Desenvolvimento de Plataforma para Auxiliar nas Aulas de Anatomia da Madeira

Paulo Henrique Alves Santos

Universidade Federal de Uberlândia
Faculdade de Computação
Bacharelado em Sistemas de Informação

Uberlândia - MG
2026

Paulo Henrique Alves Santos

Desenvolvimento de Plataforma para Auxiliar nas Aulas de Anatomia da Madeira

Trabalho de Conclusão de Curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, Minas Gerais, como requisito exigido parcial à obtenção do grau de Bacharel em Sistemas de Informação.

Área de concentração: Sistemas de Informação

Orientador: Prof. Dr. Thiago Pirola Ribeiro

Uberlândia - MG
2026

Dedico este trabalho à minha família, que sempre me apoiou durante toda a minha trajetória. Em particular minha mãe Tânia e meu irmão César pelo apoio incondicional, e aos meus amigos pelo incentivo e apoio durante esta jornada.

Agradecimentos

Agradeço à minha mãe e ao meu pai pelo apoio incondicional durante minha jornada na graduação.
Agradeço aos meus amigos Carlos Humberto, Lucas Gabriel, Rafaela Peres, Paulo Peral, Guilherme Santa, Matheus Reis, que tornaram a jornada mais leve e significativa.
Agradeço ao meu irmão e à minha cunhada pela motivação e apoio durante toda a jornada.
Agradeço à minha família cujo carinho e apoio tornaram essa jornada possível.
Agradeço ao meu orientador, Prof. Dr. Thiago Ribeiro Pirola, pelo apoio, paciência e contribuições que tornaram essa conquista possível.
Agradeço à Universidade Federal de Uberlândia (UFU) e à Faculdade de Computação (FACOM) por todo o aprendizado, oportunidades e experiências ao longo de minha formação em Sistemas de Informação.
Agradeço ao PET Sistemas de Informação pelos conhecimentos adquiridos, amizades construídas e pela oportunidade de contribuir com o aprendizado de outros alunos.

"A mente que se abre a uma nova ideia jamais voltará ao seu tamanho original."
(Albert Einstein)

Resumo

Esse trabalho apresenta o desenvolvimento de uma plataforma web para o processamento e análise de imagens para o auxílio nas aulas de anatomia da madeira. A plataforma permite o envio, processamento e análise de imagens por meio de filtros e geração de gráficos e tabelas para interpretação dos resultados. O sistema foi estruturado e dividido em duas camadas: backend e frontend. A camada backend foi desenvolvida utilizando a linguagem Python com auxílio do framework FastAPI, com persistência das informações utilizando PostgreSQL e mecanismos de segurança usando SHA-256 com salt. A camada frontend foi implementada com HTML, CSS e JavaScript e com auxílio do framework Bootstrap e utilização de SessionStorage e IndexedDB no armazenamento de dados no lado do cliente, garantindo a construção de uma interface responsiva. Como resultado da integração de ambas as camadas e tecnologias a plataforma desenvolvida apresenta potencial de apoio ao ensino e à análise de imagens de madeira.

Palavras-chave: Processamento de Imagens, Aplicação Web, API, Anatomia da Madeira, Ensino.

Lista de ilustrações

Lista de tabelas

Lista de Siglas

API *Application Programming Interface*

ACID *Atomicity, Consistency, Isolation, Durability*

ASGI *Asynchronous Server Gateway Interface*

CSS *Cascading Style Sheets*

HTML *Hypertext Markup Language*

HTTP *Hypertext Transfer Protocol*

ID Identificador

JPEG *Joint Photographic Experts Group*

JWT *JSON Web Token*

OpenCV *Open Source Computer Vision Library*

PNG *Portable Network Graphics*

REST *Representational State Transfer*

SAIM Sistema de Análise de Imagens de Madeira

SGBD Sistema Gerencial de Banco de Dados

SHA-256 *Secure Hash Algorithm* 256-bit

UX *User Experience*

UI U*ser Interface*

Sumário

Capítulo 1

Introdução

Com os avanços na área da computação verificou-se um gradativo e significativo processo de automatização das análises laboratoriais nas diferentes áreas da ciência. Os resultados imediatos podem ser comprovados pela redução do tempo de realização e aumento no número das análises, além da melhoria da qualidade das informações e do ganho significativo em precisão. Diversos laboratórios particulares e de Universidades utilizam a computação, mais especificamente o processamento digital de imagens para automatizar e acelerar as análises.

Nesse contexto, os laboratórios que desenvolvem pesquisas nas áreas de biologia, anatomia, ciência e tecnologia da madeira (BURGER; RICHTER, 1991), dentre outros carecem de softwares específicos para auxiliar na melhoria da precisão e também na redução do tempo de análise, permitindo a automação do processamento e análises de imagens.

Existem diversos estudos na área da anatomia da madeira voltados à automação da análise e quantificação de amostras por meio do processamento digital de imagens (RIBEIRO; CARNEIRO-TOMAZELLO; FILHO, 1999; RIBEIRO; FILHO; CRUVINEL, 2001; RIBEIRO et al., 2005). As empresas que desenvolvem equipamentos de microscopia já disponibilizam, juntamente com o equipamento, um *software* para análise das imagens obtidas por microscopia. Apesar do *software* ter itens que auxiliam nas análises de imagens de madeira, esses softwares precisam ser mais genéricos por poderem ser utilizados para analisar quaisquer imagens que sejam obtidas pelo equipamento. Com isso, análises específicas voltadas para imagens de madeira acabam não sendo priorizadas e muitas vezes não são implementadas.

Dentro da Universidade nas disciplinas relacionadas à Anatomia de Madeira, especialmente em cursos da área de Ciências Florestais, os discentes aprendem técnicas de análise e mensuração de características anatômicas da madeira. Essas atividades podem

envolver medições manuais realizadas com réguas ou escalas presentes nos equipamentos de microscopia. Nesse contexto, o contato com ferramentas computacionais de análise de imagens pode contribuir para aproximar a formação acadêmica das tecnologias utilizadas no mercado de trabalho.

Pensando nisso, Ribeiro (2002) propôs um sistema baseado na metodologia utilizada na universidade para realizar análises, por meio de imagens de microscopia, auxiliadas por técnicas de processamento digital de imagens (GONZALEZ; WOODS, 2018). Foram realizados diversos estudos dos algoritmos e técnicas (RIBEIRO; CARNEIRO-TOMAZELLO; FILHO, 1999) (RIBEIRO; FILHO; CRUVINEL, 2001) (RIBEIRO et al., 2003) (RIBEIRO et al., 2005) para a proposição desse sistema e da metodologia descrita em Ribeiro (2002).

1.1 Objetivos

O objetivo deste projeto é a consolidação e evolução da plataforma voltada ao apoio das disciplinas de Anatomia da Madeira e Dendrologia, por meio do aprimoramento de suas funcionalidades, finalização do desenvolvimento e integração eficiente entre seus componentes. A proposta contempla a articulação entre módulos de processamento de imagens, *backend* e interface de usuário, garantindo um fluxo coeso de dados e operações. Além disso, permitir aos discentes a experiência da utilização de uma ferramenta especializada em analisar imagens de madeira e, que posteriormente, poderá ser utilizada em seu dia a dia de pesquisas e no mercado de trabalho.

1.2 Justificativa

A necessidade deste projeto está relacionada às limitações do Sistema de Análise de Imagens de Imagens de Madeira (SAIM), desenvolvido por (RIBEIRO, 2002), que utiliza tecnologias destinadas a ambientes Windows legados e não recebeu atualizações posteriores. Essas limitações dificultam sua utilização em ambientes modernos e restringem o acesso dos discentes a ferramenta. Dessa forma, a modernização do sistema para uma plataforma *web* visa ampliar sua acessibilidade e sua utilização nas atividades acadêmicas relacionas à anatomia da madeira.

1.3 Metodologia

Nesse projeto foram analisados os algoritmos propostos por Ribeiro (2002) no sistema SAIM e, a partir dessas análises, foram selecionadas abordagens atualizadas com o objetivo de melhorar o desempenho e reduzir o custo computacional dos métodos originalmente propostos.

A plataforma foi implementada na linguagem de programação Python e disponibilizada por meio de um sistema WEB, permitindo assim, que seja utilizada em qualquer lugar com conexão com a Internet e sem a necessidade de nenhuma instalação por parte do discente independente do sistema operacional utilizado. Nesse contexto, foram definidos requisitos funcionais e não funcionais, incluindo autenticação de usuários, envio e gerenciamento de imagens, aplicação de filtros de processamento, geração de gráficos e armazenamento dos resultados obtidos.

Para o desenvolvimento da solução, adotou-se a separação entre a interface de acesso e a camada de processamento, estruturando o sistema no modelo de cliente-servidor. Dessa forma, um servidor *web* é responsável pela interface com o usuário, enquanto outro componente realiza o processamento dos algoritmos e o gerenciamento dos dados, provendo melhor organização, escalabilidade e manutenibilidade da aplicação.

No desenvolvimento da interface *Web* serão utilizadas folhas de estilo (CSS), além de conceitos de Interface de Usuários (UI) e Experiência de Usuário (UX). O desenvolvimento da parte de análise das imagens será feito na linguagem Python, com o uso da biblioteca OpenCV, sendo integrado a um Sistema Gerenciador de Banco de Dados para o armazenamento das informações e posterior utilização pelo usuário.

Para validação operacional da plataforma, foi utilizado um conjunto de imagens previamente disponíveis, adequado ao contexto de anatomia da madeira, permitindo a verificação prática das funcionalidades implementadas. A estratégia de testes concentrou-se na execução do fluxo do sistema, envolvendo autenticação, envio de imagens, aplicação dos métodos de processamento, salvamento dos resultados e reutilização das imagens processadas em diferentes operações.

Como critérios de validação, foram considerados o funcionamento correto das funcionalidades implementadas, a integração entre os módulos do sistema e a estabilidade geral da aplicação. Como etapa futura, prevê-se a realização de testes e validações da plataforma por especialistas da área, com o objetivo de avaliar sua aplicabilidade em contexto acadêmico e de pesquisa. Para isso, poderão ser conduzidos estudos com discentes das disciplinas relacionadas, mediante aprovação do Comitê de Ética, possibilitando o acompanhamento por meio de grupos de testes e controle. Adicionalmente, espera-se estabelecer parcerias com instituições e pesquisadores da área de Ciências Florestais para obtenção de *feedback* especializado e identificação de melhorias e novas funcionalidades para a evolução da plataforma.

1.4 Organização da Monografia

Esta monografia está estruturada em quatro capítulos.

1. Capítulo 1 é apresentada a introdução do trabalho, incluindo contexto, os objetivos, justificativa e metodologia adotada para o desenvolvimento da plataforma.

2. Capítulo 2 é abordada a fundamentação teórica, contemplando as tecnologias utilizadas, os conceitos de processamento digital de imagens e análise de trabalhos correlatos.

3. Capítulo 3 é descrito o desenvolvimento da plataforma, detalhando suas arquitetura, implementação e resultados obtidos.

4. Capítulo 4 são apresentadas as conclusões do trabalho, bem como as considerações finais e proposta de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Nesse capítulo será apresentado o referencial teórico sobre os temas que compõem esse trabalho.

2.1 Tecnologias

Nessa seção serão apresentadas as tecnologias utilizadas no desenvolvimento da plataforma proposta neste trabalho. São descritas as principais ferramentas, bibliotecas e arquitetura adotadas para a implementação do sistema.

2.1.1 Python

Python[1] é uma linguagem de programação interpretada e de alto nível, criada pelo matemático e programador holandês Guido van Rossum e lançada oficialmente em 1991 pela Python Software Foundation, sendo acessível e de fácil entendimento e uso tanto para programadores experientes e novatos (ROSSUM; DRAKE, 2009).

A linguagem conta com uma sintaxe clara e de fácil legibilidade, execução, entendimento e desenvolvimento simples sendo possível a construção de *softwares* mais simples ou complexos utilizando múltiplos paradigmas, incluindo programação funcional, programação imperativa e programação orientada a objetos (Python Core Team, 2019).

Com uma ampla comunidade de desenvolvedores e vasta quantidade de bibliotecas terceiras disponíveis para uso, como Numpy, Math, OpenCV, que corroboram com o desenvolvimento de ações diversas e construção de aplicações, desde as mais simples até as mais complexas.

2.1.2 FastAPI

FastAPI é um *framework web* moderno e veloz, desenvolvido em Python e lançado em 2018 pelo seu criador Sebastián Ramírez. Desde o início o *framework* está acessível e

[1] Mais informações no site <https://www.python.org/downloads/>

disponível aos demais desenvolvedores Python.

O FastAPI[2] permite o desenvolvimento de APIs Python possibilitando aplicações escaláveis, com alta *performance* para diferentes cenários. Para atingir tais resultados o *framework* se baseia no protocolo ASGI, microframework Starlette e a biblioteca Pydantic.

Uma das vantagens apresentadas pelo *framework* é a geração de documentação de forma automática e interativa em tempo real se utilizando do padrão OpenAPI. Essa documentação é acessada por ferramentas como Swagger UI e ReDoc por uma interface de usuário *web*, que possibilitam a exploração de todos os endpoints da aplicação e realização de testes em cada endpoint individualmente (RAMÍREZ, 2018).

