Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Departamento de Engenharia Elétrica

Relatório de Estágio Supervisionado

Desenvolvimento de um Sistema de Gerenciamento de Documentos para o LABMET

Bruna Eduarda Cruz Maciel

Campina Grande - PB
Novembro de 2024

Bruna Eduarda Cruz Maciel

Desenvolvimento de um Sistema de Gerenciamento de Documentos para o LABMET

Relatório de Estágio Supervisionado submetido à Coordenação do Curso de Engenharia Elétrica da Universidade Federal de Campina Grande – UFCG, como parte dos requisitos para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Universidade Federal de Campina Grande - UFCG

Centro de Engenharia Elétrica e Informática - CEEI

Departamento de Engenharia Elétrica - DEE

Coordenação de Graduação em Engenharia Elétrica - CGEE

Edmar Candeia Gurjão, D.Sc.

(Orientador)

Campina Grande - PB

Novembro de 2024

Bruna Eduarda Cruz Maciel

Desenvolvimento de um Sistema de Gerenciamento de Documentos para o LABMET

Relatório de Estágio Supervisionado submetido à Coordenação do Curso de Engenharia Elétrica da Universidade Federal de Campina Grande – UFCG, como parte dos requisitos para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Aprovado em ____ /____ /____

Leocarlos Bezerra da Silva Lima
Universidade Federal de Campina Grande
Avaliador

Edmar Candeia Gurjão
Universidade Federal de Campina Grande
Orientador

Campina Grande - PB
Novembro de 2024

Dedico este trabalho a Deus,que me proporcionou sabedoria para superar desafios e aos meus pais, pela educação, amor e cuidado ao longo de toda a minha vida e me deram oportunidades de chegar onde cheguei.

Agradecimentos

À Deus por me proporcionar a benção de poder realizar um curso superior, e me manter perseverante na elaboração deste trabalho, me trazendo foco e me direcionando no caminho correto. Em cada desafio, pude perceber Sua providência cuidando dos detalhes e abrindo portas que jamais imaginei. É pela Sua infinita graça que esta caminhada foi possível, e eu reconheço que tudo o que alcancei é fruto da Sua vontade e destinado à Sua glória.

À minha família, um agradecimento especial. Vocês sempre estiveram presentes em todos os momentos da minha vida. O apoio incondicional e a fé que depositaram em mim foram essenciais para que eu alcançasse este marco. Em especial, sou eternamente grata à meus pais, Marta e Jailton, por todo sacrificio, apoio e nunca deixarem faltar nada,principalmente o mais importante,amor.

Aos amigos que compartilharam esta jornada comigo na universidade, dividindo noites em claro e celebrando conquistas, saibam que os laços que criamos são valiosos e levarei comigo por toda a vida. Agradeço por sempre acreditarem em mim.

Aos amigos que me acompanham desde a escola e aos que fiz ao longo do caminho, sou grata por estarem sempre presentes e por serem fontes de apoio e amizade verdadeira. Obrigada por me mostrarem que posso contar com vocês em qualquer momento.

A todos que contribuíram para este trabalho, oferecendo auxílio e ideias valiosas, meu sincero agradecimento por seu apoio fundamental.

Por último, gostaria de expressar minha gratidão aos professores que me acompanharam ao longo desta jornada acadêmica, especialmente ao professor Edmar. Sua orientação e comprometimento foram verdadeiramente valiosos. O seu jeito divertido e motivador faz o aprendizado muito mais leve e especial, e por isso escolhi trabalhar com ele durante meu estágio. Agradeço também aos colaboradores do LABMET por todo o apoio e contribuições. A presença e o suporte de vocês foram fundamentais para meu crescimento e conquistas.

"Dificuldades preparam pessoas comuns para destinos extraordinários",
(C.S Lewis)

Resumo

Este trabalho tem como objetivo o desenvolvimento de um sistema de gerenciamento de documentos para o Laboratório de Metrologia (LABMET) buscando não apenas facilitar o acesso à informação, mas também acelerar o processo de busca, tornando a rotina da equipe mais ágil e produtiva. Reconhecemos que as informações são o maior patrimônio de qualquer organização. Por isso, gerenciá-las de forma simples e eficaz é essencial modernização,otimização e destaque. Com este novo sistema, esperamos transformar a maneira como os documentos são geridos, ajudando a equipe a se concentrar no que realmente importa: a inovação e a excelência nas pesquisas.

Palavras-chave: Gestão de documentos, React. Node.js,Aplicação Web,JavaScript,Typescript.

Abstract

This work aims to develop a document management system for the Metrology Laboratory (LABMET), seeking not only to facilitate access to information but also to accelerate the search process, making the team's routine more agile and productive. We recognize that information is the greatest asset of any organization. Therefore, managing it simply and effectively is essential for modernization, optimization, and prominence. With this new system, we hope to transform the way documents are managed, helping the team focus on what truly matters: innovation and excellence in research.

Keywords: Document Management, React, Node.js, Web Application,JavaScript, TypeScript.

Lista de ilustrações

Figura 1 – Requisição HTTP no Insomnia 7
Figura 2 – Rest API 9
Figura 3 – Visual Studio Code 11
Figura 4 – Sistema de Controle de Versão Distribuído 12
Figura 5 – Sistema de Controle de Versão Distribuído 14
Figura 6 – Diagrama de caso de uso 17
Figura 7 – Diagrama de caso de uso 18
Figura 8 – Diagrama de classes 18
Figura 9 – Tela de Login dark mode 20
Figura 10 – Tela de Login white mode 21
Figura 11 – Tela de Cadastro white mode 22
Figura 12 – Tela de Cadastro dark mode 22
Figura 13 – Modal de Cadastro de usuários 23
Figura 14 – Tela de Arquivos 23
Figura 15 – Tela de Arquivos Internos 24
Figura 16 – Tela de Lista de Arquivos 25
Figura 17 – Tela Modal de Cadastro de Arquivos 26
Figura 18 – Tela de Configurações de perfil 26
Figura 19 – Tela Home 27
Figura 20 – Tela de Calendário 28

Lista de abreviaturas e siglas

UFCG	Universidade Federal de Campina Grande
MERN	*MongoDB, ExpressJS, React Native e NodeJS*
API	*Application Programming Interface*
IDE	*Integrated Development Environment*
JSON	*JavaScript Object Notation*
PDF	*Portable Document Format*
TI	*Tecnologia da Informação*
UI	*User Interface*
UX	*User Experience*
HTTP	*Hypertext Transfer Protocol*
AWS	*Amazon Web Services*
LABMET	*Laboratório de Metrologia*

Sumário

1 INTRODUÇÃO 1
1.1 Objetivos 1
1.1.1 Objetivos Gerais 1
1.1.2 Objetivos Específicos 1
1.2 Justificativa 2
1.3 Metodologia 2
1.3.1 Levantamento de Requisitos 2
1.3.2 Análise e Planejamento 2
1.3.3 Desenvolvimento 2
1.3.4 Testes 3
1.3.5 Implantação 3
1.3.6 Treinamento e Suporte 3
1.3.7 Avaliação e Melhoria Contínua 3
1.4 Resultados Esperados 3
1.5 Estrutura do documento 4

2 REFERENCIAL TEÓRICO 5
2.1 Engenharia de Software 5
2.1.1 Diagrama de casos de uso 5
2.1.2 Diagrama de classes 5
2.1.3 Requisitos funcionais 5
2.1.4 Requisitos não funcionais 6
2.1.5 Model-View-Controller ou MVC 6
2.2 Desenvolvimento Web 6
2.2.1 Protocolo HTTP 6
2.2.2 Frontend 7
2.2.3 Backend 8
2.2.4 Banco de dados 9
2.2.5 Ferramentas 10
2.3 Fundamentação Tecnológica 11
2.3.1 Visual Studio Code 11
2.3.2 GIT E GITHUB 11
2.3.3 React JS 12
2.3.4 Node JS 13
2.3.5 Node.js Express 13

2.3.6 MongoDB . . . 13
2.4 Arquitetura da Aplicação . . . 14

3 METODOLOGIA DA PESQUISA . . . 15
3.1 Requisitos . . . 15
3.1.1 Requisitos funcionais . . . 16
3.1.2 Requisitos não funcionais . . . 16
3.1.3 Modelagem do sistema . . . 16

4 RESULTADOS E DISCUSSÕES . . . 19
4.1 Levantamento de Requisitos . . . 19
4.2 Visão Geral do Sistema . . . 19
4.3 Análise das páginas funcionais . . . 20
4.3.1 Login . . . 20
4.3.2 Cadastro . . . 21
4.3.3 Modal Cadastro . . . 22
4.3.4 Tela de Arquivos Geral . . . 23
4.3.5 Tela de Arquivos Internos . . . 24
4.3.6 Tela de Lista de Arquivos . . . 24
4.3.7 Tela Modal de Cadastro de Arquivos . . . 25
4.3.8 Tela de Configurações . . . 26
4.3.9 Tela Home . . . 27
4.3.10 Tela de Calendário . . . 28

5 CONSIDERAÇÕES FINAIS . . . 29
5.0.1 Conclusões . . . 29
5.0.2 Perspectivas Futuras . . . 29
5.0.3 Experiência Adquirida . . . 30

REFERÊNCIAS BIBLIOGRÁFICAS . . . 32

1 Introdução

O estágio supervisionado desempenha um papel fundamental na formação, proporcionando aos estudantes a oportunidade de aplicar o conhecimento teórico adquirido na universidade enquanto desenvolvem novas habilidades e competências essenciais para o crescimento profissional. Neste relatório, serão apresentadas as atividades desempenhadas por Bruna Eduarda Cruz Maciel durante o período de 26/08/2024 a 12/10/2024, no Laboratório de Metrologia (LABMET), localizada em Campina Grande-PB,na Universidade Federal de Campina Grande (UFCG) sob supervisão de Edmar Candeia Gurjão.

O estágio teve uma carga horária de 30 horas semanais, totalizando 200 horas ao final do programa. Durante este período, a aluna se dedicou ao desenvolvimento de um sistema de gerenciamento de documentos, com foco na organização,segurança e otimização do acervo do laboratório.

As atividades realizadas incluíram o levantamento de requisitos junto aos usuários para entender suas necessidades, a análise e planejamento do sistema, e a implementação de funcionalidades que visam otimizar o acesso e o controle de arquivos, assim como também foi responsável por realizar testes do sistema. Ao longo do estágio, a aluna não apenas adquiriu experiência prática em gestão de documentos, mas também desenvolveu habilidades em programação, design de interfaces e modelagem de sistemas.

1.1 Objetivos

Esta seção tem por finalidade apresentar e descrever os objetivos gerais e específicos deste trabalho.

1.1.1 Objetivos Gerais

Desenvolver um sistema de gerenciamento de documentos para o LabMet que otimize a organização, acesso e compartilhamento de informações, visando melhorar a eficiência operacional e a segurança dos dados.

1.1.2 Objetivos Específicos

- Criar uma interface intuitiva que facilite a interação dos usuários com o sistema.
- Garantir a segurança dos dados através de autenticação e controle de acesso.

- Implementar funcionalidades de busca e filtragem para agilizar o acesso aos documentos.
- Desenvolver um sistema de controle de versões que permita rastrear alterações e manter a integridade dos documentos.
- Treinar os usuários para a correta utilização do sistema e promover a adaptação à nova ferramenta.

