CENTRO ESTADUAL DE EDUCAÇÃO TECNOLÓGICA PAULA SOUZA
Faculdade de Tecnologia da Praia Grande
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

HEALTHCARE

SISTEMA PARA VERIFICAÇÃO DE QUANTIDADE DE PROFISSIONAIS DISPONÍVEIS POR UNIDADE DE SAÚDE

Bahjet Mohamad Khalil
Pedro Henrique Bezerra Severino
Pedro Vinicius da Silva Oliveira

PRAIA GRANDE
2024

Bahjet Mohamad Khalil
Pedro Henrique Bezerra Severino
Pedro Vinicius da Silva Oliveira

HEALTHCARE

SISTEMA PARA VERIFICAÇÃO DE QUANTIDADE DE PROFISSIONAIS DISPONÍVEIS POR UNIDADE DE SAÚDE

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia da Praia Grande, como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.
Orientador: Prof. Joseffe Barroso de Oliveira

PRAIA GRANDE
2024

Khalil, Bahjet Mohamad.
Severino, Pedro Henrique Bezerra.
Oliveira, Pedro Vinicius da Silva.

Healthcare – Sistema para verificação de quantidade de profissionais disponíveis por unidade de saúde / Bahjet Mohamad Khalil, Pedro Henrique Bezerra Severino, Pedro Vinicius da Silva Oliveira – Praia Grande: Centro Estadual de Educação Tecnológica Paula Souza (CEETEPS). - Junho, 2024.
95f.: il.

Relatório Técnico (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) – Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia da Baixada Santista (FATEC Praia Grande), 2024.

Orientador: Joseffe Barroso de Oliveira.

1. Demora no atendimento médico. 2.Divulgar hospitais e clinicas de Praia Grande. 3.Históricos e armazenamento de exames. I Título. II. Centro Estadual de Educação Tecnológica Paula Souza – FATEC Praia Grande.

Bahjet Mohamad Khalil
Pedro Henrique Bezerra Severino
Pedro Vinicius da Silva Oliveira

HEALTHCARE

SISTEMA PARA VERIFICAÇÃO DE QUANTIDADE DE PROFISSIONAIS DISPONÍVEIS POR UNIDADE DE SAÚDE

Trabalho de Conclusão de Curso apresentado à Faculdade de Tecnologia da Praia Grande, como exigência parcial para obtenção do título de tecnólogo em Análise e Desenvolvimento de Sistemas.

Praia Grande, 21 de junho de 2024.

Banca Examinadora

Joseffe Barroso de Oliveira
Faculdade de Tecnologia da Praia Grande
Presidente

Jônatas Cerqueira Dias
Faculdade de Tecnologia da Praia Grande

Raphael Naves
Faculdade de Tecnologia da Praia Grande

AGRADECIMENTOS

Gostaríamos de expressar nossa profunda gratidão a Deus, por nossas vidas e por nos ajudar a ultrapassar todos os obstáculos encontrados ao longo deste curso.

Aos nossos pais, irmãos, amigos, nossos sinceros agradecimentos pelo incentivo nos momentos difíceis e pela compreensão diante de nossa ausência enquanto nos dedicávamos à realização deste trabalho.

Aos nossos professores, agradecemos pelas correções e ensinamentos, que foram fundamentais para que pudéssemos apresentar um desempenho aprimorado em nosso processo de formação profissional.

Quero agradecer a minha namorada Bianca Drumond, que me apoiou incansavelmente em todas as fases deste trabalho. Sua paciência, compreensão e carinho foram fundamentais para que eu pudesse manter o equilíbrio emocional e alcançar a conclusão deste TCC.

RESUMO

O “HealthCare” é um sistema de software desenvolvido para melhorar o acesso à informação e a qualidade dos cuidados de saúde na região da Praia Grande. É constituída por uma aplicação web dirigida a hospitais ou clínicas, privadas ou públicas, e por uma aplicação móvel dirigida ao público em geral, permitindo aos utilizadores obter informação essencial sobre hospitais públicos e privados, listas de espera e médicos especializados disponíveis nas unidades hospitalares.

Na aplicação Web, o acesso será restrito a hospitais ou clínicas com credenciais específicas para que se possam registar na plataforma e depois de efetuada a autenticação possam atualizar o número de médicos disponíveis por especialidades.

O aplicativo Mobile estará disponível ao público em geral para download em dispositivos móveis, possibilitando os usuários pesquisarem especialidades cadastradas em hospitais ou clínicas públicas e privadas da região de Praia Grande, possibilitando a visualização de informações como o número de médicos disponíveis por especialidades e a avaliação dos demais usuários/pacientes em relação à fila de atendimento.

Palavras-Chave: Demora no atendimento médico; Divulgar hospitais e clínicas de Praia Grande; Históricos e armazenamento de exames.

ABSTRACT

"HealthCare" is a software system designed to improve access to information and the quality of care in the healthcare sector in the Praia Grande region. It consists of a web application for private hospitals and a mobile application for the public, allowing users to obtain essential information about both public and private hospitals, waiting times, and available specialized doctors in registered hospital units.

In the Web application for Hospitals, access will be restricted to hospitals with specific credentials so they can register on the platform and, upon authentication, update the number of available doctors by specialties.

The Mobile application will be available for download to the general public on mobile devices, where users can search for public and private hospitals in the Praia Grande region and view their information, including the number of doctors available by specialties and user/patient ratings regarding the waiting queue for treatment.

Keywords: Delay in medical care; Promote hospitals and clinics in Praia Grande; Exam histories and storage.

LISTA DE ABREVIAÇÕES E SIGLAS

API	Application Programming Interface
AWS	Amazon Web Services
CSS	Cascading Style Sheets
DER	Diagrama Entidade Relacionamento
HTML	HyperText Markup Language
IOS	Iphone Operating System
IP	Internet Protocol
ORM	Object Relational Mapping
POO	Programação Orientada a Objetos
REST	Representational State Transfer
SCRUM	Sprint Cycle Review Update Meeting
SPA	Single Page Application
SQL	Structured Query Language
UML	Unified Modeling Language

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Diante do contexto contemporâneo, muitas pessoas expressam apreensão em buscar atendimento hospitalar, devido à preocupação com as filas de espera e a incerteza sobre a disponibilidade dos profissionais de saúde necessários. Essa preocupação não apenas afeta a experiência do paciente, mas também desafia a capacidade dos hospitais de fornecer um serviço eficiente e de qualidade. Diante desse cenário, surge a necessidade de desenvolver soluções inovadoras que utilizem a tecnologia para enfrentar tais desafios.

Este sistema foi desenvolvido com o objetivo de demonstrar como a tecnologia bem aplicada e pensada pode beneficiar a vida da sociedade como um todo, automatizando diversos procedimentos e conseguindo ter uma maior clareza em relação aos dados e informações enviadas. A falta de transparência na comunicação sobre o tempo de espera e a disponibilidade de profissionais em hospitais tem gerado uma percepção negativa entre o público em relação à eficiência do atendimento hospitalar. Muitas pessoas evitam buscar assistência médica, a menos que seja absolutamente necessário, devido à incerteza em relação à rapidez do atendimento e à capacidade de acesso à informação dos profissionais especializados disponíveis. Esse paradigma impacta negativamente a utilização dos serviços de saúde e pode ter consequências graves para a saúde pública.

Do outro lado, o próprio estabelecimento também não tem nenhuma plataforma que lhe dê acesso a uma comunicação direta com seu paciente. Sendo assim, a troca de informações entre os dois acaba não sendo feita de maneira efetiva, prejudicando ambos os lados: o estabelecimento, que não consegue fornecer precisamente suas informações e situações em determinado momento, e o paciente, que não consegue ter grandes informações sobre a situação atual de determinado estabelecimento sem antes ter que se dirigir a ele, tornando o processo de ser atendido muito mais lento e prejudicando a experiência do usuário, em um processo simples que demanda tempo desnecessário.

Além disso, os próprios pacientes não possuem uma plataforma que forneça uma comunicação colaborativa entre eles. Ter tal ferramenta facilita as tomadas de decisões dos pacientes, sendo necessário criar uma plataforma que forneça esta comunicação direta entre os pacientes, funcionando como uma ferramenta de feedback. Considerando as questões previamente mencionadas, foi identificada uma oportunidade de desenvolver uma plataforma que possa resolver e automatizar os desafios enfrentados tanto pelo paciente quanto pelo estabelecimento. Essa solução visa proporcionar benefícios para ambas as partes e simplificar os processos envolvidos.

À medida que novas tecnologias surgem para solucionar problemas cotidianos, é notável a necessidade de cada vez mais soluções automatizadas na área da saúde, essenciais para otimizar procedimentos e beneficiar a sociedade. Diante dessa lacuna, foi necessário desenvolver um sistema que agilize o fornecimento de informações e o atendimento, visando melhorar um dos processos mais importantes nesse setor. Foi reconhecido que os conhecimentos adquiridos durante o curso foram aplicados, sendo fundamental para resolver os problemas mencionados anteriormente. O principal objetivo é desenvolver um sistema flexível para o público. Para isso, optaram por uma plataforma móvel para os pacientes, visando facilitar sua utilização e comunicação tanto entre eles quanto com os estabelecimentos de saúde. Por outro lado, a versão destinada aos estabelecimentos será web, permitindo fornecer informações precisas aos pacientes com rapidez.

Segundo a AWS (S.D), uma aplicação web é um software executado em um navegador da web, usado como um meio de troca de informações com os usuários de forma conveniente e segura. Esse tipo de aplicação possui diversos benefícios, entre eles estão sua acessibilidade, desenvolvimento eficiente, simplicidade para o usuário e sua escalabilidade. No que diz respeito a aplicações para dispositivos móveis, a Azure (2024) define quatro tipos de aplicações existentes:

- Aplicações Nativas: São específicas de um determinado dispositivo ou plataforma, como o Android ou o iOS.

- Aplicações Multiplataforma: Podem ser codificadas nas linguagens de sua escolha e posteriormente são compiladas para cada sistema operativo no qual a aplicação deseja ser executada.
- Aplicações Web Progressivas: Assim como as aplicações web, estes tipos de aplicações são executados em browsers, porém em um contexto móvel.
- Aplicações Híbridas: Sendo uma mistura entre aplicações nativas e progressivas, esta combinação faz com que a aplicação tenha acesso às funcionalidades e hardware do dispositivo, tornando-a possível de ser executada em diversos tipos de dispositivos.

A abordagem utilizada foi de criar uma aplicação híbrida, para aproveitar os conceitos familiares aos integrantes da equipe, facilitando o desenvolvimento do projeto. Além disso, essa abordagem permite tornar a aplicação mais flexível, garantindo sua acessibilidade em diversos tipos de dispositivos.

2 PROBLEMATIZAÇÃO DE PESQUISA

A demora no atendimento em filas de espera hospitalar é uma questão amplamente discutida na sociedade, conforme destacado pelos autores Krishnamurti, Shiro e Arthur (2005, p.256), que definem a fila como uma lista de pacientes necessitando de um mesmo tratamento ou serviço médico, cuja demanda excede a oferta. Nesse contexto, os sistemas foram concebidos com o propósito de oferecer soluções que beneficiassem a sociedade como um todo.

A tecnologia desempenha um papel fundamental no funcionamento de diversas atividades na sociedade, que enumera benefícios significativos, incluindo educação, comunicação, trabalho, processos produtivos e relações sociais. Esse contexto ressalta a importância da tecnologia na sociedade contemporânea. Diante disso, a pesquisa buscou explorar maneiras de utilizar a tecnologia para solucionar problemas na área da saúde.

A construção dessa plataforma é sem dúvida um desafio significativo, pois requer que a equipe saia de sua zona de conforto e lide com um tema que carece de muitos sistemas de referência para orientar o desenvolvimento. Isso demandou uma análise detalhada e a concepção do funcionamento do sistema, incluindo o uso de tecnologias com as quais não estávamos familiarizados.

3 SOLUÇÃO E HIPÓTESE

A plataforma HealthCare foi desenvolvida com o intuito de solucionar um dos principais desafios enfrentados pelos pacientes: a demora no atendimento médico devido ao alto volume de pacientes, o que resulta em frustração e demanda de tempo significativa para se locomover até o hospital e esperar na fila. Dependendo da gravidade da situação do paciente, é crucial que o atendimento seja realizado com a máxima celeridade possível.

> “[...] a espera na fila de um hospital: muitas vezes, ainda que os horários sejam marcados, acontece de a consulta ou o exame demorar mais do que o esperado, o que pode ser uma situação um tanto quanto estressante.” (DIAGRAD, S.D)

Pode-se concluir que a espera prolongada na fila de atendimento pode ser extremamente desgastante para o paciente, resultando em frustrações que impactam diretamente sua rotina diária. Diante desse cenário, foi identificado uma oportunidade de solucionar esse problema por meio da tecnologia. O relatório "Congestão e Superlotação dos Serviços Hospitalares de Urgência", publicado pelo Ministério da Saúde (2020), reforça essa percepção ao afirmar:

> “A utilização de tecnologia da informação e comunicação à distância, entre médicos, entre o médico e enfermeiros, entre o médico e o paciente, e entre o médico e a equipe para diversas ações de saúde, como consultas e laudos de exame deve ser ampliada”

Com base nessas considerações, pode-se afirmar que a tecnologia ainda não foi totalmente explorada para lidar efetivamente com as questões relacionadas à fila de atendimento. De acordo com Pereira (2023), a crescente utilização da tecnologia da tecnologia para otimizar custos e recursos em diversos projetos, acaba salientando a urgência de um método eficiente para melhorar o atendimento via SUS e ressaltando o potencial da tecnologia para salvar vidas.

O sistema desenvolvido para dispositivos móveis será completamente livre e aberto para qualquer usuário se cadastrar e utilizar todas as funcionalidades após autenticação, com interface extremamente fácil e simples, projetado especificamente para dispositivos móveis para maior acessibilidade ao público em geral.

Para uso dos hospitais, foi desenvolvida uma aplicação web prática e simples, facilitando o registro e fornecimento de informações sobre a disponibilidade de profissionais por especialidade após autenticação.

Diante das questões levantadas, evidencia-se a importância do projeto e sua proposta para resolver problemas, destacando a necessidade da tecnologia como solução para diversos desafios da sociedade, sendo essa a abordagem adotada para o desenvolvimento da ideia.

3.1 OBJETIVO GERAL

Através dos conhecimentos que foram adquiridos ao decorrer do curso, foi desenvolvido a plataforma HealthCare, na qual proporciona aos hospitais compartilharem suas informações para os seus pacientes além de auxiliar com publicidade, ao mesmo tempo que ajuda os pacientes a relatarem suas experiências em relação à fila de atendimento de determinado hospital. Essa abordagem visa promover uma experiência colaborativa, cujos relatos podem ser visualizados por outros usuários.

A plataforma planeja se destacar ao atribuir um papel de destaque ao paciente, permitindo-lhe total liberdade para relatar sua experiência ao se dirigir a determinado estabelecimento e interagir com outros usuários da plataforma. Além disso, será disponibilizado espaço para os hospitais se divulgarem e fornecerem aos seus pacientes informações úteis que facilitem a experiência para ambos os lados.

De maneira geral, a equipe não planeja simplesmente construir mais uma plataforma, mas sim moldar um meio em que os pacientes possam sentir-se mais seguros e confiantes ao decidir se deslocar para determinado hospital, sem qualquer dúvida sobre sua escolha. Ao mesmo tempo, fornecerão um método pelo qual os hospitais possam comunicar suas informações de forma precisa e eficaz aos seus pacientes.

3.2 OBJETIVOS ESPECÍFICOS

A área da saúde é reconhecida como uma das mais fundamentais para o progresso da sociedade, o que suscita a reflexão de que o desenvolvimento de soluções voltadas para esse setor tende a trazer benefícios para a comunidade em geral. O software proposto será responsável por aprimorar tanto a experiência do paciente, que busca atendimento ágil e satisfatório, quanto oferecer oportunidades para os hospitais se divulgarem e fornecerem informações cruciais. Para alcançar esse objetivo, a equipe utilizará as tecnologias dominadas em seu desenvolvimento.

A aplicação será dividida em duas partes distintas. Uma delas consistirá em uma versão para plataforma web, destinada aos hospitais para que possam registrar-se e fornecer informações sobre a disponibilidade de profissionais por especialidade. A outra parte será uma versão para dispositivos móveis, permitindo que os usuários localizem hospitais próximos, visualizem suas informações e forneçam feedback sobre determinados estabelecimentos.

4 JUSTIFICATIVA

Como mencionado anteriormente, o desenvolvimento de uma plataforma destinada a auxiliar a área da saúde é projetado para trazer benefícios para a sociedade como um todo, e é exatamente esse o propósito do nosso sistema. O objetivo é criar uma ferramenta que facilite a vida dos pacientes que buscam atendimento rápido em determinada unidade de saúde, permitindo-lhes visualizar as especialidades dos médicos locais. Além disso, os pacientes terão a liberdade de expressar seu feedback sobre a fila de atendimento específica daquela unidade, sendo que esse feedback será visível para todos os usuários que encontrarem essa unidade no aplicativo

Inicialmente, o aplicativo será focado na cidade de Praia Grande, onde existe uma lacuna nesse tipo de iniciativa, o que proporcionará uma vantagem significativa para nossa plataforma. Nos planos futuros, a conclui expansão para atender às necessidades dos pacientes em todo o Brasil está prevista.

Os objetivos da plataforma visam proporcionar uma melhoria significativa na interação entre o paciente e o hospital no qual ele deseja ser atendido. O foco não se limita apenas à melhoria na velocidade do atendimento do estabelecimento. Os sistemas estão sendo desenvolvidos para aprimorar a comunicação e o relacionamento do hospital com o paciente.

Ao mesmo tempo, busca-se oferecer liberdade ao paciente para relatar sua experiência a outros usuários. Isso faz com que o hospital assuma uma responsabilidade maior ao fornecer as informações necessárias, pois essas podem ser confrontadas por outros pacientes. Estes terão um meio de relatar sua experiência, que será visualizada por outros usuários, permitindo que avaliem se vale a pena ou não se deslocar para serem atendidos neste estabelecimento específico. Isso trará uma segurança maior para essa escolha.

