



UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE SOBRAL
DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

JOSÉ CARLOS SILVA GADELHA

WEB CEUA - DESENVOLVIMENTO DA API DO SISTEMA WEB CEUA

SOBRAL

2023

JOSÉ CARLOS SILVA GADELHA

WEB CEUA - DESENVOLVIMENTO DA API DO SISTEMA WEB CEUA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Campus de Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Orientador: Prof. Dr. Iális Cavalcante de Paula Júnior.

SOBRAL

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

G12w Gadelha, Jose Carlos Silva.
Web Ceua - Desenvolvimento da API do sistema Web Ceua / Jose Carlos Silva Gadelha. – 2023.
37 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral,
Curso de Engenharia da Computação, Sobral, 2023.
Orientação: Prof. Dr. Prof. Dr. Iális Cavalcante de Paula Júnior.

1. Ceua. 2. Express. 3. Projetos. 4. API. I. Título.

CDD 621.39

JOSÉ CARLOS SILVA GADELHA

WEB CEUA - DESENVOLVIMENTO DA API DO SISTEMA WEB CEUA

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia da Computação do Campus de Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia da Computação.

Aprovada em: 14/12/2023.

BANCA EXAMINADORA

Prof. Dr. Iális Cavalcante de Paula
Júnior (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Antônio Josefran de Oliveira Bastos
Universidade Federal do Ceará (UFC)

Tecg^o. Iago Magalhães de Mesquita
Instituto Federal de Educação, Ciência e Tecnologia
do Ceará (IFCE)

Carlos Gadelha, lembre-se você conseguiu continue acreditando e se priorizando. A minha família e em especial a Deus

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, minha base, meu tudo.

Também desejo agradecer à minha família, que sempre me apoiou na medida do possível. Desde cedo, percebi que, por meio da educação, poderia proporcionar uma vida melhor tanto para mim quanto para meus pais.

Hoje, sou grato pelas escolhas que fiz durante a graduação; optei por concluir o curso, mesmo diante das diversas dificuldades, e estou colhendo os frutos agora.

Agradeço a toda a universidade, em especial aos professores que contribuíram para o meu crescimento profissional.

Contudo, gostaria de expressar meu agradecimento especial ao **Prof. Dr. Iális Cavalcante de Paula Júnior** por sua orientação e paciência ao longo deste trabalho.

“Todos os dias somos apresentados a duas escolhas, mudar ou repetir”
(autor desconhecido)

RESUMO

A digitalização dos processos da *Comissão de Ética no Uso de Animais* (CEUA) em Fortaleza é de extrema necessidade para aprimorar a eficiência, facilitar a tomada de decisões e impulsionar o crescimento. Atualmente, toda a gestão dos projetos é conduzida manualmente, utilizando *e-mails*, formulários e planilhas em cada etapa, desde o cadastro até o controle das atividades. Diante do aumento contínuo no número de projetos, essa abordagem torna-se cada vez mais complexa. Frente a esse desafio, a CEUA optou por implementar um sistema que simplificasse a submissão de projetos e o gerenciamento. Neste contexto, desenvolveu-se a API do sistema webCeua, oferecendo diferentes níveis de acesso, como pesquisador, parecerista e administrador. Essa diferenciação de acesso visa proporcionar uma experiência personalizada, permitindo que cada usuário desempenhe seu papel de maneira eficaz. Além disso, a API possibilita o acompanhamento do histórico entre as diversas versões de um projeto, proporcionando transparência e facilitando as consultas. Ao integrar essas características, a API contribui significativamente para a criação de uma aplicação de fácil utilização e intuitiva, alinhada com os objetivos da CEUA de Fortaleza para modernizar e otimizar seus processos.

Palavras-chave: API; Express; Projetos; Ceua

ABSTRACT

The digitalization of the processes of the Ethics Committee on the Use of Animals (CEUA) in Fortaleza is extremely necessary to improve efficiency, facilitate decision-making and boost growth. Currently, all project management is conducted manually, using emails, forms and spreadsheets at each stage, from registration to activity control. Given the continuous increase in the number of projects, this approach becomes increasingly complex. Faced with this challenge, CEUA chose to implement a system that facilitates project submission and management. For that reason, the webCeua system *Application Programming Interface* (API) was developed, offering different levels of access, such as researcher, reviewer and administrator. This differentiation of access aims to provide a personalized experience, allowing each user to play their role effectively. Furthermore, the API makes it possible to monitor the history between the different versions of a project, providing transparency and facilitating queries. By integrating these features, the API significantly contributes to the creation of an easy-to-use and intuitive application, aligned with CEUA in Fortaleza's objectives to modernize and optimize its processes.

Keywords: API; Express; Projects; Ceua

LISTA DE ILUSTRAÇÕES

Figura 1 – Arquitetura <i>Model Service Controller</i> (MSC)	16
Figura 2 – <i>Schema</i> do banco de dados utilizando o Prisma ORM	18
Figura 3 – Diagrama do banco de dados	21
Figura 4 – Cadastro de novos usuários pelo <i>insomnia</i>	24
Figura 5 – Cadastro atualizado de novos usuários pelo <i>insomnia</i>	26
Figura 6 – Implementação do módulo de <i>e-mails</i> usando o <i>nodemailer</i>	27
Figura 7 – Tela de Login	28
Figura 8 – Tela seleção dos perfis	29
Figura 9 – Tela Inicial: Projetos do Pesquisador	29
Figura 10 – Tela Inicial: Gerenciamento de Projetos	30
Figura 11 – Menu do projeto em listagem de projetos perfil da coordenação	30
Figura 12 – Selecionar Parecerista	31
Figura 13 – Tela gerenciamento de usuários	32
Figura 14 – Menu gerenciar usuário	32
Figura 15 – Criação de novos usuários	33
Figura 16 – <i>E-mail</i> enviado após o usuário ser criado	33
Figura 17 – Listagem de Projeto Parecerista	34
Figura 18 – Parecer	34

LISTA DE ABREVIATURAS E SIGLAS

API	<i>Application Programming Interface</i>
CEUA	<i>Comissão de Ética no Uso de Animais</i>
HTML	Linguagem de Marcação de Hipertexto (do inglês: <i>HyperText Markup Language</i>)
MSC	<i>Model Service Controller</i>
MVP	<i>Minimum viable product</i>
ORM	<i>Object Relational Mapper</i>
PDF	<i>Portable Document Format</i>
SQL	Linguagem de Consulta Estruturada(do inglês: <i>Structured Query Language</i>)
UFC	<i>Universidade Federal do Ceará</i>

SUMÁRIO

1	INTRODUÇÃO	12
1.1	Justificativa	13
1.2	Objetivos	13
1.2.1	<i>Objetivo Geral</i>	13
1.2.2	<i>Objetivos Específicos</i>	13
2	FUNDAMENTAÇÃO TEÓRICA	15
2.1	Pesquisadores	15
2.2	Coordenação	15
2.3	Pareceristas	15
2.4	Arquitetura MSC	16
2.5	<i>JavaScript</i>	16
2.6	Node	17
2.7	Express	17
2.8	Bancos de Dados e Prisma ORM	17
2.9	Docker	17
3	METODOLOGIA	19
3.1	Análise e reconstrução da API	19
3.1.1	<i>Levantamento de requisitos</i>	19
3.1.2	<i>Desenvolvimento da aplicação no back end</i>	20
3.1.2.1	<i>Modelagem do banco de dados</i>	20
3.1.2.2	<i>Rotas da API</i>	22
3.1.2.3	<i>Versões de um projeto.</i>	23
3.1.2.4	<i>Gerenciamento inicial de usuários.</i>	23
3.2	Desenvolvimento de <i>Minimum viable product</i> (MVP) do sistema	24
3.2.1	<i>Perfil do Pesquisador</i>	24
3.2.2	<i>Perfil do Parecerista</i>	25
3.2.3	<i>Perfil da Coordenação</i>	25
3.3	Coleta de feedbacks do MVP do sistema e refatoração	25
3.3.1	<i>Geração de arquivos no formato PDF</i>	25
3.3.2	<i>Gerenciamento de usuários</i>	26

3.3.3	<i>Notificações via e-mail</i>	26
4	RESULTADOS	28
4.1	Acesso ao Sistema	28
4.2	Perfil do Pesquisador	29
4.3	Perfil da Coordenação	30
4.4	Perfil do Parecerista	34
5	CONCLUSÕES E TRABALHOS FUTUROS	35
	REFERÊNCIAS	36

1 INTRODUÇÃO

As Comissões de Ética no Uso de Animais constituem estruturas fundamentais para assegurar o bem-estar dos animais envolvidos em pesquisas científicas. Elas garantem a condução ética dos experimentos, observando princípios e diretrizes que buscam minimizar o sofrimento dos animais, ao mesmo tempo em que promovem avanços significativos na área de pesquisa.

