UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

Faculdade de ciências exatas

Sistemas de Informação

Pedro Henrique Pimenta da Silva

SISTEMA WEB PARA REGISTRO E GESTÃO DE DADOS DAS ESPÉCIES DE ERIOCAULACEAE

Diamantina

2025

PEDRO HENRIQUE PIMENTA DA SILVA

SISTEMA WEB PARA REGISTRO E GESTÃO DE DADOS DAS ESPÉCIES DE ERIOCAULACEAE

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Alessandro Vivas Andrade

Diamantina
2025

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI

FOLHA DE APROVAÇÃO

Pedro Henrique Pimenta da Silva

SISTEMA WEB PARA REGISTRO E GESTÃO DE DADOS DAS ESPÉCIES DE ERIOCAULACEAE

Trabalho de Conclusão de Curso apresentado ao Curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisito parcial para conclusão do curso.

Orientador: Prof. Dr. Alessandro Vivas Andrade

Aprovado em 26 de novembro de 2025

BANCA EXAMINADORA

Prof. Dr. Alessandro Vivas Andrade
Faculdade de Ciências Exatas - DECOM - UFVJM

Profa. Dra. Fabiane Nepomuceno Costa
Faculdade FCBS - DCBIO/UFVJM

Prof. Dr. Rafael Santin
Faculdade de Ciências Exatas - DECOM - UFVJM

Documento assinado eletronicamente por **Alessandro Vivas Andrade**, **Servidor(a)**, em 26/11/2025, às 15:08, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rafael Santin**, **Servidor(a)**, em 26/11/2025, às 15:09, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Fabiane Nepomuceno da Costa**, **Docente**, em 27/11/2025, às 19:24, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://sei.ufvjm.edu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1945296** e o código CRC **FB9C8E7B**.

Referência: Processo nº 23086.000729/2023-83 SEI nº 1945296

AGRADECIMENTOS

Agradeço primeiramente a Deus, pela força e sabedoria concedidas ao longo desta caminhada. À minha família pelo apoio constante e companheirismo em todos os momentos. Ao professor Alessandro Vivas, pela orientação, conselhos e dedicação que foram essenciais no desenvolvimento deste trabalho. Agradeço também aos colegas e professores do curso de Sistemas de Informação da UFVJM, pelo conhecimento e pelas experiências compartilhadas, que contribuíram significativamente para minha formação e para a realização deste projeto.

RESUMO

Este trabalho apresenta o desenvolvimento de um sistema Web para o registro e gestão de dados sobre espécies de Eriocaulaceae, plantas características do bioma brasileiro, com destaque para as regiões do Vale do Jequitinhonha, em Minas Gerais, e para as regiões da Bahia. Nessas regiões predominam espécies de Eriocaulaceae, Poaceae, Xyridaceae, possuindo considerável valor ecológico, cultural e econômico, o que leva a desafios relacionados à sua preservação e documentação. O sistema foi projetado para auxiliar no registro, consulta e organização das espécies de forma eficiente e acessível, utilizando o Django como framework principal e o PostgreSQL para gerenciamento de banco de dados. No processo de desenvolvimento foi priorizado a criação de uma interface amigável e intuitiva, visando atender as partes interessadas na preservação dessas espécies.

Palavras chave: Django, Eriocaulaceae, postgresql, sempre-vivas.

ABSTRACT

This work presents the development of a web-based system for registering and managing data on Eriocaulaceae species, plants characteristic of the Brazilian biome, with emphasis on the Jequitinhonha Valley region in Minas Gerais and the regions of Bahia. These regions are predominantly home to species of Eriocaulaceae, Poaceae, and Xyridaceae, possessing considerable ecological, cultural, and economic value, which leads to challenges related to their preservation and documentation. The system was designed to assist in the efficient and accessible registration, consultation, and organization of species, using Django as the main framework and PostgreSQL for database management. The development process prioritized the creation of a user-friendly and intuitive interface, aiming to meet the needs of stakeholders in the preservation of these species.

Keywords: Django, Eriocaulaceae,postgresql, sempre-vivas.

LISTA DE ABREVIATURAS E SIGLAS

CSS - Cascading Style Sheets

DRY - Don’t Repeat Yourself

HTML - HyperText Markup Language (Linguagem de Marcação de HiperTexto)

MVC - Model View Controller

MVT - Model view template

MVVM -Model View ViewModel

ORM - Object- Relational Mapping

SGBD - Sistema gerenciador de Banco de Dados

SUMÁRIO

1 INTRODUÇÃO

As espécies pertencentes a família Eriocaulaceae representam um patrimônio de grande relevância para a biodiversidade brasileira, sua localização é concentrada em regiões como o cerrado e a Serra do Espinhaço, em Minas Gerais e em algumas regiões da Bahia. São famílias reconhecidas principalmente como “Sempre-vivas”. Essa denominação refere-se ao nome popular dado a escapos e inflorescências de plantas que mantêm a aparência de estruturas vivas, mesmo depois de destacadas e secas” (Giulietti et al., 1988, p.1). Essas espécies não são reconhecidas apenas por sua beleza única, mas também por seu valor ecológico, econômico e cultural. Entretanto, a devastação do seu habitat natural, juntamente com a coleta desordenada e a ineficiência do estado em promover políticas públicas eficazes voltadas à conservação dessas plantas têm contribuído para sua extinção gradual.

Diante deste cenário, pesquisadores e ambientalistas enfrentam desafios significativos no que diz respeito à catalogação, armazenamento e o fácil acesso às informações referentes a essas espécies. De acordo com Mourão (2023, p.2) “No Brasil são encontradas cerca de 630 das 1.200 espécies de sempre-vivas conhecidas”. Atualmente, grande parte deste extenso conjunto de dados coletados encontra-se fragmentado em documentos físicos, planilhas ou arquivos, esse fato dificulta a continuidade e qualidade das pesquisas realizadas, a manipulação dos dados e, consequentemente, nas tomadas de decisões referentes à preservação da espécie.

Apesar da existência de plataformas que oferecem informações gerais sobre a flora nacional, como o Flora e Funga do Brasil, o escopo destas plataformas é amplo e não oferece funcionalidades específicas propostas neste sistema, como manutenção no histórico de alterações e buscas avançadas. Assim, evidencia-se a necessidade de um sistema informatizado voltado ao armazenamento, organização e a disponibilização de informações de forma segura, prática e acessível aos pesquisadores e aos demais interessados. Ao implementar funcionalidades como fluxo de aprovação,buscas avançadas, histórico de alterações, controle de acesso e uma interface adaptada a pesquisadores, espera-se complementar os sistemas similares, ampliando a qualidade e a confiabilidade das informações.

1.1 Justificativa

Diante do exposto cenário, torna-se evidente a necessidade do auxílio de um sistema informatizado que possibilite contribuir para a organização e o armazenamento de informações referentes às espécies.

Embora existam sistemas amplamente utilizados para consulta de espécies da flora nacional, como o Flora e Funga do Brasil, nota-se que tais iniciativas possuem escopo abrangente e voltado ao contexto nacional. Nesse sentido, torna-se relevante o desenvolvimento de um sistema informatizado direcionado e específico para as espécies de Eriocaulaceae,visando proporcionar um ambiente que facilite o acompanhamento e o enriquecimento contínuo das informações por parte de pesquisadores, professores e demais colaboradores.

Ao promover o registro estruturado dessas espécies, o sistema busca incentivar a documentação científica, apoiar ações de preservação e ampliar o conhecimento sobre as referidas espécies, contribuindo para a sua valorização e continuidade dos estudos relacionados a esse grupo botânico.

1.2 Objetivos

1.2.1 Objetivo Geral

Desenvolver um sistema Web, para o registro e gestão de dados das espécies de eriocaulaceae, com o propósito de auxiliar para sua preservação e apoiar no trabalho de pesquisadores e ambientalistas.

1.2.2 Objetivos Específicos

Com o intuito de propor uma solução tecnológica que atenda às necessidades reais dos pesquisadores e as demais partes interessadas com o estudo das espécies da família Eriocaulaceae, este trabalho tem como objetivos específicos:

- Encontrar as necessidades e dificuldades enfrentadas por pesquisadores e demais interessados no acesso, registro e organização de informações sobre espécies da família Eriocaulaceae, tomando como referência o contexto regional de estudo;
- Projetar e implementar um sistema que permita realizar o cadastro estruturado das espécies, possibilitando também a leitura, atualização e exclusão dos dados, assegurando consistência e controle das informações armazenadas;
- Desenvolver uma interface intuitiva e de fácil navegação, que favoreça a utilização do sistema em diferentes dispositivos e facilite o trabalho dos usuários na manipulação dos dados cadastrados;
- Disponibilizar um ambiente digital que contribua para a documentação e divulgação de informações relevantes sobre a família Eriocaulaceae, apoiando ações de pesquisa e conservação ambiental.

1.3 Metodologia

A metodologia adotada neste trabalho caracteriza-se como pesquisa aplicada, orientada para o desenvolvimento de um sistema informatizado destinado ao registro e gestão dos dados das espécies da família Eriocaulaceae. A abordagem utilizada combina métodos qualitativos, para compreensão das necessidades informacionais, e quantitativos, relacionados à organização e estruturação dos dados no sistema.

O trabalho foi conduzido por meio de etapas sequenciais e complementares, que possibilitaram alcançar os objetivos propostos:

- Revisão bibliográfica: foram analisados artigos científicos e publicações relacionadas às espécies da família Eriocaulaceae, à importância de sua conservação e às soluções tecnológicas aplicadas à gestão de dados ambientais. Essa etapa forneceu embasamento teórico para a definição do problema e das funcionalidades essenciais do sistema.
- Levantamento de requisitos: com base em reuniões exploratórias com pesquisadores da área, foram identificadas as principais dificuldades no armazenamento e acesso às informações sobre as espécies, permitindo definir as funcionalidades essenciais para atender às demandas reais de uso, tais como cadastro, atualização, consulta dos registros e gerenciamento do histórico de edição.

- Modelagem: a partir dos requisitos levantados, foi elaborada a estrutura do sistema utilizando diagramas e modelos de dados, estabelecendo a forma como as informações seriam organizadas e acessadas pelos diferentes perfis de usuários (administrador, pesquisador e visitante).
- Implementação: o sistema foi desenvolvido utilizando o framework Django, associado à linguagem Python e às tecnologias de marcação e estilização HTML e CSS, além do PostgreSQL como sistema gerenciador de banco de dados. Nessa etapa, foram desenvolvidas as funcionalidades definidas anteriormente, observando aspectos de usabilidade, confiabilidade e rastreabilidade das alterações.
- Testes: foram realizadas verificações funcionais com foco em validar o funcionamento das funcionalidades de cadastro, edição, aprovação, consulta e navegação. Os testes buscaram garantir coerência dos dados, clareza na apresentação das informações e estabilidade do sistema.
- Verificação dos resultados:O Sistema atinge os objetivos propostos no que tange à organização das informações e ao suporte às atividades de pesquisa e preservação das espécies estudadas.

2 FRAMEWORK

No contexto do desenvolvimento de sistemas computacionais, o termo framework pode ser definido como um conjunto de componentes de software reutilizáveis que por sua vez, torna mais eficiente o desenvolvimento de novas aplicações. Segundo Gamma et al. (2007), “um framework é um conjunto de classes cooperantes que constroem um projeto reutilizável para uma determinada categoria de software”.

No âmbito do desenvolvimento de software, os frameworks representam uma base sólida para o desenvolvimento de novas aplicações, fornecendo não apenas componentes reutilizáveis, mas também um conjunto de boas práticas de programação, visando garantir que tantos os projetistas quantos os programadores possam ter um foco maior detalhando as exigências de negócio do software do que com detalhes de baixo nível do sistema (Willermann;Ibarra,2007,p.41).

