Makhles Reuter Lange

Atualização Tecnológica do Formulário de Inscrição do Processo Seletivo da Pós-Graduação da UFSC

Florianópolis

2 de julho de 2018

Makhles Reuter Lange

Atualização Tecnológica do Formulário de Inscrição do Processo Seletivo da Pós-Graduação da UFSC

Trabalho de Conclusão de Curso submetido ao Curso de Ciências da Computação da Universidade Federal de Santa Catarina para a obtenção do grau de Bacharel em Ciências da Computação.

Universidade Federal de Santa Catarina - UFSC

Ciências da Computação

Orientador: Andréia Alves dos Santos Schwaab

Coorientador: Leandro José Komosinski

Florianópolis

2 de julho de 2018

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Lange, Makhles Reuter
Atualização Tecnológica do Formulário de Inscrição do Processo Seletivo da Pós-Graduação da UFSC / Makhles Reuter Lange ; orientadora, Andréia Alves dos Santos Schwaab, coorientador, Leandro José Komosinski, 2018.
105 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2018.

Inclui referências.

1. Ciências da Computação. 2. Desenvolvimento Web. 3. Formulário de inscrição. 4. JavaServer Faces. 5. Responsive design. I. Schwaab, Andréia Alves dos Santos. II. Komosinski, Leandro José. III. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. IV. Título.

RESUMO

Utilizando-se a linguagem Java e as tecnologias para desenvolvimento Web próprias do JavaEE, desenvolveu-se um sistema de inscrição para os Programas de Pós-Graduação da UFSC. A produção deste trabalho foi inspirada em um sistema atualmente em uso que está carente de algumas funcionalidades. O novo sistema permite o prenchimento do formulário durante todo o período definido no cronograma do processo seletivo, além do envio de documentos obrigatórios pré-estabelecidos pelas secretarias dos cursos e da importação dos dados de inscrições passadas. A interface do sistema foi desenvolvida de forma responsiva, *i.e.*, a usabilidade em dispositivos móveis também foi levada em consideração no desenvolvimento das páginas. Ainda, empregou-se um conjunto de bibliotecas mais atual do que o do sistema em uso, proporcionando mais recursos ao desenvolvedor. Atualmente o sistema se encontra no estágio de homologação e à espera de testes e identificação de possíveis *bugs* e melhorias.

Palavras-chave: Formulário de Inscrição; Processo Seletivo; Desenvolvimento Web; JavaServer Faces; PrimeFaces; Responsive Web Design.

ABSTRACT

JavaEE Web development technologies were used in the development of a new online Postgraduate Enrollment System for the Federal University of Santa Catarina. The current enrollment system, which lacks some important sought out functionalities, was used as an inspiration in the development of the new one. Candidates can now fill out the application forms at any time during the selection process, upload the required documents and import personal data used in the last submitted application. Responsive web design was taken into consideration while designing the system, *i.e.*, mobile users will not feel left behind when filling out the application form. Lastly, an up-to-date set of libraries were used instead of the old ones, providing a more resourceful environment to developers. The system is currently awaiting tests and improvements.

Keywords: Application Form; Selection Process; Web Development; JavaServer Faces; PrimeFaces; Responsive Web Design.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CAS	Sistema de Autenticação Centralizada
EIS	Enterprise Information System
Java EE	Java Enterprise Edition
JSF	JavaServer Faces
JPA	Java Persistence API
PROPG	Pró-Reitoria de Pós-Gradução
RUP	Rational Unified Process
SeTIC	Superintendência de Governança Eletrônica e Tecnologia da Informação e Comunicação
UFSC	Universidade Federal de Santa Catarina
UML	Unified Modelling Language

SUMÁRIO

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

O acesso aos programas de pós-graduação oferecidos pela Universidade Federal de Santa Catarina é feito através de processos seletivos lançados por suas respectivas secretarias. Todos os programas possuem algum modo para realizar as inscrições em seus processos seletivos. Dentre as formas utilizadas, destacam-se um sistema informatizado oficial, formulários de inscrição manuais e sistemas informatizados não-oficiais, responsáveis pela inscrição de 7097 candidatos em 2017[1]. O uso desses diversos sistemas, com suas funcionalidades e interfaces distintas, acarreta em diversos problemas, tais como:

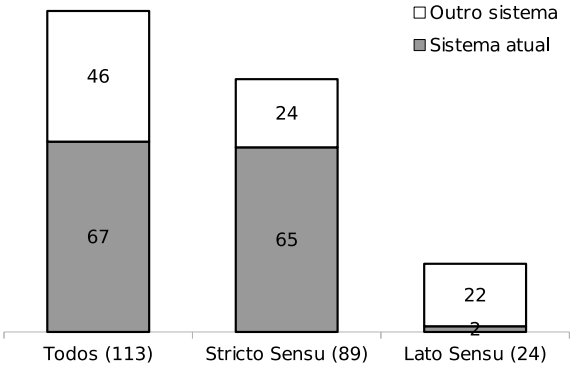
- Falta de padronização dos requisitos e tecnologias utilizadas.
- Autenticação de forma não-centralizada: além de ser uma vulnerabilidade em termos de segurança, implica em duplicação de código.
- Difícil manutenibilidade;
- Difícil obtenção de estatísticas relacionadas aos candidatos.

Em meados de 2010, esperava-se mais de cinco mil inscrições no Programa de Especialização em Saúde da Família. Visando atender essas inscrições, a solução adotada pela Pró-Reitoria de Pós-Gradução (PROPG), em parceria com a Superintendência de Governança Eletrônica e Tecnologia da Informação e Comunicação (SeTIC), foi a criação de um sistema informatizado[2]. Posteriormente, esse sistema foi adotado por outros programas.

A Figura 1 mostra o uso desse sistema pelos programas da UFSC em 2017. Dos 113 programas, apenas 67 (59%) adotaram o sistema para realizar as inscrições. A figura também mostra o uso de acordo com o tipo do programa (*stricto sensu* ou *lato sensu*).

[1] Para os níveis Mestrado, Mestrado Profissional e Doutorado.
[2] Vide http://www.capg.ufsc.br/inscricao

Figura 1 – Utilização dos sistemas de inscrição em 2017.



Fonte: Dados obtidos em consultas ao banco de dados do CAPG.

Não obstante, esse sistema não foi desenvolvido visando a integração dos sistemas existentes e está carente de algumas funcionalidades frequentemente solicitadas pelos programas de pós-graduação. A implementação de tais funcionalidades no sistema atual é custosa, pois sua estrutura contempla os requisitos definidos na época de sua elaboração. Adicionalmente, as bibliotecas e tecnologias utilizadas são antigas (algumas estão, inclusive, descontinuadas), dificultando seu desenvolvimento e sua manutenibilidade. Justifica-se, portanto, o desenvolvimento de um novo sistema de inscrição, que terá como inspiração o sistema atual, porém com a atualização das tecnologias utilizadas e a adição das funcionalidades descritas nas seções 1.1.1.1 e 1.1.1.4.

1.1.1 Funcionalidades Desejadas

1.1.1.1 Envio de Documentos

A funcionalidade mais importante que será implementada neste novo sistema é o envio de documentos de forma digital por parte dos candidatos, que atualmente têm o trabalho de comparecer às secretarias dos cursos com as cópias físicas dos documentos. A praticidade dessa

funcionalidade beneficia tanto o candidato quanto a universidade.

1.1.1.2 Salvamento Progressivo das Informações

Caso um candidato queira fazer a inscrição em algum dos programas de Pós-Graduação da UFSC através do sistema de inscrição atualmente em uso, este deve, essencialmente, fornecer todas as informações em uma única sessão. Ao final, o sistema fornece um número de inscrição e a possibilidade de impressão de um comprovante de inscrição.

Devido à extensão do formulário, pretende-se oferecer ao candidato a opção de salvar o progresso do seu preenchimento. O candidato poderá, dessa forma, distribuir o preenchimento do formulário em várias sessões até que esteja seguro das informações fornecidas e decida-se por finalizar sua inscrição.

1.1.1.3 Ordem de Preenchimento do Formulário

No sistema atual, o preenchimento do formulário segue uma ordem predefinida. Muitas vezes, no entanto, o candidato não possui todas as informações necessárias e/ou relevantes ao iniciar o preenchimento do formulário. Pretende-se dar a possibilidade de preenchimento do formulário sem nenhuma ordem imposta neste processo (a não ser em situações nas quais exista uma ordem implícita).

1.1.1.4 Interface Responsiva

A grande maioria dos *websites* existentes hoje na internet foram desenvolvidos para serem visualizados nas versões *desktop* dos navegadores. Não é novidade, no entanto, que a utilização de dispositivos móveis para a navegação na internet tem crescido nos últimos anos. Segundo dados obtidos na própria SeTIC, aproximadamente 30% dos usuários que fizeram inscrição nos sistemas NDI (Núcleo de Desenvolvimento Infantil) e CAPL (Colégio da Aplicação) em 2017 utilizaram dispositivos móveis (celulares e tablets).

O recente aumento na diversificação de dispositivos utilizados para a navegação na internet tem exigido das aplicações web a capacidade de disponibilizar interfaces específicas para cada dispositivo. Identifica-se, portanto, a necessidade do desenvolvimento de uma interface com o usuário que se ajuste da melhor forma possível aos diversos dispositivos por ele utilizados.

1.2 SETIC

A Superintendência de Governança Eletrônica e Tecnologia da Informação e Comunicação (SeTIC), setor de informática da Universidade Federal de Santa Catarina (UFSC), é responsável pelo planejamento, pesquisa, aplicação e desenvolvimento de produtos e serviços de tecnologia da informação e comunicação da universidade[3]. As aplicações desenvolvidas pela SeTIC são solicitadas e acompanhadas por usuários dos diversos setores da UFSC.

Ressalta-se que o autor foi estagiário na SeTIC por dois anos, período no qual se familiarizou com as tecnologias utilizadas no presente trabalho, e que teve uma bolsa de extensão fornecida pela FEESC para o seu desenvolvimento. Todas as atividades foram desenvolvidas nas instalações da SeTIC, com equipamentos e tecnologias disponibilizados por esta. Assim, o último estágio do desenvolvimento do presente sistema, *i.e.*, a *operacionalização* e a *manutenção*[4], serão feitas pela SeTIC.

1.3 OBJETIVOS

1.3.1 Objetivo Geral

O objetivo principal deste Trabalho de Conclusão de Curso é desenvolver um novo sistema informatizado para a inscrição de candidatos nos processos seletivos dos diversos programas de Pós-Graduação da

[3] Mais informações em http://setic.ufsc.br/apresentacao/

[4] O sistema é liberado e implantado no ambiente de produção. Eventuais erros que não foram descobertos nos estágios anteriores e novos requisitos podem surgir ao longo do uso do sistema, justificando a sua constante manutenção.

UFSC, tendo como inspiração o sistema de inscrição que está em uso atualmente.

1.3.2 Objetivos Específicos

- Implementar o *back-end* da aplicação, *i.e.*, as *regras de negócio* relativas à inscrição em um processo seletivo.
- Criar as tabelas necessárias no banco de dados.
- Implementar o *front-end* da aplicação, *i.e.*, as páginas que o candidato terá acesso ao realizar sua inscrição.
- Tornar o sistema acessível em dispositivos móveis.
- Disponibilizar o sistema em ambiente de homologação.

1.4 RESULTADOS ESPERADOS

- Sistema CAPG Inscrição publicado no ambiente de homologação.
- Maior grau de adesão ao novo sistema por parte dos Programas de Pós-Graduação da UFSC.
- Melhoria da gestão dos dados dos processos de inscrição, tanto por parte dos programas, quanto da própria PROPG.
- Artigo referente à produção realizada anexo ao TCC.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 PLATAFORMA JAVA, EDIÇÃO EMPRESARIAL

As **aplicações empresariais** têm o propósito de fornecer a *lógica do negócio* de uma empresa. O seu gerenciamento é feito de forma centralizada e é comum a sua interação com outros softwares empresariais. A plataforma Java, Edição Empresarial (Java EE), é um conjunto de tecnologias Java que são utilizadas para o desenvolvimento de aplicações empresariais. O objetivo da plataforma é fornecer aos desenvolvedores um conjunto de especificações / APIs que permitam a diminuição do tempo de desenvolvimento, a redução da complexidade e o aumento do desempenho de aplicações empresariais (JENDROCK et al., 2014). Tais especificações são *contratos* que são implementados por diversos fornecedores, *e.g.*, GlassFish, Oracle WebLogic, Apache TomEE, etc.

2.1.1 Aplicações Multi-Camadas

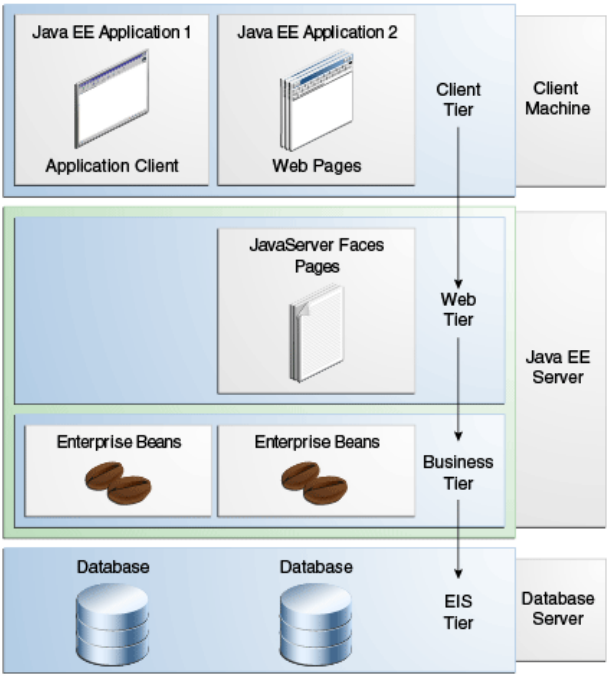
Segundo o modelo Java EE, as aplicações são distribuídas em camadas (*tiered design*)[1]. Define-se, desta forma, responsabilidades distintas para as diferentes partes do sistema, *i.e.*, para os diferentes componentes que compõem uma aplicação Java EE (vide Figura 2) e que são distribuídos nas camadas Cliente, Web, Negócio e EIS.

2.1.1.1 Camada Cliente

Esta camada é composta de clientes aplicativos que acessam o servidor Java EE e que geralmente se localizam em máquinas diferentes da máquina do servidor:

[1] Os termos *tier* e *layer* são ambos traduzidos para o português como "camada". No entanto, uma *tier* representa uma unidade física, na qual um código ou processo é executado. Uma *layer*, por sua vez, representa uma unidade lógica, responsável pela organização lógica do código através da abstração dos dados. Diversas *layers* podem existir em computadores diferentes, ou em processos diferentes em um único computador, ou ainda em um único processo em um único computador (LHOTKA, 2005).

Figura 2 – Componentes Java EE distribuídos nas diversas camadas de duas aplicações Web.



Fonte: adaptado de Jendrock et al. (2014)

- Clientes Web - também chamados de *clientes magros*, são compostos de páginas dinâmicas (*e.g.*, XHTML) geradas na camada Web e por um navegador responsável por renderizá-las.

- Clientes Aplicativos - quando os usuários necessitam realizar atividades que requerem uma interface mais rica do que a disponibilizada por linguagens de marcação, utilizam-se clientes aplicativos baseados nas APIs Swing ou AWT (*Abstract Window Toolkit*). Não obstante, pode-se utilizar uma interface em linha de comando. Embora seja possível o estabelecimento de uma conexão HTTP com um *servlet* na camada Web, clientes aplicativos costumam

acessar diretamente os *beans* gerenciais da camada de negócio.

- *Applets* - componentes embarcados que são incluídos nas páginas web. São escritos em Java e, portanto, necessitam da máquina virtual Java instalada no navegador web do cliente (e provavelmente do *plugin* Java e de um arquivo de política de segurança).

2.1.1.2 Camada Web

É a camada responsável pela interação entre os clientes e a camada de negócios. Suas principais funções são:

- Gerar, dinamicamente, conteúdo para o cliente em diversos formatos.
- Obter as entradas (dados e ações) da interface com o cliente e retornar os resultados dos componentes da camada de negócios.
- Controlar o fluxo das páginas no cliente.
- Manter o estado dos dados da sessão do usuário.
- Executar um pouco de lógica simples e armazenar dados em componentes JavaBeans temporariamente.

Servlets (vide Anexo A) e páginas web criadas com a tecnologia *JavaServer Faces* (vide seção 2.1.2) são os componentes desta camada.

2.1.1.3 Camada de Negócio

A camada de negócios possui componentes que provêm a lógica do negócio da aplicação, ou seja, código que provê funcionalidades para um determinado domínio do negócio. As tecnologias utilizadas nessa camada são as seguintes:

- Componentes Enterprise JavaBeans (EJB), que fornecem funcionalidades como gerenciamento de sessão, segurança, gerenciamento de transações, etc.

- *Web services* (JAX-RS, JAX-WS).
- Java Persistence API (JPA), para a criação e gerenciamento de entidades (vide Seção 2.1.5).

2.1.1.4 Camada EIS

A camada EIS (*Enterprise Information System*) consiste de servidores de bancos de dados, sistemas de planejamento de recursos empresariais e outras fontes de dados legados (JENDROCK et al., 2014). Esses recursos geralmente se localizam em suas próprias máquinas e são acessados pela camada de negócios. As tecnologias utilizadas nessa camada são:

- Java Database Connetivity API (JDBC) - uma interface que permite a conexão às bases de dados.
- Java Persistence API (JPA).
- Java Transaction API (JTA) - uma interface para a realização de transações.

2.1.2 JavaServer Faces

A tecnologia JavaServer Faces (JSF) é uma *especificação* de uma API Java utilizada para a criação de interfaces de usuário em aplicações web. Com essa API, é possível:

- Representar componentes[2] e gerenciar seus estados;
- Fazer o controle de eventos gerados;
- Realizar a validação e a conversão de dados no lado do servidor;
- Definir regras para a navegação entre páginas;
- Prover o suporte à internacionalização e acessibilidade;

[2] Componentes, ou *widgets*, são entidades autônomas e reutilizáveis da interface de usuário.

Existem diversas implementações da especificação — Apache MyFaces e Oracle Mojarra são as principais. Ambas contêm pelo menos os componentes padrões, ou seja, os componentes responsáveis por gerar qualquer um dos elementos HTML básicos (tabelas, caixas de entrada de texto, botões, seletores, etc). Bibliotecas de componentes podem ser utilizadas como suporte às implementações (vide Seção 2.1.4).