1.2 Justificativa

A gestão eficiente de documentos é crucial para o bom funcionamento de qualquer laboratório, especialmente no LabMet, onde a quantidade de informações e documentos pode ser elevada. O uso do Google Drive, embora útil, apresenta limitações em termos de organização e controle, podendo resultar em dificuldade de acesso e na perda de dados. Com o desenvolvimento de um sistema específico, é possível melhorar a rastreabilidade, aumentar a segurança das informações e otimizar a colaboração entre os membros da equipe. Além disso, um sistema customizado atenderá às necessidades particulares do LabMet, promovendo um ambiente de trabalho mais produtivo.

1.3 Metodologia

A metodologia para o desenvolvimento do sistema será dividida nas seguintes etapas:

1.3.1 Levantamento de Requisitos

Realizar reuniões com os usuários do LabMet para identificar suas necessidades e expectativas em relação ao sistema.

1.3.2 Análise e Planejamento

Elaborar um documento de especificação de requisitos que detalhe as funcionalidades desejadas e a arquitetura do sistema.

1.3.3 Desenvolvimento

Utilizar uma abordagem ágil (como Scrum) para o desenvolvimento do sistema, permitindo iterações e adaptações conforme o feedback dos usuários. Implementar as funcionalidades de gerenciamento de documentos, incluindo upload, download, busca e controle de versões.

1.3.4 Testes

Realizar testes unitários e testes com usuários para garantir que o sistema atenda aos requisitos e funcione conforme esperado.

1.3.5 Implantação

Realizar a migração dos documentos existentes para o novo sistema e disponibilizar o sistema para uso.

1.3.6 Treinamento e Suporte

Promover treinamentos para os usuários e oferecer suporte técnico para a resolução de problemas durante a adaptação ao novo sistema.

1.3.7 Avaliação e Melhoria Contínua

Coletar feedback dos usuários após a implementação e realizar ajustes conforme necessário para melhorar a experiência do usuário.

1.4 Resultados Esperados

O novo sistema de gerenciamento de documentos para o LabMet tem como objetivo transformar a forma como a equipe lida com informações. Esperamos que, com uma estrutura mais organizada, os membros do laboratório consigam encontrar documentos de maneira rápida e eficiente, economizando tempo e reduzindo a frustração.

A automação de tarefas manuais permitirá que todos se dediquem a atividades mais importantes e criativas, potencializando a produtividade e a colaboração. Acesso fácil e seguro às informações fará com que a troca de ideias entre os colegas se torne mais fluida, fortalecendo o trabalho em equipe.

Além disso, a segurança dos dados será uma prioridade. Com controles de acesso, garantiremos que apenas pessoas autorizadas possam visualizar informações sensíveis, trazendo mais tranquilidade a todos. O sistema também permitirá o rastreamento de versões dos documentos, o que ajudará a manter a precisão e a integridade das informações.

Por fim, queremos que todos se sintam confortáveis e satisfeitos com a nova ferramenta. Por meio de treinamentos e suporte contínuo, estaremos prontos para ajudar na transição e na adaptação ao novo sistema. A coleta de feedback será essencial para aprimorá-lo ao longo do tempo, assegurando que ele atenda às necessidades de todos no LabMet.

1.5 Estrutura do documento

O trabalho está dividido da seguinte forma: neste capítulo, foi apresentada uma introdução sobre o tema, incluindo a contextualização do problema, a motivação para o desenvolvimento do sistema, a definição dos desafios enfrentados e os objetivos pretendidos. No Capítulo 2, é apresentado o referencial teórico, onde são discutidos conceitos relevantes sobre gerenciamento de documentos, suas melhores práticas e tecnologias, além de abordar alguns sistemas existentes na área que se relacionam com o desenvolvimento deste trabalho.

No Capítulo 3, é demonstrada a metodologia utilizada no desenvolvimento do sistema de gerenciamento de documentos, detalhando as etapas de levantamento de requisitos, análise, desenvolvimento, testes e implantação. No Capítulo 4, são apresentados os resultados obtidos, incluindo a análise das funcionalidades implementadas e discussões sobre as decisões de design e desenvolvimento do código da solução proposta.

Por fim, o Capítulo 5 contém a conclusão, onde são discutidas as contribuições do trabalho, além de possíveis melhorias e direções para pesquisas e desenvolvimentos futuros que podem ser explorados a partir deste projeto.

2 Referencial Teórico

Para a realização deste trabalho, faz-se necessário o entendimento de duas áreas do conhecimento: Engenharia de Software e Desenvolvimento Web. A primeira área tem o objetivo de apoiar o desenvolvimento de todo o sistema. A segunda área elucida como as aplicações se comunicam e compartilham dados, além de mostrar o que é necessário para criação e manutenção da parte web do sistema.

2.1 Engenharia de Software

A engenharia de software é uma disciplina da informática que se ocupa de todos os aspectos da produção de software, desde as primeiras etapas de especificação do sistema, até a manutenção desse sistema e depois que o mesmo já se encontra em uso (SOMMERVILLE, 2011). Neste trabalho serão utilizados diversos conceitos presentes na área da engenharia de software, como o modelo arquitetural Model-View-Controller, ou MVC, e algumas ferramentas como os diagramas de caso de uso e os diagramas de classes.

2.1.1 Diagrama de casos de uso

Este tipo de diagrama descreve uma possível utilização do sistema do ponto de vista do usuário. Para tanto, ele descreve as principais funcionalidades e interações dessas funcionalidades com os usuários (JACOBSON, 1992).

2.1.2 Diagrama de classes

O diagrama de classes é útil para ilustrar e documentar a estrutura e as relações das classes que servem de modelo para os objetos (FALBO, 2017). Este tipo de diagrama mapeia os atributos, operações e relações entre os objetos do sistema, descrevendo exatamente aquilo que deve estar presente no sistema a ser modelado.

2.1.3 Requisitos funcionais

Os requisitos funcionais representam todas as ações que o software deve realizar por meio de suas funções ou serviços . São descrições das interações entre o sistema e o ambiente e de seus comportamentos através de entradas específicas.

2.1.4 Requisitos não funcionais

Utilizados para definir restrições e limitações no desenvolvimento dos sistemas, os requisitos não funcionais têm origem em restrições de orçamento, necessidades dos usuários finais ou mesmo integração com outros sistemas, a fim de garantir a interoperabilidade entre eles (FALBO, 2017). Um exemplo de requisito não funcional neste trabalho seria "o sistema deve ser construído utilizando a linguagem de programação JavaScript".

2.1.5 Model-View-Controller ou MVC

Este padrão arquitetural é composto por três camadas que estão presentes em seu nome: Model, a camada responsável pela lógica da aplicação, que realiza as operações do sistema de fato; View, a parte que o usuário interage, responsável por toda a exibição das informações para o usuário; e Controller, responsável por gerenciar o fluxo de dados e ações entre as requisições dos usuários e o que é repassado para a Model. O MVC é muito utilizado por conta da sua separação de responsabilidades entre as diferentes partes do mesmo sistema (MACORATTI, 2019).

2.2 Desenvolvimento Web

Ao desenvolvimento web é conferida a responsabilidade de elaborar e codificar páginas e soluções para a internet. Esse trabalho se dá tanto pela parte que toca o usuário, o frontend, quanto pelo que está por trás do que o usuário enxerga, o backend. Apesar dessa divisão de responsabilidades entre as partes do desenvolvimento, a comunicação entre elas se dá a partir da utilização de um mesmo protocolo: o Hypertext Transfer Protocol.

2.2.1 Protocolo HTTP

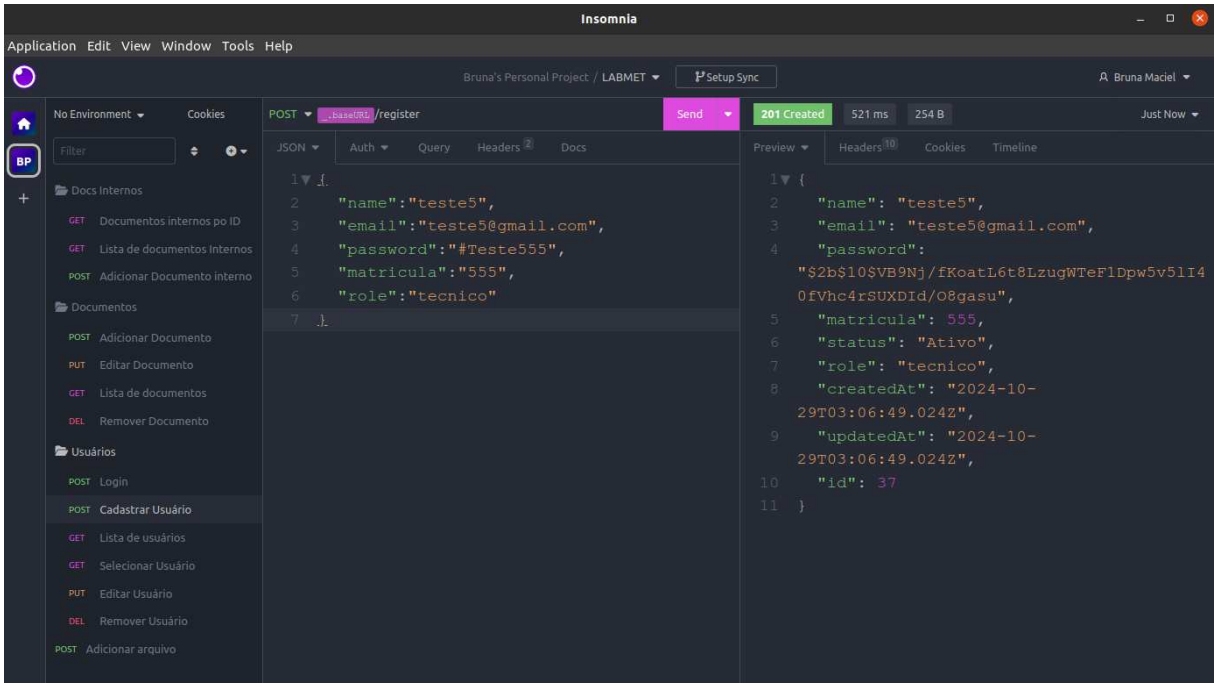
Como base para toda troca de dados realizada na internet temos o Hypertext Transfer Protocol, ou HTTP. É um protocolo que estabelece uma relação cliente-servidor entre as partes, onde o cliente inicia as requisições a fim de obter recursos de um servidor (NETWORK, 2021). Esses recursos podem ser imagens, vídeos, textos ou mesmo páginas web inteiras.

O HTTP evoluiu com o tempo e hoje possui diversos métodos de requisição, dentre eles os mais utilizados neste trabalho são:

- **GET**, para solicitar algum dado específico.
- **POST**, para enviar dados ao servidor (geralmente criando novas entradas na base de dados).
- **PUT**, utilizado para alterar ou substituir uma instância de um recurso.

- **DELETE**, utilizado para remover um recurso específico da base de dados.

Figura 1 – Requisição HTTP no Insomnia



Fonte: Autoria Própria

A figura acima, obtida utilizando o software Insomnia (INSOMINIA, 2024), mostra o exemplo de uma requisição HTTP contida neste trabalho. A fim de realizar o cadastro na aplicação, o frontend faz uma requisição do tipo POST para a rota http://localhost:3001/register, enviando as informações no formato JSON. O backend, por sua vez, processa a requisição e, em caso de sucesso, retorna as informações do usuário logado. É com base nesses métodos de requisição que a interface do usuário (frontend) interage com o servidor (backend) da aplicação. Assim como o protocolo HTTP, o frontend e o backend evoluíram ao longo do tempo. [1]

2.2.2 Frontend

O frontend é a parte da aplicação com a qual o usuário interage, funcionando como um intermediário entre o usuário e o backend. Essa interface pode ser web, mobile ou desktop. Neste trabalho, a interface web foi construída em Typescript, utilizando a biblioteca React.

