



UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM  
FACULDADE DE TECNOLOGIA - FT  
ENGENHARIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UMA PLATAFORMA WEB PARA  
IMPORTAÇÃO E EXPORTAÇÃO DE DADOS EDUCACIONAIS

Gabriel Sobrinho Sahdo

Manaus, 15 de Dezembro de 2025

Gabriel Sobrinho Sahdo

DESENVOLVIMENTO DE UMA PLATAFORMA WEB PARA  
IMPORTAÇÃO E EXPORTAÇÃO DE DADOS EDUCACIONAIS

Monografia de Graduação apresentada à  
Coordenação de Engenharia da Computação,  
UFAM, da Universidade Federal do Amazonas,  
como parte dos requisitos necessários à obtenção  
do título de Engenheiro da Computação.

Orientador: Dr. Edson Nascimento Silva Júnior

Manaus, 15 de Dezembro de 2025

Ficha Catalográfica

Elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

---

S131d Sahdo, Gabriel Sobrinho

Desenvolvimento de uma plataforma web para importação e exportação de dados educacionais / Gabriel Sobrinho Sahdo. - 2025.

53 f. : il., color. ; 31 cm.

Orientador(a): Edson Nascimento Silva Júnior.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal do Amazonas, Faculdade de Tecnologia, Curso de Engenharia da Computação, Manaus, 2025.

1. Desenvolvimento web. 2. Sistema educacional. 3. Importação de dados. 4. React. 5. Node.js. I. Silva Júnior, Edson Nascimento. II. Universidade Federal do Amazonas. Faculdade de Tecnologia. Curso de Engenharia da Computação. III. Título


---

# DESENVOLVIMENTO DE UMA PLATAFORMA WEB PARA IMPORTAÇÃO E EXPORTAÇÃO DE DADOS EDUCACIONAIS

Gabriel Sobrinho Sahdo


MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO CURSO DE ENGENHARIA DA COMPUTAÇÃO DA UNIVERSIDADE FEDERAL DO AMAZONAS COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM ENGENHARIA DE COMPUTAÇÃO.

Aprovada por:

Documento assinado digitalmente  
 **EDSON NASCIMENTO SILVA JUNIOR**  
Data: 15/12/2025 18:44:19-0300  
Verifique em <https://validar.iti.gov.br>


---

Prof. Dr. Edson Nascimento Silva Júnior

Documento assinado digitalmente  
 **RUI TER BRAGA CALDAS**  
Data: 15/12/2025 18:55:25-0300  
Verifique em <https://validar.iti.gov.br>

---

Prof. Dr. Ruiteir Braga Caldas

Documento assinado digitalmente  
 **JOAO MARCOS BASTOS CAVALCANTI**  
Data: 16/12/2025 18:41:14-0300  
Verifique em <https://validar.iti.gov.br>

---

Prof. João Marcos Bastos Cavalcante, Ph.D.

Manaus, 15 de dezembro de 2025

# DESENVOLVIMENTO DE UMA PLATAFORMA WEB PARA IMPORTAÇÃO E EXPORTAÇÃO DE DADOS EDUCACIONAIS

Monografia apresentada ao curso de Engenharia da Computação da Universidade Federal do Amazonas como parte dos requisitos para a obtenção do título de Engenheiro da Computação.

Aprovado em: 15/12/2025

## Sumário

<b>Capítulo 1</b>	<b>12</b>
Introdução	12
1.1 Contextualização e Motivação	12
1.2 Objetivos	13
1.3 Justificativa	13
1.4 Estrutura da Monografia	14
<b>Capítulo 2</b>	<b>15</b>
Fundamentação Teórica	15
2.1 Desenvolvimento Web e Arquitetura de Sistemas	15
2.2 Tecnologias Utilizadas	15
2.2.1 React e Bootstrap (Frontend)	15
2.2.2 Node.js, Express e Typescript (Backend)	16
2.2.3 Prisma ORM e MySQL (Banco de Dados)	16
2.3 Importação e Exportação de Planilhas	17
2.4 Segurança e Autenticação	17
<b>Capítulo 3</b>	<b>18</b>
Metodologia de Desenvolvimento	18
3.1 Levantamento de Requisitos	18
3.2 Planejamento do Projeto	19
3.3 Recursos utilizados	19
3.3.1 Essenciais	19
3.3.2 Auxiliares	21
<b>Capítulo 4</b>	<b>22</b>
Desenvolvimento da Ferramenta	22
4.1 Descrição Geral do Sistema	22
4.2 Arquitetura do Sistema	23
4.3 Banco de Dados	27
4.4 Fluxo de Navegação e Experiência do Usuário	31
4.5 Módulo de Autenticação de Usuários	32
4.6 Módulo de Cadastro de Professores	34
4.7 Cadastro de Turmas, Alunos e Importação de Planilhas	37
4.8 Interface de Preenchimento de Dados pelos Alunos	39
<b>Capítulo 5</b>	<b>42</b>
Resultados Obtidos	42
5.1 Funcionalidades Concretizadas	42
5.2 Pontos Fortes da Plataforma	43
5.3 Desafios Encontrados	43
5.4 Contribuições Práticas	44

<b>Capítulo 6</b>	<b>45</b>
Conclusões e Trabalhos Futuros	45
6.1 Considerações Finais	45
6.2 Trabalhos Futuros	46
<b>Referências Bibliográficas</b>	<b>47</b>
<b>Apêndice A - Estrutura do Banco de Dados</b>	<b>49</b>
<b>Apêndice B - Fluxo Geral do Sistema</b>	<b>53</b>

# Lista de Figuras

<b>4.1</b>	Organização dos diretórios do <i>frontend</i> _____	24
<b>4.2</b>	Organização dos diretórios do <i>backend</i> _____	25
<b>4.3</b>	Diagrama do modelo de dados do sistema _____	29
<b>4.4</b>	Fluxograma Completo do Sistema. _____	32
<b>4.5</b>	Tela de login da aplicação _____	33
<b>4.6</b>	Interface de formulário de solicitação do professor _____	34
<b>4.7</b>	Interface de consulta da situação de cadastro _____	35
<b>4.8</b>	Interface de listagem de solicitação de cadastro _____	36
<b>4.9</b>	Página de detalhes da solicitação de cadastro de professor _____	37
<b>4.10</b>	Interface para cadastro e listagem de turmas do professor _____	38
<b>4.11</b>	Turmas listadas na página após o cadastro _____	38
<b>4.12</b>	Interface de importação e exportação de dados dos alunos _____	39
<b>4.13</b>	Página de atualização de dados do aluno _____	40

# Agradecimentos

Primeiramente, agradeço a Deus, por ter me sustentado em cada etapa dessa caminhada, me concedendo força, sabedoria e perseverança mesmo nos momentos mais difíceis.

Aos meus pais e à minha irmã, por serem meu alicerce, por cada gesto de carinho, cada palavra de apoio e cada demonstração de confiança no meu potencial. Vocês são minha base e inspiração.

Aos meus amigos — tanto da faculdade, quanto de fora — que, com palavras, gestos, companheirismo e incentivos constantes, fizeram com que esse percurso se tornasse mais leve. Obrigado por acreditarem em mim e celebrar cada pequena conquista comigo.

Agradeço também ao meu orientador, pela orientação técnica, paciência e incentivo contínuo durante o desenvolvimento deste trabalho. Sua contribuição foi fundamental para que este projeto ganhasse forma.

Por fim, agradeço aos professores do Web Academy e da graduação, cujos ensinamentos, exemplos e desafios ao longo da jornada acadêmica moldaram minha trajetória profissional e pessoal.

## **Resumo**

Este trabalho apresenta o desenvolvimento de uma plataforma web voltada ao ambiente educacional, com o objetivo de facilitar o processo de coleta e gerenciamento de dados estudantis. A ferramenta permite que professores importem planilhas contendo informações básicas dos alunos (CPF e e-mail), a partir das quais são criadas turmas e contas de acesso. Os alunos, por sua vez, acessam a plataforma para completar seus dados cadastrais. Ao final do processo, o professor pode exportar uma nova planilha com as informações preenchidas. O sistema foi construído utilizando tecnologias modernas como React, Node.js, TypeScript, PrismaORM e MySQL, com foco em segurança, usabilidade e escalabilidade. A metodologia aplicada envolveu levantamento de requisitos, modelagem de dados, desenvolvimento incremental, testes e validações com usuários reais. Os resultados demonstraram que a plataforma é funcional, segura e eficaz para sua proposta, sendo aplicável em diferentes contextos escolares.

Palavras-chave: desenvolvimento web, sistema educacional, importação de dados, React, Node.js, TypeScript, PrismaORM.

## **Abstract**

This work presents the development of a web platform aimed at the educational environment, with the goal of streamlining the process of collecting and managing student data. The tool allows teachers to import spreadsheets containing students basic information (CPF and email), from which student accounts and class groups are automatically created. Students log into the platform to complete their personal data, and at the end of the process, the teacher can export a new spreadsheet containing the completed records. The system was built using modern technologies such as React, Node.js, TypeScript, PrismaORM, and MySQL, with a focus on security, usability, and scalability. The applied methodology involved requirement gathering, data modeling, incremental development, testing, and validation with real users. The results show that the platform is functional, secure, and effective for its intended use, and applicable in various educational contexts.

Keywords: web development, educational system, data import, React, Node.js, TypeScript, PrismaORM

# Capítulo 1

## Introdução

A ausência de uma ferramenta que automatize, de forma segura e organizada, o fluxo de coleta de dados estudantis a partir de planilhas dificulta o gerenciamento de dados escolares, que é essencial para professores e instituições de ensino. Pensando nisso, o presente trabalho apresenta o desenvolvimento de uma plataforma web voltada para o cadastro e acompanhamento de informações de alunos, com funcionalidades de importação e exportação de planilhas, autenticação por perfil (professor e aluno) e gerenciamento de turmas.

A plataforma foi desenvolvida utilizando uma stack moderna composta por React e Bootstrap no frontend, Node.js com Express e Typescript no backend, e PrismaORM com MySQL no gerenciamento de banco de dados. Essa arquitetura visa oferecer uma solução eficiente, responsiva e segura para o fluxo de entrada e saída de dados.