2.1.3 OpenCV

A OpenCV (*Open Source Computer Vision Library*) é uma biblioteca de *software* para visão computacional de código aberto e livre para modificações criada pela Intel Corporation em 2000.

A biblioteca fornece um abrangente conjunto de funções e algoritmos prontos para uso, abrangendo técnicas desde detecção de bordas até a análise de movimento. Com uma alta abrangência de técnicas a biblioteca se torna adequada para projetos de diferentes tamanhos, escalas e complexidades (BRADSKI, 2000).

A OpenCV[3], além de sua vasta gama de funções apresenta como diferencial sua eficiência, desempenho e otimização, decorrentes de sua implementação original, que utiliza linguagens de baixo nível sendo elas o C/C++. Outros benefícios se encontram na sua integração com outras bibliotecas Python de forma fácil e documentada, reforçando seu uso na visão computacional (OpenCV, 2025).

2.1.4 NumPy

A NumPy (*Numerical Python*)[4] é uma biblioteca de código aberto e livre voltada para a computação numérica contando com uma grande coleção de funções matemáticas em Python. Lançada em 2005 pelo cientista de dados e desenvolvedor Travis Oliphant, a NumPy incorporou e modificou as bibliotecas matemáticas Numeric e Numarray para suportar o processamento de *arrays* e matrizes multidimensionais (HARRIS et al., 2020).

A biblioteca fornece uma ampla gama de funções para cálculos matemáticos, manipulação de *arrays* e matrizes e apresenta como diferencial seu desempenho pela sua implementação em grande parte utilizando a linguagem de programação C de baixo nível, o que garante sua rapidez de execução (OLIPHANT, 2006).

[2] Mais informações no site <https://fastapi.tiangolo.com/>
[3] Link da biblioteca OpenCV <https://opencv.org/releases/>
[4] Link da biblioteca NumPy <https://numpy.org/install/>

2.1.5 Banco de Dados

Banco de dados é uma coleção de dados de forma organizada e que requer um Sistema Gerencial de Banco de Dados (SGBD) para realizar seu controle e gerenciamento (ORACLE, 2020), além de ser responsável por garantir integridade, consistência e disponibilidade das informações. Existem diferentes modelos de banco de dados que entregam diferentes características e benefícios para as mais diversas necessidades para o armazenamento de dados. De forma geral os bancos de dados podem ser definidos em banco de dados relacionais ou banco de dados não relacionais.

Para este trabalho foi adotado o uso de um banco de dados relacional, implementado por meio do PostgreSQL[5], um sistema gerenciador de banco de dados de código aberto, reconhecido por sua fácil escalabilidade, flexibilidade e sua confiabilidade.

O PostgreSQL se destaca na sua capacidade em lidar com um alto volume de informações e operações e mantendo a consistência dos dados. Além disso o PostgreSQL oferece suporte a transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade) o que garante a segurança e integridade nas operações de escrita e leitura (POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2022).

2.1.6 REST

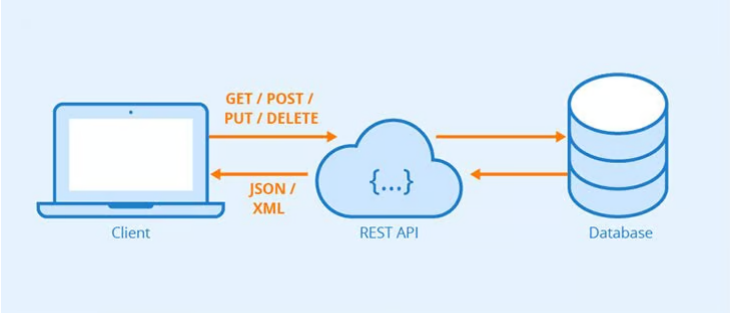
O *Representational State Transfer* (REST) é um modelo arquitetural definido por Roy Fielding em 2000 que possui como base um conjunto de seis princípios que garantem a simplicidade, escalabilidade e interoperabilidade entre sistemas distribuídos.

Os princípios do padrão REST são: (i) *Client-Server*, que estabelece uma arquitetura clara de separação entre cliente e servidor; (ii) *Stateless*, no qual cada requisição deve conter todas as informações para processamento e sem dependência de estado ou requisições anteriores; (iii) *Cacheable*, que se utiliza de cache para a redução de requisições e melhora na eficiência do sistema; (iv) *Uniform Interface* (Interface Uniforme) padronizando a comunicação entre o servidor e o cliente se utilizando operações HTTP; (v) *Layered System* (Sistema em Camadas) que garante a modularidade e segurança; (vi) *Code on Demand* (Código sob demanda) princípio opcional que possibilita o envio de código executável diretamente do servidor.

Quando uma API ou serviço *web* adotam a arquitetura REST é chamada de *RESTful*. Esse tipo de API utiliza os métodos do protocolo HTTP sendo eles GET para consulta de informações, POST para criação e armazenamento de novos recursos, PUT voltado à atualização e substituição de informações existentes, DELETE para exclusão de dados informados (Figura 1).

[5] Mais informações no site <https://www.postgresql.org/download/>

Figura 1 – Representação API REST.



Fonte: Seobility (2023).

2.2 Processamento Digital de Imagens

O processamento digital de imagens engloba diversas técnicas direcionadas para a análise, transformação e interpretação de imagens digitais, as quais são representações discretizadas de imagens em coordenadas espaciais e níveis de intensidade de brilho, capturadas por meio de sensores de imageamento (GONZALEZ; WOODS, 2009). O objetivo do processamento digital de imagens é possibilitar a manipulação e análise de imagens para a extração de informações relevantes, alimentar sistemas e auxiliar a análise e interpretação humana.

Entre as diversas técnicas de processamento de imagens, destacam-se o pré-processamento, a segmentação, as transformações geométricas e a extração de características que podem ser aplicadas isoladamente ou em conjunto (GONZALEZ; WOODS, 2018).

2.3 Trabalhos Correlatos

2.3.1 Sistemas para Análise de Imagens de Madeira

No trabalho de Ribeiro (2002) foi realizada uma pesquisa que demonstra, por meio de técnicas do processamento digital de imagens, um método complementar de análise da qualidade de madeira que cria um vetor de parâmetros que indicam qualidade tendo como base a caracterização de aspectos microscópicos e sub-microscópicos. Esse trabalho deu origem ao Sistema de Análise de Imagens de Madeira (SAIM).

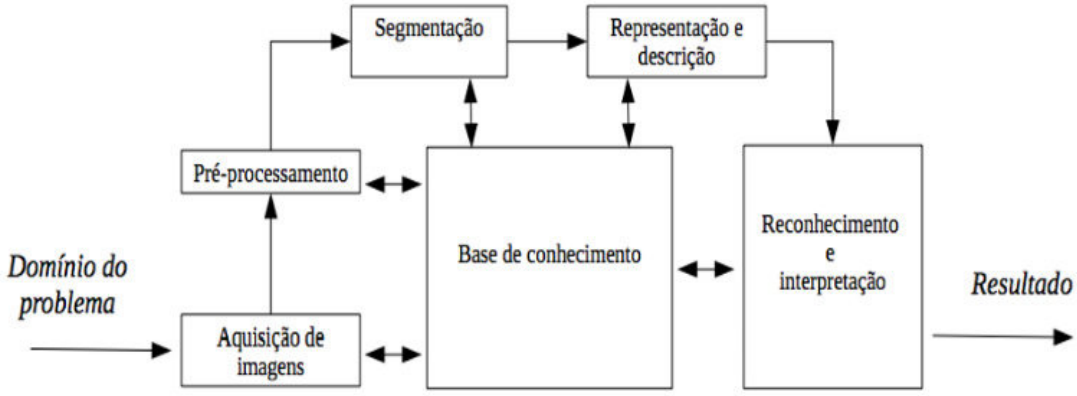
Apesar de sua relevância, o sistema apresenta algumas limitações relacionadas ao contexto tecnológico distinto que foi desenvolvido, evidenciando a necessidade de soluções mais modernas, acessíveis e integradas, como a proposta neste trabalho.

2.3.2 Aplicações de Visão Computacional em Análise Anatômica e Agronômica

O trabalho de Machado (2023) aplica técnicas de visão computacional para identificação de padrões em folhas de café, evidenciando o potencial dessas abordagens na automação de análises visuais. Embora não seja específico para madeira, o trabalho contribui metodologicamente ao demonstrar o uso de algoritmos de segmentação, filtragem e extração de características, que também são aplicáveis à análise anatômica da madeira.

Além disso, os conceitos fundamentais apresentados no livro Processamento Digital de Imagens dos autores Gonzalez e Woods (2009), fornecem a base teórica para o processamento digital de imagens, incluindo técnicas essenciais utilizadas neste trabalho, como filtragem, transformação e análise de imagens. (Figura 2).

Figura 2 – Passos fundamentais em processamento de imagens.



Fonte: Gonzalez e Woods (2009).

2.3.3 Arquiteturas de Sistemas Web e Integração de Componentes

Como suporte à implementação da plataforma proposta, destacam-se trabalhos relacionados à arquitetura de sistemas *web* e integração entre componentes. O trabalho de Nascimento (2025) apresenta o desenvolvimento de um *backend* com ênfase na modelagem de dados, utilizando arquitetura em camadas, API *Restful* juntamente com PostgreSQL, contribuindo para organização e consistência das informações do sistema.

De forma complementar, o trabalho de Cardoso (2025) aborda a implementação de *backend* com foco na integração entre *frontend* e *backend*, utilizando API´s e requisições HTTP para garantir a comunicação eficiente entre os componentes do sistema.

Esses trabalhos são relevantes não apenas pela implementação tecnológica, mas também por evidenciarem boas práticas de arquitetura de *software*, separação de responsabili-

dades e integração entre módulos. Tais aspectos são fundamentais para o desenvolvimento da plataforma, que depende de uma comunicação eficiente entre todas as partes.

Capítulo 3

Desenvolvimento e Resultados

Nesse capítulo serão apresentadas as etapas de desenvolvimento e funcionamento do sistema, sua estrutura e por fim os resultados obtidos.

3.1 Visão Geral

Para o desenvolvimento do sistema foram criados dois projetos independentes que se comunicam entre si, sendo eles o *backend* que fica responsável pela conexão com o banco de dados e implementação das regras de negócio; e o *frontend*, sendo responsável pelas interfaces de usuário e comunicação com o *backend*.

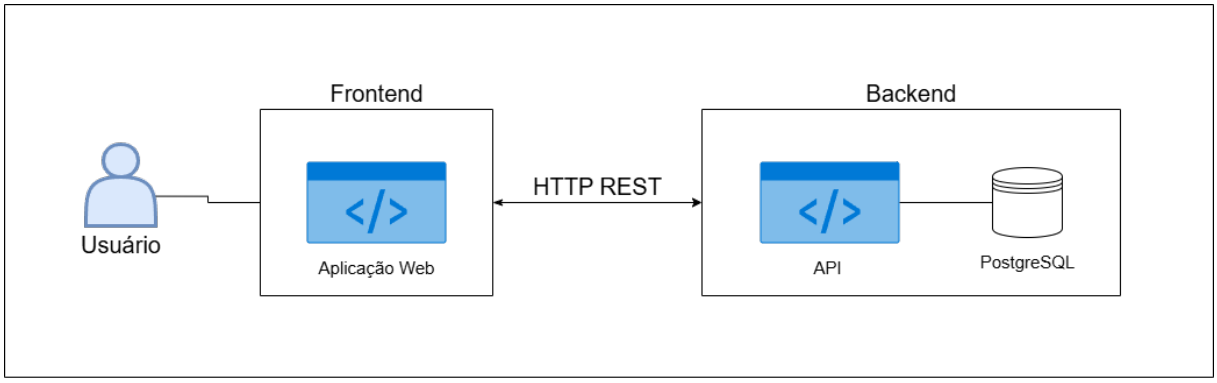
No *backend* foi utilizado o *framework* FastAPI, juntamente com a linguagem de programação Python, escolhidos por sua simplicidade, escalabilidade e facilidade de integração com outras bibliotecas Python. Essa combinação de fatores resultou em uma estrutura de *backend* eficiente, escalável e com comunicação no modelo cliente-servidor.

O *frontend* foi desenvolvido utilizando as tecnologias fundamentais para o desenvolvimento *web*, sendo elas *Hypertext Markup Language* (HTML), responsável pelo esqueleto das páginas; CSS (*Cascading Style Sheets*), para estilização e organização visual; Javascript, utilizado para viabilizar a comunicação com o *backend* e proporcionar interações pelo sistema, também foi usado o IndexedDB para armazenamento das imagens processadas para reutilização rápida. Além disso, adotou-se o Bootstrap, um *framework* CSS para padronização visual e facilitador na construção das páginas *web*.

Para a camada de persistência das informações, optou-se pelo sistema gerenciador de banco de dados relacional PostgreSQL, que apresenta fácil utilização, escalabilidade e robustez. Sua adoção e integração ao *backend* garantiu operações de escrita e leitura eficientes e seguras dos dados.

A combinação e integração das tecnologias utilizadas resultou em um sistema composto por diferentes camadas, com comunicação entre componentes, conforme apresentado na Figura 3.

Figura 3 – Estrutura Comunicação do Sistema



Fonte: Próprio Autor.

