UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Henrique Hideki Saito

Software para feedback de saúde para operadores de atividades críticas em ambiente industrial

Florianópolis
2021

Henrique Hideki Saito

Software para feedback de saúde para operadores de atividades críticas em ambiente industrial

Relatório final da disciplina DAS5511 (Projeto de Fim de Curso) como Trabalho de Conclusão do Curso de Graduação em Engenharia de Controle e Automação da Universidade Federal de Santa Catarina em Florianópolis.

Orientador: Professor Jefferson Luiz Brum Marques.

Supervisora: Marcela Purificação.

Florianópolis

2021

Henrique Hideki Saito

Software para feedback de saúde para operadores de atividades críticas em ambiente industrial

Esta monografia foi julgada no contexto da disciplina DAS5511 (Projeto de Fim de Curso) e aprovada em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação

Florianópolis, 12 de Setembro de 2021.

Prof. Hector Bessa Silveira,
Coordenador do Curso

Banca Examinadora:

Prof. Jefferson Luiz Brum Marques,
Orientador UFSC/CTC/IEB

Marcela Purificação, Bióloga.
Supervisora
Centro de Inovação SESI em Tecnologias para Saúde

Prof. Jomi Fred Hübner,
Avaliador
Instituição UFSC/CTC/DAS

Prof. Marcelo de Lellis Costa de Oliveira,
Presidente da Banca
UFSC/CTC/DAS

Este trabalho é dedicado aos meus pais, amigos e à minha irmãzinha.

Agradecimentos

Ao meu orientador Prof. Jefferson Luiz Brum Marques, por disponibilizar o seu tempo para ajudar e compartilhar do seu conhecimento.

A minha supervisora na empresa Marcela Conceição, que compartilha dos agradecimentos como orientadora e colega de trabalho.

Ao CIS por me dar a oportunidade de trabalhar em projetos incríveis que impactam nas vidas de muitos trabalhadores.

A todos meus colegas de trabalho pelo respeito, ajuda, paciência, carinho e boas risadas.

A minha família, minha irmãzinha e nossa cachorra Funny.

E aos meus amigos, pela paciência e companheirismo.

RESUMO

O presente trabalho foi desenvolvido no Centro de Inovação SESI em Tecnologias para Saúde em parceria com a startup País Digital com foco na saúde para operadores de atividades críticas em ambiente industrial. Seu objetivo principal é promover o estudo com foco na saúde dos trabalhadores e aumentar a eficácia de melhores práticas nas empresas. Com motivações econômicas, sociais e tecnológicas, o projeto tem como objetivo desenvolver um protótipo que, a partir da medição de parâmetros fisiológicos, possa identificar sinais relacionados à atenção, fadiga e estresse, auxiliando as equipes de SST. Diminuindo assim a quantidade de acidentes devido a estes fatores, melhorando as condições de trabalho, aumentando a competitividade da empresa e gerando avanços tecnológicos. O projeto foi dividido em três partes: a coleta de dados dos colaboradores, a lógica do negócio, que centraliza a aplicação, gerenciando o banco de dados e a interface de usuário que disponibiliza as informações em dashboard.

Palavras-chave: Parâmetros Fisiológicos, Saúde e Segurança no Trabalho, Inovação.

ABSTRACT

This work was developed at Centro de Inovação SESI em Tecnologias para Saúde in partnership with the startup País Digital with focus on the wellbeing of operators working in critical activities in an industrial environment. Its main objective is to develop the study with focus on workers's health and increase the effectiveness of best practices in companies. With financial, social and technological motivations, the project's objective is to develop a prototype that, based on the assessment of physiological parameters, can identify signs related to attention, fatigue and stress, helping OSH (Occupational safety and health) teams. Thus, reducing the number of accidents due to these factors, improving working conditions, increasing the company's competitiveness and generating technological advances. The project was divided into three parts: data acquisition, business logic, which centralizes the application, managing the database, and user interface that makes information available on the dashboard.

Keywords: Physiological Parameters, Health and Safety at Work, Innovation.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ECG - Eletrocardiograma
SST - Saúde e Segurança do Trabalho
OSH - Occupational safety and health
CIS - Centro de Inovação SESI
CRUD - Create, read, update and delete
API - Application Programming Interface
REST - Representational state transfer
SPA - Single page application
PWA - Progressive Web Apps
UI - User Interface
CLI - Command-line interface
VFC - Variabilidade da frequência cardíaca
SNA - Sistema Nervoso Autônomo
BPM - Batimentos por minuto
DTO - Data Transfer Object
HTML - HyperText Markup Language
CSS - Cascading Style Sheets
Px - Pixel
Vh - View Height
Vw - View Width
HTTP - Hypertext Transfer Protocol

SUMÁRIO

1 INTRODUÇÃO

No contexto de indústria 4.0, em que a incorporação de novas tecnologias cresce cada vez mais em busca de vantagens e maior competitividade no mercado, o projeto Driver4.0, em parceria com a startup País Digital, visa auxiliar as empresas na gestão de seus trabalhadores acompanhando a saúde e aptidão dos colaboradores através do monitoramento de sinais vitais, mais especificamente acerca da fadiga, cansaço e atenção, para assim auxiliar os gestores na tomada de decisão, evitando que trabalhadores cansados realizem tarefas perigosas.

O sistema se propõe, em um primeiro momento, a ser aplicado e testado em motoristas de empilhadeiras. A escolha se deve ao fato de ser um veículo muito comum e utilizado em indústrias, com potencial de graves danos se envolvido em incidentes, mas futuramente pode ser aplicado para motoristas no geral.

Acidentes envolvendo empilhadeiras acontecem com maior frequência do que o esperado e geralmente são oriundos de má operação, treinamento deficiente e condições de trabalho adversas. Sendo assim, o projeto possui uma motivação social em relação a melhoria da segurança e da qualidade de vida dos trabalhadores, diminuindo os riscos em seu cotidiano laboral.

Além disso, o projeto possui uma motivação econômica, visando diminuir os gastos com acidentes de trabalho (afastamentos, absenteísmo, destruição ou prejuízos de infraestrutura) e assim impactando positivamente na produtividade da indústria, já que esse tipo de acidente prejudica sua operação logística.

Com o monitoramento em tempo real e de forma contínua, traz ao trabalhador uma melhoria nas condições de trabalho, conciliando a elevação da sua qualidade de vida com o aumento da produtividade.

A identificação ativa das condições de estresse e fadiga dos colaboradores permite um direcionamento mais eficiente das ações de promoção de saúde personalizadas. No contexto empresarial, um ambiente mais seguro influencia diretamente em uma imagem mais positiva da empresa e ajuda a sociedade a aumentar

o número de trabalhadores produtivos, diminuindo consequentemente possíveis gastos com saúde.

Existe também o impacto tecnológico, dado seu caráter inovador que visa ajudar na tomada de decisão de equipes da SST (Saúde e Segurança do Trabalho), uma tendência mundial.

1.1 OBJETIVOS

O objetivo principal deste projeto é desenvolver um protótipo capaz de medir parâmetros fisiológicos de motoristas e, a partir disso, identificar sinais sobre atenção, cansaço e fadiga dos mesmos, auxiliando assim as equipes de SST. A solução deve ser capaz de captar, armazenar e processar os dados do colaborador, assim como interagir com os usuários para adquirir feedbacks e retornar em forma de dashboard para os gestores possibilitando assim alertar os trabalhadores e seus supervisores sobre potenciais riscos e a necessidade de atenção ou intervenção.

Para isso, alguns objetivos específicos devem ser alcançados, como:

- Definir os parâmetros a serem coletados.
- Escolha de tecnologias para analisar os dados fisiológicos.
- Escolha de linguagens a serem utilizadas para o desenvolvimento da interface de usuário.
- Criar interfaces de usuário para os gestores e para os colaboradores.
- Implementação de medidas de segurança.
- Escolha de tecnologia a ser utilizada para desenvolvimento do backend.
- Desenvolver o backend para implementar as regras do negócio e manipular os dados no banco de dados.
- Escolha, desenvolvimento e teste de hardware para coleta.
- Desenvolver interação hardware-software.
- Integração do software de captação e interface de usuário com o backend (modelo 3 camadas).
- Manipulação do banco de dados pelo backend.
- Transformar informações em dashboard para visualização.
- Criar histórico das coletas realizadas.

2 EMPRESA

O Centro de Inovação SESI em Tecnologias para Saúde (CIS Tecnologias para Saúde) desenvolve e aplica tecnologias através de projetos multidisciplinares voltados a melhoria da saúde e segurança dos trabalhadores e ambientes de trabalho com a motivação da redução de custos e crescimento de produtividade da indústria.

A multidisciplinaridade característica do centro permite a incorporação de uma grande variedade de tecnologias inovadoras como inteligência artificial, robótica, visão computacional, gamificação e realidade virtual, para assim, torná-los mais imersivos, interessantes e eficazes.

Os projetos podem ser de edital, onde as indústrias enviam planos de projeto baseados em problemas de seu cotidiano, ou internos, voltados a problemas e interesses do Centro de Inovação como divulgação, melhorias na gestão, etc. Ambos passam por várias etapas que por sua vez são divididas em sub etapas até virar um produto. Dentro da etapa inicial de captação, o projeto passa pelas sub etapas de prospecção, entendimento, definição e negociação para então passar para a etapa intermediária de design de produto onde a solução começa a tomar forma, passando pelas etapas de planejamento, entendimento, ideação, decisão, prototipagem e validação, sendo necessário muitas vezes voltar para etapas anteriores caso a solução não esteja condizente com o problema, ou não seja viável. Por fim o produto entra na fase de desenvolvimento, onde é necessário construir o backlog, planejar sprints, desenvolver e validar.

Para o desenvolvimento de projetos é utilizada a metodologia ágil, onde tarefas são separadas em intervalos de duas semanas (sprints). No início de cada sprint é realizada uma reunião (planning) para decidir quais tarefas devem ser priorizadas nas próximas duas semanas. O mais importante nesta metodologia é a comunicação, constantes testes e validações, por este motivo são realizadas reuniões diárias (dailies), onde é discutido o que foi feito no dia anterior, assim como problemas encontrados.

Um exemplo de projeto de edital, onde o autor ficou responsável pelo sistema de feedback, foi o treinamento ergonômico Ergovirtual que foi desenvolvido baseado em um sistema de identificação postural utilizando algoritmo e câmera para, juntamente com

realidade aumentada e um método já consolidado de avaliação de posturas baseado nos ângulos entre os membros, passar feedback em tempo real com o intuito de sensibilizar e condicionar os colaboradores para comportamentos mais seguros no ambiente de trabalho a fim de prevenir lesões, faltas e acidentes. O sistema de feedback é encarregado de receber os dados captados pela câmera e processado pelo algoritmo, salvar no banco de dados e mostrar para o usuário algumas recomendações baseadas em suas posturas de risco. O projeto pode ser dividido em partes onde o sistema de feedback consiste em interface do usuário desenvolvida em Angular, lógica do negócio em Java Spring Boot, com banco de dados MariaDB, Liquibase para gerenciá-lo e sistema de visão computacional, desenvolvido em Python.

Já o Processos CIS, um projeto interno que o autor participou, consiste em um site tipo CRUD desenvolvido em Angular (frontend) e GO (backend), utilizando MariaDB e Liquibase, com a finalidade de divulgar o centro de inovação, apresentar as fases dos projetos, as atividades que são realizadas em cada etapa e os profissionais que atuam em cada atividade, com foco em design, acessibilidade e experiência do usuário.

Em ambos os projetos o autor trabalhou na interface do usuário e na lógica do negócio, analisando, gerenciando dados, e realizando a integração de diferentes APIs.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 FADIGA

Cansaço e fadiga são condições diferentes, enquanto o primeiro é temporário, ou seja, depois de uma boa noite de sono, ou de descanso adequado o indivíduo se recupera, o segundo é um fenômeno complexo e, portanto, uma debilidade acumulada de cansaço, apresentando privação de sono, alterações no ritmo circadiano, ou período de vigília estendida. A fadiga apresenta efeitos mais profundos e pode exigir tratamentos de recuperação específicos (Pinheiro, 2020).

Quando o corpo está fatigado, reage mais lentamente aos sinais internos e externos, com reduzida capacidade de resposta e reação à estímulos, incluindo até mesmo reações mais lentas a situações, ficando mais propenso a exercer julgamentos inadequados. Isso aumenta consideravelmente o risco de intercorrências (Almeida, 2020; Brasil, 2015;). Este efeito pode ser entendido em etapas, primeiramente o cansaço, que se não tratado se torna fadiga acumulada que pode intensificar para fadiga crônica aumentando a debilidade no desempenho (Pinheiro, 2020).

Algumas variáveis que influenciam na detecção de fadiga incluem a hora do dia, período de vigília (período transcorrido desde o último sono), carga de trabalho (atividades monótonas contribuem para o aumento da sonolência e degradação do desempenho, assim como atividades mais complexas ou com elevada carga de trabalho aumentam a susceptibilidade à fadiga), duração e qualidade do sono.

3.2 ESTRESSE

O estresse é caracterizado pela dificuldade em relaxar, tensão, impaciência, irritabilidade e agitação. O distresse é considerado ruim, comumente o que consideramos estresse, e pode ser classificado em estresse agudo, intenso e rápido e o estresse crônico, mais ameno porém prolongado (Albino & Conde, 2019; Seaward, 2009). Existe também o estresse ocupacional, relacionado a alto esforço e baixa recompensa, as principais respostas psicológicas ao estresse ocupacional têm consistido na insatisfação no trabalho, ansiedade e depressão (Sousa 2005).

3.3 ATENÇÃO

Pode-se definir “atenção” como a capacidade cognitiva que processa a informação proveniente dos sentidos ou da memória, capacidade do indivíduo em responder predominantemente aos estímulos que lhe são comportamentalmente relevantes, e ignorar as distrações arredores. Esta é uma função crucial, pois permite a interação do ser com seu ambiente, além de auxiliar na organização dos processos mentais (Santos, 2018).

A literatura estabelece que o nível de VFC (variabilidade da frequência cardíaca) em repouso possui correlação com desempenho cognitivo e comportamento psicofisiológico, esclarecendo que aqueles com maior VFC possuem respostas mais adequadas quando comparados com aqueles com baixa VFC (Bosquet et.al ,2008 ; Brunetto et.al, 2008).

3.4 ELETROCARDIOGRAMA

Eletrocardiograma, também chamado de eletrocardiografia, é um exame simples utilizado para medir em tempo real atividade elétrica do coração.

Após comparações entre possíveis parâmetros fisiológicos a serem monitorados para a identificação da fadiga do funcionário foi selecionado o ECG por este representar bom custo-benefício de desenvolvimento, implantação e sua eficácia na identificação de tal parâmetro. Entre outras vantagens do ECG estão a inexistência de contra indicações, sua simplicidade, resposta em tempo real e sua ampla utilização, o que acarreta em uma grande quantidade de documentação e pesquisas.

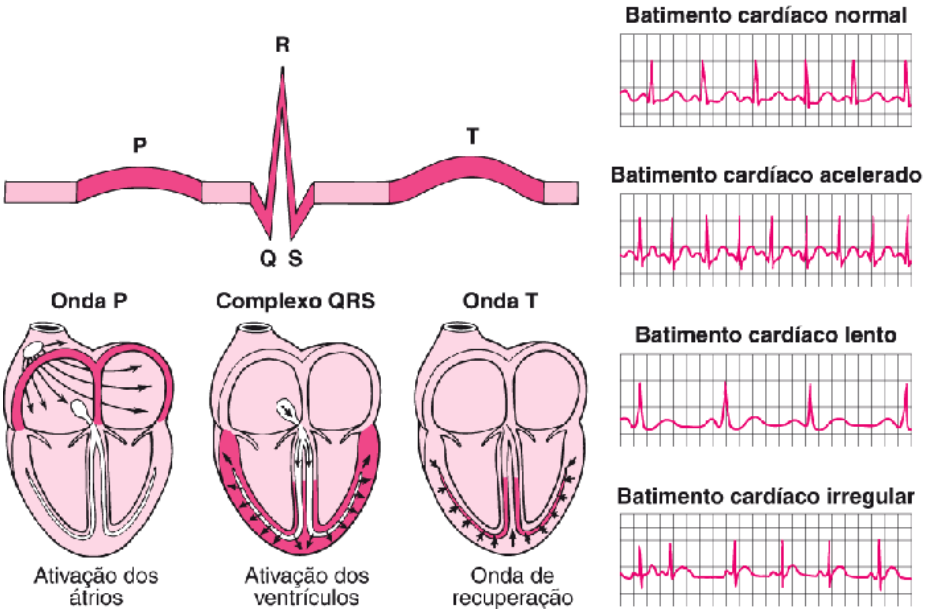
3.5.1 DADOS DO ECG

Para escolher o melhor sinal fisiológico para monitoramento e inferência foi realizada uma busca na literatura científica acerca de quais sinais fisiológicos podem estar relacionados à variabilidade da frequência cardíaca (VFC). Sabe-se que a VFC fornece bons indicadores para avaliação de problemas de saúde, sendo a alta VFC um sinal de boa adaptabilidade e a baixa VFC, geralmente, um indicador de adaptação insuficiente do SNA (sistema nervoso autônomo), o que pode indicar disfunções fisiológicas no indivíduo (Pumprla, Howorka, Groves, Chester & Nolan, 2002; Vanderlei, Pastre, Hoshi, Carvalho & Godoy, 2009).

Estudos na área da saúde têm utilizado os índices de VFC para compreensão de diversas condições patológicas graves, mas também para diagnosticar desordens fisiológicas e psicológicas em indivíduos aparentemente saudáveis quando submetidos a atividades físicas (Dishman, Nakamura, Garcia, Thompson, Dunn & Blair, 2000).

Os batimentos cardíacos, possuem vários parâmetros a serem coletados, entretanto existem algumas limitações como interferências de sinais, sensibilidade de sensores, confiabilidade de dados e etc. Levando em conta estas dificuldades, os dados foram coletados e analisados da seguinte forma:

Figura 1 - Forma de ondas de batimentos cardíacos.



