UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

CINTIA MARQUES VALENTE

Sistema para cadastramento e orientação de exames para pacientes

Monografia apresentada como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação

Orientador: Prof. Dr. Dennis Giovani Balreira

Porto Alegre
2024

RESUMO

Os exames de rotina tem o importante papel de monitorar as funções corporais, antecipando a proteção de uma série de condições. Este trabalho apresenta o desenvolvimento de um sistema que consiste em uma aplicação web de cadastramento e orientação médica. O principal objetivo é facilitar as informações referentes a exames de rotina, incentivando sua realização, através de um sistema de cadastramento dos seguintes dados: nome, sexo e idade. A implementação desta aplicação é realizada, no Backend, em linguagem C Sharp, utilizando o framework .Net. Para a persistência de dados é utilizado um banco de dados relacional PostgreSQL, implementado através do Docker, plataforma de execução de aplicativos em contêiners. Para os testes unitários, foi utilizada a biblioteca xUnit. No Frontend, a implementação é realizada em linguagem TypeScript, utilizando o framework Angular. Foram realizados testes com os usuários para validação e avaliação da utilidade e usabilidade, a partir da validação foram identificados resultados positivos, principalmente no quesito utilidade.

Palavras-chave: Engenharia de Sofware. Aplicação. Exames médicos.

Registration and guidance system examinations for patients

ABSTRACT

Routine exams have the important role of monitoring bodily functions, anticipating protection from a series of conditions. However, studies show that less than half of Brazilians undergo routine exams. This work presents the development of a web system that consists of a desktop application for registration and medical guidance. The main objective is to facilitate information regarding routine examinations, through a system for registering the following data: name, age and sex. The implementation of this application is carried out, in the Backend, in C Sharp language, using the .Net framework. For data persistence, a PostgreSQL relational database is used, implementing it through Docker, a platform for running applications in containers. For unit tests, the xUnit library was used. In Frontend, implementation is carried out in TypeScript language, using the Angular framework. Have been tested with users for validation and evaluation of usefulness and usability, positive results were identified from validation, mainly in terms of usefulness.

Keywords: Software Engineering. Application. Medical exams.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUS	Sistema Único de Saúde
LDL	Low-density lipoprotein
HDL	High-density lipoprotein
VLDL	Very low-density lipoprotein
HPV	Human Papiloma Virus
AVC	Acidente vascular cerebral
PSA	Prostate-Specific Antigens
TSH	Thyroid Stimulanting Hormone
T4	Thyroxine
ID	Identification
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
IDE	Integrated Development Environment
DDD	Domain Driven Design
API	Application Programming Interface
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
XML	Extensible Markup Language
SVG	Scalable Vector Graphics
MathML	Mathematical Markup Language
XHTML	Extensible Hypertext Markup Language
SACV	Sistema de Agendamento de Clínica Veterinária
JSON	JavaScript Object Notation
UML	Unified Modeling Language

DTO	Data Transfer Objects
HTTP	Hypertext Transfer Protocol
CRUD	Create, Read, Update, Delete
URL	Uniform Resource Locator

SUMÁRIO

1 INTRODUÇÃO

Exames de rotina são fundamentais na prevenção e detecção precoce de doenças. Muitas condições, quando identificadas em estágios iniciais, aumentam significativamente as perspectivas de completa recuperação. Exames de rotina nos permitem agir proativamente em relação à nossa saúde, não só combatendo doenças mais complexas, mas também assumindo o controle de fatores de risco potencialmente prejudiciais (HOSPITAL, 2023). Com base em decisões informadas, possibilitadas pelos resultados dos exames, podemos realizar ajustes no estilo de vida, hábitos alimentares e incorporar rotinas de exercícios, que não apenas contribuem para a prevenção de doenças, como também para uma vida mais plena e ativa. Porém, estudos mostram que menos da metade dos brasileiros checam como está sua saúde através de exames de rotina (POPULAR, 2023). Muitas pessoas buscam atendimento apenas em situações de urgência ou por questões de doenças já estabelecidas. Diversos fatores podem contribuir para isso, como a falta de conscientização sobre a importância dos exames e limitações financeiras para realizar alguns procedimentos. Este estudo tem como objetivo facilitar as informações referentes a exames, tanto no setor privado quanto no Sistema Único de Saúde (SUS) e estimular sua realização através de um sistema de cadastramento e de orientação médica. Espera-se que os resultados obtidos contribuam para uma maior propagação de informações e para o estímulo à realização de exames de rotina, ajudando a promover a cultura da prevenção e o cuidado contínuo com a saúde.

1.1 Organização

Esse trabalho está estruturado em um total de seis capítulos. Este capítulo apresentou a motivação e seus objetivos. O Capítulo 2 apresenta os conceitos médicos com relação aos exames utilizados e os conceitos de engenharia de software; no Capítulo 3 destaca alguns trabalhos relacionados que serviram como inspiração. O Capítulo 4 apresenta o desenvolvimento utilizado no trabalho; o Capítulo 5 apresenta os experimentos e seus resultados. Por fim, no Capítulo 6 é apresentada a conclusão, limitações e trabalhos futuros.

2 CONCEITOS BÁSICOS

Este capítulo apresenta os principais conceitos e referenciais teóricos usados no trabalho, além de ferramentas e tecnologias utilizadas, relacionadas ao processo de desenvolvimento da aplicação.

2.1 Exames médicos

Neste trabalho foram utilizados 15 exames de rotina, os quais foram selecionados por meio de uma busca no site de busca Google (GOOGLE, 2023a), utilizando a string "exames de rotina". Os exames escolhidos são sugestões e não substituem a consulta com um especialista. Portanto, é recomendável consultar um profissional antes de realizar qualquer procedimento.

Abaixo estão listados os exames escolhidos, com a sua respectiva especialidade principal (PADRAO, 2024):

- **Densitometria óssea - Reumatologia** Avaliação da massa óssea para diagnóstico da osteoporose e osteopenia, é este exame que analisa a quantidade de cálcio presente no osso, o que é determinante para a densidade. (BRONSTEIN, 2020).
- **Eletrocardiograma - Cardiologia** Um dos exames realizados para a avaliação da saúde cardiovascular. Com ele, é possível identificar anormalidades do ritmo cardíaco, presença de arritmias, hipertrofias e até o infarto do miocárdio. (EXAME, 2020).
- **Exame de urina - Urologia** Utilizado para identificar infecções do trato urinário, doenças renais e doenças sistêmicas, além de ajudar a detectar bactérias e se há perda de nutrientes. (MEDICINA, 2024).
- **Glicemia - Endocrinologia** Verifica, principalmente, a quantidade de açúcares presentes na circulação do sangue. A análise detecta quando há sinais de hipoglicemia, ou seja, pouco açúcar na corrente sanguínea. Além disso, apresenta os indícios de hiperglicemia, quando existe o excesso de sacarose. Consequentemente, essa é uma das verificações mais comuns para o diagnóstico e o acompanhamento da diabetes. (VARGAS, 2024).
- **Glicemia e insulina - Endocrinologia** Importantes exames para serem realizado durante a fase infantil, essenciais para verificar se a criança está com os níveis

corretos e não apresenta um quadro de diabetes. (CELLA, 2023).

- **Hemograma completo - Hematologia** Detecta possíveis infecções, alterações e doenças, faz uma avaliação quantitativa e qualitativa sobre os três principais tipos de células que compõem o sangue e mostra se seus níveis são baixos, normais ou altos. (ESTAR, 2024).
- **Lipidograma - Cardiologia** Exame para análise de colesterol, LDL (Low-density lipoprotein), HDL (High-density lipoprotein), VLDL (Very low-density lipoprotein) e triglicerídeos que, quando estão em valores fora do normal, representam um grande risco para o desenvolvimento de diversas doenças. (D'OR, 2023)
- **Mamografia - Mastologia** Exame radiológico feito nas mamas. Possui alta resolução e fornece imagens detalhadas capazes de identificar precocemente o câncer de mama, antes mesmo que a mulher tenha sintomas. (CAMILO, 2023).
- **Papanicolau - Ginecologia** Verifica anomalias ou lesões no colo do útero, além de servir para a detecção de lesões precursoras do câncer do colo do útero e da infecção pelo HPV (Human Papiloma Virus), também indica se você tem alguma outra infecção que precisa ser tratada. (SAUDE, 2023).
- **Parasitológico - Parasitologia** Exame de fezes, que identifica parasitas, patologias e infecções. (PADRAO, 2024).
- **PCR (Polymerase Chain Reaction) - Cardiologia** PCR (Polymerase Chain Reaction), uma proteína que é produzida pelo fígado. Detecta infecções, doenças reumáticas e cardiovasculares, como infarto e AVC (Acidente vascular cerebral). (MD.SAUDE, 2022).
- **PSA (Prostate-Specific Antigens) e toque retal - Urologia** Possibilita identificar sinais de doenças na próstata, como prostatite (inflamação da próstata), hiperplasia (aumento da próstata) e câncer de próstata. (VARELLA, 2023).
- **Sorologia - Hepatologia** Identifica o tipo de vírus causador da hepatite, pode detectar e diferenciar as hepatites A, B, C, D ou E, avalia o sistema imunológico e a necessidade de reforço na vacinação. (DIAGNOSTICA, 2024).
- **Teste ergométrico - Cardiologia** Avalia as funções cardíacas em movimento. Durante o teste, o paciente caminha ou corre em uma esteira ou bicicleta ergométrica, enquanto é monitorado por um equipamento que registra a frequência cardíaca, a pressão arterial e outros sinais vitais. (DIAGNÓSTICA, 2023).
- **TSH (Thyroid Stimulanting Hormone) e T4 (Thyroxine) - Endocrinologia** Ava-

lia a tireoide no sangue e analisa o hormônio que possa estar em excesso causando o hipertireoidismo ou hipotireoidismo. (CLINICAS, 2023)

2.2 Engenharia de software

Ian Sommervile, em seu livro "Engenharia de Software" define a Engenharia de Software como: "É uma disciplina de engenharia que se preocupa com todos os aspectos de produção de software." (SOMMERVILLE, 2024). Segundo Ian Sommervile, existem muitos processos de software diferentes, mas todos devem incluir quatro atividades fundamentais para a engenharia de software:

- **Especificação de software**: funcionalidade do software e as restrições a seu funcionamento devem ser definidas. Nessa fase, é onde se entende completamente os requisitos do cliente e as restrições do projeto.
- **Projeto e implementação de software**: o software deve ser produzido para atender às especificações. É nessa fase que cria-se uma arquitetura eficiente e é feita a definição das estruturas de dados e algoritmos necessários.
- **Validação de software**: o software deve ser produzido para atender às especificações. Nessa fase podem ser feitos testes para identificar e corrigir defeitos, garantindo a qualidade do software.
- **Evolução de software**: o software deve evoluir para atender às necessidades de mudança dos clientes. É o processo de fazer ajustes e melhorias para garantir que o software continue a atender às demandas em evolução, o que pode incluir a introdução de novas funcionalidades, correção de bugs e adaptação a novas tecnologias.