### 1.1 Contextualização e Motivação

O uso de planilhas é uma alternativa usada na rotina de professores para organizar turmas, registrar presenças e controlar dados escolares. Entretanto, a gestão manual desses documentos pode gerar erros, retrabalhos e perda de informações. A necessidade de uma ferramenta que automatize o processo de coleta, organização e exportação de dados motivou o desenvolvimento deste projeto. O projeto surge, portanto, como resposta a uma demanda concreta por digitalização segura, organizada e acessível dos dados estudantis.

## **1.2 Objetivos**

Objetivo Geral:

- Desenvolver uma plataforma web para facilitar o cadastro de informações escolares a partir da importação e exportação de planilhas.

Objetivos Específicos:

- Implementar login diferenciado para alunos e professores.
- Permitir importação de planilhas com CPF e e-mail dos alunos.
- Criar interface intuitiva para preenchimento de dados pelos alunos.
- Gerar exportação dos dados completos em planilha ao final do processo.
- Garantir a segurança dos dados trafegados e armazenados.
- Tornar o sistema responsivo e acessível em diferentes dispositivos.

## **1.3 Justificativa**

O uso de sistemas personalizados para esse tipo de fluxo torna o processo de coleta de dados em massa algo eficiente [1]. Dentro do contexto apresentado, professores têm de lidar com um grande volume de dados de alunos, o que muitas vezes demanda tempo, atenção redobrada e controle constante para evitar erros e inconsistências na abordagem do uso de planilhas.

A ferramenta proposta busca suprir essa lacuna, proporcionando uma solução que centraliza as informações, automatiza tarefas repetitivas, e ainda permite a personalização das turmas de acordo com as necessidades de cada professor [1]. O uso de tecnologias modernas e acessíveis também contribui para a sua manutenção a longo prazo e para futuras escalabilidades [1].

#### **1.4 Estrutura da Monografia**

O Capítulo 2 apresenta os fundamentos teóricos e as tecnologias utilizadas na construção da ferramenta. O Capítulo 3 detalha a metodologia e abordagem adotadas no desenvolvimento. O Capítulo 4 mostra a implementação da ferramenta e suas funcionalidades. O Capítulo 5 discute os resultados obtidos e o Capítulo 6 apresenta as conclusões e possibilidades de trabalhos futuros.

# Capítulo 2

## Fundamentação Teórica

Este capítulo apresenta os conceitos e tecnologias fundamentais para o desenvolvimento da plataforma proposta. A fundamentação teórica visa contextualizar o leitor sobre os recursos empregados e justificar as escolhas técnicas realizadas.

### 2.1 Desenvolvimento Web e Arquitetura de Sistemas

O desenvolvimento web é uma área que contempla uma gama de conhecimentos e boas práticas voltadas à criação de sistemas acessíveis via navegadores [2]. Divide-se em duas grandes frentes: o frontend, que é a interface visual com a qual o usuário interage; e o backend, que é responsável por tratar as regras de negócio e se comunicar com bancos de dados.

A arquitetura adotada neste projeto é do tipo cliente-servidor, com o navegador operando como cliente que consome serviços disponibilizados por uma API (Interface de Programação de Aplicativos). Essa API, por sua vez, manipula os dados e retorna as respostas necessárias à visualização ou à persistência de informações.

### 2.2 Tecnologias Utilizadas

#### 2.2.1 React e Bootstrap (Frontend)

React [3] é uma biblioteca mantida pelo Facebook que revolucionou o desenvolvimento de interfaces ao introduzir o conceito de componentes reutilizáveis e reatividade no DOM [4] virtual. Com React, a construção de interfaces torna-se mais modular, escalável e eficiente. O uso do React permite também melhor integração com bibliotecas auxiliares, como React Router para navegação e Axios para requisições HTTP [3].

Bootstrap [5] complementa o React com uma abordagem responsiva baseada em grid e estilos pré-definidos que aceleram o desenvolvimento. Sua sintaxe intuitiva permite criar layouts responsivos com facilidade, contribuindo para uma experiência mais fluida nos diferentes tamanhos de tela.

### **2.2.2 Node.js, Express e Typescript (Backend)**

Node.js [6] é uma tecnologia que possibilita o uso de JavaScript fora do navegador, permitindo desenvolver aplicações backend com uma linguagem já familiar ao desenvolvedor frontend. Sua alta performance e *event loop* tornam o Node ideal para aplicações que demandam escalabilidade e resposta rápida.

Express [7] é um framework minimalista e flexível que roda sobre o Node.js, facilitando a criação de rotas, *middleware* e integração com banco de dados. Sua ampla documentação e comunidade ativa facilitam a resolução de problemas e expansão do projeto.

Typescript [8] entra como um reforço na legibilidade e segurança do código, por trazer tipagem estática e recursos de desenvolvimento mais robustos. Ele reduz a ocorrência de erros em tempo de execução e aumenta a manutenção do sistema.

### **2.2.3 Prisma ORM e MySQL (Banco de Dados)**

O Prisma [9] é um ORM moderno que automatiza grande parte das operações com o banco de dados, gerando mapeamentos em tempo real das entidades e permitindo o uso de comandos mais seguros e intuitivos. Ele suporta migrações de esquema, introspecção de banco e validações integradas.

Para este projeto, o banco relacional escolhido foi o MySQL [10], uma opção amplamente usada em projetos de pequeno a grande porte. O MySQL combina estabilidade, compatibilidade com serviços de nuvem e excelente desempenho, sendo ideal para aplicações que necessitam de integridade e consistência nos dados.

## 2.3 Importação e Exportação de Planilhas

Planilhas eletrônicas possibilitam a visualização e organização rápida de listas de alunos, presenças, notas e outros registros. No contexto institucional, esse formato já é amplamente utilizado para consolidar informações administrativas e acadêmicas, de modo que a própria entrada do processo, isto é, os dados enviados pelos professores ou coordenações já chega ao sistema como uma planilha previamente preenchida. Assim, aproveitar esse formato familiar não apenas facilitou a adesão à ferramenta, como também reduziu a necessidade de etapas manuais adicionais, garantindo compatibilidade direta entre o arquivo enviado e o que a aplicação precisa tratar, desde a leitura dos dados até sua formatação para visualização.

A leitura desses dados é realizada a partir de um arquivo separado por vírgulas — *Comma Separated Values* (.csv). Esse arquivo contém as informações que o usuário deseja importar ou exportar e pode ser processado pelo **SheetJS** [11], para fazer a leitura e posteriormente pelo **ExcelJS** [12] para gerar arquivos de planilhas contendo os dados consolidados. Ambas as bibliotecas permitem converter o conteúdo das planilhas em objetos manipuláveis no código, possibilitando que o sistema trate, valide e reestruture esses dados com segurança.

## 2.4 Segurança e Autenticação

A segurança dos dados dos usuários é uma prioridade em qualquer sistema. Para tanto, a ferramenta de autenticação escolhida utiliza criptografia de senha (no caso, utilizamos a biblioteca *bcrypt* [13] para Node.js para criptografar as senhas), criação de sessões de usuário e proteção contra ataques comuns, como CSRF [14] e injeção de SQL [14] — que é reforçada pela proteção nativa do ORM Prisma contra esse tipo de ataque.

Essa base teórica fornece os subsídios necessários para compreender as decisões de projeto e a implementação da ferramenta descrita nos próximos capítulos.

# Capítulo 3

## Metodologia de Desenvolvimento

A metodologia adotada foi baseada em práticas de “metodologia ágil” [15] com conceito de Sprints com tempos variáveis, sistema local de controle baseado no *kanban* [16] essencial para garantir a organização do trabalho, a qualidade do código e a manutenção da coerência entre os objetivos definidos e os resultados alcançados.

### 3.1 Levantamento de Requisitos

O processo de levantamento de requisitos foi realizado com base em observações práticas do cotidiano de professores que lidam com grandes volumes de dados estudantis. Foram identificadas as principais dores e dificuldades, tais como:

- Necessidade de criar turmas rapidamente a partir de listas de alunos;
- Falta de controle padronizado sobre os dados dos alunos;
- Dificuldade em acompanhar quem completou ou não o preenchimento de informações;
- Inexistência de ferramentas para exportar os dados preenchidos com confiabilidade.

Com isso, os requisitos funcionais e não funcionais da plataforma foram definidos e priorizados. Dentre os principais requisitos funcionais destacam-se:

- Autenticação de usuários (aluno e professor);
- Importação de planilhas em formato .csv;
- Cadastro automático de alunos com base nos dados importados;
- Interface intuitiva para preenchimento de dados pelos alunos;
- Exportação dos dados completos da turma.

## 3.2 Planejamento do Projeto

O projeto foi dividido em três fases principais:

- **Fase 1: Estruturação e configuração do ambiente de desenvolvimento**, com escolha de tecnologias, configuração do banco de dados e setup do servidor;
- **Fase 2: Implementação funcional**, com desenvolvimento da API, integração com o banco, construção do frontend e testes parciais;
- **Fase 3: Validação e refinamento**, com testes em cenários reais, ajustes visuais e melhorias na usabilidade.

Essas fases foram conduzidas de maneira iterativa, adotando-se boas práticas inspiradas em metodologias ágeis, como a divisão do projeto em entregáveis menores (sprints) e revisões frequentes para adaptação das funcionalidades.

## 3.3 Recursos utilizados

Seguindo da conceituação descrita na fundamentação teórica, existiram alguns recursos tecnológicos que foram utilizados para o desenvolvimento do sistema de pré-processamento de dados. Segue a descrição mais detalhada destes recursos.

### 3.3.1 Essenciais

Os recursos tecnológicos considerados essenciais foram aqueles diretamente envolvidos na construção da plataforma, viabilizando a implementação das funcionalidades centrais, o processamento dos dados e a interação com os usuários finais.

- **Node.js**, em conjunto com o framework **Express**, foi utilizado para o desenvolvimento do backend da aplicação. Essa combinação permitiu estruturar a API responsável pelo recebimento das requisições, execução das regras de negócio e comunicação com o banco de dados. A escolha dessas tecnologias favoreceu um desenvolvimento flexível e alinhado ao modelo cliente-servidor adotado.