Fonte - manual MDS 2021

A onda P corresponde a contração dos átrios, QRS contração ventricular e T a repolarização dos ventrículos. Além das ondas, também analisamos os períodos das ondas PP e RR, sendo respectivamente, frequência arterial e frequência ventricular.

A análise das informações que mostram ter relevância com o projeto seriam:

- Período QT - representa o período de despolarização e repolarização dos ventrículos.
- Período QRS - representa a contração ventricular, tendo como valores de referências de 60 a 100ms.
- Período PR - representa o intervalo de atividade da contração dos átrios a contração ventricular.

Além dos parâmetros do ECG também é levado em consideração o BPM (batimento por minuto) e VFC (variação da frequência cardíaca).

- BPM - valor de referência de 60 a 100 bpm, quando em repouso.
- VFC - possui diferentes valores dependendo do método utilizado (linear e não-linear)

3.6 ARQUITETURA DE SOFTWARE

Arquitetura de software se trata de escolhas fundamentais sobre a estrutura do software e a metodologia para desenvolvê-lo, estas estruturas são compostas não só de elementos, como também do relacionamento entre eles e suas propriedades. Tais escolhas são fundamentais para o sucesso do projeto, uma vez que influenciam diretamente em suas características, fazendo com que atendam ou não os requisitos.

Um padrão arquitetural é um modelo documentado criado depois de inúmeros estudos e testes que ajuda na tomada de decisões de projeto, alguns deles são:

- Model-view-controller (MVC) - O modelo divide o projeto em três elementos interconectados separando a interface do usuário (view) da lógica do negócio (model) e o controller converte ações do usuário em comandos para ambos model e view.
- Cliente-servidor - Uma arquitetura com estrutura distribuída dividida entre servidores que fornecem recursos (serviços) e clientes que os utilizam.
- Modelo 3 camadas - Semelhante à arquitetura cliente-servidor, diferencia no desenvolvimento do lado do cliente, retirando sua camada de negócio, unificando todo gerenciamento de dados no backend da aplicação, aumentado o controle do projeto e diminuindo o tempo de resposta.
- Micro Serviços - Uma arquitetura orientada a serviços onde o projeto consiste em uma coleção de serviços autônomos.

3.6.1 API REST

APIs são rotinas e padrões estabelecidos por uma aplicação para criação de uma interface de software com a finalidade de disponibilizar serviços a outras aplicações funcionando como intermediador para troca de informações.

REST é um estilo de arquitetura de software que define algumas regras para comunicação via internet, essas regras restringem as maneiras como o servidor pode processar e responder às solicitações do cliente para que o sistema ganhe propriedades não funcionais desejáveis, como desempenho, escalabilidade, simplicidade e confiabilidade. Um sistema que segue estas regras é então chamado de RESTful.

A arquitetura define as seguintes regras:

- Client-server - separação entre o cliente, que necessita do serviço e do servidor que os fornece.
- Stateless - cada requisição do cliente deve fornecer todas as informações que o servidor necessita para responder.
- Cacheable - as respostas devem, implícita ou explicitamente, definir-se como armazenáveis ou não armazenáveis em cache, eliminando parcial ou completamente algumas interações cliente-servidor, melhorando ainda mais a escalabilidade e desempenho.
- Layered system - o cliente não tem conhecimento da complexidade da conexão com o servidor, se está conectado diretamente ou se existem proxys ou balanceadores de carga. Isso facilita o uso de servidores intermediários que, ao prover balanceamento de cargas e caches compartilhados, aumentam a escalabilidade do sistema.
- Code on demand (opcional) - os servidores podem estender temporariamente ou personalizar a funcionalidade de um cliente transferindo código executável.
- Uniform interface - o uso de uma interface uniforme simplifica e desacopla a arquitetura, o que permite que cada parte evolua de forma independente.

3.7 FRONTEND - ANGULAR

Frontend, front end ou front-end é a parte visual da aplicação, implementa a interface de usuário responsável por mostrar informações e interagir com o usuário.

Typescript é uma linguagem de programação desenvolvida e mantida pela Microsoft muito similar ao Javascript,com a adição de tipagem estática opcional o que simplifica o desenvolvimento principalmente de aplicações de grande escala.

Angular é um framework gratuito e de código aberto de aplicações web baseado em typescript liderado pelo time Angular da Google juntamente com alguns indivíduos da comunidade. Este é usado principalmente em aplicativos de larga escala, aplicativos dinâmicos sem recarregamento de páginas (SPA) e PWAs, uma solução econômica que permite o uso de tecnologias da web para executar aplicativos móveis em plataformas online e offline. Os PWAs funcionam em um navegador, mas se comportam de maneira semelhante aos aplicativos nativos, eliminando a necessidade de ir ao Google Play ou App Store para baixar o aplicativo, podendo usar o aplicativo imediatamente.

Angular é uma plataforma complexa difícil de aprender, possui numerosos elementos estruturais que incluem Injectors, Components, Directives, Pipes, Services, entre outros o que agrava sua curva de aprendizado.

Dentre as vantagens do framework podemos citar:

- Desenvolvimento eficaz entre plataformas - desenvolvimento de soluções de aplicativos web progressivos econômicos que podem ser executadas em plataformas móveis
- Alta qualidade das aplicações - ao mesmo tempo que a grande quantidade de elementos estruturais dificulta sua utilização, utilizados corretamente se tornam poderosas ferramentas na criação de inúmeras soluções.
- Velocidade e desempenho - a diversidade de recursos como template syntax, Angular CLI, roteadores, entre outros, permite o carregamento mais rápido do aplicativo e torna sua estrutura compatível com vários tipos de linguagens de programação de back-end para exibir os dados coletados na UI com eficiência.
- Agilização no processo de desenvolvimento - a grande quantidade de documentação detalhada, vinculação de dados bidirecional e Angular CLI são

alguns dos fatores que tornam o desenvolvimento de aplicações utilizando o framework mais rápido.

3.8 BACKEND

Backend é a camada que o usuário não enxerga, responsável pela implementação das lógicas de negócio e gerência dos dados.

Java é uma linguagem de programação de alto nível, baseada em classes e orientada a objetos, projetada para ter o mínimo possível de dependências de implementação. É uma linguagem de programação de propósito geral destinada a permitir que os desenvolvedores de aplicativos escrevam uma vez, execute em qualquer lugar (WORA) o que significa que o código Java compilado pode ser executado em todas as plataformas que suportam Java sem a necessidade de recompilação.

Spring é uma framework de código aberto leve amplamente usado para desenvolver APIs REST. Sendo assim, Java Spring Boot (Spring Boot) é uma ferramenta que agiliza o desenvolvimento de aplicativos da web e microsserviços com Spring Framework.

Tomcat, Jetty e Undertow são embedded servlet containers que facilitam testes na aplicação pois aproximam o ambiente de desenvolvimento da produção.

Entre as vantagens de utilizar Spring Boot podemos citar:

- Desenvolvimento rápido e fácil de aplicativos baseados em Spring
- Capacidade de criar aplicativos autônomos
- Incorporação diretamente do Tomcat, Jetty ou Undertow.
- Quantidades reduzidas de código-fonte

4 REQUISITOS

Os requisitos do projeto foram extraídos da concepção do sistema realizado no workshop com presença do CIS-SC, CIS-RS e País Digital.

4.1 GERAIS

- O sistema deve ser alocado em *cloud* e será mantido pela País Digital.
- A arquitetura deve ser desenvolvida em modelo de microsserviços.
- O sistema deve conter três perfis de usuário: usuário(trabalhador), gerente e administrador.
- O perfil de usuário/trabalhador poderá logar no sistema, resetar sua senha, visualizar o próprio dashboard/informações de sua coleta e responder questionários.
- O perfil de Gerente de fábrica terá login, resetar senha, cadastro de novos indivíduos no sistema e visualizar o dashboard dos funcionários.
- O perfil de administrador terá todas as funcionalidades dos gerentes de fábrica e será responsável pelo cadastro destes.
- O sistema deve ser compliance com a LGPD, utilizando melhores práticas de segurança e encriptação de dados.

4.2 FRONTEND

- Deve possuir etapa de login para acessar o sistema.
- Deve possuir funcionalidade de recuperação de senha.
- A plataforma web apresenta a tradução dos valores capturados pela visão computacional em formas de gráficos e cards, podendo ser utilizado ferramentas de business intelligence embarcada na página.
- Na página do usuário deve conter um questionário.
- Deve possuir funcionalidade de cadastro, edição e deleção de usuários e gerentes.
- Se comunica com o(s) backend(s) através de protocolo HTTP via REST API.
- Deve apresentar ao usuário os erros do sistema como, mas não somente: sensores não conectados, erro em conexão do banco de dados, erros em comunicação com MQTT.

4.3 BACKEND

- Realiza comunicação com o frontend realizando troca de informações de login, armazenando o token de autenticação e sessões de usuário do banco de dados.
- Recebe informações do frontend e realiza cadastro, edição e deleção de usuários no banco de dados.
- Conecta-se com o banco de dados para realização de queries via SQL retornando informações para o Frontend.
- Deve possuir sistema de segurança de acesso com permissões de usuário.
- Realiza comunicação com o software de captação de dados salvando os dados do usuário no banco de dados.

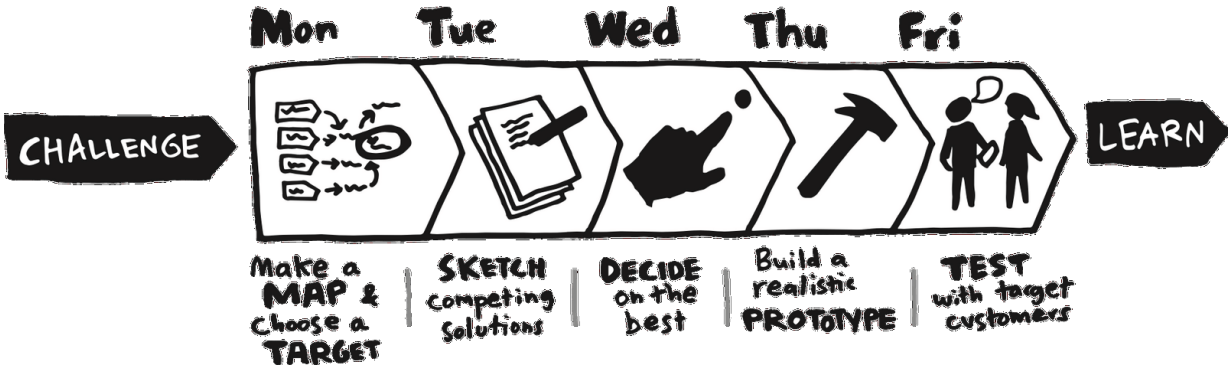
4.4 BANCO DE DADOS

- Deve conter estrutura para armazenamento de usuários e senhas de forma encriptada.
- Deve conter estrutura para armazenamento de sessões e token de usuários.
- Deve conter estrutura para armazenamento dos questionários.
- Deve conter estrutura para armazenamento de perfis de usuário.
- Deve conter estrutura para armazenamento dos dados fisiológicos do usuário transformados em indicadores.l

5 DETALHES DO PROJETO

Diversas etapas e sub etapas foram necessárias na criação do protótipo, começando pela captação, onde foi necessário prospectar, entender e definir o problema a ser resolvido por meio de estudos e pesquisas, para então começar o design do produto onde foi realizada uma Design Sprint (figura 2). Durante cinco dias a equipe se reuniu por meio de vídeo chamadas para diversas atividades, cada dia compreendendo uma etapa, sendo elas: mapear, entender, decidir, prototipar e validar. A partir disso, foi possível perceber qual solução seria mais adequada para o problema encontrado.

Figura 2 - Design Sprint



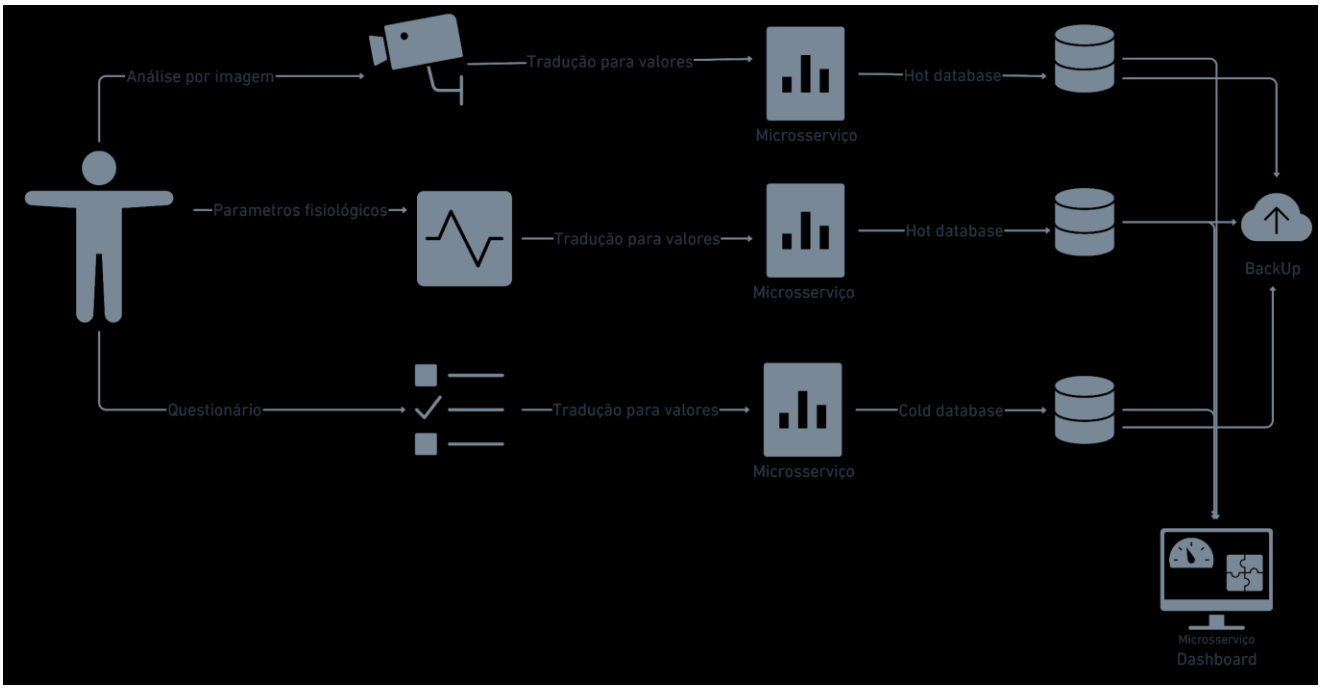
Fonte - https://www.thesprintbook.com/

O último dia do Design Sprint é reservado para testes do produto com usuários. Sendo assim, foi agendado com a empresa C-Pack, que atua na fabricação de tubos extrudados, localizada no município de São José. Lá a equipe encontrou o responsável pelo setor de logística e os funcionários do mesmo, que participariam dos testes. Foi utilizada uma sala para coleta de dados do eletrocardiograma, e a simulação de uso do dashboard para coleta de respostas sobre sono, disposição e emoção. Um dos pontos a ser validado era o uso de uma câmera na empilhadeira, que foi dispensada devido à alta vibração do equipamento e a necessidade de validar essa implementação com as empresas fabricantes de empilhadeiras.

Participaram dos testes operadores de empilhadeira, líder de equipe e gestor, e validou-se o interesse na visualização e acompanhamento dos dados que demonstram a fadiga dos operadores.

Finalmente foi definido uma primeira solução como mostra a figura a seguir:

Figura 3 - Protótipo da solução



Fonte - Autoral

Onde o usuário deve realizar duas medições ao dia, uma no início e outra ao fim do expediente e responder a um questionário, esses dados são enviados ao backend e salvo no banco de dados para posteriormente serem transformados em indicadores e exibidos na forma de dashboard.

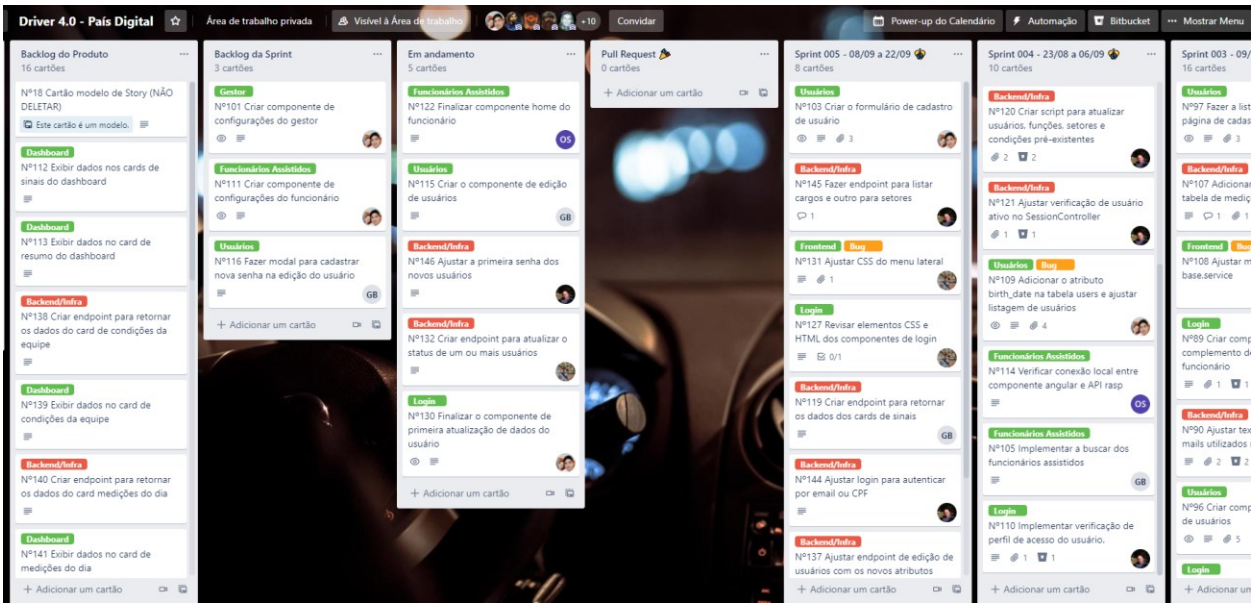
Depois de definida a solução foi iniciada a etapa de desenvolvimento utilizando a metodologia ágil, mais especificamente Kanban, na qual as tarefas são divididas em todo, a serem feitas, doing, sendo realizadas e done, aquelas que já foram finalizadas. Para organizar tais tarefas, é utilizado o Trello (figura 4) onde os cards são divididos em colunas e ordenados por prioridade (de cima para baixo). O método separa o tempo de desenvolvimento em sprints, no caso do projeto, sprints de duas semanas, com uma reunião de planejamento no início de cada sprint e reuniões diárias para alinhamento da equipe, tirar dúvidas, discutir problemas encontrados ou só atualizar as atividades.