2.3 Ferramentas de desenvolvimento

Nesta seção são apresentadas as tecnologias, arquitetura e ferramentas utilizadas para o desenvolvimento do sistema, envolvendo linguagens de programação, *frameworks* e bibliotecas para implementação do *backend*, *frontend* e testes unitários, além de tecnologias para gerenciamento e criação do banco de dados.

2.3.1 Banco de Dados Relacional

Um banco de dados relacional é um tipo de banco de dados que armazena e fornece acesso a pontos de dados relacionados entre si. Baseados no modelo relacional, uma maneira intuitiva e direta de representar dados em tabelas. Em um banco de dados relacional, cada linha na tabela é um registro com uma ID (Identification) exclusiva chamada chave. As colunas da tabela contêm atributos dos dados e cada registro geralmente tem um valor para cada atributo, facilitando o estabelecimento das relações entre os pontos de dados. (ORACLE, 2024)

2.3.2 PostgreSQL

O PostgreSQL é um sistema gerenciador de banco de dados objeto relacional (SGBD), desenvolvido como projeto de código aberto. (POSTGRESQL, 2023)

2.3.3 Docker

O Docker é um conjunto de produtos de plataforma como serviço que usam virtualização de nível de sistema operacional para entregar software em pacotes chamados contêineres. Os contêineres são isolados uns dos outros e agrupam seus próprios softwares, bibliotecas e arquivos de configuração. Eles podem se comunicar uns com os outros por meio de canais bem definidos. Todos os contêineres são executados por um único kernel do sistema operacional e, portanto, usam menos recursos do que as máquinas virtuais. (WIKIPEDIA, 2021)

2.3.4 DBeaver

O DBeaver é uma ferramenta gratuita de gerenciamento de banco de dados multiplataforma para desenvolvedores, administradores de banco de dados, analistas e todos que trabalham com dados, o DBeaver oferece uma interface intuitiva e robusta para executar consultas SQL (Structured Query Language), visualizar esquemas de banco de dados e gerenciar conexões . Ele suporta todos os bancos de dados SQL populares como MySQL, MariaDB, PostgreSQL, SQLite, Apache Family e muito mais. (DBEAVER, 2023)

2.3.5 Migrations

O recurso de migrações no Entity Framework Core oferece uma maneira de atualizar de forma incremental o esquema de banco de dados para mantê-lo em sincronia com o modelo de dados do aplicativo, preservando os dados existentes no banco de dados. (MICROSOFT, 2023b)

2.3.6 Entity Framework Core

O Entity Framework Core é uma tecnologia pertecente ao ecossistema .Net, uma biblioteca de mapeamento objeto-relacional desenvolvida pela Microsoft e é uma parte integral do framework .Net Core e .Net 5/6. Utilizado para mapear classe de objetos em banco de dados relacionais, facilitando a manipulação de dados através de código orientado a objetos em aplicativos .Net. Oferece suporte para uma variedade de banco de dados relacional, incluindo SQL Server, PostgreSQL, MySQL, SQLite, entre outros. (MICROSOFT, 2023a)

2.3.7 Microsoft Visual Studio

O Visual Studio é o IDE (Integrated Development Environment) mais abrangente para desenvolvedores .NET e C++ no Windows. Totalmente empacotado com uma bela matriz de ferramentas e recursos para elevar e aprimorar cada estágio de desenvolvimento de software. (MICROSOFT, 2024b)

2.3.8 DDD (Domain Driven Design)

O DDD (Domain Driven Design) é uma abordagem importante de design de software, com foco na modelagem de software para corresponder a um domínio de acordo com as informações dos especialistas desse domínio. No design orientado a domínio, a estrutura e a linguagem do código de software (nomes de classe, métodos de classe , variáveis de classe) devem corresponder ao domínio de negócios. Esta arquitetura segue princípios de separação de responsabilidade e ajuda na organização do código de forma a torná-lo mais modular, escalável e de fácil manutenção. Cada camada possui responsabi-

lidades específicas e comunica-se com outras camadas de forma estruturada, facilitando a compreensão e evolução do sistema ao longo do tempo. (DDD, 2023)

2.3.9 Swagger

O Swagger é uma ferramenta popular para documentação de API (Application Programming Interface). Permite que desenvolvedores criem, documentem e consumam uma API de maneira mais eficiente e organizada, com uma interface interativa chamada Swagger UI, que permite visualizar e testar os endpoints diretamente no navegador. (SWAGGER, 2024)

2.3.10 C Sharp

C Sharp é uma linguagem de programação moderna, orientada a objetos e fortemente tipada. Permite que os desenvolvedores criem muitos tipos de aplicativos seguros e robustos que são executados no .NET. (SHARP, 2023)

2.3.11 .NET

O .NET é um framework de código aberto com suporte da Microsoft. C Sharp é a linguagem de programação para o .NET. É fortemente tipado e tem simultaneidade integrada e gerenciamento automático de memória. (MICROSOFT, 2024a)

2.3.12 xUnit

xUnit.net é uma ferramenta de teste de unidade gratuita, de código aberto e focada na comunidade para o .NET Framework. Escrito pelo inventor original do NUnit v2, xUnit.net é a tecnologia mais recente para testes de unidade C Sharp, F Sharp, VB.NET e outras linguagens .NET. xUnit.net funciona com ReSharper, CodeRush, TestDriven.NET e Xamarin. Faz parte da .NET Foundation e opera sob seu código de conduta . É licenciado sob Apache 2 (uma licença aprovada pela OSI). (XUNIT, 2024)

2.3.13 TypeScript

TypeScript é uma linguagem de programação fortemente tipada baseada em JavaScript. (TYPESCRIPT, 2024)

2.3.14 Visual Studio Code

O Visual Studio Code é um editor de código-fonte autônomo que é executado no Windows, macOS e Linux. Recomendado para desenvolvedores JavaScript e Web, com muitas extensões, oferece suporte a praticamente qualquer linguagem de programação. (MICROSOFT, 2024b)

2.3.15 Angular

Angular é um framework de código-fonte aberto e de frontend, baseado em TypeScript, liderado pela Equipe Angular do Google e por uma comunidade de indivíduos e corporações. Angular é uma reescrita completa do AngularJS, feito pela mesma equipe que o construiu. (WIKIPEDIA, 2023a)

2.3.16 HTML

HTML (HyperText Markup Language) é o bloco de construção mais básico da web. Define o significado e a estrutura do conteúdo da web. (MOZILLA, 2023)

2.3.17 CSS

CSS (Cascading Style Sheets) é uma linguagem de estilo usado para descrever a apresentação de um documento escrito em HTML ou em XML (Extensible Markup Language), incluindo várias linguagens em XML como SVG (Scalable Vector Graphics), MathML (Mathematical Markup Language) ou XHTML (Extensible Hypertext Markup Language). O CSS descreve como elementos são mostrados na tela, no papel, na fala ou em outras mídias. (MOZILLA, 2024b)

2.3.18 Bootstrap

Bootstrap é um framework web com código-fonte aberto para desenvolvimento de componentes de interface e frontend para sites e aplicações web, usando HTML, CSS e JavaScript, baseado em modelos de design para a tipografia, melhorando a experiência do usuário em um site amigável e responsivo. (WIKIPEDIA, 2023b)

2.3.19 Git

Git é um sistema de controle de versão distribuído, gratuito e de código aberto, projetado para lidar com tudo, desde projetos pequenos a muito grandes, com velocidade e eficiência. (GIT, 2023)

2.3.20 GitHub

Uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o Git. Ele permite que programadores, utilitários ou qualquer usuário cadastrado na plataforma contribuam em projetos privados e/ou Open Source de qualquer lugar do mundo. GitHub é amplamente utilizado por programadores para divulgação de seus trabalhos ou para que outros programadores contribuam com o projeto, além de promover fácil comunicação através de recursos que relatam problemas ou misturam repositórios remotos (issues, pull request). (WIKIPEDIA, 2024)

3 TRABALHOS RELACIONADOS

Neste capítulo serão apresentados três sistemas que serviram como inspiração para este projeto, os quais foram selecionados por meio de uma busca no site de busca Google Acadêmico (ACADEMICO, 2023).

3.1 Goemergency

Sistema web de orientação médica, onde o paciente informa seus sintomas e a aplicação relaciona às possíveis doenças disponíveis no banco de dados e exibe um possível diagnóstico. Dentre as funcionalidades implementadas, o usuário tem a possibilidade de conversar com médicos especialistas na área escolhida e saber mais informações sobre sua situação, possibilitando agilizar o atendimento e fazer uma análise da possível causa dos sintomas, além de poder manter um contato mesmo com distanciamento físico.

Esse trabalho foi proposto durante a pandemia, quando houve uma alta demanda em vários setores, com recursos humanos e materiais frequentemente insuficientes, resultando em sobrecarga dos profissionais. Muitos serviços ainda usavam métodos manuais, dificultando a gestão de informações. Esses desafios foram agravados pela relutância dos pacientes em procurar ajuda médica devido às restrições de acesso aos hospitais causado pelo risco de transmissão da doença. (SILVA, 2022). Na Figura 3.1 temos uma tela dessa aplicação:

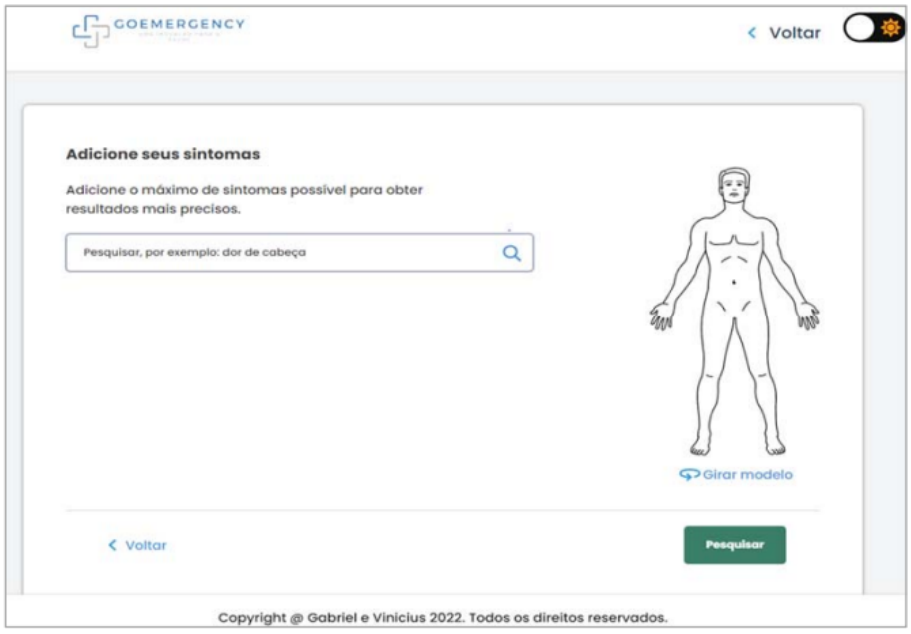

Figura 3.1 – Tela da Aplicação Goemergency

3.2 SACV - Sistema de Agendamento de Clínica Veterinária

SACV (Sistema web de agendamento de consultas veterinárias) consiste em uma aplicação desktop de agendamento de consultas veterinárias que tem como objetivo realizar a implementação de uma aplicação Web que possibilite o agendamento de consultas veterinárias através da internet, mantendo os cadastros de tutores, médicos e animais de estimação em um sistema de banco de dados. Além disso, com os dados obtidos há a possibilidade de acesso ao histórico de saúde dos animais e a geração de relatórios dos atendimentos realizados que possibilita o agendamento de consultas como também o cadastro de tutores, médicos e animais de estimação em um sistema de banco de dados.