4.1 RELEVÂNCIA SOCIAL

A área da saúde é, sem dúvida, uma das mais importantes para a sociedade como um todo. Com isso em mente, uma tecnologia que facilite questões e até mesmo evite situações nesta área traria diversos benefícios para a sociedade. Isso facilitaria a vida do paciente que busca atendimento rápido e eficiente, enquanto também simplificaria o lado dos hospitais. Eles poderiam se promover em uma plataforma e fornecer informações importantes para possíveis futuros pacientes que possam frequentar seus estabelecimentos.

À medida que a saúde se torna cada vez mais uma prioridade global, integrar a tecnologia como um meio de melhorar todos os processos nessa área é crucial. As duas aplicações desenvolvidas foram projetadas com a intenção de aprimorar os procedimentos realizados neste campo. A plataforma web serve como facilitador para os hospitais, enquanto a plataforma mobile atua como uma ferramenta de entrega de informação e feedback em tempo real para o paciente.

As aplicações não são apenas mais uma ferramenta produzida; o foco é transformá-las em catalisadores de grandes mudanças que afetem positivamente a sociedade. Ao simplificar a gestão dos hospitais, permite-se que os profissionais se concentrem nos pacientes, ao mesmo tempo em que facilita a vida do paciente, proporcionando acesso a informações transparentes e promovendo a comunicação direta entre os próprios pacientes.

Integrar a tecnologia é um passo crucial para aprimorar o setor da saúde. Ao conectar os hospitais com seus pacientes, visa-se construir uma sociedade mais resiliente, onde a informação se torna uma aliada na busca por um atendimento médico mais eficiente.

5 METODOLOGIA

A principal questão abordada é conseguir fornecer uma tecnologia que forneça uma plataforma de visualização de informações e de feedback em tempo real, por conta dessa questão, foi abordada a de pesquisa qualitativa. “É uma pesquisa focada em entender aspectos mais subjetivos, como comportamentos, ideias, pontos de vista, entre outros.” afirma o autor Lucas Mathias (2022), tempo em mente essa definição, utilizamos deste tipo de pesquisa para conseguir obter um panorama geral sobre os pontos principais que insatisfazem os pacientes em relação a fila de espera de um hospital.

Todo o processo de pesquisa teve como referência a leitura de literatura especializada sobre o tema abordado, a experiência prévia da equipe e conhecimentos sobre o assunto, bem como as opiniões de profissionais que pudessem orientar o rumo do projeto. Por fim, conduziu-se uma pesquisa de campo para obter uma variedade de pontos de vista. Todos esses procedimentos permitiram à equipe chegar a uma conclusão sobre o assunto e como utilizar os conhecimentos em tecnologia para resolver todas as questões levantadas durante esses procedimentos.

Essa pesquisa teve ênfase em conseguir detectar as causas e padrões por trás de todos os problemas levantados e estudados, com o intuito de conseguir identificar as possíveis soluções que podem ser utilizadas alinhadas a diversas tecnologia para conseguir resolver os problemas levantados.

Durante toda a fase de planejamento e posteriormente na fase de desenvolvimento, a equipe baseou-se em diversos conceitos da metodologia ágil SCRUM, uma das mais estabelecidas e utilizadas no mercado como um todo. “Todos os tipos de equipes, como RH, marketing e design, usam o Scrum de maneira eficaz. Porém, o Scrum se destaca mais em equipes de desenvolvimento e engenharia de software.” (AWS, S.D). Um dos principais conceitos adotados pela equipe foram as reuniões regulares, realizadas frequentemente para alinhar ideias da equipe e acompanhar o desenvolvimento do sistema.

Por se basear na metodologia Scrum, a utilização das Sprints foi fundamental. As Sprints consistem na divisão do trabalho dos membros da equipe em períodos fixos,

que variam de duas a quatro semanas (Awari, 2023). A variação do tempo foi definida pela equipe com base na complexidade da implementação de cada funcionalidade, concentrando todos os esforços dos integrantes para alcançar a implementação bem-sucedida de novas funcionalidades.

Ao longo do desenvolvimento do projeto, nem todas as funcionalidades implementadas foram planejadas desde o início. Isso tornou as Sprints essenciais para manter o foco da equipe no que precisava ser implementado. No início de cada Sprint, era realizado um planejamento dos requisitos das funcionalidades e de como cada integrante contribuiria para sua implementação.

A característica da metodologia Scrum que mais impactou a equipe foi seu foco na colaboração entre os membros do grupo. Isso facilitou o desenvolvimento do sistema e, ao mesmo tempo, permitiu que os integrantes da equipe se ajudassem em tarefas difíceis, promovendo aprendizado e agilizando o processo de desenvolvimento.

É importante ressaltar que a construção do projeto foi baseada na metodologia Scrum, embora não tenha seguido todos os seus conceitos rigidamente. Foi adaptada de uma maneira que a equipe considerou mais fácil para o desenvolvimento do projeto e que se adequasse ao perfil de cada um dos integrantes. Graças a essas práticas, o processo de desenvolvimento das aplicações se tornou mais eficiente e eficaz, garantindo a implementação bem-sucedida de todas as funcionalidades.

6 PLANEJAMENTO DO PROJETO

As aplicações têm como objetivo facilitar tanto a vida do hospital quanto a do paciente. O hospital pode se promover na plataforma web e informar a quantidade de profissionais disponíveis por especialidade. Por sua vez, o paciente, através da plataforma mobile, consegue visualizar essas informações sobre o hospital, ler os feedbacks de outros pacientes e deixar seu próprio feedback sobre a situação de determinado hospital.

Para desenvolver a aplicação web, foi utilizada a linguagem de programação JavaScript em conjunto com o *framework*[1] React, possibilitando a criação de um sistema simples capaz de atender às necessidades dos hospitais que utilizam essa plataforma. Já para a aplicação mobile, também foi usada a linguagem de programação JavaScript, porém com o framework React Native, que permite a criação de aplicativos para dispositivos móveis.

Por fim, para conectar o sistema ao banco de dados, foi utilizado o framework Express com a tecnologia NodeJS e Typescript. O Express é amplamente utilizado para a criação de *API's*[2], garantindo uma conexão eficiente entre o sistema e o banco de dados.

O framework React é bastante escolhido para a criação de aplicações *SPA*[3], que fornecem uma melhor experiência para o usuário que a está utilizando, além de facilitar o processo de comunicação entre o front-end com o back-end da aplicação e

---

[1] Framework é um conjunto de técnicas, ferramentas ou conceitos pré-definidos usados para resolver um problema de um projeto ou domínio específico. [1] https://www.escoladnc.com.br/blog/o-que-sao-frameworks/

[2] API significa Application Programming Interface, se trata de mecanismos que permitem que dois componentes de software se comuniquem usando um conjunto de definições e protocolos. [2] https://aws.amazon.com/pt/whatis/api/#:~:text=API%20significa%20Application%20Programming%20Interface,de%20servi%C3%A7o%20entre%20duas%20aplica%C3%A7%C3%B5es.

[3] SPA significa Single Page Application, são utilizadas para uma melhor experiência do usuário durante a navegação entre páginas. [3] https://www.treinaweb.com.br/blog/spa-e-ssr-quais-as-diferencas#:~:text=SPA%20%C3%A9%20uma%20sigla%20para,ter%C3%A3o%20apenas%20uma%20%C3%BAnica%20p%C3%A1gina.

conseguir dividir o projeto em várias componentes, que podem ser reaproveitados em diversas páginas do sistema, acelerando o processo de desenvolvimento da aplicação web.

O framework React Native foi selecionado devido à sua semelhança com o React, porém projetado para atender a todos os sistemas operacionais de dispositivos móveis. É uma tecnologia extremamente flexível, adequada para todos os públicos, e oferece bom desempenho, proporcionando uma excelente experiência ao usuário que utiliza a aplicação.

Para o back-end da aplicação, foi utilizado o framework ExpressJS em conjunto com a tecnologia Typescript. Essa escolha foi feita devido à facilidade que essa combinação oferece na criação de uma API que interage com o banco de dados e envia todas as informações necessárias para o front-end, tanto da plataforma web quanto da plataforma de dispositivos móveis.

Todas essas tecnologias têm como base a linguagem de programação Javascript. Segundo o autor Munhas (2023), o Javascript é altamente valorizado por sua capacidade de desenvolver uma variedade de aplicações em diversos ambientes. No caso do projeto, essa linguagem permite que seja utilizado tanto no ambiente web quanto no ambiente de dispositivos móveis, o que a torna essencial devido à sua flexibilidade, juntamente com todos os frameworks que são baseados nessa linguagem.

6.1 PROTOTIPAÇÃO

Para proporcionar uma visão geral de como funcionaria o fluxo dos sistemas, foram desenvolvidos dois tipos de protótipos. O primeiro foi um protótipo de baixa fidelidade, que serviu como ponto de partida para conceber como as aplicações operariam e para identificar os requisitos necessários. Posteriormente, foi criado um protótipo de alta fidelidade, projetando a aparência final do sistema, sendo muito próximo do resultado final.

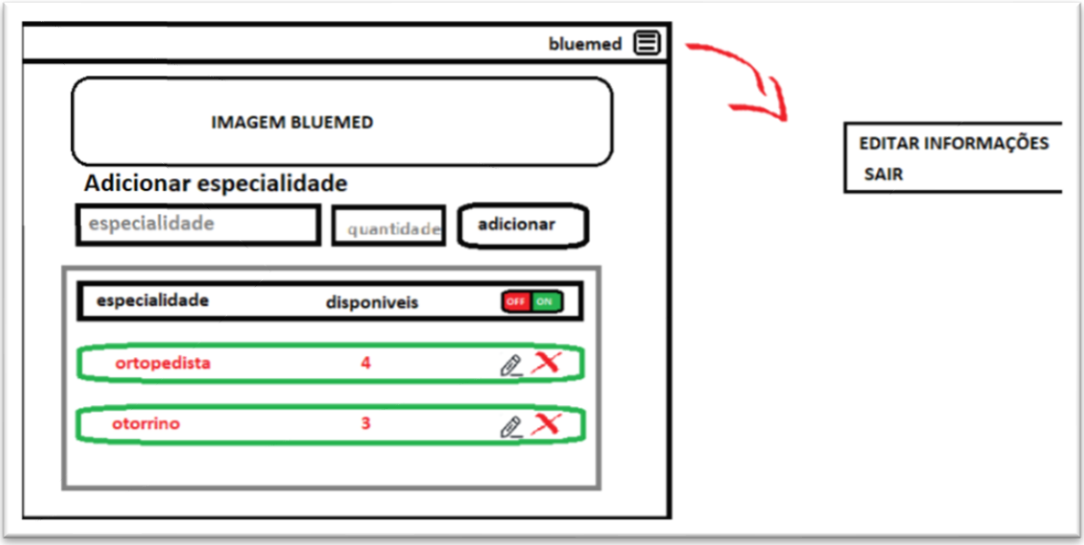
6.2 PROTÓTIPO DE BAIXA FIDELIDADE

Esses protótipos são feitos antes de começar o desenvolvimento do sistema, para ter certeza se vale a pena ou não o investimento no projeto, assim como define o site SoftDesign:

> “Imagine o seguinte: você tem uma ideia de um novo produto digital, e a você parece que ele poderia resolver o problema de muitas pessoas. Pode ser um novo aplicativo ou até mesmo um novo sistema que irá facilitar o trabalho de empresas de um determinado ramo. Mas, para ter certeza antes de fazer um alto investimento, é preciso planejar, testar e avaliar essa ideia. Nesse caso, um protótipo de baixa fidelidade pode ser uma das formas de começar a validar e amadurecer a ideia.” (SOFTDESIGN, 2020)

Assim como o diagrama de caso de uso, foi utilizado a ferramenta Draw IO para desenvolver o diagrama de classes, a partir da figura 1 é possível visualizar a representação gráfica do diagrama desenvolvido:

Figura 1 - Protótipo de baixa fidelidade



Fonte: Elaborado pelos autores, 2024.

6.3 PROTÓTIPO DE ALTA FIDELIDADE

Posteriormente, após os avanços na ideia, baseados no que o protótipo de baixa fidelidade foi capaz de proporcionar, as telas do protótipo de alta fidelidade do sistema foram criadas. Esse estágio se aproximou muito mais da versão final das telas do sistema.

> “Um protótipo de alta fidelidade (às vezes chamado de high-fi ou hi-fi) é uma representação interativa do produto, baseada no computador ou em dispositivos móveis. Esse protótipo já apresenta maior semelhança com o design final em termos de detalhes e funcionalidade.” (MEDIUM, 2020)

Após a finalização das telas desse protótipo, foi possível iniciar a programação das telas do próprio sistema. Nas figuras 2, 3 e 4, serão apresentadas as principais telas que foram desenvolvidas para serem integradas ao sistema:

Figura 2 - Protótipo de alta fidelidade página inicial.

Fonte: Elaborado pelos autores, 2024.

Figura 3 - Protótipo de alta fidelidade página entrar.

Fonte: Elaborado pelos autores, 2024.

Figura 4 - Protótipo de alta fidelidade página especialidades.

Fonte: Elaborado pelos autores, 2024.

6.4 ENTIDADES DO SISTEMA

Pensar em quais serão as entidades em seu sistema é uma parte fundamental para conseguir dar início ao desenvolvimento de um projeto, segundo a definição do site DevMedia (2007), as entidades são representações de várias informações sobre algum determinado conceito do sistema, possuindo atributos, que são as informações que irão referenciá-las. Parafraseando, uma entidade é feita para representar de forma clara alguma função dentro de um determinado negócio.

No artigo publicado na Alura por Almeida (2023), o conceito de entidades é especificado da seguinte maneira: "As entidades são objetos ou conceitos do mundo real que são representados no banco de dados." Com base nessa definição, ao identificarmos as entidades em um sistema, estamos reconhecendo todas as representações reais que terão interações com a aplicação desenvolvida.

Definir as entidades do sistema é um passo fundamental para posteriormente conseguir dar início a modelagem do banco de dados, mas também tem sua importância para poder dar sequência no projeto como um todo, onde cada entidade terá um papel diferente na forma como atuará nas diferentes separações do sistema, assumindo papéis diferentes no contexto na qual a entidade será utilizada.

São as entidades do nosso sistema que serão responsáveis por definir a estrutura do banco de dados, para que posteriormente, cada uma delas se torne uma tabela no banco de dados, sendo que os atributos delas se tornarão as colunas desta tabela que foi criada.

Para o front end tanto da versão Web, quanto da versão mobile do sistema, as entidades serão responsáveis por guiar o desenvolvimento das interfaces e componentes do sistema, já que eles serão criados tendo como parâmetros as regras de negócios, para que assim os hospitais e pacientes consigam interagir com o sistema da maneira que foi planejado.

Como no back end são utilizados dados que ou chegam diretamente do banco de dados, ou são enviados para lá, as entidades aparecem como variáveis do sistema, que serão representadas conforme a necessidade de interação que esta parte terá com o banco de dados.

Após criteriosa análise, as seguintes entidades foram identificadas para serem implementadas:

- Pacientes;
- Hospitais;
- Profissionais Disponíveis;
- Exames;
- Comentários.

Essas entidades serão utilizadas em todos os contextos da aplicação, seja no front end, back end, banco de dados, requisitos e diagramas UML.

6.5 REQUISITOS FUNCIONAIS

Os requisitos funcionais definem como o sistema deve ser utilizado e moldado para atender às necessidades dos usuários.

Esses requisitos são visíveis para os usuários e se traduzem em funcionalidades que permitem interações através da interface fornecida pela aplicação. A seguir, são apresentados os requisitos funcionais identificados:

Tabela 1 - Requisitos Funcionais

[RF001] CADASTRAR USUÁRIO
Descrição do caso de uso: O sistema deverá permitir o cadastro de pacientes e de empresa, com cada um desses cadastros com as devidas informações necessárias a serem fornecidas.
[RF002] AUTENTICAR USUÁRIO

Descrição do caso de uso: O sistema deverá permitir o cadastro de pacientes e de empresa, com cada um desses cadastros com as devidas informações necessárias a serem fornecidas.
[RF003] REALIZAR LOGIN E LOGOFF
Descrição do caso de uso: O sistema deverá fornecer em sua interface da possibilidade de o usuário conseguir se conectar e desconectar da aplicação.
[RF004] CONTROLE DAS INFORMAÇÕES DOS USUÁRIOS
Descrição do caso de uso: O sistema deverá fornecer para os pacientes e empresas a possibilidade de visualizar, alterar e remover suas informações cadastradas.
[RF005] CONTROLE DE PROFISSIONAIS DISPONÍVEIS POR ESPECIALIDADE
Descrição do caso de uso: O sistema deve permitir que as empresas controlem os profissionais disponíveis por especialidade e a quantidade que ela possui disponível, além da remoção em caso de necessidade.
[RF006] CONTROLE DE EXAMES
Descrição do caso de uso: A aplicação mobile será responsável por fornecer através de sua interface a possibilidade de o paciente cadastrar, visualizar e remover seus exames.
[RF007] VISUALIZAR ESTABELECIMENTOS ABERTOS
Descrição do caso de uso: O sistema deve fornecer para o paciente a funcionalidade de visualizar todos os estabelecimentos abertos e suas informações principais, além de listar a quantidade de profissionais disponíveis por especialidade da área de saúde naquele determinado estabelecimento.
[RF008] REALIZAR COMENTÁRIO
Descrição do caso de uso: A aplicação mobile deverá fornecer ao paciente a possibilidade de realizar um comentário sobre determinada empresa que está cadastrada no sistema.
[RF009] LISTAR COMENTÁRIOS
Descrição do caso de uso: No sistema deverá ser possível o paciente visualizar os comentários de outros pacientes sobre uma determinada empresa.

6.6 REQUISITOS NÃO FUNCIONAIS

Utilizados para descrever as limitações e restrições do sistema projetado, os requisitos não funcionais, ao contrário dos funcionais, especificam como o sistema executa suas funcionalidades. Já os requisitos funcionais descrevem as funcionalidades propriamente ditas (V/SURE, 2024).