No mercado, existem várias bibliotecas e frameworks JavaScript para facilitar o desenvolvimento, entre os quais se destacam Angular, React e Vue. O Angular, criado

[1] <https://insomnia.rest/>

pelo Google, utiliza TypeScript e tem uma curva de aprendizagem mais acentuada, pois exige uma estruturação rígida do código. O React, por sua vez, é mais leve e modular, permitindo que os desenvolvedores criem componentes de forma funcional, o que facilita a adição de novas funcionalidades. O Vue se destaca pela sua simplicidade e documentação clara, sendo conhecido por sua rapidez e organização do código, onde template, JavaScript e CSS podem ser organizados em um único arquivo.(MASTERS, 2021)

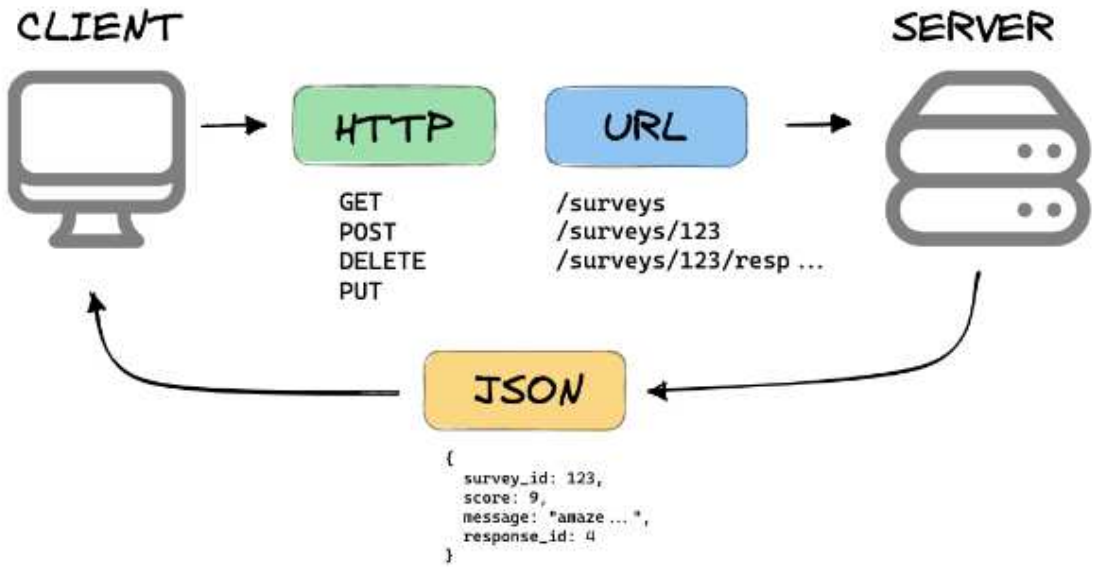
A escolha do React para este trabalho foi motivada pela sua flexibilidade e eficiência no desenvolvimento. Sua abordagem baseada em componentes permite uma organização clara do código, facilitando a reutilização e a manutenção. Além disso, o React possui um ecossistema robusto e uma comunidade ativa, o que oferece suporte e recursos valiosos. Essa combinação de modularidade, desempenho e facilidade de integração com outras bibliotecas torna o React uma solução ideal para construir interfaces de usuário dinâmicas e responsivas.[2] [3] [4]

2.2.3 Backend

O backend, fundamental para o pleno funcionamento de uma aplicação, desempenha um papel vital ao viabilizar a execução das tarefas apresentadas no front-end. Esta camada, responsável pelo processamento efetivo dos dados essenciais, constitui o ambiente onde a aplicação é executada.[5]

[2] <https://angular.io/>
[3] <https://vuejs.org/>
[4] <https://www.typescriptlang.org/>
[5] <https://nodejs.org/pt-br/>

Figura 2 – Rest API



Fonte: (Howie Mann Startup Strategy, 2023)

2.2.4 Banco de dados

Conforme o site da Oracle (2023), um banco de dados é uma coleção organizada de informações armazenadas eletronicamente em sistemas de computador. Essas informações são estruturadas em tabelas, facilitando o acesso, manutenção e controle dos dados.

De acordo com Elmasri e Navathe (2011), existem várias abordagens para organizar um banco de dados:

- **Relacional:** Os dados são organizados em tabelas, utilizando relacionamentos para representar a estrutura.
- **Relacional a Objeto:** Os dados são representados como objetos, permitindo uma modelagem mais próxima do mundo real.
- **Distribuído:** Os dados podem estar em diferentes locais, mas são integrados em uma única base de dados.
- **NoSQL ou Não Relacional:** Permite a manipulação de dados não organizados, ideal para ambientes que requerem flexibilidade na gestão de dados.

Essas diferentes abordagens atendem a necessidades específicas de aplicação. A escolha entre elas depende dos requisitos do projeto, da natureza dos dados e das metas de desempenho e escalabilidade.

2.2.5 Ferramentas

Para a construção do sistema descrito neste trabalho, diversas ferramentas de desenvolvimento foram empregadas. O editor de código Visual Studio Code foi escolhido como base para o desenvolvimento do código, graças à sua flexibilidade e à ampla gama de extensões disponíveis. Para a configuração inicial da aplicação, utilizou-se o Docker, uma plataforma que permite a conteinerização de softwares. Com o Docker, o desenvolvedor pode descrever em um arquivo os softwares necessários e, com apenas algumas linhas de comando, configurar rapidamente um ambiente de desenvolvimento completo. O Docker foi crucial para agilizar o setup inicial da aplicação, permitindo que diferentes computadores pudessem colaborar no desenvolvimento.

No que diz respeito ao desenvolvimento do backend, o Node.js se destacou como um ambiente de execução que possibilita o uso da linguagem JavaScript na criação da API. Juntamente com o Node.js, o framework Express foi amplamente utilizado. O Express é um framework JavaScript que simplifica o desenvolvimento de APIs em ambientes Node.js, tornando a criação de rotas, middlewares e a manipulação de erros mais eficientes. Para testar as rotas desenvolvidas com o Express, o software Insomnia foi utilizado como cliente de API REST, permitindo realizar testes de forma rápida e prática.

Por fim, para o desenvolvimento das interfaces, a principal ferramenta utilizada foi o React, que se especializa na construção de aplicações de página única (SPAs, do inglês Single Page Applications). O React facilita a criação de interfaces ao tratar todos os seus componentes como funções, permitindo a reutilização de código de maneira eficiente. Essa funcionalidade é ainda mais destacada com a utilização da biblioteca TailwindCss,juntamente com o shadcn comobiblioteca de componentes estilizavéis.

A seguir, apresentamos um resumo das ferramentas mencionadas nesta seção, destacando o contexto de uso de cada uma durante o desenvolvimento do projeto.

Ferramenta	**Contexto**
Ubuntu 22.0.4	Backend e Frontend
Visual Studio Code	Backend e Frontend
Docker	Backend e Frontend
MongoDB	Backend
Node.js	Backend
Express	Backend
Insomnia	Backend
React	Frontend
Shadcn	Frontend
TailwindCSS	Frontend
Google Chrome	Frontend

Tabela 1 – Ferramentas utilizadas e seus contextos

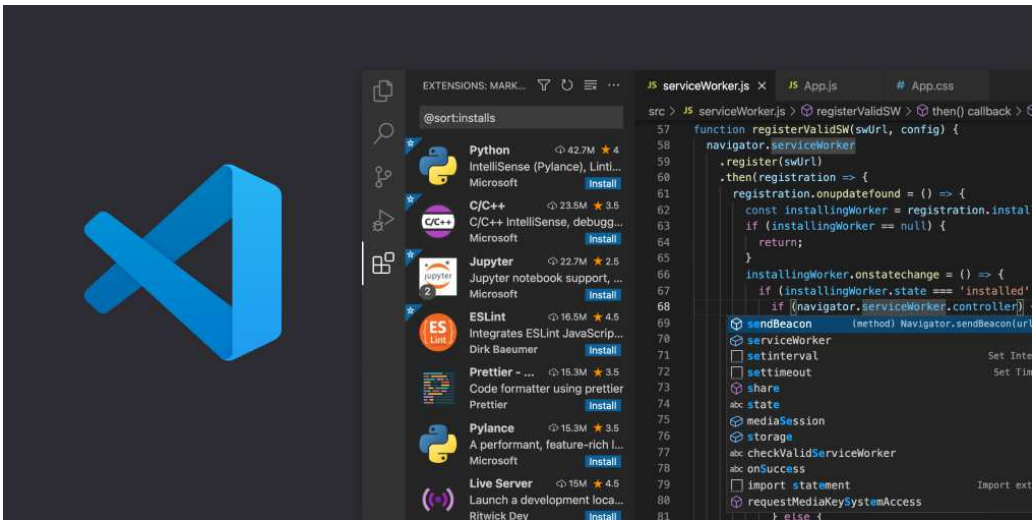
2.3 Fundamentação Tecnológica

Nesta seção será explicado um pouco mais sobre as tecnologias que foram utilizadas para a criação do Front-End da aplicação.

2.3.1 Visual Studio Code

Visual Studio Code é um editor de código-fonte leve e intuitivo para desenvolvimento, estando disponível para os sistemas operacionais macOS, Linux e Windows. Com suporte já integrado para a linguagem JavaScript. (CODE, 2024) Ele será utilizado para desenvolvimento do software, sendo utilizado hibridamente com as linguagens HTML, CSS e JS.

Figura 3 – Visual Studio Code



Fonte:(CODE, 2024)

2.3.2 GIT E GITHUB

De acordo com a Atlassian (2022), o GIT é um Sistema de Controle de Versão Distribuído (DVCS). Isso significa que, ao contrário de sistemas mais antigos, como o Subversion (SVN), onde havia um único local para armazenar todo o histórico de mudanças do software, no GIT cada programador tem uma cópia completa do código em sua própria máquina. Isso transforma cada cópia em um repositório independente.

Um repositório é a estrutura que organiza e controla o código por meio do GIT, geralmente composta de pastas. Nele, podem existir várias ramificações, ou "branches", que são cópias do repositório principal. As branches permitem que os desenvolvedores adicionem novas funcionalidades sem afetar o código original, garantindo que, caso algo dê errado, o repositório principal permaneça intacto.

