UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Luiz Ricardo Flôres Maestri

FERRAMENTA DE SUPORTE AO PROCESSO DE VOLUNTARIADO DA INICIATIVA COMPUTAÇÃO NA ESCOLA

Florianópolis

2018

Luiz Ricardo Flôres Maestri

FERRAMENTA DE SUPORTE AO PROCESSO DE VOLUNTARIADO DA INICIATIVA COMPUTAÇÃO NA ESCOLA

TCC submetido ao curso de Sistemas de Informação para a obtenção do Grau de Bacharel em Sistemas de Informação.
Orientador: Prof. Dr. Jean Carlo Rossa Hauck

Florianópolis

2018

Catalogação na fonte elaborada pela biblioteca da
Universidade Federal de Santa Catarina

A ficha catalográfica é confeccionada pela Biblioteca Central.

Tamanho: 7cm x 12 cm

Fonte: Times New Roman 9,5

Maiores informações em:
http://www.bu.ufsc.br/design/Catalogacao.html

Luiz Ricardo Flôres Maestri

FERRAMENTA DE SUPORTE AO PROCESSO DE VOLUNTARIADO DA INICIATIVA COMPUTAÇÃO NA ESCOLA

Este TCC foi julgado aprovado para a obtenção do Título de "Bacharel em Sistemas de Informação", e aprovado em sua forma final pelo curso de Sistemas de Informação.

Florianópolis, 27 de Novembro 2018.

Prof. Cristian Koliver, Dr.
Coordenador do Curso

Banca Examinadora:

Prof. Dr. Jean Carlo Rossa Hauck
Orientador

Prof. Leandro José Komosinski, Dr.

Nathalia da Cruz Alves, Bsc

Minha mãe que me ensinou o que é fé e perseverança.

Todas as pessoas deveriam aprender a programar, pois isso ensina a pensar.

Steve Jobs

RESUMO

Atualmente o mercado na área de tecnologia apresenta-se no mundo em grande crescimento. Todavia no Brasil não se tem percebido interesse suficiente na capacitação de jovens para atuarem na área. Iniciativas como a Computação na Escola (CnE), que utilizam esforços voluntários para apoiar o ensino de computação para crianças e jovens, nascem procurando atender esta deficiência na educação básica. A gestão do processo completo de voluntariado se torna algo complexo e trabalhoso, principalmente pelas dificuldades de comunicação devido à troca de documentos por meio físico entre as entidades envolvidas. Nesse contexto, este trabalho propõem o desenvolvimento e implantação de um sistema web a fim de auxiliar a gestão do processo de voluntariado. O sistema foi concebido, modelado, desenvolvido, testado e implantado. Através disto busca-se favorecer a CnE a realizar sua proposta de estimular o ensino de computação a crianças e adolescentes. Tendo este objetivo em mente foi averiguado por meio de pesquisa na literatura e entrevista com os responsáveis, melhorias necessárias no gerenciamento de processos de iniciativas voluntárias para que as mesmas possam ser corrigidas com a implementação de um sistema web.

Palavras-chave: Sistema Web, Computação na Escola, Gestão de Processos, Voluntariado, Ensino de Programação.

ABSTRACT

Currently, the market in technology is in the world in great growth, however, in Brazil, there has not been perceived enough interest in the training of young people to work in the area. Initiatives such as Computação na Escola (CnE), which use voluntary efforts to support computer education for children and young people, are born seeking to address this deficiency in basic education, yet managing the entire process of volunteering becomes complex and cumbersome, especially due to communication difficulties due to the physical exchange of documents between the entities involved. In this context, this work proposes the development and implementation of a web system in order to assist the management of the volunteer process. The system had been designed, modeled, developed, tested and deployed. Through this, it seems to favor the CnE to carry out its proposal to stimulate the teaching of computation to children and adolescents. Having this goal in mind had been ascertained through research in the literature and interview with those responsible, necessary improvements in the management of voluntary initiatives processes so that they can be corrected with the implementation of a web system.

Keywords: Web System, Computação na Escola, Process Management, Voluntary, Programming Teaching.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

A demanda de profissionais nas áreas de Ciências, Tecnologia, Engenharia e Matemática tem a previsão, nos Estados Unidos, de triplicar no período entre 2011 e 2021 (LANGDON et al., 2011), chegando a 1,1 milhão em 2024, porém somente 45% destas mesmas vagas serão preenchidas (DUBOW, 2013).

Mesmo com essa demanda crescente na área da TI , em 2015 um estudo divulgado (ABDUL-ALIM, 2015) aponta que alunos do ensino médio não seriam capacitados em sala de aula com a finalidade de ocupar essas vagas e, dada esta falta de conhecimento prévio, poucos estudantes na graduação de tecnologia acham que aprender a programar seja fácil (JENKINS, 2002).

Jenkins (2002) afirma que o ensino de programação nos cursos de graduação na área tecnológica, normalmente lecionada nas fases iniciais, é assunto fundamental do curso. Jenkins explica que este momento na vida de muitos estudantes é uma transição difícil e que os materiais mais básicos que os mesmos encontrarão podem ser potencialmente desafiadores em meio a essa mudança.

Muitos estudantes deixam de acreditar no seu sucesso, e com isso perdem motivação e sem a mesma não há aprendizado (JENKINS, 2001). Segundo Allende (2012), no Brasil, 38% dos alunos de nível superior abandonam o curso em seu primeiro ano, deste valor 25,93% não tinha informações o suficiente sobre o curso no início (MELLO et al., 2013).

Uma possível solução seria o ensino de programação e de conceitos de computação já desde as séries iniciais. A implantaçãodo ensino de computação na educação depende basicamente de quatro fatores: o computador, o software educativo, o aluno e da disponibilidade de professor capacitado a usar o computador como ferramenta educacional (VALENTE, 1993)

Para Marques et al. (2011) a prática de programação no ensino médio é interessante e o uso de jogos pode servir de fator motivacional forte, tal qual têm servido em cursos de programação introdutórios em universidades.

Alguns programas governamentais pontuais como o "Um Computador por Aluno", iniciado no ano de 2007, tem como objetivo incentivar o ensino da informática na educação pública básica entretanto muitas vezes o que se encontra nas escolas são laboratórios abandonados, com-

putadores quebrados e ultrapassados. E mesmo que os laboratórios estejam em boas condições, há falta de profissionais para ocupar o cargo de professor de Informática (LACERDA, 2012).

Segundo o censo sobre o ensino superior do INEP (2016) no ano de 2015 somente 1476 do concluintes, maior taxa dos últimos anos, estariam em cursos de Licenciatura em Computação (informática). E contrapartida, o INEP (2017) afirma que "o País conta com 186,1 mil escolas de educação básica", o que evidencia a carência de profissionais para o ensino de computação nas escolas.

Iniciativas público-privadas como a escola NAVE[1] e voluntárias como code.org[2] e Computação na escola(CnE)[3] surgiram com a intenção de levar o ensino da computação para crianças e jovens a um maior público. Muitas dessas iniciativas tem contado com a participação de profissionais da área tecnológica como voluntários para apoiar o ensino de programação.

O Centro de Voluntariado de São Paulo define o voluntariado como "a doação de tempo, trabalho e talento para causas de interesse social, a fim de melhorar a qualidade de da comunidade" (GOLDSTEIN, 2007).

A Iniciativa Computação na Escola tem utilizado o envolvimento de voluntários no ensino de programação para crianças (CNE, 2017). No entanto, gerenciar o processo de voluntariado é complexo, especialmente sem o apoio de uma ferramenta. Tipicamente o processo de voluntariado na Iniciativa Computação na Escola envolve:

- A CnE, o voluntário de alguma empresa parceira, a empresa e uma escola alvo.
- Envios de diversos documentos e Termos de Responsabilidade por e-mail/fax.
- Avaliações de perfil ao início do processo e avaliações de desempenho ao fim do mesmo.
- Agendamento das datas de treinamento e monitoria.

Para Tenório (2001) ONG (voluntárias) tem um "estilo próprio de gestão" construído ao longo de sua existência e demonstrando uma resistência aos modelos mais estruturados, apresentando dificuldades na execução de tarefas administrativas.

[1] http://www.oifuturo.org.br/educacao/nave/
[2] http://code.org
[3] http://www.computacaonaescola.ufsc.br

Assim, espera-se que a existência de uma ferramenta web para gerência do processo de voluntariado venha a contribuir com o aumento do ensino de computação nas escolas e o interesse de crianças e adolescentes na área tecnológica.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Desenvolver e avaliar uma ferramenta web de gestão do processo de voluntariado para o ensino de computação nas escolas, no contexto da Iniciativa Computação na Escola do Departamento de Informática e Estatística da UFSC.

1.2.2 Objetivos Específicos

1. Analisar da fundamentação teórica sobre gestão de voluntariado e desenvolvimento web.
2. Mapeamento sistemático do estado da arte sobre ferramentas de auxílio ao voluntariado.
3. Modelar e implementar um sistema web para auxiliar no processo de voluntariado na iniciativa Computação na Escola.
4. Implantar e avaliar do sistema desenvolvido.

1.3 MÉTODO DE PESQUISA

Este trabalho será desenvolvido em 6 etapas:

1. Análise da fundamentação teórica (E-1):
   Será analisada a fundamentação teórica sobre gestão de voluntariado e desenvolvimento web.
2. Mapeamento sistemático do estado da arte (E-2):
   Nesta etapa será realizado Mapeamento sistemático do estado da arte sobre ferramentas de auxílio ao voluntariado.
3. Realizar um estudo comparativo de tecnologias de mercado para desenvolvimento web (E-3)

Nesta etapa será realizada uma análise das tecnologias de desenvolvimento web selecionadas, com base nos requisitos do sistema, as ferramentas mais indicadas para o desenvolvimento.

4. Modelar e implementar um sistema web para auxiliar no processo de voluntariado na iniciativa Computação na Escola (E-4):
   Nesta etapa será realizada a modelagem do sistema web e do banco de dados, utilizando a UML e diagrama entidade relacional. Esta etapa possui as seguintes atividades:

   (a) Implementar Módulo administrativo para a CnE.

   (b) Implementar Módulo avaliativo para as escolas participantes.

   (c) Implementar Módulo de autorização para as empresas parceiras.

   (d) Implementar Módulo de cadastro para candidatos a voluntários.

5. Implantação e avaliação (E-5):
   Nesta etapa o sistema será implantado no servidor fornecido pela Setic/UFSC e avaliado pelos responsáveis da iniciativa Computação na Escola.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos mais importantes utilizados no desenvolvimento deste trabalho como os aspectos relacionados ao voluntariado, área de aplicação deste trabalho, e aspectos técnicos envolvidos no desenvolvimento do sistema proposto.

2.1 VOLUNTARIADO

Segundo a Lei nº 9.608/1998 do Voluntariado (BRASIL, 1998), o serviço voluntário no Brasil é definido por:

- Não ser remunerado.
- Não gerar vínculo empregatício.
- Não pode ser exigido em contrapartida de algum benefício.
- Ser exercido em entidade pública ou privada, sem fins lucrativos e com objetivos sociais.
- Exigir a assinatura do Termo de Adesão descrevendo objetivo e condições de trabalho e reembolso de despesas.

Marcos e Amador (2014) salientam que o voluntariado mesmo não sendo uma pratica recente, somente em 2011 entrou nas pautas politicas da UE. A gestão do voluntariado também é nova nos estudos da administração, tornando-se um desafio para o setores de RH, habituados com as relações contratuais típicas da gestão mais clássica (VIDAL et al., 2007).

O ciclo de gestão do voluntariado como apresentado na figura 1 pode ser dividido em 6 fases como objetivos bem específicos (MARCOS; AMADOR, 2014):

1. Preparação - Contém a criação de um plano de voluntariado (explicado abaixo) e de uma ferramenta de gestão que permita a replicação de processos na organização.
2. Definição - Esta fase do ciclo se constitui da criação de perfis sociodemográficos, escolares e profissionais desejáveis para missão da empresa e da seleção de voluntários com tais perfis.

3. Acolhimento - Este é o momento onde há o comprometimento, com os deveres e direitos, do voluntário com a organização e vice-versa. Também é o momento de esclarecimento das dúvidas do voluntário sobre suas tarefas na organização.

4. Desenvolvimento - Este é o momento de formação do voluntário para as tarefas que lhe serão designadas, uma forte comunicação com o líder e demais integrantes do grupo de trabalho. E um acompanhamento para analisar o grau de satisfação das expectativas e identificação de necessidades.

5. Reconhecimento - Esta fatia do ciclo de gestão do voluntariado consiste de ações que visam reconhecer o trabalho do voluntário no dia a dia.

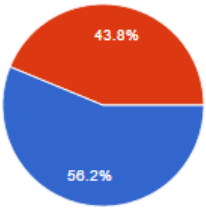
6. Desvinculação - Implica em um processo de orientação ao voluntário de saída e a manutenção da relação de mesmo com a organização após o período de voluntariado.

Figura 1 – As fases do ciclo de gestão do voluntariado (MARCOS; AMADOR, 2014).

2.1.1 Sobre o perfil Voluntário

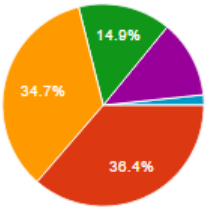
O voluntário é, em senso comum, todo aquele que por valores solidários e sociais, doa seu tempo, seu trabalho e seus talentos espontaneamente e sem remuneração, para causas de interesse comunitário, humanitário e social.(YAZBEK, 2002).

Segundo MENDES (2015) os voluntários no Brasil, durente o periodo da copa de do mundo da FIFA de 2014, em sua maioria são mulheres, entre 18 e 35 anos, com boa estabilidade financeira, possivelmente estudantes de nível superior ou pós-graduação, sem filhos como pode-se inferir pelos gráficos abaixo:



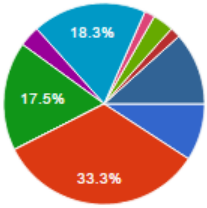
Feminino	68	56.2%
Masculino	53	43.8%
Prefere não responder	0	0%

Figura 2 – Gênero dos voluntários (MENDES, 2015).



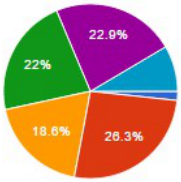
Menor que 18 anos	0	0%
Entre 18 e 25 anos	44	36.4%
Entre 26 e 35 anos	42	34.7%
Entre 36 e 45 anos	18	14.9%
Entre 46 e 60 anos	15	12.4%
Maior que 60 anos	2	1.7%
Prefere não responder	0	0%

Figura 3 – Idade dos voluntários (MENDES, 2015).



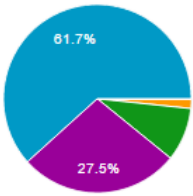
Autônomo	11	9.2%
Estudante	40	33.3%
Empresário	0	0%
Servidor Público	21	17.5%
Profissional Liberal	4	3.3%
Empregado na inciativa privada	22	18.3%
Empregado em ONG's	2	1.7%
Aposentado	4	3.3%
Prefere não responder	2	1.7%
Outros	14	11.7%

Figura 4 – Profissão dos voluntários (MENDES, 2015).



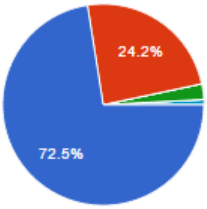
Menos que 1 salário mínimo (- R$ 788,00)	2	1.7%
1 a 3 salários mínimos (R$ 788,00 a 2.364,00)	31	25.8%
3 a 5 salários mínimos (R$ 2.364,00 a 3.940,00)	22	18.3%
5 a 10 salários mínimos (R$ 3.940,00 a 7.880,00)	26	21.7%
Mais que 10 salários mínimos (+ R$ 7.880,00)	27	22.5%
Prefere não responder	10	8.3%

Figura 5 – Renda dos voluntários (MENDES, 2015).



Ensino Fundamental Incompleto	0	0%
Ensino Fundamental Completo	0	0%
Ensino Médio Incompleto	2	1.7%
Ensino Médio Completo	11	9.2%
Nível Superior Incompleto	33	27.5%
Nível Superior Completo ou mais	74	61.7%
Não se Aplica	0	0%
Prefere não responder	0	0%

Figura 6 – Escolaridade dos voluntários (MENDES, 2015).



Solteiro	87	72.5%
Casado	29	24.2%
Viúvo	0	0%
Divorciado	3	2.5%
Prefere não responder	0	0%
Outros	1	0.8%

Figura 7 – Estado Civil dos voluntários (MENDES, 2015).

Segundo Centro de Voluntariado de São Paulo (2016), 77% dos voluntários se sentem completamente satisfeitos com a atividade desempenhada, 87% estão motivados a continuar no voluntariado sendo as principais motivações, solidariedade (67%), melhorar o mundo fazendo a diferença (32%) e religiosidade (22%).

2.1.2 Voluntariado e educação

Na área da educação o voluntariado tem repercutido com inúmeros projetos nos diversos níveis de ensino tais como Universidade Solidária, Educação Solidária, Alfabetização Solidária, Projeto Amigos da Escola, entre outros. Estas iniciativas são financiadas por diversas fontes como UNESCO, empresas privadas, ONG's e entidades governamentais (SILVA, 2009).

De todas os projetos supracitados, o de maior destaque foi o Projeto Amigos da Escola – Todos pela Educação, lançado pela rede Globo durante o segundo mandato do governo FHC. Visa fortalecer o ensino público, sendo um dos empreendimento sociais mais importantes do país (GLOBO, 2012)(SILVA, 2009).

Figura 8 – Logo Amigos da Escola – Todos pela Educação (DIáRIO CATARINENSE, 2012).

2.2 APLICAÇÕES WEB

Mora (2002) define aplicações web como sendo uma categoria especial de cliente-servidor. Caivano e Villoria (2009) complementam ressaltando que aplicações web não são mais do que ferramentas acessíveis por meio de um navegador web e acesso à rede, internet ou intranet, na web 2.0.

Tanenbaum e Steen (2007) descreve a arquitetura cliente-servidor como um sistema distribuído dividido em dois processos, que se comunicam por meio de um protocolo definido, o HTTP no caso de aplicações web (MORA, 2002), o servidor que implementa um serviço específico e o cliente requisita o serviço, e espera o retorno, do servidor (TANENBAUM; STEEN, 2007) como exemplificado na figura 9.

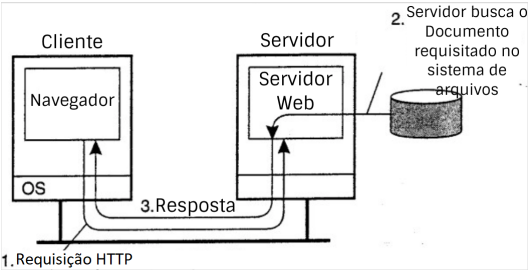


Figura 9 – Típica organização cliente-servidor de um web site/sistema (TANENBAUM; STEEN, 2007).

2.2.1 Tecnologias

Na seção abaixo são apresentadas as tecnologias utilizadas para o servidor e cliente no contexto deste trabalho, são elas ReST e Spring Project para o servidor e no lado do cliente ReactJS e Flux.

2.2.1.1 Servidor

2.2.1.1.1 Arquitetura Rest

Representational State Transfer, ReST, foi concebido por Roy Thomas Fielding na sua tese de doutorado em 2000, como uma arquitetura para sistemas de hipermídia distribuídos. Fielding (2000) elenca algumas características, restrições, que compõem a arquitetura ReST:

1. Arquitetura cliente-servidor - Para Fielding (2000) a arquitetura cliente-servidor é a resposta para uma requisição na qual servidor envia os dados em um formato fixo (representação). Separando a interface do armazenamento dos dados, assim, melhorando a portabilidade e escalabilidade.

2. Stateless - O servidor não detém conhecimento nenhum sobre os clientes a ele conectado, assim como conteúdo dos dados e como são executadas as funções do cliente (BERKENBROCK, 2005). Assim cada interação do cliente com o servidor deve conter todas as informações necessárias para a execução do pedido (FIELDING, 2000).

3. Cache - A utilização de cache, uma restrição facultativa, pode reduzir as interações entre cliente e servidor aumentando a eficiência e escalabilidade do sistema (FIELDING, 2000). Berkenbrock (2005) realça que a adoção de cache reduz a confiabilidade nos dados apresentados no cliente uma vez que podem não corresponder a versão mais recente no servidor.

4. Interface uniforme - Para Fielding (2000) a principal distinção do ReST para outras arquiteturas baseadas em rede. A interface uniforme degrada a eficiência por ser aplicado o princípio de generalidade da engenharia de software.

5. Sistemas em camadas - Fielding (2000) explicita que a divisão do sistema em camadas hierárquicas melhoram a escalabilidade, performance e segurança do sistema, por um melhor balanceamento da carga de processamento.

6. Código sob demanda - Outra restrição facultativa permite a funcionalidade opcional de estender código para execução no cliente por intermédio de download e instalação de applets e scripts (FIELDING, 2000).

2.2.1.1.2 Spring Project

O Spring Project ou Spring IO é uma plataforma modular do ecossistema Java mantida pela Pivotal (2018a) para prover a infraestrutura para as aplicações, desde segurança, aplicações web até big data. Os Módulos usados neste projeto são:

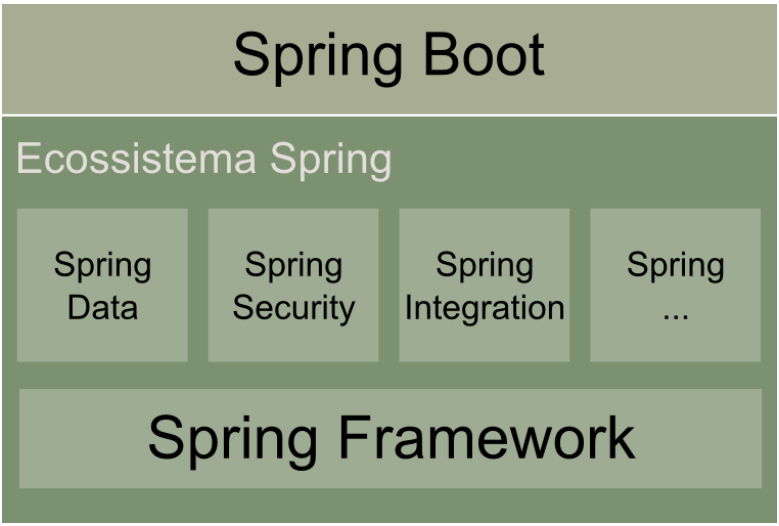


Figura 10 – Ecossistema da plataforma Spring IO (DEVMIDIA, 2015).

- Spring Boot - Facilita a criação de projetos Spring. Contém injeção de dependências, servidor embarcado e configurações mais amigáveis.

- Spring Data - Abstração de ORM construída em cima da JPA e da JTA

- Spring Security - Provê mecanismos de autenticação e autorização que podem ser facilmente estendidos para atender as necessidades da aplicação.

2.2.1.1.3 MySQL

Segundo a Oracle Corporation (2018), o MySQL é o banco de dados de *Open source* mais conhecido no mundo. Com desempenho, confiabilidade e facilidade de uso comprovados por gigantes da Web, incluindo Facebook, Twitter, YouTube, sendo a principal opção de banco de dados para aplicação Web. Além disso, é uma opção grande como banco de dados integrado (ORACLE CORPORATION, 2018).

2.2.1.2 Cliente

2.2.1.2.1 Virtual DOM

A virtual DOM, é uma cópia em memória da DOM renderizada, sendo um recurso implementado por diversos frameworks e bibliotecas JS para otimização de re-renderização da interface. A virtual DOM ao sofrer uma alteração de estado é comparada com a DOM real e esta por sua vez é alterada somente no ponto necessário da árvore (TECH, 2017) como apresentado na figura 11.

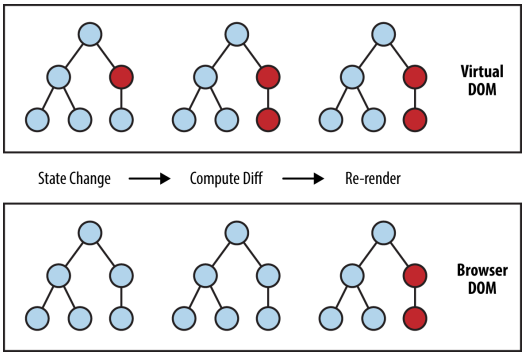


Figura 11 – Virtual Dom calculando a diferença após alteração de estado para otimização da nova renderização (EISENMAN, 2015).

2.2.1.2.2 ReactJS

ReactJS, ou simplesmente React, é uma biblioteca JS, criada pelo Facebook (2018), para a construção e gerenciamento de estado de interface. Utilizando-se de uma virtual DOM para otimização de renderização durante mudança de estado da aplicação (TECH, 2017).

O React se propõem a mudar a mentalidade ao construir a interface web, quebrando-a em componentes flexíveis e de fácil manutenção, e utiliza-se da sintaxe, opcional, JSX para uma maior legibilidade do código(HUNT, 2013). Outras vantagens do React é a melhor performance quando há alteração do estado visto que somente o nó da DOM que sofreu a alteração será novamente renderizado. A possibilidade de renderização em *server side* o que melhorar o SEO (TECH, 2017)(HUNT,

2013).

2.2.1.2.3 Arquitetura Flux

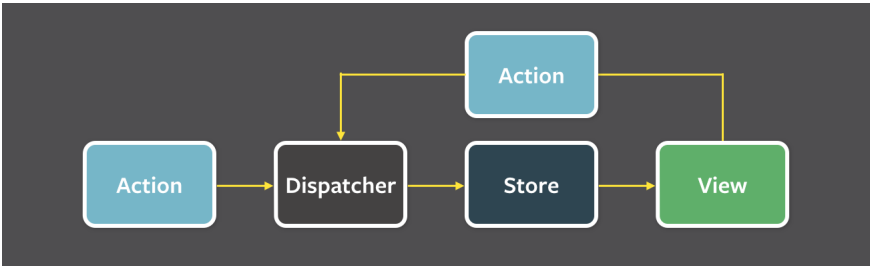


Figura 12 – Fluxo de dados proposto pela arquitetura Flux (FACEBOOK, 2015).

Flux é a proposta do Facebook (2015) para o fluxo dos dados dentro das aplicações construídas com ReactJS, trabalhando com um fluxo unidirecional como ilustrado na figura 12, dividido em quatro estruturas diferente que são:

- *Action* - Capturam as maneiras pelas quais qualquer coisa pode interagir com a aplicação. São objetos simples que possuem um campo "*type*"e alguns dados.
- *Dispatcher* - Toda *action* é enviada para os *stores* registrados no *dispatcher*.
- *Store* - Após os *stores* serem atualizados em resposta a *action*, eles emitem um evento de alteração.
- *View/Controller-View* - Os dados dos *stores* são apresentados na *view* e esta quando esta recebe o sinal de um evento de alteração, realiza uma nova renderização utilizando os novos dados do *store*.

3 ESTADO DA ARTE

Este capítulo apresenta o estado da arte das ferramentas para gestão de processos de voluntariado. A análise do estado da arte é realizada seguindo as principais etapas do método de mapeamento sistemático de literatura definido por Petersen K. (2008), adaptadas para uma pesquisa de trabalho de conclusão de curso de graduação. Esse método é realizado definindo o mapeamento da literatura disponível, executando a busca e fazendo a extração dos resultados, bem como sua análise.

3.1 DEFINIÇÃO DO PROTOCOLO DE MAPEAMENTO

Este mapeamento tem como objetivo realizar uma pesquisa de ferramentas/sistemas web para o gerenciamento de processos de voluntariado. Com essa premissa foi definida a seguinte pergunta de pesquisa: "Existem sistemas informatizados para a gestão de processos para programas de voluntariado e quais seus benefícios?".

A base de pesquisa utilizada foi o Google Scholar[1], pelo acesso a diversas bases de dados. As buscas foram realizadas com os termos em português e suas traduções em inglês. Os resultados analisados da busca foram apenas os que estavam nestes idiomas, e dentro do período de 2010 a 2017.

3.1.1 Critérios de inclusão/exclusão

O material escolhido deve obedecer ao seguintes critérios para serem considerados proveitosos no mapeamento:

- O material encontrado deve conter *features* que os sistemas existentes apresentam.

- O material encontrado deve ser focado em processos do terceiro setor, iniciativas de responsabilidade social e voluntariado.

[1] https://scholar.google.com.br

3.1.2 Critério de qualidade

O critério de qualidade foi definido como:

- Deve preferencialmente conter necessidades que o sistema deveria atender.

3.1.3 Dados a serem extraídos

Para responder a pergunta de pesquisa com base nos trabalhos selecionados são extraídos os seguintes dados:

- Tecnologias utilizadas no desenvolvimento das ferramentas.
- Funcionalidades implementadas.
- Limitações encontradas.

3.1.4 Strings de busca

Os termos utilizados na busca são derivados da pergunta de pesquisa e são apresentados no Quadro 1:

Tabela 1 – Sinônimos e traduções dos termos de busca

Termo	Sinônimo	Tradução
"Sistema informatizado"	"Sistema web", "Aplicação"	"Computerized system", "Web system","Application"
"Gestão de processos"	"Gerenciamento"	"Process Management","Management"
"Empresa de Responsabilidade Social"	"Voluntariado", "Terceiro Setor"	"Corporate Social Responsibility/ CSR", "voluntary", "voluntary sector/community sector"
"Benefícios"		"Benefits"

As Strings de busca derivadas aparecem no Quadro 2, bem como o número total de resultados retornados:

Tabela 2 – Termos de busca

Termos de busca	Resultados retornados
("Sistema informatizado"OR "Sistema web"OR "Aplicação"OR "Computerized system"OR "Web system"OR "Application") AND ("Gestão de processos"OR "Gerenciamento"OR "Process Management"OR "Management") AND ("Empresa de Responsabilidade Social"OR "Voluntariado"OR "Terceiro Setor"OR "Corporate Social Responsibility/ CSR"OR "voluntary"OR "voluntary sector/community sector") AND ("Beneficios"OR "Benefits") PERÍODO (2010 a 2017)	13
("Web system"OR "Application") AND "Process Management"AND ("voluntary"OR "voluntary sector/community sector") AND "Benefits"PERÍODO (2010 a 2017)	23
"volunteer management web system"	995.000

3.2 EXECUÇÃO DE BUSCA

As buscas foram feitas em dezembro de 2017. No total foram necessárias 3 iterações de busca, apresentadas na sequência.

3.2.1 Primeira e Segunda Execução

A primeira execução retornou um total de 13 registros, nenhum, contudo atendeu aos critérios de inclusão, e um reforçou as dificuldades de gestão neste setor.

A segunda execução, usando somente alguns termos de busca na língua inglesa, elencados na segunda linha da tabela 2 também se mostrou escassa.

3.2.2 Terceira Execução

Em razão do baixo número de resultados encontrados, a busca por ferramentas correlatas também foi realizada utilizando o Google. Os critérios para busca foram considerados os mesmos.

Após uma pesquisa sobre ferramentas de gerência de voluntariado, foram elencadas para análise 6 ferramentas existentes no mercado (KNOW HOW NON PROFIT(ORG.), 2018), são elas:

1. Better Impact [2].
2. Team Kinetic [3].
3. Volistics [4].
4. Do-it [5] [6] [7].
5. Samaritan [8].
6. Volunteer Hub [9].

3.2.3 Ferramentas selecionadas

Cada uma das ferramentas selecionadas é brevemente apresentada a seguir:

3.2.3.1 Better Impact

Better Impact é uma ferramenta web paga para gestão de voluntariado. A ferramenta tem a preocupação de ser genérica permitindo que o gestor da iniciativa voluntária escolher entre templates já existentes para os mais diversos e comuns tipos de iniciativas voluntárias, ou um template branco com menos customização previamente já realizada.

[2] https://www.betterimpact.co.uk/
[3] https://teamkinetic.co.uk/
[4] https://www.volgistics.com/
[5] https://do-it.org/
[6] https://doit.life/
[7] http://doittrust.org/
[8] https://samaritan.com/
[9] https://www.volunteerhub.com/

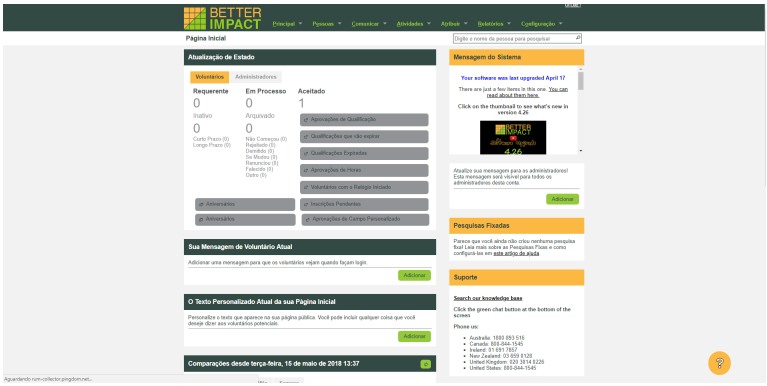


Figura 13 – Tela inicial do sistema Better Impact (2018).

3.2.3.2 Team Kinetic

Team Kinetic é uma ferramenta web paga com uma versão gratuita até 9 voluntários, a partir de 10 voluntários para gerir os preços variam entre £20pm à £150pm. A principal preocupação dessa ferramenta é com o recrutamento e a retenção de voluntários. Permitindo o envio de e-mails e mensagens de texto(SMS). A versão enterprise(£150pm) possibilita ao gestor elencar um perfil de usuário que pode visualizar as oportunidades.

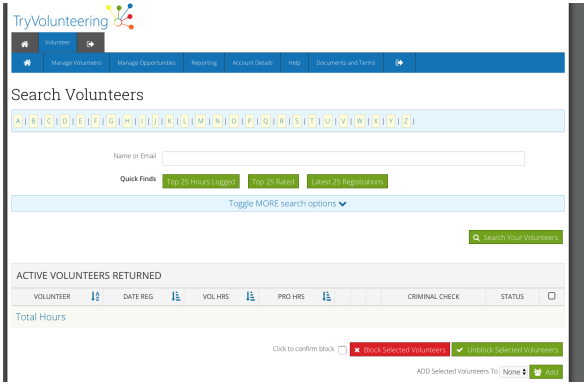


Figura 14 – Listagem de Voluntários (TEAM KINECTIC, 2018).

3.2.3.3 Volistics

Outra ferramenta web existente no mercado, Volistics, é customizável para a necessidade da iniciativa. Inicialmente o gestor utiliza uma versão de testes com alguns recursos limitados por 60 dias, podendo evoluir a qualquer momento para versão paga ou instantaneamente após os 60 dias. A Volistics conta com suporte de 13 horas/dia de segunda a sexta-feira.

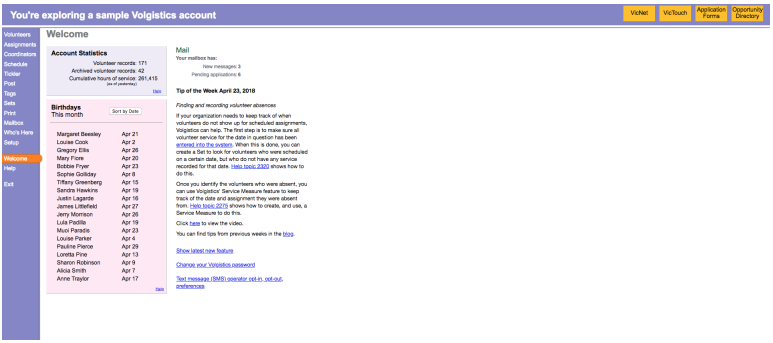


Figura 15 – Tela de boas vindas do Volgistics (2018).

3.2.3.4 Do-it

A Do-it, é uma organização de caridade que disponibiliza uma plataforma web gratuita para gestão de iniciativas voluntárias. Porém diferentemente dos sistemas/plataformas anteriores há a necessidade de uma verificação dos dados da organização antes da divulgação das vagas de voluntariado. Ela não contém uma possibilidade alta de customização, somente elencar um perfil de voluntário que possa se candidatar a vaga.

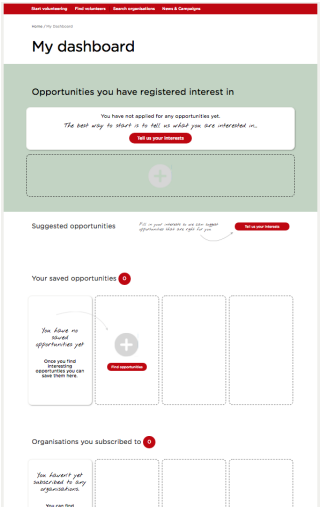


Figura 16 – Dashboard do voluntário (DO-IT, 2018).

3.2.3.5 Samaritan

A ferramenta Samaritan, é uma plataforma completa para gestão de voluntários. Contando com todas as funcionalidades dos outros sistemas supracitados e outras funções tais como verificação de antecedentes criminais e sistema para treinamento dos voluntários. O preço inicial da plataforma é de US$2500 anuais, entretanto, para organizações com orçamento anual menor de US$400 mil ou membro da Association of Leaders in Volunteer Engagement (AL!VE) tem direito a uma versão gratuita com os recursos mais básicos da plataforma.

Figura 17 – Site da Samaritan Technologies (2018).

3.2.3.6 Volunteer Hub

O Volunteer Hub, está no mercado desde 1996, sendo o sistema mais antigo dos estudados. Possui integração com diversos CRMs, ferramentas de angariação de fundos, deduplicação.Diferentemente das demais ferramentas não existe uma versão gratuita para teste, sendo o plano de menor valor de US$150 mensais e US$595 de instalação e configuração inicial.

Figura 18 – Site da Volunteer Hub (2018).

3.3 ANÁLISE DOS RESULTADOS

A Know How Non Profit(Org.) (2018) lista algumas funcionalidades que devem ser avaliadas pelo gestor na escolha de uma ferramenta para o fim de gerir uma organização sem fins lucrativos, dentre elas:

1. A inscrição de novos voluntários.
2. Coleta de dados dos voluntários.
3. Cadastro de oportunidades de voluntáriado.
4. Cadastro e manutenção de Eventos voluntários.
5. Relatórios e feedbacks.
6. Gestão de comunicação com os envolvidos.
7. Incentivos ao voluntário.
8. Treinamento e cursos.

Para analisar as ferramentas encontradas, foram definidos alguns requisitos iniciais, tomando por base aqueles citados pela Know How Non Profit(Org.) (2018), requisitos funcionais, e por meio de entrevista com um dos coordenadores da CnE, requisitos não funcionais, são elas:

- RF1 - O sistema deve permitir o cadastro dos voluntários.
- RF2 - O sistema deve permitir o cadastro dos beneficiados.
- RF3 - O sistema deve permitir o cadastro de eventos de voluntariado.
- RF4 - O sistema deve permitir a alocação de voluntários.
- RF5 - O sistema deve permitir o gerenciamento de treinamentos.
- RF6 - O sistema deve permitir a avaliação dos voluntários.
- RF7 - O sistema deve permitir a avaliação dos treinamentos.
- RNF1 - O sistema deve ser Integrado ao site da CnE.
- RNF2 - Gratuito e Open-Source.
- RNF3 - Aplicação web.

A Tabela 3 apresenta um comparativo entre as ferramentas analisadas, de acordo com os requisitos funcionais de mais alto nível, identificados para a ferramenta desenvolvida neste trabalho. Para avaliar o grau de atendimento de cada requisito, foram usados os seguintes conceitos: T (Totalmente Atendido), P (Parcialmente Atendido), N (Não Atendido), NT (Não Testado).

Tabela 3 – Funcionalidades atendidas

	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RNF1	RNF2	RNF3
Better Impact	T	N	N	T	P	T	N	N	N	T
Team Kinetic	N	P	T	T	N	T	N	N	P	T
Volistics	T	T	P	T	N	T	N	N	N	T
Do-it	T	N	T	P	N	N	N	N	P	T
Samaritan	NT	NT	NT	NT	NT	NT	NT	N	P	T
Volunteer Hub	NT	NT	NT	NT	NT	NT	NT	N	N	T

Como apresentado na Tabela 3 as ferramentas "Samaritan"e "Volunteer Hub"não poderam ser testadas devido ao custo de licença além do orçamento disponível para o projeto.

Por meio de uma breve pesquisa nos sites destas ferramentas, foi constatado que ambas ferramentas não atenderiam as funcionalidades RF5 e RF7, entretanto elas contemplam as demais funcionalidades não testadas.

Dentre as demais plataformas elencadas as mais próximas do ideal são "Team Kinetic"e "Volistics"atendendo a maior parte das funcionalidades em sua totalidade. Há possibilidade de que a ferramenta "Better Impact"tenha sido prejudicada na pesquisa devido a escolha de um *template* de testes, voltado para gerenciamento escolar.

Devido a nenhum dos sistemas serem de código aberto não foi possível encontrar informações sobre tecnologias e método de desenvolvimento utilizados, deixando a analise carente de informações para este tópico.

Existem diversas aplicações para o auxílio do controle de voluntários. Porém nenhuma das ferramentas supracitadas pode atender a todos os requisitos elencados para o projeto, resultando na necessidade da implementação de um sistema específico para a Computação na Escola.

4 PROPOSTA DE SOLUÇÃO

Este capitulo apresenta o processo de elaboração da solução para o presente trabalho. Primeiramente é abordada a elicitação dos requisitos funcionais e não-funcionais, para então ser apresentado o fluxo das atividades da solução. Também é apresentado o diagrama de casos de uso e seus respectivos detalhamentos, sendo que, para cada detalhamento de caso de uso, são apresentados alguns protótipos de telas. Após a definição dos requisitos, é apresentada uma pesquisa comparativa para escolha das tecnologias da solução implementada.

A solução desenvolvida tem como objetivo apoiar a iniciativa Computação na Escola do GQS/INE/UFSC durante o processo de seleção dos tutores voluntários e avaliações dos mesmos, sendo um sistema para controle de tutores voluntários (CuLTiVAR).

Para o desenvolvimento da solução deste trabalho foi adotado a linguagem Java para criação do *backend* devido a grande probabilidade de encontrar programadores dentro da universidade com conhecimento prévio na linguagem, visto ser a linguagem ensinada, caso haja necessidade de implementações de novas funcionalidade no futuro.

4.1 REQUISITOS

Visando o desenvolvimento de uma solução que atendesse às necessidades da iniciativa CnE e expectativas quanto às funcionalidades contempladas pela mesma, o levantamento dos requisitos foi realizado em 3 etapas.

Em um primeiro momento por meio de entrevista com o coordenador do projeto foi criado um esboço inicial dos requisitos tendo como base o fluxo, que automatizado pela solução proposta neste trabalho, apresentado na figura 19 fornecida por um dos coordenadores da CnE. Após isso foi realizado o mapeamento bibliográfico onde foram identificados novos requisitos e lapidados os requisitos já existentes usando como base os itens citados pela Know How Non Profit(Org.) (2018).

Em um terceiro momento, com telas já prototipadas utilizando os requisitos elencados anteriormente, foram entrevistados um ex-voluntário e uma pessoa responsável na CnE para confirmação de requisitos dos requisitos.

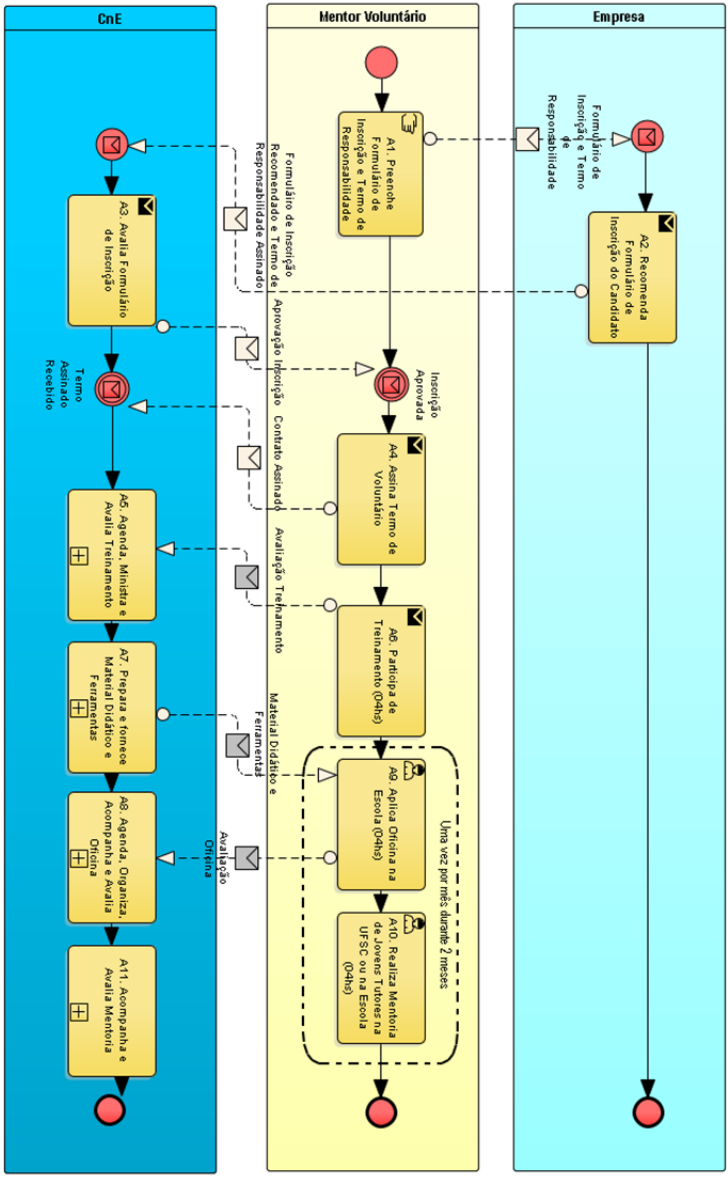


Figura 19 – Diagrama do processo de voluntariado (fornecido por um dos coordenadores da CnE).

Como resultado deste processo foram elicitados os seguintes requisitos vistos nas tabelas 4 e 5:

Tabela 4 – Requisitos Funcionais

Identificação	Descrição
RF 1	O sistema deve permitir ao voluntário se cadastrar da plataforma.
RF 2	O sistema deve permitir ao voluntário enviar a documentação necessária, termos de responsabilidade e de voluntariado.
RF 3	O sistema deve permitir ao responsável pela empresa recomendar a inscrição do voluntário.
RF 4	O sistema deve permitir ao responsável na CnE avaliar a inscrição recomendada.
RF 5	O sistema deve permitir ao responsável na CnE disponibilizar materiais para treinamento via documentos digitais.
RF 6	O sistema deve permitir ao responsável na CnE fornecer materiais didáticos para as oficinas.
RF 7	O sistema deve permitir ao responsável na CnE agendar eventos (Treinamentos, Oficinas, Reuniões, etc...).
RF 8	O sistema deve permitir ao Voluntário avaliar o treinamento recebido.
RF 9	O sistema deve permitir ao Voluntário e ao responsável da escola avaliar a mentoria.
RF 10	O sistema deve permitir ao responsável na CnE visualizar relatórios sobre os treinamentos.
RF 11	O sistema deve permitir ao responsável na CnE visualizar relatórios sobre os voluntários e mentorias.
RF 12	O sistema deve avisar os usuários envolvidos nos eventos.
RF 13	O sistema deve permitir ao responsável na CnE cadastrar novas empresas parceiras.
RF 14	O sistema deve permitir ao responsável na CnE cadastrar novas escolas beneficiadas.
RF 15	O sistema deve permitir ao responsável na CnE cadastrar novos locais para eventos.
RF 16	O sistema deve permitir ao Usuário alterar seu dados cadastrais.
RF 17	O Sistema deve permitir ao Responsável na CnE Cadastrar novos projetos

Continua na próxima página

Tabela 4 – *Continuação da página anterior*

Identificação	Descrição
RF 18	O Sistema deve permitir ao Responsável na CnE Cadastrar novos tipos de evento

Tabela 5 – Requisitos não-Funcionais

Identificação	Descrição
RNF 1	O sistema deve ser uma aplicação web
RNF 2	O sistema deve ser implementado na linguagem Java
RNF 3	O sistema deve ter integração com o Portal da CnE via banco de dado MySQL

Conforme abordado anteriormente, uma parte dos requisitos da solução deste trabalho foram criados com base na fundamentação teórica, algumas recomendações realizadas pela Know How Non Profit(Org.) (2018) sobre sistemas de gerenciamento de voluntáriado:

- Rec 1 - Os voluntários se cadastram via website.
- Rec 2 - Definir questionário de cadastro.
- Rec 3 - Definir quais informações devem ser coletadas após o cadastro (CV e qualificações).
- Rec 4 - O administrador do sistema cadastra todas as oportunidades de voluntariado.
- Rec 5 - Você precisará de ferramentas para comunicação com uma grande quantidade de voluntários?
- Rec 6 - Você gostaria que os voluntários gravassem suas próprias horas?
- Rec 7 - Você quer registrar o feedback do voluntário?
- Rec 8 - Deseja enviar e-mails automatizados ou mensagens de texto para voluntários (quando eles se inscrevem, participam de uma oportunidade, um dia antes de sua oportunidade acontecer, etc.)

- Rec 9 - Você quer anunciar cursos de treinamento para seus voluntários?

A tabela 6 apresenta uma comparação dos requisitos da fundamentação teórica com os da solução.

Tabela 6 – Comparação dos requisitos da fundamentação teórica com os da solução

Recomendação	Requisitos
Rec 1	RF 1
Rec 2	RF 1
Rec 3	RF 2
Rec 4	RF 7, RF 17, RF 18
Rec 5	RF 12
Rec 6	Não atendido
Rec 7	RF 8
Rec 8	RF 12
Rec 9	RF 5, RF 7, RF 8 e RF 12

O demais requisitos foram elencados por meio da análise do fluxo apresentado na imagem 19 e pelas entrevistas com participantes da iniciativa CnE.

4.2 CASOS DE USO

Por meio da análise do fluxo apresentado na imagem 19 foram identificadas quatro entidades externas que irão interagir com a solução a ser desenvolvida, ou seja, os atores do diagrama de casos de uso. São estes: Responsável na CnE, Responsável pela empresa, Responsável da escola e Voluntário.

A imagem 20 apresenta o diagrama de casos de uso referente ao sistema CuLTiVAR, na tabela 7 são apresentados de forma descritiva o que cada ator representa no mundo real e cada uma das funcionalidades acessivas a ele no sistema.

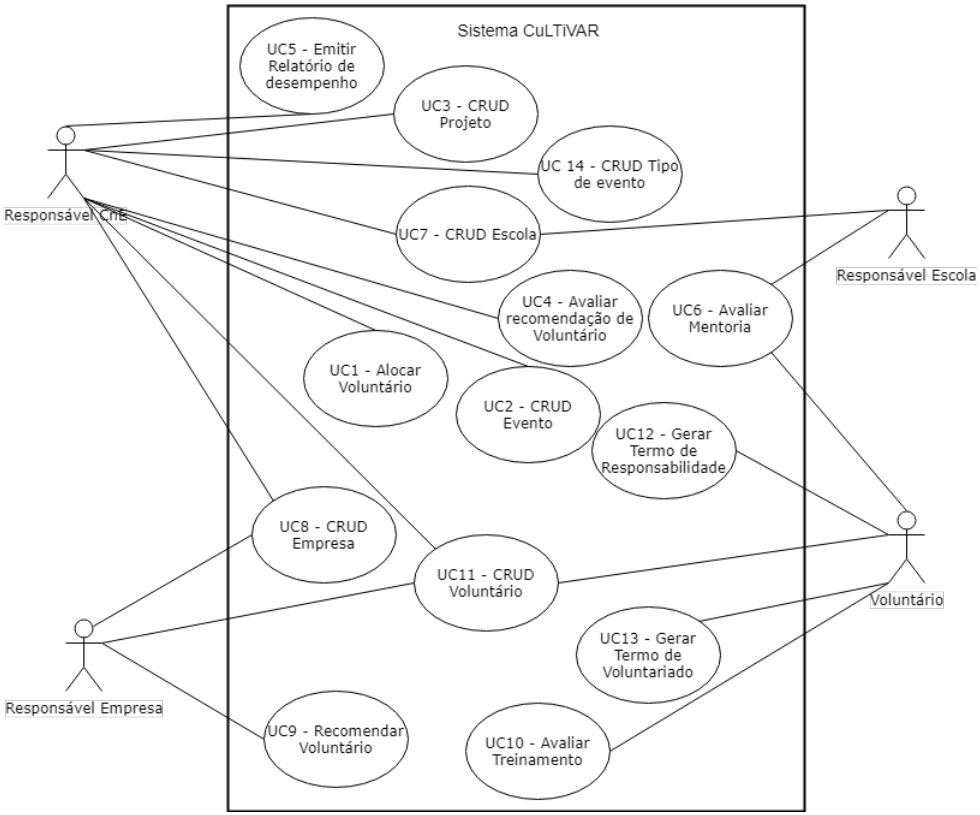


Figura 20 – Diagrama de casos de uso.

Tabela 7 – Atores e responsabilidades

Ator	Mundo Real	Casos de Uso Realizados
Responsável na CnE	Representa alguém da gestão da CnE	UC 1 - Alocar Voluntário. UC 2 - CRUD Evento. UC 3 - CRUD Projeto UC 4 -Avaliar recomendação de voluntário. UC 5 - Emitir relatório de desempenho UC 7 - CRUD escola UC 8 - CRUD empresa UC 11 - CRUD Voluntário UC 14 - CRUD Tipo de evento
Responsável pela Empresa	Representa a pessoa responsável na empresa parceira pela comunicação com a CnE	UC 8 - CRUD empresa UC 9 - Recomendar voluntário UC 11 - CRUD Voluntário
Responsável da Escola	Representa a pessoa responsável na escola parceira pela comunicação com a CnE	UC 6 - Avaliar mentoria UC 7 - CRUD escola
Voluntário	Representa a pessoa realizará a mentoria em nome da CnE	UC 6 - Avaliar mentoria UC 10 - Avaliar treinamento UC 11 - CRUD Voluntário UC 12 - Gerar termo de responsabilidade UC 13 - Gerar termo de Voluntariado

Cada casos de uso está relacionado com os requisitos funcionais por ele atendido na tabela 8 abaixo:

Tabela 8 – Rastreamento Casos de Uso - Requisitos Funcionais

Casos de Uso	Requisitos
UC 1	RF 7, RF 12
UC 2	RF 5, RF 7, RF 12, RF15
UC 3	RF 17
UC 4	RF 4
UC 5	RF 10, RF 11
UC 6	RF 9
UC 7	RF 14, RF 16
UC 8	RF 13, RF 16
UC 9	RF 3
UC 10	RF 8
UC 11	RF 1, RF 16
UC 12	RF 2
UC 13	RF 2
UC 14	RF 5, RF 6, RF 18

Como visto na tabela 8 é possível atender todos os requisitos elicitados com os caso de usos apresentados na imagem 20.

As tabelas 24, 10 e 11 abaixo apresentam, como exemplo, o detalhamento dos casos de uso “Alocar Voluntário”, “Recomendar Voluntário” e “Avaliar Treinamento” respectivamente.

Tabela 9 – Detalhamento de caso de uso: UC1 - Alocar Voluntário.

Nome do Caso de Uso	UC1 - Alocar Voluntário.
Pré-Condições	Existir pelo menos um evento e um voluntário.
Atores Envolvidos	Responsável na CnE.
Resumo	Aborda os passos necessários para o Responsável na CnE alocar um voluntário em algum evento (Treinamento, Reunião, mentoria, etc...).
Cenário principal:	
1. O Responsável CnE seleciona o evento em que quer adicionar o voluntário.	
2. O sistema apresenta os detalhes do evento.	
3. O responsável CnE clica em “Adicionar Voluntário”.	
4. O Sistema Mostra a lista de Voluntários aptos para o evento.	
5. O Responsável CnE seleciona o Voluntário a ser alocado.	
6. O sistema atualiza a alocação do Voluntário.	
7. O sistema envia uma notificação ao Voluntário por e-mail.	
Cenário alternativo: No passo 3 caso o evento já tenha alcançado o limite de de Voluntários alocado.	
3.1. O sistema deve apresentar a mensagem “O evento escolhido já tem o limite de Voluntários”.	
3.2. O usuário deverá desalocar algum Voluntário clicando no “-” ao lado do nome do Voluntário.	
3.3. Volta ao passo 3.	

Tabela 10 – Detalhamento de caso de uso: UC 9 - Recomendar Voluntário.

Nome do Caso de Uso	UC 9 - Recomendar Voluntário.
Pré-Condições	Existir um Voluntário que já tenha preenchido e enviado os termos de responsabilidade e voluntariado que ainda não tenha sido aprovado pelo Responsável na CnE.
Atores Envolvidos	Responsável pela Empresa.
Resumo	Aborda os passos necessários para o voluntário ser recomendado pelo Responsável pela Empresa.
Cenário principal:	
1. O Responsável pela Empresa seleciona o funcionário que deseja recomendar. **[Imagem 21]**	
2. O sistema mostra os detalhes do usuário. **[Imagem 22]**	
3. O Responsável Empresa clica em "Recomendar".	
4. O sistema envia um e-mail ao Responsável CnE e ao Voluntário informando da recomendação.	
Cenário alternativo: No passo 2 caso o usuário não esteja no status "Aguardando recomendação da Empresa"	
2.1. O sistema não mostrará o botão "Recomendar".	
2.2 Termina o caso de uso.	

Tabela 11 – Detalhamento de caso de uso: UC 10 - Avaliar Treinamento.

Nome do Caso de Uso	UC 10 - Avaliar Treinamento.
Pré-Condições	O voluntário ter participado de um treinamento e ainda não o ter avaliado.
Atores Envolvidos	Voluntário.
Resumo	Aborda os passos necessários para o voluntário avaliar o treinamento a ele oferecido.
Cenário principal:	
1. O voluntário deve clicar no evento anterior de treinamento na tela inicial **[Imagem 23]**	
2. O sistema mostra os um pop-up com o formulário avaliação. **[Imagem 24]**	
3. O Voluntário deve preencher o formulário e clicar em "Avaliar".	

4.3 PROTOTIPAÇÃO DE TELAS

Esta Seção apresenta alguns dos protótipos de tela para os casos de uso "Recomendar Voluntário"e "Avaliar Treinamento":

4.3.1 UC 9 - Recomendar Voluntário

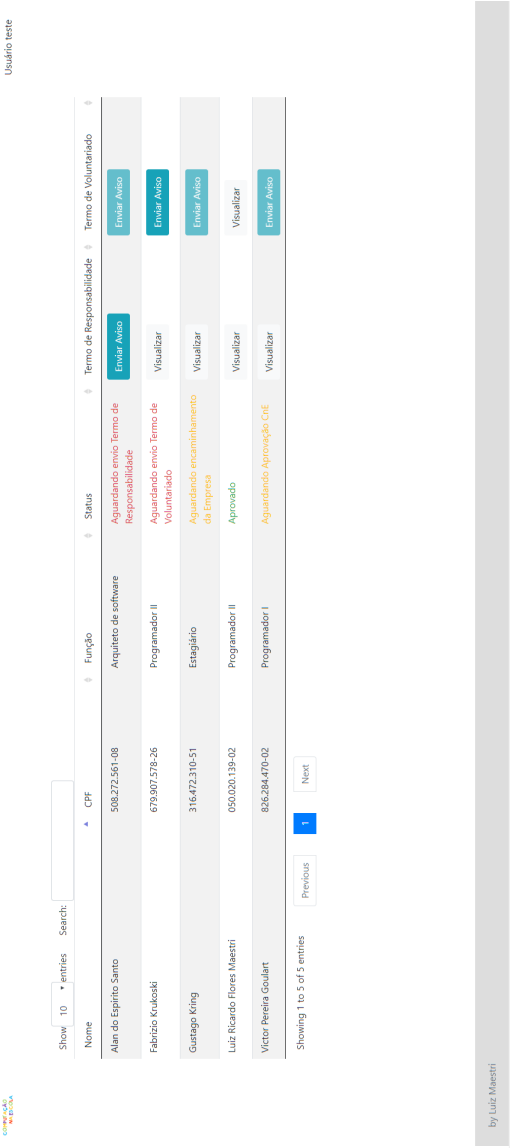


Figura 21 – Tela inicial do Responsável pela empresa.

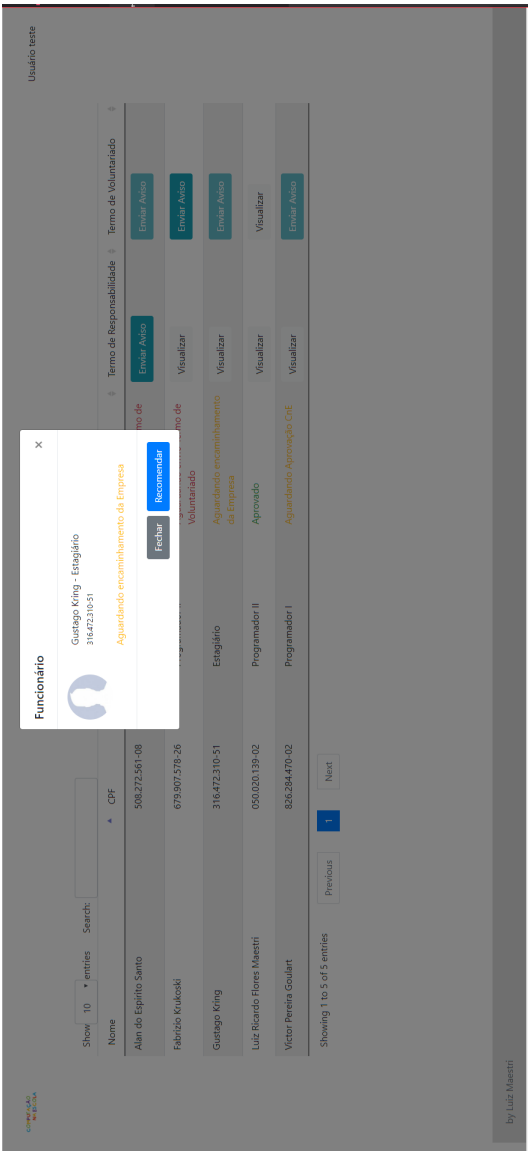


Figura 22 – Tela de detalhes do funcionário.

4.3.2 UC 10 - Avaliar Treinamento.

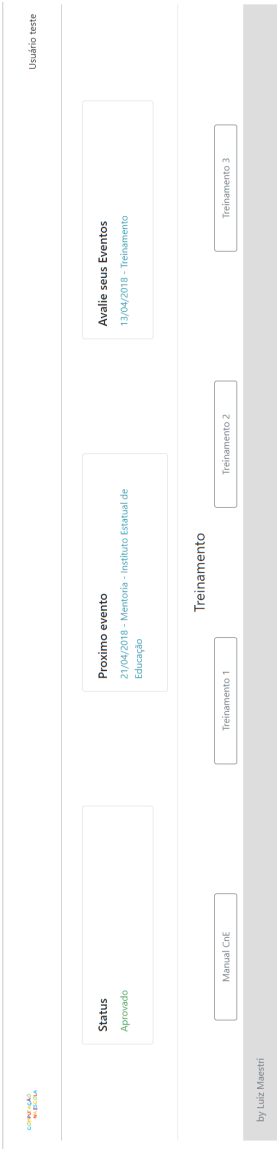


Figura 23 – Tela inicial do Voluntário.

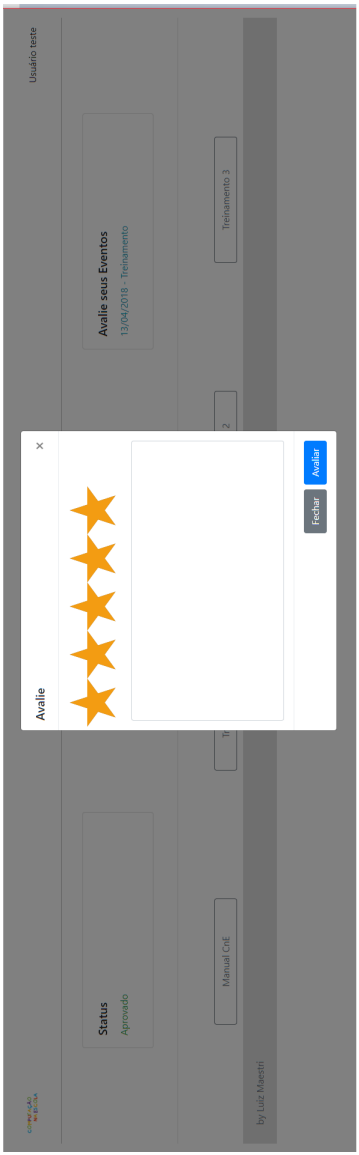


Figura 24 – Tela de avaliação de treinamento.

4.4 MODELAGEM DO BANCO DE DADOS

Esta seção apresenta informações de suma importância para que seja atendido o requisito "RNF 3 - O sistema deve ter integração com o Portal da CnE via banco de dado MySQL", na figura 25, é apresentado o mapeamento de entidades relacionais do sistema CuLTiVAR, sem as tabelas relativas ao site da CnE.

No presente trabalho, o sistema de gerenciamento de banco de dados (SGBD) adotado para desenvolver a ferramenta para gerenciar as inscrições das oficinas e o ambiente colaborativo foi o MySQL, SGBD esse aceito pelo WordPress (WORDPRESS, 2018), tecnologia em que foi criada o atual site da Computação na Escola e demais sistemas preexistentes da iniciativa.

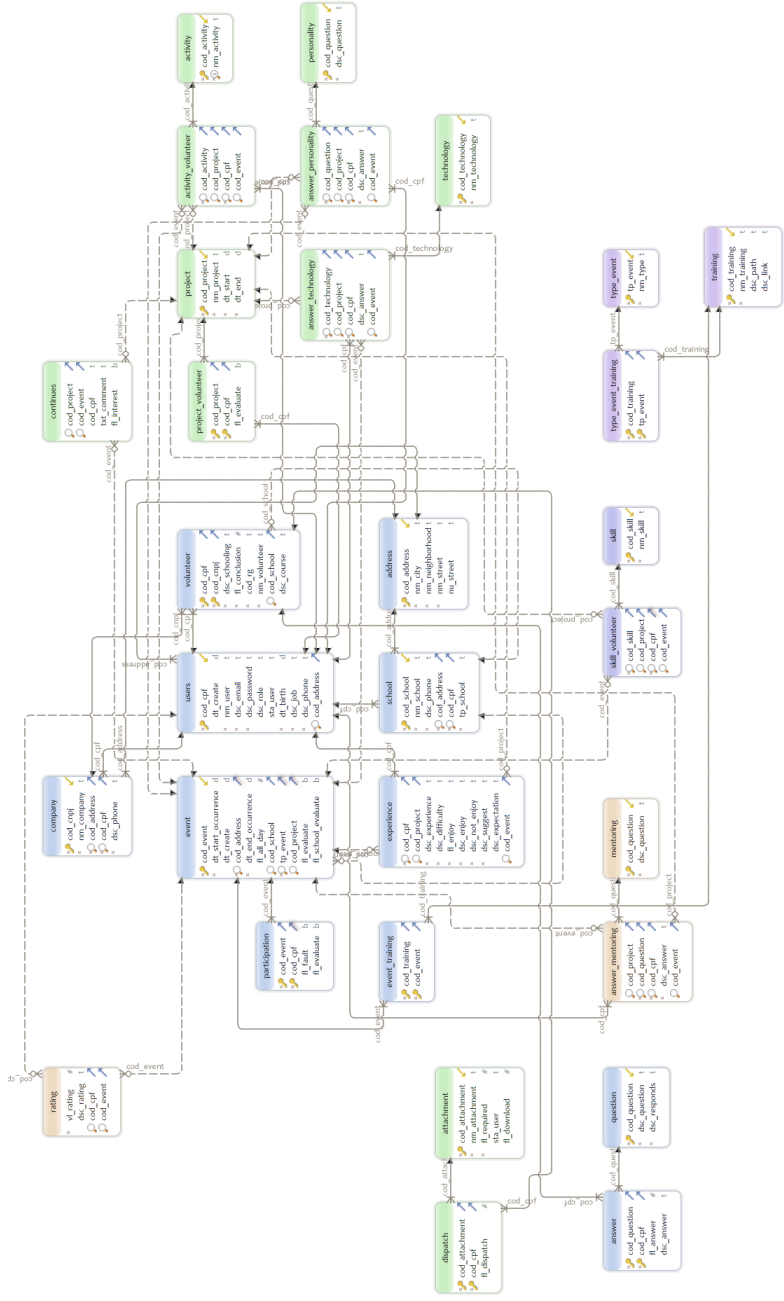


Figura 25 – Diagrama Entidade-Relacional

5 DESENVOLVIMENTO

Este Capitulo tem como objetivo apresentar passo a passo do desenvolvimento, explicando decisões de projeto como adoções de *Design Patterns*, tais como *Service*, *Builder*, *Immutable* entre outros. Na sequência são apresentados detalhes do desenvolvimentos das camadas de backend e de frontend.

5.1 BACKEND

Esta seção tem como objetivo apresentar a camada de backend do sistema desenvolvido, utilizando o framework spring e a biblioteca Java, Lombok.

5.1.1 Spring

Para a implementação do backend foi utilizado o framework Spring[1], mais especificamente o Spring Boot[2], um dos projetos da Pivotal[3], que tem como objetivo simplificar *setup* inicial para a plataforma, trazendo algumas configurações já realizadas internamente, e permitindo que sejam reescritas programaticamente ou por meio de um arquivo nomeado *application.properties*(PIVOTAL, 2018a). Exemplos de ambas configurações apresentadas, em ordem, na Figuras 26 e 27. O trecho de código apresentado, habilita a capacidade de execução assíncrona para o Spring, e injeta no seu contexto um objeto do tipo Executor para utilização (explicado na Tabela 12).

[1] https://spring.io/
[2] https://spring.io/projects/spring-boot
[3] https://pivotal.io/

```java
@Configuration
@EnableAsync
public class AsyncConfig {
    @Bean
    public Executor asyncExecutor() {
        ThreadPoolTaskExecutor executor = new
            ThreadPoolTaskExecutor();
        executor.setCorePoolSize(2);
        executor.setMaxPoolSize(2);
        executor.setQueueCapacity(50);
        executor.setThreadNamePrefix("email-");
        executor.initialize();
        return executor;
    }
}
```

Figura 26 – Código de configuração para funcionamento assíncrono

```
spring.datasource.url=jdbc:mysql://localhost:3306/cultivar?
    useSSL=false&allowPublicKeyRetrieval=true
spring.datasource.username========= //Dados sensveis
spring.datasource.password========= //Dados sensveis

logging.level.org.springframework.data=INFO
logging.level.org.springframework.jdbc.core.JdbcTemplate=INFO

scheduler.email.send.cron=0 27 3 * * *

spring.mail.host=smtp.gmail.com
spring.mail.port=465
spring.mail.username==========@gmail.com //Dados sensveis
spring.mail.password======== //Dados sensveis
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.properties.mail.smtp.starttls.required=true
spring.mail.properties.mail.smtp.ssl.enable=true
spring.mail.test-connection=true
```

Figura 27 – Exemplo de application.properties para configuração do projeto

O arquivo apresentado na figura 27 exemplifica a configuração por meio de arquivos de chave-valor, esse exemplo traz algumas configurações como conexão ao banco de dados, nível de *log* e credenciais para envio de e-mail. Por padrão esse arquivo deve ser salvo no servidor no mesmo diretório onde se encontra o executável gerado na compilação.

O framework Spring fornece abstrações, JSR-330[4], *Dependency Injection*, e utiliza da JSR-303[5], *Bean Validation*, por meio das *Java Annotations* *@Autowired* e *@Valid* respectivamente, ambas utilizadas para o projeto apresentado, como demonstrado no trecho de código abaixo:

```
1   @RestController
2   @RequestMapping(path = "/activity")
3   @AllArgsConstructor(onConstructor = @__(@Autowired))
4   @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
5   public class ActivityResource {
6
7       ActivityService service;
8
9       @PostMapping(consumes =
            MediaType.APPLICATION_JSON_UTF8_VALUE)
10      @ResponseStatus(code = HttpStatus.CREATED)
11      public void create(@Valid @RequestBody final Activity
            activity) {
12          service.create(activity);
13      }
14      ...
15      @DeleteMapping(path = "/{codQuestion}", produces =
            MediaType.APPLICATION_JSON_UTF8_VALUE)
16      public Activity delete(@PathVariable final Long
            codQuestion) throws ServiceException{
17          return service.delete(codQuestion);
18      }
19  }
```

Figura 28 – Classe ActivityResource

Outras *Java Annotations* do Spring apresentadas nos trechos de código acima são *@Configuration, @EnableAsync, @Bean, @RestController, @RequestMapping, @PostMapping, @DeleteMapping, @Respon-*

[4]https://www.jcp.org/en/jsr/detail?id=330
[5]https://www.jcp.org/en/jsr/detail?id=303

seBody, @RequestBody @ResponseStatus e *@PathVariable*, a Tabela 12 traz uma breve explicação de cada *Annotations* apresentada juntamente com *@Controller* e *@RequestBody*.