No contexto do backend, algumas bibliotecas desempenharam papéis fundamentais:

- **Cors**: viabilizar a comunicação entre as camadas da aplicação, permitindo que o frontend consumisse os serviços disponibilizados pelo backend.
  - **Dotenv**: organizar e proteger as configurações sensíveis do sistema, separando dados de ambiente do código-fonte.
  - **Prisma ORM**: atuar como camada intermediária entre a aplicação e o banco de dados, facilitando a modelagem, consulta e manutenção dos dados de forma estruturada.
  - **MySQL2**: estabelecer a conexão com o banco de dados e permitir a execução de operações de leitura e escrita.
  - **Json Web Token (JWT)**: controlar o acesso às funcionalidades da plataforma por meio de autenticação baseada em tokens.
  - **Bcrypt**: assegurar o armazenamento seguro das credenciais dos usuários por meio de técnicas de criptografia.
  - **SheetJS (xlsx)**: realizar a leitura e o processamento inicial dos arquivos de planilhas enviados pelos professores.
  - **ExcelJS**: gerar arquivos de planilhas contendo os dados consolidados, possibilitando a exportação das informações ao final do processo.
- 
- O **MySQL** foi utilizado como sistema gerenciador de banco de dados relacional, sendo responsável pelo armazenamento das informações cadastradas na plataforma. Sua utilização permitiu organizar os dados em tabelas relacionadas, garantindo consistência e integridade durante o ciclo de uso da aplicação.
  - O **React** foi adotado no desenvolvimento do frontend, permitindo a construção de interfaces dinâmicas baseadas em componentes. Essa abordagem facilitou a atualização das telas conforme as interações do usuário, como autenticação, preenchimento de formulários e visualização de turmas.

- O **Bootstrap** foi empregado para auxiliar na estilização da interface, fornecendo componentes visuais prontos e um sistema de grid que contribuiu para a responsividade da aplicação em diferentes dispositivos.
- O **TypeScript** foi utilizado como extensão do JavaScript tanto no frontend quanto no backend, contribuindo para uma melhor organização do código, redução de erros e maior clareza na definição de tipos e estruturas de dados.

### 3.3.2 Auxiliares

Os recursos tecnológicos auxiliares foram utilizados como apoio ao processo de desenvolvimento, testes, organização e validação da plataforma, contribuindo para a eficiência do trabalho e a qualidade do produto final.

- O **Figma** foi utilizado na etapa inicial do projeto para a elaboração de protótipos das telas, permitindo visualizar previamente a disposição dos elementos e facilitar o planejamento da interface antes da implementação.
- O **Insomnia** foi empregado para testar e validar os endpoints da API durante o desenvolvimento do backend, possibilitando verificar o comportamento das requisições e respostas sem depender da interface gráfica.
- Como ambiente de desenvolvimento integrado, utilizou-se o **Visual Studio Code**, que ofereceu suporte à escrita, organização e manutenção do código por meio de recursos como extensões, depuração e integração com o controle de versões.
- O navegador **Google Chrome** foi utilizado para testar as interfaces desenvolvidas, inspecionar elementos visuais e identificar ajustes necessários relacionados à responsividade e ao comportamento da aplicação.

# Capítulo 4

## Desenvolvimento da Ferramenta

Neste capítulo, será descrito em detalhes o processo de desenvolvimento da plataforma web de pré-processamento de dados de alunos. Serão apresentados os módulos implementados, as decisões técnicas tomadas em cada etapa, bem como a lógica empregada para garantir a funcionalidade e a integração do sistema como um todo.

### 4.1 Descrição Geral do Sistema

O sistema é dividido em três perfis principais de usuários: **administrador**, **professor** e **aluno**. Cada perfil possui permissões e funcionalidades distintas, que foram desenvolvidas para atender às necessidades específicas de cada um.

A plataforma foi idealizada para facilitar a coleta e organização de dados dos alunos de forma automatizada e digital, especialmente em contextos escolares e educacionais. Um professor pode solicitar se cadastrar no sistema preenchendo seus dados cadastrais e enviando fotos de documentos que comprovem sua identidade (CPF e RG). O administrador é responsável por analisar as solicitações de cadastro do professor, verificando os documentos e as informações cadastrais, podendo rejeitar a solicitação em caso de inconsistências ou aprová-la caso não exista nenhuma inconsistência dos dados cadastrais com os documentos. Isso garante não somente que o professor é quem está solicitando o cadastro, como que as informações cadastradas estão corretas.

Uma vez cadastrado, o professor pode criar turmas e cadastrar nela alunos, por meio da importação de uma planilha .csv contendo o CPF na primeira coluna e o e-mail na segunda coluna dos estudantes. O sistema, então, cria automaticamente os cadastros e vincula os alunos a uma turma específica. Os alunos, por sua vez, acessam a plataforma com as credenciais cadastradas e preenchem as demais informações

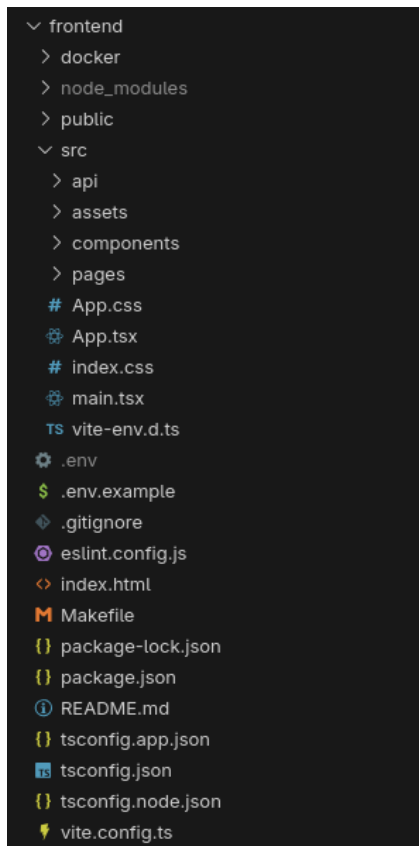
requeridas. Posteriormente, o professor pode exportar uma nova planilha, agora com os dados completos, preenchidos pelos alunos.

## 4.2 Arquitetura do Sistema

A plataforma foi arquitetada de forma modular, visando facilitar a manutenção e a evolução do sistema. No frontend, foi adotada uma arquitetura baseada em **componentes reutilizáveis** utilizando React, na qual cada parte recorrente da interface como formulários, botões, campos de entrada e elementos de navegação foi encapsulada em componentes independentes. Essa abordagem permitiu reduzir a duplicação de código e tornar a interface mais organizada.

Os componentes reutilizáveis estão centralizados no diretório **components**, enquanto cada página do sistema, como login, cadastro de professor e painéis de acesso, foi implementada em arquivos específicos no diretório **pages**. Essa separação possibilitou distinguir claramente os elementos reutilizáveis das telas completas da aplicação, facilitando ajustes pontuais e futuras expansões.

A **Figura 4.1** apresenta a organização dos diretórios do frontend, evidenciando a separação entre componentes reutilizáveis e páginas do sistema.



**Figura 4.1:** Organização dos diretórios do *frontend*.

O backend foi organizado em uma arquitetura orientada a recursos, com separação clara entre rotas, controladores, serviços e acesso aos dados. Essa estrutura permitiu isolar responsabilidades e facilitar a manutenção do código ao longo do desenvolvimento. Cada recurso representa uma funcionalidade do sistema, como autenticação, gestão de turmas, cadastro de alunos ou manipulação de arquivos, reunindo os componentes necessários para o seu funcionamento.

As **rotas** definem os endpoints da aplicação e direcionam as requisições para os **controllers**, responsáveis por tratar a entrada e saída dos dados da requisição HTTP. Os **services** concentram a lógica de negócio, como validações, criação de registros e processamento de planilhas, enquanto o acesso ao banco de dados é realizado por meio do Prisma ORM, garantindo consistência e organização das operações. Elementos como **middlewares** e funções utilitárias complementam essa estrutura, realizando validações e tarefas comuns a diferentes recursos.



A organização dos diretórios do backend segue padrões amplamente adotados em aplicações Node.js com Express e Prisma ORM, sendo uma estrutura consolidada na comunidade de desenvolvimento e não uma organização criada exclusivamente para este trabalho. A adoção desse padrão teve como objetivo facilitar a manutenção do código, a separação de responsabilidades e a escalabilidade da aplicação.

A pasta **prisma** é utilizada conforme a convenção do ORM Prisma, armazenando os modelos do banco de dados e os arquivos de migração responsáveis pela evolução do schema ao longo do desenvolvimento. Já a pasta **uploads** é destinada ao armazenamento, no servidor, de arquivos enviados pelo frontend, como imagens, permitindo que esses recursos possam ser posteriormente acessados pelo cliente por meio de rotas específicas de visualização.

O diretório **src** concentra o código-fonte da aplicação. Dentro dele, a pasta **resources** organiza as funcionalidades do sistema de acordo com o padrão de separação por recursos (*resource-based organization*), bastante comum em APIs REST. Esses recursos podem representar tanto entidades do banco de dados, como *professor*, *aluno* e *dadosUsuario*, quanto funcionalidades específicas do sistema, como *auth*, responsável pela autenticação de usuários, e *images*, voltado à manipulação de arquivos no servidor.

Cada recurso agrupa seus respectivos componentes, seguindo um padrão arquitetural conhecido:

- **controllers**, responsáveis por tratar a entrada e saída das requisições HTTP;
- **helpers**, que concentram trechos reutilizáveis de código associados ao recurso;
- **schemas**, que definem a estrutura esperada das requisições e realizam validações de tipos e campos;
- **services**, que encapsulam a lógica de acesso e manipulação dos dados no banco;
- **types**, que armazenam os tipos personalizados utilizados pelo TypeScript naquele recurso.

Além disso, cada recurso pode estar associado a um conjunto de rotas, responsáveis por definir os caminhos de acesso aos seus respectivos controllers a partir das URLs base do servidor.

Fora do diretório **resources**, a aplicação mantém pastas de uso global, também seguindo convenções comuns em projetos backend:

- **helpers**, que agrupam funções reutilizáveis por toda a aplicação;
- **middlewares**, responsáveis por validar dados das requisições e controlar o acesso às rotas;
- **routers**, onde são centralizadas as definições de rotas da aplicação;
- **utils**, que reúnem funções genéricas utilizadas em diferentes contextos do sistema.

Essa estrutura, embora não seja exclusiva deste projeto, mostrou-se adequada para organizar o código de forma clara, modular e alinhada às boas práticas recomendadas para aplicações backend modernas.