Após fazer as alterações e registrar essas modificações, os desenvolvedores realizam o que chamamos de "merge", que significa integrar as mudanças da ramificação de volta à principal. Normalmente, antes de fazer o merge, as alterações são enviadas para um repositório remoto, como o GitHub ou o Azure DevOps, que abordaremos em seguida.[6]

O GIT foi fundamental no desenvolvimento deste sistema, pois nos permitiu gerenciar o histórico de alterações do código de forma eficiente. Assim, conseguimos reverter facilmente qualquer modificação que pudesse ter causado problemas.[7]

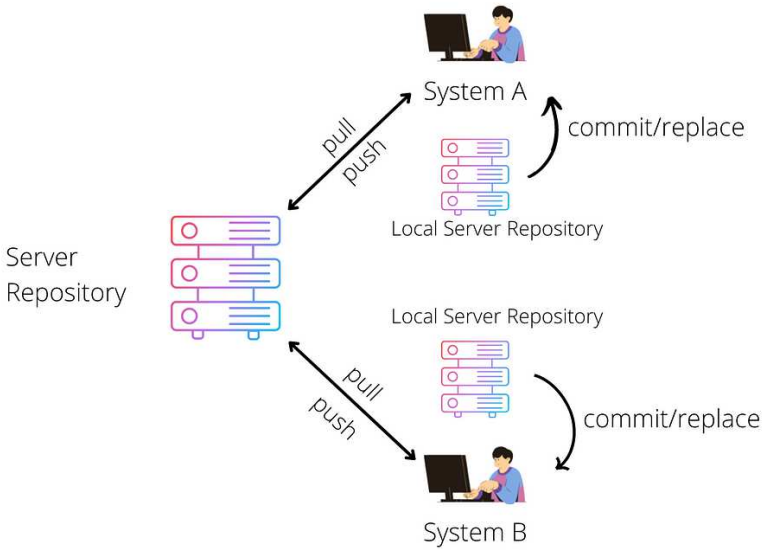


Figura 4 – Sistema de Controle de Versão Distribuído

2.3.3 React JS

Atualmente temos visto falar bastante do ReactJS, sendo uma biblioteca JavaScript de código aberto voltada para a criação de UI (User Interfaces) para aplicações SPA (Single Page Application). SPAs são aplicações de página única ou seja todo o conteúdo é acessado e visualizado com base em apenas uma página, isso é possível graças a divisão das páginas em pequenas partes às quais chamamos de componentes, estes são representados dentro de tags como as que encontramos no HTML e cada componente por sua vez tem um comportamento e funcionalidades específicas definidas pelas propriedades e estados que o componente pode assumir, cada componente do React é responsável por gerar elementos HTML na página visualizada, desta forma podemos atualizar apenas os componentes que tiveram alteração em seu estado (Kenzie, 2022).[8]

[6] <https://github.com/>
[7] <https://git-scm.com/>
[8] <https://react.dev/>

2.3.4 Node JS

O Node.js é definido como um ambiente de execução javascript serverside. Que faz com que aplicações javascript sejam criadas para rodar como uma aplicação standalone em uma máquina, não dependendo de um browser para execução. (ALURA, 2022) Com o Node.js foi possível diminuir partes da integração da aplicação, como por exemplo a API (Application Programming Interface), pois ela é feita direta pelo Node.js e consumida pelo front-end. O uso de uma mesma linguagem para toda a aplicação também foi uma vantagem bem funcional, sem ter a necessidade de gastar mais tempo aprendendo uma outra linguagem o tempo de desenvolvimento será maior(Alura, 2022).[9]

2.3.5 Node.js Express

Node.js Express é um framework descrito como uma biblioteca alternativa para uma série de outros frameworks do Node.js. Com o Node.js Express abrese a possibilidade de se usar diferentes verbos HTTP, como por exemplo GET, POST e DELETE, dentro da sua aplicação, o que não seria possível apenas com o Node.js. Além de gerenciar requisições de diferentes verbos HTTP com Express também é possível fazer uma integração com "view engines"para inserir dados nos templates, definir portas a serem usadas para conexão e a localização dos modelos que são usados para renderizar respostas e adicionar "middleware"para novos processos de requisições. Dentro do Express é possível usar outros frameworks, como o bcrypt. (MDN WEB DOCS, 2022) Pelo fato de se poder usar verbos HTTP dentro da aplicação, facilita muita o uso do framework. Devido a esse fato, pode-se gerenciar diferentes requisições URLs, as rotas (MDN Web Docs, 2022).

2.3.6 MongoDB

O MongoDB é um banco de dados NoSQL projetado para oferecer escalabilidade e flexibilidade, sendo uma ótima opção para aplicações modernas. Diferente dos bancos de dados relacionais, que usam tabelas e colunas, o MongoDB armazena dados em documentos no formato JSON. Isso permite que desenvolvedores manipulem os dados de maneira mais intuitiva e adaptável (Devmedia, 2022).

A estrutura em JSON possibilita que os dados variem ou evoluam com o tempo, o que é especialmente útil em cenários dinâmicos. Além disso, o MongoDB se destaca pela sua capacidade de escalar horizontalmente, lidando eficientemente com grandes volumes de dados em múltiplos servidores. Essas características tornam o MongoDB uma escolha ideal para aplicações que exigem agilidade e um gerenciamento eficaz de dados. [10]

[9] <https://nodejs.org/docs/latest/api/>
[10] <https://www.mongodb.com/pt-br>

2.4 Arquitetura da Aplicação

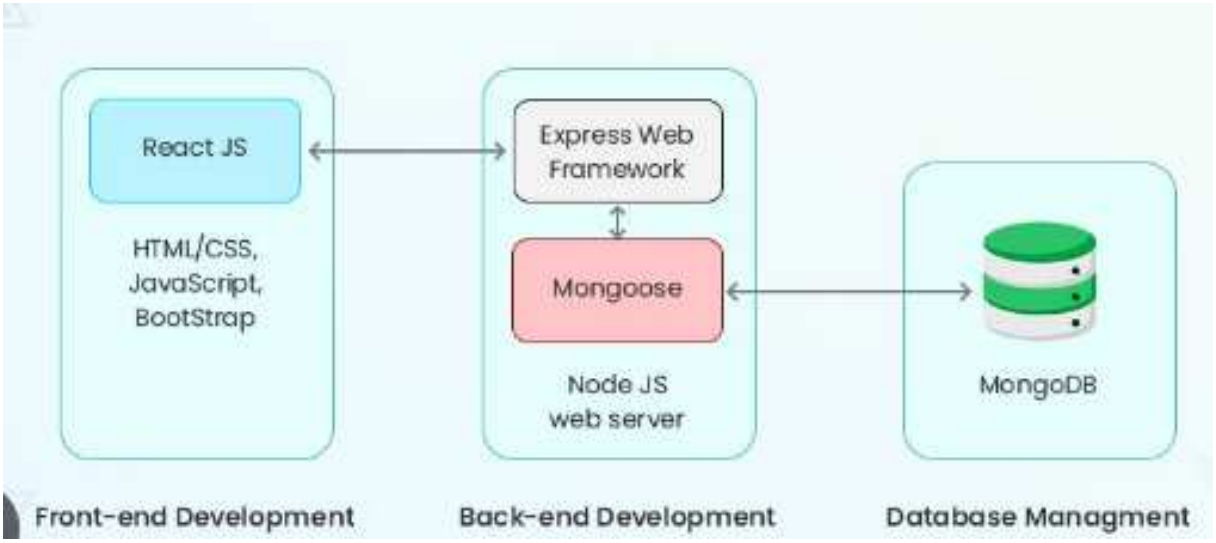
A aplicação em questão segue o modelo cliente-servidor, dividida em duas partes que desempenham funções específicas. No backend, utilizamos Node.js, que é responsável por gerenciar a camada de dados e fornecer informações para o frontend, que é construído em React.

Nesse modelo de arquitetura, conhecido como Cliente-Servidor, um programa (o cliente) faz solicitações a outro programa (o servidor) em busca de serviços. Essa separação traz vários benefícios, permitindo que o servidor funcione como uma entidade com uma API que disponibiliza todos os nossos dados.

Do lado esquerdo, temos o cliente, que é o frontend em React. Ele inicia as solicitações, representadas pelo número 1 (make request) na figura. Essas solicitações são enviadas para o servidor, que foi desenvolvido com Node.js. O servidor analisa o pedido e o encaminha ao Express, que gerencia as requisições diretas ao banco de dados. Se a solicitação estiver correta, o Express recupera os dados necessários do banco.

Por fim, o servidor devolve essa informação ao frontend, que então apresenta os dados na tela para o usuário. Essa interação entre cliente e servidor torna a aplicação mais organizada e eficiente.

Figura 5 – Sistema de Controle de Versão Distribuído



3 Metodologia da Pesquisa

Neste trabalho, desenvolvemos um sistema de gerenciamento de documentos para o LabMet, com o objetivo de otimizar a organização, acesso e compartilhamento de informações essenciais para a equipe. A metodologia adotada visa facilitar o fluxo de trabalho, reduzindo o tempo de busca por documentos e aumentando a eficiência nas atividades diárias.

Para a construção das interfaces, foi usado de inspiração outras plataformas que já se destacam pela usabilidade e design amigável, assegurando uma experiência do usuário agradável. Foi usada a linguagem de programação Typescript, junto com a plataforma Node.js para o backend, que proporciona um ambiente leve e eficiente. Para o frontend, foi escolhido o React, que permite criar componentes reutilizáveis e responsivos, garantindo que a interface se adapte a diferentes dispositivos. Além disso, foi utilizado ferramentas de gerenciamento de dados, como MongoDB, para garantir que todas as informações sejam armazenadas de maneira segura e acessível.

Nas próximas seções, detalharemos as metodologias específicas utilizadas na construção das diferentes partes do sistema, destacando como cada uma contribui para o gerenciamento eficiente dos documentos no LabMet.

3.1 Requisitos

Para construir este projeto foi necessário definir algumas funcionalidades essenciais para a consistência da aplicação. Essas definições estão apresentadas abaixo, começando pelos requisitos funcionais, que ditam o que a aplicação deve fazer, enquanto os requisitos não funcionais dão suporte aos requisitos funcionais informando como o sistema deve realizar tais funções. Nas tabelas abaixo temos a apresentação desses requisitos com seus identificadores e uma descrição sobre cada um deles.

3.1.1 Requisitos funcionais

Tabela 2 – Requisitos Funcionais do Sistema

Identificador	**Descrição**
RF01	O sistema deve permitir que os usuários sejam autenticados informando login e senha.
RF02	O sistema deve permitir o cadastro de diferentes tipos de usuários: administrador,técnico e coordenador.
RF03	O sistema deve permitir que apenas os usuário com privilégio "coordenador"possa fazer o cadastro de novos usuários no sistema
RF04	O sistema deve permitir que os usuarios administradores possam visualizar,editar e cadastrar arquivos administrativos
RF05	O sistema deve permitir que os usuarios técnicos possam visualizar,editar e cadastrar arquivos técnicos
RF06	O sistema deve permitir que o coordenador possa liberar acesso para qualquer usuário visualizar, editar ou cadastrar os arquivos,independente da sua função.
RF07	O sistema deve permitir que usuários pesquisem os arquivos
RF08	O sistema deve permitir que os usuários visualizem os arquivos cadastrados