Tabela 12 – Spring *Java Annotations* (PIVOTAL, 2018b)

Identificação	Descrição
@Configuration	Indica que uma classe declara ao menos um método *@Bean* e deve ser processada pelo contêiner/contexto do Spring gerando definições de *bean*.
@EnableAsync	Habilita a capacidade de execução de métodos de modo assíncrono pelo Spring.
@Bean	Indica que o método gerará um *bean* gerenciado pelo contexto do Spring, geralmente utilizando o *pattern* de *Singleton*(explicado a anteriormente).
@Controller	A classe anotada com esta *Annotation* é considerada um *Controller* para o Spring e permite que a mesma seja detectada pelo scan de *classpath*.
@ResponseBody	Indica que o valor retornado pelo método deve ser deve ser encapsulado no corpo da resposta.
@RestController	Composição das *Annotations @Controller* e *@ResponseBody*.
@RequestBody	O parâmetro com está *Annotation* será extraído do corpo da requisição.
@RequestMapping	*Annotation* para mapeamento de requisições Web para métodos em classes de manipulação de Requisições.
@PostMapping	É um abstração da *Annotation @RequestMapping* para responsder somente requisições utilizando o verbo http POST.
@DeleteMapping	É um abstração da *Annotation @RequestMapping* para responsder somente requisições utilizando o verbo http DELETE.
@ResponseStatus	Mapeia o *status* http da resposta, caso não seja usado os *status* padrões são 200 para sucesso e 500 para qualquer exceção não tratada.
@PathVariable	Indica que o parâmetro é mapeado pela url da requisição, que é indicada na proriedade *path* da *Annotation @RequestMapping* e suas abstrações, a variável é indicada na propriedade por {propriedade}.

5.1.2 Project Lombok

O Lombok[6] é uma biblioteca java que utiliza-se de *Annotations* para gerar código em tempo de compilação, deixando o código Java mais simples e amigável, este é um exemplo de como código fica com as *Annotations* do Lombok, Figura 29.

```
1    @Value
2    @Wither
3    @Builder(builderClassName = "Builder")
4    @...
5    @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
6    public class School {
7        Long codSchool;
8        @...
9        String name;
10       @...
11       String phone;
12       @...
13       Address address;
14       @...
15       User responsible;
16       @...
17       SchoolType type;
18
19       ...
20   }
```

Figura 29 – Exemplo de Classe Java utilizando *Annotations* do Lombok

[6] https://projectlombok.org/

Para cada *Annotation* apresentada na figura Figura 29 será compilada para os trechos representados nas figuras 30, 31, 32 e 33.

A figura 30 apresenta alguns trechos de código gerados pela *Annotation @Value* responsável por gerar um construtor com todos os atributos da classe, os métodos de acesso, *Getters*, e os métodos equals e hashCode.

```
1   public class School {
2       ...
3       @java.beans.ConstructorProperties({"codSchool", "name",
            "phone", "address", "responsible", "type"})
4       School(Long codSchool, @NotBlank @NotNull String name,
            @NotBlank @NotNull String phone, @Valid @NotNull
            Address address, @Valid @NotNull User responsible,
            @NotNull SchoolType type) {
5           ...
6       }
7
8       public Long getCodSchool() {
9           return this.codSchool;
10      }
11
12      public @NotBlank @NotNull String getName() {
13          return this.name;
14      }
15      ...
16      public boolean equals(Object o) {
17          ...
18      }
19
20      public int hashCode() {
21          ...
22      }
23  }
```

Figura 30 – Código gerado pela *Annotation @Value*

A *Annotation @Wither* gera métodos, figura 31, para "alteração"dos atributos do objeto, explicado na seção 5.1.3.

```
1   public class School {
2       ...
3       public School withPhone(@NotBlank @NotNull String phone)
            {
4           return this.phone == phone ? this : new
                School(this.codSchool, this.name, phone,
                this.address, this.responsible, this.type);
5       }
6
7       public School withAddress(@Valid @NotNull Address
            address) {
8           return this.address == address ? this : new
                School(this.codSchool, this.name, this.phone,
                address, this.responsible, this.type);
9       }
10      ...
11  }
```

Figura 31 – Método gerado pela *Annotation @Wither*

A *Annotation @Builder*, como apresentado na figura 32, é utilizada para se trabalhar como o *Pattern Builder*, mais explicações na seção 5.1.3.

```
1   public class School {
2       ...
3       public static Builder builder() {
4           return new Builder();
5       }
6       ...
7        public static class Builder{
8           ...
9        }
10      ...
11  }
```

Figura 32 – código gerado pela *Annotation @Builder*

A *Annotation @FieldDefaults* tem o objetivo de configurar a visibilidade dos atributos da classe por ela anotada, como mostra a Figura 33

```
1   public class School {
2       private final Long codSchool;
3       @NotBlank
4       @NotNull
5       private final String name;
6       @NotBlank
7       @NotNull
8       private final String phone;
9       @Valid
10      @NotNull
11      private final Address address;
12      @Valid
13      @NotNull
14      private final User responsible;
15      @NotNull
16      private final SchoolType type;
17      ...
18  }
```

Figura 33 – código gerado pela *Annotation @FieldDefaults*

5.1.3 Design Patterns

Esta subseção é destinada a explicações das adoções dos *patterns* adotados no backend.

O Spring trabalha com o conceito de *Proxy*, um *Design Pattern* estrutural definido pela GoF. Esse *Design Pattern* encapsula um objeto através de um outro objeto que possui a mesma interface de comunicação, métodos, de forma que o segundo objeto, *Proxy*, controla o acesso ao primeiro (DEVMIDIA, 2006).

Outro *Pattern* do Spring é o *Singleton*, permite a criação de somente um instância da classe implementada utilizando o *Pattern*, evitando assim a criação de instâncias desnecessárias e por sua vez o consumo de memória.(DEVMIDIA, 2010)

Uma aplicação construída com Spring, é divida em camada chamadas no ecossistema da plataforma de *Stereotypes*, são elas, *Repository* para acesso ao banco de dados, comumente chamada de DAO , Service sendo a camada de negócio, sendo a camada mais robusta da aplicação e a camada de *Controller* que faz a comunicação entre a visão, *View*, e o *Service*.

Alguns dos *Design Patterns* usados no projeto foram *Builder, Service, DAO, DTO, PO, dependency injection* e *Immutable*, que serão exemplificados por meios dos trechos de códigos da figura 34 à figura 38:

```
1   @Value
2   @Wither
3   @Builder(builderClassName = "Builder")
4   @JsonDeserialize(builder = Address.Builder.class)
5   @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
6   public class Address {
7       Long codAddress;
8       @NotBlank
9       @NotNull
10      String city;
11      @NotBlank
12      @NotNull
13      String neighborhood;
14      @NotBlank
15      @NotNull
16      String street;
17      String number;
18
19      @JsonPOJOBuilder(withPrefix = "")
20      public static class Builder{
21          public Builder(){}
22      }
23  }
```

Figura 34 – Exemplo de PO com as *Annotations* do Lombok

Por meio das *Annotations @Value, @Wither, @Builder* e *@FieldDefaults* providas lombok, a classe *Address* é uma representação do modelo de dados, *PO*, com os atributos privados e *final*, impedindo alterações após a instanciação da classe, *immutable*, somente com os métodos de acesso e comparação gerados pela *Annotation @Value*. A *Annotation @Wither* gera métodos similares ao *Setters*, contanto, esses métodos não alteram o próprio objeto, mas instanciam e retornam um novo objeto com somente o atributo que o método propõe-se a alterar diferente do objeto original, como mostra a figura 35.

```
1     public Address withCodAddress(Long codAddress) {
2        return this.codAddress == codAddress ? this : new
             Address(codAddress, this.city, this.neighborhood,
             this.street, this.number);
3     }
```

Figura 35 – Exemplo método gerado pela *Annotation @Wither* do Lombok

```
1     @Repository
2     @AllArgsConstructor(onConstructor = @__(@Autowired))
3     @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4     public class PersonalityRepository {
5
6         NamedParameterJdbcTemplate jdbcTemplate;
7
8         ...
9
10        private Personality build(ResultSet rs , int i) throws
              SQLException {
11            return Personality.builder()
12                    .codQuestion(rs.getLong("cod_question"))
13                    .question(rs.getString("dsc_question"))
14                    .build();
15        }
16
17        ...
18    }
```

Figura 36 – Exemplo de uso de *Builder*

O código da figura 36 o uso de *Builder* para construção de objetos dentro do método **build(ResultSet rs , int i)**. A classe *PersonalityRepository* de acesso a base de dados comumente chamada de *DAO*. Dentro da arquitetura do Spring recebe o nome de *repository*. Sobre esta classe é também aplicado o *Pattern* de *dependency injection* por meio Annotation @Autowired, que será aplicada sobre o construtor gerado durante compilação pelo Lombok, figura 37.

```
1     @java.beans.ConstructorProperties({"jdbcTemplate"})
2     @Autowired
3     public PersonalityRepository(NamedParameterJdbcTemplate
       jdbcTemplate) {
4         this.jdbcTemplate = jdbcTemplate;
5     }
```

Figura 37 – Exemplo de uso de *Builder*

A figura 38 apresenta um exemplo de classe Service, este tipo de classe segundo Fowler et al. (2002) encapsulam a lógica de negócio da aplicação, controlando transações e coordenando respostas na implementação de suas operações. O método da classe apresentado na figura 38 é a criação de um novo usuário onde pode encontrar um validação do objeto *Address* antes da execução da inserção no banco de dados.

```
1     @Service
2     @...
3     public class UserService {
4         ...
5         public void create(final User user) throws
           ServiceException {
6             val address = user.getAddress();
7             if (!ValidateUtils.isValid(address)) {
8                 throw new InvalidException(null);
9             }
10            userRepository.create(
11                user.withAddress(
12                    address.withCodAddress(
13                        addressRepository.create(
14                            address
15                        )
16                    )
17                )
18            );
19
20        }
21        ...
22    }
```

Figura 38 – Classe *Service*

5.1.3.1 Immutable

Esta subseção visa explicar o motivo do uso de *immutable* e como a JVM trabalha com esse tipo de objetos. Infelizmente o servidor fornecida para implantação do sistema implementado tinha um hardware limitando quando a memória. Assim obrigando a utilização de patterns que utilizem uma meno quantidade deste recurso, tal qual, é o pattern Immutable, explicado abaixo. Os objetos imutáveis muitas vezes não são empregados por programadores, devido a uma superestimação do custo para criação de um novo objeto no lugar de somente alterar o existente. Porém, pode ser compensado por algumas das eficiências associadas a objetos imutáveis(ORACLE CORPORATION, 2014). Isso inclui uma otimização no coletor de lixo, visto que não se faz necessário a execução do código para proteger os objetos mutáveis contra corrupção(WIKIWIKIWEB, 2011)(ORACLE CORPORATION, 2014).

5.2 FRONT-END

Nesta seção são apresentados alguns detalhes da interface de interação com o usuário, como principais telas e funcionalidades e componentes.

5.2.1 React JS

Para implementação da camada de front-end foi utilizada a biblioteca de JavaScript React JS [7]. React foi pensado para que os elementos da tela podessem ter seus comportamentos encapsulados em forma de componentes flexíveis, independente e reutilizável(EIS, 2016)(TABLELESS, 2018), como mostram as figuras 39, 40.

[7] https://reactjs.org

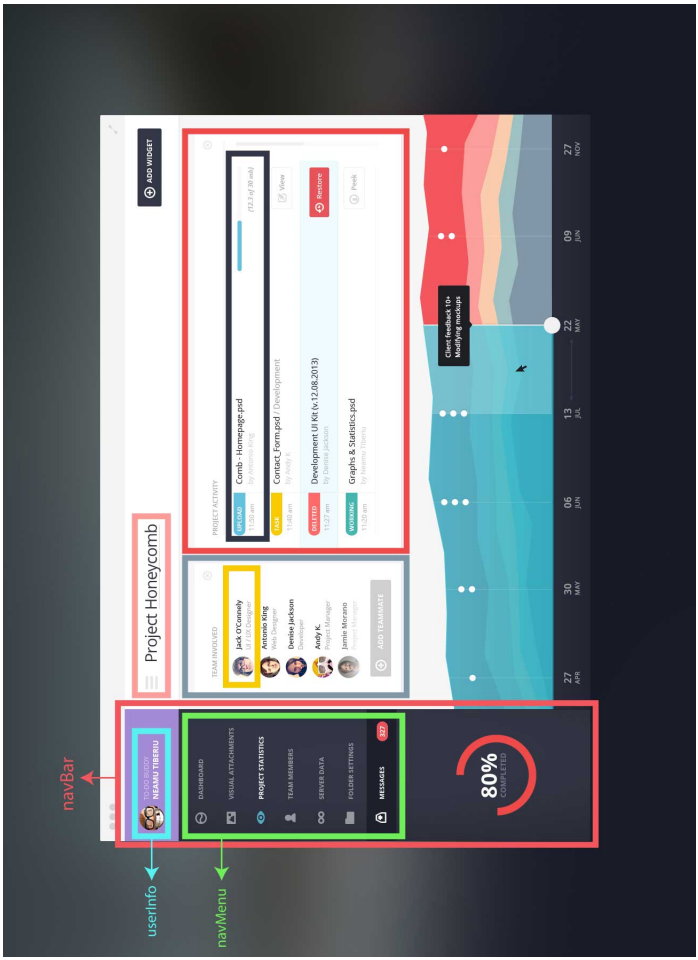


Figura 39 – Exemplos de Tela criada com React JS e componentes (EIS, 2016).

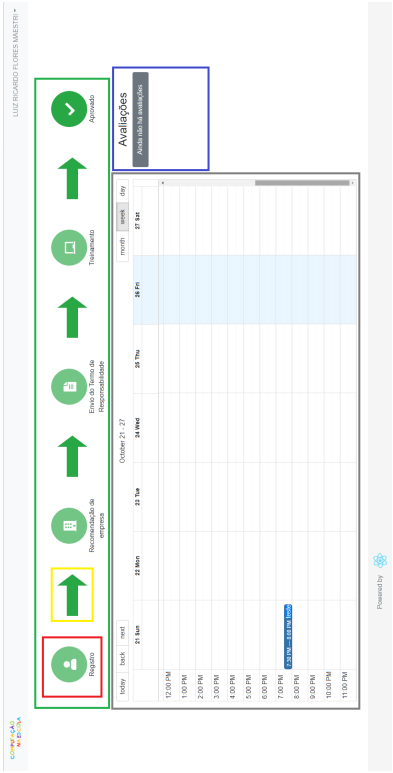


Figura 40 – Tela desenvolvida para o Sistema Cultivar com alguns componentes identificados.

Com a ideia de reuso de componentes em primeiro plano, foram utilizadas bibliotecas de componentes prontos, tais como: Reactstrap[8], um biblioteca de componentes já estilizados com bootstrap[9].

React-big-calendar[10] para criação de uma agenda com layout similar ao Google Calendar[11]

O código apresentado na figura 41 representa o componente da tela principal do voluntário responsável por iniciar a avaliação de um evento ou projeto que o voluntário tenha participado.

[8] http://reactstrap.github.io/
[9] https://getbootstrap.com
[10] https://www.npmjs.com/package/react-big-calendar
[11] https://www.google.com/calendar

```
import React, { Component, Fragment } from 'react';
import { Row, Col, Button } from 'reactstrap';
import Form from './form';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            isOpen : false
        };
        this.toggle = this.toggle.bind(this);
    }

    toggle(){
        const { isOpen } = this.state;
        this.setState({ isOpen: !isOpen });
    }

    render(){
        const { evaluate, cpf, afterSubmit } = this.props;
        const { isOpen } = this.state
        return (
            <Fragment>
                <Row>
                    <Col>
                        <Button color="primary" style={{
                            width: 'inherit', height:
                            '40px' }} onClick={this.toggle}>
                            {evaluate.title}
                        </Button>
                    </Col>
                </Row>
                {isOpen && <Form title={evaluate.title}
                    code={evaluate.codEvent ||
                    evaluate.codProject} cpf={cpf}
                    afterSubmit={afterSubmit}
                    isOpen={isOpen} close={this.toggle}
                    isProject={evaluate.isProject}/>}
                <br />
            </Fragment>
        )
    }
}
```

Figura 41 – Exemplo de Componente React.

O componente apresentado na figura 41, foi escrito utilizando ECMAScript 6[12] e JSX[13], que são, a linguagem script padronizada base para o JavaScript (MOZILLA, 2016) e uma extensão do ECMAScript(FACEBOOK, 2014), respectivamente.

Para Facilitar a explicação do código apresentado na figura 41, será explicado método a método nas figuras 42, 43 e 44.

A figura 42 demonstra o construtor do componente configurando o estado inicial do mesmo.

```
1    constructor(){
2        super();
3        this.state = {
4            isOpen : false
5        };
6        this.toggle = this.toggle.bind(this);
7    }
```

Figura 42 – Construtor do componente.

O método *toggle* demonstrado na figura 43 é responsável por abrir o formulário de avaliação alterando o estado do componente por meio do método *setState*, esta alteração faz com que o o componente execute a função *render*, explicada na figura 44, novamente utilizando agora o novo estado.

```
1    toggle(){
2        const { isOpen } = this.state;
3        this.setState({ isOpen: !isOpen });
4    }
```

Figura 43 – Método *toggle*.

A figura 44 apresenta o método *render*, que é o responsável por transformar renderizar o JSX na tela.

O componente é composto por outros componentes são eles:

- *Row* que será renderizado como uma *<div class="row»*.
- *Col* que será renderizado como uma *<div class="col»*.

[12] http://www.ecma-international.org/memento/tc39.htm
[13] https://facebook.github.io/jsx/

- *Button* que será renderizado como uma *<button type="button"class="btn btn-primary»* e executará o método *toggle* quando clicado.

- *Form*, que é o formulário de avaliação. em forma de um *modal*.

```
1   render(){
2       const { evaluate, cpf, afterSubmit } = this.props;
3       const { isOpen } = this.state
4       return (
5           <Fragment>
6               <Row>
7                   <Col>
8                       <Button color="primary" style={{ width:
                            'inherit', height: '40px' }}
                            onClick={this.toggle}>
9                           {evaluate.title}
10                      </Button>
11                  </Col>
12              </Row>
13              {isOpen && <Form title={evaluate.title}
                    code={evaluate.codEvent ||
                    evaluate.codProject} cpf={cpf}
                    afterSubmit={afterSubmit} isOpen={isOpen}
                    close={this.toggle}
                    isProject={evaluate.isProject}/>}
14              <br />
15          </Fragment>
16      )
17  }
```

Figura 44 – Método render.

5.2.2 Interfaces

Esta seção tem como objetivo demonstra a utilização das telas do Sistema Cultivar.

5.2.2.1 Login

Na figura 45 é demonstrada a tela de *login* com um campo Usuário(1) e Senha(2) que o usuário deve preencher com seu cpf/email e senha no sistema, e clicar no botão "Entrar"(3) para realizar sua autenticação no sistema. Logo abaixo do botão "Entrar"há o link "Quero ser voluntário"(4) que redireciona para o formulário inicial de cadastro do voluntário.

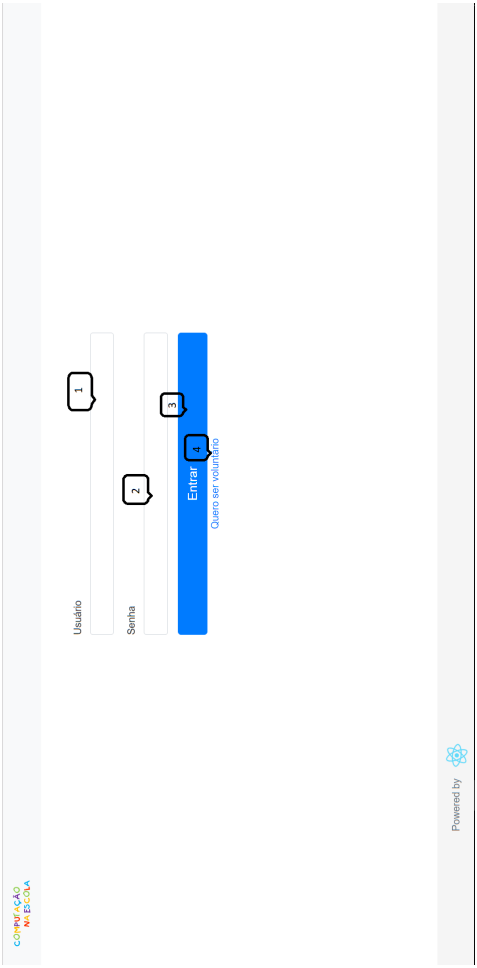


Figura 45 – Tela de Login.

5.2.2.2 Voluntário

As figuras 46, 47 e 48 demonstram o formulário inicial, termo de voluntariado, para o seu cadastro como de acordo com os casos de uso UC11 e UC13.

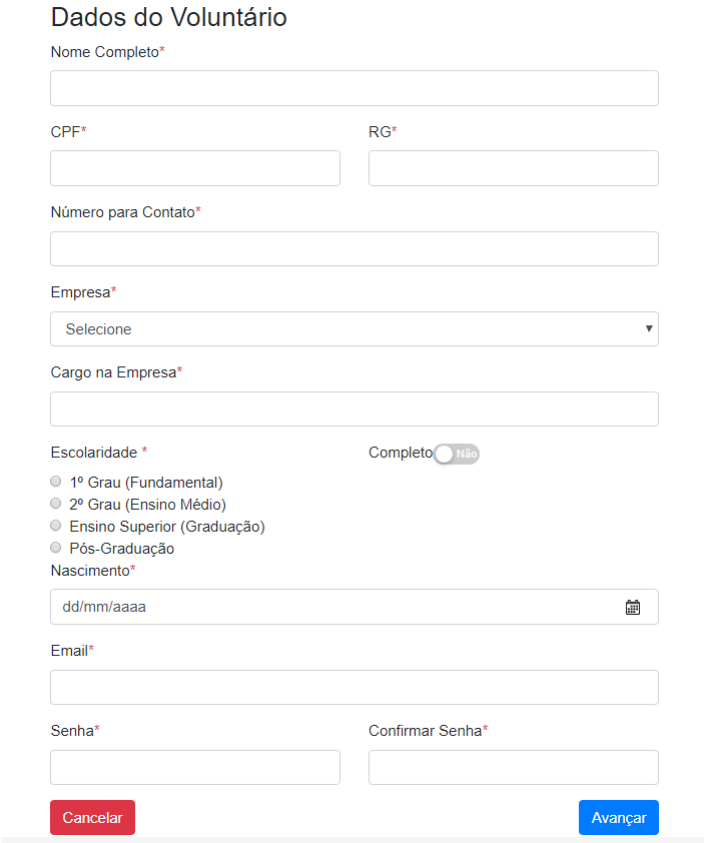


Figura 46 – Primeiro passo do *wizard* de registro do de voluntário.

Figura 47 – Segundo passo do *wizard* de registro do de voluntário.

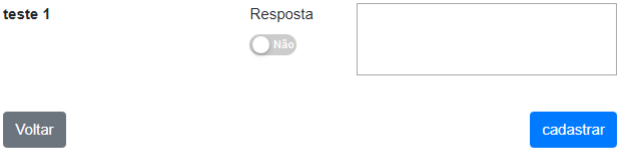


Figura 48 – Último passo do *wizard* de registro do de voluntário.

A Figura 49 apresenta a tela do voluntário na primeira vez que entrar no sistema onde permite que o mesmo termine seu registro realizando *uploads* dos arquivos solicitados/necessários, como especificado pelo RF 2 e UC11.

Após executar o envio dos arquivos o usuário será transferido para tela presente na figura 50, onde terá de esperar o Responsável da empresa ligado a ele o recomendar para a CnE.

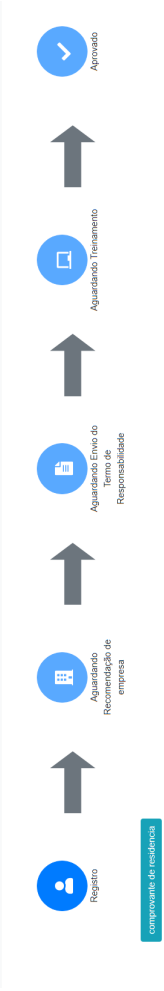


Figura 49 – Primeiro Login do voluntário.

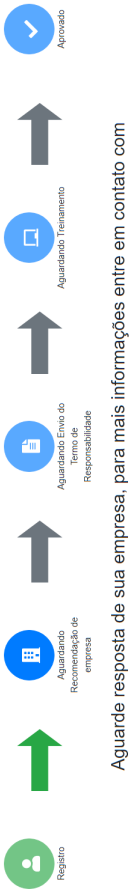


Figura 50 – Tela de Espera pela recomendação da empresa.

Na Figura 51 apresenta a tela referente ao UC12 - Gerar termo de responsabilidade, onde o usuário poderá realizar o *download* e *upload* do mesmo.

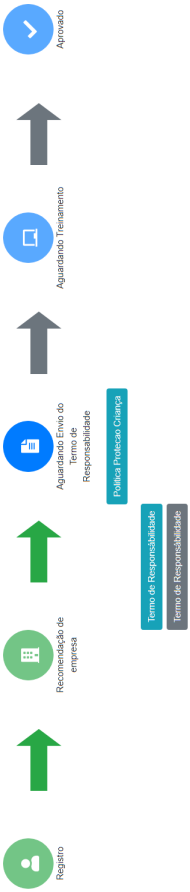


Figura 51 – Tela de envio do termo de responsabilidade.

A figuras 52 e 53 apresentam o *dashbord* principal do Voluntário. Onde ele pode visualizar os eventos do quais participará pela CnE, e poderá avaliar os eventos anteriores como explicitado nos UCs 6 e 10.

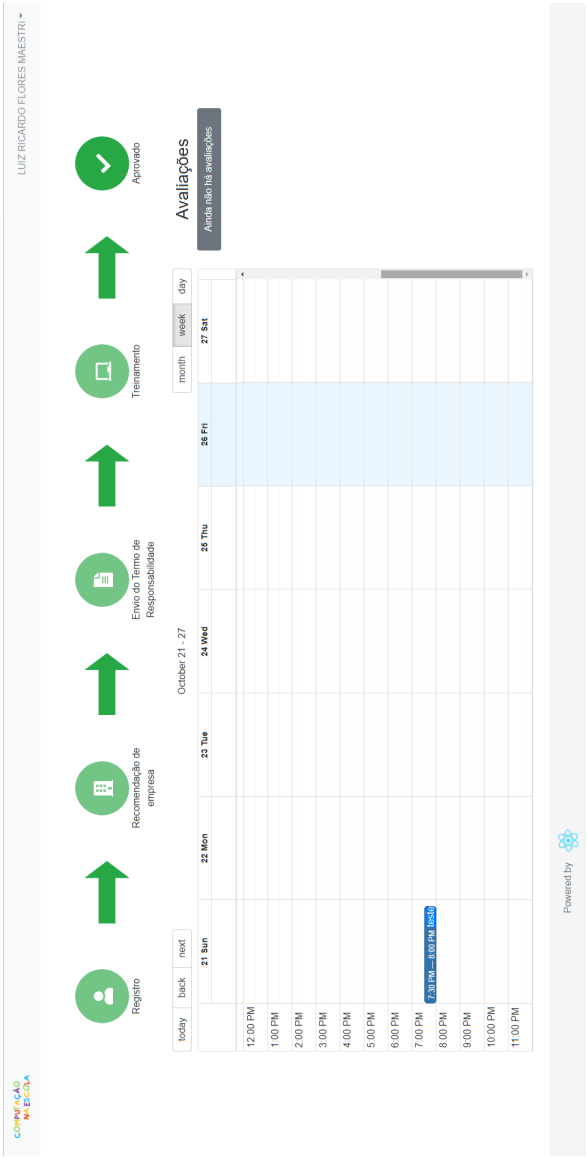


Figura 52 – *Dashboard* voluntário.

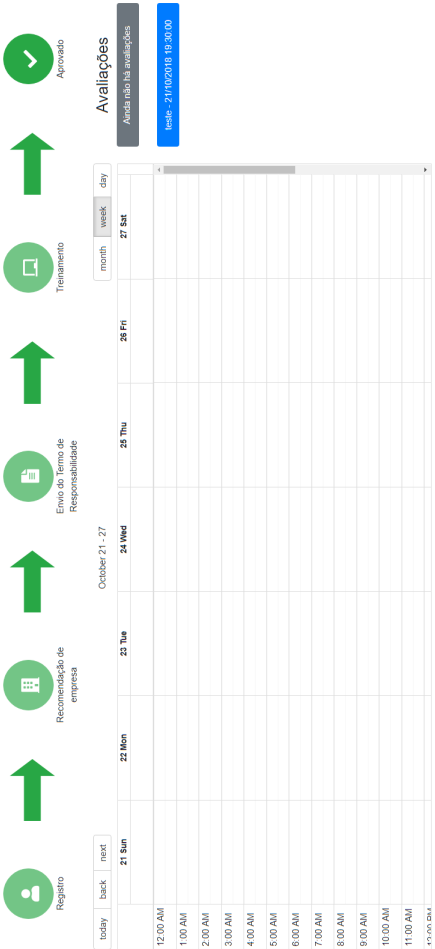


Figura 53 – *Dashboard* voluntário.

Ao clicar no evento marcado no calendário ou no botão em avaliações abrirá um *pop-up*, figura 54, com dados do evento ou o um *pop-up*, figura 55, com o formulário para avaliar o evento já ocorrido como previstos nos UC 6 e 10.

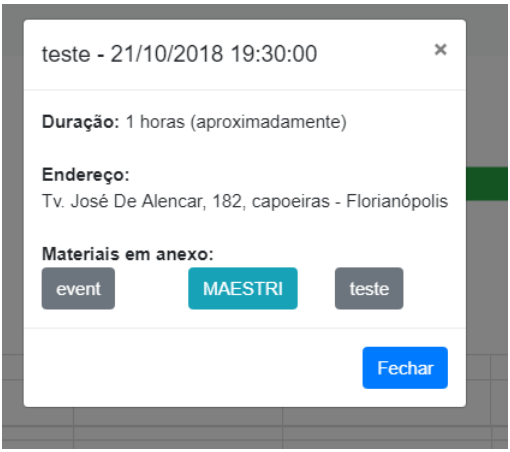


Figura 54 – Detalhes do evento.

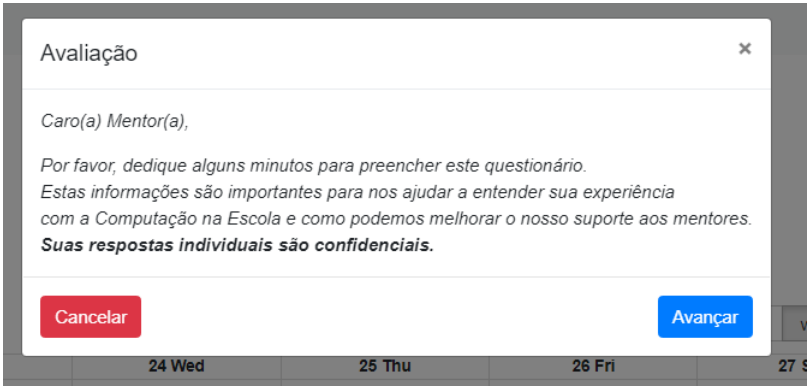


Figura 55 – Formulário de avaliação de eventos do Voluntário.

5.2.2.3 Responsável da Empresa

A figura 56 apresenta a tela principal do Responsável da empresa, onde lhe é apresentado uma listagem dos voluntários da empresa(1) e o Responsável a empresa pode realizar as seguintes ações:

- Filtrar a tabela pelo nome do voluntário(2), após a terceira letra começará a filtrar.
- Visualizar o status da candidatura do Voluntário(3).
- Visualizar mais detalhes do Voluntário(4), como previsto no UC11. Ao clicar no botão abrirá um *pop-up* com as informações do Voluntário como pode ser visto na figura 57.
- Visualizar locais onde o Voluntário já esteve representando a empresa na CnE(5).
- Recomendar a CnE a candidatura do Voluntário(6), como previsto no UC9. Ao clicar no botão abrirá um *pop-up* com com perguntas de sim ou não sobre a recomendação do Voluntário como pode ser visto na figura 58.
- Deletar o Voluntário(7), como previsto no UC11, quando o mesmo não tem nenhum evento futuro associado a ele.

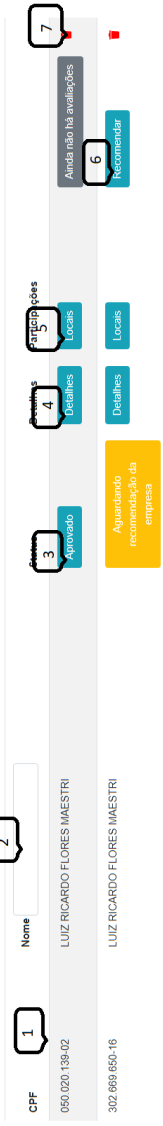


Figura 56 – Tela de Espera pela recomendação da empresa.

Figura 57 – Tela de detalhes do usuário.

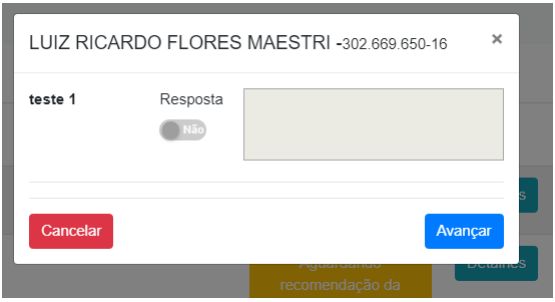


Figura 58 – Tela de recomendação da empresa.

5.2.2.4 Responsável na CnE

A figura 59 apresenta a tela inicial do Responsável na CnE. Onde o mesmo pode realiza:

- UC 8 - CRUD de Empresa, clicando nos itens 1 e 2 que abrirá um *pop-up* com o formulário para cadastro da empresa ou a excluirá a empresa, respectivamente.
- UC 2 - CRUD Evento, selecionando uma faixa de tempo dentro do calendário.
- UC 7 - CRUD de escola, clicando nos itens 4 e 5 que abrirá um *pop-up* com o formulário para cadastro da escola ou a excluirá a empresa, respectivamente.

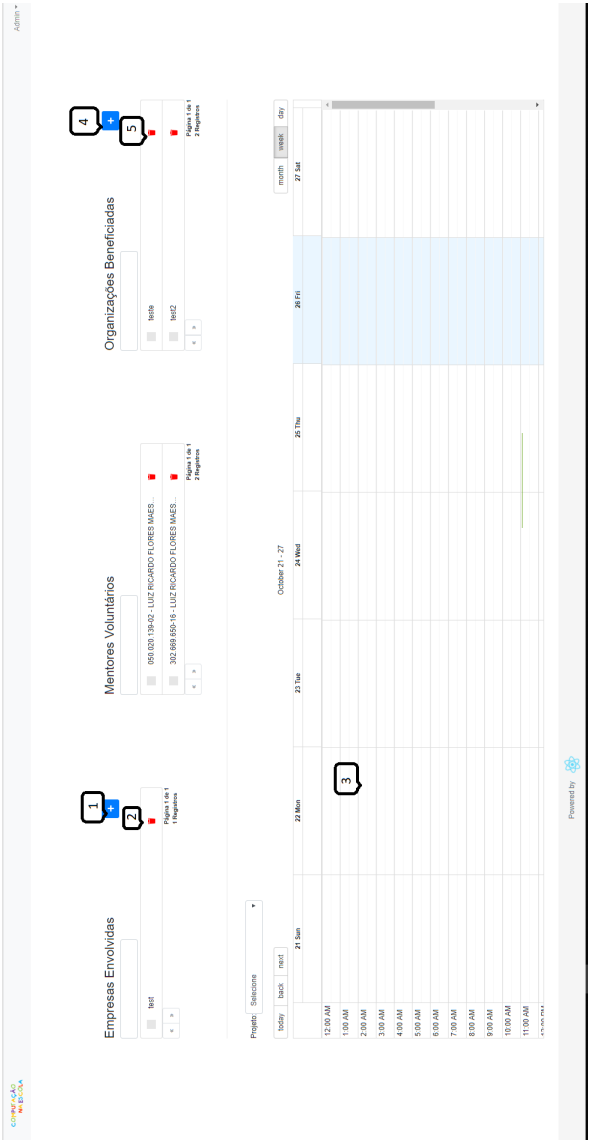


Figura 59 – *Dashboard* Responsável na CnE.

6 AVALIAÇÃO DO SISTEMA CULTIVAR

Com o objetivo de avaliar a qualidade do Sistema Cultivar foram realizadas avaliações com usuários entre os dias 20 e 26 de outubro de 2018, em relação a utilidade, funcionalidade, eficiência e usabilidade pela perspectiva de dos voluntários.

6.1 TESTE COM USUÁRIOS

Segundo Konaiagari (2017) o teste com usuários é projetado para avaliar o produto segundo a visão de usuários reais, buscando descobrir problemas e pontos de melhorias(CAELUM, 2018).

Os testes foram feitos com ex-voluntários da CnE, para que houvesse uma comparação do processo atual, realizado manualmente, com o proporcionado pelo sistema desenvolvido neste projeto.

Todos os voluntários participaram da CnE entre os anos de 2014 e 2015, quando estavam entre 2ª e 4ª fase dos cursos de Ciências da Computação e Sistemas de Informação. Os mesmos foram selecionados por meio do grupo do curso de Sistemas.

O objetivo desta avaliação é analisar a qualidade do Sistema Cultivar em termos de utilidade, funcionalidade, e usabilidade pelo ponto de vista dos voluntários.

6.1.1 Utilidade

Sobre a utilidade do sistema foi aplicado um breve questionário para usuários, apresentado na tabela 13. De forma geral os voluntários consideraram o sistema útil para sua finalidade de facilitar o "dia-a-dia"dentro da Computação na Escola, apesar de reclamações sobre os email enviados diariamente ao invés de semanalmente alertando de compromissos da CnE e da falta de sincronização com o *google calendar*.

Sendo este o motivo pelo qual metade dos usuários consideraram muito mais incomodo do que útil para essa função.

Questão de Análise	SIM	NÂO
O Sistema Cultivar foi útil para sua candidatura a vaga de voluntário?	4	0
O Sistema Cultivar foi útil para organizar-se com os horários das oficinas/projetos?	2	2
O Sistema Cultivar foi útil para auxiliar a preparação das oficinas?	4	0

Tabela 13 – Respostas do questionário de utilidade

6.1.2 Funcionalidade

Para o quesito funcionalidade, o sistema foi avaliado de maneira levemente positiva pelos usuários como apresentado na tabela 14, não apresentando erros durante a execução do módulo do voluntário.

Entretanto foi quase unânime para os usuários que somente uma *label* colorida não é o suficiente para o usuário compreender o motivo do *feedback* da escola quando negativo. Mesmo que compreendam que os textos de *feedbacks* fornecidos pelas escolas possam acabar ferindo suas pessoas caso expostos sem um "filtro"de linguagem antes.

Questão de Análise	SIM	NÂO
Você acha que as informações disponibilizadas no evento são suficientes?(localização, data e horário)	4	0
Você observou algum erro *(bug)* em relação a funcionalidade da ferramenta?	0	4
Você acha que as informações de feedback são suficiente?	1	3

Tabela 14 – Respostas do questionário de Funcionalidade

6.1.3 Usabilidade

A avaliação de usabilidade, em termos de satisfação foi realizada por meio de um diagrama de SUS, que consiste em um questionário de 10 perguntas, com respontas em uma escala de 1 a 5, onde 1 significa Discordo Completamente e 5 significa Concordo Completamente(TEIXEIRA, 2015). São as questões(TEIXEIRA, 2015):

1. Eu acho que gostaria de usar esse sistema com frequência.
2. Eu acho o sistema desnecessariamente complexo.
3. Eu achei o sistema fácil de usar.
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o sistema.
5. Eu acho que as várias funções do sistema estão muito bem integradas.
6. Eu acho que o sistema apresenta muita inconsistência.
7. Eu imagino que as pessoas aprenderão como usar esse sistema rapidamente.
8. Eu achei o sistema atrapalhado de usar.
9. Eu me senti confiante ao usar o sistema.
10. Eu precisei aprender várias coisas novas antes de conseguir usar o sistema.

Observou-se avaliações acima de 85 pontos. Como apresentado na Tabela 15, as respostas de usuário para cada questão são apresentadas na tabela 16.

Tabela 15 – Resultado da pesquisa SUS

Pontuação SUS				Média
97,5	92,5	92,5	87,5	92,5

Tabela 16 – Respostas dos voluntários

Voluntário	1	2	3	4	5	6	7	8	9	10	Nota
Victor Goulart	5	1	5	1	5	1	5	1	4	1	97,5
Felipe Senna	5	1	5	1	4	1	4	1	4	1	92,5
Fabrizio Krukoski	5	2	4	1	5	1	5	1	4	1	92,5
Gabriela Mattos	4	2	4	1	5	2	5	1	4	1	87,5

6.1.4 Pontos Fortes

Baseado no feedback dos usuários o modulo de voluntário do Sistema Cultivar, é considerado simples de usar por adotar padrões de mercado como o uso do componente de calendário similar ao componente utilizado no Google Calendar.

Motivo pelo qual os voluntários sentiram-se bem familiarizados com o sistema, não precisando de muito tempo para se adaptarem ao sistema.

6.2 AMEAÇAS A VALIDADE DA AVALIAÇÃO

A maior ameaça a validade da avaliação feita é a quantidade de usuários disponíveis para avaliar a ferramenta e a falta de acesso a usuários representantes dos atores Responsável da Empresa e Responsável da Escola, devido a falta de tempo para o contato com os mesmos, entende-se que para uma maior precisão e confiabilidade do resultados é necessário mais usuários e representantes dos atores não testados.

7 CONCLUSÃO

O objetivo principal deste trabalho foi desenvolver e avaliar uma ferramenta web de gestão do processo de voluntariado para o ensino de computação nas escolas, no contexto da Iniciativa Computação na Escola do Departamento de Informática e Estatística da UFSC. Neste contexto, foi feita a análise da fundamentação teórica sobre gestão de voluntariado e desenvolvimento web (O1).

Também foi realizada a análise do estado da arte identificando as principais web ferramentas de auxílio ao voluntariado (O2).

Deste modo, baseando-se na fundamentação teórica e na análise do estado da arte, juntamente a entrevistas com responsáveis pela Computação na Escola, foram elicitados os requisitos, modelado a base de dados e arquitetura do sistema, para viabilizar a implementação do mesmo (O3).

Os resultados da avaliação do Sistema Cultivar fornecem uma indicação inicial da sua utilidade, funcionalidade, desempenho e usabilidade muito positiva (O4), porém ainda incompleta.

O sistema Cultivar permite cadastro e avaliação dos mentores voluntários, disponibilizando uma ferramenta de feedback mais segura para a CnE, e simplificando a iniciativa do voluntário de participar como mentor.

Como trabalhos futuros, recomenda-se algumas pequenas alterações da interface para deixa-lá mais amigável a dispositivos móveis permitindo a compatibilidade com PWAs, aplicações webs que, nos dispositivos moveis, progressivamente assumem funcionalidades que antes eram possíveis somente em aplicações nativas(LIMA, 2017). E com o possível aumento dos voluntários em projetos da Computação na Escola, a troca da máquina servidor para uma máquina mais robusta.

REFERÊNCIAS

ABDUL-ALIM, J. Students not prepared for careers in computer science. *Diverse – Issues in Higher Education*, 2015. <http://diverseeducation.com/article/77408/>. Acessado em 10/09/2017.

ALLENDE, F. O entra e sai nas universidades. *G1*, São Paulo, Brasil, p. 1–1, Out 2012. <http://g1.globo.com/sp/santos-regiao/blog-do-allende/platb/2012/10/17/o-entra-esai-nas-universidades/>. Acessado em 21/09/2017.

BERKENBROCK, C. D. M. Mestre em Ciências da Computação, *Investigação e Implementação de Estratégias de Notificação de Invalidação para Coerência de Cache em Ambientes de Computação Móvel sem Fio*. Florianópolis, Brasil: [s.n.], 2005. <https://repositorio.ufsc.br/bitstream/handle/123456789/102464/214169.pdf?sequence=1>. Acessado em 24/05/2018.

BETTER IMPACT. *Dashboard*. 2018. <https://www.betterimpact.co.uk/>. Acessado em 16/06/2018.

BRASIL. Lei nº 9.608, de 18 de fevereiro de 1998. dispõe sobre o serviço voluntário e dá outras providências. *Diário Oficial [da] República Federativa do Brasil*, Brasília, DF, 18 fev. 1998. <http://www.planalto.gov.br/ccivil_03/LEIS/L9608.htm>. Acessado em 03/06/2018.

CAELUM. *Apêndice - Testes de Usabilidade*. 2018. <https://www.caelum.com.br/apostila-ux-usabilidade-mobile-web/usabilidade/>. Acessado em 30/10/2018.

CAIVANO, R. M.; VILLORIA, L. N. *Aplicaciones WEB 2.0 - Google Docs*. Villa María, Argentina: Editorial Universitaria de Villa María, 2009.

CENTRO DE VOLUNTARIADO DE SãO PAULO. *Palestra Voluntariado e a Transformação Social - 2016*. 2016. <http://www.voluntariado.org.br/sms/files/2016%20%20Palestra%20Voluntariado%20e%20Transforma%C3%A7%C3%A3o%20Social.pdf>. Acessado em 25/05/2018.

CNE, C. N. E. F. *Iniciando o projeto Jovens Tutores de Programação*. 2017. <http://www.computacaonaescola.ufsc.br/?p=2217>. Acessado em 14/09/2017.

DEVMIDIA. *Design Patterns – Singleton – Parte 3*. 2006. <https://www.devmedia.com.br/conheca-o-pattern-proxy-gof-gang-of-four/4066>. Acessado em 25/10/2018.

DEVMIDIA. *Conheça o Pattern Proxy - GoF (Gang of Four)*. 2010. <https://www.devmedia.com.br/design-patterns-singleton-parte-3/16782>. Acessado em 25/10/2018.

DEVMIDIA. *Spring Boot: simplificando o Spring*. 2015. <https://www.devmedia.com.br/spring-boot-simplificando-o-spring/31979>. Acessado em 26/10/2018.

DIáRIO CATARINENSE. Terceira edição do de 2012 do projeto amigos da escola acontece hoje. *Diário Catarinense*, Florianópolis, Brasil, 2012. <http://dc.clicrbs.com.br/sc/noticias/noticia/2012/08/terceira-edicao-do-de-2012-do-projeto-amigos-da-escola-acontece-hoje-3871028.html>. Acessado em 16/06/2018.

DO-IT. *Dashboard*. 2018. <https://do-it.org/>. Acessado em 16/06/2018.

DUBOW, W. *By the numbers*. Boulder, 2013. <http://www.ncwit.org/bythenumbers>. Acessado em 02/09/2017.

EIS, D. *Hello World com React, do rascunho até o primeiro componente*. 2016. <https://tableless.com.br/hello-world-com-react-do-rascunho-ate-o-primeiro-componente/>. Acessado em 30/10/2018.

EISENMAN, B. *Learning React Native*. Sebastopol, Califórnia, EUA: O'Reilly Media, 2015.

FACEBOOK. *Draft: JSX Specification*. 2014. <https://facebook.github.io/jsx/>. Acessado em 30/10/2018.

FACEBOOK. *Flux*. 2015. <https://facebook.github.io/flux/>. Acessado em 23/05/2018.

FACEBOOK. *React*. 2018. <https://reactjs.org/>. Acessado em 23/05/2018.

FIELDING, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Tese (Doutorado em Ciências da Computação) — UNIVERSITY OF CALIFORNIA, Irvine, EUA, 2000. <https://www.ics.uci.edu/ fielding/pubs/dissertation/top.htm>. Acessado em 23/05/2018.

FOWLER, M. et al. *Patterns of Enterprise Application Architecture*. Boston, Massachusetts, EUA: Addison-Wesley Professional, 2002.

GLOBO, R. *Manual AMIGOS DA ESCOLA - Guia de Ação*. 2012. <http://download.globo.com/amigosdaescola/manual.pdf>. Acessado em 30/11/2018.

GOLDSTEIN, I. S. *Responsabilidade Social - Das Grandes Corporações ao Terceiro Setor*. São Paulo, Brasil: Ed.Atica, 2007.

HUNT, P. *Why did we build React?* 2013. <https://reactjs.org/blog/2013/06/05/why-react.html>. Acessado em 16/06/2018.

INEP. *Sinopse Estatistíca de Educação Superior 2015*. Brasília, 2016. <http://portal.inep.gov.br/web/guest/sinopses-estatisticas-da-educacao-superior>. Acessado em 14/09/2017.

INEP. *Censo Escolar da Educação Básica 2016 : Notas Estatísticas*. Brasília, 2017. <http://download.inep.gov.br/educacao_basica/censo_escolar/notas_estatisticas/2017/notas_estatisticas_censo_escolar_da_educacao_basica_2016.pdf>. Acessado em 14/09/2017.

JENKINS, T. *THE MOTIVATION OF STUDENTS OF PROGRAMMING*. Tese (Mestrado em Ciências da Computação) — University Of Kent, Cantuária, Reino Unido, 2001. <https://www.cs.kent.ac.uk/pubs/2001/1401/content.pdf>. Acessado em 21/09/2017.

JENKINS, T. On the difficulty of learning to program. In: *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*. Leeds, Reino Unido: [s.n.], 2002. v. 1, p. 53–58. <http://www.psy.gla.ac.uk/ steve/localed/jenkins.html/>. Acessado em 20/08/2017.

KNOW HOW NON PROFIT(ORG.). *HOW TO CHOOSE A VOLUNTEER MANAGEMENT SYSTEM*. 2018.

<https://knowhownonprofit.org/how-to/how-to-choose-a-volunteer-management-system>. Acessado em 22/04/2018.

KONAIAGARI, A. *Teste de usuário: Introdução, benefícios, como criar e melhores práticas*. 2017. <https://imasters.com.br/design-ux/teste-de-usuario-introducao-beneficios-como-criar-e-melhores-praticas>. Acessado em 30/10/2018.

LACERDA, M. Informática como disciplina obrigatória na educação básica. In: *Anais do Encontro Virtual de Documentação em Software Livre e Congresso Internacional de Linguagem e Tecnologia Online*. Belo Horizonte, Brasil: [s.n.], 2012. v. 1, n. 1. <http://www.periodicos.letras.ufmg.br/index.php/anais_linguagem_tecnologia/article/view/1882>. Acessado em 03/11/2017.

LANGDON, D. et al. Stem: Good jobs now and for the future. *ESA Issue Brief #03-11*, US Department of Commerce, Jul 2011. <http://www.esa.doc.gov/sites/default/files/stemfinalyjuly14_1.pdf>. Acessado em 21/09/2017.

LIMA, M. *Introdução aos Progressive Web Apps*. 2017. <https://tableless.com.br/introducao-aos-progressive-web-apps/>. Acessado em 02/12/2018.

MARCOS, V.; AMADOR, C. A gestão do voluntariado. *EMPREENDEDORISMO SOCIAL EM PORTUGAL*, UNIVERSIDADE DO PORTO, 2014. <https://repositorio-aberto.up.pt/bitstream/10216/74145/2/88717.pdf>. Acessado em 12/06/2018.

MARQUES, D. L. et al. Atraindo alunos do ensino médio para a computação: Uma experiência prática de introdução a programação utilizando jogos e python. In: *SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO*. Aracajú, Brasil: [s.n.], 2011. p. 1138 – 1147. <http://www.br-ie.org/pub/index.php/wie/article/view/1954/1713>. Acessado em 20/11/2017.

MELLO, S. P. T. de et al. O fenômeno evasão nos cursos superiores de tecnologia: Um estudo de caso em uma universidade pública no sul do brasil. In: *Xiii Coloquio de Gestión Universitaria En Américas*. Arequipa, Peru: [s.n.], 2013. v. 0, n. 0, p. 1–10. <https://repositorio.ufsc.br/bitstream/handle/123456789/113096/2013129 - O fenômeno evasão nos cursos

superiores.pdf?sequence=1&isAllowed=y>. Acessado em 01/09/2017.

MENDES, N. C. *Análise do Perfil Socioeconômico e Motivacional dos Voluntários da Copa do Mundo FIFA 2014*. Tese (BACHERELADO EM TURISMO) — UNIVERSIDADE DE BRASÍLIA, Brasília, Brasil, 2015. <http://www.bdm.unb.br/bitstream/10483/12986/1/2015_NathaliaCorreaMendes.pdf>. Acessado em 14/06/2018.

MORA, S. L. *Programación de aplicaciones web: historia, principios básicos y clientes web*. Alicante, Espanha: Editorial Club Universitario, 2002.

MOZILLA. *Recursos de linguagem JavaScript*. 2016. <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Language$_Resources$>.*Acessadoem*30/10/2018.

ORACLE CORPORATION. *Immutable Objects*. 2014. <https://docs.oracle.com/javase/tutorial/essential/concurrency/immutable.htm Acessado em 26/10/2018.

ORACLE CORPORATION. *MySQL | O Banco de Dados de Código Aberto mais Conhecido no Mundo*. 2018. <https://www.oracle.com/br/mysql/index.html>. Acessado em 16/06/2018.

PETERSEN K., F. R. M. S. . M. M. Systematic mapping studies in software engineering. In: *Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*. Swindon, Reino Unido: [s.n.], 2008. p. 68–77. <http://www.robertfeldt.net/publications/petersen$_ease08_ysmap_studies_in_se.pa$

PIVOTAL. *Spring*. 2018. <https://spring.io/>. Acessado em 25/05/2018.

PIVOTAL. *Spring Docs*. 2018. <https://docs.spring.io/>. Acessado em 25/10/2018.

SAMARITAN TECHNOLOGIES. *Home Page*. 2018. <https://samaritan.com/>. Acessado em 16/06/2018.

SILVA, A. M. Dilemas do voluntariado na educaÇÃo em aracaju/se. *IV Jornada Internacional de politicas públicas*, São Luís, Brasil, 2009. <http://www.joinpp.ufma.br/jornadas/joinppIV/OLD/eixos_OLD/11.%20Impasses%20e%20Desafios%20das%20Politicas%20de%20Educa%

%A7%C3%A3o/DILEMAS%20DO%20VOLUNTARIADO%20NA%20EDUCA%C3%87%C3%83O%20EM%20ARACAJU%20SE1.pdf>. Acessado em 16/06/2018.

TABLELESS. *Componentes alem do reuso*. 2018. <https://tableless.com.br/componentes-alem-do-reuso/>. Acessado em 30/10/2018.

TANENBAUM, A. S.; STEEN, M. V. *Distributed Systems: Principles and Paradigms*. Nova Jersey, EUA: Pearson, Prentice Hall, 2007.

TEAM KINECTIC. *Dashboard*. 2018. <https://teamkinetic.co.uk/>. Acessado em 16/06/2018.

TECH, H. ponto. *React: o framework onipresente*. out. 2017. 1 MP3 (48min). (Hipsters). Podcast. <https://hipsters.tech/react-o-framework-onipresente-hipsters-66/>. Acessado em 23/05/2018.

TEIXEIRA, F. *O que é o SUS (System Usability Scale) e como usá-lo em seu site*. 2015. <https://brasil.uxdesign.cc/o-que-%C3%A9-o-sus-system-usability-scale-e-como-us%C3%A1-lo-em-seu-site-6d63224481c8>. Acessado em 30/10/2018.

TENóRIO, F. *Gestão de ongs: principais funções gerenciais*. Rio de Janeiro, Brasil: Ed.Fgv, 2001.

VALENTE, J. A. Diferentes usos do computador na educação. In: *Em Aberto*. Brasília, Brasil: [s.n.], 1993. v. 12, n. 53, p. 1–14. <http://emaberto.inep.gov.br/index.php/emaberto/article/view/1876>. Acessado em 21/07/2017.

VIDAL, F. A. B. et al. Gestão participativa e voluntariado: Sinais de uma racionalidade substantiva na administração de organizações do terceiro setor. In: *Anima - Revista da Faculdade Integrada do Ceará*. Fortaleza, Brasil: [s.n.], 2007. <http://publica-estaciofic.com.br/edicoes/ANIMA/ANIMA-2007-N11.pdf#page=28>. Acessado em 12/06/2018.

VOLGISTICS. *Dashboard*. 2018. <https://www.volgistics.com/>. Acessado em 16/06/2018.

VOLUNTEER HUB. *Home Page*. 2018. <https://www.volunteerhub.com/>. Acessado em 16/06/2018.

WIKIWIKIWEB. *Immutable Objects And Garbage Collection*. 2011. <http://wiki.c2.com/?ImmutableObjectsAndGarbageCollection>. Acessado em 26/10/2018.

WORDPRESS. *Home Page*. 2018. <https://wordpress.org/>. Acessado em 21/06/2018.

YAZBEK, M. C. Voluntariado e profissionalidade na intervenção social. *Revista Intervenção Social*, Brasil, 2002. <http://www.periodicoseletronicos.ufma.br/index.php/rppublica/article/view/3718/1749>. Acessado em 14/06/2018.

APÊNDICE A – Artigo

Processo de voluntariado dentro da iniciativa Computação na Escola

Luiz Ricardo Flores Maestri[1], Jean Carlo Rossa Hauck[1]

[1]Departamente de Informatica e Estatística (INE) – Universidade Federal de Santa Catarina (UFSC)

luiz.maestri@grad.ufsc.br, jean.hauck@ufsc.br

Abstract. *Currently, the market in technology is in the world in great growth, however, in Brazil, there has not been perceived enough interest in the training of young people to work in the area. Initiatives such as Computação na Escola (CnE), which use volunture efforts to support computer education for children and young people, are born seeking to address this deficiency in basic education, yet managing the entire process of volunteering becomes complex and cumbersome, especially due to communication difficulties due to the physical exchange of documents between the entities involved. In this context, this work proposes the development and implementation of a web system in order to assist the management of the volunteer process.*

Resumo. *Atualmente o mercado na área de tecnologia apresenta-se no mundo em grande crescimento. Todavia no Brasil não se tem percebido interesse suficiente na capacitação de jovens para atuarem na área. Iniciativas como a Computação na Escola (CnE), que utilizam esforços voluntários para apoiar o ensino de computação para crianças e jovens, nascem procurando atender esta deficiência na educação básica. A gestão do processo completo de voluntariado se torna algo complexo e trabalhoso, principalmente pelas dificuldades de commicação devido à troca de documentos por meio físico entre as entidades envolvidas.*

1. Introdução

A demanda de profissionais nas áreas de Ciências, Tecnologia, Engenharia e Matemática tem a previsão, nos Estados Unidos, de triplicar no período entre 2011 e 2021 [LANGDON 2011].

Mesmo com essa demanda crescente na área da TI, em 2015 um estudo divulgado [ABDUL-ALIM 2015] aponta que alunos do ensino médio não seriam capacitados em sala de aula com a finalidade de ocupar essas vagas.

Iniciativas público-privadas como a escola NAVE[1] e voluntárias como code.org[2] e Computação na escola(CnE)[3] surgiram com a intenção de levar o ensino da computação para crianças e jovens a um maior público.

A Iniciativa Computação na Escola tem utilizado o envolvimento de voluntários no ensino de programação para crianças [CNE 2017]. No entanto, gerenciar o processo de voluntariado é complexo, especialmente sem o apoio de uma ferramenta. Tipicamente o processo de voluntariado na Iniciativa Computação na Escola envolve:

- A CnE, o voluntário de alguma empresa parceira, a empresa e uma escola alvo.
- Envios de diversos documentos e Termos de Responsabilidade por e-mail/fax.
- Avaliações de perfil ao início do processo e avaliações de desempenho ao fim do mesmo.
- Agendamento das datas de treinamento e monitoria.

Para Tenório (2001) ONG (voluntárias) tem um "estilo próprio de gestão" construído ao longo de sua existência e demonstrando uma resistência aos modelos mais estruturados, apresentando dificuldades na execução de tarefas administrativas.

2. Contexto

A experiência relatada consiste na busca de ferramentas que possam auxiliar na gestão do processo de voluntariado da CnE. Com base em alguns requisitos elencados pela Know How Non Profit (Org.) (2018), requisitos funcionais, e por meio de entrevista com um dos coordenadores da CnE, requisitos não funcionais, são elas:

- RF1 - O sistema deve permitir o cadastro dos voluntários.
- RF2 - O sistema deve permitir o cadastro dos beneficiados.
- RF3 - O sistema deve permitir o cadastro de eventos de voluntariado.
- RF4 - O sistema deve permitir a alocação de voluntários.
- RF5 - O sistema deve permitir o gerenciamento de treinamentos.
- RF6 - O sistema deve permitir a avaliação dos voluntários.
- RF7 - O sistema deve permitir a avaliação dos treinamentos.
- RNF1 - O sistema deve ser Integrado ao site da CnE.
- RNF2 - Gratuito e Open-Source.
- RNF3 - Aplicação web.

3. Procedimentos metodológicos (coleta/análise dos dados)

A análise do estado da arte é realizada seguindo as principais etapas do método de mapeamento sistemático de literatura definido por Petersen K. (2008). Esse método é realizado definindo o mapeamento da literatura disponível, executando a busca e fazendo a extração dos resultados, bem como sua análise.

Definição do estudo: O estudo é definido em termos do objetivo e perguntas de pesquisa e o design de pesquisa. A partir do objetivo e das perguntas de análise são sistematicamente derivadas as medidas para a coleta de dados utilizando o método GQM [Basili, Caldeira e Rombach 1994].

Análise e interpretação do estudo: Nesta etapa são analisados os dados obtidos pergunta/requisitos de pesquisa, usando métodos quantitativos e qualitativos. Ao final, os resultados são interpretados e discutidos.

4. Principais Conclusões

Existem diversas aplicações de para o auxílio do controle de voluntários. porém nenhuma das ferramentas supracitadas pode atender a todos os requisitos elencados para o projeto, resultando na necessidade da implementação de um sistema específico para a Computação na Escola.

Referências

ABDUL-ALIM, J. "Students not prepared for careers in computer science." 2015. Disponível: http://diverseeducation.com/article/77408/. Acesso Set. 2017.

Basili, V. R., Caldeira G., Rombach, H. D. "Goal Question Metric Paradigm. Encyclopedia of Software Engineering", John Wiley & Sons, 1994.

CNE, C. N. E. F. "Iniciando o projeto Jovens Tutores de Programação." 2017. Disponível: http://www.computacaonaescola.ufsc.br/?p=2217. Acesso: Set. 2017.

KNOW HOW NON PROFIT (ORG.). "HOW TO CHOOSE A VOLUNTEER MANAGEMENT SYSTEM". 2018. Disponível: https://knowhownonprofit.org/how-to/how-to-choose-a-volunteer-management-system. Acesso: Abr. 2018.

LANGDON, D. et al. "Stem: Good jobs now and for the future." 2011. Disponível: http://www.esa.doc.gov/sites/default/files/stemfinalyjuly14_1.pdf. Acesso: Set. 2017.

TENÓRIO, F. Gestão de ongs: principais funções gerenciais. Rio de Janeiro, Brasil: Ed.Fgv, 2001.

1 http://www.oifuturo.org.br/educacao/nave/

2 http://code.org

3 http://www.computacaonaescola.ufsc.br

Figura 60 – Artigo

APÊNDICE B – Detalhamento dos casos de uso

Tabela 17 – Detalhamento de caso de uso: UC 2 - CRUD Evento.

Nome do Caso de Uso	UC 2 - CRUD Evento.
Pré-Condições	Existir escola já cadastrada e voluntários associados a escola
Atores Envolvidos	Responsável na CnE
Resumo	Aborda os passos necessários para o Responsável na CnE cadastrar um novo evento.
Cenário principal:	
1. O Responsável na CnE seleciona no calendário a data e e horários que o evento ocorrerá.	
2. O sistema abre um modal com formulário, em forma de *wizard* com para o cadastro de evento, apresentando os campos relativos ao endereço como desabilitados.	
3. O Responsável na CnE preenche os dados e seleciona a escola participante.	
4. O Sistema preenche os dados de endereço com os dados da escola e habilita para edição.	
5. O Responsável na CnE seleciona os voluntários participantes do evento dentre os associados a escola.	
6. O Responsável na CnE seleciona documentos digitais para disponibilizar aos voluntários participantes.	

Tabela 18 – Detalhamento de caso de uso: UC 3 - CRUD Projeto.

Nome do Caso de Uso	UC 3 - CRUD Projeto.
Pré-Condições	Sem precondições
Atores Envolvidos	Responsável na CnE
Resumo	Aborda os passos necessários para o Responsável na CnE Cadastrar um projeto
Cenário principal:	
1. O Responsável na CnE clica em "Administração"no canto superior direito.	
2. O sistema Abre a tela de administração.	
3. O Responsável na CnE clica em "Projetos".	
4. O Sistema apresenta a tela de projetos com uma listagem dos projetos existentes a esquerda e um formulário a direita.	
5. O Responsável empresa preenche os dados e clica em "Salvar".	
6. O Sistema apresenta o novo projeto na listagem a direita e e configura os campos do formulário para em branco.	

Tabela 19 – Detalhamento de caso de uso: UC 4 -Avaliar recomendação de voluntário.

Nome do Caso de Uso	UC 4 -Avaliar recomendação de voluntário.
Pré-Condições	Voluntário com o *status* "esperando recomendação da empresa"
Atores Envolvidos	Responsável pela Empresa
Resumo	Aborda os procedimentos realizados pelo Responsável na Empresa para recomendar um voluntário.
Cenário principal:	
1. O Responsável pela Empresa clica em "Recomendar"	
2. O sistema apresenta um modal com as respostas do voluntário ao questionário de cadastro.	
3. O Responsável pela Empresa clica em "Avançar".	
4. O sistema apresenta um questionário sobre a recomendação do voluntário.	
5. O Responsável pela Empresa preenche o questionário e clica em "Recomendar".	
6. O Sistema altera o *status* do voluntário para "aguardando envio do termo de responsabilidade".	

Tabela 20 – Detalhamento de caso de uso: UC 5 - Emitir relatório de desempenho.

Nome do Caso de Uso	UC 5 - Emitir relatório de desempenho.
Pré-Condições	Existir evento avaliado.
Atores Envolvidos	Responsável na CnE.
Resumo	Aborda como o Responsável na CnE emite os relatórios sobre os voluntários.
Cenário principal:	
1. O Responsável na CnE clica em "Relatórios"no canto superior direito.	
2. O sistema apresenta a tela de relatórios com dois botões "Relatório por escola"e "Relatório por empresa".	
3. O Responsável na CnE clica em "Relatório por escola".	
4. O sistema apresenta o relatório de eventos separados por escola.	
Cenário alternativo: No passo 4 caso o o Responsável na CnE tenha clicado em "Relatório por empresa"no passo 3.	
4.1. O sistema apresenta o relatório de eventos separados por empresa.	

Tabela 21 – Detalhamento de caso de uso: UC 7 - CRUD escola.

Nome do Caso de Uso	UC 7 - CRUD escola.
Pré-Condições	Sem pré-condições
Atores Envolvidos	Responsável na CnE
Resumo	Aborda os passos realizados pelo Responsável na CnE para cadastrar uma escola beneficiada
Cenário principal:	
1. O Responsável na CnE clica no botão "+"acima da listagem de escolas beneficiadas.	
2. O sistema abre o formulário em forma de *wizard* de cadastro de escola beneficiada em um modal.	
3. O Responsável na CnE preenche os dados da escola e clica em "Salvar"	
4. O Sistema salva os dados e atualiza a listagem de escolas beneficiadas.	

Tabela 22 – Detalhamento de caso de uso: UC 8 - CRUD empresa.

Nome do Caso de Uso	UC 8 - CRUD empresa.
Pré-Condições	Sem pré-condições
Atores Envolvidos	Responsável na CnE
Resumo	Aborda os passos realizados pelo Responsável na CnE para cadastrar uma Empresa Parceira
Cenário principal:	
1. O Responsável na CnE clica no botão "+"acima da listagem de escolas beneficiadas.	
2. O sistema abre o formulário em forma de *wizard* de cadastro de escola beneficiada em um modal.	
3. O Responsável na CnE preenche os dados da escola e clica em "Salvar"	
4. O Sistema salva os dados e atualiza a listagem de escolas beneficiadas.	

Tabela 23 – Detalhamento de caso de uso: UC 11 - CRUD Voluntário.

Nome do Caso de Uso	UC 11 - CRUD Voluntário.
Pré-Condições	Ter Empresa parceira cadastrada.
Atores Envolvidos	Voluntário.
Resumo	Aborda os passo realizados pelo Voluntário
Cenário principal:	
1. O Voluntário clica em "Quero ser voluntário".	
2. O sistema apresenta o formulário de cadastro de voluntário.	
3. O Voluntário preenche seus dados e clica em cadastrar.	
4. O sistema salva os dados envia um e-mail e realiza o *login* do voluntário.	
5. O sistema apresenta a tela para envio de documentos.	
6. O Voluntário realiza o *upload* dos documentos.	
7. Após o último *upload* o sistema avisa o Voluntário por um *pop-up* e altera o *status* para "aguardando recomendação da empresa".	

Tabela 24 – Detalhamento de caso de uso: UC 14 - CRUD Tipo de evento.

Nome do Caso de Uso	UC 14 - CRUD Tipo de evento.
Pré-Condições	Sem precondições
Atores Envolvidos	Responsável na CnE
Resumo	Aborda os passos necessários para o Responsável na CnE Cadastrar um tipo de evento
Cenário principal:	
1. O Responsável na CnE clica em "Administração"no canto superior direito.	
2. O sistema Abre a tela de administração.	
3. O Responsável na CnE clica em "Tipo de Evento".	
4. O Sistema apresenta a tela de projetos com uma listagem dos tipo de evento existentes a esquerda e um formulário a direita.	
5. O Responsável empresa preenche os dados e clica em "Salvar".	
6. O Sistema apresenta o novo projeto na listagem a direita e e configura os campos do formulário para em branco.	

APÊNDICE C – Diagramas de Classe

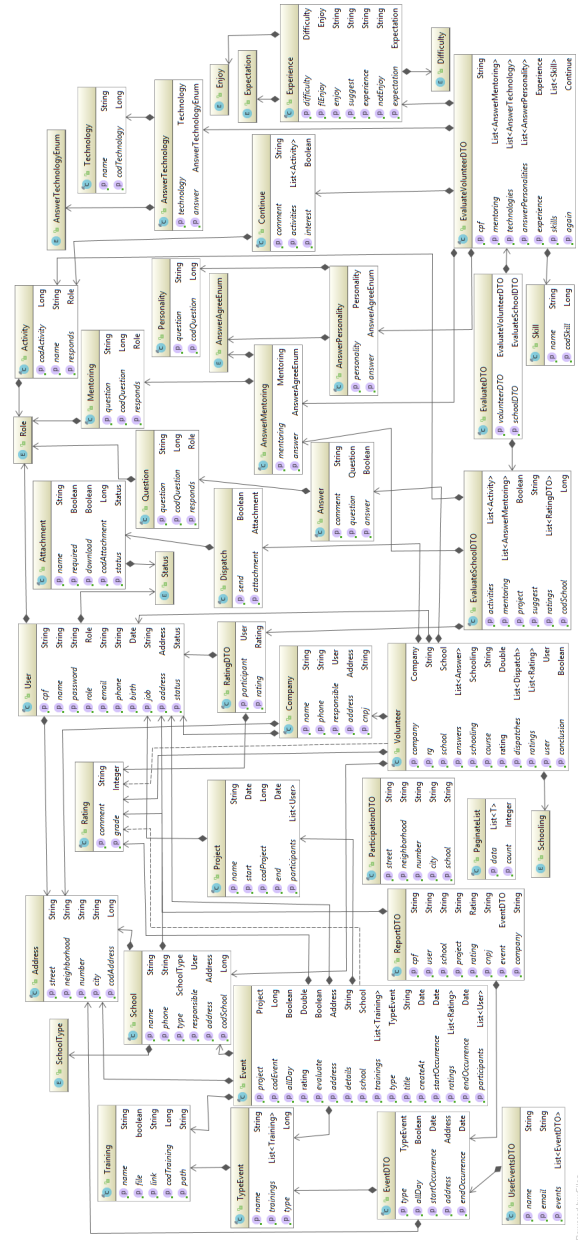


Figura 61 – Diagrama de classes - classes de modelo e DTOs.