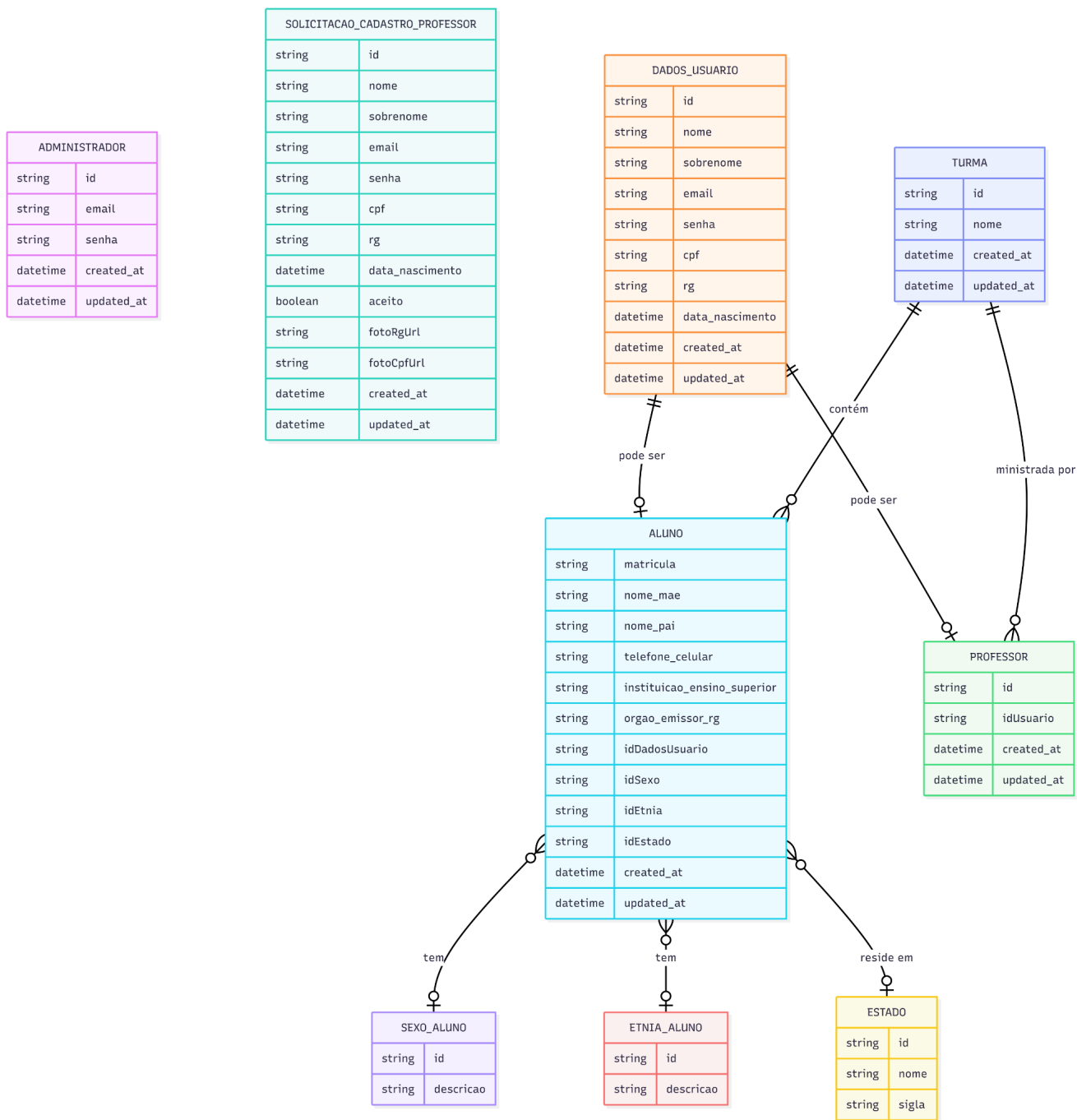
### 4.3 Banco de Dados

O banco de dados utilizado foi utilizado o MySQL, escolhido por sua robustez e ampla adoção em sistemas de médio e grande porte. Em conjunto com o MySQL, foi utilizado o Prisma ORM como ferramenta de modelagem e acesso ao banco de dados, atuando como uma camada de abstração entre a aplicação e o banco de dados relacional. O uso do ORM justifica-se devido a critérios técnicos e metodológicos que visam assegurar maior consistência, produtividade e escalabilidade ao desenvolvimento da aplicação. Diferentemente do uso de SQL puro, em que o desenvolvedor é responsável pela escrita manual das consultas, o Prisma oferece uma camada de abstração que traduz as estruturas do banco de dados em modelos tipados, permitindo maior integração com linguagens modernas como *TypeScript*. Essa característica reduz a ocorrência de erros em tempo de execução, uma vez que possíveis inconsistências podem ser identificadas já na etapa de compilação.

As entidades do banco de dados são:

- **Administrador** (representa os usuários responsáveis pelas análises das solicitações de cadastro de professores, podendo aceitar ou rejeitar uma solicitação).
- **SolicitacaoCadastroProfessor** (armazena os dados de solicitações de professores que desejam se cadastrar).
- **DadosUsuario** (entidade central que concentra informações pessoais básicas de qualquer usuário do sistema, sejam alunos ou professores).
- **Aluno** (especialização de usuário que representa os estudantes matriculados, vinculados a turmas por meio de relacionamentos muitos-para-muitos).
- **Professor** (especialização de usuário que representa os docentes, vinculados a turmas que podem ser ministradas por um ou mais professores).
- **Turma** (representa as turmas da instituição, associadas a alunos e professores em relacionamentos de muitos-para-muitos).

A **Figura 4.3**, abaixo, mostra o diagrama do modelo de dados utilizado no sistema.



**Figura 4.3:** Diagrama do modelo de dados do sistema

A modelagem do banco de dados seguiu as etapas clássicas da modelagem relacional, passando pelos níveis **conceitual**, **lógico** e **físico**. No nível **conceitual**, foi elaborado o Modelo Entidade-Relacionamento (MER), apresentado na **Figura 4.3**, no qual foram

definidas as principais entidades do sistema, seus atributos e relacionamentos, de acordo com os requisitos funcionais da aplicação.

Nesse nível, optou-se por separar os dados básicos dos usuários na entidade **DadosUsuario**, a qual concentra informações comuns tanto a alunos quanto a professores, como nome, e-mail, CPF, RG e data de nascimento. Essa decisão permitiu evitar redundâncias e facilitar a manutenção das informações compartilhadas. As entidades **Aluno** e **Professor** foram modeladas como perfis associados a **DadosUsuario**, armazenando apenas os atributos específicos de cada tipo de usuário, como matrícula e dados complementares no caso dos alunos, e a vinculação funcional no caso dos professores.

Ainda no nível conceitual, foi definida a entidade **SolicitacaoCadastroProfessor**, responsável por armazenar temporariamente os dados enviados durante o processo de solicitação de cadastro de professores. Embora compartilhe campos semelhantes aos de **DadosUsuario**, essa entidade não foi modelada como uma especialização, pois possui um papel distinto no sistema: registrar informações pessoais, documentos comprobatórios e o estado de aprovação da solicitação. Dessa forma, apenas após a aprovação por um administrador, os dados são efetivamente inseridos em **DadosUsuario**, sendo o registro correspondente removido de **SolicitacaoCadastroProfessor**, o que confere maior controle e transparência ao processo de validação.

No **nível lógico**, o modelo conceitual foi transformado em estruturas relacionais, com a definição das tabelas, chaves primárias e chaves estrangeiras, bem como a aplicação dos princípios de **normalização**, visando reduzir redundâncias e garantir a consistência dos dados. Nesse estágio, foram introduzidas tabelas auxiliares, como **Sexo\_Aluno**, **Etnia\_Aluno** e **Estado**, utilizadas para padronizar valores e facilitar futuras expansões do sistema.

Por fim, no **nível físico**, foram tomadas decisões relacionadas à implementação do banco de dados no MySQL, como o uso de identificadores do tipo UUID como chaves primárias na maioria das entidades, com exceção da entidade **Aluno**, cuja chave

primária é a matrícula. Além disso, o uso do Prisma ORM possibilitou a aplicação de migrações controladas, permitindo a evolução segura e incremental do esquema de dados conforme novas demandas da aplicação surgissem.

A modelagem adotada foi pensada para oferecer **flexibilidade**, **escalabilidade** e **manutenção simplificada**, atendendo tanto aos requisitos atuais quanto a possíveis extensões futuras do sistema.

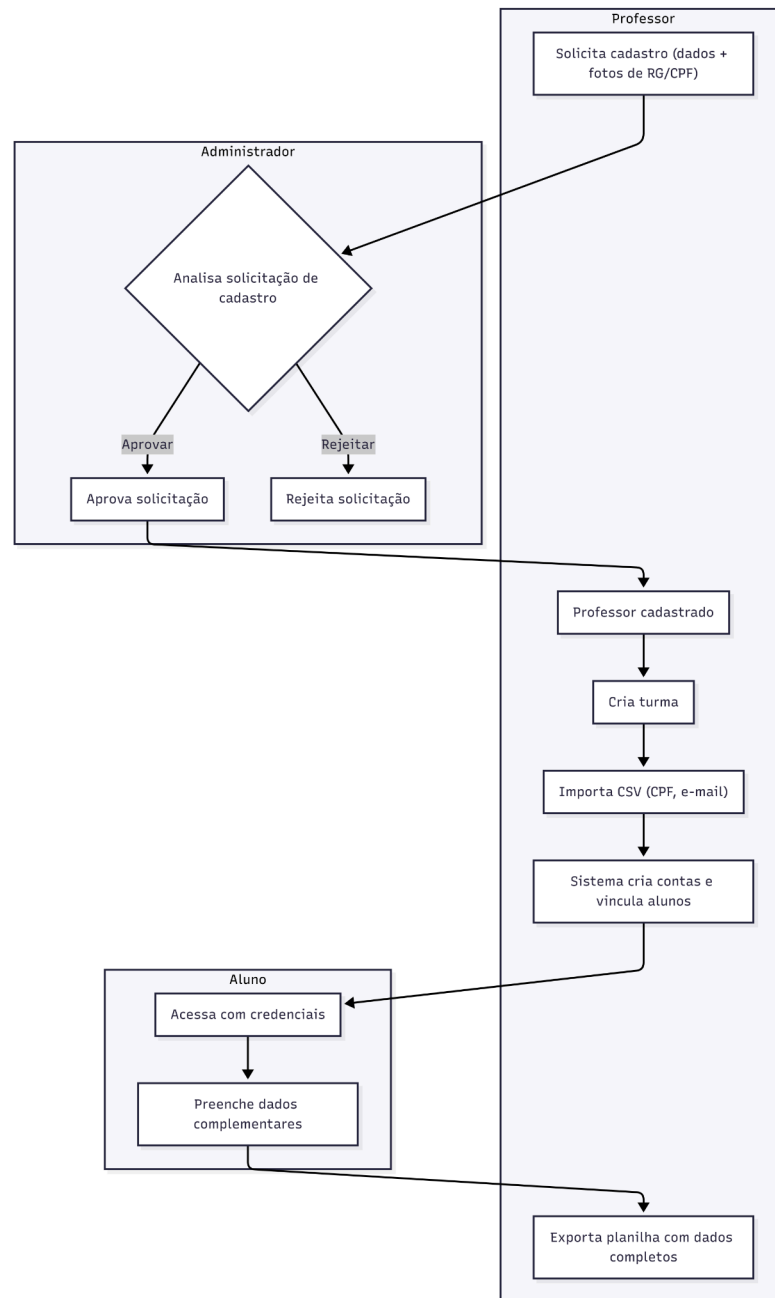
#### **4.4 Fluxo de Navegação e Experiência do Usuário**

O fluxo de navegação da aplicação foi cuidadosamente pensado para ser o mais intuitivo possível. Após o *login*, o usuário administrador é redirecionado para uma página que lista as solicitações de cadastro dos professores; ao clicar em uma das solicitações, a página de detalhes da solicitação é exibida, com os dados e a foto do professor e os botões para aceitar ou rejeitar o cadastro.

