Desenvolvimento de Software *Web* para Gerenciamento de Dados e Documentos de Estudantes

Matheus Mayrer[1], Dr. Rafael Vieira Coelho[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS) – Campus Farroupilha - 95174-274– Farroupilha – RS – Brazil

matheusmayrer20@gmail.com, rafael.coelho@farroupilha.ifrs.edu.br

Abstract. *Educational institutions, especially public ones, frequently face challenges with the manual management of student data and documents, a scenario susceptible to errors, information loss, and high time consumption. The absence of adequate technological tools results in low-efficiency methods, directly impacting the quality of academic management. This paper describes the development of a web software for the management of student data and documents, aiming to modernize and optimize these administrative processes. The system's objective is to centralize information and standardize school procedures on a practical and intuitive platform. The developed solution allows for the digital archiving of documents, ensuring data integrity and traceability, as well as the complete management of the student lifecycle. The system operates with distinct access profiles, providing personalized interfaces according to the users' roles. For the software's construction, a client-server architecture was used, with the backend developed in Java Spring and the frontend in React, presenting itself as a strategic tool that meets the demand for modernization in the educational sector.*

Keywords: *technology, web system, school management*

Resumo. As instituições de ensino, principalmente públicas, frequentemente enfrentam desafios com o gerenciamento manual de dados e documentos de estudantes, um cenário suscetível a falhas, perdas de informações e alto consumo de tempo. A ausência de ferramentas tecnológicas adequadas resulta em métodos de baixa eficiência, impactando diretamente a qualidade da gestão acadêmica. O presente trabalho descreve o desenvolvimento de um software *web* para o gerenciamento de dados e documentos de estudantes, visando modernizar e otimizar estes processos administrativos. O objetivo do sistema é centralizar as informações e padronizar os procedimentos escolares em uma plataforma prática e intuitiva. A solução desenvolvida permite o arquivamento digital de documentos, garantindo a integridade e rastreabilidade dos dados, bem como a gestão completa do ciclo estudantil. O sistema opera com perfis de acesso distintos, disponibilizando interfaces personalizadas de acordo com os cargos dos usuários. Para a construção do software foi utilizada a arquitetura cliente-servidor, com o *backend* desenvolvido em *Java Spring* e o *frontend* em *React*, se apresentando como uma ferramenta estratégica que atende a demanda por modernização no setor educacional.

Palavras-Chave: tecnologia, sistema *web*, gestão escolar

1. Introdução

O ordenamento jurídico brasileiro, pela lei nº 8.159, de 8 de janeiro de 1991, dispõe sobre a preservação de documentos públicos e privados, garantindo que estes sejam

preservados de forma íntegra e acessível no longo prazo (Brasil, 1991). Os riscos de extravio e degradação de documentos decorrentes de métodos tradicionais de arquivamento, bem como estratégias para minimizá-los, são questões frequentemente debatidas (Figueirêdo; Negreiros, 2025, p. 02).

A falta de acesso a ferramentas tecnológicas impõe a necessidade de gerenciamento manual de dados como registros acadêmicos e recursos humanos, gerando um cenário suscetível a falhas e imprecisões, com baixa eficiência e alto consumo de tempo (Calseverini; Silva, 2024, p. 01). Esse fenômeno também se reflete nas escolas, especialmente nas públicas. Diante disso, é fundamental a atuação integrada de profissionais de diversas áreas, promovendo o acesso e o uso das tecnologias em prol da educação (Ferreira; Rosado; Carvalho, 2017, p. 93).

A partir de tal análise, estruturou-se o presente projeto, sob a justificativa da necessidade de informatização na gestão dos dados escolares, acompanhando a evolução tecnológica dos demais setores da sociedade.

A implementação de um sistema para gestão de dados e documentos no âmbito estudantil visa atender à demanda por soluções tecnológicas modernas e eficientes, facilitando a administração e preservação de documentos, além de aprimorar a qualidade do ensino e dos serviços oferecidos aos alunos e à comunidade. O sistema proposto integra e padroniza processos em instituições de ensino, garantindo controle e rastreamento eficaz dos dados dos estudantes. O software permite a gestão das escolas e suas turmas, transferências entre escolas, atribuição de notas e controle de frequência dos discentes. Auxilia ainda no armazenamento seguro e padronizado de documentos.

O desenvolvimento do software foi pensado visando sua implantação em municípios e estados que não possuem sistemas integrados e digitais na educação e no gerenciamento dos dados de seus estudantes de ensino fundamental e médio, servindo como apoio aos professores, secretários de escola e demais funcionários vinculados à educação. Este poderá, ainda, ser aplicado em redes privadas de ensino, embora não seja o foco principal do projeto.

Este trabalho é dividido em tópicos. O referencial teórico apresenta a fundamentação bibliográfica do projeto; em seguida, no campo metodológico, são expostas as análises de requisitos, modelagem de diagramas, as tecnologias e os procedimentos que foram utilizados no desenvolvimento do software; no desenvolvimento, são abordados alguns tópicos sobre a implementação do projeto; após, nos resultados, são demonstradas as interfaces e funcionalidades. O trabalho é concluído com as considerações finais juntamente de sugestões de trabalhos futuros e as referências bibliográficas utilizadas.