Segundo os autores Willemann e Ibarra(2007):

> Um framework ou arcabouço é uma estrutura de suporte definida em que um outro projeto de software pode ser organizado e desenvolvido, quando se analisa o conceito no âmbito do desenvolvimento de software. Um framework pode incluir programas de suporte, bibliotecas de código, linguagens de script e outros softwares para ajudar a desenvolver e juntar diferentes componentes de um projeto de software. Os Frameworks são projetados com o propósito de facilitar o desenvolvimento de software, habilitando projetistas e programadores a gastarem mais tempo detalhando as exigências de negócio do software do que com detalhes tediosos de baixo nível do sistema.(Willermann;Ibarra,2007,p.41).

Desta forma, destaca-se como é útil os frameworks no desenvolvimento de uma aplicação mais ágil, segura e estável. A sua utilização também proporciona maior padronização de código, manutenção e a detecção de erros, aspectos essenciais para sistemas escaláveis e contínuos, como é o caso do sistema proposto neste trabalho.

2.1 Vantagens e Desvantagens

A utilização dos frameworks no desenvolvimento de sistemas traz uma série de vantagens e benefícios que impactam positivamente na produtividade, na qualidade do software e na segurança. Todavia, como em qualquer produto tecnológico, seu uso não fica isento de algumas limitações. A seguir, serão apresentadas algumas das principais vantagens e desvantagens relacionados a utilização de frameworks cujo foco é no desenvolvimento de sistemas web.

Dentre as principais vantagens da adoção de um framework é a redução de custos, permitindo a reutilização de código através de estrutura e padrões já estabelecidos, isso

auxilia no desenvolvimento garantindo que o desenvolvedor possa se concentrar nas regras específicas de negócio (Willermann;Ibarra,2007,p.42).

Outro aspecto relevante é a padronização no desenvolvimento, uma vez que os principais frameworks utilizados impõem arquiteturas bem definidas e robustas, baseando-se em padrões já consolidados no mercado, tais como Model-View-Controller (MVC), Model-View-Template (MVT), Model-View-ViewModel (MVVM), etc. Ademais, a existência de uma documentação abrangente, a incorporação de práticas consolidadas de segurança e suporte contínuo auxiliando na resolução de problemas promovendo um ambiente de desenvolvimento colaborativo, auxiliando na evolução contínua do projeto.

Entretanto, é necessário levar em consideração determinadas limitações que os frameworks impõem. A primeira delas refere-se ao custo de treinamento para adoção dos frameworks que é consideravelmente alto, a curva de aprendizagem inicial leva um tempo considerável, pois exige que o desenvolvedor compreenda sua arquitetura e seu funcionamento até que os desenvolvedores estejam aptos a utilizarem todas as suas funcionalidades de maneira eficaz.

Adicionalmente, os frameworks podem apresentar uma sobrecarga estrutural, adicionando complexidade e recursos adicionais que, embora robustos, podem ser desnecessários para projetos menores, impactando no desempenho e utilização de recursos. Além disso, a rigidez na estrutura de um framework pode limitar a flexibilidade, pois os desenvolvedores devem seguir os paradigmas adotados pela ferramenta, o que, em determinadas situações podem ser prejudiciais.

Diante deste cenário, é possível concluir que os frameworks são instrumentos valiosos no desenvolvimento de um sistema, desde que esteja devidamente alinhado às necessidades do projeto. Cabe aos gestores avaliarem cuidadosamente os benefícios proporcionados, refletindo sobre as eventuais limitações, visando garantir que a escolha do framework contribui efetivamente para a qualidade, evolução e a sustentabilidade do sistema.No contexto do presente trabalho, essas consideracoes foram essenciais para a adoção do framework Django,que se destaca por sua robustez e flexibilidade que se mostraram adequadas para as demandas do sistema proposto.

2.2 Tipos de frameworks

Os frameworks são classificados de acordo com o nível de controle e interação que é oferecido ao desenvolvedor, sendo assim, eles podem ser caracterizados em caixa branca, caixa preta e caixa cinza.

Os frameworks caixa branca (whitebox) são descritos como frameworks que tem como característica principal a existência de orientação a objetos. Nesses frameworks, a reutilização se dá através da herança e sobrescrita de métodos, onde os usuários utilizam subclasses que estendem as classes abstratas para criar aplicações específicas (Maldonado et al., 2002, p.23).

Já os frameworks caixa preta destacam-se por serem menos flexíveis, o reuso é através da composição, consequentemente, neste tipo de framework as funcionalidades internas não podem ser vistas nem modificadas e devem-se utilizar as interfaces fornecidas pelo framework (Willermann;Ibarra,2007,p.43).

Assis e Suzano(2003, p.8) afirmam que:

> Os de caixa preta se baseiam na composição de objetos, ou seja, eles definem a interface com os componentes de modo que elas sejam conectadas ao framework por meio da composição de objetos, tendo maior facilidade de uso e extensão que os de caixa branca.

Por fim, os de caixa cinza são um balanceamento entre os de caixa preta e branca, provendo tanto flexibilidade quanto capacidade de extensão através de herança e composição (Assis;Suzano, 2003, p.8). Desta forma, compreender os diferentes tipos de frameworks é fundamental para que os desenvolvedores possam adotar a melhor abordagem que irá se adequar às demandas específicas de cada projeto.

2.3 O framework Django

O desenvolvimento de sistemas web modernos demanda por ferramentas que possam oferecer agilidade, segurança e escalabilidade. Diante deste cenário, o framework Django

destaca-se por suas características robustas e eficiência na construção de aplicações, fator impactante para sua consolidação no mercado.

O django é um framework de alto nível de código aberto que incentiva o desenvolvimento rápido, além de um design limpo e pragmático. Escrito em Python criado em 2005 por Adrian Holovaty e Simon Willison nos Estados Unidos, destacando-se por oferecer um ambiente que simplifica a criação de soluções escaláveis (Simoni, 2024, p.47). Ele adota o princípio DRY(“Don’t Repeat Yourself”), incentivando a reutilização de código e a redução da duplicidade, promovendo um desenvolvimento mais ágil, limpo e eficiente (Chen et al., 2020, p.99).

O django tem como objetivo principal automatizar tarefas recorrentes no desenvolvimento web, permitindo assim que os desenvolvedores foquem nas regras de negócios da aplicação, sem se preocupar em desenvolver do zero elementos básicos como autenticação, acesso a banco de dados, administração, etc. Além disso, possui um sistema administrativo automatizado, que é gerado dinamicamente a partir das modelos definidos, facilitando o gerenciamento dos dados.

O django segue o padrão arquitetural Model-View-Template (MVT), sendo a model é a estrutura responsável por representar os dados da aplicação, possuindo uma ligação direta com um banco de dados que utilizam o Object-Relational-Mapping (ORM) do Django(Simoni, 2024, p.48). A camada da view atua como um intermediário entre a model e o template, cuja responsabilidade é receber uma requisição e enviar uma resposta, sendo feito através de uma função ou classe Python. Por fim, a camada de template é responsável por renderizar a interface com o usuário.

Além disso, o Django também faz uso da arquitetura Model View Controller (MVC), onde a aplicação é separada entre as camadas Model, View e controller.Segundo Dias (2020, p.18)

> No MVC, o Model é a camada que representa dos dados da aplicação (KAMIDA, 2017), sendo o principal responsável pela leitura, escrita e validação dos dados (TABLELESS, 2015). A View é a camada responsável pela interação web como o usuário através do navegador, onde são exibidos e coletados os dados por meio de XML ou HTML (TABLELESS, 2015). O Controller é a camada que faz o controle de todas as requisições, é o responsável por determinar quais informações serão extraídas do banco de dados pelo Model e exibidas aos usuários pela View (Kamida, 2017).

A escolha do Django para este projeto se justifica pela combinação de diversos fatores. Primeiramente, sua facilidade de trabalhar com bancos de dados relacionais de forma simples e intuitiva, por meio de sua integração nativa com ORM

(Object-Relational-Mapping) permitindo que o desenvolvedor trabalhe com os bancos de dados abstraindo os comandos SQL e favorecendo a manipulação dos dados diretamente por mediante as classes Python. Ademais, a segurança que o Django oferece é outro fator determinante,já que ele incorpora proteções as ameaças mais comuns, como cross-site request forgery (CSRF), injeção de SQL, cross-site- scripting(XSS) sendo essenciais quando se lidam com dados sensíveis.

Sucintamente, o django se apresenta como uma solução robusta e eficiente para o sistema proposto, visando organizar e armazenar dados sobre as espécies de plantas eriocaulaceae, a fim de contribuir para sua preservação e armazenamento de informações sobre determinadas plantas.

3 BANCO DE DADOS

No contexto da tecnologia de informação, um banco de dados pode ser definido como um conjunto organizado de dados, que são estruturados de forma que possam ser facilmente acessados, gerenciados e atualizados através dos sistemas computacionais. Sua projeção tem como finalidade armazenar informações de forma eficiente, segura e consistente, permitindo que as aplicações possam realizar operações de inserção, consulta, atualização e exclusão dos dados de modo rápido e preciso. De acordo com Date (2004), "um banco de dados é uma coleção de dados persistentes, usada pelos sistemas de aplicação de uma determinada empresa". Sendo assim, os bancos de dados são fundamentais para o desenvolvimento de sistemas, sendo a base em diversas aplicações que necessitam manipular dados de maneira íntegra e segura, garantindo não apenas a organização dos dados, mas também com relação a possibilidade de escalabilidade, integridade e segurança dos sistemas, aspectos essenciais para os sistemas atuais.

3.1 Sistema de Gerenciamento de Banco de Dados

Para que um de dados seja utilizado de maneira eficiente, é necessário o uso de Sistema de Gerenciamento de Banco de Dados (SGBD). Um SGBD é um software cuja responsabilidade é gerenciar, organizar, armazenar e manipular dados, garantindo aspectos essenciais para um banco de dados, como integridade, segurança, disponibilidade e consistência. De acordo com Elmasri et al. (2005, p.3):

> Um sistema gerenciador de banco de dados (SGBD) é uma coleção de programas que permite aos usuários criar e manter um banco de dados. O SGBD é, portanto, um sistema de software de propósito geral que facilita os processos de definição, construção, manipulação e compartilhamento de bancos de dados entre vários usuários e aplicações. A definição de um banco de dados implica especificar os tipos de dados, as estruturas e as restrições para os dados a serem armazenados em um banco de dados.

Ademais, um SGBD oferece uma interface entre os dados que estão armazenados e as aplicações que os utilizam, além de fornecer funcionalidades específicas para controle de acesso, concorrência, proteção e recuperação contra mau funcionamento ou falhas (Elmasri, et al., 2005, p.4).

3.2 Tipos de Banco de Dados

Os bancos de dados podem ser classificados em diferentes tipos, de acordo com a sua modelagem e estrutura. Entre os principais tipos, destacam-se os banco de dados relacionais, orientado a objetos, NoSQL. Os bancos relacionais é o mais consolidado entre os demais, principalmente por sua estrutura ser baseada em tabelas com colunas e linhas, sendo assim, os dados são organizados através de uma convenção de regras e os relacionamentos são definidos pelas chaves primárias e estrangeiras (Barros et al., 2017, p. 16). No que tange a definição conceitual do modelo relacional, Silberschatz, Korth e Sudarshan (2006, p.25) definem sua estrutura e funcionamento desta forma :

> Um banco de dados relacional consiste em uma coleção de tabelas, cada uma com um nome único atribuído.Uma Linha em uma tabela representa uma relação entre um conjunto de valores. Informalmente, uma tabela é um conjunto de entidades, e uma linha é uma entidade (Silberschatz; Korth; Sudarshan, 2006, p.25).

Esse modelo foi criado por Ted Codd, da IBM Research em 1970, fundado com base em conceitos matemáticos, como a teoria de conjuntos e a lógica de predicados, visando favorecer a consistência e integridade, além de permitir a execução de consultas complexas através da linguagem SQL. Tais características tornaram os bancos relacionais os mais adequados para sistemas empresariais, governamentais, acadêmicos e a qualquer sistema que exija o forte controle e a integridade de seus dados (Elmasri e Navathe, 2005, p.89).