Por se tratar de uma aplicação para plataformas Web e Mobile, foi necessária a identificação precisa dos requisitos não funcionais. Abaixo estão os requisitos não funcionais levantados:

Tabela 2 - Requisitos não funcionais

[RNF010] CONEXÃO COM BANCO DE DADOS
Descrição do caso de uso: Todas as aplicações e suas camadas deverão se comunicar com o banco de dados feito com a linguagem MySQL.
[RNF011] PORTABILIDADE
Descrição do caso de uso: A aplicação Web deverá estar disponível para plataformas desktop e mobile, enquanto a aplicação Mobile deve estar disponível para todos os possíveis sistemas operacionais de dispositivos móveis.
[RNF012] SEGURANÇA
Descrição do caso de uso: Todas as informações sensíveis dos usuários deverão estar criptografadas e o acesso ao banco de dados só deverá ser realizado pelos administradores da aplicação.
[RNF013] VALIDAÇÕES
Descrição do caso de uso: Para realizar o cadastro, informações como CEP e CNPJ serão todas validadas para verificação das informações inseridas pelos usuários.
[RNF014] CONEXÃO COM A INTERNET
Descrição do caso de uso: O sistema deve manter uma conexão de maneira ativa em todos os momentos com a internet para o envio e uso de dados.

6.7 DIAGRAMAS UML (LINGUAGEM DE MODELAGEM UNIFICADA)

Após o levantamento de todas as entidades do sistema, foi possível dar início ao desenvolvimento dos diagramas *UML*, que segundo a definição do Lucidchart (2024), foi uma linguagem criada com o intuito de conseguir estabelecer uma linguagem de modelagem visual comum, que seja rica tanto semanticamente quanto sintaticamente, visando servir para arquitetura, design e posteriormente implementar sistemas de softwares complexos.

Os diagramas UML utilizam o paradigma POO (Programação Orientada a Objetos), esse conceito está presente em diversas etapas do desenvolvimento de uma aplicação, por conseguir retratar e modelar objetos do mundo real no contexto de desenvolvimento, esse conceito facilita a criação de softwares que são mais fáceis de manter, expandir e ainda promove práticas de programação mais eficientes e seguras.

> “A Programação Orientada a Objetos (POO) é um paradigma de programação que revolucionou a forma como desenvolvemos software. Baseada em conceitos como encapsulamento, herança e polimorfismo, a POO oferece uma abordagem estruturada e modular para o desenvolvimento de sistemas complexos. Neste artigo, exploraremos as vantagens da programação orientada a objetos e como ela contribui para a criação de código mais eficiente, reutilizável e fácil de manter.” (DIO, 2023)

Segundo a definição feita pela Lucidchart, a UML utiliza os principais conceitos da Programação Orientada a Objetos para conseguir apresentar uma metodologia mais consiste e fácil de utilizar, por conta desses e de outros diversos fatores, essa linguagem representa as melhores práticas para desenvolver e documentar os aspectos diferentes da modelagem de software ou de sistemas de negócios.

Como existem vários tipos de diagramas UML que podem ser criados para servir como uma ferramenta para o desenvolvimento do projeto, utilizou-se neste projeto três destes diagramas, sendo eles:

- **Diagrama de Caso de Uso**: Um diagrama que é utilizado como uma representação gráfica para ilustrar como um sistema deve interagir com seus usuários (que nesse caso são chamados de “atores”).

- **Diagrama de Classe**: Utilizado para ilustrar a estrutura de um software, exibindo todas as classes do sistema, atributos, métodos e as relações entre as classes estabelecidas.
- **Diagrama de Entidade e Relacionamento**: É uma representação gráfica que é utilizada no design de banco de dados, sendo usada para modelar os dados e as relações entre eles.

No próximo tópico, será detalhado a definição de cada um desses diagramas e sua importância para o desenvolvimento da aplicação, dando ênfase maior para cada um deles e suas devidas explicações e conceitos.

6.7.1 DIAGRAMAS DE CASO DE USO

Esses diagramas exercem um papel fundamental no que diz respeito a interação do usuário com o sistema, sendo o usuário retratado como “ator” nesse contexto, os casos de uso e os atores descrevem o que o sistema faz e a maneira em que os usuários o utilizam, mas não entrando no mérito de como o sistema funciona e opera internamente (IBM 2021).

Essa é uma ferramenta de grande importância para a modelagem dos requisitos do software, sendo uma representação gráfica simples e intuitiva, facilitando o entendimento das funcionalidades oferecidas pelo sistema e deixando claro os caminhos que os desenvolvedores usarão durante a criação da aplicação.

Os diagramas de caso de uso são representados por diversos fatores, porém seus elementos principais são:

- **Atores**: São representados por bonecos no estilo palito, representando os usuários ou sistemas externos que irão interagir com o sistema.
- **Caso de Uso**: Representado por uma elipse, são utilizadas para representar uma sequência específica de ações e interações dos atores com o sistema.

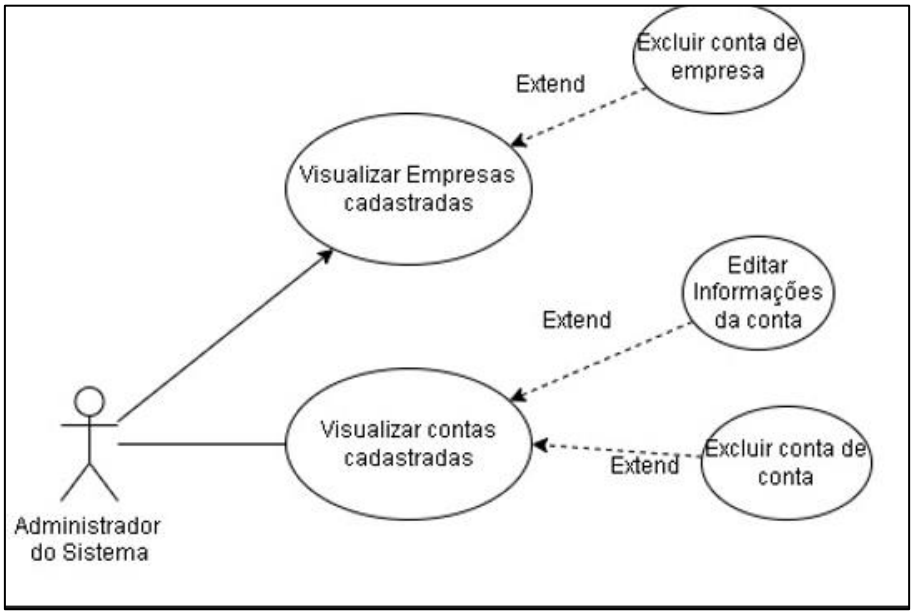
- **Relações**: As relações podem ser associações (linhas entre atores e casos de uso), generalização (definir hierarquias entre os atores) e as relações de inclusão e extensão (para indicar como os casos de uso estão relacionados).

A criação desse diagrama foi crucial para dar início ao desenvolvimento da aplicação, servindo como referência de como funcionaria diversas das funcionalidades do software, os atores identificados foram:

- **Público**: Sendo tanto a empresa quanto o paciente, esse ator exemplifica como o usuário possuí a opção de se cadastrar em alguma dessas duas opções.
- **Empresa**: Se o usuário escolher se cadastrar como empresa, ele poderá ter outras ações no sistema, como visualizar as informações cadastrar e informar a quantidade de especialistas disponíveis.
- **Paciente**: Após autenticação, conseguirá visualizar suas próprias informações cadastradas e todas as empresas do sistema, tendo a opção importante de realizar um comentário sobre determinada empresa.
- **Administrador**: Possuindo um dashboard para conseguir ter controle tanto dos pacientes cadastrados, quanto das empresas.

Para elaborar do diagrama de caso de uso, utiliza-se a ferramenta Draw IO, sendo uma ferramenta bastante conhecida para a criação de diagramas UML de maneira rápida e efetiva, abaixo está disponível as representações gráficas dos diagramas, detalhando todos os processos envolvidos.

Figura 5 – Caso de uso administrador do sistema.



Fonte: Elaborado pelos autores, 2024

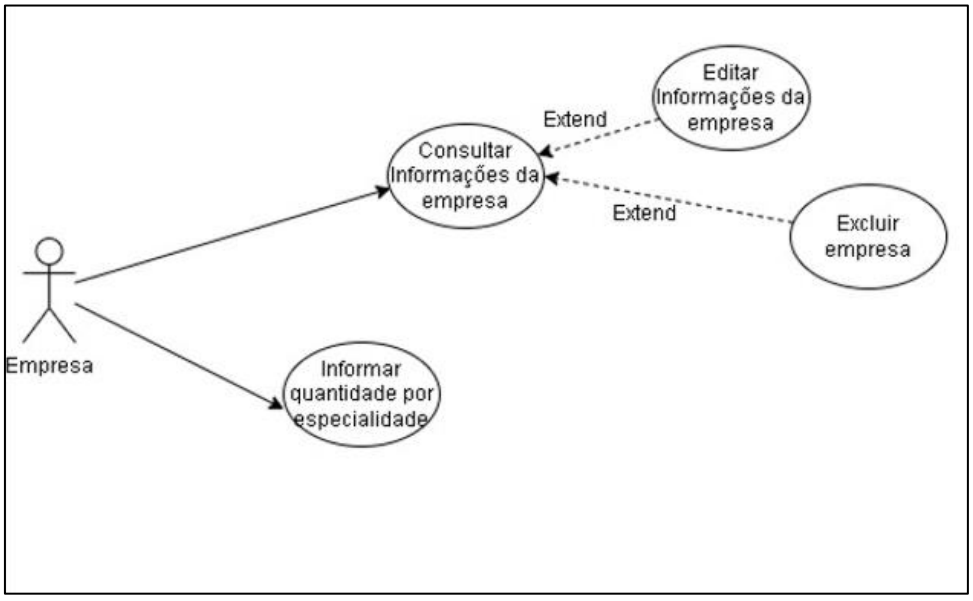
A figura 5 descreve um dos diagramas de caso de uso, especificamente focado nas funcionalidades acessíveis pelo Administrador do Sistema. O diagrama apresenta dois casos de uso principais, cada um com suas extensões. Conforme demonstrado na imagem a seguinte especificação e detalhamento dele é descrito abaixo:

- Visualizar Empresas Cadastradas: Este caso de uso permite que o Administrador do Sistema visualize todas as empresas cadastradas no sistema.
- Excluir Conta de Empresa: Este caso de uso estendido permite que o Administrador do Sistema exclua a conta de uma empresa específica. É um comportamento adicional que pode ser invocado durante a visualização das empresas cadastradas.
- Visualizar Contas Cadastradas: Este caso de uso permite que o Administrador do Sistema visualize todas as contas de usuários cadastradas no sistema.
- Editar Informações da Conta: Este caso de uso estendido permite que o Administrador do Sistema edite as informações de uma conta específica. Esta funcionalidade pode ser acessada durante a visualização das contas cadastradas.

- Excluir Conta de Conta: Este caso de uso estendido permite que o Administrador do Sistema exclua uma conta de usuário específica. Assim como a edição de informações, esta funcionalidade é um comportamento adicional acessível durante a visualização das contas cadastradas.

Cada uma das extensões é representada por uma linha pontilhada no diagrama, indicando que são comportamentos adicionais que se estendem dos casos de uso principais, e que podem ser executados dependendo da necessidade do Administrador do Sistema enquanto ele realiza as operações principais de visualização.

Figura 6– Caso de uso empresa.



Fonte: Elaborado pelos autores, 2024

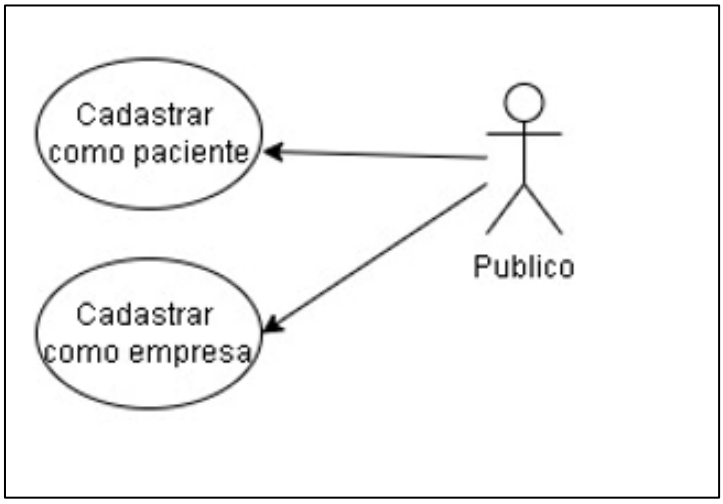
A figura 6 descreve o diagrama de caso de uso para a entidade Empresa no sistema. O diagrama apresenta dois casos de uso principais, cada um com suas respectivas extensões. Conforme demonstrado na imagem a seguinte especificação e detalhamento dele é descrito abaixo:

- Consultar Informações da Empresa: Este caso de uso permite que a Empresa consulte suas próprias informações cadastradas no sistema.

- Editar Informações da Empresa: Este caso de uso estendido permite que a Empresa edite suas próprias informações. Este é um comportamento adicional que pode ser invocado durante a consulta das informações da empresa.
- Excluir Empresa: Este caso de uso estendido permite que a Empresa exclua seu próprio registro do sistema. Assim como a edição, esta é uma funcionalidade adicional que pode ser acessada durante a consulta das informações da empresa.
- Informar Quantidade por Especialidade: Este caso de uso permite que a Empresa informe a quantidade de recursos, pessoas ou itens por especialidade. Este caso de uso não possui extensões associadas, conforme mostrado no diagrama.

As extensões são representadas por linhas pontilhadas no diagrama, indicando que são comportamentos adicionais aos casos de uso principais e podem ser acionados conforme necessário enquanto a Empresa realiza as operações principais de consulta das informações.

Figura 7 – Caso de uso público.



Fonte: Elaborado pelos autores, 2024

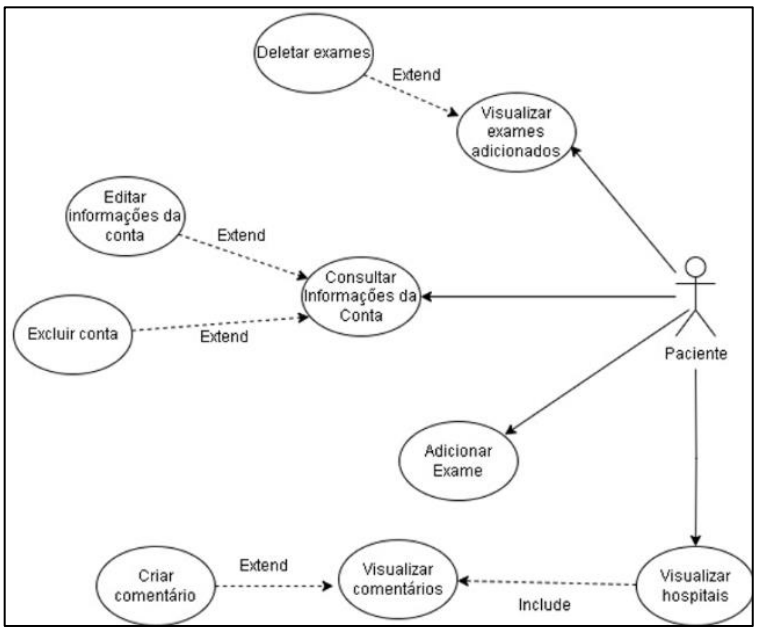
A figura 7 descreve o diagrama de casos de uso para a entidade Público no sistema. O diagrama apresenta dois casos de uso principais, sem extensões

associadas. Conforme demonstrado na imagem a seguinte especificação e detalhamento dele é descrito abaixo:

- Cadastrar como Paciente: Este caso de uso permite que um usuário do público se cadastre no sistema como paciente. É um processo em que o usuário fornece suas informações pessoais necessárias para ser registrado como paciente no sistema.
- Cadastrar como Empresa: Este caso de uso permite que um usuário do público se cadastre no sistema como empresa. Neste caso, o usuário fornece as informações necessárias para registrar a empresa no sistema.

Ambos os casos de uso são acessados diretamente pela entidade Público, que representa qualquer pessoa ou organização que deseja se registrar no sistema.

Figura 8 – Caso de uso paciente.



Fonte: Elaborado pelos autores, 2024

A figura 8 descreve o diagrama de casos de uso e interações entre um paciente e um sistema de gerenciamento de informações de saúde. Ele mostra várias funcionalidades que o paciente pode executar, além de como algumas

funcionalidades estão relacionadas entre si através de relações de extensão ("extend") e inclusão ("include"). Conforme demonstrado na imagem as seguintes especificações e relações deles é descrito abaixo:

- Consultar Informações da Conta: Caso de uso principal que permite ao paciente visualizar as informações da conta.
- Editar Informações da Conta: Extensão deste caso de uso, permitindo ao paciente modificar as informações da conta.
- Excluir Conta: Extensão deste caso de uso, permitindo ao paciente excluir a conta do sistema.
- Adicionar Exame: Caso de uso que permite ao paciente adicionar informações sobre exames médicos ao sistema.
- Visualizar Exames Adicionados: Incluído no caso de uso "Adicionar Exame", permitindo ao paciente ver os exames que foram adicionados.
- Deletar Exames: Extensão do caso de uso "Visualizar Exames Adicionados", permitindo ao paciente deletar exames previamente adicionados.
- Visualizar Comentários: Caso de uso que permite ao paciente ver comentários feitos no sistema.
- Criar Comentário: Extensão deste caso de uso, permitindo ao paciente adicionar novos comentários ao sistema.
- Visualizar Hospitais: Caso de uso que permite ao paciente ver a lista de hospitais disponíveis no sistema.

As relações de extensão indicam funcionalidades adicionais ou alternativas que estão disponíveis dependendo do contexto de uso do caso principal. Já as inclusões indicam funcionalidades que são necessariamente parte do caso de uso principal.

Essas descrições ajudam a entender como o paciente pode interagir com o sistema e como essas interações são organizadas e relacionadas entre si.

6.7.2 DIAGRAMA DE CLASSES

Responsáveis por estabelecer todas as classes do sistema, seus atributos e de que maneira as classes se relacionam entre si, esses diagramas são de extrema importância para facilitar o desenvolvimento da aplicação, visto que servem como base para o fluxo do sistema, a Lucidchart (2024) faz a seguinte definição destes tipos de diagrama: “Diagramas de classes estão entre os tipos mais úteis de diagramas UML pois mapeiam de forma clara a estrutura de um determinado sistema ao modelar suas classes, seus atributos, operações e relações entre objetos.”, a partir desta definição, fica claro a importância destes diagramas para o desenvolvimento.

Esses diagramas são responsáveis por trazer uma série de benefícios que incluem a ilustração de modelos de dados, uma melhor compreensão da visão geral de um sistema, uma representação gráfica que destaca a maneira que o código deverá ser programado e consegue fornecer uma descrição independente de uma implementação de tipos utilizados no sistema Lucidchart (2024).