2. Referencial Teórico

Os sistemas informatizados são encontrados nos diversos ramos da sociedade, estando presentes em diversas áreas e sendo essencial para o mundo moderno (Sommerville, 2011, p. 2). Além disso, não atuam apenas como produto final, mas também figuram como um mecanismo intermediário indispensável, gerando e gerenciando informações (Pressman, 2001, p. 4-9).

O desenvolvimento de um software pode ser bastante complexo e, por isso, a engenharia de software estuda, entre outras coisas, os padrões arquiteturais (Sommerville, 2011, p. 108). Esses padrões dividem as aplicações em grandes partes denominadas componentes e preocupam-se em como apresentar, organizar e estruturar, definindo regras sobre as comunicações que ocorrem dentro do sistema e entre o sistema e o usuário (Valente, 2020, p. 153-155).

Um dos padrões de arquitetura mais conhecido em serviços *web* é o *model-view-controller* (MVC), sendo uma base para o gerenciamento de interação (Sommerville, 2011, p. 108). Ele organiza o código em três componentes principais: o modelo, responsável por gerenciar os dados e a lógica de negócios; a visão, que apresenta a interface gráfica ao usuário e recebe suas interações; e o controlador, que interpreta as entradas do usuário, atualiza o modelo e ajusta a visão conforme necessário (Valente, 2020, p. 158-162).

Outro padrão muito utilizado, principalmente em sistemas distribuídos, é o cliente-servidor (Sommerville, 2011, p. 113). Neste padrão, a estrutura é composta por um conjunto de servidores, que oferecem serviços, e um conjunto de clientes, que requisitam e consomem esses serviços (Valente, 2020, p. 173). A troca de informações entre as partes nessa arquitetura é viabilizada por protocolos que trafegam em uma rede, de modo que permita a independência dos componentes, ou seja, que estes sejam modificados sem afetar o sistema como um todo (Sommerville, 2011, p. 113-114).

A transferência dos dados ocorre por protocolos como o protocolo de transferência de hipertexto (HTTP), um dos principais utilizado na internet para a transferência de dados entre clientes e servidores (Silva, 2024, p. 13). As comunicações entre aplicações e serviços ocorrem, geralmente, por meio de interfaces de programação de aplicações (API), permitindo que dois programas consigam trocar dados de maneira padronizada (Pandolfo, 2024, p. 15).

Para organizar e padronizar essa comunicação via HTTP foi proposto o estilo arquitetural *Representational State Transfer (*REST), estabelecendo diretrizes para a construção e integração de API na internet. (Pita, 2024, p 5-9). De acordo com essas diretrizes, a comunicação deve ser *stateless,* cada requisição deve conter todos os dados necessários para que o servidor consiga processar a resposta corretamente (Mendes, 2024, p 40-41). Além da comunicação sem estado, o princípio da interface uniforme dita que a interação com todos os recursos da API deve seguir um modelo consistente, utilizando os métodos HTTP de forma padronizada (Pita, 2024, p 8-16).

No que diz respeito ao emprego de tecnologias no âmbito da educação, de acordo com o censo escolar de 2023, realizado pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP), o Brasil tem aumentado o percentual de escolas de ensino fundamental com acesso à internet (Brasil, 2023). No entanto, as escolas da rede municipal, apesar de representarem a maior parte das instituições de ensino fundamental, são as que menos dispõem de recursos tecnológicos.

Dessa forma, a estrutura escolar, bem como o trabalho de professores e educadores, tem enfrentado constantes modificações, influenciadas pelo processo de globalização e pela evolução das tecnologias. (Barreto, 2004, p. 1183). É fundamental a redução das lacunas existentes entre as atividades desenvolvidas no ambiente escolar e as práticas sociais que permeiam o cotidiano de docentes e discentes, tendo em vista que, no dia a dia, o acesso a ferramentas tecnológicas que potencializam a realização de tarefas é amplo, sendo indispensável sua integração ao contexto educacional (Moreira; Kramer, 2007, p. 1037).

3. Metodologia

A primeira etapa realizada foi relacionada a engenharia de requisitos, uma etapa fundamental no desenvolvimento de sistemas, envolvendo a elicitação, análise, especificação e verificação destes (Sommerville, 2011, p. 60). Após essa fase, os requisitos foram documentados e validados para garantir que estavam corretos e sem inconsistências (Valente, 2020, p. 28-34).

Com os requisitos documentados e organizados, foi possível avançar para a modelagem do sistema, criando assim representações detalhadas das funcionalidades e da estrutura do sistema, preenchendo lacunas existentes entre os requisitos abstratos. Esses modelos auxiliam no entendimento, análise e validação das soluções propostas (Valente, 2020, p. 59-65).

Com relação a metodologia de desenvolvimento, optou-se pelo Scrum devido aos diversos benefícios, principalmente pela flexibilidade e adaptação a mudanças, bem como a possibilidade da melhoria contínua (Sommerville, 2011, p. 39-53).

3.1 Elicitação e Classificação dos Requisitos

A elicitação dos requisitos é uma etapa essencial para o planejamento e desenvolvimento de sistemas, pois estes definem as funcionalidades e características esperadas. A identificação e categorização permite alinhar as necessidades dos usuários com o resultado final entregue.