APÊNDICE D – Código

```
1
2
3  src/main/java/br/ufsc/cultivar/dto/ParticipationDTO.java
4
5  package br.ufsc.cultivar.dto;
6
7  import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
8  import lombok.AccessLevel;
9  import lombok.Builder;
10 import lombok.Value;
11 import lombok.experimental.FieldDefaults;
12
13 @Value
14 @Builder(builderClassName = "Builder")
15 @JsonDeserialize(builder = ParticipationDTO.Builder.class)
16 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
17 public class ParticipationDTO {
18     String city;
19     String neighborhood;
20     String street;
21     String number;
22     String school;
23 }
24
25
26 src/main/java/br/ufsc/cultivar/dto/EvaluateDTO.java
27
28 package br.ufsc.cultivar.dto;
29
30 import br.ufsc.cultivar.model.Answer;
31 import br.ufsc.cultivar.model.evaluate.AnswerTechnology;
32 import br.ufsc.cultivar.model.evaluate.AnswerPersonality;
33 import br.ufsc.cultivar.model.evaluate.Experience;
34 import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
35 import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
36 import lombok.AccessLevel;
37 import lombok.Builder;
38 import lombok.Value;
39 import lombok.experimental.FieldDefaults;
40
41 import java.util.List;
```

```
42
43  @Value
44  @Builder(builderClassName = "Builder")
45  @JsonDeserialize(builder = EvaluateDTO.Builder.class)
46  @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
47  public class EvaluateDTO {
48
49      String cpf;
50      Long project;
51      List<AnswerTechnology> technologies;
52      List<AnswerPersonality> answerPersonalities;
53      Experience experience;
54
55      @JsonPOJOBuilder(withPrefix = "")
56      public static class Builder{
57          public Builder(){}
58      }
59  }
60
61
62  src/main/java/br/ufsc/cultivar/dto/EventDTO.java
63
64  package br.ufsc.cultivar.dto;
65
66  import br.ufsc.cultivar.model.Address;
67  import br.ufsc.cultivar.model.TypeEvent;
68  import lombok.AccessLevel;
69  import lombok.Builder;
70  import lombok.Value;
71  import lombok.experimental.FieldDefaults;
72
73  import java.util.Date;
74
75  @Value
76  @Builder
77  @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
78  public class EventDTO {
79      TypeEvent type;
80      Address address;
81      Date startOccurrence;
82      Date endOccurrence;
83      Boolean allDay;
84  }
85
86
```

src/main/java/br/ufsc/cultivar/dto/PaginateList.java

```java
package br.ufsc.cultivar.dto;

import br.ufsc.cultivar.model.Answer;
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;

import java.util.List;

@Value
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = PaginateList.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class PaginateList<T> {
    Integer count;
    List<T> data;
}
```

src/main/java/br/ufsc/cultivar/dto/UserEventsDTO.java

```java
package br.ufsc.cultivar.dto;

import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;

import java.util.List;

@Value
@Builder
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class UserEventsDTO {
    String email;
    String name;
    List<EventDTO> events;
}
```

src/main/java/br/ufsc/cultivar/repository/evaluate/PersonalityRepository.java

```
132
133 package br.ufsc.cultivar.repository.evaluate;
134
135 import br.ufsc.cultivar.model.evaluate.Personality;
136 import br.ufsc.cultivar.utils.DatabaseUtils;
137 import lombok.AccessLevel;
138 import lombok.AllArgsConstructor;
139 import lombok.experimental.FieldDefaults;
140 import org.springframework.beans.factory.annotation.Autowired;
141 import
        org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
142 import
        org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
143 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
144 import org.springframework.stereotype.Repository;
145
146 import java.sql.ResultSet;
147 import java.sql.SQLException;
148 import java.util.List;
149
150 @Repository
151 @AllArgsConstructor(onConstructor = @__(@Autowired))
152 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
153 public class PersonalityRepository {
154
155     NamedParameterJdbcTemplate jdbcTemplate;
156
157     public void create(Personality question) {
158         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
159                 .withTableName("personality")
160                 .usingGeneratedKeyColumns("cod_question")
161                 .execute(
162                         new MapSqlParameterSource("dsc_question",
                                question.getQuestion())
163                 );
164     }
165
166     public List<Personality> get() {
167         return jdbcTemplate.query(
168                 "select * from personality",
169                 (rs, i) -> this.build(rs)
170         );
171     }
172
173     public Personality get(Long codQuestion) {
```

```
174         return jdbcTemplate.query(
175                 "select * from personality where
                        cod_question=:cod_question",
176                 new MapSqlParameterSource("cod_question",
                        codQuestion),
177                 this::build
178         );
179     }
180
181     private Personality build(ResultSet rs) throws SQLException {
182         if(!DatabaseUtils.isNotEmpty(rs)){
183             return null;
184         }
185         return Personality.builder()
186                 .codQuestion(rs.getLong("cod_question"))
187                 .question(rs.getString("dsc_question"))
188                 .build();
189     }
190
191     public void delete(Long codQuestion) {
192         jdbcTemplate.update(
193                 "delete from personality where
                        cod_question=:cod_question",
194                 new MapSqlParameterSource("cod_question",
                        codQuestion)
195         );
196     }
197 }
198
199
200 src/main/java/br/ufsc/cultivar/repository/evaluate/MentoringRepository.java
201
202 package br.ufsc.cultivar.repository.evaluate;
203
204 import br.ufsc.cultivar.model.evaluate.Mentoring;
205 import br.ufsc.cultivar.utils.DatabaseUtils;
206 import lombok.AccessLevel;
207 import lombok.AllArgsConstructor;
208 import lombok.experimental.FieldDefaults;
209 import org.springframework.beans.factory.annotation.Autowired;
210 import
        org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
211 import
        org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
212 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
```

```
213 import org.springframework.stereotype.Repository;
214
215 import java.sql.ResultSet;
216 import java.sql.SQLException;
217 import java.util.List;
218
219 @Repository
220 @AllArgsConstructor(onConstructor = @__(@Autowired))
221 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
222 public class MentoringRepository {
223     NamedParameterJdbcTemplate jdbcTemplate;
224
225     public void create(Mentoring mentoring){
226         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
227             .withTableName("mentoring")
228             .usingGeneratedKeyColumns("cod_question")
229             .execute(
230                 new MapSqlParameterSource("dsc_question",
                        mentoring.getQuestion())
231             );
232     }
233
234     public List<Mentoring> get(){
235         return jdbcTemplate.query(
236                 "select * from mentoring",
237                 (rs, i) -> this.build(rs)
238         );
239     }
240
241     public Mentoring get(Long codQuestion){
242         return jdbcTemplate.query(
243                 "select * from mentoring where
                        cod_question=:cod_question",
244                 new MapSqlParameterSource("cod_question",
                        codQuestion),
245                 this::build
246         );
247     }
248
249     public void delete(Long codQuestion){
250         jdbcTemplate.update(
251                 "delete from mentoring where
                        cod_question=:cod_question",
252                 new MapSqlParameterSource("cod_question",
                        codQuestion)
```

```
253         );
254     }
255 
256     private Mentoring build(ResultSet rs) throws SQLException {
257         if(!DatabaseUtils.isNotEmpty(rs)){
258             return null;
259         }
260         return Mentoring.builder()
261                 .codQuestion(rs.getLong("cod_question"))
262                 .question(rs.getString("dsc_question"))
263                 .build();
264     }
265 }
266 
267 
268 src/main/java/br/ufsc/cultivar/repository/evaluate/SkillRepository.java
269 
270 package br.ufsc.cultivar.repository.evaluate;
271 
272 import br.ufsc.cultivar.model.evaluate.Skill;
273 import br.ufsc.cultivar.utils.DatabaseUtils;
274 import lombok.AccessLevel;
275 import lombok.AllArgsConstructor;
276 import lombok.experimental.FieldDefaults;
277 import org.springframework.beans.factory.annotation.Autowired;
278 import
        org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
279 import
        org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
280 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
281 import org.springframework.stereotype.Repository;
282 
283 import java.sql.ResultSet;
284 import java.sql.SQLException;
285 import java.util.List;
286 
287 @Repository
288 @AllArgsConstructor(onConstructor = @__(@Autowired))
289 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
290 public class SkillRepository {
291     NamedParameterJdbcTemplate jdbcTemplate;
292 
293     public void create(Skill skill){
294         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
295             .withTableName("skill")
```

```
296             .usingGeneratedKeyColumns("cod_skill")
297             .execute(
298                 new MapSqlParameterSource("nm_skill",
                        skill.getName())
299             );
300     }
301
302     public Skill get(Long codSkill){
303         return jdbcTemplate.query(
304                 "select * from skill where cod_skill=:cod_skill",
305                 new MapSqlParameterSource("cod_skill", codSkill),
306                 this::build
307         );
308     }
309
310     public List<Skill> get(){
311         return jdbcTemplate.query(
312                 "select * from skill",
313                 (rs, i) -> this.build(rs)
314         );
315     }
316
317     public void delete(Long codSkill){
318         jdbcTemplate.update(
319                 "delete from skill where cod_skill=:cod_skill",
320                 new MapSqlParameterSource("cod_skill", codSkill)
321         );
322     }
323
324     private Skill build(ResultSet rs) throws SQLException {
325         if (!DatabaseUtils.isNotEmpty(rs)){
326             return null;
327         }
328         return Skill.builder()
329                 .codSkill(rs.getLong("cod_skill"))
330                 .name(rs.getString("nm_skill"))
331                 .build();
332     }
333 }
334
335
336 src/main/java/br/ufsc/cultivar/repository/evaluate/EvaluateRepository.java
337
338 package br.ufsc.cultivar.repository.evaluate;
339
```

```
340 import br.ufsc.cultivar.model.evaluate.*;
341 import lombok.AccessLevel;
342 import lombok.AllArgsConstructor;
343 import lombok.experimental.FieldDefaults;
344 import lombok.val;
345 import org.springframework.beans.factory.annotation.Autowired;
346 import
        org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
347 import
        org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
348 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
349 import org.springframework.stereotype.Repository;
350
351 import java.util.List;
352
353 @Repository
354 @AllArgsConstructor(onConstructor = @__(@Autowired))
355 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
356 public class EvaluateRepository {
357
358     NamedParameterJdbcTemplate jdbcTemplate;
359
360     public void saveTechnology(AnswerTechnology
            answerTechnology, Long codProject, String cpf) {
361         val technology = answerTechnology.getTechnology();
362         val answer = answerTechnology.getAnswer();
363         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
364             .withTableName("answer_technology")
365             .execute(
366                 new MapSqlParameterSource()
367                     .addValue("cod_technology",
                            technology.getCodTechnology())
368                     .addValue("cod_project", codProject)
369                     .addValue("cod_cpf", cpf)
370                     .addValue("dsc_answer", answer.name())
371             );
372     }
373
374     public void saveAnswerVolunteer(AnswerPersonality
            answerPersonality, Long codProject, String cpf) {
375         val question = answerPersonality.getQuestion();
376         val answer = answerPersonality.getAnswer();
377         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
378             .withTableName("answer_technology")
379             .execute(
```

```
380                 new MapSqlParameterSource()
381                     .addValue("cod_question",
                            question.getCodQuestion())
382                     .addValue("cod_project", codProject)
383                     .addValue("cod_cpf", cpf)
384                     .addValue("dsc_answer", answer.name())
385             );
386     }
387
388     public void saveExperience(Experience experience, Long
            codProject, String cpf) {
389         val difficulty = experience.getDifficulty();
390         val enjoy = experience.getFlEnjoy();
391         val expectation = experience.getExpectation();
392         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
393             .withTableName("experience")
394             .execute(
395                 new MapSqlParameterSource()
396                     .addValue("cod_project", codProject)
397                     .addValue("cod_cpf", cpf)
398                     .addValue("dsc_experience",
                            experience.getExperience())
399                     .addValue("dsc_difficulty", difficulty.name())
400                     .addValue("fl_enjoy", enjoy.name())
401                     .addValue("dsc_enjoy", experience.getEnjoy())
402                     .addValue("dsc_not_enjoy",
                            experience.getNotEnjoy())
403                     .addValue("dsc_suggest",
                            experience.getSuggest())
404                     .addValue("dsc_expectation",
                            expectation.name())
405             );
406     }
407
408     public List<String> getCpfs(Long codProject) {
409         return jdbcTemplate.queryForList(
410                 "select cod_cpf from experience where
                        cod_project=:cod_project",
411                 new MapSqlParameterSource("cod_project",
                        codProject),
412                 String.class
413         );
414     }
415
```

```
416     public List<AnswerPersonality> getAnswersVolunteer(Long
            codProject, String cpf) {
417         return jdbcTemplate.query(
418                 "select av.dsc_answer, vq.* from
                        answer_personality av natural join
                        personality vq where cod_project=:cod_project
                        and cod_cpf=:cod_cpf",
419                 new MapSqlParameterSource("cod_project",
                        codProject).addValue("cod_cpf", cpf),
420                 (rs, i) ->
421                     AnswerPersonality.builder()
422                         .question(
423                             Personality.builder()
424                                 .codQuestion(rs.getLong("cod_question"))
425                                 .question(rs.getString("dsc_question"))
426                                 .build()
427                         ).answer(
428                             AnswerPersonalityEnum
429                                 .valueOf(rs.getString("dsc_answer"))
430                         ).build()
431         );
432     }
433
434     public List<AnswerTechnology> getAnswersTechnologies(Long
            codProject, String cpf) {
435         return jdbcTemplate.query(
436             "select at.dsc_answer ,t.* from answer_technology at
                    natural join technology t where
                    cod_project=:cod_project and cod_cpf=:cod_cpf",
437                 new MapSqlParameterSource("cod_project",
                        codProject).addValue("cod_cpf", cpf),
438                 (rs, i) ->
439                     AnswerTechnology.builder()
440                             .technology(
441                                     Technology.builder()
442                                             .name("")
443                                             .build()
444                             )
445                             .build()
446         );
447     }
448 }
449
450
451 src/main/java/br/ufsc/cultivar/repository/evaluate/TechnologyRepository.java
```

```
452
453 package br.ufsc.cultivar.repository.evaluate;
454
455 import br.ufsc.cultivar.model.evaluate.Technology;
456 import br.ufsc.cultivar.utils.DatabaseUtils;
457 import lombok.AccessLevel;
458 import lombok.AllArgsConstructor;
459 import lombok.experimental.FieldDefaults;
460 import org.springframework.beans.factory.annotation.Autowired;
461 import
        org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
462 import
        org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
463 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
464 import org.springframework.stereotype.Repository;
465
466 import java.sql.ResultSet;
467 import java.sql.SQLException;
468 import java.util.List;
469
470 @Repository
471 @AllArgsConstructor(onConstructor = @__(@Autowired))
472 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
473 public class TechnologyRepository {
474     NamedParameterJdbcTemplate jdbcTemplate;
475
476     public void create(Technology technology) {
477         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
478                 .withTableName("technology")
479                 .usingGeneratedKeyColumns("cod_technology")
480                 .execute(
481                         new MapSqlParameterSource("nm_technology",
                                technology.getName())
482                 );
483     }
484
485     public List<Technology> get() {
486         return jdbcTemplate.query("select * from technology",
                (rs, i) -> this.build(rs));
487     }
488
489     private Technology build(ResultSet rs) throws SQLException {
490         if(!DatabaseUtils.isNotEmpty(rs)){
491             return null;
492         }
```

```
493         return Technology.builder()
494                 .codTechnology(rs.getLong("cod_technology"))
495                 .name(rs.getString("nm_technology"))
496                 .build();
497     }
498
499     public void delete(Long codTechnology) {
500         jdbcTemplate.update(
501                 "delete from technology where
                        cod_technology=:cod_technology",
502                 new MapSqlParameterSource("cod_technology",
                        codTechnology)
503         );
504     }
505
506     public Technology get(Long codTechnology) {
507         return jdbcTemplate.query(
508                 "select * from technology where
                        cod_technology=:cod_technology",
509                 new MapSqlParameterSource("cod_technology",
                        codTechnology),
510                 this::build
511         );
512     }
513 }
514
515
516 src/main/java/br/ufsc/cultivar/repository/TrainingRepository.java
517
518 package br.ufsc.cultivar.repository;
519
520 import br.ufsc.cultivar.model.Training;
521 import br.ufsc.cultivar.utils.DatabaseUtils;
522 import lombok.AccessLevel;
523 import lombok.AllArgsConstructor;
524 import lombok.experimental.FieldDefaults;
525 import org.springframework.beans.factory.annotation.Autowired;
526 import
        org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
527 import
        org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
528 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
529 import org.springframework.stereotype.Repository;
530
531 import java.sql.ResultSet;
```

```
532 import java.sql.SQLException;
533 import java.util.List;
534
535 @Repository
536 @AllArgsConstructor(onConstructor = @__(@Autowired))
537 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
538 public class TrainingRepository {
539
540     NamedParameterJdbcTemplate jdbcTemplate;
541
542
543     public Long create(final Training training, final Long
          codEvent) {
544         return new
              SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
545                 .withTableName("training")
546                 .usingGeneratedKeyColumns("cod_training")
547                 .executeAndReturnKey(
548                         new MapSqlParameterSource()
549                                 .addValue("nm_training",
                                      training.getName())
550                                 .addValue("dsc_path",
                                      training.getPath())
551                                 .addValue("dsc_link",
                                      training.getLink())
552                 ).longValue();
553     }
554
555     public List<Training> getByTypeEvent(final Long tpEvent) {
556         return jdbcTemplate.query(
557                 "select t.* from training t natural join
                      type_event_training where tp_event=:tp_event",
558                 new MapSqlParameterSource("tp_event", tpEvent),
559                 (rs, i) -> this.build(rs)
560         );
561     }
562
563     public List<Training> getByEvent(final Long codEvent) {
564         return jdbcTemplate.query(
565                 "select t.* from training t natural join
                      event_training where cod_event=:cod_event",
566                 new MapSqlParameterSource("cod_event", codEvent),
567                 (rs, i) -> this.build(rs)
568         );
569     }
```

```
570
571     public Training get(final Long codTraining) {
572         return jdbcTemplate.query(
573                 "select * from training where
                        cod_training=:cod_training",
574                 new MapSqlParameterSource("cod_training",
                        codTraining),
575                 this::build
576         );
577     }
578
579     public void deleteByEvent(final Long tpEvent) {
580         jdbcTemplate.update(
581                 "select * from training where tp_event=:tp_event",
582                 new MapSqlParameterSource("tp_event", tpEvent)
583         );
584     }
585
586
587
588     private Training build(ResultSet rs) throws SQLException {
589         if(!DatabaseUtils.isNotEmpty(rs)){
590             return null;
591         }
592         return Training.builder()
593                 .codTraining(rs.getLong("cod_training"))
594                 .name(rs.getString("nm_training"))
595                 .path(rs.getString("dsc_path"))
596                 .link(rs.getString("dsc_link"))
597                 .build();
598     }
599 }
600
601
602 src/main/java/br/ufsc/cultivar/repository/QuestionRepository.java
603
604 package br.ufsc.cultivar.repository;
605
606 import br.ufsc.cultivar.model.Question;
607 import br.ufsc.cultivar.model.Role;
608 import br.ufsc.cultivar.utils.DatabaseUtils;
609 import lombok.AccessLevel;
610 import lombok.AllArgsConstructor;
611 import lombok.experimental.FieldDefaults;
612 import org.springframework.beans.factory.annotation.Autowired;
```

```
613 import
        org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
614 import
        org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
615 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
616 import org.springframework.stereotype.Repository;
617
618 import java.sql.ResultSet;
619 import java.sql.SQLException;
620 import java.util.List;
621
622 @Repository
623 @AllArgsConstructor(onConstructor = @__(@Autowired))
624 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
625 public class QuestionRepository {
626
627     NamedParameterJdbcTemplate jdbcTemplate;
628
629     public void create(final Question question) {
630         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
631                 .withTableName("question")
632                 .usingGeneratedKeyColumns("cod_question")
633                 .execute(
634                         new MapSqlParameterSource("dsc_question",
                                question.getQuestion())
635                             .addValue("dsc_responds",
                                    question.getResponds().name())
636                 );
637     }
638
639     public List<Question> get() {
640         return jdbcTemplate.query(
641                 "select * from question",
642                 (rs, i) -> this.build(rs)
643         );
644     }
645
646     public List<Question> get(final Role responds) {
647         return jdbcTemplate.query(
648                 "select * from question where
                        dsc_responds=:dsc_responds",
649                 new MapSqlParameterSource("dsc_responds",
                        responds.name()),
650                 (rs, i) -> this.build(rs)
651         );
```

```
652     }
653
654     public Question get(final Long codQuestion) {
655         return jdbcTemplate.query(
656                 "select * from question where
                        cod_question=:cod_question",
657                 new MapSqlParameterSource("cod_question",
                        codQuestion),
658                 this::build
659         );
660     }
661
662     public void delete(Long codQuestion) {
663         jdbcTemplate.update(
664                 "delete from question where
                        cod_question=:cod_question",
665                 new MapSqlParameterSource("cod_question",
                        codQuestion)
666         );
667     }
668
669     public void update(Question question) {
670         jdbcTemplate.update(
671                 "update question set dsc_question=:dsc_question
                        where cod_question=:cod_question",
672                 new MapSqlParameterSource("dsc_question",
                        question.getQuestion())
673                     .addValue("cod_question",
                            question.getQuestion())
674         );
675     }
676
677     private Question build(ResultSet rs) throws SQLException {
678         if(!DatabaseUtils.isNotEmpty(rs)){
679             return null;
680         }
681         return Question.builder()
682                 .codQuestion(rs.getLong("cod_question"))
683                 .question(rs.getString("dsc_question"))
684                 .responds(
685                         Role.valueOf(rs.getString("dsc_responds"))
686                 )
687                 .build();
688     }
689 }
```

src/main/java/br/ufsc/cultivar/repository/CompanyRepository.java

```java
package br.ufsc.cultivar.repository;

import br.ufsc.cultivar.model.Address;
import br.ufsc.cultivar.model.Company;
import br.ufsc.cultivar.model.User;
import br.ufsc.cultivar.utils.DatabaseUtils;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import lombok.val;
import org.springframework.beans.factory.annotation.Autowired;
import
    org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import
    org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import
    org.springframework.jdbc.core.namedparam.SqlParameterSource;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;
import java.util.Map;
import java.util.Objects;
import java.util.Optional;

@Repository
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class CompanyRepository {

    NamedParameterJdbcTemplate jdbcTemplate;

    public void create(final Company company) {
        new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
                .withTableName("company")
                .execute(getParams(company));
    }

```

```
731     public List<Company> get(final String filter, final Long
            page) {
732         val sql = new StringBuilder("select * from company");
733         val params = new MapSqlParameterSource();
734         Optional.ofNullable(filter)
735             .ifPresent(
736                 s -> {
737                     sql.append(" where nm_company like
                            :nm_company");
738                     params.addValue("nm_company", filter + "%");
739                 }
740         );
741         Optional.ofNullable(page)
742             .ifPresent(
743                 aLong -> {
744                     sql.append(" limit 5 offset :offset");
745                     params.addValue("offset", page*5);
746                 }
747         );
748         return jdbcTemplate.query(
749                 sql.toString(),
750                 params,
751                 (rs, i) -> this.build(rs)
752         );
753     }
754
755     public Integer count(final String filter) {
756         val sql = new StringBuilder("select count(cod_cnpj) from
                company");
757         val params = new MapSqlParameterSource();
758         Optional.ofNullable(filter)
759             .ifPresent(s -> {
760                 sql.append(" where nm_company like :nm_company");
761                 params.addValue("nm_company", filter + "%");
762             }
763         );
764         return jdbcTemplate.queryForObject(sql.toString(),
                params, Integer.class);
765     }
766
767     public Company get(final String cnpj) {
768         return jdbcTemplate.query(
769                 "select * from company where cod_cnpj=:cod_cnpj",
770                 new MapSqlParameterSource("cod_cnpj", cnpj),
771                 this::build
```

```
772         );
773     }
774
775     public void delete(final String cnpj) {
776         jdbcTemplate.update(
777                 "delete from company where cod_cnpj=:cod_cnpj",
778                 new MapSqlParameterSource("cod_cnpj", cnpj)
779         );
780     }
781
782     public void update(final Company company) {
783         jdbcTemplate.update(
784                 "update company set nm_company=:nm_company,
                        dsc_phone=:dsc_phone,
                        cod_address=:cod_address," +
785                        "cod_cpf=:cod_cpf where
                            cod_cnpj=:cod_cnpj",
786                 getParams(company)
787         );
788     }
789
790     private SqlParameterSource getParams(final Company company) {
791         return new MapSqlParameterSource()
792                 .addValue("cod_cnpj", company.getCnpj())
793                 .addValue("nm_company", company.getName())
794                 .addValue("dsc_phone", company.getPhone())
795                 .addValue("cod_address",
                        company.getAddress().getCodAddress())
796                 .addValue("cod_cpf",
                        company.getResponsible().getCpf());
797     }
798
799     private Company build(final ResultSet rs) throws
            SQLException {
800         if(!DatabaseUtils.isNotEmpty(rs)){
801             return null;
802         }
803         return Company.builder()
804                 .cnpj(rs.getString("cod_cnpj"))
805                 .name(rs.getString("nm_company"))
806                 .phone(rs.getString("dsc_phone"))
807                 .address(
808                         Address.builder()
809                                 .codAddress(rs.getLong("cod_address"))
810                                 .build()
```

```
811                 )
812                 .responsible(
813                         User.builder()
814                                 .cpf(rs.getString("cod_cpf"))
815                                 .build()
816                 )
817                 .build();
818     }
819 }
820
821
822 src/main/java/br/ufsc/cultivar/repository/DispatchRepository.java
823
824 package br.ufsc.cultivar.repository;
825
826 import br.ufsc.cultivar.model.Attachment;
827 import br.ufsc.cultivar.model.Dispatch;
828 import br.ufsc.cultivar.utils.DatabaseUtils;
829 import lombok.AccessLevel;
830 import lombok.AllArgsConstructor;
831 import lombok.experimental.FieldDefaults;
832 import org.springframework.beans.factory.annotation.Autowired;
833 import
        org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
834 import
        org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
835 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
836 import org.springframework.stereotype.Repository;
837
838 import java.sql.ResultSet;
839 import java.sql.SQLException;
840 import java.util.List;
841
842 @Repository
843 @AllArgsConstructor(onConstructor = @__(@Autowired))
844 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
845 public class DispatchRepository {
846
847     NamedParameterJdbcTemplate jdbcTemplate;
848
849
850     public void save(Dispatch dispatch, String cpf) {
851         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
852                 .withTableName("dispatch")
853                 .execute(
```

```
854                         new MapSqlParameterSource()
855                             .addValue("cod_cpf", cpf)
856                             .addValue("cod_attachment",
                                    dispatch.getAttachment().getCodAttachment())
857                             .addValue("fl_dispatch",
                                    dispatch.getSend())
858                 );
859     }
860
861     public Dispatch get(String cpf, Long codAttachment) {
862         return jdbcTemplate.query(
863             "select * from dispatch where cod_cpf=:cod_cpf and
                    cod_attachment=:cod_attachment",
864             new MapSqlParameterSource()
865                 .addValue("cod_cpf", cpf)
866                 .addValue("cod_attachment", codAttachment),
867             this::build
868         );
869     }
870
871     public List<Dispatch> get(String cpf) {
872         return jdbcTemplate.query(
873                 "select * from dispatch where cod_cpf=:cod_cpf",
874                 new MapSqlParameterSource()
875                         .addValue("cod_cpf", cpf),
876                 (rs, i) -> this.build(rs)
877         );
878     }
879
880     private Dispatch build(ResultSet rs) throws SQLException {
881         if(!DatabaseUtils.isNotEmpty(rs)){
882             return null;
883         }
884         return Dispatch.builder()
885                 .attachment(
886                         Attachment.builder()
887                                 .codAttachment(rs.getLong("cod_attachment"))
888                                 .build()
889                 ).send(rs.getBoolean("fl_dispatch"))
890                 .build();
891     }
892 }
893
894
895 src/main/java/br/ufsc/cultivar/repository/AttachmentRepository.java
```

```
896
897 package br.ufsc.cultivar.repository;
898
899 import br.ufsc.cultivar.model.Attachment;
900 import br.ufsc.cultivar.model.Status;
901 import lombok.AccessLevel;
902 import lombok.AllArgsConstructor;
903 import lombok.experimental.FieldDefaults;
904 import lombok.val;
905 import org.springframework.beans.factory.annotation.Autowired;
906 import
        org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
907 import
        org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
908 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
909 import org.springframework.stereotype.Repository;
910
911 import java.sql.ResultSet;
912 import java.sql.SQLException;
913 import java.util.List;
914 import java.util.Optional;
915
916 @Repository
917 @AllArgsConstructor(onConstructor = @__(@Autowired))
918 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
919 public class AttachmentRepository {
920
921     NamedParameterJdbcTemplate jdbcTemplate;
922
923     public long create(final Attachment attachment) {
924         return new
              SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
925                 .withTableName("attachment")
926                 .usingGeneratedKeyColumns("cod_attachment")
927                 .executeAndReturnKey(getParams(attachment))
928                 .longValue();
929     }
930
931     public List<Attachment> get(final String filter, final Long
          page) {
932         val sql = new StringBuilder("select * from attachment ");
933         val params = new MapSqlParameterSource();
934         Optional.ofNullable(filter)
935             .ifPresent(
936                 s -> {
```

```
937                     sql.append("where nm_attachment like
                            :nm_attachment ");
938                     params.addValue("nm_attachment", filter + "%");
939                 }
940             );
941         sql.append("limit 20 offset :offset");
942         params.addValue("offset", page*20);
943         return jdbcTemplate.query(sql.toString(), params, (rs,
                i) -> this.build(rs));
944     }
945
946     public Integer count(final String filter){
947         val sql = new StringBuilder("select
                count(cod_attachment) from attachment ");
948         val params = new MapSqlParameterSource();
949         Optional.ofNullable(filter)
950             .ifPresent(
951                 s -> {
952                     sql.append("where nm_attachment like
                            :nm_attachment ");
953                     params.addValue("nm_attachment", filter + "%");
954                 }
955             );
956         return jdbcTemplate.queryForObject(sql.toString(),
                params, Integer.class);
957     }
958
959     public Attachment get(final Long codAttachment) {
960         return jdbcTemplate.query(
961                 "select * from attachment where
                        cod_attachment=:cod_attachment",
962                 new MapSqlParameterSource("cod_attachment",
                        codAttachment),
963                 this::build
964         );
965     }
966     public List<Attachment> get(final Status status) {
967         return jdbcTemplate.query(
968                 "select * from attachment where
                        sta_user=:sta_user",
969                 new MapSqlParameterSource("sta_user",
                        status.name()),
970                 (rs, i) -> this.build(rs)
971         );
972     }
```

```
973
974     public void delete(Long codAttachment) {
975         jdbcTemplate.update(
976                 "delete from attachment where
                        cod_attachment=:cod_attachment",
977                 new MapSqlParameterSource("cod_attachment",
                        codAttachment)
978         );
979     }
980
981     public void update(final Attachment attachment) {
982         jdbcTemplate.update(
983                 "update attachment set
                        nm_attachment=:nm_attachment,
                        fl_required=:fl_required " +
984                         "where cod_attachment=:cod_attachment",
985                 getParams(attachment).addValue("cod_attachment",
                        attachment.getCodAttachment())
986         );
987     }
988
989     private Attachment build(ResultSet rs) throws SQLException {
990         if(rs.isBeforeFirst()){
991             rs.first();
992         }
993         return Attachment.builder()
994                 .codAttachment(rs.getLong("cod_attachment"))
995                 .name(rs.getString("nm_attachment"))
996                 .status(
997                         Status.valueOf(rs.getString("sta_user"))
998                 )
999                 .required(rs.getBoolean("fl_required"))
1000                .download(rs.getBoolean("fl_download"))
1001                .build();
1002    }
1003
1004    private MapSqlParameterSource getParams(final Attachment
            attachment) {
1005        return new MapSqlParameterSource()
1006                .addValue("nm_attachment", attachment.getName())
1007                .addValue("sta_user",
                        attachment.getStatus().name())
1008                .addValue("fl_required", attachment.getRequired())
1009                .addValue("fl_download",
                        attachment.getDownload());
```

```
1010        }
1011    }
1012
1013
1014    src/main/java/br/ufsc/cultivar/repository/RatingRepository.java
1015
1016    package br.ufsc.cultivar.repository;
1017
1018    import br.ufsc.cultivar.model.Rating;
1019    import lombok.AccessLevel;
1020    import lombok.AllArgsConstructor;
1021    import lombok.experimental.FieldDefaults;
1022    import org.springframework.beans.factory.annotation.Autowired;
1023    import
            org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
1024    import
            org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
1025    import
            org.springframework.jdbc.core.namedparam.SqlParameterSource;
1026    import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
1027    import org.springframework.stereotype.Repository;
1028
1029    import java.util.List;
1030
1031    @Repository
1032    @AllArgsConstructor(onConstructor = @__(@Autowired))
1033    @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
1034    public class RatingRepository {
1035
1036        NamedParameterJdbcTemplate jdbcTemplate;
1037
1038        public void create(Rating rating, String cpf) {
1039            new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
1040                    .withTableName("rating")
1041                    .usingColumns("vl_rating", "dsc_rating",
                            "cod_cpf")
1042                    .execute(getParams(rating, cpf));
1043        }
1044
1045        public void create(Rating rating, Long codEvent) {
1046            new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
1047                    .withTableName("rating")
1048                    .usingColumns("vl_rating", "dsc_rating",
                            "cod_event")
1049                    .execute(getParams(rating, codEvent));
```

```
1050     }
1051
1052     private SqlParameterSource getParams(Rating rating, Object
             cod) {
1053         return new MapSqlParameterSource()
1054                 .addValue("vl_rating", rating.getGrade())
1055                 .addValue("dsc_rating", rating.getComment())
1056                 .addValue("cod_cpf", cod)
1057                 .addValue("cod_event", cod);
1058     }
1059
1060     public List<Rating> get(String cpf) {
1061         return jdbcTemplate.query(
1062                 "select * from rating where cod_cpf=:cod_cpf",
1063                 new MapSqlParameterSource("cod_cpf", cpf),
1064                 (rs, i) -> Rating.builder()
1065                         .comment(rs.getString("dsc_rating"))
1066                         .grade(rs.getInt("vl_rating"))
1067                         .build()
1068         );
1069     }
1070
1071     public List<Rating> get(Long codEvent) {
1072         return jdbcTemplate.query(
1073                 "select * from rating where cod_event=:cod_event",
1074                 new MapSqlParameterSource("cod_event", codEvent),
1075                 (rs, i) -> Rating.builder()
1076                         .comment(rs.getString("dsc_rating"))
1077                         .grade(rs.getInt("vl_rating"))
1078                         .build()
1079         );
1080     }
1081 }
1082
1083
1084 src/main/java/br/ufsc/cultivar/repository/SchoolRepository.java
1085
1086 package br.ufsc.cultivar.repository;
1087
1088 import br.ufsc.cultivar.model.Address;
1089 import br.ufsc.cultivar.model.School;
1090 import br.ufsc.cultivar.model.SchoolType;
1091 import br.ufsc.cultivar.model.User;
1092 import br.ufsc.cultivar.utils.DatabaseUtils;
1093 import lombok.AccessLevel;
```

```
1094 import lombok.AllArgsConstructor;
1095 import lombok.experimental.FieldDefaults;
1096 import lombok.val;
1097 import org.springframework.beans.factory.annotation.Autowired;
1098 import
         org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
1099 import
         org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
1100 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
1101 import org.springframework.stereotype.Repository;
1102
1103 import java.sql.ResultSet;
1104 import java.sql.SQLException;
1105 import java.util.List;
1106 import java.util.Map;
1107 import java.util.Objects;
1108 import java.util.Optional;
1109
1110 @Repository
1111 @AllArgsConstructor(onConstructor = @__(@Autowired))
1112 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
1113 public class SchoolRepository {
1114
1115     NamedParameterJdbcTemplate jdbcTemplate;
1116
1117     public Long create(final School school) {
1118         return new
                 SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
1119                 .withTableName("school")
1120                 .usingGeneratedKeyColumns("cod_school")
1121                 .executeAndReturnKey(getParams(school))
1122                 .longValue();
1123     }
1124
1125     public List<School> get(final String filter, final Long
             page) {
1126         val sql = new StringBuilder("select * from school");
1127         val params = new MapSqlParameterSource();
1128         Optional.ofNullable(filter)
1129             .ifPresent(
1130                 s -> {
1131                     sql.append(" where nm_school like :nm_school");
1132                     params.addValue("nm_school", filter + "%");
1133                 }
1134             );
```

```
        Optional.ofNullable(page)
            .ifPresent(
                aLong -> {
                    sql.append(" limit 5 offset :offset");
                    params.addValue("offset", page * 5);
                }
            );
        return jdbcTemplate.query(
                sql.toString(),
                params,
                (rs, i) -> this.build(rs)
        );
    }

    public Integer count(final String filter) {
        val sql = new StringBuilder("select count(cod_school)
            from school");
        val params = new MapSqlParameterSource();
        Optional.ofNullable(filter)
                .ifPresent(s -> {
                           sql.append(" where nm_school like
                               :nm_school");
                           params.addValue("nm_school", filter +
                               "%");
                       }
                );
        return jdbcTemplate.queryForObject(sql.toString(),
            params, Integer.class);
    }

    public School get(final Long codSchool) {
        return jdbcTemplate.query(
                "select * from school where
                    cod_school=:cod_school",
                new MapSqlParameterSource("cod_school",
                    codSchool),
                this::build
        );
    }

    public void delete(final Long codSchool) {
        jdbcTemplate.update(
                "delete from school where cod_school=:cod_school",
                new MapSqlParameterSource("cod_school", codSchool)
        );
```

```
1174        }
1175
1176        public void update(final School school) {
1177            jdbcTemplate.update(
1178                    "update school set nm_school=:nm_school,
                            dsc_phone=:dsc_phone,
                            cod_address=:cod_address," +
1179                           "cod_cpf=:cod_cpf where
                                cod_school=:cod_school",
1180                    getParams(school).addValue("cod_school",
                            school.getCodSchool())
1181            );
1182        }
1183
1184        private MapSqlParameterSource getParams(final School school)
                {
1185            return new MapSqlParameterSource()
1186                    .addValue("nm_school", school.getName())
1187                    .addValue("dsc_phone", school.getPhone())
1188                    .addValue("cod_address",
                            school.getAddress().getCodAddress())
1189                    .addValue("cod_cpf",
                            school.getResponsible().getCpf())
1190                    .addValue("tp_school", school.getType().name());
1191        }
1192
1193        private School build(final ResultSet rs) throws SQLException
                {
1194            if(!DatabaseUtils.isNotEmpty(rs)){
1195                return null;
1196            }
1197            return School.builder()
1198                    .codSchool(rs.getLong("cod_school"))
1199                    .name(rs.getString("nm_school"))
1200                    .phone(rs.getString("dsc_phone"))
1201                    .type(
1202                            SchoolType.valueOf(rs.getString("tp_school"))
1203                    ).address(
1204                            Address.builder()
1205                                    .codAddress(rs.getLong("cod_address"))
1206                                    .build()
1207                    )
1208                    .responsible(
1209                            User.builder()
1210                                    .cpf(rs.getString("cod_cpf"))
```

```
                                .build()
                )
                .build();
    }
}


src/main/java/br/ufsc/cultivar/repository/VolunteerRepository.java

package br.ufsc.cultivar.repository;

import br.ufsc.cultivar.model.*;
import br.ufsc.cultivar.utils.DatabaseUtils;
import javafx.scene.chart.PieChartBuilder;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import lombok.val;
import org.springframework.beans.factory.annotation.Autowired;
import
    org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import
    org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.*;

@Repository
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class VolunteerRepository {

    NamedParameterJdbcTemplate jdbcTemplate;

    public void create(Volunteer volunteer) {
        new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
                .withTableName("volunteer")
                .execute(getParams(volunteer));
    }

    public List<Volunteer> get(final List<String> filterCompany,
        final List<Long> filterSchool,
```

```
1253                                final String filter, final Integer
                                        page) {
1254            val sql = new StringBuilder("select * from volunteer
                    where 1=1");
1255            val params = new MapSqlParameterSource();
1256
1257            if (!Optional.ofNullable(filterCompany)
1258                    .orElseGet(ArrayList::new)
1259                    .isEmpty()
1260                    ){
1261                sql.append(" and cod_cnpj in(:cod_cnpj)");
1262                params.addValue("cod_cnpj", filterCompany);
1263            }
1264            if (!Optional.ofNullable(filterSchool)
1265                    .orElseGet(ArrayList::new)
1266                    .isEmpty()
1267                    ){
1268                sql.append(" and cod_school in(:cod_school)");
1269                params.addValue("cod_school", filterSchool);
1270            }
1271            Optional.ofNullable(filter)
1272                .ifPresent(
1273                    s -> {
1274                        sql.append(" and nm_volunteer like
                                :nm_volunteer");
1275                        params.addValue("nm_volunteer", filter + "%");
1276                    }
1277                );
1278            Optional.ofNullable(page)
1279                .ifPresent(
1280                    aLong -> {
1281                        sql.append(" limit 5 offset :offset");
1282                        params.addValue("offset", page * 5);
1283                    }
1284                );
1285            return jdbcTemplate.query(
1286                    sql.toString(),
1287                    params,
1288                    (rs, i) -> this.build(rs)
1289            );
1290        }
1291
1292        public Volunteer get(String cpf) {
1293            return jdbcTemplate.query(
1294                    "select * from volunteer where cod_cpf=:cod_cpf",
```

```
1295                new MapSqlParameterSource("cod_cpf", cpf),
1296                this::build
1297        );
1298    }
1299
1300    public void delete(String cpf) {
1301        jdbcTemplate.update(
1302                "delete from volunteer where cod_cpf=:cod_cpf",
1303                new MapSqlParameterSource("cod_cpf", cpf)
1304        );
1305    }
1306
1307    public void update(Volunteer volunteer) {
1308        jdbcTemplate.update(
1309                "update volunteer set
                        dsc_schooling=:dsc_schooling,
                        fl_conclusion=:fl_conclusion, " +
1310                        "cod_rg=:cod_rg, cod_school=:cod_school
                            where cod_cpf=:cod_cpf",
1311                getParams(volunteer)
1312        );
1313    }
1314
1315    private MapSqlParameterSource getParams(Volunteer volunteer)
            {
1316        val user = volunteer.getUser();
1317        val school = volunteer.getSchool();
1318        return new MapSqlParameterSource()
1319                .addValue("cod_cpf", user.getCpf())
1320                .addValue("nm_volunteer", user.getName())
1321                .addValue("cod_cnpj",
                            volunteer.getCompany().getCnpj())
1322                .addValue("cod_school", Objects.nonNull(school) ?
                            school.getCodSchool() : null)
1323                .addValue("dsc_schooling",
                            volunteer.getSchooling().name())
1324                .addValue("fl_conclusion",
                            volunteer.getConclusion())
1325                .addValue("cod_rg", volunteer.getRg())
1326                .addValue("dsc_course", volunteer.getCourse());
1327    }
1328
1329    private Volunteer build(ResultSet rs) throws SQLException {
1330        if(!DatabaseUtils.isNotEmpty(rs)){
1331            return null;
```

```
1332            }
1333            return Volunteer.builder()
1334                    .user(
1335                            User.builder()
1336                                    .cpf(rs.getString("cod_cpf"))
1337                                    .build()
1338                    )
1339                    .company(
1340                            Company.builder()
1341                                    .cnpj(rs.getString("cod_cnpj"))
1342                                    .build()
1343                    ).school(
1344                            School.builder()
1345                                    .codSchool(rs.getLong("cod_school"))
1346                                    .build()
1347                    ).schooling(
1348                            Schooling.valueOf(rs.getString("dsc_schooling"))
1349                    ).course(rs.getString("dsc_course"))
1350                    .conclusion(rs.getBoolean("fl_conclusion"))
1351                    .rg(rs.getString("cod_rg"))
1352                    .build();
1353        }
1354
1355        public Integer count(final List<String> filterCompany, final
                List<Long> filterSchool, final String filter) {
1356            val sql = new StringBuilder("select count(cod_cpf) from
                    volunteer where 1=1");
1357            val params = new MapSqlParameterSource();
1358
1359            if (!Optional.ofNullable(filterCompany)
1360                    .orElseGet(ArrayList::new)
1361                    .isEmpty()
1362                    ){
1363                sql.append(" and cod_cnpj in(:cod_cnpj)");
1364                params.addValue("cod_cnpj", filterCompany);
1365            }
1366            if (!Optional.ofNullable(filterSchool)
1367                    .orElseGet(ArrayList::new)
1368                    .isEmpty()
1369                    ){
1370                sql.append(" and cod_school in(:cod_school)");
1371                params.addValue("cod_school", filterSchool);
1372            }
1373            Optional.ofNullable(filter)
1374                    .ifPresent(
```

```
                        s -> {
                            sql.append(" and nm_volunteer like
                                :nm_volunteer");
                            params.addValue("nm_volunteer", filter
                                + "%");
                        }
                );
        return jdbcTemplate.queryForObject(sql.toString(),
            params, Integer.class);
    }
}


src/main/java/br/ufsc/cultivar/repository/EventRepository.java

package br.ufsc.cultivar.repository;

import br.ufsc.cultivar.dto.EventDTO;
import br.ufsc.cultivar.model.*;
import br.ufsc.cultivar.utils.DatabaseUtils;
import br.ufsc.cultivar.utils.DateUtils;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import lombok.val;
import org.springframework.beans.factory.annotation.Autowired;
import
    org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import
    org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.stereotype.Repository;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.*;

@Repository
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class EventRepository {

    NamedParameterJdbcTemplate jdbcTemplate;

    public Long create(final Event event) {
```

```
1415            return new
                    SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
1416                    .withTableName("event")
1417                    .usingGeneratedKeyColumns("cod_event")
1418                    .usingColumns(
1419                            "dt_start_occurrence",
1420                            "dt_end_occurrence",
1421                            "fl_all_day",
1422                            "tp_event",
1423                            "cod_address",
1424                            "fl_evaluate",
1425                            "cod_school"
1426                    ).executeAndReturnKey(getParams(event))
1427                    .longValue();
1428        }
1429
1430        public List<Event> get(final List<String> filterVolunteer,
                final List<Long> filterSchool, final Long filterProject)
                {
1431            val sql = new StringBuilder("select e.*, te.nm_type from
                    event e join participation natural join type_event
                    te where 1=1");
1432            val params = new MapSqlParameterSource();
1433            if (!Optional.ofNullable(filterVolunteer)
1434                    .orElseGet(ArrayList::new)
1435                    .isEmpty()
1436                    ){
1437                sql.append(" and cod_cpf in(:cod_cpf)");
1438                params.addValue("cod_cpf", filterVolunteer);
1439            }
1440            if (!Optional.ofNullable(filterSchool)
1441                    .orElseGet(ArrayList::new)
1442                    .isEmpty()
1443                    ){
1444                sql.append(" and cod_school in(:cod_school)");
1445                params.addValue("cod_school", filterSchool);
1446            }
1447            if (Objects.nonNull(filterProject)){
1448                sql.append(" and cod_project in(:cod_project)");
1449                params.addValue("cod_project", filterProject);
1450            }
1451            return jdbcTemplate.query(
1452                    sql.toString(),
1453                    params,
1454                    (rs, i) -> this.build(rs)
```

```
1455        );
1456    }
1457
1458    public Event get(final Long codEvent) {
1459        return jdbcTemplate.query(
1460                "select e.*, te.nm_type from event e natural join
                        type_event te where cod_event=:cod_event",
1461                new MapSqlParameterSource("cod_event", codEvent),
1462                this::build
1463        );
1464    }
1465
1466    public void delete(final Long codEvent) {
1467        jdbcTemplate.update(
1468                "delete from event where cod_event=:cod_event",
1469                new MapSqlParameterSource("cod_event", codEvent)
1470        );
1471    }
1472
1473    public void update(final Event event) {
1474        jdbcTemplate.update(
1475                "update event set
                            dt_start_occurrence=:dt_start_occurrence,
                            dt_end_occurrence=:dt_end_occurrence," +
1476                           "tp_event=:tp_event,
                                fl_all_day=:fl_all_day,
                                cod_address=:cod_address,
                                fl_evaluate=:fl_evaluate," +
1477                           "cod_school=:cod_school where
                                cod_event=:cod_event",
1478                getParams(event)
1479        );
1480    }
1481
1482    private Event build(final ResultSet rs) throws SQLException {
1483        if(!DatabaseUtils.isNotEmpty(rs)){
1484            return null;
1485        }
1486        val codProject = rs.getLong("cod_project");
1487        val isNull = rs.wasNull();
1488        return Event.builder()
1489                .codEvent(rs.getLong("cod_event"))
1490                .startOccurrence(DateUtils.toDate(rs.getTimestamp("dt_start_occurr
1491                .endOccurrence(DateUtils.toDate(rs.getTimestamp("dt_end_occurrence
1492                .allDay(rs.getBoolean("fl_all_day"))
```

```
1493                .createAt(DateUtils.toDate(rs.getTimestamp("dt_create")))
1494                .type(
1495                        TypeEvent.builder()
1496                                .type(rs.getLong("tp_event"))
1497                                .name(rs.getString("nm_type"))
1498                                .build()
1499                )
1500                .address(
1501                        Address.builder()
1502                                .codAddress(rs.getLong("cod_address"))
1503                                .build()
1504                ).school(
1505                        School.builder()
1506                                .codSchool(rs.getLong("cod_school"))
1507                                .build()
1508                ).project(
1509                        isNull ?
1510                                Project.builder()
1511                                        .codProject(codProject)
1512                                        .build() :
1513                                null
1514                ).evaluate(rs.getBoolean("fl_evaluate"))
1515                .build();
1516    }
1517
1518    private MapSqlParameterSource getParams(final Event event) {
1519        return new MapSqlParameterSource()
1520                .addValue("dt_start_occurrence",
                            event.getStartOccurrence())
1521                .addValue("dt_end_occurrence",
                            event.getEndOccurrence())
1522                .addValue("fl_all_day", event.getAllDay())
1523                .addValue("tp_event", event.getType().getType())
1524                .addValue("cod_address",
                            event.getAddress().getCodAddress())
1525                .addValue("cod_school",
                            event.getSchool().getCodSchool())
1526                .addValue("cod_event", event.getCodEvent())
1527                .addValue("fl_evaluate", event.getEvaluate());
1528    }
1529
1530    public List<Event> eventsByVolunteer(final String cpf, final
            Long type) {
1531        val sb = new StringBuilder("select e.*, te.nm_type from
                event e natural join participation p natural join
```

```
            type_event te where cod_cpf=:cod_cpf");
1532        val params = new MapSqlParameterSource("cod_cpf", cpf);
1533        Optional.ofNullable(type)
1534            .ifPresent(
1535                aLong -> {
1536                    sb.append(" and te.tp_event=:tp_event");
1537                    params.addValue("tp_event", type);
1538                }
1539            );
1540        return jdbcTemplate.query(sb.toString(), params, (rs, i)
                -> this.build(rs));
1541    }
1542
1543    public List<Event> eventsBySchool(Long codSchool, Long type)
        {
1544        val sb = new StringBuilder("select * from event e
            natural join type_event te where
            cod_school=:cod_school");
1545        val params = new MapSqlParameterSource("cod_school",
            codSchool);
1546        Optional.ofNullable(type)
1547            .ifPresent(
1548                aLong -> {
1549                    sb.append(" and te.tp_event=:tp_event");
1550                    params.addValue("tp_event", type);
1551                }
1552            );
1553        return jdbcTemplate.query(sb.toString(), params, (rs, i)
                -> this.build(rs));
1554    }
1555
1556    public List<EventDTO> getEventsToAlert(String cpf) {
1557        val sql = "select dt_start_occurrence,
                dt_end_occurrence, " +
1558            "a.*, fl_all_day, te.nm_type from event " +
1559            "natural join type_event te natural join address a
                    natural join participation " +
1560            "where cod_cpf=:cod_cpf and " +
1561            "dt_start_occurrence between CURDATE() and (CURDATE()
                    + INTERVAL 7 DAY)";
1562        return jdbcTemplate.query(
1563            sql,
1564            new MapSqlParameterSource("cod_cpf", cpf),
1565            (rs, i) ->
1566                EventDTO.builder()
```

```
                    .type(
                        TypeEvent.builder()
                            .name(rs.getString("nm_type"))
                            .build()
                    ).startOccurrence(rs.getTimestamp("dt_start_occurrence")
                    .endOccurrence(rs.getTimestamp("dt_end_occurrence"))
                    .allDay(rs.getBoolean("fl_all_day"))
                    .address(
                        Address.builder()
                            .city(rs.getString("nm_city"))
                            .neighborhood(rs.getString("nm_neighborhood"))
                            .street(rs.getString("nm_street"))
                            .number(rs.getString("nu_street"))
                            .build()
                    ).build()
        );
    }

    public List<Event> getEventsToEvaluateByVolunteer(final
        String cpf) {
        return null;
    }
}


src/main/java/br/ufsc/cultivar/repository/AddressRepository.java

package br.ufsc.cultivar.repository;

import br.ufsc.cultivar.model.Address;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import lombok.val;
import org.springframework.beans.factory.annotation.Autowired;
import
    org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import
    org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.stereotype.Repository;

import java.util.Objects;

@Repository
```

```
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class AddressRepository {

    NamedParameterJdbcTemplate jdbcTemplate;

    public Long create(final Address address) {
        val insert = new
             SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
                .withTableName("address")
                .usingGeneratedKeyColumns("cod_address");
        if (Objects.isNull(address.getNumber())){
            insert.usingColumns("nm_city", "nm_neighborhood",
                 "nm_street");
        }
        return
             insert.executeAndReturnKey(getParams(address)).longValue();
    }

    public Address get(final Long codAddress) {
        return jdbcTemplate.query(
            "select * from address where
                 cod_address=:cod_address",
            new MapSqlParameterSource("cod_address", codAddress),
            rs -> {
                if(rs.isBeforeFirst()){
                    rs.first();
                }
                return Address.builder()
                        .codAddress(codAddress)
                        .city(rs.getString("nm_city"))
                        .neighborhood(rs.getString("nm_neighborhood"))
                        .street(rs.getString("nm_street"))
                        .number(rs.getString("nu_street"))
                        .build();
            }
        );
    }

    public void update(final Address address) {
        jdbcTemplate.update(
                "update address set nm_city=:nm_city,
                    nm_neighborhood=:nm_neighborhood,
                    nm_street=:nm_street," +
```

```
                        "nu_street=:nu_street where
                            cod_address=:cod_address",
                getParams(address)
                        .addValue("cod_address",
                            address.getCodAddress())
        );
    }

    private MapSqlParameterSource getParams(final Address
        address) {
        return new MapSqlParameterSource()
                .addValue("nm_city", address.getCity())
                .addValue("nm_neighborhood",
                    address.getNeighborhood())
                .addValue("nm_street", address.getStreet())
                .addValue("nu_street", address.getNumber());
    }
}


src/main/java/br/ufsc/cultivar/repository/ParticipationRepository.java

package br.ufsc.cultivar.repository;

import br.ufsc.cultivar.dto.ParticipationDTO;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import lombok.val;
import org.springframework.beans.factory.annotation.Autowired;
import
    org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import
    org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class ParticipationRepository {

    NamedParameterJdbcTemplate jdbcTemplate;
```

```
1686
1687        public void create(Long codEvent, String cpf) {
1688            new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
1689                    .withTableName("participation")
1690                    .execute(
1691                            new MapSqlParameterSource()
1692                                .addValue("cod_cpf", cpf)
1693                                .addValue("cod_event", codEvent)
1694                    );
1695        }
1696
1697        public List<ParticipationDTO> getVolunteerLocals(final
                String codCpf){
1698            val sql = "select a.*, nm_school from volunteer v" +
1699                    "  natural join participation p" +
1700                    "  join event e on p.cod_event = e.cod_event " +
1701                    "  natural join address a" +
1702                    "  join school s on e.cod_school = s.cod_school "
                            +
1703                    "where p.cod_cpf=:cod_cpf and e.dt_end_occurrence
                            < current_date";
1704            return jdbcTemplate.query(
1705                    sql,
1706                    new MapSqlParameterSource("cod_cpf", codCpf),
1707                    (rs, i) -> ParticipationDTO.builder()
1708                            .school(rs.getString("nm_school"))
1709                            .city(rs.getString("nm_city"))
1710                            .neighborhood(rs.getString("nm_neighborhood"))
1711                            .street(rs.getString("nm_street"))
1712                            .number(rs.getString("nu_street"))
1713                            .build()
1714            );
1715        }
1716    }
1717
1718
1719    src/main/java/br/ufsc/cultivar/repository/ProjectRepository.java
1720
1721    package br.ufsc.cultivar.repository;
1722
1723    import br.ufsc.cultivar.model.Project;
1724    import br.ufsc.cultivar.utils.DatabaseUtils;
1725    import lombok.AccessLevel;
1726    import lombok.AllArgsConstructor;
1727    import lombok.experimental.FieldDefaults;
```

```
1728 import lombok.val;
1729 import org.springframework.beans.factory.annotation.Autowired;
1730 import
         org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
1731 import
         org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
1732 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
1733 import org.springframework.stereotype.Repository;
1734
1735 import java.sql.ResultSet;
1736 import java.sql.SQLException;
1737 import java.util.List;
1738 import java.util.Optional;
1739
1740 @Repository
1741 @AllArgsConstructor(onConstructor = @__(@Autowired))
1742 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
1743 public class ProjectRepository {
1744
1745     NamedParameterJdbcTemplate jdbcTemplate;
1746
1747     public void create(Project project) {
1748         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
1749             .withTableName("project")
1750             .usingGeneratedKeyColumns("cod_project")
1751             .execute(
1752                 new MapSqlParameterSource()
1753                     .addValue("nm_project", project.getName())
1754                     .addValue("dt_start", project.getStart())
1755                     .addValue("dt_end", project.getEnd())
1756             );
1757     }
1758
1759     public List<Project> get(final String filter, final Long
             page) {
1760         val sql = new StringBuilder("select * from project ");
1761         val params = new MapSqlParameterSource();
1762         Optional.ofNullable(filter)
1763             .ifPresent(
1764                 s -> {
1765                     sql.append("where nm_project like :nm_project
                                 ");
1766                     params.addValue("nm_project", filter + "%");
1767                 }
1768             );
```

```
1769            sql.append("order by dt_start desc ");
1770            Optional.ofNullable(page)
1771                .ifPresent(
1772                    aLong -> {
1773                        sql.append("limit 20 offset :offset");
1774                        params.addValue("offset", page*20);
1775                    }
1776                );
1777            return jdbcTemplate.query(sql.toString(), params, (rs,
                    i) -> this.build(rs));
1778        }
1779
1780        public Integer count(final String filter) {
1781            val sql = new StringBuilder("select count(cod_project)
                    from project ");
1782            val params = new MapSqlParameterSource();
1783            Optional.ofNullable(filter)
1784                    .ifPresent(
1785                            s -> {
1786                                sql.append("where nm_project like
                                        :nm_project ");
1787                                params.addValue("nm_project", filter +
                                        "%");
1788                            }
1789                    );
1790            return jdbcTemplate.queryForObject(sql.toString(),
                    params, Integer.class);
1791        }
1792
1793        public Project get(Long codProject) {
1794            return jdbcTemplate.query(
1795                "select * from project where
                        cod_project=:cod_project",
1796                new MapSqlParameterSource("cod_project", codProject),
1797                this::build
1798            );
1799        }
1800
1801        public void delete(Long codProject) {
1802            jdbcTemplate.update(
1803                "delete from project where cod_project=:cod_project",
1804                new MapSqlParameterSource("cod_project", codProject)
1805            );
1806        }
1807
```

```
1808        private Project build(ResultSet rs) throws SQLException {
1809            if(!DatabaseUtils.isNotEmpty(rs)){
1810                return null;
1811            }
1812            return Project.builder()
1813                .codProject(rs.getLong("cod_project"))
1814                .name(rs.getString("nm_project"))
1815                .start(rs.getDate("dt_start"))
1816                .end(rs.getDate("dt_end"))
1817                .build();
1818        }
1819    }
1820
1821
1822    src/main/java/br/ufsc/cultivar/repository/AnswerRepository.java
1823
1824    package br.ufsc.cultivar.repository;
1825
1826    import br.ufsc.cultivar.model.Answer;
1827    import br.ufsc.cultivar.model.Question;
1828    import lombok.AccessLevel;
1829    import lombok.AllArgsConstructor;
1830    import lombok.experimental.FieldDefaults;
1831    import org.springframework.beans.factory.annotation.Autowired;
1832    import
            org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
1833    import
            org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
1834    import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
1835    import org.springframework.stereotype.Repository;
1836
1837    import java.util.List;
1838
1839    @Repository
1840    @AllArgsConstructor(onConstructor = @__(@Autowired))
1841    @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
1842    public class AnswerRepository {
1843
1844        NamedParameterJdbcTemplate jdbcTemplate;
1845
1846        public void create(final Answer answer, final String cpf) {
1847            new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
1848                    .withTableName("answer")
1849                    .execute(
1850                            new MapSqlParameterSource()
```

```
1851                            .addValue("cod_cpf", cpf)
1852                            .addValue("cod_question",
                                    answer.getQuestion().getCodQuestion())
1853                            .addValue("fl_answer",
                                    answer.getAnswer())
1854                            .addValue("dsc_answer",
                                    answer.getComment())
1855                );
1856        }
1857
1858        public List<Answer> get(final String cpf) {
1859            return jdbcTemplate.query(
1860                    "select * from answer where cod_cpf=:cod_cpf",
1861                    new MapSqlParameterSource("cod_cpf", cpf),
1862                    (rs, i) -> Answer.builder()
1863                            .answer(rs.getBoolean("fl_answer"))
1864                            .comment(rs.getString("dsc_answer"))
1865                            .question(
1866                                    Question.builder()
1867                                            .codQuestion(rs.getLong("cod_question"))
1868                                            .build()
1869                            )
1870                            .build()
1871            );
1872        }
1873
1874        public void delete(String cpf) {
1875            jdbcTemplate.update(
1876                    "delete from answer where cod_cpf=:cod_cpf",
1877                    new MapSqlParameterSource("cod_cpf", cpf)
1878            );
1879        }
1880    }
1881
1882
1883    src/main/java/br/ufsc/cultivar/repository/UserRepository.java
1884
1885    package br.ufsc.cultivar.repository;
1886
1887    import br.ufsc.cultivar.model.Address;
1888    import br.ufsc.cultivar.model.Role;
1889    import br.ufsc.cultivar.model.Status;
1890    import br.ufsc.cultivar.model.User;
1891    import lombok.AccessLevel;
1892    import lombok.AllArgsConstructor;
```

```
1893 import lombok.experimental.FieldDefaults;
1894 import lombok.val;
1895 import org.springframework.beans.factory.annotation.Autowired;
1896 import
         org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
1897 import
         org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
1898 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
1899 import org.springframework.stereotype.Repository;
1900
1901 import java.sql.ResultSet;
1902 import java.sql.SQLException;
1903 import java.util.List;
1904 import java.util.Map;
1905 import java.util.Optional;
1906
1907 @Repository
1908 @AllArgsConstructor(onConstructor = @__(@Autowired))
1909 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
1910 public class UserRepository {
1911     NamedParameterJdbcTemplate jdbcTemplate;
1912
1913     public void create(final User user) {
1914         new SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
1915             .withTableName("users")
1916             .usingColumns("cod_cpf", "nm_user", "dsc_email",
                      "dsc_password", "dsc_role",
1917                       "sta_user", "dt_birth", "cod_address",
                              "dsc_job", "dsc_phone")
1918             .execute(getParams(user));
1919     }
1920
1921     public List<User> get(final Map<String, Object> filter) {
1922         val sql = new StringBuilder("select * from users where
                  1=1");
1923         val params = new MapSqlParameterSource();
1924         Optional.ofNullable(filter)
1925             .ifPresent(
1926                 map -> map.forEach(
1927                     (key, value) -> {
1928                         sql.append(" and ")
1929                             .append(key)
1930                             .append("=:")
1931                             .append(key);
1932                         params.addValue(key, value);
```

```
                }
            )
        );
    return jdbcTemplate.query(
        sql.toString(),
        params,
        (rs, i) -> this.build(rs)
    );
}

public User get(final String cpf) {
    return jdbcTemplate.query(
        "select * from users where cod_cpf=:cod_cpf",
        new MapSqlParameterSource("cod_cpf", cpf),
        this::build
    );
}

public void delete(final String cpf) {
    jdbcTemplate.update(
        "delete from users where cod_cpf=:cod_cpf",
        new MapSqlParameterSource("cod_cpf", cpf)
    );
}

public void update(final User user) {
    val sql = "update users " +
        "set nm_user=:nm_user, dsc_email=:dsc_email,
            dsc_password=:dsc_password, dsc_role=:dsc_role," +
        "sta_user=:sta_user, dt_birth=:dt_birth,
            cod_address=:cod_address, dsc_job=:dsc_job," +
        "dsc_phone=:dsc_phone where cod_cpf=:cod_cpf;";
    jdbcTemplate.update(sql, getParams(user));
}

public List<User> getParticipants(final Long codEvent) {
    return jdbcTemplate.query(
            "select u.* from users u join participation p on
                u.cod_cpf=p.cod_cpf where
                p.cod_event=:cod_event",
            new MapSqlParameterSource("cod_event", codEvent),
            (rs, i) -> this.build(rs)
    );
}
```

```
1974      private MapSqlParameterSource getParams(final User user){
1975          return new MapSqlParameterSource()
1976              .addValue("cod_cpf", user.getCpf())
1977              .addValue("nm_user", user.getName())
1978              .addValue("dsc_email", user.getEmail())
1979              .addValue("dsc_password", user.getPassword())
1980              .addValue("dsc_role", user.getRole().name())
1981              .addValue("sta_user", user.getStatus().name())
1982              .addValue("dt_birth", user.getBirth())
1983              .addValue("cod_address",
                      user.getAddress().getCodAddress())
1984              .addValue("dsc_job", user.getJob())
1985              .addValue("dsc_phone", user.getPhone());
1986      }
1987
1988      private User build(final ResultSet rs) throws SQLException {
1989          if(rs.isBeforeFirst()){
1990              rs.first();
1991          }
1992          return User.builder()
1993                  .cpf(rs.getString("cod_cpf"))
1994                  .name(rs.getString("nm_user"))
1995                  .email(rs.getString("dsc_email"))
1996                  .password(rs.getString("dsc_password"))
1997                  .role(Role.valueOf(rs.getString("dsc_role")))
1998                  .status(Status.valueOf(rs.getString("sta_user")))
1999                  .birth(rs.getDate("dt_birth"))
2000                  .address(
2001                      Address.builder()
2002                          .codAddress(rs.getLong("cod_address"))
2003                          .build()
2004                  )
2005                  .job(rs.getString("dsc_job"))
2006                  .phone(rs.getString("dsc_phone"))
2007                  .build();
2008      }
2009  }
2010
2011
2012  src/main/java/br/ufsc/cultivar/repository/TypeEventRepository.java
2013
2014  package br.ufsc.cultivar.repository;
2015
2016  import br.ufsc.cultivar.model.Project;
2017  import br.ufsc.cultivar.model.TypeEvent;
```

```
2018 import br.ufsc.cultivar.utils.DatabaseUtils;
2019 import javafx.scene.chart.PieChartBuilder;
2020 import lombok.AccessLevel;
2021 import lombok.AllArgsConstructor;
2022 import lombok.experimental.FieldDefaults;
2023 import lombok.val;
2024 import org.springframework.beans.factory.annotation.Autowired;
2025 import
         org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
2026 import
         org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
2027 import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
2028 import org.springframework.stereotype.Repository;
2029
2030 import java.sql.ResultSet;
2031 import java.sql.SQLException;
2032 import java.util.List;
2033 import java.util.Optional;
2034
2035 @Repository
2036 @AllArgsConstructor(onConstructor = @__(@Autowired))
2037 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
2038 public class TypeEventRepository {
2039
2040     NamedParameterJdbcTemplate jdbcTemplate;
2041
2042     public Long create(String name) {
2043         return new
                 SimpleJdbcInsert(jdbcTemplate.getJdbcTemplate())
2044                 .withTableName("type_event")
2045                 .usingGeneratedKeyColumns("tp_event")
2046                 .executeAndReturnKey(
2047                         new MapSqlParameterSource("nm_type", name)
2048                 ).longValue();
2049     }
2050
2051     public List<TypeEvent> get(final String filter, final Long
             page) {
2052         val sql = new StringBuilder("select * from type_event ");
2053         val params = new MapSqlParameterSource();
2054         Optional.ofNullable(filter)
2055             .ifPresent(
2056                 s -> {
2057                     sql.append("where nm_type like :nm_type ");
2058                     params.addValue("nm_type", filter + "%");
```

```
2059                }
2060            );
2061        Optional.ofNullable(page)
2062            .ifPresent(
2063                aLong -> {
2064                    sql.append("limit 20 offset :offset");
2065                    params.addValue("offset", page*20);
2066                }
2067            );
2068        return jdbcTemplate.query(sql.toString(), params, (rs,
                i) -> this.build(rs));
2069    }
2070
2071    public TypeEvent get(Long tpEvent) {
2072        return jdbcTemplate.query(
2073                "select * from type_event where
                        tp_event=:tp_event",
2074                new MapSqlParameterSource("tp_event", tpEvent),
2075                this::build
2076        );
2077    }
2078
2079    private TypeEvent build(ResultSet rs) throws SQLException {
2080        if(!DatabaseUtils.isNotEmpty(rs)){
2081            return null;
2082        }
2083        return TypeEvent.builder()
2084                .type(rs.getLong("tp_event"))
2085                .name(rs.getString("nm_type"))
2086                .build();
2087    }
2088
2089    public void delete(Long tpEvent) {
2090        jdbcTemplate.update(
2091                "delete from type_event where tp_event=:tp_event",
2092                new MapSqlParameterSource("tp_event", tpEvent)
2093        );
2094    }
2095
2096    public Integer count(String filter) {
2097        val sql = new StringBuilder("select count(tp_event) from
                type_event ");
2098        val params = new MapSqlParameterSource();
2099        Optional.ofNullable(filter)
2100            .ifPresent(
```

```
                s -> {
                    sql.append("where nm_type like :nm_type ");
                    params.addValue("nm_type", filter + "%");
                }
            );
        return jdbcTemplate.queryForObject(sql.toString(),
            params, Integer.class);
    }
}
```

src/main/java/br/ufsc/cultivar/config/SecurityConfig.java

```
package br.ufsc.cultivar.config;

import org.springframework.context.annotation.Bean;
import
    org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
    org.springframework.security.config.annotation.web.configuration.EnableWebSe
import
    org.springframework.security.config.annotation.web.configuration.WebSecurity
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;
import
    org.springframework.web.cors.UrlBasedCorsConfigurationSource;

@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter
    {

    @Override
    protected void configure(HttpSecurity http) throws Exception
    {
        http.cors()
            .and()
            .csrf()
            .disable();
    }

    @Bean
    CorsConfigurationSource corsConfigurationSource(){
        CorsConfiguration corsConfiguration = new
            CorsConfiguration();
        corsConfiguration.addAllowedHeader("*");
```

```
2138            corsConfiguration.addAllowedMethod("*");
2139            corsConfiguration.addAllowedOrigin("*");
2140
2141            UrlBasedCorsConfigurationSource source = new
                    UrlBasedCorsConfigurationSource();
2142            source.registerCorsConfiguration("/**",
                    corsConfiguration);
2143            return source;
2144        }
2145    }
2146
2147
2148    src/main/java/br/ufsc/cultivar/config/EmailConfig.java
2149
2150    package br.ufsc.cultivar.config;
2151
2152    import lombok.Getter;
2153    import org.springframework.beans.factory.annotation.Value;
2154    import org.springframework.context.annotation.Configuration;
2155
2156    @Configuration
2157    @Getter
2158    public class EmailConfig {
2159        @Value("${spring.mail.username}")
2160        String from;
2161    }
2162
2163
2164    src/main/java/br/ufsc/cultivar/config/AsyncConfig.java
2165
2166    package br.ufsc.cultivar.config;
2167
2168    import org.springframework.context.annotation.Bean;
2169    import org.springframework.context.annotation.Configuration;
2170    import org.springframework.scheduling.annotation.EnableAsync;
2171    import
            org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor;
2172
2173    import java.util.concurrent.Executor;
2174
2175    @Configuration
2176    @EnableAsync
2177    public class AsyncConfig {
2178        @Bean
2179        public Executor asyncExecutor() {
```

```
        ThreadPoolTaskExecutor executor = new
            ThreadPoolTaskExecutor();
        executor.setCorePoolSize(2);
        executor.setMaxPoolSize(2);
        executor.setQueueCapacity(50);
        executor.setThreadNamePrefix("email-");
        executor.initialize();
        return executor;
    }
}
```