Foi utilizado o Bitbucket, um serviço de hospedagem de repositório de código-fonte baseado em Git de propriedade da Atlassian para gerenciar versões do código, permitindo que cada desenvolvedor pudesse realizar sua tarefa utilizando uma branch separada, mitigando o risco de perder ou alterar o código principal.

Vale ressaltar que, no decorrer do desenvolvimento algumas alterações foram necessárias para adaptar e ajustar o protótipo ou resolver problemas que não haviam sido previstos.

No contexto deste projeto o autor atuou nas partes do frontend, backend e banco de dados que serão explicados mais à frente.

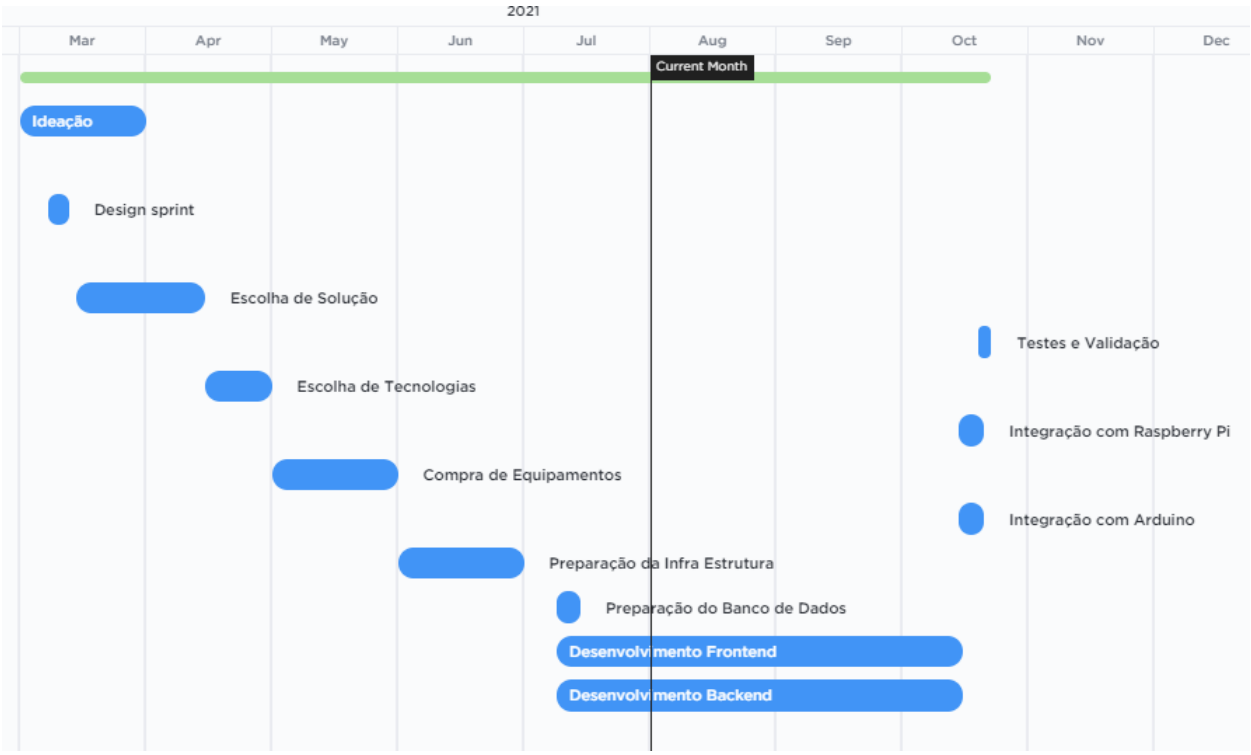
Figura 4 - Trello



Fonte - Autoral

Segue uma imagem do diagrama criado previamente do planejamento das atividades do projeto:

Figura 5 - Diagrama de Gantt

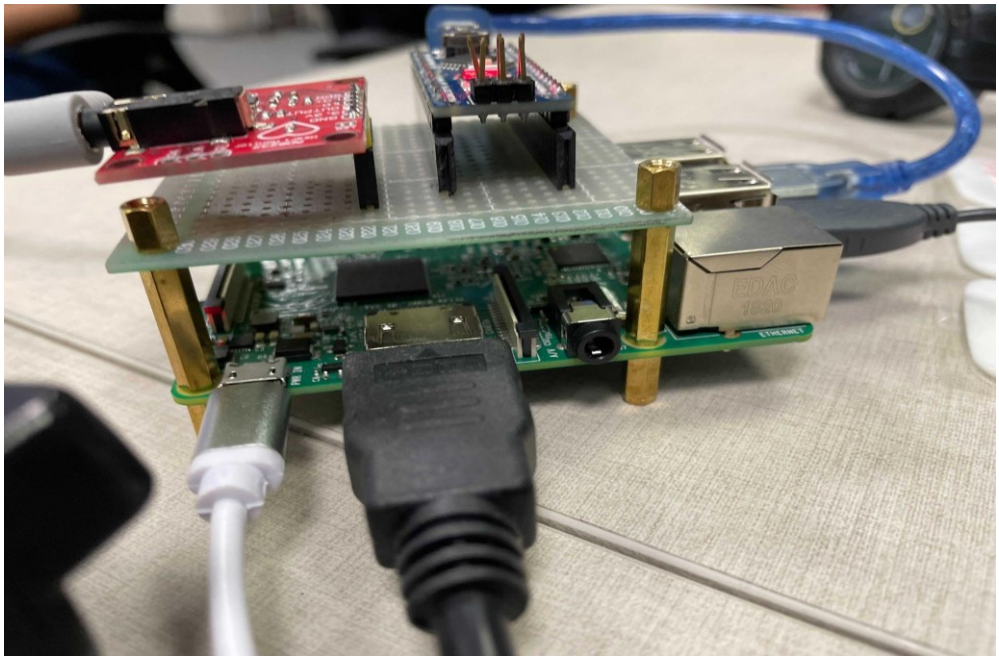


Fonte - Autoral

5.1 SISTEMA DE CAPTAÇÃO DE DADOS

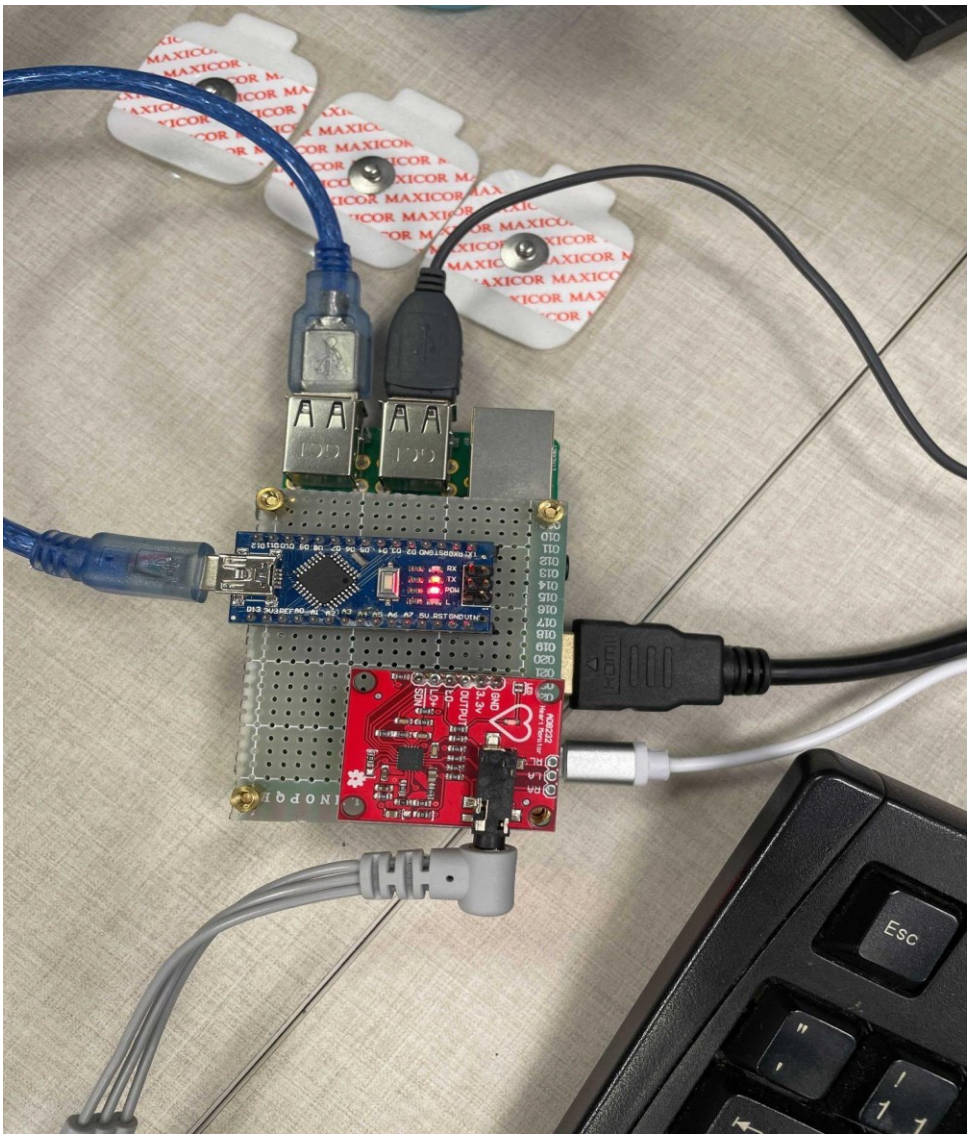
Na parte de aquisição de dados foram selecionados o microchip EPS8266, para armazenar os valores e posteriormente processá-los em um computador virtual, como EC2 (processamento em nuvem) ou AWS Lambda. Para facilitar este lado do desenvolvimento foi escolhido o Raspberry Pi, que armazena e processa os dados e possui protocolo de conexão com a internet. Foi escolhido também um microcontrolador arduino nano por disponibilidade e a placa de sensor AD8232, que é projetada para extrair, amplificar e filtrar pequenos sinais biopotenciais na presença de condições ruidosas, como aquelas criadas pelo movimento ou colocação remota do eletrodo. Esta placa permite que um microcontrolador seja incorporado para adquirir o sinal de saída facilmente. O estrutura do hardware more ser vista nas imagens a seguir:

Figura 6 - Hardware de captação vista lateral



Fonte - Autoral

Figura 7 - Hardware de captação vista superior



Fonte - Autoral

Também foi necessária a escolha de eletrodos, onde foi optado pelo eletrodo alicate por não precisar de cola. A figura a seguir demonstra a posição dos eletrodos para realização da captação.

Figura 8 - Posição dos eletrodos durante captação

Fonte - Autoral

O desenvolvimento do software de tratamento de dados foi realizado nas linguagens C (arduino) e Python (tratamento de dados) com a utilização de bibliotecas como flask (API), Serial para receber, Neurokit2 para processar e Panda para tratamento de dados (figuras 9 e 10).

Figura 9 - Análise de parâmetros

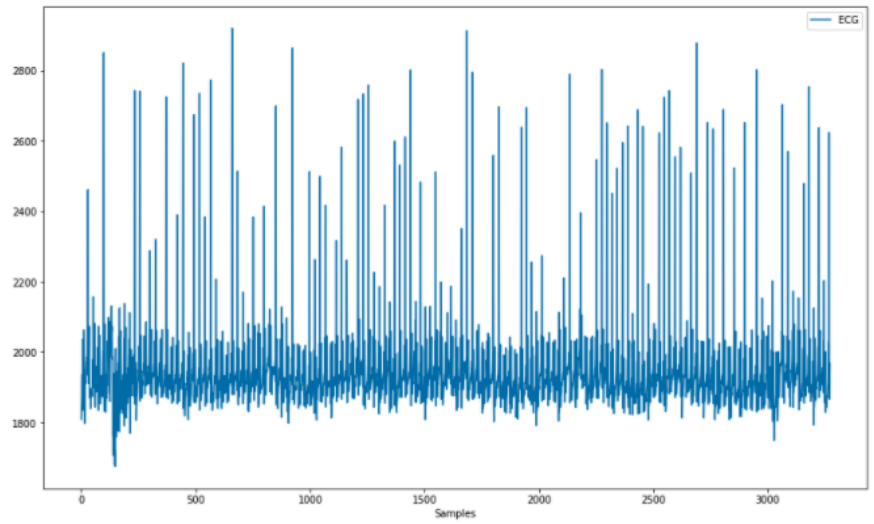
```
import pandas as pd

data = pd.read_csv("G1PreT.csv")  #coloque o nome do arquivo
print(data)
#retorna o conteudo do arquivo
```

```
      Pulse
0      1931
1      1905
2      2112
3      2030
4      1840
...     ...
3031   1901
3032   1959
3033   1931
```

Fonte - Autoral

Figura 10 - Visualização dos dados

```
data = pd.read_csv('G2.csv')

data = pd.DataFrame({"ECG": data['Pulse']})

nk.signal_plot(data)
#Retorna o dado plotado (visualizar graficamento o conteudo)
```



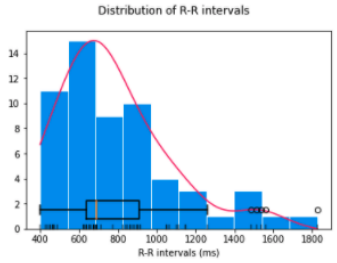
Fonte - Autoral

Foram checados os métodos/funções para checar se efetuam o cálculo de maneira correta e comparando a performance com outras bibliotecas, a neurokit2 (figura 11) apresentou melhor.

Figura 11 - Avaliação do ponto R-R

```
#AVALIAÇÃO RR(MeanNN),SDNN, SDANN, rMSSD, pNN50, pNN20

# Find peaks
peaks, info = nk.ecg_peaks(data["ECG"],sampling_rate=100)
#analise RR
hrv_time = nk.hrv_time(peaks,
                       sampling_rate=80,
                       show=True)
hrv_time
```

	HRV_RMSSD	HRV_MeanNN	HRV_SDNN	HRV_SDSD	HRV_CVNN	HRV_CVSD	HRV_MedianNN	HRV_MadNN	HRV_MCVNN	HRV_IQRNN	HRV_pNN50	HRV_pNN20	HRV_TINN	HRV_HTI
0	448.407269	808.189655	319.337923	452.392324	0.395127	0.554829	687.5	287.25375	0.417824	271.875	79.310345	89.655172	1425.0	3.866667



Fonte - Autoral

5.2 BACKEND

Como descrito anteriormente, é a parte responsável pela implementação das lógicas do negócio, gerência de dados e integração da solução como um todo. Tendo sido desenvolvido pelo autor juntamente com os desenvolvedores do CIS, implementa a comunicação com o frontend e software de captação utilizando a aquitetura de API REST onde o endpoint é contruido dentro de controllers (figura 12)onde as requisições SQL para trazer dados do banco de dados e enviar para o frontend são feitas por meio de repositórios (figura 13) e alterações feitas no banco de dados são feitas através de serviços (figura 14), ambos devem ser incluídos no controller.

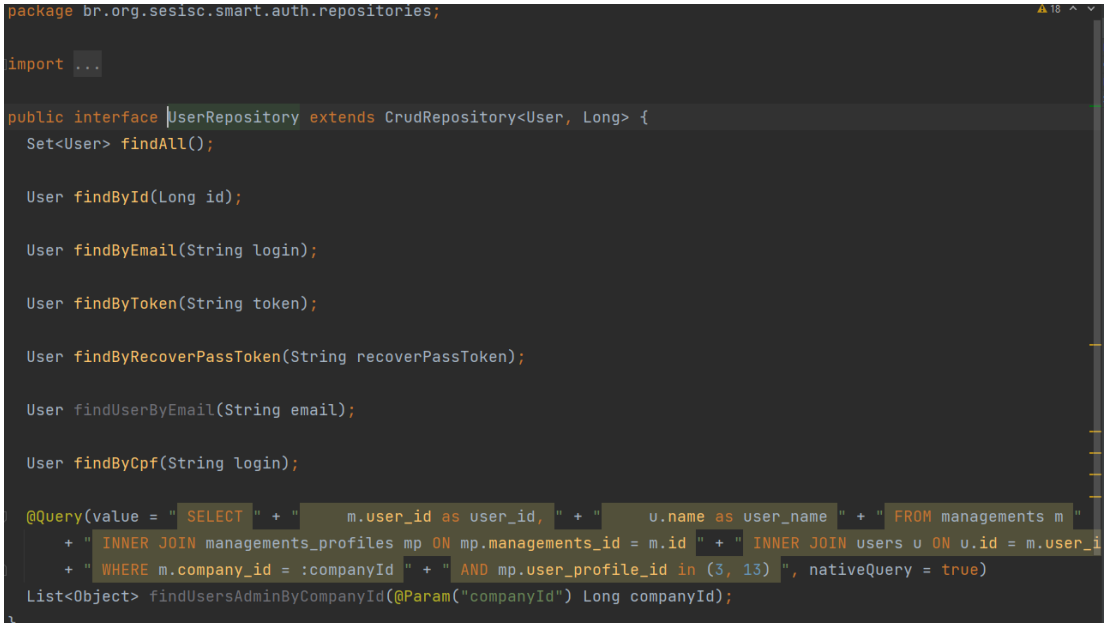
O controller então disponibiliza serviços para o frontend e o software de captação que podem utilizá-los enviando requisições HTTP do tipo GET, POST, PUT, DELETE.