No sistema em questão, os requisitos foram organizados em três categorias principais: requisitos funcionais (RF), que descrevem as funcionalidades específicas para diferentes perfis de usuários; requisitos não funcionais (RNF), que detalham atributos de qualidade e restrições técnicas; e regras de negócio (RN), que estabelecem as políticas e restrições relacionadas à aplicação.

Os requisitos funcionais foram organizados de acordo com os cargos dos usuários, sendo dividido em requisitos funcionais de secretário (RFS), requisitos funcionais de professor (RFP) e requisitos funcionais de administrador (RFA), que são os três cargos para os usuários do sistema.

Requisitos funcionais de secretário (RFS):

- RFS1: Deve ser possível matricular novos alunos;
- RFS2: Deve ser possível rematricular alunos antigos;
- RFS3: Deve ser possível transferir alunos de outras escolas;

- RFS4: Deve ser possível atualizar dados de alunos;
- RFS5: Deve ser possível pesquisar e visualizar dados dos alunos;
- RFS6: Deve ser possível gerenciar turmas nas escolas;
- RFS7: Deve ser possível gerenciar disciplinas nas turmas das escolas;
- RFS8: Deve ser possível gerenciar os documentos dos estudantes.

Requisitos funcionais de administrador (RFA):

- RFA1: Deve ser possível gerenciar escolas;
- RFA2: Deve ser possível gerenciar usuários.

Requisitos funcionais de professor (RFP):

- RFP1: Deve ser possível gerenciar as notas dos alunos;
- RFP2: Deve ser possível gerenciar a frequência dos alunos;
- RFP3: Deve ser possível pesquisar e visualizar dados dos alunos;
- RFP4: Deve ser possível pesquisar e visualizar dados das turmas;
- RFP5: Deve ser possível gerenciar os documentos dos estudantes.

Requisitos não funcionais:

- RNF1: O sistema deve possuir uma interface responsiva;
- RNF2: O sistema deve ser desenvolvido para a *web;*
- RNF3: As senhas dos usuários devem ser criptografadas;
- RNF4: O sistema deve ser intuitivo e de fácil uso para os diferentes perfis.

Regras de negócio:

- RN1: Um aluno não pode estar matriculado em duas escolas ao mesmo tempo;
- RN2: Uma turma não pode ser concluída com matrículas de estudantes ativas;
- RN3: Somente usuários “administrador” podem cadastrar escolas e novos usuários;
- RN4: Os usuários “administrador” devem possuir todas as permissões dos usuários “secretário”;
- RN5: Somente usuários “professor” podem atribuir notas às disciplinas que os estudantes estão matriculados;
- RN6: Somente usuários “secretário” e “administrador” podem matricular, rematricular e transferir estudantes;
- RN7: O acesso ao sistema deverá conter verificações de credenciais.

3.2. Casos de Uso

Com base nos requisitos elencados, os casos de usos foram elaborados para atender as demandas do sistema. A confecção do diagrama de casos de uso ocorreu utilizando da ferramenta *diagrams.net* (Diagrams, 2024). Tal representação visual, apresentada na

Figura 1, auxilia no entendimento das principais funcionalidades do sistema e nos atores que possuem as permissões para realizá-las.