No caso do usuário professor, ele é redirecionado a uma página de visualização e criação de turmas. Ao clicar em uma das turmas listadas, é então redirecionado para a página de cadastro e exportação de dados dos alunos. Nesta página, é possível importar as planilhas para realizar os cadastros e exportar as planilhas com os dados preenchidos também é possível ver quais alunos foram cadastrados e quais já preencheram os dados. Para os alunos, o acesso é direto ao formulário de preenchimento.

Além disso, foram implementadas mensagens de *feedback*, *spinners* de progresso e validações visuais para garantir que o usuário esteja sempre ciente do que está acontecendo. A usabilidade foi prioridade em todo o projeto, reforçando a proposta de ser uma ferramenta acessível, funcional e prática.

Com essa estrutura, a plataforma atinge seu objetivo central de digitalizar, organizar e facilitar o processo de coleta e gerenciamento de dados educacionais. A **Figura 4.4** descreve bem o fluxograma do sistema.

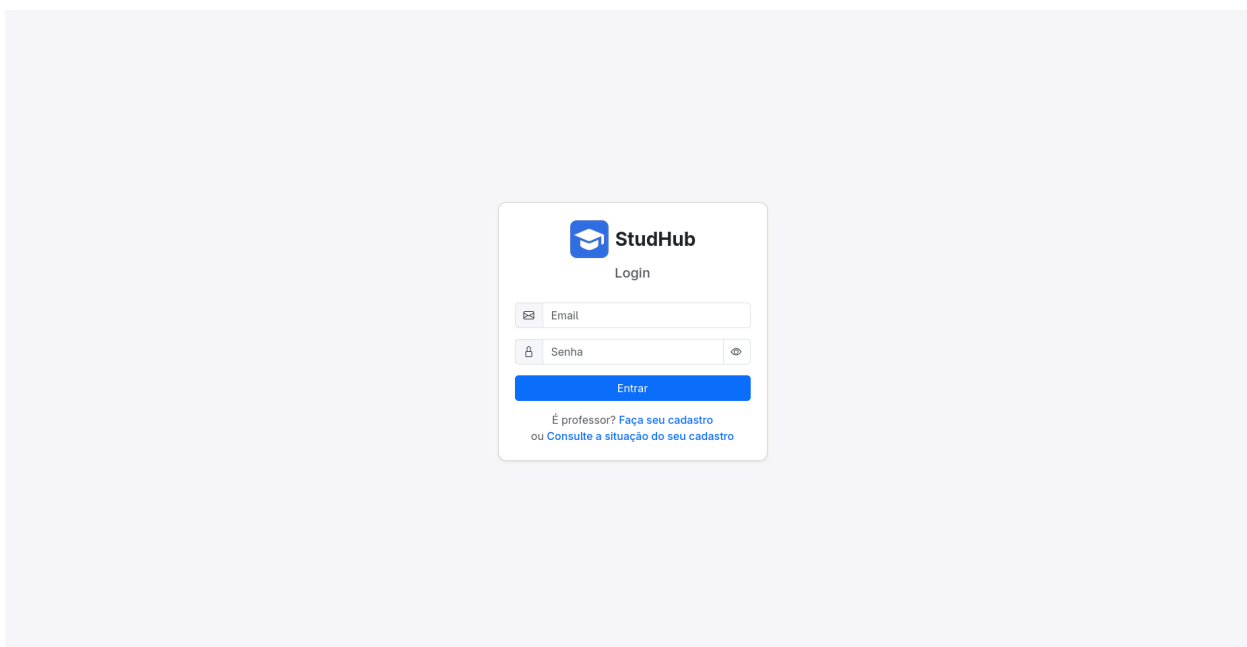


**Figura 4.4:** Fluxograma Completo do Sistema.

#### 4.5 Módulo de Autenticação de Usuários

A autenticação foi implementada com base no modelo de *login* tradicional, onde o professor acessa o sistema via e-mail e senha previamente cadastrados. No sistema,

tanto para o professor quanto para o aluno, sua senha é o próprio CPF, que é automaticamente definido como senha no momento do cadastro do usuário, uma prática para facilitar o acesso inicial. Desta forma, quando o administrador aceita uma solicitação de cadastro de um professor, um usuário com o e-mail informado pelo professor é criado, e o CPF do professor é definido como sua senha. De modo semelhante, quando o professor cadastra os alunos com a planilha, o CPF do aluno é definido como senha. As senhas são armazenadas de forma segura no banco de dados com o uso de algoritmos de *hash*. Neste caso, a biblioteca *bcrypt* é utilizada para criptografar a senha, a fim de salvá-la no banco de dados de uma forma mais segura. A **Figura 4.5** exibe a página de login de usuários

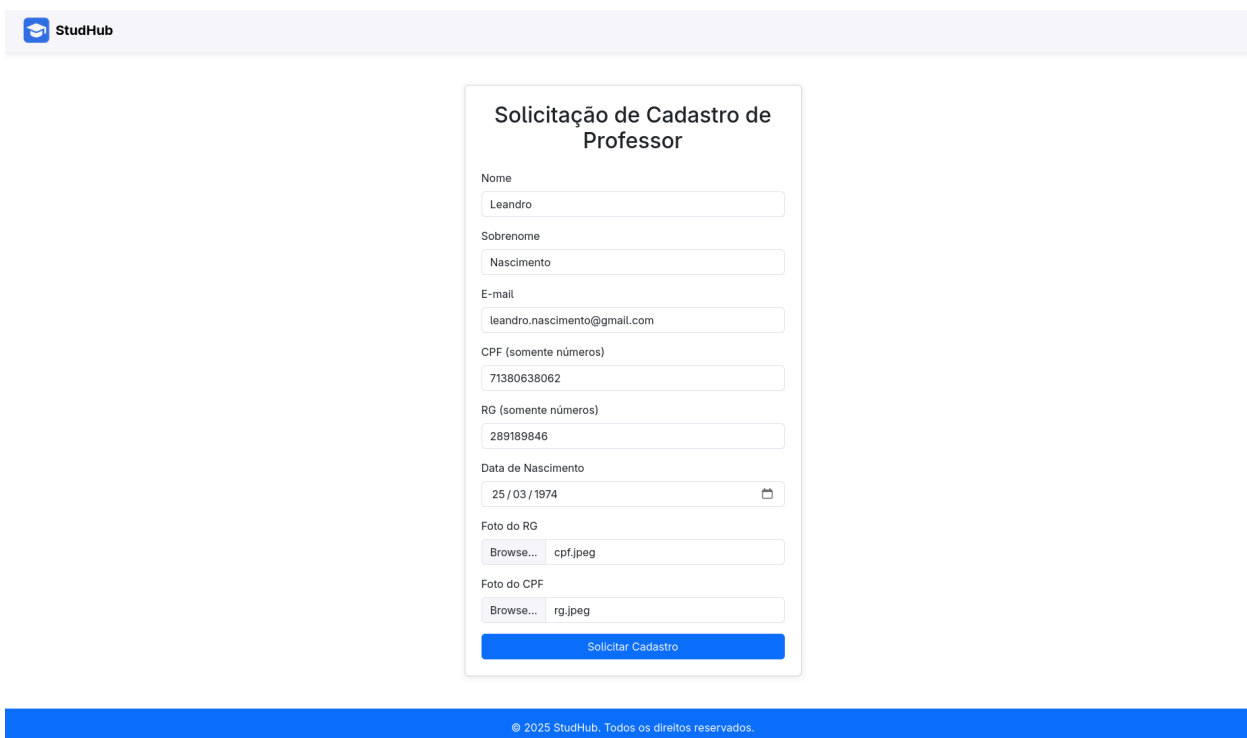


**Figura 4.5:** Tela de login da aplicação.

No servidor, foi implementado um controle de sessão, garantindo que apenas usuários autenticados pudessem acessar as rotas protegidas da aplicação. Além disso, implementou-se *middlewares* para verificação do tipo de usuário ativo no *backend*, a fim de impedir que um tipo de usuário acesse rotas com funcionalidades restritas a outros tipos de usuários. Um aluno, por exemplo, recebe como resposta do servidor um erro ao tentar acessar a rota de cadastro de alunos, que é destinada ao professor.

## 4.6 Módulo de Cadastro de Professores

O processo de cadastro de professores traz aos docentes e administradores uma experiência intuitiva. Logo na página de login, conforme exibido na **Figura 4.5** o professor encontra dois links principais: um para solicitar o cadastro e outro para verificar a situação da solicitação. Ao clicar em “Faça seu cadastro”, ele é redirecionado para um formulário no qual deve preencher algumas informações pessoais, como nome, sobrenome, CPF, RG e e-mail. Além disso, é necessário realizar o *upload* das imagens do CPF e do RG, que servirão para a validação documental. Essa interface apresenta-se de forma clara, organizada em campos de preenchimento direto seguidos pela área de envio dos documentos, conforme exibido na **Figura 4.6**, abaixo.