3.2 Desenvolvimento

O desenvolvimento do *backend* foi conduzido a partir das técnicas apresentadas na Seção 3.1. A explicação será dividida em três partes fundamentais: Banco de dados e sua modelagem; Segurança e criptografia; Ambiente e documentação.

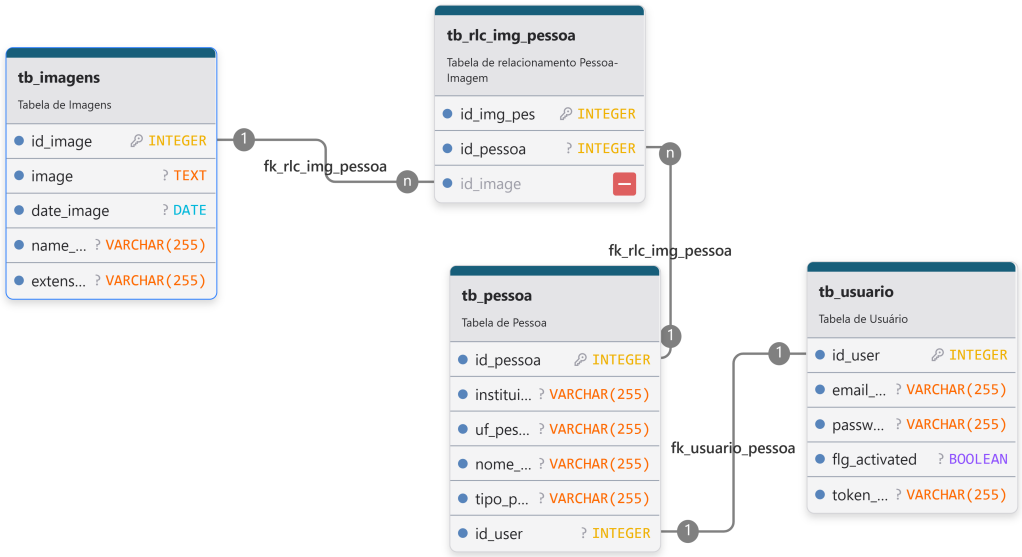
3.2.1 Banco de Dados

Para o correto funcionamento do sistema, foi utilizado o PostgreSQL, no qual foi realizada a modelagem do banco de dados com base nas telas já existentes no trabalho de Ribeiro (2002). O objetivo da modelagem foi garantir a consistência entre as informações e a minimização de informações redundantes.

Para todo o projeto foi definido um único schema no banco de dados para todo o sistema e utilização dos usuários, representados por alunos e professores. Após análise do sistemas proposto por Ribeiro (2002), mapeou-se as imagens como a principal entidade de informação utilizada no processamento. Com base nessa informação e visando a implementação de um sistema *web* moderno, foram desenhados e estruturadas quatro tabelas principais (Figura 4): (i) tabela imagens, para o armazenamento das imagens a serem processadas, convertidas em string no formato Base64, juntamente com informações auxiliares da imagem; (ii) tabela de pessoa, destinada ao armazenamento de informações de usuários, sendo eles alunos ou professores, contendo informações básicas e institucionais; (iii) tabela de usuário, voltada para gerenciamento de *login* e acesso ao sistema; (iv) tabela de relacionamento entre pessoa e imagem para identificação de quais usuários realizaram o *upload* e posterior processamento de cada imagem.

Para todas as tabelas foram definidos campos de identificação única (IDs), utilizados para estabelecer os relacionamentos entre as tabelas e sua integridade referencial. Essa modelagem também possibilitou a identificação da cardinalidade entre as tabelas, sendo eles: (i) 1:1 entre usuário e pessoa, em que cada usuário tem somente uma informação

Figura 4 – Diagrama tabelas banco de dados.



Fonte: Próprio Autor.

relacionada de pessoa; (ii) 1:N entre pessoa e imagem, em que uma pessoa pode ter realizado vários *uploads* de imagens; (iii) 1:1 imagem e pessoa-imagem, em que cada ID de imagem pode estar relacionada a uma única pessoa.

3.2.2 Criptografia

Com a disponibilização do sistema via *web*, surgiu a necessidade de autenticação de usuários e, consequentemente, a necessidade de proteção das senhas dos usuários. Para assegurar maior segurança para a senha como informação sensível foi utilizada função *hash* SHA-256 que garante que a senha não seja decifrada diretamente em texto puro.

Além disso, foi realizada a adição de um *salt*, gerado previamente de forma aleatória e incorporado ao processo de geração do *hash* com o algoritmo SHA-256. Esse valor é armazenado fora do sistema e reutilizado durante a autenticação. A combinação do *salt* com o algoritmo SHA-256 garante que mesmo em caso de vazamento de dados, a informação da senha do usuário permanece segura.

Com o uso do algoritmo de função *hash* SHA-256 em conjunto com o *salt*, o sistema nunca armazena a senha dos usuários na forma de texto puro (Figura 5). Durante o processo de autenticação, após o usuário informar a senha, o sistema realiza a geração do *hash* da senha do usuário novamente utilizando o mesmo *salt* previamente associado, e com o resultado gerado é realizada a comparação com o *hash* já armazenado em banco. Essa abordagem garante a segurança do processo de *login* no sistema sem expor a informações sensíveis durante o processo.

Embora essa abordagem represente um avanço significativo em relação ao armazena-

mento de senhas dos usuários da aplicação, destaca-se que algoritmos específicos para *hash* de senhas como *bycrpt*, *scrypt* e *Argon2*, são mais adequados para esse propósito, pois incluem mecanismos adicionais de segurança, como resistência a ataques de força bruta e maior custo computacional configurável.

Figura 5 – Encriptação de senha dos usuários.

```python
async def encript_password_user(password: str):
    '''
    Method to encript user password
    '''
    salt = helper.get_salt_data()
    hash_object = hashlib.sha256()
    password_bytes = password.encode('utf-8')
    salt_bytes = salt.encode('utf-8')
    hash_object.update(salt_bytes+password_bytes)
    hash_hex = hash_object.hexdigest()
    return hash_hex
```

Fonte: Próprio Autor.

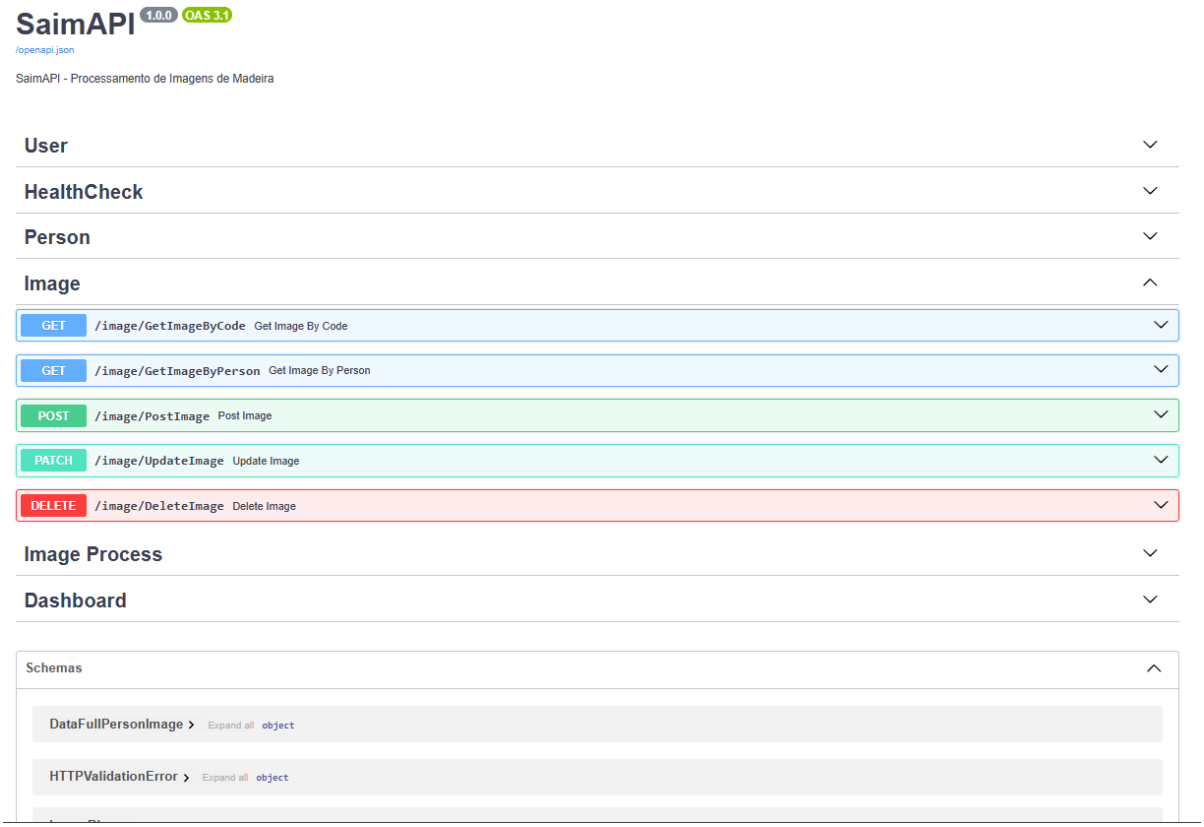
3.2.3 Ambiente e Documentação

Para a realização de testes e documentação das rotas do projeto, que são consumidas pela camada do cliente, foi utilizado um conjunto de ferramentas da Swagger, baseado nas especificações OpenAPI. Essa abordagem permite a geração de uma documentação interativa que possibilita a interação e exploração pelo próprio navegador através do Swagger UI, disponibilizado após a execução do *backend* no endereço http://127.0.0.1:8000/docs. A interface do Swagger UI e as rotas criadas para o projeto podem ser observadas na Figura 6.

Além de fornecer uma documentação interativa, o Swagger UI organiza as rotas em blocos de grupos, facilitando a compreensão da API, conforme ilustrado na Figura 7. Cada rota é apresentada de forma intuitiva e de fácil legibilidade, separando os métodos HTTP por cores: azul - Método POST, verde - Método GET, verde-água - Método PATCH e vermelho - Método DELETE. Ademais, cada rota possui uma descrição individual bem como ao selecionar a rota podemos observar os dados esperados na requisição (*request*) e os dados de retorno da requisição (*response*), como observado nas Figura 7 e 8.

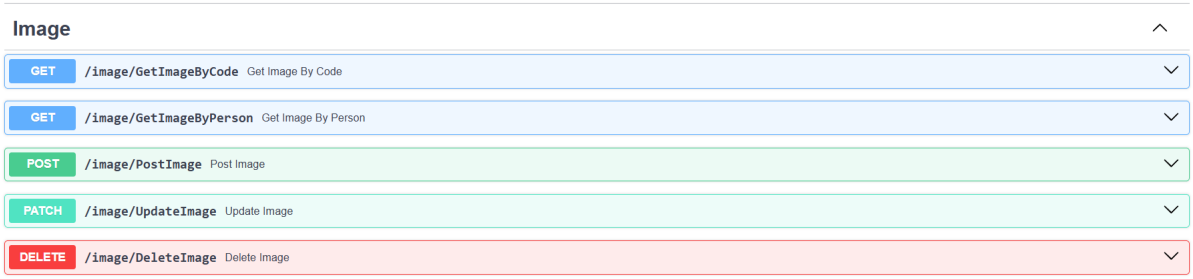
Para padronização das respostas do *backend*, foi criado um middleware responsável pelo tratamento de todas as respostas de requisições, conforme Figura 9. Esse mecanismo garante sempre a mesma estrutura de resposta, incluindo o status de sucesso da requisição, os dados do processo executado na rota e, em casos de erro, a mensagem detalhada do erro ocorrido.

Figura 6 – Swagger SAIMAPI



Fonte: Próprio Autor.

Figura 7 – Rotas de imagem SAIMAPI.