Figura 12 - Controller do usuário no backend

```java
package br.org.sesisc.smart.auth.controllers;

import ...

@RestController
@RequestMapping("/users")
public class UserController {

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private UserService userService;

    @Autowired
    private StorageService storageService;

    private UserMapper userMapper = new UserMapper();

    // Define the logger object for this class
    private static final Logger LOG = LoggerFactory.getLogger(UserController.class);

    private static final String ERROR_USER_CONTROLLER = "Erro UserController : ";
```

Fonte - Autoral

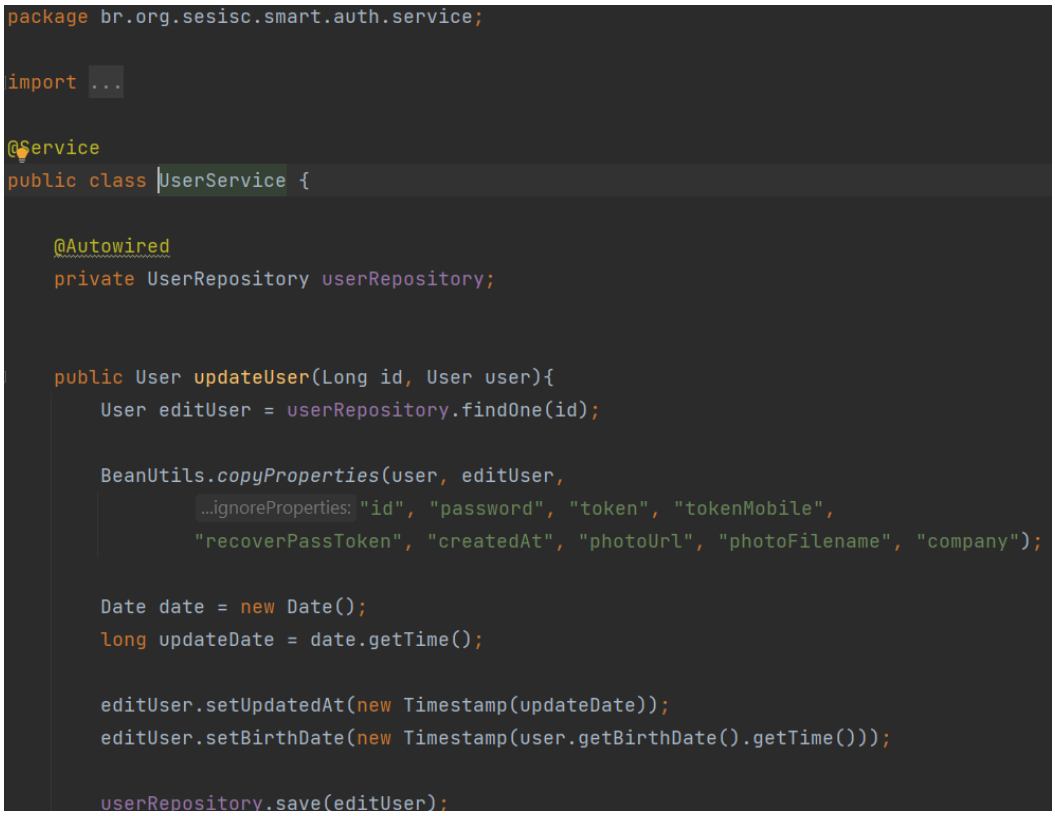
Figura 13 - Repository do usuário no backend



```java
package br.org.sesisc.smart.auth.repositories;

import ...

public interface UserRepository extends CrudRepository<User, Long> {
  Set<User> findAll();

  User findById(Long id);

  User findByEmail(String login);

  User findByToken(String token);

  User findByRecoverPassToken(String recoverPassToken);

  User findUserByEmail(String email);

  User findByCpf(String login);

  @Query(value = " SELECT " + "      m.user_id as user_id, " + "      u.name as user_name " + " FROM managements m "
      + " INNER JOIN managements_profiles mp ON mp.managements_id = m.id " + " INNER JOIN users u ON u.id = m.user_i
      + " WHERE m.company_id = :companyId " + " AND mp.user_profile_id in (3, 13) ", nativeQuery = true)
  List<Object> findUsersAdminByCompanyId(@Param("companyId") Long companyId);
}
```

Fonte - Autoral

Figura 14 - Service do usuário no backend

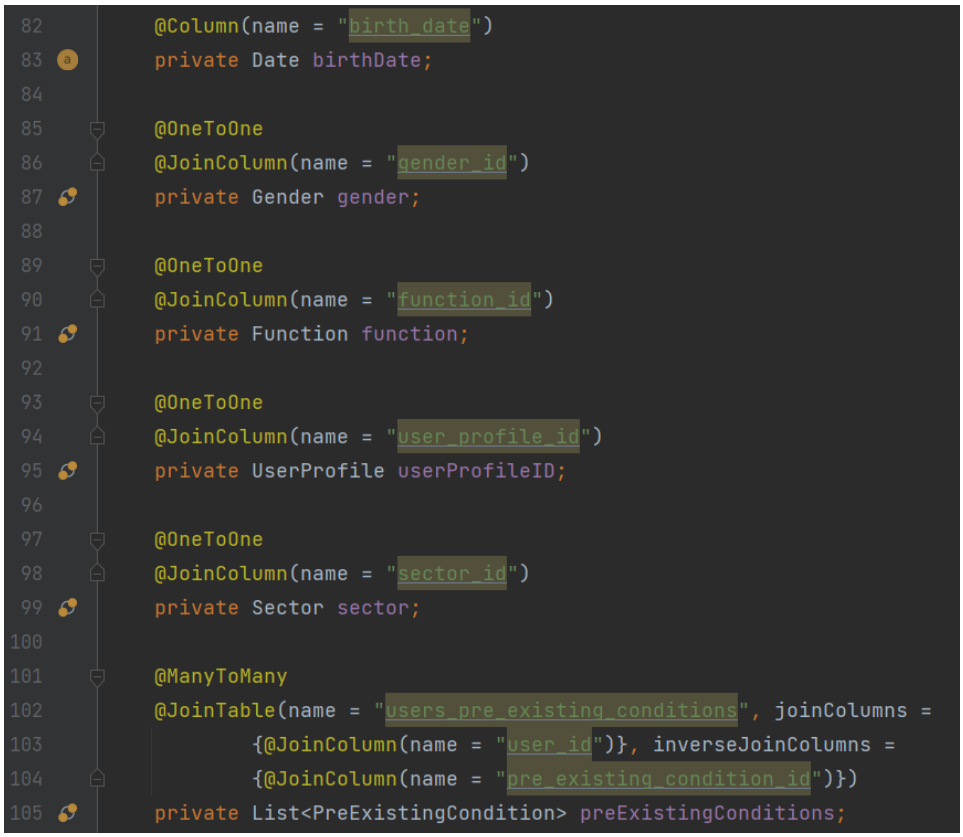


```
package br.org.sesisc.smart.auth.service;

import ...

@Service
public class UserService {

    @Autowired
    private UserRepository userRepository;


    public User updateUser(Long id, User user){
        User editUser = userRepository.findOne(id);

        BeanUtils.copyProperties(user, editUser,
                ...ignoreProperties: "id", "password", "token", "tokenMobile",
                "recoverPassToken", "createdAt", "photoUrl", "photoFilename", "company");

        Date date = new Date();
        long updateDate = date.getTime();

        editUser.setUpdatedAt(new Timestamp(updateDate));
        editUser.setBirthDate(new Timestamp(user.getBirthDate().getTime()));

        userRepository.save(editUser);
```

Fonte - Autoral

A manipulação de dados utiliza models (classes) que representam tabelas no banco de dados e portanto possuem os mesmos atributos, foreign keys, junções com outras tabelas (figuras 15 e 16) e DTOs (figura 17) usado para guardar e transportar apenas dados que precisam ser manipulados entre processos (implementam getter e setters) e podem ser convertidos em models utilizando Mapper (figura 18).

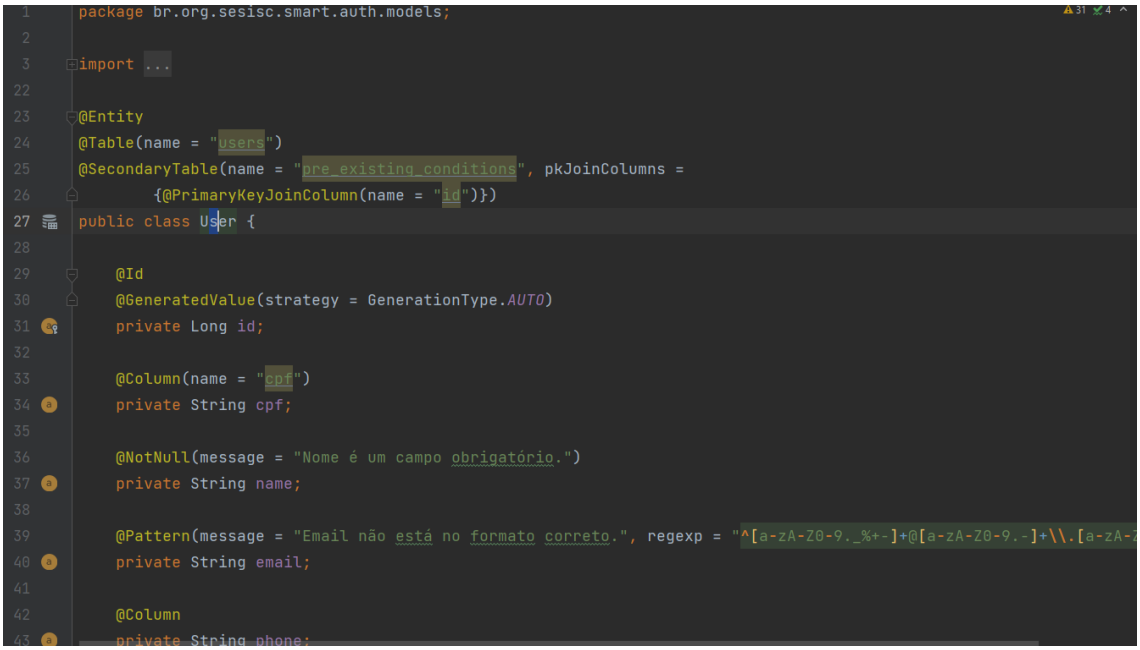
Figura 15 - Conexão entre tabelas no backend



```
82      @Column(name = "birth_date")
83      private Date birthDate;
84
85      @OneToOne
86      @JoinColumn(name = "gender_id")
87      private Gender gender;
88
89      @OneToOne
90      @JoinColumn(name = "function_id")
91      private Function function;
92
93      @OneToOne
94      @JoinColumn(name = "user_profile_id")
95      private UserProfile userProfileID;
96
97      @OneToOne
98      @JoinColumn(name = "sector_id")
99      private Sector sector;
100
101     @ManyToMany
102     @JoinTable(name = "users_pre_existing_conditions", joinColumns =
103             {@JoinColumn(name = "user_id")}, inverseJoinColumns =
104             {@JoinColumn(name = "pre_existing_condition_id")})
105     private List<PreExistingCondition> preExistingConditions;
```

Fonte - Autoral

Figura 16 - Model do usuário no backend



```java
package br.org.sesisc.smart.auth.models;

import ...

@Entity
@Table(name = "users")
@SecondaryTable(name = "pre_existing_conditions", pkJoinColumns =
        {@PrimaryKeyJoinColumn(name = "id")})
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @Column(name = "cpf")
    private String cpf;

    @NotNull(message = "Nome é um campo obrigatório.")
    private String name;

    @Pattern(message = "Email não está no formato correto.", regexp = "^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-
    private String email;

    @Column
    private String phone;
```

Fonte - Autoral

Figura 17 - DTO do usuário no backend

```java
package br.org.sesisc.smart.auth.dto;

import ...

public class UserDTO {

    private long id;
    private String cpf;
    private String name;
    private String phone;
    private String email;
    private String photoUrl;
    private String photoFilename;
    private Gender gender;
    private Function function;
    private Sector sector;
    private List<PreExistingCondition> preExistingCondition;
    private Boolean resumeMailMonthly;
    private Boolean resumeMailWeekly;
    private Boolean alertMailEmployees;
    private Boolean firstAccess;
    private Boolean status;
    private Date birthDate;
    private UserProfile userProfileID;
    private ExercisePractice exercisePractice;
```

Fonte - Autoral

Figura 18 - Método que transforma DTO em Model

```
public User userDtoToModel(UserDTO userDto) {
    User user = new User();

    user.setId(userDto.getId());
    user.setCpf(user.getCpf());
    user.setEmail(userDto.getEmail());
    user.setPhone(userDto.getPhone());
    user.setName(userDto.getName());
    user.setPhotoUrl(userDto.getPhotoUrl());
    user.setPhotoFilename(userDto.getPhotoFilename());
    user.setSector(userDto.getSector());
    user.setPreExistingConditions(userDto.getPreExistingConditions());
    user.setFunction(userDto.getFunction());
    user.setGender(userDto.getGender());
    user.setBirthDate(userDto.getBirthDate());
    user.setResumeMailMonthly(userDto.getResumeMailMonthly());
    user.setResumeMailWeekly(userDto.getResumeMailWeekly());
    user.setAlertMailEmployees(userDto.getAlertMailEmployees());
    user.setFirstAccess(userDto.getFirstAccess());
    user.setStatus(userDto.getStatus());
    user.setUserProfileID(userDto.getUserProfileID());
    userDto.setExercisePractice(user.getExercisePractice());
    userDto.setCompany(user.getCompany());
```

Fonte - Autoral

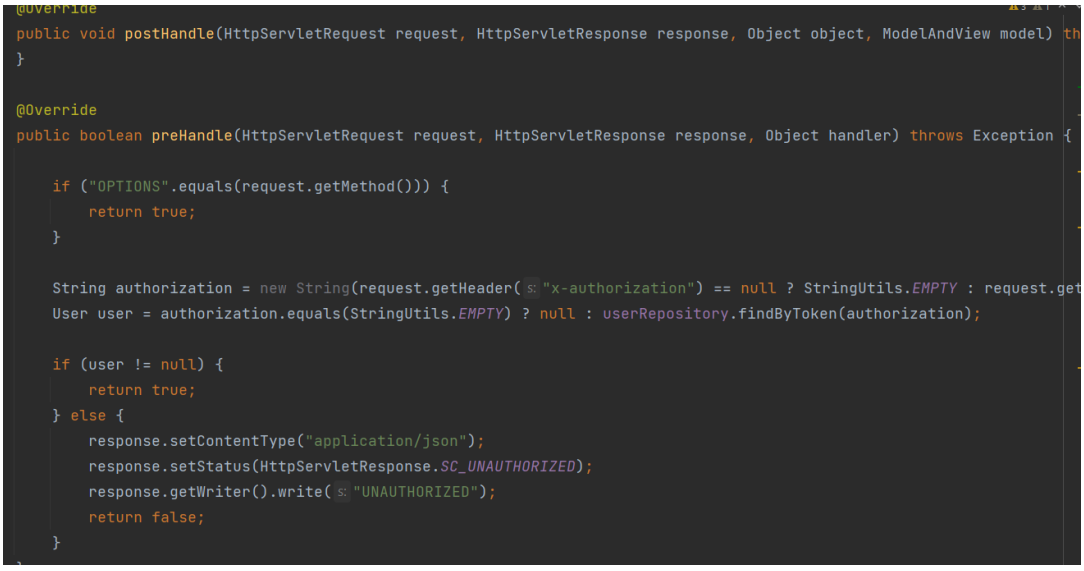
A parte de segurança é implementada no WebConfig (figura 19) que utiliza o Securityinterceptor para checar as permissões (token de autorização) das requisições que chegam no backend (figura 20).

Figura 19 - WebConfig

```
    @Autowired
    private SecurityInterceptor interceptor;

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(interceptor)
                .addPathPatterns("/**")
                .excludePathPatterns("/sessions/**")
                .excludePathPatterns("/users/**/photo_profile")
                .excludePathPatterns("/companies/**/logo")
                .excludePathPatterns("/password/**");
    }

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping( pathPattern: "/**")
                .allowedOrigins("*")
                .allowedMethods("GET", "POST", "PUT", "PATCH", "DELETE", "OPTIONS")
                .allowedHeaders("*")
                .allowCredentials(false)
                .maxAge(3600);
    }
```

Fonte - Autoral

Figura 20 - Securityinterceptor



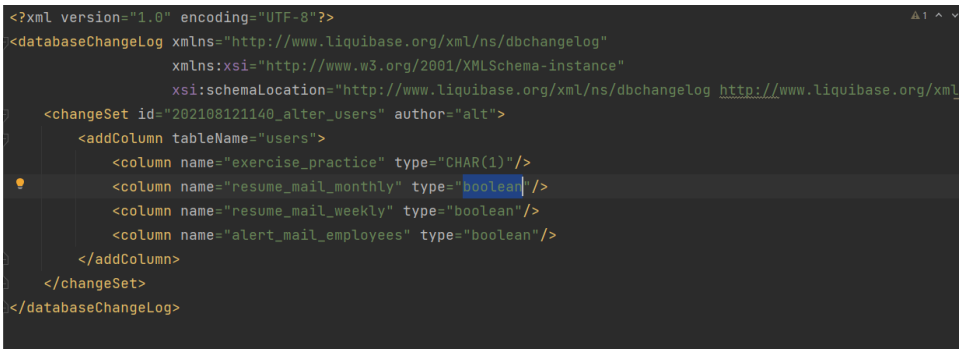
```
@Override
public void postHandle(HttpServletRequest request, HttpServletResponse response, Object object, ModelAndView model) th
}

@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {

    if ("OPTIONS".equals(request.getMethod())) {
        return true;
    }

    String authorization = new String(request.getHeader( s: "x-authorization") == null ? StringUtils.EMPTY : request.get
    User user = authorization.equals(StringUtils.EMPTY) ? null : userRepository.findByToken(authorization);

    if (user != null) {
        return true;
    } else {
        response.setContentType("application/json");
        response.setStatus(HttpServletResponse.SC_UNAUTHORIZED);
        response.getWriter().write( s: "UNAUTHORIZED");
        return false;
    }
```

Fonte - Autoral

O controle do banco de dados é realizado pelo Liquibase, uma biblioteca independente de banco de dados de código aberto usada para rastrear, gerenciar e

aplicar mudanças no esquema do banco de dados através de arquivos XML (figura 21) que podem manipular tabelas (figura 22) ou rodar scripts em SQL(figura 23).