No cenário brasileiro, essas comissões estão distribuídas em diversas instituições de ensino e pesquisa por todo o país. Cada uma opera de acordo com normas e regulamentações específicas, alinhadas aos princípios éticos estabelecidos nacional e internacionalmente. Essa descentralização reflete a importância atribuída à ética na pesquisa com animais, contribuindo para uma abordagem mais abrangente e adaptada às particularidades de cada instituição. Tais comissões desempenham papel crucial em instituições de ensino e pesquisa, sendo responsáveis por avaliar e deliberar sobre a aprovação ou reprovação de projetos de pesquisa que envolvem protocolos experimentais com animais.

Atualmente, a CEUA é composta por pesquisadores, pareceristas e membros administrativos, incluindo a secretaria e a coordenação. Os pesquisadores submetem seus projetos para análise via *e-mail*, utilizando formulários específicos. A coordenação registra esses novos projetos em uma planilha eletrônica e realiza uma triagem de pareceristas com base na área de estudo de cada projeto.

Concluída a triagem, o projeto é encaminhado para análise de um parecerista, que pode sugerir correções ou melhorias. Após a análise do parecerista, o projeto está pronto para ser apreciado na reunião da CEUA. Caso a comissão aceite as modificações propostas pelo parecerista, o projeto retorna à coordenação, que o reenvia ao pesquisador para as devidas correções. O projeto será novamente apreciado na próxima reunião.

Caso o projeto não receba sugestões de alteração ou melhoria e seja aprovado pela comissão durante a reunião, é então emitido o certificado de autorização.

Entretanto, é importante ressaltar que, embora tenha havido uma tentativa inicial de implementar a API, essa iniciativa não foi concluída. Diante disso, está em curso uma nova fase de desenvolvimento, com o objetivo de finalizar e aprimorar a aplicação. Essa segunda abordagem visa atender de maneira mais eficiente às demandas da CEUA da *Universidade Federal do Ceará* (UFC) em Fortaleza. A nova implementação da API pretende oferecer funcionalidades como o cadastro e a edição de projetos, bem como a definição de níveis de

acesso ao sistema, contemplando categorias como Pesquisador, Parecerista e Administrador.

1.1 Justificativa

Atualmente, para o controle tanto dos projetos quanto dos pareceres, utiliza-se a ferramenta de gerenciamento de planilhas da *Google*. No entanto, esse método apresenta limitações e dificuldades de escalabilidade, além do fator humano, pois a planilha, a cada dia, torna-se mais complexa, aumentando a possibilidade de ocorrência de erros.

Conforme mencionado anteriormente, houve uma tentativa inicial de desenvolver uma ferramenta para gerenciar os processos da CEUA, abrangendo desde o cadastro de novos projetos até sua aprovação ou rejeição. Contudo, apenas o *front end*, ou seja, a interface com o usuário, foi desenvolvido, integrada a uma solução da Google chamada *firebase* para o *back end*.

A eventual descontinuidade do *firebase* resultaria na inoperância da aplicação, levando à perda de acesso aos dados. Além disso, deve-se considerar o fator custo, uma vez que é necessário pagar para utilizar a ferramenta. A utilização de um serviço de terceiros a longo prazo não se mostra vantajosa, tanto devido ao custo de uso quanto à limitação de personalização, quando comparada a uma aplicação própria.

Contudo, este trabalho é de suma importância para modernizar os processos da CEUA, facilitando o acesso e gerenciamento de seus procedimentos.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver uma aplicação web, onde os usuários possam interagir, cadastrar novos projetos, acompanhar o andamento dos mesmos ou seja os pareceres. Além de facilitar o gerenciamento dos projetos, histórico e controle de acesso às informações.

1.2.2 Objetivos Específicos

- Reconstruir a API, com as melhorias identificadas durante a análise, utilizando-se de padrões de projeto modernos e vastamente utilizados no mercado.
- Implementar diferentes níveis de acesso tais como pesquisador, parecerista e administrador.
- Possibilitar que o histórico entre as diferentes versões de um projeto esteja disponível para

consulta.

- Implementar a funcionalidade de geração de documentos em formato PDF.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção aborda as tecnologias utilizadas no projeto e os perfis dos usuários. Ela apresenta os conceitos e características empregados no desenvolvimento da aplicação, bem como as ferramentas escolhidas.

2.1 Pesquisadores

Os pesquisadores necessitam da aprovação da CEUA, ou seja, do conselho de ética, para conduzir suas respectivas pesquisas. Sua principal responsabilidade envolve o preenchimento do formulário com os dados relativos à sua pesquisa e submeter à Coordenação.

Em caso de observações ou sugestões de melhoria, cabe aos pesquisadores efetuar as correções no projeto.

2.2 Coordenação

A Coordenação detém a responsabilidade de gerenciar os projetos e selecionar os pareceristas encarregados da análise. Uma de suas principais atribuições inclui a organização das reuniões do colegiado da CEUA, com a escolha dos projetos a serem analisados.

No *software*, a Coordenação adquire uma responsabilidade adicional, que é a gestão dos usuários. Isso implica em adicionar novos usuários, administrar os níveis de acesso e, se necessário, desativar um usuário.

2.3 Pareceristas

Os pareceristas são responsáveis por analisar se os projetos estão em conformidade com as normas éticas vigentes. Possuem autonomia para aceitar ou recusar um projeto, sendo a recusa geralmente motivada por conflitos de interesse por parte do parecerista ou se o projeto está infringindo alguma norma.

No caso da aceitação do projeto, o parecerista dispõe de duas opções: sugerir modificações ou aprovar o projeto.

2.4 Arquitetura MSC

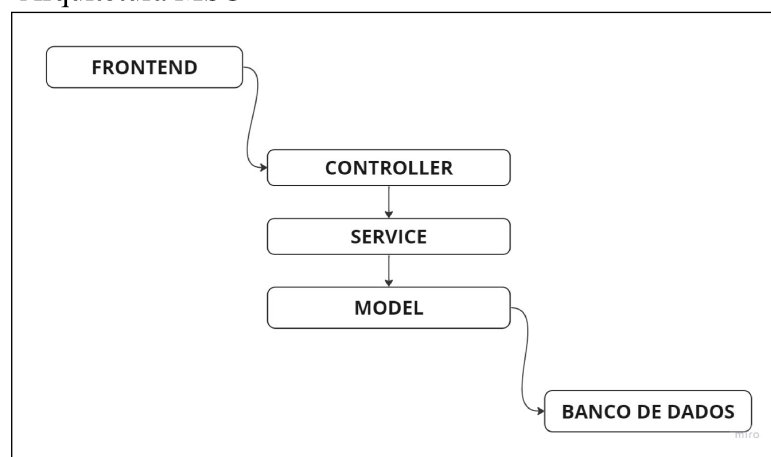
A arquitetura MSC é amplamente empregada na construção de aplicações no *back end*, organizando o código em camadas para facilitar a compreensão do mesmo (MEDIUM, 2023).

A primeira camada consiste no *Controller*, responsável por receber as requisições dos clientes da API e encaminhar as informações para a próxima camada, o *Service*.

A segunda camada, o *Service*, abriga as regras de negócio da aplicação. Após receber os dados da camada anterior, aplica-se as regras de negócio e realizam-se as requisições necessárias na camada *Model*.

A terceira camada, o *Model*, interage com o banco de dados, obtendo e armazenando informações. Neste trabalho, optamos por utilizar um *Object Relational Mapper* (ORM), que será detalhado em uma seção específica dedicada a esse tema.

Figura 1 – Arquitetura MSC



Fonte: O próprio autor.

2.5 JavaScript

JavaScript é uma linguagem de programação leve, estruturada e baseada em funções. Utilizada principalmente na Web para criar páginas dinâmicas, atualmente é amplamente utilizada em ambientes sem o *browser* tais como o Node.js e o Apache (MOZILLA, 2023).

2.6 Node

O Node.js é um software de código aberto, fundamentado no interpretador V8 do *Google*, que viabiliza a execução de códigos *JavaScript* fora do ambiente do navegador (GEEK BLOG, 2023).

2.7 Express

O Express é uma estrutura de aplicativos do Node.js que disponibiliza recursos fundamentais para aplicações web, destacando-se, por exemplo, no desenvolvimento de APIs (EXPRESS, 2023).

2.8 Bancos de Dados e Prisma ORM

Para o gerenciamento do banco de dados, optou-se por empregar um ORM, que essencialmente é uma camada que estabelece uma relação entre objetos e os dados armazenados no banco de dados.

O uso dessa ferramenta possibilita a realização de consultas ao banco sem a necessidade de utilizar Linguagem de Consulta Estruturada (do inglês: *Structured Query Language*) (SQL) puro, simplificando o processo de desenvolvimento. Uma vantagem adicional é que a modelagem fica armazenada no próprio repositório da aplicação, conforme ilustrado na Figura 3.