Este trabalho foi proposto para automatizar processos e facilitar o agendamento que não precisa ser feito presencialmente ou via telefone, além de evitar a manutenção de fichas e cadastros manuscritos de tutores e animais. (GOMES, 2023). Na Figura 3.2 temos uma tela dessa aplicação:

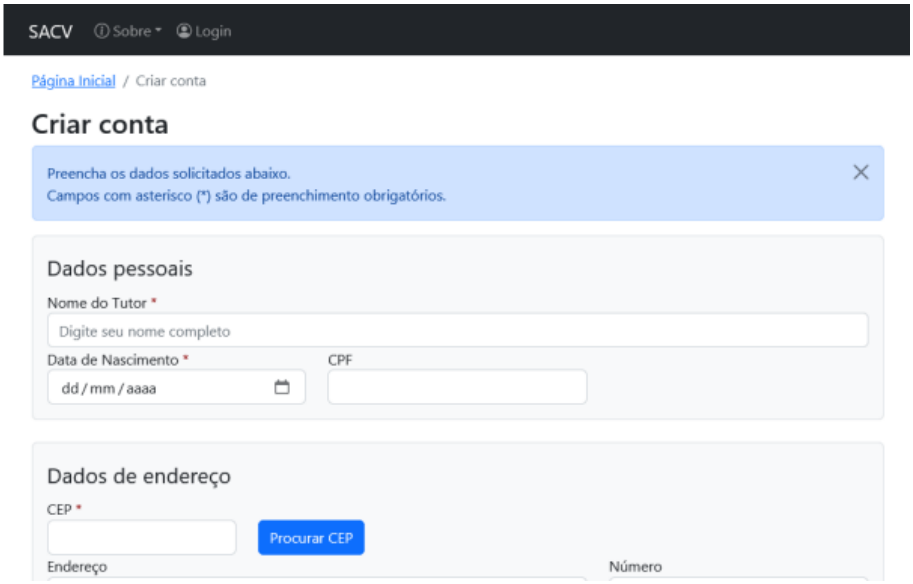

Figura 3.2 – Tela do Sistema de Agendamento Clínica Veterinária

3.3 Odonto Sistem

Sistema web que tem como objetivo propor uma aplicação web abrangente para a gestão administrativa completa de uma clínica. Ao observar os processos diários realizados em clínicas odontológicas, tornou-se evidente a necessidade de um sistema web eficiente para otimizar as atividades administrativas. Embora essas tarefas sejam simples,

consomem uma quantidade significativa de tempo, assim, a adoção de um software que auxilie nos processos administrativos da clínica se torna crucial.

Dentre as funcionalidades implementadas, destacam-se o agendamento de consultas, a atualização de prontuários contendo informações relevantes, como procedimentos realizados, anamnese e observações. Além disso, o sistema permite o cadastro de pacientes, clínicas e dentistas, proporcionando uma visão organizada e abrangente de todas as informações necessárias para o funcionamento adequado da clínica odontológica. A implementação do sistema web foi guiada pela simplicidade e facilidade de uso, com o intuito de atender às demandas específicas desse ambiente.

Este trabalho foi proposto para automatização de atividades administrativas, diminuindo assim, a quantidade de tempo para realizar essas tarefas. (CORDEIRO, 2023). Na Figura 3.3 temos uma tela dessa aplicação:

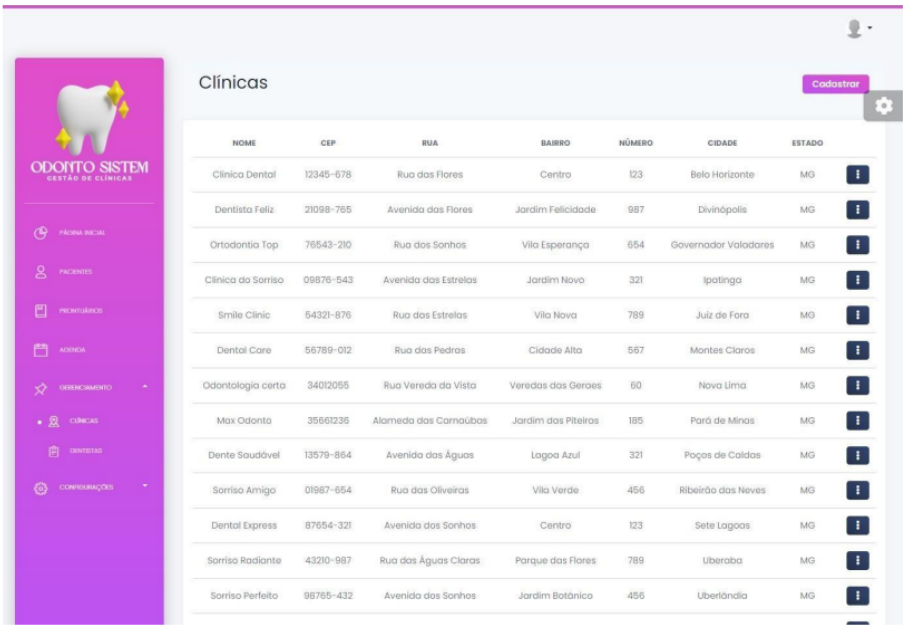

Figura 3.3 – Tela da Aplicação Odonto Sistem

3.4 Comparação

Com relação aos sistemas apresentados, todos tem a funcionalidade de realizar login, cadastramento e listagem, com a diferença que o Goemergency, utiliza as informações cadastradas para exibir um possível dignóstico, o Sistema de Agendamento de Clínica Veterinária possibilita o cadastramento e listagem de consultas, tutores, médicos e animais e o Odonto Sistem cadastra e lista também outras clínicas.

Nosso trabalho utiliza características desses sistemas, como cadastramento e lista-

gem. Ao analisarmos esses sistemas, identificamos a ausência de uma aplicação dedicada a exames, o que nos motivou a desenvolver um sistema capaz de apresentar uma lista de exames de rotina, a partir do cadastro de informações do usuário, bem como a localização real onde é possível realizar cada exame.

Essas funcionalidades não estão contempladas no comparativo, conforme mostrado na Tabela 3.1. Diante disso, desenvolvemos uma solução especializada na obtenção e organização de informações relacionadas a exames de rotina, simplificando o acesso a essas informações.

	Goemergency	*SACV*	*Odonto Sistem*	*Sistema Proposto*
Login	Sim	Sim	Sim	Sim
Cadastrar Formulário	Sim	Sim	Sim	Sim
Lista de Exames	Não	Não	Não	Sim
Localização	Não	Não	Não	Sim

Tabela 3.1 – Tabela comparativa entre os sistemas

4 DESENVOLVIMENTO

Neste capítulo serão apresentados uma visão geral do trabalho e os detalhes do desenvolvimento da aplicação, como as funcionalidades, banco de dados, *backend*, *frontend* e interface.

4.1 Visão geral

O desenvolvimento teve início com uma pesquisa das soluções já existentes. A partir dessa pesquisa, surgiu a ideia do sistema. Assim, se iniciou um desenvolvimento baseado em uma metodologia interativa, onde foram definidas as funcionalidades, arquitetura e interface. Após a finalização da aplicação, foi realizado um experimento com usuários, que será apresentado no próximo capítulo. A Figura 4.1 apresenta esta visão geral.

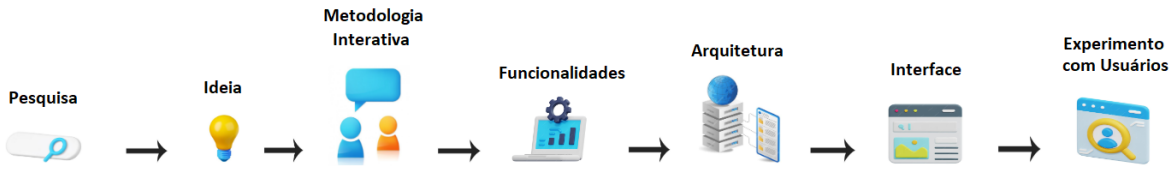

Figura 4.1 – Visão geral do trabalho

4.2 Funcionalidades

Nesta seção serão apresentadas as funcionalidades, os passos necessários para realização de cada tarefa, juntamente com as telas exibidas ao usuário durante o processo.

4.2.1 Tela de Login

Para efetuar o login no sistema, é necessário inserir um e-mail e uma senha. Posteriormente, deve-se clicar no botão *Entrar*, conforme a Figura 4.2.