Figura 21 - Arquivo XML de alteração de tabela

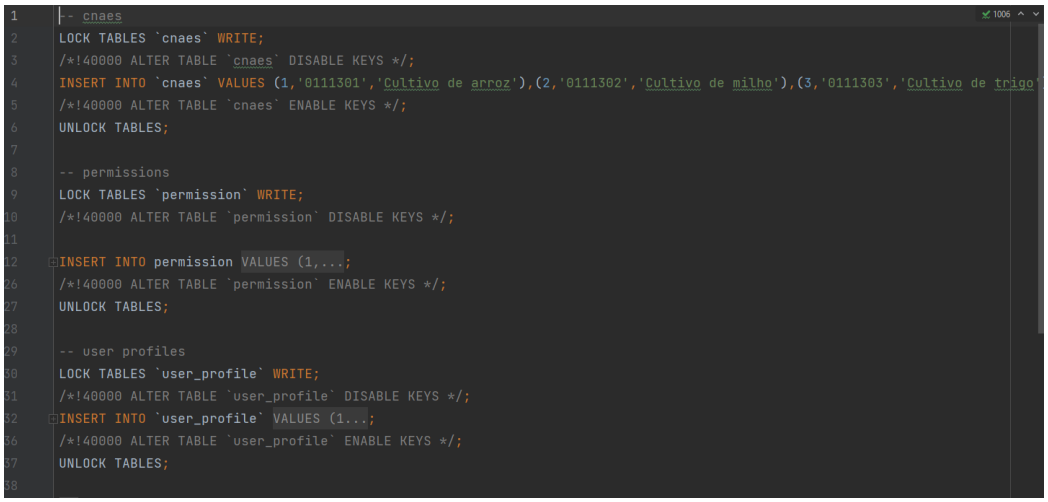


Fonte - Autoral

Figura 22 - Arquivo XML para rodar script SQL

Fonte - Autoral

Figura 23 - Script SQL

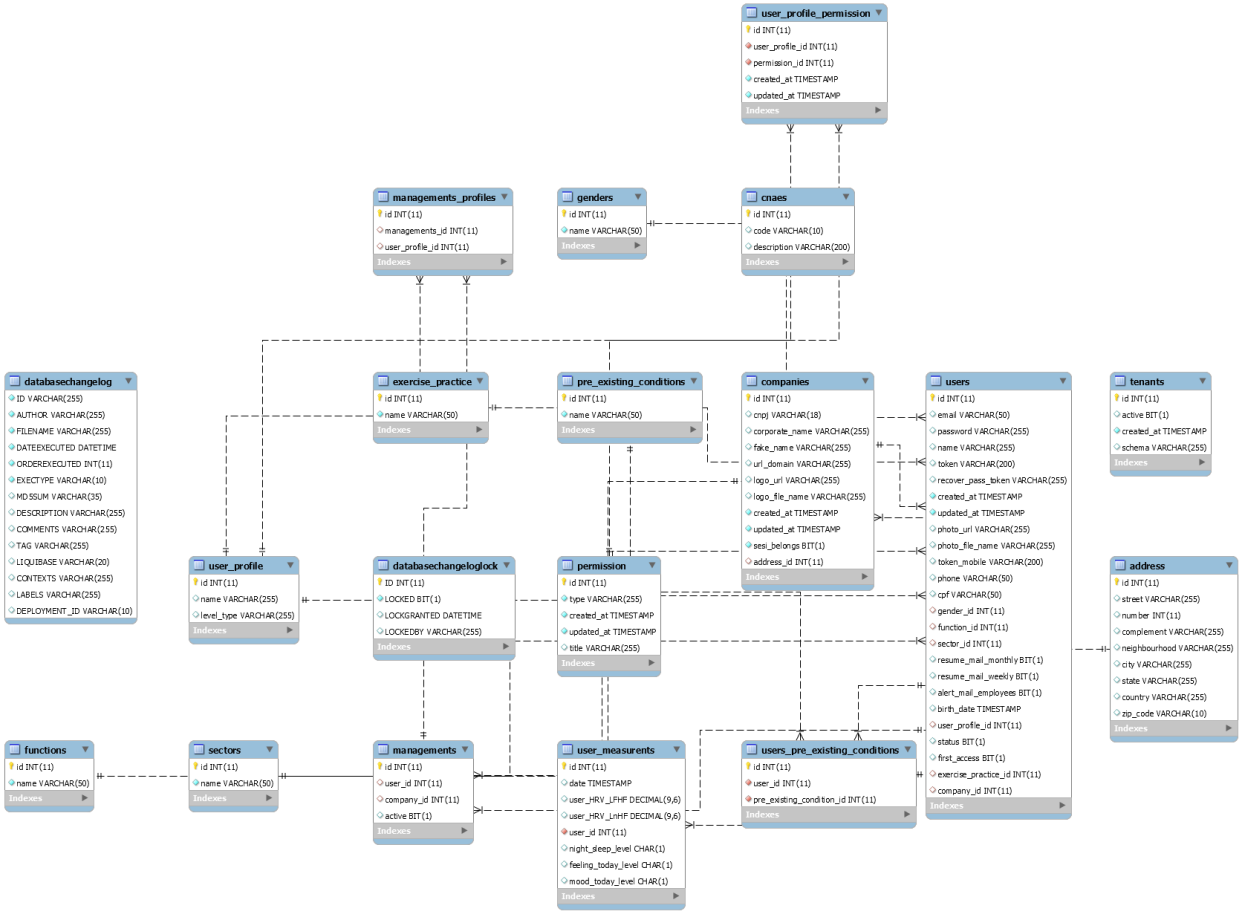


```
-- cnaes
LOCK TABLES `cnaes` WRITE;
/*!40000 ALTER TABLE `cnaes` DISABLE KEYS */;
INSERT INTO `cnaes` VALUES (1,'0111301','Cultivo de arroz'),(2,'0111302','Cultivo de milho'),(3,'0111303','Cultivo de trigo'
/*!40000 ALTER TABLE `cnaes` ENABLE KEYS */;
UNLOCK TABLES;

-- permissions
LOCK TABLES `permission` WRITE;
/*!40000 ALTER TABLE `permission` DISABLE KEYS */;

INSERT INTO permission VALUES (1,...;
/*!40000 ALTER TABLE `permission` ENABLE KEYS */;
UNLOCK TABLES;

-- user profiles
LOCK TABLES `user_profile` WRITE;
/*!40000 ALTER TABLE `user_profile` DISABLE KEYS */;
INSERT INTO `user_profile` VALUES (1...;
/*!40000 ALTER TABLE `user_profile` ENABLE KEYS */;
UNLOCK TABLES;
```

Fonte - Autoral

O banco de dados final é representado pelo diagrama a seguir:

Figura 24 - Diagrama UML do banco de dados



Fonte - Autoral

5.3 INTERFACE DE USUÁRIOS

Planejada em conjunto com a designer da equipe, tem como função interagir com com os diferentes tipos de usuários, se comunicar com backend utilizando API REST para receber dados e então mostrar de forma acessível, intuitiva e agradável para quem estiver usando. As funcionalidades de cadastro, edição e deleção de usuários são implementadas pelos serviços utilizando os métodos HTTP Put, Post e Delete respectivamente e as requisições de dados pelo método Get (figura 25). Ao implementar os métodos HTTP, é necessário realizar tratamento de erros, alinhar com o que o backend espera receber no cabeçalho da mensagem enviando as identificações requeridas.

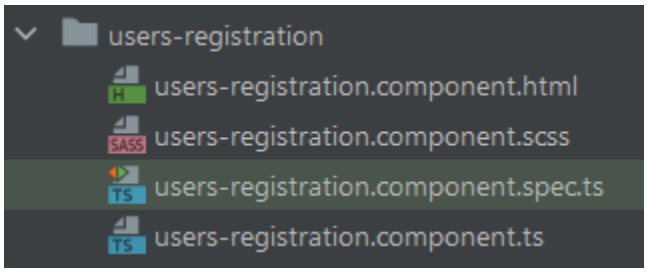
O desenvolvimento do sistema se encontra em sua fase final e portanto alguns recursos ainda estão sendo implementados.

Figura 25 - Método Get do frontend

Fonte - Autoral

O angular utiliza HTML e CSS para mostrar elementos e estilizá-los, respectivamente e typescript para implementar lógicas, fluxos, manipular dados, etc. Cada componente gerado (figura 26) possui um arquivo de cada tipo (e um arquivo .spec.ts usado para testes) já integrados entre si (figura 27).

Figura 26 - Componente gerado em Angular



Fonte - Autoral

Figura 27 - Conexão entre HTML, CSS e Typescript do componente em Angular

```
@Component({
  selector: 'app-users-registration',
  templateUrl: './users-registration.component.html',
  styleUrls: ['./users-registration.component.scss']
})
```

Fonte - Autoral

A camada de apresentação se comunica com a camada de lógica de negócios utilizando serviços (figura 28) onde para cada tipo de ação diferente como requisitar todos os usuários cadastrados no banco de dados (linha 14) ou criar um novo usuário (linha 20) é utilizado um endpoint. Componentes mais complexos tipicamente utilizam vários endpoints para poder realizar suas funções corretamente.

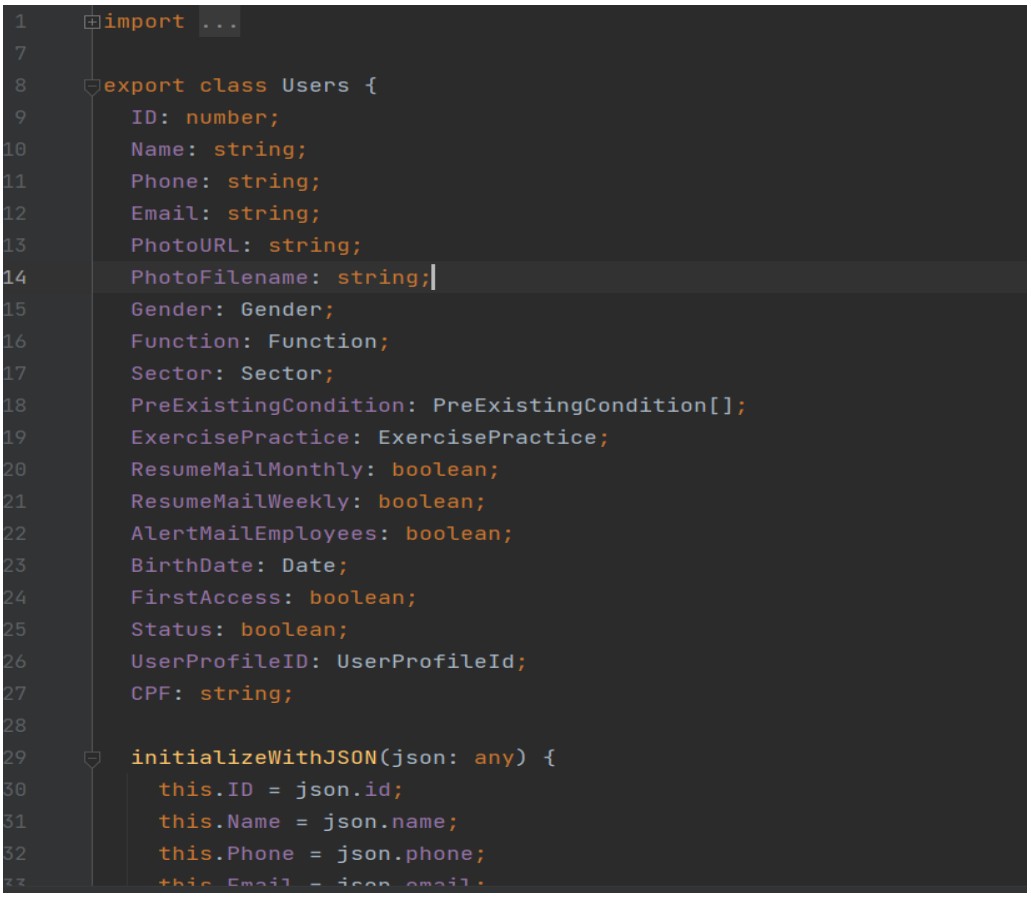
A transferência e manipulação de dados, utiliza classes (figura 29), chamadas aqui de models que, quando necessário, devem ser compatíveis com as classes do backend para que haja integração entre os dois. Ao receber dados, o frontend necessita de realizar validações (figura 30) pois os dados salvos, como data de nascimento, não corresponde ao dados mostrados na tela, como idade, mas existe uma relação (linha 48).

Figura 28 - Service do frontend

```
import {Injectable} from '@angular/core';
import {BaseService} from '../base.service';

@Injectable({
  providedIn: 'root'
})
export class UsersService {

  private endpointAllUsers = '/users';

  constructor(private service: BaseService) {
  }

  async getAllUsers() {
    return await this.service.Get(this.endpointAllUsers).toPromise().then((res: any) => {
      return res;
    });
  }

  async createUser(user: string) {
    return await this.service.Post(this.endpointAllUsers, user).toPromise().then((res: any) => {
      return res;
    });
  }
}
```

Fonte - Autoral

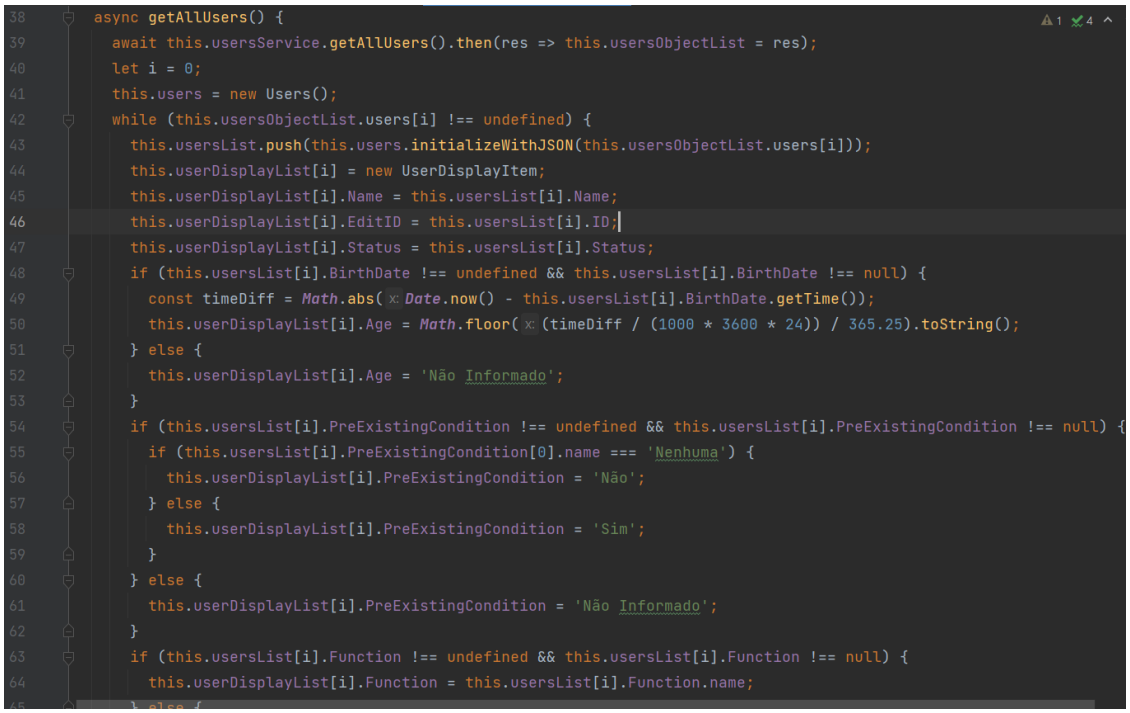
Figura 29 - Model Users



```
import ...

export class Users {
  ID: number;
  Name: string;
  Phone: string;
  Email: string;
  PhotoURL: string;
  PhotoFilename: string;
  Gender: Gender;
  Function: Function;
  Sector: Sector;
  PreExistingCondition: PreExistingCondition[];
  ExercisePractice: ExercisePractice;
  ResumeMailMonthly: boolean;
  ResumeMailWeekly: boolean;
  AlertMailEmployees: boolean;
  BirthDate: Date;
  FirstAccess: boolean;
  Status: boolean;
  UserProfileID: UserProfileId;
  CPF: string;

  initializeWithJSON(json: any) {
    this.ID = json.id;
    this.Name = json.name;
    this.Phone = json.phone;
```

Fonte - Autoral

Figura 30 - Validação realizadas ao buscar receber lista de usuários



```
async getAllUsers() {
  await this.usersService.getAllUsers().then(res => this.usersObjectList = res);
  let i = 0;
  this.users = new Users();
  while (this.usersObjectList.users[i] !== undefined) {
    this.usersList.push(this.users.initializeWithJSON(this.usersObjectList.users[i]));
    this.userDisplayList[i] = new UserDisplayItem;
    this.userDisplayList[i].Name = this.usersList[i].Name;
    this.userDisplayList[i].EditID = this.usersList[i].ID;
    this.userDisplayList[i].Status = this.usersList[i].Status;
    if (this.usersList[i].BirthDate !== undefined && this.usersList[i].BirthDate !== null) {
      const timeDiff = Math.abs( x: Date.now() - this.usersList[i].BirthDate.getTime());
      this.userDisplayList[i].Age = Math.floor( x: (timeDiff / (1000 * 3600 * 24)) / 365.25).toString();
    } else {
      this.userDisplayList[i].Age = 'Não Informado';
    }
    if (this.usersList[i].PreExistingCondition !== undefined && this.usersList[i].PreExistingCondition !== null) {
      if (this.usersList[i].PreExistingCondition[0].name === 'Nenhuma') {
        this.userDisplayList[i].PreExistingCondition = 'Não';
      } else {
        this.userDisplayList[i].PreExistingCondition = 'Sim';
      }
    } else {
      this.userDisplayList[i].PreExistingCondition = 'Não Informado';
    }
    if (this.usersList[i].Function !== undefined && this.usersList[i].Function !== null) {
      this.userDisplayList[i].Function = this.usersList[i].Function.name;
```

Fonte - Autoral

Como dito anteriormente, para mostrar elementos, dados, é utilizado HTML (figura 31) que pode acessar os dados de algumas formas, como interpolation, property binding, class binding, style binding, attribute binding, event binding, two-way binding. No projeto é utilizado majoritariamente two-way binding onde as mudanças da view (HTML) alteram o model (Typescript) e vice e versa, event binding que permite ouvir e responder às ações do usuário (figura 32), como pressionamentos de tecla, movimentos do mouse, cliques e toques e class binding que permite estilizar os elementos HTML com classes do CSS (figura 33).