src/main/java/br/ufsc/cultivar/config/JdbcConfig.java

```
package br.ufsc.cultivar.config;

import org.springframework.context.annotation.Configuration;
import
    org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration
@EnableTransactionManagement
public class JdbcConfig {

}
```

src/main/java/br/ufsc/cultivar/config/SwaggerConfig.java

```
package br.ufsc.cultivar.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
    org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
import
    org.springframework.web.servlet.config.annotation.WebMvcConfigurationSupport
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import
    springfox.documentation.swagger2.annotations.EnableSwagger2;
```

```
2220
2221 import static springfox.documentation.builders.PathSelectors.ant;
2222
2223 @Configuration
2224 @EnableSwagger2
2225 public class SwaggerConfig extends WebMvcConfigurationSupport {
2226
2227     @Bean
2228
2229     public Docket productApi() {
2230         return new Docket(DocumentationType.SWAGGER_2)
2231                 .select()
2232                 .apis(RequestHandlerSelectors.basePackage("br.ufsc.cultivar.
2233                 .paths(ant("/api/**"))
2234                 .build()
2235                 .apiInfo(metaData());
2236
2237     }
2238
2239     private ApiInfo metaData() {
2240         return new ApiInfoBuilder()
2241                 .title("Sistema CulTiVar")
2242                 .description("Sistema de Controle de Tutores
                         Voluntrios")
2243                 .version("0.2.0")
2244                 .license("Apache License Version 2.0")
2245                 .licenseUrl("https://www.apache.org/licenses/LICENSE-2.0")
2246                 .contact(new Contact("Luiz Maestri", null,
                         "luizricardomaestri@gmail.com"))
2247                 .build();
2248     }
2249
2250     @Override
2251     protected void addResourceHandlers(ResourceHandlerRegistry
         registry) {
2252         registry.addResourceHandler("swagger-ui.html")
2253                 .addResourceLocations("classpath:/META-INF/resource/");
2254
2255         registry.addResourceHandler("/webjars/**")
2256                 .addResourceLocations("classpath:/META-INF/resource/webjars/
2257     }
2258 }
2259
2260
2261 src/main/java/br/ufsc/cultivar/security/AuthResponseDTO.java
```

```
2262
2263 package br.ufsc.cultivar.security;
2264
2265 import br.ufsc.cultivar.model.User;
2266 import com.fasterxml.jackson.annotation.JsonInclude;
2267 import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
2268 import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
2269 import lombok.AccessLevel;
2270 import lombok.Builder;
2271 import lombok.Value;
2272 import lombok.experimental.FieldDefaults;
2273 import lombok.experimental.Wither;
2274
2275 @Value
2276 @Wither
2277 @Builder(builderClassName = "Builder")
2278 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
2279 @JsonDeserialize(builder = AuthResponseDTO.Builder.class)
2280 @JsonInclude(JsonInclude.Include.NON_NULL)
2281 class AuthResponseDTO {
2282     User user;
2283     String token;
2284
2285     @JsonPOJOBuilder(withPrefix = "")
2286     static class Builder{}
2287 }
2288
2289
2290 src/main/java/br/ufsc/cultivar/security/AuthException.java
2291
2292 package br.ufsc.cultivar.security;
2293
2294 public class AuthException extends Exception {
2295     AuthException(String message) {
2296         super(message);
2297     }
2298 }
2299
2300
2301 src/main/java/br/ufsc/cultivar/security/AuthService.java
2302
2303 package br.ufsc.cultivar.security;
2304
2305 import br.ufsc.cultivar.exception.ServiceException;
2306 import br.ufsc.cultivar.model.User;
```

```
2307 import br.ufsc.cultivar.service.UserService;
2308 import lombok.AccessLevel;
2309 import lombok.AllArgsConstructor;
2310 import lombok.experimental.FieldDefaults;
2311 import org.springframework.beans.factory.annotation.Autowired;
2312 import org.springframework.stereotype.Service;
2313
2314 @Service
2315 @AllArgsConstructor(onConstructor = @__(@Autowired))
2316 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
2317 class AuthService {
2318     UserService userService;
2319
2320     AuthResponseDTO authenticate(AuthRequestDTO dto) throws
          AuthException, ServiceException {
2321         User user = userService.get(dto.getCpf());
2322         if (user.getPassword().equals(dto.getPassword())) {
2323             return AuthResponseDTO.builder()
2324                     .user(user)
2325                     .token("")
2326                     .build();
2327         }
2328         throw new AuthException("Usurio e/ou senha incorretos.");
2329     }
2330 }
2331
2332
2333 src/main/java/br/ufsc/cultivar/security/AuthRequestDTO.java
2334
2335 package br.ufsc.cultivar.security;
2336
2337 import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
2338 import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
2339 import lombok.AccessLevel;
2340 import lombok.Builder;
2341 import lombok.Value;
2342 import lombok.experimental.FieldDefaults;
2343 import org.hibernate.validator.constraints.br.CPF;
2344
2345 import javax.validation.constraints.NotBlank;
2346
2347 @Value
2348 @Builder
2349 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
```

```
@JsonDeserialize(builder =
    AuthRequestDTO.AuthRequestDTOBuilder.class)
class AuthRequestDTO {
    @CPF
    String cpf;
    @NotBlank
    String password;

    @JsonPOJOBuilder(withPrefix = "")
    public static class AuthRequestDTOBuilder{}
}
```

src/main/java/br/ufsc/cultivar/security/AuthResource.java

```
package br.ufsc.cultivar.security;

import br.ufsc.cultivar.exception.ServiceException;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import javax.validation.Valid;

@RestController
@RequestMapping(value = "/auth")
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class AuthResource {

    AuthService service;

    @PostMapping(consumes =
        MediaType.APPLICATION_JSON_UTF8_VALUE,
           produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
    public AuthResponseDTO authenticate(@RequestBody @Valid
        AuthRequestDTO dto) throws AuthException,
        ServiceException {
       return service.authenticate(dto);
```

```
2391     }
2392 }
2393
2394
2395
2396 src/main/java/br/ufsc/cultivar/scheduler/EventEmailScheduler.java
2397
2398 package br.ufsc.cultivar.scheduler;
2399
2400 import br.ufsc.cultivar.dto.UserEventsDTO;
2401 import br.ufsc.cultivar.email.EmailClient;
2402 import br.ufsc.cultivar.model.User;
2403 import br.ufsc.cultivar.service.EventService;
2404 import lombok.AccessLevel;
2405 import lombok.AllArgsConstructor;
2406 import lombok.experimental.FieldDefaults;
2407 import lombok.extern.slf4j.Slf4j;
2408 import lombok.val;
2409 import org.springframework.beans.factory.annotation.Autowired;
2410 import org.springframework.scheduling.annotation.Scheduled;
2411 import org.springframework.stereotype.Component;
2412 import org.thymeleaf.context.Context;
2413 import org.thymeleaf.spring5.SpringTemplateEngine;
2414
2415 import java.util.List;
2416
2417 @Slf4j
2418 @Component
2419 @AllArgsConstructor(onConstructor = @__(@Autowired))
2420 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
2421 public class EventEmailScheduler {
2422
2423     SpringTemplateEngine templateEngine;
2424     EventService service;
2425     EmailClient client;
2426
2427     @Scheduled(cron = "${scheduler.email.send.cron}")
2428     public void sendWarning(){
2429         service.getEventsToAlert()
2430             .forEach(
2431                 userEventsDTO ->
2432                     client.sendEmailSync(
2433                         userEventsDTO.getEmail(),
2434                         "Eventos",
2435                         buildEmail(userEventsDTO)
```

```
                    )
            );
    }

    private String buildEmail(UserEventsDTO userEventsDTO){
        val context = new Context();
        context.setVariable("message", "teste");
        context.setVariable("user", userEventsDTO);
        return templateEngine.process("eventEmail", context);
    }
}
```

src/main/java/br/ufsc/cultivar/utils/FileUtils.java

```
package br.ufsc.cultivar.utils;

import br.ufsc.cultivar.exception.ServiceException;
import br.ufsc.cultivar.exception.Type;
import br.ufsc.cultivar.exception.UploadException;
import br.ufsc.cultivar.model.Attachment;
import br.ufsc.cultivar.model.Dispatch;
import lombok.extern.slf4j.Slf4j;
import lombok.val;
import org.springframework.core.io.FileSystemResource;
import org.springframework.core.io.Resource;
import org.springframework.core.io.UrlResource;
import org.springframework.stereotype.Component;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.IOException;
import java.net.MalformedURLException;
import java.nio.file.Files;
import java.nio.file.Paths;

@Component
@Slf4j
public class FileUtils {
    public Dispatch save(final Attachment attachment, final
        MultipartFile multipartFile, final String cpf) throws
        UploadException {
        val path = String.format(
                "./files/users/%s/attachments/%s.pdf",
```

```
2479                cpf,
2480                attachment.getName()
2481                        .replaceAll(" ", "_")
2482        );
2483        val file = new File(path);
2484        save(file, multipartFile);
2485        return
                Dispatch.builder().attachment(attachment).send(true).build();
2486    }
2487
2488    public String save(final MultipartFile multipartFile, final
            Long codEvent) throws UploadException {
2489        val path = String.format(
2490                "./files/events/%d/attachments/%d.pdf",
2491                codEvent,
2492                System.currentTimeMillis()
2493        );
2494        return save(new File(path), multipartFile);
2495    }
2496
2497    public String saveAttachment(final MultipartFile
            multipartFile, final Long codAttachment) throws
            UploadException {
2498        val path = String.format(
2499                "./files/attachments/%d.pdf",
2500                codAttachment
2501        );
2502        return save(new File(path), multipartFile);
2503    }
2504
2505    private String save(final File file, final MultipartFile
            multipart) throws UploadException{
2506        try {
2507            if (file.getParentFile().mkdirs()){
2508                log.info("create dir: " + file.getParent());
2509            }
2510            Files.copy(multipart.getInputStream(),
                    Paths.get(file.toURI()));
2511            return file.getName();
2512        } catch (IOException e) {
2513            throw new UploadException(String.format("No foi
                    possvel salvar o arquivo %s.", file.getPath()),
                    e);
2514        }
2515    }
```

```
2516
2517        public FileSystemResource get(final Long codAttachment) {
2518            val path = String.format(
2519                    "./files/attachments/%d.pdf",
2520                    codAttachment
2521            );
2522            return new FileSystemResource(path);
2523        }
2524    }
2525
2526
2527    src/main/java/br/ufsc/cultivar/utils/ValidateUtils.java
2528
2529    package br.ufsc.cultivar.utils;
2530
2531    import lombok.val;
2532
2533    import javax.validation.Validation;
2534
2535    public class ValidateUtils {
2536
2537        public static <T> Boolean isValid(T entity){
2538            if (entity == null){
2539                return false;
2540            }
2541            val factory = Validation.buildDefaultValidatorFactory();
2542            val validator = factory.getValidator();
2543            val validate = validator.validate(entity);
2544            return validate.isEmpty();
2545        }
2546    }
2547
2548
2549    src/main/java/br/ufsc/cultivar/utils/DatabaseUtils.java
2550
2551    package br.ufsc.cultivar.utils;
2552
2553    import java.sql.ResultSet;
2554    import java.sql.SQLException;
2555
2556    public class DatabaseUtils {
2557
2558        public static boolean isNotEmpty(ResultSet rs) throws
                SQLException {
2559            if (rs.isBeforeFirst()){
```

```
            rs.first();
        }
        return rs.isLast() || rs.next() && rs.previous();
    }
}


src/main/java/br/ufsc/cultivar/utils/DateUtils.java

package br.ufsc.cultivar.utils;

import java.sql.Date;
import java.sql.Timestamp;

public class DateUtils {
    public static java.util.Date toDate(Timestamp timestamp) {
        return Date.from(timestamp.toInstant());
    }
}


src/main/java/br/ufsc/cultivar/model/evaluate/AnswerTechnologyEnum.java

package br.ufsc.cultivar.model.evaluate;

public enum AnswerTechnologyEnum {
    NOTHING, BEGINNER, PROFICIENT, EXPERT
}


src/main/java/br/ufsc/cultivar/model/evaluate/Skill.java

package br.ufsc.cultivar.model.evaluate;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

@Value
```

```
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Skill.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Skill {
    Long codSkill;
    @NotNull
    @NotBlank
    String name;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}
```

src/main/java/br/ufsc/cultivar/model/evaluate/Expectation.java

```
package br.ufsc.cultivar.model.evaluate;

public enum Expectation {
    ALL, SOME, NOTHING
}
```

src/main/java/br/ufsc/cultivar/model/evaluate/Difficulty.java

```
package br.ufsc.cultivar.model.evaluate;

public enum Difficulty {
    VERY_EASY, EASY, HARD, VERY_HARD
}
```

src/main/java/br/ufsc/cultivar/model/evaluate/Enjoy.java

```
package br.ufsc.cultivar.model.evaluate;

public enum Enjoy {
    VERY_FUNNY, FUNNY, BORED, VERY_BORED
}
```

src/main/java/br/ufsc/cultivar/model/evaluate/Technology.java

```
2650 package br.ufsc.cultivar.model.evaluate;
2651
2652 import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
2653 import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
2654 import lombok.AccessLevel;
2655 import lombok.Builder;
2656 import lombok.Value;
2657 import lombok.experimental.FieldDefaults;
2658
2659 @Value
2660 @Builder(builderClassName = "Builder")
2661 @JsonDeserialize(builder = Technology.Builder.class)
2662 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
2663 public class Technology {
2664
2665     Long codTechnology;
2666     String name;
2667
2668     @JsonPOJOBuilder(withPrefix = "")
2669     public static class Builder{
2670         public Builder(){}
2671     }
2672 }
2673
2674
2675 src/main/java/br/ufsc/cultivar/model/evaluate/AnswerPersonalityEnum.java
2676
2677 package br.ufsc.cultivar.model.evaluate;
2678
2679 public enum AnswerPersonalityEnum {
2680     TOTALLY_AGREE, AGREE, DISAGREE, TOTALLY_DISAGREE
2681 }
2682
2683
2684 src/main/java/br/ufsc/cultivar/model/evaluate/Experience.java
2685
2686 package br.ufsc.cultivar.model.evaluate;
2687
2688 import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
2689 import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
2690 import lombok.AccessLevel;
2691 import lombok.Builder;
2692 import lombok.Value;
2693 import lombok.experimental.FieldDefaults;
2694
```

```
@Value
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Experience.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Experience {

    String experience;
    Difficulty difficulty;
    Expectation expectation;
    Enjoy flEnjoy;
    String enjoy;
    String notEnjoy;
    String suggest;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/evaluate/Personality.java

package br.ufsc.cultivar.model.evaluate;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

@Value
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Personality.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Personality {

    Long codQuestion;
    @NotNull
    @NotBlank
    String question;
```

```
2740
2741        @JsonPOJOBuilder(withPrefix = "")
2742        public static class Builder{
2743            public Builder(){}
2744        }
2745    }
2746
2747
2748    src/main/java/br/ufsc/cultivar/model/evaluate/Mentoring.java
2749
2750    package br.ufsc.cultivar.model.evaluate;
2751
2752    import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
2753    import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
2754    import lombok.AccessLevel;
2755    import lombok.Builder;
2756    import lombok.Value;
2757    import lombok.experimental.FieldDefaults;
2758
2759    @Value
2760    @Builder(builderClassName = "Builder")
2761    @JsonDeserialize(builder = Mentoring.Builder.class)
2762    @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
2763    public class Mentoring {
2764        Long codQuestion;
2765        String question;
2766
2767        @JsonPOJOBuilder(withPrefix = "")
2768        public static class Builder{
2769            public Builder(){}
2770        }
2771    }
2772
2773
2774    src/main/java/br/ufsc/cultivar/model/evaluate/AnswerTechnology.java
2775
2776    package br.ufsc.cultivar.model.evaluate;
2777
2778    import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
2779    import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
2780    import lombok.AccessLevel;
2781    import lombok.Builder;
2782    import lombok.Value;
2783    import lombok.experimental.FieldDefaults;
2784
```

```
@Value
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Technology.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class AnswerTechnology {
    Technology technology;
    AnswerTechnologyEnum answer;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/evaluate/AnswerPersonality.java

package br.ufsc.cultivar.model.evaluate;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;

@Value
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = AnswerPersonality.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class AnswerPersonality {
    Personality question;
    AnswerPersonalityEnum answer;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/Answer.java

package br.ufsc.cultivar.model;

```

```
import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;
import lombok.experimental.Wither;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

@Value
@Wither
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Answer.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Answer {
    @Valid
    @NotNull
    Question question;
    @NotBlank
    @NotNull
    Boolean answer;
    String comment;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/Status.java

package br.ufsc.cultivar.model;

public enum Status {
    APPROVED,
    WAIT_STATEMENT,
    WAIT_COMPANY,
    WAIT_TRAINING,
    REGISTER;

    public boolean isValid(Status next){
```

```
2875        switch (this){
2876            case APPROVED: return this.equals(next);
2877            case WAIT_TRAINING: return APPROVED.equals(next) ||
                    this.equals(next);
2878            case WAIT_STATEMENT: return WAIT_TRAINING.equals(next)
                    || this.equals(next);
2879            case WAIT_COMPANY: return WAIT_STATEMENT.equals(next)
                    || this.equals(next);
2880            case REGISTER: return WAIT_COMPANY.equals(next) ||
                    this.equals(next);
2881            default: return false;
2882        }
2883    }
2884
2885
2886 }
2887
2888
2889 src/main/java/br/ufsc/cultivar/model/School.java
2890
2891 package br.ufsc.cultivar.model;
2892
2893 import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
2894 import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
2895 import lombok.AccessLevel;
2896 import lombok.Builder;
2897 import lombok.Value;
2898 import lombok.experimental.FieldDefaults;
2899 import lombok.experimental.Wither;
2900
2901 import javax.validation.Valid;
2902 import javax.validation.constraints.NotBlank;
2903 import javax.validation.constraints.NotNull;
2904
2905 @Value
2906 @Wither
2907 @Builder(builderClassName = "Builder")
2908 @JsonDeserialize(builder = School.Builder.class)
2909 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
2910 public class School {
2911     Long codSchool;
2912     @NotBlank
2913     @NotNull
2914     String name;
2915     @NotBlank
```

```
    @NotNull
    String phone;
    @Valid
    @NotNull
    Address address;
    @Valid
    @NotNull
    User responsible;
    @NotNull
    SchoolType type;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/Volunteer.java

package br.ufsc.cultivar.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;
import lombok.experimental.Wither;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

@Value
@Wither
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Volunteer.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Volunteer {
    @Valid
```

```
    @NotNull
    User user;
    @Valid
    @NotNull
    Company company;
    @NotNull
    Schooling schooling;
    @NotNull
    Boolean conclusion;
    @NotBlank
    @NotNull
    String course;
    @NotBlank
    @NotNull
    String rg;
    @Valid
    @NotNull
    School school;
    List<@Valid Dispatch> dispatches;
    @NotNull
    @NotEmpty
    List<Answer> answers;
    List<@Valid Rating> ratings;

    public Double getRating() {
        return Optional.ofNullable(ratings)
                .orElse(new ArrayList<>())
                .stream()
                .mapToInt(Rating::getGrade)
                .average()
                .orElse(0.);
    }

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/User.java

package br.ufsc.cultivar.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
```

```
3006 import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
3007 import lombok.AccessLevel;
3008 import lombok.Builder;
3009 import lombok.Value;
3010 import lombok.experimental.FieldDefaults;
3011 import lombok.experimental.Wither;
3012
3013 import javax.validation.Valid;
3014 import javax.validation.constraints.NotBlank;
3015 import javax.validation.constraints.NotNull;
3016 import java.util.Date;
3017
3018 @Value
3019 @Wither
3020 @Builder(builderClassName = "Builder")
3021 @JsonDeserialize(builder = User.Builder.class)
3022 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
3023 public class User {
3024     String cpf;
3025     @NotBlank
3026     @NotNull
3027     String name;
3028     @NotBlank
3029     @NotNull
3030     String email;
3031     @NotBlank
3032     String password;
3033     @NotNull
3034     Role role;
3035     @NotNull
3036     Status status;
3037     @NotNull
3038     Date birth;
3039     @NotBlank
3040     @NotNull
3041     String job;
3042     @NotBlank
3043     @NotNull
3044     String phone;
3045     @Valid
3046     @NotNull
3047     Address address;
3048
3049     @JsonPOJOBuilder(withPrefix = "")
3050     public static class Builder{
```

```
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/SchoolType.java

package br.ufsc.cultivar.model;

public enum SchoolType {
    MUNICIPAL,
    STATE,
    FEDERAL
}

src/main/java/br/ufsc/cultivar/model/Address.java

package br.ufsc.cultivar.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;
import lombok.experimental.Wither;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

@Value
@Wither
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Address.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Address {
    Long codAddress;
    @NotBlank
    @NotNull
    String city;
    @NotBlank
    @NotNull
    String neighborhood;
    @NotBlank
    @NotNull
```

```
3096        String street;
3097        String number;
3098
3099        @JsonPOJOBuilder(withPrefix = "")
3100        public static class Builder{
3101            public Builder(){}
3102        }
3103    }
3104
3105
3106    src/main/java/br/ufsc/cultivar/model/Event.java
3107
3108    package br.ufsc.cultivar.model;
3109
3110    import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
3111    import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
3112    import lombok.AccessLevel;
3113    import lombok.Builder;
3114    import lombok.Value;
3115    import lombok.experimental.FieldDefaults;
3116    import lombok.experimental.Wither;
3117
3118    import javax.validation.Valid;
3119    import javax.validation.constraints.NotBlank;
3120    import javax.validation.constraints.NotEmpty;
3121    import javax.validation.constraints.NotNull;
3122    import java.util.ArrayList;
3123    import java.util.Date;
3124    import java.util.List;
3125    import java.util.Optional;
3126
3127    @Value
3128    @Wither
3129    @Builder(builderClassName = "Builder")
3130    @JsonDeserialize(builder = Event.Builder.class)
3131    @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
3132    public class Event {
3133        Long codEvent;
3134        @NotBlank
3135        @NotNull
3136        Date startOccurrence;
3137        @NotBlank
3138        @NotNull
3139        Date endOccurrence;
3140        Date createAt;
```

```
    @NotBlank
    @NotNull
    TypeEvent type;
    @NotNull
    Boolean allDay;
    @Valid
    @NotNull
    Address address;
    @NotNull
    Boolean evaluate;
    @Valid
    @NotNull
    School school;
    @Valid
    Project project;
    List<@Valid Training> trainings;
    @NotNull
    @NotEmpty
    List<@Valid User> participants;
    List<@Valid Rating> ratings;

    public Double getRating() {
        return Optional.ofNullable(ratings)
                .orElseGet(ArrayList::new)
                .stream()
                .mapToInt(Rating::getGrade)
                .average()
                .orElse(0.);
    }

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/Dispatch.java

package br.ufsc.cultivar.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
```

```
import lombok.Value;
import lombok.experimental.FieldDefaults;

import javax.validation.Valid;
import javax.validation.constraints.NotNull;

@Value
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Dispatch.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Dispatch {
    @NotNull
    @Valid
    Attachment attachment;
    Boolean send;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/Question.java

package br.ufsc.cultivar.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

@Value
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Question.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Question {
    Long codQuestion;
    @NotBlank
    @NotNull
```

```
    String question;
    @NotBlank
    @NotNull
    Role responds;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/Schooling.java

package br.ufsc.cultivar.model;

public enum Schooling {
    ELEMENTARY_SCHOOL,
    HIGH_SCHOOL,
    UNIVERSITY_GRADUATE,
    POSTGRADUATE_STUDIES
}


src/main/java/br/ufsc/cultivar/model/Attachment.java

package br.ufsc.cultivar.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

@Value
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Attachment.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Attachment {
    Long codAttachment;
    @NotBlank
```

```
    @NotNull
    String name;
    @NotBlank
    @NotNull
    Boolean required;
    @NotBlank
    @NotNull
    Status status;
    Boolean download;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/model/Rating.java

package br.ufsc.cultivar.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;

import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

@Value
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Rating.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Rating {
    @NotBlank
    @NotNull
    Integer grade;
    String comment;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
```

```
}


src/main/java/br/ufsc/cultivar/model/Company.java

package br.ufsc.cultivar.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;
import lombok.experimental.Wither;
import org.hibernate.validator.constraints.br.CNPJ;

import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

@Value
@Wither
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Company.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Company {
    @NotBlank
    @NotNull
    @CNPJ
    String cnpj;
    @NotBlank
    @NotNull
    String name;
    @NotBlank
    @NotNull
    String phone;
    @Valid
    @NotNull
    Address address;
    @Valid
    @NotNull
    User responsible;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
```

```
3366         public Builder(){}
3367     }
3368 }
3369 
3370 
3371 src/main/java/br/ufsc/cultivar/model/Project.java
3372 
3373 package br.ufsc.cultivar.model;
3374 
3375 import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
3376 import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
3377 import lombok.AccessLevel;
3378 import lombok.Builder;
3379 import lombok.Value;
3380 import lombok.experimental.FieldDefaults;
3381 
3382 import javax.validation.constraints.NotBlank;
3383 import javax.validation.constraints.NotNull;
3384 import java.util.Date;
3385 
3386 @Value
3387 @Builder(builderClassName = "Builder")
3388 @JsonDeserialize(builder = Project.Builder.class)
3389 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
3390 public class Project {
3391 
3392     Long codProject;
3393     @NotNull
3394     @NotBlank
3395     String name;
3396     @NotNull
3397     Date start;
3398     @NotNull
3399     Date end;
3400 
3401     @JsonPOJOBuilder(withPrefix = "")
3402     public static class Builder{
3403         public Builder(){}
3404     }
3405 }
3406 
3407 
3408 src/main/java/br/ufsc/cultivar/model/TypeEvent.java
3409 
3410 package br.ufsc.cultivar.model;
```

```
3411
3412 import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
3413 import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
3414 import lombok.AccessLevel;
3415 import lombok.Builder;
3416 import lombok.Value;
3417 import lombok.experimental.FieldDefaults;
3418 import lombok.experimental.Wither;
3419
3420 import javax.validation.Valid;
3421 import javax.validation.constraints.NotBlank;
3422 import javax.validation.constraints.NotEmpty;
3423 import javax.validation.constraints.NotNull;
3424 import java.util.List;
3425
3426 @Value
3427 @Wither
3428 @Builder(builderClassName = "Builder")
3429 @JsonDeserialize(builder = TypeEvent.Builder.class)
3430 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
3431 public class TypeEvent {
3432
3433     Long type;
3434     @NotNull
3435     @NotBlank
3436     String name;
3437     @NotNull
3438     @NotEmpty
3439     List<@Valid @NotNull Training> trainings;
3440
3441     @JsonPOJOBuilder(withPrefix = "")
3442     public static class Builder{
3443         public Builder(){}
3444     }
3445 }
3446
3447
3448 src/main/java/br/ufsc/cultivar/model/Role.java
3449
3450 package br.ufsc.cultivar.model;
3451
3452 public enum Role {
3453     ADMIN,
3454     VOLUNTEER,
3455     COMPANY_ADMIN,
```

```
    SCHOOL_ADMIN
}


src/main/java/br/ufsc/cultivar/model/Training.java

package br.ufsc.cultivar.model;

import com.fasterxml.jackson.databind.annotation.JsonDeserialize;
import com.fasterxml.jackson.databind.annotation.JsonPOJOBuilder;
import lombok.AccessLevel;
import lombok.Builder;
import lombok.Value;
import lombok.experimental.FieldDefaults;
import lombok.experimental.Wither;

import javax.validation.Valid;
import javax.validation.constraints.NotNull;

@Value
@Wither
@Builder(builderClassName = "Builder")
@JsonDeserialize(builder = Training.Builder.class)
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class Training {
    Long codTraining;
    @Valid
    @NotNull
    String name;
    String path;
    String link;

    @JsonPOJOBuilder(withPrefix = "")
    public static class Builder{
        public Builder(){}
    }
}


src/main/java/br/ufsc/cultivar/service/evaluate/SkillService.java

package br.ufsc.cultivar.service.evaluate;

import br.ufsc.cultivar.exception.ServiceException;
import br.ufsc.cultivar.exception.Type;
```

```
import br.ufsc.cultivar.model.evaluate.Skill;
import br.ufsc.cultivar.repository.evaluate.SkillRepository;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import lombok.val;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class SkillService {
    SkillRepository repository;

    public void create(Skill skill){
        repository.create(skill);
    }

    public List<Skill> get(){
        return repository.get();
    }

    public Skill delete(Long codSkill) throws ServiceException {
        val skill = Optional.ofNullable(
            repository.get(codSkill)
        ).orElseThrow(
            () -> new ServiceException(null, null, Type.NOT_FOUND)
        );
        repository.delete(codSkill);
        return skill;
    }
}


src/main/java/br/ufsc/cultivar/service/evaluate/TechnologyService.java

package br.ufsc.cultivar.service.evaluate;

import br.ufsc.cultivar.exception.ServiceException;
import br.ufsc.cultivar.exception.Type;
import br.ufsc.cultivar.model.evaluate.Technology;
```

```
3546 import br.ufsc.cultivar.repository.evaluate.TechnologyRepository;
3547 import lombok.AccessLevel;
3548 import lombok.AllArgsConstructor;
3549 import lombok.experimental.FieldDefaults;
3550 import org.springframework.beans.factory.annotation.Autowired;
3551 import org.springframework.stereotype.Service;
3552
3553 import java.util.List;
3554 import java.util.Optional;
3555
3556 @Service
3557 @AllArgsConstructor(onConstructor = @__(@Autowired))
3558 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
3559 public class TechnologyService {
3560     TechnologyRepository repository;
3561
3562     public void create(Technology technology) throws
          ServiceException {
3563         repository.create(technology);
3564     }
3565
3566     public List<Technology> get() throws ServiceException {
3567         return repository.get();
3568     }
3569
3570
3571     public Technology delete(Long codTechnology) throws
          ServiceException {
3572         Technology technology = repository.get(codTechnology);
3573         Optional.ofNullable(technology).orElseThrow(() -> new
              ServiceException(null, null, Type.NOT_FOUND));
3574         repository.delete(codTechnology);
3575         return technology;
3576     }
3577 }
3578
3579
3580 src/main/java/br/ufsc/cultivar/service/evaluate/MentoringService.java
3581
3582 package br.ufsc.cultivar.service.evaluate;
3583
3584 import br.ufsc.cultivar.model.evaluate.Mentoring;
3585 import br.ufsc.cultivar.repository.evaluate.MentoringRepository;
3586 import lombok.AccessLevel;
3587 import lombok.AllArgsConstructor;
```

```
import lombok.experimental.FieldDefaults;
import lombok.val;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class MentoringService {

    MentoringRepository repository;

    public void create(Mentoring mentoring){
        repository.create(mentoring);
    }

    public List<Mentoring> get(){
        return repository.get();
    }

    public Mentoring delete(Long codQuestion){
        val mentoring = repository.get(codQuestion);
        repository.delete(codQuestion);
        return mentoring;
    }
}
```

src/main/java/br/ufsc/cultivar/service/evaluate/EvaluateService.java

```
package br.ufsc.cultivar.service.evaluate;

import br.ufsc.cultivar.dto.EvaluateDTO;
import br.ufsc.cultivar.exception.ServiceException;
import br.ufsc.cultivar.repository.evaluate.EvaluateRepository;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import lombok.val;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
```

```
3633 import java.util.stream.Collectors;
3634
3635 @Service
3636 @AllArgsConstructor(onConstructor = @__(@Autowired))
3637 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
3638 public class EvaluateService {
3639     EvaluateRepository repository;
3640
3641     public void create(EvaluateDTO dto) throws ServiceException {
3642         val cpf = dto.getCpf();
3643         val codProject = dto.getProject();
3644         dto.getTechnologies()
3645             .forEach(
3646                 answerTechnology ->
                        repository.saveTechnology(answerTechnology,
                        codProject, cpf)
3647             );
3648         dto.getAnswerPersonalities()
3649                 .forEach(
3650                         answerVolunteer ->
                                repository.saveAnswerVolunteer(answerVolunteer,
                                codProject, cpf)
3651                 );
3652         repository.saveExperience(dto.getExperience(),
                codProject, cpf);
3653     }
3654
3655     public List<EvaluateDTO> get(Long codProject) {
3656         return repository.getCpfs(codProject)
3657             .stream()
3658             .map(
3659                 cpf -> EvaluateDTO.builder()
3660                     .cpf(cpf)
3661                     .project(codProject)
3662                     .answerPersonalities(
3663                         repository.getAnswersVolunteer(codProject,
                                cpf)
3664                     ).technologies(
3665                         repository.getAnswersTechnologies(codProject,
                                cpf)
3666                     ).build()
3667             ).collect(Collectors.toList());
3668     }
3669 }
3670
```

src/main/java/br/ufsc/cultivar/service/evaluate/PersonalityService.java

```java
package br.ufsc.cultivar.service.evaluate;

import br.ufsc.cultivar.model.evaluate.Personality;
import
    br.ufsc.cultivar.repository.evaluate.PersonalityRepository;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class PersonalityService {
    PersonalityRepository repository;

    public void create(Personality question) {
        repository.create(question);
    }

    public List<Personality> get() {
        return repository.get();
    }


    public Personality delete(Long codQuestion) {
        Personality question = repository.get(codQuestion);
        repository.delete(codQuestion);
        return question;
    }
}
```

src/main/java/br/ufsc/cultivar/service/UserService.java

```java
package br.ufsc.cultivar.service;

import br.ufsc.cultivar.exception.ServiceException;
import br.ufsc.cultivar.model.User;
```

```
3715 import br.ufsc.cultivar.repository.AddressRepository;
3716 import br.ufsc.cultivar.repository.UserRepository;
3717 import br.ufsc.cultivar.utils.ValidateUtils;
3718 import lombok.AccessLevel;
3719 import lombok.AllArgsConstructor;
3720 import lombok.experimental.FieldDefaults;
3721 import lombok.val;
3722 import org.springframework.beans.factory.annotation.Autowired;
3723 import org.springframework.stereotype.Service;
3724
3725 import java.util.List;
3726 import java.util.Map;
3727
3728 @Service
3729 @AllArgsConstructor(onConstructor = @__(@Autowired))
3730 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
3731 public class UserService {
3732
3733     UserRepository userRepository;
3734     AddressRepository addressRepository;
3735
3736     public void create(final User user) throws ServiceException {
3737         val address = user.getAddress();
3738         if (!ValidateUtils.isValid(address)) {
3739             throw new ServiceException(null, null, null);
3740         }
3741         userRepository.create(
3742                 user.withAddress(
3743                         address.withCodAddress(
3744                                 addressRepository.create(
3745                                         address
3746                                 )
3747                         )
3748                 )
3749         );
3750
3751     }
3752
3753     public List<User> get(final Map<String, Object> filter)
          throws ServiceException {
3754         try {
3755             return userRepository.get(filter);
3756         } catch (RuntimeException e){
3757             val throwable = e.getCause();
3758             if (throwable instanceof ServiceException) {
```

```
3759                throw (ServiceException) throwable;
3760            }
3761            throw e;
3762        }
3763    }
3764
3765    public User get(final String cpf) throws ServiceException {
3766        val user = userRepository.get(cpf);
3767        return user.withAddress(
3768                addressRepository.get(
3769                        user.getAddress().getCodAddress()
3770                )
3771        );
3772    }
3773
3774    public User delete(final String cpf) throws ServiceException
        {
3775        val user = get(cpf);
3776        userRepository.delete(cpf);
3777        return user;
3778    }
3779
3780    public void update(final User user, final String cpf) throws
         ServiceException {
3781        if (!user.getCpf().equals(cpf)) {
3782            throw new ServiceException(null, null, null);
3783        }
3784        if (!get(cpf).getStatus().isValid(user.getStatus())){
3785            throw new ServiceException(null, null, null);
3786        }
3787        userRepository.update(user);
3788    }
3789
3790    List<User> getParticipants(final Long codEvent) {
3791        return userRepository.getParticipants(codEvent);
3792    }
3793 }
3794
3795 src/main/java/br/ufsc/cultivar/service/CompanyService.java
3796
3797 package br.ufsc.cultivar.service;
3798
3799 import br.ufsc.cultivar.dto.PaginateList;
3800 import br.ufsc.cultivar.exception.ServiceException;
3801 import br.ufsc.cultivar.model.Company;
```

```
3802 import br.ufsc.cultivar.repository.AddressRepository;
3803 import br.ufsc.cultivar.repository.CompanyRepository;
3804 import lombok.AccessLevel;
3805 import lombok.AllArgsConstructor;
3806 import lombok.experimental.FieldDefaults;
3807 import lombok.val;
3808 import org.springframework.beans.factory.annotation.Autowired;
3809 import org.springframework.stereotype.Service;
3810
3811 import java.util.ArrayList;
3812 import java.util.List;
3813 import java.util.Map;
3814 import java.util.Optional;
3815 import java.util.stream.Collectors;
3816
3817 @Service
3818 @AllArgsConstructor(onConstructor = @__(@Autowired))
3819 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
3820 public class CompanyService {
3821
3822     CompanyRepository companyRepository;
3823     UserService userService;
3824     AddressRepository addressRepository;
3825
3826     public void create(final Company company) throws
          ServiceException {
3827         userService.create(company.getResponsible());
3828         val address = company.getAddress();
3829         companyRepository.create(
3830             company.withAddress(
3831                 address.withCodAddress(
3832                     addressRepository.create(
3833                         address
3834                     )
3835                 )
3836             )
3837         );
3838     }
3839
3840     public PaginateList get(final String filter, final Long
          page) throws ServiceException {
3841         return PaginateList.builder()
3842             .count(companyRepository.count(filter))
3843             .data(
```

```
                new ArrayList<>(companyRepository.get(filter,
                    page))
            ).build();
    }

    public Company get(final String cnpj) throws
        ServiceException {
        val company = companyRepository.get(cnpj);
        return Optional.ofNullable(company)
                .orElseThrow(
                    () -> new ServiceException(null, null, null)
                ).withAddress(
                    addressRepository.get(
                        company.getAddress()
                            .getCodAddress()
                    )
                ).withResponsible(
                    userService.get(
                        company.getResponsible()
                            .getCpf()
                    )
                );
    }

    public Company delete(String cnpj) throws ServiceException {
        val company = get(cnpj);
        companyRepository.delete(cnpj);
        return Optional.ofNullable(company).orElseThrow(() ->
            new ServiceException(null, null, null));
    }

    public void update(Company company, String cnpj) throws
        ServiceException {
        if(!company.getCnpj().equals(cnpj)){
            throw new ServiceException(null, null, null);
        }
        addressRepository.update(company.getAddress());
        companyRepository.update(company);
    }
}


src/main/java/br/ufsc/cultivar/service/VolunteerService.java

package br.ufsc.cultivar.service;
```

```
3885
3886 import br.ufsc.cultivar.dto.PaginateList;
3887 import br.ufsc.cultivar.email.EmailClient;
3888 import br.ufsc.cultivar.exception.ServiceException;
3889 import br.ufsc.cultivar.exception.Type;
3890 import br.ufsc.cultivar.model.Volunteer;
3891 import br.ufsc.cultivar.repository.*;
3892 import br.ufsc.cultivar.utils.ValidateUtils;
3893 import lombok.AccessLevel;
3894 import lombok.AllArgsConstructor;
3895 import lombok.experimental.FieldDefaults;
3896 import lombok.extern.slf4j.Slf4j;
3897 import lombok.val;
3898 import org.springframework.beans.factory.annotation.Autowired;
3899 import org.springframework.stereotype.Service;
3900 import org.thymeleaf.context.Context;
3901 import org.thymeleaf.spring5.SpringTemplateEngine;
3902
3903 import java.util.ArrayList;
3904 import java.util.Date;
3905 import java.util.List;
3906 import java.util.Optional;
3907 import java.util.stream.Collectors;
3908
3909 @Slf4j
3910 @Service
3911 @AllArgsConstructor(onConstructor = @__(@Autowired))
3912 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
3913 public class VolunteerService {
3914
3915     VolunteerRepository volunteerRepository;
3916     UserService userService;
3917     CompanyService companyService;
3918     RatingRepository ratingRepository;
3919     AnswerRepository answerRepository;
3920     QuestionRepository questionRepository;
3921     EventRepository eventRepository;
3922     DispatchService dispatchService;
3923     EmailClient emailClient;
3924     SpringTemplateEngine templateEngine;
3925
3926     public void create(final Volunteer volunteer) throws
          ServiceException {
3927         val user = volunteer.getUser();
3928         if (!ValidateUtils.isValid(user)) {
```

```
3929            throw new ServiceException(null, null, null);
3930        }
3931        userService.create(user);
3932        volunteerRepository.create(volunteer);
3933        Optional.ofNullable(volunteer.getAnswers())
3934                .ifPresent(
3935                        answers -> answers.forEach(
3936                                answer -> answerRepository.create(
3937                                        answer,
3938                                        user.getCpf()
3939                                )
3940                        )
3941                );
3942        emailClient.sendEmailAsync(user.getEmail(), "Bem vindo",
                buildEmail());
3943    }
3944
3945    private String buildEmail(){
3946        val context = new Context();
3947        context.setVariable("message", "teste volunteer");
3948        return templateEngine.process("newVolunteerEmail",
                context);
3949    }
3950
3951    public Volunteer get(final String cpf) throws
             ServiceException {
3952        val volunteer = volunteerRepository.get(cpf);
3953        return Optional.ofNullable(volunteer)
3954                .orElseThrow(() -> new ServiceException(null,
                         null, Type.NOT_FOUND))
3955                .withUser(
3956                    userService.get(cpf)
3957                ).withCompany(
3958                    companyService.get(
3959                            volunteer.getCompany().getCnpj()
3960                    )
3961                ).withAnswers(
3962                    answerRepository.get(cpf)
3963                        .stream()
3964                        .map(answer -> answer.withQuestion(
3965                                questionRepository.get(
3966                                        answer.getQuestion().getCodQuestion()
3967                                )
3968                            )
3969                        ).collect(Collectors.toList())
```

```
3970                ).withRatings(
3971                    ratingRepository.get(cpf)
3972                ).withDispatches(
3973                    dispatchService.get(cpf)
3974                );
3975        }
3976
3977        public Volunteer delete(final String cpf) throws
                ServiceException {
3978            val hasEvents = eventRepository.eventsByVolunteer(cpf,
                    null)
3979                    .stream()
3980                    .filter(
3981                            event -> event.getEndOccurrence()
3982                                    .after(new Date())
3983                    ).collect(Collectors.toList())
3984                    .isEmpty();
3985            if(hasEvents){
3986                throw new ServiceException(null, null, null);
3987            }
3988            val volunteer = get(cpf);
3989            volunteerRepository.delete(cpf);
3990            return volunteer;
3991        }
3992
3993        public void update(final Volunteer volunteer, final String
                cpf) throws ServiceException {
3994            val user = volunteer.getUser();
3995            if (!user.getCpf().equals(cpf)){
3996                throw new ServiceException(null, null, null);
3997            }
3998            userService.update(user, cpf);
3999            answerRepository.delete(cpf);
4000            volunteer.getAnswers()
4001                    .forEach(answer ->
                            answerRepository.create(answer, cpf));
4002            volunteerRepository.update(volunteer);
4003        }
4004
4005        public PaginateList get(final List<String> filterCompany,
                final List<Long> filterSchool,
4006                                final String filter, final Integer
                                    page) throws ServiceException {
4007            try {
4008                return PaginateList.builder()
```

```
                .count(volunteerRepository.count(filterCompany,
                    filterSchool, filter))
                .data(
                    volunteerRepository.get(filterCompany,
                        filterSchool, filter, page)
                        .stream()
                        .map(volunteer -> {
                            try {
                                return volunteer.withUser(
                                    userService.get(
                                        volunteer.getUser()
                                            .getCpf()
                                    )
                                ).withRatings(
                                    ratingRepository.get(
                                        volunteer.getUser()
                                            .getCpf()
                                    )
                                );
                            } catch (ServiceException e) {
                                throw new RuntimeException(e);
                            }
                        }).collect(Collectors.toList())
                ).build();
        } catch (RuntimeException e){
            throw new ServiceException(null, null, null);
        }
    }
}
```

src/main/java/br/ufsc/cultivar/service/TypeEventService.java

```
package br.ufsc.cultivar.service;

import br.ufsc.cultivar.dto.PaginateList;
import br.ufsc.cultivar.exception.ServiceException;
import br.ufsc.cultivar.exception.Type;
import br.ufsc.cultivar.model.Training;
import br.ufsc.cultivar.model.TypeEvent;
import br.ufsc.cultivar.repository.TrainingRepository;
import br.ufsc.cultivar.repository.TypeEventRepository;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
```

```
4052 import lombok.val;
4053 import org.springframework.beans.factory.annotation.Autowired;
4054 import org.springframework.stereotype.Service;
4055
4056 import java.util.ArrayList;
4057 import java.util.List;
4058 import java.util.Optional;
4059
4060 @Service
4061 @AllArgsConstructor(onConstructor = @__(@Autowired))
4062 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4063 public class TypeEventService {
4064
4065     TypeEventRepository typeEventRepository;
4066     TrainingRepository trainingRepository;
4067
4068     public void create(TypeEvent typeEvent) {
4069         val tpEvent =
                 typeEventRepository.create(typeEvent.getName());
4070         typeEvent.getTrainings()
4071                 .forEach(
4072                     training ->
                                 trainingRepository.create(training,
                                 tpEvent)
4073                 );
4074     }
4075
4076     public PaginateList get(final String filter, final Long
          page) {
4077         return PaginateList.builder()
4078                 .count(typeEventRepository.count(filter))
4079                 .data(
4080                     new
                                 ArrayList<>(typeEventRepository.get(filter,
                                 page))
4081                 ).build();
4082     }
4083
4084     public TypeEvent get(Long tpEvent) {
4085         return typeEventRepository.get(tpEvent)
4086                 .withTrainings(trainingRepository.getByTypeEvent(tpEvent));
4087     }
4088
4089     public TypeEvent delete(Long tpEvent) {
4090         val typeEvent = get(tpEvent);
```

```
4091        typeEventRepository.delete(tpEvent);
4092        return typeEvent;
4093    }
4094
4095    public List<Training> getTrainings(Long tpEvent) {
4096        return trainingRepository.getByTypeEvent(tpEvent);
4097    }
4098 }
4099
4100
4101 src/main/java/br/ufsc/cultivar/service/DispatchService.java
4102
4103 package br.ufsc.cultivar.service;
4104
4105 import br.ufsc.cultivar.exception.ServiceException;
4106 import br.ufsc.cultivar.exception.Type;
4107 import br.ufsc.cultivar.exception.UploadException;
4108 import br.ufsc.cultivar.model.Attachment;
4109 import br.ufsc.cultivar.model.Dispatch;
4110 import br.ufsc.cultivar.repository.DispatchRepository;
4111 import br.ufsc.cultivar.utils.FileUtils;
4112 import lombok.AccessLevel;
4113 import lombok.AllArgsConstructor;
4114 import lombok.experimental.FieldDefaults;
4115 import org.springframework.beans.factory.annotation.Autowired;
4116 import org.springframework.stereotype.Service;
4117 import org.springframework.web.multipart.MultipartFile;
4118
4119 import java.util.List;
4120 import java.util.Optional;
4121
4122 @Service
4123 @AllArgsConstructor(onConstructor = @__(@Autowired))
4124 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4125 public class DispatchService {
4126
4127     DispatchRepository repository;
4128     FileUtils fileUtils;
4129
4130     public void save(final Attachment attachment, final
          MultipartFile file, final String cpf) throws
          ServiceException {
4131         try {
4132             repository.save(fileUtils.save(attachment, file,
                  cpf), cpf);
```

```
4133            } catch (UploadException e) {
4134                throw new ServiceException(null, null, Type.FILE);
4135            }
4136        }
4137
4138        public Dispatch get(final String cpf, final Long
                codAttachment) {
4139            return repository.get(cpf, codAttachment);
4140        }
4141
4142        public List<Dispatch> get(String cpf) throws
                ServiceException {
4143            return Optional.ofNullable(repository.get(cpf))
4144                    .orElseThrow(() -> new ServiceException(null,
                            null, Type.NOT_FOUND));
4145        }
4146    }
4147
4148
4149    src/main/java/br/ufsc/cultivar/service/AttachmentService.java
4150
4151    package br.ufsc.cultivar.service;
4152
4153    import br.ufsc.cultivar.dto.PaginateList;
4154    import br.ufsc.cultivar.exception.ServiceException;
4155    import br.ufsc.cultivar.exception.Type;
4156    import br.ufsc.cultivar.exception.UploadException;
4157    import br.ufsc.cultivar.model.Attachment;
4158    import br.ufsc.cultivar.model.Status;
4159    import br.ufsc.cultivar.repository.AttachmentRepository;
4160    import br.ufsc.cultivar.utils.FileUtils;
4161    import lombok.AccessLevel;
4162    import lombok.AllArgsConstructor;
4163    import lombok.experimental.FieldDefaults;
4164    import lombok.val;
4165    import org.springframework.beans.factory.annotation.Autowired;
4166    import org.springframework.core.io.Resource;
4167    import org.springframework.stereotype.Service;
4168    import org.springframework.web.multipart.MultipartFile;
4169
4170    import java.util.ArrayList;
4171    import java.util.List;
4172    import java.util.Objects;
4173    import java.util.Optional;
4174
```

```
@Service
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class AttachmentService {

    AttachmentRepository repository;
    FileUtils fileUtils;

    public void create(final Attachment attachment, final
        MultipartFile file) throws ServiceException {
        if(attachment.getDownload() && Objects.isNull(file)){
            throw new ServiceException(null, null, null);
        }
        val codAttachment = repository.create(attachment);
        if (attachment.getDownload()){
            try {
                fileUtils.saveAttachment(file, codAttachment);
            } catch (UploadException e) {
                throw new ServiceException(e.getMessage(), e,
                    Type.FILE);
            }
        }
    }

    public PaginateList get(final String filter, final Long
        page) throws ServiceException {
        return PaginateList.builder()
                .data(
                        new ArrayList<>(repository.get(filter,
                            page))
                ).count(
                        repository.count(filter)
                ).build();
    }
    public List<Attachment> get(final Status status) throws
        ServiceException {
        return repository.get(status);
    }

    public Attachment get(final Long codAttachment) throws
        ServiceException {
        val attachment = repository.get(codAttachment);
        return Optional.ofNullable(attachment)
            .orElseThrow(() -> new ServiceException(null, null,
                null));
```

```
4213     }
4214
4215     public Resource getAsFile(final Long codAttachment) throws
             ServiceException {
4216         return fileUtils.get(
4217             Optional.ofNullable(repository.get(codAttachment))
4218                 .orElseThrow(
4219                     () -> new ServiceException(null, null,
                                 Type.NOT_FOUND)
4220                 ).getCodAttachment()
4221         );
4222     }
4223
4224     public Attachment delete(final Long codAttachment) throws
             ServiceException {
4225         val attachment = get(codAttachment);
4226         repository.delete(codAttachment);
4227         return attachment;
4228     }
4229
4230     public void update(final Attachment attachment, final Long
             codAttachment) throws ServiceException {
4231         if (attachment.getCodAttachment().equals(codAttachment)){
4232             throw new ServiceException(null, null, null);
4233         }
4234         repository.update(attachment);
4235     }
4236 }
4237
4238
4239 src/main/java/br/ufsc/cultivar/service/ProjectService.java
4240
4241 package br.ufsc.cultivar.service;
4242
4243 import br.ufsc.cultivar.dto.PaginateList;
4244 import br.ufsc.cultivar.model.Project;
4245 import br.ufsc.cultivar.repository.ProjectRepository;
4246 import lombok.AccessLevel;
4247 import lombok.AllArgsConstructor;
4248 import lombok.experimental.FieldDefaults;
4249 import lombok.val;
4250 import org.springframework.beans.factory.annotation.Autowired;
4251 import org.springframework.stereotype.Service;
4252
4253 import java.util.ArrayList;
```

```
import java.util.List;

@Service
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class ProjectService {

    ProjectRepository projectRepository;

    public void create(Project project) {
        projectRepository.create(project);
    }

    public PaginateList get(final String filter, final Long
        page) {
        return PaginateList.builder()
                .count(projectRepository.count(filter))
                .data(
                    new ArrayList<>(projectRepository.get(filter,
                        page))
                ).build();
    }

    public Project get(Long codProject) {
        return projectRepository.get(codProject);
    }

    public Project delete(Long codProject) {
        val project = get(codProject);
        projectRepository.delete(codProject);
        return project;
    }
}
```

src/main/java/br/ufsc/cultivar/service/QuestionService.java

```
package br.ufsc.cultivar.service;

import br.ufsc.cultivar.exception.ServiceException;
import br.ufsc.cultivar.model.Question;
import br.ufsc.cultivar.model.Role;
import br.ufsc.cultivar.repository.QuestionRepository;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
```

```
4297 import lombok.experimental.FieldDefaults;
4298 import lombok.val;
4299 import org.springframework.beans.factory.annotation.Autowired;
4300 import org.springframework.stereotype.Service;
4301
4302 import java.util.List;
4303 import java.util.Map;
4304
4305 @Service
4306 @AllArgsConstructor(onConstructor = @__(@Autowired))
4307 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4308 public class QuestionService {
4309
4310     QuestionRepository repository;
4311
4312     public void create(final Question question) throws
          ServiceException {
4313         repository.create(question);
4314     }
4315
4316     public List<Question> get() throws ServiceException {
4317         return repository.get();
4318     }
4319
4320     public List<Question> get(Role responds) throws
          ServiceException {
4321         return repository.get(responds);
4322     }
4323
4324     public Question get(final Long codQuestion) throws
          ServiceException {
4325         return repository.get(codQuestion);
4326     }
4327
4328     public Question delete(final Long codQuestion) throws
          ServiceException {
4329         val question = get(codQuestion);
4330         repository.delete(codQuestion);
4331         return question;
4332     }
4333
4334     public void update(final Question question, final Long
          codQuestion) throws ServiceException {
4335         if (question.getCodQuestion().equals(codQuestion)){
4336             throw new ServiceException(null, null, null);
```

```
        }
        repository.update(question);
    }
}


src/main/java/br/ufsc/cultivar/service/SchoolService.java

package br.ufsc.cultivar.service;

import br.ufsc.cultivar.dto.PaginateList;
import br.ufsc.cultivar.exception.ServiceException;
import br.ufsc.cultivar.model.School;
import br.ufsc.cultivar.repository.AddressRepository;
import br.ufsc.cultivar.repository.SchoolRepository;
import br.ufsc.cultivar.utils.ValidateUtils;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import lombok.val;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

@Service
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class SchoolService {

    SchoolRepository schoolRepository;
    AddressRepository addressRepository;
    UserService userService;

    public School create(final School school) throws
         ServiceException {
        val address = school.getAddress();
        if (!ValidateUtils.isValid(address)){
            throw new ServiceException(null, null, null);
        }
        userService.create(school.getResponsible());
        return school.withCodSchool(
                schoolRepository.create(
```

```
4381                        school.withAddress(
4382                                address.withCodAddress(
4383                                        addressRepository.create(address)
4384                                )
4385                        )
4386                )
4387        );
4388    }
4389
4390    public PaginateList get(final String filter, final Long
        page) throws ServiceException {
4391        return PaginateList.builder()
4392                .count(schoolRepository.count(filter))
4393                .data(new
                    ArrayList<>(schoolRepository.get(filter,
                    page)))
4394                .build();
4395    }
4396
4397    public School get(final Long codSchool) throws
        ServiceException {
4398        val school = schoolRepository.get(codSchool);
4399        return school.withAddress(
4400                addressRepository.get(
4401                        school.getAddress().getCodAddress()
4402                )
4403        ).withResponsible(
4404                userService.get(
4405                        school.getResponsible().getCpf()
4406                )
4407        );
4408    }
4409
4410    public School delete(final Long codSchool) throws
        ServiceException {
4411        val school = get(codSchool);
4412        schoolRepository.delete(codSchool);
4413        return school;
4414    }
4415
4416    public void update(final School school, final Long
        codSchool) throws ServiceException {
4417        if (school.getCodSchool().equals(codSchool)){
4418            throw new ServiceException(null, null, null);
4419        }
```

```
4420            schoolRepository.update(school);
4421        }
4422    }
4423
4424
4425    src/main/java/br/ufsc/cultivar/service/EventService.java
4426
4427    package br.ufsc.cultivar.service;
4428
4429    import br.ufsc.cultivar.dto.ParticipationDTO;
4430    import br.ufsc.cultivar.dto.UserEventsDTO;
4431    import br.ufsc.cultivar.exception.ServiceException;
4432    import br.ufsc.cultivar.exception.Type;
4433    import br.ufsc.cultivar.exception.UploadException;
4434    import br.ufsc.cultivar.model.Event;
4435    import br.ufsc.cultivar.repository.*;
4436    import br.ufsc.cultivar.utils.FileUtils;
4437    import lombok.AccessLevel;
4438    import lombok.AllArgsConstructor;
4439    import lombok.experimental.FieldDefaults;
4440    import lombok.val;
4441    import org.springframework.beans.factory.annotation.Autowired;
4442    import org.springframework.stereotype.Service;
4443    import org.springframework.web.multipart.MultipartFile;
4444
4445    import java.util.*;
4446    import java.util.stream.Collectors;
4447
4448    @Service
4449    @AllArgsConstructor(onConstructor = @__(@Autowired))
4450    @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4451    public class EventService {
4452
4453        FileUtils fileUtils;
4454        EventRepository eventRepository;
4455        AddressRepository addressRepository;
4456        RatingRepository ratingRepository;
4457        UserRepository userRepository;
4458        ParticipationRepository participationRepository;
4459        TrainingRepository trainingRepository;
4460
4461        public String upload(MultipartFile file, Long codEvent)
                 throws ServiceException{
4462            try {
4463                return fileUtils.save(file, codEvent);
```

```
4464            } catch (UploadException e) {
4465                throw new ServiceException(e.getMessage(), e,
                        Type.FILE);
4466            }
4467        }
4468
4469        public void create(final Event event) throws
                ServiceException {
4470            val address = event.getAddress();
4471            val codEvent = eventRepository.create(
4472                    event.withAddress(
4473                            address.withCodAddress(
4474                                    addressRepository.create(
4475                                            address
4476                                    )
4477                            )
4478                    )
4479            );
4480            Optional.ofNullable(event.getParticipants())
4481                    .orElseGet(ArrayList::new)
4482                    .forEach(
4483                            user ->
                                participationRepository.create(codEvent,
                                user.getCpf())
4484                    );
4485        }
4486
4487        public Set<Event> get(final List<String> filterVolunteer,
                final List<Long> filterSchool,
4488                              final Long filterProject) throws
                                  ServiceException {
4489            return new
                    HashSet<>(eventRepository.get(filterVolunteer,
                    filterSchool, filterProject));
4490        }
4491
4492        public Event get(final Long codEvent) throws
                ServiceException {
4493            val event =
                    Optional.ofNullable(eventRepository.get(codEvent))
4494                    .orElseThrow(() -> new ServiceException(null, null,
                            null));
4495            val type = event.getType();
4496            return event.withAddress(
4497                    addressRepository.get(
```

```
4498                event.getAddress().getCodAddress()
4499            )
4500        ).withParticipants(
4501            userRepository.getParticipants(codEvent)
4502        ).withRatings(
4503            ratingRepository.get(codEvent)
4504        ).withTrainings(
4505            trainingRepository.getByEvent(codEvent)
4506        ).withType(
4507            type.withTrainings(
4508                trainingRepository.getByTypeEvent(
4509                    type.getType()
4510                )
4511            )
4512        );
4513    }
4514
4515    public Event delete(final Long codEvent) throws
         ServiceException {
4516        val event = get(codEvent);
4517        eventRepository.delete(codEvent);
4518        return event;
4519    }
4520
4521    public void update(final Event event, final Long codEvent)
         throws ServiceException {
4522        if (event.getCodEvent().equals(codEvent)){
4523            throw new ServiceException(null, null, null);
4524        }
4525        eventRepository.update(event);
4526    }
4527
4528    public Set<ParticipationDTO> getVolunteerLocals(final String
         codCpf){
4529        return new
             HashSet<>(participationRepository.getVolunteerLocals(codCpf));
4530    }
4531
4532    public List<Event> eventsByVolunteer(final String cpf, final
         Long type) {
4533        return eventRepository.eventsByVolunteer(cpf, type);
4534    }
4535
4536    public List<Event> eventsBySchool(Long codSchool, Long type)
         {
```

```
4537            return eventRepository.eventsBySchool(codSchool, type);
4538        }
4539
4540        public List<UserEventsDTO> getEventsToAlert() {
4541            return userRepository.get((Map<String, Object>) null)
4542                .stream()
4543                .map(
4544                    user ->
4545                        UserEventsDTO.builder()
4546                            .email(user.getEmail())
4547                            .name(user.getName())
4548                            .events(
4549                                eventRepository.getEventsToAlert(user.getCpf())
4550                            ).build()
4551                ).collect(Collectors.toList());
4552        }
4553
4554        public List<Event> getEventsToEvaluateByVolunteer(final
                String cpf) {
4555            return
                    eventRepository.getEventsToEvaluateByVolunteer(cpf);
4556        }
4557    }
4558
4559
4560    src/main/java/br/ufsc/cultivar/service/RatingService.java
4561
4562    package br.ufsc.cultivar.service;
4563
4564    import br.ufsc.cultivar.model.Rating;
4565    import br.ufsc.cultivar.repository.RatingRepository;
4566    import lombok.AccessLevel;
4567    import lombok.AllArgsConstructor;
4568    import lombok.experimental.FieldDefaults;
4569    import org.springframework.beans.factory.annotation.Autowired;
4570    import org.springframework.stereotype.Service;
4571
4572    @Service
4573    @AllArgsConstructor(onConstructor = @__(@Autowired))
4574    @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4575    public class RatingService {
4576
4577        RatingRepository repository;
4578
4579        public void create(Rating rating, String cpf) {
```

```
        repository.create(rating, cpf);
    }

    public void create(Rating rating, Long codEvent) {
        repository.create(rating, codEvent);
    }
}


src/main/java/br/ufsc/cultivar/resource/evaluate/PersonalityResource.java

package br.ufsc.cultivar.resource.evaluate;

import br.ufsc.cultivar.exception.ServiceException;
import br.ufsc.cultivar.model.evaluate.Personality;
import br.ufsc.cultivar.service.evaluate.PersonalityService;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping(path = "/personality")
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class PersonalityResource {
    PersonalityService service;

    @PostMapping(consumes =
        MediaType.APPLICATION_JSON_UTF8_VALUE)
    @ResponseStatus(code = HttpStatus.CREATED)
    public void create(@RequestBody final Personality question)
        throws ServiceException {
        service.create(question);
    }

    @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
    public List<Personality> get() throws ServiceException{
        return service.get();
    }
```

```
4623
4624        @DeleteMapping(path = "/{codQuestion}", produces =
                MediaType.APPLICATION_JSON_UTF8_VALUE)
4625        public Personality delete(@PathVariable final Long
                codQuestion) throws ServiceException{
4626            return service.delete(codQuestion);
4627        }
4628    }
4629
4630
4631    src/main/java/br/ufsc/cultivar/resource/evaluate/MentoringResource.java
4632
4633    package br.ufsc.cultivar.resource.evaluate;
4634
4635    import br.ufsc.cultivar.exception.ServiceException;
4636    import br.ufsc.cultivar.model.evaluate.Mentoring;
4637    import br.ufsc.cultivar.model.evaluate.Personality;
4638    import br.ufsc.cultivar.service.evaluate.MentoringService;
4639    import br.ufsc.cultivar.service.evaluate.PersonalityService;
4640    import lombok.AccessLevel;
4641    import lombok.AllArgsConstructor;
4642    import lombok.experimental.FieldDefaults;
4643    import org.springframework.beans.factory.annotation.Autowired;
4644    import org.springframework.http.HttpStatus;
4645    import org.springframework.http.MediaType;
4646    import org.springframework.web.bind.annotation.*;
4647
4648    import java.util.List;
4649
4650    @RestController
4651    @RequestMapping(path = "/mentoring")
4652    @AllArgsConstructor(onConstructor = @__(@Autowired))
4653    @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4654    public class MentoringResource {
4655        MentoringService service;
4656
4657        @PostMapping(consumes =
                MediaType.APPLICATION_JSON_UTF8_VALUE)
4658        @ResponseStatus(code = HttpStatus.CREATED)
4659        public void create(@RequestBody final Mentoring question)
                throws ServiceException {
4660            service.create(question);
4661        }
4662
4663        @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
```

```
4664     public List<Mentoring> get() throws ServiceException{
4665         return service.get();
4666     }
4667 
4668     @DeleteMapping(path = "/{codQuestion}", produces =
             MediaType.APPLICATION_JSON_UTF8_VALUE)
4669     public Mentoring delete(@PathVariable final Long
             codQuestion) throws ServiceException{
4670         return service.delete(codQuestion);
4671     }
4672 }
4673 
4674 src/main/java/br/ufsc/cultivar/resource/evaluate/SkillResource.java
4675 
4676 package br.ufsc.cultivar.resource.evaluate;
4677 
4678 import br.ufsc.cultivar.exception.ServiceException;
4679 import br.ufsc.cultivar.model.evaluate.Skill;
4680 import br.ufsc.cultivar.service.evaluate.SkillService;
4681 import lombok.AccessLevel;
4682 import lombok.AllArgsConstructor;
4683 import lombok.experimental.FieldDefaults;
4684 import org.springframework.beans.factory.annotation.Autowired;
4685 import org.springframework.http.HttpStatus;
4686 import org.springframework.http.MediaType;
4687 import org.springframework.web.bind.annotation.*;
4688 
4689 import java.util.List;
4690 
4691 @RestController
4692 @RequestMapping(path = "/skill")
4693 @AllArgsConstructor(onConstructor = @__(@Autowired))
4694 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4695 public class SkillResource {
4696     SkillService service;
4697 
4698     @PostMapping(consumes =
             MediaType.APPLICATION_JSON_UTF8_VALUE)
4699     @ResponseStatus(code = HttpStatus.CREATED)
4700     public void create(@RequestBody final Skill skill) throws
             ServiceException {
4701         service.create(skill);
4702     }
4703 
4704     @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
```

```
4705        public List<Skill> get() throws ServiceException{
4706            return service.get();
4707        }
4708
4709        @DeleteMapping(path = "/{codSkill}", produces =
                MediaType.APPLICATION_JSON_UTF8_VALUE)
4710        public Skill delete(@PathVariable final Long codSkill)
                throws ServiceException{
4711            return service.delete(codSkill);
4712        }
4713    }
4714
4715
4716    src/main/java/br/ufsc/cultivar/resource/evaluate/EvaluateProjectResource.j
4717
4718    package br.ufsc.cultivar.resource.evaluate;
4719
4720    import br.ufsc.cultivar.dto.EvaluateDTO;
4721    import br.ufsc.cultivar.exception.ServiceException;
4722    import br.ufsc.cultivar.service.evaluate.EvaluateService;
4723    import lombok.AccessLevel;
4724    import lombok.AllArgsConstructor;
4725    import lombok.experimental.FieldDefaults;
4726    import org.springframework.beans.factory.annotation.Autowired;
4727    import org.springframework.http.HttpStatus;
4728    import org.springframework.http.MediaType;
4729    import org.springframework.web.bind.annotation.*;
4730
4731    import javax.validation.Valid;
4732    import java.util.List;
4733
4734    @RestController
4735    @RequestMapping(path = "/evaluate/project")
4736    @AllArgsConstructor(onConstructor = @__(@Autowired))
4737    @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4738    public class EvaluateProjectResource {
4739
4740        EvaluateService evaluateService;
4741
4742        @PostMapping(consumes =
                MediaType.APPLICATION_JSON_UTF8_VALUE)
4743        @ResponseStatus(HttpStatus.CREATED)
4744        public void create(@RequestBody @Valid final EvaluateDTO
             dto) throws ServiceException {
4745            evaluateService.create(dto);
```

```
4746     }
4747
4748     @GetMapping(path = "/{codProject}", produces =
             MediaType.APPLICATION_JSON_UTF8_VALUE)
4749     public List<EvaluateDTO> get(@PathVariable final Long
             codProject){
4750         return evaluateService.get(codProject);
4751     }
4752 }
4753
4754
4755 src/main/java/br/ufsc/cultivar/resource/evaluate/TechnologyResource.java
4756
4757 package br.ufsc.cultivar.resource.evaluate;
4758
4759 import br.ufsc.cultivar.exception.ServiceException;
4760 import br.ufsc.cultivar.model.evaluate.Technology;
4761 import br.ufsc.cultivar.service.evaluate.TechnologyService;
4762 import lombok.AccessLevel;
4763 import lombok.AllArgsConstructor;
4764 import lombok.experimental.FieldDefaults;
4765 import org.springframework.beans.factory.annotation.Autowired;
4766 import org.springframework.http.HttpStatus;
4767 import org.springframework.http.MediaType;
4768 import org.springframework.web.bind.annotation.*;
4769
4770 import java.util.List;
4771
4772 @RestController
4773 @RequestMapping(path = "/technology")
4774 @AllArgsConstructor(onConstructor = @__(@Autowired))
4775 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4776 public class TechnologyResource {
4777     TechnologyService service;
4778
4779     @PostMapping(consumes =
             MediaType.APPLICATION_JSON_UTF8_VALUE)
4780     @ResponseStatus(code = HttpStatus.CREATED)
4781     public void create(@RequestBody final Technology technology)
             throws ServiceException {
4782         service.create(technology);
4783     }
4784
4785     @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
4786     public List<Technology> get() throws ServiceException{
```

```
4787            return service.get();
4788        }
4789
4790        @DeleteMapping(path = "/{codTechnology}", produces =
                MediaType.APPLICATION_JSON_UTF8_VALUE)
4791        public Technology delete(@PathVariable final Long
                codTechnology) throws ServiceException{
4792            return service.delete(codTechnology);
4793        }
4794    }
4795
4796
4797    src/main/java/br/ufsc/cultivar/resource/AttachmentResource.java
4798
4799    package br.ufsc.cultivar.resource;
4800
4801    import br.ufsc.cultivar.dto.PaginateList;
4802    import br.ufsc.cultivar.exception.ServiceException;
4803    import br.ufsc.cultivar.model.Attachment;
4804    import br.ufsc.cultivar.model.Status;
4805    import br.ufsc.cultivar.service.AttachmentService;
4806    import lombok.AccessLevel;
4807    import lombok.AllArgsConstructor;
4808    import lombok.experimental.FieldDefaults;
4809    import org.springframework.beans.factory.annotation.Autowired;
4810    import org.springframework.core.io.Resource;
4811    import org.springframework.http.HttpStatus;
4812    import org.springframework.http.MediaType;
4813    import org.springframework.lang.Nullable;
4814    import org.springframework.web.bind.annotation.*;
4815    import org.springframework.web.multipart.MultipartFile;
4816
4817    import java.util.List;
4818
4819    @RestController
4820    @RequestMapping(path = "/attachment")
4821    @AllArgsConstructor(onConstructor = @__(@Autowired))
4822    @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4823    public class AttachmentResource {
4824
4825        AttachmentService service;
4826
4827        @PostMapping(consumes = MediaType.MULTIPART_FORM_DATA_VALUE)
4828        @ResponseStatus(code = HttpStatus.NO_CONTENT)
```

```
4829        public void create(@RequestPart final Attachment attachment,
                @RequestPart(required=false) final MultipartFile file)
                throws ServiceException {
4830            service.create(attachment, file);
4831        }
4832
4833        @GetMapping(path = "/{status}", produces =
                MediaType.APPLICATION_JSON_UTF8_VALUE)
4834        public List<Attachment> get(@PathVariable final Status
                status) throws ServiceException{
4835            return service.get(status);
4836        }
4837
4838        @GetMapping(path = "/{codAttachment}/download", produces =
                MediaType.APPLICATION_PDF_VALUE)
4839        public Resource get(@PathVariable final Long codAttachment)
                throws ServiceException{
4840            return service.getAsFile(codAttachment);
4841        }
4842
4843        @GetMapping(path = "/page/{page}", produces =
                MediaType.APPLICATION_JSON_UTF8_VALUE)
4844        public PaginateList get(@RequestParam(required = false)
                final String filter, @PathVariable final Long page)
                throws ServiceException{
4845            return service.get(filter, page);
4846        }
4847
4848        @DeleteMapping(path = "/{codAttachment}", produces =
                MediaType.APPLICATION_JSON_UTF8_VALUE)
4849        public Attachment delete(@PathVariable final Long
                codAttachment) throws ServiceException{
4850            return service.delete(codAttachment);
4851        }
4852
4853        @PutMapping(path = "/{codAttachment}", consumes =
                MediaType.APPLICATION_JSON_UTF8_VALUE,
4854                   produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
4855        @ResponseStatus(code = HttpStatus.NO_CONTENT)
4856        public void update(@RequestBody final Attachment attachment,
                @PathVariable final Long codAttachment)throws
                ServiceException{
4857            service.update(attachment, codAttachment);
4858        }
4859    }
```