3.1.2 Requisitos não funcionais

Tabela 3 – Requisitos Não Funcionais do Sistema

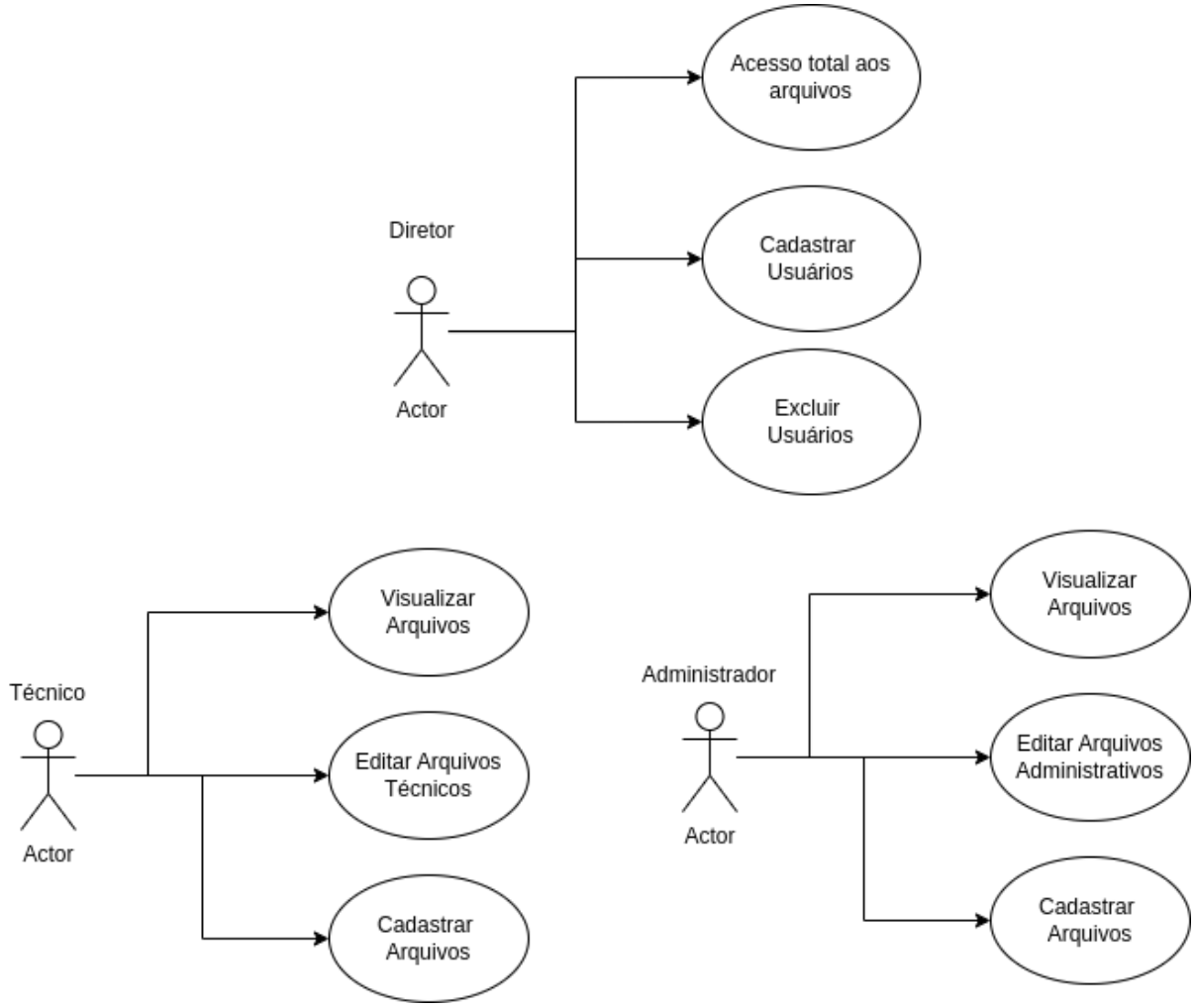
Identificador	**Descrição**
RF01	O sistema deve ser construído utilizando uma linguagem de fácil manutenção.
RF02	O usuário deve ser capaz de utilizar as principais atividades do sistema em, no máximo, 3 minutos.
RF03	O sistema deve ser feito para web.
RF04	As pesquisas dos usuários devem ser retornadas em menos de 10 segundos.
RF05	O sistema deve dar um feedback claro à respeito dos erros que eventualmente ocorrerem.
RF06	Os módulos que compõem o sistema devem estar bem separados.

3.1.3 Modelagem do sistema

Diagramas de Casos de Uso

Além da disposição dos elementos que compõem o sistema, para o seu desenvolvimento foram desenhados alguns cenários utilizando diagramas de caso de uso. O primeiro deles Representa as funções de cada usuario no sistema.

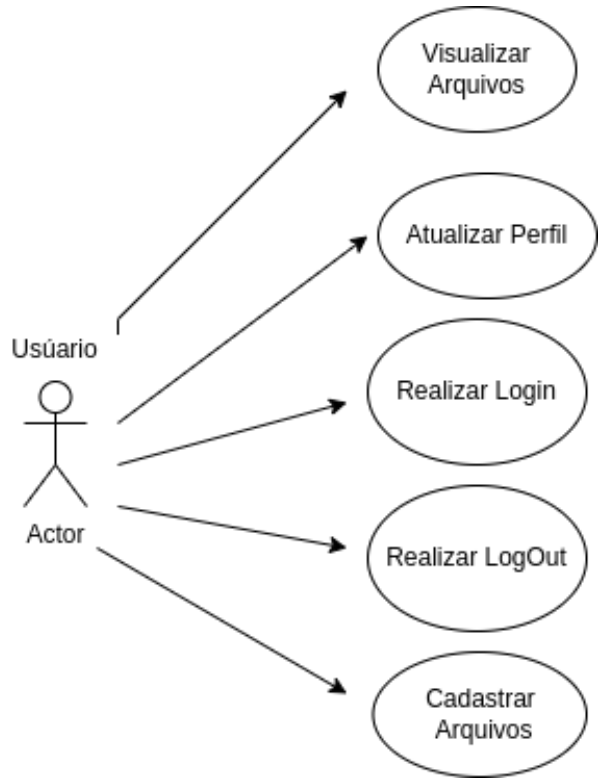
Figura 6 – Diagrama de caso de uso



Fonte: Autoria Própria.

O segundo diagrama mostra as ações que o usário pode realizar por meio do sistema.

Figura 7 – Diagrama de caso de uso

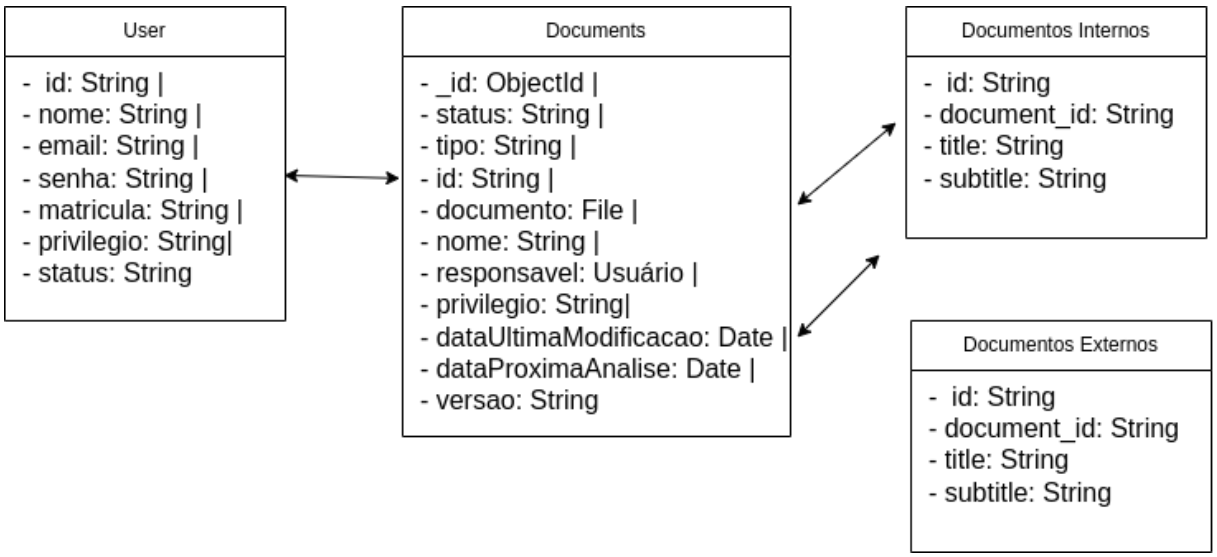


Fonte: Autoria Própria.

Diagrama de classes

As classes do backend do sistema foram criadas de acordo com a necessidade no decorrer do desenvolvimento do código.

Figura 8 – Diagrama de classes



Fonte: Autoria Própria.

4 Resultados e discussões

Neste capítulo serão apresentados os principais resultados obtidos com o desenvolvimento do protótipo do sistema.

4.1 Levantamento de Requisitos

A fase inicial do projeto envolveu um levantamento de requisitos realizado por meio de reuniões com os funcionários do laboratório. Essas conversas, apesar de serem informais, foram fundamentais para obter uma compreensão inicial das necessidades e expectativas dos usuários. Durante esses encontros, foi possível identificar aspectos-chave relacionados à usabilidade, funcionalidade e experiência geral de usuário desejada para o sistema.

Apesar da amostra limitada, as informações coletadas forneceram insights valiosos para a concepção do projeto. As discussões focaram em identificar problemas comuns enfrentados pelos usuários na gestão atual com o google drive.As percepções obtidas dessas conversas foram cuidadosamente documentadas e analisadas, formando a base para o desenvolvimento das especificações iniciais do sistema. Esta abordagem direcionada e focada em usuários reais assegurou que o projeto começasse com uma sólida compreensão das necessidades do mercado, preparando o terreno para as fases subsequentes de desenvolvimento.

4.2 Visão Geral do Sistema

No desenvolvimento do sistema de gerenciamento de documentos, optou-se por ferramentas principais como MongoDB Atlas e Visual Studio Code. O MongoDB Atlas foi essencial para o armazenamento e gestão eficiente dos dados, garantindo segurança e integridade das informações.

Para a edição de código, escolhemos o Visual Studio Code devido ao seu ambiente rico em recursos, que inclui depuração, controle de versão integrado e uma ampla gama de extensões. Sua flexibilidade e interface personalizável tornam o desenvolvimento mais eficiente e adaptado às necessidades dos desenvolvedores.

No backend, utilizamos Node.js, uma plataforma leve e orientada a eventos, ideal para aplicações que requerem um alto volume de operações, como o nosso sistema de gerenciamento de documentos. O Node.js, aliado ao vasto ecossistema de módulos disponíveis via npm, aumentou nossa produtividade. A linguagem Typescript foi escolhida por sua versatilidade, escalabilidade e manutenção a longo prazo

No frontend,foi ultilizado o framework React por proporcionar uma interface modular e reutilizável, otimizando o desempenho com atualizações eficientes, e facilitando a colaboração, além de ter uma grande comunidade e ecossistema de suporte,juntamente com o tailwind que otimizou a construção da estilização das páginas e o shadcn que facilitou a elaboração dos componentes.

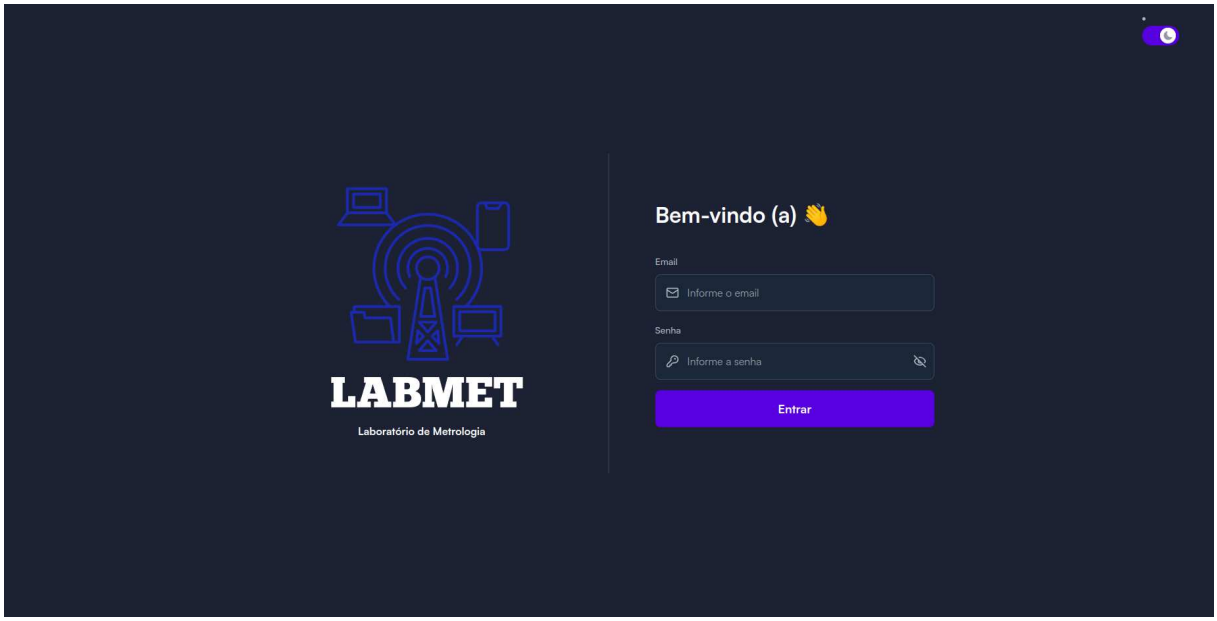
4.3 Análise das páginas funcionais

Nas próximas seções serão apresentadas as páginas do protótipo desenvolvido.