Figura 31 - HTML

```html
<div class="form-row">
  <div class="input-field">
    <label>Nome completo</label>
    <input #inputName #nameValidationVariable="ngModel" (change)="inputNameValidation()"
           [(ngModel)]="inputNameValidator"
           placeholder="{{placeHolderName}}" required type="text">
    <p *ngIf="!validName && nameValidationVariable.touched" class="error-message">Insira um nome válido.</p>
  </div>
  <div class="input-field">
    <label>Data de nascimento</label>
    <input #birthValidationVariable="ngModel" #inputBirth (change)="inputBirthValidation()"
           [(ngModel)]="inputBirthValidator"
           placeholder="{{placeHolderBirth}}" required type="date">
    <p *ngIf="birthValidationVariable.invalid && birthValidationVariable.touched" class="error-message">
      Insira uma data de nascimento válida.</p>
  </div>
  <div class="input-field">
    <label>Gênero</label>
    <select #selectGender>
      <option disabled selected value="">Selecione</option>
      <option value="1">Feminino</option>
      <option value="2">Masculino</option>
      <option value="3">Prefiro não informar</option>
    </select>
  </div>
</div>
```

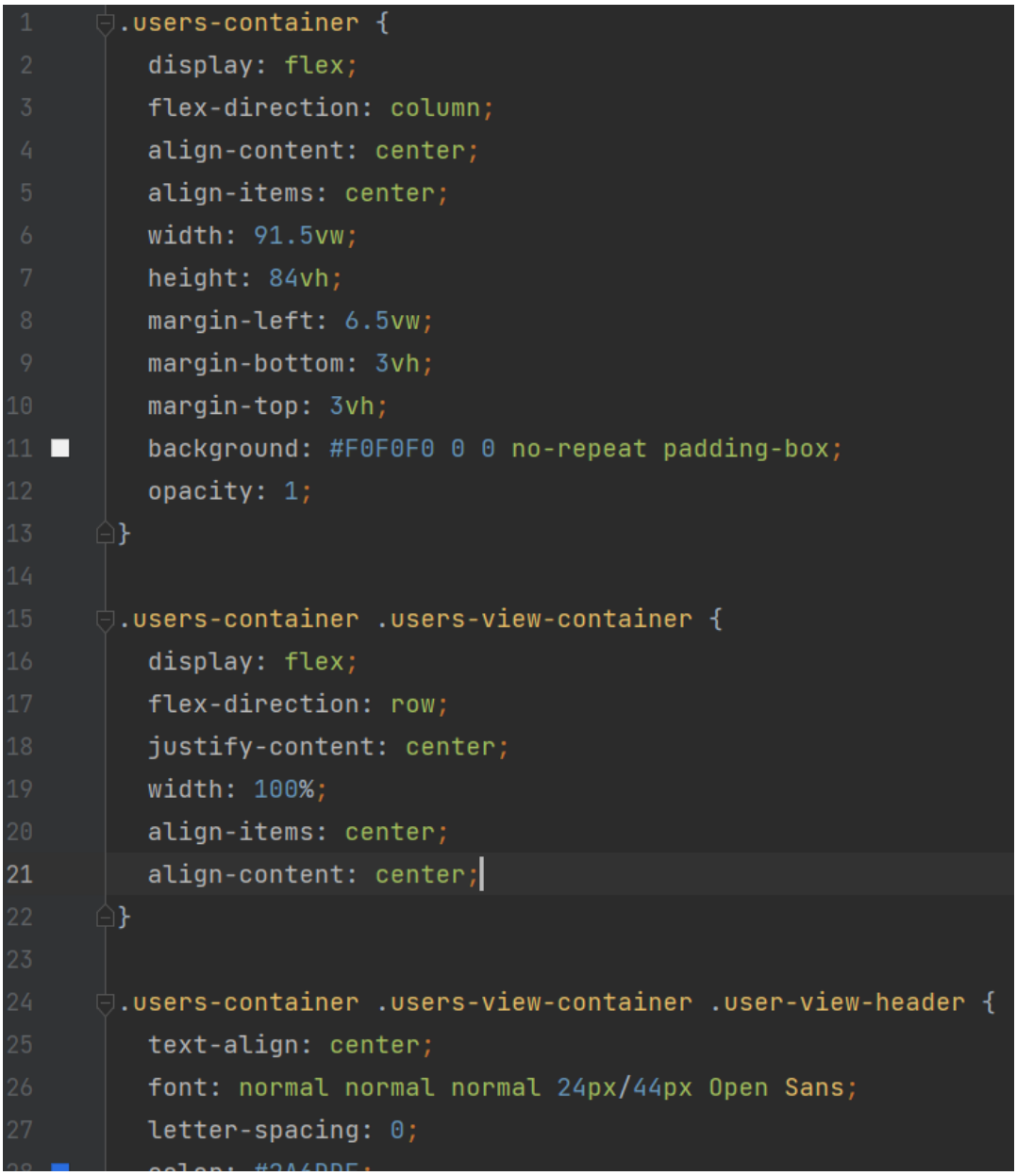
Fonte - Autoral

Figura 32 - Event binding

```
(change)="inputNameValidation()"
```

Fonte - Autoral

Figura 33 - CSS



```css
.users-container {
  display: flex;
  flex-direction: column;
  align-content: center;
  align-items: center;
  width: 91.5vw;
  height: 84vh;
  margin-left: 6.5vw;
  margin-bottom: 3vh;
  margin-top: 3vh;
  background: #F0F0F0 0 0 no-repeat padding-box;
  opacity: 1;
}

.users-container .users-view-container {
  display: flex;
  flex-direction: row;
  justify-content: center;
  width: 100%;
  align-items: center;
  align-content: center;
}

.users-container .users-view-container .user-view-header {
  text-align: center;
  font: normal normal normal 24px/44px Open Sans;
  letter-spacing: 0;
```

Fonte - Autoral

Cuidados especiais são necessários durante a implementação de componentes no que se refere a responsividade da aplicação, por este motivo, sempre que possível, é recomendável utilizar distâncias dinâmicas ao invés de distâncias fixas.

Porcentagens em CSS são relativas ao elemento-ancestral mais próximo enquanto a medida vh é igual a 1/100 da altura da viewport. Então, por exemplo, se a

altura do navegador é 900px, 1vh equivale a 9px e, analogamente, se a largura da viewport é 750px, 1vw equivale a 7.5px.

Mesmo tomando esses cuidados, tamanhos de telas muito diferentes podem causar efeitos indesejáveis na aplicação, desconfigurando os elementos, para resolver este problema é utilizado media queries que consiste em uma técnica CSS introduzida no CSS3 incluindo um bloco de propriedades apenas se uma determinada condição for verdadeira, como tamanho de tela menor ou maior que um determinado tamanho, por exemplo (figura 34).

Figura 34 - Media query

```css
@media (min-width: 900px) {
  .login-page-processos {
    width: 100%;
    height: 100%;
    display: flex;
    justify-content: flex-end;
    background-color: #F0F0F0;
  }
  .logo-background > img {
    width: 50vw;
    height: 100%;
  }

  .white-box {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    height: 100%;
    width: 100%;
  }
  .login-card {
    display: grid;
    justify-items: center;
    justify-self: center;
    vertical-align: middle;
    box-shadow: none;
```

Fonte - Autoral

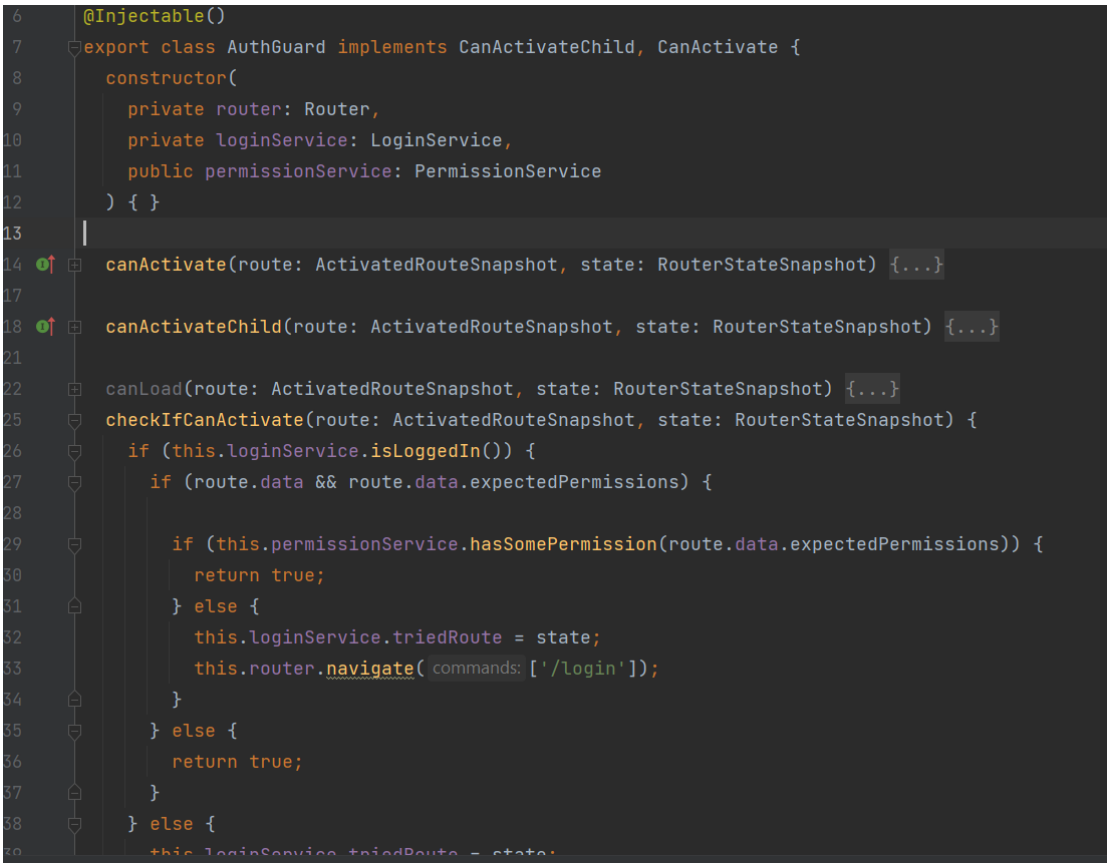
A navegação entre componentes do sistema é implementada através de um módulo de rotas (figura 35) que checa se o usuário está logado e suas permissões antes que este possa acessar a tela (figura 36). Na parte do frontend, ao realizar o login (figura

37), o sistema checa o tipo de usuário e se é a primeira vez que este usuário entra no sistema para direcioná-lo às telas corretas.

Figura 35 - Rotas de componentes no frontend

Fonte - Autoral

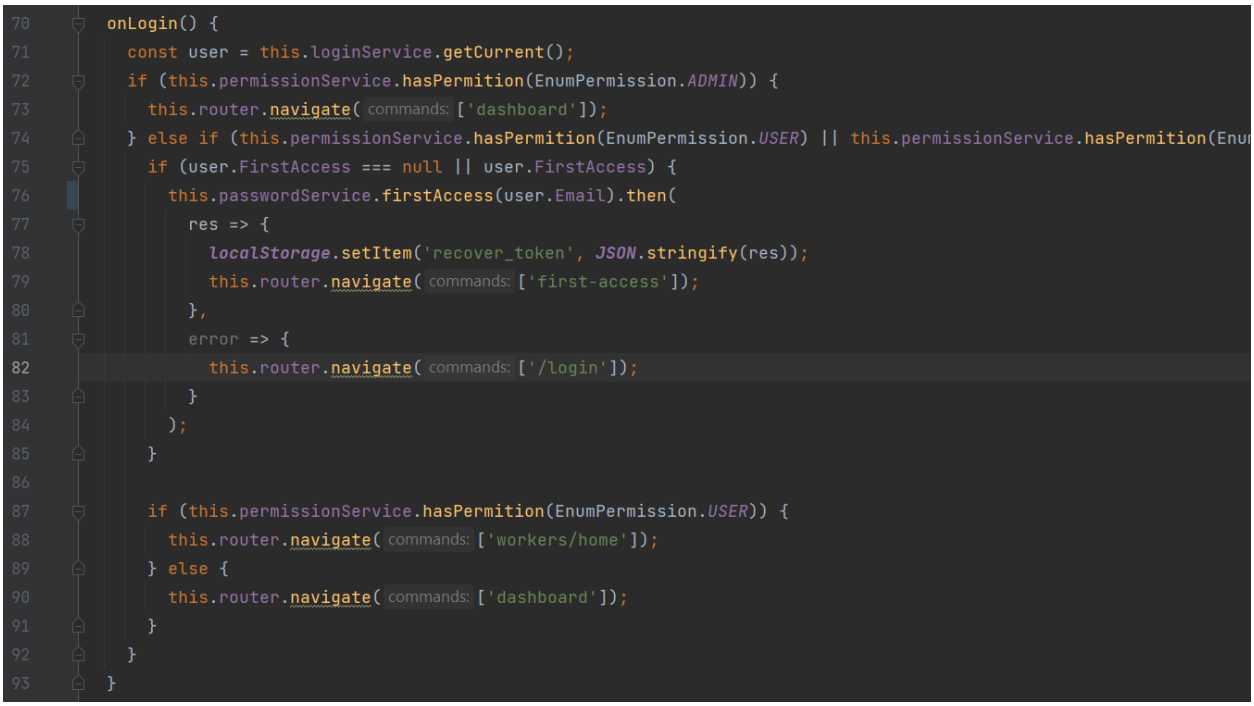
Figura 36 - Checagem de permissões de usuário



```
@Injectable()
export class AuthGuard implements CanActivateChild, CanActivate {
  constructor(
    private router: Router,
    private loginService: LoginService,
    public permissionService: PermissionService
  ) { }

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {...}

  canActivateChild(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {...}

  canLoad(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {...}
  checkIfCanActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {
    if (this.loginService.isLoggedIn()) {
      if (route.data && route.data.expectedPermissions) {

        if (this.permissionService.hasSomePermission(route.data.expectedPermissions)) {
          return true;
        } else {
          this.loginService.triedRoute = state;
          this.router.navigate( commands: ['/login']);
        }
      } else {
        return true;
      }
    } else {
```

Fonte - Autoral

Figura 37 - Checagem do tipo de usuário ao realizar login

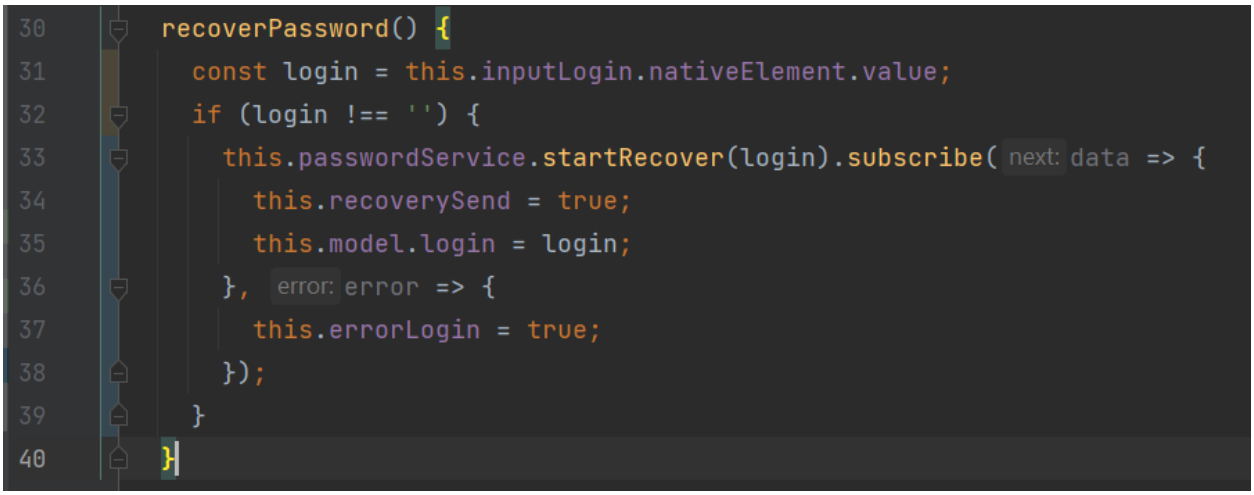


```
70  onLogin() {
71    const user = this.loginService.getCurrent();
72    if (this.permissionService.hasPermition(EnumPermission.ADMIN)) {
73      this.router.navigate( commands: ['dashboard']);
74    } else if (this.permissionService.hasPermition(EnumPermission.USER) || this.permissionService.hasPermition(Enu
75      if (user.FirstAccess === null || user.FirstAccess) {
76        this.passwordService.firstAccess(user.Email).then(
77          res => {
78            localStorage.setItem('recover_token', JSON.stringify(res));
79            this.router.navigate( commands: ['first-access']);
80          },
81          error => {
82            this.router.navigate( commands: ['/login']);
83          }
84        );
85      }
86
87      if (this.permissionService.hasPermition(EnumPermission.USER)) {
88        this.router.navigate( commands: ['workers/home']);
89      } else {
90        this.router.navigate( commands: ['dashboard']);
91      }
92    }
93  }
```

Fonte - Autoral

Atendendo aos requisitos do sistema, foi implementado também a funcionalidade de recuperação de senha (figura 38) que envia um e-mail para o usuário, gestor ou administrador responsável (caso o usuário não tenha e-mail cadastrado) utilizando novamente serviços (linha 33).