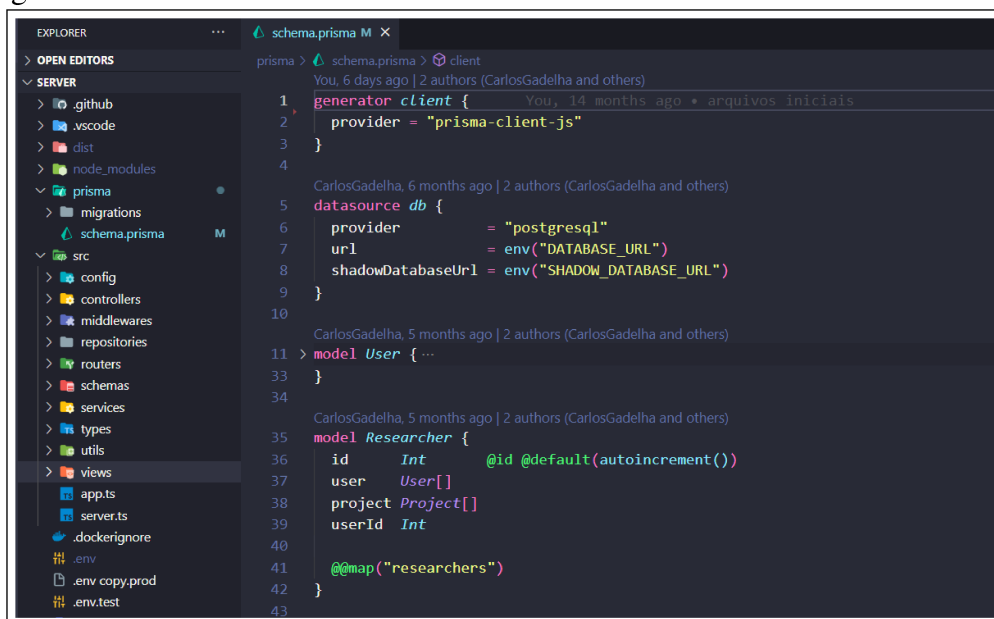
Dentre os diversos ORM disponíveis no mercado, a escolha recaiu sobre o Prisma, um conjunto de ferramentas de código aberto que facilita o acesso e a manipulação de dados no banco de dados (DIGITALOCEAN, 2023).

Nesse contexto, o banco de dados utilizado pode ser facilmente substituído, uma vez que o Prisma ORM gera um *schema* personalizado conforme a escolha do banco. No âmbito deste trabalho, optamos por utilizar o *PostgreSQL*.

2.9 Docker

O Docker é uma plataforma de código aberto que simplifica a criação e administração de ambientes isolados. Essa tecnologia permite o empacotamento de uma aplicação, tornando-a portátil para qualquer outra máquina que tenha o Docker instalado (BLOG TREINAWEB, 2023).

No entanto, para que a aplicação esteja disponível para o usuário final, é necessário

Figura 2 – *Schema* do banco de dados utilizando o Prisma ORM

```
1 generator client {
2   provider = "prisma-client-js"
3 }
4
5
6 datasource db {
7   provider         = "postgresql"
8   url              = env("DATABASE_URL")
9   shadowDatabaseUrl = env("SHADOW_DATABASE_URL")
10 }
11
12 model User {
13   ...
14 }
15
16 model Researcher {
17   id      Int      @id @default(autoincrement())
18   user    User[]
19   project Project[]
20   userId  Int
21
22   @@map("researchers")
23 }
```

Fonte: O próprio autor.

utilizar uma infraestrutura adequada. Entre as opções disponíveis no mercado, o Docker se destaca por facilitar a implantação em ambientes de produção e será empregado nesta aplicação.

3 METODOLOGIA

O presente trabalho tem como objetivo o desenvolvimento da API, que será responsável por gerenciar integralmente as regras de negócios e a comunicação com o banco de dados. Sobre as regras de negócios podemos citar os diferentes níveis de acesso, tais como pesquisador, pareceristas e o perfil administrador.

Este capítulo detalha o processo de desenvolvimento, sendo a metodologia do projeto dividida em três etapas:

- **Análise e reconstrução da API**

- Inicialmente, será realizada uma análise da API já desenvolvida, com o propósito de mapear as funcionalidades existentes, identificar falhas, sugerir melhorias e avaliar a possibilidade de adição de novas funcionalidades.

- **Desenvolvimento de MVP do sistema**

- Será desenvolvida uma API que se integrará ao *front end* já desenvolvido.

- **Coleta de *feedback* do MVP do sistema e refatoração**

- Após a coleta de *feedback* dos usuários sobre a API, serão implementadas as melhorias sugeridas, correções de *bugs* e refatoração do código.

3.1 Análise e reconstrução da API

3.1.1 *Levantamento de requisitos*

O levantamento de requisitos é uma das etapas iniciais no desenvolvimento de um *software*, neste caso os principais requisitos estavam definidos mas não implementados na aplicação.

- **Principais funcionais:**

- Salvar histórico de versões de um projeto.
- Diferentes perfis de acessos na aplicação.
- Gerenciamento dos usuários na tela de coordenação.
- Gerar certificado do projeto após aprovação.

- **Principais não funcionais:**

- Aplicação deve ser intuitiva e de fácil utilização.
- Aplicação deve ser desenvolvida com o máximo de modularidade e o mínimo

de acoplamento, garantindo assim a sua manutenção.

- Aplicação deve ser escalável, ou seja suportar um grande volume de requisições de dados.

3.1.2 Desenvolvimento da aplicação no back end

Nessa subseção será detalhado o desenvolvimento da API, a divisão em etapas e a sua construção lógica para fornecer as soluções necessárias para a aplicação webCeua.

3.1.2.1 Modelagem do banco de dados

Após a análise dos requisitos funcionais, concluiu-se que um banco relacional seria ideal para solucionar a problemática da WebCeua, como dito anteriormente na introdução todo o processo da CEUA é feito através de planilhas.

Com isso em mente definimos as primeiras tabelas do banco de dados, sendo que todos os nomes de variáveis e tabelas na aplicação serão em inglês, padrão amplamente utilizado no mercado.

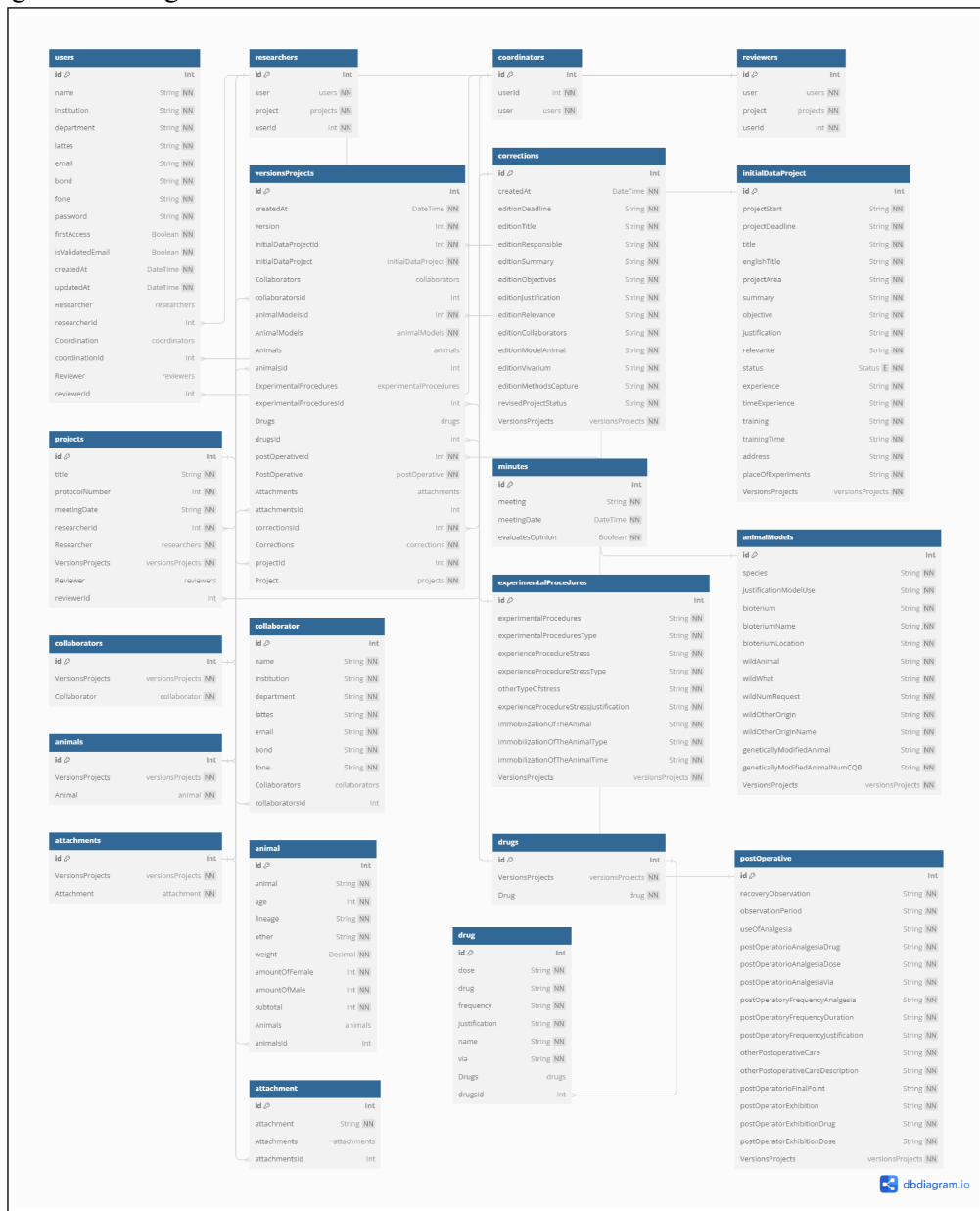
Temos as seguintes tabelas:

- **Users**, dados gerais de todos os usuários.
- **Researcher**, dados dos pesquisadores.
- **Coordination**, dados da coordenação.
- **Reviewer**, dados dos pareceristas.
- **Project**, dados gerais do projeto tais como número do protocolo entre outros.
- **VersionsProjects**, armazena as diferentes versões de um projeto, como as versões do projeto é algo crítico na API terá uma subseção detalhando seu funcionamento.
- **InitialDataProject**, dados iniciais do projeto.
- **Collaborators**, armazena os colaboradores de um projeto.
- **Collaborator**, dados dos colaboradores.
- **AnimalModel**, armazena os animais utilizados em um determinado projeto de pesquisa.
- **Animals**, dados dos animais.
- **Animal**, dados dos pesquisadores.
- **ExperimentalProcedures**, dados referentes aos procedimentos experimentais.
- **PostOperative**, dados referente ao pós operatório.

- **Drugs**, armazena os fármacos utilizados no projeto.
- **Drug**, dados dos fármacos.
- **Attachment**, dados anexos.
- **Minute**, dados das reuniões.

Entretanto, um projeto de pesquisa necessita de muitas informações. O volume significativo de dados resultou em um diagrama relacional de proporções consideráveis. Devido a isso, apenas uma fração dos dados de algumas tabelas foi representada integralmente na figura 3.

Figura 3 – Diagrama do banco de dados



Fonte: O próprio autor.

Podemos dividir as tabelas em duas partes: a primeira consiste nos usuários, com a tabela principal *User* armazenando os dados gerais dos usuários, e três tabelas associadas *Researcher*, *Reviewer* e *Coordination* - que armazenam os dados dos pesquisadores, pareceristas e coordenadores, respectivamente.

A segunda refere-se aos projetos, com a tabela principal *Project* e a tabela dependente *VersionsProjects*; esse relacionamento é 1:N, ou seja, um projeto pode ter inúmeras versões. A tabela *VersionsProjects* serve apenas como um agregador das informações de um projeto, ou seja, das demais tabelas representadas na figura 3.

3.1.2.2 Rotas da API

– A aplicação possui os seguintes rotas:

1. **"/newProject"**, do tipo POST, rota de registro de novos projetos.
2. **"/updateDataProject/{projectId}"**, do tipo POST, rota de criação de uma nova versão, dado que o projeto já esteja salvo no banco de dados. Se os dados do projeto tiverem alguma alteração, o sistema cria uma nova versão.
3. **"/newCorrections"**, do tipo POST, rota para inserção dos dados do parecer.
4. **"/getProject/"**, do tipo GET, retorna uma lista contendo todos os projetos.
5. **"/getProject/{projectId}"**, do tipo GET, retorna os dados de um projeto em específico.
6. **"/getProjectByUser/{researcherId}"**, do tipo GET, retorna todos os projetos de um determinado pesquisador.
7. **"/getProjectReviewer/{reviewerId}"**, do tipo GET, retorna todos os projetos de um determinado parecerista.
8. **"/getProjectsCoordination"**, do tipo GET, retorna todos os projetos, exceto os com *status* "Salvo".
9. **"/getCertificate/{projectId}"**, do tipo GET, retorna um arquivo PDF contendo o certificado do projeto.
10. **"/updateReviewer/{projectId}"**, do tipo PATCH, atualiza os dados do projeto adicionando um parecerista.
11. **"/signup"**, do tipo POST, criação de novos usuários.
12. **"/signin"**, do tipo POST, login do sistema.
13. **"/updatePassword"**, do tipo POST, atualiza a senha de um usuário.

14. **"/updateData/{id}"**, do tipo POST, atualiza os dados de um usuário.
15. **"/user/{id}"**, do tipo GET, retorna os dados de usuário
16. **"/user"**, do tipo GET, retorna uma lista com todos os usuários do sistema.
17. **"/newResearcher"**, do tipo POST, atribui a um usuário o acesso ao perfil pesquisador
18. **"/getResearchers"**, do tipo GET, retorna uma lista com todos os pesquisadores
19. **"/newReviewer"**, do tipo POST, atribui a um usuário o acesso ao perfil parecerista
20. **"/getReviewers"**, do tipo GET, retorna uma lista com todos os pareceristas
21. **"/newCoordination"**, do tipo POST, atribui a um usuário o acesso ao perfil da Coordenação
22. **"/getCoordinations"**, do tipo GET, retorna uma lista com todos os coordenadores.

3.1.2.3 *Versões de um projeto.*

Toda a aplicação foi modelada em torno da problemática do gerenciamento de versões dos projetos, ou seja, o histórico de alterações. Ao cadastrar um novo projeto o mesmo encontra-se na versão 01 no sistema. A cada modificação essa numeração é incrementada. A API verifica se houve alterações, caso tenha alterações uma nova versão é criada.

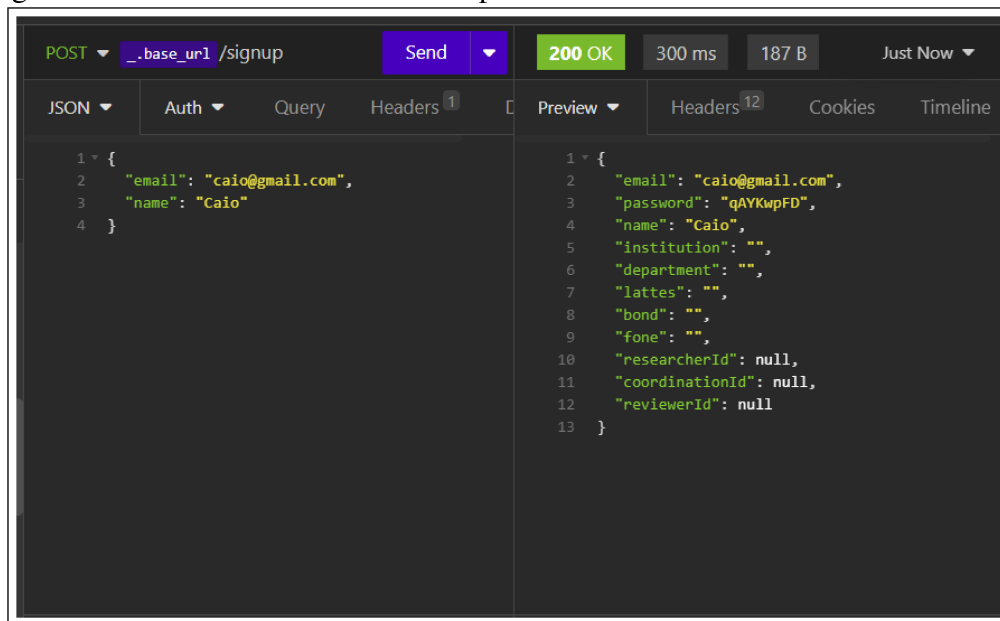
3.1.2.4 *Gerenciamento inicial de usuários.*

Inicialmente o gerenciamento limitava-se ao cadastro de novos usuários via 4 rotas da API.

1. **"/signup"**, do tipo POST, criação de novos usuários.
2. **"/newResearcher"**, do tipo POST, atribui a um usuário o acesso ao perfil pesquisador
3. **"/newReviewer"**, do tipo POST, atribui a um usuário o acesso ao perfil parecerista
4. **"/newCoordination"**, do tipo POST, atribui a um usuário o acesso ao perfil da Coordenação

Esse cadastro era feito diretamente na API por meio do *insomnia* como demonstrado na figura 4. Após ser criado, esse novo usuário era vinculado aos perfis de acesso por meio das outras rotas.

Figura 4 – Cadastro de novos usuários pelo *insomnia*



Fonte: O próprio autor.

3.2 Desenvolvimento de MVP do sistema

Nessa etapa houve a integração da primeira versão API com o *front end* da aplicação. Tivemos que disponibilizar a aplicação para o uso para isso utilizamos duas soluções voltadas para essa finalidade: A *vercel* para o *front end* e a *heroku* para a API

Com a aplicação em produção, os primeiros módulos foram testados tais como a submissão dos novos projetos e o gerenciamento dos mesmos. O sistema webCeua se divide em 3 perfis como descrito na introdução, o perfil do pesquisador, parecerista e coordenação.