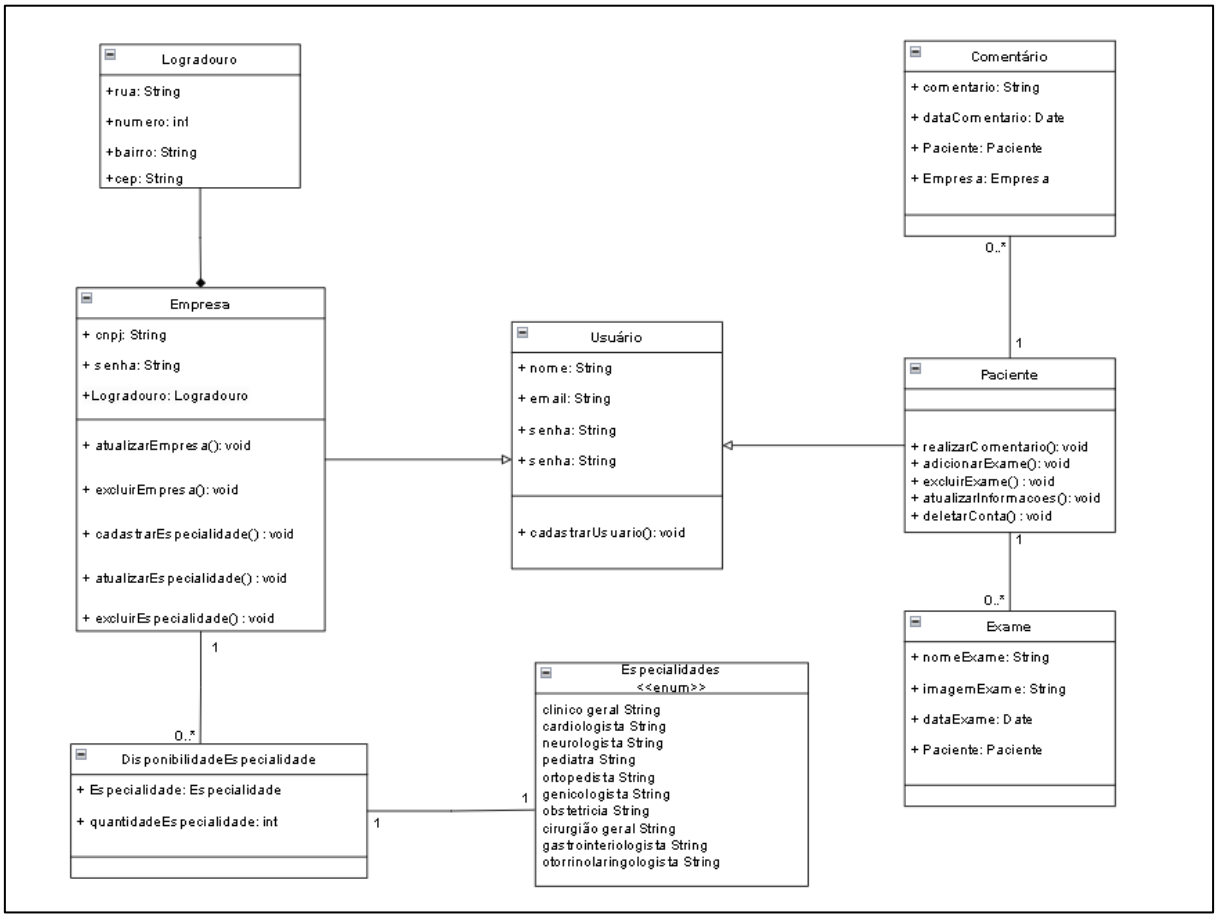
Para desenvolver as classes deste sistema, foram utilizados alguns dos conceitos já estabelecidos no diagrama de caso de uso, porém adaptando para esse novo contexto, além de algumas classes adicionais para se encaixar nas necessidades que enxergamos, as principais classes criadas foram as seguintes:

- **Usuário**: A classe principal do sistema que serve como base tanto para a Empresa, quanto para o Paciente, possuindo os atributos nome, e-mail e senha.
- **Empresa**: Possuindo uma relação de associação com a classe Usuário, porém também definindo seus atributos próprios como cnpj, telefone, logradouro e disponibilidadeEspecialidade.
- **Paciente**: Assim com a classe Empresa, possui com relação de associação com a classe Usuário, porém, não possuindo nenhum atributo própria, apenas aqueles já definidos em Usuário.

- **Comentário**: Possuindo relacionamento de associação tanto com a classe Empresa, quanto com a Usuário, possui os atributos comentário e dataComentário.
- **Disponibilidade Especialidade**: Possui os nomes das especialidades já definidos, sendo necessário apenas informar o atributo quantidadeEspecialidade.

Assim como o diagrama de caso de uso, foi utilizado a ferramenta Draw IO para desenvolver o diagrama de classes, a figura 9 se trata da representação gráfica do diagrama desenvolvido:

Figura 9 - Diagrama de Classes



Fonte: Elaborado pelos autores, 2024

6.7.3 DIAGRAMA DE ENTIDADE E RELACIONAMENTO

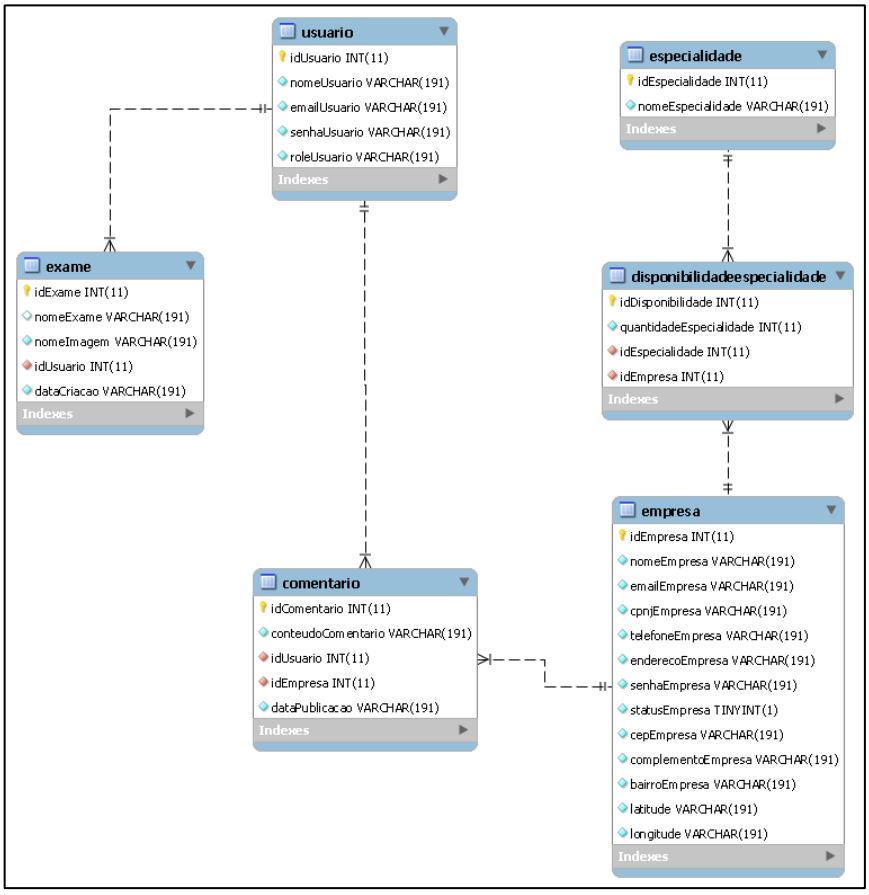
De acordo com a Lucidchart (2024), os diagramas de entidade e relacionamento (DER) são representações gráficas que ilustram como as entidades se relacionam entre si dentro de um sistema, sendo eles mais utilizados para projetor ou depurar bancos de dados relacionais, utilizando um conjunto definido de símbolos para representar a conexão entre as entidades e seus atributos.

Esses diagramas são compostos por entidades, atributos e os relacionamentos entre as entidades, sendo que cada um desses aspectos é parte fundamental para a criação deste diagrama, de maneira resumida, cada um desses aspectos possui a seguinte definição:

- **Entidades**: Representam objetos ou conceitos do sistema, sendo identificados de maneira única, como por exemplo, uma entidade “Aluno”, “Professor”, “Escola” etc.
- **Atributos**: São as características que definem uma entidade, como por exemplo, um atributo da entidade “Pessoa” seria “Nome”.
- **Relacionamentos de Um para Um (1:1)**: Uma instância de uma entidade está somente relacionada a uma única instância de outra entidade.
- **Relacionamentos de Um para Muitos (1:N)**: Quando uma instância de uma entidade está relacionada a mais de uma instância de outra entidade.
- **Relacionamentos de Muitos para Muitos (N:N)**: Casos extremos quando múltiplas instâncias de uma entidade estão relacionadas a outra múltiplas instâncias de outra entidade.

Para criação desse diagrama, utiliza-se a ferramenta MYSQL Workbench, que é uma ferramenta que fornece não apenas a possibilidade de modelagem de dados, mas também o desenvolvimento do banco, sendo um software extremamente versátil e utilizado, na figura 10 é demonstrado a representação gráfica do diagrama de entidade e relacionamento.

Figura 10 - Diagrama de Entidade e Relacionamento



Fonte: Elaborado pelos autores, 2024.

Abaixo é possível visualizar o dicionário de dados do nosso Diagrama de Entidade e Relacionamento, que possui o seguinte significado: “O dicionário de dados contém informações sobre as restrições de integridade definidas no banco de dados, como chaves primárias, chaves estrangeiras e restrições de exclusão.”, (AWARI, 2023).

Tabela 3 - Dicionário de Dados da Tabela de Usuários

Tabela	Usuário
Descrição	Armazena todos os usuários cadastrados
Observações	

CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
IdUsuario	Código de identificação de usuário	Int	11	PK / Identity
nomeUsuario	Nome do usuário	Varchar	191	Not Null
emailUsuario	Email do Usuário	Varchar	191	Unique / Not Null
senhaUsuario	Senha do Usuário	Varchar	191	Not Null
roleUsuario	Função do Usuário dentro do sistema (Administrador ou Usuário Comum)	Varchar	191	Not Null / Default

Tabela 4 - Dicionário de Dados da Tabela de Exames

Tabela	Exame			
Descrição	Armazena todos os exames cadastrados pelos usuários			
Observações	Possui uma chave estrangeira da tabela usuário			
CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
IdExame	Código de identificação de cada exame	Int	11	PK / Identity
nomeExame	Nome registrado pelo usuário para o exame	Varchar	191	Not Null

nomeImagem	Nome da imagem do exame	Varchar	191	Unique / Not Null
dataCriacao	Data que o exame foi adicionado	Varchar	191	Not Null / Default
idUsuario	Identificador do usuário que cadastrou o exame	Int	11	FK

Tabela 5 - Dicionário de Dados da Tabela de Empresa

Tabela	Empresa			
Descrição	Armazena todos as empresas cadastradas			
Observações				
CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
idEmpresa	Código de identificação de Empresa	Int	11	PK / Identity
nomeEmpresa	Nome da Empresa	Varchar	191	Not Null
emailEmpresa	Email da Empresa	Varchar	191	Unique / Not Null
cnpjEmpresa	Cnpj da Empresa	Varchar	191	Unique / Not Null
telefoneEmpresa	Telefone da Empresa	Varchar	191	Not Null / Default
enderecoEmpresa	Endereço da Empresa	Varchar	191	Not Null
senhaEmpresa	Senha da Empresa	Varchar	191	Not Null
statusEmpresa	Status da Empresa (aberta ou fechada)	Tinyint	1	Not Null / Default
cepEmpresa	Cep da Empresa	Varchar	191	Unique / Not Null

complementoEmpresa	Complemento do endereço da empresa (Referência)	Varchar	191	
bairroEmpresa	Bairro da Empresa	Varchar	191	Not Null
latitude	Latitude do endereço da Empresa (para usar no mapa)	Varchar	191	Default
longitude	Longitude do endereço da Empresa (para usar no mapa)	Varchar	191	Default

Tabela 6 - Dicionário de Dados de Especialidade

Tabela	Especialidade			
Descrição	Armazena todos os tipos de especialistas na área da saúde			
Observações				
CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
IdEspecialidade	Código de identificação de cada especialidade	Int	11	PK / Identity
nomeEspecialidade	Nome do tipo de especialista da área da saúde	Varchar	191	Not Null

Tabela 7 - Dicionário de Dados da Tabela de Disponibilidade

Tabela	Disponibilidade Especialidade

Descrição	Armazena todos os profissionais disponíveis por especialidade de uma determinada empresa			
Observações	Possui uma chave estrangeira da tabela Especialidade e outra da tabela Empresa			
CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
idDisponibilidade	Código de identificação de cada disponibilidade	Int	11	PK / Identity
quantidade Especialidade	Quantidade de profissionais disponíveis em determinada empresa	Int	11	Not Null
idEmpresa	Identificador da Empresa	Int	11	FK
idEspecialidade	Identificador da Especialidade	Int	11	FK

Tabela 8 - Dicionário de Dados da Tabela de Comentário

Tabela	Comentário
Descrição	Armazena todos os comentários realizados pelos usuários
Observações	Possui uma chave estrangeira da tabela usuário

CAMPOS				
NOME	**DESCRIÇÃO**	**TIPO DE DADO**	**TAMANHO**	**RESTRIÇÕES**
idComentario	Código de identificação de cada Comentário	Int	11	PK / Identity
conteudoComentario	Mensagem do comentário	Varchar	191	Not Null
dataPublicacao	Data em que o comentário foi realizado	Varchar	191	Default
idEmpresa	Identificador da empresa em que o comentário foi realizado	Int	11	FK
idUsuario	Identificador do usuário que realizou o comentário	Int	11	FK

7 TECNOLOGIAS E FERRAMENTAS

Este tópico abordará algumas das principais tecnologias que foram utilizadas e necessárias para o desenvolvimento ou planejamento do projeto, sendo cada uma delas uma parte crucial para a criação das plataformas.

7.1 FIGMA

A plataforma colaborativa Figma foi feita com a intenção de fornecer um meio para que os usuários consigam desenvolver interfaces e protótipos, feito para ser utilizado no navegador web, mas que posteriormente recebeu ferramentas offline feitas para serem utilizadas no desktop para plataformas Windows, Linux e macOS.

O Figma teve seu lançamento em 2016, desde então, ganhou um grande destaque no mercado por conta de sua abordagem colaborativa e que ao mesmo tempo fornece um método de conduzir o trabalho de várias pessoas em tempo real em um único projeto, o que dá uma maior velocidade para a produção e desenvolvimento de um determinado projeto, fora que além disso, ele é uma plataforma de fácil acesso por estar disponível para a maioria dos navegadores.

Essa ferramenta foi utilizada no projeto para produzir os protótipos de alta fidelidade, tanto para a parte web quanto para a mobile da aplicação. O Figma foi escolhido devido à sua usabilidade em uma ampla gama de navegadores, eliminando a necessidade de instalar arquivos executáveis. Sua característica colaborativa foi fundamental para o desenvolvimento do projeto, permitindo a interação entre todos os membros da equipe em tempo real. Isso proporcionou um meio de feedback imediato, permitindo que cada membro demonstrasse suas ideias para determinadas telas, o que foi essencial para chegar a uma conclusão sobre o design final de todas as telas do sistema. O Figma desempenhou um papel essencial nesta fase inicial do desenvolvimento do projeto, na figura 11 é possível a visualização de seu logotipo.

Figura 11 - Figma

Fonte: Figma, 2024.

7.2 DRAW.IO

O Draw.io é um editor gráfico que permite a criação de desenhos, gráficos e uma variedade de outros projetos sem a necessidade de baixar arquivos ou qualquer outra ação desse tipo. Funciona na plataforma web de forma totalmente remota, semelhante ao Figma, sendo também reconhecido por sua leveza e facilidade de uso.

Esse editor é, sem dúvida, extremamente eficiente no que se propõe a ser, que é um editor leve e com uma curva de aprendizado simples. O blog TechTudo (S.D) faz a seguinte afirmação a respeito do Draw.io: "Com um layout simples e fácil, o serviço permite ao usuário criar seus próprios processos, inserindo uma grande variedade de formas, textos, setas indicativas, e alterar cores e espessuras de traços." Com essa afirmação como base, pode-se perceber o quão efetivo e essencial esse editor é em sua proposta.

A equipe optou por utilizar esse editor devido a todas as questões levantadas. Assim como o Figma, ele é uma plataforma colaborativa, o que facilita a interação dos membros da equipe no projeto realizado. Eles utilizaram esse editor para criar tanto o diagrama de classe quanto o diagrama de caso de uso, e graças a essa ferramenta, conseguiram elaborar esses diagramas com relativa facilidade, a figura 12 representa o logo tipo do DRAW IO.

Figura 12 - DRAW IO.

Fonte: Snapcraft, 2024.

7.3 GITHUB

De acordo com a Hostinger (2023), de maneira resumidade, o GitHub é um serviço que funciona baseado em nuvem que serve para hospedar um sistema de controle de versão que se chama Git, o que permite que desenvolvedores compartilhem projetos e consigam trabalhar de maneira colaborativo enquanto mantêm um registro com detalhes de seu progresso.

O GitHub atualmente é sem dúvidas uma das ferramentas mais importantes e utilizadas no mercado de trabalho entre os desenvolvedores, em sua definição, podemos observar o quanto ele facilita o trabalho em conjunto de equipes de programadores, que conseguem colaborar entre si graças ao suporte remoto e seguro que é fornecido por essa plataforma, além das questões de colaboração, ele fornece uma segurança maior para os desenvolvedores, pois armazena todas as versões do código em sua nuvem, o que facilita situações em que for necessário realizar ações como por exemplo backup no código desejado, proporcionando um ambiente seguro para o desenvolvimento de aplicações.

Essa plataforma desempenhou um papel crucial no desenvolvimento da aplicação. Cada membro da equipe foi designado para se concentrar em uma parte

específica do sistema, resultando em um aumento significativo na velocidade de desenvolvimento. Além disso, promoveu uma colaboração mais eficaz dentro da equipe, permitindo que cada integrante visualizasse todas as versões do código e expressasse sua opinião sobre possíveis melhorias que poderiam otimizar o código como um todo. Isso contribuiu para aprimorar todas as fases da aplicação, na figura 13 é possível a visualização de seu logotipo.