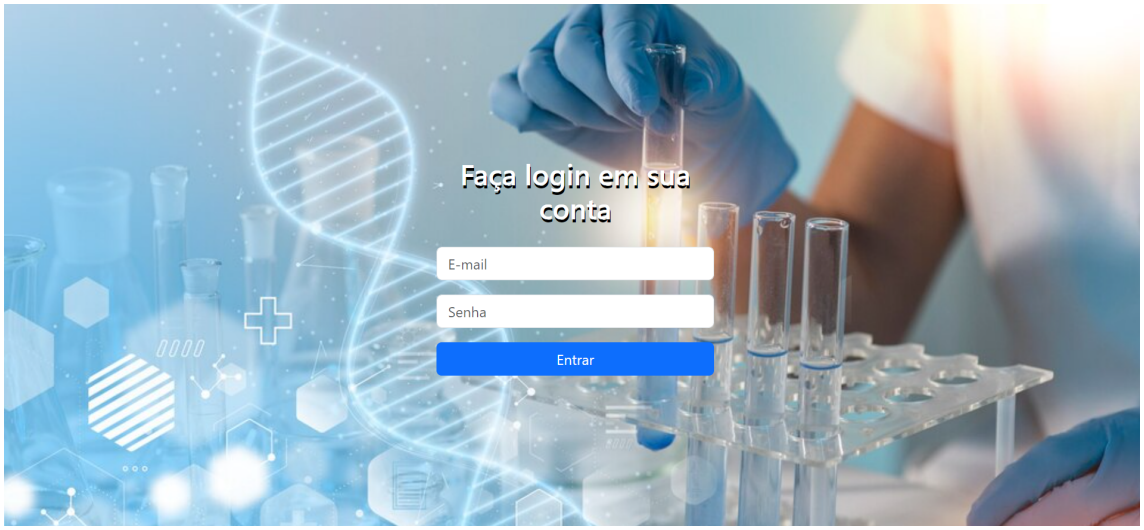

Figura 4.2 – Tela de Login

4.2.2 Tela de Cadastro

Ao entrar na aplicação o usuário será redirecionado para a tela de cadastro, mostrada na Figura 4.3, onde deverá preencher o formulário, com nome, sexo e idade. Posteriormente, deve-se clicar no botão *Buscar*. Nessa tela, também tem um menu com o botão *Sair*, onde o usuário é redirecionado para a tela de login.

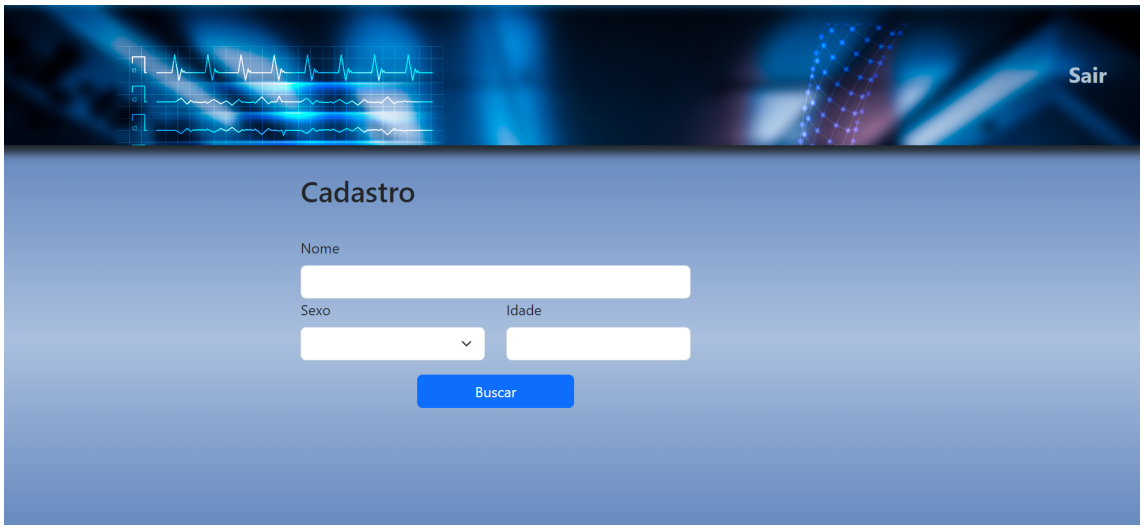

Figura 4.3 – Tela de Cadastro

4.2.3 Tela de Exames recomendados

Ao clicar no botão *Buscar* será mostrado a lista de exames de rotina recomendados, com base no sexo e idade. Nas Figuras 4.4 e 4.5 respectivamente, é mostrado um

exemplo de um usuário do sexo Feminino (F) de 33 anos e um usuário do sexo Masculino (M) de 42 anos.

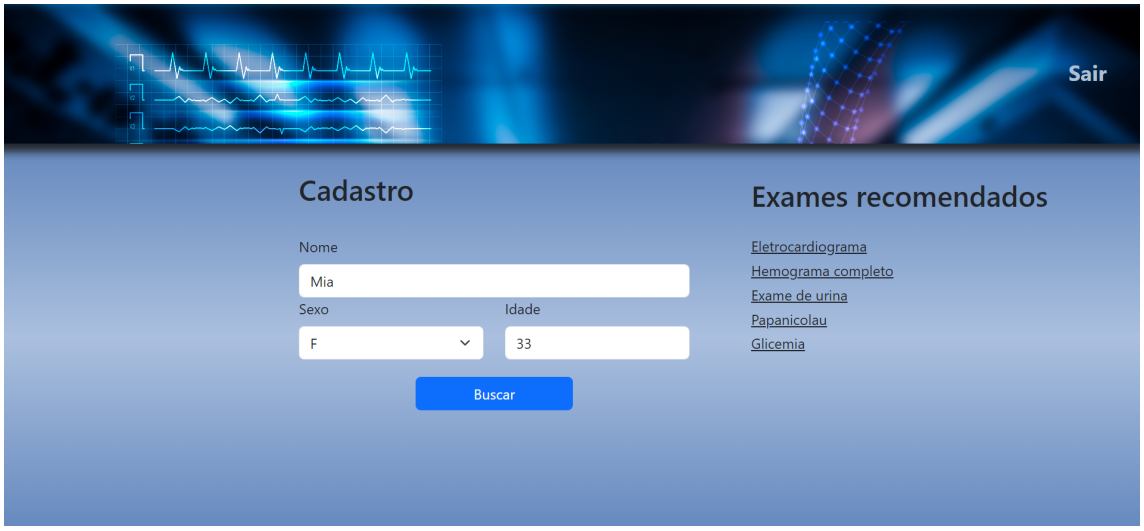

Figura 4.4 – Tela de Exames

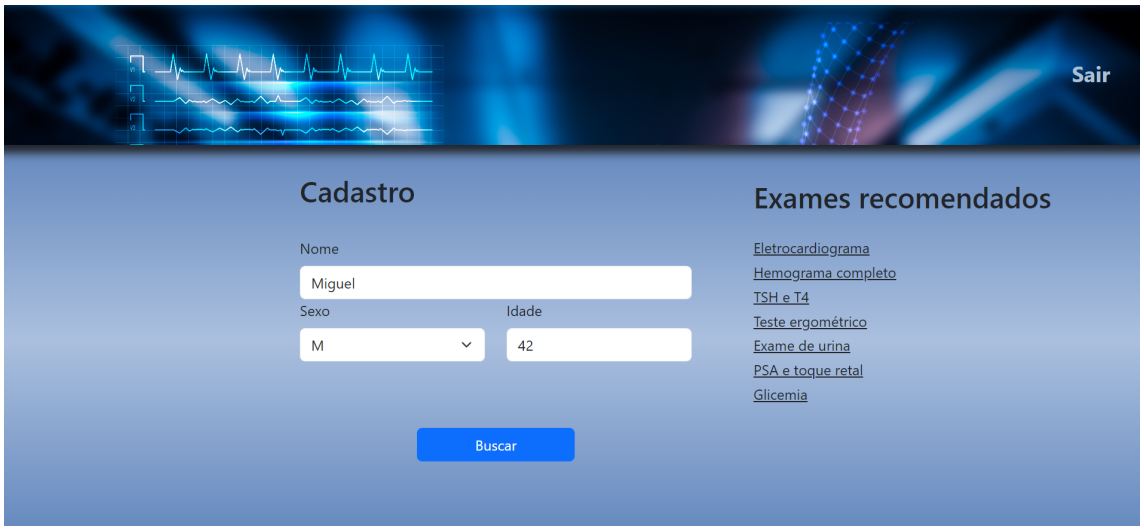

Figura 4.5 – Tela de Exames

4.2.4 Tela de Descrição do Exame

Ao clicar em um exame, na lista de exames, na tela anterior, o usuário será redirecionado para a tela de descrição do exame. Nessa tela, conforme a Figura 4.6, serão apresentados o nome do exame, uma imagem ilustrativa relacionada ao exame, descrição do exame, bem como os botões *Voltar*, que redireciona o usuário para a tela de cadastro, o botão *Sair*, que redireciona o usuário para a tela de login e o botão *Clique aqui*. Este último botão, identificado por uma label específica, tem como finalidade mostrar os locais

onde o exame pode ser realizado na cidade de Porto Alegre e região.

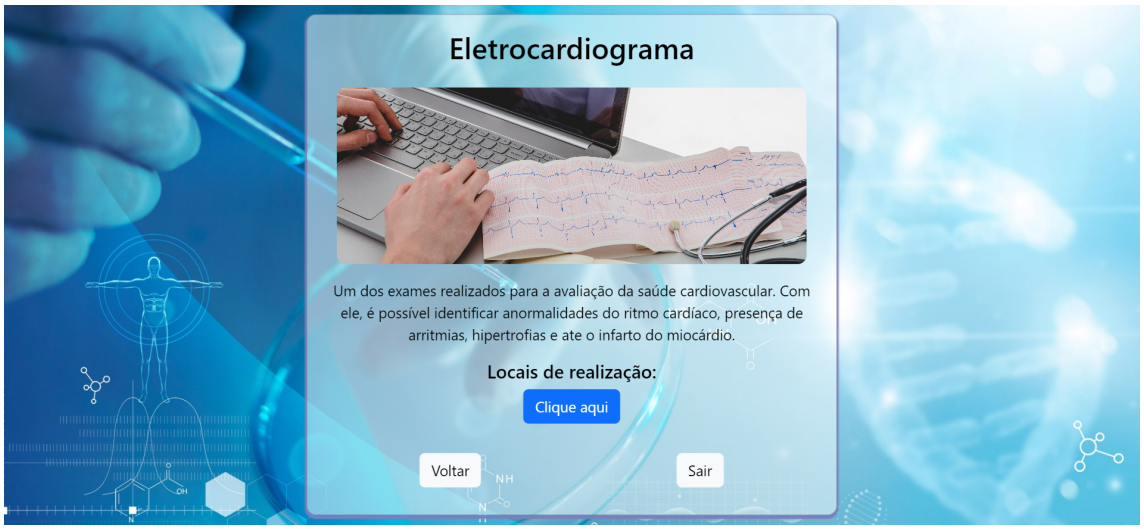

Figura 4.6 – Tela de Descrição de um Exame

4.2.5 Tela do Mapa

Ao clicar no botao *Clique aqui*, na tela anterior, o usuário será redirecionado para a tela do mapa, nessa tela são apresentados os locais, no Google Maps (GOOGLE, 2023c), onde é possível realizar o respectivo exame na cidade de Porto Alegre e região, abrangendo tanto instituições privadas, quanto aquelas vinculadas ao Sistema Único de Saúde (SUS), conforme a Figura 4.7.