Por outro lado, os bancos de dados orientados a objetos são uma alternativa aos relacionais representando os dados na forma de objetos. Essa abordagem integra os conceitos do paradigma da programação orientada a objetos (POO), foram desenvolvidos para satisfazer as demandas de aplicações mais complexas, ditas não convencionais(Vieira, 2001, p.48).

Os bancos NoSQL, por sua vez, surgem através da necessidade de lidar com grandes volumes de dados não estruturados, alta escalabilidade e requisitos de desempenho em tempo real, como em sistemas atuais de aplicações em nuvem, redes, IoT, etc. Esse tipo de banco de dados rompe com o modelo relacional tradicional, cujo objetivo não é substituir o modelo tradicional, mas sim ser uma opção quando é necessário bancos de dados com estruturas mais flexíveis (Frozza; Schreiner; Dos Santos Mello, 2022, p.28).

A utilização de bancos de dados NoSQL por empresas gigantes de tecnologia demonstra sua capacidade de lidar com desafios modernos e flexíveis de gerenciamento de dados. Nesse contexto, é importante destacar que:

> Os bancos de dados NoSQL tem sido amplamente adotados em empresas como Facebook, Amazon e Google com o intuito de atender às suas demandas de escalabilidade, alta disponibilidade e dados não estruturados. Além disso, atualmente, diversos bancos de dados NoSQL de código livre estão disponíveis, como: Cassandra1, Hypertable2, MongoDB3, Redis4, CouchDB5 e Dynamo6. (Loscio; Oliveira; Pontes, 2011, p.1).

Dessa forma, a compreensão dos diferentes tipos de banco de dados é fundamental para a escolha tecnológica coerente com as demandas de cada projeto. Cada modelo possui suas vantagens e limitações que os gestores devem levar em consideração antes de adotá-las em suas aplicações. A escolha adequada contribui diretamente na eficiência, manutenção e escalabilidade do sistema.

Para o desenvolvimento do sistema proposto neste trabalho, optou-se por um banco de dados relacional, uma vez que esse modelo proporciona alguns benefícios já citados anteriormente como robustez, integridade e segurança.

3.3 PostgreSQL

O PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto, amplamente reconhecido pela sua robustez e extensibilidade. Criado na década de 1980 como parte do projeto POSTGRES na universidade da Califórnia em Berkeley pelo professor Michael Stonebraker, o PostgreSQL evoluiu consideravelmente ao decorrer dos anos, tornando-se um dos bancos de dados mais completos, seguros e sendo amplamentes respeitados no mercado.Sua arquitetura segue o modelo relacional, sendo assim, a modelagem dos dados ocorre por meio de tabelas inter-relacionadas, tendo suporte a linguagem SQL e a seus recursos como triggers, views, análise de texto, entre outros (Elgrably, 2015, p.21).

Entre os diferenciais do PostgreSQL em relação aos demais SGBS, destaca-se sua robustez em relação às transações e concorrência, como destacam os autores:

> O PostgreSQL implementa o modelo ACID, garantindo a integridade dos dados e a confiabilidade das transações mesmo em ambientes de alta concorrência. Em termos de desempenho, o PostgreSQL é altamente aprimorado, com um otimizador de consultas sofisticado que busca encontrar o plano de execução mais eficiente para consultas complexas. Com a introdução de recursos como particionamento de tabelas e paralelismo de consultas, o PostgreSQL tem sido capaz de lidar com cargas de trabalho cada vez mais exigentes em ambientes de produção.(Merlo et al., 2024, p. 65).

Outro aspecto relevante a se considerar é a segurança. O PostgreSQL possui vários mecanismos de autenticação e autorização, como os certificados SSL, métodos de controle de

acesso, autenticação por senha, entre outros. Além disso, contém uma comunidade ativa e por permitir que desenvolvedores do mundo inteiro possam contribuir diretamente no desenvolvimento de novas funcionalidades e correções das vulnerabilidades, contribuindo para a melhoria contínua (Elgrably, 2015, p.22).

No contexto deste trabalho, optou-se pelo PostgreSQL devido a sua natureza gratuita e de código aberto e pela sua capacidade de lidar com diversos volumes de dados estruturados, com integridade e confiabilidade. A sua flexibilidade aliada à compatibilidade com o framework Django, utilizado no sistema proposto foi um fator determinante, já a integração entre essas tecnologias são estáveis e bem documentadas. Por fim, sua ampla adoção em diferentes contextos reforça a sua maturidade e confiabilidade, o que assegura que o sistema proposto estará apoiado por uma base sólida e validada no mercado.

4. DOCKER: CONTEINERIZAÇÃO E PORTABILIDADE

O Docker atualmente é uma das principais ferramentas de conteinerização utilizadas atualmente no desenvolvimento de sistemas, permitindo com que os ambientes executem de forma isolada, portátil e reprodutível. Além disso, aliado ao baixo uso dos recursos disponibilizados pelo sistema quando comparado com as máquinas virtuais (Curvel, 2022).O docker é uma plataforma de código aberto utilizada para o desenvolvimento, distribuição, e execução de aplicações, permitindo separar a aplicação da infraestrutura agilizando o desenvolvimento do sistema. (Docker,Inc., 2025). Desenvolvido e apresentado ao mundo em 2013 por Solomon Hykes, fundador e CEO da empresa dotCloud em uma palestra na Python Conference realizada na Califórnia.

A conteinerização consiste no empacotamento das aplicações juntamente com suas dependências, bibliotecas, frameworks e configurações necessárias visando garantir que o software opere de maneira consistente em diferentes ambientes, sendo possível sua execução independentemente do sistema operacional.

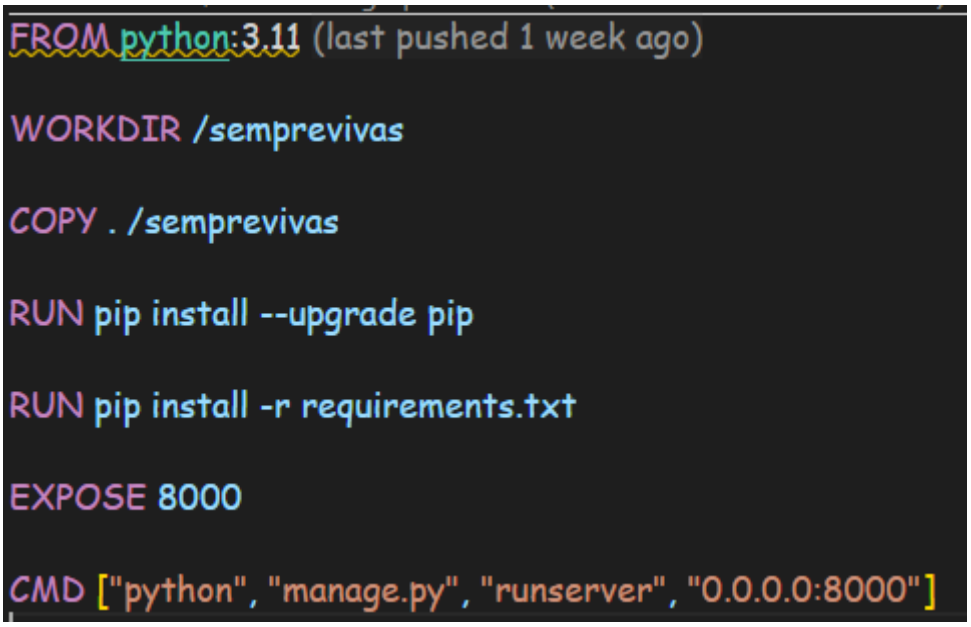
O uso do Docker no contexto de desenvolvimento e implementação traz vantagens significativas, entre as quais destacam-se a portabilidade, a escalabilidade e a padronização nos ambientes de desenvolvimento. Dessa forma, justifica-se a utilização do Docker no desenvolvimento do sistema proposto não apenas otimizando o processo de desenvolvimento, mas também proporcionando uma maior robustez e consistência na implantação, atributos essenciais para aplicações que demandam confiabilidade e melhoria contínua.

4.1 Configuração do Ambiente com Docker

A utilização do Docker foi essencial para assegurar a reprodutibilidade do ambiente de desenvolvimento e o ambiente de produção. O arquivo do Dockerfile foi elaborado para a definição da imagem principal da aplicação, incluindo as dependências necessárias para a execução e a configuração do servidor web. Como podemos ver na imagem abaixo, a instrução FROM python: 3.11 especifica qual será a imagem base utilizada, sendo neste caso a versão 3.11 do Python, essencial para a execução do Django e bibliotecas necessárias. Na linha posterior, o comando WORKDIR /semprevivas define o diretório de utilização dentro do container, aqui será o local onde todos os arquivos e comandos serão executados. O próximo comando COPY ./semprevivas realiza a cópia do projeto local para dentro do container.

Posteriormente, os comandos RUN pip install --upgrade pip e RUN pip install -r requirements.txt fazem a instalação das dependências do sistema. Já a instrução EXPOSE 8000 avisa onde que disponibilizará a porta da aplicação, que no caso é a 8000. Por fim, o último comando CMD ["python", "manage.py", "runserver", "0.0.0.0:8000"] define o que será executado quando o container for inicializado. Sendo assim, o Dockerfile proporciona de modo prático que o sistema execute em um ambiente controlado, independente do sistema operacional.

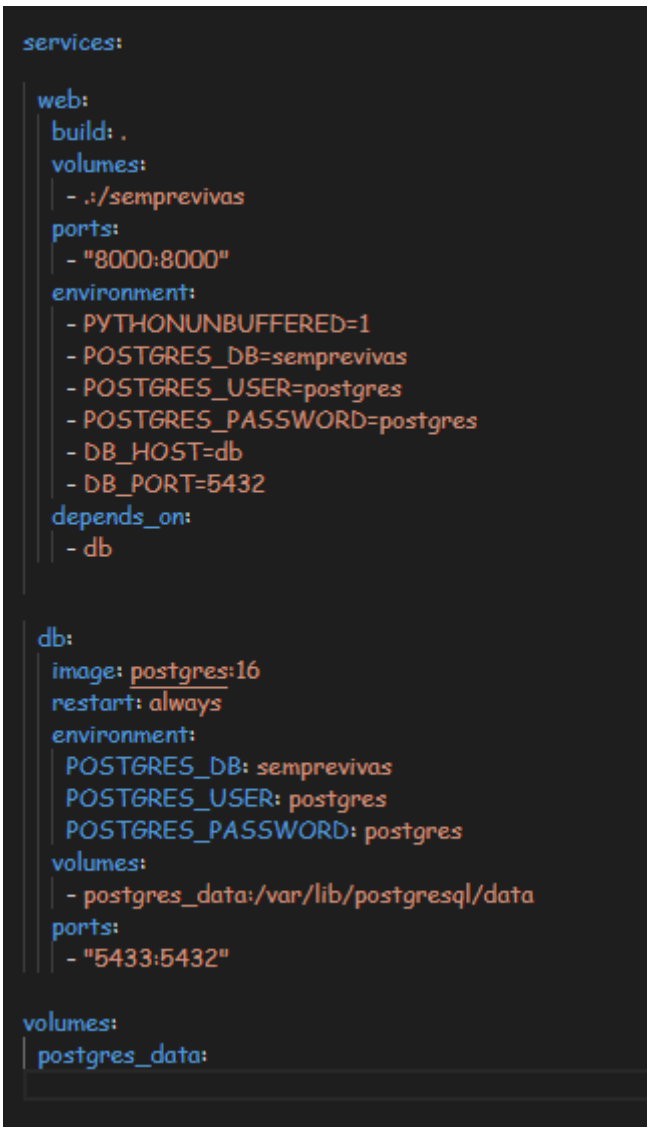
Figura 01- Dockerfile



Fonte: Autor.