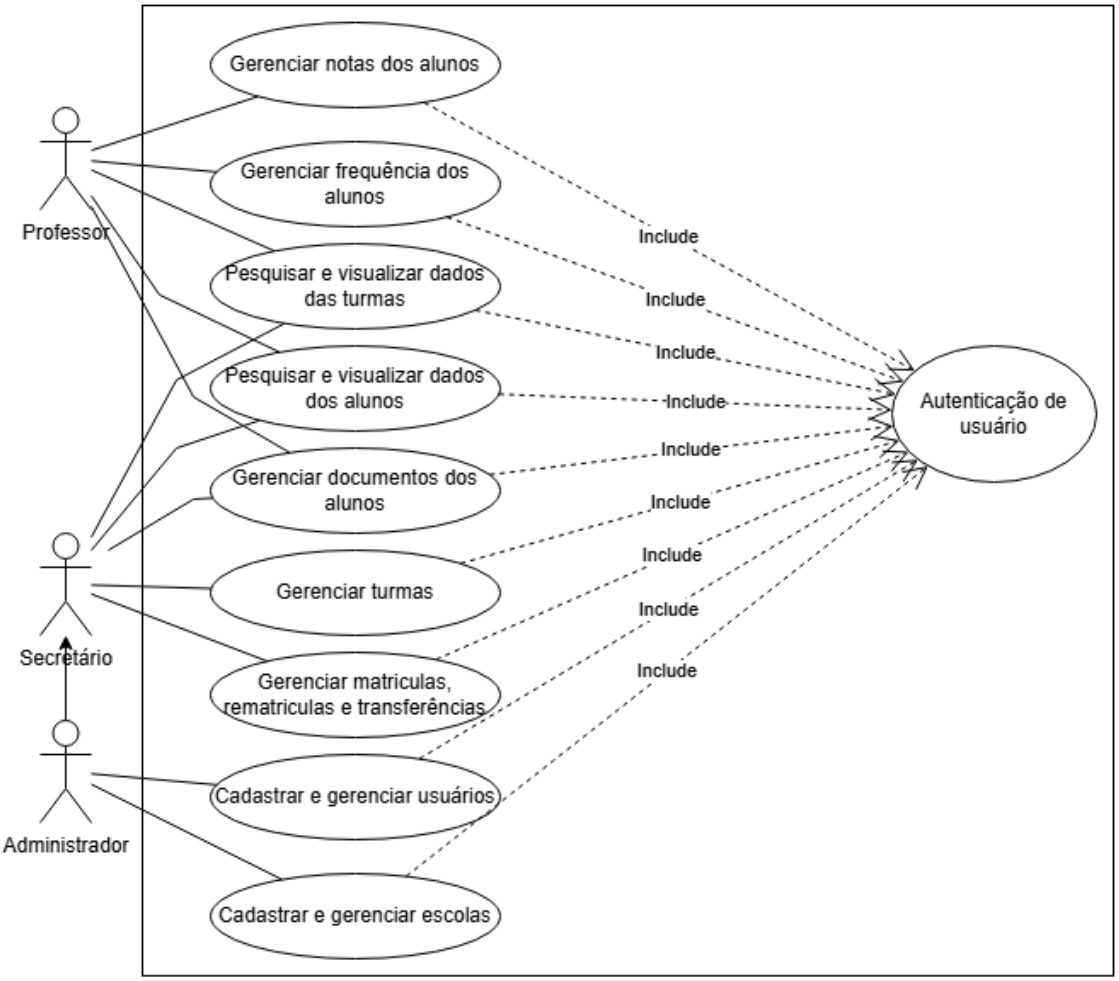

Figura 1: Diagrama de casos de uso. Fonte: Autor, 2024.

O diagrama de casos de uso apresenta os três atores do sistema e as suas principais funcionalidades. Para a utilização do sistema é necessário a autenticação do usuário, uma vez que todos os casos de uso possuem uma relação de inclusão com a "Autenticação de usuário".

3.3. Modelo Entidade-Relacionamento

O modelo entidade-relacionamento (MER), apresentado na Figura 2, foi confeccionado com base nos casos de uso e nos requisitos do projeto. Para a elaboração foi utilizada a ferramenta *dbdiagram* (Holistics, 2024).

Figura 2: Diagrama entidade relacionamento. Fonte: Autor, 2024.

Por meio do diagrama, é possível visualizar e analisar a estrutura e os relacionamentos entre as entidades do banco de dados. O modelo expõe os atributos e o tipo de dado que será atribuído a estes, garantindo a integridade e a consistência das informações.

3.4. Tecnologias

O *backend* do sistema, onde estão as principais lógicas acerca do funcionamento, foi desenvolvido em *Java*, utilizando o framework *Spring*. Essa ferramenta oferece um modelo abrangente de programação e configuração para aplicativos empresariais, facilitando o desenvolvedor a focar na lógica de negócio. Além disso, oferece suporte robusto ao desenvolvimento de APIs e aplicações MVC, podendo lidar com cargas de trabalho intensas e sendo altamente escalável (Spring, 2024).

No *frontend* do software empregou-se o *React*, uma biblioteca *JavaScript* para a construção de interfaces de usuários *web*. A estrutura é composta de componentes reutilizáveis, o que resulta em uma construção mais organizada e modular. Sua

utilização permite uma interação rápida e eficiente entre o usuário e a aplicação, melhorando a experiência do uso por meio de atualizações de interface em tempo real (React, 2024).

Para a gestão dos dados foi utilizado o *MySQL*, um banco de dados relacional que utiliza a linguagem *Structured Query Language* (SQL) padrão para manipulação e consulta de dados estruturados. Entre as principais características, é possível citar a flexibilidade e escalabilidade, sendo compatível com diversos sistemas operacionais (Oracle, 2024). Ainda com relação a dados, os documentos dos usuários são armazenados em diretórios organizados de acordo com o identificador do mesmo.

Para auxiliar no desenvolvimento, foi utilizado *Integrated Development Environment* (IDE), ambiente que oferece ferramentas de depuração, complemento de código, suporte a várias linguagens e integração com sistemas de controle de versão. Com essa finalidade, o *Intellij IDEA* e o *Visual Studio Code* foram utilizados, tornando a construção do software mais eficiente e organizada (Jetbrains, 2024; Microsoft, 2024). Além disso, a modelagem das entidades e dos casos de uso foi realizada com o auxílio de ferramentas como o *diagrams*, que disponibiliza formas e modelos, e o *dbdiagram,* específico para diagramas de banco de dados (Diagrams, 2024; Holistics, 2024).

4. Desenvolvimento

O software desenvolvido consiste em uma aplicação *web*, acessível por meio da internet ou, em casos e ambientes específicos dos usuários, por uma rede local. O sistema contém uma autenticação inicial, responsável por atribuir permissões de acesso a determinadas funcionalidades de acordo com o cargo do usuário. Essa abordagem visa garantir segurança e evitar erros na utilização.

Para assegurar o correto funcionamento da aplicação *web*, é essencial que todos os seus componentes estejam devidamente hospedados em um serviço de qualidade. Isso possibilita o acesso à aplicação a partir de dispositivos comuns como celulares, computadores e notebooks, que estão presentes na rotina dos usuários alvos.

O principal paradigma utilizado no sistema é a programação orientada a objetos, amplamente empregada em algoritmos *Java*. Além disso, alguns algoritmos clássicos da literatura estão presentes como o de autenticação, controle de acesso, criptografia de dados, operações básicas de armazenamento de dados e ordenação.