The image shows a web browser window with the StudHub logo in the top left corner. The main content is a form titled "Solicitação de Cadastro de Professor". The form contains the following fields and values:

- Nome: Leandro
- Sobrenome: Nascimento
- E-mail: leandro.nascimento@gmail.com
- CPF (somente números): 71380638062
- RG (somente números): 289189846
- Data de Nascimento: 25 / 03 / 1974
- Foto do RG: Browse... cpf.jpeg
- Foto do CPF: Browse... rg.jpeg

At the bottom of the form is a blue button labeled "Solicitar Cadastro". Below the form, there is a blue footer bar with the text "© 2025 StudHub. Todos os direitos reservados."

**Figura 4.6:** Interface de formulário de solicitação do professor

Já na opção “Consulte a situação do seu cadastro”, o professor pode acompanhar o andamento de sua solicitação informando apenas o CPF. O sistema, então, retorna a situação do cadastro atualizada, indicando se está em análise, se foi aprovado ou se houve recusa, proporcionando ao usuário uma forma prática e imediata de obter retorno sem a necessidade de contato adicional. A **Figura 4.7** exibe a interface de consulta da situação cadastral.

Consulta de Situação do Cadastro

71380638062 Buscar

Data da solicitação	Situação
01/10/2025	Em análise

© 2025 StudHub. Todos os direitos reservados.

**Figura 4.7:** Interface de consulta da situação de cadastro

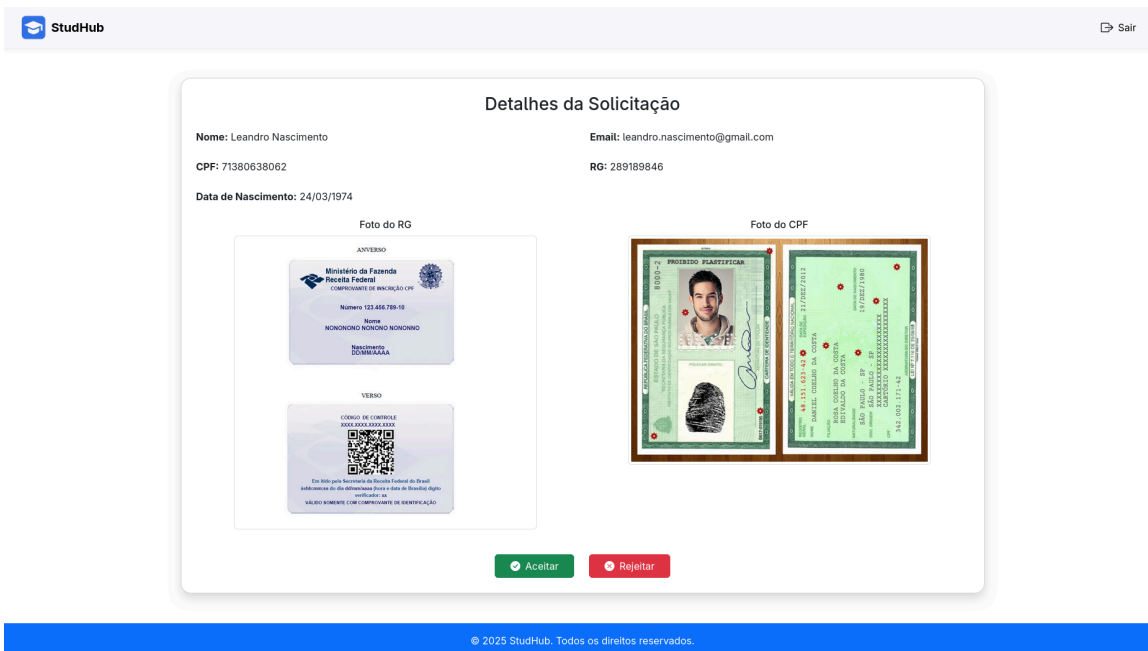
Do ponto de vista administrativo, o sistema também dispõe de funcionalidades específicas para a equipe responsável pela validação. O administrador tem acesso a uma página que lista todas as solicitações ainda não analisadas, funcionando como um painel de acompanhamento, conforme exibido na **Figura 4.8**.

## Solicitações de Cadastro de Professores

Nome do professor	Data da solicitação
Leandro Nascimento	01/10/2025

**Figura 4.8:** Página de listagem de solicitações de cadastro de professores

Ao clicar em qualquer solicitação, o administrador é redirecionado para a página de detalhes da solicitação, exibida na **Figura 4.9**, onde pode visualizar os dados preenchidos pelo professor, bem como os documentos enviados, para que analise se a solicitação de cadastro deve ser aprovada ou recusada.

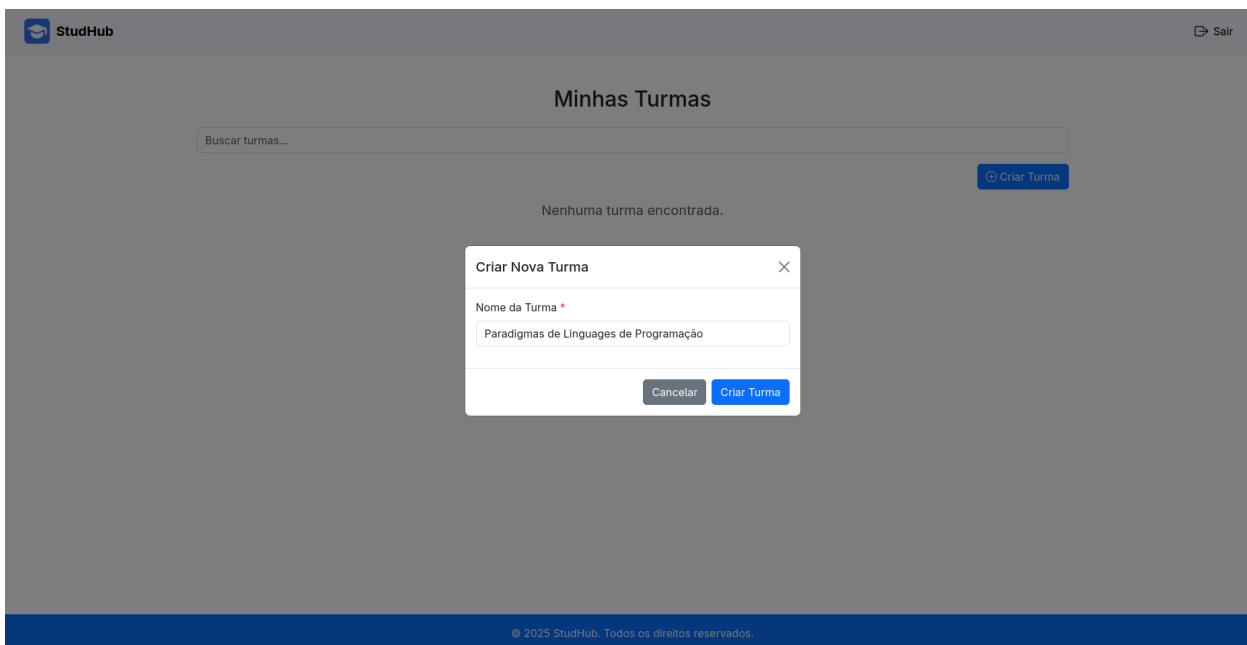


**Figura 4.9:** Página de detalhes da solicitação de cadastro de professor

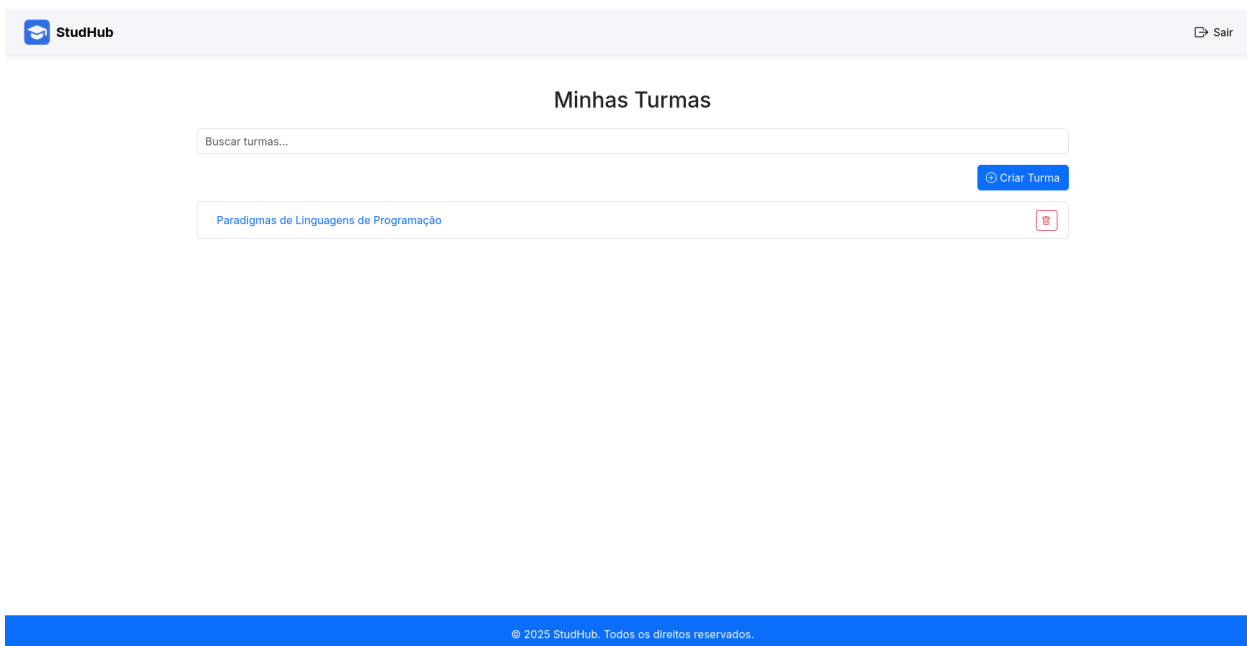
Esse fluxo de trabalho assegura que o processo de cadastramento seja ao mesmo tempo ágil para o professor e criterioso para a administração, conciliando usabilidade, segurança e transparência.