Na imagem abaixo mostra o funcionamento do arquivo de configuração Docker-compose.yml é responsável por organizar os serviços que serão necessários para a execução do sistema, integrando o Django e o PostgreSQL em um ambiente unificado. O serviço web é onde é definida a aplicação, gerando a imagem a partir do Dockerfile, utilizando a porta 8000 e a variáveis de ambiente para realizar a conexão com o banco. Posteriormente, o serviço db utiliza a imagem oficial do postgreSQL na versão 16, juntamente das configurações necessárias para a sua utilização, tais como configuração do usuário,senha e banco de dados, além da utilização de um volume que assegura a integridade dos dados.

Figura 02- Docker-compose



```yaml
services:

  web:
    build: .
    volumes:
      - .:/semprevivas
    ports:
      - "8000:8000"
    environment:
      - PYTHONUNBUFFERED=1
      - POSTGRES_DB=semprevivas
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=postgres
      - DB_HOST=db
      - DB_PORT=5432
    depends_on:
      - db

  db:
    image: postgres:16
    restart: always
    environment:
      POSTGRES_DB: semprevivas
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5433:5432"

volumes:
  postgres_data:
```

Fonte: Autor

5. GIT E GITHUB: CONTROLE DE VERSÃO

No desenvolvimento de software especialmente em ambientes onde envolvem equipes cooperativas ou em projetos de pequeno e grande porte, o uso de ferramentas que auxiliem no controle, rastreio e controle de modificações do código fonte, é imprescindível. Um problema evidente que ocorria antes do surgimento dos sistemas de controle de versão, era comum que as equipes de desenvolvimento gerenciavam as alterações manualmente, o que acarretava em diversos problemas como no controle do histórico, sobreposição de mudanças e o impacto negativo na manutenção da integridade do projeto. Diante deste cenário, fica evidente a necessidade dos sistemas de controle de versão que são apoiadas por ferramentas que automatizam e auxiliam na garantia da manutenção, sendo o GIT a ferramenta mais utilizada atualmente (Barbosa; Da Silva, 2018, p.2).

O Git é um sistema de controle de versão que permite aos desenvolvedores acompanhar todas as alterações realizadas em um projeto, registrando cada mudança no histórico e possibilitando que os desenvolvedores realizem a reversão para estados anteriores caso haja necessidade, ele permite o compartilhamento de código de uma forma gerenciada sem a necessidade do uso de um repositório principal (Cunha, 2018, p.9).

A utilização do GIT permite a criação de branches (ramificações) permitindo o desenvolvimento e testes de novas funcionalidades de forma isolada, sem comprometer a versão principal do software e a realização dos commits (registros) das alterações. Sua importância pode ser notada em sua capacidade de facilitar a colaboração e controle nos projetos permitindo que os desenvolvedores atuem simultaneamente em um mesmo projeto, ao mesmo tempo em que as mudanças são feitas em cada arquivo (Morais Júnior,2024, p.18). Além disso, a realização de merges que integram as modificações realizadas de volta ao branch principal e as pull requests que funcionam como um mecanismo de controle revisando e aprovando as alterações antes que elas sejam incorporadas ao branch principal, fortalecendo a integridade e a qualidade do software. (Git, 2025).

No contexto deste trabalho, o Git foi utilizado em conjunto com o GitHub, plataforma cujo a finalidade é oferecer repositórios remotos, permitindo o armazenamento, compartilhamento e o trabalho juntamente de outras pessoas no desenvolvimento de um projeto. A plataforma tem um funcionamento semelhante a uma rede social permitindo interação e comunicação entre os programadores (Barbosa Da Silva, 2018, p.2). O uso de GitHub trouxe vantagens significativas, dentre elas destacam-se, o acesso do repositório de qualquer local, o compartilhamento e a preservação do histórico de versões do projeto.

Assim, a utilização do Git e GitHub contribuiu significativamente para o desenvolvimento do sistema de forma estruturada, segura e alinhada às boas práticas de programação atuais.

6.DESENVOLVIMENTO DO SISTEMA

Este capítulo tem como objetivo apresentar de maneira detalhada o sistema desenvolvido e suas funcionalidades relacionadas ao gerenciamento das informações sobre as espécies de plantas de eriocaulaceae. Serão apresentadas as suas principais funcionalidades, os arquivos de configuração do ambiente, bem como o detalhamento dos frameworks e testes adotados. Além disso, incluem-se os diagramas e imagens cuja finalidade é demonstrar o funcionamento do sistema e reforçar a sua necessidade.

6.1 Funcionalidades

O sistema foi criado visando atender as necessidades dos pesquisadores e ambientalistas que lidam com o gerenciamento das informações relacionadas às espécies de plantas eriocaulaceae. Tem como principal finalidade o cadastro, edição, atualização e exclusão das espécies, no qual é possível inserir informações referentes a elas, tais como taxonômicas, descrições, imagens e características específicas de cada planta. Além do CRUD das espécies cujas funcionalidades são permitidas apenas pelos pesquisadores e administradores, o sistema permite com que o usuário realize consultas e buscas avançadas, possibilitando que o usuário consiga encontrar de forma ágil e fácil os dados específicos de seu interesse. Essas consultas podem ser feitas a partir de variados atributos das espécies, como buscar pelo nome científico de uma dada espécie, família botânica, nome popular e demais atributos.

No que diz respeito à edição é possível com que o usuário atualize ou complemente as informações referentes às espécies, o sistema mantém um histórico destas mudanças, preservando o conhecimento científico quanto à própria informação das plantas ao longo do tempo. De modo similar, o sistema contempla a exclusão das espécies permitindo a manutenção de banco de dados consistente e coeso e organizado. O sistema foi projetado visando atender a todo tipo de usuário, com foco na usabilidade, oferecendo uma interface simples e intuitiva de modo que mesmo usuários com pouca familiaridade com tecnologia possam ver e manipular os dados de forma fácil.

O sistema foi concebido para funcionar com três perfis distintos de usuários: administrador, pesquisador e convidado. Cada um desses perfis apresenta funções e responsabilidades específicas. O usuário convidado, ao realizar seu cadastro e acessar o sistema através do login ou sem uma conta cadastrada, possui acesso restrito à página inicial,

onde são disponibilizadas as informações referentes às espécies cadastradas. O Pesquisador, por sua vez, dispõe das mesmas permissões atribuídas ao convidado, acrescidas da possibilidade de realizar alterações nas espécies de plantas. Todavia, para que tais modificações sejam efetivadas, é necessário que o administrador avalie e aprove as mudanças. Este último é responsável pela gestão do sistema, atuando no gerenciamento dos membros da equipe e também na aprovação das mudanças solicitadas pelos Pesquisadores e por fim, o administrador atribui e modifica os perfis de acesso dos usuários, podendo defini-los como Administrador, Pesquisador ou Convidado.

Logo, com essas necessidades, foram definidos os requisitos funcionais e não funcionais do sistema do sistema proposto, de acordo com Valente(2020) como:

> Requisitos definem o que um sistema deve fazer e sob quais restrições. Requisitos relacionados com a primeira parte dessa definição — o que um sistema deve fazer, ou seja, suas funcionalidades — são chamados de Requisitos Funcionais. Já os requisitos relacionados com a segunda parte — sob que restrições — são chamados de Requisitos Não-Funcionais.

No sistema proposto, os requisitos funcionais foram levantados mediante reuniões com pesquisadores onde foram definidos os requisitos essenciais para que o sistema opere da maneira esperada.Consequentemente, os requisitos propostos dos sistema são listados como:

Requisitos Funcionais (RF)

RF01 – O sistema deve permitir o cadastro de usuários com autenticação por login e senha.

RF02 – O sistema deve possibilitar que os pesquisadores e administradores cadastrem novas espécies de eriocaulaceae.

RF03 – O sistema deve permitir que os administradores realizem o gerenciamento do cadastro, aprovando ou rejeitando os cadastros das espécies que são submetidas pelos pesquisadores.

RF04 – O sistema deve possibilitar a edição e exclusão de espécies cadastradas por meio dos usuários do tipo pesquisador e administrador, mediante aprovação do administrador do sistema.

RF05 – O sistema deve disponibilizar uma área de listagem e busca de espécies, facilitando a consulta.

RF06 – O sistema deve permitir que administradores gerenciam os usuários, podendo editar e desativar contas e alterar o status de cada um.

RF07 – O sistema deve exibir detalhes de cada espécie, incluindo informações científicas e histórico de nomes.

RF08 – O sistema deve ter um histórico de edição sobre cada espécie de eriocaulaceae.

RF09 – O sistema deve integrar-se ao banco de dados PostgreSQL para persistência dos dados.

RF10 – O sistema deve utilizar containers Docker para garantir a portabilidade do ambiente de execução.

Já para os requisitos não funcionais do sistema:

Requisitos Não Funcionais (RNF)

RNF01 – O sistema deve ser desenvolvido utilizando o framework Django.

RNF02 – O banco de dados utilizado deve ser o PostgreSQL.

RNF03 – O sistema deve oferecer uma interface intuitiva e responsiva, garantindo usabilidade adequada em diferentes dispositivos.

RNF04 – O sistema deve possuir mecanismos de segurança no processo de autenticação e autorização de usuários.

RNF05 – Os dados cadastrados do sistema devem ser íntegros e consistentes.

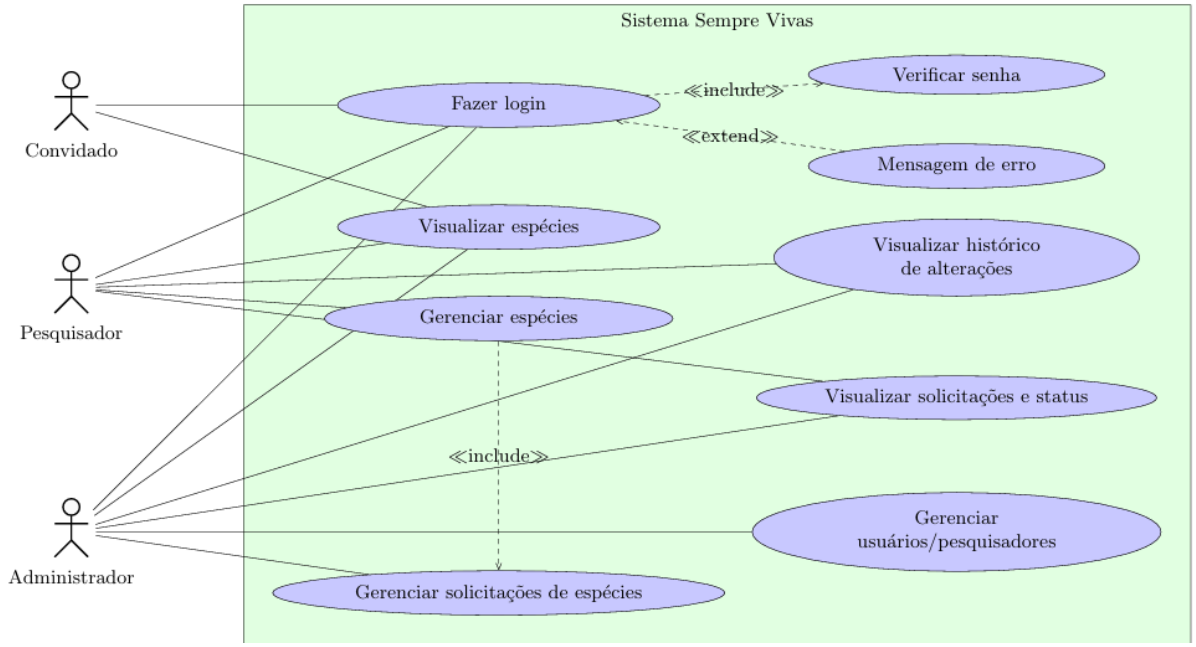
RNF06 – O sistema deve ser versionado utilizando Git, com repositório hospedado no GitHub.

RNF07 – O sistema deve ser projetado para ser escalável, permitindo futuras expansões.