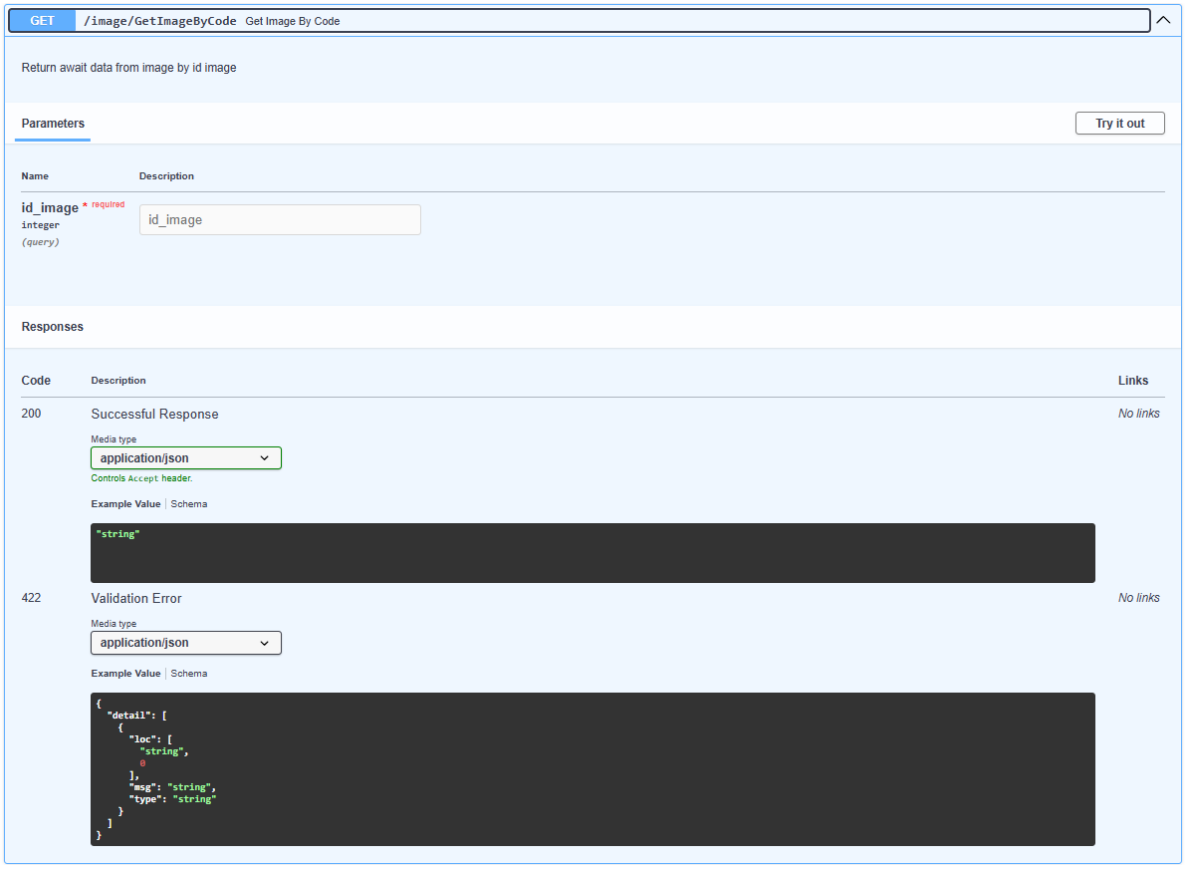


Fonte: Próprio Autor.

3.3 Frontend

O desenvolvimento do *Frontend* foi conduzido a partir das técnicas apresentadas na Seção 3.1. A explicação será dividida em duas partes fundamentais: Armazenamento no *Client-Side*; Utilização do Sistema.

Figura 8 – Detalhes request/response de rota de imagem SAIMAPI.



Fonte: Próprio Autor.

Figura 9 – Estrutura de retorno SAIMAPI.

```python
class SaimApiResponse:
    async def create_response(self, success: Optional[bool] = False, data: Optional[Any] = None, message: Optional[str] = None):
        response = {
            "data": data,
            "message": message,
            "success": success,
        }
        return response

    async def create_error_response(self, message: str):
        raise HTTPException(status_code=404, detail=message)

saim_api_response = SaimApiResponse()
```

Fonte: Próprio Autor.

3.3.1 Armazenamento no Client-Side

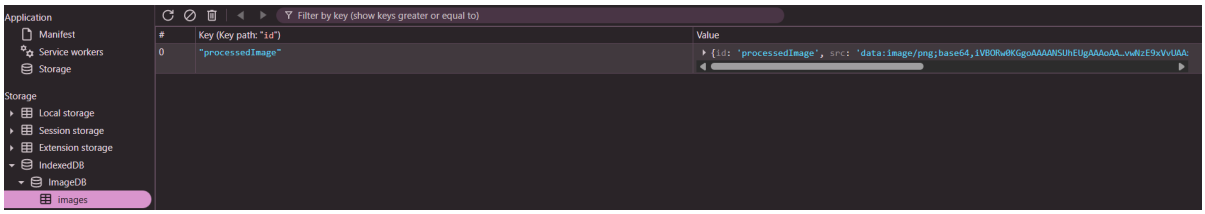
Na camada de *frontend*, identificou-se a necessidade de armazenar algumas informações específicas para cada usuário durante o uso do sistema. Para atender essa necessidade foi utilizado o `SessionStorage`, que permitiu armazenar as informações básicas de usuário e sua autenticação de forma temporária, o que garante que, ao expirar a sessão, os dados do usuário sejam limpos, evitando problemas de inconsistências e exposição de informações.

Além disso, outra solução utilizada foi o IndexedDB, um banco de dados orientado a

objetos executado no *Client-Side*, que possibilita o armazenamento de objetos e recuperação da informação por índices. O IndexedDB possibilitou o armazenamento das imagens que o usuário realizou processamento sem a necessidade de salvamento e *upload* manual (Figura 10).

Essa abordagem permitiu que, após o uso dos métodos disponíveis o resultado seja salvo e reutilizado em outros métodos, garantindo uma fluidez e simplicidade no manuseio da aplicação e de todas suas funções.

Figura 10 – IndexedDB Client-Side.



Fonte: Próprio Autor.

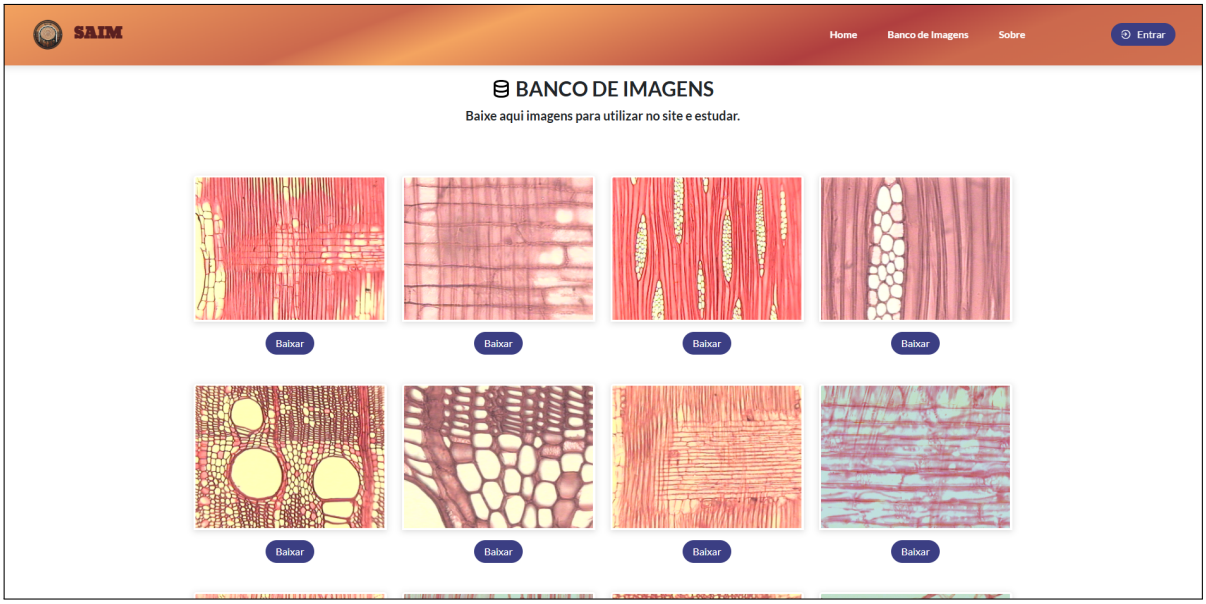
3.3.2 Utilização do Sistema

Ao acessar o sistema é possível visualizar que a aplicação possui quatro seções disponíveis sem autenticação na parte superior da aplicação. São elas `Home`, contendo informações gerais da aplicação; `Sobre`, com detalhes e informações do desenvolvimento; `Banco de Imagens`, que disponibiliza imagens básicas para utilização nos processamentos da aplicação; e, por fim, seção de `Login`, para acessar a aplicação completa. Essas interfaces podem ser observadas nas Figuras 11, 12, 13 e 14.

Figura 11 – Tela Inicial do SAIM.

Fonte: Próprio Autor.

Figura 12 – Banco de Imagens do SAIM.

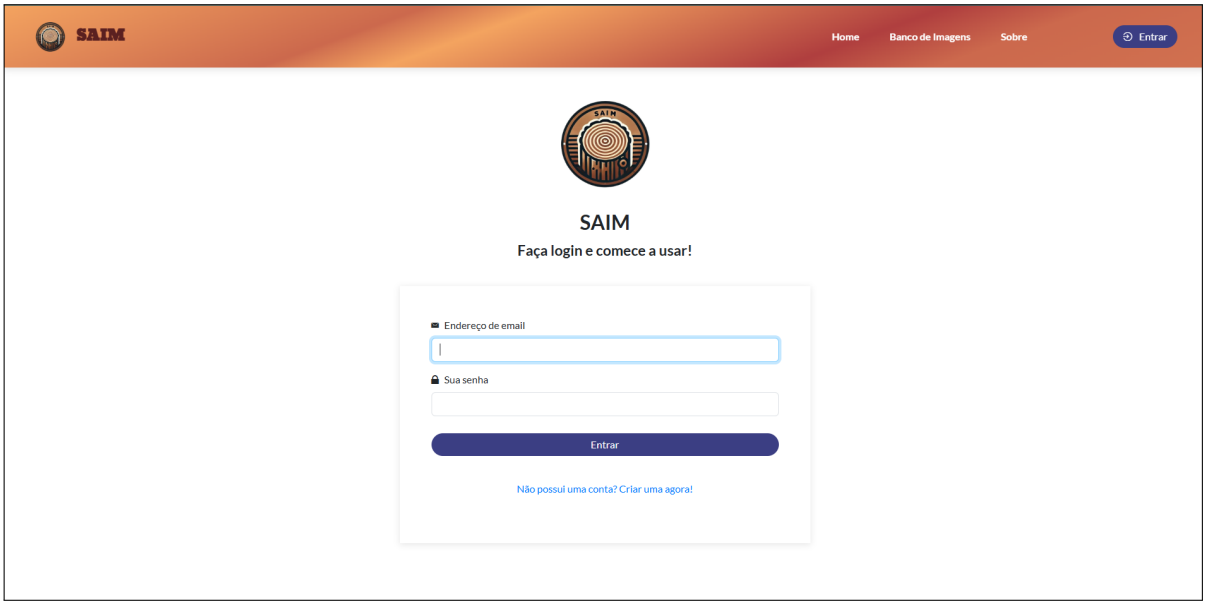


Fonte: Próprio Autor.

Figura 13 – Sobre o SAIM.

Fonte: Próprio Autor.

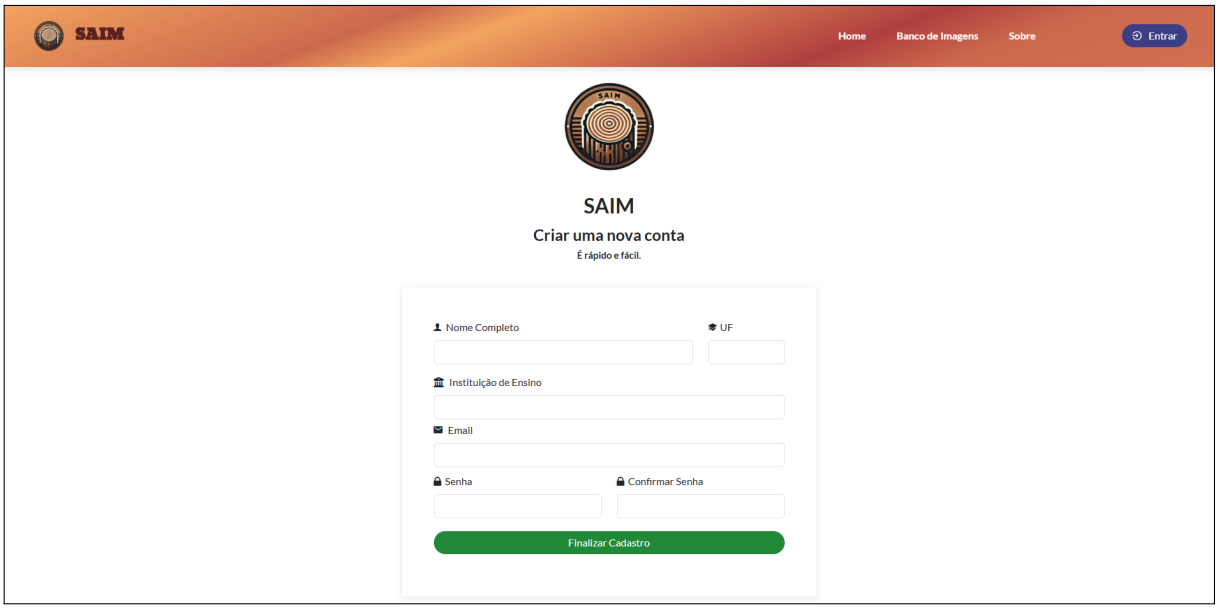
Figura 14 – Login SAIM.



Fonte: Próprio Autor.

Para realização da autenticação na aplicação, o usuário deve realizar o *login* utilizando e-mail e senha ou se for seu primeiro acesso pode realizar o cadastro de uma nova conta, conforme ilustrado na Figura 15. Após a autenticação bem-sucedida, três novas seções ficam disponíveis na barra superior da aplicação, sendo elas Arquivos, Filtros e Madeira.

Figura 15 – Registro novo usuário SAIM.



Fonte: Próprio Autor.