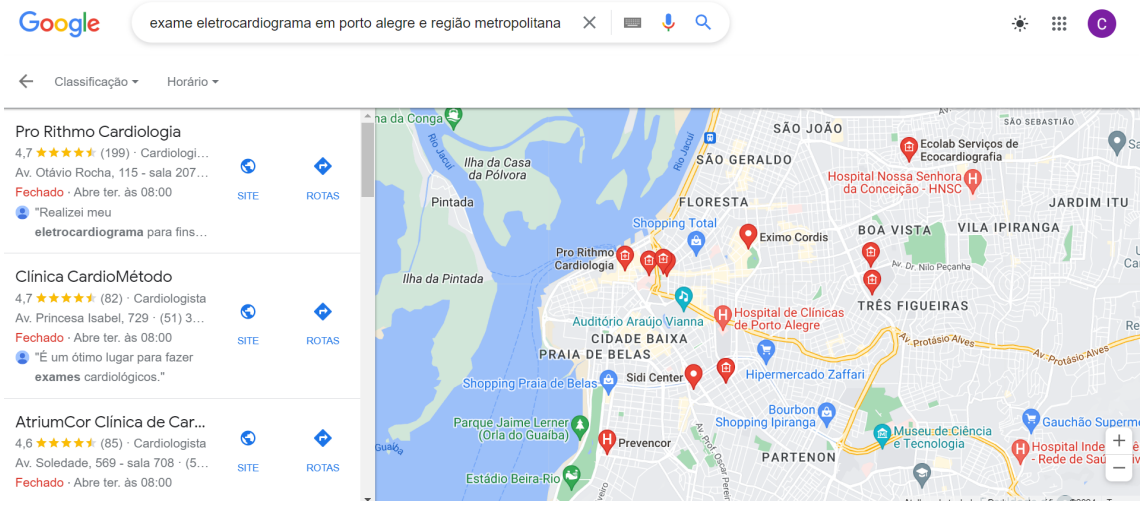

Figura 4.7 – Tela do Google Maps

4.3 Arquitetura

Esta seção descreve três dos principais detalhes arquiteturais utilizados no sistema: (i) Banco de Dados, (ii) Backend, e (iii) Frontend. A Figura 4.8 ilustra este *pipeline*, o qual está descrito nas próximas seções.

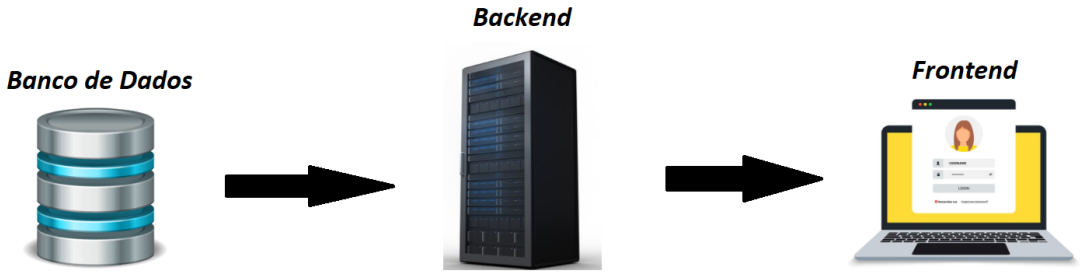

Figura 4.8 – Arquitetura do Sistema

4.3.1 Banco de dados

Para armazenar e gerenciar os dados foi utilizado o banco de dados relacional PostgreSQL. Para hospedar o banco de dados foi empregado um contêiner Docker, conforme mostrado na Figura 4.9, com isso é possível isolar o ambiente PostgreSQL, garantindo portabilidade e consistência em diferentes sistemas operacionais.

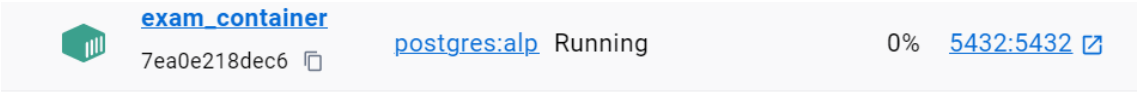

Figura 4.9 – Detalhes do Contêiner Usado Com a Tecnologia Docker

Para gerenciar o banco de dados e garantir a integridade das estruturas ao longo do tempo, foram utilizadas Migrations, que descrevem as alterações no esquema de banco de dados ao longo do tempo. Para interagir com o banco de dados PostgreSQL, foi utilizado o DBeaver. Ao criar as Migrations, a estrutura de pastas no DBeaver é automaticamente organizada para facilitar a visualização e o gerenciamento das entidades e das Migrations. Foram criadas duas entidades no banco de dados denominadas: *ExamData* e *PatientData*. Para preencher o banco de dados, está sendo utilizado um arquivo JSON (JavaScript Object Notation) como fonte de dados. Este arquivo contém uma estrutura de dados que representa a entidade, denominada *Exam* no *Backend*, o qual será discutido no próximo capítulo. Os dados contidos neste arquivo JSON podem ser facilmente modificados ou atualizados sem a necessidade de alterações no código-fonte. Na modelagem do banco de

dados, foi utilizado um relacionamento *Many to One* (Muitos para Um) para representar a relação entre as entidades *ExamData* e *PatientData*, o que significa que muitos exames estão associados a um único paciente e um paciente pode ter muitos exames, conforme mostrado no diagrama UML (Unified Modeling Language), da Figura 4.10.

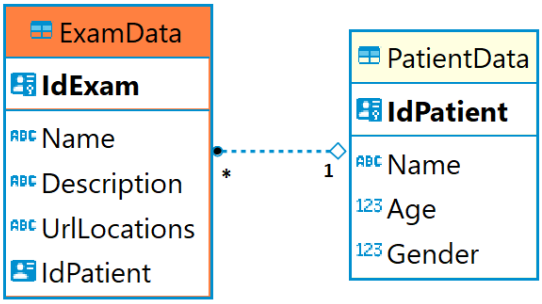

Figura 4.10 – Diagrama de classes UML contendo as entidades e seus relacionamentos

A entidade *ExamData*, contém os seguintes atributos:

- **Name**: nome do exame.
- **Description**: descrição do exame.
- **UrlLocations**: url que direciona para o Google Maps, exibindo a localização geográfica dos lugares em Porto Alegre e região onde o exame pode ser realizado.
- **IdPatient**: id do paciente que é usado para estabelecer a ligação entre as entidades, permitindo o relacionamento entre elas.

A entidade *PatientData*, contém os seguintes atributos:

- **Name**: nome do paciente.
- **Age**: idade do paciente.
- **Gender**: sexo do paciente.

Na Figura 4.11 é mostrado como a estrutura de pastas é organizada, no DBeaver, após a criação de uma Migration. A pasta *Tables* contém as tabelas que representam as entidades do projeto, juntamente com a tabela da *Migration* criada.

O PostgreSQL, por padrão, é configurado para escutar na porta 5432, que é utilizada para comunicação entre aplicativos e o banco de dados. No contexto do contêiner Docker, essa porta é exposta para que aplicativos externos, como o DBeaver, possam se conectar e interagir com o banco de dados hospedado no contêiner.

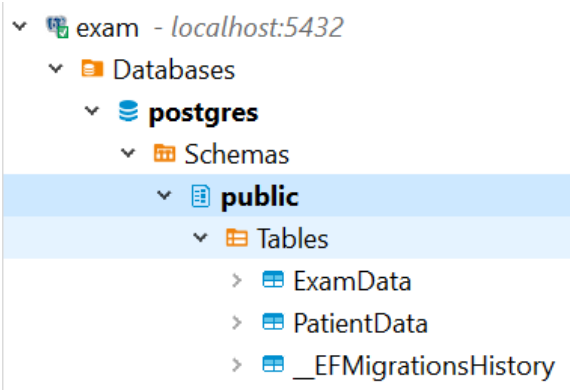

Figura 4.11 – Estrutura de Pastas no DBeaver

4.3.2 Backend

Para o desenvolvimento do Backend foi utilizado o framework .Net e para a documentação foi utilizado o Swagger. É no Backend onde está o arquivo JSON, que contém os dados da entidade *Exam*. Além disso, para a organização do projeto foi adotada a arquitetura DDD, conforme mostrado na Figura 4.12, onde as camadas foram divididas da seguinte forma:

- **Application**: nessa camada é onde ficam os *services* que são responsáveis por coordenar a lógica de negócios, manipulando as entidades e realizando operações específicas. Atuam como uma camada intermediária entre a interface do usuário e o domínio, encapsulando a complexidade da lógica de negócios.
- **Domain**: esta é a camada onde ficam as entidades e os objetos de transferência de dados (DTOs - Data Transfer Objects). A entidades representam conceitos do domínio da aplicação como *Exam* e *Patient*. Os *DTOs* são utilizados para transferir dados entre as camadas da aplicação.
- **Persistence**: nessa camada é onde ficam as *Migrations* e o código relacionado à persistência de dados. É responsável por lidar com o armazenamento e recuperação dos dados.
- **Repository**: esta camada é responsável por abstrair o acesso aos dados do banco de dados. Fornece métodos encapsulando a lógica de acesso aos dados, e uma interface de alto nível para interagir com o banco de dados, permitindo que outras camadas acessem os dados sem conhecer os detalhes da implementação.
- **UI**: nessa camada ficam os *Controllers* e o arquivo JSON. Os *Controllers* recebem requisições HTTP (Hypertext Transfer Protocol) para a interface do usuário, chamam os services na camada de *Application* para executar a lógica de negócios e

retornam as respostas apropriadas para a interface do usuário. Esta camada é responsável por lidar com a interação direta com os usuários da aplicação.