2.1.2.1 Arquitetura do Framework

O *framework* JSF segue o padrão arquitetural[3] MVC baseado em componentes. A grande vantagem do padrão MVC é a separação entre apresentação e comportamento (lógica) da aplicação:

- **M**odelo - representa o comportamento da aplicação em termos do domínio do problema.
- **V**isão - fornece a representação da informação para o usuário da aplicação. Pode-se ter diversas visões do mesmo modelo. A tecnologia padrão utilizada pelo JSF é a Facelets (vide Seção 2.1.3).
- **C**ontrolador - converte entradas/eventos em comandos para a visão ou para o modelo.

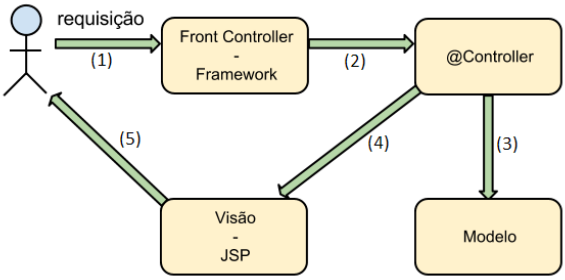
Os *frameworks* MVC podem ser baseados em ações (*action-based*) ou em componentes (*component-based*). Em ambos os tipos, todas as requisições feitas à aplicação devem passar primeiramente por um controlador frontal, como pode ser visto nas Figuras 3 e 4.

Quando se utiliza um *framework* baseado em ação (*e.g.*, SpringMVC, Struts, ASP.NET, etc), o controlador frontal é um *servlet* que irá lidar com objetos `HTTPServletRequest` e `HTTPServletResponse`, e delegar as requisições para métodos de outros controladores (classes que estendem a classe `HttpServlet` - vide o Algoritmo 4, no Anexo A) de acordo

[3] Um padrão arquitetural expressa um esquema de organização estrutural para sistemas baseados em *software*. O padrão provê um conjunto de subsistemas predefinidos, especifica suas responsabilidades, e inclui regras e diretrizes para organizar a relação entre eles (BUSCHMANN et al., 1996).

com o caminho do recurso desejado e dos parâmetros da requisição. A obtenção, conversão e validação dos parâmetros das requisições e a atualização dos valores do modelo são responsabilidades do desenvolvedor da aplicação.

Figura 3 – *Framework* MVC baseado em ações. Os números ao lado das setas indicam a ordem do processamento de uma requisição.



Fonte: Adaptado de Almeida (2012).

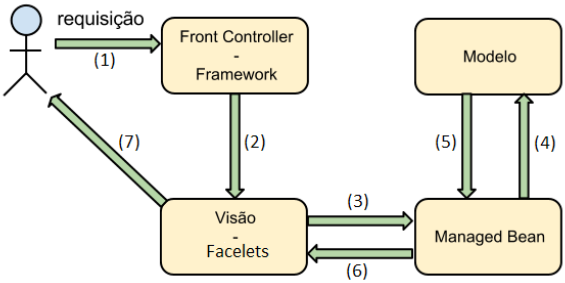
O JSF é um *framework* baseado em componentes. Como tal, possui um controlador frontal, o `FacesServlet`, que é o *servlet* responsável por gerenciar o ciclo de vida do processamento de requisições. Dessa forma, o desenvolvedor da aplicação não precisa se preocupar em escrever código *boilerplate*[4], tal como no caso de um framework baseado em ações.

Resumidamente, a requisição feita pelo usuário da aplicação é recebida pelo `FacesServlet`, que delega o processamento para a visão responsável pela geração da página acessada. O componente acessado (por exemplo, um botão) invoca um método em um *managed bean*[5]. Este, por sua vez, pode buscar dados do modelo e retorná-los para a visão. Finalmente, a visão envia a resposta ao usuário.

[4] Trechos de código que são incluídos em diversos lugares com um mínimo de alteração.

[5] Uma classe Java que trabalha junto à visão.

Figura 4 – *Framework* MVC baseado em componentes. Os números ao lado das setas indicam a ordem do processamento de uma requisição.



Fonte: Adaptado de Almeida (2012).

O ciclo de vida do processamento de requisições do JSF é, na verdade, bem mais complexo do que o apresentado, pois envolve a criação de uma árvore de componentes, a conversão e validação dos dados obtidos destes componentes, o gerenciamento de eventos oriundos destes componentes e a propagação dos dados para os *beans*.

2.1.3 Facelets

Facelets é uma tecnologia de apresentação utilizada em aplicações baseadas em JSF. Atualmente, é também a tecnologia definida na especificação do JSF, substituindo assim a descontinuada JSP (JavaServer Pages).

Com esta tecnologia, as páginas web são criadas em XHTML e os componentes são inseridos através de *tags* de diversas bibliotecas. No Algoritmo 1, por exemplo, utiliza-se o componente `<h:inputText>`, da biblioteca JSF HTML, para a entrada de texto, e `<f:validateLongRange>`, da biblioteca Core, para a validação do texto entre um valor mínimo e um máximo.

Algoritmo 1 – Utilização de *tags* em um arquivo XHTML.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core">

<h:body>
  <h:form>
    <h:inputText id="userNumber"
                 title="Insira um numero entre 0 e 10:"
                 value="#{aBean.userNumber}">
      <f:validateLongRange minimum="#{aBean.minimum}"
                           maximum="#{aBean.maximum}"/>
    </h:inputText>
    <h:commandButton id="submit"
                     value="Submit"
                     action="response"/>
  </h:form>
</h:body>
```

Fonte: Adaptado de Jendrock et al. (2014).

Dentre suas vantagens, Jendrock et al. (2014) destacam:

- Reuso de código através de *templates* e da composição de componentes.
- Redução do tempo de compilação.
- Validação de expressões em *Expression Language*[6] em tempo de compilação.

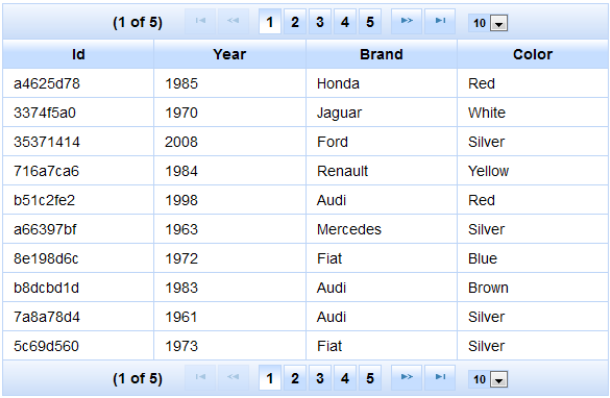
2.1.4 Primefaces

Bibliotecas de componentes oferecem funcionalidades previamente testadas e eventualmente úteis, tais como tabelas capazes de ordenar, filtrar e selecionar linhas, organização do conteúdo em diversos tipos de painéis e menus, exibição de coleções em listas e árvores, etc, além da possibilidade de escolha de diversos temas. Existem diversas bibliotecas

[6] Mecanismo que permite a comunicação entre as páginas da web e os *managed beans*.

disponíveis — PrimeFaces, RichFaces e ICEFaces são as mais conhecidas. Nos sistemas baseados em Java atualmente desenvolvidos pela SeTIC, utiliza-se a biblioteca PrimeFaces. A Figura 5 mostra um exemplo de tabela feita com essa biblioteca.

Figura 5 – Componente DataTable da biblioteca PrimeFaces. Evidencia-se o uso de *paginators* no cabeçalho e rodapé, e o uso do tema *bluesky*.



Fonte: PrimeFaces (2017).

2.1.5 Java Persistence API

A API de Persistência do Java (JPA) possibilita o gerenciamento de dados relacionais nas aplicações Java através de um mapeamento objeto-relacional. Esse mapeamento é feito através de *entidades*, *i.e.*, objetos do domínio de persistência. Uma entidade representa uma tabela de um banco de dados no modelo relacional, e suas instâncias representam as linhas desta tabela.

Segundo Jendrock et al. (2014), para ser considerada uma entidade, uma classe Java precisa:

1. Possuir a anotação `javax.persistence.entity`;

2. Possuir ao menos um construtor sem argumentos com visibilidade *public* ou *protected*;

3. A classe, suas instâncias persistíveis e seus métodos não podem ser declarados *final*;

4. Instâncias persistíveis devem ser declaradas como *private*, *protected* ou *package private* e podem ser acessadas somente através dos métodos da classe.

Tais restrições podem ser vistos na entidade `Empregado`, destacadas em forma de comentários no Algoritmo 2.

Algoritmo 2 – Entidade Empregado e seus mapeamentos.

```
@Entity // 1)
public class Empregado {
    @Id
    @Column(name="EMP_ID")
    private long id;
    private String nome;
    @ManyToOne
    private Setor setor; // 3 e 4