Figura 38 - Funcionalidade de recuperar senha



```
30  recoverPassword() {
31    const login = this.inputLogin.nativeElement.value;
32    if (login !== '') {
33      this.passwordService.startRecover(login).subscribe( next: data => {
34        this.recoverySend = true;
35        this.model.login = login;
36      }, error: error => {
37        this.errorLogin = true;
38      });
39    }
40  }
```

Fonte - Autoral

Por fim, alguns componentes como resumo da coleta e tela das coletas do dia do colaborador e partes de funcionalidades do dashboard do gestor ainda estão sendo desenvolvidos.

5.3.1 COLABORADORES

Neste capítulo será explicado o fluxo de telas do colaborador que deve ser primeiramente pré cadastrado pelo gestor responsável, sendo obrigatório o preenchimento do nome do usuário, data de nascimento e CPF sendo a primeira senha por padrão a data de nascimento do colaborador.

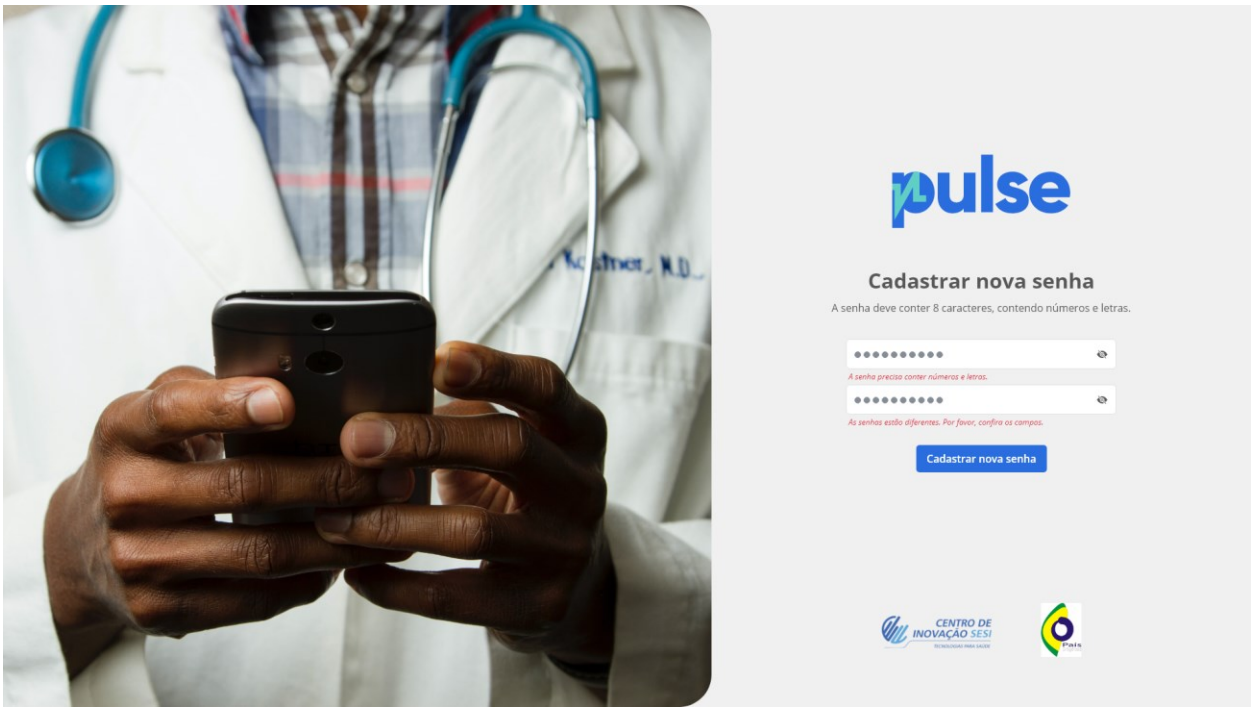
Após ser cadastrado pelo gestor o usuário pode realizar seu primeiro acesso (figura 39) onde o ele deve escolher uma nova senha. O login pode ser realizado tanto com e-mail quanto CPF para casos onde o trabalhador não possui e-mail.

Figura 39 - Tela de primeiro acesso

Fonte - Autoral

Neste primeiro acesso é checado se a senha contém caracteres e números para aumentar a efetividade da senha (figura 40). Assim como se ambas as senhas inseridas são iguais.

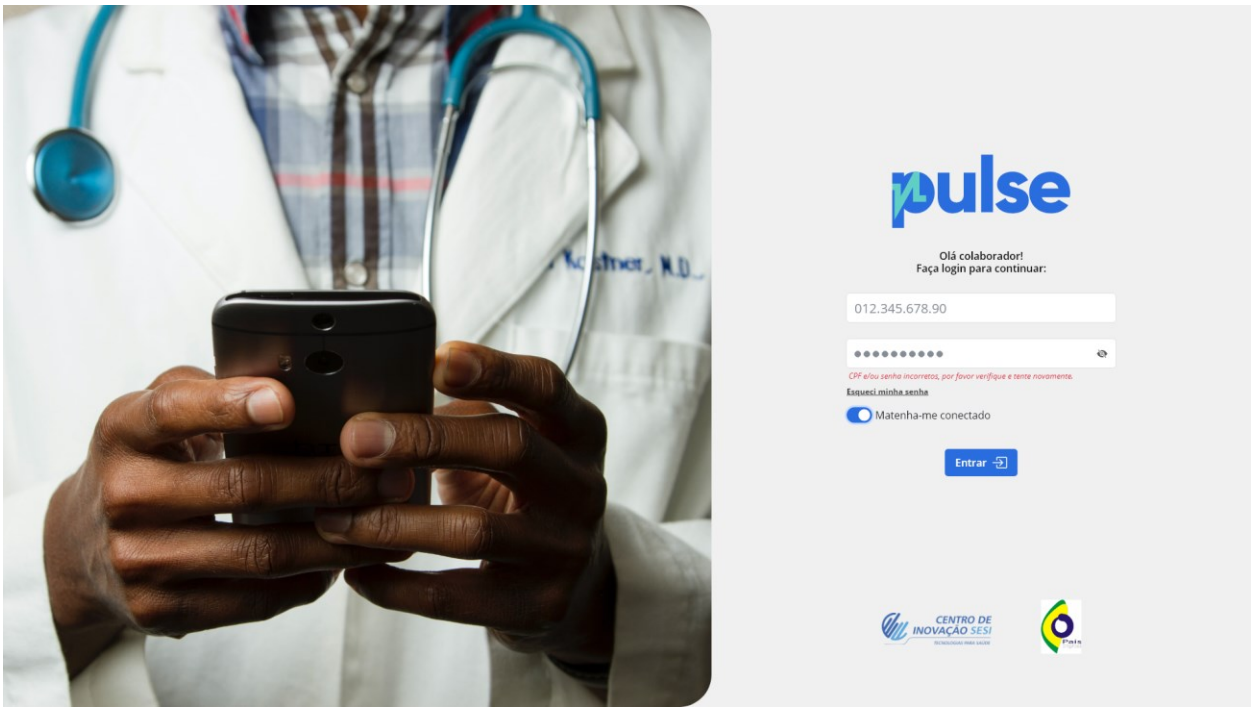
Figura 40 - Erro mostrado nas validações



Fonte - Autoral

Desde o primeiro até futuros acessos, com senha padrão ou já escolhida pelo usuário, sempre é realizada validação no banco de dados para checar se senha e usuário estão corretos (figura 41).

Figura 41 - Erro caso usuário ou senha não sejam válidos

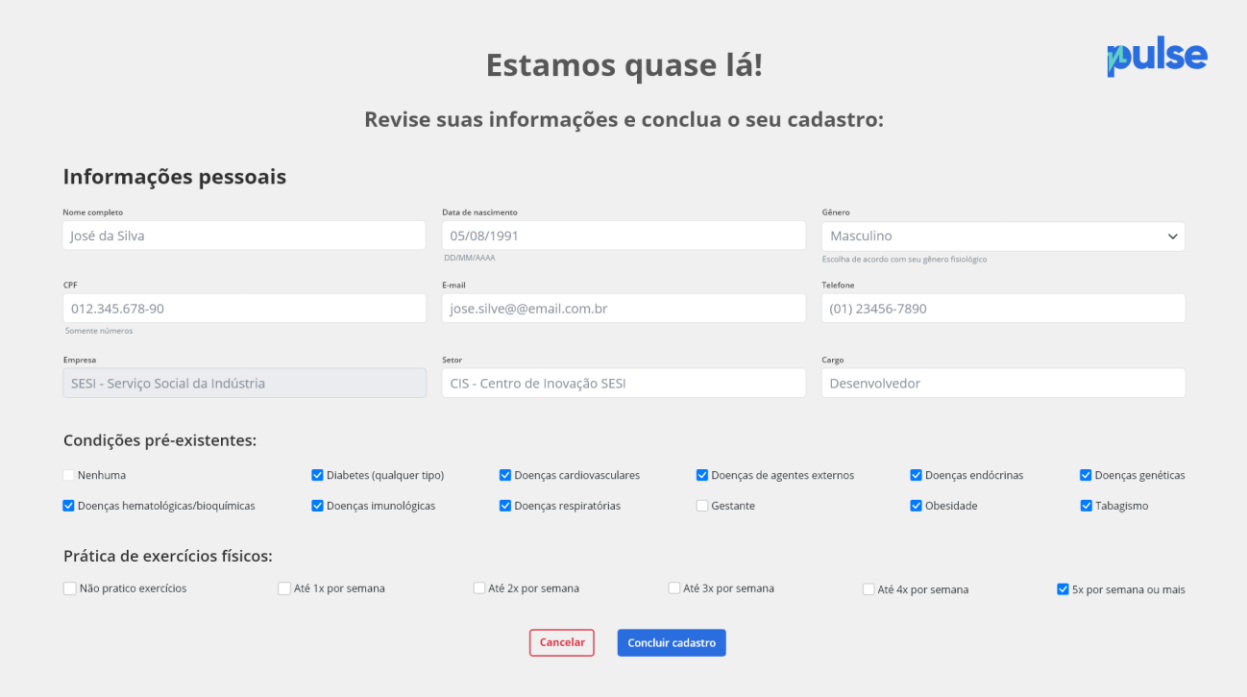


Fonte - Autoral

Após a validação da senha o colaborador é redirecionado para a tela de primeiro cadastro (figura 42) onde este deve terminar seu cadastro, o fluxo foi feito desta forma levando em consideração que o gestor pode não ter acesso ou conhecimento de todas as informações de seu funcionário. Portanto, o cadastro realizado pelo próprio funcionário tem mais campos obrigatórios.

Foram definidas também algumas regras de validações na seleção de condições pré-existentes e prática de exercício físicos, no segundo, por exemplo, apenas um campo pode ser selecionado enquanto no primeiro podem haver múltiplos campos selecionados que devem ser enviados para o backend na forma de array de condições pré-existentes, porém, devem ser desselecionados caso o usuário selecione o primeiro elemento (nenhum).

Figura 42 - Formulário de cadastro no primeiro acesso



Fonte - Autoral

Após cadastro o colaborador é redirecionado para uma tela inicial onde, caso não seja seu primeiro acesso, pode ver o resumo de suas medições e está apto a iniciar medições, para tal necessita apenas clicar em um botão (figura 43).

Figura 43 - Tela de boas vindas do usuário.

Fonte - Autoral

Ao clicar no botão aparecerão algumas recomendações que servem como guia mostradas a seguir:

Figura 44 - Recomendação do posicionamento dos eletrodos

Fonte - Autoral

Figura 45 - Recomendação geral de como realizar a captação

Fonte - Autoral

Durante a medição o usuário permanece em uma tela com algumas instruções e um GIF para auxiliá-lo como na imagem a seguir:

Figura 46 - Tela de espera enquanto realiza medição

Fonte - Autoral

Ao completar a captação o sistema automaticamente redireciona o colaborador a tela de conclusão a seguir:

Figura 47 - Tela de conclusão de coleta

Fonte - Autoral

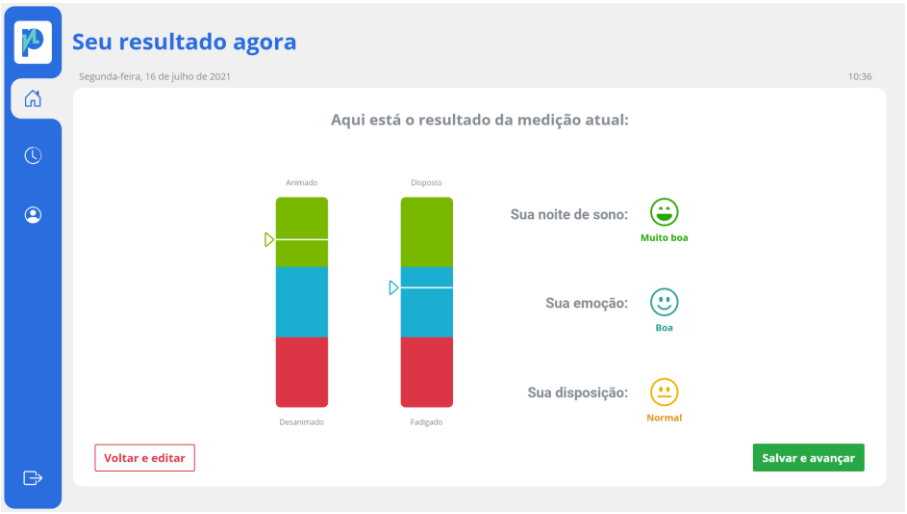
O usuário deve então responder um rápido questionário como ilustrado a seguir:

Figura 48 - Questionário

Fonte - Autoral

Por fim é exibido um feedback (figura 49) onde as barras representam os dados coletados e os emojis às respostas do formulário.

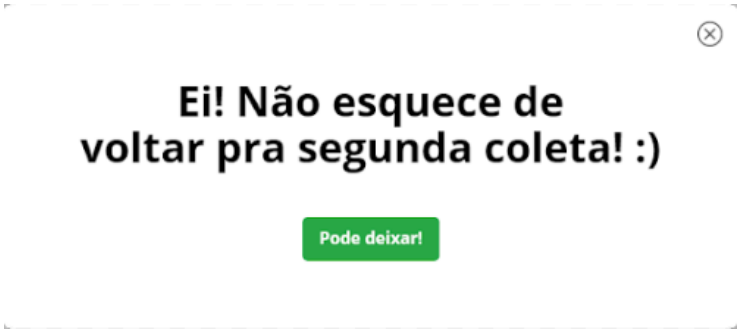
Figura 49 - Resumo da coleta



Fonte - Autoral

Ao finalizar a primeira medição do dia é exibido um lembrete de que são duas medições diárias como visto na figura a seguir:

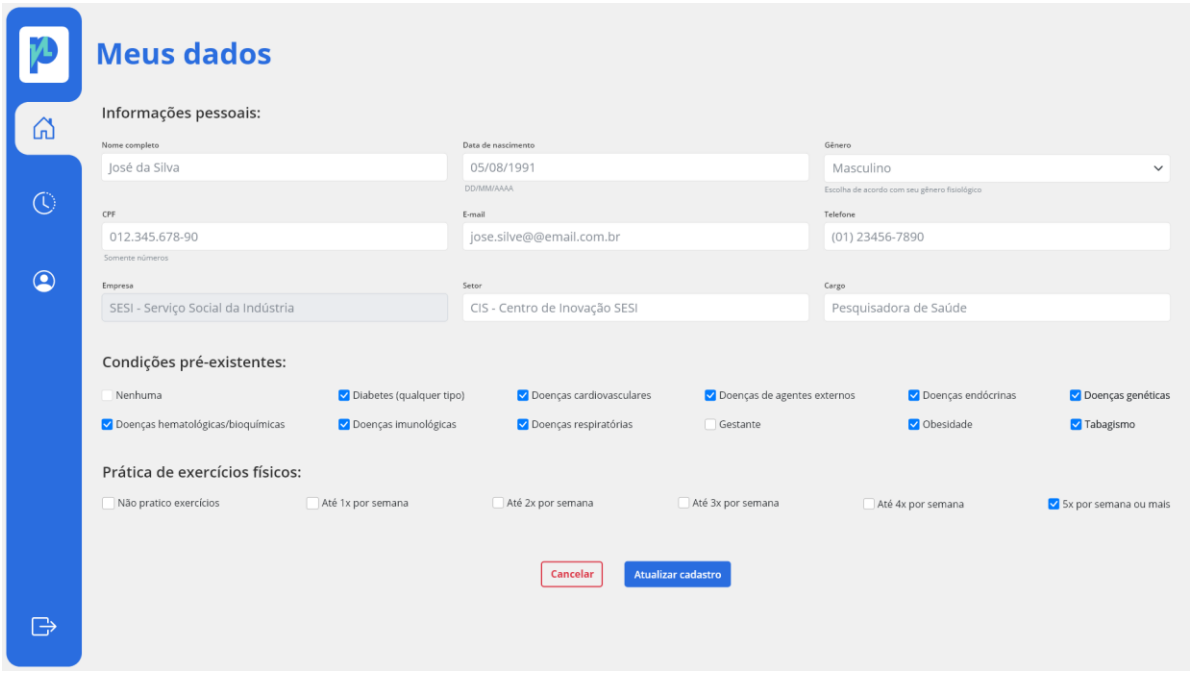
Figura 50 - Lembrete



Fonte - Autoral

Fora as medições diárias, o colaborador pode ver e editar seus dados (figura 51) e, caso já tenha realizado as medições, ver seu resultado diário (figura 52).