#### 4.7 Cadastro de Turmas, Alunos e Importação de Planilhas

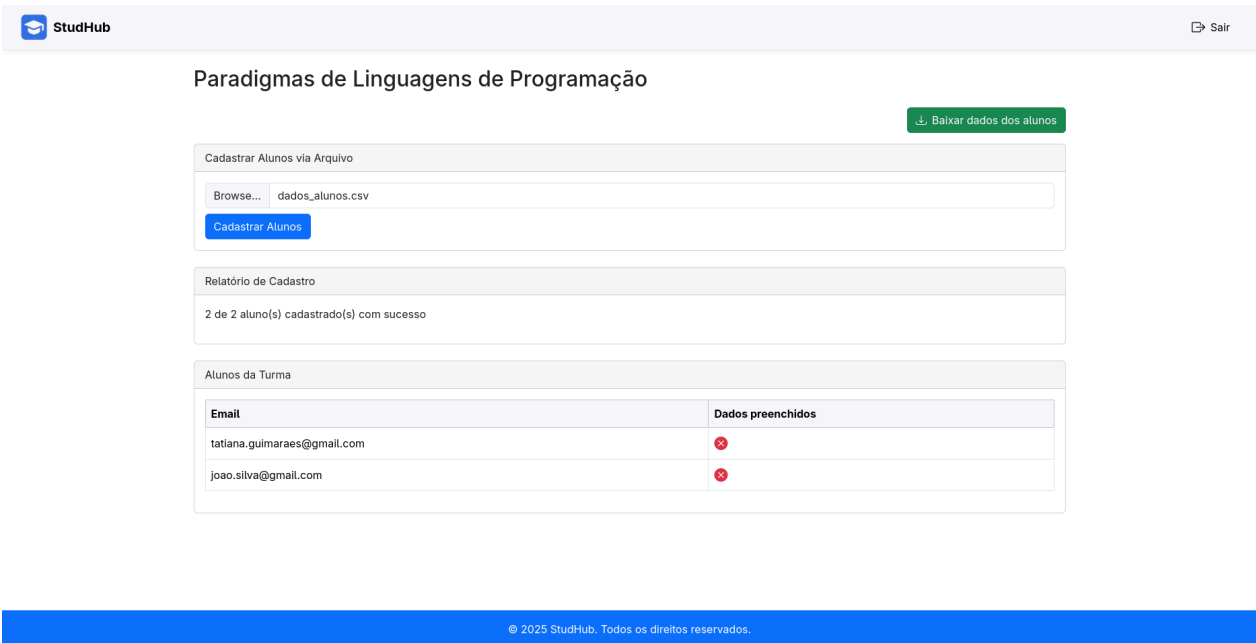
O professor tem à sua disposição uma interface intuitiva que permite o cadastro de turmas, conforme exibido na **Figura 4.10**. Uma vez criada a turma, o sistema também permite que ela seja excluída por meio de um clique no botão que possui o ícone da lixeira, conforme exibido na **Figura 4.11**. Além da página de criação de turmas, o professor tem acesso a uma interface para o *upload* de arquivos *.csv*, que é exibida na **Figura 4.12**. Essa funcionalidade foi implementada por meio de um *input* de arquivos que aceita apenas o tipo *.csv*. Uma vez que o *upload* é realizado, o arquivo é lido no *frontend* e tem suas informações inseridas nos devidos campos da requisição que é enviada para o cadastro dos alunos no servidor.



**Figura 4.10:** Interface para cadastro e listagem de turmas do professor



**Figura 4.11:** Turmas listadas na página após o cadastro



**Figura 4.12:** Interface de cadastro de alunos e exportação das planilhas com os dados preenchidos pelos alunos

Após o preenchimento de dados pelos alunos, o professor, na mesma página onde cadastrou os discentes, pode visualizar quais alunos completaram o preenchimento e, com um clique no botão “Baixar dados dos alunos”, exportar uma nova planilha com todos os dados coletados.

Essa exportação foi implementada com uso das biblioteca *xlsx*, que permite a criação de arquivos *.xlsx* diretamente *frontend*. O arquivo gerado mantém a estrutura da planilha original, acrescida das colunas com as informações adicionais.

#### 4.8 Interface de Preenchimento de Dados pelos Alunos

Uma vez logado, o aluno tem acesso a uma página personalizada onde poderá preencher os dados complementares solicitados, conforme mostrado na **Figura 4.13**. A interface foi projetada com foco em simplicidade e responsividade, permitindo que os estudantes acessem facilmente via computador, tablet ou smartphone.

61574766058

E-mail  
aluno1@gmail.com

---

**Dados Pessoais**

Nome

Sobrenome

Nome do Pai

Nome da Mãe

RG

Órgão Emissor

Estado

Data de Nascimento

---

**Contato**

Telefone Celular

---

**Instituição**

Instituição de Ensino Superior

---

**Outros Dados**

Sexo

Etnia

---

**Documentos (PDF)**

RG (PDF)  
 Nenhum arquivo escolhido

CPF (PDF)  
 Nenhum arquivo escolhido

Diploma (PDF)  
 Nenhum arquivo escolhido

Certidão de Casamento (PDF) — caso houver  
 Nenhum arquivo escolhido

**Figura 4.13:** Página de atualização de dados do aluno

Os hooks no React são funções especiais que permitem aos componentes funcionais acessar recursos que antes eram exclusivos dos componentes de classe, como gerenciamento de estado, ciclo de vida, contexto e otimizações. Eles tornam o código mais simples, reutilizável e declarativo, ao permitir que lógica de estado e efeitos

colaterais seja encapsulada em funções independentes. Entre os hooks mais utilizados estão o `useState`, para controlar valores dinâmicos dentro do componente, e o `useEffect`, responsável por lidar com operações assíncronas e reações a mudanças no estado ou nas propriedades. Esses hooks compõem a base da arquitetura moderna em React e possibilitam a criação de interfaces reativas e organizadas.

No desenvolvimento do StudHub, especialmente nos fluxos de cadastro, atualização de dados e gerenciamento de informações do usuário, a utilização dos hooks `useState` e `useEffect`, mostrou-se essencial para garantir organização, previsibilidade e boa experiência de uso.

O hook `useState` foi empregado para controlar estados locais dos componentes, como valores dos campos, indicadores de carregamento e validações dinâmicas. Em interfaces que exibem dados previamente cadastrados — como ocorre na atualização de informações do aluno — é fundamental manter um estado local reativo para refletir corretamente tanto os valores retornados do backend quanto as interações do usuário. Isso evita inconsistências entre o valor exibido, o valor editado e a lógica interna do componente.

Já o `useEffect` desempenha um papel determinante na execução de efeitos colaterais, em especial na realização de requisições ao backend. No fluxo de preenchimento automático dos formulários, o efeito é responsável por acionar a chamada à rota `/dadosUsuario`, carregar os dados existentes do estudante e armazená-los no estado local. Esse padrão garante que a interface seja sincronizada com o backend assim que o componente é montado, além de permitir o tratamento adequado de atualizações futuras caso algum parâmetro dependente do efeito venha a mudar. Assim, `useEffect` assegura que as operações assíncronas ocorram de forma controlada, evitando chamadas desnecessárias e garantindo que o formulário sempre reflita o estado real da aplicação.

# Capítulo 5

## Resultados Obtidos

Este capítulo apresenta uma análise crítica dos resultados alcançados com o desenvolvimento e testes da plataforma, destacando as principais funcionalidades implementadas, os desafios enfrentados ao longo do processo e as contribuições práticas do sistema para o contexto educacional ao qual se propõe.

### 5.1 Funcionalidades Concretizadas

A ferramenta implementada conseguiu atender a todos os requisitos funcionais propostos no planejamento inicial. A seguir, apresenta-se os resultados de cada um dos requisitos.

- Login individualizado para alunos e professores, com autenticação segura;
- Importação automatizada de planilhas .csv, com geração de contas para os alunos;
- Cadastro completo dos dados complementares realizado diretamente pelos alunos;
- Exportação final da planilha com todas as informações coletadas;
- Interface limpa, responsiva e adaptada a diferentes dispositivos;
- Painel de controle para o professor visualizar e gerenciar suas turmas.

Essas funcionalidades foram testadas individualmente e em conjunto, garantindo o correto funcionamento do sistema tanto em ambientes controlados quanto em simulações com dados reais.

## 5.2 Pontos Fortes da Plataforma

A partir da validação feita com usuários-teste (docentes e discentes), alguns pontos fortes foram identificados:

- **Automatização de processos repetitivos:** a importação e exportação de planilhas elimina a necessidade de preenchimentos manuais e reduz consideravelmente o risco de erros.
- **Interface amigável:** o uso de React e Bootstrap resultou em uma experiência visual intuitiva e moderna, permitindo que o usuário não precise de treinamento prévio.
- **Arquitetura escalável:** a separação entre frontend e backend, o uso do Prisma ORM e a organização modular do código permitem que futuras melhorias sejam realizadas com facilidade.
- **Segurança e integridade dos dados:** com autenticação por sessão e senhas criptografadas, o sistema se mostrou adequado para proteger informações sensíveis dos usuários.

## 5.3 Desafios Encontrados

Apesar dos bons resultados, o processo de desenvolvimento também apresentou dificuldades que precisaram ser superadas:

- **Leitura de planilhas com dados mal formatados:** algumas planilhas possuíam inconsistências que exigiram validações adicionais e tratamento de exceções no backend;
- **Controle de estados complexos no frontend:** a manipulação de múltiplas etapas no formulário exigiu atenção especial à lógica de navegação e sincronização dos dados entre os componentes React;
- **Ambiente de testes limitado:** por se tratar de um projeto acadêmico, os testes com usuários reais foram realizados em pequena escala, o que pode limitar a avaliação de desempenho em larga escala;

- **Hospedagem e deploy:** embora o sistema tenha sido desenvolvido localmente com sucesso, a publicação em ambiente de produção foi adiada para uma etapa futura, dependendo de infraestrutura adicional.

#### **5.4 Contribuições Práticas**

A plataforma desenvolvida representa uma solução viável, aplicável e replicável em diferentes realidades educacionais. Sua arquitetura moderna e seu foco na automação permitem:

- Reduzir a carga de trabalho dos professores;
- Aumentar a precisão e integridade dos dados estudantis;
- Oferecer aos alunos um meio ágil e seguro de preenchimento de informações;
- Servir como base para outras ferramentas escolares mais completas, como sistemas de matrícula, geração de históricos ou relatórios de desempenho.

Os resultados alcançados mostram que mesmo soluções desenvolvidas em contexto acadêmico podem alcançar níveis de aplicabilidade e relevância prática consideráveis, desde que bem planejadas e construídas com foco no usuário final.