3.2.1 Perfil do Pesquisador

O perfil do pesquisador compreende a tela inicial que solicita à API todos os projetos associados àquele pesquisador específico, listando-os na tela. Outra funcionalidade diz respeito à criação de um novo projeto. Dado que um projeto envolve diversos campos, o sistema permite que versões incompletas sejam salvas. A cada vez que os dados são salvos no banco de dados, uma nova versão é gerada. Em outras palavras, um projeto pode ter um número indefinido de versões, mas o sistema considera a última versão como a versão oficial do projeto.

3.2.2 Perfil do Parecerista

Quando um projeto é atribuído para o relator o mesmo é exibido na tela inicial, o relator possui a funcionalidade de analisar o projeto que consiste em preencher o parecer.

Nesse processo três funcionalidades da API são utilizados:

- retornar os projetos do parecerista.
- retornar dados de um determinado projeto que será analisado.
- salvar o parecer.

3.2.3 Perfil da Coordenação

Nesse primeiro momento o perfil da coordenação foi disponibilizado o gerenciamento inicial dos projetos e a escolha do relator. Quando o projeto muda para o status submetido, o mesmo é exibido no perfil da coordenação liberando a escolha do relator.

3.3 Coleta de feedbacks do MVP do sistema e refatoração

Após a disponibilização do MVP para as professoras do curso de medicina que fazem parte da CEUA da UFC, várias melhorias foram aplicadas na API de modo geral.

3.3.1 Geração de arquivos no formato PDF

A etapa final de um projeto no sistema é a sua aprovação ou a sua reprovação. Após a etapa de cadastro, revisão pelo parecerista e finalmente a apreciação na reunião da CEUA. Caso esse projeto seja aprovado, é disponibilizado o certificado de autorização da pesquisa.

O certificado é gerado acessando um *endpoint* destinado na API para essa funcionalidade, necessitando apenas passar o *id* do projeto em questão. Contudo, o arquivo em *Portable Document Format* (PDF) é gerado apenas se a data da reunião e a finalidade do projeto estiverem preenchidas corretamente, o que significa que o mesmo passou por todas as etapas de validação.

Pois essas informações serão atualizadas no banco de dados após a coordenação aprovar o projeto, processo que insere essas informações.

3.3.2 Gerenciamento de usuários

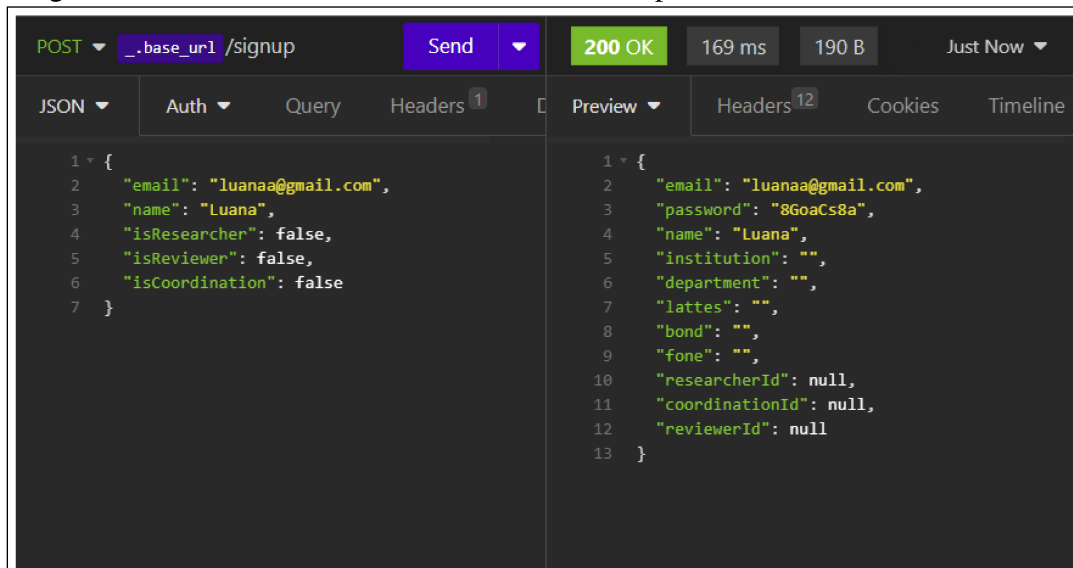
Atualmente o cadastro de novos usuários é gerenciado pelo perfil da Coordenação, utilizando apenas a rota: **"/signup"**, do tipo POST.

Para vincular aos perfis de acesso, na requisição foram adicionados três atributos:

- "isResearcher", caso seja *true* vincula o usuário ao perfil do Pesquisador.
- "isReviewer", caso seja *true* vincula o usuário ao perfil do Parecerista.
- "isCoordination", caso seja *true* vincula o usuário ao perfil da Coordenação.

No próximo capítulo será demonstrado como ficou essa integração com o *front end* desenvolvido.

Figura 5 – Cadastro atualizado de novos usuários pelo *insomnia*



Fonte: O próprio autor.

3.3.3 Notificações via e-mail

Uma das melhorias propostas consistiu na utilização de *e-mails*; consequentemente, desenvolvemos um módulo dedicado para essa finalidade, empregando a biblioteca *nodemailer* para o envio dos *e-mails*.

Nessa abordagem utiliza-se *templates*, ou seja modelos desenvolvidos em Linguagem de Marcação de Hipertexto (do inglês: *HyperText Markup Language*) (HTML) com todo o *layout* necessário para o *email*. esse *template* recebe os dados por parâmetro. Por exemplo, ao adicionar um novo usuário na plataforma é gerada uma senha provisória aleatória que é enviada por *email*.

Outras funcionalidades de notificação podem ser implementadas utilizando esse módulo de notificação, necessitando apenas criar um novo *template*.

Figura 6 – Implementação do módulo de *e-mails* usando o *nodemailer*

```
12 class Email {
13
14   async sendEmail(this: TypeEmail) {
15
16     const transport = nodemailer.createTransport({
17       service: 'hotmail',
18       auth: {
19         user: 'ceua@hotmail.com',
20         pass: '*****',
21       },
22     })
23
24     const templateOptions = {
25       viewEngine: {
26         extName: ".email",
27         partialDir: path.resolve('./src/views'),
28         defaultLayout: false,
29       },
30       viewPath: path.resolve('./src/views'),
31       extName: ".email"
32     }
33     transport.use('compile', hbs(templateOptions))
34     const info = await transport.sendMail(this)
35
36   }
37 }
38
39 export default Email
```

Fonte: O próprio autor.

4 RESULTADOS

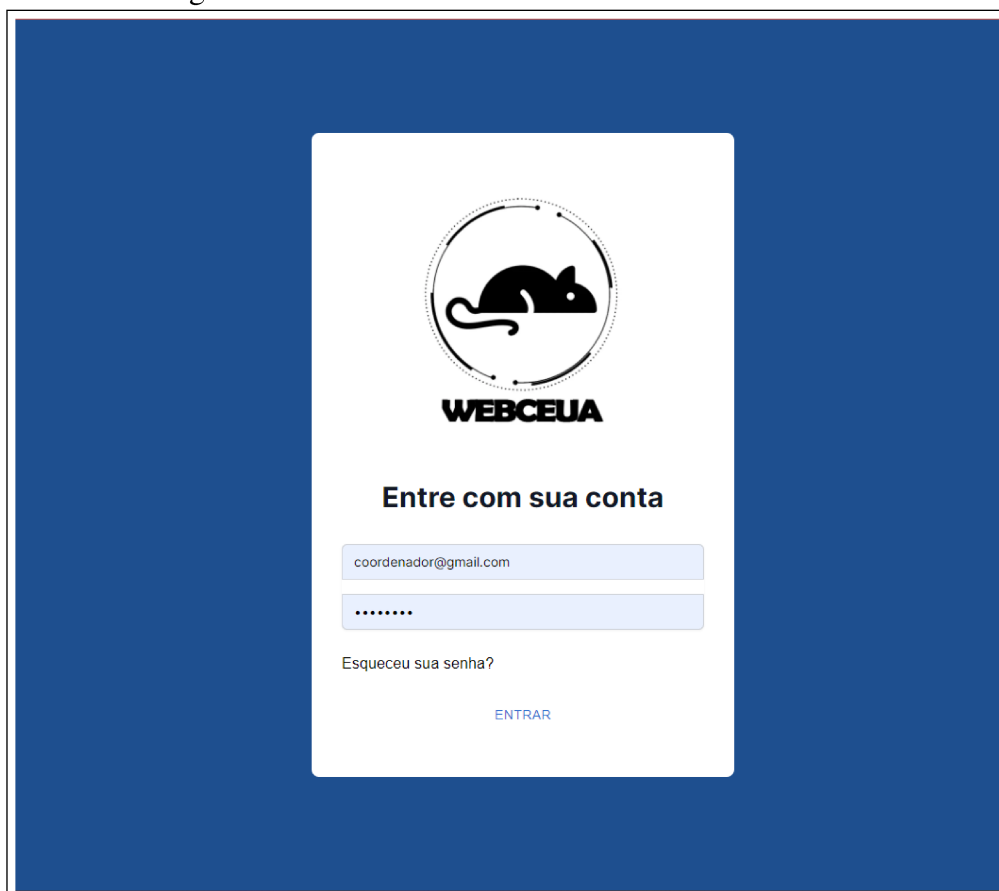
Neste capítulo, serão expostos os resultados alcançados durante o desenvolvimento deste trabalho, destacando as funcionalidades da API integradas às telas propostas.