    public Empregado() {}  // 2
    // ...
}
```

Ainda com relação ao Algoritmo 2, é interessante observar as seguintes anotações:

- `@Id` - indica que o campo é a chave primária da tabela.
- `@Column` - indica que o campo é uma coluna da tabela. Quando o parâmetro `name` não é especificado, assume-se que o nome da coluna é idêntico ao nome do campo (é o caso do campo `nome`).
- `@ManyToOne` - indica a relação entre as entidades `Empregado` e `Setor`. Neste caso, diversos empregados estão relacionados ao

mesmo setor. Keith e Schincariol (2013) definem outras anotações utilizadas no relacionamento de entidades além de `@ManyToOne`. Para maiores detalhes, vide Anexo B.

3 ANÁLISE E PROJETO - MÓDULO DE INSCRIÇÃO

Sommerville (2011) explica que um *processo de software* é um conjunto de atividades relacionadas que levam à produção de um produto de software. Embora existam diversos processos de software, todos devem incluir as seguintes atividades fundamentais: especificação, projeto e implementação, validação, e evolução do software. Cada uma destas atividades é uma tarefa complexa em si mesma e inclui subatividades.

Por não existir um modelo de processo de software único a ser utilizado na SeTIC, o autor decidiu-se por seguir algumas das melhores práticas estabelecidas pelo *Rational Unified Process* (RUP), as quais já foram utilizadas em projetos anteriores em que o autor participou na SeTIC como estagiário (vide Anexo C).

Ressalta-se que o desenvolvimento teve como inspiração o sistema existente. Fez-se uso de diversas tabelas já existentes, bem como dos campos presentes no formulário de inscrição. Não obstante, novas tabelas e campos foram criados, e a implementação das regras de negócio foi realizada pelo autor deste trabalho.

3.1 ELICITAÇÃO, ANÁLISE E DEFINIÇÃO DE REQUISITOS

O desenvolvimento de um software/sistema segundo o modelo adotado pode ser dividido em diversos estágios. O primeiro consiste na elicitação, análise e definição de requisitos. Segundo Sommerville (2011), os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento. Esses requisitos refletem as necessidades dos clientes para um sistema que serve a uma finalidade determinada, como controlar um dispositivo, fazer um pedido ou encontrar informações. Esta etapa é muito importante pois, além de servir de base para as demais etapas, fornece um rumo a ser seguido e um objetivo a ser alcançado. Não menos importante, evita a implementação de funcionalidades desnecessárias.

A elicitação ou descoberta dos requisitos do sistema pode ser

feita de diversas maneiras: entrevistas, cenários, casos de uso, etnografia, etc. A abordagem adotada, comum aos demais sistemas desenvolvidos na SeTIC, foi a de realizar de entrevistas abertas, isto é, uma série de questões relacionadas ao uso do sistema foi explorada junto à Pró-Reitoria de Pós-Gradução (PROPG), o cliente deste projeto. Dessa forma, pôde-se obter uma melhor compreensão de suas necessidades.

Um conjunto de *wireframes*, ou protótipos, das páginas do sistema foi desenvolvido[1] e apresentado com o objetivo de validar a estrutura, o conteúdo, as diversas funções e os diversos caminhos de interações dos usuários com o sistema. A confecção deste conjunto de *wireframes* teve como base o sistema informatizado atualmente em uso. Em cima deste foram aplicadas todas as mudanças que se faziam necessárias e que justificam o desenvolvimento do novo sistema. A Figura 6 fornece um exemplo retirado deste conjunto: após o início da sessão no sistema, a página inicial exibida para o usuário Fulano apresenta um histórico de inscrições já realizadas por este usuário. Evidencia-se o uso de balões explicativos para agilizar o entendimento dos diversos elementos da página. Ressalta-se que o visual da página não é o foco do *wireframe*, e sim o que ela permite que o usuário realize.

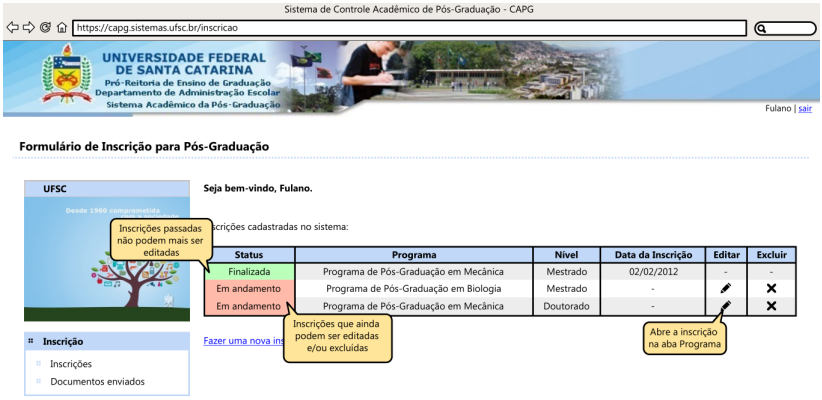
Na reunião com o cliente, os requisitos propostos e exemplificados nos *wireframes* foram analisados. Destes, alguns foram removidos, outros adicionados. No fim, um conjunto de requisitos iniciais foi aprovado e serviu como uma especificação formal para o desenvolvimento do sistema. Embora a distinção entre os diferentes tipos de requisitos não seja tão clara, os requisitos de software são frequentemente classificados como *funcionais* e *não funcionais*. Essa classificação é abordada nos itens a seguir.

3.1.1 Requisitos Funcionais

Os requisitos funcionais descrevem os serviços que o sistema deve fornecer, como deve reagir a entradas específicas e como deve

[1] Utilizou-se um *plugin* do WireframeSketcher feito para o Eclipse.

Figura 6 – *Wireframe* que representa a página de inscrições realizadas pelo usuário Fulano após este ter feito *login* no sistema.



Fonte: Elaborada pelo autor (2018).

se comportar em determinadas situações. Visando o entendimento por todas as partes interessadas (*stakeholders*), a especificação dos requisitos é feita aqui de forma abstrata (em alto nível).

Requisito 1 - Visualizar programas oferecidos. O usuário poderá visualizar a lista de programas de pós-graduação oferecidos pela UFSC sem a necessidade de realizar *login* no sistema.

Requisito 2 - Realizar cadastro no CAS. O usuário poderá se cadastrar no Sistema de Autenticação Centralizada a partir da página inicial do sistema.

Requisito 3 - Iniciar sessão. O usuário poderá iniciar a sessão no sistema através de *login*.

Requisito 4 - Encerrar sessão. O usuário poderá encerrar a sessão no sistema a qualquer momento através de *logout*.

Requisito 5 - Salvar a inscrição. O usuário poderá salvar as informações inseridas no formulário para um acesso futuro. Após o primeiro salvamento, o usuário receberá um número de inscrição.

Requisito 6 - Alterar informações. O usuário poderá alterar as informações inseridas durante todo o período de inscrição do programa selecionado, desde que não tenha finalizado a sua inscrição.

Requisito 7 - Finalizar a inscrição. O usuário poderá finalizar a sua inscrição. Após finalizada, os dados inseridos não poderão mais ser modificados.

Requisito 8 - Receber um comprovante de inscrição. Após finalizada a inscrição, o usuário receberá um comprovante de inscrição através do e-mail cadastrado.

Requisito 9 - Importar dados do CAS. Os dados que foram utilizados para fazer o cadastro no CAS deverão ser importados automaticamente.

Requisito 10 - Importar dados de inscrições anteriores. Ao iniciar uma nova inscrição, o usuário poderá importar dados pessoais e dados de contato da última inscrição realizada.

Requisito 11 - Fazer *upload* de arquivos. O usuário poderá fazer o *upload* de arquivos solicitados pelo programa escolhido.

Requisito 12 - Importar arquivos. O usuário poderá buscar arquivos utilizados em inscrições anteriores.

Requisito 13 - Inserir informações. O usuário poderá inserir diversos tipos de informações, as quais são agrupadas em dados do programa, pessoais, econômicos, de contato e de formação. Embora esses dados não estejam descritos em pormenores neste relatório, estavam presentes nos protótipos apresentados ao cliente na reunião de definição dos requisitos.

3.1.2 Requisitos Não Funcionais

Segundo Sommerville (2011), os requisitos não funcionais são aqueles que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários. Relacionam-se às propriedades emergentes do sistema e definem restrições sobre a sua implementação. Foram identificados os seguintes requisitos:

- Usuários do sistema devem autenticar-se utilizando o Sistema de Autenticação Centralizada (CAS).
- A aplicação Web deve ser feita utilizando a linguagem de programação Java e as tecnologias que são comumente utilizadas no desenvolvimento Web com Java e que estão disponíveis na SeTIC (vide Seção 2.1).
- A aplicação Web deve utilizar a estrutura e seguir o visual dos demais sistemas desenvolvidos na SeTIC[2]. Para tanto, deve-se utilizar o projeto denominado *Projeto Base*, criado por desenvolvedores da SeTIC para o desenvolvimento de novos sistemas.
- O sistema deve possuir uma interface responsiva, *i.e.*, capaz de ajustar a disposição do conteúdo em função das dimensões da janela do navegador e do tipo de dispositivo utilizado (*desktop* ou móvel).

3.2 PROJETO DO SISTEMA

O segundo estágio do modelo de desenvolvimento de software adotado consiste em desenvolver a arquitetura geral do sistema. Aqui descrevem-se as principais abstrações do sistema e seus relacionamentos através dos diagramas de casos de uso e de classes.

[2] Este requisito se refere à versão *desktop* da aplicação.

3.2.1 Diagrama de Casos de Uso

Diagramas de casos de uso fornecem uma representação em alto nível da interação entre os diversos usuários (chamados de atores) com o sistema. A Figura 7 relaciona os casos de uso identificados para o sistema de inscrição. As interações são representadas por elipses e os atores por bonecos palito. Identificam-se os seguintes atores:

- **Candidato**, o principal ator e que atua em todos os casos de uso identificados.
- **CAS**, ator que interage apenas com os casos de uso relacionados à autenticação do ator Candidato.
- **Sybase**, uma interface para o sistema de gerenciamento do banco de dados e que atua em todos os casos de uso que necessitam de acesso ao banco de dados.
- **Storage**, uma interface para o repositório de documentos enviados pelo ator Candidato.

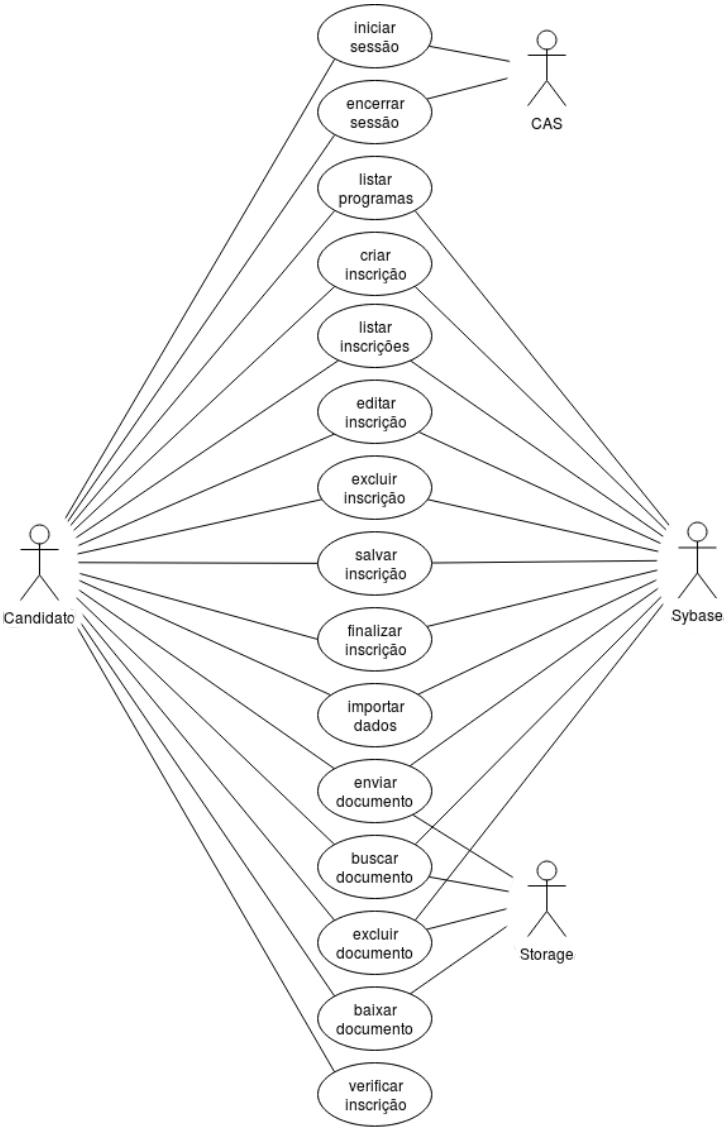
Nota-se também que o caso de uso *verificar inscrição* é o único caso de uso que interage com apenas um ator. Isto se deve ao fato de que a verificação de campos não-preenchidos e de campos com erros de preenchimento é feita em parte pelo JSF e em parte no *bean* associado a *view* de inscrição (ambos na camada Web).

3.2.2 Diagrama de Visão Geral de Interação

Introduzido na versão 2.0 da UML, este diagrama é uma versão modificada do diagrama de atividades, no qual ao invés de ações ou atividades, os nós correspondem a *interações* que são modeladas em outros diagramas. Sua finalidade é modelar as sequências possíveis de ocorrência das interações identificadas.

Uma possível aplicação do diagrama de visão geral de interação é a modelagem do fluxo entre as interações associadas aos casos de uso

Figura 7 – Casos de uso identificados no sistema.



Fonte: Elaborada pelo autor (2018).

identificados anteriormente. Tem-se, assim, uma visão geral do sistema em execução durante um processo de inscrição. O fluxo de execução pode ser visto na Figura 8. Algumas guardas foram omitidas para não saturar o diagrama.

Identificam-se os seguintes elementos:

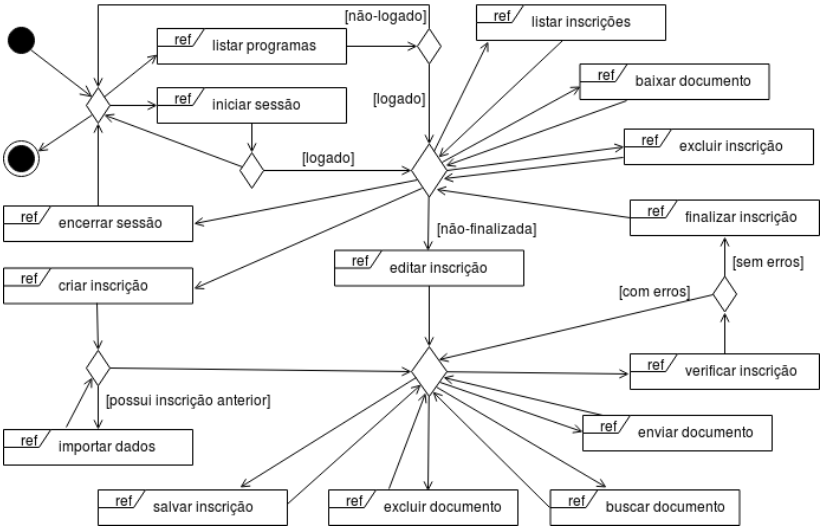
- **Nós de uso de interação**, que referenciam as interações associadas aos casos de uso.
- **Nós de decisão**, responsáveis por receber o fluxo a partir de uma ou mais arestas e selecionar a direção do fluxo de execução de acordo com as guardas.
- **Guardas**, ou condições booleanas escritas entre colchetes, condicionam a saída do nó de decisão.
- **Nó inicial**, representado por um círculo preto, a partir do qual inicia-se o fluxo de execução.
- **Nó final**, representado por um círculo preto inscrito em uma circunferência preta, no qual termina o fluxo de execução.

3.2.3 Diagrama de Classes

Um diagrama de classes é responsável por mostrar as diversas classes de uma aplicação e as associações (agregação, composição, generalização e dependências) entre elas.

O diagrama de classes do sistema foi dividido em duas imagens por questão de espaço. A Figura 9 foi desenvolvida com a ferramenta ObjectAid UML Explorer e mostra as classes envolvidas na obtenção dos programas disponíveis em uma determinada data. Pode-se perceber uma associação de generalização entre as classes `ProgramasService` e `ProgramasServiceImpl` e dependências entre as classes `ProgramasBean` e `ProgramaAux`, e `ProgramasBean` e `ProgramasService`.

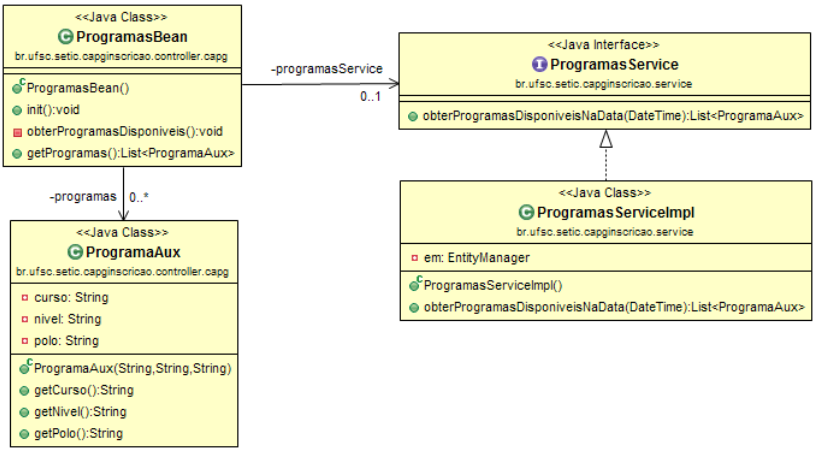
Figura 8 – Diagrama de Visão Geral de Interação.



Fonte: Elaborada pelo autor (2018).

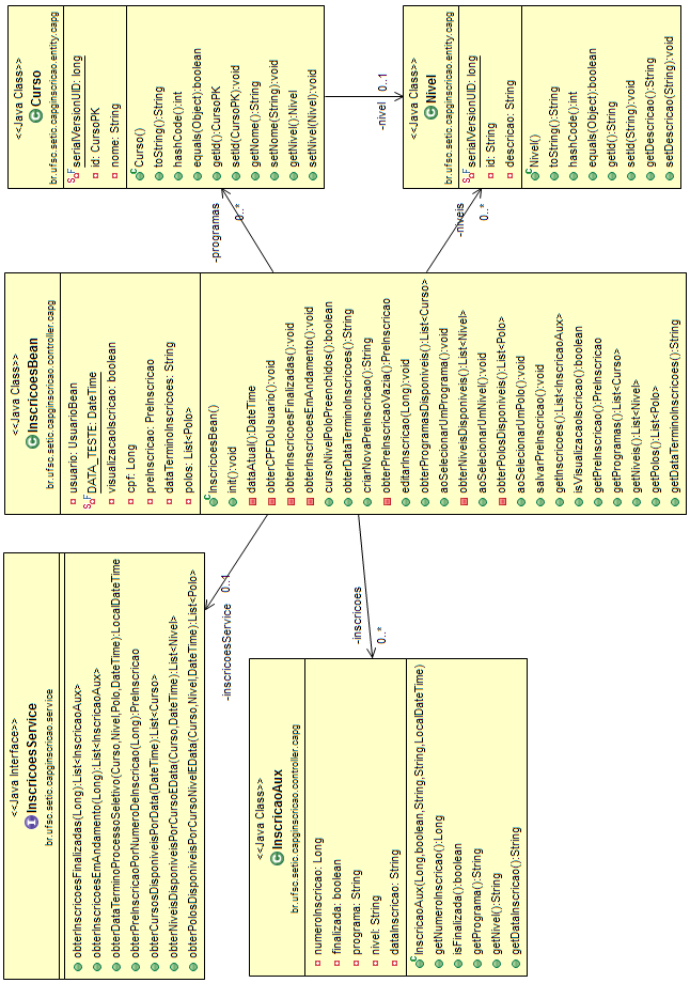
A Figura 10 ilustra a classe `InscricoesBean` e algumas das suas associações de dependência. As classes `Curso` e `Nivel` são entidades e estão relacionadas entre si. `InscricoesService` é uma interface que provê serviços à classe `InscricoesBean`. Finalmente, `InscricaoAux` é uma classe auxiliar que é utilizada junto à *view* para exibir informações das inscrições cadastradas para um determinado usuário.

Figura 9 – Diagrama de classes relacionado à obtenção de programas.



Fonte: Elaborada pelo autor (2018).

Figura 10 – Diagrama de classes relacionado às inscrições.

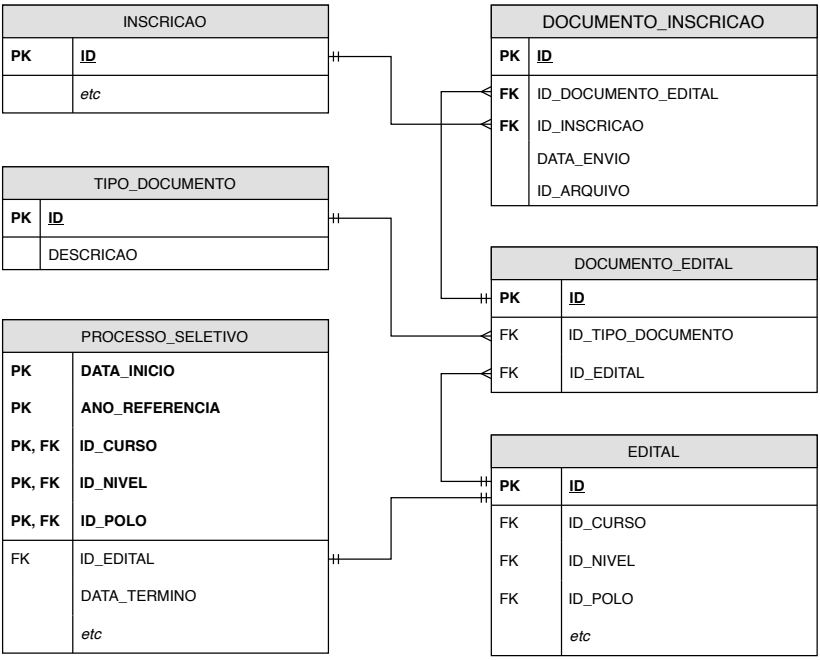


Fonte: Elaborada pelo autor (2018).

3.2.4 Diagrama de Entidade-Relacionamento

Tendo em vista o requisito funcional de permitir ao candidato enviar e importar documentos, foi desenvolvido o modelo de relacionamento entre entidades exibido na Figura 11.

Figura 11 – Modelo entidade-relacionamento associado ao salvamento de documentos.



Fonte: Elaborada pelo autor (2018).

As entidades `INSCRICAO`, `PROCESSO_SELETIVO` e `EDITAL` já existiam na base de dados em forma de tabelas. A primeira engloba os dados pessoais do candidato; as outras duas representam processos seletivos e seus respectivos editais. As demais entidades foram modeladas para atender aos requisitos 11 e 12 (vide Seção 3.1.1):

- `TIPO_DOCUMENTO` - contém os tipos dos documentos obrigatórios

para a inscrição nos diversos programas, *e.g.*, histórico escolar, diploma de graduação, RG, *etc.*

- `DOCUMENTO_INSCRICAO` - entidade que armazena o código dos documentos enviados por cada candidato. Os documentos propriamente ditos são armazenados em um repositório e seus identificadores salvos no campo `ID_ARQUIVO`. Pode-se obter, por exemplo, uma lista de códigos de documentos enviados por um determinado candidato e, a partir da relação com a entidade `DOCUMENTO_EDITAL`, qual o tipo de cada documento e a qual processo seletivo pertence.

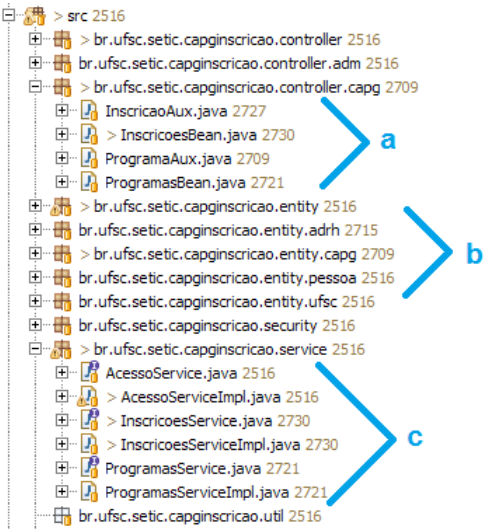
- `DOCUMENTO_EDITAL` - entidade associativa entre as entidades `EDITAL` e `TIPO_DOCUMENTO`, com chave própria.

4 IMPLEMENTAÇÃO

4.1 CONFIGURAÇÃO DO PROJETO

Após definida a estrutura inicial do sistema, deu-se início ao processo de implementação. O projeto foi configurado para atender às necessidades do sistema. Além de arquivos de configuração, diversos pacotes foram criados para a separação lógica das classes, como pode ser visto na Figura 12.

Figura 12 – Alguns pacotes e classes da aplicação vistos no Eclipse.



Fonte: Elaborada pelo autor (2018).

O pacote `(a)` é composto de *beans* e classes auxiliares da camada Web (vide Seção 2.1.1.2) que, em conjunto com os arquivos .xhtml, são responsáveis por gerar as páginas da aplicação.

O mapeamento objeto-relacional se encontra no pacote destacado por `(b)`. Aqui se encontram as classes das inúmeras entidades do sistema (vide Seção 2.1.1.3) e, por serem muitas, não foram incluídas na imagem.

As classes representadas por (c), assim como as classes do item (b), pertencem à camada de negócios. São classes ditas de serviço, responsáveis pelas operações básicas de persistência: *Create*, *Read*, *Update*, *Delete*, comumente abreaviadas por CRUD. Na Figura 12, as classes `AcessoService`, `InscricoesService` e `ProgramasService` são interfaces[1], enquanto que `AcessoServiceImpl`, `InscricoesServiceImpl` e `ProgramasServiceImpl` representam suas implementações.

Para ilustrar a operação *read*, o Algoritmo 3 contém o método que obtém a lista de programas disponíveis em uma determinada data. Percebe-se que o tipo do elemento contido na lista é um `ProgramaAux`, uma das classes auxialiares mencionadas no item (a). A instância do `EntityManager`, que está associada a um contexto de persistência, é representada por `em`.

Algoritmo 3 – Obtenção da lista de programas disponíveis.