src/main/java/br/ufsc/cultivar/resource/CompanyResource.java

```java
package br.ufsc.cultivar.resource;

import br.ufsc.cultivar.dto.PaginateList;
import br.ufsc.cultivar.exception.ServiceException;
import br.ufsc.cultivar.model.Company;
import br.ufsc.cultivar.service.CompanyService;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.web.bind.annotation.*;

import java.util.List;
import java.util.Map;

@RestController
@RequestMapping(path = "/company")
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class CompanyResource {

    CompanyService service;

    @PostMapping(consumes =
        MediaType.APPLICATION_JSON_UTF8_VALUE)
    @ResponseStatus(code = HttpStatus.NO_CONTENT)
    public void create(@RequestBody final Company company)
        throws ServiceException {
        service.create(company);
    }

    @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
    public List get(@RequestParam(required=false) String filter)
        throws ServiceException{
        return service.get(filter, null).getData();
    }

    @GetMapping(path = "/page/{page}", produces =
        MediaType.APPLICATION_JSON_UTF8_VALUE)
```

```
4901     public PaginateList get(@RequestParam(required=false) final
             String filter,
4902                             @PathVariable final Long page) throws
                                     ServiceException{
4903         return service.get(filter, page);
4904     }
4905
4906     @GetMapping(path = "/{cnpj}", produces =
             MediaType.APPLICATION_JSON_UTF8_VALUE)
4907     public Company getOne(@PathVariable final String cnpj)
             throws ServiceException{
4908         return service.get(cnpj);
4909     }
4910
4911     @DeleteMapping(path = "/{cnpj}", produces =
             MediaType.APPLICATION_JSON_UTF8_VALUE)
4912     public Company delete(@PathVariable final String cnpj)
             throws ServiceException{
4913         return service.delete(cnpj);
4914     }
4915
4916     @PutMapping(path = "/{cnpj}", consumes =
             MediaType.APPLICATION_JSON_UTF8_VALUE,
4917                produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
4918     @ResponseStatus(code = HttpStatus.NO_CONTENT)
4919     public void update(@RequestBody final Company company,
             @PathVariable final String cnpj)throws ServiceException{
4920         service.update(company, cnpj);
4921     }
4922 }
4923
4924
4925 src/main/java/br/ufsc/cultivar/resource/QuestionResource.java
4926
4927 package br.ufsc.cultivar.resource;
4928
4929 import br.ufsc.cultivar.exception.ServiceException;
4930 import br.ufsc.cultivar.model.Question;
4931 import br.ufsc.cultivar.model.Role;
4932 import br.ufsc.cultivar.service.QuestionService;
4933 import lombok.AccessLevel;
4934 import lombok.AllArgsConstructor;
4935 import lombok.experimental.FieldDefaults;
4936 import org.springframework.beans.factory.annotation.Autowired;
4937 import org.springframework.http.HttpStatus;
```

```
4938 import org.springframework.http.MediaType;
4939 import org.springframework.web.bind.annotation.*;
4940
4941 import java.util.List;
4942 import java.util.Map;
4943
4944 @RestController
4945 @RequestMapping(path = "/question")
4946 @AllArgsConstructor(onConstructor = @__(@Autowired))
4947 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
4948 public class QuestionResource {
4949
4950     QuestionService service;
4951
4952     @PostMapping(consumes =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
4953     @ResponseStatus(code = HttpStatus.NO_CONTENT)
4954     public void create(@RequestBody final Question question)
          throws ServiceException {
4955         service.create(question);
4956     }
4957
4958     @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
4959     public List<Question> get() throws ServiceException{
4960         return service.get();
4961     }
4962
4963     @GetMapping(path = "/{responds}", produces =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
4964     public List<Question> get(@PathVariable final Role responds)
          throws ServiceException{
4965         return service.get(responds);
4966     }
4967
4968     @DeleteMapping(path = "/{codQuestion}", produces =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
4969     public Question delete(@PathVariable final Long codQuestion)
          throws ServiceException{
4970         return service.delete(codQuestion);
4971     }
4972
4973     @PutMapping(path = "/{codQuestion}", consumes =
          MediaType.APPLICATION_JSON_UTF8_VALUE,
4974             produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
4975     @ResponseStatus(code = HttpStatus.NO_CONTENT)
```

```
4976      public void update(@RequestBody final Question question,
              @PathVariable final Long codQuestion)throws
              ServiceException{
4977          service.update(question, codQuestion);
4978      }
4979  }
4980
4981
4982  src/main/java/br/ufsc/cultivar/resource/ProjectResource.java
4983
4984  package br.ufsc.cultivar.resource;
4985
4986  import br.ufsc.cultivar.dto.PaginateList;
4987  import br.ufsc.cultivar.exception.ServiceException;
4988  import br.ufsc.cultivar.model.Project;
4989  import br.ufsc.cultivar.service.ProjectService;
4990  import lombok.AccessLevel;
4991  import lombok.AllArgsConstructor;
4992  import lombok.experimental.FieldDefaults;
4993  import org.springframework.beans.factory.annotation.Autowired;
4994  import org.springframework.http.HttpStatus;
4995  import org.springframework.http.MediaType;
4996  import org.springframework.web.bind.annotation.*;
4997
4998  import javax.validation.Valid;
4999  import java.util.List;
5000
5001  @RestController
5002  @RequestMapping(path = "/project")
5003  @AllArgsConstructor(onConstructor = @__(@Autowired))
5004  @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
5005  public class ProjectResource {
5006
5007      ProjectService projectService;
5008
5009      @PostMapping(consumes =
              MediaType.APPLICATION_JSON_UTF8_VALUE)
5010      @ResponseStatus(HttpStatus.CREATED)
5011      public void create(@Valid @RequestBody final Project
          project){
5012          projectService.create(project);
5013      }
5014
5015      @GetMapping(path = "/page/{page}", produces =
              MediaType.APPLICATION_JSON_UTF8_VALUE)
```

```
5016      public PaginateList get(@RequestParam(required = false)
              final String filter, @PathVariable final Long page)
              throws ServiceException {
5017          return projectService.get(filter, page);
5018      }
5019
5020      @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5021      public List get(){
5022          return projectService.get(null, null).getData();
5023      }
5024
5025      @GetMapping(path = "/{codProject}", produces =
              MediaType.APPLICATION_JSON_UTF8_VALUE)
5026      public Project get(@PathVariable final Long codProject){
5027          return projectService.get(codProject);
5028      }
5029
5030      @DeleteMapping(path = "/{codProject}", produces =
              MediaType.APPLICATION_JSON_UTF8_VALUE)
5031      public Project delete(@PathVariable final Long codProject){
5032          return projectService.delete(codProject);
5033      }
5034  }
5035
5036
5037  src/main/java/br/ufsc/cultivar/resource/EventResource.java
5038
5039  package br.ufsc.cultivar.resource;
5040
5041  import br.ufsc.cultivar.exception.ServiceException;
5042  import br.ufsc.cultivar.model.Event;
5043  import br.ufsc.cultivar.model.Rating;
5044  import br.ufsc.cultivar.service.EventService;
5045  import br.ufsc.cultivar.service.RatingService;
5046  import lombok.AccessLevel;
5047  import lombok.AllArgsConstructor;
5048  import lombok.experimental.FieldDefaults;
5049  import org.springframework.beans.factory.annotation.Autowired;
5050  import org.springframework.http.HttpStatus;
5051  import org.springframework.http.MediaType;
5052  import org.springframework.web.bind.annotation.*;
5053  import org.springframework.web.multipart.MultipartFile;
5054
5055  import java.util.List;
5056  import java.util.Map;
```

```
5057 import java.util.Set;
5058
5059 @RestController
5060 @RequestMapping(path = "/event")
5061 @AllArgsConstructor(onConstructor = @__(@Autowired))
5062 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
5063 public class EventResource {
5064
5065     EventService eventService;
5066     RatingService ratingService;
5067
5068     @PostMapping(consumes =
         MediaType.APPLICATION_JSON_UTF8_VALUE)
5069     @ResponseStatus(code = HttpStatus.NO_CONTENT)
5070     public void create(@RequestBody final Event event) throws
         ServiceException {
5071         eventService.create(event);
5072     }
5073
5074     @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5075     public Set<Event> get(
5076             @RequestParam(value = "cod_cpf", required = false)
                 final List<String> filterVolunteer,
5077             @RequestParam(value = "cod_school", required = false)
                 final List<Long> filterSchool,
5078             @RequestParam(value = "cod_project", required =
                 false) final Long filterProject) throws
                 ServiceException{
5079         return eventService.get(filterVolunteer, filterSchool,
             filterProject);
5080     }
5081
5082     @GetMapping(path = "/{codEvent}", produces =
         MediaType.APPLICATION_JSON_UTF8_VALUE)
5083     public Event get(@PathVariable final Long codEvent) throws
         ServiceException{
5084         return eventService.get(codEvent);
5085     }
5086
5087     @DeleteMapping(path = "/{codEvent}", produces =
         MediaType.APPLICATION_JSON_UTF8_VALUE)
5088     public Event delete(@PathVariable final Long codEvent)
         throws ServiceException{
5089         return eventService.delete(codEvent);
5090     }
```

```
5091
5092        @PutMapping(path = "/{codEvent}", consumes =
                MediaType.APPLICATION_JSON_UTF8_VALUE,
5093                produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5094        @ResponseStatus(code = HttpStatus.NO_CONTENT)
5095        public void update(@RequestBody final Event event,
                @PathVariable final Long codEvent)throws
                ServiceException{
5096                eventService.update(event, codEvent);
5097        }
5098
5099        @PostMapping(path="/{codEvent}/upload", consumes =
                MediaType.APPLICATION_PDF_VALUE,
5100                produces = MediaType.TEXT_PLAIN_VALUE)
5101        public String upload(@RequestPart final MultipartFile file,
                @PathVariable final Long codEvent) throws
                ServiceException {
5102                return eventService.upload(file, codEvent);
5103        }
5104
5105        @PutMapping(path = "/{codEvent}/evaluate", consumes =
                MediaType.APPLICATION_JSON_UTF8_VALUE)
5106        @ResponseStatus(code = HttpStatus.NO_CONTENT)
5107        public void evaluate(@RequestBody final Rating rating,
                @PathVariable Long codEvent){
5108                ratingService.create(rating, codEvent);
5109        }
5110    }
5111
5112
5113    src/main/java/br/ufsc/cultivar/resource/VolunteerResource.java
5114
5115    package br.ufsc.cultivar.resource;
5116
5117    import br.ufsc.cultivar.dto.PaginateList;
5118    import br.ufsc.cultivar.dto.ParticipationDTO;
5119    import br.ufsc.cultivar.exception.ServiceException;
5120    import br.ufsc.cultivar.model.*;
5121    import br.ufsc.cultivar.service.DispatchService;
5122    import br.ufsc.cultivar.service.EventService;
5123    import br.ufsc.cultivar.service.RatingService;
5124    import br.ufsc.cultivar.service.VolunteerService;
5125    import lombok.AccessLevel;
5126    import lombok.AllArgsConstructor;
5127    import lombok.experimental.FieldDefaults;
```

```
5128 import org.springframework.beans.factory.annotation.Autowired;
5129 import org.springframework.http.HttpStatus;
5130 import org.springframework.http.MediaType;
5131 import org.springframework.web.bind.annotation.*;
5132 import org.springframework.web.multipart.MultipartFile;
5133
5134 import java.util.List;
5135 import java.util.Set;
5136
5137 @RestController
5138 @RequestMapping(path = "/volunteer")
5139 @AllArgsConstructor(onConstructor = @__(@Autowired))
5140 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
5141 public class VolunteerResource {
5142
5143     VolunteerService volunteerService;
5144     RatingService ratingService;
5145     DispatchService dispatchService;
5146     EventService eventService;
5147
5148     @PostMapping(consumes =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
5149     @ResponseStatus(code = HttpStatus.NO_CONTENT)
5150     public void create(@RequestBody final Volunteer volunteer)
          throws ServiceException {
5151         volunteerService.create(volunteer);
5152     }
5153
5154     @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5155     public List get(
5156             @RequestParam(value = "cod_cnpj", required = false)
                  final List<String> filterCompany,
5157             @RequestParam(value = "cod_school", required = false)
                  final List<Long> filterSchool,
5158             @RequestParam(required = false) final String filter)
                  throws ServiceException{
5159         return volunteerService.get(filterCompany, filterSchool,
              filter, null).getData();
5160     }
5161
5162     @GetMapping(path = "/page/{page}", produces =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
5163     public PaginateList get(
5164             @RequestParam(value = "cod_cnpj", required = false)
                  final List<String> filterCompany,
```

```
5165            @RequestParam(value = "cod_school", required = false)
                    final List<Long> filterSchool,
5166            @RequestParam(required = false) final String filter,
5167            @PathVariable final Integer page) throws
                    ServiceException{
5168        return volunteerService.get(filterCompany, filterSchool,
                filter, page);
5169    }
5170
5171
5172    @GetMapping(path = "/{cpf}", produces =
            MediaType.APPLICATION_JSON_UTF8_VALUE)
5173    public Volunteer get(@PathVariable final String cpf) throws
            ServiceException{
5174        return volunteerService.get(cpf);
5175    }
5176
5177    @DeleteMapping(path = "/{cpf}", produces =
            MediaType.APPLICATION_JSON_UTF8_VALUE)
5178    public Volunteer delete(@PathVariable final String cpf)
            throws ServiceException{
5179        return volunteerService.delete(cpf);
5180    }
5181
5182    @PutMapping(path = "/{cpf}", consumes =
            MediaType.APPLICATION_JSON_UTF8_VALUE,
5183            produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5184    @ResponseStatus(code = HttpStatus.NO_CONTENT)
5185    public void update(@RequestBody final Volunteer volunteer,
            @PathVariable final String cpf)throws ServiceException{
5186        volunteerService.update(volunteer, cpf);
5187    }
5188
5189    @GetMapping(path = {"/{cpf}/event", "/{cpf}/event/{type}"})
5190    public List<Event> getEvents(@PathVariable final String cpf,
            @PathVariable(required = false) final Long type){
5191        return eventService.eventsByVolunteer(cpf, type);
5192    }
5193
5194    @GetMapping(path = "/{cpf}/attachment/{codAttachment}")
5195    public Dispatch getDispatch(@PathVariable final String cpf,
            @PathVariable Long codAttachment){
5196        return dispatchService.get(cpf, codAttachment);
5197    }
5198
```

```
5199      @PostMapping(path="/{cpf}/upload", consumes =
              MediaType.MULTIPART_FORM_DATA_VALUE,
5200              produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5201      public void upload(@RequestPart final Attachment attachment,
              @RequestPart final MultipartFile file,
5202                          @PathVariable final String cpf) throws
                                ServiceException {
5203          dispatchService.save(attachment, file, cpf);
5204      }
5205
5206      @PutMapping(path = "/{cpf}/evaluate", consumes =
              MediaType.APPLICATION_JSON_UTF8_VALUE)
5207      @ResponseStatus(code = HttpStatus.NO_CONTENT)
5208      public void evaluate(@RequestBody final Rating rating,
              @PathVariable final String cpf) throws ServiceException {
5209          ratingService.create(rating, cpf);
5210      }
5211
5212      @GetMapping(path = "/{cpf}/evaluate", consumes =
              MediaType.APPLICATION_JSON_UTF8_VALUE)
5213      @ResponseStatus(code = HttpStatus.NO_CONTENT)
5214      public List<Event> evaluate(@PathVariable final String cpf)
              throws ServiceException {
5215          return eventService.getEventsToEvaluateByVolunteer(cpf);
5216      }
5217
5218      @GetMapping(path = "/{cpf}/participations", produces =
              MediaType.APPLICATION_JSON_UTF8_VALUE)
5219      public Set<ParticipationDTO>
              getVolunteerLocals(@PathVariable final String cpf){
5220          return eventService.getVolunteerLocals(cpf);
5221      }
5222  }
5223
5224
5225  src/main/java/br/ufsc/cultivar/resource/UserResource.java
5226
5227  package br.ufsc.cultivar.resource;
5228
5229  import br.ufsc.cultivar.exception.ServiceException;
5230  import br.ufsc.cultivar.model.User;
5231  import br.ufsc.cultivar.service.CompanyService;
5232  import br.ufsc.cultivar.service.UserService;
5233  import lombok.AccessLevel;
5234  import lombok.AllArgsConstructor;
```

```
5235 import lombok.experimental.FieldDefaults;
5236 import lombok.val;
5237 import org.springframework.beans.factory.annotation.Autowired;
5238 import org.springframework.http.HttpStatus;
5239 import org.springframework.http.MediaType;
5240 import org.springframework.web.bind.annotation.*;
5241
5242 import java.util.List;
5243 import java.util.Map;
5244
5245 @RestController
5246 @RequestMapping(path = "/user")
5247 @AllArgsConstructor(onConstructor = @__(@Autowired))
5248 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
5249 public class UserResource {
5250
5251     UserService service;
5252
5253     @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5254     public List<User> get(@RequestParam final Map<String,
          Object> filter) throws ServiceException{
5255         return service.get(filter);
5256     }
5257
5258     @GetMapping(path = "/{cpf}", produces =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
5259     public User get(@PathVariable final String cpf) throws
          ServiceException{
5260         return service.get(cpf);
5261     }
5262
5263     @DeleteMapping(path = "/{cpf}", produces =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
5264     public User delete(@PathVariable final String cpf) throws
          ServiceException{
5265         return service.delete(cpf);
5266     }
5267
5268     @PutMapping(path = "/{cpf}", consumes =
          MediaType.APPLICATION_JSON_UTF8_VALUE,
5269             produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5270     @ResponseStatus(code = HttpStatus.NO_CONTENT)
5271     public void update(@RequestBody final User user,
          @PathVariable final String cpf)throws ServiceException{
5272         service.update(user, cpf);
```

```
5273     }
5274 }
5275
5276
5277 src/main/java/br/ufsc/cultivar/resource/TypeEventResource.java
5278
5279 package br.ufsc.cultivar.resource;
5280
5281 import br.ufsc.cultivar.dto.PaginateList;
5282 import br.ufsc.cultivar.exception.ServiceException;
5283 import br.ufsc.cultivar.model.Training;
5284 import br.ufsc.cultivar.model.TypeEvent;
5285 import br.ufsc.cultivar.service.TypeEventService;
5286 import lombok.AccessLevel;
5287 import lombok.AllArgsConstructor;
5288 import lombok.experimental.FieldDefaults;
5289 import org.springframework.beans.factory.annotation.Autowired;
5290 import org.springframework.http.HttpStatus;
5291 import org.springframework.http.MediaType;
5292 import org.springframework.web.bind.annotation.*;
5293
5294 import java.util.List;
5295
5296 @RestController
5297 @RequestMapping(path = "/typeEvent")
5298 @AllArgsConstructor(onConstructor = @__(@Autowired))
5299 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
5300 public class TypeEventResource {
5301
5302     TypeEventService typeEventService;
5303
5304     @PostMapping(consumes =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
5305     @ResponseStatus(code = HttpStatus.NO_CONTENT)
5306     public void create(@RequestBody final TypeEvent typeEvent)
          throws ServiceException {
5307         typeEventService.create(typeEvent);
5308     }
5309
5310     @GetMapping(path = "/page/{page}", produces =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
5311     public PaginateList get(@RequestParam(required = false)
          final String filter, @PathVariable final Long page)
          throws ServiceException {
5312         return typeEventService.get(filter, page);
```

```
5313     }
5314
5315     @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5316     public List get() throws ServiceException{
5317         return typeEventService.get(null, null).getData();
5318     }
5319
5320     @GetMapping(path = "/{tpEvent}", produces =
             MediaType.APPLICATION_JSON_UTF8_VALUE)
5321     public TypeEvent get(@PathVariable final Long tpEvent)
             throws ServiceException{
5322         return typeEventService.get(tpEvent);
5323     }
5324
5325     @DeleteMapping(path = "/{tpEvent}", produces =
             MediaType.APPLICATION_JSON_UTF8_VALUE)
5326     public TypeEvent delete(@PathVariable final Long tpEvent)
             throws ServiceException{
5327         return typeEventService.delete(tpEvent);
5328     }
5329
5330     @GetMapping(path = "/{tpEvent}/trainings", produces =
             MediaType.APPLICATION_JSON_UTF8_VALUE)
5331     public List<Training> getTrainings(@PathVariable final Long
             tpEvent) throws ServiceException{
5332         return typeEventService.getTrainings(tpEvent);
5333     }
5334 }
5335
5336
5337 src/main/java/br/ufsc/cultivar/resource/SchoolResource.java
5338
5339 package br.ufsc.cultivar.resource;
5340
5341 import br.ufsc.cultivar.dto.PaginateList;
5342 import br.ufsc.cultivar.exception.ServiceException;
5343 import br.ufsc.cultivar.model.Event;
5344 import br.ufsc.cultivar.model.School;
5345 import br.ufsc.cultivar.model.TypeEvent;
5346 import br.ufsc.cultivar.service.EventService;
5347 import br.ufsc.cultivar.service.SchoolService;
5348 import lombok.AccessLevel;
5349 import lombok.AllArgsConstructor;
5350 import lombok.experimental.FieldDefaults;
5351 import org.springframework.beans.factory.annotation.Autowired;
```

```
5352 import org.springframework.http.HttpStatus;
5353 import org.springframework.http.MediaType;
5354 import org.springframework.web.bind.annotation.*;
5355
5356 import java.util.List;
5357 import java.util.Map;
5358
5359 @RestController
5360 @RequestMapping(path = "/school")
5361 @AllArgsConstructor(onConstructor = @__(@Autowired))
5362 @FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
5363 public class SchoolResource {
5364     SchoolService service;
5365     EventService eventService;
5366
5367     @PostMapping(consumes =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
5368     @ResponseStatus(code = HttpStatus.NO_CONTENT)
5369     public void create(@RequestBody final School school) throws
          ServiceException {
5370         service.create(school);
5371     }
5372
5373     @GetMapping(produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5374     public List get(@RequestParam(required=false) String filter)
          throws ServiceException{
5375         return service.get(filter, null).getData();
5376     }
5377
5378     @GetMapping(path = "/page/{page}", produces =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
5379     public PaginateList get(@RequestParam(required=false) String
          filter, @PathVariable final Long page) throws
          ServiceException{
5380         return service.get(filter, page);
5381     }
5382
5383     @GetMapping(path = "/{codSchool}", produces =
          MediaType.APPLICATION_JSON_UTF8_VALUE)
5384     public School get(@PathVariable final Long codSchool) throws
          ServiceException{
5385         return service.get(codSchool);
5386     }
5387
```

```
5388      @GetMapping(path = {"/{codSchool}/event",
              "/{codSchool}/event/{type}"}, produces =
              MediaType.APPLICATION_JSON_UTF8_VALUE)
5389      public List<Event> getEvents(@PathVariable final Long
              codSchool, @PathVariable(required = false) final Long
              type){
5390          return eventService.eventsBySchool(codSchool, type);
5391      }
5392
5393      @DeleteMapping(path = "/{codSchool}", produces =
              MediaType.APPLICATION_JSON_UTF8_VALUE)
5394      public School delete(@PathVariable final Long codSchool)
              throws ServiceException{
5395          return service.delete(codSchool);
5396      }
5397
5398      @PutMapping(path = "/{codSchool}", consumes =
              MediaType.APPLICATION_JSON_UTF8_VALUE,
5399              produces = MediaType.APPLICATION_JSON_UTF8_VALUE)
5400      @ResponseStatus(code = HttpStatus.NO_CONTENT)
5401      public void update(@RequestBody final School school,
              @PathVariable final Long codSchool)throws
              ServiceException{
5402          service.update(school, codSchool);
5403      }
5404  }
5405
5406
5407  src/main/java/br/ufsc/cultivar/exception/ServiceException.java
5408
5409  package br.ufsc.cultivar.exception;
5410
5411  import lombok.Getter;
5412
5413  public class ServiceException extends Exception {
5414
5415      @Getter
5416      private final Type type;
5417
5418      public ServiceException(String message, Throwable cause,
              Type type) {
5419          super(message, cause);
5420          this.type = type;
5421      }
5422
```

```
    public ServiceException(String message, Throwable cause) {
        super(message, cause);
        this.type = Type.DEFAULT;
    }
}


src/main/java/br/ufsc/cultivar/exception/Type.java

package br.ufsc.cultivar.exception;

public enum Type {
    NOT_FOUND,
    INVALID,
    FILE,
    DEFAULT
}


src/main/java/br/ufsc/cultivar/exception/UploadException.java

package br.ufsc.cultivar.exception;

public class UploadException extends Exception {
    public UploadException(String message, Throwable cause) {
        super(message, cause);
    }
}


src/main/java/br/ufsc/cultivar/email/EmailClient.java

package br.ufsc.cultivar.email;

import br.ufsc.cultivar.config.EmailConfig;
import lombok.AccessLevel;
import lombok.AllArgsConstructor;
import lombok.experimental.FieldDefaults;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.mail.javamail.JavaMailSender;
import org.springframework.mail.javamail.MimeMessageHelper;
import org.springframework.mail.javamail.MimeMessagePreparator;
import org.springframework.scheduling.annotation.Async;
import org.springframework.stereotype.Component;

```

```
import java.util.Optional;
import java.util.concurrent.CompletableFuture;

@Component
@AllArgsConstructor(onConstructor = @__(@Autowired))
@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)
public class EmailClient {
    EmailConfig emailConfig;
    JavaMailSender mailSender;

    public void sendEmailSync(String to, String subject, String message){
        MimeMessagePreparator messagePreparator = mimeMessage -> {
            MimeMessageHelper messageHelper = new MimeMessageHelper(mimeMessage);
            messageHelper.setFrom(emailConfig.getFrom());
            messageHelper.setTo(to);
            messageHelper.setSubject(subject);
            messageHelper.setText(message, true);
        };
        mailSender.send(messagePreparator);

    }

    @Async
    public CompletableFuture<Void> sendEmailAsync(String to, String subject, String message){
        sendEmailSync(to, subject, message);
        return CompletableFuture.completedFuture(null);
    }
}


src/main/java/br/ufsc/cultivar/Application.java

package br.ufsc.cultivar;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.scheduling.annotation.EnableScheduling;
import org.springframework.web.bind.annotation.RequestMapping;

```

```
@SpringBootApplication
@EnableScheduling
public class Application {
    public static void main(String... args) {
        SpringApplication.run(Application.class, args);
    }
}
```

D.2 FRONTEND

```
1
2
3  webapp/src/utils/formatter.js
4
5  export default {
6      cnpj: s => s.replace(/^(\d{2})(\d{3})(\d{3})(\d{4})(\d{2})/,
           "$1.$2.$3/$4-$5"),
7      cpf: s => s.replace(/^(\d{3})(\d{3})(\d{3})(\d{2})/,
           "$1.$2.$3-$4"),
8      phone: s => s.length === 10 ?
           s.replace(/^(\d{2})(\d{4})(\d{4})/, "($1)$2-$3") :
           s.replace(/^(\d{2})(\d{5})(\d{4})/, "($1)$2-$3"),
9  }
10
11 webapp/src/utils/http.js
12
13 import axios from 'axios';
14 import { get } from './storage';
15
16 axios.defaults.baseURL = process.env.REACT_APP_ENDPOINT;
17
18 export const setHeaders = (headers) =>
       axios.defaults.headers.common = {
19     'X-Requested-With': 'XMLHttpRequest',
20     'Authorization': get('token'),
21     ...headers
22 }
23
24 const empty = () => {};
25
26 const setCallback = (request, success, failed) =>
       request.then(success ? success : empty).catch(failed ?
       failed : empty);
27
28 export const getRequest = (url, success, failed) =>
       setCallback(axios.get(url), success, failed);
29 export const postRequest = (url, data, success, failed) =>
       setCallback(axios.post(url, data), success, failed);
30 export const putRequest = (url, data, success, failed) =>
       setCallback(axios.put(url, data), success, failed);
```

```
export const deleteRequest = (url, success, failed) =>
    setCallback(axios.delete(url), success, failed);

setHeaders();

webapp/src/utils/storage.js

export const saveObject = (key, obj) =>
    localStorage.setItem(key, JSON.stringify(obj));

export const getAsObject = key =>
    JSON.parse(localStorage.getItem(key));

export const save = (key, data) => localStorage.setItem(key,
    data);

export const get = key => localStorage.getItem(key);

webapp/src/components/pagination/index.js

import Pagination from './Pagination.jsx';
export default Pagination;

webapp/src/components/pagination/Pagination.jsx

import React from 'react';
import { Row, Col, Pagination, PaginationItem, PaginationLink }
    from 'reactstrap';

export default class Example extends React.Component {
   constructor(){
       super();
       this.state = {
           page: 0
       };
       this.increment = this.increment.bind(this);
       this.decrement = this.decrement.bind(this);
   }

   increment(){
       const page = this.state.page + 1;
       this.props.onChangePage(page);
       this.setState({page});
   }

```

```
71      decrement() {
72          const page = this.state.page - 1;
73          this.props.onChangePage(page);
74          this.setState({ page });
75      }
76
77      render() {
78          const { pages, count } = this.props;
79          const { page } = this.state;
80          return (
81              <Row>
82                  <Col>
83                      <Pagination aria-label="Page navigation
                            example">
84                          <PaginationItem disabled={page === 0}>
85                              <PaginationLink previous href="#"
                                    onClick={this.decrement}/>
86                          </PaginationItem>
87                          <PaginationItem disabled={page ===
                                parseInt(pages)}>
88                              <PaginationLink next href="#"
                                    onClick={this.increment}/>
89                          </PaginationItem>
90                      </Pagination>
91                  </Col>
92                  <Col>
93                      <strong className="float-right"
                            style={{fontSize: '.8rem'}}>
94                          Pgina  {page + 1} de {parseInt(pages) + 1}
95                          <br/>
96                          {count} Registros
97                      </strong>
98                  </Col>
99              </Row>
100         );
101     }
102 }
103
104 webapp/src/components/footer/Footer.jsx
105
106 import React from 'react';
107 import { Container, Col, Row } from 'reactstrap';
108 import logo from './logo.svg'
109 import './footer.css';
110
```

```
export default () => (
    <footer className="footer">
        <Container>
            <Row>
                <Col>
                    <span className="text-muted">Powered by <img
                        className="react-logo" src={logo}
                        alt="React"/></span>
                </Col>
            </Row>
        </Container>
    </footer>
);
```

webapp/src/components/footer/footer.sass

```
$footer-height: 60px

.footer
  position: fixed
  bottom: 0
  width: 100%
  height: $footer-height
  line-height: 60px /* Vertically center the text there */
  background-color: #f5f5f5
  .react-logo
    animation: react-logo-spin infinite 20s linear
    height: $footer-height - 15px

@keyframes react-logo-spin
  from
    transform: rotate(0deg)
  to
    transform: rotate(360deg)
```

webapp/src/components/footer/logo.svg

```
<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 841.9
    595.3">
    <g fill="#61DAFB">
        <path d="M666.3 296.5c0-32.5-40.7-63.3-103.1-82.4
            14.4-63.6
            8-114.2-20.2-130.4-6.5-3.8-14.1-5.6-22.4-5.6v22.3c4.6
            0 8.3.9 11.4 2.6 13.6 7.8 19.5 37.5 14.9 75.7-1.1
```

9.4-2.9 19.3-5.1
29.4-19.6-4.8-41-8.5-63.5-10.9-13.5-18.5-27.5-35.3-41.6-50
32.6-30.3 63.2-46.9 84-46.9V78c-27.5 0-63.5
19.6-99.9
53.6-36.4-33.8-72.4-53.2-99.9-53.2v22.3c20.7 0 51.4
16.5 84 46.6-14 14.7-28 31.4-41.3 49.9-22.6 2.4-44
6.1-63.6 11-2.3-10-4-19.7-5.2-29-4.7-38.2 1.1-67.9
14.6-75.8 3-1.8 6.9-2.6 11.5-2.6V78.5c-8.4 0-16
1.8-22.6 5.6-28.1 16.2-34.4 66.7-19.9 130.1-62.2
19.2-102.7 49.9-102.7 82.3 0 32.5 40.7 63.3 103.1
82.4-14.4 63.6-8 114.2 20.2 130.4 6.5 3.8 14.1 5.6
22.5 5.6 27.5 0 63.5-19.6 99.9-53.6 36.4 33.8 72.4
53.2 99.9 53.2 8.4 0 16-1.8 22.6-5.6 28.1-16.2
34.4-66.7 19.9-130.1 62-19.1 102.5-49.9
102.5-82.3zm-130.2-66.7c-3.7 12.9-8.3 26.2-13.5
39.5-4.1-8-8.4-16-13.1-24-4.6-8-9.5-15.8-14.4-23.4
14.2 2.1 27.9 4.7 41 7.9zm-45.8 106.5c-7.8 13.5-15.8
26.3-24.1 38.2-14.9 1.3-30 2-45.2 2-15.1
0-30.2-.7-45-1.9-8.3-11.9-16.4-24.6-24.2-38-7.6-13.1-14.5-26.4-
6.2-13.4 13.2-26.8 20.7-39.9 7.8-13.5 15.8-26.3
24.1-38.2 14.9-1.3 30-2 45.2-2 15.1 0 30.2.7 45 1.9
8.3 11.9 16.4 24.6 24.2 38 7.6 13.1 14.5 26.4 20.8
39.8-6.3 13.4-13.2 26.8-20.7 39.9zm32.3-13c5.4 13.4
10 26.8 13.8 39.8-13.1 3.2-26.9 5.9-41.2 8 4.9-7.7
9.8-15.6 14.4-23.7 4.6-8 8.9-16.1 13-24.1zM421.2
430c-9.3-9.6-18.6-20.3-27.8-32 9 .4 18.2.7 27.5.7
9.4 0 18.7-.2 27.8-.7-9 11.7-18.3 22.4-27.5
32zm-74.4-58.9c-14.2-2.1-27.9-4.7-41-7.9 3.7-12.9
8.3-26.2 13.5-39.5 4.1 8 8.4 16 13.1 24 4.7 8 9.5
15.8 14.4 23.4zM420.7 163c9.3 9.6 18.6 20.3 27.8
32-9-.4-18.2-.7-27.5-.7-9.4 0-18.7.2-27.8.7 9-11.7
18.3-22.4 27.5-32zm-74 58.9c-4.9 7.7-9.8 15.6-14.4
23.7-4.6 8-8.9 16-13 24-5.4-13.4-10-26.8-13.8-39.8
13.1-3.1 26.9-5.8 41.2-7.9zm-90.5
125.2c-35.4-15.1-58.3-34.9-58.3-50.6 0-15.7
22.9-35.6 58.3-50.6 8.6-3.7 18-7 27.7-10.1 5.7 19.6
13.2 40 22.5 60.9-9.2 20.8-16.6 41.1-22.2
60.6-9.9-3.1-19.3-6.5-28-10.2zM310
490c-13.6-7.8-19.5-37.5-14.9-75.7 1.1-9.4 2.9-19.3
5.1-29.4 19.6 4.8 41 8.5 63.5 10.9 13.5 18.5 27.5
35.3 41.6 50-32.6 30.3-63.2 46.9-84
46.9-4.5-.1-8.3-1-11.3-2.7zm237.2-76.2c4.7 38.2-1.1
67.9-14.6 75.8-3 1.8-6.9 2.6-11.5 2.6-20.7
0-51.4-16.5-84-46.6 14-14.7 28-31.4 41.3-49.9
22.6-2.4 44-6.1 63.6-11 2.3 10.1 4.1 19.8 5.2

```
            29.1zm38.5-66.7c-8.6 3.7-18 7-27.7
            10.1-5.7-19.6-13.2-40-22.5-60.9 9.2-20.8 16.6-41.1
            22.2-60.6 9.9 3.1 19.3 6.5 28.1 10.2 35.4 15.1 58.3
            34.9 58.3 50.6-.1 15.7-23 35.6-58.4 50.6zM320.8
            78.4z"/>
        <circle cx="420.9" cy="296.5" r="45.7"/>
        <path d="M520.5 78.1z"/>
    </g>
</svg>


webapp/src/components/footer/footer.css

.footer {
  position: fixed;
  bottom: 0;
  width: 100%;
  height: 60px;
  line-height: 60px;
  background-color: #f5f5f5; }
  .footer .react-logo {
    animation: react-logo-spin infinite 20s linear;
    height: 45px; }

@keyframes react-logo-spin {
  from {
    transform: rotate(0deg); }
  to {
    transform: rotate(360deg); } }


webapp/src/components/form/maskInput/index.js

import MaskInput from './Mask'
export default MaskInput;

webapp/src/components/form/maskInput/Mask.jsx

import React, { Component } from 'react';
import Input from '../input';
import InputMask from 'react-input-mask';
import PropTypes from 'prop-types';
import requiredIf from 'react-required-if';

export default class extends Component {
```

```
190     static propTypes = {
191         id: PropTypes.string.isRequired,
192         mask: PropTypes.string.isRequired,
193         label: PropTypes.string.isRequired,
194         invalidMessage: requiredIf(PropTypes.string, props =>
                props.required),
195         required: PropTypes.bool,
196         placeholder: PropTypes.string,
197         maskChar: PropTypes.string,
198         onChange: PropTypes.func
199     }
200
201     static defaultProps = {
202         maskChar: '_',
203     }
204
205     componentDidMount() {
206         const { id, value } = this.props;
207         document.getElementById(id).value = value;
208     }
209
210     render() {
211         const { id, label, placeholder, invalidMessage,
                onChange, required, mask, maskChar, value, ...rest }
                = this.props
212         return (
213             <InputMask onChange={onChange} mask={mask}
                    maskChar={maskChar} >
214                 {
215                     inputProps =>
216                         (
217                             <Input id={id} {...inputProps}
                                    label={label}
                                    invalidMessage={invalidMessage}
                                    placeholder={placeholder}
                                    required={required} {...rest} />
218                         )
219                 }
220             </InputMask>
221         );
222     }
223 }
224
225 webapp/src/components/form/input/Input.jsx
226
```

```
227 import React, { Component } from 'react';
228 import PropTypes from 'prop-types';
229 import { FormGroup, Input, Label, FormFeedback } from
        'reactstrap';
230 import Required from '../Required.jsx'
231 import requiredIf from 'react-required-if';
232
233 export default class extends Component{
234     static propTypes = {
235         id: PropTypes.string.isRequired,
236         label: PropTypes.string.isRequired,
237         invalidMessage: requiredIf(PropTypes.string, props =>
                props.required),
238         type: PropTypes.string,
239         placeholder: PropTypes.string,
240         required: PropTypes.bool,
241         children: requiredIf(PropTypes.node, props => props.type
                === 'select')
242     }
243
244     static defaultProps = {
245         type: 'text',
246     }
247
248     componentDidMount() {
249         const { id, value } = this.props;
250         document.getElementById(id).value = value;
251     }
252
253     render(){
254         const { id, label, type, placeholder, invalidMessage,
                required, children, ...rest } = this.props
255         return (
256             <FormGroup>
257                 <Label for={id}>{label}{required ? <Required /> :
                        null}</Label>
258                 <Input id={id} type={type}
                        placeholder={placeholder} {...rest}>
259                     {children}
260                 </Input>
261                 <FormFeedback>{invalidMessage}</FormFeedback>
262             </FormGroup>
263         );
264     }
265 }
```

```
webapp/src/components/form/input/index.js

import Input from './Input'
export default Input;

webapp/src/components/form/datePicker/datePicker.css

input[type="date"] {
  background: #fff url(./calendar.png) 97% 50% no-repeat; }
  input[type="date"]::-webkit-inner-spin-button {
    display: none; }
  input[type="date"]::-webkit-calendar-picker-indicator {
    opacity: 0; }


webapp/src/components/form/datePicker/datePicker.sass

input[type="date"]
    background: #fff url(./calendar.png) 97% 50% no-repeat
    &::-webkit-inner-spin-button
        display: none;
    &::-webkit-calendar-picker-indicator
        opacity: 0;


webapp/src/components/form/datePicker/index.js

import DatePicker from './DatePicker';
export default DatePicker;

webapp/src/components/form/datePicker/DatePicker.jsx

import React from 'react';
import PropTypes from 'prop-types';
import requiredIf from 'react-required-if';
import Input from '../input';
import './datePicker.css';

const DatePicker = ({ id, label, placeholder, invalidMessage,
    required, ...rest}) => (
    <Input id={id} type="date" label={label}
        placeholder={placeholder}
        invalidMessage={invalidMessage} required={required}
        {...rest}/>
```

```
);

DatePicker.propTypes = {
    id: PropTypes.string.isRequired,
    label: PropTypes.string.isRequired,
    invalidMessage: requiredIf(PropTypes.string, props =>
        props.required),
    placeholder: PropTypes.string,
    required: PropTypes.bool
};

DatePicker.defaultProps = {
    placeholder: 'DD/MM/AAAA',
};

export default DatePicker;
```

webapp/src/components/form/datePicker/calendar.png

```
 PNG


         IHDR7bKGD           pHYsHHFk > IDAT (   } M
         P0          p5
   j   -   vbKj   '              m6q@E              ;          SPf          =
     ;         mbW2@TpRXkx          ~    1Xu    |8     w6     - HEZW ,~      |:      "(


        fPD       ~          {

     j     %tEXtdate:create2018-08-03T09:53:38-04:00    %tEXtdate:modify2017-10-(
```

webapp/src/components/form/wizard/Navigation.jsx

```
import React from 'react';
import PropTypes from 'prop-types';
import { WithWizard } from "react-albus/lib";
import { Button, Row, Col } from 'reactstrap';

const Navigation = ({ onCancel, validate, onSubmit, submitLabel,
    nextLabel, previousLabel, cancelLabel}) => (
    <WithWizard
        render={({ next, previous, step, steps }) => (
            <Row>
                {
                    steps.indexOf(step) === 0 && (
```

```
348                         <Col>
349                             <Button color="danger"
                                    onClick={onCancel}>
350                                 {cancelLabel}
351                             </Button>
352                         </Col>
353                     )
354                 }
355                 {
356                     steps.indexOf(step) > 0 && (
357                         <Col>
358                             <Button color="secondary"
                                    onClick={previous}>
359                                 {previousLabel}
360                             </Button>
361                         </Col>
362                     )
363                 }
364                 {
365                     steps.indexOf(step) < steps.length - 1 && (
366                         <Col>
367                             <Button id="next" type="button"
                                    color="primary"
                                    className="float-right"
                                    onClick={() => {
368                                 validate && validate(step);
369                                 next();
370                             }}>
371                                 {nextLabel}
372                             </Button>
373                         </Col>
374                     )
375                 }
376                 {
377                     steps.indexOf(step) === steps.length - 1 && (
378                         <Col>
379                             <Button id="next" type="button"
                                    color="primary"
                                    className="float-right"
                                    onClick={onSubmit}>
380                                 {submitLabel}
381                             </Button>
382                         </Col>
383                     )
384                 }
```

```
            </Row>
        )}
    />
);

Navigation.propTypes = {
    onCancel: PropTypes.func.isRequired,
    onSubmit: PropTypes.func.isRequired,
    submitLabel : PropTypes.string.isRequired,
    validate: PropTypes.func,
    nextLabel: PropTypes.string,
    previousLabel: PropTypes.string,
    cancelLabel: PropTypes.string
};

Navigation.defaultProps = {
    validate: () => null,
    nextLabel: 'Avanar',
    previousLabel: 'Voltar',
    cancelLabel: 'Cancelar'
};

export default Navigation;
```

webapp/src/components/form/wizard/index.js

```
import Wizard from './Wizard';
export default Wizard;
```

webapp/src/components/form/wizard/Wizard.jsx

```
import React from 'react';
import PropTypes from 'prop-types';
import { Wizard, Steps, Step } from "react-albus/lib";
import { Form } from 'reactstrap';
import Navigation from './Navigation';

const WizardForm = ({ children, onCancel, validate, onSubmit,
    submitLabel, nextLabel, previousLabel, cancelLabel}) => {
    return (
            <Form>
                <Wizard>
                    <Steps>
                        {
```

```
428                             React.Children.map(children, (child,
                                    index) => (
429                                 <Step id={'step-' + index}>
430                                     {child}
431                                 </Step>
432                             ))
433                         }
434                     </Steps>
435                 <Navigation {...{ onCancel, validate, onSubmit,
                        submitLabel, nextLabel, previousLabel,
                        cancelLabel}}/>
436                 </Wizard>
437             </Form>
438     );
439 }
440
441
442 Navigation.propTypes = {
443     children: PropTypes.element,
444     onCancel: PropTypes.func.isRequired,
445     onSubmit: PropTypes.func.isRequired,
446     submitLabel: PropTypes.string.isRequired,
447     validate: PropTypes.func,
448     nextLabel: PropTypes.string,
449     previousLabel: PropTypes.string,
450     cancelLabel: PropTypes.string
451 };
452
453 Navigation.defaultProps = {
454     validate: () => null,
455     nextLabel: 'Avanar',
456     previousLabel: 'Voltar',
457     cancelLabel: 'Cancelar'
458 };
459
460 export default WizardForm;
461
462 webapp/src/components/form/checkbox/Checkbox.jsx
463
464 import React from 'react';
465 import './checkbox.css';
466
467 export default ({ value, label, disabled, checked, onChange}) =>
        (
468     <label className="control control--checkbox">{label}
```

```
        <input type="checkbox" value={value} disabled={disabled}
            checked={checked} onChange={onChange}/>
        <div className="control__indicator"></div>
    </label>
)

webapp/src/components/form/checkbox/index.js

import Checkbox from './Checkbox.jsx';
export default Checkbox;

webapp/src/components/form/checkbox/checkbox.css

.control {
  display: block;
  position: relative;
  padding-left: 30px;
  margin-bottom: 15px;
  cursor: pointer;
  font-size: 18px; }
  .control input {
    position: absolute;
    z-index: -1;
    opacity: 0; }
    .control input:focus ~ .control__indicator {
      background: #ccc; }
    .control input:checked ~ .control__indicator {
      background: #2aa1c0; }
      .control input:checked ~ .control__indicator:after {
        display: block; }
    .control input:checked:focus ~ .control__indicator {
      background: #0e647d; }
    .control input:disabled ~ .control__indicator {
      background: #e6e6e6;
      opacity: 0.6;
      pointer-events: none; }
  .control:hover input ~ .control__indicator {
    background: #ccc; }
  .control:hover input:not([disabled]):checked ~
      .control__indicator {
    background: #0e647d; }

.control__indicator {
  position: absolute;
  top: 2px;
```

```
512   left: 0;
513   height: 20px;
514   width: 20px;
515   background: #e6e6e6; }
516   .control__indicator:after {
517     content: '';
518     position: absolute;
519     display: none; }
520
521 .control--checkbox .control__indicator:after {
522   left: 8px;
523   top: 3px;
524   width: 5px;
525   height: 10px;
526   border: solid #fff;
527   border-width: 0 2px 2px 0;
528   transform: rotate(45deg); }
529
530 .control--checkboxinput:disabled ~ .control__indicator:after {
531   border-color: #7b7b7b; }
532
533
534 webapp/src/components/form/checkbox/checkbox.sass
535
536 .control
537     display: block
538     position: relative
539     padding-left: 30px
540     margin-bottom: 15px
541     cursor: pointer
542     font-size: 18px
543     input
544         position: absolute
545         z-index: -1
546         opacity: 0
547         &:focus ~ .control__indicator
548             background: #ccc
549         &:checked ~ .control__indicator
550             background: #2aa1c0
551             &:after
552                 display: block
553         &:checked:focus ~ .control__indicator
554             background: #0e647d
555         &:disabled ~ .control__indicator
556             background: #e6e6e6
```

```
557             opacity: 0.6
558             pointer-events: none
559
560     &:hover
561         input ~ .control__indicator
562             background: #ccc
563         input:not([disabled]):checked ~ .control__indicator
564             background: #0e647d
565
566 .control__indicator
567     position: absolute
568     top: 2px
569     left: 0
570     height: 20px
571     width: 20px
572     background: #e6e6e6
573     &:after
574         content: ''
575         position: absolute
576         display: none
577
578 .control--checkbox
579     .control__indicator:after
580         left: 8px;
581         top: 3px;
582         width: 5px;
583         height: 10px;
584         border: solid #fff
585         border-width: 0 2px 2px 0
586         transform: rotate(45deg)
587     &input:disabled ~ .control__indicator:after
588         border-color: #7b7b7b
589
590
591 webapp/src/components/form/option/index.js
592
593 import Option from './Option.jsx'
594 export default Option;
595
596 webapp/src/components/form/option/Option.jsx
597
598 import React from 'react';
599 import { FormGroup, Input, Label, FormFeedback } from
        'reactstrap';
600 import Required from '../Required.jsx'
```

```
import PropTypes from 'prop-types';
import requiredIf from 'react-required-if';

const Option = ({label, type, onChange, required,
    invalidMessage, ...rest}) => (
    <FormGroup check>
        <Label check>
            <Input type={type} onChange={onChange} {...rest}/>
            {' ' + label}
            {required? <Required/>: null}
        </Label>
        <FormFeedback>{invalidMessage}</FormFeedback>
    </FormGroup>
);

Option.propTypes = {
    label: PropTypes.string.isRequired,
    invalidMessage: requiredIf(PropTypes.string, props =>
        props.required),
    type: PropTypes.oneOf(['checkbox', 'radio']),
    required: PropTypes.bool,
};

Option.defaultProps = {
    type: 'radio',
};

export default Option;
```

webapp/src/components/form/switch/Switch.jsx

```
import React from 'react';
import { Label } from 'reactstrap';
import Switch from 'rc-switch';
import 'rc-switch/assets/index.css';
import './switch.css';

export default ({ id, label, onChange, ...rest}) => (
    <div>
        <Label for={id}>{label}</Label>
        <Switch id={id} onChange={onChange}
            checkedChildren="Sim" unCheckedChildren="No"
            style={{width: '50px'}} {...rest}/>
    </div>
)
```

webapp/src/components/form/switch/switch.sass

```
.rc-switch-checked
    &:after
        left: 65%
```

webapp/src/components/form/switch/index.js

```
import Switch from './Switch.jsx'
export default Switch;
```

webapp/src/components/form/switch/switch.css

```
.rc-switch-checked:after {
  left: 65%; }
```

webapp/src/components/form/fileInput/FileInput.jsx

```
import React from 'react';
import PropTypes from 'prop-types';
import { Input, Label } from 'reactstrap';
import './fileInput.css'

const FileInput = ({label, onChange, className, color, ...rest})
    => (
    <Label className={`btn btn-${color} ${className}`}>
        <Input type="file" name="file" onChange={onChange}
            {...rest}/>
        {label}
    </Label>
);

FileInput.propTypes = {
    label: PropTypes.string.isRequired,
    onChange: PropTypes.func.isRequired,
    className: PropTypes.string,
    color: PropTypes.oneOf([
        'primary',
        'success',
        'info',
        'warning',
        'danger'
    ])
```

```
};

FileInput.defaultProps = {
    color: 'info'
};

export default FileInput;
```

webapp/src/components/form/fileInput/fileInput.sass

```
label
    cursor: pointer
    input[type=file]
        display: none;
```

webapp/src/components/form/fileInput/index.js

```
import FileInput from './FileInput';
export default FileInput;
```

webapp/src/components/form/fileInput/fileInput.css

```
label {
  cursor: pointer; }
  label input[type=file] {
    display: none; }
```

webapp/src/components/form/Required.jsx

```
import React from 'react';
export default () => (<span className="text-danger">*</span>);
```

webapp/src/components/form/index.js

```
export {default as Input} from './input';
export {default as FileInput} from './fileInput';
export {default as DatePicker} from './datePicker';
export {default as MaskInput} from './maskInput';
export {default as Wizard} from './wizard';
export {default as Option} from './option';
export {default as Switch} from './switch';
export {default as Checkbox} from './checkbox';
export {default as Required} from './Required.jsx';
```

730 webapp/src/components/calender/Calendar.jsx

```
731
732 import React, { Component } from 'react';
733 import BigCalendar from 'react-big-calendar'
734 import PropTypes from 'prop-types';
735 import requiredIf from 'react-required-if';
736 import dates from 'react-big-calendar/lib/utils/dates'
737 import moment from 'moment'
738 import 'react-big-calendar/lib/css/react-big-calendar.css';
739 import './calendar.css';
740
741 BigCalendar.setLocalizer(BigCalendar.momentLocalizer(moment))
742
743 const { MONTH, WEEK, DAY } = BigCalendar.Views
744 const allViews = [MONTH, WEEK, DAY]
745
746 export default class extends Component{
747     static propsTypes = {
748         selectable: PropTypes.bool,
749         events: PropTypes.array,
750         onSelectEvent: requiredIf(PropTypes.func, props =>
                props.selectable),
751         onSelectSlot: PropTypes.func
752     };
753     static defaultProps = {
754         events: [],
755         onSelectEvent: () => null,
756         onSelectSlot: () => null
757     };
758     render(){
759         const { events, selectable, onSelectEvent, onSelectSlot,
                defaultView } = this.props;
760         return (
761             <BigCalendar
762                 selectable={selectable}
763                 onSelectEvent={onSelectEvent}
764                 onSelectSlot={onSelectSlot}
765                 events={events}
766                 defaultView={defaultView}
767                 views={allViews}
768                 max={dates.endOf(new Date(), 'day')}
769                 defaultDate={new Date()}
770             />
771         );
772     }
```

```
773 }
774
775 webapp/src/components/calender/index.js
776
777 import Calendar from './Calendar.jsx';
778 export default Calendar;
779
780 webapp/src/components/calender/calendar.sass
781
782 .rbc-event-content
783     background-color: #007BFF
784     border-color: #007BFF
785
786 .rbc-toolbar
787   flex-wrap: wrap
788   justify-content: center
789
790
791 webapp/src/components/calender/calendar.css
792
793 .rbc-event-content {
794   background-color: #007BFF;
795   border-color: #007BFF; }
796
797 .rbc-toolbar {
798   flex-wrap: wrap;
799   justify-content: center; }
800
801
802 webapp/src/components/filter/Filter.jsx
803
804 import React from 'react';
805 import { Row, Col, Form } from 'reactstrap';
806 import PropTypes from 'prop-types'
807 import { Input } from '../form'
808
809 const Filter = ({ value, label, handlerFilter }) => (
810     <Row style={{ marginBottom: '5px' }}>
811         <Col>
812             <Form inline>
813                 <Input id="filter" value={value} label={label}
                        onChange={handlerFilter} />
814             </Form>
815         </Col>
816     </Row>
```

```
);

Filter.propTypes = {
    handlerFilter: PropTypes.func.isRequired,
    value: PropTypes.string,
    label: PropTypes.string
}

Filter.defaultProps ={
    label: " "
}

export default Filter;
```

webapp/src/components/filter/index.js

```
import Filter from './Filter';
export default Filter;
```

webapp/src/components/header/Header.jsx

```
import React, { Fragment } from 'react';
import { Navbar, NavbarBrand, Nav, UncontrolledDropdown,
    DropdownToggle, DropdownMenu, DropdownItem } from
    'reactstrap';
import { NavLink } from 'react-router-dom';
import logo from './logo.png';
import { Roles } from '../../model';

export default ({role, name, logged, logout}) => (
    <Navbar color="light" light>
        <NavbarBrand tag="button" className="btn
            btn-outline-secondary" style={{ borderWidth: '0'} }>
            <NavLink to="/">
                <img src={logo} alt="Logo" height="35px"/>
            </NavLink>
        </NavbarBrand>
        {
            logged && (
                <Nav className="ml-auto" navbar>
                    <UncontrolledDropdown nav inNavbar>
                        <DropdownToggle nav caret>
                            {name}
                        </DropdownToggle>
                        <DropdownMenu right>
```

```
                        {
                            role === Roles.ADMIN && (
                                <Fragment>
                                    <DropdownItem>
                                        <NavLink to="/admin">
                                            administrao
                                        </NavLink>
                                    </DropdownItem>
                                    <DropdownItem divider />
                                </Fragment>
                            )
                        }
                            <DropdownItem onClick={logout}>
                                Sair
                            </DropdownItem>
                        </DropdownMenu>
                    </UncontrolledDropdown>
                </Nav>
            )
        }
    </Navbar>
);
```

webapp/src/components/index.js

```
export { default as Header } from './header/Header';
export { default as Footer } from './footer/Footer';
export {
    Input,
    FileInput,
    DatePicker,
    MaskInput,
    Wizard,
    Option,
    Switch,
    Required,
    Checkbox
} from './form';
export { default as Calendar } from './calender';
export { default as Pagination } from './pagination';
export { default as Filter } from './filter';
```

webapp/src/model/role.js

```
const Roles = {
```

```
904     ADMIN: 'ADMIN',
905     VOLUNTEER: 'VOLUNTEER',
906     COMPANY_ADMIN: 'COMPANY_ADMIN',
907     SCHOOL_ADMIN: 'SCHOOL_ADMIN',
908     values: () => [
909         Roles.ADMIN,
910         Roles.VOLUNTEER,
911         Roles.COMPANY_ADMIN,
912         Roles.SCHOOL_ADMIN
913     ],
914     has: role => Roles.values().includes(role),
915     translate: role =>{
916         return {
917             ADMIN: 'Administrador',
918             VOLUNTEER: 'Voluntrio',
919             COMPANY_ADMIN: 'Responsavel na empresa',
920             SCHOOL_ADMIN: 'Responsavel da escola',
921         }[role]
922     }
923 };
924
925 export default Roles;
926
927 webapp/src/model/question.js
928
929 export default class{
930     constructor(codQuestion, question = '', responds = '') {
931         this.codQuestion = codQuestion;
932         this.question = question;
933         this.responds = responds;
934     }
935 }
936
937 webapp/src/model/event.js
938
939 import Address from './address';
940 import School from './school';
941
942 export default class{
943     constructor(codEvent, startOccurrence, endOccurrence,
        createAt, project, type = '', allDay,
944         address = new Address(), school = new School(),
            trainings = [],
945         participants = [], ratings = []) {
946         this.codEvent = codEvent;
```

```
947         this.startOccurrence = startOccurrence;
948         this.endOccurrence = endOccurrence;
949         this.createAt = createAt;
950         this.project = project;
951         this.type = type;
952         this.allDay = allDay;
953         this.address = address;
954         this.trainings = trainings;
955         this.participants = participants;
956         this.ratings = ratings;
957         this.school = school;
958     }
959 }
960
961 webapp/src/model/user.js
962
963 import Address from './address';
964
965 export default class{
966     constructor(cpf='', name='', email='', password='', role='',
          status='',
967             birth='', job='', phone='', address=new Address() ) {
968         this.cpf = cpf;
969         this.name = name;
970         this.email = email;
971         this.password = password;
972         this.role = role;
973         this.status = status;
974         this.birth = birth;
975         this.job = job;
976         this.phone = phone;
977         this.address = address;
978     }
979 }
980
981 webapp/src/model/school.js
982
983 import Address from './address';
984 import User from './user';
985 import Roles from './role';
986 import Status from './status';
987
988 export default class {
989     constructor(codSchool, name = '', phone = '', schoolType =
          '', address = new Address(), responsible = new User()) {
```

```
990         this.codSchool = codSchool;
991         this.name = name
992         this.phone = phone;
993         this.schoolType = schoolType;
994         this.address = address;
995         responsible.role = Roles.SCHOOL_ADMIN;
996         responsible.status = Status.APPROVED
997         this.responsible = responsible
998     }
999 }
1000
1001 webapp/src/model/address.js
1002
1003 export default class {
1004     constructor(codAddress, city = '', neighborhood = '', street
          = '', number = '') {
1005         this.codAddress = codAddress;
1006         this.city = city;
1007         this.neighborhood = neighborhood;
1008         this.street = street;
1009         this.number = number;
1010     }
1011     toString(){
1012         return `${this.street}, ${this.number},
              ${this.neighborhood} - ${this.city}`;
1013     }
1014 }
1015
1016 webapp/src/model/index.js
1017
1018 export { default as Address } from './address';
1019 export { default as Roles } from './role';
1020 export { default as Status } from './status';
1021 export { default as User } from './user';
1022 export { default as Volunteer } from './volunteer';
1023 export { default as Question } from './question';
1024 export { default as Answer } from './answer';
1025 export { default as Schooling } from './schooling';
1026 export { default as SchoolType } from './schoolType';
1027 export { default as School } from './school';
1028 export { default as Event } from './event';
1029 export { default as EventType } from './eventType';
1030 export { default as Attachment } from './attachment';
1031 export { default as Training } from './training';
1032 export { default as Project } from './project';
```

webapp/src/model/volunteer.js

```
import User from './user';
import Roles from './role';
import Status from './status';
import Company from './company';
import School from './school';

export default class{
    constructor(user = new User(), company = new Company(),
        school = new School(),
        schooling = '', conclusion = false, course ='', rg = '',
            answers = [], ratings = []) {
        user.role = Roles.VOLUNTEER;
        user.status = Status.REGISTER;
        this.user = user;
        this.company = company;
        this.schooling = schooling;
        this.conclusion = conclusion;
        this.course = course;
        this.rg = rg;
        this.answers = answers;
        this.ratings = ratings;
        this.school = school;
    }
}
```

webapp/src/model/answer.js

```
import Question from './question';
export default class {
    constructor(question = new Question(), answer = false,
        comment = '') {
        this.question = question;
        this.answer = answer;
        this.comment = comment;
    }
}
```

webapp/src/model/company.js

```
import Address from './address';
import User from './user';
import Roles from './role';
```

```
1075 import Status from './status';
1076
1077 export default class{
1078     constructor(cnpj='', name='', phone='', address = new
         Address(), responsible = new User()) {
1079         this.cnpj = cnpj;
1080         this.name = name
1081         this.phone = phone;
1082         this.address = address;
1083         responsible.role = Roles.COMPANY_ADMIN;
1084         responsible.status = Status.APPROVED
1085         this.responsible = responsible
1086     }
1087 }
1088
1089 webapp/src/model/project.js
1090
1091 export default class{
1092     constructor(codProject, start, end, name='') {
1093         this.codProject = codProject;
1094         this.name = name;
1095         this.start = start;
1096         this.end = end;
1097     }
1098 }
1099
1100 webapp/src/model/status.js
1101
1102 const Status = {
1103     APPROVED: 'APPROVED',
1104     WAIT_STATEMENT: 'WAIT_STATEMENT',
1105     WAIT_COMPANY: 'WAIT_COMPANY',
1106     WAIT_TRAINING: 'WAIT_TRAINING',
1107     REGISTER: 'REGISTER',
1108     values: () => [
1109         Status.APPROVED,
1110         Status.WAIT_STATEMENT,
1111         Status.WAIT_COMPANY,
1112         Status.WAIT_TRAINING,
1113         Status.REGISTER
1114     ],
1115     has: status => Status.values().includes(status),
1116     translate: status => {
1117         return {
1118             APPROVED: 'Aprovado',
```

```
1119            WAIT_STATEMENT: 'Aguardando envio do termo de
                    resposabilidade',
1120            WAIT_COMPANY: 'Aguardando recomendao da empresa',
1121            WAIT_TRAINING: 'Aguardando treinamento',
1122            REGISTER: 'Registrado',
1123        }[status];
1124    },
1125    next: status => {
1126        return {
1127            APPROVED: Status.APPROVED,
1128            WAIT_STATEMENT: Status.WAIT_TRAINING,
1129            WAIT_COMPANY: Status.WAIT_STATEMENT,
1130            WAIT_TRAINING: Status.APPROVED,
1131            REGISTER: Status.WAIT_COMPANY,
1132        }[status];
1133    }
1134 };
1135
1136 export default Status;
1137
1138 webapp/src/model/training.js
1139
1140 export default class{
1141     constructor(codTraining, name = '', path = '', link = '') {
1142         this.codTraining = codTraining;
1143         this.name = name;
1144         this.path = path;
1145         this.link = link;
1146     }
1147 }
1148
1149 webapp/src/model/schooling.js
1150
1151 const Schooling = {
1152     ELEMENTARY_SCHOOL: 'ELEMENTARY_SCHOOL',
1153     HIGH_SCHOOL: 'HIGH_SCHOOL',
1154     UNIVERSITY_GRADUATE: 'UNIVERSITY_GRADUATE',
1155     POSTGRADUATE_STUDIES: 'POSTGRADUATE_STUDIES',
1156     values: () => [
1157         Schooling.ELEMENTARY_SCHOOL,
1158         Schooling.HIGH_SCHOOL,
1159         Schooling.UNIVERSITY_GRADUATE,
1160         Schooling.POSTGRADUATE_STUDIES,
1161     ],
1162     has: schooling => Schooling.values().includes(schooling),
```

```
1163        translate: schooling => {
1164            return {
1165                ELEMENTARY_SCHOOL: '1 Grau (Fundamental)',
1166                HIGH_SCHOOL: '2 Grau (Ensino Mdio)',
1167                UNIVERSITY_GRADUATE: 'Ensino Superior (Graduao)',
1168                POSTGRADUATE_STUDIES: 'Ps- Graduao '
1169            }[schooling];
1170        }
1171    }
1172
1173    export default Schooling;
1174
1175    webapp/src/model/attachment.js
1176
1177    export default class{
1178        constructor(codAttachment, name = '', status = '', required
                = false, download = false) {
1179            this.codAttachment = codAttachment;
1180            this.name = name;
1181            this.required = required;
1182            this.status = status;
1183            this.download = download;
1184        }
1185    }
1186
1187    webapp/src/model/schoolType.js
1188
1189    const SchoolType = {
1190        MUNICIPAL: 'MUNICIPAL',
1191        STATE: 'STATE',
1192        FEDERAL: 'FEDERAL',
1193        values: () => [
1194            SchoolType.MUNICIPAL,
1195            SchoolType.STATE,
1196            SchoolType.FEDERAL
1197        ],
1198        has: type => SchoolType.values().includes(type),
1199        translate: type => {
1200            return {
1201                MUNICIPAL: 'Municipal',
1202                STATE: 'Estadual',
1203                FEDERAL: 'Federal'
1204            }[type];
1205        }
1206    }
```

```
1207
1208 export default SchoolType;
1209
1210 webapp/src/model/eventType.js
1211
1212 export default class{
1213     constructor(type, name, trainings=[]){
1214         this.type = type;
1215         this.name = name;
1216         this.trainings = trainings;
1217     }
1218 }
1219
1220 webapp/src/pages/public/register/Page.jsx
1221
1222 import React, { Component } from 'react';
1223 import { Row, Col, Form, Label } from 'reactstrap';
1224 import { Input, Wizard, DatePicker, MaskInput, Option, Switch,
        Required } from '../../../components';
1225 import { Volunteer, Schooling, Answer, Roles } from
        '../../../model';
1226 import { getRequest, postRequest } from '../../../utils/http';
1227 import axios from 'axios';
1228
1229 export default class extends Component{
1230     constructor(){
1231         super();
1232         this.state = {
1233             volunteer: new Volunteer(),
1234             questions: [],
1235             companies: [],
1236             confirmPassword: ''
1237         }
1238         this.handlerName = this.handlerName.bind(this);
1239         this.handlerCpf = this.handlerCpf.bind(this);
1240         this.handlerRg = this.handlerRg.bind(this);
1241         this.handlerPhone = this.handlerPhone.bind(this);
1242         this.handlerComplete = this.handlerComplete.bind(this)
1243         this.handlerCourse = this.handlerCourse.bind(this)
1244         this.handlerSchooling = this.handlerSchooling.bind(this)
1245         this.handlerJob = this.handlerJob.bind(this);
1246         this.handlerCompany = this.handlerCompany.bind(this);
1247         this.handlerEmail = this.handlerEmail.bind(this);
1248         this.handlerBirth = this.handlerBirth.bind(this);
1249         this.handlerPassword = this.handlerPassword.bind(this);
```

```
1250        this.handlerConfirmPassword =
                this.handlerConfirmPassword.bind(this);
1251        this.handlerCity = this.handlerCity.bind(this);
1252        this.handlerNeighborhood =
                this.handlerNeighborhood.bind(this);
1253        this.handlerStreet = this.handlerStreet.bind(this);
1254        this.handlerNumber = this.handlerNumber.bind(this);
1255        this.handlerAnswer = this.handlerAnswer.bind(this);
1256        this.handlerAnswerComment =
                this.handlerAnswerComment.bind(this);
1257        this.handlerSubmit = this.handlerSubmit.bind(this);
1258    }
1259
1260    componentWillMount(){
1261        axios.all([
1262            getRequest(`/question/${Roles.VOLUNTEER}`, res =>
                    res.data),
1263            getRequest('/company', res => res.data )
1264        ]).then(res => {
1265            const { volunteer } = this.state;
1266            volunteer.answers = res[0].map(question => new
                    Answer(question))
1267            this.setState({ questions: res[0], companies: res[1]
                    })
1268        });
1269    }
1270
1271    // user
1272    handlerName(event){
1273        const { volunteer } = this.state;
1274        const { user } = volunteer;
1275        user.name = event.target.value;
1276        volunteer.user = user;
1277        this.setState({ volunteer });
1278    }
1279
1280    handlerRg(event){
1281        const { volunteer } = this.state;
1282        volunteer.rg = event.target.value.replace('_', '');
1283        this.setState({ volunteer });
1284    }
1285
1286    handlerCpf(event){
1287        const { volunteer } = this.state;
1288        const { user } = volunteer;
```

```
1289            user.cpf = event.target.value.replace(/[.-]/g, '');
1290            volunteer.user = user;
1291            this.setState({ volunteer });
1292        }
1293
1294        handlerPhone(event) {
1295            const { volunteer } = this.state;
1296            const { user } = volunteer;
1297            user.phone = event.target.value.replace(/[()-]/g, '');
1298            volunteer.user = user;
1299            this.setState({ volunteer });
1300        }
1301
1302        handlerJob(event){
1303            const { volunteer } = this.state;
1304            const { user } = volunteer;
1305            user.job = event.target.value;
1306            volunteer.user = user;
1307            this.setState({ volunteer });
1308        }
1309
1310        handlerCompany(event){
1311            const { volunteer, companies } = this.state;
1312            for (const company of companies) {
1313                if(company.cnpj === event.target.value){
1314                    volunteer.company = company;
1315                    break;
1316                }
1317            }
1318            this.setState({ volunteer });
1319        }
1320
1321        handlerEmail(event){
1322            const { volunteer } = this.state;
1323            const { user } = volunteer;
1324            user.email = event.target.value;
1325            volunteer.user = user;
1326            this.setState({ volunteer });
1327        }
1328
1329        handlerBirth(event){
1330            const { volunteer } = this.state;
1331            const { user } = volunteer;
1332            const { value } = event.target;
1333            user.birth = value ? new Date(value) : null;
```

```
1334        volunteer.user = user;
1335        this.setState({ volunteer });
1336    }
1337
1338    handlerComplete(value){
1339        const { volunteer } = this.state;
1340        volunteer.conclusion = value;
1341        this.setState({ volunteer });
1342    }
1343
1344    handlerSchooling(event){
1345        const { volunteer } = this.state;
1346        volunteer.schooling = event.target.value;
1347        this.setState({ volunteer });
1348    }
1349
1350    handlerCourse(event){
1351        const { volunteer } = this.state;
1352        volunteer.course = event.target.value;
1353        this.setState({ volunteer });
1354    }
1355
1356    handlerPassword(event){
1357        const { volunteer } = this.state;
1358        const { user } = volunteer;
1359        user.password = event.target.value;
1360        volunteer.user = user;
1361        this.setState({ volunteer });
1362    }
1363
1364    handlerConfirmPassword(event){
1365        let { confirmPassword } = this.state;
1366        confirmPassword = event.target.value;
1367        this.setState({ confirmPassword });
1368    }
1369
1370    //address
1371    handlerCity(event) {
1372        const { volunteer } = this.state;
1373        const { address } = volunteer.user;
1374        address.city = event.target.value;
1375        volunteer.user.address = address;
1376        this.setState({ volunteer });
1377    }
1378
```

```
1379        handlerNeighborhood(event) {
1380            const { volunteer } = this.state;
1381            const { address } = volunteer.user;
1382            address.neighborhood = event.target.value;
1383            volunteer.user.address = address;
1384            this.setState({ volunteer });
1385        }
1386
1387        handlerStreet(event) {
1388            const { volunteer } = this.state;
1389            const { address } = volunteer.user;
1390            address.street = event.target.value;
1391            volunteer.user.address = address;
1392            this.setState({ volunteer });
1393        }
1394
1395        handlerNumber(event) {
1396            const { volunteer } = this.state;
1397            const { address } = volunteer.user;
1398            address.number = event.target.value;
1399            volunteer.user.address = address;
1400            this.setState({ volunteer });
1401        }
1402
1403        //Question
1404        handlerAnswer(value, codQuestion){
1405            const { volunteer } = this.state;
1406            for (const answer of volunteer.answers) {
1407                if (answer.question.codQuestion === codQuestion){
1408                    answer.answer = value;
1409                    break;
1410                }
1411            }
1412            this.setState({ volunteer });
1413        }
1414
1415        handlerAnswerComment(event, codQuestion){
1416            const { volunteer } = this.state;
1417            for (const answer of volunteer.answers) {
1418                if (answer.question.codQuestion === codQuestion){
1419                    answer.answer = event.target.value;
1420                    break;
1421                }
1422            }
1423            this.setState({ volunteer });
```

```
1424        }
1425
1426        handlerSubmit(){
1427            const { afterSubmit } = this.props;
1428            postRequest(
1429                '/volunteer',
1430                this.state.volunteer,
1431                () => postRequest('/auth', {}, afterSubmit)
1432            );
1433        }
1434
1435        render(){
1436            const { questions, companies, volunteer, confirmPassword
                        } = this.state;
1437            const { user, company } = volunteer;
1438            const maxDate = new Date().toJSON().split('T')[0];
1439            return (
1440                <Row>
1441                    <Col/>
1442                    <Col>
1443                        <Wizard onCancel={()=> null}
                                submitLabel="cadastrar"
                                onSubmit={this.handlerSubmit}>
1444                            <div>
1445                                <h3>Dados do Voluntrio</h3>
1446                                <Input id="nameReponsible" label="Nome
                                        Completo" invalidMessage="Nome
                                        Completo  obrigatrio"
                                        value={user.name}
                                        onChange={this.handlerName}
                                        required />
1447                                <Row>
1448                                    <Col>
1449                                        <MaskInput id="cpf" label="CPF"
                                                invalidMessage="CPF
                                                obrigatrio"
                                                mask="999.999.999-99"
                                                value={user.cpf}
                                                onChange={this.handlerCpf}
                                                required />
1450                                    </Col>
1451                                    <Col>
1452                                        <MaskInput id="rg" label="RG"
                                                invalidMessage="RG
                                                obrigatrio"
```

```
                        mask="99999999999"
                        value={volunteer.rg}
                        onChange={this.handlerRg}
                        required />
1453                </Col>
1454            </Row>
1455            <MaskInput id="phoneResponsible"
                    label="Nmero para Contato"
                    invalidMessage="Nmero para Contato
                     obrigatrio" value={user.phone}
                    mask="(99)9999-9999"
                    onChange={this.handlerPhone}
                    required />
1456            <Input id="company" type="select"
                    label="Empresa"
                    invalidMessage="Empresa
                    obrigatrio" value={company.cnpj}
                    onChange={this.handlerCompany}
                    required >
                   <option value="">Selecione</option>
1457               {
1458                   companies.length ?
1459                       companies.map(
1460                           company => <option
1461                               key={company.cnpj}
                                    value={company.cnpj}>{company.n
1462                       ) : null
1463               }
1464            </Input>
1465            <Input id="job" label="Cargo na
                    Empresa" invalidMessage="Cargo na
                    Empresa  obrigatrio"
                    value={user.job}
                    onChange={this.handlerJob}
                    required />
1466            <Row>
1467                <Col>
1468                    <Label
                            for="radio1">Escolaridade
                            <Required/></Label>
1469                    {
1470                        Schooling.values().map(
1471                            schooling => (
1472                                <Option
                                        name="radio1"
```

```
                                    key={schooling}
                                    value={schooling}
                                    label={Schooling.translate(school
                                    onChange={this.handlerSchooling}/
                )
            )
        }
    </Col>
    <Col>
        <Row>
            <Col>
                <Switch id="complete"
                    label="Completo"
                    onChange={this.handlerComplete}
                    value={volunteer.conclusion}
                    checkedChildren="Sim"
                    unCheckedChildren="No"/>
            </Col>
        </Row>
        {
            volunteer.schooling ===
                Schooling.UNIVERSITY_GRADUATE
                ||
            volunteer.schooling ===
                Schooling.POSTGRADUATE_STUDIES?
                (
            <div>
                <br/>
                <Row>
                    <Col>
                        <Form inline>
                            <Input
                                id="course"
                                label="Curso"
                                invalidMessage="Curso

                                obrigatrio"
                                value={volunteer.cours
                                onChange={this.handler
                                required/>
                        </Form>
                    </Col>
                </Row>
            </div>
            ):
```

```
1497                                    null
1498                                }
1499                            </Col>
1500                        </Row>
1501                        <DatePicker id="birth"
                                label="Nascimento"
                                invalidMessage="Nascimento
                                obrigatrio" max={maxDate}
                                onChange={this.handlerBirth}
                                required />
1502                        <Input id="email" type="email"
                                label="Email"
                                invalidMessage="Email  obrigatrio"
                                value={user.email}
                                onChange={this.handlerEmail}
                                required />
1503                        <Row>
1504                            <Col>
1505                                <Input id="password"
                                        type="password"
                                        label="Senha"
                                        invalidMessage="Senha
                                        obrigatrio"
                                        value={user.password}
                                        onChange={this.handlerPassword}
                                        required />
1506                            </Col>
1507                            <Col>
1508                                <Input id="passwordConfirm"
                                        type="password"
                                        label="Confirmar Senha"
                                        invalidMessage="As senhas
                                        devem ser idnticas"
                                        value={confirmPassword}
                                        onChange={this.handlerConfirmPassword}
                                        required />
1509                            </Col>
1510                        </Row>
1511                    </div>
1512                    <div>
1513                        <h3> Endero  do  Voluntrio </h3>
1514                        <Input id="city" label="Cidade"
                                invalidMessage="Cidade
                                obrigatrio"
                                onChange={this.handlerCity}
```

```
                    required />
                <Input id="neighborhood"
                    label="Bairro"
                    invalidMessage="Bairro
                    obrigatrio"
                    onChange={this.handlerNeighborhood}
                    required />
                <Row>
                    <Col>
                        <Input id="street"
                            label="Lougradouro"
                            invalidMessage="Lougradouro
                             obrigatrio"
                            onChange={this.handlerStreet}
                            required />
                    </Col>
                    <Col md="3">
                        <Input id="number"
                            label="Nmero"
                            onChange={this.handlerNumber}
                            />
                    </Col>
                </Row>
            </div>
            <div>
                {
                    questions.map((question, index) => (
                        <div key={question.codQuestion}>
                            <Row >
                                <Col>
                                    <strong>{question.question}</strong>
                                </Col>
                                <Col md="2">
                                    <Switch
                                        id={`question-${question.codQuest
                                        label="Resposta"
                                        onChange={(value)
                                        =>
                                        this.handlerAnswer(value,
                                        question.codQuestion)}/>
                                </Col>
                                <Col>
                                    <textarea
                                        id={`details-${question.codQuesti
                                        cols="30"
```

```
                                                    rows="3"
                                                    onChange={(event)=>
                                                    this.handlerAnswerComment(e
                                                    question.codQuestion)}/>
1538                                            </Col>
1539                                        </Row>
1540                                        { index !==
                                                question.length-1
                                                ?<hr/> : null}
1541                                    </div>
1542                                ))
1543                            }
1544                        </div>
1545                    </Wizard>
1546                </Col>
1547                <Col/>
1548            </Row>
1549        );
1550    }
1551 }
1552
1553 webapp/src/pages/public/register/index.js
1554
1555 import Page from './Page.jsx';
1556 export default Page;
1557
1558 webapp/src/pages/public/login/Page.jsx
1559
1560 import React, { Component } from 'react';
1561 import { Redirect, Link } from 'react-router-dom';
1562 import { Button, Row, Col, Form } from 'reactstrap';
1563 import { Input } from '../../../components';
1564 import { postRequest } from '../../../utils/http';
1565
1566 export default class extends Component{
1567     constructor(){
1568         super();
1569         this.state={
1570             cpf: '',
1571             password: ''
1572         };
1573         this.handlerUsername = this.handlerUsername.bind(this);
1574         this.handlerPassword = this.handlerPassword.bind(this);
1575         this.handlerSubmit = this.handlerSubmit.bind(this);
1576     }
```

```
1577
1578        handlerUsername(event){
1579            this.setState({ cpf: event.target.value});
1580        }
1581
1582        handlerPassword(event){
1583            this.setState({password: event.target.value});
1584        }
1585
1586        handlerSubmit(){
1587            postRequest('/auth', this.state, res =>
                    this.props.afterLogin(res.data.user));
1588        }
1589
1590        render(){
1591            const { logged } = this.props;
1592            const { cpf, password } = this.state;
1593            return !logged ? (
1594                <Row>
1595                    <Col/>
1596                    <Col md="4" sm="10">
1597                        <Row>
1598                            <Col>
1599                                <Form onKeyPress={({ key }) => key ===
                                        'Enter' && this.handlerSubmit()}>
1600                                    <Input id="login" label="Usurio"
                                            value={cpf}
                                            onChange={this.handlerUsername}/>
1601                                    <Input id="password" label="Senha"
                                            type="password"
                                            value={password}
                                            onChange={this.handlerPassword}/>
1602                                    <Button type="button"
                                            color="primary" size="lg"
                                            onClick={this.handlerSubmit}
                                            block>Entrar</Button>
1603                                </Form>
1604                            </Col>
1605                        </Row>
1606                        <Row>
1607                            <Col className="text-center">
1608                                <Link to="cadastrar">Quero ser
                                        voluntrio</Link>
1609                            </Col>
1610                        </Row>
```

```
                </Col>
                <Col/>
            </Row>
        ) : (
            <Redirect to="/dashboard"/>
        );
    }
}
```