As funcionalidades, conforme identificadas nos requisitos funcionais, foram implementadas na sua totalidade. No entanto, os resultados serão apresentados com ênfase nas telas, mas com foco principal na API, que é o elemento central deste estudo.

4.1 Acesso ao Sistema

O acesso ao sistema requer um endereço de e-mail e uma senha, conforme ilustrado na Figura 7. Ao longo do desenvolvimento da aplicação final, conclui-se que a centralização do cadastro de novos usuários pela coordenação seria a opção mais viável.

Figura 7 – Tela de Login

A imagem mostra a tela de login de um sistema. O fundo é azul escuro. No centro, há um formulário branco. No topo do formulário, há um ícone de um mouse dentro de um círculo com pontos, e o texto "WEBCEUA" em negrito. Abaixo disso, o texto "Entre com sua conta" é exibido. Há dois campos de entrada: o primeiro contém o e-mail "coordenador@gmail.com" e o segundo contém pontos para mascarar a senha. Abaixo dos campos, há o link "Esqueceu sua senha?". No final do formulário, há um botão azul com o texto "ENTRAR".

Fonte: O próprio autor.

Caso o Login seja válido, o acesso é liberado na aplicação pela API que retorna os dados do usuário contendo os seus níveis de acesso como demonstrado na figura 8. Se o usuário

possuir apenas um nível de acesso essa tela dos perfis não é exibida.

Figura 8 – Tela seleção dos perfis

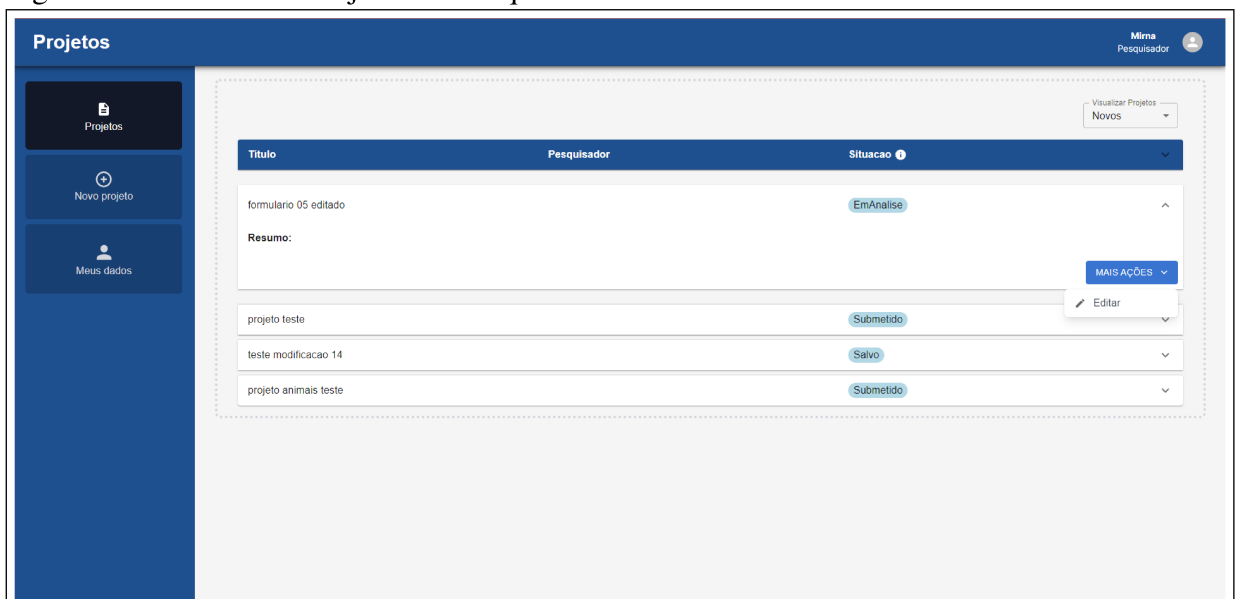


Fonte: O próprio autor.

4.2 Perfil do Pesquisador

A tela inicial deste perfil, lista todos os projetos adicionados pelo pesquisador como demonstrado na figura 9. Ao acessar essa tela uma requisição é realizada para a API que retorna como os dados.

Figura 9 – Tela Inicial: Projetos do Pesquisador



Fonte: O próprio autor.

4.3 Perfil da Coordenação

A tela inicial do perfil da coordenação é similar aos demais perfis contendo a listagem dos projetos, mas nesse caso a API retorna todos os projetos que já foram submetidos na aplicação mas omitindo os *status* salvo ou seja aqueles incompletos como demonstrado na figura 10

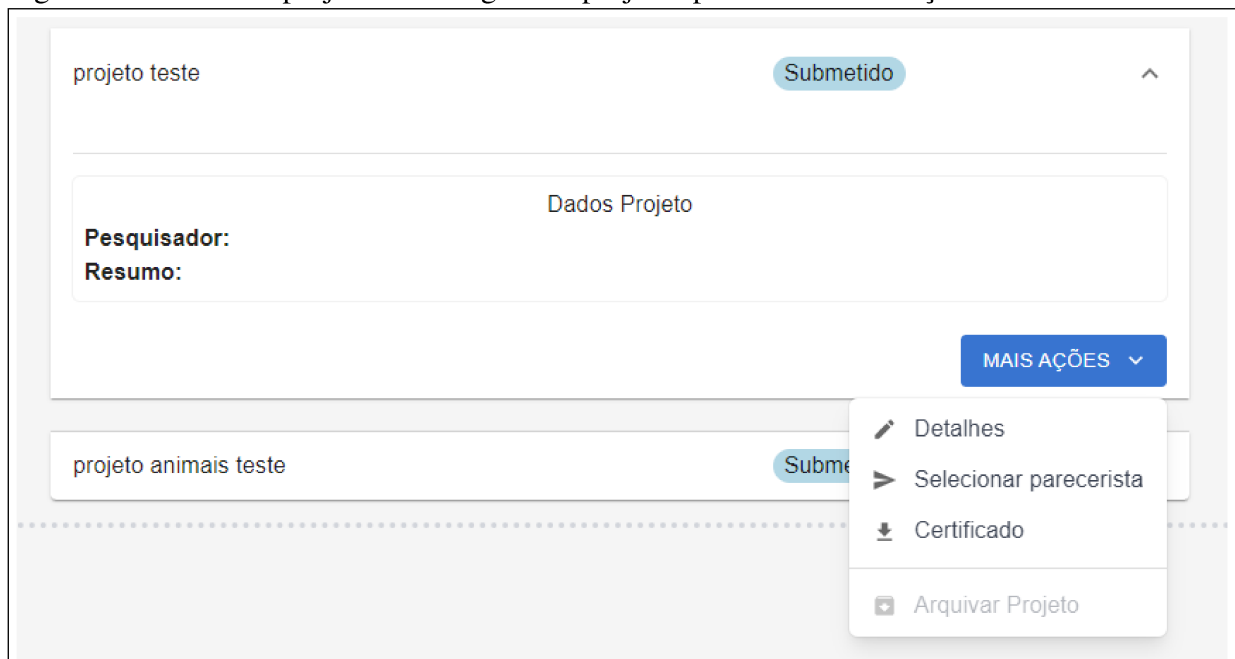
Figura 10 – Tela Inicial: Gerenciamento de Projetos



Fonte: O próprio autor.

Cada projeto possui o menu mais ações com algumas funcionalidades tais como "Detalhes", "selecionar parecerista", "Aprovar projeto" e "Certificado"