# Capítulo 6

## Conclusões e Trabalhos Futuros

Este capítulo tem como objetivo apresentar as principais conclusões do trabalho desenvolvido, bem como refletir sobre o impacto da ferramenta proposta e indicar caminhos possíveis para a sua continuidade e aperfeiçoamento.

### 6.1 Considerações Finais

O desenvolvimento da plataforma web apresentada nesta monografia demonstrou, na prática, como tecnologias modernas podem ser aplicadas para solucionar problemas concretos no contexto educacional. A proposta de facilitar a coleta, organização e exportação de dados estudantis foi plenamente atendida, com a criação de um sistema funcional, responsivo, seguro e de uso intuitivo para professores e alunos.

A construção da aplicação possibilitou também um exercício aprofundado de integração entre múltiplas tecnologias – React, Node.js, Typescript, PrismaORM e MySQL –, promovendo uma compreensão sólida de como essas ferramentas podem trabalhar juntas de forma coesa. Além disso, o processo contribuiu para o desenvolvimento de habilidades importantes no campo da Engenharia de Software, como levantamento de requisitos, modelagem de dados, organização de código, testes, validação com usuários e documentação.

A experiência obtida demonstra que soluções tecnológicas acadêmicas podem ter impacto real se alinhadas a demandas existentes e projetadas com foco no usuário final. A adoção de boas práticas de desenvolvimento e uma abordagem metodológica clara contribuíram diretamente para o sucesso do projeto.

## 6.2 Trabalhos Futuros

Embora o sistema tenha cumprido seus objetivos iniciais, algumas possibilidades de melhoria e expansão foram identificadas ao longo do processo de desenvolvimento. Dentre elas, destacam-se:

- **Hospedagem em ambiente de produção:** disponibilizar a aplicação em um servidor real, com domínio e certificado SSL, para permitir seu uso contínuo por instituições de ensino;
- **Painel de acompanhamento em tempo real:** permitir que professores visualizem, em tempo real, quais alunos já preencheram seus dados e quais ainda estão pendentes;
- **Notificações por e-mail:** implementar envio automático de e-mails aos alunos lembrando-os do prazo de preenchimento;
- **Recuperação de senha e atualização de dados pessoais:** dar maior autonomia aos usuários para gerenciar seus dados com segurança;
- **Dashboard analítico:** incluir gráficos e indicadores que ajudem os professores a visualizar estatísticas gerais das turmas;
- **Integração com sistemas educacionais existentes:** permitir exportação para formatos compatíveis com softwares de gestão escolar.

Essas melhorias podem ser implementadas em etapas futuras e incrementariam significativamente o valor da ferramenta como produto. Além disso, podem servir como ponto de partida para novos projetos que tenham como foco a transformação digital da educação.

O presente trabalho, portanto, além de representar uma solução prática, também se consolida como base para explorações mais amplas no desenvolvimento de sistemas inteligentes e acessíveis para contextos educacionais reais.

## Referências Bibliográficas

[1] Leão, Luan Leite, et al. "Banco de dados aplicados em processos de pesquisa com redes neurais artificiais:: uma alternativa às planilhas eletrônicas." *Observatorio de la Economía Latinoamericana* 22.12 (2024): 59.

[2] CONSÓRCIO WORLD WIDE WEB – W3C. Architecture of the World Wide Web, Volume One. 2004. Disponível em: <https://www.w3.org/TR/webarch/>.

[3] React – A JavaScript library for building user interfaces. Meta Platforms, Inc. Disponível em: <https://reactjs.org/>.

[4] WORLD WIDE WEB CONSORTIUM (W3C). Document Object Model (DOM) Level 2 Specification. 2000. Disponível em: <https://www.w3.org/TR/DOM-Level-2-Core/>. Acesso em: 8 dez. 2025.

[5] Bootstrap – The most popular HTML, CSS, and JS library. The Bootstrap Authors. Disponível em: <https://getbootstrap.com/>.

[6] Node.js – JavaScript runtime built on Chrome's V8 engine. OpenJS Foundation. Disponível em: <https://nodejs.org/>.

[7] Express – Fast, unopinionated, minimalist web framework for Node.js. OpenJS Foundation. Disponível em: <https://expressjs.com/>.

[8] TypeScript – Typed JavaScript at Any Scale. Microsoft. Disponível em: <https://www.typescriptlang.org/>.

[9] Prisma ORM – Modern database toolkit. Prisma Data, Inc. Disponível em: <https://www.prisma.io/>.

[10] MySQL – The world's most popular open source database. Oracle Corporation. Disponível em: <https://www.mysql.com/>.

[11] SheetJS – Excel for the Web. SheetJS LLC. Disponível em: <https://sheetjs.com/>.

[12] ExcelJS – Excel Workbook Manager. GitHub – exceljs/exceljs. Disponível em: <https://github.com/exceljs/exceljs>.

[13] bcrypt – A library to help you hash passwords. GitHub – bcrypt. Disponível em: <https://github.com/kelektiv/node.bcrypt.js/>.

[14] JUNIOR, ANTONIO CARLOS GONÇALVES. "UM ESTUDO DE SEGURANÇA DA INFORMAÇÃO: INJEÇÃO DE SQL."

[15] SUTHERLAND, Jeff. *Scrum: A Arte de Fazer o Dobro de Trabalho na Metade do Tempo*. 2. ed. Porto Alegre: Bookman, 2014.

[16] Anderson, David J. *Kanban: Successful Evolutionary Change for Your Technology Business*. Blue Hole Press, 2010.

## Apêndice A – Estrutura do Banco de Dados

Este apêndice apresenta a **estrutura lógica do banco de dados** utilizado na aplicação, conforme o **Modelo Entidade–Relacionamento (MER)** adotado. O banco foi modelado utilizando o **Prisma ORM** e implementado em um sistema gerenciador de banco de dados relacional **MySQL**, respeitando princípios de normalização, integridade referencial e organização dos dados.

### Tabela: ADMINISTRADOR

Armazena os dados dos usuários responsáveis pela administração do sistema.

- **id** (string, PK)
- **email** (string)
- **senha** (string)
- **created\_at** (datetime)
- **updated\_at** (datetime)

### Tabela: SOLICITACAO\_CADASTRO\_PROFESSOR

Responsável por armazenar as solicitações de cadastro de professores, permitindo o controle e a validação administrativa antes da criação efetiva do usuário no sistema.

- **id** (string, PK)
- **nome** (string)
- **sobrenome** (string)
- **email** (string)
- **senha** (string)
- **cpf** (string)
- **rg** (string)
- **data\_nascimento** (datetime)
- **aceito** (boolean)
- **fotoRgUrl** (string)
- **fotoCpfUrl** (string)

- **created\_at** (datetime)
- **updated\_at** (datetime)

### **Tabela: DADOS\_USUARIO**

Armazena os dados básicos dos usuários efetivamente cadastrados no sistema, independentemente do perfil (aluno ou professor).

- **id** (string, PK)
- **nome** (string)
- **sobrenome** (string)
- **email** (string)
- **senha** (string)
- **cpf** (string)
- **rg** (string)
- **data\_nascimento** (datetime)
- **created\_at** (datetime)
- **updated\_at** (datetime)

### **Tabela: PROFESSOR**

Representa o perfil de professor no sistema, associado a um registro em **DADOS\_USUARIO**.

- **id** (string, PK)
- **idUsuario** (string, FK → DADOS\_USUARIO.id)
- **created\_at** (datetime)
- **updated\_at** (datetime)

### **Tabela: TURMA**

Armazena as informações das turmas criadas no sistema, associadas a um professor responsável.

- **id** (string, PK)
- **nome** (string)
- **created\_at** (datetime)
- **updated\_at** (datetime)

Relacionamento:

- Uma turma é **ministrada por** um professor.
- Uma turma **contém** múltiplos alunos.

### Tabela: ALUNO

Armazena os dados específicos dos alunos, vinculados aos seus dados básicos e a informações complementares.

- **matricula** (string, PK)
- **nome\_mae** (string)
- **nome\_pai** (string)
- **telefone\_celular** (string)
- **instituicao\_ensino\_superior** (string)
- **orgao\_emissor\_rg** (string)
- **idDadosUsuario** (string, FK → DADOS\_USUARIO.id)
- **idSexo** (string, FK → SEXO\_ALUNO.id)
- **idEtnia** (string, FK → ETNIA\_ALUNO.id)
- **idEstado** (string, FK → ESTADO.id)
- **created\_at** (datetime)
- **updated\_at** (datetime)

### Tabela: SEXO\_ALUNO

Tabela auxiliar utilizada para normalizar os dados referentes ao sexo do aluno.

- **id** (string, PK)

- **descricao** (string)

### **Tabela: ETNIA\_ALUNO**

Tabela auxiliar responsável pelo armazenamento das opções de etnia dos alunos.

- **id** (string, PK)
- **descricao** (string)

### **Tabela: ESTADO**

Armazena os estados federativos associados ao endereço do aluno.

- **id** (string, PK)
- **nome** (string)
- **sigla** (string)

### **Considerações sobre o Modelo**

O modelo adotado separa **dados de autenticação e identificação básica (DADOS\_USUARIO)** dos **dados específicos de cada perfil**, permitindo flexibilidade e reaproveitamento das informações. O uso de tabelas auxiliares para sexo, etnia e estado contribui para a normalização do banco de dados e reduz redundâncias.

Essa estrutura favorece a escalabilidade do sistema, possibilitando a inclusão de novos perfis de usuários ou atributos adicionais sem a necessidade de grandes reestruturações no banco de dados.

## **Apêndice B – Fluxo Geral do Sistema**

### **Fluxo para o Professor:**

1. Login na plataforma com e-mail e senha;
2. Criação de nova turma via formulário;
3. Importação de planilha com lista de alunos (e-mail e CPF);
4. Acompanhamento do preenchimento pelos alunos;
5. Exportação da planilha final com todos os dados.

### **Fluxo para o Aluno:**

1. Login com e-mail e CPF;
2. Acesso ao formulário de preenchimento dos dados complementares;
3. Submissão dos dados e confirmação de envio.

Esse fluxo é simples, direto e garante que ambos os perfis realizem suas tarefas com facilidade e autonomia. O sistema guia o usuário com mensagens de sucesso, feedbacks de erro e interfaces responsivas, contribuindo para uma boa experiência geral de uso.