webapp/src/pages/public/login/index.js

```
import Page from './Page.jsx';
export default Page;
```

webapp/src/pages/public/index.js

```
export {default as Login} from './login';
export {default as Register} from './register';
```

webapp/src/pages/protected/admin/attachment/Form.jsx

```
import React, { Component } from 'react';
import { Attachment, Status } from '../../../../model';
import PropTypes from 'prop-types';
import { postRequest } from '../../../../utils/http';
import { Row, Col, Form, Button, Card, CardBody } from
    'reactstrap';
import { Input, FileInput, Switch } from
    '../../../../components';

export default class extends Component{
    constructor(){
        super();
        this.state= {
            attachment: new Attachment(),
            file: null
        };
        this.handlerAttatchment =
            this.handlerAttatchment.bind(this);
        this.handlerRequired = this.handlerRequired.bind(this);
        this.handlerDownload = this.handlerDownload.bind(this);
        this.handlerStatus = this.handlerStatus.bind(this);
        this.handlerFile = this.handlerFile.bind(this);
        this.handlerSubmit = this.handlerSubmit.bind(this);
    }
```

```
1653
1654        static propTypes = {
1655            afterSubmit: PropTypes.func.isRequired
1656        }
1657
1658        handlerAttatchment(event){
1659            const { attachment } = this.state;
1660            attachment.name = event.target.value;
1661            this.setState({ attachment });
1662        }
1663
1664        handlerRequired(value){
1665            const { attachment } = this.state;
1666            attachment.required = value;
1667            this.setState({ attachment });
1668        }
1669
1670        handlerDownload(value){
1671            const { attachment } = this.state;
1672            attachment.download = value;
1673            this.setState({ attachment });
1674        }
1675
1676        handlerStatus(event){
1677            const { attachment } = this.state;
1678            attachment.status = event.target.value;
1679            this.setState({ attachment });
1680        }
1681
1682        handlerFile(event){
1683            this.setState({ file: event.target.files[0] });
1684        }
1685
1686        handlerSubmit(){
1687            const { afterSubmit } = this.props;
1688            const { attachment, file } = this.state;
1689            const json = JSON.stringify(attachment);
1690            const blob = new Blob([json], {
1691                type: 'application/json'
1692            });
1693            let form = new FormData();
1694            form.append('attachment', blob);
1695            if(file){
1696                form.append('file', file);
1697            }
```

```
1698            postRequest('/attachment', form, afterSubmit)
1699        }
1700
1701        render(){
1702            const { attachment } = this.state;
1703            return (
1704                <Card>
1705                    <CardBody>
1706                        <Form>
1707                            <Input id="name" label="Anexo"
                                    onChange={this.handlerAttatchment}
                                    invalidMessage="Anexo  Obrigatrio"
                                    required/>
1708                            <Row>
1709                                <Col md="5">
1710                                    <Switch id="required" label="Envio
                                            Obrigatrio "
                                            onChange={this.handlerRequired}/>
1711                                </Col>
1712                                <Col>
1713                                    <Switch id="dowload"
                                            label="Disponibilizado Pela CnE
                                            "
                                            onChange={this.handlerDownload}/>
1714                                </Col>
1715                            </Row>
1716                            {
1717                                attachment.download && <FileInput
                                        color="info" label="Upload"
                                        onChange={this.handlerFile}
                                        accept="application/pdf" required/>
1718                            }
1719                            <Input id="status" type="select"
                                    label="Enviado Durante"
                                    onChange={this.handlerStatus}
                                    invalidMessage="Enviado Durante
                                    Obrigatrio" required>
1720                                <option value="">Selecione</option>
1721                                {
1722                                    Status.values().map(status =>
                                            <option key={status}
                                            value={status}>{Status.translate(status)}
1723                                    }
1724                            </Input>
```

```
                        <Button type="button" color="primary"
                            className="float-right"
                            onClick={this.handlerSubmit}>Cadastrar</Button>
                    </Form>
                </CardBody>
            </Card>
        );
    }
}
```

webapp/src/pages/protected/admin/attachment/index.js

```
import List from './List';
export default List;
```

webapp/src/pages/protected/admin/attachment/ListItem.jsx

```
import React, { Component, Fragment } from 'react';
import PropTypes from 'prop-types';
import { deleteRequest } from '../../../../utils/http';
import { FaTrash } from 'react-icons/fa';
import Details from './Details.jsx';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            isOpen: false
        }
        this.handlerDelete = this.handlerDelete.bind(this);
        this.handlerDetails = this.handlerDetails.bind(this);
    }

    static propTypes = {
        attachment: PropTypes.object.isRequired,
        afterDelete: PropTypes.func.isRequired
    }

    handlerDelete(){
        const { attachment, afterDelete } = this.props;
        deleteRequest(
            `/attachment/${attachment.codAttachment}`,
            afterDelete
        )
    }
```

```
1768
1769        handlerDetails(){
1770            const { isOpen } = this.state;
1771            this.setState({ isOpen: !isOpen });
1772        }
1773
1774        render(){
1775            const { attachment } = this.props;
1776            const { isOpen } = this.state;
1777            return (
1778                <Fragment>
1779                    <tr>
1780                        <td onClick={this.handlerDetails} style={{
                                cursor: 'pointer' }}>
1781                            <strong>
1782                                {attachment.name}
1783                            </strong>
1784                        </td>
1785                        <td onClick={this.handlerDetails} style={{
                                cursor: 'pointer' }}>
1786                            <strong>
1787                                {attachment.required ? "Sim" : "No"}
1788                            </strong>
1789                        </td>
1790                        <td>
1791                            <FaTrash style={{ cursor: 'pointer' }}
                                    color="red"
                                    onClick={this.handlerDelete} />
1792                        </td>
1793                    </tr>
1794                    {
1795                        isOpen ? <Details attachment={attachment}
                                isOpen={isOpen}
                                close={this.handlerDetails}/> : null
1796                    }
1797                </Fragment>
1798            );
1799        }
1800    }
1801
1802    webapp/src/pages/protected/admin/attachment/Details.jsx
1803
1804    import React from 'react';
1805    import { Modal, ModalBody, ModalHeader, ModalFooter, Row, Col,
        Button } from 'reactstrap';
```

```
1806 import { Status } from '../../../../model';
1807
1808 export default ({ close, attachment, isOpen }) => attachment && (
1809     <Modal toggle={close} isOpen={isOpen}>
1810         <ModalHeader
                toggle={close}>{attachment.name}</ModalHeader>
1811         <ModalBody>
1812             <Row>
1813                 <Col>
1814                     <strong>
1815                         Obrigatrio :
1816                     </strong>
1817                 </Col>
1818                 <Col>
1819                     {attachment.required ? 'Sim': 'No'}
1820                 </Col>
1821             </Row>
1822             <Row>
1823                 <Col>
1824                     <strong>
1825                         Disponibilizado pela CnE:
1826                     </strong>
1827                 </Col>
1828                 <Col>
1829                     {attachment.download ? 'Sim': 'No'}
1830                 </Col>
1831             </Row>
1832             <Row>
1833                 <Col>
1834                     <strong>
1835                         Necessrio  envio durante:
1836                     </strong>
1837                 </Col>
1838                 <Col>
1839                     {Status.translate(attachment.status)}
1840                 </Col>
1841             </Row>
1842         </ModalBody>
1843         <ModalFooter>
1844             <Button color="default" type="button"
                    onClick={close}>Fechar</Button>
1845         </ModalFooter>
1846     </Modal>
1847 );
1848
```

webapp/src/pages/protected/admin/attachment/List.jsx

```
import React, { Component } from 'react';
import { getRequest } from '../../../../utils/http';
import { Row, Col, Table } from 'reactstrap';
import ListItem from './ListItem.jsx';
import Form from './Form.jsx';
import { Filter, Pagination } from '../../../../components';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            attachments: [],
            filter: '',
            count: 0,
            pages: 0
        };
    }

    componentWillMount(){
        getRequest(
            '/attachment/page/0',
            res => {
                const { data: attachments, count } = res.data
                this.setState({
                    pages: count / 20,
                    attachments,
                    count
                })
            }
        );
    }

    handlerFilter(event){
        const filter = event.target.value;
        if (filter.length >= 3 || filter.length === 0){
            getRequest(
                `/attachment/pages/0?filter=${filter}`,
                res => {
                    const { data: attachments, count} = res.data
                    this.setState({
                        pages: count/20,
                        attachments,
                        count,
```

```
                    filter
                })
            }
        );
    } else {
        this.setState({ filter })
    }
}

onChangePage(pageNumber) {
    const { filter } = this.state;
    getRequest(
        `/attachment/page/${pageNumber}?filter=${filter.lenght
            > 3 ? filter : ''}`,
        res => {
            const { data: attachments, count } = res.data;
            this.setState({
                pages: count / 20,
                attachments,
                count
            });
        }
    );
}

render(){
    const { attachments, filter, pages, count } = this.state;
    return (
        <div>
            <Row>
                <Col>
                    <h3>Anexos</h3>
                </Col>
            </Row>
            <br/>
            <Row>
                <Col>
                    <Table striped>
                        <thead>
                            <tr>
                                <th>
                                    <Filter
                                        label="Nome  "
                                        value={filter}
                                        handlerFilter={this.handlerFilter.bind(t
```

```
                                        />
                                    </th>
                                    <th>Obrigatrio</th>
                                    <th></th>
                                </tr>
                            </thead>
                            <tbody>
                                {
                                    attachments.length ?
                                        attachments.map(
                                            attachment => (
                                                <ListItem
                                                    key={attachment.codAttachme
                                                    attachment={attachment}
                                                    afterDelete={this.component
                                            )
                                        ) : (
                                            <tr>
                                                <td colSpan="3">
                                                    <strong>
                                                        Nenhum
                                                            Registro
                                                            encontrado
                                                    </strong>
                                                </td>
                                            </tr>
                                        )
                                }
                            </tbody>
                        </Table>
                        <Pagination pages={pages} count={count}
                            onChangePage={this.onChangePage.bind(this)}
                            />
                    </Col>
                    <Col md="4">
                        <Form
                            afterSubmit={this.componentWillMount.bind(this)},
                    </Col>
                </Row>
            </div>
        );
    }

}
```

webapp/src/pages/protected/admin/question/Form.jsx

```
import React, { Component } from 'react';
import { Question, Roles } from '../../../../model';
import PropTypes from 'prop-types';
import { postRequest } from '../../../../utils/http';
import { Form, Button, Card, CardBody } from 'reactstrap';
import { Input } from '../../../../components';

export default class extends Component{
    constructor() {
        super()
        this.state = {
            newQuestion: new Question(),
            invalid: false
        }
        this.handlerQuestion = this.handlerQuestion.bind(this);
        this.handlerResponds = this.handlerResponds.bind(this);
        this.handlerSubmit = this.handlerSubmit.bind(this);
    }

    static propTypes = {
        afterSubmit: PropTypes.func.isRequired
    }

    handlerQuestion(event) {
        const { newQuestion } = this.state;
        newQuestion.question = event.target.value;
        this.setState({ newQuestion, invalid: false });
    }

    handlerResponds(event) {
        const { newQuestion } = this.state;
        newQuestion.responds = event.target.value;
        this.setState({ newQuestion, invalid: false });
    }

    handlerSubmit() {
        const { newQuestion } = this.state;
        if (!newQuestion.question || !newQuestion.responds) {
            this.setState({ invalid: true })
            return;
        }
        const { afterSubmit } = this.props;
        postRequest(
```

```
            '/question',
            newQuestion,
            () =>{
                afterSubmit();
                this.setState({ newQuestion: new Question()});
            }
        );
    }

    render(){
        const { newQuestion } = this.state;
        return (
            <Card>
                <CardBody>
                    <Form>
                        <Input id="newQuestion" label="Questo"
                            invalidMessage="Campo Obrigatrio"
                            value={newQuestion.question}
                            invalid={this.state.invalid}
                            onChange={this.handlerQuestion}
                            required />
                        <Input id="destiny" type="select"
                            label="Destinada"
                            invalidMessage="Campo Obrigatrio"
                            value={newQuestion.responds}
                            invalid={this.state.invalid}
                            onChange={this.handlerResponds}
                            required>
                            <option>Selecione</option>
                            <option
                                value={Roles.VOLUNTEER}>{Roles.translate(Role
                            <option
                                value={Roles.COMPANY_ADMIN}>{Roles.translate(
                        </Input>
                        <Button type="button" color="primary"
                            className="float-right"
                            onClick={this.handlerSubmit}>Cadastrar</Button>
                    </Form>
                </CardBody>
            </Card>
        );
    }
}
```

webapp/src/pages/protected/admin/question/index.js

```
2046
2047 import List from './List';
2048 export default List;
2049
2050 webapp/src/pages/protected/admin/question/List.jsx
2051
2052 import React, { Component, Fragment } from 'react';
2053 import { Question, Roles } from '../../../../model';
2054 import { getRequest, deleteRequest } from
         '../../../../utils/http';
2055 import { Row, Col, Table} from 'reactstrap';
2056 import { FaTrash } from 'react-icons/fa';
2057 import Form from './Form.jsx';
2058
2059 export default class extends Component{
2060     constructor(){
2061         super()
2062         this.state = {
2063             questions: []
2064         }
2065         this.handlerDelete = this.handlerDelete.bind(this);
2066     }
2067
2068     componentWillMount(){
2069         getRequest('/question', res => this.setState({
              questions: res.data }));
2070     }
2071
2072     handlerDelete(codQuestion){
2073         deleteRequest(`/question/${codQuestion}`,
              this.componentWillMount);
2074     }
2075
2076     render(){
2077         const { questions } = this.state;
2078         return (
2079             <Fragment>
2080                 <Row>
2081                     <Col>
2082                         <h3> Questes </h3>
2083                     </Col>
2084                 </Row>
2085                 <br/>
2086                 <Row>
2087                     <Col>
```

```
<Table striped>
    <thead>
        <tr>
            <td>
                <strong>Questo</strong>
            </td>
            <td>
                <strong>Destinada</strong>
            </td>
            <td/>
        </tr>
    </thead>
    <tbody>
        {
            questions.length ?
            questions.map(
                question => (
                        <tr
                            key={question.codQuestion}>
                            <td>
                                <strong>{question.questi
                            </td>
                            <td>
                                <strong>{Roles.translate
                            </td>
                            <td>
                                <FaTrash
                                    style={{cursor:
                                    'pointer'}}
                                    color="red"
                                    onClick={()
                                    =>
                                    this.handlerDelete(q
                            </td>
                        </tr>
                    )
                ) : (
                    <tr>
                        <td colSpan="3">
                            Nenhuma Questo
                                cadastrada
                                Cadastrada
                        </td>
                    </tr>
                )
```

```
                                    }
                                </tbody>
                            </Table>
                        </Col>
                        <Col md="4">
                            <Form
                                afterSubmit={this.componentWillMount.bind(this)}
                                />
                        </Col>
                    </Row>
                </Fragment>

        );
    }
}
```

webapp/src/pages/protected/admin/project/Form.jsx

```
import React, { Component } from 'react';
import { Project } from '../../../../model';
import PropTypes from 'prop-types';
import { postRequest } from '../../../../utils/http';
import { Form, Button, Card, CardBody } from 'reactstrap';
import { Input, DatePicker } from '../../../../components';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            project: new Project(),
            invalid: false
        };
        this.handlerName = this.handlerName.bind(this);
        this.handlerStart = this.handlerStart.bind(this);
        this.handlerEnd = this.handlerEnd.bind(this);
        this.handlerSubmit = this.handlerSubmit.bind(this);
    }

    static propTypes = {
        afterSubmit: PropTypes.func.isRequired
    }

    handlerName(event) {
        const { project } = this.state;
        project.name = event.target.value;
```

```
2167            this.setState({ invalid: false, project });
2168        }
2169
2170        handlerStart(event) {
2171            const { project } = this.state;
2172            const { value } = event.target;
2173            project.start = value ? new Date(value) : null;
2174            this.setState({ invalid: false, project });
2175        }
2176
2177        handlerEnd(event) {
2178            const { project } = this.state;
2179            const { value } = event.target;
2180            project.end = value ? new Date(value) : null;
2181            this.setState({ invalid: false, project });
2182        }
2183
2184        handlerSubmit() {
2185            const { project } = this.state;
2186            if (!project.name || !project.start || !project.end) {
2187                this.setState({ invalid: true })
2188                return;
2189            }
2190            const { afterSubmit } =this.props;
2191            postRequest(
2192                '/project',
2193                project,
2194                () => {
2195                    afterSubmit();
2196                    this.setState({ project: new Project() });
2197                }
2198            );
2199        }
2200        render(){
2201            const { project, invalid } = this.state;
2202            const { start } = project;
2203            const minDateStart = new Date().toJSON().split('T')[0];
2204            const minDateEnd = start ? start.toJSON().split('T')[0]
                    : minDateStart;
2205            return (
2206                <Card>
2207                    <CardBody>
2208                        <Form>
2209                            <Input id="name" label="Nome do Projeto"
                                    onChange={this.handlerName}
```

```
                            invalidMessage="Nome do Projeto
                            Obrigatrio" value={project.name}
                            invalid={invalid} required />
                        <DatePicker id="start" label="Incio do
                            Projeto" invalidMessage="Incio do
                            Projeto  obrigatrio"
                            min={minDateStart}
                            onChange={this.handlerStart}
                            invalid={invalid} required />
                        <DatePicker id="end" label="Final do
                            Projeto" invalidMessage="Final do
                            Projeto  obrigatrio" min={minDateEnd}
                            onChange={this.handlerEnd}
                            invalid={invalid} required />
                        <Button type="button" color="primary"
                            className="float-right"
                            onClick={this.handlerSubmit}>Cadastrar</Button>
                    </Form>
                </CardBody>
            </Card>
        );
    }
}
```

webapp/src/pages/protected/admin/project/index.js

```
import List from './List';
export default List;
```

webapp/src/pages/protected/admin/project/ListItem.jsx

```
import React, { Component, Fragment } from 'react';
import PropTypes from 'prop-types';
import { deleteRequest } from '../../../../utils/http';
import { FaTrash } from 'react-icons/fa';

export default class extends Component {

    static propTypes = {
        typeEvent: PropTypes.object.isRequired,
        afterDelete: PropTypes.func.isRequired
    }

    handlerDelete() {
        const { project, afterDelete } = this.props;
```

```
2241            deleteRequest(
2242                `/project/${project.codProject}`,
2243                afterDelete
2244            )
2245        }
2246
2247        render() {
2248            const { project } = this.props;
2249            return (
2250                <tr>
2251                    <td>
2252                        <strong>
2253                            {project.name}
2254                        </strong>
2255                    </td>
2256                    <td>
2257                        <strong>
2258                            {new
                                Date(project.start).toLocaleDateString()}
2259                        </strong>
2260                    </td>
2261                    <td>
2262                        <strong>
2263                            {new
                                Date(project.end).toLocaleDateString()}
2264                        </strong>
2265                    </td>
2266                    <td>
2267                        <FaTrash style={{ cursor: 'pointer' }}
                             color="red"
                             onClick={this.handlerDelete.bind(this)} />
2268                    </td>
2269                </tr>
2270            );
2271        }
2272    }
2273
2274    webapp/src/pages/protected/admin/project/List.jsx
2275
2276    import React, { Component, Fragment } from 'react';
2277    import { getRequest } from '../../../../utils/http';
2278    import { Row, Col, Table } from 'reactstrap';
2279    import ListItem from './ListItem.jsx';
2280    import Form from './Form.jsx';
2281    import { Filter, Pagination } from '../../../../components';
```

```
2282
2283 export default class extends Component{
2284     constructor(){
2285         super();
2286         this.state ={
2287             projects: [],
2288             filter: '',
2289             count: 0,
2290             pages: 0
2291         };
2292     }
2293
2294     componentWillMount(){
2295         getRequest(
2296             '/project/page/0',
2297             res => {
2298                 const { data: projects, count } = res.data
2299                 this.setState({
2300                     pages: count / 20,
2301                     projects,
2302                     count
2303                 });
2304             }
2305         );
2306     }
2307
2308     handlerFilter(event) {
2309         const filter = event.target.value;
2310         if (filter.length >= 3 || filter.length === 0) {
2311             getRequest(
2312                 `/project/page/0?filter=${filter}`,
2313                 res => {
2314                     const { data: projects, count } = res.data
2315                     this.setState({
2316                         pages: count / 20,
2317                         projects,
2318                         count,
2319                         filter
2320                     })
2321                 }
2322             );
2323         } else {
2324             this.setState({ filter })
2325         }
2326     }
```

```
2327
2328        onChangePage(pageNumber) {
2329            const { filter } = this.state;
2330            getRequest(
2331                `/project/page/${pageNumber}?filter=${filter.lenght >
                        3 ? filter : ''}`,
2332                res => {
2333                    const { data: projects, count } = res.data;
2334                    this.setState({
2335                        pages: count / 20,
2336                        projects,
2337                        count
2338                    });
2339                }
2340            );
2341        }
2342
2343        render(){
2344            const { projects, filter, count, pages } = this.state;
2345            return (
2346                <Fragment>
2347                    <Row>
2348                        <Col>
2349                            <h3>Projetos</h3>
2350                        </Col>
2351                    </Row>
2352                    <br/>
2353                    <Row>
2354                        <Col>
2355                            <Table striped>
2356                                <thead>
2357                                    <tr>
2358                                        <th>
2359                                            <Filter
                                                    label="Nome  "
                                                    value={filter}
                                                    handlerFilter={this.handlerFilter.
                                                    />
2360                                        </th>
2361                                        <th> Incio </th>
2362                                        <th>Fim</th>
2363                                        <th></th>
2364                                    </tr>
2365                                </thead>
2366                                <tbody>
```

```
                                {
                                    projects.length ?
                                        projects.map(project =>
                                            <ListItem
                                            key={project.codProject}
                                            project={project}
                                            afterDelete={this.componentWillMount}
                                            />):
                                        <tr>
                                            <td colSpan="4">
                                                <strong>No  h
                                                    Projetos
                                                    registrados</strong>
                                            </td>
                                        </tr>
                                }
                            </tbody>
                        </Table>
                        <Pagination pages={pages} count={count}
                            onChangePage={this.onChangePage.bind(this)}
                            />
                    </Col>
                    <Col md="4">
                        <Form
                            afterSubmit={this.componentWillMount.bind(this)}
                            />
                    </Col>
                </Row>
            </Fragment>
        )
    }
}

webapp/src/pages/protected/admin/evaluate/skill/Form.jsx

import React, { Component } from 'react';
import { Card, CardBody, Form, Button } from 'reactstrap';
import { Input } from '../../../../../components';
import { postRequest } from '../../../../../utils/http';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            value : '',
```

```
2401            invalid: false
2402        };
2403        this.handlerSkill = this.handlerSkill.bind(this);
2404        this.handlerSubmit = this.handlerSubmit.bind(this);
2405    }
2406
2407    handlerSkill(event){
2408        this.setState({ value: event.target.value, invalid:
                false });
2409    }
2410
2411    handlerSubmit(){
2412        const { value } = this.state;
2413        if(!value){
2414            this.setState({invalid: true})
2415        }
2416        const { afterSubmit } = this.props;
2417        postRequest(
2418            '/skill',
2419            { name: value },
2420            () => {
2421                this.setState({value: ''});
2422                afterSubmit();
2423            }
2424        );
2425    }
2426
2427    render(){
2428        return (
2429            <Card>
2430                <CardBody>
2431                    <Form>
2432                        <Input id="skill" label="Competncia"
                                {...this.state} invalidMessage="Campo
                                Obrigatrio"
                                onChange={this.handlerSkill} required
                                />
2433                        <Button type="button" color="primary"
                                className="float-right"
                                onClick={this.handlerSubmit}>Cadastrar</Button>
2434                    </Form>
2435                </CardBody>
2436            </Card>
2437        )
2438    }
```

```
}

webapp/src/pages/protected/admin/evaluate/skill/index.js

import List from './List';
export default List;

webapp/src/pages/protected/admin/evaluate/skill/List.jsx

import React, { Component, Fragment } from 'react';
import { getRequest, deleteRequest } from
    '../../../../../utils/http';
import { Col, Row, ListGroup, ListGroupItem } from 'reactstrap';
import { FaTrash } from 'react-icons/fa';
import Form from './Form.jsx';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            skills: []
        };
        this.handlerDelete = this.handlerDelete.bind(this);
        this.componentWillMount =
            this.componentWillMount.bind(this);
    }

    componentWillMount(){
        getRequest('/skill', res => this.setState({ skills:
            res.data }));
    }

    handlerDelete(codSkill){
        deleteRequest(`/skill/${codSkill}`,
            this.componentWillMount);
    }

    render(){
        const { skills } = this.state;
        return (
            <Fragment>
                <Row>
                    <Col>
                        <h3>Competncias</h3>
                    </Col>
```

```
2480                </Row>
2481                <br/>
2482                <Row>
2483                    <Col>
2484                        <ListGroup>
2485                            {
2486                                skills.length ?
2487                                    skills.map(
2488                                        skill =>
2489                                            (
2490                                                <ListGroupItem
                                                        key={skill.codSkill}>
2491                                                <Row>
2492                                                    <Col>
2493                                                        <strong>{skill.name}
2494                                                    </Col>
2495                                                    <Col md="1">
2496                                                        <FaTrash
                                                                style={{
                                                                cursor:
                                                                'pointer'
                                                                }}
                                                                color="red"
                                                                onClick={()
                                                                =>
                                                                this.handlerDele
2497                                                    </Col>
2498                                                </Row>
2499                                            </ListGroupItem>
2500                                        )
2501                                ) : (
2502                                    <ListGroupItem>
2503                                        <strong>Nenhuma questo
                                                sobre a Competncia
                                                encontrada</strong>
2504                                    </ListGroupItem>
2505                                )
2506
2507                            }
2508                        </ListGroup>
2509                    </Col>
2510                    <Col md="4">
2511                        <Form
                                afterSubmit={this.componentWillMount}/>
2512                    </Col>
```

```
                </Row>
            </Fragment>
        );
    }
}

webapp/src/pages/protected/admin/evaluate/technology/Form.jsx

import React, { Component } from 'react';
import { Card, CardBody, Form, Button } from 'reactstrap';
import { Input } from '../../../../../components';
import { postRequest } from '../../../../../utils/http';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            value : '',
            invalid: false
        };
        this.handlerSkill = this.handlerSkill.bind(this);
        this.handlerSubmit = this.handlerSubmit.bind(this);
    }

    handlerSkill(event){
        this.setState({ value: event.target.value, invalid:
            false });
    }

    handlerSubmit(){
        const { value } = this.state;
        if(!value){
            this.setState({invalid: true})
        }
        const { afterSubmit } = this.props;
        postRequest(
            '/technology',
            { name: value },
            () => {
                this.setState({value: ''});
                afterSubmit();
            }
        );
    }
```

```
2557      render(){
2558          return (
2559              <Card>
2560                  <CardBody>
2561                      <Form>
2562                          <Input id="skill" label="Conhecimento"
                                {...this.state} invalidMessage="Campo
                                Obrigatrio"
                                onChange={this.handlerSkill} required
                                />
2563                          <Button type="button" color="primary"
                                className="float-right"
                                onClick={this.handlerSubmit}>Cadastrar</Button>
2564                      </Form>
2565                  </CardBody>
2566              </Card>
2567          )
2568      }
2569  }
2570
2571  webapp/src/pages/protected/admin/evaluate/technology/index.js
2572
2573  import List from './List';
2574  export default List;
2575
2576  webapp/src/pages/protected/admin/evaluate/technology/List.jsx
2577
2578  import React, { Component, Fragment } from 'react';
2579  import { getRequest, deleteRequest } from
        '../../../../../utils/http';
2580  import { Col, Row, ListGroup, ListGroupItem } from 'reactstrap';
2581  import { FaTrash } from 'react-icons/fa';
2582  import Form from './Form.jsx';
2583
2584  export default class extends Component{
2585      constructor(){
2586          super();
2587          this.state = {
2588              technologies: []
2589          };
2590          this.handlerDelete = this.handlerDelete.bind(this);
2591          this.componentWillMount =
                 this.componentWillMount.bind(this);
2592      }
2593
```

```
2594    componentWillMount(){
2595        getRequest('/technology', res => this.setState({
                technologies: res.data }));
2596    }
2597
2598    handlerDelete(codTechnology){
2599        deleteRequest(`/technology/${codTechnology}`,
                this.componentWillMount);
2600    }
2601
2602    render(){
2603        const { technologies } = this.state;
2604        return (
2605            <Fragment>
2606                <Row>
2607                    <Col>
2608                        <h3>Competncias</h3>
2609                    </Col>
2610                </Row>
2611                <br/>
2612                <Row>
2613                    <Col>
2614                        <ListGroup>
2615                            {
2616                                technologies.length ?
2617                                    technologies.map(
2618                                        technology =>
2619                                            (
2620                                                <ListGroupItem
                                                        key={technology.codTechnology}>
2621                                                <Row>
2622                                                    <Col>
2623                                                        <strong>{technology.name}<
2624                                                    </Col>
2625                                                    <Col md="1">
2626                                                        <FaTrash
                                                                style={{
                                                                cursor:
                                                                'pointer'
                                                                }}
                                                                color="red"
                                                                onClick={()
                                                                =>
                                                                this.handlerDelete(tec
2627                                                    </Col>
```

```
                                                    </Row>
                                                </ListGroupItem>
                                            )
                                        ) : (
                                            <ListGroupItem>
                                                <strong>Nenhum tipo de
                                                    Conhecimento
                                                    encontrado</strong>
                                            </ListGroupItem>
                                        )

                                    }
                                </ListGroup>
                            </Col>
                            <Col md="4">
                                <Form
                                    afterSubmit={this.componentWillMount}/>
                            </Col>
                        </Row>
                    </Fragment>
                );
            }
        }
```

webapp/src/pages/protected/admin/evaluate/personality/Form.jsx

```
import React, { Component } from 'react';
import { Card, CardBody, Form, Button } from 'reactstrap';
import { Input } from '../../../../../components';
import { postRequest } from '../../../../../utils/http';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            value : '',
            invalid: false
        };
        this.handlerQuestion = this.handlerQuestion.bind(this);
        this.handlerSubmit = this.handlerSubmit.bind(this);
    }

    handlerQuestion(event){
        this.setState({ value: event.target.value, invalid:
            false });
```

```
    }

    handlerSubmit(){
        const { value } = this.state;
        if(!value){
            this.setState({invalid: true})
        }
        const { afterSubmit } = this.props;
        postRequest(
            '/personality',
            { question: value },
            () => {
                this.setState({value: ''});
                afterSubmit();
            }
        );
    }

    render(){
        return (
            <Card>
                <CardBody>
                    <Form>
                        <Input id="personality" label="Questo"
                            {...this.state } invalidMessage="Campo
                            Obrigatrio"
                            onChange={this.handlerQuestion}
                            required />
                        <Button type="button" color="primary"
                            className="float-right"
                            onClick={this.handlerSubmit}>Cadastrar</Button>
                    </Form>
                </CardBody>
            </Card>
        )
    }
}
```

webapp/src/pages/protected/admin/evaluate/personality/index.js

```
import List from './List';
export default List;
```

webapp/src/pages/protected/admin/evaluate/personality/List.jsx

```
2708 import React, { Component, Fragment } from 'react';
2709 import { getRequest, deleteRequest } from
         '../../../../../utils/http';
2710 import { Col, Row, ListGroup, ListGroupItem } from 'reactstrap';
2711 import { FaTrash } from 'react-icons/fa';
2712 import Form from './Form.jsx';
2713
2714 export default class extends Component{
2715     constructor(){
2716         super();
2717         this.state = {
2718             personalities: []
2719         };
2720         this.handlerDelete = this.handlerDelete.bind(this);
2721         this.componentWillMount =
              this.componentWillMount.bind(this);
2722     }
2723
2724     componentWillMount(){
2725         getRequest('/personality', res => this.setState({
              personalities: res.data }));
2726     }
2727
2728     handlerDelete(codQuestion){
2729         deleteRequest(`/personality/${codQuestion}`,
              this.componentWillMount);
2730     }
2731
2732     render(){
2733         const { personalities } = this.state;
2734         return (
2735             <Fragment>
2736                 <Row>
2737                     <Col>
2738                         <h3>Personalidade</h3>
2739                     </Col>
2740                 </Row>
2741                 <br/>
2742                 <Row>
2743                     <Col>
2744                         <ListGroup>
2745                             {
2746                                 personalities.length ?
2747                                     personalities.map(
2748                                         personality =>
```

```
                                        (
                                            <ListGroupItem
                                                key={personality.codQuestion}>
                                                <Row>
                                                    <Col>
                                                        <strong>{personality.quest
                                                    </Col>
                                                    <Col md="1">
                                                        <FaTrash
                                                            style={{cursor:
                                                            'pointer'}}
                                                            color="red"
                                                            onClick={()
                                                            =>
                                                            this.handlerDelete(per
                                                    </Col>
                                                </Row>
                                            </ListGroupItem>
                                        )
                                    ) : (
                                        <ListGroupItem>
                                            <strong>Nenhuma questo
                                                sobre a
                                                personalidade do
                                                voluntrio
                                                encontrada</strong>
                                        </ListGroupItem>
                                    )

                                }
                            </ListGroup>
                        </Col>
                        <Col md="4">
                            <Form
                                afterSubmit={this.componentWillMount}/>
                        </Col>
                    </Row>
                </Fragment>
            );
        }
    }

webapp/src/pages/protected/admin/evaluate/task/Form.jsx

import React, { Component } from 'react';
```

```
2782 import { Card, CardBody, Form, Button } from 'reactstrap';
2783 import { Input } from '../../../../../components';
2784 import { postRequest } from '../../../../../utils/http';
2785
2786 export default class extends Component{
2787     constructor(){
2788         super();
2789         this.state = {
2790             value : '',
2791             invalid: false
2792         };
2793         this.handlerQuestion = this.handlerQuestion.bind(this);
2794         this.handlerSubmit = this.handlerSubmit.bind(this);
2795     }
2796
2797     handlerQuestion(event){
2798         this.setState({ value: event.target.value, invalid:
                 false });
2799     }
2800
2801     handlerSubmit(){
2802         const { value } = this.state;
2803         if(!value){
2804             this.setState({invalid: true})
2805         }
2806         const { afterSubmit } = this.props;
2807         postRequest(
2808             '/task',
2809             { task: value },
2810             () => {
2811                 this.setState({value: ''});
2812                 afterSubmit();
2813             }
2814         );
2815     }
2816
2817     render(){
2818         return (
2819             <Card>
2820                 <CardBody>
2821                     <Form>
2822                         <Input id="mentoring" label="Questo"
                                 {...this.state } invalidMessage="Campo
                                 Obrigatrio"
                                 onChange={this.handlerQuestion}
```

```
                            required />
                        <Button type="button" color="primary"
                            className="float-right"
                            onClick={this.handlerSubmit}>Cadastrar</Button>
                    </Form>
                </CardBody>
            </Card>
        )
    }
}
```

webapp/src/pages/protected/admin/evaluate/task/index.js

```
import List from './List';
export default List;
```

webapp/src/pages/protected/admin/evaluate/task/List.jsx

```
import React, { Component, Fragment } from 'react';
import { getRequest, deleteRequest } from
    '../../../../../utils/http';
import { Col, Row, ListGroup, ListGroupItem } from 'reactstrap';
import { FaTrash } from 'react-icons/fa';
import Form from './Form.jsx';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            tasks: []
        };
        this.handlerDelete = this.handlerDelete.bind(this);
        this.componentWillMount =
            this.componentWillMount.bind(this);
    }

    componentWillMount(){
        getRequest('/task', res => this.setState({
            personalities: res.data }));
    }

    handlerDelete(codTask){
        deleteRequest(`/task/${codTask}`,
            this.componentWillMount);
    }
```

```
2861
2862        render(){
2863            const { tasks } = this.state;
2864            return (
2865                <Fragment>
2866                    <Row>
2867                        <Col>
2868                            <h3>Feedback do Voluntrio</h3>
2869                        </Col>
2870                    </Row>
2871                    <br/>
2872                    <Row>
2873                        <Col>
2874                            <ListGroup>
2875                                {
2876                                    tasks.length ?
2877                                        tasks.map(
2878                                            task =>
2879                                                (
2880                                                    <ListGroupItem
                                                        key={task.codTask}>
2881                                                    <Row>
2882                                                        <Col>
2883                                                            <strong>{task.questi
2884                                                        </Col>
2885                                                        <Col md="1">
2886                                                            <FaTrash
                                                                style={{
                                                                cursor:
                                                                'pointer'
                                                                }}
                                                                color="red"
                                                                onClick={()
                                                                =>
                                                                this.handlerDele
2887                                                        </Col>
2888                                                    </Row>
2889                                                </ListGroupItem>
2890                                                )
2891                                    ) : (
2892                                        <ListGroupItem>
2893                                            <strong>Nenhuma questo
                                                sobre a
                                                personalidade do
                                                voluntrio
```

```
                                                encontrada</strong>
                                        </ListGroupItem>
                                    )

                                }
                            </ListGroup>
                        </Col>
                        <Col md="4">
                            <Form
                                afterSubmit={this.componentWillMount}/>
                        </Col>
                    </Row>
                </Fragment>
            );
        }
    }

webapp/src/pages/protected/admin/evaluate/mentoring/Form.jsx

    import React, { Component } from 'react';
    import { Card, CardBody, Form, Button } from 'reactstrap';
    import { Input } from '../../../../../components';
    import { postRequest } from '../../../../../utils/http';

    export default class extends Component{
        constructor(){
            super();
            this.state = {
                value : '',
                invalid: false
            };
            this.handlerQuestion = this.handlerQuestion.bind(this);
            this.handlerSubmit = this.handlerSubmit.bind(this);
        }

        handlerQuestion(event){
            this.setState({ value: event.target.value, invalid:
                false });
        }

        handlerSubmit(){
            const { value } = this.state;
            if(!value){
                this.setState({invalid: true})
            }
```

```
2936            const { afterSubmit } = this.props;
2937            postRequest(
2938                '/mentoring',
2939                { question: value },
2940                () => {
2941                    this.setState({value: ''});
2942                    afterSubmit();
2943                }
2944            );
2945        }
2946
2947        render(){
2948            return (
2949                <Card>
2950                    <CardBody>
2951                        <Form>
2952                            <Input id="mentoring" label="Questo"
                                    {...this.state } invalidMessage="Campo
                                    Obrigatrio"
                                    onChange={this.handlerQuestion}
                                    required />
2953                            <Button type="button" color="primary"
                                    className="float-right"
                                    onClick={this.handlerSubmit}>Cadastrar</Button>
2954                        </Form>
2955                    </CardBody>
2956                </Card>
2957            )
2958        }
2959    }
2960
2961    webapp/src/pages/protected/admin/evaluate/mentoring/index.js
2962
2963    import List from './List';
2964    export default List;
2965
2966    webapp/src/pages/protected/admin/evaluate/mentoring/List.jsx
2967
2968    import React, { Component, Fragment } from 'react';
2969    import { getRequest, deleteRequest } from
        '../../../../../utils/http';
2970    import { Col, Row, ListGroup, ListGroupItem } from 'reactstrap';
2971    import { FaTrash } from 'react-icons/fa';
2972    import Form from './Form.jsx';
2973
```

```
2974 export default class extends Component{
2975     constructor(){
2976         super();
2977         this.state = {
2978            mentorings: []
2979         };
2980         this.handlerDelete = this.handlerDelete.bind(this);
2981         this.componentWillMount =
                 this.componentWillMount.bind(this);
2982     }
2983
2984     componentWillMount(){
2985         getRequest('/mentoring', res => this.setState({
                 personalities: res.data }));
2986     }
2987
2988     handlerDelete(codQuestion){
2989         deleteRequest(`/mentoring/${codQuestion}`,
                 this.componentWillMount);
2990     }
2991
2992     render(){
2993         const { mentorings } = this.state;
2994         return (
2995             <Fragment>
2996                 <Row>
2997                     <Col>
2998                         <h3>Feedback do Voluntrio</h3>
2999                     </Col>
3000                 </Row>
3001                 <br/>
3002                 <Row>
3003                     <Col>
3004                         <ListGroup>
3005                             {
3006                                 mentorings.length ?
3007                                     mentorings.map(
3008                                         mentoring =>
3009                                             (
3010                                                 <ListGroupItem
                                                        key={mentoring.codQuestion}>
3011                                                 <Row>
3012                                                     <Col>
3013                                                         <strong>{mentoring.questio
3014                                                     </Col>
```

```
3015                                                    <Col md="1">
3016                                                        <FaTrash
                                                                style={{
                                                                cursor:
                                                                'pointer'
                                                                }}
                                                                color="red"
                                                                onClick={()
                                                                =>
                                                                this.handlerDele
3017                                                    </Col>
3018                                                </Row>
3019                                            </ListGroupItem>
3020                                        )
3021                                    ) : (
3022                                        <ListGroupItem>
3023                                            <strong>Nenhuma questo
                                                    sobre a
                                                    personalidade do
                                                    voluntrio
                                                    encontrada</strong>
3024                                        </ListGroupItem>
3025                                    )
3026
3027                                }
3028                            </ListGroup>
3029                        </Col>
3030                        <Col md="4">
3031                            <Form
                                    afterSubmit={this.componentWillMount}/>
3032                        </Col>
3033                    </Row>
3034                </Fragment>
3035            );
3036        }
3037    }
3038
3039    webapp/src/pages/protected/admin/evaluate/Page.jsx
3040
3041    import React, { Component } from 'react';
3042    import { Row, Col, Button, Collapse } from 'reactstrap';
3043    import ListPersonality from './personality';
3044    import ListSkill from './skill';
3045    import ListTechnology from './technology';
3046    import ListMentoring from './mentoring';
```

```
3047
3048 const styleButton = {
3049     width: '100%'
3050 };
3051
3052 export default class extends Component {
3053     constructor() {
3054         super();
3055         this.state = {
3056             show: {
3057                 knowledge: false,
3058                 personality: false,
3059                 skill: false,
3060                 mentoring: false,
3061                 others: false
3062             }
3063         };
3064     }
3065
3066     changeShow(collapse) {
3067         const { show } = this.state;
3068         for (const key in show) {
3069             if (show.hasOwnProperty(key)) {
3070                 show[key] = key === collapse ? !show[key] : false;
3071             }
3072         }
3073         this.setState({ show })
3074     }
3075
3076     render() {
3077         const { role } = this.props;
3078         const { show } = this.state;
3079         return (
3080             <Row>
3081                 <Col>
3082                     <Row>
3083                         <Col>
3084                             <Button type="button" color="info"
                                        style={styleButton} onClick={() =>
                                        this.changeShow('knowledge')}>Conhecimento</Button>
3085                         </Col>
3086                         <Col md="1" />
3087                         <Col>
3088                             <Button type="button" color="info"
                                        style={styleButton} onClick={() =>
```

```
                                this.changeShow('personality')}>Personalidade
                                do voluntrio</Button>
3089                        </Col>
3090                        <Col md="1" />
3091                        <Col>
3092                            <Button type="button" color="info"
                                    style={styleButton} onClick={() =>
                                    this.changeShow('skill')}>Competncias
                                    Desenvolvidas</Button>
3093                        </Col>
3094                        <Col md="1" />
3095                        <Col>
3096                            <Button type="button" color="info"
                                    style={styleButton} onClick={() =>
                                    this.changeShow('mentoring')}>Feedback
                                    do Voluntrio</Button>
3097                        </Col>
3098                        <Col md="1" />
3099                        <Col>
3100                            <Button type="button" color="info"
                                    style={styleButton} onClick={() =>
                                    this.changeShow('others')}>Avaliao</Button>
3101                        </Col>
3102                    </Row>
3103                    <br />
3104                    <Row>
3105                        <Col>
3106                            <Collapse isOpen={show.knowledge}>
3107                                <ListTechnology/>
3108                            </Collapse>
3109                            <Collapse isOpen={show.personality}>
3110                                <ListPersonality/>
3111                            </Collapse>
3112                            <Collapse isOpen={show.skill}>
3113                                <ListSkill/>
3114                            </Collapse>
3115                            <Collapse isOpen={show.mentoring}>
3116                                <ListMentoring/>
3117                            </Collapse>
3118                            <Collapse isOpen={show.others}>
3119                                teste
3120                            </Collapse>
3121                        </Col>
3122                    </Row>
3123                </Col>
```

```
            </Row>
        );
    }
}

webapp/src/pages/protected/admin/evaluate/index.js

import Page from './Page.jsx';
export default Page;

webapp/src/pages/protected/admin/eventType/Form.jsx

import React, { Component, Fragment } from 'react';
import { EventType, Training } from '../../../../model';
import PropTypes from 'prop-types';
import { postRequest } from '../../../../utils/http';
import { Row, Col, Button, Form, Card, CardBody } from
    'reactstrap';
import { Input, FileInput, Switch } from
    '../../../../components';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            typeEvent: new EventType(),
            files: []
        };
        this.handlerName = this.handlerName.bind(this);
        this.handlerAdd = this.handlerAdd.bind(this);
        this.handlerSubmit = this.handlerSubmit.bind(this);
    }

    static propTypes = {
        afterSubmit: PropTypes.func.isRequired,
    }

    handlerName(event){
        const { typeEvent } = this.state;
        typeEvent.name = event.target.value;
        this.setState({ typeEvent });
    }

    handlerAdd() {
        const { typeEvent } = this.state;
```

```
3167            let training = new Training();
3168            training.isFile = false;
3169            typeEvent.trainings.push(training);
3170            this.setState({ typeEvent })
3171        }
3172
3173        handlerNameAttachemnt(event, index) {
3174            const { typeEvent } = this.state;
3175            typeEvent.trainings[index].name = event.target.value;
3176            /*const training = typeEvent.trainings[index];
3177            training.name = event.target.value;
3178            if(training.isFile && files[index]){
3179                this.upload()
3180            } else{
3181                this.setState({ typeEvent });
3182            }*/
3183            this.setState({ typeEvent });
3184        }
3185
3186        handlerLinkAttachemnt(event, index) {
3187            const { typeEvent } = this.state;
3188            typeEvent.trainings[index].link = event.target.value;
3189            this.setState({ typeEvent });
3190        }
3191
3192        handlerUploadAttachemnt(event, index) {
3193            const file = event.target.files[0];
3194            console.log(file);
3195
3196        }
3197
3198        /*upload(index){
3199            a
3200        }*/
3201
3202        handlerRemove(index) {
3203            const { typeEvent } = this.state;
3204            typeEvent.trainings.splice(index, 1);
3205            this.setState({ typeEvent });
3206        }
3207
3208        handlerIsFile(value, index) {
3209            const { typeEvent } = this.state;
3210            typeEvent.trainings[index].isFile = value
3211            this.setState({ typeEvent });
```

```
3212        }
3213
3214        handlerSubmit(){
3215            const { afterSubmit } = this.props;
3216            const { typeEvent } = this.state;
3217            /*const json = JSON.stringify(typeEvent);
3218            const blob = new Blob([json], {
3219                type: 'application/json'
3220            });
3221            let form = new FormData();
3222            form.append('typeEvent', blob);
3223            if(file){
3224                form.append('file', file);
3225            }*/
3226            postRequest('/typeEvent', typeEvent, () => {
3227                afterSubmit();
3228                this.setState({ typeEvent: new EventType() });
3229            })
3230        }
3231
3232        render(){
3233            const { typeEvent } = this.state;
3234            const { trainings } = typeEvent;
3235            return (
3236                <Card>
3237                    <CardBody>
3238                        <Form>
3239                            <Row>
3240                                <Col>
3241                                    <Input id="name" label="Tipo de
                                            Evento"
                                            onChange={this.handlerName}
                                            invalidMessage="Tipo de Evento
                                            Obrigatrio"
                                            value={typeEvent.type}
                                            required/>
3242                                </Col>
3243                            </Row>
3244                            <Row>
3245                                <Col>
3246                                    <h4>Materiais</h4>
3247                                </Col>
3248                            </Row>
3249                            {
3250                                trainings.length ?
```

```
3251                        trainings.map(
3252                            (training, index) => (
3253                                <Fragment
                                        key={`training-${index}`}>
3254                                    <Row>
3255                                        <Col>
3256                                            <Input
                                                    id={`training-${index}`
                                                    label="Nome"
                                                    invalidMessage="Nome
                                                     Obrigatrio"
                                                    value={training.name}
                                                    onChange={event
                                                    =>
                                                    this.handlerNameAttache
                                                    index)}
                                                    required />
3257                                        </Col>
3258                                        <Col md="5">
3259                                            <Switch
                                                    id={`file-${index}`}
                                                    label="Arquivo
                                                    pra Upload"
                                                    value={training.isFile}
                                                    onChange={value
                                                    =>
                                                    this.handlerIsFile(valu
                                                    index)} />
3260                                        </Col>
3261                                    </Row>
3262                                    <Row>
3263                                        <Col>
3264                                            {
3265                                                training.isFile
                                                        ?
3266                                                        (
3267                                                            <FileInput
                                                                    id={`upload-
                                                                    label="Anexa
                                                                    arquivo"
                                                                    invalidMessa

                                                                    Obrigatrio"
                                                                    onChange={ev
                                                                    =>
```

```
                                                this.handlerUpload/
                                                index)}
                                                accept="applicatio
                                                required
                                                />
3268                                        ) : (
3269                                            <Input
                                                    id={`link-${index}
                                                    label="Link"
                                                    invalidMessage="Li

                                                    Obrigatrio"
                                                    value={training.li
                                                    onChange={event
                                                    =>
                                                    this.handlerLinkAt
                                                    index)}
                                                    required
                                                    />
3270                                        )
3271                                    }
3272                                </Col>
3273                                <Col md="3">
3274                                    {!training.isFile
                                            &&
                                            <label> </label>}
3275                                    <Button outline
                                            color="secondary"
                                            onClick={()
                                            =>
                                            this.handlerRemove(index)}>Re
3276                                </Col>
3277                            </Row>
3278                        </Fragment>
3279                    )
3280                ): null
3281            }
3282            <Row>
3283                <Col>
3284                    <Button outline color="secondary"
                            onClick={this.handlerAdd}>Incluir</Button>
3285                </Col>
3286            </Row>
3287            <br/>
3288            <Row>
```

```
                            <Col>
                                <Button type="button"
                                    color="primary"
                                    className="float-right"
                                    onClick={this.handlerSubmit}>Cadastrar</B
                            </Col>
                        </Row>
                    </Form>
                </CardBody>
            </Card>
        );
    }
}
```

webapp/src/pages/protected/admin/eventType/index.js

```
import List from './List';
export default List;
```

webapp/src/pages/protected/admin/eventType/ListItem.jsx

```
import React, { Component, Fragment } from 'react';
import PropTypes from 'prop-types';
import { deleteRequest } from '../../../../utils/http';
import { FaTrash } from 'react-icons/fa';
import Trainings from './Trainings.jsx';

export default class extends Component{

    static propTypes = {
        typeEvent: PropTypes.object.isRequired,
        afterDelete: PropTypes.func.isRequired
    }

    handlerDelete(){
        const { typeEvent, afterDelete } = this.props;
        deleteRequest(
            `/typeEvent/${typeEvent.type}`,
            afterDelete
        )
    }

    render(){
        const { typeEvent } = this.props;
        return (
```

```
            <Fragment>
                <tr>
                    <td>
                        <strong>
                            {typeEvent.name}
                        </strong>
                    </td>
                    <td>
                        <Trainings type={typeEvent.type}/>
                    </td>
                    <td>
                        <FaTrash style={{ cursor: 'pointer' }}
                            color="red"
                            onClick={this.handlerDelete.bind(this)}
                            />
                    </td>
                </tr>
            </Fragment>
        );
    }
}
```