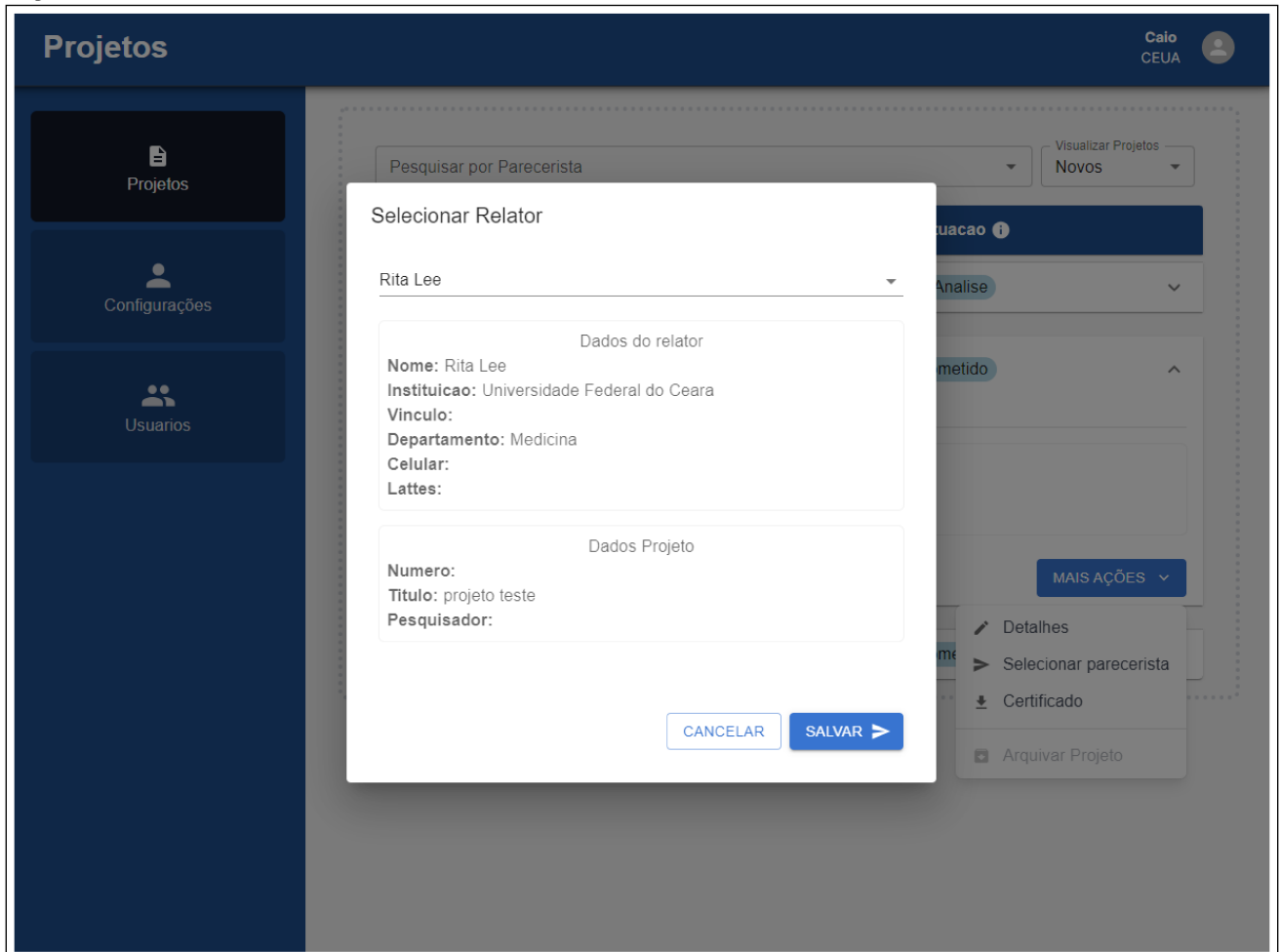
Figura 11 – Menu do projeto em listagem de projetos perfil da coordenação



Fonte: O próprio autor.

Em "Detalhes", pode-se visualizar uma cópia do projeto. Ao clicar em selecionar um parecerista o sistema abre um modal que lista todos os pareceristas disponíveis como demonstrado na figura 12. Nesse caso em específico há um requisição para API solicitando todos os pareceristas, ao confirmar a escolha do parecerista clicando no botão salvar, o projeto é atualizado com os dados do novo parecerista.

Figura 12 – Selecionar Parecerista



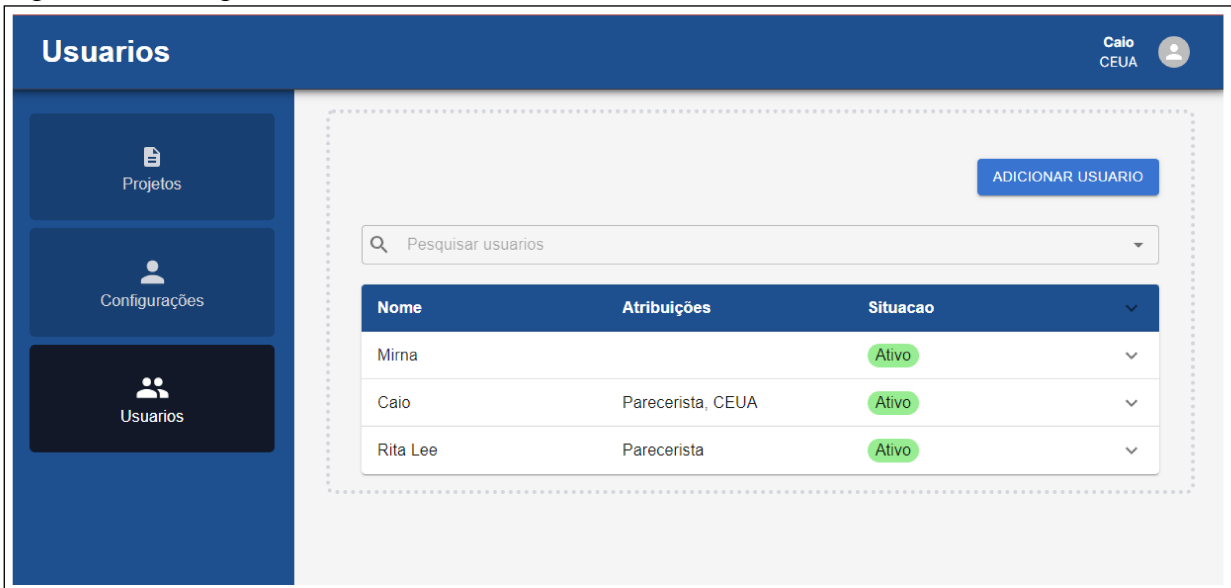
Fonte: O próprio autor.

A etapa final de um projeto na coordenação é sua aprovação, após o parecer e a apreciação na reunião da CEUA. Caso esse projeto seja aprovado a coordenação aprova o mesmo via sistema, ou seja inserindo a data da reunião e a finalidade do projeto em questão.

Com esse dados inseridos no banco de dados é liberado o certificado. O usuário poderá fazer o *download* clicando no botão "Certificado" que dispara uma requisição para a API gera um arquivo PDF contendo o certificado.

Outro módulo que a coordenação possui trata-se do gerenciamento de usuários da aplicação, que consiste na listagem de todos os usuários como demonstrado na figura 13.

Figura 13 – Tela gerenciamento de usuários

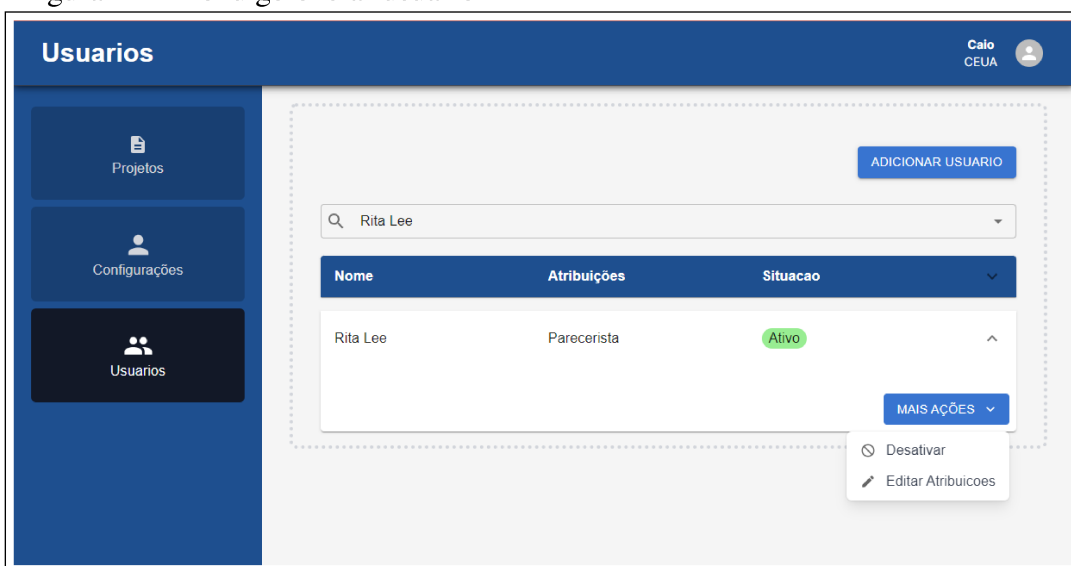


Fonte: O próprio autor.

O menu que gerencia um usuário contém duas funcionalidades, a primeira possibilita inativar a conta e a segunda editar as atribuições, ou seja, adicionar ou remover o acessos os perfis pesquisador, parecerista e a coordenação como demonstrado na figura 14

A aplicação não permite deletar usuários, pois perderia a consistência dos dados. Um exemplo seria um pesquisador que ficaria um ano de licença, se por algum motivo sua conta fosse deletada o histórico dos seus projetos seria perdido. A solução implementada no *back end* consiste na utilização de um atributo que indica se aquela conta está ativa ou não.