- **Test**: além das camadas mencionadas, foi implementada a camada *Test*, onde são realizados os testes unitários dos services das duas entidades existentes (*Exam* e *Patient*). Esta camada garante a qualidade do sistema, permitindo validar o comportamento individual de cada *service*. Para a execução dos testes, foi utilizado o framework xUnit. Além disso, para estruturar os testes, foi utilizado o padrão **Triple-A** (Arrange, Act, Assert). Ao seguir este padrão, os testes foram organizados em três etapas distintas:
  - **Arrange**: preparação do ambiente de teste, configurando o estado inicial para a execução do teste;
  - **Act**: realização das ações ou operações que queremos testar;
  - **Assert**: verifica-se se o resultado da ação ou operação está de acordo com o esperado. Isso permite identificar eventuais problemas ou falhas no código, garantindo um software mais confiável e livre de erros;

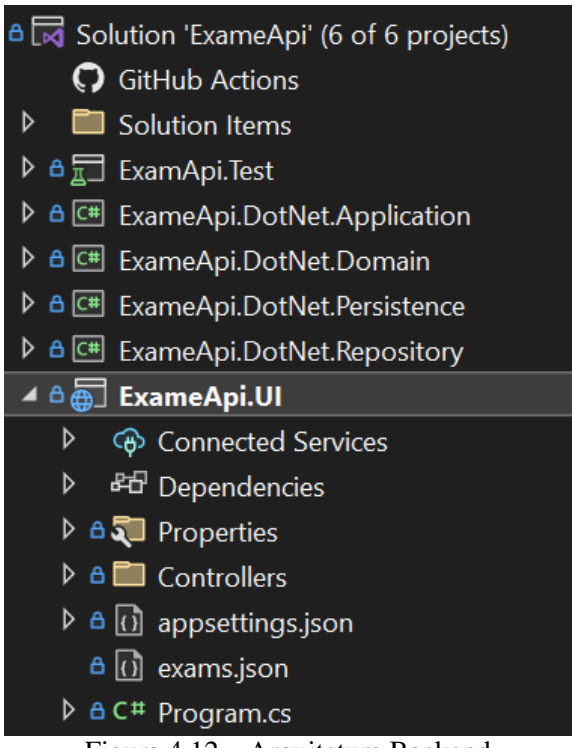

Figura 4.12 – Arquitetura Backend

4.3.3 Frontend

Para o desenvolvimento do Frontend foi utilizado o framework Angular. Foi adotada uma estrutura de pastas comumente utilizada em projetos Angular, que adota uma organização hierárquica, seguindo uma abordagem baseada em componentes, conforme mostrado na Figura 4.13, com os seguintes diretórios:

- **app**: este diretório serve como ponto de entrada para a aplicação, que contém todos os componentes, módulos, serviços e outros recursos específicos da aplicação.
- **components**: neste diretório contém os components que são parte reutilizáveis e independentes da interface do usuário.
- **models**: contém as definições de modelos de dados utilizados. Esses modelos representam as entidades *Exam* e *Patient* manipulados pela aplicação.
- **pages**: contém os componentes que representam as páginas individuais da aplicação. Cada página pode conter vários componentes e é responsável por uma parte específica da interface do usuário.
- **services**: neste diretório ficam os *services*, que são as classes injetáveis que encapsulam a lógica de negócios e fornecem funcionalidades específicas para diferentes partes da aplicação. Realizam a comunicação com o backend por meio de chamadas HTTP usando o módulo HttpClient fornecido pelo Angular. Implementam os métodos que definem operações CRUD, acrônimo que representa Create, Read, Update e Delete (MOZILLA, 2024a).
- **assets**: neste diretório estão armazenadas as imagens, que constituem os recursos estáticos fundamentais do sistema. Essas imagens são elementos que desempenham um papel importante na interface do usuário, contribuindo para a experiência visual.
- **environments**: esse diretório contém os arquivos de configuração relacionados a diferentes ambientes de desenvolvimento, como a URL (Uniform Resource Locator), que é o endereço local onde o servidor está sendo executado.

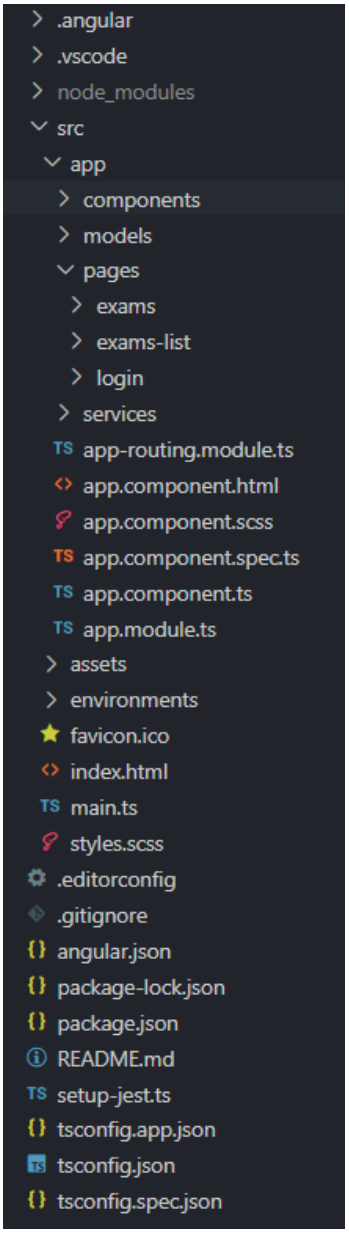

Figura 4.13 – Arquitetura Frontend

4.4 Interface

O desenvolvimento da interface foi inspirado no site do Hospital Moinhos de Vento (MOINHOS, 2024) e no site Doctoralia (DOCTORALIA, 2023).

Com base nestes sites, escolhemos utilizar cores frias, uma escolha comum em sites da área da saúde, que segundo a psicologia das cores (MATILDEFILMES, 2022), transmite uma sensação de tranquilidade e segurança. Para a tela de login, foi utilizado como base a tela do Doctoralia, conforme mostrada na Figura 4.14.

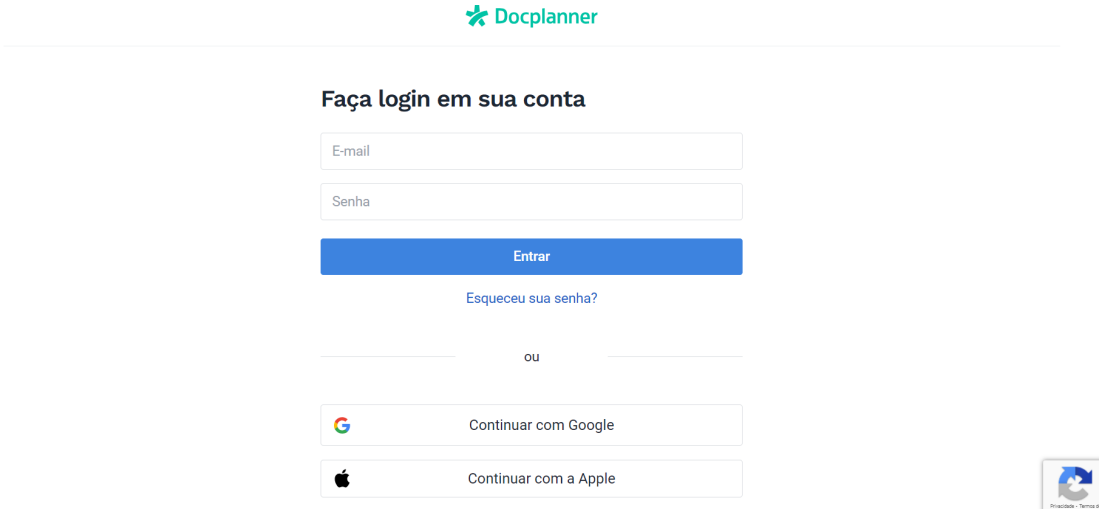

Figura 4.14 – Página de Login do Site Doctoralia

A utilização de imagens foi inspirada no site do Hospital Moinhos de Vento, que utiliza imagens, conforme mostrado na Figura 4.15.

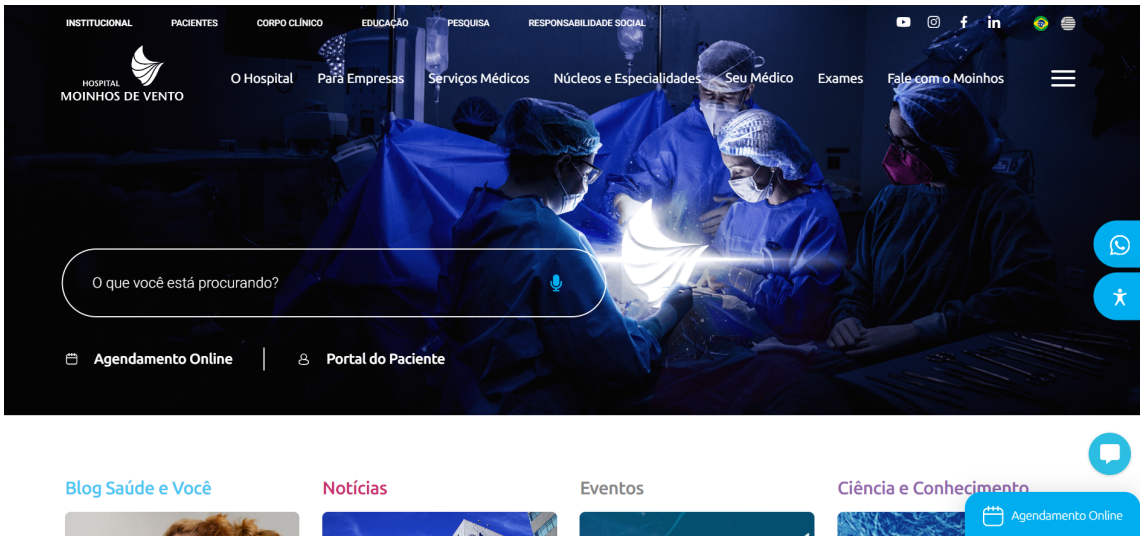

Figura 4.15 – Página Inicial do Site Moinhos de Vento

5 VALIDAÇÃO

Neste capítulo, serão apresentados os resultados da avaliação do projeto, baseado na experência direta dos usuários que interagiram com a aplicação. Esses resultados foram coletados por meio do experimento prático e pelas respostas fornecidas pelos participantes. Para ilustrar foram utilizados gráficos, gerados pela ferramenta disponível no Google Drive. (GOOGLE, 2023b)

5.1 Experimentos

Os experimentos foram feitos com 17 participantes. Dentre as pessoas selecionadas, uma era da área de tecnologia e uma era da área da saúde. O experimento consistiu em elaborar perguntas para coleta de dados sobre os participantes, conforme mostrado na Tabela 5.1. Cada pessoa fez o experimento no mesmo computador onde o sistema foi desenvolvido. Antes de cada participante começar o experimento, foi explicado o objetivo do sistema e as tarefas que deveriam realizar:

- fazer login
- preencher formulário
- clicar no botão *Buscar*
- clicar em um exame da lista
- ler as informações na tela de descrição do exame
- clicar no botão *Clique aqui*
- observar o mapa

O sistema começa com a tela de login, onde os participantes foram orientados que poderiam colocar e-mail e senha fictícios ou clicar diretamente no botão *Entrar*. Ao entrar no sistema, seguiram o fluxo de tarefas explicado.