4.3.1 Login

A página de login é fundamental para a segurança e acessibilidade do sistema, garantindo que apenas usuários autorizados possam acessar o sistema fazendo uma conferência com o banco de dados para verificar se os dados estão cadastrados.

Figura 9 – Tela de Login dark mode



Fonte: Autoria Própria

Figura 10 – Tela de Login white mode

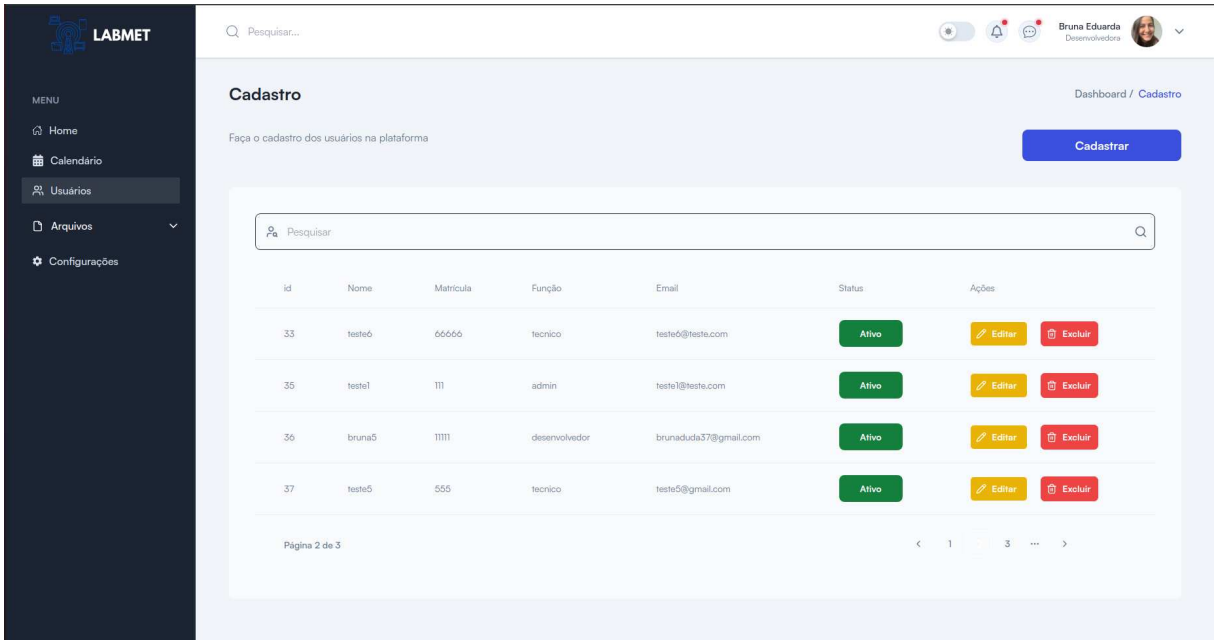
Fonte: Autoria Própria

Após realizar o login, a tela a ser exibida dependerá do tipo de usuário que está acessando o sistema. No caso deste roteiro, o usuário é o coordenador e, portanto, ele tem acesso a uma tela a mais que os demais que é a de cadastro.

4.3.2 Cadastro

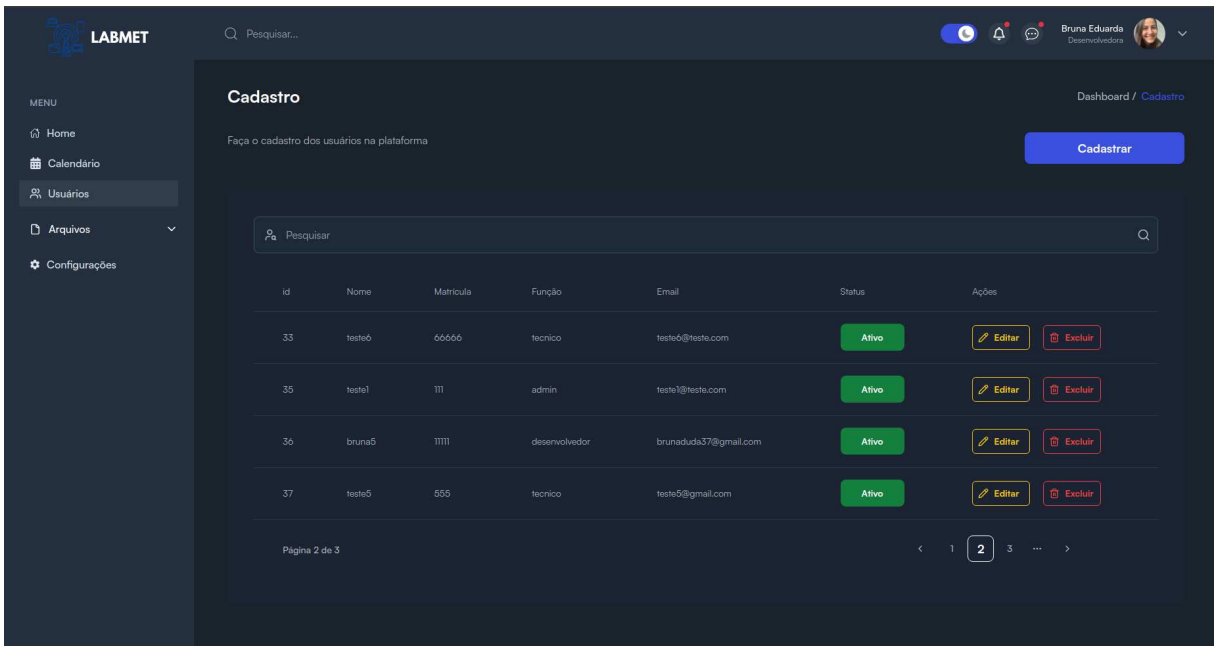
A página de cadastro é essencial para permitir a inclusão de novos usuários no sistema. Ela coleta e valida as informações necessárias para criar uma nova conta, garantindo que os dados sejam registrados de forma segura no banco de dados. Um cadastro eficiente e intuitivo contribui para uma melhor experiência do usuário, facilitando o acesso ao sistema desde o início. A página só esta disponível para o coordenadores realizarem a inclusão de usuários ao sistema.

Figura 11 – Tela de Cadastro white mode



Fonte: Autoria Própria

Figura 12 – Tela de Cadastro dark mode



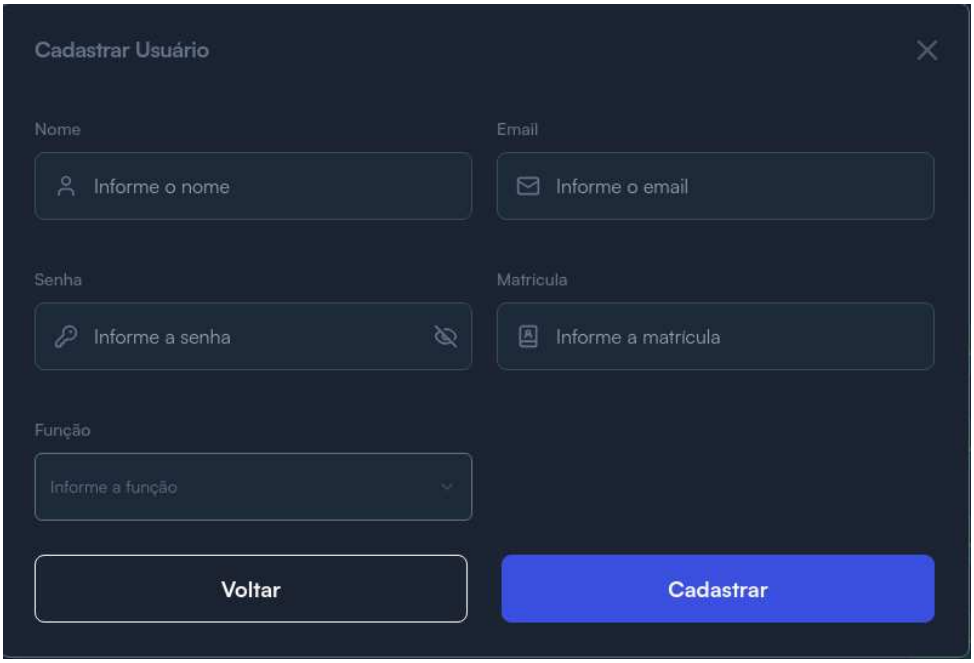
Fonte: Autoria Própria

4.3.3 Modal Cadastro

Para criar um novo usuário,deve-se clicar no botão "Cadastrar". Após o clique uma modal surgirá na tela solicitando algumas informações a respeito do usuário a ser

cadastrado.

Figura 13 – Modal de Cadastro de usuários

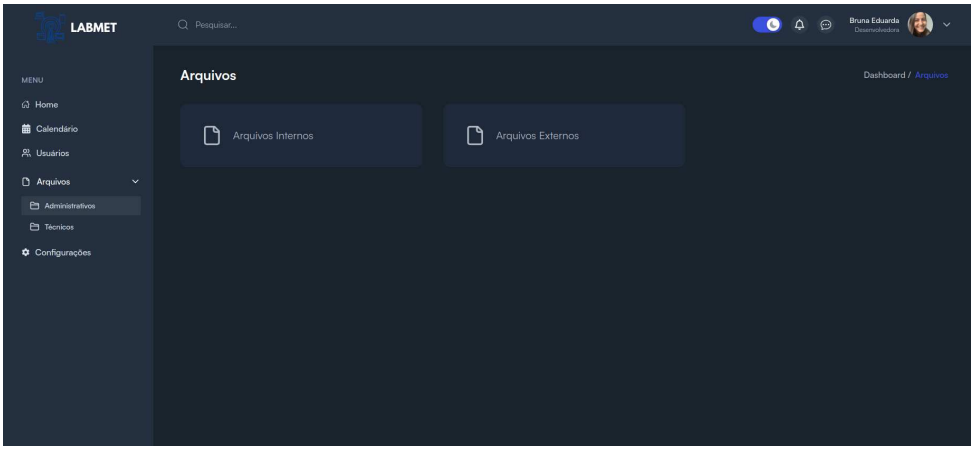


Fonte: Autoria Própria

4.3.4 Tela de Arquivos Geral

A tela geral de arquivos apresenta uma visão estruturada e intuitiva dos documentos, permitindo que tanto usuários administrativos quanto técnicos acessem facilmente arquivos internos e externos.