```
public List<ProgramaAux>
obterProgramasDisponiveisNaData(DateTime data) {
    StringBuffer query = new StringBuffer();
    query.append("SELECT new br.ufsc...ProgramaAux(");
    query.append("ps.curso.nome, ps.nivel.descricao, ");
    query.append("ps.polo.nome) ");
    query.append("FROM ProcessoSeletivo ps ");
    query.append("WHERE :dt BETWEEN ps.dtInicioProcesso ");
    query.append("AND ps.dtTerminoProcesso ");
    query.append("ORDER BY ps.curso");

    return em.createQuery(query.toString(),
                          ProgramaAux.class)
             .setParameter("dt", data.toLocalDateTime())
             .getResultList();
}
```

Fonte: Produzido pelo autor.

Ressalta-se que a maioria das classes presentes na Figura 12 já existiam no Projeto Base e provêm funcionalidades como segurança

[1] Em Java, uma interface é um tipo abstrato que especifica um comportamento que deve ser implementado.

(através do Spring), acesso a Web Services, comunição com a base de dados Centura, validadores e conversores do JSF, *etc.*

4.2 DISPOSITIVOS MÓVEIS

A adaptação da interface do sistema para a utilização em dispositivos móveis é um requisito não-funcional, cuja implementação teve de levar em consideração duas abordagens:

- A utilização de um *add-on* da biblioteca PrimeFaces chamado PrimeFaces Mobile (PFM), o qual permite a criação de aplicações Web baseadas em JSF otimizadas para o uso em dispositivos móveis. Nesta abordagem, utilizam-se *views* separadas para navegadores de dispositivos móveis e para navegadores comuns, e reutilizam-se os métodos da camada de negócio da aplicação. Embora isso permita maior usabilidade e experiência mais agradável para usuários acostumados a utilizar aplicativos móveis, a duplicação de código dificulta a manutenibilidade da aplicação.
- A utilização de páginas responsivas: as mesmas *views* são utilizadas em navegadores comuns e navegadores de dispositivos móveis, e a disposição dos elementos é controlada a partir de *CSS media queries*.

Tendo em vista a manutenibilidade do código, optou-se pela segunda abordagem. As imagens utilizadas na exemplificação das funcionalidades mostradas a seguir foram obtidas em navegadores comuns. Imagens obtidas em um navegador de um dispositivo móvel[2] podem ser vistas no Anexo D.

4.3 FUNCIONALIDADES

A Tabela 1 mostra um mapeamento entre os casos de uso identificados para o ator Candidato (vide Seção 3.2.1) e as páginas nas quais

[2] Motorola Moto G5 com resolução de 1920x1080 pixels.

foram implementados. A implementação de cada item está descrita nas subseções a seguir.

Quadro 1 – Casos de uso implementados e respectivas páginas envolvidas.

Caso de Uso	**Página da Aplicação**
Listar programas	`programas.xhtml`
Iniciar sessão	`login.xhtml`
Encerrar sessão	A partir de qualquer página
Listar inscrições Criar inscrição Editar inscrição Excluir inscrição	`inscricoes.xhtml`
Enviar documento	`enviar_documento.xhtml`
Buscar documento	`buscar_documento.xhtml`
Baixar documento	`documentos_enviados.xhtml`
Excluir documento Importar dados Salvar inscrição Finalizar inscrição Verificar inscrição	`inscricao.xhtml`

Fonte: Elaborada pelo autor (2018).

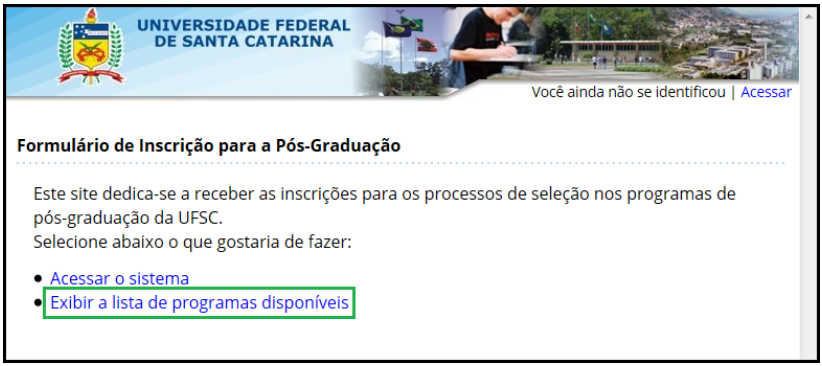
4.3.1 Lista de Programas Disponíveis

O acesso à lista de programas disponíveis para a inscrição é feita a partir da página inicial da aplicação, conforme destacado na Figura 13 por um retângulo. O candidato não precisa estar logado no sistema para acessá-la. A lista exibe os programas, os níveis e os polos que estão com inscrições abertas no momento do acesso à página (vide Figura 14).

4.3.2 Autenticação no Sistema

Como pode ser visto no diagrama de casos de uso, Figura 7, o cadastro e a autenticação no sistema são feitos através do Sistema de Autenticação Centralizada (CAS), que permite a navegação entre os diversos sistemas da UFSC com apenas uma autenticação. Ao clicar nos links de acesso (vide destaque na Figura 15), o candidato é automaticamente transferido à página do CAS (vide Figura 16), onde pode se cadastrar e autenticar-se perante o sistema. Salienta-se que o CAS

Figura 13 – Acesso à lista de programas disponíveis em destaque.



Fonte: Elaborada pelo autor (2018).

Figura 14 – Lista de programas disponíveis para inscrição.

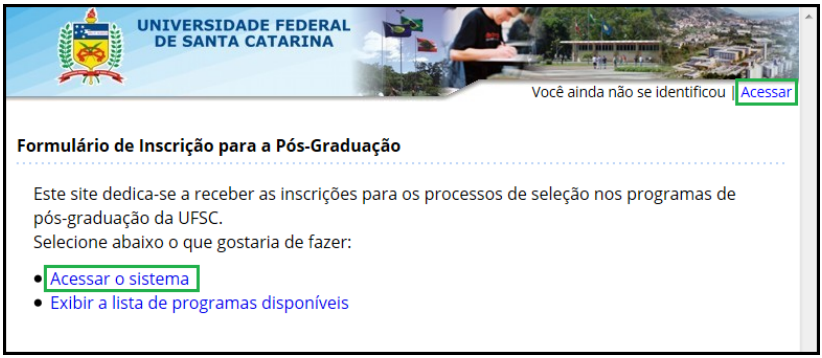
Programa	Nível	Polo
Programa de Pós-Graduação em Urbanismo, História e Arquitetura da Cidade	Pós-Doutorado	Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Ciência da Computação	Pós-Doutorado	Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Neurociências	Pós-Doutorado	Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Engenharia Elétrica	Pós-Doutorado	Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Engenharia Mecânica	Pós-Doutorado	Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Bioquímica	Pós-Doutorado	Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Ciências Médicas-Novo	Pós-Doutorado	Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Ecologia	Pós-Doutorado	Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Engenharia de Automação e Sistemas	Pós-Doutorado	Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Engenharia de Produção	Pós-Doutorado	Universidade Federal de Santa Catarina
Programa de Pós-Graduação em Odontologia	Pós-Doutorado	Universidade Federal de Santa Catarina

Fonte: Elaborada pelo autor (2018).

é um sistema já existente e utilizado por diversos outros sistemas da UFSC, e que o CAPG Inscrição atua apenas como cliente.

Após autenticado, o candidato é automaticamente transferido

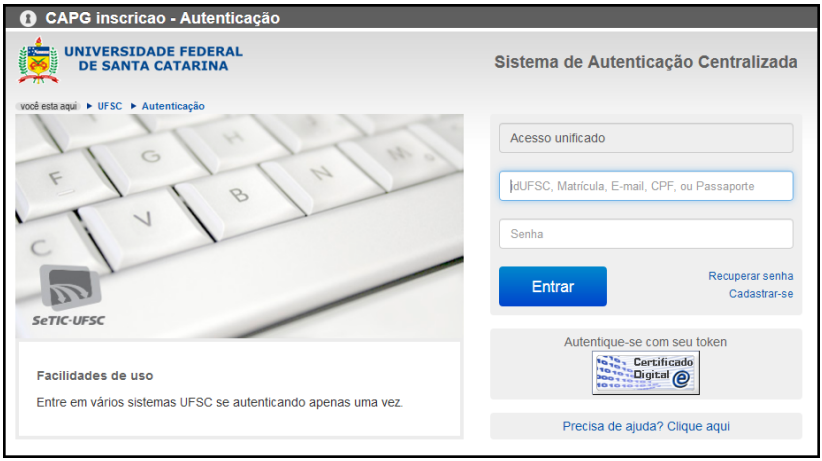
Figura 15 – Acesso ao sistema de inscrição do CAPG.



Fonte: Elaborada pelo autor (2018).

para a página de inscrições (vide Figura 17). Nesta, destaca-se o nome do usuário autenticado no sistema por um retângulo, e à sua direita o link para encerrar a sua sessão no sistema.

Figura 16 – Cadastro e autenticação no CAS.



Fonte: Elaborada pelo autor (2018).

Figura 17 – Lista de inscrições do candidato.

Status	Programa	Nível	Data de Inscrição	Editar	Excluir
Em andamento	Programa de Pós-Graduação em Aquicultura	Mestrado	-		
Em andamento	Programa de Pós-Graduação em Direito	Mestrado	-		
Finalizada	Programa de Pós-Graduação em Administração	Pós-Doutorado	22/11/2017	-	-
Finalizada	Programa de Pós-Graduação em Administração	Pós-Doutorado	21/11/2017	-	-
Finalizada	Programa de Pós-Graduação em Administração	Pós-Doutorado	21/11/2017	-	-
Finalizada	Programa de Pós-Graduação em Agroecossistemas	Pós-Doutorado	21/11/2017	-	-
Finalizada	Programa de Pós-Graduação em Administração	Pós-Doutorado	21/11/2017	-	-
Finalizada	Programa de Pós-Graduação em Ciência da Computação	Doutorado	29/11/2012	-	-
Finalizada	Programa de Pós-Graduação em Ciência da Computação	Doutorado	29/11/2012	-	-

Fonte: Elaborada pelo autor (2018).

4.3.3 Gerenciamento de Inscrições

O usuário autenticado no sistema tem acesso ao histórico de inscrições realizadas. Neste histórico, identificam-se o programa, o nível e a data de finalização da inscrição, como pode ser visto na tabela da Figura 17. Cada inscrição possui ainda um *status*:

- **Finalizada** – inscrição pertencente a um processo seletivo passado que não pode ser editada ou excluída.
- **Em andamento** – inscrição pertencente a um processo seletivo em vigência que pode ser editada e/ou excluída.

Ressalta-se que a tabela com o histórico de inscrições não é visível para candidatos que nunca realizaram uma inscrição em algum

dos processos seletivos para cursos de pós-graduação da UFSC. Para estes, apenas o link `Fazer uma nova inscrição`[3] estará disponível.

4.3.4 Criação de Uma Inscrição

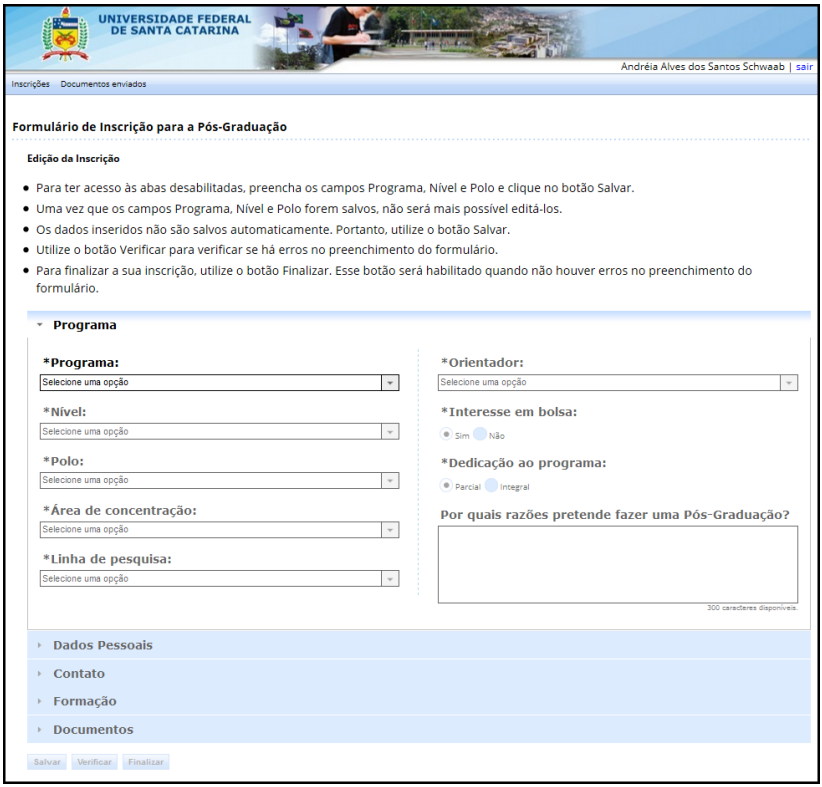
É possível criar uma nova inscrição a partir da tela de inscrições. Ao fazê-lo, o candidato se depara com uma tela tal qual a da Figura 18, na qual identificam-se alguns pontos:

- No *cabeçalho* da página encontram-se algumas mensagens informativas que têm o propósito de auxiliar o usuário sobre a utilização da aplicação.
- No *corpo* da página encontra-se um componente da biblioteca PrimeFaces chamado `AccordionPanel`[4], composto de diversas abas. Quando uma aba recebe um clique do mouse, o seu conteúdo é exibido progressivamente (daí a referência a um acordeão).
- No *rodapé* da página encontram-se os botões de ação `Salvar`, `Validar` e `Finalizar`. O botão `Salvar` é habilitado após o preenchimento do campo `Polo`, e salva as informações fornecidas pelo usuário na base de dados. Deve-se atentar para o fato de que as informações fornecidas pelos usuários não são salvas automaticamente, daí o uso do botão. A utilização dos demais botões será explicada na seção 4.3.7.

Com exceção da aba `Programa`, as demais estão desabilitadas no início do preenchimento de uma inscrição. Para habilitá-las, é preciso preencher os campos `Programa`, `Nível` e `Polo`, e clicar no botão `Salvar`. Após o salvamento, o candidato terá a opção de importar os dados da última inscrição por si realizada, caso exista (vide Figura 19). A partir deste ponto, os casos de uso *criar inscrição* e *editar inscrição* têm as mesmas funcionalidades.

[3] Link na parte inferior da tabela da Figura 17.
[4] Vide https://www.primefaces.org/showcase/ui/panel/accordionPanel.xhtml.

Figura 18 – Criação de uma nova inscrição.



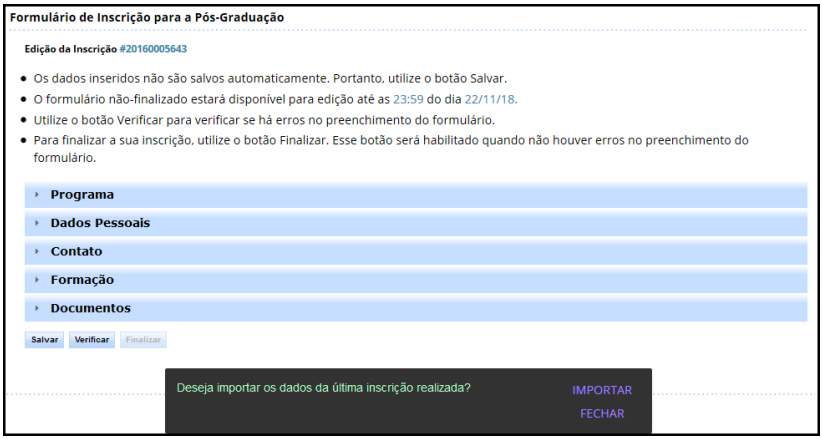
Fonte: Elaborada pelo autor (2018).

4.3.5 Edição de Uma Inscrição

A Figura 19 ilustra uma página típica de quando se inicia a edição de uma inscrição, com exceção da mensagem de importação de dados. Nota-se que, ao contrário da página inicial da criação de uma nova inscrição, o número da inscrição e a data limítrofe para a finalização da inscrição são exibidas no cabeçalho da página.

A Figura 20 exibe os campos da aba `Dados Pessoais` preenchidos com dados fictícios. É importante ressaltar que alguns dos campos

Figura 19 – Importação de dados na criação de uma nova inscrição.



Fonte: Elaborada pelo autor (2018).

dependem do preenchimento do campo `País de origem`:

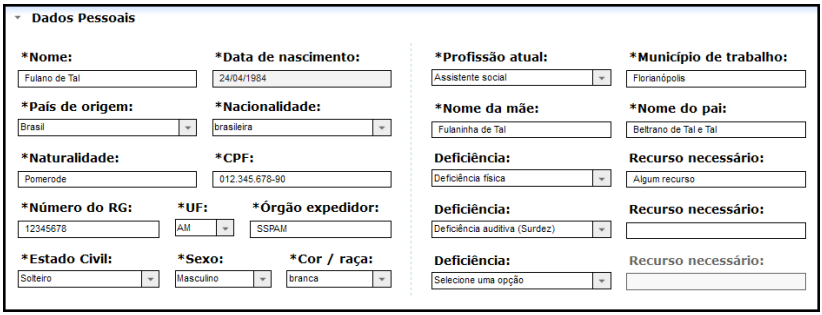
- Quando o país de origem selecionado for `Brasil`, os campos `Naturalidade` e `Município de trabalho` são capazes de completar automaticamente o valor inserido pelo usuário[5]. Para isso, requerem que o usuário insira ao menos três caracteres.

- Ao escolher um país estrangeiro, os campos `Naturalidade` e `Município de trabalho` tornam-se campos de entrada de texto normais, sem a funcionalidade de auto-completar a entrada fornecida. Ainda, os campos `Número do RG`, `UF` e `Órgão expedidor` são substituídos pelos campos `Número do passaporte` e `Validade do passaporte`, e o campo `CPF` deixa de ser obrigatório.

Os pares formados pelos campos `Deficiência` e `Recurso necessário` têm o seguinte funcionamento:

[5] Utiliza-se um componente PrimeFaces chamado `AutoComplete`.

Figura 20 – Campos da aba `Dados Pessoais`.



Fonte: Elaborada pelo autor (2018).

- Inicialmente, apenas o campo `Deficiência` do primeiro par está habilitado.

- Se o usuário selecionar um valor para um campo `Deficiência`, habilita-se o campo `Recurso necessário` do seu par e o campo `Deficiência` do par seguinte.