Após o experimento foi solicitado que cada participante desse uma nota com relação à Usabilidade e Utilidade, conforme mostrado na Tabela 5.2.

5.1.1 Dados pré-experimento

Na Tabela 5.1 temos os dados, que foram coletados no pré-experimento: idade, sexo, profissão e escolaridade.

	Idade	*Sexo*	*Profissão*	*Escolaridade*
Usuário 1	18	M	Estudante	Ensino médio completo
Usuário 2	22	M	Economista	Ensino superior completo
Usuário 3	25	F	Nutricionista	Ensino superior completo
Usuário 4	28	F	Design	Ensino médio completo
Usuário 5	30	M	Segurança	Ensino médio incompleto
Usuário 6	34	F	Desenvolvedora	Ensino superior incompleto
Usuário 7	35	M	Publicitário	Ensino superior incompleto
Usuário 8	36	M	Contador	Ensino superior completo
Usuário 9	49	F	Diarista	Ensino fundamental incompleto
Usuário 10	50	F	Cozinheira	Ensino médio completo
Usuário 11	55	M	Zelador	Ensino médio completo
Usuário 12	55	F	Zeladora	Ensino fundamental completo
Usuário 13	56	F	Cozinheira	Ensino médio completo
Usuário 14	58	M	Porteiro	Ensino médio completo
Usuário 15	59	M	Zelador	Ensino fundamental incompleto
Usuário 16	60	F	Aposentada	Ensino fundamental incompleto
Usuário 17	67	M	Aposentado	Ensino fundamental incompleto

Tabela 5.1 – Tabela de Distribuição dos Dados

Na Figura 5.1 temos o percentual referente à faixa etária dos participantes, conforme mostrado a seguir:

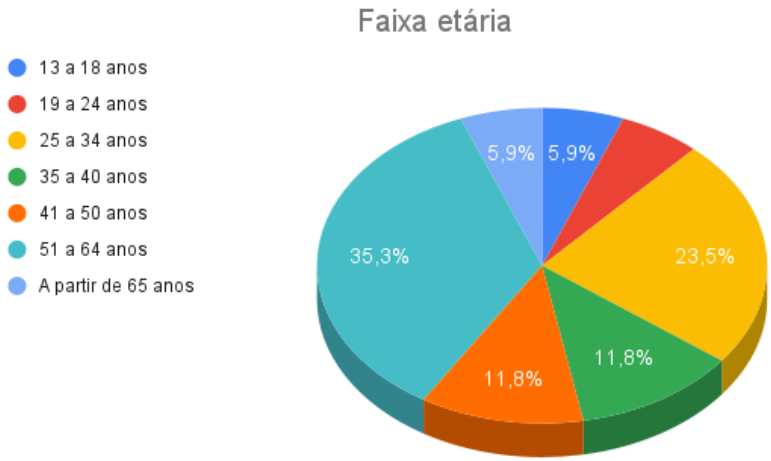

Figura 5.1 – Distribuição por Faixa etária

Na Figura 5.2 temos o percentual referente ao sexo dos participantes, conforme

mostrado a seguir:

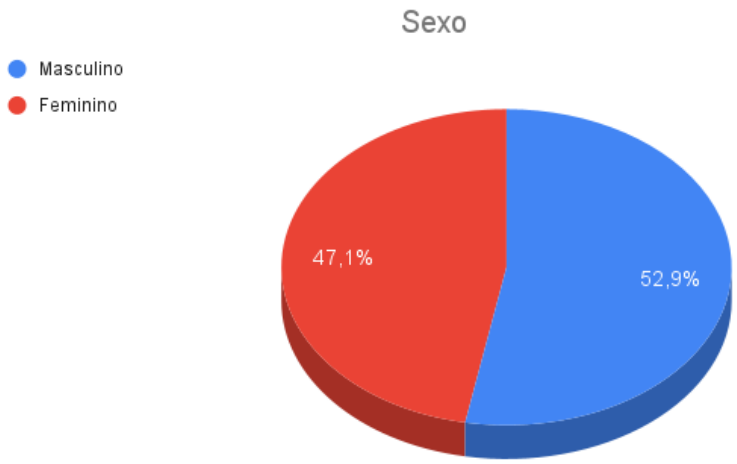

Figura 5.2 – Distribuição por Sexo

Na Figura 5.3 temos o percentual referente à escolaridade dos participantes, conforme mostrado a seguir:

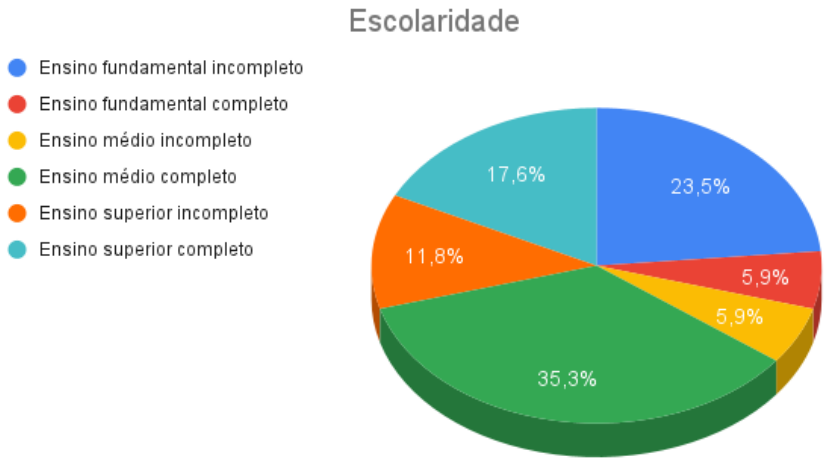

Figura 5.3 – Distribuição por Escolaridade

5.1.2 Dados pós-experimento

Na Tabela 5.2 temos os dados, que foram coletados no pós-experimento, através de respostas dos usuários quanto a Usabilidade e Utilidade.

	Usabilidade	*Utilidade*
Usuário 1	5	5
Usuário 2	5	5
Usuário 3	5	5
Usuário 4	5	5
Usuário 5	4	5
Usuário 6	5	5
Usuário 7	5	5
Usuário 8	5	5
Usuário 9	4	5
Usuário 10	4	5
Usuário 11	4	5
Usuário 12	4	5
Usuário 13	4	5
Usuário 14	4	5
Usuário 15	4	5
Usuário 16	4	5
Usuário 17	4	5

Tabela 5.2 – Tabela das Avaliações de Usabilidade e Utilidade

Na **Usabilidade**, foi avaliado a facilidade do uso do sistema, foi utilizado uma escala de 1 a 5:

- **1**- Muito Difícil
- **2**- Difícil
- **3**- Neutro
- **4**- Fácil
- **5**- Muito Fácil

Na **Utilidade**, foi avaliado a utilidade com relação ao objetivo proposto, foi utilizado uma escala de 1 a 5:

- **1**- Completamente Inútil
- **2**- Inútil
- **3**- Neutro
- **4**- Útil
- **5**- Muito Útil

5.2 Resultados e discussão

Os resultados dos testes mostraram que os usuários consideraram a aplicação extremamente útil em relação ao fornecimento das informações sobre exames de rotina. Com relação à usabilidade, foi possível observar que os usuários com menor escolaridade e os usuários com faixa etária mais alta relataram maior dificuldade ao utilizar o sistema, visto que a nota média deste perfil foi 4 (Fácil). Por outro lado, aqueles com escolaridade mais alta e faixa etária mais baixa acharam muito fácil, atribuindo uma nota média de 5 (Muito Fácil). Esse resultado mostra uma relação entre o nível de escolaridade, idade e a facilidade na interação com o sistema, mostrando a importância de considerar a diversidade etária e educacional ao projetar interfaces e experiências de usuário.

Após o experimento com um usuário da área da saúde, foi detectada a seguinte limitação: ao entrar na tela de descrição de um exame, escolhido na lista de exames e voltar para a tela de cadastro, a lista de exames não fica salva, o usuário deve preencher novamente o formulário. Com relação à escolha e informações em geral sobre os exames, esse usuário relatou ter tido uma experiência muito positiva.

Após o experimento com um usuário da área de design, foram apontadas as seguintes melhorias: a lista de exames poderia aparecer em uma outra página, separada da página de cadastro e poderia ter uma mensagem para deixar o site mais íntimo para o usuário, como por exemplo: "Olá 'nome do usuário', esta é a sua lista de exames recomendados!".

Após o experimento com um usuário da área de tecnologia, também foi detectada a limitação com relação à lista de exames não ficar salva. Além disso, foram apontadas melhorias como implementação da autenticacão utilizando um email da conta do Google (GOOGLE, 2024) e implementação da edição dos dados do formulário, deixando também esses dados salvos quando voltar para tela de cadastro, além da inserção de mensagens de validações no formulário.

Mesmo diante dessas variações, foi identificada uma facilidade de uso entre todos os usuários, mostrando que o sistema pode oferecer uma experiência acessível e intuitiva, estando alinhado com as necessidades e habilidades do público-alvo, alcançando o objetivo de incentivo e facilitação de acesso às informações referentes aos principais exames de rotina.

6 CONCLUSÃO

Neste trabalho foi proposto o desenvolvimento de uma aplicação web de cadastramento e orientação médica, mostrando os conceitos médicos, referenciais teóricos, as etapas da Engenharia de Software, tecnologias, arquitetura e ferramentas utilizadas.

Foram apresentados os trabalhos relacionados que serviram como inspiração, o desenvolvimento da aplicação com as funcionalidades e passos para uso do sistema, juntamente com as telas.

Foi realizado um experimento de teste de usabilidade e utilidade envolvendo a interação direta dos usuários com o sistema, onde foram feitas perguntas para coleta de dados. Os resultados do experimento foram altamente satisfatórios, onde os participantes consideraram extremamente útil e não tiveram maiores dificuldades no uso do sistema. Também foram identificadas algumas sugestões de melhorias e detalhes que requerem ajustes.

Portanto, é possível concluir que a aplicação desenvolvida desempenha um papel crucial na facilitação do acesso às informações relacionadas aos exames de rotina e na promoção da cultura da prevenção e cuidados com a saúde. Através do sistema, os usuários são incentivados a realizar exames de rotina de forma mais consciente e proativa, contribuindo para a detecção precoce e adoção de medidas preventivas adequadas.