Figura 14 – Tela de Arquivos

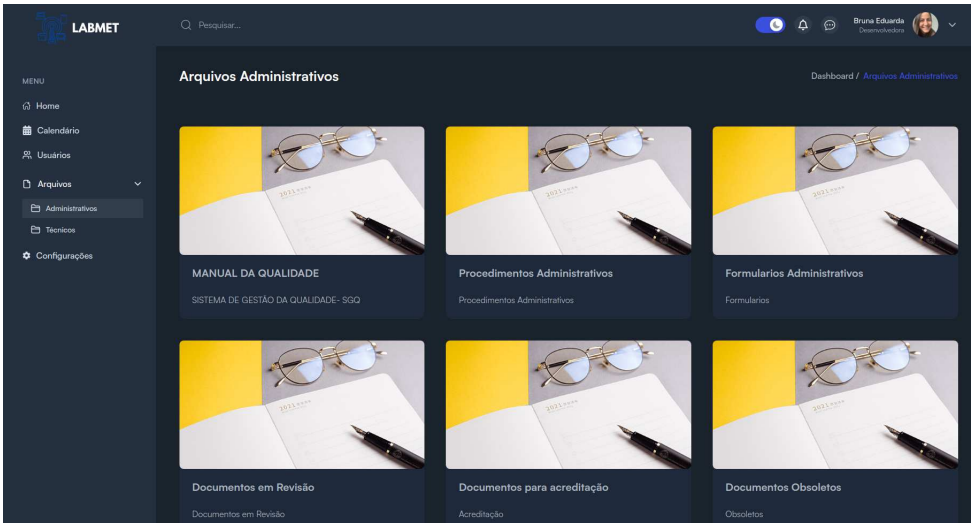


Fonte: Autoria Própria

4.3.5 Tela de Arquivos Internos

A tela de arquivos internos é organizada em pastas, proporcionando uma estrutura clara para a gestão dos documentos essenciais do laboratório. Os usuários podem navegar facilmente entre as pastas para acessar protocolos, relatórios e procedimentos operacionais padrão. Além disso, essa interface permite o cadastro de novas pastas, oferecendo flexibilidade para adaptar a organização dos arquivos conforme as necessidades do laboratório evoluem. Essa funcionalidade facilita a categorização e o armazenamento de informações, garantindo que os usuários possam manter um ambiente de trabalho sempre bem organizado e acessível. OBS: As imagens serão substituídas!

Figura 15 – Tela de Arquivos Internos



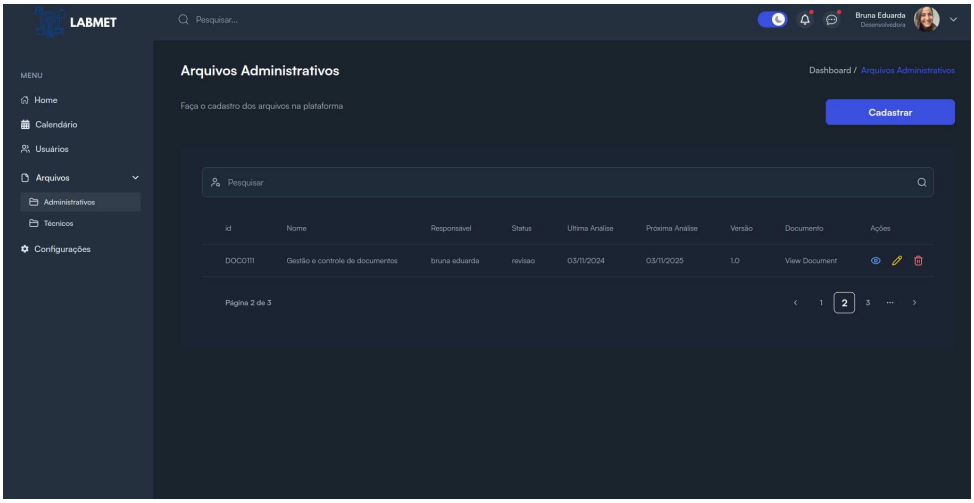
Fonte: Autoria Própria

4.3.6 Tela de Lista de Arquivos

A tela de listagem de arquivos é uma ferramenta essencial para a gestão eficiente dos documentos do laboratório, proporcionando uma visão clara e organizada de todos os arquivos disponíveis. Nela, os usuários podem visualizar os arquivos agrupados por seção ou ID, o que facilita a localização e o acesso rápido às informações necessárias.

A interface permite que os usuários filtrem e classifiquem os arquivos conforme suas necessidades, garantindo que a pesquisa por documentos específicos seja ágil e prática. Além disso, essa tela oferece funcionalidades como a busca por palavras-chave e a visualização de detalhes dos arquivos, como data de criação e autor, promovendo um controle mais rigoroso sobre a documentação.

Figura 16 – Tela de Lista de Arquivos



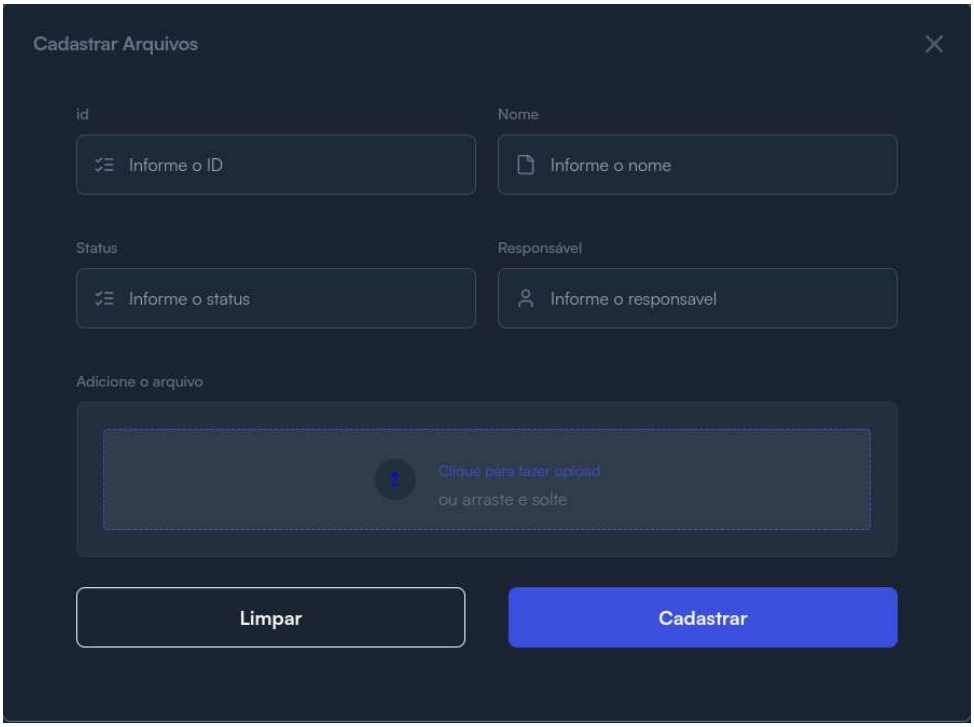
Fonte: Autoria Própria

4.3.7 Tela Modal de Cadastro de Arquivos

A tela modal de cadastro de arquivos é uma interface projetada para facilitar o registro de novos documentos no sistema. Nela, os usuários podem inserir informações essenciais, como nome do arquivo, descrição, categoria e tags, garantindo que o cadastro seja completo e organizado.

A modal oferece campos intuitivos e opções de validação, evitando erros e promovendo a consistência dos dados. Além disso, permite o upload de arquivos diretamente, simplificando o processo de adição de novos documentos ao sistema.

Figura 17 – Tela Modal de Cadastro de Arquivos

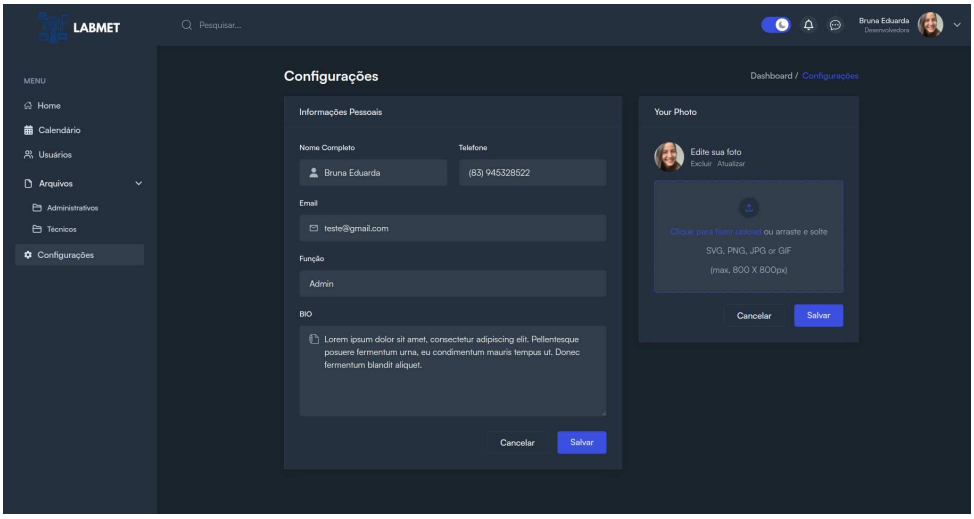


Fonte: Autoria Própria

4.3.8 Tela de Configurações

A tela de configurações de perfil permite aos usuários atualizar informações pessoais, como nome e e-mail e outros. A interface intuitiva garante fácil navegação, personalizando a experiência no sistema de acordo com as necessidades individuais.

Figura 18 – Tela de Configurações de perfil



Fonte: Autoria Própria

A seguir, algumas telas já produzidas de funcionalidades futuras que seria interessante a implementação.

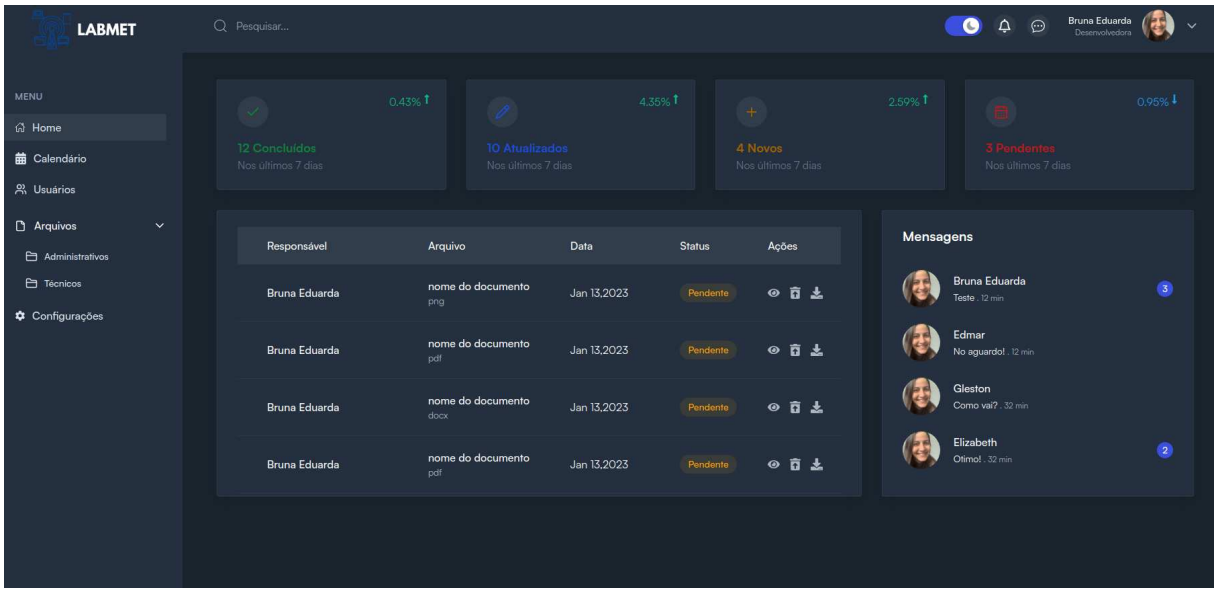
Ao acessar o sistema, o usuário é recebido por uma página inicial intuitiva, que oferece um resumo abrangente das atividades do laboratório. Nesta visão geral, é possível visualizar a situação atual das tarefas em andamento, acompanhando o progresso das atividades de forma clara e objetiva.