4.1. Implementação

As bibliotecas de códigos, também conhecidas como dependências de um projeto, fornecem funcionalidades adicionais para o desenvolvimento, manipulação de dados e conexões com ferramentas externas. Para a implementação foram utilizadas bibliotecas como *Spring Security*, *MySQL Driver*, *Lombok*, *Java JSON Web Token (JWT)*, *Spring Web* e Spring Data *JPA*, sendo o gerenciamento destas feito com o auxílio do *Apache Maven*.

O uso das bibliotecas pode ser analisado no código da classe “User”, apresentado na Figura 3. É possível observar as anotações provenientes das bibliotecas *Lombok* e *Spring Data JPA*.

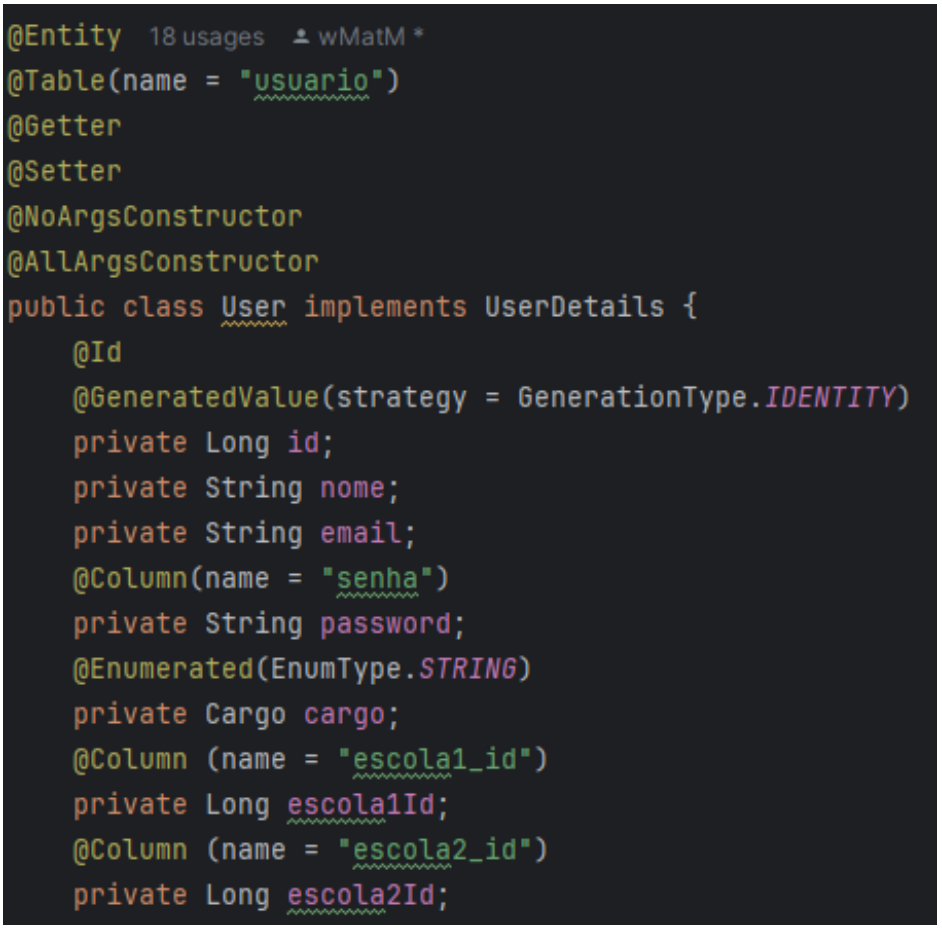

Figura 3: Código da classe "User". Fonte: Autor, 2025.

As anotações "@Getter" e "@Setter" substituem os métodos *get* e *set* que seriam criados para os atributos da classe. As anotações "@NoArgsConstructor" e "@AllArgsConstructor" geram automaticamente métodos construtores da classe sem argumentos e com todos os argumentos, respectivamente. Essas quatro anotações pertencem ao *Lombok*, tendo como finalidade reduzir o código repetitivo e verboso em classes *Java*.

As demais anotações pertencem ao *Spring Data JPA*. A "@Entity" marca a classe "User" como uma entidade JPA. Isso informa ao framework que ele deve gerenciar essa classe e que ela corresponde a uma tabela no banco de dados. A "@Table (name = "usuario") especifica o nome exato da tabela no banco de dados. Caso o nome da classe fosse igual ao nome da tabela no banco de dados, não seria necessária essa anotação.

A anotação "@Id" indica que o campo é a chave primária da tabela enquanto a anotação "@GeneratedValue" configura a estratégia de geração dessa chave. Neste caso é utilizado a estratégia "IDENTITY", indicando que a geração da chave primária será realizada pelo banco de dados.

As anotações "@Column" mapeiam o atributo abaixo delas para uma coluna no banco de dados. É necessário seu uso quando o nome do atributo e da coluna não são iguais. Note que os atributos "id", "nome" e "email" não possuem essa anotação

justamente por terem uma coluna correspondente na tabela, sendo assim, o framework consegue mapear. A anotação "@Enumerated" define como o *Enum* será representado no banco de dados. Nesse caso, como o cargo é uma cadeia de caracteres, será do tipo *String*.