webapp/src/pages/protected/admin/eventType/Trainings.jsx

```
import React, { Component, Fragment } from 'react';
import PropTypes from 'prop-types';
import { Button, Tooltip } from 'reactstrap';
import { getRequest } from '../../../../utils/http';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            isOpen: false,
            trainings: []
        };
        this.handlerTrainings = this.handlerTrainings.bind(this);
    }

    static propTypes = {
        type: PropTypes.number.isRequired
    }

    handlerTrainings(){
        const { isOpen } = this.state;
```

```
3373         const { type } = this.props;
3374         if (!isOpen){
3375             getRequest(
3376                 `/typeEvent/${type}/trainings`,
3377                 res => this.setState({ isOpen: !isOpen,
                        trainings: res.data })
3378             );
3379         } else {
3380             this.setState({ isOpen: !isOpen, trainings: [] });
3381         }
3382     }
3383
3384     render(){
3385         const { isOpen, trainings } = this.state;
3386         const { type } = this.props;
3387         return (
3388             <Fragment>
3389                 <Button id={`type-${type}`} type="button"
                            color="info"
                            onClick={this.handlerTrainings}>Anexos</Button>
3390                 { isOpen && <Tooltip target={`type-${type}`}
                            isOpen={isOpen}>{ trainings.map(training =>
                            training.name) }</Tooltip> }
3391             </Fragment>
3392         );
3393     }
3394 }
3395
3396 webapp/src/pages/protected/admin/eventType/List.jsx
3397
3398 import React, { Component, Fragment } from 'react';
3399 import { getRequest } from '../../../../utils/http';
3400 import { Row, Col, Table } from 'reactstrap';
3401 import ListItem from './ListItem.jsx';
3402 import Form from './Form.jsx';
3403 import { Filter, Pagination } from '../../../../components';
3404
3405 export default class extends Component{
3406     constructor(){
3407         super();
3408         this.state = {
3409             typesEvent: [],
3410             filter: '',
3411             count: 0,
3412             pages: 0
```

```
3413            };
3414            this.componentWillMount =
                    this.componentWillMount.bind(this);
3415        }
3416
3417        componentWillMount(){
3418            getRequest(
3419                '/typeEvent/page/0',
3420                res => {
3421                    const { data: typesEvent, count } = res.data
3422                    this.setState({
3423                        pages: count / 20,
3424                        typesEvent,
3425                        count
3426                    });
3427                }
3428            );
3429        }
3430
3431        handlerFilter(event) {
3432            const filter = event.target.value;
3433            if (filter.length >= 3 || filter.length === 0) {
3434                getRequest(
3435                    `/typeEvent/page/0?filter=${filter}`,
3436                    res => {
3437                        const { data: typesEvent, count } = res.data
3438                        this.setState({
3439                            pages: count / 20,
3440                            typesEvent,
3441                            count,
3442                            filter
3443                        })
3444                    }
3445                );
3446            } else {
3447                this.setState({ filter })
3448            }
3449        }
3450
3451        onChangePage(pageNumber) {
3452            const { filter } = this.state;
3453            getRequest(
3454                `/typeEvent/page/${pageNumber}?filter=${filter.lenght
                        > 3 ? filter : ''}`,
3455                res => {
```

```
3456                const { data: typesEvent, count } = res.data;
3457                this.setState({
3458                    pages: count / 20,
3459                    typesEvent,
3460                    count
3461                });
3462            }
3463        );
3464    }
3465
3466    render(){
3467        const { typesEvent, filter, count, pages } = this.state;
3468        return (
3469            <Fragment>
3470                <Row>
3471                    <Col>
3472                        <h3>Tipo de Eventos</h3>
3473                    </Col>
3474                    <Col md="1">
3475                    </Col>
3476                </Row>
3477                <br/>
3478                <Row>
3479                    <Col>
3480                        <Table striped>
3481                            <thead>
3482                                <tr>
3483                                    <th>
3484                                        <Filter
                                                label="Nome  "
                                                value={filter}
                                                handlerFilter={this.handlerFilter.
                                                />
3485                                    </th>
3486                                    <th>Anexos</th>
3487                                    <th></th>
3488                                </tr>
3489                            </thead>
3490                            <tbody>
3491                                {
3492                                    typesEvent.length?
3493                                    typesEvent.map(
3494                                        typeEvent => (
3495                                            <ListItem
                                                    key={typeEvent.type}
```

```
                                                typeEvent={typeEvent}
                                                afterDelete={this.componentWillMount}
                                            )
                                        ) : (
                                            <tr>
                                                <td colSpan="3">
                                                    <strong>Nenhum
                                                        Tipo de
                                                        Evento
                                                        cadastrado</strong>
                                                </td>
                                            </tr>
                                        )
                                    }
                                </tbody>
                            </Table>
                            <Pagination pages={pages} count={count}
                                onChangePage={this.onChangePage.bind(this)}
                                />
                        </Col>
                        <Col md="4">
                            <Form
                                afterSubmit={this.componentWillMount}/>
                        </Col>
                    </Row>
                </Fragment>
            );
        }
    }

    webapp/src/pages/protected/admin/Page.jsx

    import React, { Component } from 'react';
    import { Redirect } from 'react-router-dom';
    import { Row, Col, Button, Collapse } from 'reactstrap';
    import ListAttachment from './attachment';
    import ListQuestion from './question';
    import ListEventType from './eventType';
    import ListProject from './project';
    import EvaluatePage from './evaluate';
    import { Roles } from '../../../model';

    const styleButton = {
        width: '100%'
    };
```

```
3533
3534 export default class extends Component{
3535     constructor(){
3536         super();
3537         this.state = {
3538             show: {
3539                 project: false,
3540                 eventType: false,
3541                 attachment: false,
3542                 question: false,
3543                 evaluate: false
3544             }
3545         };
3546     }
3547
3548     changeShow(collapse) {
3549         const { show } = this.state;
3550         for (const key in show) {
3551             if (show.hasOwnProperty(key)) {
3552                 show[key] = key === collapse ? !show[key] : false;
3553             }
3554         }
3555         this.setState({ show })
3556     }
3557
3558     render(){
3559         const { role, logged } = this.props;
3560         const { show } = this.state;
3561         if (role === Roles.ADMIN) {
3562             return (
3563                 <Row>
3564                     <Col md="1" />
3565                     <Col>
3566                         <Row>
3567                             <Col>
3568                                 <Button type="button" color="info"
                                        style={styleButton} onClick={()
                                        =>
                                        this.changeShow('project')}>Projetos</But
3569                             </Col>
3570                             <Col md="1" />
3571                             <Col>
3572                                 <Button type="button" color="info"
                                        style={styleButton} onClick={()
                                        =>
```

```
                                this.changeShow('eventType')}>Tipos
                                de Evento</Button>
3573                        </Col>
3574                        <Col md="1" />
3575                        <Col>
3576                            <Button type="button" color="info"
                                    style={styleButton}
                                    onClick={()=>this.changeShow('attachment')}>Ane
3577                        </Col>
3578                        <Col md="1" />
3579                        <Col>
3580                            <Button type="button" color="info"
                                    style={styleButton}
                                    onClick={()=>this.changeShow('question')}>Quest
3581                        </Col>
3582                        <Col md="1" />
3583                        <Col>
3584                            <Button type="button" color="info"
                                    style={styleButton}
                                    onClick={()=>this.changeShow('evaluate')}>Avali
3585                        </Col>
3586                    </Row>
3587                    <br/>
3588                    <Row>
3589                        <Col>
3590                            <Collapse isOpen={show.project}>
3591                                <ListProject />
3592                            </Collapse>
3593                            <Collapse isOpen={show.eventType}>
3594                                <ListEventType />
3595                            </Collapse>
3596                            <Collapse isOpen={show.attachment}>
3597                                <ListAttachment />
3598                            </Collapse>
3599                            <Collapse isOpen={show.question}>
3600                                <ListQuestion />
3601                            </Collapse>
3602                            <Collapse isOpen={show.evaluate}>
3603                                <EvaluatePage/>
3604                            </Collapse>
3605                        </Col>
3606                    </Row>
3607                </Col>
3608                <Col md="1" />
3609            </Row>
```

```
            );
        }
        if(logged){
            return <Redirect to="/dashboard" />
        }
        return <Redirect to="/login" />
    }
}

webapp/src/pages/protected/admin/index.js

import Page from './Page.jsx';
export default Page;

webapp/src/pages/protected/dashboard/admin/volunteer/index.js

import List from './List.jsx';
export default List;

webapp/src/pages/protected/dashboard/admin/volunteer/ListItem.jsx

import React, { Component } from 'react';
import { Row, Col, ListGroupItem } from 'reactstrap';
import { FaTrash } from 'react-icons/fa';
import Details from './Details.jsx';
import PropTypes from 'prop-types';
import { Checkbox } from '../../../../../components';
import { deleteRequest } from '../../../../../utils/http';
import formatter from '../../../../../utils/formatter';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            isOpen: false
        }
        this.toggle = this.toggle.bind(this);
        this.handlerDelete = this.handlerDelete.bind(this);
    }

    static propTypes = {
        volunteer: PropTypes.object.isRequired,
        afterDelete: PropTypes.func.isRequired
    }
```

```
3655        toggle(){
3656            const { isOpen } = this.state;
3657            this.setState({isOpen: !isOpen});
3658        }
3659
3660        handlerDelete(){
3661            const { volunteer, afterDelete } = this.props;
3662            const { user } = volunteer;
3663            deleteRequest(‘/user/${user.cpf}‘, afterDelete)
3664        }
3665
3666        render(){
3667            const { volunteer, onSelectVolunteer } = this.props;
3668            const { user } = volunteer;
3669            const { isOpen } = this.state;
3670            return (
3671                <ListGroupItem>
3672                    <Row>
3673                        <Col md="1">
3674                            <Checkbox value={user.cpf}
                                    onChange={onSelectVolunteer} />
3675                        </Col>
3676                        <Col onClick={this.toggle}
                                 className="text-ellipsis" style={{
                                 cursor: ’pointer’ }}>
3677                            {formatter.cpf(user.cpf)} - {user.name}
3678                        </Col>
3679                        <Col md="2">
3680                            <FaTrash style={{ cursor: ’pointer’ }}
                                    color="red"
                                    onClick={this.handlerDelete} />
3681                        </Col>
3682                    </Row>
3683                    { isOpen && <Details cpf={user.cpf}
                             isOpen={isOpen} close={this.toggle}
                             deleteFunc={this.handlerDelete}/> }
3684                </ListGroupItem>
3685            );
3686        }
3687    }
3688
3689    webapp/src/pages/protected/dashboard/admin/volunteer/Details.jsx
3690
3691    import React, { Component } from ’react’;
```

```
import { getRequest, putRequest } from '../../../../../utils/http';
import formatter from '../../../../../utils/formatter';
import { Input } from '../../../../../components';
import { Status, Address, Schooling, School } from '../../../../../model';
import axios from 'axios';
import { Button, Modal, ModalHeader, ModalBody, ModalFooter, Row, Col, Label, Form } from 'reactstrap';
import pic from './pic.jpeg';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            volunteer: null,
            trainings: [],
            schools: [],
            err: {
                invalid: false
            }
        };
        this.handlerAprroved = this.handlerAprroved.bind(this);
        this.HandlerAssociate = this.HandlerAssociate.bind(this);
        this.handlerDelete = this.handlerDelete.bind(this);
        this.HandlerSubmit = this.HandlerSubmit.bind(this);
    }

    componentWillMount(){
        const { cpf } = this.props;
        axios.all([
            getRequest(`/volunteer/${cpf}`, res => res.data),
            getRequest('/school', res => res.data)
        ]).then(
            res => this.setState({
                volunteer: res[0],
                schools: res[1]
            })
        );
    }

    handlerDelete(){
        const { deleteFunc, close } = this.props;
        deleteFunc();
        close();
```

```
3734        }
3735
3736        handlerAprroved(){
3737            const { close } = this.props;
3738            const { volunteer } = this.state;
3739            const { user } = volunteer;
3740            user.status = Status.APPROVED;
3741            volunteer.user = user;
3742            putRequest(`/volunteer/${user.cpf}`, volunteer, close);
3743        }
3744
3745        HandlerAssociate(event){
3746            debugger;
3747            const { volunteer, schools } = this.state;
3748            const { value } = event.target;
3749            if(value){
3750                for (const school of schools) {
3751                    if(school.codSchool === parseInt(value, 10)){
3752                        volunteer.school = school;
3753                        break;
3754                    }
3755                }
3756            } else {
3757                volunteer.school = new School()
3758            }
3759            this.setState({
3760                volunteer,
3761                err: {
3762                    invalid: false
3763                }
3764            });
3765        }
3766
3767        HandlerSubmit(){
3768            const { volunteer } = this.state;
3769            const { user } = volunteer;
3770            if (volunteer.school.codSchool){
3771                putRequest(`/volunteer/${user.cpf}`, volunteer);
3772            } else {
3773                this.setState({
3774                    err: {
3775                        invalid: true
3776                    }
3777                })
3778            }
```

```
3779        }
3780
3781        isApprovable(){
3782            const { volunteer, trainings } = this.state;
3783            const { user } = volunteer;
3784            return user.status === Status.WAIT_TRAINING &&
                    trainings.length && !(trainings.filter(training =>
                    training.endOccurrence > new
                    Date().getTime()).length)
3785        }
3786
3787        render(){
3788            const { volunteer, schools, err } = this.state;
3789            const { close, isOpen } = this.props;
3790            if(volunteer){
3791                const { user, school } = volunteer;
3792                const address = new Address();
3793                Object.assign(address, user.address);
3794                return (
3795                    <Modal toggle={close} isOpen={isOpen}>
3796                        <ModalHeader toggle={close}>{user.name} -
                                <small>{formatter.cpf(user.cpf)}</small></ModalHeade
3797                        <ModalBody>
3798                            <Row>
3799                                <Col md="1">
3800                                    <img src={pic} alt=""
                                            className="rounded-circle"
                                            width="75px" />
3801                                </Col>
3802                                <Col md="2" />
3803                                <Col>
3804                                    <Row>
3805                                        <Col>
3806                                            <Label className="btn
                                                    btn-info"
                                                    style={{cursor:
                                                    'default'}}>
3807                                                {Status.translate(user.status)}
3808                                            </Label>
3809                                        </Col>
3810                                    </Row>
3811                                    <Row>
3812                                        <Col>
3813                                            <strong>Esoclaridade:</strong>
3814                                        </Col>
```

```
                </Row>
                <Row>
                    <Col>
                        {
                            Schooling.translate(volunteer.schooling)
                            } -
                            {volunteer.conclusion ?
                            'Completo':
                            'Incompleto'}
                        {
                            volunteer.course && <div>
                                { volunteer.course }
                            </div>
                        }
                    </Col>
                </Row>
                <br />
                <Row>
                    <Col>
                        <strong>Contato:</strong>
                    </Col>
                </Row>
                <Row>
                    <Col>
                        {user.email} /
                            {formatter.phone(user.phone)}
                    </Col>
                </Row>
                <br/>
                <Row>
                    <Col>
                        <strong>Endereo:</strong>
                    </Col>
                </Row>
                <Row>
                    <Col>
                        {address.toString()}
                    </Col>
                </Row>
                <br />
                <Row>
                    <Col>
                        <strong> Avaliaes :</strong>
                    </Col>
                </Row>
```

```
3854                        <Row>
3855                            <Col>
3856                                {
3857                                    volunteer.ratings.length?
3858                                        ‘ Avaliao   mdia  de
                                                ${user.name}
                                                ${volunteer.rating}‘:
                                                ’Voluntrio ainda
                                                no foi avaliado’
3859                                }
3860                            </Col>
3861                        </Row>
3862                        <br/>
3863                        <Row>
3864                            <Col>
3865                                <Form inline>
3866                                    <Input id="school"
                                            type="select"
                                            label="Escola"
                                            invalidMessage="O
                                            voluntrio deve ser
                                            associado a uma
                                            escola"
3867                                        onChange={this.HandlerAssociate
                                            value={school?
                                            school.codSchool
                                            : ’’} {...err}
                                            required>
3868                                        <option
                                            value="">Selecione</option>
3869                                        {
3870                                            schools.map(school
                                                => <option
                                                key={school.codSchool}
                                                value={school.codSchool
3871                                        }
3872                                    </Input>
3873                                    <Button color="primary"
                                            type="button"
                                            onClick={this.HandlerSubmit}>A
3874                                </Form>
3875                            </Col>
3876                        </Row>
3877                    </Col>
3878                </Row>
```

```
3879                    </ModalBody>
3880                    <ModalFooter style={{ display: 'inline' }}>
3881                        <Row>
3882                            <Col>
3883                                <Button color="danger"
                                        onClick={this.handlerDelete}>Excluir</Button>
3884                            </Col>
3885                            <Col>
3886                                {
3887                                    this.isApprovable()? (
3888                                        <Button color="info"
                                                className="float-right"
                                                onClick={this.handlerAprroved}>Aprovar</
3889                                    ) : (
3890                                        <Button color="default"
                                                className="float-right"
                                                onClick={close}>Fechar</Button>
3891                                    )
3892                                }
3893                            </Col>
3894                        </Row>
3895                    </ModalFooter>
3896                </Modal>
3897            )
3898        }
3899        return (
3900            <Modal toggle={close} isOpen={isOpen}/>
3901        );
3902    }
3903 }
3904
3905 webapp/src/pages/protected/dashboard/admin/volunteer/List.jsx
3906
3907 import React, { Component, Fragment } from 'react';
3908 import { getRequest } from '../../../../../utils/http';
3909 import { Row, Col, ListGroup, ListGroupItem } from 'reactstrap';
3910 import VolunteerItem from './ListItem.jsx';
3911 import { Pagination, Filter } from '../../../../../components';
3912
3913 export default class extends Component{
3914     constructor() {
3915         super()
3916         this.state = {
3917             filter: '',
3918             volunteers: [],
```

```
3919            count: 0,
3920            pages: 0
3921        }
3922        this.onChangePage = this.onChangePage.bind(this);
3923        this.handlerFilter = this.handlerFilter.bind(this);
3924    }
3925
3926    componentWillMount() {
3927        const { filterByCompany, filterBySchool } = this.props;
3928        const { filter } = this.state;
3929        getRequest(
3930            `/volunteer/page/0?cod_cnpj=${filterByCompany.join(',')}&cod_sch
3931            res => {
3932                const page = res.data;
3933                this.setState({
3934                    volunteers: page.data,
3935                    count: page.count,
3936                    pages: page.count / 5
3937                });
3938            }
3939        );
3940    }
3941
3942    componentWillReceiveProps(){
3943        this.componentWillMount();
3944    }
3945
3946    handlerFilter(event) {
3947        const { filterByCompany, filterBySchool } = this.props;
3948        const filter = event.target.value;
3949        if (filter.length >= 3 || filter.length === 0){
3950            getRequest(
3951                `/volunteer/page/0?cod_cnpj=${filterByCompany.join(',')}&cod
3952                res => {
3953                    const page = res.data;
3954                    this.setState({
3955                        volunteers: page.data,
3956                        count: page.count,
3957                        pages: page.count / 5,
3958                        filter
3959                    });
3960                }
3961            );
3962        } else {
3963            this.setState({filter});
```

```
3964        }
3965    }
3966
3967    onChangePage(pageNumber){
3968        const { filterByCompany, filterBySchool } = this.props;
3969        const { filter } = this.state;
3970        getRequest(
3971            `/volunteer/page/${pageNumber}?cod_cnpj=${filterByCompany.join(',')}&c
                > 3 ? filter : ''}`,
3972            res => {
3973                const page = res.data;
3974                this.setState({
3975                    volunteers: page.data,
3976                    count: page.count,
3977                    pages: page.count / 5
3978                });
3979            }
3980        );
3981    }
3982    render(){
3983        const { volunteers, pages, count } = this.state;
3984        const { onSelectVolunteer } = this.props;
3985        return (
3986            <Fragment>
3987                <Row>
3988                    <Col>
3989                        <h3>Mentores Voluntrios</h3>
3990                    </Col>
3991                </Row>
3992                <Filter handlerFilter={this.handlerFilter} />
3993                <Row>
3994                    <Col>
3995                        <ListGroup>
3996                            {
3997                                volunteers.length ?
3998                                    volunteers.map(
3999                                        volunteer => (
4000                                            <VolunteerItem
                                                    key={volunteer.user.cpf}
                                                    volunteer={volunteer}
                                                    onSelectVolunteer={onSelectVolunteer
                                                    afterDelete={this.componentWillMount
4001                                        )
4002                                    ) : (
4003                                        <ListGroupItem>
```

```
                                        <strong>
                                            Nenhum Voluntrio
                                                Cadastrado
                                        </strong>
                                    </ListGroupItem>
                                )
                            }
                        </ListGroup>
                    </Col>
                </Row>
                <Pagination pages={pages} count={count}
                    onChangePage={this.onChangePage}/>
            </Fragment>
        );
    }
}

webapp/src/pages/protected/dashboard/admin/school/Form.jsx

import React, { Component } from 'react';
import { School, SchoolType } from '../../../../../model';
import PropTypes from 'prop-types';
import { postRequest } from '../../../../../utils/http';
import { Row, Col, Modal, ModalHeader, ModalBody } from
    'reactstrap';
import { Wizard, Input, MaskInput, DatePicker } from
    '../../../../../components';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            school: new School(),
            confirmPassword: null
        };
        this.handlerName = this.handlerName.bind(this);
        this.handlerPhone = this.handlerPhone.bind(this);
        this.handlerType = this.handlerType.bind(this);
        this.handlerNameResponsible =
            this.handlerNameResponsible.bind(this);
        this.handlerCpf = this.handlerCpf.bind(this);
        this.handlerPhoneResponsible =
            this.handlerPhoneResponsible.bind(this);
        this.handlerJob = this.handlerJob.bind(this);
        this.handlerEmail = this.handlerEmail.bind(this);
```

```
4043            this.handlerBirth = this.handlerBirth.bind(this);
4044            this.handlerPassword = this.handlerPassword.bind(this);
4045            this.handlerConfirmPassword =
                    this.handlerConfirmPassword.bind(this);
4046            this.handlerCity = this.handlerCity.bind(this);
4047            this.handlerNeighborhood =
                    this.handlerNeighborhood.bind(this);
4048            this.handlerStreet = this.handlerStreet.bind(this);
4049            this.handlerNumber = this.handlerNumber.bind(this);
4050            this.handlerSubmit = this.handlerSubmit.bind(this);
4051        }
4052
4053        static propTypes = {
4054            afterSubmit: PropTypes.func.isRequired,
4055            isOpen: PropTypes.bool.isRequired,
4056            close: PropTypes.func.isRequired
4057        }
4058
4059
4060        //school
4061        handlerName(event){
4062            const { school } = this.state;
4063            school.name = event.target.value;
4064            this.setState({ school });
4065        }
4066
4067        handlerPhone(event){
4068            const { school } = this.state;
4069            school.phone = event.target.value.replace(/[()-]/g, '');
4070            this.setState({ school });
4071        }
4072
4073        handlerType(event) {
4074            const { school } = this.state;
4075            school.type = event.target.value;
4076            this.setState({ school });
4077        }
4078
4079        // responsible
4080        handlerNameResponsible(event){
4081            const { school } = this.state;
4082            const { responsible } = school;
4083            responsible.name = event.target.value;
4084            school.responsible = responsible;
4085            this.setState({ school });
```

```
4086        }
4087
4088        handlerCpf(event){
4089            const { school } = this.state;
4090            const { responsible } = school;
4091            responsible.cpf = event.target.value.replace(/[.-]/g,
                    '');
4092            school.responsible = responsible;
4093            this.setState({ school });
4094        }
4095
4096        handlerPhoneResponsible(event) {
4097            const { school } = this.state;
4098            const { responsible } = school;
4099            responsible.phone = event.target.value.replace(/[()-]/g,
                    '');
4100            school.responsible = responsible;
4101            this.setState({ school });
4102        }
4103
4104        handlerJob(event){
4105            const { school } = this.state;
4106            const { responsible } = school;
4107            responsible.job = event.target.value;
4108            school.responsible = responsible;
4109            this.setState({ school });
4110        }
4111
4112        handlerEmail(event){
4113            const { school } = this.state;
4114            const { responsible } = school;
4115            responsible.email = event.target.value;
4116            school.responsible = responsible;
4117            this.setState({ school });
4118        }
4119
4120        handlerBirth(event){
4121            const { school } = this.state;
4122            const { responsible } = school;
4123            const { value } = event.target;
4124            responsible.birth = value ? new Date(value) : null;
4125            school.responsible = responsible;
4126            this.setState({ school });
4127        }
4128
```

```
4129        handlerPassword(event){
4130            const { school } = this.state;
4131            const { responsible } = school;
4132            responsible.password = event.target.value;
4133            school.responsible = responsible;
4134            this.setState({ school });
4135        }
4136
4137        handlerConfirmPassword(event){
4138            let { confirmPassword } = this.state;
4139            confirmPassword = event.target.value;
4140            this.setState({ confirmPassword });
4141        }
4142
4143        //address
4144        handlerCity(event){
4145            const { school } = this.state;
4146            const { responsible, address } = school;
4147            address.city = event.target.value;
4148            school.address = responsible.address = address;
4149            school.responsible = responsible;
4150            this.setState({ school });
4151        }
4152
4153        handlerNeighborhood(event){
4154            const { school } = this.state;
4155            const { responsible, address } = school;
4156            address.neighborhood = event.target.value;
4157            school.address = responsible.address = address;
4158            school.responsible = responsible;
4159            this.setState({ school });
4160        }
4161
4162        handlerStreet(event){
4163            const { school } = this.state;
4164            const { responsible, address } = school;
4165            address.street = event.target.value;
4166            school.address = responsible.address = address;
4167            school.responsible = responsible;
4168            this.setState({ school });
4169        }
4170
4171        handlerNumber(event){
4172            const { school } = this.state;
4173            const { responsible, address } = school;
```

```
4174            address.number = event.target.value;
4175            school.address = responsible.address = address;
4176            school.responsible = responsible;
4177            this.setState({ school });
4178        }
4179
4180        handlerSubmit(){
4181            const { afterSubmit, close } = this.props;
4182            const { school } = this.state;
4183            postRequest(
4184                '/school',
4185                school,
4186                () => {
4187                    afterSubmit();
4188                    close();
4189                }
4190            );
4191        }
4192
4193        render(){
4194            const { close, isOpen } = this.props;
4195            const { school, confirmPassword } = this.state;
4196            const { address, responsible } = school;
4197            const maxDate = new Date().toJSON().split('T')[0];
4198            return (
4199                <Modal isOpen={isOpen} toggle={close}>
4200                    <ModalHeader toggle={close}>Nova
                            Empresa</ModalHeader>
4201                    <ModalBody>
4202                        <Wizard onCancel={close}
                                submitLabel="cadastrar"
                                onSubmit={this.handlerSubmit}>
4203                            <div>
4204                                <h3>Dados da Escola</h3>
4205                                <Input id="nameCompany" label="Nome da
                                        Escola" invalidMessage="Nome da
                                        Escola  obrigatrio"
                                        value={school.name}
                                        onChange={this.handlerName}
                                        required/>
4206                                <MaskInput id="phone" label="Contato
                                        da Empresa"
                                        invalidMessage="Contato da Escola
                                        obrigatrio" value={school.phone}
                                        mask="(99)9999-9999"
```

```
                        onChange={this.handlerPhone}
                        required/>
4207                <Input id="schoolType" type="select"
                        label="Nvel Federativo"
                        invalidMessage="Nvel Federativo
                        obrigatrio"
                        value={school.schoolType}
                        onChange={this.handlerType}
                        required >
4208                    <option value="">Selecione</option>
4209                    {
4210                        SchoolType.values().map(
4211                            type => <option key={type}
                                value={type}>{SchoolType.translate(type)
4212                        )
4213                    }
4214                </Input>
4215                <hr className="row"/>
4216            </div>
4217            <div>
4218                <h3>Dados do responsvel</h3>
4219                <Input id="nameReponsible" label="Nome
                        Completo" invalidMessage="Nome
                        Completo  obrigatrio"
                        value={responsible.name}
                        onChange={this.handlerNameResponsible}
                        required />
4220                <MaskInput id="cpf" label="CPF"
                        invalidMessage="CPF  obrigatrio"
                        value={responsible.cpf}
                        mask="999.999.999-99"
                        onChange={this.handlerCpf}
                        required />
4221                <MaskInput id="phoneResponsible"
                        label="Contato do Responsvel"
                        invalidMessage="Contato do
                        Responsvel  obrigatrio"
                        value={responsible.phone}
                        mask="(99)9999-9999"
                        onChange={this.handlerPhoneResponsible}
                        required />
4222                <Input id="job" label="Cargo do
                        Responsvel" invalidMessage="Cargo
                        do Responsvel  obrigatrio"
                        value={responsible.job}
```

```
                        onChange={this.handlerJob}
                        required />
4223                <DatePicker id="birth"
                        label="Nascimento"
                        invalidMessage="Nascimento
                        obrigatrio" max={maxDate}
                        onChange={this.handlerBirth}
                        required />
4224                <Input id="email" type="email"
                        label="Email"
                        invalidMessage="Email  obrigatrio"
                        value={responsible.email}
                        onChange={this.handlerEmail}
                        required />
4225                <Row>
4226                    <Col>
4227                        <Input id="password"
                                type="password"
                                label="Senha"
                                invalidMessage="Senha
                                obrigatrio"
                                value={responsible.password}
                                onChange={this.handlerPassword}
                                required />
4228                    </Col>
4229                    <Col>
4230                        <Input id="passwordConfirm"
                                type="password"
                                label="Confirmar Senha"
                                invalidMessage="As senhas
                                devem ser idnticas"
                                value={confirmPassword}
                                onChange={this.handlerConfirmPassword}
                                required />
4231                    </Col>
4232                </Row>
4233                <hr className="row"/>
4234            </div>
4235            <div>
4236                <h3>Endereo  da Escola</h3>
4237                <Input id="city" label="Cidade"
                        invalidMessage="Cidade
                        obrigatrio" value={address.city}
                        onChange={this.handlerCity}
                        required/>
```

```
                            <Input id="neighborhood"
                                label="Bairro"
                                invalidMessage="Bairro
                                obrigatrio"
                                value={address.neighborhood}
                                onChange={this.handlerNeighborhood}
                                required/>
                            <Row>
                                <Col>
                                    <Input id="street"
                                        label="Lougradouro"
                                        invalidMessage="Lougradouro
                                         obrigatrio"
                                        value={address.street}
                                        onChange={this.handlerStreet}
                                        required/>
                                </Col>
                                <Col md="3">
                                    <Input id="number"
                                        label="Nmero"
                                        value={address.number}
                                        onChange={this.handlerNumber}/>
                                </Col>
                            </Row>
                            <hr className="row"/>
                        </div>
                    </Wizard>
                </ModalBody>
            </Modal>
        );
    }
}
```

webapp/src/pages/protected/dashboard/admin/school/index.js

```
import List from './List.jsx';
export default List;
```

webapp/src/pages/protected/dashboard/admin/school/ListItem.jsx

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { deleteRequest } from '../../../../../utils/http';
import { Checkbox } from '../../../../../components';
import { Row, Col, ListGroupItem } from 'reactstrap';
```

```
4268 import { FaTrash } from 'react-icons/fa';
4269 import Details from './Details.jsx';
4270
4271 export default class extends Component{
4272     constructor(){
4273         super();
4274         this.state = {
4275             isOpen: false
4276         }
4277         this.handlerDelete = this.handlerDelete.bind(this);
4278         this.handlerDetails = this.handlerDetails.bind(this);
4279     }
4280
4281     static propTypes = {
4282         school: PropTypes.object.isRequired,
4283         afterDelete: PropTypes.func.isRequired
4284     }
4285
4286     handlerDelete(){
4287         const { school, afterDelete } = this.props;
4288         deleteRequest(`/school/${school.codSchool}`,
                afterDelete);
4289     }
4290
4291     handlerDetails(){
4292         const { isOpen } = this.state;
4293         this.setState({ isOpen: !isOpen });
4294     }
4295
4296     render(){
4297         const { school, onSelectSchool } = this.props;
4298         const { isOpen } = this.state;
4299         return (
4300             <ListGroupItem key={school.codSchool}>
4301                 <Row>
4302                     <Col md="1">
4303                         <Checkbox value={school.codSchool}
                                onChange={onSelectSchool}/>
4304                     </Col>
4305                     <Col md="9" onClick={this.handlerDetails}
                            className="text-ellipsis" style={{
                            cursor: 'pointer' }}>
4306                         {school.name}
4307                     </Col>
4308                     <Col>
```

```
                        <FaTrash style={{ cursor: 'pointer' }}
                            color="red"
                            onClick={this.handlerDelete} />
                    </Col>
                </Row>
                {
                    isOpen ? <Details code={school.codSchool}
                        isOpen={isOpen}
                        close={this.handlerDetails}/> : null
                }
            </ListGroupItem>
        );
    }
}
```

webapp/src/pages/protected/dashboard/admin/school/Add.jsx

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { Button } from 'reactstrap';
import { FaPlus } from 'react-icons/fa';
import Form from './Form.jsx';

export default class extends Component {
    constructor() {
        super();
        this.state = {
            isOpen: false
        };
    }

    static propTypes = {
        afterSubmit: PropTypes.func.isRequired
    }

    handlerAdd() {
        const { isOpen } = this.state;
        this.setState({ isOpen: !isOpen });
    }

    render() {
        const { isOpen } = this.state;
        const { afterSubmit } = this.props;
        return (
            <div>
```

```
                <Button type="button" color="primary"
                    onClick={this.handlerAdd.bind(this)}>
                    <FaPlus />
                </Button>
                <Form isOpen={isOpen} afterSubmit={afterSubmit}
                    close={this.handlerAdd.bind(this)} />
            </div>
        );
    }
}
```

webapp/src/pages/protected/dashboard/admin/school/Details.jsx

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { Address, SchoolType } from '../../../../../model';
import { getRequest } from '../../../../../utils/http';
import formatter from '../../../../../utils/formatter';
import { Button, Modal, ModalHeader, ModalBody, ModalFooter,
    Row, Col } from 'reactstrap';
import pic from './pic.jpeg';

export default class extends Component {
    constructor(){
        super();
        this.state = {
            school: null
        }
    }
    static propTypes = {
        code: PropTypes.string.isRequired,
        isOpen: PropTypes.bool.isRequired,
        close: PropTypes.func.isRequired
    };

    componentWillMount() {
        const { code } = this.props;
        getRequest(`/school/${code}`, res => this.setState({
            school: res.data }));
    }

    render() {
        const { isOpen, close } = this.props;
        const { school } = this.state;
        if (school) {
```

```
4391            const address = new Address();
4392            Object.assign(address, school.address);
4393            return (
4394                <Modal toggle={close} isOpen={isOpen} >
4395                    <ModalHeader toggle={close}>
4396                        <Row>
4397                            <Col>
4398                                {school.name}
4399                            </Col>
4400                        </Row>
4401                        <Row>
4402                            <Col>
4403                                <small>
4404                                    <strong>Escola
                                        {SchoolType.translate(school.type)}</strong>
4405                                </small>
4406                            </Col>
4407                        </Row>
4408                    </ModalHeader>
4409                    <ModalBody>
4410                        <Row>
4411                            <Col md={1}>
4412                                <img src={pic} alt=""
                                     className="rounded-circle"
                                     width="75px" />
4413                            </Col>
4414                            <Col md="2" />
4415                            <Col>
4416                                <Row>
4417                                    <Col>
4418                                        <strong>Endereo:</strong>
4419                                    </Col>
4420                                </Row>
4421                                <Row>
4422                                    <Col>
4423                                        {address.toString()}
4424                                    </Col>
4425                                </Row>
4426                                <br />
4427                                <Row>
4428                                    <Col>
4429                                        <strong>Contato:</strong>
4430                                    </Col>
4431                                </Row>
4432                                <Row>
```

```
                                    <Col>
                                        {school.responsible.email} /
                                            {formatter.phone(school.phone)}
                                    </Col>
                                </Row>
                            </Col>
                        </Row>
                        <br />
                        <Row>
                            <Col md="3" />
                            <Col>
                                <strong>Responsvel:</strong>
                            </Col>
                        </Row>
                        <Row>
                            <Col md="3" />
                            <Col>
                                <div>{school.responsible.name}</div>
                                <small>{formatter.cpf(school.responsible.cpf)}
                            </Col>
                        </Row>
                    </ModalBody>
                    <ModalFooter>
                        <Button color="primary"
                            onClick={close}>Fechar</Button>
                    </ModalFooter>
                </Modal>
            );
        } else {
            return (
                <Modal toggle={close} isOpen={isOpen} ></Modal>
            );
        }
    }
}
```

webapp/src/pages/protected/dashboard/admin/school/List.jsx

```
import React, { Component } from 'react';
import { getRequest } from '../../../../../utils/http';
import { Row, Col, ListGroup, ListGroupItem } from 'reactstrap';
import { Pagination, Filter } from '../../../../../components';
import SchoolItem from './ListItem.jsx';
import Add from './Add.jsx';
```

```
4476 export default class extends Component{
4477     constructor(){
4478         super();
4479         this.state = {
4480             filter: '',
4481             schools: [],
4482             count: 0,
4483             pages: 0
4484         };
4485         this.onChangePage = this.onChangePage.bind(this);
4486         this.handlerFilter = this.handlerFilter.bind(this);
4487         this.componentWillMount =
                 this.componentWillMount.bind(this)
4488     }
4489
4490     componentWillMount(){
4491         getRequest(
4492             '/school/page/0',
4493             res => {
4494                 const page = res.data;
4495                 this.setState({
4496                     schools: page.data,
4497                     count: page.count,
4498                     pages: page.count / 5
4499                 });
4500             }
4501         );
4502     }
4503
4504     handlerFilter(event) {
4505         const filter = event.target.value;
4506         if (filter.length >= 3 || filter.length === 0){
4507             getRequest(
4508                 `/school/page/0?filter=${filter}`,
4509                 res => {
4510                     const page = res.data;
4511                     this.setState({
4512                         companies: page.data,
4513                         count: page.count,
4514                         pages: page.count / 5,
4515                         filter
4516                     });
4517                 }
4518             );
4519         } else {
```

```
4520            this.setState({filter});
4521        }
4522    }
4523
4524    onChangePage(pageNumber) {
4525        const { filter } = this.state;
4526        getRequest(
4527            `/school/page/${pageNumber}?filter=${filter.lenght >
                    3 ? filter : ''}`,
4528            res => {
4529                const page = res.data;
4530                this.setState({
4531                    schools: page.data,
4532                    count: page.count,
4533                    pages: page.count / 5
4534                });
4535            }
4536        );
4537    }
4538
4539    render(){
4540        const { schools, pages, count } = this.state;
4541        const { onSelectSchool } = this.props;
4542        return (
4543            <div>
4544                <Row>
4545                    <Col>
4546                        <h3>
4547                             Organizaes  Beneficiadas
4548                        </h3>
4549                    </Col>
4550                    <Col md="2">
4551                        <Add afterSubmit={this.componentWillMount}
                                />
4552                    </Col>
4553                </Row>
4554                <Filter handlerFilter={this.handlerFilter} />
4555                <Row>
4556                    <Col>
4557                        <ListGroup>
4558                            {
4559                                schools.length ?
4560                                    schools.map(
4561                                        school =>
```

```
                                        <SchoolItem
                                            key={school.codSchool}
                                            onSelectSchool={onSelectSchool}
                                            school={school}
                                            afterDelete={this.componentWillMount}
                                            />
                                    ) : (
                                        <ListGroupItem>
                                            <strong>
                                                Nenhuma  Organizao
                                                    Cadastrada
                                            </strong>
                                        </ListGroupItem>
                                    )
                                }
                            </ListGroup>
                        </Col>
                    </Row>
                    <Pagination pages={pages} count={count}
                        onChangePage={this.onChangePage} />
                </div>
            );
        }
    }
```

webapp/src/pages/protected/dashboard/admin/event/form/Form.jsx

```
import React, { Component, Fragment } from 'react';
import PropTypes from 'prop-types';
import { postRequest, getRequest } from
    '../../../../../../utils/http';
import formatter from '../../../../../../utils/formatter';
import { Event, School, Training } from
    '../../../../../../model';
import { Row, Col, Modal, ModalHeader, ModalBody, Button } from
    'reactstrap';
import { Wizard, Input, Switch, FileInput } from
    '../../../../../../components';
import axios from 'axios';
import './form.css';

export default class extends Component {
    constructor(props){
        super(props);
        this.state = {
```

```
4596            event: props.event,
4597            volunteers: [],
4598            schools: [],
4599            typesEvent: [],
4600            trainingsOfType: []
4601        };
4602        this.handlerType = this.handlerType.bind(this);
4603        this.handlerCity = this.handlerCity.bind(this);
4604        this.handlerNeighborhood =
                this.handlerNeighborhood.bind(this);
4605        this.handlerStreet = this.handlerStreet.bind(this);
4606        this.handlerNumber = this.handlerNumber.bind(this);
4607        this.handlerParticipants =
                this.handlerParticipants.bind(this);
4608        this.handlerSelectSchool =
                this.handlerSelectSchool.bind(this);
4609        this.handlerAdd = this.handlerAdd.bind(this);
4610        this.handlerRemove = this.handlerRemove.bind(this);
4611        this.handlerNameAttachemnt =
                this.handlerNameAttachemnt.bind(this);
4612        this.handlerLinkAttachemnt =
                this.handlerLinkAttachemnt.bind(this);
4613        this.handlerIsFile = this.handlerIsFile.bind(this);
4614        this.handlerEvaluate = this.handlerEvaluate.bind(this);
4615        this.handlerSubmit = this.handlerSubmit.bind(this);
4616    }
4617
4618    static propTypes = {
4619        event: PropTypes.objectOf(Event).isRequired,
4620        afterSubmit: PropTypes.func.isRequired,
4621        isOpen: PropTypes.bool.isRequired,
4622        close: PropTypes.func.isRequired
4623    }
4624
4625    componentWillMount(){
4626        const { event } = this.state;
4627        event.allDay = event.startOccurrence ===
                event.endOccurrence;
4628        axios.all([
4629            getRequest('/school', res => res.data),
4630            getRequest('/typeEvent', res => res.data)
4631        ]).then(
4632            res => this.setState({
4633                event,
4634                schools: res[0],
```

```
4635                typesEvent: res[1]
4636            })
4637        );
4638    }
4639
4640    handlerEvaluate(value){
4641        const { event } = this.state;
4642        event.evaluate = value;
4643        this.setState({event});
4644    }
4645
4646    handlerType(userEvent){
4647        const { event, typesEvent } = this.state;
4648        const { value } = userEvent.target;
4649        event.type = null;
4650        if(value){
4651            for (const key in typesEvent) {
4652                if (typesEvent.hasOwnProperty(key)) {
4653                    const typeEvent = typesEvent[key];
4654                    if(typeEvent.type === parseInt(value, 10)){
4655                        event.type = typeEvent;
4656                    }
4657                }
4658            }
4659            getRequest(
4660                `/typeEvent/${value}/trainings`,
4661                res => this.setState({ trainingsOfType: res.data,
                        event }),
4662                () => this.setState({ trainingsOfType: [], event
                        })
4663            );
4664        }
4665    }
4666
4667    //address
4668    handlerCity(userEvent) {
4669        const { event } = this.state;
4670        const {address } = event;
4671        address.city = userEvent.target.value;
4672        event.address = address;
4673        this.setState({ event });
4674    }
4675
4676    handlerNeighborhood(userEvent) {
4677        const { event } = this.state;
```

```
4678            const {address } = event;
4679            address.neighborhood = userEvent.target.value;
4680            event.address = address;
4681            this.setState({ event });
4682        }
4683
4684        handlerStreet(userEvent) {
4685            const { event } = this.state;
4686            const {address } = event;
4687            address.street = userEvent.target.value;
4688            event.address = address;
4689            this.setState({ event });
4690        }
4691
4692        handlerNumber(userEvent) {
4693            const { event } = this.state;
4694            const {address } = event;
4695            address.number = userEvent.target.value;
4696            event.address = address;
4697            this.setState({ event });
4698        }
4699
4700        //participants
4701        handlerSelectSchool(userEvent){
4702            const { value } = userEvent.target;
4703            const { event, schools } = this.state
4704            event.school = new School()
4705            for(const school of schools){
4706                if(school.codSchool === parseInt(value, 10)){
4707                    event.school = school;
4708                }
4709            }
4710            if(value){
4711                axios.all([
4712                    getRequest(
4713                        `/volunteer?cod_school=${value}`,
4714                        res => res.data
4715                    ),
4716                    getRequest(
4717                        `/school/${value}`,
4718                        res => res.data
4719                    )
4720                ]).then(
4721                    res => {
4722                        const school = res[1];
```

```
4723                    event.school = school;
4724                    event.address = school.address;
4725                    event.address.codAddress = null;
4726                    this.setState({volunteers: res[0], event});
4727                }
4728            )
4729        } else{
4730            this.setState({volunteers: [], event});
4731        }
4732    }
4733    handlerParticipants(userEvent){
4734        const { event, volunteers } = this.state;
4735        let participants = [], opt, participant;
4736        for (let i = 0; !!(opt =
            userEvent.target.options[i++]);) {
4737
4738            if (opt.selected) {
4739                for (let j = 0; !!(participant =
                    volunteers[j++].user);){
4740                    if (opt.value === participant.cpf){
4741                        participants.push(participant);
4742                        break;
4743                    }
4744                }
4745            }
4746        }
4747        event.participants = participants;
4748        this.setState({ event });
4749    }
4750
4751    handlerAdd(){
4752        const { event } = this.state;
4753        let training = new Training();
4754        training.isFile = false;
4755        event.trainings.push(training);
4756        this.setState({event})
4757    }
4758
4759    handlerNameAttachemnt(userEvent, index){
4760        const { event } = this.state;
4761        event.trainings[index].name = userEvent.target.value;
4762        this.setState({event});
4763    }
4764
4765    handlerLinkAttachemnt(userEvent, index) {
```

```
4766            const { event } = this.state;
4767            event.trainings[index].link = userEvent.target.value;
4768            this.setState({ event });
4769        }
4770
4771        handlerUploadAttachemnt(userEvent, index){
4772            const file = userEvent.target.files[0];
4773            console.log(file);
4774        }
4775
4776        handlerRemove(index){
4777            const { event } = this.state;
4778            event.trainings.splice(index, 1);
4779            this.setState({event});
4780        }
4781
4782        handlerIsFile(value, index){
4783            const { event } = this.state;
4784            event.trainings[index].isFile = value
4785            this.setState({ event });
4786        }
4787
4788        handlerSubmit(){
4789            const { afterSubmit } = this.props
4790            let { event } = this.state;
4791            const [start, end] = [ event.startOccurrence,
                    event.endOccurrence ]
4792            event.startOccurrence = event.startOccurrence.toJSON()
4793            event.endOccurrence = event.endOccurrence.toJSON()
4794            postRequest(
4795                '/event',
4796                event,
4797                afterSubmit,
4798                () => [event.startOccurrence, event.endOccurrence] =
                        [start, end]
4799            );
4800        }
4801
4802        render(){
4803            const { isOpen, close } = this.props;
4804            const { event, volunteers, schools, typesEvent,
                    trainingsOfType } = this.state;
4805            const dateTitle = event.allDay ?
4806                event.startOccurrence.toLocaleString() :
```

```
4807            event.startOccurrence.toLocaleString() + ' - ' +
                    event.endOccurrence.toLocaleString();
4808        const { codSchool } = event.school;
4809        const { trainings, type, address } = event;
4810        return (
4811            <Modal isOpen={isOpen} toggle={close} style={{width:
                    'max-content'}}>
4812                <ModalHeader toggle={close}>Novo Evento -
                        {dateTitle}</ModalHeader>
4813                <ModalBody>
4814                    <Wizard onCancel={close}
                            submitLabel="cadastrar"
                            onSubmit={this.handlerSubmit}>
4815                        <div>
4816                            <h3>Dados do evento</h3>
4817                            <Row>
4818                                <Col>
4819                                    <Input id="type" type="select"
                                            label="Tipo de Evento"
                                            invalidMessage="Tipo de
                                            Evento  obrigatrio"
                                            value={type.type}
                                            onChange={this.handlerType}
                                            required >
4820                                        <option
                                                value="">Selecione</option>
4821                                        {
4822                                            typesEvent.map(
4823                                                typeEvent => <option
                                                        key={typeEvent.type}
                                                        value={typeEvent.type}>{typeEvent
4824                                                )
4825                                        }
4826                                    </Input>
4827                                </Col>
4828                                <Col md="3">
4829                                    <Switch id="evaluate"
                                            label="Avaliado"
                                            value={event.evaluate}
                                            onChange={this.handlerEvaluate}
                                            />
4830                                </Col>
4831                            </Row>
4832                            <Input id="filter" type="select"
                                    label="Escola" value={codSchool}
```

```
                    onChange={this.handlerSelectSchool}>
4833                <option>Selecione</option>
4834                {
4835                    schools.map(school => <option
                        key={school.codSchool}
                        value={school.codSchool}>{school.name}
4836                }
4837            </Input>
4838            <Input id="city" label="Cidade"
                invalidMessage="Cidade
                obrigatrio" value={address.city}
                disabled={!codSchool}
                onChange={this.handlerCity}
                required />
4839            <Input id="neighborhood"
                label="Bairro"
                invalidMessage="Bairro
                obrigatrio"
                value={address.neighborhood}
                disabled={!codSchool}
                onChange={this.handlerNeighborhood}
                required />
4840            <Row>
4841                <Col>
4842                    <Input id="street"
                        label="Lougradouro"
                        invalidMessage="Lougradouro
                         obrigatrio"
                        value={address.street}
                        disabled={!codSchool}
                        onChange={this.handlerStreet}
                        required />
4843                </Col>
4844                <Col md="3">
4845                    <Input id="number"
                        label="Nmero"
                        value={address.number}
                        onChange={this.handlerNumber}
                        disabled={!codSchool} />
4846                </Col>
4847            </Row>
4848            <hr className="row" />
4849        </div>
4850        <div>
4851            <h3>Participantes</h3>
```

```
4852                        <Input id="type" type="select"
                                label="Selecione"
                                invalidMessage="Tipo de Evento
                                obrigatrio"
                                onChange={this.handlerParticipants}
                                style={{ height: '500px'}}
                                multiple required
                                disabled={!codSchool} >
4853                            {
4854                                volunteers.map(
4855                                    volunteer => {
4856                                        const { user } =
                                                volunteer;
4857                                        return (
4858                                            <option
                                                    key={user.cpf}
                                                    value={user.cpf}>{formatter.cpf(u
                                                    -
                                                    {user.name}</option>
4859                                        );
4860                                    }
4861                                )
4862                            }
4863                        </Input>
4864                        <hr className="row" />
4865                    </div>
4866                    <div>
4867                        <h3>Materias</h3>
4868                        {
4869                            trainingsOfType.length?
4870                                trainingsOfType.map(
4871                                    training => (
4872                                            <Col
                                                    key={training.codTraining}>
4873                                            {training.link &&
                                                <a
                                                    className="btn
                                                    btn-info"
                                                    href={training.link}
                                                    target="_blank">{training.nam
4874                                            </Col>
4875                                    )
4876                                ) : (
4877                                    <div>
```

```
4878                                    <strong>Para esse tipo
                                            de evento no h
                                            materiais prviamente
                                            cadastrados</strong>
4879                                </div>
4880                            )
4881                        }
4882                        <hr className="row" />
4883                    </div>
4884                    <div>
4885                        <h3>Materiais Extra</h3>
4886                        {
4887                            trainings.map((training, index) => (
4888                                <Fragment>
4889                                    <Row>
4890                                        <Col>
4891                                            <Input
                                                    id={`training-${index}`}
                                                    label="Nome"
                                                    invalidMessage="Nome
                                                     Obrigatrio"
                                                    value={training.name}
                                                    onChange={event
                                                    =>
                                                    this.handlerNameAttachemnt(
                                                    index)}
                                                    required/>
4892                                        </Col>
4893                                        <Col md="5">
4894                                            <Switch
                                                    id={`file-${index}`}
                                                    label="Arquivo
                                                    pra Upload"
                                                    value={training.isFile}
                                                    onChange={value
                                                    =>
                                                    this.handlerIsFile(value,
                                                    index)} />
4895                                        </Col>
4896                                    </Row>
4897                                    <Row>
4898                                        <Col>
4899                                            {
4900                                                training.isFile ?
4901                                                    (
```

```
4902                                                    <FileInput
                                                            id={`upload-${index}`}
                                                            label="Anexar
                                                            arquivo"
                                                            invalidMessage="Upload

                                                            Obrigatrio"
                                                            onChange={event
                                                            =>
                                                            this.handlerUploadAtta
                                                            index)}
                                                            accept="application/pd
                                                            required/>
4903                                                    ) : (
4904                                                    <Input
                                                            id={`link-${index}`}
                                                            label="Link"
                                                            invalidMessage="Link

                                                            Obrigatrio"
                                                            value={training.link}
                                                            onChange={event
                                                            =>
                                                            this.handlerLinkAttach
                                                            index)}
                                                            required/>
4905                                                    )
4906                                                }
4907                                            </Col>
4908                                            <Col md="3">
4909                                                {!training.isFile &&
                                                        <label> </label>}
4910                                                <Button outline
                                                        color="secondary"
                                                        onClick={() =>
                                                        this.handlerRemove(index)}>Remove
4911                                            </Col>
4912                                        </Row>
4913                                    </Fragment>
4914                            ))
4915                        }
4916                        <Row>
4917                            <Col>
4918                                <Button outline
                                        color="secondary"
```

```
                                        onClick={this.handlerAdd}>Incluir</But
                                    </Col>
                                </Row>
                                <hr className="row" />
                            </div>
                        </Wizard>
                    </ModalBody>
                </Modal>
            );
        }
    }
```

webapp/src/pages/protected/dashboard/admin/event/form/index.js

```
import Form from './Form.jsx';
export default Form;
```

webapp/src/pages/protected/dashboard/admin/event/form/form.css

```
.modal-content {
  width: inherit; }
```

webapp/src/pages/protected/dashboard/admin/event/form/form.sass

```
.modal-content
    width: inherit
```

webapp/src/pages/protected/dashboard/admin/event/list.css

```
.rbc-day-bg, .rbc-events-container {
  cursor: pointer; }
```

webapp/src/pages/protected/dashboard/admin/event/index.js

```
import List from './List.jsx';
export default List;
```

webapp/src/pages/protected/dashboard/admin/event/Details.jsx

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { getRequest, deleteRequest } from
    '../../../../../utils/http';
```

```
4962 import { Address } from '../../../../../model';
4963 import { Button, Modal, ModalHeader, ModalBody, ModalFooter,
         Row, Col } from 'reactstrap';
4964
4965
4966 export default class extends Component{
4967     constructor(){
4968         super();
4969         this.state = {
4970             eventDetails : null
4971         }
4972         this.handlerDelete = this.handlerDelete.bind(this);
4973     }
4974
4975     static propTypes = {
4976         code: PropTypes.number.isRequired,
4977         isOpen: PropTypes.bool.isRequired,
4978         close: PropTypes.func.isRequired,
4979         afterDelete: PropTypes.func.isRequired,
4980     };
4981
4982     componentWillMount() {
4983         const { code } = this.props;
4984         getRequest(
4985             `/event/${code}`,
4986             res => {
4987                 let eventDetails = res.data;
4988                 eventDetails.title = eventDetails.type.name + ' -
                        ' + new
                        Date(eventDetails.startOccurrence).toLocaleString();
4989                 this.setState({ eventDetails });
4990             }
4991         );
4992     }
4993
4994     handlerDelete() {
4995         const { code, afterDelete} = this.props;
4996         deleteRequest(`/event/${code}`, afterDelete);
4997     }
4998
4999     render(){
5000         const { isOpen, close } = this.props;
5001         const { eventDetails } = this.state;
5002         if (eventDetails){
```

```
5003            const { address: eventAddress, trainings:
                    eventTrainings, participants, type } =
                    eventDetails;
5004            const { trainings: typeTrainings } = type;
5005            const trainings =
                    eventTrainings.concat(typeTrainings);
5006            const duration = Math.ceil(
5007                Math.abs(
5008                    eventDetails.startOccurrence -
                            eventDetails.endOccurrence
5009                ) / (1000 * 3600)
5010            );
5011            const address = new Address();
5012            Object.assign(address, eventAddress);
5013            return (
5014                <Modal toggle={close} isOpen={isOpen} >
5015                    <ModalHeader
                            toggle={close}>{eventDetails.title}</ModalHeader>
5016                    <ModalBody>
5017                        <Row>
5018                            <Col>
5019                                <strong> Durao :</strong>
                                        {eventDetails.allDay ? 'Dia
                                        Inteiro' : duration + ' horas
                                        (aproximadamente)'}
5020                            </Col>
5021                        </Row>
5022                        <br />
5023                        <Row>
5024                            <Col>
5025                                <strong>Endereo:</strong>
5026                            </Col>
5027                        </Row>
5028                        <Row>
5029                            <Col>
5030                                {address.toString()}
5031                            </Col>
5032                        </Row>
5033                        <br />
5034                        <Row>
5035                            <Col>
5036                                <strong>Participantes:</strong>
5037                            </Col>
5038                        </Row>
5039                        <Row>
```

```
5040                        {
5041                            participants.map(
5042                                participant => (
5043                                    <Col md="4"
                                            style={{marginTop:
                                            '20px'}}>
5044                                        {participant.cpf} -
                                                {participant.name}
5045                                    </Col>
5046                                )
5047                            )
5048                        }
5049                    </Row>
5050                    <br />
5051                    <Row>
5052                        <Col>
5053                            <strong>Materiais em anexo:</strong>
5054                        </Col>
5055                    </Row>
5056                    <Row>
5057                        <Col>
5058                            {
5059                                trainings.length ?
5060                                (
5061                                    <Row>
5062                                        {
5063                                            trainings.map(
5064                                                training => (
5065                                                    <Col
                                                            key={training.codTraining}
5066                                                        {training.link
                                                            && <a
                                                            className="btn
                                                            btn-info"
                                                            href={training.link}
                                                            target="_blank">{train
5067                                                    </Col>
5068                                                )
5069                                            )
5070                                        }
5071                                    </Row>
5072                                ): 'Nenhum anexo para esse
                                        evento.'
5073                            }
5074                        </Col>
```

```
                        </Row>
                    </ModalBody>
                    <ModalFooter style={{display: 'inline'}}>
                        <Row>
                            <Col>
                                <Button color="danger"
                                    onClick={this.handlerDelete}>Excluir</But
                            </Col>
                            <Col>
                                <Button color="primary"
                                    className="float-right"
                                    onClick={close}>Fechar</Button>
                            </Col>
                        </Row>
                    </ModalFooter>
                </Modal>
            )
        } else {
            return (
                <Modal toggle={close} isOpen={isOpen} ></Modal>
            );
        }
    }
}

webapp/src/pages/protected/dashboard/admin/event/List.jsx

import React, { Component } from 'react';
import { Calendar, Input } from '../../../../../components';
import { Row, Col, Form } from 'reactstrap';
import BigCalendar from 'react-big-calendar';
import FormEvent from './form';
import Details from './Details.jsx'
import { Event } from '../../../../../model';
import { getRequest } from '../../../../../utils/http';
import axios from 'axios';
import './list.css'

const mapEvents =
    events => events.map(
        event =>{
            event.end = new Date(event.endOccurrence);
            event.start = new Date(event.startOccurrence);
            event.title = event.type.name + ' - ' +
                event.start.toLocaleString()
```

```
5116            return event
5117        }
5118    );
5119
5120 export default class extends Component{
5121     constructor(){
5122         super();
5123         this.state = {
5124             events: [],
5125             projects: [],
5126             isOpenForm: false,
5127             isOpenDetails: false,
5128             newEvent: new Event(),
5129             codEvent: null,
5130             codProject: ''
5131         };
5132         this.closeForm = this.closeForm.bind(this);
5133         this.closeDetails = this.closeDetails.bind(this);
5134         this.handlerSelectSlot =
                 this.handlerSelectSlot.bind(this);
5135         this.handlerSelectEvent =
                 this.handlerSelectEvent.bind(this);
5136         this.handlerSelectProject =
                 this.handlerSelectProject.bind(this);
5137         this.componentWillMount =
                 this.componentWillMount.bind(this);
5138     }
5139
5140     componentWillMount(){
5141         const { filterBySchool, filterByVolunteer } = this.props;
5142         axios.all([
5143             getRequest(
5144                 `/event?cod_cpf=${filterByVolunteer.join(',')}&cod_school=${filter
5145                 res => res.data
5146             ),
5147             getRequest('/project', res => res.data)
5148         ]).then(
5149             res => this.setState({
5150                 events: mapEvents(res[0]),
5151                 projects: res[1],
5152                 isOpenForm: false,
5153                 newEvent: new Event(),
5154                 isOpenDetails: false,
5155                 codEvent: null
5156             })
```

```
5157            )
5158        }
5159
5160        closeForm(){
5161            this.setState({isOpenForm: false, newEvent: new
                    Event()});
5162        }
5163
5164        closeDetails(){
5165            this.setState({ isOpenDetails: false, codEvent: null });
5166        }
5167
5168        handlerSelectSlot({ start, end }){
5169            if(start < new Date()){
5170                return;
5171            }
5172            const { newEvent } = this.state;
5173            newEvent.startOccurrence = start;
5174            newEvent.endOccurrence = end;
5175            this.setState({isOpenForm: true, newEvent});
5176        }
5177
5178        handlerSelectEvent(event){
5179            const { codEvent } = event
5180            this.setState({ isOpenDetails: true, codEvent });
5181        }
5182
5183        handlerSelectProject(event){
5184            const { value } = event.target;
5185            const { filterBySchool, filterByVolunteer } = this.props;
5186            getRequest(
5187                `/event?cod_cpf=${filterByVolunteer.join(',')}&cod_school=${filt
5188                res => this.setState({ events: mapEvents(res.data),
                        codProject: value})
5189            );
5190        }
5191
5192        render(){
5193            const { events, projects, isOpenForm, isOpenDetails,
                    newEvent, codEvent, codProject } = this.state;
5194            return (
5195                <Row>
5196                    <Col>
5197                        <Row>
5198                            <Col>
```

```
5199                        <Form inline>
5200                            <Input id="filter" type="select"
                                    value={codProject}
                                    label="Projeto:"
                                    onChange={this.handlerSelectProject}>
5201                                <option
                                        value="">Selecione</option>
5202                                {
5203                                    projects.map(project =>
                                            <option
                                            key={project.codProject}
                                            value={project.codProject}>{project.name
5204                                }
5205                            </Input>
5206                        </Form>
5207                    </Col>
5208                </Row>
5209                <br />
5210                <Row style={{ height: '600px', marginBottom:
                        '1%' }}>
5211                    <Col>
5212                        <Calendar events={events}
                                onSelectEvent={this.handlerSelectEvent}
                                onSelectSlot={this.handlerSelectSlot}
                                selectable
                                defaultView={BigCalendar.Views.WEEK}
                                />
5213                        {
5214                            isOpenForm ?
5215                                <FormEvent isOpen={isOpenForm}
                                        close={this.closeForm}
                                        event={newEvent}
                                        afterSubmit={this.componentWillMount}
                                        /> :
5216                                null
5217                        }
5218                        {
5219                            isOpenDetails ?
5220                                <Details code={codEvent}
                                        isOpen={isOpenDetails}
                                        close={this.closeDetails}
                                        afterDelete={this.componentWillMount}
                                        /> :
5221                                null
5222                        }
```

```
                        </Col>
                    </Row>
                </Col>
            </Row>
        );
    }
}
```

webapp/src/pages/protected/dashboard/admin/event/list.sass

```
.rbc-day-bg, .rbc-events-container
    cursor: pointer
```

webapp/src/pages/protected/dashboard/admin/company/Form.jsx

```
import React, { Component } from 'react';
import Company from '../../../../../model/company';
import PropTypes from 'prop-types';
import { postRequest } from '../../../../../utils/http';
import { Row, Col, Modal, ModalHeader, ModalBody } from
    'reactstrap';
import { Wizard, Input, MaskInput, DatePicker } from
    '../../../../../components';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            company: new Company(),
            confirmPassword: ''
        };
        this.handlerName = this.handlerName.bind(this);
        this.handlerPhone = this.handlerPhone.bind(this);
        this.handlerCnpj = this.handlerCnpj.bind(this);
        this.handlerNameResponsible =
            this.handlerNameResponsible.bind(this);
        this.handlerCpf = this.handlerCpf.bind(this);
        this.handlerPhoneResponsible =
            this.handlerPhoneResponsible.bind(this);
        this.handlerJob = this.handlerJob.bind(this);
        this.handlerEmail = this.handlerEmail.bind(this);
        this.handlerBirth = this.handlerBirth.bind(this);
        this.handlerPassword = this.handlerPassword.bind(this);
        this.handlerConfirmPassword =
            this.handlerConfirmPassword.bind(this);
```

```
5263            this.handlerCity = this.handlerCity.bind(this);
5264            this.handlerNeighborhood =
                    this.handlerNeighborhood.bind(this);
5265            this.handlerStreet = this.handlerStreet.bind(this);
5266            this.handlerNumber = this.handlerNumber.bind(this);
5267            this.handlerSubmit = this.handlerSubmit.bind(this);
5268        }
5269
5270        static propTypes = {
5271            afterSubmit: PropTypes.func.isRequired,
5272            isOpen: PropTypes.bool.isRequired,
5273            close: PropTypes.func.isRequired
5274        }
5275
5276
5277        //company
5278        handlerName(event){
5279            const { company } = this.state;
5280            company.name = event.target.value;
5281            this.setState({ company });
5282        }
5283
5284        handlerCnpj(event){
5285            const { company } = this.state;
5286            company.cnpj = event.target.value.replace(/[./-]/g, '');
5287            this.setState({ company });
5288        }
5289
5290        handlerPhone(event){
5291            const { company } = this.state;
5292            company.phone = event.target.value.replace(/[()-]/g, '');
5293            this.setState({ company });
5294        }
5295
5296        // responsible
5297        handlerNameResponsible(event){
5298            const { company } = this.state;
5299            const { responsible } = company;
5300            responsible.name = event.target.value;
5301            company.responsible = responsible;
5302            this.setState({ company });
5303        }
5304
5305        handlerCpf(event){
5306            const { company } = this.state;
```

```
5307            const { responsible } = company;
5308            responsible.cpf = event.target.value.replace(/[.-]/g,
                    '');
5309            company.responsible = responsible;
5310            this.setState({ company });
5311        }
5312
5313        handlerPhoneResponsible(event) {
5314            const { company } = this.state;
5315            const { responsible } = company;
5316            responsible.phone = event.target.value.replace(/[()-]/g,
                    '');
5317            company.responsible = responsible;
5318            this.setState({ company });
5319        }
5320
5321        handlerJob(event){
5322            const { company } = this.state;
5323            const { responsible } = company;
5324            responsible.job = event.target.value;
5325            company.responsible = responsible;
5326            this.setState({ company });
5327        }
5328
5329        handlerEmail(event){
5330            const { company } = this.state;
5331            const { responsible } = company;
5332            responsible.email = event.target.value;
5333            company.responsible = responsible;
5334            this.setState({ company });
5335        }
5336
5337        handlerBirth(event){
5338            const { company } = this.state;
5339            const { responsible } = company;
5340            const { value } = event.target;
5341            responsible.birth = value ? new Date(value) : null;
5342            company.responsible = responsible;
5343            this.setState({ company });
5344        }
5345
5346        handlerPassword(event){
5347            const { company } = this.state;
5348            const { responsible } = company;
5349            responsible.password = event.target.value;
```

```
        company.responsible = responsible;
        this.setState({ company });
    }

    handlerConfirmPassword(event){
        let { confirmPassword } = this.state;
        confirmPassword = event.target.value;
        this.setState({ confirmPassword });
    }

    //address
    handlerCity(event){
        const { company } = this.state;
        const { responsible, address } = company;
        address.city = event.target.value;
        company.address = responsible.address = address;
        company.responsible = responsible;
        this.setState({ company });
    }

    handlerNeighborhood(event){
        const { company } = this.state;
        const { responsible, address } = company;
        address.neighborhood = event.target.value;
        company.address = responsible.address = address;
        company.responsible = responsible;
        this.setState({ company });
    }

    handlerStreet(event){
        const { company } = this.state;
        const { responsible, address } = company;
        address.street = event.target.value;
        company.address = responsible.address = address;
        company.responsible = responsible;
        this.setState({ company });
    }

    handlerNumber(event){
        const { company } = this.state;
        const { responsible, address } = company;
        address.number = event.target.value;
        company.address = responsible.address = address;
        company.responsible = responsible;
        this.setState({ company });
```

```
5395        }
5396
5397        handlerSubmit(){
5398            const { afterSubmit, close } = this.props;
5399            const { company } = this.state;
5400            postRequest(
5401                '/company',
5402                company,
5403                () => {
5404                    afterSubmit();
5405                    close();
5406                }
5407            );
5408        }
5409
5410        render(){
5411            const { close, isOpen } = this.props;
5412            const { company, confirmPassword } = this.state;
5413            const { address, responsible } = company;
5414            const maxDate = new Date().toJSON().split('T')[0];
5415            return (
5416                <Modal isOpen={isOpen} toggle={close}>
5417                    <ModalHeader toggle={close}>Nova
                            Empresa</ModalHeader>
5418                    <ModalBody>
5419                        <Wizard onCancel={close}
                                submitLabel="cadastrar"
                                onSubmit={this.handlerSubmit}>
5420                            <div>
5421                                <h3>Dados da Empresa</h3>
5422                                <Input id="nameCompany" label="Nome
                                        Fantasia" invalidMessage="Nome
                                        Fantasia  obrigatrio"
                                        value={company.name}
                                        onChange={this.handlerName}
                                        required/>
5423                                <MaskInput id="cnpj" label="CNPJ"
                                        invalidMessage="CNPJ  obrigatrio"
                                        mask="99.999.999/9999-99"
                                        value={company.cnpj}
                                        onChange={this.handlerCnpj}
                                        required/>
5424                                <MaskInput id="phone" label="Contato
                                        da Empresa"
                                        invalidMessage="Contato da Empresa
```

```
                        obrigatrio" value={company.phone}
                        mask="(99)9999-9999"
                        onChange={this.handlerPhone}
                        required/>
5425                <hr className="row"/>
5426            </div>
5427            <div>
5428                <h3>Dados do responsvel</h3>
5429                <Input id="nameReponsible" label="Nome
                        Completo" invalidMessage="Nome
                        Completo  obrigatrio"
                        value={responsible.name}
                        onChange={this.handlerNameResponsible}
                        required/>
5430                <MaskInput id="cpf" label="CPF"
                        invalidMessage="CPF  obrigatrio"
                        value={responsible.cpf}
                        mask="999.999.999-99"
                        onChange={this.handlerCpf}
                        required />
5431                <MaskInput id="phoneResponsible"
                        label="Contato do Responsvel"
                        invalidMessage="Contato do
                        Responsvel  obrigatrio"
                        value={responsible.phone}
                        mask="(99)9999-9999"
                        onChange={this.handlerPhoneResponsible}
                        required/>
5432                <Input id="job" label="Cargo do
                        Responsvel" invalidMessage="Cargo
                        do Responsvel  obrigatrio"
                        value={responsible.job}
                        onChange={this.handlerJob}
                        required/>
5433                <DatePicker id="birth"
                        label="Nascimento"
                        invalidMessage="Nascimento
                        obrigatrio" max={maxDate}
                        onChange={this.handlerBirth}
                        required/>
5434                <Input id="email" type="email"
                        label="Email"
                        invalidMessage="Email  obrigatrio"
                        value={responsible.email}
                        onChange={this.handlerEmail}
```

```
                    required/>
5435                <Row>
5436                    <Col>
5437                        <Input id="password"
                                type="password"
                                label="Senha"
                                invalidMessage="Senha
                                obrigatrio"
                                value={responsible.password}
                                onChange={this.handlerPassword}
                                required/>
5438                    </Col>
5439                    <Col>
5440                        <Input id="passwordConfirm"
                                type="password"
                                label="Confirmar Senha"
                                invalidMessage="As senhas
                                devem ser idnticas"
                                value={confirmPassword}
                                onChange={this.handlerConfirmPassword}
                                required/>
5441                    </Col>
5442                </Row>
5443                <hr className="row"/>
5444            </div>
5445            <div>
5446                <h3>Endereo da Empresa</h3>
5447                <Input id="city" label="Cidade"
                        invalidMessage="Cidade
                        obrigatrio" value={address.city}
                        onChange={this.handlerCity}
                        required />
5448                <Input id="neighborhood"
                        label="Bairro"
                        invalidMessage="Bairro
                        obrigatrio"
                        value={address.neighborhood}
                        onChange={this.handlerNeighborhood}
                        required />
5449                <Row>
5450                    <Col>
5451                        <Input id="street"
                                label="Lougradouro"
                                invalidMessage="Lougradouro
                                 obrigatrio"
```

```
                                        value={address.street}
                                        onChange={this.handlerStreet}
                                        required />
                                </Col>
                                <Col md="3">
                                    <Input id="number"
                                        label="Nmero"
                                        value={address.number}
                                        onChange={this.handlerNumber}
                                        />
                                </Col>
                            </Row>
                            <hr className="row"/>
                        </div>
                    </Wizard>
                </ModalBody>
            </Modal>
        );
    }
}
```

webapp/src/pages/protected/dashboard/admin/company/index.js

```
import List from './List.jsx';
export default List;
```

webapp/src/pages/protected/dashboard/admin/company/ListItem.jsx

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { deleteRequest } from '../../../../../utils/http';
import { Checkbox } from '../../../../../components';
import { Row, Col, ListGroupItem } from 'reactstrap';
import { FaTrash } from 'react-icons/fa';
import Details from './Details.jsx';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            isOpen: false
        }
        this.handlerDelete = this.handlerDelete.bind(this);
        this.handlerDetails = this.handlerDetails.bind(this);
    }
```

```
5490
5491      static propTypes = {
5492          company: PropTypes.object.isRequired,
5493          afterDelete: PropTypes.func.isRequired
5494      }
5495
5496      handlerDelete(){
5497          const { company, afterDelete } = this.props;
5498          deleteRequest(‘/company/${company.cnpj}‘, afterDelete);
5499      }
5500
5501      handlerDetails(){
5502          const { isOpen } = this.state;
5503          this.setState({ isOpen: !isOpen });
5504      }
5505
5506      render(){
5507          const { company, onSelectCompany } = this.props;
5508          const { isOpen } = this.state;
5509          return (
5510              <ListGroupItem key={company.cnpj}>
5511              <Row>
5512                      <Col md="1">
5513                          <Checkbox value={company.cnpj}
                                onChange={onSelectCompany}/>
5514                      </Col>
5515                      <Col md="9" onClick={this.handlerDetails}
                            className="text-ellipsis" style={{
                            cursor: ’pointer’ }}>
5516                          {company.name}
5517                      </Col>
5518                      <Col>
5519                          <FaTrash style={{ cursor: ’pointer’ }}
                                color="red"
                                onClick={this.handlerDelete} />
5520                      </Col>
5521                  </Row>
5522                  {
5523                      isOpen ? <Details cnpj={company.cnpj}
                            isOpen={isOpen}
                            close={this.handlerDetails}/> : null
5524                  }
5525              </ListGroupItem>
5526          );
5527      }
```

```
}

webapp/src/pages/protected/dashboard/admin/company/Add.jsx

import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { Button } from 'reactstrap';
import { FaPlus } from 'react-icons/fa';
import Form from './Form.jsx';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            isOpen: false
        };
    }

    static propTypes = {
        afterSubmit: PropTypes.func.isRequired
    }

    handlerAdd(){
        const { isOpen } = this.state;
        this.setState({ isOpen: !isOpen });
    }

    render(){
        const { isOpen } = this.state;
        const { afterSubmit } = this.props;
        return (
            <div>
                <Button type="button" color="primary"
                    onClick={this.handlerAdd.bind(this)}>
                    <FaPlus />
                </Button>
                <Form isOpen={isOpen} afterSubmit={afterSubmit}
                    close={this.handlerAdd.bind(this)}/>
            </div>
        );
    }
}

webapp/src/pages/protected/dashboard/admin/company/Details.jsx
```

```
5571 import React, { Component } from 'react';
5572 import PropTypes from 'prop-types';
5573 import Address from '../../../../../model/address';
5574 import { getRequest } from '../../../../../utils/http';
5575 import formatter from '../../../../../utils/formatter';
5576 import { Button, Modal, ModalHeader, ModalBody, ModalFooter,
         Row, Col } from 'reactstrap';
5577 import pic from './pic.jpeg';
5578
5579 export default class extends Component {
5580     constructor(){
5581         super();
5582         this.state = {
5583             company: null
5584         }
5585     }
5586     static propTypes = {
5587         cnpj: PropTypes.string.isRequired,
5588         isOpen: PropTypes.bool.isRequired,
5589         close: PropTypes.func.isRequired
5590     };
5591
5592     componentWillMount() {
5593         const { cnpj } = this.props;
5594         getRequest(`/company/${cnpj}`, res => this.setState({
              company: res.data }));
5595     }
5596
5597     render() {
5598         const { isOpen, close } = this.props;
5599         const { company } = this.state;
5600         if (company) {
5601             const address = new Address();
5602             Object.assign(address, company.address);
5603             return (
5604                 <Modal toggle={close} isOpen={isOpen} >
5605                     <ModalHeader toggle={close}>{company.name} -
                          <small>{formatter.cnpj(company.cnpj)}</small></Modal
5606                     <ModalBody>
5607                         <Row>
5608                             <Col md={1}>
5609                                 <img src={pic} alt=""
                                        className="rounded-circle"
                                        width="75px" />
5610                             </Col>
```

```
                        <Col md="2" />
                        <Col>
                            <Row>

                                <Col>
                                    <strong>Endereo:</strong>
                                </Col>
                            </Row>
                            <Row>
                                <Col>
                                    {address.toString()}
                                </Col>
                            </Row>
                            <br />
                            <Row>
                                <Col>
                                    <strong>Contato:</strong>
                                </Col>
                            </Row>
                            <Row>
                                <Col>
                                    {company.responsible.email}
                                        /
                                        {formatter.phone(company.phone)}
                                </Col>
                            </Row>
                        </Col>
                    </Row>
                    <br />
                    <Row>
                        <Col md="3" />
                        <Col>
                            <strong>Responsvel:</strong>
                        </Col>
                    </Row>
                    <Row>
                        <Col md="3" />
                        <Col>
                            <div>{company.responsible.name}</div>
                            <small>{formatter.cpf(company.responsible.cpf)}</sma
                        </Col>
                    </Row>
                </ModalBody>
                <ModalFooter>
```

```
                    <Button color="primary"
                        onClick={close}>Fechar</Button>
                </ModalFooter>
            </Modal>
        );
    } else {
        return (
            <Modal toggle={close} isOpen={isOpen} ></Modal>
        );
    }
  }
}
```

webapp/src/pages/protected/dashboard/admin/company/List.jsx

```
import React, { Component } from 'react';
import { getRequest } from '../../../../../utils/http';
import { Row, Col, ListGroup, ListGroupItem, Form } from
    'reactstrap';
import { Pagination, Filter } from '../../../../../components';
import CompanyItem from './ListItem.jsx';
import Add from './Add.jsx';

export default class extends Component {
    constructor() {
        super()
        this.state = {
            filter: '',
            companies: [],
            count: 0,
            pages: 0
        }
        this.onChangePage = this.onChangePage.bind(this);
        this.handlerFilter = this.handlerFilter.bind(this);
        this.componentWillMount =
            this.componentWillMount.bind(this)
    }

    componentWillMount() {
        getRequest(
            '/company/page/0',
            res => {
                const page = res.data;
                this.setState({
                    companies: page.data,
```

```
                count: page.count,
                pages: page.count / 5
            });
        }
    );
}

handlerFilter(event) {
    const filter = event.target.value;
    if (filter.length >= 3 || filter.length === 0){
        getRequest(
            `/company/page/0?filter=${filter}`,
            res => {
                const page = res.data;
                this.setState({
                    companies: page.data,
                    count: page.count,
                    pages: page.count / 5,
                    filter
                });
            }
        );
    } else {
        this.setState({filter});
    }
}

onChangePage(pageNumber){
    const { filter } = this.state;
    getRequest(
    `/company/page/${pageNumber}?filter=${filter.lenght > 3
        ? filter : ''}`,
        res => {
            const page = res.data;
            this.setState({
                companies: page.data,
                count: page.count,
                pages: page.count / 5
            });
        }
    );
}

render() {
    const { companies, pages, count } = this.state;
```

```
5739            const { onSelectCompany } = this.props;
5740            return (
5741                <div>
5742                    <Row>
5743                        <Col>
5744                            <h3>
5745                                Empresas Envolvidas
5746                            </h3>
5747                        </Col>
5748                        <Col md="2">
5749                            <Add
                                    afterSubmit={this.componentWillMount}/>
5750                        </Col>
5751                    </Row>
5752                    <Filter handlerFilter={this.handlerFilter}/>
5753                    <Row>
5754                        <Col>
5755                            <ListGroup>
5756                                {
5757                                    companies.length ?
5758                                        companies.map(
5759                                            company =>
5760                                                <CompanyItem
                                                        onSelectCompany={onSelectCompa
                                                        key={company.cnpj}
                                                        company={company}
                                                        afterDelete={this.componentWil
5761                                        ) : (
5762                                            <ListGroupItem>
5763                                                <strong>
5764                                                    Nenhuma Empresa
                                                            Cadastrada
5765                                                </strong>
5766                                            </ListGroupItem>
5767                                        )
5768                                }
5769                            </ListGroup>
5770                        </Col>
5771                    </Row>
5772                    <Pagination pages={pages} count={count}
                            onChangePage={this.onChangePage} />
5773                </div>
5774            );
5775        }
5776    }
```

webapp/src/pages/protected/dashboard/admin/Page.jsx

```
import React, { Component, Fragment } from 'react';
import ListCompany from './company';
import ListVolunteer from './volunteer';
import ListEvent from './event'
import ListSchool from './school';
import { Row, Col } from 'reactstrap';

const Hr = () => (
    <Fragment>
        <br />
        <hr />
        <br />
    </Fragment>
);

export default class extends Component{
    constructor(){
        super();
        this.state = {
            filterByCompany: [],
            filterBySchool: [],
            filterByVolunteer: []
        }
        this.onSelectCompany = this.onSelectCompany.bind(this);
        this.onSelectSchool = this.onSelectSchool.bind(this);
        this.onSelectVolunteer =
            this.onSelectVolunteer.bind(this);
    }

    onSelectCompany(event){
        const { filterByCompany } = this.state;
        const { value } = event.target;
        if(filterByCompany.includes(value)){
            filterByCompany.splice(filterByCompany.indexOf(value),
                1);
        } else {
            filterByCompany.push(value);
        }
        this.setState({ filterByCompany });
    }

    onSelectSchool(event) {
```

```
5820            const { filterBySchool } = this.state;
5821            const { value } = event.target;
5822            if (filterBySchool.includes(value)) {
5823                filterBySchool.splice(filterBySchool.indexOf(value),
                        1);
5824            } else {
5825                filterBySchool.push(value);
5826            }
5827            this.setState({ filterBySchool });
5828        }
5829
5830        onSelectVolunteer(event) {
5831            const { filterByVolunteer } = this.state;
5832            const { value } = event.target;
5833            if (filterByVolunteer.includes(value)) {
5834                filterByVolunteer.splice(filterByVolunteer.indexOf(value),
                        1);
5835            } else {
5836                filterByVolunteer.push(value);
5837            }
5838            this.setState({ filterByVolunteer });
5839        }
5840
5841        render(){
5842            const { filterByCompany, filterBySchool,
                     filterByVolunteer } = this.state;
5843            return (
5844                <Row>
5845                    <Col md="1" />
5846                    <Col>
5847                        <Row>
5848                            <Col />
5849                        </Row>
5850                        <Hr />
5851                        <Row>
5852                            <Col>
5853                                <ListCompany
                                        onSelectCompany={this.onSelectCompany}/>
5854                            </Col>
5855                            <Col md="1" />
5856                            <Col>
5857                                <ListVolunteer
                                        onSelectVolunteer={this.onSelectVolunteer}
                                        filterByCompany={filterByCompany}
                                        filterBySchool={filterBySchool}/>
```