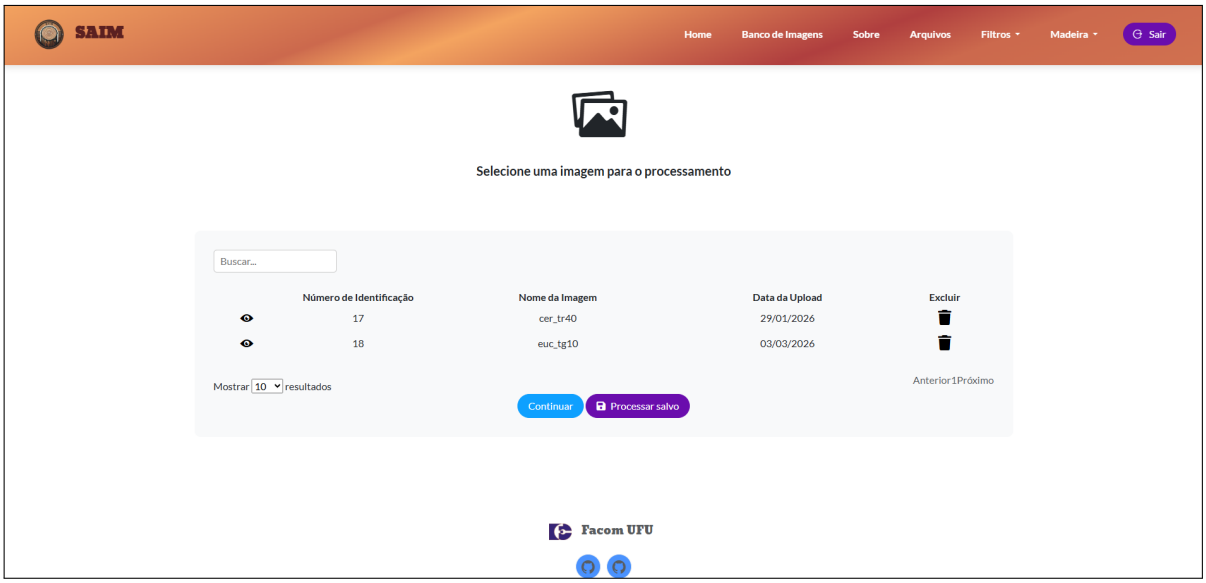
3.3.2.1 Arquivos

A seção Arquivos é responsável pelo envio de novas imagens (*upload*) que o usuário deseja processar. Para isso, basta o usuário selecionar ou arrastar uma imagem no formato PNG ou JPEG e confirmar seu *upload*. Após o procedimento de *upload*, a imagem ficará disponível para utilização em todos os processamentos da aplicação (Figura 16 e 20).

Figura 16 – Upload de nova imagem para processamento.

Fonte: Próprio Autor.

Figura 17 – Imagens de upload disponíveis para processamento.



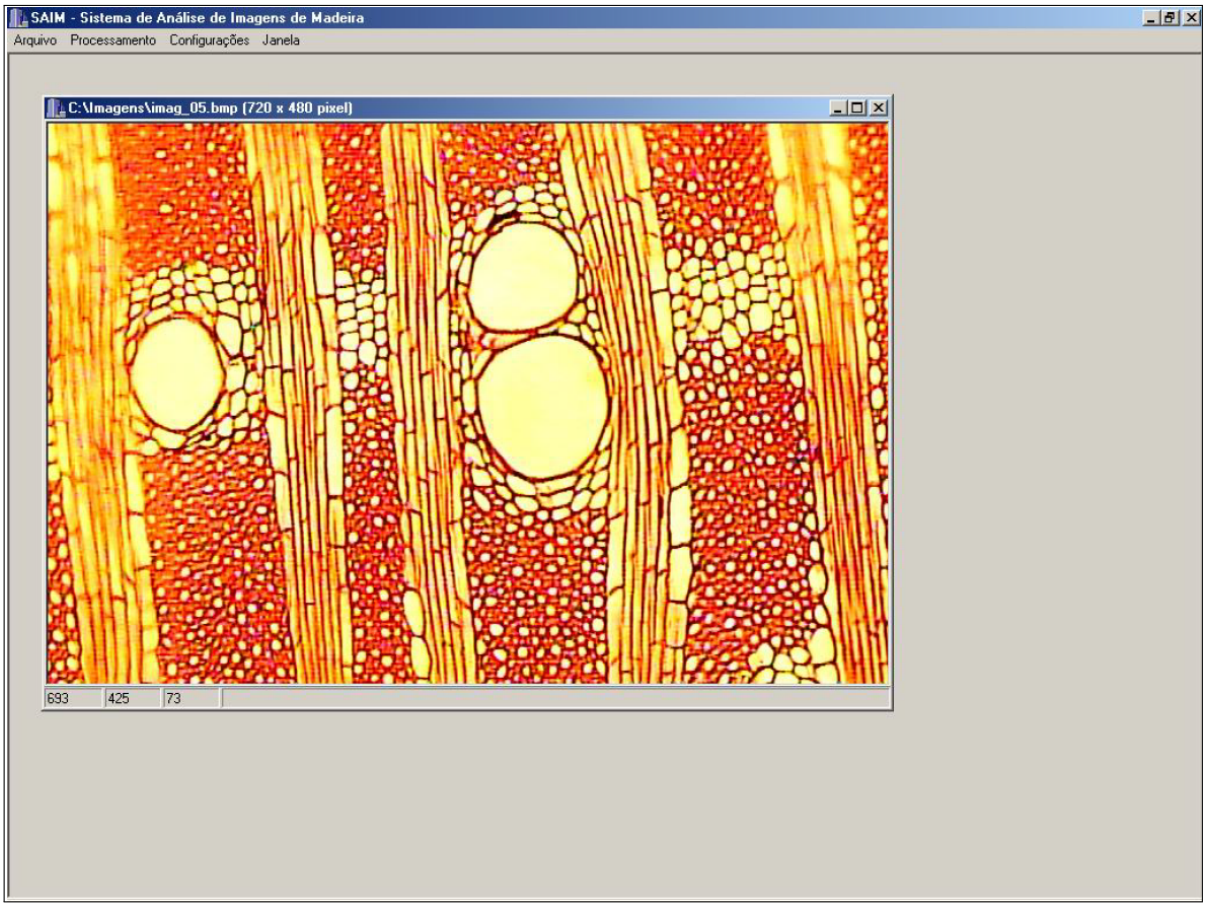
Fonte: Próprio Autor.

3.3.2.2 Filtros

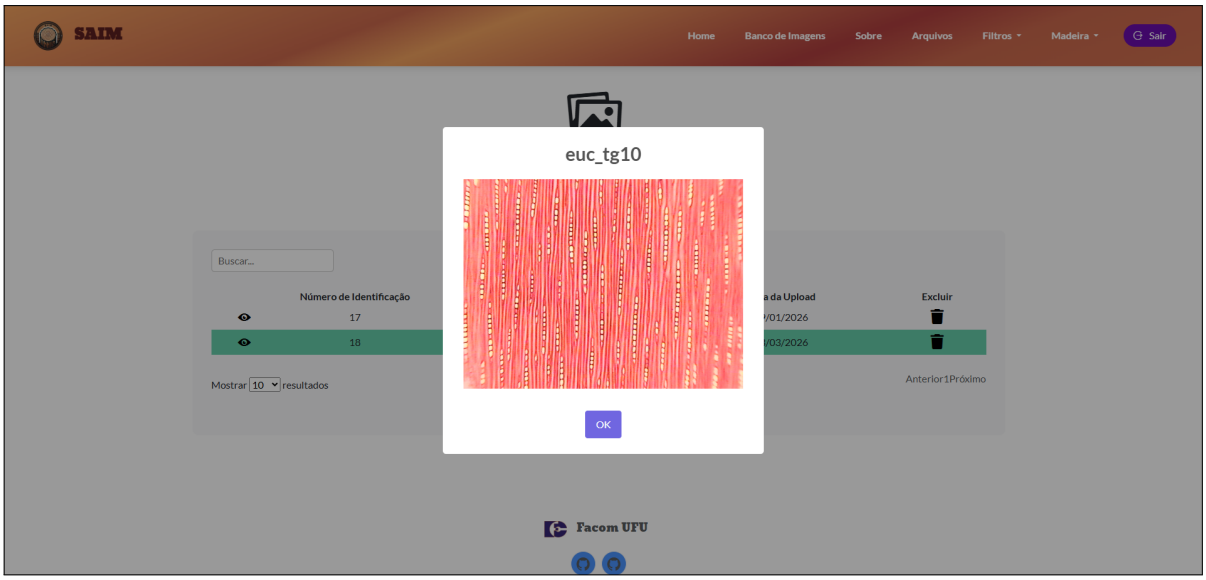
A funcionalidade de aplicação de filtros constitui uma das principais etapas do sistema, sendo responsável pelo processamento das imagens carregadas e selecionadas. Nesta seção, são apresentadas as informações referentes aos filtros implementados, bem como a evolução da interface da aplicação, por meio da comparação entre o sistema original e a solução desenvolvida neste trabalho.

A Figura 18 é apresentada a comparação entre a visualização da imagem selecionada para processamento no sistema original e na solução desenvolvida. Observa-se que, no sistema original a visualização era limitada, enquanto a solução proposta oferece uma interface mais completa, permitindo ao usuário não apenas visualizar a imagem, mas também gerenciar suas imagens enviadas, incluindo a possibilidade de exclusão quando necessário (Figura 19).

Figura 18 – Visualização de detalhes da imagem selecionada.

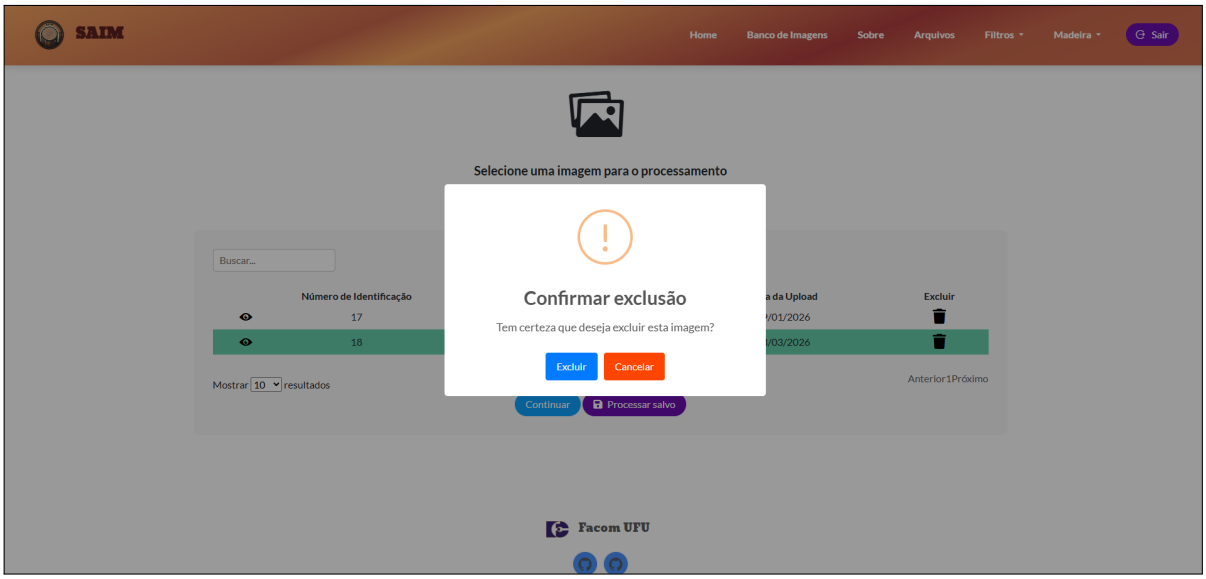


Fonte: Ribeiro (2002).



Fonte: Próprio Autor.