- Se o usuário limpar algum campo `Deficiência`, então o campo `Recurso necessário` do seu par e todos os pares seguintes serão desabilitados.

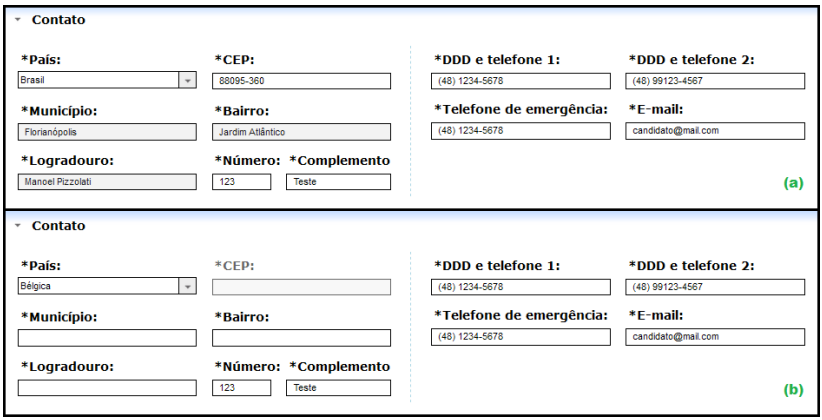
- Podem existir ao todo três pares.

O preenchimento da aba `Contato` depende do campo `País`. A Figura 21 exibe duas situações:

(a) Se o endereço for nacional, então os campos `Município`, `Bairro` e `Logradouro` serão automaticamente preenchidos quando o usuário preencher o campo `CEP` e pressionar a tecla `<TAB>` (ou clicar fora do campo).

(b) Se o endereço for internacional, então o campo `CEP` é desabilitado e o usuário deve preencher os campos `Município`, `Bairro` e `Logradouro`.

Figura 21 – Preenchimento da aba `Contato`. Em `(a)`, tem-se um endereço nacional. Em `(b)`, internacional.



Fonte: Elaborada pelo autor (2018).

Ainda, os campos relacionados a números de telefone são automaticamente formatados conforme o usuário os preenche.

Os candidatos podem adicionar cursos de formação já concluídos ou em andamento na aba `Formação`, através do link `Adicionar curso` (vide Figuras 22 e 23). Nesta última, se a caixa `Instituição nacional` estiver selecionada, então o campo `Instituição` completa automaticamente o valor inserido pelo usuário (a partir de três caracteres).

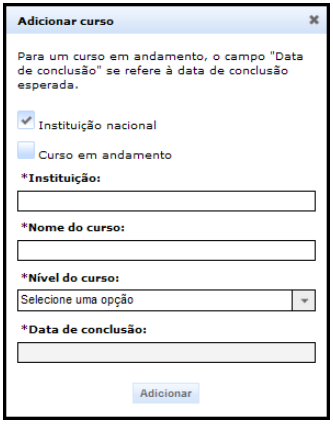
4.3.6 Envio de Documentos

Uma das principais funcionalidades providas por este sistema é permitir que o candidato faça o envio de documentos. Não somente se reduz a quantidade de papéis que ficariam armazenados nas secretarias dos cursos (economizando espaço), como também potencializa-se a

Figura 22 – Campos da aba `Formação`.

Fonte: Elaborada pelo autor (2018).

Figura 23 – Caixa de diálogo modal que permite a adição de um curso de formação.

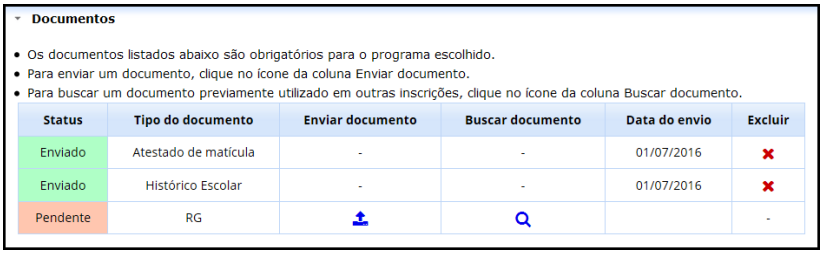


Fonte: Elaborada pelo autor (2018).

segurança dos documentos (documentos em papel estão sempre sujeitos a danos e avarias) e facilita-se a sua busca. Essa funcionalidade está implementada na aba `Documentos`, como pode ser visto na Figura 24. As linhas da tabela representam os documentos que são exigidos pela secretaria a qual o programa escolhido está vinculado. No exemplo

fictício mostrado, são exigidos três documentos, dois dos quais já foram enviados, e um está pendente.

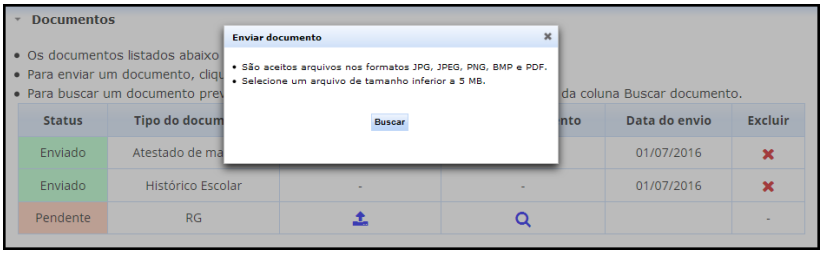
Figura 24 – Campos da aba `Documentos`.



Fonte: Elaborada pelo autor (2018).

A partir da tabela, pode-se *enviar* novos documentos e *buscar* documentos de inscrições passadas realizadas pelo candidato.

Figura 25 – Envio de documentos.



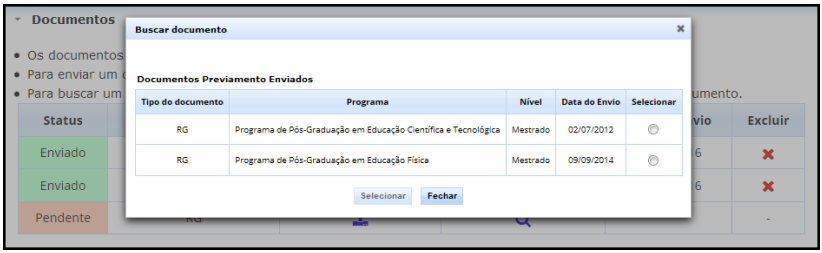
Fonte: Elaborada pelo autor (2018).

No envio de documentos (vide Figura 25), uma caixa de diálogo modal informa os formatos aceitos pela aplicação e o tamanho máximo de cada arquivo. Ao clicar no botão `Buscar`, abre-se uma janela de busca de documentos nativa ao sistema operacional do computador do candidato para que este possa escolher o documento a ser enviado. Os documentos

enviados são armazenados em um repositório e os identificadores dos arquivos são armazenados em um banco de dados (vide Figura 11).

Na busca de documentos (vide Figura 26), uma janela modal exibe uma lista de inscrições passadas nas quais esse mesmo tipo de documento foi utilizado. As colunas `Programa`, `Nível` e `Data do envio` auxiliam o candidato na escolha do documento a ser reutilizado na inscrição atual.

Figura 26 – Busca de documentos utilizados em inscrições passadas.



Fonte: Elaborada pelo autor (2018).

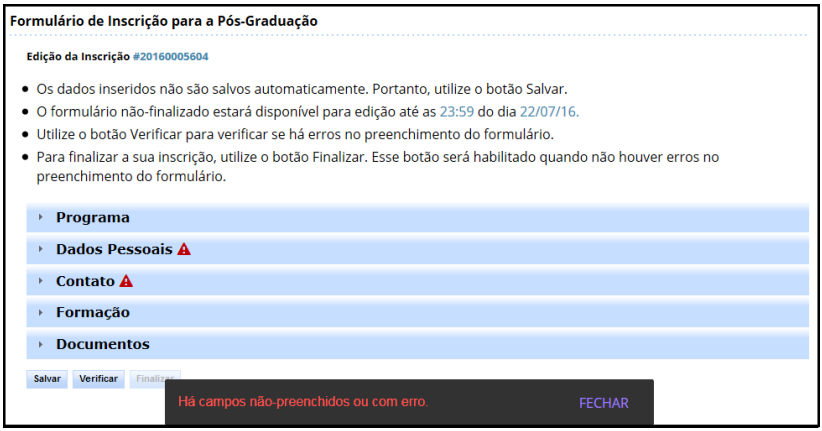
Uma das preocupações que podem ser levantadas em relação ao envio de documentos é a quantidade de conexões simultâneas que o servidor onde está instalado o repositório de documentos suporta. Segundo dados obtidos na própria SeTIC, são possíveis sete conexões simultâneas ao repositório. Até o momento, o sistema atual não teve problemas relacionados à falta de conexões disponíveis. Embora o sistema em desenvolvimento vise atender a uma gama maior de secretarias, os processo seletivos dos diferentes programas costumam ter cronogramas diferentes, diminuindo a concorrência por conexões ao repositório.

4.3.7 Verificação de Erros e Finalização de Uma Inscrição

Pode-se verificar a existência de erros e o não-preenchimento de campos obrigatórios em qualquer momento durante o preenchimento do formulário por meio do botão `Verificar` (salvo a situação em que o usuário está criando uma nova inscrição e ainda não salvou o formulário).

Na presença de tais erros, uma mensagem é exibida para o usuário e as abas que possuem campos não-preenchidos ou com erros são destacadas com um ícone de exclamação, como pode ser visualizado na Figura 27 para as abas `Dados Pessoais` e `Contato`.

Figura 27 – Erros encontrados nas abas `Dados Pessoais` e `Contato`.

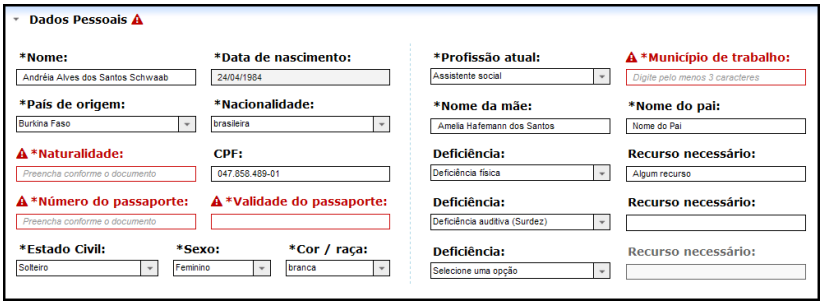


Fonte: Elaborada pelo autor (2018).

Ao abrir tais abas, os campos não-validados estão destacados na cor vermelha e com um ícone de exclamação ao lado de seus rótulos (vide Figura 28). Neste momento, tendo o usuário corrigido os erros ou não, se o formulário for salvo, os ícones de erro desaparecem.

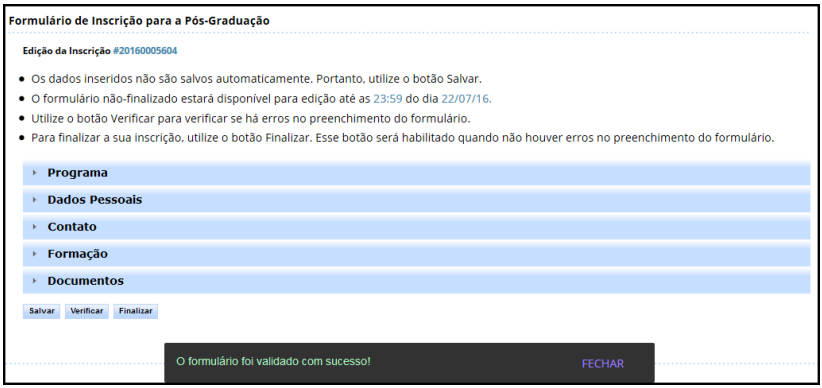
Quando o formulário for verificado e não houver mais erros, uma mensagem de sucesso é exibida (vide Figura 29), e o botão `Finalizar` é habilitado. Ao clicar no botão `Finalizar`, disponibiliza-se um comprovante de inscrição em formato PDF e a edição da inscrição deixa de ser possível a partir deste momento. O documento é automaticamente descarregado no navegador e também enviado para o e-mail cadastrado pelo usuário no preenchimento do formulário. Um exemplo de comprovante é mostrado no Anexo E.

Figura 28 – Campos não-preenchidos da aba `Dados Pessoais`.



Fonte: Elaborada pelo autor (2018).

Figura 29 – Formulário verificado e sem erros.



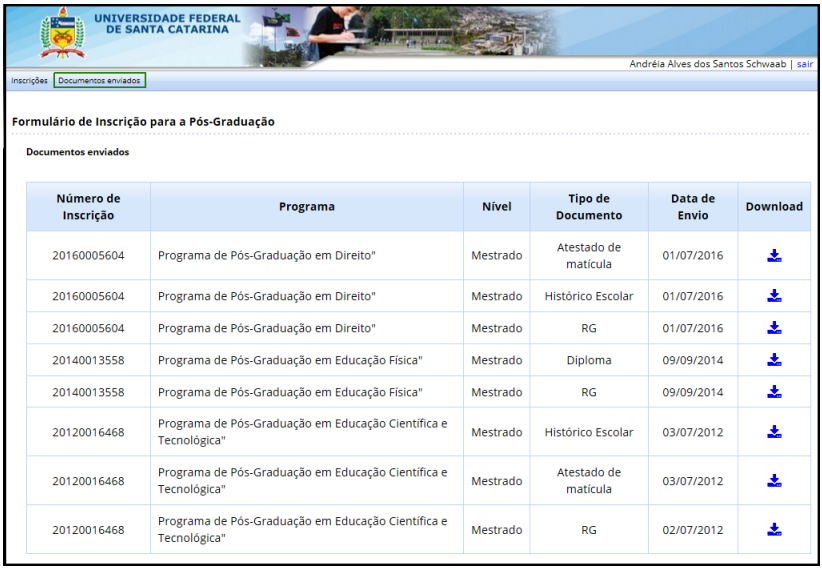
Fonte: Elaborada pelo autor (2018).

4.3.8 Documentos Enviados

Todos os documentos enviados pelo candidato podem ser obtidos a partir da *view documentos enviados*, a qual pode ser acessada através do item de menu `Documentos enviados`. O usuário pode baixar os documentos desejados clicando no ícone de *download* da linha respectiva (vide Figura 30). Informações tais como número de inscrição, programa,

nível, tipo de documento e data de envio também estão presentes para auxiliar o usuário a encontrar o documento desejado.

Figura 30 – Lista de documentos enviados pelo candidato.



Fonte: Elaborada pelo autor (2018).

5 CONCLUSÕES

As maneiras de inscrição nos cursos de Pós-Graduação da UFSC atualmente são variadas. Grande parte dos programas utiliza um sistema informatizado oficial, que não atende integralmente os requisitos existentes. Alguns cursos utilizam seus próprios sistemas, dificultando a sua manutenibilidade e introduzindo duplicações de código e vulnerabilidades.

Torna-se relevante, portanto, o desenvolvimento de um novo sistema de inscrição unificado, que dê suporte a novas funcionalidades e que utilize as tecnologias atualmente em uso na SeTIC.

Ao longo deste projeto, diversas atividades previstas foram desenvolvidas para que o objetivo fosse atingido, especificamente:

- Levantamento das funcionalidades desejadas para o sistema através de entrevistas com o cliente (PROPG);
- Especificação dos requisitos funcionais e não-funcionais do sistema a partir destas funcionalidades;
- Definição da arquitetura do sistema a partir de diagramas feitos segundo as especificações da UML: diagrama de casos de uso, diagrama de visão geral de interação e diagramas de classes;
- Prototipação a partir do desenvolvimento de um conjunto de *wireframes*.
- Adequação do Projeto Base para atender a estrutura definida;
- Implementação dos casos de uso discutidos na Seção 4.3.
- Adaptação da interface do sistema para sua utilização em dispositivos móveis de forma responsiva.

Como resultado, obteve-se um novo sistema de inscrição que atende às necessidades dos diversos Programas de Pós-Graduação da UFSC. O aumento no grau de adesão destes programas permitirá um

aumento na eficiência administrativa através da melhoria da gestão dos dados dos processos de inscrição por parte tanto dos programas quanto da PROPG.

Embora este projeto não seja classificado como um projeto de pesquisa, comum aos trabalhos de conclusão de curso em Ciências da Computação, seu mérito reside na aplicação de conhecimentos obtidos ao longo do curso para solucionar uma demanda real da própria universidade.

O sistema desenvolvido está atualmente no estágio de homologação. A identificação de *bugs*, as aprimorações e a adição de novas funcionalidades serão feitas pela SeTIC em colaboração com a Pró-Reitoria de Pós-Graduação, bem como as secretarias dos cursos de Pós-Graduação.

REFERÊNCIAS

ALMEIDA, A. *Entenda os MVCs e os frameworks Action e Component Based*. 2012. [Acesso em 25/04/2017]. Disponível em: <http://blog.caelum.com.br/entenda-os-mvcs-e-os-frameworks-action-e-component-based/>. Citado 2 vezes nas páginas 16 e 17.

BUSCHMANN, F. et al. *Pattern-Oriented Software Architecture: A System Of Patterns*. Chichester, Inglaterra: John Wiley & Sons Ltd, 1996. Citado na página 15.

HUNTER, J.; CRAWFORD, W. *Java Servlet Programming*. Sebastopol, California: O'Reilly & Associates, Inc., 1998. Citado na página 59.

JENDROCK, E. et al. *Java Platform, Enterprise Edition: The Java EE Tutorial*. 2014. [Acesso em 22/02/2017]. Disponível em: <https://docs.oracle.com/javaee/7/tutorial/>. Citado 5 vezes nas páginas 11, 12, 14, 18 e 19.

KEITH, M.; SCHINCARIOL, M. *Pro JPA 2*: A definitive guide to mastering the java persistence api. Second. Chichester, Inglaterra: Apress, 2013. Citado 7 vezes nas páginas 21, 61, 62, 63, 64, 65 e 66.

LHOTKA, R. *Should all apps be n-tier?* 2005. [Acesso em 23/02/2017]. Disponível em: <http://www.lhotka.net/weblog/ShouldAllAppsBeNtier.aspx>. Citado na página 11.

PRIMEFACES. *PrimeFaces Showcase*. 2017. [Acesso em 25/04/2017]. Disponível em: <https://www.primefaces.org/showcase/>. Citado na página 19.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. São Paulo: Pearson Prentice Hall, 2011. Citado 2 vezes nas páginas 23 e 27.

ANEXO A – SERVLETS

Servlets são classes Java que possuem métodos adequados a receber requisições e construir respostas às tais requisições dinamicamente. O trecho de código do Algoritmo 4 representa um `HttpServlet` que atende a requisições HTTP do tipo GET através da sobrescrita do método `doGet()`. Os parâmetros `HttpServletRequest` e `HttpServletResponse` representam, respectivamente, a requisição feita pelo cliente e a resposta do servidor.

Algoritmo 4 – Um *servlet* que imprime "*Hello World*".

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
  public void doGet(HttpServletRequest req,
                    HttpServletResponse res)
      throws ServletException, IOException {
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<html>");
    out.println("<head><title>Hello World</title></head>");
    out.println("<body>");
    out.println("<big>Hello World</big>");
    out.println("</body></html>");
  }
}
```