6.1 Limitações

Nesta seção serão apresentadas as limitações que foram identificadas na aplicação:

- **recomendação dos exames**: os exames recomendados são uma sugestão, não teve consulta com um especialista.
- **o sistema não é completamente responsivo**: embora tenha sido utilizado o framework Bootstrap em algumas partes, que tem a capacidade de tornar os elementos responsivos, outras partes do sistema foram desenvolvidas sem o framework, resultando em uma falta de responsividade completa.
- **o login apresentado não possui autenticação, trata-se de uma simulação**: não existe validação dos dados inseridos pelo usuário em relação à existência ou correção, com isso o sistema aceita qualquer entrada de e-mail e senha, mesmo que esses dados não existam de fato.

- **falta de persistência dos dados da tela de cadastro**: com base nos experimentos, alguns usuários identificaram que quando entram na tela de um exame específico e voltam para a tela de cadastro, é necessário preencher os dados novamente, pois a lista de exames não fica salva.
- **o sistema não foi colocado em produção**: o sistema executa apenas localmente, ou seja, no computador onde foi desenvolvido ou no computador de qualquer pessoa que tenha baixado o código-fonte do sistema, isso implica que ainda não está disponível publicamente.

6.2 Trabalhos futuros

Nesta seção serão apresentadas melhorias que futuramente poderão ser implementadas:

- **consulta de um especialista**: consultar um especialista para validar os exames.
- **experimento com profissionais da área da saúde**: fazer a validação do sistema com mais usuários que sejam da área da saúde.
- **inclusão de mais locais**: incluir mais locais, não somente Porto Alegre e região. Uma das formas mais simples de fazer isso é ampliar a área geográfica, ajustando as coordenadas de latitude e longitude no atributo UrlLocations, no arquivo JSON, no *backend*, para assim, incluir uma área maior, como mais locais na região já coberta, incluir o estado inteiro ou até mesmo outros estados.
- **considerar planos de saúde**: quem tem plano de saúde, tem restrição de local, então implementar a localização considerando também os usuários que tem plano de saúde.
- **idade**: implentar data de nascimento ao invés de idade.
- **refatoração na tela de cadastro**: para manter a lista de exames salva e evitar que o usuário precise preencher o formulário novamente ao retornar para a tela, uma abordagem seria armazenar os dados da lista de exames em algum local persistente, como no armazenamento local do navegador quando o usuário voltar para essa tela, isto pode ser implementado usando localStorage (MOZILLA, 2024c) no lado do cliente.
- **implementar autenticação**: uma maneira de implementar autenticação é baseada em tokens, que são objetos digitais que representam a autorização de um usuário

para acessar recursos específicos em um sistema, são muito utilizados para validar a identidade de usuários e conceder acesso a determinados recursos ou funcionalidades. (GOOGLE, 2023).

- **deixar o sistema completamente responsivo**: para garantir um sistema completamente responsivo, podemos utilizar técnicas como media queries CSS ou incorpar componentes responsivos do Bootstrap em todas as partes do sistema.
- **fazer deploy da aplicação**: para fazer deploy existem plataformas online que facilitam esse processo, oferecem ferramentas para preparar o ambiente de produção, implantar o código-fonte e configurar o servidor.

REFERÊNCIAS

ACADEMICO, G. **Google Acadêmico**. 2023. <https://scholar.google.com/schhp?hl=pt-BR&lr=lang_pt&as_sdt=0,5/>.

BRONSTEIN. **Bronstein**. 2020. <https://bronstein.com.br/saude/densitometria-ossea#:~:text=A%20densitometria%20%C3%B3ssea%20detecta%20problemas,possibilidade%20de%20osteopenia%20ou%20osteoporose./>.

CAMILO, H. M. S. **Hospital Maternidade Sao Camilo**. 2023. <https://www.saocamilosaogoncalo.org.br/institucional/Exames-de-Imagem-244#:~:text=MAMOGRAFIA,que%20a%20mulher%20tenha%20sintomas./>.

CELLA, L. **Laboratório Cella**. 2023. <https://laboratoriocella.com.br/check-up-infantil-exames-importantes-para-a-saude-das-criancas/#:~:text=Os%20exames%20de%20glicemia%20e,apresenta%20um%20quadro%20de%20diabetes./>.

CLINICAS, L. C. de A. **Laboratorio Centro de Analises Clinicas**. 2023. <https://blog.labcac.com.br/tsh-e-t4-livre-como-esses-exames-podem-ajudar/>.

CORDEIRO, G. C. **Odonto Sistem**. 2023. <https://monografias.ufop.br/handle/35400000/6152>.

DBEAVER. **DBeaver**. 2023. <https://dbeaver.io/>.

DDD. **DDD**. 2023. <https://en.wikipedia.org/wiki/Domain-driven_design>.

DIAGNOSTICA, C. M. **Cura Medicina Diagnostica**. 2024. <https://cura.com.br/hepatites-fique-alerta-aos-tipos-e-exames-que-auxiliam-no-diagnostico/>.

DIAGNÓSTICA, C.-M. **CRD-MEDICINA DIAGNÓSTICA**. 2023. <https://crd.med.br/blog/para-que-serve-o-teste-ergometrico/>.

DOCTORALIA. **Doctoralia**. 2023. <https://l.doctoralia.com.br/>.

D'OR, R. **Rede D'or**. 2023. <https://www.rededorsaoluiz.com.br/exames-e-procedimentos/analises-clinicas/lipidograma/>.

ESTAR, S. B. **Saude Bem Estar**. 2024. <https://www.saudebemestar.pt/pt/exame/analises-clinicas/hemograma/>.

EXAME, L. **Laboratorio Exame**. 2020. <https://laboratorioexame.com.br/saude/eletrocardiograma/>.

GIT. **Git**. 2023. <https://git-scm.com/>.

GOMES, A. G. **Sistema de agendamento de clínica veterinária**. 2023. <https://lume.ufrgs.br/handle/10183/257998>.

GOOGLE. **Google**. 2023. <https://www.google.com.br/>.

GOOGLE. **Google Drive**. 2023. <https://www.google.com/intl/pt-br/drive/about.html>.

GOOGLE. **Google Maps**. 2023. <https://maps.google.com>.

GOOGLE. **Tokens**. 2023. <https://cloud.google.com/docs/authentication/token-types?hl=pt-br#:~:text=O%20que%20s%C3%A3o%20tokens,-Para%20autentica%C3%A7%C3%A3o%20e&text=Na%20maioria%20dos%20fluxos%20de,pode%20ser%20autorizado%20a%20acessar.>

GOOGLE. **Account**. 2024. <https://www.google.com/account/about>.

HOSPITAL, A. **Austa Hospital**. 2023. <https://blog.austahospital.com.br/check-up-de-exames-a-importancia-da-prevencao-e-diagnostico-precoce/>.

MATILDEFILMES. **Psicologia das cores**. 2022. <https://www.matildefilmes.com.br/psicologia-das-cores-guia-avancado-para-profissionais//>.

MD.SAUDE. **MD.SAUDE**. 2022. <https://www.mdsaude.com/exames-complementares/proteina-c-reativa-pcr/>.

MEDICINA, C. de. **Centro de Medicina**. 2024. <https://www.centrodemed.com.br/blog/?p=1128/>.

MICROSOFT. **Entity Framework Core**. 2023. <https://learn.microsoft.com/pt-br/ef/core/>.

MICROSOFT. **Migrations**. 2023. <https://learn.microsoft.com/pt-br/ef/core/managing-schemas/migrations/?tabs=dotnet-core-cli/>.

MICROSOFT. **Dotnet**. 2024. <https://dotnet.microsoft.com/pt-br/learn/dotnet/what-is-dotnet>.

MICROSOFT. **Visual Studio Code**. 2024. <https://visualstudio.microsoft.com/pt-br/>.

MOINHOS, H. **Hospital Moinhos de Vento**. 2024. <https://www.hospitalmoinhos.org.br/institucional/>.

MOZILLA. **HTML**. 2023. <https://developer.mozilla.org/pt-BR/docs/Web/HTML>.

MOZILLA. **CRUD**. 2024. <https://developer.mozilla.org/pt-BR/docs/Glossary/CRUD/>.

MOZILLA. **CSS**. 2024. <https://developer.mozilla.org/pt-BR/docs/Web/CSS>.

MOZILLA. **Mozilla**. 2024. <https://developer.mozilla.org/pt-BR/docs/Web/API/Window/localStorage/>.

ORACLE. **Bando de Dados Relacional**. 2024. <https://www.oracle.com/br/database/what-is-a-relational-database/>.

PADRAO. **Exames de Rotina**. 2024. <https://www.padrao.com.br/blog/exames-de-rotina/>.

POPULAR, O. **O Popular**. 2023. <https://opopular.com.br/infomercial/a-maioria-da-populac-o-n-o-faz-check-up-com-regularidade-1.3018489/>.

POSTGRESQL. **PostgreSQL**. 2023. <https://pt.wikipedia.org/wiki/PostgreSQL>.

SAUDE, B. V. em. **Biblioteca Virtual em Saude**. 2023. <https://bvsms.saude.gov.br/papanicolau-exame-preventivo-de-colo-de-utero/>.

SHARP, C. **C Sharp**. 2023. <https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/>.

SILVA, V. P. A. Gabriel Moretti da. **Goemergency**. 2022. <https://ric.cps.sp.gov.br/handle/123456789/11425>.

SOMMERVILLE, I. **Engenharia de software**. Pearson Prentice Hall, 2024. ISBN 9788579361081. Available from Internet: <https://books.google.com.br/books?id=H4u5ygAACAAJ>.

SWAGGER. **Swagger**. 2024. <https://swagger.io/>.

TYPESCRIPT. **TypeScript**. 2024. <https://www.typescriptlang.org/>.

VARELLA, D. **Exame de toque**. 2023. <https://drauziovarella.uol.com.br/ambulatorio/exames/toque-retal-exame-de-toque/>.

VARGAS, L. J. **Laboratório Júlio Vargas**. 2024. <https://labjuliovargas.com.br/exame-de-glicemia-o-que-e-e-para-o-que-serve/>.

WIKIPEDIA. **Docker**. 2021. <https://pt.wikipedia.org/wiki/Docker_(software)>.

WIKIPEDIA. **Angular**. 2023. <https://pt.wikipedia.org/wiki/Angular_(framework)>.

WIKIPEDIA. **Bootstrap**. 2023. <https://pt.wikipedia.org/wiki/Bootstrap_(framework_front-end)>.

WIKIPEDIA. **Github**. 2024. <https://pt.wikipedia.org/wiki/GitHub>.

XUNIT. **xUnit**. 2024. <https://xunit.net/>.