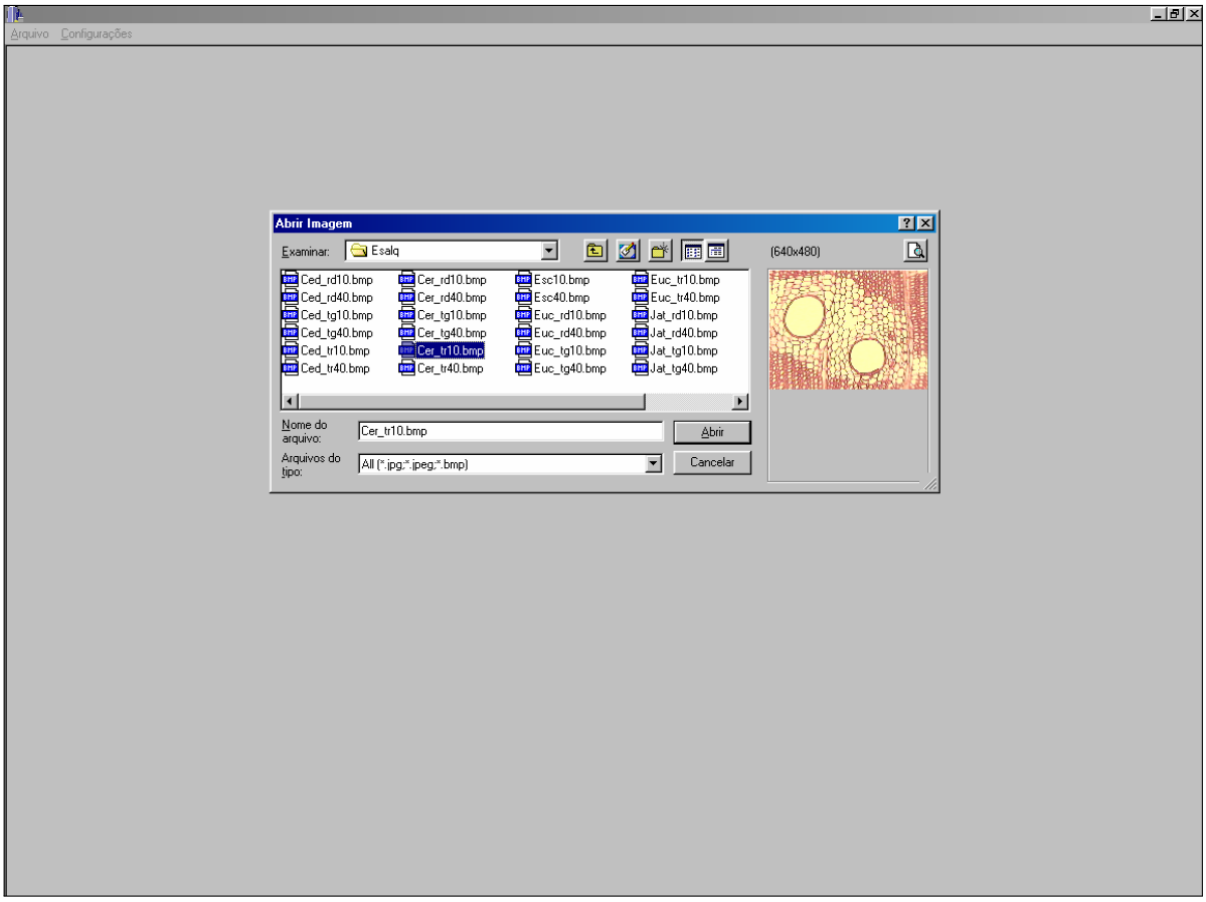
Figura 19 – Confirmação de exclusão de imagem selecionada.



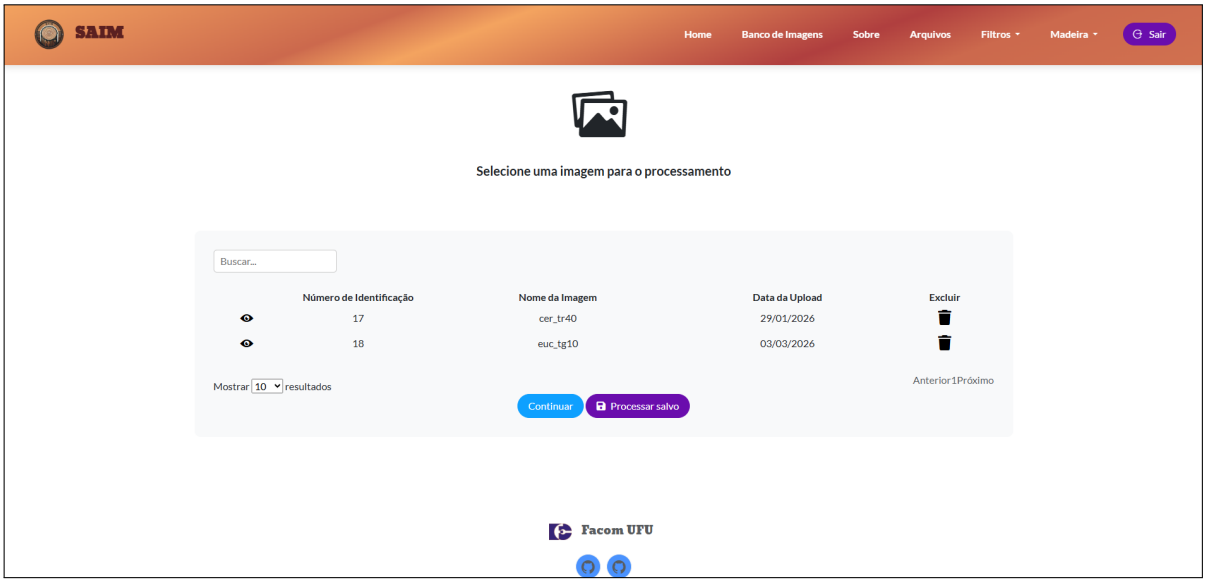
Fonte: Próprio Autor.

De forma geral, os filtros implementados foram organizados de maneira a permitir sua aplicação de forma integrada ao fluxo do sistema. Esses métodos seguem uma estrutura comum, envolvendo a seleção da imagem de entrada, a configuração de parâmetros específicos, processamento por meio da biblioteca OpenCV e a geração da imagem de saída correspondente. A Figura 20 é apresentada a comparação entre a etapa de seleção de imagens no sistema original e na solução desenvolvida.

Figura 20 – Seleção de imagens disponíveis para processamento.



Fonte: Ribeiro (2002).

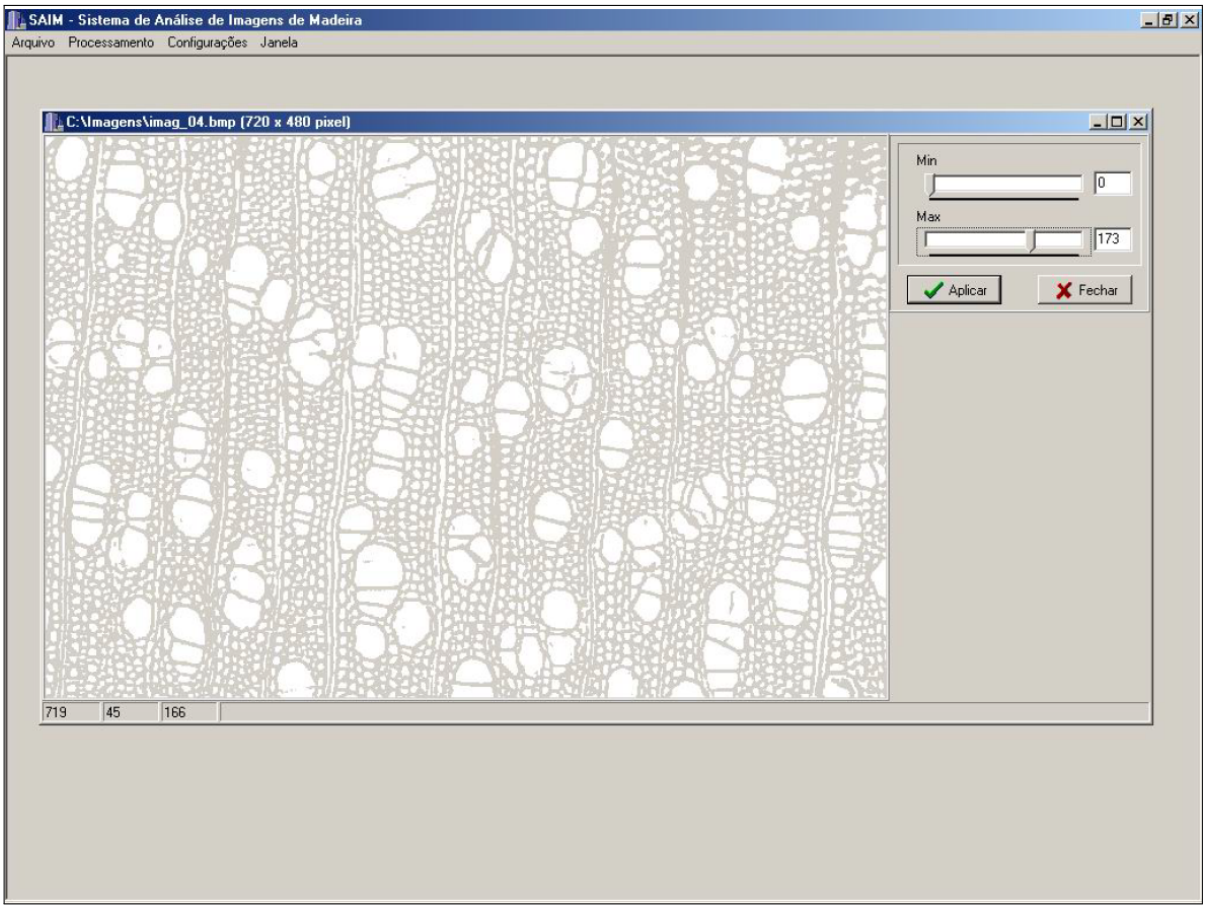


Fonte: Próprio Autor.

Os filtros disponíveis na plataforma incluem operações como Afinamento, Dilatação, Erosão, Laplaciano, Limiar, Média, Sobel (X, Y, XY). De forma geral, esses métodos recebem como entrada uma imagem e parâmetros definidos pelo usuário, como valores de

máximo e minímo, intensidade ou tamanho de *kernel*. Como saída, é gerada uma nova imagem processada, evidenciando características específicas conforme o método aplicado. A Figura 21 é apresentado um exemplo comparativo da aplicação do filtro de limiarização no sistema original e na versão desenvolvida neste trabalho.

Figura 21 – Tela filtro de Limiarização.



Fonte: Ribeiro (2002).

Fonte: Próprio Autor.

Todas as telas de filtro seguem um *layout* consistente e padronizado, dividido em duas áreas principais: À esquerda, a imagem selecionada para o processo é exibida com possibilidade de zoom da imagem para uma melhor análise; À direita, encontram-se os filtros e

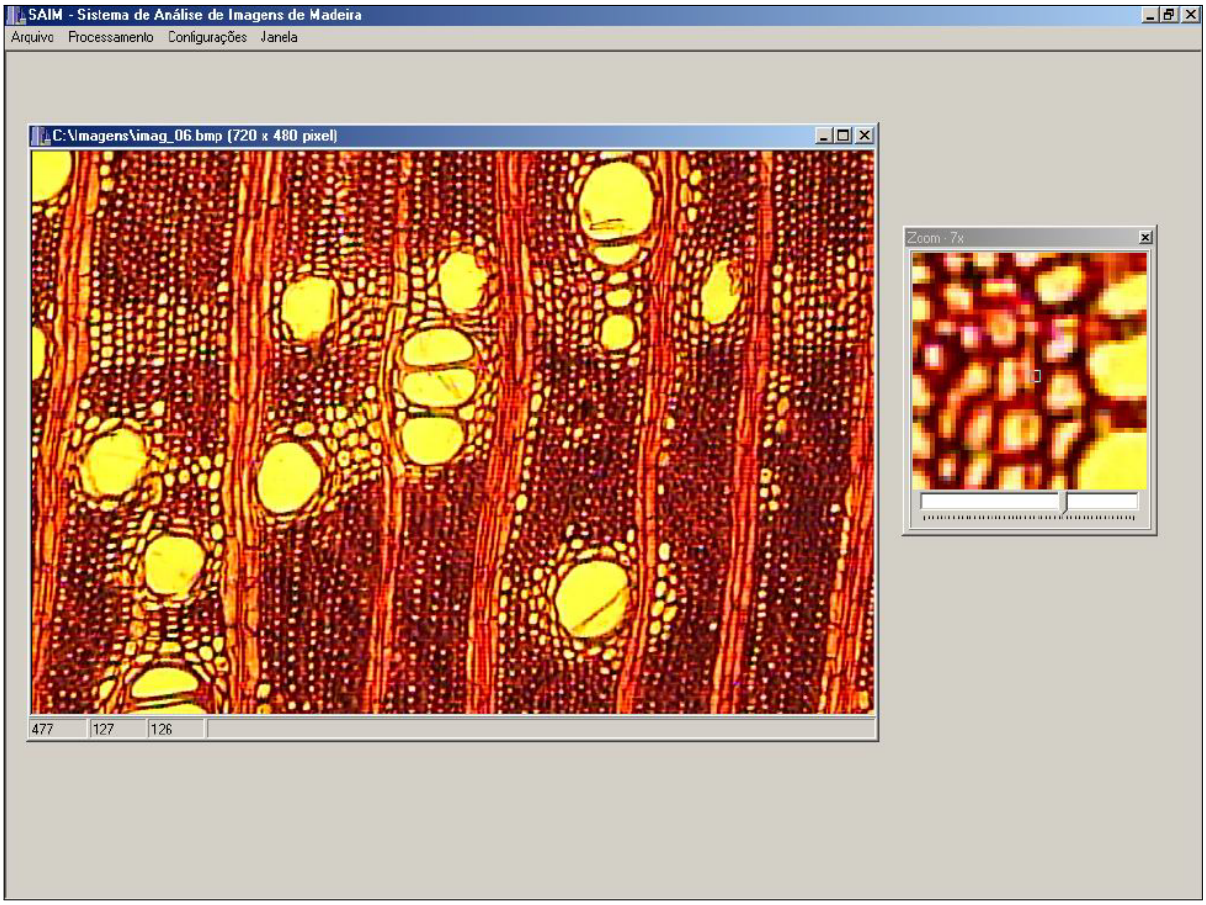
ações disponíveis, conforme ilustrado na Figura 22. Por fim, há uma área extra localizada na parte inferior da seção de visualização de imagem e dos filtros, responsável por exibir informações complementares geradas por determinados processamentos, especialmente aqueles que geram gráficos para análises detalhadas.

Figura 22 – Exemplo de filtro Dilatação.

Fonte: Próprio Autor.

Em relação ao sistema original, a Figura 23 é apresentada a evolução da funcionalidade de visualização de imagens, evidenciando melhorias no recurso de zoom e na integração da funcionalidade com todos os filtros.