A seguir será apresentado na figura 03 o diagrama de casos de uso do sistema proposto cuja finalidade é descrever as interações entre os atores do sistema e as funcionalidades disponibilizadas no sistema.

O sistema é composto por três atores sendo eles, convidado, pesquisador e o administrador. O convidado é responsável por visualizar as espécies cadastradas no sistema, por outro lado o pesquisador atua na manipulação dos dados de todas as espécies, podendo cadastrar, editar e excluir, além disso ele tem acesso ao histórico de edição de cada uma delas. E por último, o administrador possui as mesmas permissões dos demais atores, acrescidas do gerenciamento dos usuários do sistema e no controle das mudanças solicitadas pelos pesquisadores, aprovando ou não tais demandas.

Figura 03: Diagrama de casos de uso



Fonte:Autor.

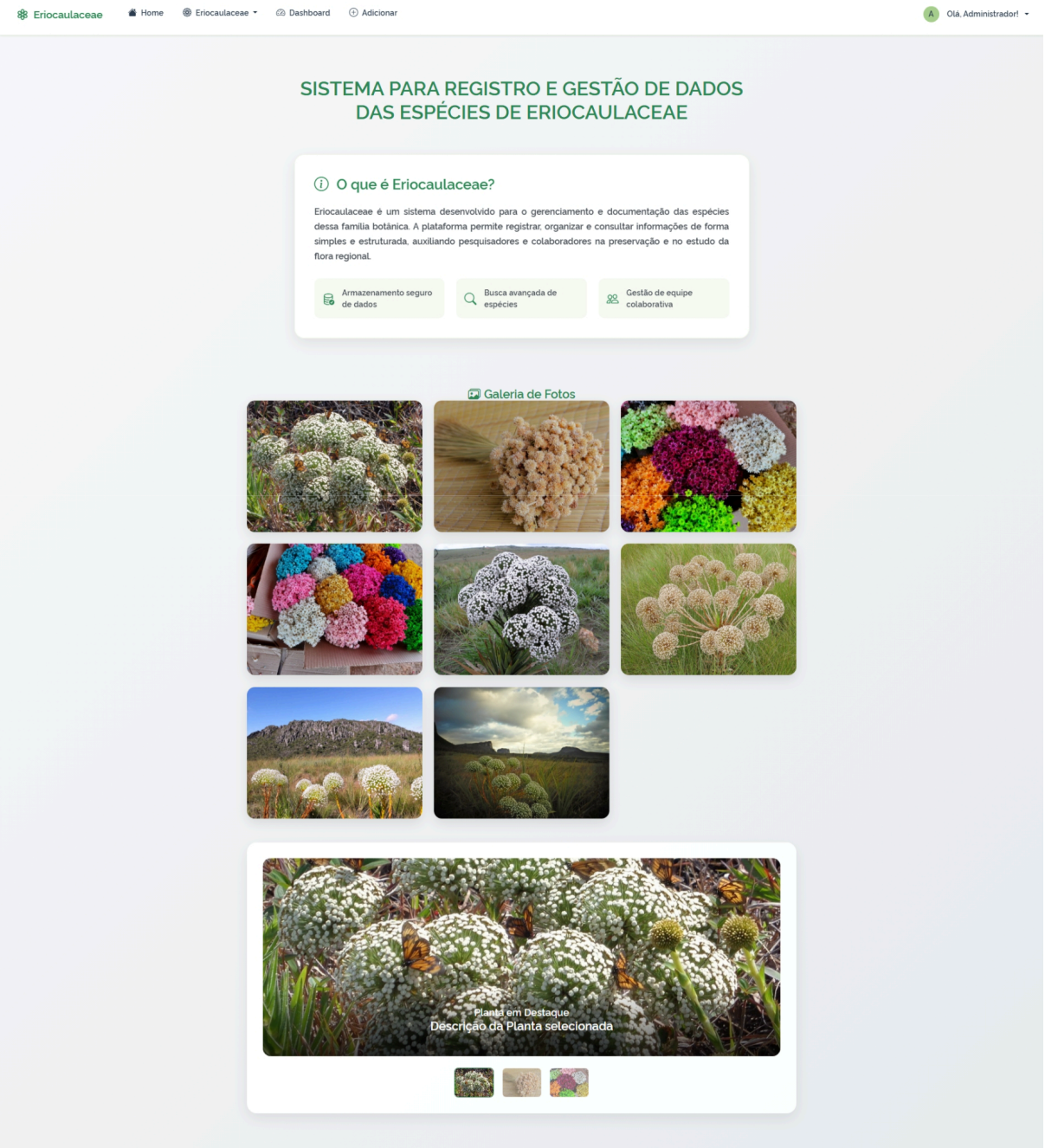
7. RESULTADOS

7.1 Principais telas do sistema

A seguir serão apresentadas as principais telas do sistema de gerenciamento de espécies de eriocaulaceae. Cujo objetivo é auxiliar no gerenciamento e controle destas plantas e servir como uma base de dados integra e coesa destas espécies.

A Figura 04 ilustra a página inicial que ao acessar o link do sistema o usuário é apresentado a esta tela inicial cuja finalidade é servir como uma porta de entrada para o acesso às funcionalidades do sistema, apresentando uma breve descrição de suas funcionalidades de forma breve, organizada e informativa. A página é estruturada em distintas seções, começando por uma definição do projeto, descrevendo sua finalidade. Posteriormente, é demonstrado uma galeria de imagens das espécies, oferecendo um espaço visual dedicado à exposição destas espécies.

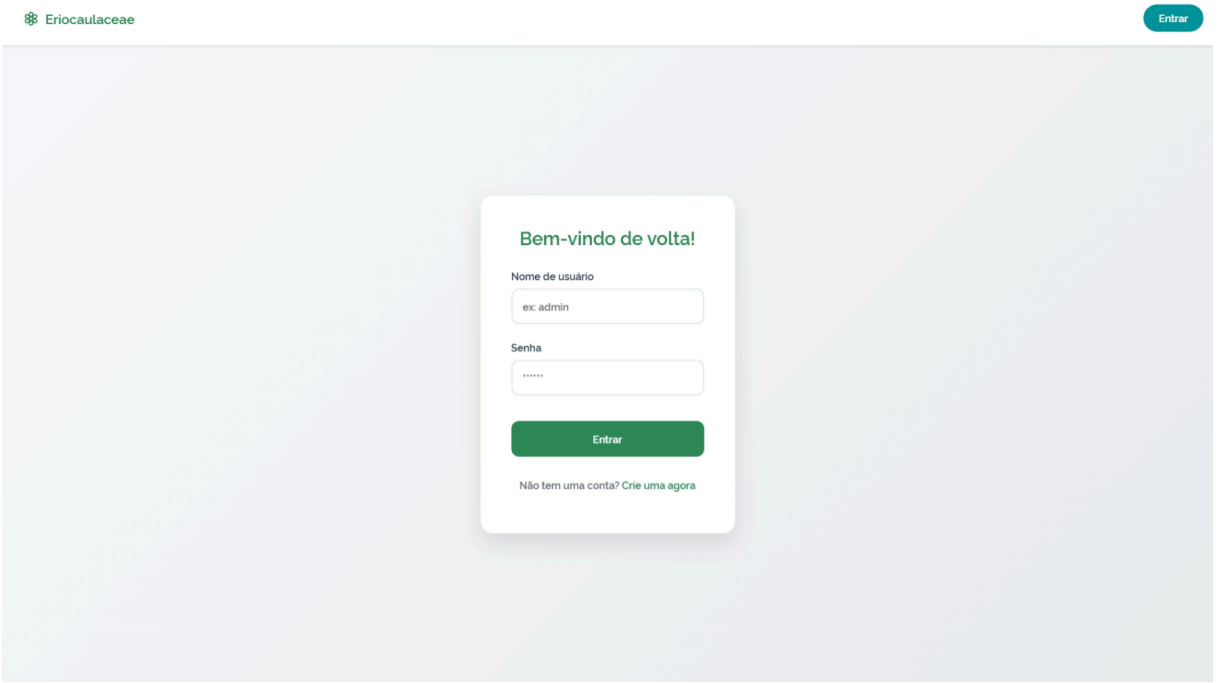
Figura 04: Página inicial do sistema



Fonte:Autor.

Como pode ser observado na Figura 05, é apresentado a tela de login onde o usuário digita o seu nome de usuário no sistema juntamente com sua senha. Após digitar os dados corretamente, o sistema já lida com o acesso a cada tipo de usuário. Cada usuário é direcionado para uma dashboard específica após o login, que serão apresentadas nas imagens posteriores.

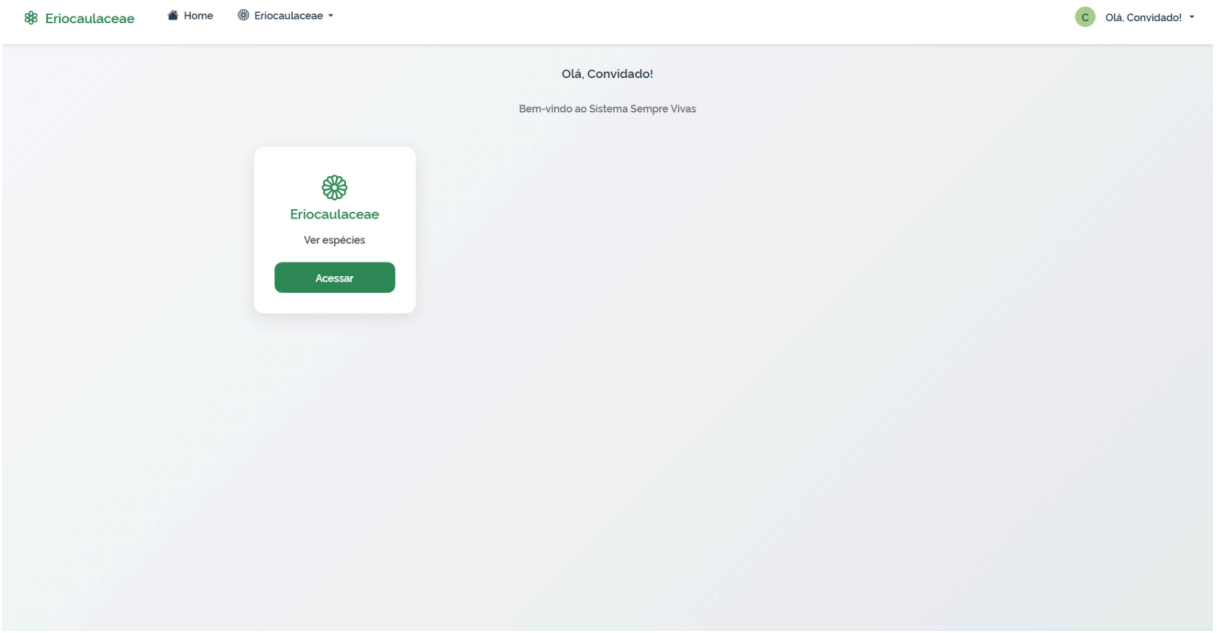
Figura 05: Página de login



Fonte:Autor.