Figura 51 - Tela de edição de dados



Fonte - Autoral

Figura 52 - Tela das coletas do dia

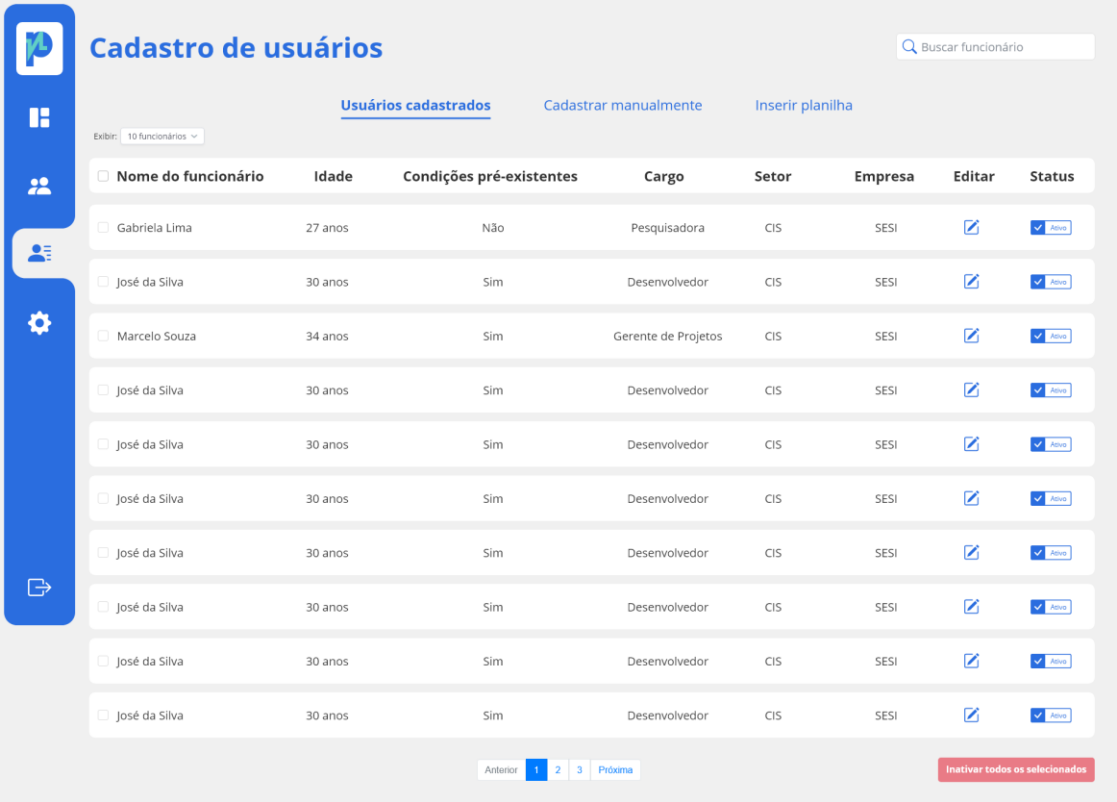
Fonte - Autoral

5.3.2 GESTORES

Como descrito anteriormente, gestores são cadastrados por administradores, e passam pelo mesmo processo em seu primeiro login, também tendo que cadastrar nova senha, podendo recuperar senha e necessitando de preencher o formulário com seus dados no primeiro acesso.

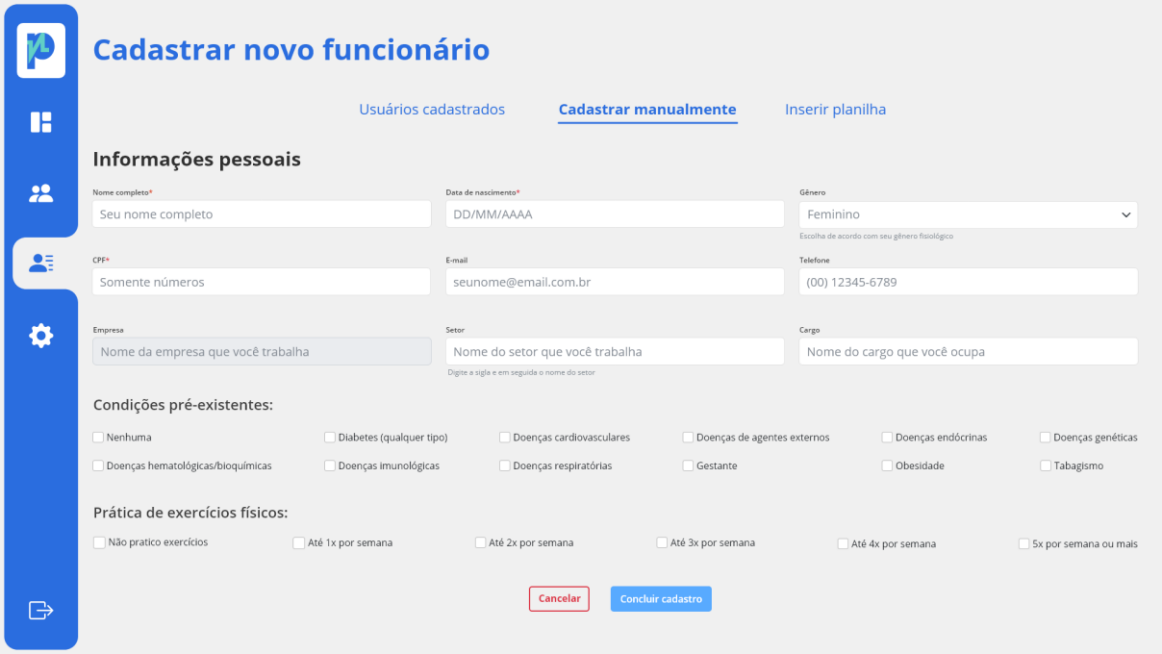
Começando funcionalidades diferentes do usuário gestor, podemos citar a visualização dos colaboradores já cadastrados (figura 53) e a inserção de novos funcionários, como previamente explicado e exemplificado na figura 54.

Figura 53 - Lista de usuários cadastrados



Fonte - Autoral

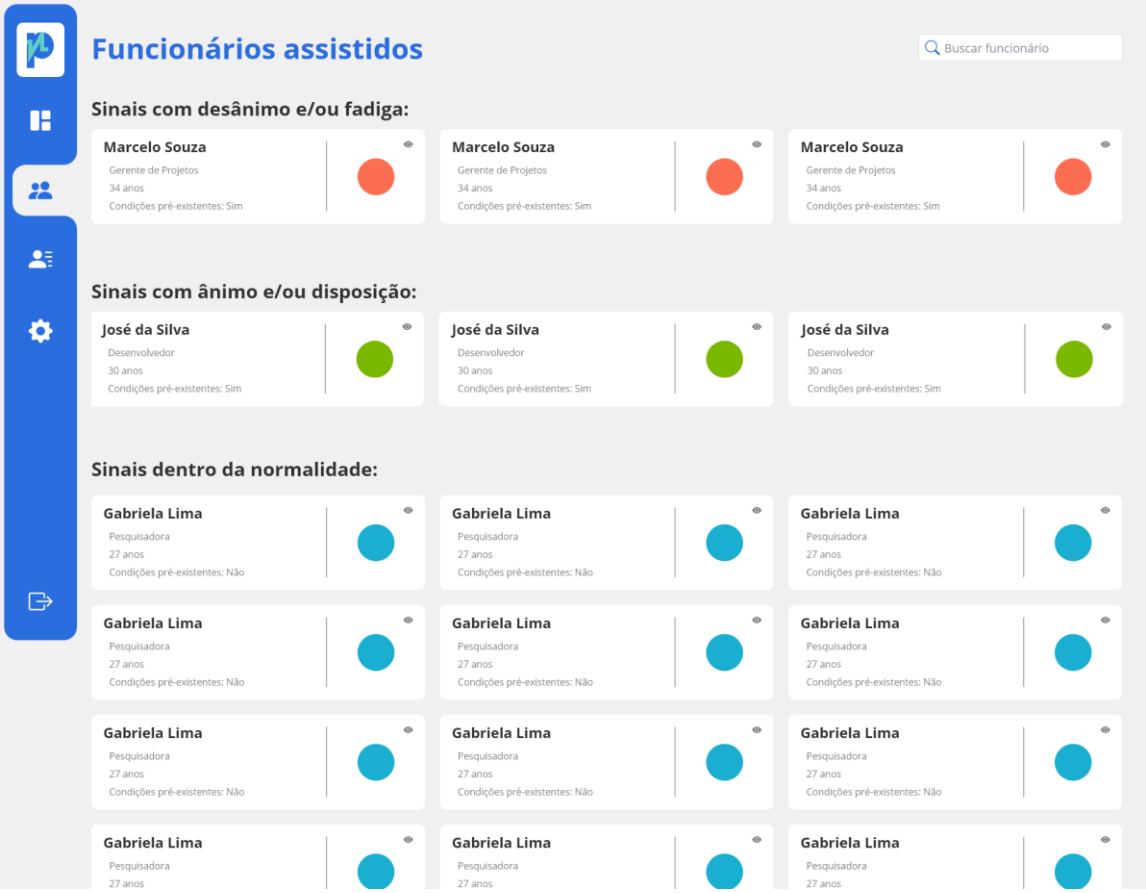
Figura 54 - Castro de novo funcionário



Fonte - Autoral

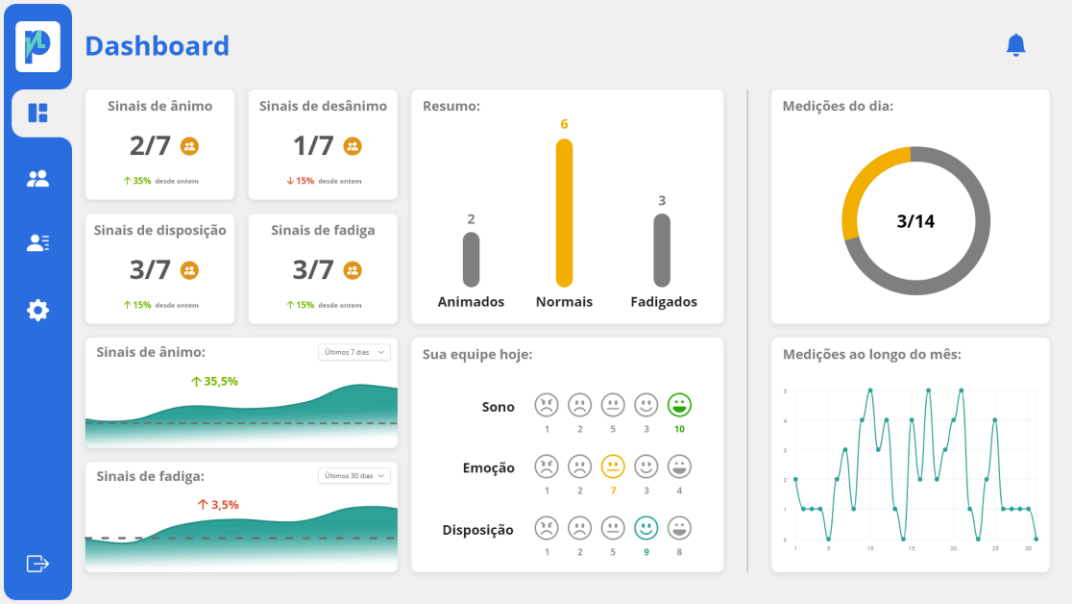
O gestor tem acesso a informações dos funcionários monitorados no dia (figura 55) e um dashboard mostrando o histórico de medições da sua equipe e alguns indicadores (figura 56).

Figura 55 - Lista de usuários separados por sinais



Fonte - Autoral

Figura 56 - Dashboard do gestor



Fonte - Autoral

É possível também acompanhar mais de perto funcionários específicos para entender melhor suas condições como exemplificado na figura a seguir:

Figura 57 - Tela de acompanhamento do usuário

Fonte - Autoral

6 ANÁLISE DE RESULTADOS

Como este projeto de conclusão de curso é parte de um projeto ainda em andamento, algumas partes como a finalização do desenvolvimento bem como a aplicação do piloto na empresa C-Pak, não foram contempladas. Porém, testes preliminares atenderam as especificações de projeto e testes de bancada realizados mostraram o desempenho adequado do protótipo.

As estratégias escolhidas para execução do projeto foram adequadas no decorrer e se mostraram aderentes aos objetivos propostos. A seleção da variabilidade da frequência cardíaca (VFC) como parâmetro fisiológico a ser medido correlacionado com a funcionalidade de classificação do sono/emoção/disposição se mostraram correlatas nos testes iniciais.

A interface implementa, conforme regras de negócio e design das telas, três perfis de usuário onde o colaborador pode realizar login, resetar a senha, fazer a coleta, responder questionário e visualizar suas informações, enquanto o gestor tem acesso aos recursos de login, resetar senha e visualização das informações traduzidas dos valores capturados de toda a equipe de funcionário através de dashboard, sendo possível também o acompanhamento individual para melhor entendimento da situação de cada trabalhador. Além de se comunicar com o backend via protocolo HTTP utilizando API REST permitindo cadastro, edição e deleção de usuários e gerentes.

O camada de lógica do negócio realiza troca de informações de login, armazenando o token de autenticação e sessões de usuário do banco de dados permitindo cadastro, edição e deleção dos mesmos e a realização de queries via SQL retornando informações para a camada de apresentação. Realiza também comunicação com o software de captação de dados salvando os dados das coletas no banco de dados. Todas as interações possuem medidas de segurança e checam permissões do usuário logado.

7 CONCLUSÃO

Como citado anteriormente, os resultados aqui apresentados neste projeto de conclusão de curso fazem parte de um projeto ainda em andamento, algumas partes como a finalização do desenvolvimento bem como a aplicação do piloto na empresa C-Pak, ainda não foram contempladas.

Através de testes internos utilizando indivíduos das empresas parceiras CIS e País Digital foi possível uma avaliação inicial demonstrando o potencial de aplicação da tecnologia desenvolvida.

O projeto pretende melhorar as condições de trabalho dos colaboradores, elevando da sua qualidade de vida e aumentando a produtividade gerando ganhos econômicos para as empresas e impacto tecnológico considerando a baixa quantidade de ferramentas tecnológicas em atuação na área de SST, muito menos voltada a acompanhar os funcionários diariamente e permitir ações antes que os incidentes ocorram. Por isso o foco na experiência do usuário, ferramenta simples e de poucos passos.

O grande diferencial do projeto é o foco em conhecer os trabalhadores e reconhecer (talvez até mesmo antes deles mesmos) mudanças sutis nos seus padrões que podem indicar sinais de fadiga.

Para a empresa o projeto apresenta uma vasta quantidade de oportunidades e já pretende ser utilizado como base em outra solução, o Safe B, adaptando os mesmos princípios para trabalho em altura.

Soluções tecnológicas para assistir os trabalhadores como essas se tornarão cada vez mais comuns e, captando esses dados e mantendo um banco, possibilitará num futuro diversos outros tipos de análises e correlações.

REFERENCIAS

ALBINO, I &. CONDE, E (2019). Revisão Sistemática: Instrumentos para avaliação do estresse e ansiedade em jogadores de futebol. Revista Brasileira de Psicologia do Esporte, Brasília, v. 9, n° 1, março 2019

ALMEIDA, Guilherme G. O Gerenciamento de Risco de Fadiga Humana e a Lei do Aeronauta. Disponível em: < https://jus.com.br/artigos/78642

BOSQUET, L.; MERKARI, S.; ARVISAIS, D.; AUBERT, A.E. Is heart rate a convenient tool to monitor over-reaching? A systematic review of the literature. British Journal of Sports Medicine, v.42, p.709-714, 2008

BRASIL. Ministério da Defesa, Comando da Aeronáutica. ICA-38 Indicadores de Desempenho da Segurança Operacional no SISCEAB. Brasília, DF. 2015. Disponível em: https://publicacoes.decea.gov.br/download.cfm?d=4287

DISHMAN RK, NAKAMURA Y, GARCIA ME, THOMPSON RW, DUNN AL, BLAIR SN (2000). Heart rate variability, train anxiety, and perceived stress among physically fit men and woman. Int J Psychophysiol, 37(2):121-33.

PINHEIRO, Paulo Sérgio Teixeira. O risco da fadiga humana na aviação civil: um estudo dos impactos na aviação comercial brasileira. Ciências Aeronáuticas-Unisul Virtual, 2020.

PUMPRLA J, HOWORKA K, GROVES D, CHESTER M & NOLAN J. (2002) Functional assessment of heart rate variability: physiological basis and practical applications. Int J Cardiol. 2002;84(1):1-14.

QUINTINO, Willian Soares. SANTOS, Roberto Márcio dos. Os riscos da fadiga humana

para a segurança operacional de voo. Revista Científica Multidisciplinar Núcleo do Conhecimento. Ano 05, Ed. 09, Vol. 04, pp. 18-34. Setembro de 2020

SEAWARD, B. L. Stress: aprenda a lidar com as tensões do dia-a-dia e melhore sua qualidade de vida. 6. ed. São Paulo: Editora Novo Conceito, 2009.

SOUSA, A. F. Estresse ocupacional em motoristas de ônibus urbanos: o papel das estratégias de coping. Salvador, 2005. Dissertação de Mestrado em Psicologia.Universidade Federal da Bahia. Disponível em http://www.pospsi.ufba.br/Aldineia_Sousa.pdf. Acesso em 05 abril de 2016.

VANDERLEI, L.C.M., SILVA, R.A., PASTRE, C.M., AZEVEDO, F.M., & GODOY, M.F. Comparison of the Polar S810i monitor and the ECG for the analysis of heart rate variability in the time and frequency domains. Polar S810i RR series for HRV analysis. Brazilian Journal of Medical and Biological Research, v.41, p.854-859, 2008.

VANDERLEI, L.C.M.; PASTRE, C.M.; HOSHI, R.A.; CARVALHO, T.D.; GODOY, M. Noções básicas de variabilidade da frequência cardíaca e sua aplicabilidade clínica. Revista Brasileira de Cirurgia Cardiovascular, v.24, n.2, p.205-207, 2009.