De acordo com a Figura 2, a classe "usuario" deveria possuir um atributo para indicar o momento de sua criação, porém, como este dado não é utilizado pelo sistema atualmente, seu gerenciamento é feito diretamente no banco de dados.

Para facilitar a gestão dos arquivos, foi realizada a divisão em pacotes. O *backend*, desenvolvido em *Java*, foi separado em cinco grupos principais: *controllers*, *entities*, *repositories*, *services* e security. Essa separação está apresentada na Figura 4.

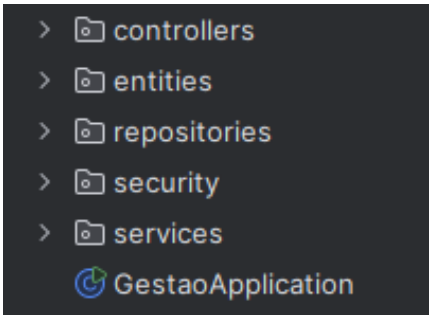

Figura 4: Separação do sistema em pacotes. Fonte: Autor, 2025.

O pacote *controllers* possui as classes que são responsáveis por implementar as APIs. Essas classes implementam controladores que seguem a arquitetura REST e são responsáveis por realizarem a comunicação com o restante do sistema.

O pacote *entities* possui as classes que representam as entidades do sistema, como estudante, escola e turma. Essas entidades são mapeadas por meio de anotações para a tabela no banco de dados.

O pacote *repositories* possui as interfaces que encapsulam a lógica de acesso aos dados. Dessa forma, as classes de serviço não interagem diretamente com a fonte dos dados.

O pacote services possui as classes que implementam a lógica da aplicação. São responsáveis por executar as operações necessárias para que se cumpra as regras de negócio e as solicitações dos usuários.

O pacote *security* é referente a criptografia, autenticação e validação dos usuários. Esse processo visa permitir a transferência de informações confidenciais de maneira segura entre um cliente e um servidor.

Além das classes e interfaces implementadas nos pacotes supracitados, a classe *GestaoApplication* está localizada na raiz do projeto e realiza a configuração automática, reconhece os componentes e inicia a aplicação *Spring Boot.*

Toda essa arquitetura em *Java* detalhada anteriormente constitui o servidor do sistema, operando de forma independente da interface do usuário. Os *controllers* atuam como a porta de entrada para o servidor, expondo *endpoints* de API que permitem a comunicação segura e padronizada.

A outra metade dessa arquitetura cliente-servidor é a interface interativa construída com a biblioteca *React*. O cliente consome os recursos disponibilizados pelo servidor por meio de chamadas de API. A interface é estruturada de acordo com o cargo do usuário, sendo organizada pelo componente principal *App.js*.

A exibição personalizada é feita pelos componentes *AdminDashboard.js, SecretaryDashboard.js e TeacherDashboard.js*. Dessa forma, é possível garantir que as funcionalidades apresentadas sejam pertinentes a cada usuário de acordo com o seu cargo, evitando erros durante o uso.

5. Resultados

Esta seção apresenta os resultados obtidos com o desenvolvimento do sistema *web* para gerenciamento escolar. O intuito é demonstrar, por meio de suas principais interfaces, o alcance dos objetivos propostos de criar uma ferramenta tecnológica e eficiente para a gestão escolar.

5.1 Controle de Acesso

O acesso ao sistema é controlado por um mecanismo de autenticação baseado em email e senha, conforme apresentado na Figura 5. Desse modo, é possível garantir que apenas usuários cadastrados e autorizados possam acessar as funcionalidades e informações do sistema, de acordo com seus níveis de permissão.

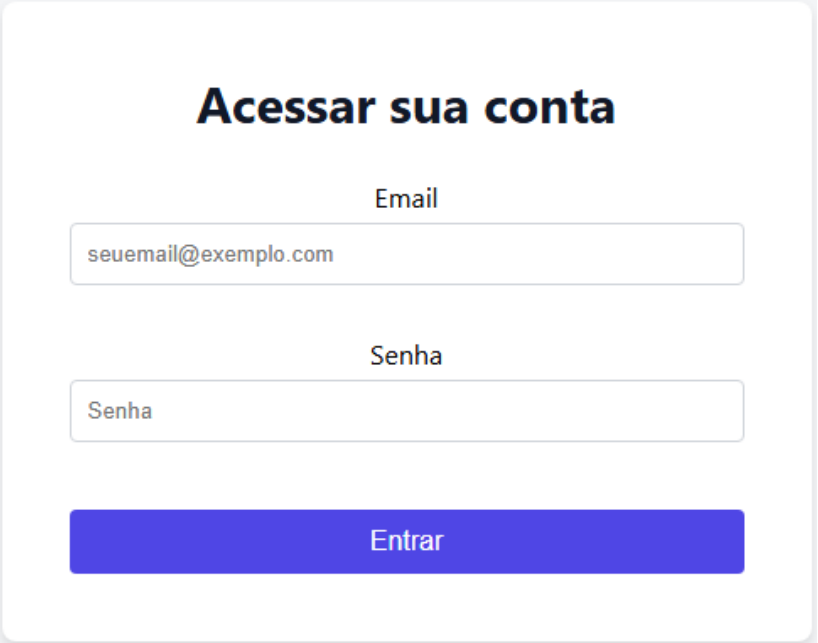

Figura 5: Tela de verificação de credenciais para acesso. Fonte: Autor, 2025.