A Figura 06 ilustra a Página exclusiva do convidado onde ele possui acesso ao menu de listagem das espécie

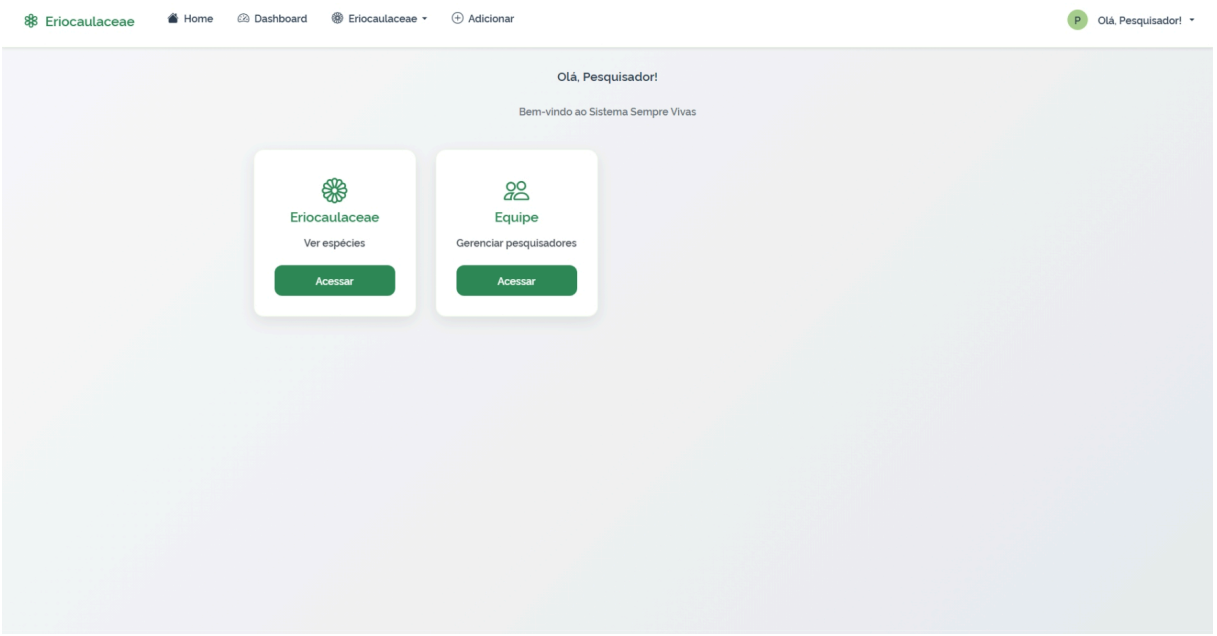
Figura 06: Dashboard do convidado



Fonte:Autor.

A Figura 07 ilustra a página exclusiva do pesquisador onde ele possui acesso ao menu de listagem das espécies e dos pesquisadores. Além disso, foi acrescido no menu superior a opção de adicionar as espécies.

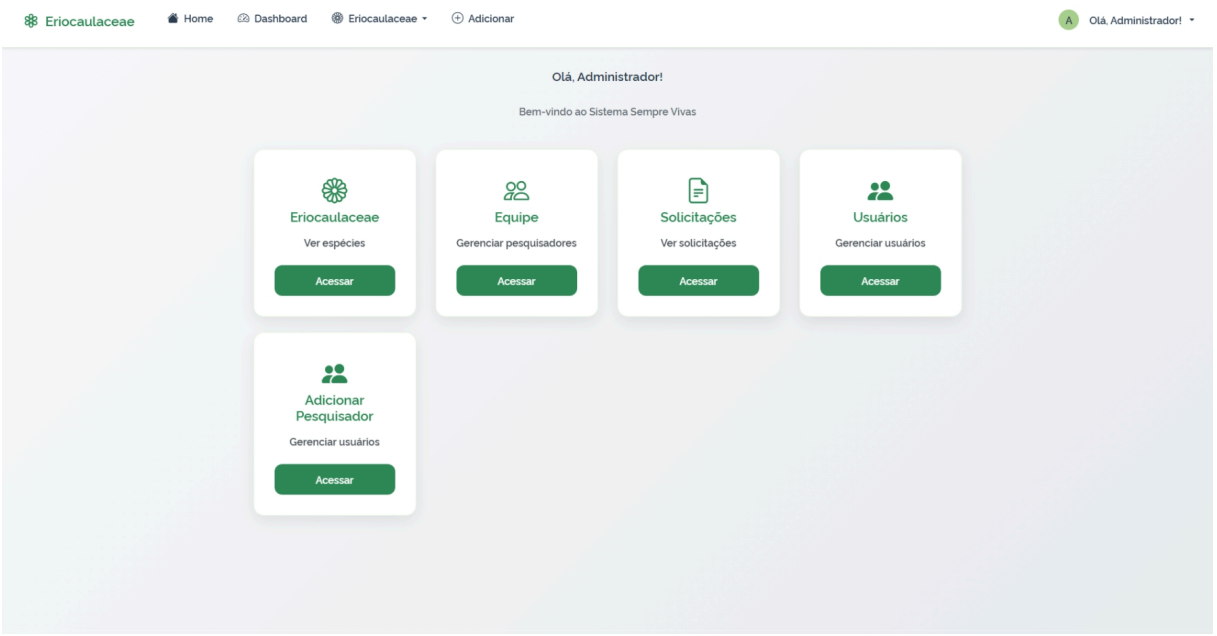
Figura 07: Dashboard do Pesquisador



Fonte:Autor.

A Figura 08 ilustra a página do administrador com o painel semelhante aos demais usuários acrescidos dos menus de cadastro e gerenciamento dos usuários e pesquisadores. Além disso, a opção de solicitações onde irá apresentar as solicitações feitas pelos pesquisadores.

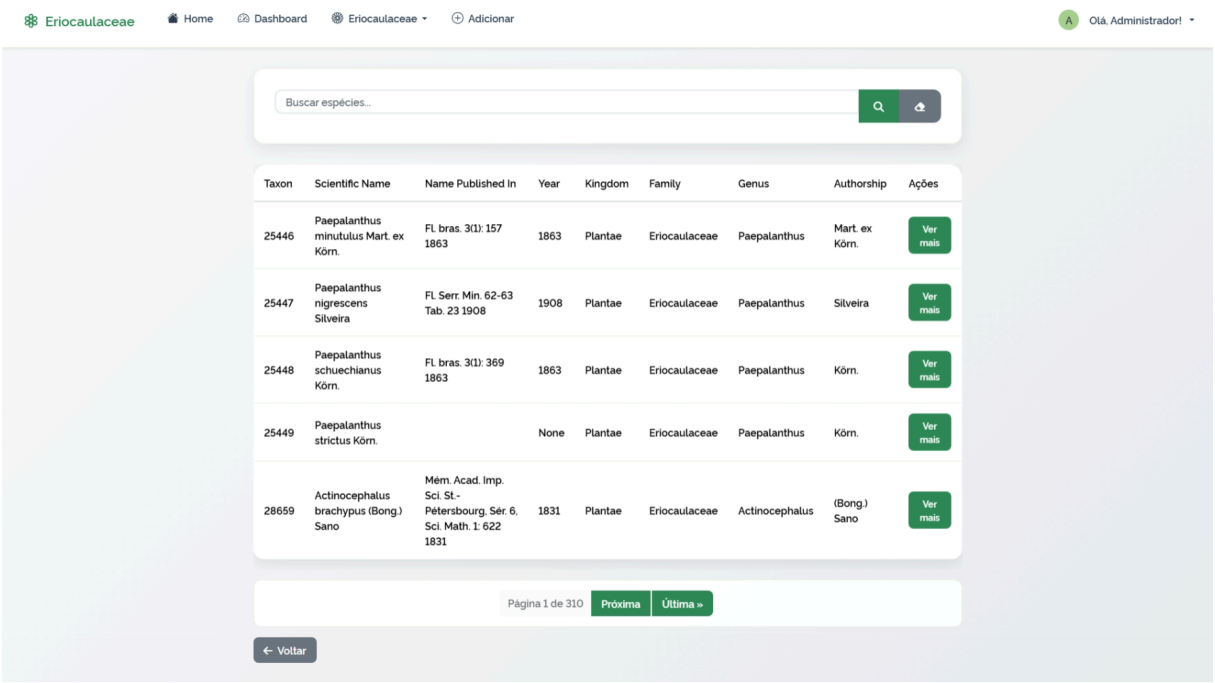
Figura 08: Dashboard do administrador



Fonte:Autor.

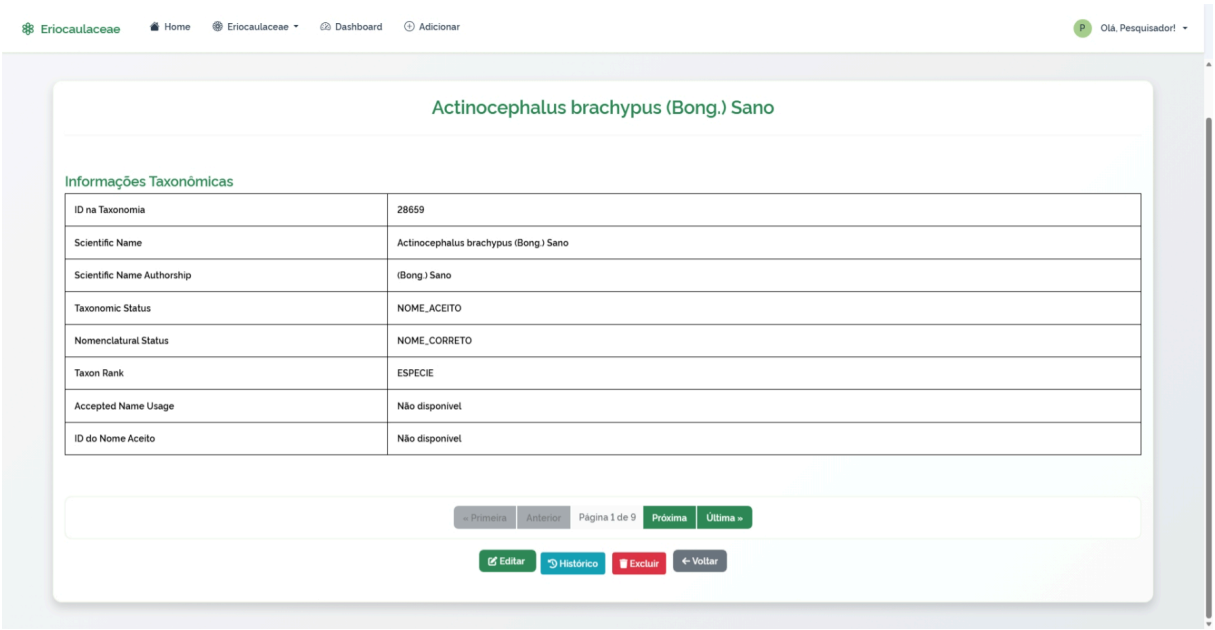
As Figuras 09 e 10 ilustram a listagem das espécies em uma tabela, na segunda imagem é possível ver o painel estendido com as demais informações de cada planta.

Figura 09: Listagem das Espécies



Fonte:Autor.

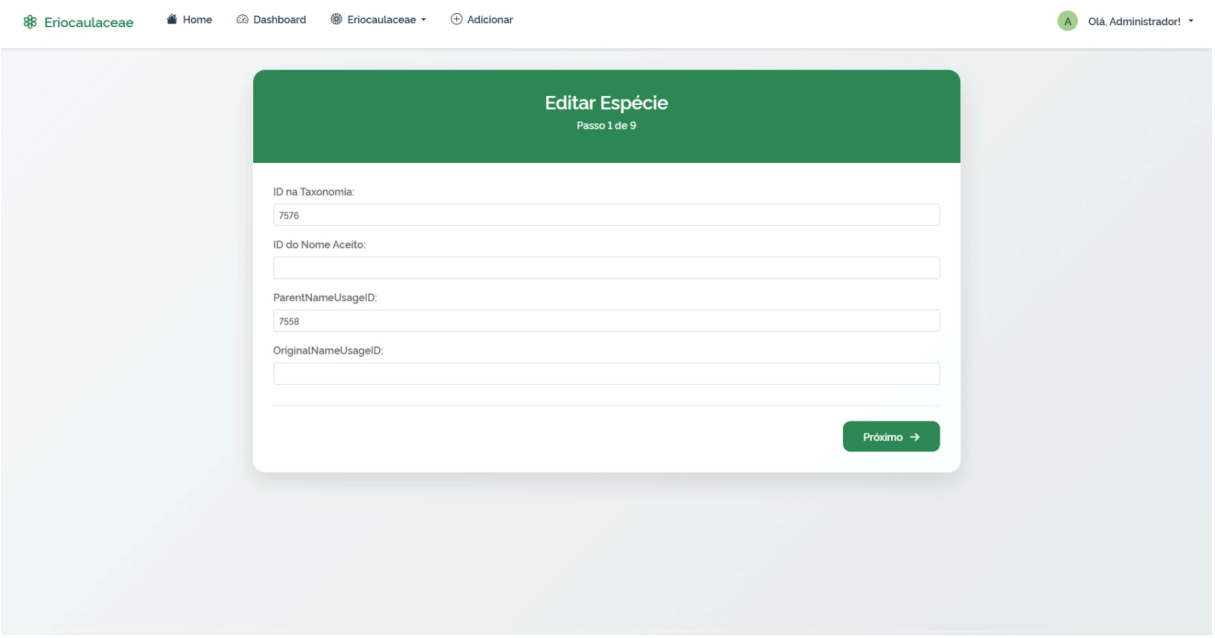
Figura 10: Informações estendidas das espécies



Fonte:Autor.