```
                        </Col>
                        <Col md="1" />
                        <Col>
                            <ListSchool
                                onSelectSchool={this.onSelectSchool}
                                />
                        </Col>
                    </Row>
                    <Hr />
                    <Row>
                        <Col>
                            <ListEvent
                                filterBySchool={filterBySchool}
                                filterByVolunteer={filterByVolunteer}/>
                        </Col>
                    </Row>
                </Col>
                <Col md="1" />
            </Row>
        );
    }
}
```

webapp/src/pages/protected/dashboard/admin/index.js

```
import Page from './Page.jsx';
export default Page;
```

webapp/src/pages/protected/dashboard/volunteer/training/index.js

```
import Step from './Step.jsx';
export default Step;
```

webapp/src/pages/protected/dashboard/volunteer/training/Details.jsx

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { getRequest } from '../../../../../utils/http';
import { Address } from '../../../../../model';
import { Button, Modal, ModalHeader, ModalBody, ModalFooter,
    Row, Col } from 'reactstrap';


export default class extends Component{
    constructor(){
```

```
5898            super();
5899            this.state = {
5900                eventDetails : null
5901            }
5902        }
5903
5904        static propTypes = {
5905            code: PropTypes.number.isRequired,
5906            isOpen: PropTypes.bool.isRequired,
5907            close: PropTypes.func.isRequired
5908        };
5909
5910        componentWillMount() {
5911            const { code } = this.props;
5912            getRequest(
5913                `/event/${code}`,
5914                res => {
5915                    let eventDetails = res.data;
5916                    eventDetails.title = eventDetails.type.name + ' -
                            ' + new
                            Date(eventDetails.startOccurrence).toLocaleString();
5917                    this.setState({ eventDetails });
5918                }
5919            );
5920        }
5921
5922        render(){
5923            const { isOpen, close } = this.props;
5924            const { eventDetails } = this.state;
5925            if (eventDetails){
5926                const { address: eventAddress, trainings } =
                        eventDetails;
5927                const duration = Math.ceil(
5928                    Math.abs(
5929                        eventDetails.startOccurrence -
                                eventDetails.endOccurrence
5930                    ) / (1000 * 3600)
5931                );
5932                const address = new Address();
5933                Object.assign(address, eventAddress);
5934                return (
5935                    <Modal toggle={close} isOpen={isOpen} >
5936                        <ModalHeader
                                toggle={close}>{eventDetails.title}</ModalHeader>
5937                        <ModalBody>
```

```
5938                    <Row>
5939                        <Col>
5940                            <strong> Durao :</strong>
                                    {eventDetails.allDay ? 'Dia
                                    Inteiro' : duration + ' horas
                                    (aproximadamente)'}
5941                        </Col>
5942                    </Row>
5943                    <br />
5944                    <Row>
5945                        <Col>
5946                            <strong>Endereo:</strong>
5947                        </Col>
5948                    </Row>
5949                    <Row>
5950                        <Col>
5951                            {address.toString()}
5952                        </Col>
5953                    </Row>
5954                    <br />
5955                    <Row>
5956                        <Col>
5957                            <strong>Materiais em anexo:</strong>
5958                        </Col>
5959                    </Row>
5960                    <Row>
5961                        <Col>
5962                            {
5963                                trainings.length ?
5964                                    (
5965                                        <Row>
5966                                            {
5967                                                trainings.map(
5968                                                    training => (
5969                                                        <Col
                                                                key={training.codTrain
5970                                                            {training.link
                                                                &&
                                                                <a
                                                                className="btn
                                                                btn-info"
                                                                href={training.linl
                                                                target="_blank">{t
5971                                                        </Col>
5972                                                    )
```

```
                                                )
                                            }
                                        </Row>
                                    ) : 'Nenhum anexo para esse
                                        evento.'
                                }
                            </Col>
                        </Row>
                    </ModalBody>
                    <ModalFooter style={{display: 'inline'}}>
                        <Row>
                            <Col>
                                <Button color="primary"
                                    className="float-right"
                                    onClick={close}>Fechar</Button>
                            </Col>
                        </Row>
                    </ModalFooter>
                </Modal>
            )
        } else {
            return (
                <Modal toggle={close} isOpen={isOpen} ></Modal>
            );
        }
    }
}

webapp/src/pages/protected/dashboard/volunteer/training/Step.jsx

import React, { Component } from 'react';
import { Calendar } from '../../../../../components';
import { Row, Col } from 'reactstrap';
import BigCalendar from 'react-big-calendar';
import Details from './Details.jsx'
import { getRequest } from '../../../../../utils/http';

const mapEvents =
    events => events.map(
        event =>{
            event.end = new Date(event.endOccurrence);
            event.start = new Date(event.startOccurrence);
            event.title = event.type.name + ' - ' +
                event.start.toLocaleString()
            return event
```

```
6014        }
6015    );
6016
6017 export default class extends Component{
6018     constructor(){
6019         super();
6020         this.state = {
6021             events: [],
6022             isOpenDetails: false,
6023             codEvent: null
6024         };
6025         this.closeDetails = this.closeDetails.bind(this);
6026         this.handlerSelectEvent =
                 this.handlerSelectEvent.bind(this);
6027         this.componentWillMount =
                 this.componentWillMount.bind(this);
6028     }
6029
6030     componentWillMount(){
6031         const { cpf } = this.props;
6032         getRequest(
6033             `/volunteer/${cpf}/event/`,
6034             res => this.setState({
6035                 events: mapEvents(res.data),
6036                 isOpenDetails: false,
6037                 codEvent: null
6038             })
6039         );
6040     }
6041
6042     closeDetails(){
6043         this.setState({ isOpenDetails: false, codEvent: null });
6044     }
6045
6046     handlerSelectEvent(event){
6047         const { codEvent } = event
6048         this.setState({ isOpenDetails: true, codEvent });
6049     }
6050
6051     render(){
6052         const { events, isOpenDetails, codEvent } = this.state;
6053         return (
6054             <Row style={{ height: '600px', marginBottom: '1%' }}>
6055                 <Col>
```

```
                    <Calendar events={events}
                        onSelectEvent={this.handlerSelectEvent}
                        defaultView={BigCalendar.Views.WEEK}/>
                    {
                        isOpenDetails ?
                            <Details code={codEvent}
                                isOpen={isOpenDetails}
                                close={this.closeDetails}
                                afterDelete={this.componentWillMount}/>
                                :
                            null
                    }
                </Col>
            </Row>
        );
    }
}
```

webapp/src/pages/protected/dashboard/volunteer/statement/Attachment.jsx

```
import React, { Component } from 'react';
import { Row, Col, Button } from 'reactstrap';
import { ClipLoader } from 'react-spinners';
import { FileInput } from '../../../../../components';
import { postRequest, getRequest } from
    '../../../../../utils/http';
import { saveAs } from 'file-saver/FileSaver';
import axios from 'axios';

export class AttachmentUploader extends Component{
    constructor(){
        super();
        this.state = {
            loading: false,
            success: false
        }
        this.upload = this.upload.bind(this);
    }

    componentWillMount(){
        const { cpf, attachment } = this.props;
        const { codAttachment } = attachment;
        getRequest(`/volunteer/${cpf}/attachment/${codAttachment}`,
            res => this.setState({success: res.data.send}))
    }
```

```
6093
6094        upload(event){
6095            const file = event.target.files[0];
6096            const { attachment, cpf, afterUpload } = this.props;
6097            if (file){
6098                this.setState({loading: true});
6099                const json = JSON.stringify(attachment);
6100                const blob = new Blob([json], {
6101                    type: 'application/json'
6102                });
6103                let form = new FormData();
6104                form.append('attachment', blob);
6105                form.append('file', file);
6106                postRequest(
6107                    `/volunteer/${cpf}/upload`,
6108                    form,
6109                    res => {
6110                        afterUpload(attachment.codAttachment);
6111                        this.setState({
6112                            loading: false,
6113                            success: true
6114                        });
6115                    },
6116                    () => this.setState({ loading: false, success:
                         false })
6117                );
6118            }
6119        }
6120
6121        render(){
6122            const { attachment } = this.props;
6123            const { loading, success } = this.state;
6124            return (
6125                <Row>
6126                    <Col md="9">
6127                        <FileInput color={success ? 'success' :
                             'info'} label={attachment.name}
                             onChange={this.upload}
                             accept="application/pdf"
                             disabled={loading || success}/>
6128                    </Col>
6129                    <Col md="2">
6130                        <ClipLoader sizeUnit="px" size={30}
                             color="#007bff" loading={loading} />
6131                    </Col>
```

```
6132            </Row>
6133        );
6134    }
6135 }
6136
6137 export class AttachmentDownloader extends Component{
6138
6139     handlerDownload(){
6140         const { codAttachment, name } = this.props.attachment;
6141         axios(`/attachment/${codAttachment}/download`, {
6142             method: 'GET',
6143             responseType: 'blob' //Force to receive data in a
                    Blob Format
6144         }).then(res =>{
6145             const file = new Blob(
6146                 [res.data],
6147                 {type: 'application/pdf'}
6148             );
6149             saveAs(file, `${name}.pdf`);
6150         });
6151     }
6152
6153     render(){
6154         const { attachment } = this.props;
6155         return (
6156             <Row>
6157                 <Col md="9">
6158                     <Button
                            onClick={this.handlerDownload.bind(this)}>{attachmen
6159                 </Col>
6160             </Row>
6161         );
6162     }
6163 }
6164
6165 webapp/src/pages/protected/dashboard/volunteer/statement/index.js
6166
6167 import Step from './Step.jsx';
6168 export default Step;
6169
6170 webapp/src/pages/protected/dashboard/volunteer/statement/Step.jsx
6171
6172 import React, { Component } from 'react';
6173 import { Row, Col } from 'reactstrap';
```

```
6174 import { AttachmentUploader, AttachmentDownloader } from
         './Attachment.jsx';
6175 import { getRequest } from '../../../../../utils/http';
6176 import { Status } from '../../../../../model';
6177
6178 export default class extends Component{
6179     constructor() {
6180         super();
6181         this.state = {
6182             attachments: []
6183         }
6184     }
6185
6186     componentWillMount() {
6187         getRequest(`/attachment/${Status.WAIT_STATEMENT}`, res
                 => this.setState({ attachments: res.data }));
6188     }
6189
6190     render(){
6191         const { attachments } = this.state;
6192         const { cpf, afterUpload } = this.props;
6193         return (
6194             <Row style={{textAlign: 'center'}}>
6195                 <Col>
6196                     <Row>
6197                         <Col>
6198                             <a className="btn btn-info"
                                     target="_blank" rel="noopener
                                     noreferrer"
                                     href="http://www.computacaonaescola.ufsc.br/wp-cont
6199                                 Politica Protecao Criana
6200                             </a>
6201                         </Col>
6202                     </Row>
6203                     <br/>
6204                     <Row>
6205                         {
6206                             attachments.map(
6207                                 attachment => (
6208                                     <Col
                                         key={attachment.codAttachment}>
6209                                     <Row>
6210                                         <Col>
6211                                             <AttachmentUploader
                                                 cpf={cpf}
```

```
                                                    attachment={attachment}
                                                    afterUpload={afterUpload}
                                                    />
6212                                            </Col>
6213                                        </Row>
6214                                        <Row>
6215                                            <Col>
6216                                                {attachment.download
                                                    &&
                                                    <AttachmentDownloader
                                                    attachment={attachment}/>}
6217                                            </Col>
6218                                        </Row>
6219                                    </Col>
6220                                )
6221                            )
6222                        }
6223                    </Row>
6224                </Col>
6225            </Row>
6226        )
6227    }
6228 }
6229
6230 webapp/src/pages/protected/dashboard/volunteer/register/Attachment.jsx
6231
6232 import React, { Component } from 'react';
6233 import { Row, Col } from 'reactstrap';
6234 import { ClipLoader } from 'react-spinners';
6235 import { FileInput } from '../../../../../components';
6236 import { postRequest, getRequest } from
        '../../../../../utils/http';
6237
6238 export default class extends Component{
6239     constructor(){
6240         super();
6241         this.state = {
6242             loading: false,
6243             success: false
6244         }
6245         this.upload = this.upload.bind(this);
6246     }
6247
6248     componentWillMount(){
6249         const { cpf, attachment } = this.props;
```

```
6250            const { codAttachment } = attachment;
6251            getRequest(‘/volunteer/${cpf}/attachment/${codAttachment}‘,
                    res => this.setState({success: res.data.send}))
6252        }
6253
6254        upload(event){
6255            const file = event.target.files[0];
6256            const { attachment, cpf } = this.props;
6257            if (file){
6258                this.setState({loading: true});
6259                const json = JSON.stringify(attachment);
6260                const blob = new Blob([json], {
6261                    type: ’application/json’
6262                });
6263                let form = new FormData();
6264                form.append(’attachment’, blob);
6265                form.append(’file’, file);
6266                postRequest(
6267                    ‘/volunteer/${cpf}/upload‘,
6268                    form,
6269                    res => {
6270                        this.props.afterUpload(attachment.codAttachment);
6271                        this.setState({
6272                            loading: false,
6273                            success: true
6274                        });
6275                    },
6276                    () => this.setState({ loading: false, success:
                        false })
6277                );
6278            }
6279        }
6280
6281        render(){
6282            const { attachment } = this.props;
6283            const { loading, success } = this.state;
6284            return (
6285                <Row>
6286                    <Col md="9">
6287                        <FileInput color={success ? ’success’ :
                                ’info’} label={attachment.name}
                                onChange={this.upload}
                                accept="application/pdf"
                                disabled={loading || success}/>
6288                    </Col>
```

```
                <Col md="2">
                    <ClipLoader sizeUnit="px" size={30}
                        color="#007bff" loading={loading} />
                </Col>
            </Row>
        );
    }
}
```

webapp/src/pages/protected/dashboard/volunteer/register/index.js

```
import Step from './Step.jsx';
export default Step;
```

webapp/src/pages/protected/dashboard/volunteer/register/Step.jsx

```
import React, { Component } from 'react';
import { Row, Col } from 'reactstrap';
import AttachmentUploader from './Attachment.jsx';
import { getRequest } from '../../../../../utils/http';
import { Status } from '../../../../../model';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            attachments: []
        }
    }

    componentWillMount(){
        getRequest(`/attachment/${Status.REGISTER}`, res =>
            this.setState({ attachments: res.data}));
    }

    render(){
        const { attachments } = this.state;
        const { cpf, afterUpload } = this.props;
        if(cpf){
            return (
                <Row>
                    {
                        attachments.map(
                            attachment => (
```

```
                                <Col key= {
                                    attachment.codAttachment }>
                                    <AttachmentUploader cpf={cpf}
                                        attachment={attachment}
                                        afterUpload={afterUpload}/>
                                </Col>
                            )
                        )
                    }
                </Row>
            );
        }
        return (null);
    }
}
```

webapp/src/pages/protected/dashboard/volunteer/steps/index.js

```
import Steps from './RegisterSteps';
export default Steps;
```

webapp/src/pages/protected/dashboard/volunteer/steps/steps.css

```
.btn-circle {
  width: 90px;
  height: 90px;
  padding: 6px 0px;
  border-radius: 100% !important;
  text-align: center;
  font-size: 2rem !important;
  line-height: 1.42857; }
```

webapp/src/pages/protected/dashboard/volunteer/steps/steps.sass

```
.btn-circle
    width: 90px
    height: 90px
    padding: 6px 0px
    border-radius: 100% !important
    text-align: center
    font-size: 2rem !important
    line-height: 1.42857
```

webapp/src/pages/protected/dashboard/volunteer/steps/RegisterSteps.jsx

```
import React from 'react';
import { Row, Col, Button } from 'reactstrap';
import { Status } from '../../../../../model';
import { FaUser, FaCheck, FaFileAlt, FaBuilding, FaChalkboard }
    from 'react-icons/fa';
import Arrow from 'react-arrow'
import './steps.css';

const Step = ({userStatus, status, icon, successContition,
    label}) => (
    <Col>
        <Row>
            <Col>
                <Button color={ !successContition ? "primary":
                    'success'} size="lg" className="btn-circle"
                    disabled={userStatus !== status}>
                    {icon}
                </Button>
            </Col>
        </Row>
        <Row>
            <Col>
                {successContition ? label.replace('Aguardando ',
                    '') : label}
            </Col>
        </Row>
    </Col>
)


const ArrowStep = ({successContition}) => {
    const color = successContition ? '#28a745' : '#6c757d';
    const style = {
        display: 'flex',
        alignItems: 'center',
        paddingBottom: '2%'
    };
    return (
        <Col md="1" style={style}>
            <Arrow direction="right" shaftWidth={30}
                shaftLength={80} headWidth={80} headLength={30}
                fill={color}/>
        </Col>
```

```
6411        )
6412    }
6413
6414    export default ({status}) => (
6415        <Row style={{ textAlign: 'center' }}>
6416            <Step userStatus={status} status={Status.REGISTER}
                    icon={<FaUser />} successContition={Status.REGISTER
                    !== status} label="Registro"/>
6417            <ArrowStep successContition={Status.REGISTER !==
                    status}/>
6418            <Step userStatus={status} status={Status.WAIT_COMPANY}
                    icon={<FaBuilding/>}
                    successContition={![Status.REGISTER,
                    Status.WAIT_COMPANY].includes(status)}
                    label="Aguardando Recomendao de empresa"/>
6419            <ArrowStep successContition={![Status.REGISTER,
                    Status.WAIT_COMPANY].includes(status)}/>
6420            <Step userStatus={status} status={Status.WAIT_STATEMENT}
                    icon={<FaFileAlt/>}
                    successContition={[Status.WAIT_TRAINING,
                    Status.APPROVED].includes(status)} label="Aguardando
                    Envio do Termo de Responsabilidade"/>
6421            <ArrowStep successContition={[Status.WAIT_TRAINING,
                    Status.APPROVED].includes(status)}/>
6422            <Step userStatus={status} status={Status.WAIT_TRAINING}
                    icon={<FaChalkboard/>}
                    successContition={Status.APPROVED === status}
                    label="Aguardando Treinamento"/>
6423            <ArrowStep successContition={Status.APPROVED ===
                    status}/>
6424            <Step userStatus={status} status={Status.APPROVED}
                    icon={<FaCheck/>} successContition={Status.APPROVED
                    === status} label="Aprovado"/>
6425        </Row>
6426    );
6427
6428    webapp/src/pages/protected/dashboard/volunteer/approved/Form.jsx
6429
6430    import React, { Component } from 'react';
6431    import Company from '../../../../../model/company';
6432    import PropTypes from 'prop-types';
6433    import { postRequest } from '../../../../../utils/http';
6434    import { Row, Col, Modal, ModalHeader, ModalBody } from
        'reactstrap';
```

```
6435 import { Wizard, Input, MaskInput, DatePicker } from
         '../../../../../components';
6436
6437 export default class extends Component{
6438     constructor(){
6439         super();
6440         this.state = {
6441             evaluate: null
6442         };
6443         this.handlerSubmit = this.handlerSubmit.bind(this);
6444     }
6445
6446     static propTypes = {
6447         afterSubmit: PropTypes.func.isRequired,
6448         isOpen: PropTypes.bool.isRequired,
6449         close: PropTypes.func.isRequired
6450     }
6451
6452     handlerSubmit(){
6453         const { afterSubmit, close } = this.props;
6454         const { evaluate } = this.state;
6455         postRequest(
6456             '/company',
6457             evaluate,
6458             () => {
6459                 afterSubmit();
6460                 close();
6461             }
6462         );
6463     }
6464
6465     render(){
6466         const { close, isOpen } = this.props;
6467         const { evaluate } = this.state;
6468         return (
6469             <Modal isOpen={isOpen} toggle={close}>
6470                 <ModalHeader toggle={close}>Avaliao</ModalHeader>
6471                 <ModalBody>
6472                     <Wizard onCancel={close}
                                submitLabel="cadastrar"
                                onSubmit={this.handlerSubmit}>
6473                         <div>
6474                             <h3></h3>
6475                             <hr className="row"/>
6476                         </div>
```

```
                    </Wizard>
                </ModalBody>
            </Modal>
        );
    }
}

webapp/src/pages/protected/dashboard/volunteer/approved/index.js

import Step from './Step.jsx';
export default Step;

webapp/src/pages/protected/dashboard/volunteer/approved/Details.jsx

import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { getRequest } from '../../../../../utils/http';
import { Address } from '../../../../../model';
import { Button, Modal, ModalHeader, ModalBody, ModalFooter,
    Row, Col } from 'reactstrap';


export default class extends Component{
    constructor(){
        super();
        this.state = {
            eventDetails : null
        }
    }

    static propTypes = {
        code: PropTypes.number.isRequired,
        isOpen: PropTypes.bool.isRequired,
        close: PropTypes.func.isRequired
    };

    componentWillMount() {
        const { code } = this.props;
        getRequest(
            `/event/${code}`,
            res => {
                let eventDetails = res.data;
                eventDetails.title = eventDetails.type.name + ' -
                    ' + new
                    Date(eventDetails.startOccurrence).toLocaleString();
```

```
6519                this.setState({ eventDetails });
6520            }
6521        );
6522    }
6523
6524    render(){
6525        const { isOpen, close } = this.props;
6526        const { eventDetails } = this.state;
6527        if (eventDetails){
6528            const { address: eventAddress, trainings } =
                     eventDetails;
6529            const duration = Math.ceil(
6530                Math.abs(
6531                    eventDetails.startOccurrence -
                             eventDetails.endOccurrence
6532                ) / (1000 * 3600)
6533            );
6534            const address = new Address();
6535            Object.assign(address, eventAddress);
6536            return (
6537                <Modal toggle={close} isOpen={isOpen} >
6538                    <ModalHeader
                             toggle={close}>{eventDetails.title}</ModalHeader>
6539                    <ModalBody>
6540                        <Row>
6541                            <Col>
6542                                <strong> Durao :</strong>
                                        {eventDetails.allDay ? 'Dia
                                        Inteiro' : duration + ' horas
                                        (aproximadamente)'}
6543                            </Col>
6544                        </Row>
6545                        <br />
6546                        <Row>
6547                            <Col>
6548                                <strong>Endereo:</strong>
6549                            </Col>
6550                        </Row>
6551                        <Row>
6552                            <Col>
6553                                {address.toString()}
6554                            </Col>
6555                        </Row>
6556                        <br />
6557                        <Row>
```

```
6558                            <Col>
6559                                <strong>Materiais em anexo:</strong>
6560                            </Col>
6561                        </Row>
6562                        <Row>
6563                            <Col>
6564                                {
6565                                    trainings.length ?
6566                                        (
6567                                            <Row>
6568                                                {
6569                                                    trainings.map(
6570                                                        training => (
6571                                                            <Col
                                                                 key={training.codTrain
6572                                                                {training.link
                                                                     &&
                                                                     <a
                                                                     className="btn
                                                                     btn-info"
                                                                     href={training.linl
                                                                     target="_blank">{t
6573                                                            </Col>
6574                                                        )
6575                                                    )
6576                                                }
6577                                            </Row>
6578                                        ) : 'Nenhum anexo para esse
                                             evento.'
6579                                }
6580                            </Col>
6581                        </Row>
6582                    </ModalBody>
6583                    <ModalFooter style={{display: 'inline'}}>
6584                        <Row>
6585                            <Col>
6586                                <Button color="primary"
                                     className="float-right"
                                     onClick={close}>Fechar</Button>
6587                            </Col>
6588                        </Row>
6589                    </ModalFooter>
6590                </Modal>
6591            )
6592        } else {
```

```
            return (
                <Modal toggle={close} isOpen={isOpen} ></Modal>
            );
        }
    }
}

webapp/src/pages/protected/dashboard/volunteer/approved/Evaluate.jsx

import React, { Component, Fragment } from 'react';
import { Row, Col, Button } from 'reactstrap';
import Form from './Form';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            isOpen : false
        };
        this.toggle = this.toggle.bind(this);
    }

    toggle(){
        const { isOpen } = this.state;
        this.setState({ isOpen: !isOpen });
    }

    render(){
        const { evaluate, cpf, afterSubmit } = this.props;
        const { isOpen } = this.state
        return (
            <Fragment>
                <Row>
                    <Col>
                        <Button color="primary" style={{ width:
                            'inherit', height: '40px' }}
                            onClick={this.toggle}>
                            {evaluate.title}
                        </Button>
                    </Col>
                </Row>
                {isOpen && <Form cpf={cpf}
                    afterSubmit={afterSubmit} isOpen={isOpen}
                    close={this.toggle}/>}
                <br />
```

```
            </Fragment>
        )
    }
}

webapp/src/pages/protected/dashboard/volunteer/approved/Step.jsx

import React, { Component,Fragment } from 'react';
import { Calendar } from '../../../../../components';
import { Row, Col, Label } from 'reactstrap';
import BigCalendar from 'react-big-calendar';
import Details from './Details'
import Evaluate from './Evaluate'
import { Volunteer } from '../../../../../model';
import { getRequest } from '../../../../../utils/http';
import axios from 'axios';

const mapEvents =
    events => events.map(
        event =>{
            event.end = new Date(event.endOccurrence);
            event.start = new Date(event.startOccurrence);
            event.title = event.type.name + ' - ' +
                event.start.toLocaleString()
            return event
        }
    );

export default class extends Component{
    constructor(){
        super();
        this.state = {
            events: [],
            evaluates: [],
            isOpenDetails: false,
            codEvent: null,
            volunteer: new Volunteer()
        };
        this.closeDetails = this.closeDetails.bind(this);
        this.handlerSelectEvent =
            this.handlerSelectEvent.bind(this);
        this.componentWillMount =
            this.componentWillMount.bind(this);
    }

```

```
6676        componentWillMount(){
6677            const { cpf } = this.props;
6678            axios.all([
6679                getRequest(‘/volunteer/${cpf}‘, res => res.data),
6680                getRequest(
6681                    ‘/volunteer/${cpf}/event‘,
6682                    res => res.data
6683                )
6684            ]).then(
6685                res => {
6686                    const events = mapEvents(res[1]);
6687                    this.setState({
6688                        volunteer: res[0],
6689                        isOpenDetails: false,
6690                        codEvent: null,
6691                        evaluates: events.filter(event =>
                                event.evaluate),
6692                        events
6693                    });
6694                }
6695            );
6696        }
6697
6698        getLabel(){
6699            const { ratings, rating } = this.state.volunteer;
6700            const lastRating = ratings[0];
6701            let className, text;
6702            if (lastRating){
6703                if (rating < 40 || lastRating.grade < 40){
6704                    className = ’danger’;
6705                } else if (rating < 70){
6706                    className = ’warning’;
6707                } else {
6708                    className = ’success’;
6709                }
6710            } else {
6711                className = ’secondary’;
6712                text = ’Ainda no  h   avaliaes ’;
6713            }
6714
6715            return (
6716                <Row>
6717                    <Col>
6718                        <Label className={‘btn btn-${className}‘}
                                style={{ cursor: ’default’, width:
```

```
                    'inherit', height: '40px'}}>
6719                    {text}
6720                </Label>
6721            </Col>
6722        </Row>
6723    );
6724  }
6725
6726  closeDetails(){
6727      this.setState({ isOpenDetails: false, codEvent: null });
6728  }
6729
6730  handlerSelectEvent(event){
6731      const { codEvent } = event
6732      this.setState({ isOpenDetails: true, codEvent });
6733  }
6734
6735  render(){
6736      const { events, isOpenDetails, codEvent, evaluates,
            volunteer } = this.state;
6737      const { cpf } = volunteer.user;
6738      return (
6739          <Row style={{ height: '600px', marginBottom: '1%' }}>
6740              <Col>
6741                  <Calendar events={events}
                        onSelectEvent={this.handlerSelectEvent}
                        defaultView={BigCalendar.Views.WEEK}/>
6742                  {
6743                      isOpenDetails ?
6744                          <Details code={codEvent}
                                isOpen={isOpenDetails}
                                close={this.closeDetails}
                                afterDelete={this.componentWillMount}/>
                                :
6745                          null
6746                  }
6747              </Col>
6748              <Col md="2">
6749                  <h3 className="text-center">Avaliaes</h3>
6750                  {this.getLabel()}
6751                  <br/>
6752                  {evaluates.length ? evaluates.map(evaluate =>
                        <Evaluate evaluate={evaluate} cpf={cpf}
                        afterSubmit={()=> null}/>) : null}
6753              </Col>
```

```
            </Row>
        );
    }
}

webapp/src/pages/protected/dashboard/volunteer/company/index.js

import Step from './Step.jsx';
export default Step;

webapp/src/pages/protected/dashboard/volunteer/company/Step.jsx

import React, { Component } from 'react';
import { Row, Col } from 'reactstrap';
import { getRequest } from '../../../../../utils/http';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            company: null
        }
    }

    componentWillMount(){
        const { cnpj } = this.props;
        getRequest(`/company/${cnpj}`, res => this.setState({
            company: res.data }));
    }

    render(){
        const { company } = this.state;
        if (company){
            const { responsible } = company;
            return (
                <Row style={{textAlign: 'center'}}>
                    <Col>
                        <h2>
                            Aguarde resposta de sua empresa, para
                                mais informaes entre em contato com
                            <br/>
                            <strong>{responsible.name}</strong>
                        </h2>
                    </Col>
                </Row>
```

```
            );
        } else {
            return null;
        }
    }
}

webapp/src/pages/protected/dashboard/volunteer/Page.jsx

import React, { Component } from 'react';
import { Row, Col } from 'reactstrap';
import RegisterSteps from './steps';
import Register from './register';
import Company from './company';
import Statement from './statement';
import Training from './training';
import Approved from './approved';
import { Volunteer, Status } from '../../../../model';
import { getRequest, putRequest } from '../../../../utils/http';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            volunteer: new Volunteer(),
            attachments: [],
            dispatches: []
        };
        this.completeUpload = this.completeUpload.bind(this);
    }

    componentWillMount(){
        const { cpf } = this.props;
        getRequest(`/volunteer/${cpf}`, resVolunteer => {
            const volunteer = resVolunteer.data;
            const { user, dispatches } = volunteer;
            getRequest(
                `/attachment/${user.status}`,
                res => {
                    const attachments = res.data;
                    dispatches.push(
                        ...(
                            attachments.map(
                                attachment => {
                                    return {
```

```
6842                                    codAttachment:
                                            attachment.codAttachment,
6843                                    required:
                                            attachment.required,
6844                                    send: false
6845                                }
6846                            }
6847                        )
6848                    )
6849                )
6850                this.setState({
6851                    volunteer,
6852                    attachments,
6853                    dispatches
6854                });
6855            }
6856        );
6857    });
6858  }
6859
6860  completeUpload(codAttachment){
6861      let conclude = true
6862      const { dispatches, volunteer } = this.state;
6863      for (const dispatch of dispatches) {
6864          if (dispatch.codAttachment === codAttachment){
6865              dispatch.send = true
6866          }
6867      }
6868      for (const dispatch of dispatches) {
6869          conclude = dispatch.send || !dispatch.required
6870          if(!conclude){
6871              break;
6872          }
6873      };
6874      if (conclude){
6875          const confirm = window.confirm('Todos os anexo
                obrigatrios j foram submetidos, deseja finalizar
                esta etapa?');
6876          if (confirm){
6877              const { status } = volunteer.user;
6878              volunteer.user.status = Status.next(status);
6879              putRequest(
6880                  `/volunteer/${volunteer.user.cpf}`,
6881                  volunteer,
6882                  res => this.setState({volunteer, dispatches})
```

```
                );
            }
        } else {
            this.setState({ dispatches });
        }
    }

    render(){
        const { volunteer } = this.state;
        const { user, company } = volunteer;
        return (
            <Row>
                <Col md="1"/>
                <Col>
                    <Row>
                        <Col>
                            <RegisterSteps status={user.status}/>
                            <br/>
                            {user.status === Status.REGISTER &&
                                <Register cpf={user.cpf}
                                afterUpload={this.completeUpload}/>}
                            {user.status === Status.WAIT_COMPANY
                                && <Company cnpj={company.cnpj}/>}
                            {user.status === Status.WAIT_STATEMENT
                                && <Statement cpf={user.cpf}
                                afterUpload={this.completeUpload}/>}
                            {user.status === Status.WAIT_TRAINING
                                && <Training cpf={user.cpf} />}
                            {user.status === Status.APPROVED &&
                                <Approved cpf={user.cpf} />}
                        </Col>
                    </Row>
                </Col>
                <Col md="1"/>
            </Row>
        );
    }
}

webapp/src/pages/protected/dashboard/volunteer/index.js

import Page from './Page.jsx';
export default Page;

webapp/src/pages/protected/dashboard/school/volunteer/index.js
```

```
6921
6922 import List from './List.jsx';
6923 export default List;
6924
6925 webapp/src/pages/protected/dashboard/school/volunteer/ListItem.jsx
6926
6927 import React, { Component } from 'react';
6928 import { Row, Col, ListGroupItem } from 'reactstrap';
6929 import Details from './Details.jsx';
6930 import PropTypes from 'prop-types';
6931 import formatter from '../../../../../utils/formatter';
6932
6933 export default class extends Component{
6934     constructor(){
6935         super();
6936         this.state = {
6937             isOpen: false
6938         }
6939         this.toggle = this.toggle.bind(this);
6940     }
6941
6942     static propTypes = {
6943         user: PropTypes.object.isRequired
6944     }
6945
6946     toggle(){
6947         const { isOpen } = this.state;
6948         this.setState({isOpen: !isOpen});
6949     }
6950
6951     render(){
6952         const { user } = this.props;
6953         const { isOpen } = this.state;
6954         return (
6955             <ListGroupItem>
6956                 <Row>
6957                     <Col onClick={this.toggle}
                             className="text-ellipsis" style={{
                             cursor: 'pointer' }}>
6958                         {formatter.cpf(user.cpf)} - {user.name}
6959                     </Col>
6960                 </Row>
6961                 { isOpen && <Details cpf={user.cpf}
                         isOpen={isOpen} close={this.toggle} /> }
6962             </ListGroupItem>
```

```
6963        );
6964    }
6965 }
6966
6967 webapp/src/pages/protected/dashboard/school/volunteer/Details.jsx
6968
6969 import React, { Component } from 'react';
6970 import { getRequest } from '../../../../../utils/http';
6971 import formatter from '../../../../../utils/formatter';
6972 import { Status, Address, Schooling } from
        '../../../../../model';
6973 import { Button, Modal, ModalHeader, ModalBody, ModalFooter,
        Row, Col, Label } from 'reactstrap';
6974 import pic from './pic.jpeg';
6975
6976 export default class extends Component{
6977     constructor(){
6978         super();
6979         this.state = {
6980             volunteer: null,
6981         };
6982     }
6983
6984     componentWillMount(){
6985         const { cpf } = this.props;
6986         getRequest(`/volunteer/${cpf}`, res =>
              this.setState({volunteer: res.data}));
6987     }
6988
6989     render(){
6990         const { volunteer } = this.state;
6991         const { close, isOpen } = this.props;
6992         if(volunteer){
6993             const { user } = volunteer;
6994             const address = new Address();
6995             Object.assign(address, user.address);
6996             return (
6997                 <Modal toggle={close} isOpen={isOpen}>
6998                     <ModalHeader toggle={close}>{user.name} -
                          <small>{formatter.cpf(user.cpf)}</small></ModalHeader>
6999                     <ModalBody>
7000                         <Row>
7001                             <Col md="1">
7002                                 <img src={pic} alt=""
                                        className="rounded-circle"
```

```
                        width="75px" />
7003                </Col>
7004                <Col md="2" />
7005                <Col>
7006                    <Row>
7007                        <Col>
7008                            <Label className="btn
                                    btn-info"
                                    style={{cursor:
                                    'default'}}>
7009                                {Status.translate(user.status)}
7010                            </Label>
7011                        </Col>
7012                    </Row>
7013                    <Row>
7014                        <Col>
7015                            <strong>Esoclaridade:</strong>
7016                        </Col>
7017                    </Row>
7018                    <Row>
7019                        <Col>
7020                            {
                                    Schooling.translate(volunteer.scho
                                    } -
                                    {volunteer.conclusion ?
                                    'Completo':
                                    'Incompleto'}
7021                            {
7022                                volunteer.course && <div>
7023                                    { volunteer.course }
7024                                </div>
7025                            }
7026                        </Col>
7027                    </Row>
7028                    <br />
7029                    <Row>
7030                        <Col>
7031                            <strong>Contato:</strong>
7032                        </Col>
7033                    </Row>
7034                    <Row>
7035                        <Col>
7036                            {user.email} /
                                    {formatter.phone(user.phone)}
7037                        </Col>
```

```
                                </Row>
                                <br/>
                                <Row>
                                    <Col>
                                        <strong>Endereo:</strong>
                                    </Col>
                                </Row>
                                <Row>
                                    <Col>
                                        {address.toString()}
                                    </Col>
                                </Row>
                                <br />
                                <Row>
                                    <Col>
                                        <strong> Avaliaes :</strong>
                                    </Col>
                                </Row>
                                <Row>
                                    <Col>
                                        {
                                            volunteer.ratings.length?
                                                ` Avaliao   mdia  de
                                                    ${user.name}
                                                    ${volunteer.rating}`:
                                                    'Voluntrio ainda
                                                    no foi avaliado'
                                        }
                                    </Col>
                                </Row>
                                <br/>
                            </Col>
                        </Row>
                    </ModalBody>
                    <ModalFooter>
                        <Button color="default"
                            className="float-right"
                            onClick={close}>Fechar</Button>
                    </ModalFooter>
                </Modal>
            )
        }
        return (
            <Modal toggle={close} isOpen={isOpen}/>
        );
```

```
    }
}

webapp/src/pages/protected/dashboard/school/volunteer/List.jsx

import React, { Component } from 'react';
import { ListGroup, ListGroupItem } from 'reactstrap';
import VolunteerItem from './ListItem.jsx';
import { getRequest } from '../../../../../utils/http';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            users: []
        };
    }

    componentWillMount(){
        const {codSchool } = this.props;
        getRequest(`/volunteer?cod_school=${codSchool}`, res =>
            this.setState({ users: res.data.map(volunteer =>
            volunteer.user)}))
    }

    render(){
        const { users } = this.state;
        return (
            <ListGroup>
                {
                    users.length ?
                        users.map(
                            user => <VolunteerItem key={user.cpf}
                                user={user} />
                        ) :
                        <ListGroupItem>
                            <strong>
                                Nenhum Voluntrio Cadastrado
                            </strong>
                        </ListGroupItem>
                }
            </ListGroup>
        )
    }
}
```

webapp/src/pages/protected/dashboard/school/event/index.js

```
import List from './List.jsx';
export default List;
```

webapp/src/pages/protected/dashboard/school/event/Details.jsx

```
import React, { Component } from 'react';
import PropTypes from 'prop-types';
import { getRequest } from '../../../../../utils/http';
import { Address } from '../../../../../model';
import { Button, Modal, ModalHeader, ModalBody, ModalFooter,
    Row, Col } from 'reactstrap';


export default class extends Component{
    constructor(){
        super();
        this.state = {
            eventDetails : null
        }
    }

    static propTypes = {
        code: PropTypes.number.isRequired,
        isOpen: PropTypes.bool.isRequired,
        close: PropTypes.func.isRequired,
    };

    componentWillMount() {
        const { code } = this.props;
        getRequest(
            `/event/${code}`,
            res => {
                let eventDetails = res.data;
                eventDetails.title = eventDetails.type.name + ' -
                    ' + new
                    Date(eventDetails.startOccurrence).toLocaleString();
                this.setState({ eventDetails });
            }
        );
    }

    render(){
```

```
7161            const { isOpen, close } = this.props;
7162            const { eventDetails } = this.state;
7163            if (eventDetails){
7164                const { address: eventAddress, trainings } =
                        eventDetails;
7165                const duration = Math.ceil(
7166                    Math.abs(
7167                        eventDetails.startOccurrence -
                                eventDetails.endOccurrence
7168                    ) / (1000 * 3600)
7169                );
7170                const address = new Address();
7171                Object.assign(address, eventAddress);
7172                return (
7173                    <Modal toggle={close} isOpen={isOpen} >
7174                        <ModalHeader
                                toggle={close}>{eventDetails.title}</ModalHeader>
7175                        <ModalBody>
7176                            <Row>
7177                                <Col>
7178                                    <strong> Durao :</strong>
                                            {eventDetails.allDay ? 'Dia
                                            Inteiro' : duration + ' horas
                                            (aproximadamente)'}
7179                                </Col>
7180                            </Row>
7181                            <br />
7182                            <Row>
7183                                <Col>
7184                                    <strong>Endereo:</strong>
7185                                </Col>
7186                            </Row>
7187                            <Row>
7188                                <Col>
7189                                    {address.toString()}
7190                                </Col>
7191                            </Row>
7192                            <br />
7193                            <Row>
7194                                <Col>
7195                                    <strong>Materiais em anexo:</strong>
7196                                </Col>
7197                            </Row>
7198                            <Row>
7199                                <Col>
```

```
                                {
                                    trainings.length ?
                                        (
                                            <Row>
                                                {
                                                    trainings.map(
                                                        training => (
                                                            <Col
                                                                key={training.codTrain
                                                                {training.link
                                                                    &&
                                                                    <a
                                                                    className="btn
                                                                    btn-info"
                                                                    href={training.linl
                                                                    target="_blank">{t
                                                            </Col>
                                                        )
                                                    )
                                                }
                                            </Row>
                                        ) : 'Nenhum anexo para esse
                                            evento.'
                                }
                            </Col>
                        </Row>
                    </ModalBody>
                    <ModalFooter>
                        <Button color="primary"
                            className="float-right"
                            onClick={close}>Fechar</Button>
                    </ModalFooter>
                </Modal>
            )
        } else {
            return (
                <Modal toggle={close} isOpen={isOpen} ></Modal>
            );
        }
    }
}

webapp/src/pages/protected/dashboard/school/event/List.jsx

import React, { Component } from 'react';
```

```
import { Calendar } from '../../../../../components';
import BigCalendar from 'react-big-calendar';
import Details from './Details.jsx'
import { getRequest } from '../../../../../utils/http';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            events: [],
            isOpenDetails: false,
            codEvent: null
        };
        this.closeDetails = this.closeDetails.bind(this);
        this.handlerSelectEvent =
            this.handlerSelectEvent.bind(this);
    }

    componentWillMount(){
        const { codSchool } = this.props;
        getRequest(
            `/school/${codSchool}/event`,
            res => this.setState({
                events: res.data.map(
                    event => {
                        event.end = new Date(event.endOccurrence);
                        event.start = new
                            Date(event.startOccurrence);
                        event.title = event.type.name + ' - ' +
                            event.start.toLocaleString()
                        return event
                    }
                )
            })
        );
    }

    closeDetails(){
        this.setState({ isOpenDetails: false, codEvent: null });
    }

    handlerSelectEvent(event){
        const { codEvent } = event
        this.setState({ isOpenDetails: true, codEvent });
    }
```

```
7277
7278        render(){
7279            const { events, isOpenDetails, codEvent } = this.state;
7280            return (
7281                <div style={{ height: '600px', marginBottom: '1%' }}>
7282                       <Calendar events={events}
                               onSelectEvent={this.handlerSelectEvent}
                               defaultView={BigCalendar.Views.WEEK}/>
7283                       { isOpenDetails && <Details code={codEvent}
                               isOpen={isOpenDetails}
                               close={this.closeDetails} />}
7284                </div>
7285            );
7286        }
7287    }
7288
7289    webapp/src/pages/protected/dashboard/school/Page.jsx
7290
7291    import React, { Component } from 'react';
7292    import { Row, Col } from 'reactstrap';
7293    import VolunteerList from './volunteer';
7294    import EventList from './event';
7295    import { getRequest } from '../../../../utils/http';
7296
7297    export default class extends Component{
7298        constructor(){
7299            super();
7300            this.state = {
7301                school: null,
7302                volunteers: [],
7303                events: []
7304            };
7305        }
7306        componentWillMount(){
7307            const { cpf } = this.props;
7308            getRequest(`/school?cod_cpf=${cpf}`, res =>
                    this.setState({ school: res.data[0]}));
7309        }
7310
7311        render() {
7312            const { school } = this.state;
7313            if (school){
7314                return(
7315                    <Row>
7316                        <Col md="1"/>
```

```
                    <Col>
                        <EventList codSchool={school.codSchool}/>
                    </Col>
                    <Col md="3">
                        <VolunteerList
                            codSchool={school.codSchool}/>
                    </Col>
                    <Col md="1"/>
                </Row>
            );
        } else {
            return null;
        }
    }
}
```

webapp/src/pages/protected/dashboard/school/index.js

```
import Page from './Page.jsx';
export default Page;
```

webapp/src/pages/protected/dashboard/company/Line.jsx

```
import React, { Component } from 'react';
import { Button, Label } from 'reactstrap';
import { FaTrash } from 'react-icons/fa';
import Details from './Details';
import Recommend from './Recommend'
import Paticipation from './Paricipations'
import formatter from '../../../../utils/formatter';
import { deleteRequest } from '../../../../utils/http';
import { Status } from '../../../../model';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            openDetails: false,
            openForm: false
        };
        this.handlerDelete = this.handlerDelete.bind(this);
        this.toggleRecommend = this.toggleRecommend.bind(this);
        this.toggleDetails = this.toggleDetails.bind(this);
    }

```

```
7361        toggleRecommend(){
7362            const { openForm } = this.state;
7363            this.setState({ openForm: !openForm, openDetails: false
                    });
7364        }
7365
7366        toggleDetails() {
7367            const { openDetails } = this.state;
7368            this.setState({ openDetails: !openDetails, openForm:
                    false });
7369        }
7370
7371        handlerDelete(){
7372            const { volunteer, afterDelete } = this.props;
7373            const { user, company } = volunteer;
7374            deleteRequest(`/user/${user.cpf}`, () =>
                    afterDelete(company));
7375        }
7376
7377        getLabel() {
7378            const { ratings, rating } = this.props.volunteer;
7379            const lastRating = ratings[0];
7380            let className, text;
7381            if (lastRating) {
7382                if (rating < 40 || lastRating.grade < 40) {
7383                    className = 'danger';
7384                } else if (rating < 70) {
7385                    className = 'warning';
7386                } else {
7387                    className = 'success';
7388                }
7389            } else {
7390                className = 'secondary';
7391                text = 'Ainda no  h   avaliaes ';
7392            }
7393
7394            return (
7395                <Label className={`btn btn-${className}`} style={{
                        cursor: 'default', width: 'inherit', height:
                        '40px' }}>
7396                    {text}
7397                </Label>
7398            );
7399        }
7400
```

```
7401        render(){
7402            const { volunteer, afterDelete } = this.props;
7403            const { user } = volunteer;
7404            const { openDetails, openForm } = this.state;
7405            return (
7406                <tr>
7407                    <td style={ {width: '15%'}}>{
                            formatter.cpf(user.cpf) }</td>
7408                    <td style={{ maxWidth: '130px' }}>{ user.name
                            }</td>
7409                    <td style={{ maxWidth: '115px' }}>
7410                        <Label className={`btn btn-${user.status ===
                                Status.WAIT_COMPANY ? 'warning':
                                'info'}`} style={{ color: 'white',
                                cursor: 'default', whiteSpace: 'unset'}}>
7411                            { Status.translate(user.status) }
7412                        </Label>
7413                    </td>
7414                    <td style={{ width: '40px' }}>
7415                        <Button type="button" color="info"
                                onClick={this.toggleDetails}>
7416                            Detalhes
7417                        </Button>
7418                        {
7419                            openDetails && (
7420                                <Details close={this.toggleDetails}
                                        openForm={openForm} cpf={user.cpf}
                                        isOpen={true}
                                        closeRecommend={this.toggleRecommend}
                                        deleteFunc={this.handlerDelete}/>
7421                            )
7422                        }
7423                    </td>
7424                    <td>
7425                        <Paticipation cpf={user.cpf}/>
7426                    </td>
7427                    <td style={{ width: user.status ===
                            Status.WAIT_COMPANY ? '60px': '200px' }}>
7428                        {
7429                            user.status === Status.WAIT_COMPANY ? (
7430                                <div>
7431                                    <Button type="button" color="info"
                                            onClick={this.toggleRecommend}>
7432                                        Recomendar
7433                                    </Button>
```

```
                                {
                                    openForm && (
                                        <Recommend
                                            close={this.toggleRecommend}
                                            cpf={user.cpf}
                                            isOpen={true}
                                            afterSubmit={afterDelete}/>
                                    )
                                }
                            </div>
                        ) : this.getLabel()
                    }
                </td>
                <td style={{maxWidth: '15px'}}>
                    <FaTrash style={{ cursor: 'pointer' }}
                        color="red" onClick={this.handlerDelete}
                        />
                </td>
            </tr>
        );
    }
}

webapp/src/pages/protected/dashboard/company/Paricipations.jsx

import React, { Component, Fragment } from 'react';
import { Button, Popover, PopoverHeader, PopoverBody } from
    'reactstrap';
import { getRequest } from '../../../../utils/http';
import { Address } from '../../../../model';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            visible: false,
            participations: []
        };
        this.toggle = this.toggle.bind(this)
    }

    toggle(){
        const { visible } = this.state;
        const { cpf } = this.props;
        if(!visible){
```

```
7472            getRequest(
7473                `volunteer/${cpf}/participations`,
7474                res => {
7475                    let participations = res.data.map(
7476                        participation => {
7477                            const { school } = participation;
7478                            const address = new Address();
7479                            Object.assign(address, participation);
7480                            return {
7481                                address, school
7482                            }
7483                        }
7484                    );
7485                    this.setState({ visible: true, participations
                        });
7486                }
7487            )
7488        } else{
7489            this.setState({ visible: false })
7490        }
7491    }
7492
7493    render(){
7494        const { cpf } = this.props;
7495        const { visible, participations } = this.state;
7496        const { length } = participations;
7497        return (
7498            <Fragment>
7499                <Button id={`volunteer-${cpf}`} color="info"
                        onClick={this.toggle}>Locais</Button>
7500                <Popover placement="auto" isOpen={visible}
                        target={`volunteer-${cpf}`}
                        toggle={this.toggle}>
7501                    <PopoverHeader> Paticipaes </PopoverHeader>
7502                    <PopoverBody>
7503                        {
7504                            length ?
7505                                participations.map(
7506                                    (participation, index) => (
7507                                        <Fragment>
7508                                            {participation.school}
                                                ->
                                                {participation.address.toStrin
7509                                            {index !== length-1 &&
                                                <hr/>}
```

```
                                    </Fragment>
                                )
                            ) : <strong>Este voluntrio ainda no
                                participou de nenhum evento
                                pela CnE.</strong>
                        }
                    </PopoverBody>
                </Popover>
            </Fragment>
        );
    }
}
```

webapp/src/pages/protected/dashboard/company/Page.jsx

```
import React, { Component } from 'react';
import { Row, Col, Table } from 'reactstrap';
import { Filter } from '../../../../components';
import Line from './Line.jsx';
import { getRequest } from '../../../../utils/http';

export default class extends Component{
    constructor(){
        super();
        this.state = {
            filter: '',
            company: null,
            volunteers: []
        };
        this.listVolunteers = this.listVolunteers.bind(this);
        this.handlerFilter = this.handlerFilter.bind(this);
    }

    componentWillMount(){
        const { cpf } = this.props;
        getRequest(
            `company?cod_cpf=${cpf}`,
            companyRes => {
                const company = companyRes.data[0];
                if (company){
                    this.listVolunteers(company)
                }
            }
        )
    }
```

```
7553
7554        handlerFilter(event) {
7555            const { company } = this.state;
7556            const filter = event.target.value;
7557            if (filter.length >= 3 || filter.length === 0){
7558                this.listVolunteers(company, filter);
7559            } else {
7560                this.setState({filter});
7561            }
7562        }
7563
7564        listVolunteers(company, filter=''){
7565            getRequest(
7566                `/volunteer?cod_cnpj=${company.cnpj}&filter=${filter}`,
7567                res => this.setState({ company, filter, volunteers:
                        res.data })
7568            )
7569        }
7570
7571        render(){
7572            const { company, volunteers, filter } = this.state;
7573            if ( company ){
7574                return (
7575                    <Row>
7576                        <Col md="1"/>
7577                        <Col>
7578                            <Table striped>
7579                                <thead>
7580                                    <tr>
7581                                        <th>CPF</th>
7582                                        <th>
7583                                            <Filter label="Nome "
                                                    value={filter}
                                                    handlerFilter={this.handlerFilter}
7584                                        </th>
7585                                        <th>Status</th>
7586                                        <th>Detalhes</th>
7587                                        <th> Participaes </th>
7588                                        <th colSpan="2"></th>
7589                                    </tr>
7590                                </thead>
7591                                <tbody>
7592                                    {
7593                                        volunteers.length ?
```

```
                                    volunteers.map(volunteer =>
                                        <Line
                                        key={volunteer.user.cpf}
                                        volunteer={volunteer}
                                        afterDelete={this.listVolunteers}/>)
                                        :
                                (
                                    <tr>
                                        <td colSpan="3">
                                            <strong>
                                                Nenhum Voluntrio
                                                    Cadastrado
                                            </strong>
                                        </td>
                                    </tr>
                                )
                            }
                        </tbody>
                    </Table>
                </Col>
                <Col md="1"/>
            </Row>
        );
    }
    return (null);
  }
}

webapp/src/pages/protected/dashboard/company/index.js

import Page from './Page.jsx';
export default Page;

webapp/src/pages/protected/dashboard/company/Recommend.jsx

import React, { Component } from 'react';
import { Modal, ModalHeader, ModalBody, Row, Col } from
    'reactstrap';
import { Wizard, Switch } from '../../../../components';
import { getRequest, putRequest } from '../../../../utils/http';
import formatter from '../../../../utils/formatter';
import { Roles, Answer, Status } from '../../../../model';
import axios from 'axios';

export default class extends Component{
```

```
7632    constructor(){
7633        super();
7634        this.state = {
7635            volunteer : null,
7636            questions: [],
7637            answers: [],
7638            volunterAnswers: []
7639        };
7640        this.handlerAnswer = this.handlerAnswer.bind(this);
7641        this.handlerAnswerComment =
                this.handlerAnswerComment.bind(this);
7642        this.handlerSubmit = this.handlerSubmit.bind(this);
7643    }
7644
7645    componentWillMount(){
7646        const { cpf } = this.props;
7647        axios.all([
7648            getRequest(`/volunteer/${cpf}`, res => res.data),
7649            getRequest(`/question/${Roles.COMPANY_ADMIN}`, res =>
                    res.data)
7650        ]).then(res => {
7651
7652            this.setState({
7653                volunteer: res[0],
7654                questions: res[1],
7655                answers: res[1].map(question => new
                        Answer(question)),
7656                volunterAnswers: res[0].answers
7657            });
7658        });
7659    }
7660
7661    handlerAnswer(value, codQuestion) {
7662        const { answers } = this.state;
7663        for (const answer of answers) {
7664            if (answer.question.codQuestion === codQuestion) {
7665                answer.answer = value;
7666                break;
7667            }
7668        }
7669        this.setState({ answers });
7670    }
7671
7672    handlerAnswerComment(event, codQuestion) {
7673        const { answers } = this.state;
```

```
        for (const answer of answers) {
            if (answer.question.codQuestion === codQuestion) {
                answer.answer = event.target.value;
                break;
            }
        }
        this.setState({ answers });
    }

    handlerSubmit(){
        const { volunteer, answers, volunterAnswers } =
            this.state;
        const { user } = volunteer;
        const { afterSubmit } = this.props;
        user.status = Status.WAIT_STATEMENT;
        volunteer.user = user;
        volunteer.answers.push(...answers);
        putRequest(
            `/volunteer/${user.cpf}`,
            volunteer,
            afterSubmit,
            () =>{
                user.status = Status.WAIT_COMPANY;
                volunteer.answers = volunterAnswers;
                volunteer.user = user;
                this.setState({volunteer});
            }
        );
    }

    render(){
        const { volunteer, questions } = this.state;
        const { close, isOpen } = this.props;
        if(volunteer){
            const { user } = volunteer;
            return (
                <Modal toggle={close} isOpen={isOpen} >
                    <ModalHeader toggle={close}>
                        {user.name} -
                        <small>
                            {formatter.cpf(user.cpf)}
                        </small>
                    </ModalHeader>
                    <ModalBody>
```

```
7717                    <Wizard onCancel={close}
                            submitLabel="Recomendar"
                            onSubmit={this.handlerSubmit}>
7718                        <div>
7719                            {
7720                                volunteer.answers.map((answer,
                                        index) => (
7721                                    <div
                                            key={answer.question.codQuestio
7722                                        <Row >
7723                                            <Col>
7724                                                <strong>{answer.question
7725                                            </Col>
7726                                            <Col md="2">
7727                                                <Switch
                                                        id={`question-${answ
                                                        label="Resposta"
                                                        value={answer.answer
                                                        disabled
                                                        />
7728                                            </Col>
7729                                            <Col>
7730                                                <textarea
                                                        id={`details-${answe
                                                        cols="30"
                                                        rows="3"
                                                        value={answer.commen
                                                        disabled
                                                        />
7731                                            </Col>
7732                                        </Row>
7733                                        {index !==
                                                answer.length -
                                                1 ? <hr /> :
                                                null}
7734                                    </div>
7735                                ))
7736                            }
7737                            <hr/>
7738                        </div>
7739                        <div>
7740                            {
7741                                questions.map((question, index)
                                        => (
```

```
7742                                    <div
                                            key={question.codQuestion}>
7743                                        <Row >
7744                                            <Col>
7745                                                <strong>{question.question}</stron
7746                                            </Col>
7747                                            <Col md="2">
7748                                                <Switch
                                                        id={`question-${question.codQ
                                                        label="Resposta"
                                                        onChange={(value)
                                                        =>
                                                        this.handlerAnswer(value,
                                                        question.codQuestion)}
                                                        />
7749                                            </Col>
7750                                            <Col>
7751                                                <textarea
                                                        id={`details-${question.codQu
                                                        cols="30"
                                                        rows="3"
                                                        value={question.comment}
                                                        onChange={(event)
                                                        =>
                                                        this.handlerAnswerComment(eve
                                                        question.codQuestion)}
                                                        />
7752                                            </Col>
7753                                        </Row>
7754                                        {index !==
                                                question.length - 1
                                                ? <hr /> : null}
7755                                    </div>
7756                                ))
7757                            }
7758                            <hr/>
7759                        </div>
7760                    </Wizard>
7761                </ModalBody>
7762            </Modal>
7763        );
7764    }
7765    return (null);
7766  }
7767 }
```

```
7768
7769 webapp/src/pages/protected/dashboard/company/Details.jsx
7770
7771 import React, { Component } from 'react';
7772 import { Button, Modal, ModalHeader, ModalBody, ModalFooter,
         Row, Col } from 'reactstrap';
7773 import Recommend from './Recommend.jsx'
7774 import { getRequest } from '../../../../utils/http';
7775 import formatter from '../../../../utils/formatter';
7776 import { Address, Schooling, Status } from '../../../../model';
7777 import pic from './pic.jpeg';
7778
7779 export default class extends Component{
7780     constructor(){
7781         super();
7782         this.state = {
7783             volunteer: null
7784         }
7785     }
7786
7787     componentWillMount(){
7788         const { cpf } = this.props;
7789         getRequest(`/volunteer/${cpf}`, res =>
              this.setState({volunteer: res.data}));
7790     }
7791
7792     handlerDelete(){
7793         const { deleteFunc, close } = this.props;
7794         deleteFunc();
7795         close();
7796     }
7797
7798     render(){
7799         const { volunteer } = this.state;
7800         const { closeRecommend, openForm, close, isOpen } =
              this.props;
7801         if(volunteer){
7802             const { user } = volunteer;
7803             const address = new Address();
7804             Object.assign(address, user.address);
7805             return (
7806                 <Modal toggle={close} isOpen={isOpen} >
7807                     <ModalHeader toggle={close}>
7808                         {user.name} -
7809                         <small>
```

```
7810                        {formatter.cpf(user.cpf)}
7811                    </small>
7812                </ModalHeader>
7813                <ModalBody>
7814                    <Row>
7815                        <Col md="1">
7816                            <img src={pic} alt=""
                                    className="rounded-circle"
                                    width="75px" />
7817                        </Col>
7818                        <Col md="2" />
7819                        <Col>
7820                            <Row>
7821                                <Col>
7822                                    <strong>Esoclaridade:</strong>
7823                                </Col>
7824                            </Row>
7825                            <Row>
7826                                <Col>
7827                                    {
                                        Schooling.translate(volunteer.schooling)
                                        } -
                                        {volunteer.conclusion ?
                                        'Completo':
                                        'Incompleto'}
7828                                    {
7829                                        volunteer.course && <div>
7830                                            { volunteer.course }
7831                                        </div>
7832                                    }
7833                                </Col>
7834                            </Row>
7835                            <br />
7836                            <Row>
7837                                <Col>
7838                                    <strong>Contato:</strong>
7839                                </Col>
7840                            </Row>
7841                            <Row>
7842                                <Col>
7843                                    {user.email} /
                                        {formatter.phone(user.phone)}
7844                                </Col>
7845                            </Row>
7846                            <br/>
```

```
7847                            <Row>
7848                                <Col>
7849                                    <strong>Endereo:</strong>
7850                                </Col>
7851                            </Row>
7852                            <Row>
7853                                <Col>
7854                                    {address.toString()}
7855                                </Col>
7856                            </Row>
7857                            <br />
7858                            <Row>
7859                                <Col>
7860                                    <strong> Avaliaes :</strong>
7861                                </Col>
7862                            </Row>
7863                            <Row>
7864                                <Col>
7865                                    {
7866                                        volunteer.ratings.length?
7867                                            ‘ Avaliao   mdia  de
                                                  ${user.name}
                                                  ${volunteer.rating}‘:
                                                  ’Voluntrio ainda
                                                  no foi avaliado’
7868                                    }
7869                                </Col>
7870                            </Row>
7871                        </Col>
7872                    </Row>
7873                </ModalBody>
7874                <ModalFooter style={{ display: ’inline’ }}>
7875                    <Row>
7876                        <Col>
7877                            <Button color="danger"
                                    onClick={this.handlerDelete}>Excluir</But
7878                        </Col>
7879                        <Col>
7880                            {
7881                                user.status ===
                                    Status.WAIT_COMPANY? (
7882                                    <Button color="info"
                                        className="float-right"
                                        onClick={closeRecommend}>Recomenda
7883                                ) : (
```

```
                                    <Button color="default"
                                        className="float-right"
                                        onClick={close}>Fechar</Button>
                                )
                            }
                            {
                                openForm && (
                                    <Recommend
                                        close={closeRecommend}
                                        cpf={user.cpf}
                                        isOpen={true} />
                                )
                            }
                        </Col>
                    </Row>
                </ModalFooter>
            </Modal>
        );
    } else {
        return null
    }
  }
}

webapp/src/pages/protected/dashboard/index.js

import Dashboard from './Dashboard.jsx';
export default Dashboard;

webapp/src/pages/protected/dashboard/Dashboard.jsx

import React, { Fragment } from 'react';
import { Redirect } from 'react-router-dom';
import AdminDashboard from './admin';
import VolunteerDashboard from './volunteer';
import CompanyDashboard from './company';
import SchoolDashboard from './school';
import { Roles } from '../../../model';

export default ({cpf, role, logged}) => {
    return logged ? (
        <Fragment>
            {
                role === Roles.ADMIN && (
                    <AdminDashboard/>
```

```
                )
            }
            {
                role === Roles.VOLUNTEER && (
                    <VolunteerDashboard cpf={cpf}/>
                )
            }
            {
                role === Roles.COMPANY_ADMIN && (
                    <CompanyDashboard cpf={cpf}/>
                )
            }
            {
                role === Roles.SCHOOL_ADMIN && (
                    <SchoolDashboard cpf={cpf}/>
                )
            }
        </Fragment>
    ) : <Redirect to='/login'/>
}
```

webapp/src/pages/protected/index.js

```
export {default as Dashboard} from './dashboard';
export {default as Admin} from './admin';
```

webapp/src/pages/index.js

```
export {Dashboard, Admin} from './protected';
export {Login, Register} from './public';
```

webapp/src/App.css

```
a.no-undeline {
  color: inherit;
  text-decoration: None; }
  a.no-undeline:hover {
    color: inherit;
    text-decoration: None; }

textarea {
  resize: none !important; }

.text-ellipsis {
  white-space: nowrap;
```

```
  overflow: hidden;
  text-overflow: ellipsis; }


webapp/src/index.js

import React from 'react';
import ReactDOM from 'react-dom';
import registerServiceWorker from './registerServiceWorker';
import App from './App.jsx';
import 'bootstrap/dist/css/bootstrap.min.css';
import './index.css';
import './utils/http';

ReactDOM.render(<App />, document.getElementById('root'));
registerServiceWorker();


webapp/src/index.css

body {
  margin: 0;
  padding: 0;
  font-family: sans-serif; }


webapp/src/App.sass

a
    &.no-undeline
        color: inherit
        text-decoration: None
        &:hover
            color: inherit
            text-decoration: None

textarea
    resize: none !important

.text-ellipsis
    white-space: nowrap
    overflow: hidden
    text-overflow: ellipsis

webapp/src/index.sass
```

```
8014
8015  body
8016    margin: 0
8017    padding: 0
8018    font-family: sans-serif
8019
8020
8021
8022  webapp/src/App.test.js
8023
8024  import React from 'react';
8025  import ReactDOM from 'react-dom';
8026  import App from './App';
8027
8028  it('renders without crashing', () => {
8029    const div = document.createElement('div');
8030    ReactDOM.render(<App />, div);
8031    ReactDOM.unmountComponentAtNode(div);
8032  });
8033
8034
8035  webapp/src/App.jsx
8036
8037  // @flow
8038  import React, { Component } from 'react';
8039  import { Header, Footer } from './components';
8040  import { BrowserRouter, Route, Redirect } from
         'react-router-dom';
8041  import { Container, Row, Col } from 'reactstrap';
8042  import { Dashboard, Login, Register, Admin } from './pages';
8043  import { User } from './model';
8044  import { getAsObject, saveObject } from './utils/storage';
8045  import './App.css';
8046
8047  export default class App extends Component {
8048     constructor(){
8049        super();
8050        const user = getAsObject('user');
8051        this.state = {
8052           logged: getAsObject('logged'),
8053           user: user ? user : new User()
8054        };
8055        this.handlerLogin = this.handlerLogin.bind(this);
8056        this.handlerLogout = this.handlerLogout.bind(this);
8057     }
```

```
8058
8059    handlerLogin(user){
8060       saveObject('user', user);
8061       saveObject('logged', true);
8062       this.setState({ logged: true, user });
8063    }
8064
8065    handlerLogout(){
8066       let user = new User();
8067       saveObject('user', user);
8068       saveObject('logged', false);
8069       this.setState({ logged: false, user });
8070    }
8071
8072    render() {
8073       const { logged, user } = this.state;
8074       return (
8075          <BrowserRouter>
8076             <div>
8077                <Header role={user.role} logged={logged}
                       logout={this.handlerLogout} name={user.name}/>
8078                <Container style={{ margin: '3% 0' }} fluid>
8079                   <Row>
8080                      <Col>
8081                         <Route path="/dashboard" render={() =>
                                (<Dashboard logged={logged}
                                cpf={user.cpf} role={user.role}/>)}/>
8082                         <Route path="/login" render={() => <Login
                                logged={logged}
                                afterLogin={this.handlerLogin}/>}/>
8083                         <Route path="/cadastrar" render={() =>
                                <Register
                                afterSubmit={this.handlerLogin}/>}/>
8084                         <Route path="/admin" render={() => <Admin
                                logged={logged} role={user.role}/>}/>
8085                         <Route exact path="/" render={() =>
                                <Redirect to="/login"/>}/>
8086                      </Col>
8087                   </Row>
8088                </Container>
8089                <Footer/>
8090             </div>
8091          </BrowserRouter>
8092       );
8093    }
```

```
}


webapp/src/registerServiceWorker.js

// In production, we register a service worker to serve assets from local cache.

// This lets the app load faster on subsequent visits in production, and gives
// it offline capabilities. However, it also means that developers (and users)
// will only see deployed updates on the "N+1" visit to a page, since previously
// cached resources are updated in the background.

// To learn more about the benefits of this model, read https://goo.gl/KwvDNy.
// This link also includes instructions on opting out of this behavior.

const isLocalhost = Boolean(
  window.location.hostname === 'localhost' ||
    // [::1] is the IPv6 localhost address.
    window.location.hostname === '[::1]' ||
    // 127.0.0.1/8 is considered localhost for IPv4.
    window.location.hostname.match(
      /^127(?:\.(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)){3}$/
    )
);

export default function register() {
  if (process.env.NODE_ENV === 'production' && 'serviceWorker' in navigator) {
    // The URL constructor is available in all browsers that support SW.
    const publicUrl = new URL(process.env.PUBLIC_URL, window.location);
    if (publicUrl.origin !== window.location.origin) {
      // Our service worker won't work if PUBLIC_URL is on a different origin
      // from what our page is served on. This might happen if a CDN is used to
      // serve assets; see https://github.com/facebookincubator/create-react-app/issues/2374
```

```
      return;
    }

    window.addEventListener('load', () => {
      const swUrl =
          `${process.env.PUBLIC_URL}/service-worker.js`;

      if (isLocalhost) {
        // This is running on localhost. Lets check if a service
            worker still exists or not.
        checkValidServiceWorker(swUrl);

        // Add some additional logging to localhost, pointing
            developers to the
        // service worker/PWA documentation.
        navigator.serviceWorker.ready.then(() => {
          console.log(
            'This web app is being served cache-first by a
                service ' +
              'worker. To learn more, visit https://goo.gl/SC7cgQ'
          );
        });
      } else {
        // Is not local host. Just register service worker
        registerValidSW(swUrl);
      }
    });
  }
}

function registerValidSW(swUrl) {
  navigator.serviceWorker
    .register(swUrl)
    .then(registration => {
      registration.onupdatefound = () => {
        const installingWorker = registration.installing;
        installingWorker.onstatechange = () => {
          if (installingWorker.state === 'installed') {
            if (navigator.serviceWorker.controller) {
              // At this point, the old content will have been
                  purged and
              // the fresh content will have been added to the
                  cache.
              // It's the perfect time to display a "New content
                  is
```

```
8165              // available; please refresh." message in your web
                      app.
8166              console.log('New content is available; please
                      refresh.');
8167            } else {
8168              // At this point, everything has been precached.
8169              // It's the perfect time to display a
8170              // "Content is cached for offline use." message.
8171              console.log('Content is cached for offline use.');
8172            }
8173          }
8174        };
8175      };
8176    })
8177    .catch(error => {
8178      console.error('Error during service worker registration:',
              error);
8179    });
8180 }
8181
8182 function checkValidServiceWorker(swUrl) {
8183   // Check if the service worker can be found. If it can't
         reload the page.
8184   fetch(swUrl)
8185     .then(response => {
8186       // Ensure service worker exists, and that we really are
            getting a JS file.
8187       if (
8188         response.status === 404 ||
8189         response.headers.get('content-type').indexOf('javascript')
              === -1
8190       ) {
8191         // No service worker found. Probably a different app.
              Reload the page.
8192         navigator.serviceWorker.ready.then(registration => {
8193           registration.unregister().then(() => {
8194             window.location.reload();
8195           });
8196         });
8197       } else {
8198         // Service worker found. Proceed as normal.
8199         registerValidSW(swUrl);
8200       }
8201     })
8202     .catch(() => {
```

```
      console.log(
        'No internet connection found. App is running in offline
            mode.'
      );
    });
}

export function unregister() {
  if ('serviceWorker' in navigator) {
    navigator.serviceWorker.ready.then(registration => {
      registration.unregister();
    });
  }
}
```