Após a validação das credenciais, o servidor gera um *JSON Web Token*, envia ao cliente e utiliza para autenticar as requisições subsequentes. Essa tecnologia viabiliza um mecanismo de autenticação *stateless*, permitindo que o servidor verifique a identidade e as permissões do usuário em cada requisição sem a necessidade de uma sessão ativa.

Embora as interfaces personalizadas não apresentem botões para realizar

requisições que o cargo do usuário não permita, com o *Spring Security* é possível a criação de filtros, garantindo que os *endpoints* da API sejam protegidos por uma camada de autorização no lado do servidor. Essa configuração define que o sistema deve primeiro validar o *token* de autenticação para extrair o cargo do requerente para em seguida verificar se o cargo tem permissão para acessar o *endpoint*. Por exemplo, requisições para alterar a nota de um estudante somente são permitidas com o cargo de professor. Para *endpoints* que não exigem um cargo específico, o sistema apenas verifica a validade do *token* para garantir que o usuário está autenticado.

5.2 Interfaces do Usuário Administrador

O *token* gerado no momento do *login* contém, além das informações de validação, dados como nome e cargo do usuário. Esses dados são essenciais para a geração de páginas personalizadas. De acordo com o diagrama de casos de uso, apresentado na Figura 1, o usuário administrador pode realizar todas as funções exceto a atribuição de nota e frequência aos estudantes. A sua página inicial, apresentada na Figura 6, é a que possui o maior número de funcionalidades.

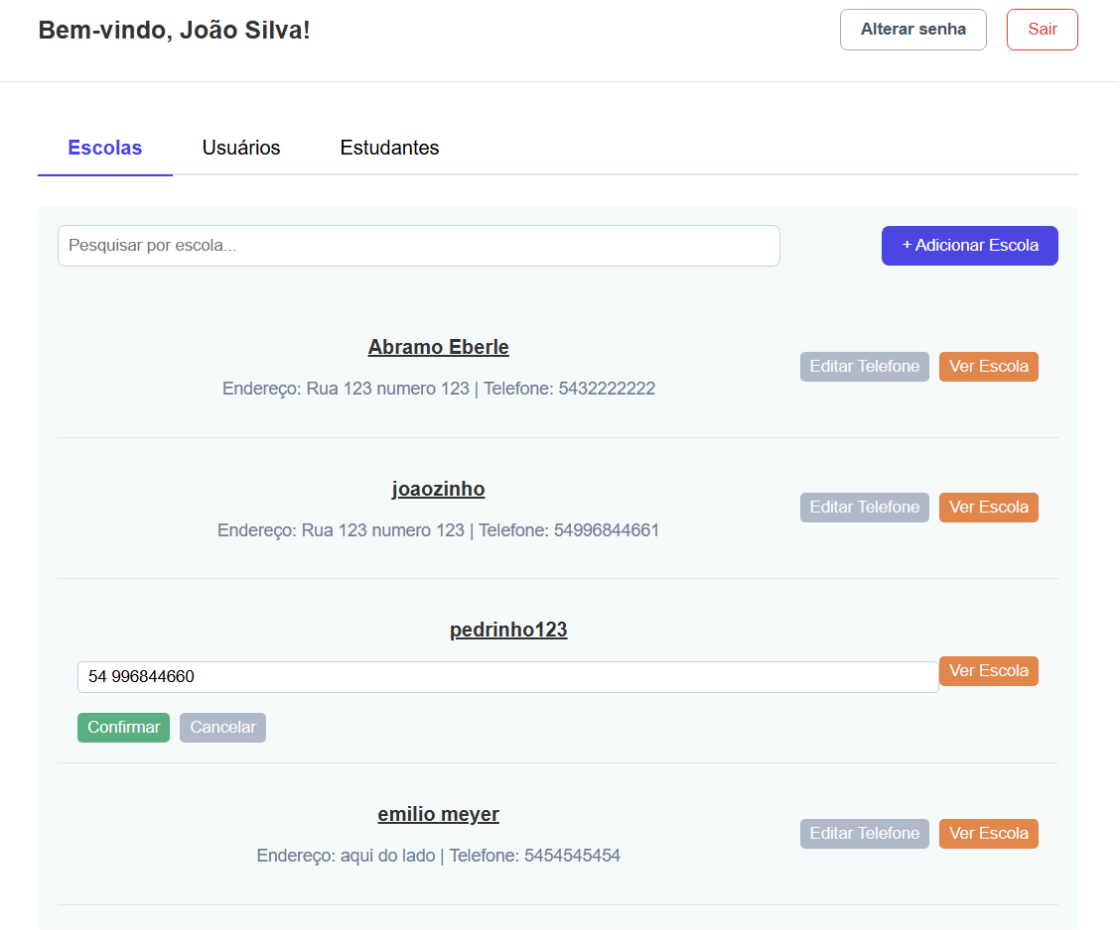

Figura 6: Página inicial do usuário com cargo administrador. Fonte: Autor, 2025.

O cabeçalho da página possui uma mensagem de bem-vindo, seguida do nome do usuário. Ainda, no lado direito do cabeçalho, são apresentados dois botões: um para alterar a senha e outro para sair do sistema, realizando o *logout*.