Figura 13 - Github

Fonte: Medium, 2019.

7.4 VISUAL STUDIO CODE

Conforme a documentação do próprio site do Visual Studio (2024), o Visual Studio Code é um editor que foi feito visando ser um editor de texto extremamente leve, mas que ao mesmo tempo possar ser poderoso e efetivo para suas aplicações, ele é um editor de textos disponível para Windows, Linux e macOS, possuindo um suporte integrado para as tecnologias JavaScript, TypeScript e NodeJS, mas que também conta com um ecossistema robusto de extensões para fornecer suporte a outros linguagens (como C++, C#, Java, Python, PHP, .NET).

Este editor é muito utilizado entre os desenvolvedores atualmente e vai continuar sendo por conta que, como citado anteriormente, ele é um editor que foi projeto para ser bem leve e ao mesmo tempo muito efetivo no que se propõe a fazer, o que fornece ao usuário que está utilizando um meio de conseguir realizar a construção de seu projeto ou aplicação de maneira simples, rápida e muito efetiva.

Sendo fundamental para a construção do projeto, o Visual Studio Code foi utilizado em todas as etapas da construção do sistema. Devido às suas qualidades previamente mencionadas e à sua ampla biblioteca de extensões, proporciona uma experiência extremamente agradável ao ser utilizado como editor de texto. A capacidade de instalar extensões que facilitam o uso de determinadas tecnologias agiliza consideravelmente o processo de construção do sistema, tornando-o muito mais agradável, abaixo na figura 14 é possível identificar o logo tipo desta ferramenta.

Figura 14 - Visual Studio Code

Fonte: TMS Software, 2024.

7.5 MYSQL

Segundo o site da Google Cloud (S.D), o MySQL é definido como um banco de dados relacional, de código aberto e que é um dos mais conhecidos do mundo, essa popularidade tão alta sendo resultado de seu uso generalizado em sites de comércio eletrônico, mídias sociais e aplicativos, sendo classificado pelo DB-Engines como o segundo banco de dados mais utilizado no mundo.

A utilização de um banco de dados relacional é uma ótima maneira de trabalhar com dados mais complexos e que exigem conectividade entre si, sendo que cada tabela neste tipo de banco possui um identificador único que pode ser utilizado para facilitar a identificação desta tabela e simplificar seu relacionamento com outras.

> “Um banco de dados relacional é um tipo de banco de dados que armazena e fornece acesso a pontos de dados relacionados entre si. Bancos

> de dados relacionais são 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 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)

Devido à sua popularidade, o banco de dados MySQL já era conhecido e utilizado por todos os membros da equipe, tornando-se a primeira escolha em mente para qual banco de dados utilizar. Conscientes de que trabalhariam com uma grande quantidade de dados e com relações entre tabelas, sabiam que um banco de dados relacional seria uma boa opção para lidar com essas situações. O fato de o MySQL ser amplamente utilizado e familiar para os integrantes da equipe fez dele o banco de dados escolhido para o projeto, a figura 15 representa o logo tipo desta tecnologia.

Figura 15 - MySQL

Fonte: Oracle, 2024.

7.6 JAVASCRIPT

De acordo com o blog Developer Mozilla (2022), o Javascript é uma linguagem leve e que é comumente conhecida como uma linguagem de script para as páginas Web, mas que também posteriormente passou a ser utilizada em outros contextos, ela é uma linguagem baseada em protótipos, multi-paradigma e dinâmica, suportando vários estilos de programação com a orientada a objetos, imperativa e declarativa.

Sem dúvida alguma, o JavaScript emerge como uma linguagem de extrema importância no cenário das aplicações web. De acordo com uma pesquisa de desenvolvedores conduzida pelo Stack Overflow em 2022, ela se destaca como a linguagem mais popular do mundo. Este dado por si só já evidencia o alcance significativo dessa tecnologia. Em conjunto com o HTML e o CSS, o JavaScript constitui a base fundamental de toda a experiência de navegação na web.

Pode-se afirmar com segurança que essa foi a tecnologia mais importante para o desenvolvimento e conclusão do projeto, uma vez que todas as tecnologias utilizadas no ambiente de programação foram baseadas no JavaScript. Esta linguagem de programação foi predominante em todas as etapas do projeto, abaixo na figura 16 é possível a visualização de seu logotipo.

Figura 16 - JavaScript

Fonte: World Vector Logo, S.D.

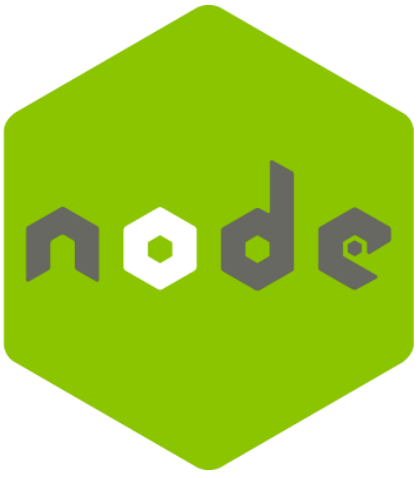
7.7 NODEJS

Na documentação do NodeJS (2024), ele se define como um software de código aberto, multiplataforma e que faz com que a execução de um código JavaScript que antes só era possível dentro de um contexto de um navegador web, ser realizada sem necessariamente estar rodando neste ambiente, fornecendo desta forma, diversos cenários no qual está tecnologia pode ser usada como forma de resolução.

Essa tecnologia é cada vez mais requisitada e utilizada no mercado de trabalho e entre os desenvolvedores, sendo definida pela Alura (2023) como uma tecnologia extremamente versátil e que possui qualidades como escalabilidade, ser multiplataforma, código aberto e por ser uma linguagem de programação multi-paradigma.

A utilização do Node foi fundamental para o desenvolvimento do projeto, pois é um pré-requisito para a utilização de outras tecnologias presentes nele. Além disso, possibilitou a execução do JavaScript em várias etapas do projeto, sendo está a linguagem mais conhecida e dominada pela equipe. Isso facilitou significativamente o desenvolvimento de todas as etapas do sistema, o logotipo desta tecnologia é visualizado na figura 17.

Figura 17 - NodeJS



Fonte: Medium, 2023.

7.8 REACT

Com base na documentação do próprio React (2024), ele é uma biblioteca que permite a criação de interfaces para o usuário a partir de várias peças individuais que são chamadas de componentes, esses componentes são pequenas partes do código que são individualizados e depois juntas formam uma página principal, preservando a qualidade do código do sistema.

O React é uma biblioteca que permite a construção de uma Single Page Aplicattion, que segundo o blog DevMedia, são aplicações que tem suas funcionalidades centradas em apenas uma página, ao invés das páginas tradicionais que recarregam ou redirecionam o usuário para uma outra página após realizar uma ação, esse tipo de aplicação atualiza apenas o conteúdo principal de forma assíncrona e mantém a estrutura da página estática, fornecendo uma experiência melhor para o usuário.

A utilização do React desempenhou um papel fundamental na construção da aplicação web. A capacidade de dividir o código em pequenos componentes para posterior reutilização foi um aspecto crucial para a qualidade e eficiência do código. Além disso, por ser uma biblioteca que opera no modelo de Single Page Application (SPA), foi possível desenvolver um sistema que proporciona uma experiência superior ao usuário, sendo uma aplicação ágil e eficiente, o logo tipo do React é representado na figura 18.

Figura 18 - React

Fonte: React, 2024.

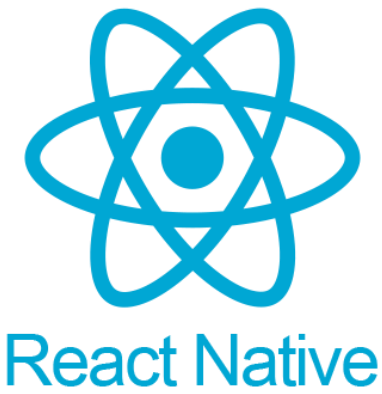
7.9 REACT NATIVE

Segundo a documentação do próprio React Native (2024), esta tecnologia combina as melhores partes do desenvolvimento nativo com React, porém desta vez para ser aplicada em um contexto de dispositivos móveis, sendo possível construir aplicações para as plataformas Android, IOS e muitos outras, sendo extremamente versátil e conseguindo reaproveitar os aspectos principais do React.

Ao utilizar esta tecnologia, é possível utilizar uma grande quantidade de componentes nativos do próprio Framework, que são componentes já moldados para atender os padrões que são necessários para conseguir desenvolver uma aplicação para um dispositivo móvel.

Necessário para o desenvolvimento da parte mobile da aplicação, esse framework simplificou significativamente a criação de uma aplicação acessível para a maior quantidade de usuários possível. Graças a ele, foi possível oferecer um sistema rápido e capaz de cumprir seu objetivo de ser fácil e prático de usar, fornecendo interfaces agradáveis e intuitivas para os pacientes. Além disso, é compatível com todos os sistemas operacionais, garantindo uma experiência consistente em diferentes dispositivos, é possível visualizar seu logotipo a partir da figura 19.

Figura 19 - React Native



Fonte: React Native, 2024.

7.10 POSTMAN

O Postman em sua própria documentação (2024), se define como uma plataforma para a construção e utilização de APIs, simplificando as etapas do ciclo de vida de uma API e fornecendo uma maneira na qual você possa construi-las e testá-las de maneira eficiente.

Essa tecnologia simplifica a criação, documentação e teste de uma API. Além de ser gratuita e de fácil utilização, ela não requer necessariamente instalação, pois oferece uma versão web que funciona de forma similar à versão desktop. Ambas as versões são igualmente válidas e podem ser utilizadas em diferentes contextos.

O Postman foi empregado para testar e documentar todas as rotas da API. Essa ferramenta desempenhou um papel crucial ao facilitar os testes de cada rota, garantindo que os objetivos fossem alcançados antes da integração com o front-end da aplicação. Essa abordagem contribuiu significativamente para a eficiência do projeto como um todo, agilizando os processos e assegurando sua qualidade, a figura 20 representa seu logotipo.

Figura 20 - Postman

Fonte: Medium, 2019.

7.11 TYPESCRIPT

Ao ler a documentação do TypeScript (2024), podemos perceber que ele é um tipo de extensão do JavaScript, adicionando uma sintaxe própria, mas que no final

das contas será convertida em código JavaScript, sendo uma ferramenta feita para facilitar exclusivamente a parte de desenvolvimento de sua aplicação, fornecendo uma segurança maior aos desenvolvedores para conseguirem ter certeza que estão seguindo os passos certos para a finalização de determinado projeto.

Esta ferramenta tem como principal objetivo transformar o JavaScript que é uma linguagem não tipada e interpretada, em uma linguagem tipada, o que de certa forma dificulta sua utilização, mas que ao mesmo tempo, fornece diversos benefícios para a parte de desenvolvimento de sua aplicação, o que torna este desenvolvimento mais seguro e efetivo, diminuindo a chance de algum erro passar batido durante a produção da aplicação.

Essa tecnologia foi utilizada durante o desenvolvimento do back-end da aplicação. Devido à natureza sensível dos dados tratados nessa parte do projeto, que são posteriormente enviados para o banco de dados, a equipe reconheceu a necessidade de uma tecnologia que fornecesse maior segurança na verificação de todas essas variáveis, a fim de evitar possíveis erros durante o envio para o banco de dados, abaixo é possível visualizar seu logotipo a partir da figura 21.

Figura 21 - Typescript

Fonte: Typescript, 2024.

7.12 PRISMA STUDIO

Conforme a documentação do Prisma (2024), ele é uma *ORM* [4] baseada em NodeJS e TypeScript que fornece ao desenvolvedor uma plataforma de última geração e experiência ao trabalhar com a interação com o banco de dados, sendo uma tecnologia extremamente intuitiva, segura e que realiza suas migrações de maneira totalmente automatizada.

O uso de ORMs tem sido cada vez mais comum no desenvolvimento de pedaços da aplicação que irão interagir com o banco de dados, o blog Cubos Academy (2023) cita como vantagens de utilizar uma ORM benefícios como: simplicidade, independência de banco de dados, prevênção de *SQL injection* [5] e facilidade de manutenção.

O Prisma Studio teve um papel fundamental no desenvolvimento da conexão da aplicação com o banco de dados. Conforme mencionado anteriormente, ele simplifica significativamente todos os processos relacionados à criação do banco e à interação do código back-end com o banco de dados. Além disso, por meio de suas migrações, proporciona uma maneira rápida de alterar informações no banco quando necessário, a logotipo desta tecnologia é visualizada a partir da figura 22.

Figura 22 - Prisma

[4] Object-Relational Mapping (ORM), em português, mapeamento objeto-relacional, é uma técnica para aproximar o paradigma de desenvolvimento de aplicações orientadas a objetos ao paradigma do banco de dados relacional. (UFSM, 2022).

[5] O SQL Injection é uma técnica de ataque baseada na manipulação do código SQL, que é a linguagem utilizada para troca de informações entre aplicativos e bancos de dados relacionais. [4] https://www.devmedia.com.br/sql-injection/6102

Fonte: Prisma, 2024.

7.13 Amazon S3

O serviço do Amazon Simple Storage Service (Amazon S3) se trata de uma maneira de conseguir armazenar variados tipos de objetos, oferecendo diversos benefícios como escalabilidade, disponibilidades dos dados, segurança e uma incrível performance (AMAZON, 2023).

A ferramenta foi utilizada para conseguir armazenar os uploads que são feitos pelo usuário na aplicação, arquivos como fotos e documentos, para conseguir armazenar essas informações, se fez necessário a criação de uma Bucket[6], para que consiga armazenar os objetos no Amazon S3 e que posteriormente esses objetos consigam ser resgatados para serem utilizados na aplicação.

Esse serviço foi fundamental para conseguir permitir com que o usuário da aplicação consiga interagir com as funcionalidades da entidade de Exame, permitindo com que o paciente consiga armazenar seus exames e visualizá-los na aplicação, o logo tipo desta ferramenta é representado na figura 23.

Figura 23 - Amazon S3

[6] Um bucket é um contêiner para objetos armazenados no Amazon S3. Você pode armazenar qualquer número de objetos em um bucket e pode ter até 100 buckets na sua conta. [5] https://docs.aws.amazon.com/pt_br/AmazonS3/latest/userguide/UsingBucket.html

Fonte: Nobug, 2022.

7.14 Swagger

O Swagger se trata de uma maneira de conseguir documentar as especificações de uma API, especificamente, de uma API Rest, o que acaba se tornando crucial para conseguir orientar a equipe responsável durante todas as etapas de desenvolvimento do projeto (SWAGGER, 2024).

Essa ferramenta é de extrema importância durante o desenvolvimento de uma aplicação por permitir que testes sejam feitos durante todo o desenvolvimento do back end, e graças a sua documentação, facilita o trabalho do front end que terá que interagir com os dados retornados por uma determinada API.

Graças a possibilidade de visualizar todas as rotas da API e seus respectivos *end-points*[7], o Swagger auxiliou o desenvolvimento da aplicação permitindo que os testes necessários fossem feitos durante este desenvolvimento e a documentação auxilia todo o processo de consumo dos dados que são retornados ao requisitar todas as rotas feitas na API Rest, abaixo na figura 24 é possível a visualização de seu logotipo.

Figura 24 - Swagger

Fonte: Swagger, 2024.

[7] Um endpoint da API é o local em que essas solicitações (conhecidas como chamadas de API) são atendidas. [6] https://www.cloudflare.com/pt-br/learning/security/api/what-is-api-endpoint/

8 DESENVOLVIMENTO

Neste tópico será abordado todas as etapas do desenvolvimento da aplicação Helthcare, separando e explicando cada parte com ênfase e detalhando como realizado o desenvolvimento de cada parte e sua importância para a criação e finalização da aplicação com tudo todo, exemplificando como cada parte teve seu papel fundamental para o projeto.

Para o front-end tanto da versão Web, quanto da versão mobile do sistema, as entidades serão responsáveis por guiar o desenvolvimento das interfaces e componentes do sistema, já que eles serão criados tendo como parâmetros as regras de negócios, para que assim os hospitais e pacientes consigam interagir com o sistema da maneira que foi planejado.

Como no back-end são utilizados dados que ou chegam diretamente do banco de dados, ou são enviados para lá, as entidades aparecem como variáveis do sistema, que serão representadas conforme a necessidade de interação que esta parte terá com o banco de dados.

Essas entidades serão utilizadas em todos os contextos da aplicação, seja no front-end, back-end, banco de dados, requisitos e diagramas UML.

8.1 BANCO DE DADOS

O sistema trabalha com informações importantes que precisam ser armazenadas para posteriormente serem reaproveitadas em outro contexto, se fez a necessidade de utilizar um banco de dados para cuidar desta função, a Oracle (2024) define um banco de dados da seguinte maneira:

> “Um banco de dados é uma coleção organizada de informações - ou dados - estruturadas, normalmente armazenadas eletronicamente em um sistema de computador. Um banco de dados é geralmente controlado por um sistema de gerenciamento de banco de dados (DBMS). Juntos, os dados e o DBMS,

> juntamente com os aplicativos associados a eles, são chamados de sistema de banco de dados, geralmente abreviados para apenas banco de dados.”

A partir dessa definição sobre banco de dados, fica evidente a necessidade de empregar esse conceito para manipular os dados provenientes de nossas aplicações. Ao percebermos que algumas dessas entidades precisariam estabelecer algum tipo de relação entre si, foi utilizado um banco de dados relacional. Conforme definido pela IBM, essa escolha ressalta a importância das relações entre os dados, os quais são organizados em linhas e colunas formando tabelas. Cada tabela possui um identificador único que possibilita o reconhecimento e o estabelecimento de relacionamentos entre elas.

No mesmo artigo referenciado anteriormente, a IBM cita o termo SQL (Linguagem de Consulta Estruturada) que foi uma linguagem de programação criada com a intenção de conseguir interagir com sistemas de gerenciamento de banco de dados relacional, atualmente sua utilização em banco de dados relacional se tornou tão comum, que por muitas vezes estes tipos de bancos são chamados de “bancos de dados SQL”, através disso observa-se o quanto a utilização desta tecnologia neste tipo de contexto se tornou muito padronizada conforme os anos.