Figura 23 – Comparação funcionalidade de Zoom.



Fonte: Ribeiro (2002).

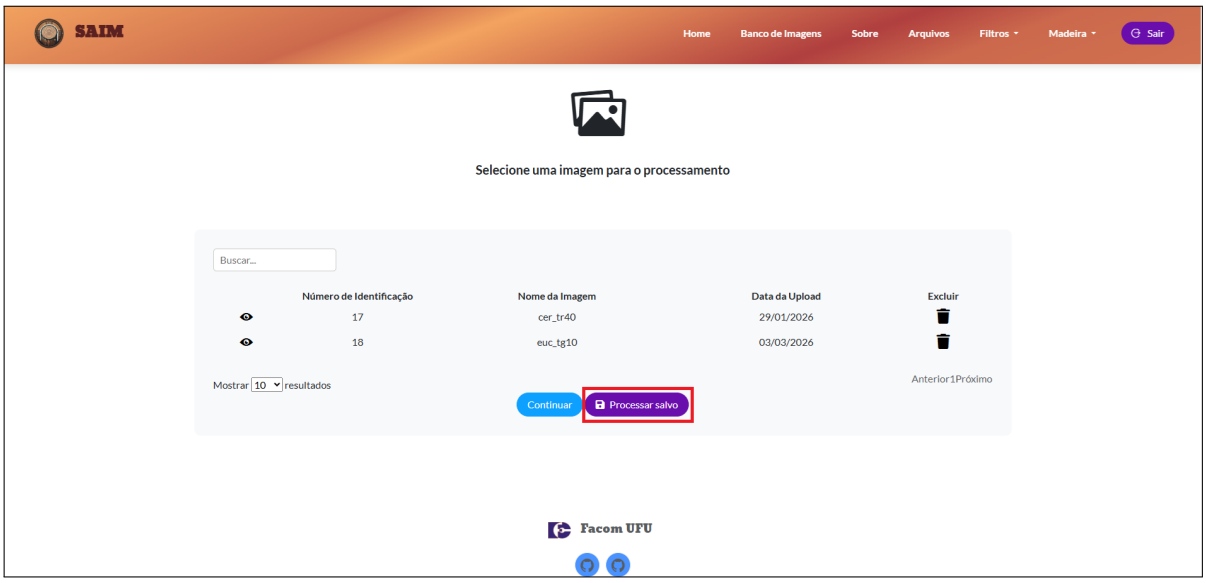
Fonte: Próprio Autor.

Após a execução de qualquer processamento, o usuário pode realizar o *download* da imagem gerada ou optar por salvá-la na própria aplicação sem *download* (Figura 24), tornando-a disponível para seleção em futuros processamentos (Figura 25).

Figura 24 – Exemplo de resultado de processamento de filtro Dilatação.

Fonte: Próprio Autor.

Figura 25 – Seleção de imagem processada anteriormente para novo processamento.

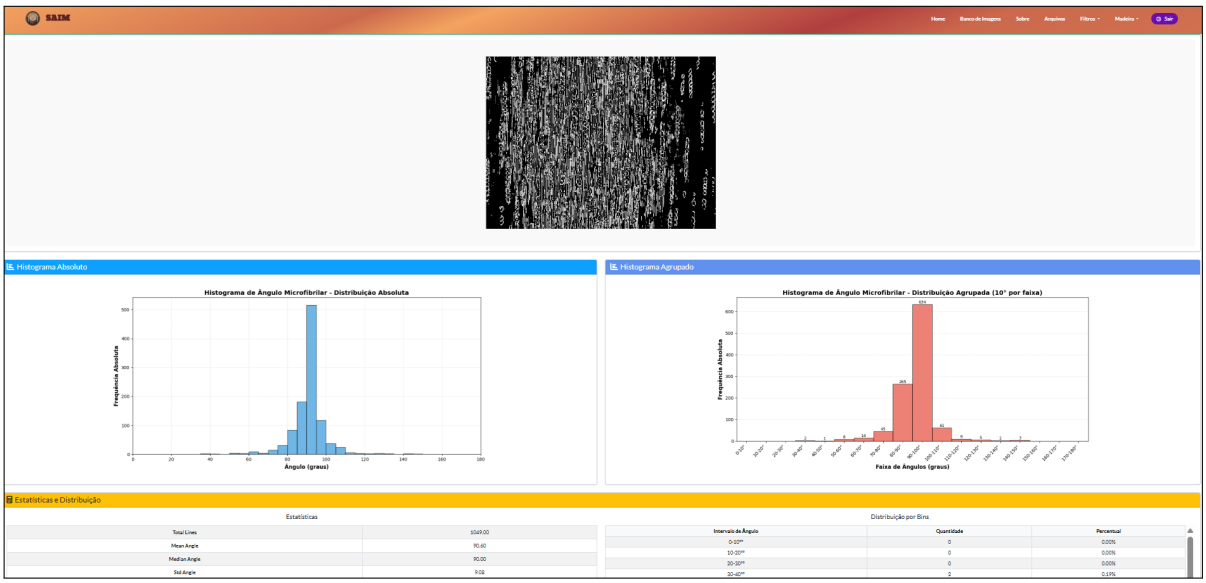


Fonte: Próprio Autor.

Por fim, tem-se a seção Madeira, que reúne os processamentos mais avançados e completos do sistema, combinando os diferentes filtros disponíveis na seção Filtros. Essa seção também oferece a geração de gráficos para análises detalhadas das informações extraídas da imagem selecionada, como o Histograma de Ângulo Microfibrilar e a Intensidade de Pixels entre dois pontos distintos (Figura 26 e 27). Destaca-se que, no caso da análise de intensidade, na figura apresentada também é incluída a comparação com o sistema original, evidenciando a evolução da interface.

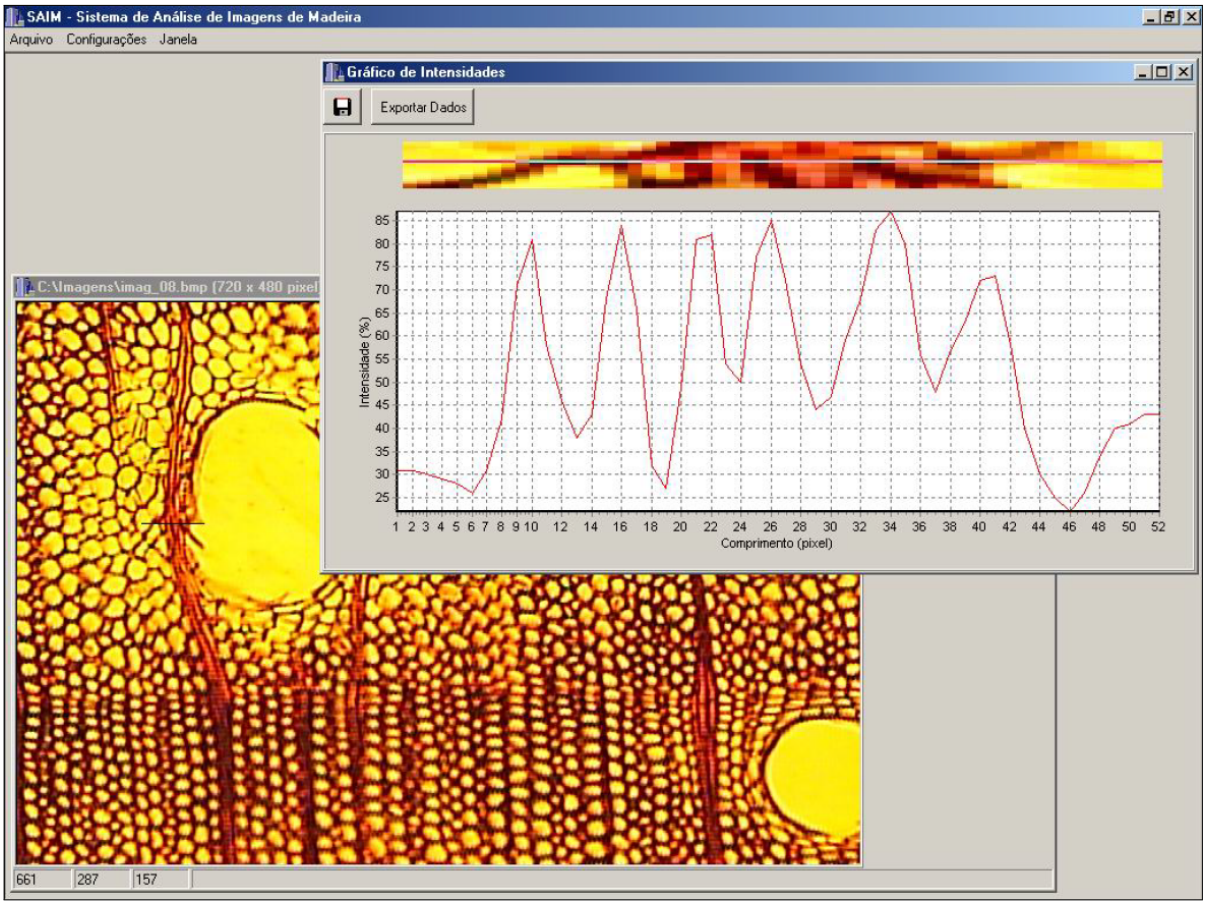
Outra funcionalidade disponível é o uso de filtros de mensuração, tanto no automático quanto manual, que oferecem informações detalhadas organizadas em tabelas (Figura 28). Esses dados possibilitam a comparação de informações e a análise aprofundada das diferentes áreas da imagem processada. Essas funcionalidades proporcionam uma análise mais aprofundada dos resultados obtidos, ampliando o potencial analítico da aplicação como um todo.

Figura 26 – Exemplo de resultado de processo de Histograma.

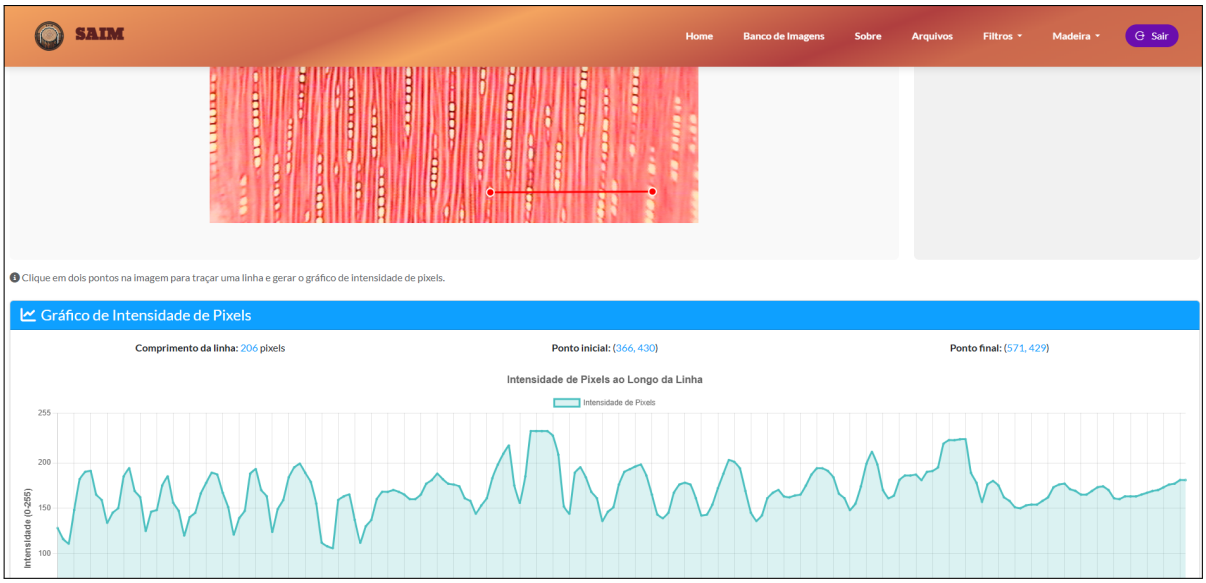


Fonte: Próprio Autor.

Figura 27 – Exemplo de resultado de processo de Intensidade.

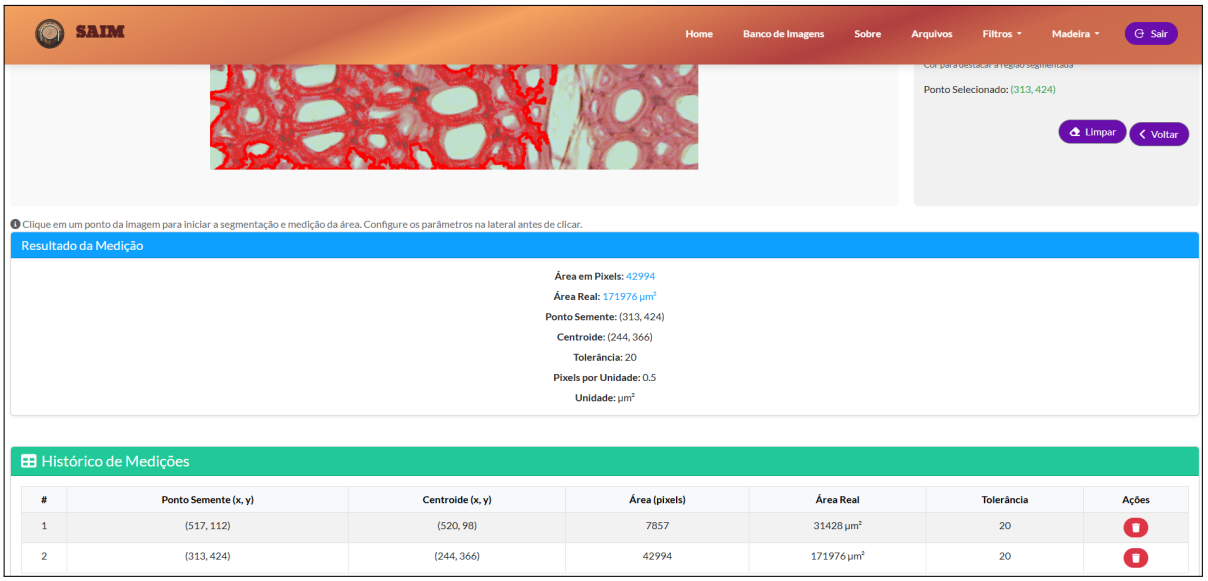


Fonte: Ribeiro (2002).