Fonte: adaptado de Hunter e Crawford (1998)

Um *servlet* HTTP pode, além do método `doGet()`, sobreescrever outros métodos, tais como `doPut()`, `doDelete()`, etc, que representam requisições HTTP do tipo PUT, DELETE, etc, respectivamente.

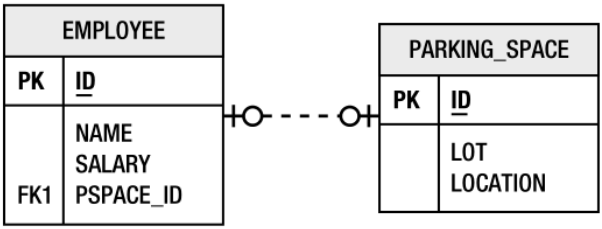
ANEXO B – RELACIONAMENTOS JPA

B.1 RELACIONAMENTO `@ONETOONE`

Apenas uma instância da entidade fonte[1] pode referenciar a mesma instância da entidade alvo. Em termos do banco de dados, isso implica na existência de uma restrição de unicidade na coluna da chave estrangeira da tabela fonte. Ainda, o JPA espera que o nome dessa coluna seja igual ao nome do atributo da entidade fonte seguido de um *underscore* e do nome da coluna da chave primária da tabela alvo. Caso esse não seja o caso, então deve-se fornecer o nome da coluna na anotação `@JoinColumn`.

Para ilustrar essa situação, considere as tabelas ilustradas na Figura 31. O mapeamento feito pelo JPA espera que a coluna com a chave estrangeira a ser utilizada seja chamada de `PARKINGSPACE_ID`, porém essa coluna chama-se `PSPACE_ID`. Neste caso, utiliza-se a anotação `@JoinColumn(name = "PSPACE_ID")`, conforme Algoritmo 5.

Figura 31 – Tabelas EMPLOYEE e PARKING_SPACE.



Fonte: Retirado de Keith e Schincariol (2013).

É interessante notar que o campo `name` não possui nenhuma anotação relacionada à sua persistência. Isso se deve ao fato de existir uma coluna de mesmo nome na tabela `EMPLOYEE`.

Algoritmo 5 – Classe Employee e seus relacionamentos.

[1] A entidade na qual se está fazendo a anotação.

```
@Entity
public class Employee {

  @Id private long id;

  @OneToOne
  @JoinColumn(name = "PSPACE_ID")
  private ParkingSpace parkingSpace;
  // ...
}
```

Fonte: Adaptado de Keith e Schincariol (2013).

Quando a entidade alvo do relacionamento também possuir uma referência à entidade fonte, tem-se então um relacionamento bidirecional, em que uma das entidades é dita ser a *dona* do relacionamento por conter a coluna com a chave estrangeira da outra entidade[2]. Nessa situação, a anotação `@JoinColumn` da entidade que *não* é a dona do relacionamento recebe o elemento `mappedBy = "X"`, onde X refere-se ao nome do campo da entidade que é dona do relacionamento, como pode ser visto no Algoritmo 6.

Algoritmo 6 – Classe ParkingSpace e seus relacionamentos.

```
@Entity
public class ParkingSpace {

  @Id private long id;
  private String location;

  @OneToOne(mappedBy = "parkingSpace")
  private Employee employee;
  // ...
}
```

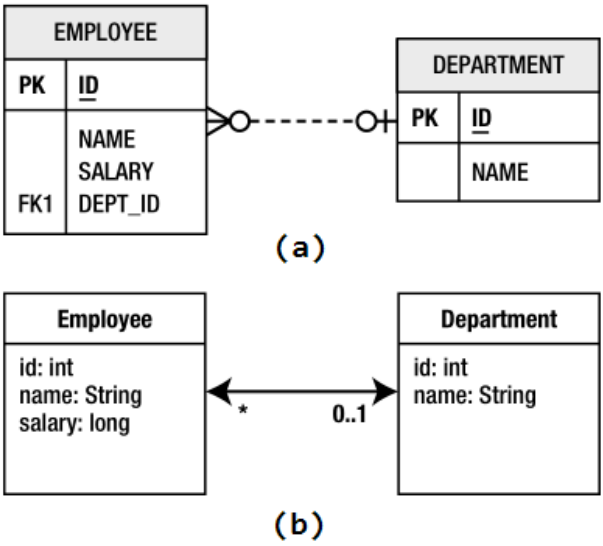
Fonte: Adaptado de Keith e Schincariol (2013).

[2] A escolha de qual das tabelas deverá conter a chave estrangeira da outra tabela é uma decisão de modelagem do banco de dados.

B.2 RELACIONAMENTO `@ONETOMANY`

Este tipo de relacionamento ocorre quando uma entidade está associada a uma coleção de entidades. Considere as tabelas `EMPLOYEE` e `DEPARTMENT` ilustradas na Figura 32. O relacionamento *um para muitos* feito em UML pode ser visto em (b).

Figura 32 – Tabelas `EMPLOYEE` e `DEPARTMENT` em (a) e seu relacionamento em (b).



Fonte: Adaptado de Keith e Schincariol (2013).

É importante notar que esse relacionamento implica em um relacionamento *muitos para um* quando analisado a partir da tabela `EMPLOYEE` e, portanto, é um relacionamento naturalmente bidirecional. Daí que uma das entidades fará o papel de dona do relacionamento, conforme discutido anteriormente no caso do relacionamento *um para um* bidirecional. Devido ao fato de não existir uma maneira escalonável de guardar inúmeras chaves em uma única linha de uma tabela, a entidade dona do relacionamento nessa situação deve ser a que estiver

no lado "muitos". Nesse caso, a entidade dona é, portanto, a `EMPLOYEE` e nela estarão as chaves estrangeiras para as linhas da tabela `DEPARTMENT`. Por sua vez, isso implica na adição do elemento `mappedBy` na anotação `@JoinColumn` dessa última entidade, conforme Algoritmo 14.

Algoritmo 7 – Classe Department e seus relacionamentos.

```
@Entity
  public class Department {

  @Id
  private long id;
  private String name;

  @OneToMany(mappedBy = "department")
  private Collection<Employee> employees;
  // ...
}
```

Fonte: Adaptado de Keith e Schincariol (2013).

Mais uma vez, devido à utilização de um nome de coluna diferente do valor padrão esperado para esse relacionamento, utiliza-se o elemento `name` na anotação `@JoinColumn` da entidade Employee (vide Algoritmo 8).

Algoritmo 8 – Classe Employee e seus relacionamentos.

```
@Entity
public class Employee {

  @Id
  private long id;

  @ManyToOne
  @JoinColumn(name = "DEPT_ID")
  private Department department;
  // ...
}
```

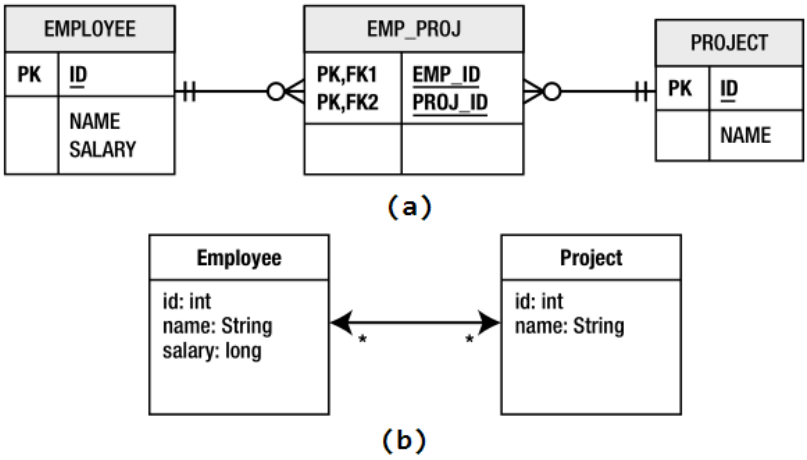
Fonte: Adaptado de Keith e Schincariol (2013).

B.3 RELACIONAMENTO `@MANYTOMANY`

O último tipo de relacionamento entre entidades apresentado é chamado de *muitos para muitos*. Esse relacionamento ocorre quando uma ou mais entidades fonte estão associadas a uma coleção de entidades alvo, e estas entidades alvo estão associadas a uma coleção de entidades fonte. A representação dessa situação em UML pode ser vista no item (b) da Figura 33 para as entidades Employee e Project.

Considere agora as tabelas ilustradas no item (a) da Figura 33. Uma linha de `EMPLOYEE` pode estar associada a diversas linhas de `PROJECT` e vice-versa. A existência da tabela `EMP_PROJ` justifica-se por não ser escalável guardar as chaves estrangeiras em cada linha de *ambas* as tabelas. Nesse caso, `EMP_PROJ` possui apenas duas colunas e estas guardam as chaves estrangeiras de cada uma das tabelas. Portanto, `EMPLOYEE` não possui uma coluna com as chaves estrageiras de `PROJECT` e vice-versa (KEITH; SCHINCARIOL, 2013).

Figura 33 – Tabelas `EMPLOYEE`, `EMP_PROJ` e `PROJECT` em (a) e seu relacionamento em (b).



Fonte: Adaptado de Keith e Schincariol (2013).

Se ambas as tabelas `EMPLOYEE` e `PROJECT` não possuem uma coluna de chaves estrangeiras, quem é a entidade dona do relacionamento? A resposta é: tanto faz, desde que essa decisão seja indicada ao JPA. As entidades Employee e Project resultantes do mapeamento podem ser vistas no Algoritmo 9. Evidencia-se o uso da anotação `joinColumns`, indicando a entidade dona do relacionamento.

Algoritmo 9 – Classes Employee e Project.

```
@Entity
public class Employee {
  @Id
  private long id;
  private String name;
  @ManyToMany
  @JoinTable(name="EMP_PROJ",
      joinColumns=@JoinColumn(name="EMP_ID"),
      inverseJoinColumns=@JoinColumn(name="PROJ_ID"))
  private Collection<Project> projects;
  // ...
}

@Entity
public class Project {
  @Id
  private long id;
  private String name;
  @ManyToMany(mappedBy="projects")
  private Collection<Employee> employees;
  // ...
}
```

Fonte: Adaptado de Keith e Schincariol (2013).

ANEXO C – RUP: MELHORES PRÁTICAS

Com o intuito de minimizar as falhas e aumentar a produtividade no desenvolvimento de software, seis idéias, conhecidas por *seis melhores práticas*, são descritas no RUP. São chamadas de *melhores* práticas por serem comumente observadas em organizações bem sucedidas.

1. **Desenvolver software iterativamente** — Identificar o problema por completo, desenvolver uma solução, construir o software e testar o produto, é uma abordagem muito arriscada de se seguir de forma sequencial, dados os sistemas de software complexos dos dias de hoje. A adoção de uma abordagem iterativa permite um entendimento do problema de forma gradativa através de múltiplas iterações. Como cada iteração resulta em um *release* executável, o time de desenvolvimento permanece sempre focado em alcançar as metas de cada iteração, além de possibilitar um melhor controle do cronograma das atividades. Ainda, os requisitos de maior risco são abordados no início de cada estágio, reduzindo, assim, o perfil de risco do projeto.

2. **Gerenciar requisitos** — O RUP descreve como elicitar, organizar e documentar as funcionalidades e restrições do sistema, bem como fazer o monitoramento das decisões adotadas ao longo do desenvolvimento do projeto. Além disso, a utilização de *casos de uso* como forma de capturar os requisitos funcionais e direcionar o design, a implementação e o teste de software, aumenta a probabilidade de que o sistema irá satisfazer as necessidades do usuário final.

3. **Usar arquiteturas baseadas em componentes** — O desenvolvimento de sistemas altamente complexos de forma monolítica aumenta substancialmente os riscos envolvidos no projeto. A utilização de módulos ou subsistemas que satisfazem funções específicas permite o reuso de código e a realização de testes antes da sua integração.

4. **Modelar o software visualmente** — A utilização de diagramas para a modelagem de atores, componentes e suas interações permite a visualização da estrutura e comportamento dos diversos componentes do sistema de forma abstrata. Segundo o RUP, a UML é considerada o alicerce para a modelagem visual.

5. **Verificar a qualidade do software** — A aceitação das aplicações de software é drasticamente inibida por baixo desempenho e confiabilidade. A avaliação da qualidade do software é feita através de testes de funcionalidade e desempenho da aplicação e do sistema, com base nos requisitos especificados.

6. **Controlar as mudanças feitas no software** — A utilização de uma sistema de controle de mudanças é indispensável em um ambiente em que mudanças são inevitáveis. A identificação de quando uma mudança ocorreu e quem a realizou é muito útil para a sincronização e estabelecimento de áreas de trabalho seguras para os desenvolvedores.

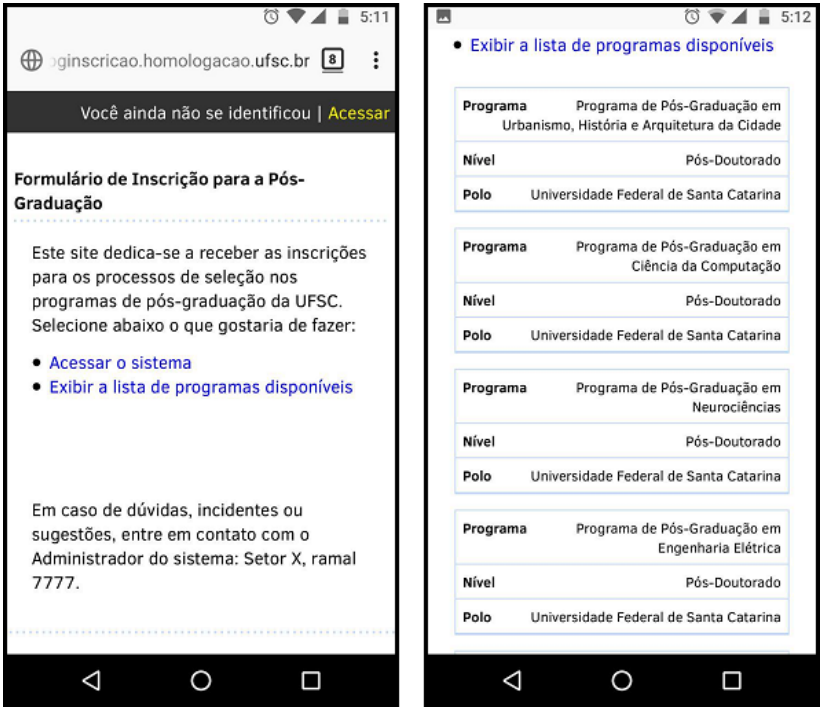
Uma *stored procedure* é utilizada no salvamento de uma inscrição na base de dados Centura. Para o sucesso de sua execução, faz-se necessário o envio de *todos* os campos do formulário de preenchimento obrigatório, tornando o desenvolvimento do software impraticável de ser realizado de forma iterativa. Não obstante, as partes do sistema não relacionadas ao preenchimento ou edição de um formulário foram desenvolvidas iterativamente.

Embora o desenvolvimento guiado por testes[1] tenha sido considerado, a sua adoção não foi feita devido à inexeperiência do autor com esse processo de desenvolvimento. A verificação da qualidade do software, através de testes de funcionalidade, integração e desempenho, deverá ser realizada pela SeTIC.

[1] Processo de desenvolvimento de software em que as funcionalidades são adicionadas ao software somente após terem sido aprovadas em casos de teste que as validem.

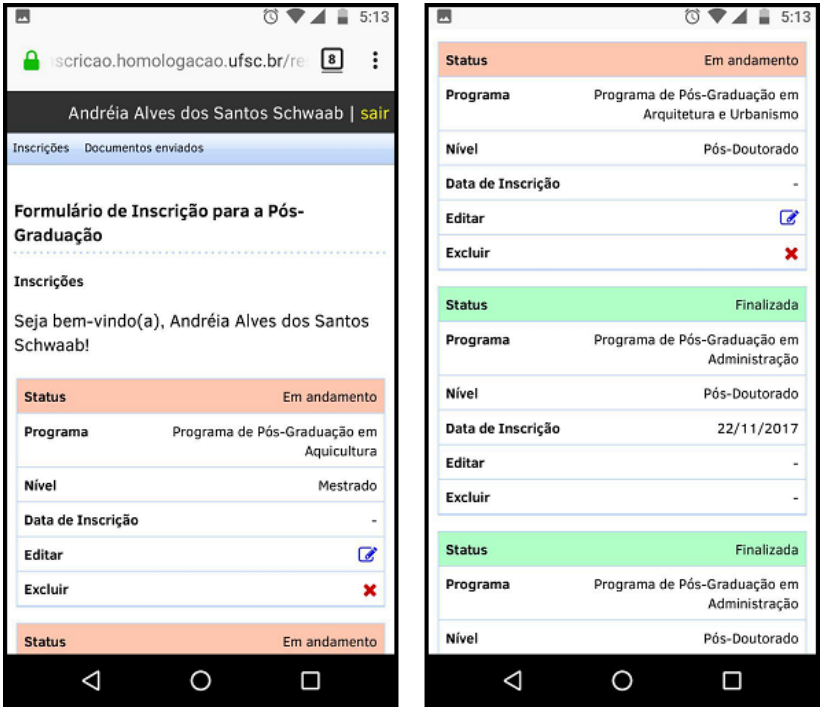
ANEXO D – EXEMPLIFICAÇÃO PARA DISPOSITIVOS MÓVEIS

Figura 34 – Página inicial do sistema (à esquerda) e lista de programas disponíveis (à direita).



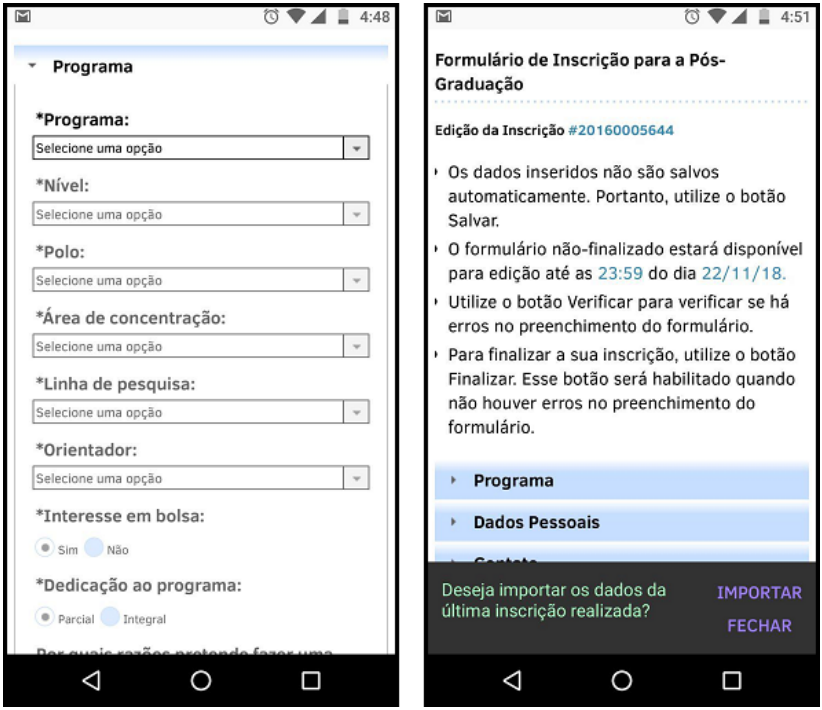
Fonte: Elaborada pelo autor (2018).