Ao escolher a linguagem de programação MySQL, escolha essa motivada pela familiaridade dos integrantes da equipe com tal tecnologia, optou-se por utilizar uma ORM, que é uma técnica que permite alinhar o código back-end com a estrutura do banco de dados, o que acaba facilitando no momento que fosse necessário executar qualquer tipo de consulta no banco de dados, o que acabou agilizando o processo de desenvolvimento da aplicação, o blog DevMedia (2011) define este conceito da seguinte maneira:

> “ORM (Object Relational Mapper) é uma técnica de mapeamento objeto relacional que permite fazer uma relação dos objetos com os dados que os mesmos representam. Ultimamente tem sido muito utilizada e vem crescendo bastante nos últimos anos.Este crescimento tem se dado principalmente pelo fato de muitos desenvolvedores não se sentirem a vontade em escrever código SQL e pela produtividade que esta técnica nos proporciona.”

A ORM utilizada no projeto foi o Prisma Studio, tecnologia que será abordada no tópico dedicado ao desenvolvimento do back-end da aplicação. No entanto, essa tecnologia foi empregada para criar as tabelas na base de dados. A seguir, é

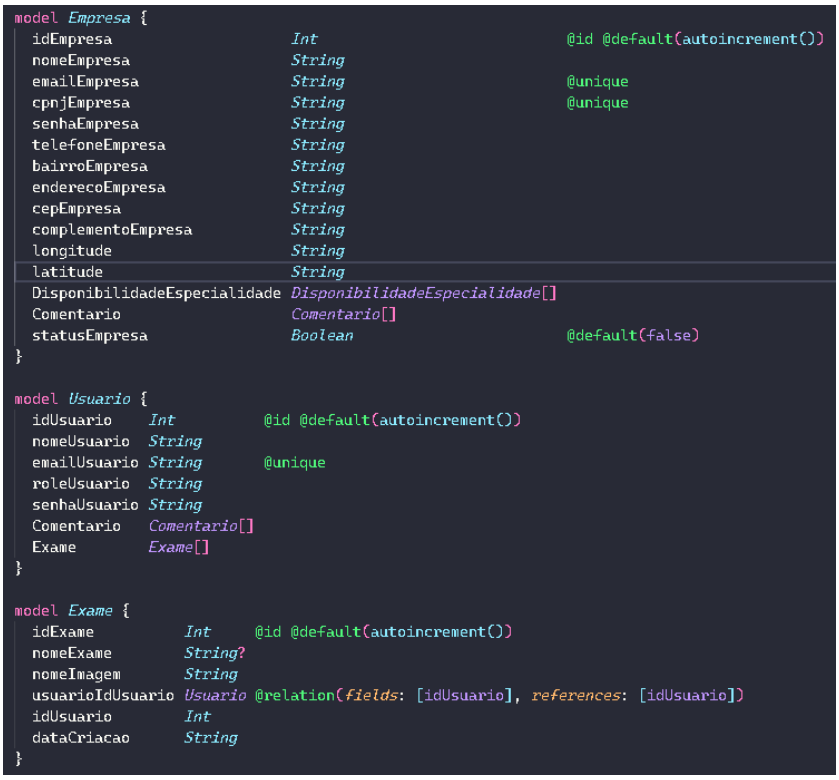
apresentada a representação dos modelos das tabelas do banco de dados a partir das figuras 25, 26 e 27:

Figura 25 - Exibição dos models feitos.

```
model Especialidade {
  idEspecialidade             Int                          @id @default(autoincrement())
  nomeEspecialidade           String
  DisponibilidadeEspecialidade DisponibilidadeEspecialidade[]
}

model DisponibilidadeEspecialidade {
  idDisponibilidade       Int           @id @default(autoincrement())
  quantidadeEspecialidade Int
  Especialidade           Especialidade @relation(fields: [idEspecialidade], references: [idEspecialidade])
  Empresa                 Empresa       @relation(fields: [idEmpresa], references: [idEmpresa])
  idEspecialidade         Int
  idEmpresa               Int
}


model Comentario {
  idComentario       Int     @id @default(autoincrement())
  conteudoComentario String
  dataPublicacao     String
  Usuario            Usuario @relation(fields: [idUsuario], references: [idUsuario])
  idUsuario          Int
  Empresa            Empresa @relation(fields: [idEmpresa], references: [idEmpresa])
  idEmpresa          Int
}
```

Fonte: Elaborado pelos autores, 2024

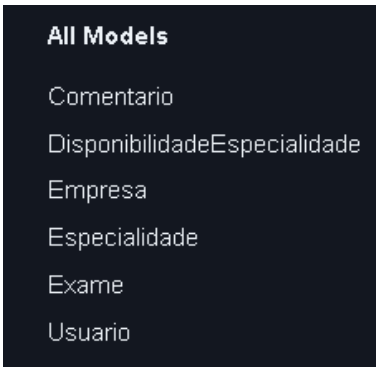
Figura 26 - Exibição dos models feitos.



```
model Empresa {
  idEmpresa                 Int                          @id @default(autoincrement())
  nomeEmpresa               String
  emailEmpresa              String                       @unique
  cpnjEmpresa               String                       @unique
  senhaEmpresa              String
  telefoneEmpresa           String
  bairroEmpresa             String
  enderecoEmpresa           String
  cepEmpresa                String
  complementoEmpresa        String
  longitude                 String
  latitude                  String
  DisponibilidadeEspecialidade DisponibilidadeEspecialidade[]
  Comentario                Comentario[]
  statusEmpresa             Boolean                      @default(false)
}

model Usuario {
  idUsuario    Int          @id @default(autoincrement())
  nomeUsuario  String
  emailUsuario String       @unique
  roleUsuario  String
  senhaUsuario String
  Comentario   Comentario[]
  Exame        Exame[]
}

model Exame {
  idExame          Int     @id @default(autoincrement())
  nomeExame        String?
  nomeImagem       String
  usuarioIdUsuario Usuario @relation(fields: [idUsuario], references: [idUsuario])
  idUsuario        Int
  dataCriacao      String
}
```

Fonte: Elaborado pelos autores, 2024

Figura 27 - Todos os Models criados.



Fonte: Elaborado pelos autores, 2024

8.2 BACK-END

O back-end será o responsável por realizar a conexão entre o ambiente front end com o banco de dados da aplicação, sendo uma parte crucial para permitir com que os usuários, por meio de interfaces, manipulem e interajam com dados existentes no banco de dados. O blog Kenzie (2021) define o back-end como sendo o responsável por questões internas do software, como banco de dados, segurança, envio e recebimento de informações, armazenamento e entre outras questões.

O back-end consiste em uma API que será a responsável por receber e enviar informações cruciais do banco de dados para que consigam ser utilizadas no front-end de da aplicação, utilizamos o conceito de API RESTful [8]para a criação dela e a

[8] Indica que o servidor transfere informações em formato-padrão. O recurso formatado é chamado de representação em REST. Esse formato pode ser diferente da representação interna do recurso na aplicação do servidor. [7] https://aws.amazon.com/pt/what-is/restful-api/#:~:text=Indica%20que%20o%20servidor%20transfere,um%20formato%20de%20representa%C3%A7%C3%A3o%20HTML.

tecnologia TypeScript em conjunto com os frameworks Express e Prisma para sua criação.

De acordo com a definição feita pela Hostinger (2023), uma API é utilizada como um mediador entre os usuários e o sistema, facilitando o acesso e o desenvolvimento de aplicações Web, de acordo com a mesma fonte, o que faz uma API ser considerada REST é o conjunto de regras estabelecidas na qual ela deve cumprir, estas regras foram criadas visando a criação de aplicações com interfaces bem definidas, rotinas padronizadas e que facilitem a comunicação entre os sistemas e os usuários, e por fim, uma API RESTful é feita através da manipulação de recursos e suas representações, sendo que essas representações são utilizadas para permitir a interação entre o usuário com o servidor, utilizando uma interface padronizada e em conjunto com um protocolo de comunicação, no qual geralmente é utilizado o protocolo *HTTP*[9], que foi o mesmo utilizado durante o desenvolvimento.

Ao utilizar o conceito RESTful durante o desenvolvimento do back-end, é necessário definir todas as rotas que serão responsáveis por retornar informações que serão utilizadas no front-end, no caso, separadas as rotas com base nas definições das entidades do sistema, sendo cada uma dessas rotas as responsáveis por realizar a comunicação entre o front-end com o banco de dados, servindo em situações nas quais o front-end envia algum parâmetro para ser manipulado e posteriormente enviado ao banco de dados, mas também sendo um meio de receber as informações do banco de dados e exibir para o usuário da aplicação, portanto, algumas rotas funcionam de maneira interativa e outros foram criadas somente como meio de leitura dos dados recebidos do banco.

[9] HTTP é a sigla para Hypertext Transfer Protocol, ou Protocolo de Transferência de Hipertexto. Esse é o principal protocolo responsável pela transferência de dados na Internet, criando as bases necessárias para a conexão entre um cliente e um servidor. [8] https://www.hostinger.com.br/tutoriais/http#:~:text=HTTP%20%C3%A9%20a%20sigla%20para,um%20cliente%20e%20um%20servidor.

8.2.1 ROTAS DE EMPRESA

As rotas criadas com o objetivo de manipular dados sobre as empresas cadastradas no sistema foram concebidas de forma a facilitar seu uso no front-end. Cada uma delas é projetada para enviar e devolver apenas as informações necessárias para o funcionamento das regras de negócio, sendo divididas de acordo com suas funcionalidades específicas. Entre todas, algumas rotas se destacam.

- **Adicionar Empresa**: Uma das mais importantes rotas da aplicação, sendo ela a responsável por permitir que as empresas se cadastrem na plataforma, possuindo todas as validações necessárias;
- **Autenticar Empresa**: Essa rota é necessária para que a empresa consiga se autenticar na aplicação, se todas as informações enviadas estiverem corretas, devolve um token[10] que será responsável para verificar posteriormente a autenticação da empresa;
- **Atualizar Empresa**: Rota essa criada para que a empresa consiga atualizar informações que foram adicionadas após o cadastro dela no sistema, sendo essencial para manter os dados atualizados;
- **Deletar Empresa**: Responsável por permitir com que a empresa em caso de necessidade, delete todas as informações dela no sistema;
- **Listar Empresas Abertas**: Esta rota é essencial para o uso do paciente na parte mobile da aplicação, sendo ela a responsável por devolver todas as empresas que definiram seu status como aberta na parte Web da aplicação, de forma com que facilite o filtro do usuário buscar um hospital que atenda suas necessidades no momento;
- **Listar Comentários**: A rota que irá devolver todos os comentários que foram feitos por usuário a respeito de uma empresa em específico, sendo outra rota essencial para a interação do usuário com a parte Mobile.

---

[10] Token, em inglês, significa ficha ou símbolo. Na área da tecnologia, o nome se refere a um dispositivo eletrônico/sistema gerador de senhas bastante utilizado por bancos. [9] https://www.infomoney.com.br/guias/tokens/

As rotas da empresa são visualizadas nas figuras 28 e 29, sendo representadas na tecnologia Swagger e diretamente no código respectivamente.

Figura 28 - Todas as rotas de Empresa representadas no Swagger.

Fonte: Elaborado pelos autores, 2024

Figura 29 - Rotas de Empresa representadas no código.

```
router.post('/adicionarEmpresa', empresaController.adicionarEmpresa);
router.get('/listarEmpresas', empresaController.listarEmpresas);
router.get('/selecionarEmpresa/:id',AuthMiddleware, empresaController.selecionarEmpresa);
router.post('/autenticarEmpresa', empresaController.autenticarEmpresa);
router.put('/atualizarEmpresa/:id',AuthMiddleware, empresaController.atualizarEmpresa);
router.delete('/deletarEmpresa/:id', AuthMiddleware, empresaController.deletarEmpresa);
router.put('/atualizarStatus/:id', empresaController.atualizarStatus);
router.post('/pesquisarEmpresa', AuthMiddleware, empresaController.pesquisarEmpresa);
router.get('/listarEmpresasAbertas', empresaController.listarEmpresasAbertas);
router.get('/listarProfissionais/:id', empresaController.listarProfissionaisDisponiveis);
router.get('/listarComentarios/:id', empresaController.listarComentarios);
router.put('/atualizarSenha/:id', empresaController.atualizarSenha);
```

Fonte: Elaborados pelos autores, 2024

8.2.2 ROTAS DE PACIENTE

Estas rotas foram criadas para que fosse possível a interação dos pacientes com nossa aplicação, sendo essas rotas feitas para serem utilizadas pela parte do front-end Mobile do sistema, destaca-se as seguintes rotas:

- **Adicionar Paciente**: Feita de maneira semelhante a rota de adicionar empresa, esta rota foi feita para que o paciente consiga se registrar no sistema, possuindo todas as validações necessárias;
- **Autenticar Paciente**: Rota que será responsável para autenticar o paciente no caso em que todas as informações enviadas estejam corretas, devolvendo o token em caso de sucesso para conseguir restringir o acesso do paciente em rotas que exigem a necessidade da autenticação;
- **Atualizar Paciente**: Responsável por entregar um método em que o paciente consiga alterar suas informações que foram enviadas durante o seu cadastro, sendo essencial para a experiência do usuário;
- **Deletar Paciente**: Caso o paciente não deseje mais utilizar a conta em que ele cadastrou, ele tem a possibilidade de deletar sua conta e todas as informações da plataforma;
- **Realizar Comentário**: Uma das rotas mais importantes da aplicação, sendo ela feita para com que o paciente consiga realizar seu comentário a respeito de alguma empresa cadastrada no sistema, sendo que este comentário posteriormente será visualizado por outros pacientes, sendo essencial para a interação entre os pacientes.
- **AdicionarExame**: Essa rota possibilita com que o usuário consiga armazenar o exame desejado dentro de seu aplicativo, para que depois possa ser visualizado através das informações obtidas de outra rota.
- **ListarExames**: A rota para listar os exames devolve todas as informações necessárias para que seja possível encontrar o caminho das imagens armazenadas pelo paciente, que estão todas disponíveis na nuvem do AWS S3.

- **DeletarExame**: Essa rota fornece ao paciente a possibilidade de conseguir deletar um determinado exame por meio de seu identificador.

Abaixo é possível visualizar as rotas dos pacientes representadas na ferramenta Swagger na figura 30 e diretamente no código na figura 31, destacando todos os parâmetros necessários para sua utilização.

Figura 30 - Todas as rotas de Paciente representadas no Swagger.

Fonte: Elaborado pelos autores, 2024

Figura 31 - Rotas de Paciente representadas no código.

```
router.get('/listarUsuarios', usuarioController.listarUsuarios);
router.post('/adicionarUsuario', usuarioController.adicionarUsuario);
router.post('/autenticarUsuario', usuarioController.autenticarUsuario);
router.put('/atualizarUsuario/:id', usuarioController.atualizarUsuario);
router.delete('/deletarUsuario/:id', usuarioController.deletarUsuario);
router.get('/selecionarUsuario/:id', usuarioController.selecionarUsuario);
router.post('/autenticarAdmin', usuarioController.autenticarAdmin);
router.post('/realizarComentario', usuarioController.realizarComentario);
router.post('/adicionarExame', upload.single('image'), usuarioController.adicionarExame);
router.get('/listarExames/:id', usuarioController.listarExames);
router.delete('/deletarExame/:id', usuarioController.deletarExame);
```

Fonte: Elaborado pelos autores, 2024

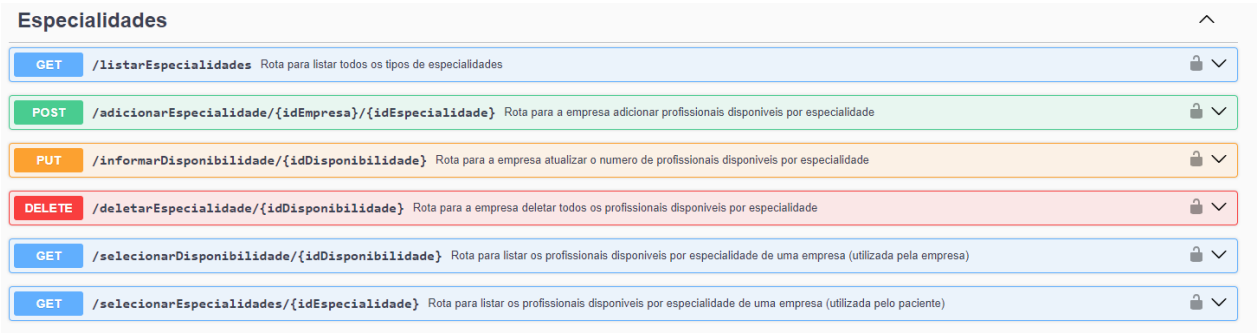
8.2.3 ROTAS DE ESPECIALIDADE

Por fim, temos as rotas que irão permitir a manipulação de dados que dizem a respeito das especialidades do sistema, sendo rotas essenciais tanto para os pacientes, quanto para as empresas, podemos destacar as seguintes rotas:

- **Listar Especialidades**: Essa rota será utilizada pela empresa na parte do front-end Web da aplicação, visto que todas as especialidades já são inseridas no banco de dados, a empresa poderá visualizar todas essas especialidades;
- **Listar Especialidades por Empresa**: Rota que será utilizada pelos pacientes no front-end Mobile da aplicação, nela será possível verificar a quantidade de profissionais disponíveis por especialidade em uma determinada empresa, sendo uma rota fundamental para o funcionamento das regras de negócio;
- **Adicionar Especialidade**: Sendo utilizada pelas empresas cadastradas, esta rota permite a possibilidade de a empresa informar a especialidade de um médico que está disponível em seu estabelecimento e a quantidade deles no momento, sendo também uma das rotas mais importantes para a implementação de regras de negócio no sistema;
- **Atualizar Disponibilidade Especialidade**: Essa rota permite com que a empresa consiga alterar a quantidade de profissionais disponível de uma especialidade em específico;
- **Deletar Disponibilidade**: A rota que permite a empresa de deletar uma especialidade de sua lista de profissionais disponíveis por especialidade.

Logo abaixo é possível visualizar as rotas de especialidades representadas no Swagger na figura 32 e diretamente no código na figura 33.