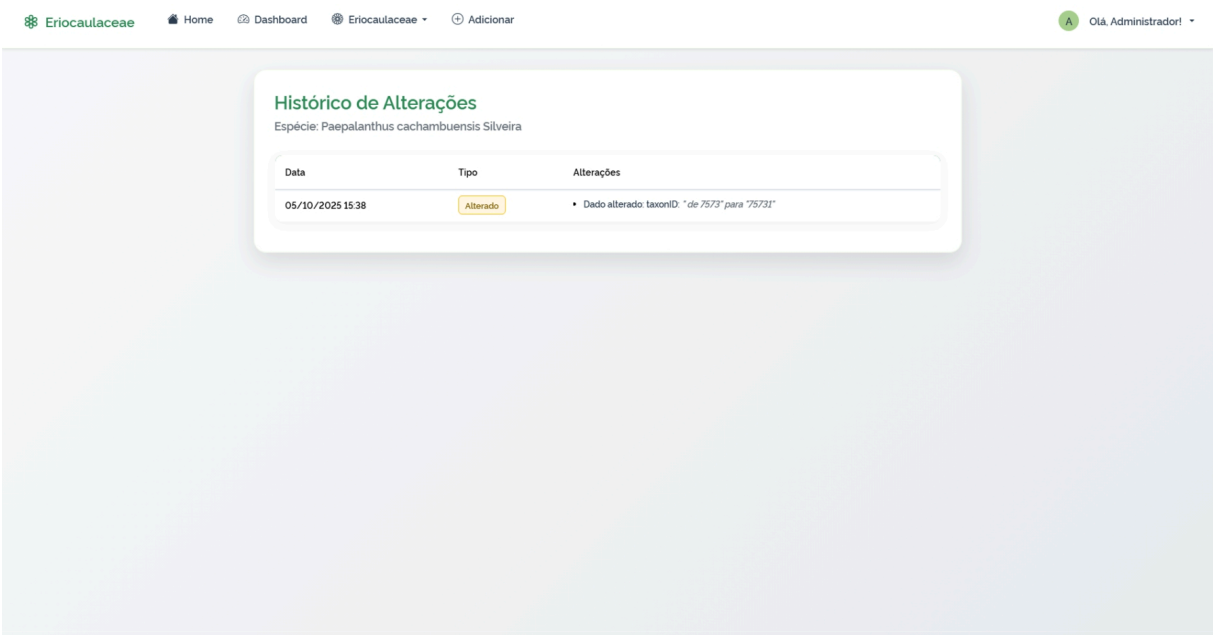
A Figura 11 ilustra a tela de edição de uma espécie. Posteriormente na Figura 12 é demonstrado uma tabela com o histórico detalhado de edições feitas em uma espécie.

Figura 11: Tela de edição de espécie



Fonte:Autor.

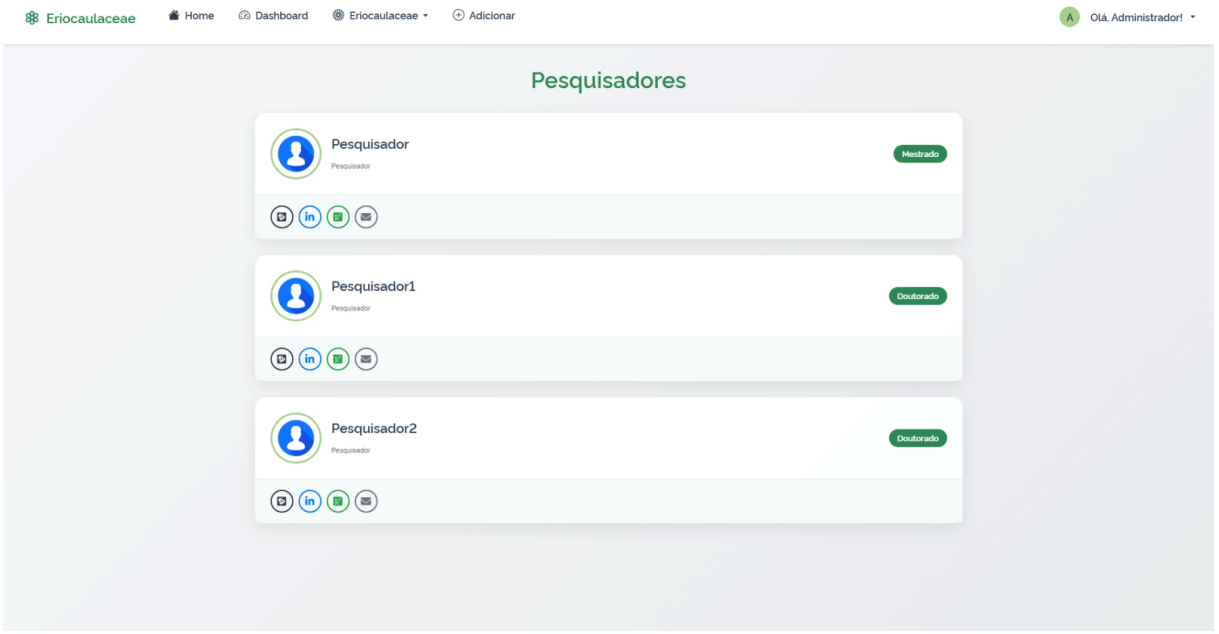
Figura 12: Tela de histórico de edição



Fonte:Autor.

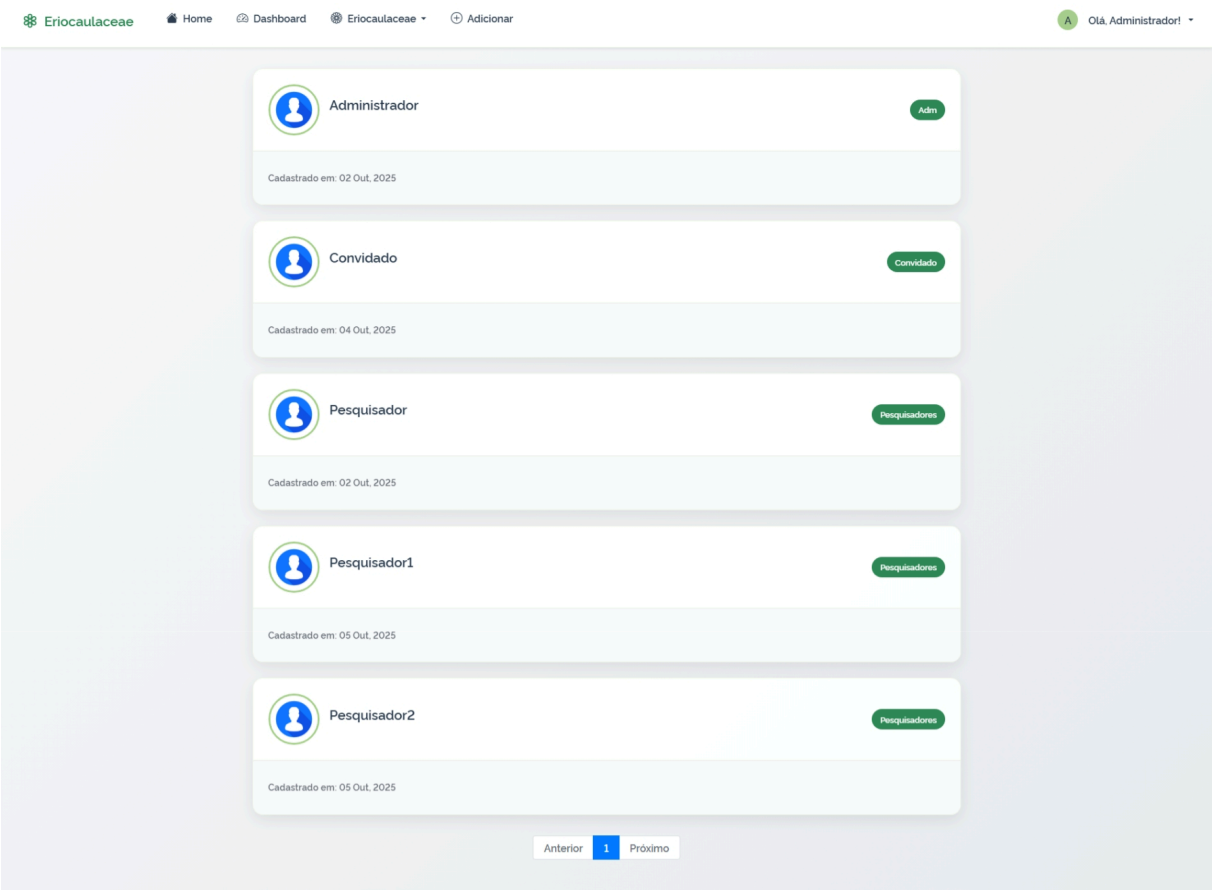
As Figuras 13 e 14 ilustram as telas dos pesquisadores e a segunda de todos os usuários cadastrados no sistema.

Figura 13: Tela de listagem dos pesquisadores cadastrados



Fonte:Autor.

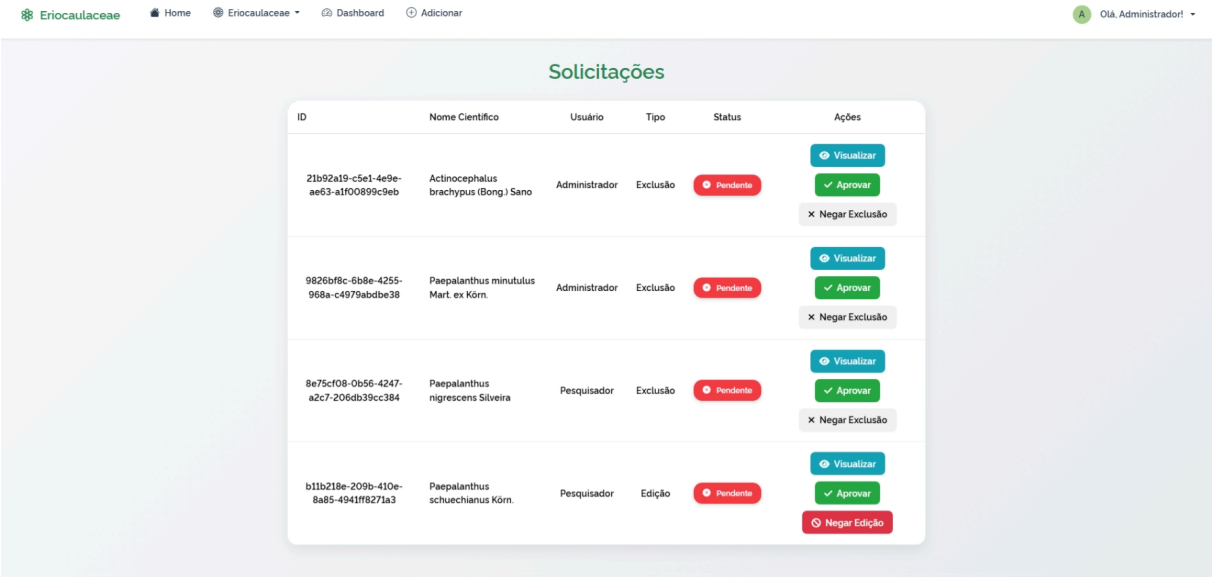
Figura 14: Tela de listagem dos usuários cadastrados



Fonte:Autor.