Figura 14 – Menu gerenciar usuário

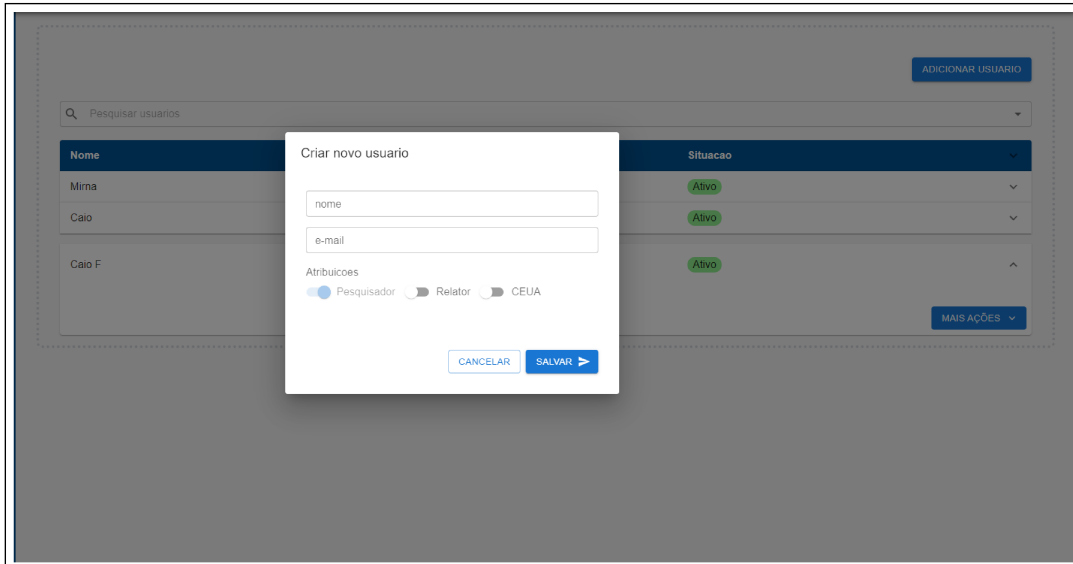


Fonte: O próprio autor.

A adição de novos usuários é feita nesse módulo, clicando no botão adicionar novos

usuários, como resultado abre um modal como demonstrado na figura 15. Para criar um novo usuário é necessário informar o nome, *e-mail* e as atribuições ou seja os níveis de acesso.

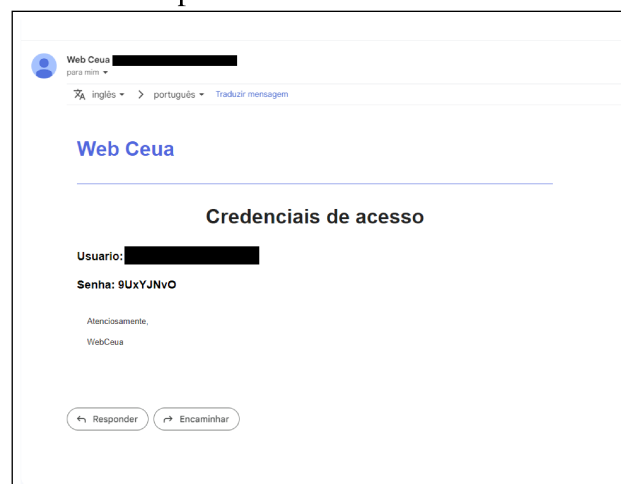
Figura 15 – Criação de novos usuários



Fonte: O próprio autor.

Após o usuário clicar no botão salvar, esses dados são enviados para API que processa e gera uma senha aleatória que será enviada por *e-mail* utilizando o módulo de *e-mail* descrito na metodologia. Na figura 16 podemos visualizar uma exemplo:

Figura 16 – *E-mail* enviado após o usuário ser criado

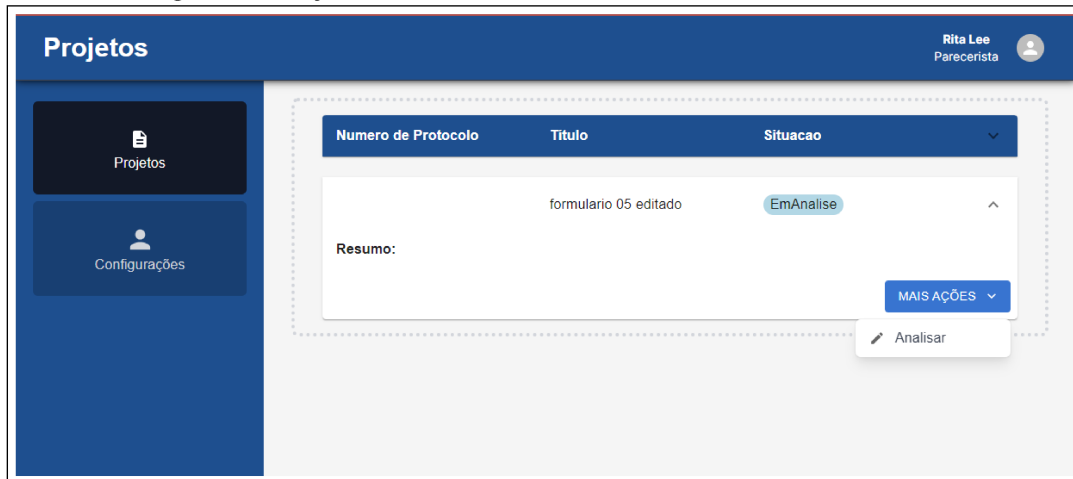


Fonte: O próprio autor.

4.4 Perfil do Parecerista

A tela inicial desse perfil consiste na lista de projetos atribuídos pela coordenação a um parecerista como demonstrado na figura 17. Nesse caso, a API retorna somente os projetos daquele parecerista.

Figura 17 – Listagem de Projeto Parecerista



Fonte: O próprio autor.

O parecerista poderá visualizar o projeto e preencher o formulário do parecer como demonstrado na figura 18, esse dados preenchidos serão vinculados ao projeto no *back end*.

Esse parecer é enviado para a coordenação que irá adicionar o projeto analisado para apreciação na próxima reunião da CEUA.

Figura 18 – Parecer

Fonte: O próprio autor.

5 CONCLUSÕES E TRABALHOS FUTUROS

Um dos principais desafios ao projetar a API do sistema webCeua foi otimizar o espaço no banco de dados, uma vez que um único projeto pode ter um número indeterminado de versões. Como destacado nas seções anteriores, um projeto contém uma quantidade significativa de dados, complicando o gerenciamento das versões, uma vez que a modificação de um único campo resulta na criação de uma nova versão.

Para superar esse desafio, foram implementadas soluções no *back end*. A primeira consistiu na modelagem do banco de dados, onde os dados do projeto foram divididos em várias tabelas, incluindo a *InitialDataProject*, responsável por armazenar os dados iniciais do projeto, como título e outras informações. No entanto, a tabela principal é a *VersionsProjects*, que armazena as diferentes versões do projeto.

Outra solução envolve uma função de comparação de dados, o sistema compara a última versão com os novos dados e, se houver alguma alteração, cria-se uma nova versão do projeto. Importante notar que nem todas as tabelas relativas aos dados do projeto recebem novas linhas; isso ocorre apenas na tabela das versões do projeto e naquela que registra as alterações efetivas nos dados.

Essa abordagem para o gerenciamento de versões demonstrou eficácia ao longo do desenvolvimento do software.

Com isso, obtivemos uma aplicação de fácil utilização e intuitiva. Os objetivos do desenvolvimento do software foram alcançados de maneira a permitir que o usuário final não se preocupe com a complexidade de gerenciar todo o fluxo dos projetos. Contudo, para trabalhos futuros, consideramos a adição de algumas funcionalidades, como a automação da distribuição dos projetos para os pareceristas, a implementação de testes automatizados no código e a pesquisa para otimizar ainda mais o espaço das versões no banco de dados.

REFERÊNCIAS

BLOG TREINAWEB. **No final das contas: o que é o Docker e como ele funciona?** 2023. <https://www.treinaweb.com.br/blog/no-final-das-contas-o-que-e-o-docker-e-como-ele-funciona>. (Acesso em: 12/09/2023).

DIGITALOCEAN. **Como construir uma API REST com o Prisma e o PostgreSQL.** 2023. Disponível em: <https://www.digitalocean.com/community/tutorials/how-to-build-a-rest-api-with-prisma-and-postgresql-pt>. (Acesso em 10/10/2023).

EXPRESS. **Express - framework de aplicativo da web Node.js.** 2023. Disponível em: <https://expressjs.com/pt-br/>. (Acesso em 05/10/2023).

GEEK BLOG. **Node.js: Por que e como usar? Várias justificativas!** 2023. Disponível em: <https://blog.geekhunter.com.br/por-que-usar-node-js-uma-justificativa-passo-a-passo/>. (Acesso em 23/08/2023).

MEDIUM. **Arquitetura MSC, o que é e por que usar?** 2023. Disponível em: <https://medium.com/@marianamohr/arquitetura-msc-o-que-%C3%A9-e-por-que-usar-42ad4cf19583>. (Acesso em: 12/07/2023).

MOZILLA. **JavaScript.** 2023. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. (Acesso em: 12/07/2023).