Figura 32 - Todas as rotas de Especialidade representadas no Swagger



Fonte: Elaborado pelos autores, 2024

Figura 33 - Rotas de especialidade representadas no código.

```
router.get('/listarEspecialidades', especialidadeController.listarEspecialidades);
router.get('/listarEspecialidades/:idEmpresa', especialidadeController.listarEspecialidadesEmpresas);
router.post('/adicionarEspecialidade/:idEmpresa/:idEspecialidade', especialidadeController.adicionarEspecialidade);
router.put('/informarEspecialidade/:idDisponibilidade', especialidadeController.informarDisponibilidade);
router.delete('/deletarEspecialidade/:idDisponibilidade', especialidadeController.deletarEspecialidade);
router.get('/selecionarDisponibilidade/:idDisponibilidade', especialidadeController.selecionarDisponibilidade);
```

Fonte: Elaborado pelos autores, 2024

8.2.4 CONTROLLERS

A aplicação foi dívida em diversos controllers, esses padrões são responsáveis por levar os métodos que serão responsáveis de interagir com o banco de dados (model) da aplicação e posteriormente, interagir com o front-end da aplicação, os controllers podem ser definidos desta seguinte maneira:

> “Como dito em sua definição, a camada de Controllers faz o "primeiro contato" com as requisições, enviando a camada de Services apenas as informações relevantes para completar a requisição. Além disso, essa é a camada que irá enviar a resposta ao cliente, seja ela positiva ou negativa.” (DEV.TO, 2022)

A partir desta definição, fica evidente a importância desses métodos, já que funcionarão como intermediários entre a camada de dados e o cliente da aplicação,

devido ao uso do Prisma no desenvolvimento do back-end, a aplicação possui êxito ao executar *Query's* [11] de uma maneira mais simplificada, facilitando o desenvolvimento desta etapa do back-end, os controllers foram divididos dessa forma:

- **Controller da Empresa**: Sendo este o controller responsável por fornecer todos os métodos que serão responsáveis por enviar as respostas necessárias principalmente para o front-end Web da aplicação, ele interage com todas as tabelas que dizem respeito a os dados das empresas, na figura 34 é possível visualizar seus principais métodos.

Figura 34 - Controller das Empresas.

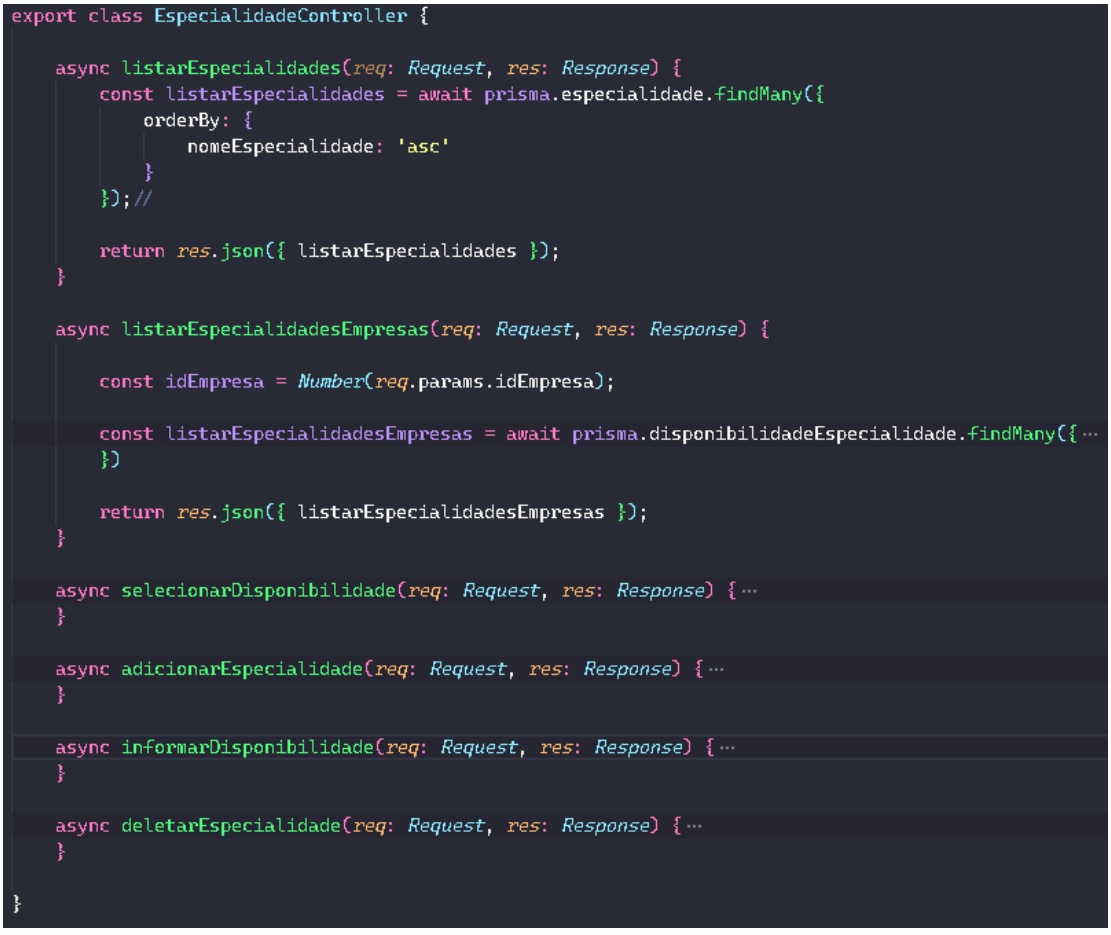
```
export class EmpresaController {

    async adicionarEmpresa(req: Request, res: Response) {···
    }

    async autenticarEmpresa(req: Request, res: Response) {···
    }

    async listarEmpresas(req: Request, res: Response) {
        const listarEmpresas = await prisma.empresa.findMany();

        return res.json({ listarEmpresas });
    }

    async selecionarEmpresa(req: Request, res: Response) {···
    }

    async listarProfissionaisDisponiveis(req: Request, res: Response) {···
    }

    async atualizarStatus(req: Request, res: Response) {
        const id = Number(req.params.id);

        const status = Boolean(req.body.status);

        const atualizarStatus = await prisma.empresa.update({
            data: {
                statusEmpresa: status
            },
            where: {
                idEmpresa: id
            }
        });

        return res.json({atualizarStatus});

    }
```

[11] Uma query é um pedido de uma informação ou de um dado. Esse pedido também pode ser entendido como uma consulta, uma solicitação ou, ainda, uma requisição. [10] https://www.hostinger.com.br/tutoriais/o-que-e-query

Fonte: Elaborado pelos autores, 2024

- **Controller das Especialidades**: Sendo o controller que possuí menos métodos, mas ainda assim, muito importante para a aplicação tendo em mente que ele interage com todos os dados que dizem respeito as especialidades, sendo essencial para o uso das entidades paciente e empresa, as principais funcionalidades deste controller são representadas na figura 35.

Figura 35 - Controller das Especialidades.



Fonte: Elaborados pelos autores, 2024

- **Controller dos Pacientes/Usuários**: Por último, o controller mais abrangente, pois é responsável por interagir com todos os dados dos usuários. optou-se por essa nomenclatura devido ao fato de que esses controllers não se limitam a interagir apenas com os pacientes, mas também com os administradores do

sistema. Portanto, foi utilizado o termo "usuário", que se adequa melhor a este contexto, os principais métodos deste controller são visualizados na figura 36.

Figura 36 - Controller dos usuários.

```
export class UsuarioController {

    async listarUsuarios(req: Request, res: Response) {
        const listarUsuarios = await prisma.usuario.findMany({
            where: {
                NOT: {
                    roleUsuario: 'ADMIN'
                }
            }
        });

        return res.json({ listarUsuarios });
    }

    async selecionarUsuario(req: Request, res: Response) {
        const id = Number(req.params.id);

        const selecionarUsuario = await prisma.usuario.findFirst({
            where: {
                idUsuario: id
            }
        });

        return res.json({ selecionarUsuario });
    }

    async adicionarUsuario(req: Request, res: Response) {···
    }

    async autenticarUsuario(req: Request, res: Response) {···
    }

    async autenticarAdmin(req: Request, res: Response) {···
    }

    async atualizarUsuario(req: Request, res: Response) {···
```

Fonte: Elaborado pelos autores, 2024

8.3 FRONT-END WEB

No desenvolvimento da aplicação web, foi empregado o framework React, o qual, conforme mencionado anteriormente, desempenha um papel essencial no contexto de Single Page Application (SPA), oferecendo uma experiência aprimorada

ao usuário. Além disso, permite a separação do código do projeto em diversos componentes, resultando em otimização no desenvolvimento do sistema.

Para a criação do projeto utilizamos a tecnologia Vite, que foi responsável por gerar a base das pastas e configurações do próprio React, essa tecnologia possuí um papel fundamental no que diz respeito ao desempenho do projeto, já que é uma tecnologia que foi feita para otimizar os fluxos de trabalho de desenvolvimento e desempenho de uma aplicação, além de deixar a criação do projeto React muito mais leve.

Observando a necessidade da aplicação de conter várias páginas, optou-se por empregar a tecnologia de React Router Dom. Conforme mencionado pelo blog DevMedia (2021), essa tecnologia desempenha um papel fundamental na criação de rotas que representam cada tela durante o desenvolvimento com o React, na figura 37 é possível visualizar todas as rotas da aplicação Web.

Figura 37 - Todas as rotas da aplicação Web.

```
export const AppRoutes = () => {
    return(
        <BrowserRouter>
            <Routes>
                <Route element={<HomePage/>} path="/"/>
                <Route element={<Entrar/>} path="/entrar"/>
                <Route element={<AlterarDados/>} path="/alterarDados"/>
                <Route element={<RegistrarDados/>} path="registrarDados"/>
                <Route element={<RecuperarSenha/>} path="recuperarSenha"/>
                <Route element={<EspecialidadeDisponivel/>} path="especialidadeDisponivel"/>
                <Route element={<AlterarEspecialidade/>} path="alterarEspecialidade/:id"/>
                <Route element={<EntrarAdmin/>} path="entrarAdmin"/>
                <Route element={<AdmEmpresa/>} path="admEmpresa"/>
                <Route element={<AdmPaciente/>} path="admPaciente"/>
                <Route element={<AlterarSenha/>} path="alterarSenha/:id" />

                <Route path="*" element={<NotFoundPage/>} />
            </Routes>
        </BrowserRouter>
    )
}
```

Fonte: Elaborado pelos autores, 2024.

8.3.1 PÁGINAS

As páginas do sistema são os componentes principais que são utilizados por cada rota definida anteriormente, carregando em seu código toda a lógica e a estrutura da página em si, separamos cada uma por pastas em que cada uma possui dentro o seu componente principal e o CSS que é responsável por estilizar a página, separamos as páginas como demonstrado na figura 38:

Figura 38 - Todas as páginas da aplicação Web.

Fonte: Elaborado pelos autores, 2024.

Além de cada uma dessas páginas possuir sua estrutura HTML própria e todas as lógicas envolvidas utilizando o Javascript, elas também carregam outros componentes que são pequenas partes do código que podem ser reaproveitadas e feitas em outro código o que simplifica o desenvolvimento da aplicação e torna a torna mais eficiente, a figura 39 representa um exemplo desta estrutura que foi utilizada em nossa aplicação.

Figura 39 - Estrutura de página da aplicação Web.

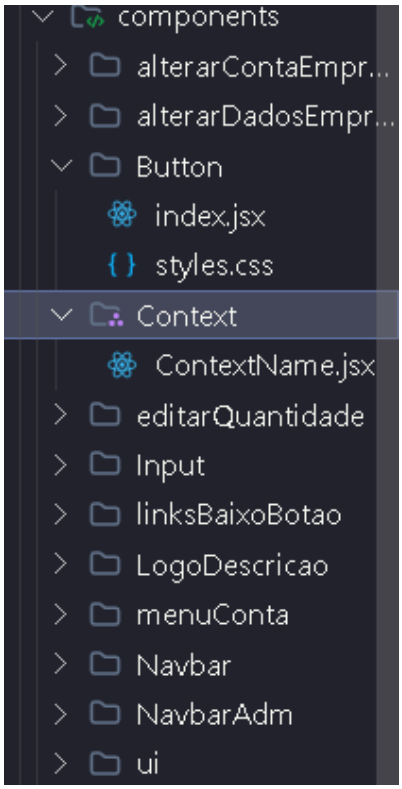
```
<body className="body-disponivel">
    <Navbar />
    <div className="container-total-especialidade">
        <div className="responsivo-disponivel">
            <div className="imagem-empresa-escolhe">
                <button type="submit" className="botao-imagem" >
                    <img className="botao-alterar-imagem" src={ImagemBotao} />
                </button>
            </div>
            <div className="adicionar-especialidade">
                <form onSubmit={editarEspecialidade}>
                    <h1 className="titulo-editar-especialidade">Editar Quantidade</h1>
                    <div className="centralizando-inputs">
                        <div className="inputs-centralizados">
                            <label className="label-especialidade" htmlFor="escolha-especialidade">Especialidade</label>
                            <select className='escolha-especialidade' value={especialidade}>
                                <option>{especialidade}</option>
                            </select>
                        </div>
                        <div className="inputs-centralizados">
                            <label htmlFor="input-quantidade">Quantidade</label>
                            <input defaultValue={quantidade} className="input-quantidade" type="number" placeholder="Qtd"
                            onChange={(e) => setQuantidade(Number(e.target.value))}
                            />
                        </div>
                        <div className="inputs-centralizados">
                            <button type="submit" className="botao-editarEspecialidade">Editar</button>
                        </div>
```

Fonte: Elaborado pelos autores, 2024.

8.3.2 COMPONENTES

Os componentes no React consistem em partes pequenas do código que são segmentadas para serem reutilizadas posteriormente nas páginas do sistema. Cada componente possui sua própria customização e oferece métodos diversos para permitir que o usuário interaja de maneiras específicas com a aplicação, na figura 40 destacamos estes componentes.

Figura 40 - Componentes da aplicação Web



Fonte: Elaborado pelos autores, 2024.

8.3.3 CONSUMO DA API

Para conseguir ter acesso aos dados que são enviados pela API do back-end na parte do front-end da aplicação, se fez necessário a utilização de algum método que consiga resgatar esses dados e armazená-los em variáveis, então escolhemos utilizar a tecnologia Axios, que de acordo com sua própria documentação, é um cliente HTTP que traz diversos benefícios para o desenvolvedor, e um dos que mais facilitou o desenvolvimento foi a questão de já transformar os dados recebidos em JSON de

maneira automática, o que diferencia ele do *Fetch*[12] nativo do JavaScript, além de outras diversas configurações que podem ser feitas, todas essas questões nos fizeram escolher o Axios como tecnologia para conseguir consumir os dados entregues pela API, na figura 41 é evidenciado a configuração desta tecnologia.

Figura 41 - Configuração da ferramenta Axios.

```
import axios from 'axios'

const PORT = import.meta.env.VITE_REACT_APP_PORT;

const blogFetch = axios.create({
    baseURL: 'http://localhost:${PORT}',
    headers: {
        "Content-Type": "application/json",
    }
})

export default blogFetch
```

Fonte: Elaborado pelos autores, 2024.

Após realizar essas configurações, foi possível começar a utilizar o Axios para consumir a API do back-end como exemplificado na figura 42, sendo que essa configuração só precisa ser feita uma única vez e após isso só sendo necessário reutilizar essa variável já criada e configurada da maneira desejada.

[12] A função fetch ou Fetch API, é uma API de busca do Javascript que permite realizar requisições HTTP assíncronas entre uma aplicação web e recursos externos. [11] https://www.dio.me/articles/fetch-a-funcao-mais-importante-do-javascript#:~:text=A%20fun%C3%A7%C3%A3o%20fetch%20ou%20Fetch,do%20Javascript%20em%20navegadores%20modernos.

Figura 42 - Requisição para listar empresas.

```
const listarEmpresas = async () => {
    try {
        const response = await blogFetch.get('/listarEmpresas');
        const data = response.data;
        setEmpresas(data.listarEmpresas);

    } catch (error) {
        console.log(error);
    }
}
```

Fonte: Elaborado pelos autores, 2024.

8.4 FRONT-END MOBILE

No desenvolvimento de aplicação mobile foi utilizado a tecnologia React Native, criada pelo Facebook em 2015, se tratando de um framework feito para o desenvolvimento de aplicativos móveis multiplataforma. (Andrade, 2020).

O framework possibilita o desenvolvimento de aplicações para diversos sistemas operacionais de dispositivos móveis, o que foi um diferencial determinante para a escolha. Além disso, ele se baseia nos conceitos do React, uma tecnologia já familiarizada pela equipe desenvolvedora. Considerando todos esses aspectos, a decisão de utilizar esse framework se mostra altamente justificada.

Assim como o React, esse framework permite a divisão do código em diversos componentes do sistema, que podem ser reutilizados em outras lógicas ou telas da aplicação, o que torna o desenvolvimento da aplicação mais eficiente, otimizado e rápido, além de possuir seus próprios componentes que servem para adaptar funções que são utilizadas em aplicativos web por meio do HTML, para o contexto de uma aplicação mobile.

A tecnologia Expo foi empregada para gerar o código base da aplicação e possibilitar a simulação dela em dispositivos móveis. Dado que a aplicação consiste em várias telas independentes que ainda necessitam interagir entre si, optou-se por

utilizar a tecnologia React Navigation. Essa ferramenta permite a definição de um componente para cada página e a especificação de como ocorrerá a interação entre elas, a figura 43 representa todas as rotas da aplicação Mobile.

Figura 43 - Todas as rotas utilizadas na aplicação Mobile.

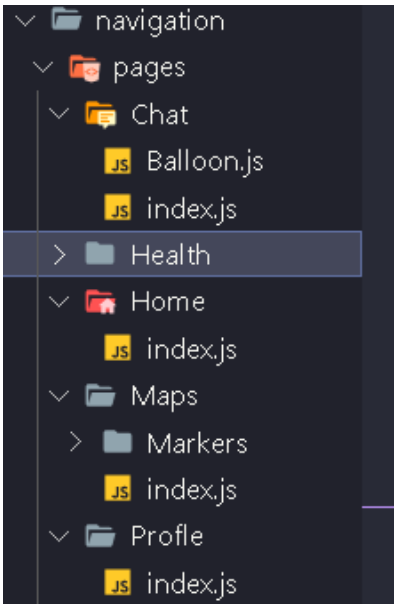
```
function App(){
  return(
    <NavigationContainer independent={true}>
      <Stack.Navigator initialRouteName="HealthScreen" screenOptions={{headerShown: false,}}>
        <Stack.Screen name="Login" component={Login} />
        <Stack.Screen name="Register" component={Register} />
        <Stack.Screen name="ProfileScreen" component={ProfileScreen} />
        <Stack.Screen name="MainContainer" component={MainContainer} />
        <Stack.Screen name="Chat" component={Chat} />
        <Stack.Screen name="HomeScreen" component={HomeScreen} />
        <Stack.Screen name="HealthScreen" component={HealthScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  )
}
```

Fonte: Elaborados pelos autores, 2024.