Figura 35 – *View* inscrições após a autenticação (à esquerda) e detalhe para os dois tipos de inscrição (à direita).



Fonte: Elaborada pelo autor (2018).

Figura 36 – Criação de uma nova inscrição (à esquerda) e mensagem de importação de dados (à direita).

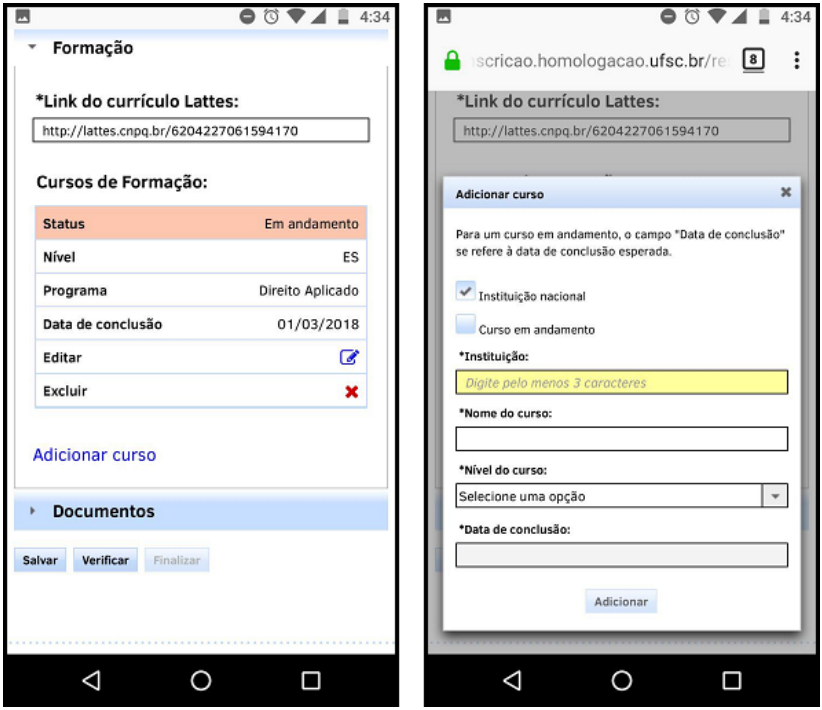


Fonte: Elaborada pelo autor (2018).

Figura 37 – Campos da aba `Dados Pessoais` (à esquerda) e campos da aba `Contato` (à direita).

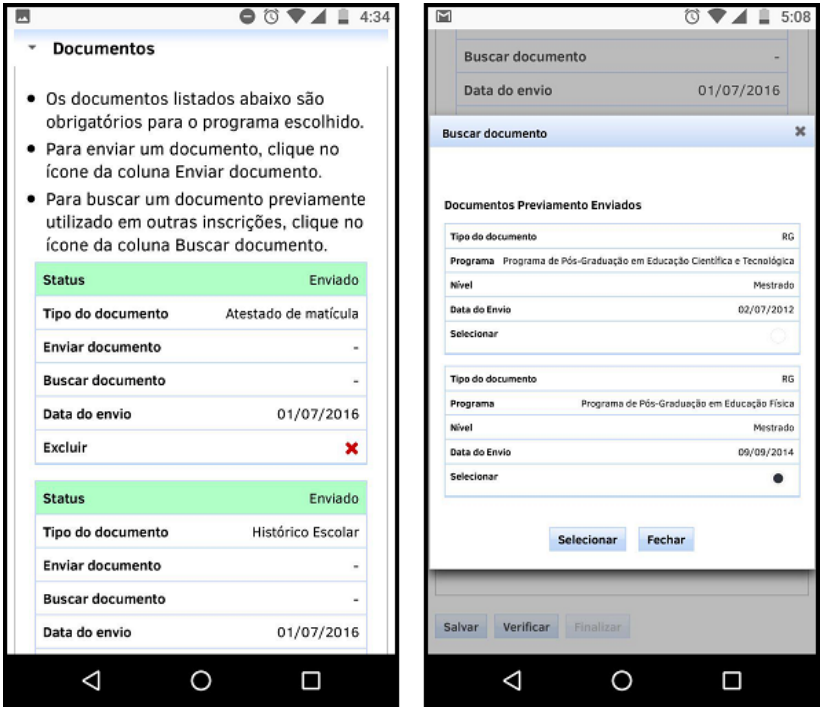
Fonte: Elaborada pelo autor (2018).

Figura 38 – Aba `Formação` (à esquerda) e inclusão de um curso de formação (à direita).



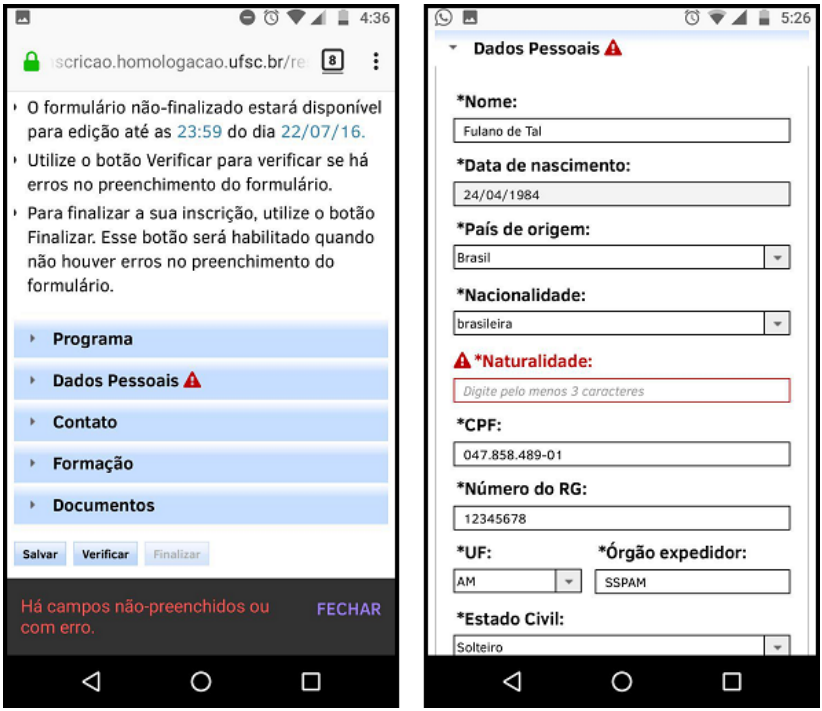
Fonte: Elaborada pelo autor (2018).

Figura 39 – Mensagens informativas da aba `Documentos` (à esquerda) e reutilização de documentos enviados (à direita).



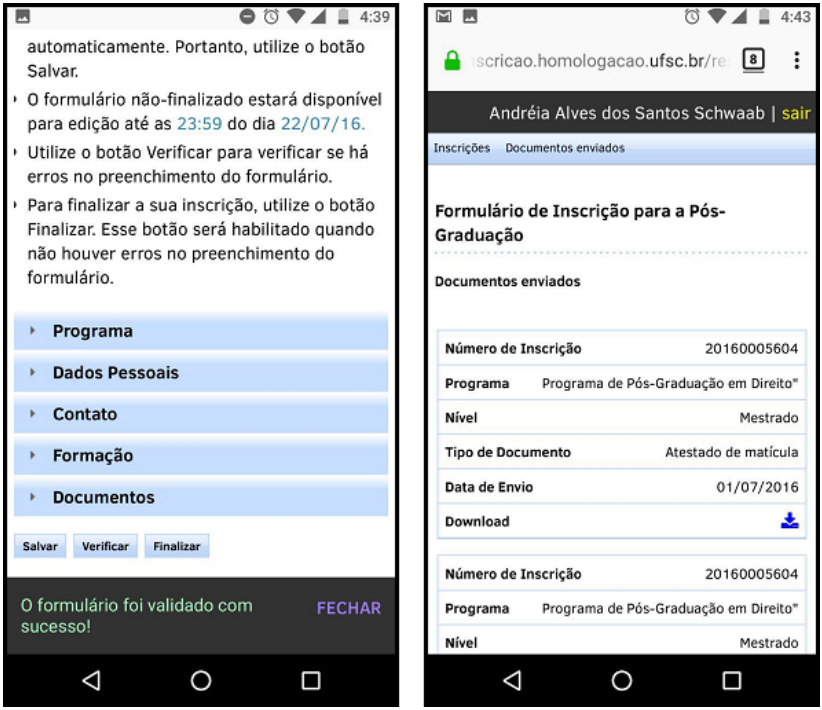
Fonte: Elaborada pelo autor (2018).

Figura 40 – Erros encontrados no preenchimento do formulário.



Fonte: Elaborada pelo autor (2018).

Figura 41 – Formulário verificado e sem erros (à esquerda) e tela de documentos enviados (à direita).



Fonte: Elaborada pelo autor (2018).

ANEXO E – EXEMPLO DE COMPROVANTE DE INSCRIÇÃO

UNIVERSIDADE FEDERAL
DE SANTA CATARINA
Pró-Reitoria de Pós-Graduação

COMPROVANTE DE INSCRIÇÃO

Atesta-se, segundo consta em nossos arquivos, que Fulano de Tal realizou inscrição no processo seletivo do Programa de Pós-Graduação em Direito, para o nível Mestrado, definido no edital de 01/2017, sob o número 20160005604.

Florianópolis, 24 de Maio de 2018.

A autenticidade deste documento pode ser verificada no endereço "http://autenticidade.ufsc.br" informando o seguinte código: CAPG-5604-c5ul-nu9w-zqe6

SeTIC - Superintendência de Governança Eletrônica e Tecnologia da Informação e Comunicação

ANEXO F – CÓDIGO-FONTE

Algoritmo 10 – Obtenção dos programas disponíveis.

```
@Override
public List<ProgramaAux> obterProgramasDisponiveisNaData(
    Date data) {
  StringBuffer query = new StringBuffer();
  query.append("SELECT new br.ufsc.setic.capginscricao.
    controller.capg.ProgramaAux(");
  query.append("ps.curso.nome, ps.nivel.descricao, ps.polo.
    nome) ");
  query.append("FROM ProcessoSeletivo ps ");
  query.append("WHERE :data BETWEEN ps.dtInicioProcesso AND
    ps.dtTerminoProcesso ");
  query.append("ORDER BY ps.curso");

  return em.createQuery(query.toString(), ProgramaAux.class)
          .setParameter("data", data)
          .getResultList();
}
```

Algoritmo 11 – Criação, atualização e remoção de entidades.

```
@Override
@Transactional
public <T> void persist(T obj) {
  em.persist(obj);
}

@Override
@Transactional
public <T> T merge(T obj) {
  return em.merge(obj);
}

@Override
@Transactional
public <T> void remove(T obj) {
  em.remove(obj);
}
```

Algoritmo 12 – Obtenção de um Programa de Pós-Graduação.

```
@Override
public Curso obterCurso(Long codigoCurso, String codigoNivel
    ) {
  StringBuffer query = new StringBuffer();
  query.append("SELECT c ");
  query.append("FROM Curso c ");
  query.append("WHERE c.id.codigo = :codigoCurso ");
  query.append("AND c.id.nivel = :codigoNivel ");

  return em.createQuery(query.toString(), Curso.class)
           .setParameter("codigoCurso", codigoCurso)
           .setParameter("codigoNivel", codigoNivel)
           .getSingleResult();
}
```

Algoritmo 13 – Obtenção do próximo número de inscrição disponível.

```
@Transactional
private Long obterNumeroInscricao(final int ano, final Long
    codigoCurso) {
  Map<String, Object> retorno = null;
  Long numeroInscricao = null;
  CenturaJdbcTemplate centuraJdbc = DbUtils.
    newCenturaJdbcTemplateNovaConn();
  SimpleJdbcCall call = new SimpleJdbcCall(centuraJdbc)
    .withCatalogName("capg")
    .withSchemaName("dbo")
    .withProcedureName("sp_inscricaoSequenceNext")
    .withReturnValue()
    .declareParameters(
      new SqlParameter("nu_ano_isq", Types.SMALLINT),
      new SqlParameter("cd_curso_isq", Types.NUMERIC),
      new SqlOutParameter("nu_lastkey_isq", Types.NUMERIC));
  retorno = (Map<String, Object>) call.execute(ano,
    codigoCurso);
  numeroInscricao = ((BigDecimal) retorno.get("
    nu_lastkey_isq")).longValue();
  return numeroInscricao;
}
```

Algoritmo 14 – *Download* de um arquivo.

```
public void baixarArquivoById(Long idArquivo) throws
    IOException {
  ArquivoDTO arquivo = storageService.getArquivoById(
    idArquivo);
  ExternalContext context = FacesContext.getCurrentInstance
    ().getExternalContext();
  InputStream inStream = storageService.getArquivoStreamById
    (idArquivo);
  OutputStream outStream = context.getResponseOutputStream()
    ;

  context.responseReset();
  context.setResponseContentType(arquivo.getContentType());
  context.setResponseContentLength(arquivo.getTamanho().
    intValue());
  context.setResponseHeader("Content-Disposition",
    "attachment; filename=\"" + arquivo.getNome() +"\"");

  int i;
  while ((i = inStream.read()) != -1) {
      outStream.write(i);
  }
  outStream.flush();
  outStream.close();
  FacesContext.getCurrentInstance().responseComplete();
}
```

ANEXO G – ARTIGO SBC

Atualização Tecnológica do Formulário de Inscrição do Processo Seletivo da Pós-Graduação da UFSC

Makhles Reuter Lange[1]

[1] Instituto de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
Florianópolis, SC – Brazil

m.r.lange@grad.ufsc.br

Abstract. *JavaEE Web development technologies were used in the development of a new online Postgraduate Enrollment System for the Federal University of Santa Catarina. The current enrollment system, which lacks some important sought out functionalities, was used as an inspiration in the development of the new one. Candidates can now fill out the application forms at any time during the selection process, upload the required documents and import personal data used in the last submitted application. Responsive web design was taken into consideration while designing the system,* i.e.*, mobile users will not feel left behind when filling out the application form. Lastly, an up-to-date set of libraries were used instead of the old ones, providing a more resourceful environment to developers.*

Resumo. *Utilizando-se a linguagem Java e as tecnologias para desenvolvimento Web próprias do JavaEE, desenvolveu-se um sistema de inscrição para os Programas de Pós-Graduação da UFSC. A produção deste trabalho foi inspirada em um sistema atualmente em uso que está carente de algumas funcionalidades. O novo sistema permite o prenchimento do formulário durante todo o período definido no cronograma do processo seletivo, além do envio de documentos obrigatórios pré-estabelecidos pelas secretarias dos cursos e da importação dos dados de inscrições passadas. A interface do sistema foi desenvolvida de forma responsiva,* i.e.*, a usabilidade em dispositivos móveis também foi levada em consideração no desenvolvimento das páginas. Ainda, empregou-se um conjunto de bibliotecas mais atual do que o do sistema em uso, proporcionando mais recursos ao desenvolvedor.*

1. Contextualização

O acesso aos programas de pós-graduação oferecidos pela Universidade Federal de Santa Catarina é feito através de processos seletivos lançados por suas respectivas secretarias. Todos os programas possuem algum modo para realizar as inscrições em seus processos seletivos. Dentre as formas utilizadas, destacam-se um sistema informatizado oficial, formulários de inscrição manuais e sistemas informatizados não-oficiais. O uso desses diversos sistemas, com suas funcionalidades e interfaces distintas, acarreta em diversos problemas, tais como:

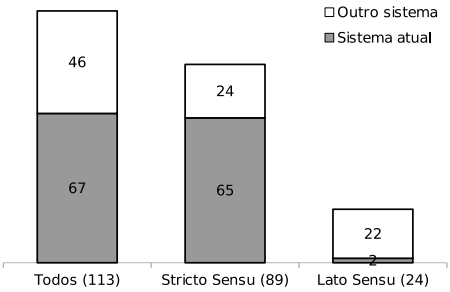
- Falta de padronização dos requisitos e tecnologias utilizadas.

- Autenticação de forma não-centralizada: além de ser uma vulnerabilidade em termos de segurança, implica em duplicação de código.
- Difícil manutenibilidade;
- Difícil obtenção de estatísticas relacionadas aos candidatos.

Em meados de 2010, esperava-se mais de cinco mil inscrições no Programa de Especialização em Saúde da Família. Visando atender essas inscrições, a solução adotada pela Pró-Reitoria de Pós-Gradução (PROPG), em parceria com a Superintendência de Governança Eletrônica e Tecnologia da Informação e Comunicação (SeTIC), foi a criação de um sistema informatizado[1]. Posteriormente, esse sistema foi adotado por outros programas.

A Figura 1 mostra o uso desse sistema pelos programas da UFSC em 2017. Dos 113 programas, apenas 67 (59%) adotaram o sistema para realizar as inscrições. A figura também mostra o uso de acordo com o tipo do programa (*stricto sensu* ou *lato sensu*).

Figura 1. Utilização dos sistemas de inscrição em 2017. Fonte: consulta do banco de dados do CAPG.



Não obstante, esse sistema não foi desenvolvido visando a integração dos sistemas existentes e está carente de algumas funcionalidades frequentemente solicitadas pelos programas de pós-graduação. A implementação de tais funcionalidades no sistema atual é custosa, pois sua estrutura contempla os requisitos definidos na época de sua elaboração. Adicionalmente, as bibliotecas e tecnologias utilizadas são antigas (algumas estão, inclusive, descontinuadas), dificultando seu desenvolvimento e sua manutenibilidade. Justifica-se, portanto, o desenvolvimento de um novo sistema de inscrição, que terá como inspiração o sistema atual, porém com a atualização das tecnologias utilizadas e a adição das funcionalidades descritas nas seções a seguir.

2. Novas Funcionalidades

2.1. Envio de Documentos

A funcionalidade mais importante implementada neste novo sistema é o envio de documentos de forma digital por parte dos candidatos, que atualmente têm o trabalho de comparecer às secretarias dos cursos com as cópias físicas dos documentos. A praticidade dessa funcionalidade beneficia tanto o candidato quanto a universidade.