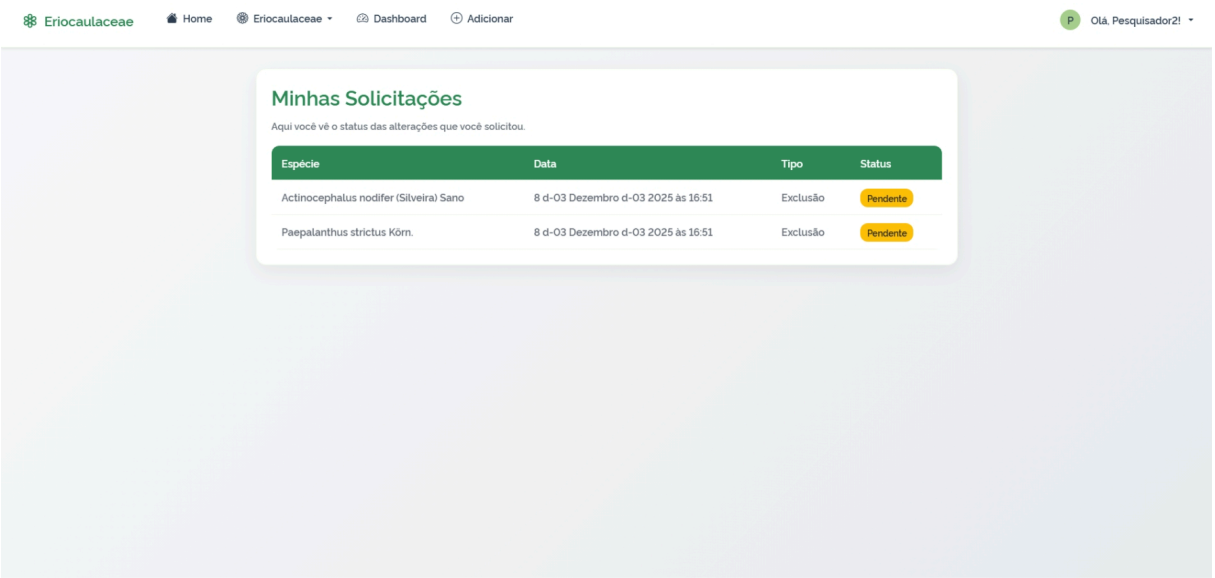
E por último as Figuras 15 e 16 ilustram a tela de solicitações feitas pelos usuários cujo acesso é restrito ao administrador e o histórico de solicitações dos usuários.

Figura 15: Tela de solicitação dos usuários



Fonte:Autor.

Figura 16: Histórico de solicitações dos usuários



Fonte:Autor.

8. CONCLUSÃO

O desenvolvimento do sistema para o gerenciamento das plantas de espécies eriocaulaceae possibilitou aplicar de forma prática os conhecimentos adquiridos ao longo da formação em Sistemas de Informação, evidenciando como a tecnologia pode ser aplicada como um instrumento de apoio às demandas da sociedade. O projeto teve como finalidade a criação de um sistema informatizado que possibilite contribuir para a organização e o armazenamento de informações referentes às espécies, contribuindo para o registro, preservação e manutenção de informações científicas, além de servir como um instrumento que auxilia na preservação ambiental.

A escolha do Django como framework proporciona uma base sólida para o desenvolvimento do sistema, sendo um ambiente de desenvolvimento robusto, escalável e seguro, fundamentado na arquitetura Model-View-Template(MVT). Tal arquitetura assegura a atribuição de responsabilidades e a maior clareza do código, permitindo com que a aplicação seja escalável e segura. A utilização do PostgreSQL como SGBD consolidou o desempenho e a confiabilidade do sistema, além de sua flexibilidade e alta compatibilidade com framework Django, auxiliando na estruturação do sistema e na realização de consultas complexas, assegurando que o sistema esteja apoiado em uma plataforma segura e validada no mercado.

Outro aspecto relevante a se considerar foi a utilização do Docker o que trouxe benefícios significativos no que tange à padronização do ambiente de desenvolvimento aliado a portabilidade, eliminando as incompatibilidades entre sistemas operacionais aliado a facilitação da implantação do sistema em diferentes contextos. De forma paralela, a utilização do Git e do Github proporcionou o versionamento e a rastreabilidade das mudanças implementadas em cada versão do sistema, auxiliando no controle mais rígido sobre cada etapa de desenvolvimento.

Com base nos resultados obtidos, acredita-se que o sistema atingiu com eficiência os objetivos propostos, a interface é simples e organizada proporcionando uma interação acessível e intuitiva, para diferentes perfis de usuários, incluindo indivíduos com limitações tecnológicas. Durante o desenvolvimento notou-se também a importância da análise dos requisitos e na atribuição de funcionalidades a diferentes perfis de usuários, o que solicitou a adoção de boas práticas de engenharia de software visando garantir uma arquitetura sólida, escalável e manutenção.

Em síntese, o trabalho demonstrou como o uso da tecnologia pode contribuir de forma significativa para atender as demandas em diferentes áreas, no presente contexto visa contribuir no gerenciamento e conservação das espécies. Como trabalhos futuros recomenda-se a abrangência a outras famílias de espécies além da implementação de novas funcionalidades para os usuários como geração de relatórios em PDF ou planilhas eletrônicas, sistemas de notificação internos, além de permitir a exportação e importação dos dados em diferentes formatos de arquivo.

REFERÊNCIAS

ANÁLISE COMPARATIVA ENTRE O BANCO DE DADOS CASSANDRA (MODELO NOSQL) E O POSTGRESQL (MODELO RELACIONAL) EM DUAS DIFERENTES ORGANIZAÇÕES EMPRESARIAIS. Colloquium Exactarum. ISSN: 2178-8332, *[S. l.]*, v. 8, n. 2, p. 39–56, 2017. Disponível em: https://journal.unoeste.br/index.php/ce/article/view/1324. Acesso em: 07 jun. 2025.

BARBOSA, Gabriel Silva; DA SILVA, Evaldo de Oliveira. Geração de Informações Gerenciais para Sistemas de Controle de Versão Um estudo de caso utilizado o GitHub. **Caderno de Estudos em Sistemas de Informação**, v. 5, n. 1, 2018.

BARROS, Juccelino Rodrigues Alves et al. Análise de desempenho de Banco de Dados Relacionais e Não Relacionais em dados genômicos. **Revista de Informática Teórica e Aplicada**, v. 24, n. 2, p. 11-27, 2017.
CHEN, Songtao et al. Django web development framework: Powering the modern web. **American Journal of Trade and Policy**, v. 7, n. 3, p. 99-106, 2020.

CUNHA, Marcela Bandeira. Entendendo o Uso do Git em Equipes de Desenvolvimento de Software. 2018.

CURVEL, Weked dos Anjos. Ensino de redes de computadores em ambiente virtual baseado em conteinerização de aplicações. 2022.

DA SILVA, EDUARDO NICOLINI SODRE. Desenvolvimento do Framework Java-Fácil. 2011.

DATE, Christopher J. **Introdução a sistemas de bancos de dados**. Elsevier Brasil, 2004.

DIAS, Luís Antonio Queiroz. Um ambiente seguro para aplicação com framework web Django. 2020.

DIAS, Luís Antonio Queiroz. Um ambiente seguro para aplicação com framework web Django. 2020.

DOCKER, Inc. Docker Documentation. Disponível em: https://docs.docker.com/. Acesso em:20 jul.2025

ELGRABLY, Isaac Souza. *Uma análise experimental entre sistemas gerenciadores de banco de dados open source*. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Universidade Federal do Pará, Belém, 2015.

ELMASRI, Ramez et al. Sistemas de banco de dados. 2005.

FROZZA, Angelo Augusto; SCHREINER, Geomar André; DOS SANTOS MELLO, Ronaldo. Projeto de Bancos de Dados NoSQL. **Capítulo de Livro em "Tópicos em Gerenciamento de Dados e Informações", anais do Simpósio Brasileiro de Banco de Dados**, 2022.

GIT. Git Documentation. Disponível em: https://git-scm.com/doc. Acesso em: 15 ago. 2025.

GIULIETTI, Ana Maria et al. Estudos em" sempre-vivas": taxonomia com ênfase nas espécies de Minas Gerais, Brasil. **Acta Botanica Brasilica**, v. 10, p. 329-377, 1996.

GIULIETTI, Nelson et al. Estudos em sempre-vivas: importância econômica do extrativismo em Minas Gerais, Brasil. **Acta botanica brasilica**, v. 1, p. 179-193, 1987.

LÓSCIO, Bernadette Farias; OLIVEIRA, HR de; PONTES, JC de S. NoSQL no desenvolvimento de aplicações Web colaborativas. **VIII Simpósio Brasileiro de Sistemas Colaborativos**, v. 10, n. 1, p. 11, 2011.

MALDONADO, José Carlos et al. Padrões e frameworks de software. Notas Didáticas, Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo, ICMC/USP, São Paulo, SP, Brasil, p. 28, 2002.

MERLO, André; PAVÃO, Jorge N. S.; FREITAS, Luiz C. de; SOARES, Jorge; BRANDÃO, Diego; BELLOZE, Kele. Estudo Comparativo entre MongoDB e PostgreSQL usando Embedding Documents. *In*: BRAZILIAN E-SCIENCE WORKSHOP (BRESCI), 18. , 2024, Florianópolis/SC. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2024. p. 64-71. ISSN 2763-8774. DOI: https://doi.org/10.5753/bresci.2024.244119.

MORAIS JÚNIOR, Pedro Pereira de. **GitGame: game para facilitar aprendizagem de Git**. 2024. Trabalho de Conclusão de Curso

.

MOURÃO, Nadja Maria. Artesanato, design e sempre-vivas em Diamantina/MG Handicrafts, design and evergreens in Diamantina/MG.

NASCIMENTO, Jamilson do et al. Visão geral sobre o padrão MVT (Model-view-template) do framework Django: implementação de uma aplicação para registro de usuários em sistema web. 2023.

PEREIRA, Adriano; COGO, Vinıcius Vielmo; CHARAO, Andrea Schwertner. Frameworks para Desenvolvimento Rápido de Aplicaçoes Web: um Estudo de Caso com CakePHP e Django. In: **Workshop de Software Livre (WSL)**. 2009.

RED HAT. O que é conteinerização? 2023. Red Hat. Disponível em: <https://www.redhat.com/pt-br/topics/cloud-native-apps/what-is-containerization#orquestra%C3%A7%C3%A3o-de-containers>. Acesso em: 17 Ago. 2025.

SANTOS, Lincoln Fernandes Paulino dos; MIOTTO, Aline Maria Malachini. Construção de um Framework Para o Desenvolvimento de Aplicações Web. Iniciação Científica Cesumar, [S. l.], v. 11, n. 2, 2009. Disponível em: https://periodicos.unicesumar.edu.br/index.php/iccesumar/article/view/1283. Acesso em: 29 maio. 2025.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. *Sistemas de banco de dados*. 5. ed. Rio de Janeiro: Campus, 2006.

SIMONI, Jhonatan Gratieri. Desenvolvimento de uma plataforma web com Django para análise de dados de qualidade de energia elétrica. 2024.

VALENTE, Marco Tulio. Engenharia de software moderna. **Princípios e práticas para desenvolvimento de software com produtividade**, v. 1, n. 24, 2020.

VIEIRA, Marina Teresa Pires. Banco de Dados Orientado a Objetos. **Universidade Federal de São Carlos Departamento de Computação, apostila**, 2001.


UFVJM