Fonte: Próprio Autor.

Figura 28 – Exemplo de resultado de processo de Mensuração Manual.



Fonte: Próprio Autor.

3.4 Avaliação dos Resultados

A avaliação dos resultados deste trabalho foi realizada com base nas funcionalidades implementadas, na arquitetura adotada e na capacidade da plataforma em atender ao objetivo proposto de apoiar as disciplinas de Anatomia da Madeira e Dendrologia por meio de uma plataforma acessível e integrada.

O desenvolvimento do sistemas envolveu a evolução progressiva da solução, iniciando pela prototipação da interface, seguida pela construção inicial, que serviu de base para a implementação final. A partir dessa estrutura inicial, o sistema foi continuamente aprimorado, com a incorporação de novas funcionalidades, melhorias na estrutura e organização do código e adequações na arquitetura da aplicação.

Como resultado, foi desenvolvida uma plataforma *web* funcional, acessível por navegador e independente de instalação local, superando limitações do sistema anterior que dependia de sistemas operacionais específicos. A solução implementada apresenta uma arquitetura moderna, baseada na separação entre *backend* e *frontend*, evitando a construção de um sistema monolítico e permitindo maior manutenibilidade e escalabilidade.

No que se refere ao processamento de imagens, os métodos implementados foram baseados nos conceitos apresentados em Ribeiro (2002), sendo adaptados e reimplementados com o uso de bibliotecas atuais. Assim, a solução proposta preserva os fundamentos teóricos do trabalho original, ao mesmo tempo em que utiliza tecnologias modernas para viabilizar sua execução em ambiente *web*.

Para evidenciar de forma objetiva as contribuições alcançadas, na tabela 1 é apresentado um resumo das principais funcionalidades implementadas, permitindo uma visuali-

zação objetiva dos recursos desenvolvidos e da abrangência da solução proposta.

Tabela 1 – Funcionalidades implementadas na plataforma

Categoria	Funcionalidade	Descrição
Autenticação e Segurança	Login e autenticação	Validação de credenciais e controle de acesso ao sistema
Autenticação e Segurança	Cadastro de usuário	Registro de novos usuários com validação de dados
Autenticação e Segurança	Gerenciamento de sessão	Armazenamento e controle de estado do usuário
Autenticação e Segurança	Logout	Encerramento seguro da sessão
Gerenciamento de Imagens	Upload de imagens	Envio de imagens para processamento
Gerenciamento de Imagens	Listagem e seleção	Visualização e escolha de imagens armazenadas
Gerenciamento de Imagens	Exclusão de imagens	Remoção de imagens do sistema
Gerenciamento de Imagens	Visualização (preview)	Exibição da imagem antes do processamento
Gerenciamento de Imagens	Armazenamento local	Persistência de imagens no navegador (IndexedDB)
Processamento de Imagens	Filtros de processamento	Aplicação de técnicas como limiarização, erosão e dilatação
Processamento de Imagens	Mensuração manual	Medição de estruturas diretamente na imagem
Processamento de Imagens	Mensuração automática	Extração automática de medidas da imagem
Processamento de Imagens	Análise de intensidade	Geração de gráficos de intensidade de pixels
Processamento de Imagens	Histogramas	Análise de distribuição de características da imagem
Arquitetura do Sistema	API REST	Comunicação entre frontend e backend
Arquitetura do Sistema	Integração entre módulos	Integração entre processamento, interface e dados
Interface	Interface web	Acesso ao sistema via navegador
Interface	Navegação dinâmica	Menus e fluxos adaptados ao usuário autenticado

De forma geral, os resultados demonstram que a plataforma atende aos objetivos propostos, oferecendo uma solução integrada que combina processamento de imagens, interface amigável e uma arquitetura moderna. Além disso, o sistema foi estruturado de forma a permitir expansões futuras, tanto no aprimoramento dos métodos de análise quanto na inclusão de novas funcionalidades.

Como limitações deste trabalho, destaca-se que a plataforma ainda não passou por um processo formal de validação com especialistas da área de Anatomia de Madeira, o que impossibilita, neste momento, uma avaliação técnica aprofundada quanto à aderência dos métodos implementados às práticas laboratoriais e acadêmicas. Da mesma forma, não foram realizados estudos com discentes das disciplinas alvo para avaliação de aspectos relacionados à usabilidade e impacto no uso em ambiente educacional.

Outra limitação observada refere-se à ausência de métricas comparativas formais entre a solução desenvolvida e o sistema anterior e seus métodos de análise, o que restringe, nessa etapa, uma avaliação de desempenho e precisão proporcionados pela plataforma. Além disso, embora tenham sido implementados mecanismos de segurança a camada de autenticação ainda pode ser evoluída com a adoção de recursos mais robustos, como a autenticação baseada em *JSON Web Token* (JWT) e controle refinado de sessões.

Também se observa como limitação o fato dos métodos de processamento de imagens terem sido reimplementados com o uso da biblioteca *Open Source Computer Vision Library* (OpenCV), em razão da indisponibilidade operacional do sistema original utilizado como base, desenvolvido para ambientes operacionais legados. Dessa forma, embora os métodos preservem seus objetivos funcionais, sua implementação foi adaptada às tecnologias atuais. Ainda assim, tal cenário não compromete os objetivos centrais deste trabalho, voltados à modernização tecnológica, acessibilidade via *web* e integração dos módulos da plataforma.

Capítulo 4

Conclusão

Este trabalho possibilitou o desenvolvimento de uma plataforma *web* voltada ao processamento e à análise de imagens, integrando diferentes tecnologias modernas tanto na camada de *backend* quanto na camada de *frontend*. No *backend*, a utilização da linguagem Python em conjunto com o FastAPI permitiu a construção de uma API REST escalável, estruturada e alinhada às boas práticas de desenvolvimento. Essa estratégia também possibilitou a adoção de um método mais seguro para o armazenamento de chaves de API e das informações de conexão com o banco de dados. Para a persistência das informações, o PostgreSQL proporcionou uma estrutura simplificada, eficiente, organizada e segura para as informações armazenadas do sistema.

No que se refere à segurança, foi utilizado o algoritmo de *hash* SHA-256 combinado com o uso de *salt*, garantindo assim uma maior proteção das credenciais dos usuários. Além disso, a padronização das comunicações originadas da API contribuiu para uma comunicação mais consistente e eficiente entre *backend* e *frontend*, facilitando também eventuais manutenções do sistema.

Na camada de *frontend*, a interface foi desenvolvida utilizando as tecnologias fundamentais da *web* (HTML, CSS, JavaScript), em conjunto com o *framework* Bootstrap, possibilitando a criação de uma interface responsiva, visualmente agradável e de fácil utilização. Para melhorar a experiência dos usuários e possibilitar uma maior fluidez na utilização da aplicação, foram adotados o uso de `SessionStorage` e `IndexedDB`, permitindo um melhor e mais eficiente gerenciamento de dados temporários.

A integração das tecnologias utilizadas, aliada à arquitetura, resultou em uma solução robusta, escalável e de fácil utilização pelos usuários. Dessa forma, a plataforma desenvolvida mostra-se como uma ferramenta de apoio às aulas de anatomia da madeira.

4.1 Trabalhos Futuros

Como evolução da plataforma, algumas melhorias e expansões podem ser implementadas:

1. Publicação do frontend e backend em servidores juntamente com a disponibilização da aplicação ao público;
2. Melhoria na autenticação da aplicação se utilizando do JSON Web Token JWT;
3. Implementação de processo de recuperação de senha por e-mail de usuário cadastrado;
4. Desenvolvimento de um painel de controle administrativo com estatísticas gerais da aplicação;
5. Validação da plataforma por especialistas em anatomia da madeira, visando avaliação prática e identificação de melhorias;
6. Avaliação de usabilidade por meio de questionário com usuários.

Referências

BRADSKI, G. The OpenCV Library. **Dr. Dobb's Journal of Software Tools**, 2000. Citado na página 17.

BURGER, L. M.; RICHTER, H. G. **Anatomia da madeira**. São Paulo: Nobel, 1991. Citado na página 12.

CARDOSO, V. S. **Implementação de um Backend e Integração com Frontend para Acompanhamento de Egressos da UFU**. 2025. Https://repositorio.ufu.br/handle/123456789/46182. Acesso em 18 mar. 2026. Citado na página 20.

GONZALEZ, R.; WOODS, R. **Processamento Digital De Imagens**. ADDISON WESLEY BRA, 2009. ISBN 9788576054016. Disponível em: <https://books.google.com.br/books?id=r5f0RgAACAAJ>. Citado 2 vezes nas páginas 19 e 20.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing**. 4. ed. São Paulo: Pearson, 2018. 1072 p. ISBN 978-0-13-335672-4. Citado 2 vezes nas páginas 13 e 19.

HARRIS, C. R. et al. Array programming with NumPy. **Nature**, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <https://doi.org/10.1038/s41586-020-2649-2>. Citado na página 17.

MACHADO, T. V. **Aplicação de técnicas de Visão Computacional para identificação de ferrugem em folhas de café**. 2023. Https://repositorio.ufu.br/handle/123456789/38294. Acesso em 18 mar. 2026. Citado na página 20.

NASCIMENTO, P. F. R. d. **Desenvolvimento do backend com ênfase na modelagem de dados para o sistema de acompanhamento de egressos na UFU**. 2025. Https://repositorio.ufu.br/handle/123456789/46512. Acesso em 18 mar. 2026. Citado na página 20.

OLIPHANT, T. E. **A Guide to NumPy**. USA: Trelgol Publishing, 2006. Citado na página 17.

OpenCV. **OpenCV: Open Source Computer Vision Library**. 2025. <https://opencv.org/>. Acessado em 28 de agosto de 2025. Citado na página 17.

ORACLE. **What Is a Database?** 2020. <https://www.oracle.com/br/database/what-is-database/>. Acessado em: 21 de junho de 2023. Citado na página 18.

POSTGRESQL GLOBAL DEVELOPMENT GROUP. **PostgreSQL Documentation PostgreSQL**. Universidade da Califórnia, Berkeley, 2022. Disponível em: <https://www.postgresql.org/docs/15/index.html>. Citado na página 18.

Python Core Team. **Python: A dynamic, open source programming language**. [S.l.], 2019. Python version 3.7. Disponível em: <https://www.python.org/>. Acesso em: Acessado em 14 Jun. de 2023. Citado na página 16.

RAMÍREZ, S. **FastAPI framework, high performance, easy to learn, fast to code, ready for production**. 2018. <https://fastapi.tiangolo.com/>. Acessado em: 20 de agosto de 2025. Citado na página 17.

RIBEIRO, T. P. **Método complementar de análise da qualidade de madeira com técnicas do processamento digital de imagens**. Dissertação (Mestrado) — Universidade Federal de São Carlos, São Carlos, 2002. Citado 9 vezes nas páginas 13, 19, 23, 33, 35, 37, 39, 42 e 43.

RIBEIRO, T. P.; CARNEIRO-TOMAZELLO, M.; FILHO, M. T. Análise de imagem da estrutura anatômica da madeira de espécies florestais nativas e introduzidas no brasil. In: USP. **Simpósio de Iniciação Científica da Universidade de São Paulo**. São Paulo, 1999. Citado 2 vezes nas páginas 12 e 13.

RIBEIRO, T. P. et al. An image processing methodology for analyzing the quality of brazilian forest wood. **Revista Árvore**, 2005. Citado 2 vezes nas páginas 12 e 13.

_____. Brazilian wood samples analysis by means of atomic force microscopy. **Acta Microscópica**, v. 12, n. A, p. 1–6, 2003. Citado na página 13.

RIBEIRO, T. P.; FILHO, M. T.; CRUVINEL, P. E. Algorithm for analyzing anatomical structure of wood from brazilian forest species based on digital imaging processes techniques. In: **Proceedings XIV Brazilian Symposium on Computer Graphics and Image Processing**. Florianópolis: IEEE, 2001. p. 392–. Citado 2 vezes nas páginas 12 e 13.

ROSSUM, G. van; DRAKE, F. L. **Python 3 Reference Manual**. Scotts Valley, CA: CreateSpace, 2009. Citado na página 16.

Seobility. **REST API**. 2023. <https://www.seobility.net/en/wiki/REST_API>. Acessado em: 10 de março de 2026. Citado na página 19.