Além disso, a página exibe uma lista de documentos que estão se aproximando da data de análise crítica, garantindo que os usuários possam se preparar adequadamente para as revisões necessárias.

Para facilitar a comunicação entre os membros da equipe, há também um espaço dedicado a um chat interno, promovendo interações em tempo real e permitindo que os usuários troquem informações e tirem dúvidas de maneira ágil e eficiente. Essa combinação de recursos torna a página inicial um ponto central para a gestão e colaboração dentro do sistema, proporcionando uma experiência mais integrada e produtiva.

4.3.9 Tela Home

Figura 19 – Tela Home

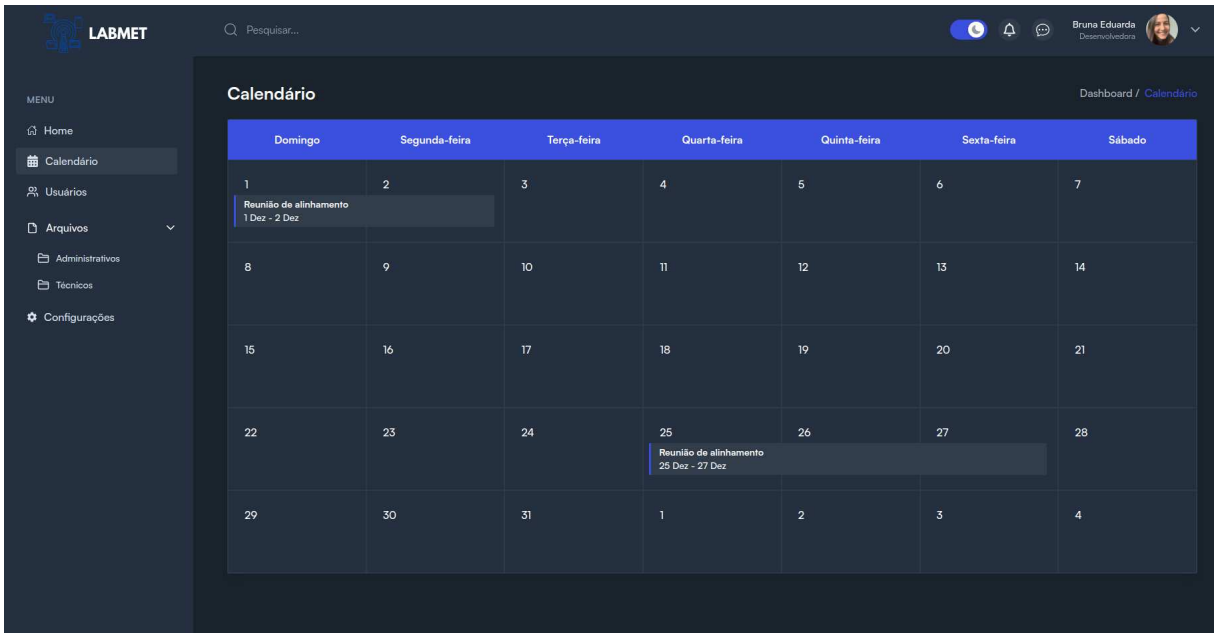


Fonte: Autoria Própria

A tela de calendário é uma ferramenta essencial para a organização das tarefas da equipe. Nela, os usuários podem visualizar compromissos e prazos importantes, além de adicionar eventos e atribuições. A interface permite fácil navegação e colaboração, ajudando a equipe a manter o foco e a produtividade em suas atividades diárias.

4.3.10 Tela de Calendário

Figura 20 – Tela de Calendário



Fonte: Autoria Própria

5 Considerações Finais

Neste capítulo, apresentamos as considerações finais sobre o desenvolvimento deste trabalho de conclusão de curso.

5.0.1 Conclusões

O desenvolvimento do sistema de gerenciamento de documentos para o Laboratório de Metrologia (LABMET) é um início de trabalho visando a transferencia de seus documentos presentes no google drive para o um sistema totalmente seu, onde a parte principal é a possibilidade de personalizar da maneira que o cliente desejar. Com a implementação de funcionalidades como a autenticação de usuários e o cadastro de novos membros, garantimos que apenas pessoas autorizadas tenham acesso a informações sensíveis. Isso é essencial para proteger o acervo contra acessos não autorizados. Além disso, a inclusão de um sistema de alertas para documentos que estão próximos da data de revisão é uma ferramenta valiosa. Essa funcionalidade assegura que a equipe mantenha a qualidade e a atualidade das informações, evitando perdas de prazos importantes.

Essa mudança se apresenta como uma solução valiosa para as necessidades de conservação do acervo, mantendo a ordem, o controle de arquivos e otimizando o acesso aos documentos. Com isso, o processo de busca se torna mais ágil e intuitivo.Essa transformação não apenas facilita o dia a dia no LABMET, mas também permite que a equipe se concentre no que realmente importa: a inovação e a excelência nas suas pesquisas. Em um ambiente onde o conhecimento é fundamental, ter um sistema robusto e confiável faz toda a diferença para o sucesso do laboratório.

5.0.2 Perspectivas Futuras

Olhando para o futuro, está previsto que o sistema de gerenciamento de documentos seja disponibilizado em produção na AWS, após a realização de testes rigorosos para garantir sua estabilidade e segurança. Essa migração para um ambiente de nuvem proporcionará maior escalabilidade e acessibilidade, permitindo que os usuários acessem o sistema de qualquer lugar e a qualquer momento.

Além disso, há planos para incluir novas funcionalidades que visam aprimorar ainda mais a experiência do usuário. Entre essas melhorias, destaca-se a implementação de um sistema de Kanban e um calendário para organização da equipe. O sistema de Kanban permitirá que os membros do LABMET gerenciem tarefas e projetos de forma visual e intuitiva, facilitando o acompanhamento do progresso e a priorização das atividades. Já o

calendário integrado ajudará na coordenação de prazos, reuniões e eventos, promovendo uma melhor organização e colaboração entre os membros da equipe.

Outras funcionalidades planejadas incluem a geração de relatórios gerenciais e a integração com outras ferramentas já utilizadas pelo laboratório. Essas adições não apenas expandirão as capacidades do sistema, mas também garantirão que ele atenda de forma mais abrangente às necessidades do LABMET, promovendo uma gestão mais eficiente e colaborativa.

Com essas melhorias, o objetivo é transformar o sistema em uma solução completa que apoie a equipe em sua busca pela excelência em pesquisa, otimizando processos e facilitando a comunicação e a organização no dia a dia do laboratório.

5.0.3 Experiência Adquirida

Desenvolver uma aplicação web completa para um cliente real foi uma jornada extremamente enriquecedora e transformadora. Desde o início, a elaboração dos requisitos apresentou desafios que me permitiram crescer tanto profissional quanto pessoalmente. Trabalhar diretamente com o cliente não exigiu apenas compreensão técnica, mas também habilidades interpessoais essenciais. Aprendi a importância de ouvir atentamente, formular perguntas relevantes e traduzir as necessidades do cliente em especificações claras e acionáveis, garantindo que o projeto estivesse alinhado com suas expectativas.

O design e o desenvolvimento do frontend foram etapas especialmente gratificantes. Criar interfaces intuitivas e visualmente atraentes me proporcionou a oportunidade de explorar e aplicar princípios de usabilidade e design. Utilizando ferramentas como protótipos e wireframes,como o figma, pude validar minhas ideias e ajustar a experiência do usuário, assegurando que o sistema fosse não apenas funcional, mas também agradável de usar.

No backend, enfrentei novos desafios, como a construção de APIs e a gestão de dados. A integração entre o que havia desenvolvido no frontend e o processamento no servidor revelou a complexidade e a beleza da conexão entre os sistemas front e back. Essa experiência me ajudou a compreender a importância de um fluxo de informações eficiente e seguro.

Atualmente, estamos na fase de testes, que é crucial para assegurar que tudo funcione conforme o planejado. Embora ainda esteja em andamento, essa etapa já tem sido reveladora. A realização de testes de usabilidade e a coleta de feedback dos usuários finais serão essenciais para identificar áreas de melhoria e garantir que o sistema atenda plenamente às necessidades do cliente.

Em resumo, o desenvolvimento desse sistema de gerenciamento de documentos não apenas ampliou meu conhecimento técnico, mas também fortaleceu minhas habilidades de trabalho em equipe e gestão de projetos. Cada desafio encontrado ao longo do caminho

foi uma oportunidade de crescimento, preparando-me melhor para enfrentar os desafios futuros na área de tecnologia. Essa experiência prática foi, sem dúvida, um dos marcos significativos da minha formação acadêmica, moldando meu entendimento sobre o papel da tecnologia em resolver problemas reais.

Referências Bibliográficas

Alura. *Node.JS: definição, características, vantagens e usos possíveis*. 2022. Acesso em: 28 de out. de 2024. Disponível em: <https://www.alura.com.br/artigos/node-js-definicaocaracteristicas-vantagens-usos>. Citado na página 13.

CODE visual studio. visual studio code. Acesso em: 09 de out. de 2024. 2024. Disponível em: <https://code.visualstudio.com/docs>. Citado na página 11.

Devmedia. *Introdução ao Mongodb*. 2022. Acesso em: 28 de out. de 2024. Disponível em: <https://www.devmedia.com.br/introducao-ao-mongodb/30792>. Citado na página 13.

FALBO, R. de A. *Engenharia de Requisitos - Notas de Aula*. 2017. [s.n.]. Disponível em: <http://www.inf.ufes.br/~vitorsouza/falbo/Notas_Aula_Engenharia_Requisitos_Falbo_2017.pdf>. Citado 2 vezes nas páginas 5 e 6.

Howie Mann Startup Strategy. *REST API Basics - 4 Things you Need to Know*. 2023. Acesso em: 28 de out. de 2024. Disponível em: <https://mannhowie.com/rest-api>. Citado na página 9.

INSOMINIA. Design, debug, test, and mock apis locally, on git, or cloud. 2024. Acesso em: 20 de out. de 2024. Disponível em: <https://insomnia.rest/>. Citado na página 7.

JACOBSON, I. *Object Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1992. Acesso em: 28 de out. de 2024. Disponível em: <http://www.amazon.com/Object-Oriented-Software-Engineering-Approach/dp/0201544350>. Citado na página 5.

Kenzie. *React: o que é, como funciona e porque usar e como aprender*. 2022. Acesso em: 28 de out. de 2024. Disponível em: <https://kenzie.com.br/blog/react/>. Citado na página 12.

MACORATTI, J. C. *Padrões de Projeto : O modelo MVC - Model View Controller*. [s.n.], 2019. Acesso em: 28 de out. de 2024. Disponível em: <https://www.macoratti.net/vbn_mvc.htm>. Citado na página 6.

MASTERS, F. What is a front-end developer? 2021. Acesso em: 20 de out. de 2024. Disponível em: <https://frontendmasters.com/guides/front-end-handbook/2018/what-is-a-FD.html>. Citado na página 8.

MDN Web Docs. *Introdução Express/Node*. 2022. Acesso em: 28 de out. de 2024. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Learn/Serverside/Express_Nodejs/Introduction>. Citado na página 13.

NETWORK, M. D. Uma visão geral do http. 2021. Acesso em: 23 de out. de 2024. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>. Citado na página 6.

SOMMERVILLE, I. *Engenharia de Software*. Pearson Prentice Hall, 2011. Acesso em: 28 de out. de 2024. ISBN 9788579361081. Disponível em: <https://books.google.com.br/books?id=H4u5ygAACAAJ>. Citado na página 5.