8.4.1 PÁGINAS

Da mesma forma que no front-end Web da aplicação, as páginas da aplicação móvel são compostas por componentes que reutilizam outros componentes para sua construção. Cada uma dessas páginas é independente e interage entre si para permitir a interação do usuário com o sistema. Na figura 44 destacamos todas as páginas desta aplicação, sendo que cada página é estilizada e projetada visando usabilidade focando sempre na experiencia do usuário.

Figura 44 - Páginas da aplicação Mobile.



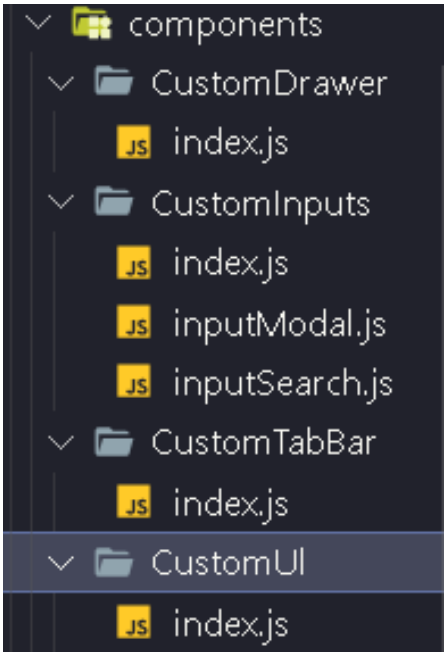
Fonte: Elaborado pelos autores, 2024.

Uma vez que o React Native mantém os princípios do React, todas essas páginas são transformadas em páginas estáticas HTML. Isso é alcançado por meio do conceito de SPA (Single Page Application), visando proporcionar uma experiência aprimorada para todos os usuários que interagem com o sistema.

8.4.2 COMPONENTES

Os componentes consistem em partes pequenas do código que são segmentadas para serem reutilizadas posteriormente nas páginas do sistema. Estes componentes estão representadas na figura 45 e cada um destes componentes possui sua própria customização e oferece métodos diversos para permitir que o usuário interaja de maneiras específicas com a aplicação.

Figura 45 - Componentes da aplicação Mobile.

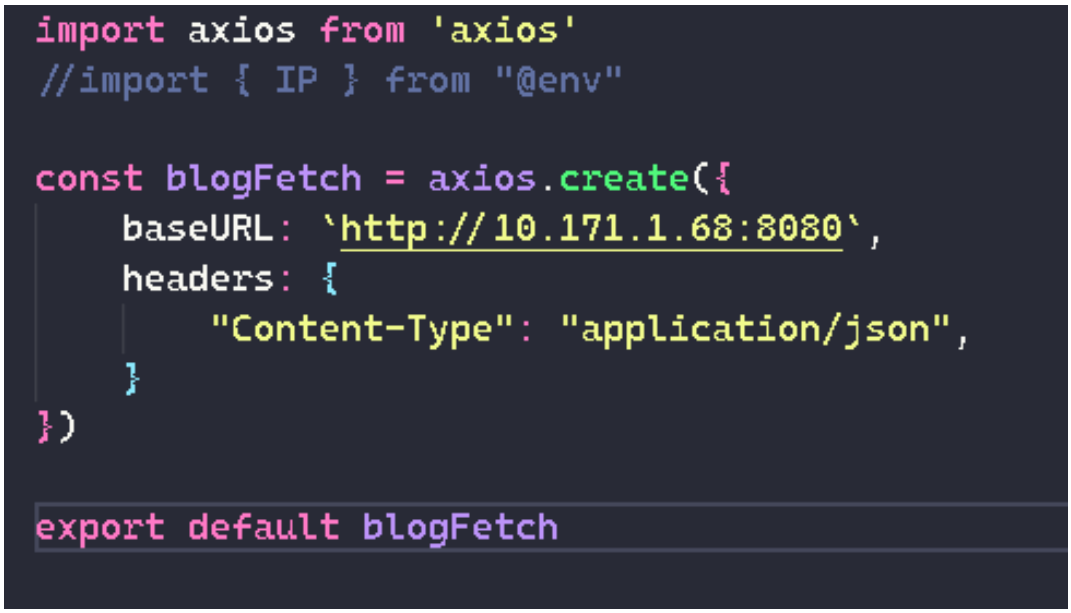


Fonte: Elaborado pelos autores, 2024.

8.4.3 CONSUMO DA API

A utilização da tecnologia Axios, que foi empregada na versão web da aplicação, possibilita a interação com as informações provenientes da API de forma simples e rápida, como mencionado anteriormente. O Axios funciona como um cliente HTTP que simplifica essas requisições, permitindo que o aplicativo móvel interaja com os dados enviados pelo back-end da aplicação, a figura 46 representa a configuração desta ferramenta.

Figura 46 - Configuração da ferramenta Axios.



Fonte: Elaborado pelos autores, 2024.

Em relação a configuração do Axios, a única diferença que existe da versão Web para a Mobile é que se faz necessário definir o IP de onde está sendo executada a API para que o Expo consiga identificar e dessa forma resgatar esses dados que são trocados entre o front e o back-end da aplicação. E após realizar essa configuração, é possível já utilizar o Axios em todas os componentes do sistema como exemplificado na figura 47, não sendo necessário repetir essa configuração que já foi feita.

Figura 47 - Requisição para listar os comentários dos pacientes.

Fonte: Elaborado pelos autores, 2024.

9 CONSIDERAÇÕES FINAIS

Inicialmente, o projeto estava concebido para fornecer uma visualização da disponibilidade de profissionais por especialidade em unidades de saúde particulares, juntamente com a localização dessas unidades em um mapa. Contudo, após a condução de pesquisas qualitativas, foi identificada a viabilidade de incorporar funcionalidades adicionais para atender a uma gama mais ampla de necessidades.

A integração de novas ideias com as já existentes elevou o projeto além da mera visualização de informações. Agora, ele também facilita a comunicação entre os usuários, ampliando assim o alcance para uma variedade maior de público e fortalecendo a proposta do sistema de forma significativa.

Após a consolidação da ideia, deu-se início ao desenvolvimento do projeto, que envolveu várias etapas. Isso incluiu a identificação das entidades, a modelagem dos principais diagramas para esclarecer o funcionamento do sistema. Foi necessário também decidir em qual plataforma o sistema estaria disponível. Através de pesquisas qualitativas, foi observada a acessibilidade proporcionada pela plataforma móvel aos usuários. Isso levou à ideia de desenvolver o projeto tanto para a plataforma web quanto para a móvel.

Para o desenvolvimento das aplicações, foi necessário dividir precisamente cada responsabilidade entre os integrantes da equipe, com base em seus pontos fortes. Essa divisão resultou em uma facilidade no desenvolvimento e na consolidação das aplicações como um todo.

Após a conclusão do projeto, apesar das dificuldades comuns enfrentadas no desenvolvimento de aplicativos, o resultado foi satisfatório. O projeto alcançou sua proposta inicial, uma vez que todas as funcionalidades planejadas foram implementadas no sistema.

REFERÊNCIAS

ALMEIDA, Marcus. **Banco de dados relacionais: conhecendo conceitos, terminologias e ferramentas**. Alura, 2023. Disponível em: https://www.alura.com.br/artigos/banco-dados-relacionais-conceitos-terminologias-ferramentas

Amazon, **Amazon S3,** [S.D]. Disponível em: https://aws.amazon.com/pt/s3/

ANDRADE, Ana Paula de. **O que é o React Native.** Treinaweb, 2020. Disponível em: https://www.treinaweb.com.br/blog/o-que-e-o-react-native

AWARI, **Guia Completo de Dicionário de Dados para Sql Server: Tudo o que Você Precisa Saber**, 2023. Disponível em: https://awari.com.br/guia-completo-de-dicionario-de-dados-para-sql-server-tudo-o-que-voce-precisa-saber/

AWARI, **Scrum: o que é e como aplicar a metodologia ágil**, 2021. Disponível em: https://awari.com.br/o-que-e-scrum/

AWS. **O que é uma aplicação Web**, 2023. Disponível em: https://aws.amazon.com/pt/what-is/web-application/#:~:text=Uma%20aplica%C3%A7%C3%A3o%20Web%20%C3%A9%20um,de%20forma%20conveniente%20e%20segura

AZURE, **O que é o desenvolvimento de aplicações móveis,** [S.D]. Disponível em: https://azure.microsoft.com/pt-pt/resources/cloud-computing-dictionary/what-is-mobile-app-development/#definition

BESSA, André. Node.JS: **o que é, como funciona esse ambiente de execução JavaScript e um Guia para iniciar**, Alura, 2023. Disponível em: https://www.alura.com.br/artigos/node-js

BOCARD, Taysa. **O que são aplicativos? Definição da desenvolvedora Usemobile**. [S.l]: Usemobile, 2021. Disponível em: https://usemobile.com.br/aplicativo-movel/

Developer Mozilla, JavaScript, [S.D]. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript

DevMedia, **Já ouviu falar em Single Page Applications**, [S.D]. Disponível em: https://www.devmedia.com.br/ja-ouviu-falar-em-single-page-applications/39009#:~:text=Single%20Page%20Applications%20(SPA)%20s%C3%A3o,a%20estrutura%20da%20p%C3%A1gina%20est%C3%A1tica.

DevMedia, **Modelagem de Dados 1: Entidades**, [S.D]. Disponível em: https://www.devmedia.com.br/modelagem-de-dados-1-entidades/4140

DiagRad, **Saiba como acalmar o paciente em filas de hospital**, [S.D]. Disponível em: https://diagrad.com.br/noticias/tempo-de-espera-em-filas-de-hospital/

DOCS, **Axios**. Disponível em: https://axios-http.com/ptbr/docs/intro

DOCS, **NodeJS**. Disponível em: https://nodejs.org/en

DOCS, **Postman**. Disponível em: https://www.postman.com/

DOCS, **Prisma**. Disponível em: https://www.prisma.io/docs/getting-started

DOCS, **React Native**. Disponível em: https://reactnative.dev/

DOCS, **React**. Disponível em: https://pt-br.react.dev/

DOCS, **Swagger**. Disponível em: https://swagger.io

DOCS, **Typescript**. Disponível em: https://www.typescriptlang.org/

DOCS, **Visual Studio Code.** Disponível em: https://code.visualstudio.com/docs

Google Cloud, **O que é o MySQL**. Disponível em: https://cloud.google.com/mysql?hl=pt-br

HUMMEL, Guilherme. **‘Triagem Digital’: fila única de espera é obsolescência hospitalar**. Saúde Business, 2022. Disponível em: https://www.saudebusiness.com/colunas/triagem-digital-fila-unica-de-espera-e-obsolescencia-hospitalar

IBM, **Diagramas de Caso de Uso**, 2021. Disponível em: https://www.ibm.com/docs/pt-br/rsm/7.5.0?topic=diagrams-use-case

IBM, **O que é um banco de dados relacional**, [S.D]. Disponível em: https://www.ibm.com/br-pt/topics/relational-databases

L., Andrei. **O Que é GitHub, Para Que Serve e Como Usar. Hostinger, 2023**. Disponível em: https://www.hostinger.com.br/tutoriais/o-que-github

Lucidchart, **O que é um diagrama de classe UML**, [S.D]. Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-diagrama-de-classe-uml

Lucidchart, **O que é um diagrama UML**, [S.D]. Disponível em: https://www.lucidchart.com/pages/pt/o-que-e-uml

MATHIAS, Lucas. **Pesquisa qualitativa e quantitativa: qual é a melhor opção**: Mind Miners, 2022. Disponível em: https://mindminers.com/blog/pesquisa-qualitativa-quantitativa/#:~:text=A%20pesquisa%20qualitativa%20%C3%A9%20aquela,pontos%20de%20vista%2C%20entre%20outros

MUNHAS, Rafael dos Santos. **Para que serve Javascript: por que implementar em sites e aplicativos**: Go Daddy, 2023. Disponível em: https://www.godaddy.com/resources/br/artigos/para-que-serve-javascript

ORACLE, **O que é um banco de dados relacional (RDBMS)**, [S.D]. Disponível em: https://www.oracle.com/br/database/what-is-a-relational-database/

ORACLE, **O que é um Banco de Dados**, [S.D]. Disponível em: https://www.oracle.com/br/database/what-is-database/

PEREIRA, Antonio. **Como a tecnologia pode auxiliar na redução das filas do SUS**. Linkedin, 2023. Disponível em: https://www.linkedin.com/pulse/como-tecnologia-pode-auxiliar-na-redu%C3%A7%C3%A3o-das-filas-do-antonio/?originalSubdomain=pt

ROMULO. **React.JS: Criando rotas com React Router Dom**. DevMedia, 2021. Disponível em: https://www.devmedia.com.br/react-js-criando-rotas-com-react-router-dom/42901

ROSSETTI, Micaela L.. **Protótipo: baixa e alta fidelidade**: Soft Design, 2020. Disponível em: https://softdesign.com.br/blog/prototipo-baixa-e-alta-fidelidade/

SARMENTO JUNIOR, Krishnamurti Matos de Araujo; TOMITA, Shiro; KOS, Arthur Octavio de Avila. **O problema da fila de espera para cirurgias otorrinolaringológicas em serviços públicos**. [S.l]: Revista Brasileira de Otorrinolaringologia, 2005. 7 p. Disponível em: https://www.scielo.br/j/rboto/a/czxcPhw7RYhr3zXqwbwYLsf/?format=pdf

SILVA, Cleuton. **Vantagens da Programação Orientada a Objetos (POO)**. Dio, 2023. Disponível em: https://www.dio.me/articles/vantagens-da-programacao-orientada-a-objetos-poo

SILVESTRE, Gabriel. **Controller e Service - Uma breve introdução.** Dev, 2022. Disponível em: https://dev.to/gabrielhsilvestre/controller-e-service-uma-breve-introducao-24hk

Somos Tera. **Prototipagem de alta fidelidade: o que é, quando, por que e como usar**, 2020. Disponível em: https://medium.com/somos-tera/prototipagem-de-alta-fidelidade-635d745b662b

TechTudo, **Draw.io online permite criar gráficos e desenhos grátis sem baixar nada**, [S.D]. Disponível em: https://www.techtudo.com.br/tudo-sobre/drawio/

V/SURE, **O que são Requisitos Funcionais: Exemplos, Definição, Guia Completo**, 2024. Disponível em: https://visuresolutions.com/pt/blog/functional-requirements/

APÊNDICE A – MIDDLEWARE DE AUTENTICAÇÃO

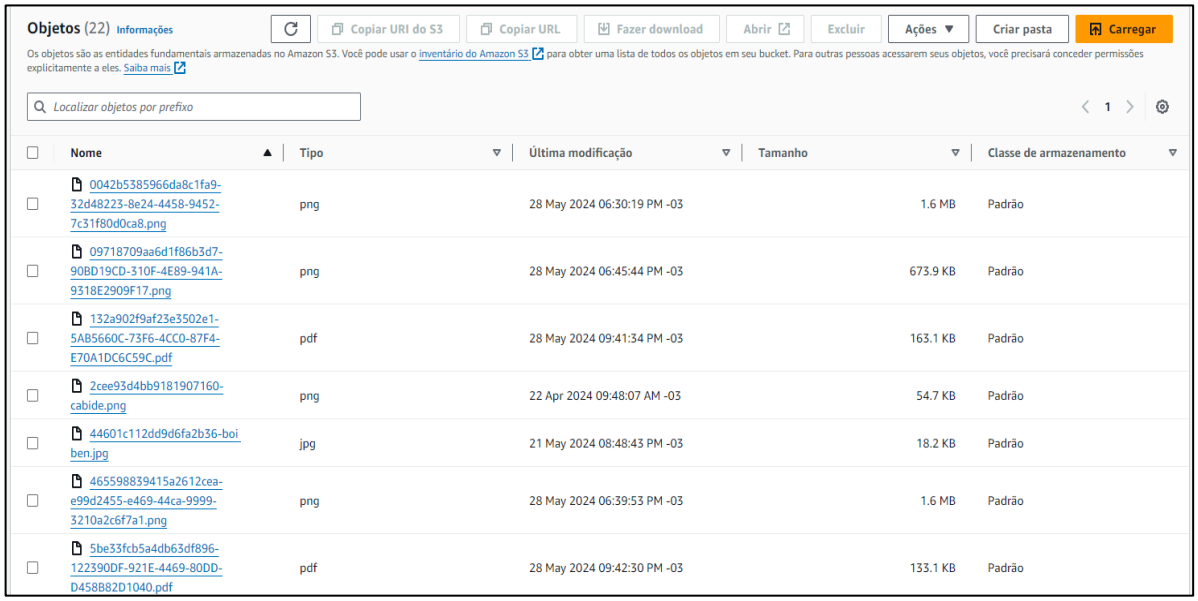
Figura 48 - Middleware de Autenticação

```
import { Request, Response, NextFunction } from "express";
import * as dotenv from 'dotenv';
import { verify } from "jsonwebtoken";
dotenv.config();

type TokenPayload = {
    id: string;
    iat: number;
    exp: number;
}

const secret = process.env.SECRET;

export function AuthMiddleware(req: Request, res: Response, next: NextFunction) {
    const { authorization } = req.headers;

    if (!authorization) {
        return res.status(401).json({ error: 'Token não fornecido!' });
    }

    const [, token] = authorization.split(" ");

    try {
        const decoded = verify(token, secret as string);
        const { id } = decoded as TokenPayload;

        req.userId = id;
        next();
    } catch (error) {
        return res.status(401).json({ error: 'Token invalido!' });
    }
}
```

Fonte: Elaborado pelos autores, 2024.

APÊNDICE B – VISÃO DA BUCKET DO AMAZON S3

Figura 49 - Bucket Amazon S3



Fonte: Elaborado pelos autores, 2024.