[1]Vide http://www.capg.ufsc.br/inscricao

2.2. Salvamento Progressivo das Informações

Caso um candidato queira fazer a inscrição em algum dos programas de Pós-Graduação da UFSC através do sistema de inscrição atualmente em uso, este deve, essencialmente, fornecer todas as informações em uma única sessão. Ao final, o sistema fornece um número de inscrição e a possibilidade de impressão de um comprovante de inscrição.

Devido à extensão do formulário, deu-se ao candidato a opção de salvar o progresso do seu preenchimento. O candidato pode, dessa forma, distribuir o preenchimento do formulário em várias sessões até que esteja seguro das informações fornecidas e decida-se por finalizar sua inscrição.

2.3. Ordem de Preenchimento do Formulário

No sistema atual, o preenchimento do formulário segue uma ordem predefinida. Muitas vezes, no entanto, o candidato não possui todas as informações necessárias e/ou relevantes ao iniciar o preenchimento do formulário. A possibilidade de preenchimento do formulário sem nenhuma ordem imposta neste processo (a não ser em situações nas quais exista uma ordem implícita) foi fornecida ao candidato.

2.4. Interface Responsiva

A grande maioria dos *websites* existentes hoje na internet foram desenvolvidos para serem visualizados nas versões *desktop* dos navegadores. Não é novidade, no entanto, que a utilização de dispositivos móveis para a navegação na internet tem crescido nos últimos anos. Segundo dados obtidos na própria SeTIC, aproximadamente 30% dos usuários que fizeram inscrição nos sistemas NDI (Núcleo de Desenvolvimento Infantil) e CAPL (Colégio da Aplicação) em 2017 utilizaram dispositivos móveis (celulares e tablets).

O recente aumento na diversificação de dispositivos utilizados para a navegação na internet tem exigido das aplicações web a capacidade de disponibilizar interfaces específicas para cada dispositivo. Identificou-se, portanto, a necessidade do desenvolvimento de uma interface com o usuário que se ajuste da melhor forma possível aos diversos dispositivos por ele utilizados.

3. Análise e Projeto

3.1. Requisitos Funcionais

Os requisitos funcionais descrevem os serviços que o sistema deve fornecer, como deve reagir a entradas específicas e como deve se comportar em determinadas situações. Visando o entendimento por todas as partes interessadas (*stakeholders*), a especificação dos requisitos é feita aqui de forma abstrata (em alto nível).

Requisito 1 - Visualizar programas oferecidos. O usuário poderá visualizar a lista de programas de pós-graduação oferecidos pela UFSC sem a necessidade de realizar *login* no sistema.

Requisito 2 - Realizar cadastro no CAS. O usuário poderá se cadastrar no Sistema de Autenticação Centralizada a partir da página inicial do sistema.

Requisito 3 - Iniciar sessão. O usuário poderá iniciar a sessão no sistema através de *login*.

Requisito 4 - Encerrar sessão. O usuário poderá encerrar a sessão no sistema a qualquer momento através de *logout*.

Requisito 5 - Salvar a inscrição. O usuário poderá salvar as informações inseridas no formulário para um acesso futuro. Após o primeiro salvamento, o usuário receberá um número de inscrição.

Requisito 6 - Alterar informações. O usuário poderá alterar as informações inseridas durante todo o período de inscrição do programa selecionado, desde que não tenha finalizado a sua inscrição.

Requisito 7 - Finalizar a inscrição. O usuário poderá finalizar a sua inscrição. Após finalizada, os dados inseridos não poderão mais ser modificados.

Requisito 8 - Receber um comprovante de inscrição. Após finalizada a inscrição, o usuário receberá um comprovante de inscrição através do e-mail cadastrado.

Requisito 9 - Importar dados do CAS. Os dados que foram utilizados para fazer o cadastro no CAS deverão ser importados automaticamente.

Requisito 10 - Importar dados de inscrições anteriores. Ao iniciar uma nova inscrição, o usuário poderá importar dados pessoais e dados de contato da última inscrição realizada.

Requisito 11 - Fazer *upload* de arquivos. O usuário poderá fazer o *upload* de arquivos solicitados pelo programa escolhido.

Requisito 12 - Importar arquivos. O usuário poderá buscar arquivos utilizados em inscrições anteriores.

Requisito 13 - Inserir informações. O usuário poderá inserir diversos tipos de informações, as quais são agrupadas em dados do programa, pessoais, econômicos, de contato e de formação. Embora esses dados não estejam descritos em pormenores neste relatório, estavam presentes nos protótipos apresentados ao cliente na reunião de definição dos requisitos.

3.2. Requisitos Não Funcionais

Os requisitos não funcionais são aqueles que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários [Sommerville 2011]. Relacionam-se às propriedades emergentes do sistema e definem restrições sobre a sua implementação. Foram identificados os seguintes requisitos:

- Usuários do sistema devem autenticar-se utilizando o Sistema de Autenticação Centralizada (CAS).
- A aplicação Web deve ser feita utilizando a linguagem de programação Java e as tecnologias que são comumente utilizadas no desenvolvimento Web com Java e que estão disponíveis na SeTIC.
- A aplicação Web deve utilizar a estrutura e seguir o visual dos demais sistemas desenvolvidos na SeTIC[2]. Para tanto, deve-se utilizar o projeto denominado *Projeto Base*, criado por desenvolvedores da SeTIC para o desenvolvimento de novos sistemas.
- O sistema deve possuir uma interface responsiva, *i.e.*, capaz de ajustar a disposição do conteúdo em função das dimensões da janela do navegador e do tipo de dispositivo utilizado (*desktop* ou móvel).

[2]Este requisito se refere à versão *desktop* da aplicação.

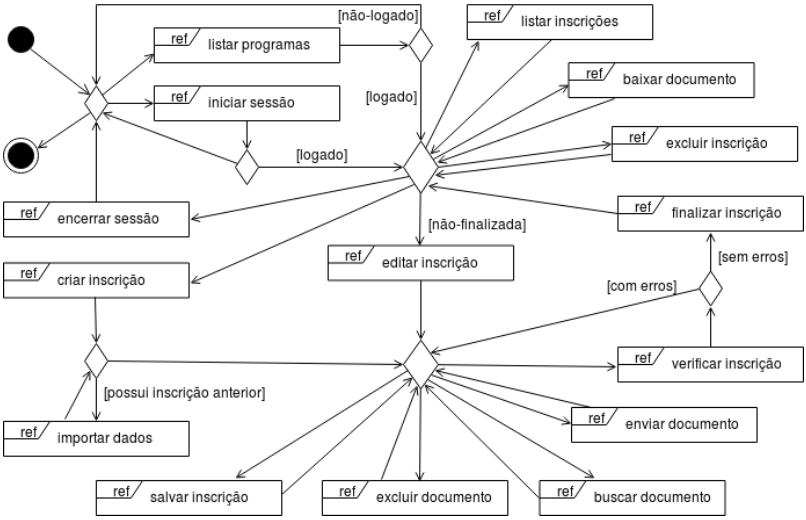
3.3. Projeto do Sistema

O segundo estágio do modelo de desenvolvimento de software adotado consiste em desenvolver a arquitetura geral do sistema. Aqui descrevem-se as principais abstrações do sistema e seus relacionamentos através dos diagramas de uso geral de interação e de entidade-relacionamento.

Introduzido na versão 2.0 da UML, este diagrama é uma versão modificada do diagrama de atividades, no qual ao invés de ações ou atividades, os nós correspondem a *interações* que são modeladas em outros diagramas. Sua finalidade é modelar as sequências possíveis de ocorrência das interações identificadas.

Uma possível aplicação do diagrama de visão geral de interação é a modelagem do fluxo entre as interações associadas aos casos de uso identificados. Tem-se, assim, uma visão geral do sistema em execução durante um processo de inscrição. O fluxo de execução pode ser visto na Figura 2. Algumas guardas foram omitidas para não saturar o diagrama.

Figura 2. Diagrama de Visão Geral de Interação. Fonte: elaborada pelo autor.

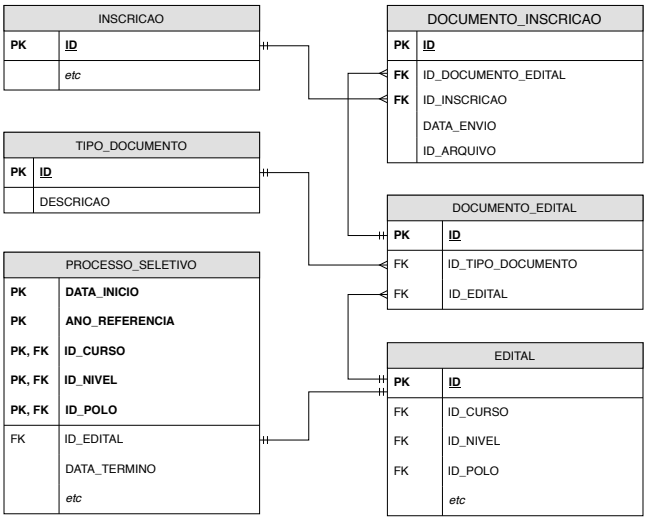


Tendo em vista o requisito funcional de permitir ao candidato enviar e importar documentos, foi desenvolvido o modelo de relacionamento entre entidades exibido na Figura 3.

As entidades `INSCRICAO`, `PROCESSO_SELETIVO` e `EDITAL` já existiam na base de dados em forma de tabelas. A primeira engloba os dados pessoais do candidato; as outras duas representam processos seletivos e seus respectivos editais. As demais entidades foram modeladas para atender aos requisitos 11 e 12 (vide Seção 3.1):

- `TIPO_DOCUMENTO` - contém os tipos dos documentos obrigatórios para a

Figura 3. Modelo entidade-relacionamento associado ao salvamento de documentos. Fonte: elaborada pelo autor.



inscrição nos diversos programas, *e.g.*, histórico escolar, diploma de graduação, RG, *etc*.

- `DOCUMENTO_INSCRICAO` - entidade que armazena o código dos documentos enviados por cada candidato. Os documentos propriamente ditos são armazenados em um repositório e seus identificadores salvos no campo `ID_ARQUIVO`. Pode-se obter, por exemplo, uma lista de códigos de documentos enviados por um determinado candidato e, a partir da relação com a entidade `DOCUMENTO_EDITAL`, qual o tipo de cada documento e a qual processo seletivo pertence.
- `DOCUMENTO_EDITAL` - entidade associativa entre as entidades `EDITAL` e `TIPO_DOCUMENTO`, com chave própria.

4. Implementação

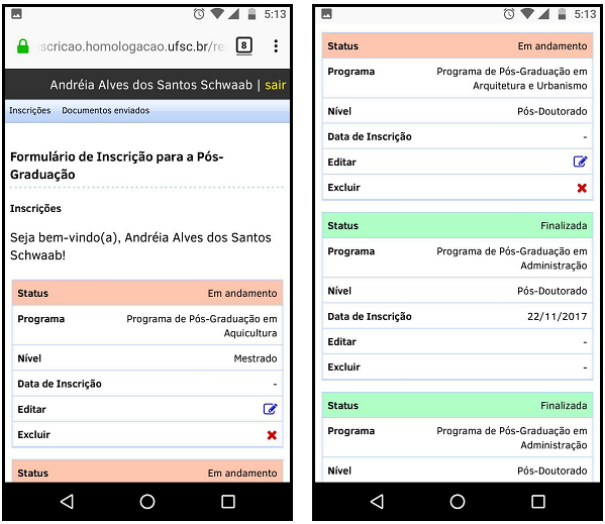
A adaptação da interface do sistema para a utilização em dispositivos móveis foi feita a partir da adoção das mesmas *views* em navegadores comuns e navegadores de dispositivos móveis, nas quais a disposição dos elementos é controlada a partir de *CSS media queries*. As imagens utilizadas na exemplificação das funcionalidades mostradas a seguir foram obtidas em um navegador de um Motorola Moto G5 com resolução de 1920x1080 pixels.

4.1. Gerenciamento de Inscrições

O usuário autenticado no sistema tem acesso ao histórico de inscrições realizadas. Neste histórico, identificam-se o programa, o nível e a data de finalização da inscrição, como pode ser visto na tabela da Figura 4. Cada inscrição possui ainda um *status*: *finalizada*, inscrição pertencente a um processo seletivo passado que não pode ser editada ou ex-

cluída; *em andamento*, inscrição pertencente a um processo seletivo em vigência que pode ser editada e/ou excluída.

Figura 4. *View* inscrições após a autenticação (à esquerda) e detalhe para os dois tipos de inscrição (à direita). Fonte: elaborada pelo autor.



4.2. Envio de Documentos

Uma das principais funcionalidades providas por este sistema é permitir que o candidato faça o envio de documentos. Não somente se reduz a quantidade de papéis que ficariam armazenados nas secretarias dos cursos (economizando espaço), como também potencializa-se a segurança dos documentos (documentos em papel estão sempre sujeitos a danos e avarias) e facilita-se a sua busca. Essa funcionalidade está implementada na aba `Documentos`, como pode ser visto na Figura 5. As linhas da tabela representam os documentos que são exigidos pela secretaria a qual o programa escolhido está vinculado. No exemplo fictício mostrado, são exigidos três documentos, dois dos quais já foram enviados, e um está pendente.

4.3. Verificação de Erros e Finalização de Uma Inscrição

Pode-se verificar a existência de erros e o não-preenchimento de campos obrigatórios em qualquer momento durante o preenchimento do formulário por meio do botão `Verificar` (salvo a situação em que o usuário está criando uma nova inscrição e ainda não salvou o formulário). Na presença de tais erros, uma mensagem é exibida para o usuário e as abas que possuem campos não-preenchidos ou com erros são destacadas com um ícone de exclamação. Ao abrir tais abas, os campos não-validados estão destacados na cor vermelha e com um ícone de exclamação ao lado de seus rótulos, como pode ser visualizado na Figura 6 para as abas `Dados Pessoais` e `Contato`.

Figura 5. Mensagens informativas da aba `Documentos` (à esquerda) e reutilização de documentos enviados (à direita). Fonte: elaborada pelo autor.

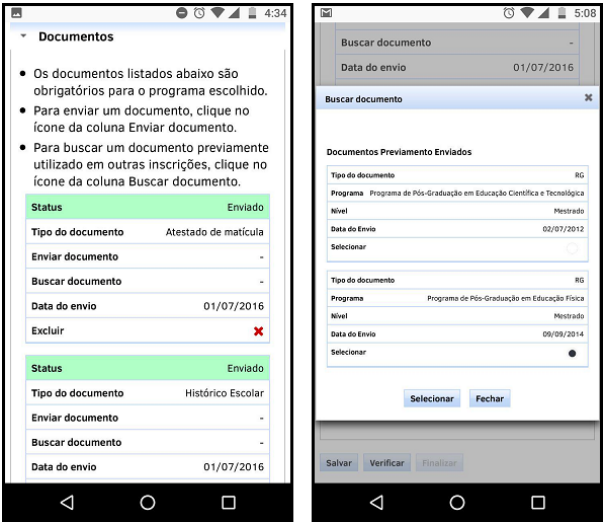
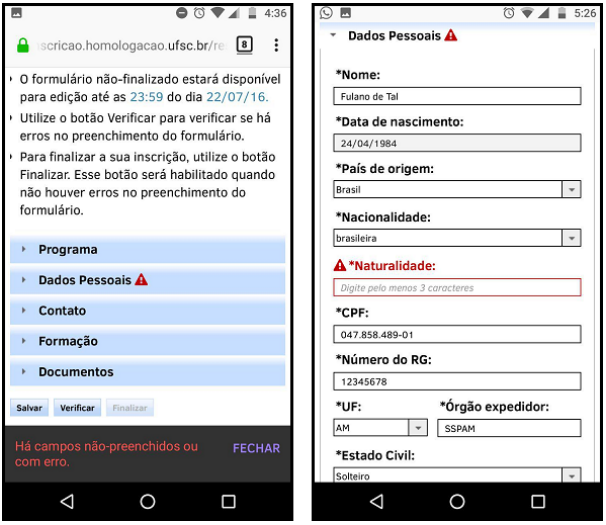


Figura 6. Erros encontrados no preenchimento do formulário. Fonte: elaborada pelo autor.



5. Conclusões

As maneiras de inscrição nos cursos de Pós-Graduação da UFSC atualmente são variadas. Grande parte dos programas utiliza um sistema informatizado oficial, que não atende integralmente os requisitos existentes. Alguns cursos utilizam seus próprios sistemas, dificultando a sua manutenibilidade e introduzindo duplicações de código e vulnerabilidades.

Torna-se relevante, portanto, o desenvolvimento de um novo sistema de inscrição unificado, que dê suporte a novas funcionalidades e que utilize as tecnologias atualmente em uso na SeTIC.

Ao longo deste projeto, diversas atividades previstas foram desenvolvidas para que o objetivo fosse atingido, especificamente:

- Levantamento das funcionalidades desejadas para o sistema através de entrevistas com o cliente (PROPG);
- Especificação dos requisitos funcionais e não-funcionais do sistema a partir destas funcionalidades;
- Definição da arquitetura do sistema a partir de diagramas feitos segundo as especificações da UML: diagrama de casos de uso, diagrama de visão geral de interação e diagramas de classes;
- Prototipação a partir do desenvolvimento de um conjunto de *wireframes*.
- Adequação do Projeto Base para atender a estrutura definida;
- Implementação dos casos de uso identificados.
- Adaptação da interface do sistema para sua utilização em dispositivos móveis de forma responsiva.

Como resultado, obteve-se um novo sistema de inscrição que atende às necessidades dos diversos Programas de Pós-Graduação da UFSC. O aumento no grau de adesão destes programas permitirá um aumento na eficiência administrativa através da melhoria da gestão dos dados dos processos de inscrição por parte tanto dos programas quanto da PROPG.

Embora este projeto não seja classificado como um projeto de pesquisa, comum aos trabalhos de conclusão de curso em Ciências da Computação, seu mérito reside na aplicação de conhecimentos obtidos ao longo do curso para solucionar uma demanda real da própria universidade.

O sistema desenvolvido está atualmente no estágio de homologação. A identificação de *bugs*, as aprimorações e a adição de novas funcionalidades serão feitas pela SeTIC em colaboração com a Pró-Reitoria de Pós-Graduação, bem como as secretarias dos cursos de Pós-Graduação.

Referências

[Sommerville 2011] Sommerville, I. (2011). *Engenharia de Software*. Pearson Prentice Hall, São Paulo, 9 edition.