Na parte central da página, a navegação é organizada em três abas: “Escolas”, “Usuários” e “Estudantes”. O menu “Escolas” apresenta uma barra de pesquisa dinâmica que retorna às escolas que possuem a letra ou a combinação de letras inseridas. Ao lado da barra de pesquisa há um botão “+Adicionar Escola” que permite adicionar uma nova escola ao sistema.

De acordo com as informações na barra de pesquisa, são listadas as escolas correspondentes, apresentando o nome da escola bem como o endereço e o número de telefone. Cada escola listada oferece opções para editar o número de telefone ou visualizar dados detalhados.

O menu “Usuários”, de maneira semelhante ao de “Escolas”, apresenta uma barra de pesquisa dinâmica que retorna os usuários cujo nome contém a sequência de caracteres inserida, conforme Figura 7. Ao lado da barra de pesquisa há um botão “+Adicionar Usuário”, que permite cadastrar um novo usuário ao sistema.

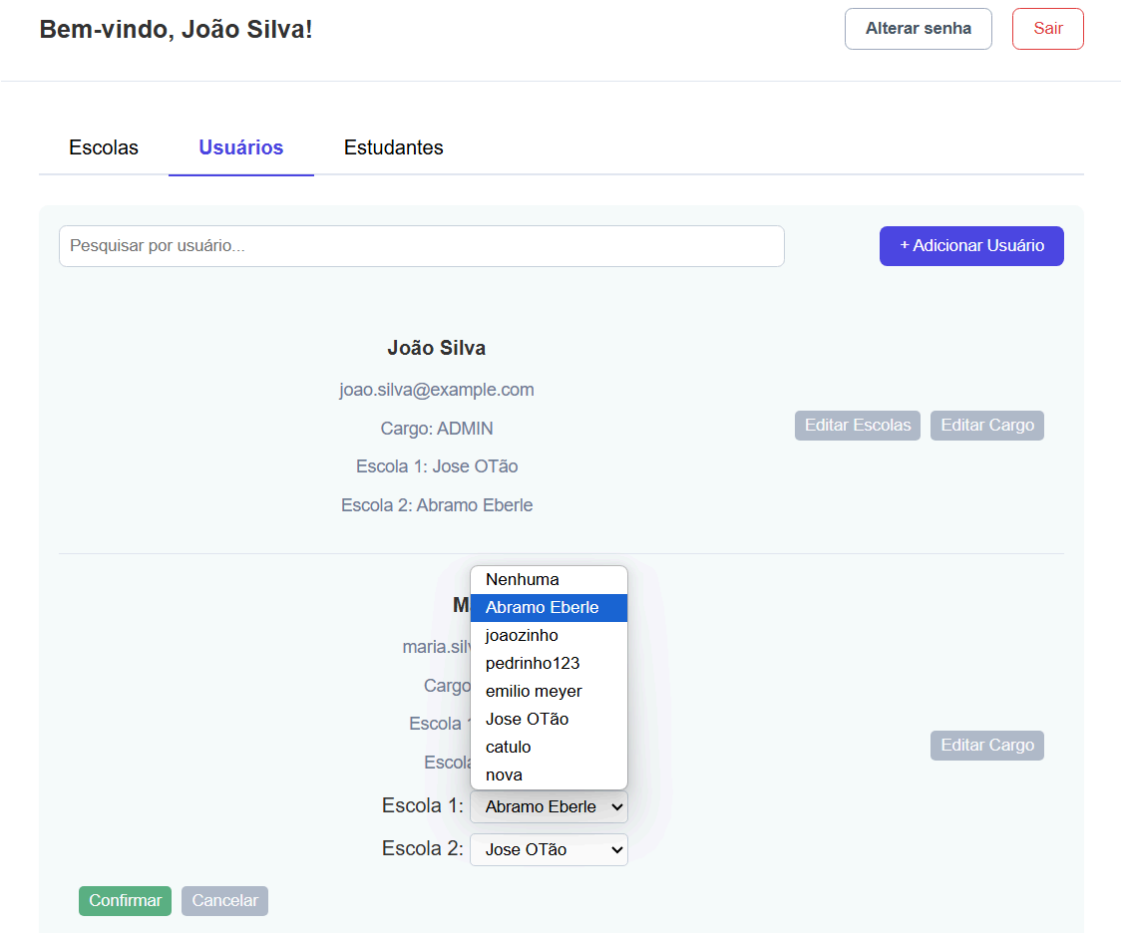

Figura 7: Menu “Usuários” de um usuário com cargo administrador. Fonte: Autor, 2025.

A Figura 7 apresenta os caracteres “si” inseridos na barra de pesquisa, retornando os usuários que possuem essa combinação em seu nome. Neste menu é possível editar as escolas em que o usuário trabalha. Ao pressionar o botão de edição de escolas, o sistema exibe uma lista de seleção com as instituições cadastradas, um mecanismo que previne erros de digitação e garante a consistência dos dados. Além

disso, é possível editar o cargo dos usuários, abrindo uma lista com as opções de cargos possíveis (administrador, secretário ou professor) para facilitar esta alteração e evitar falhas.

O menu “Estudantes” possui uma barra de pesquisa e um botão para adicionar estudantes em sua parte superior, conforme apresentado na Figura 8.

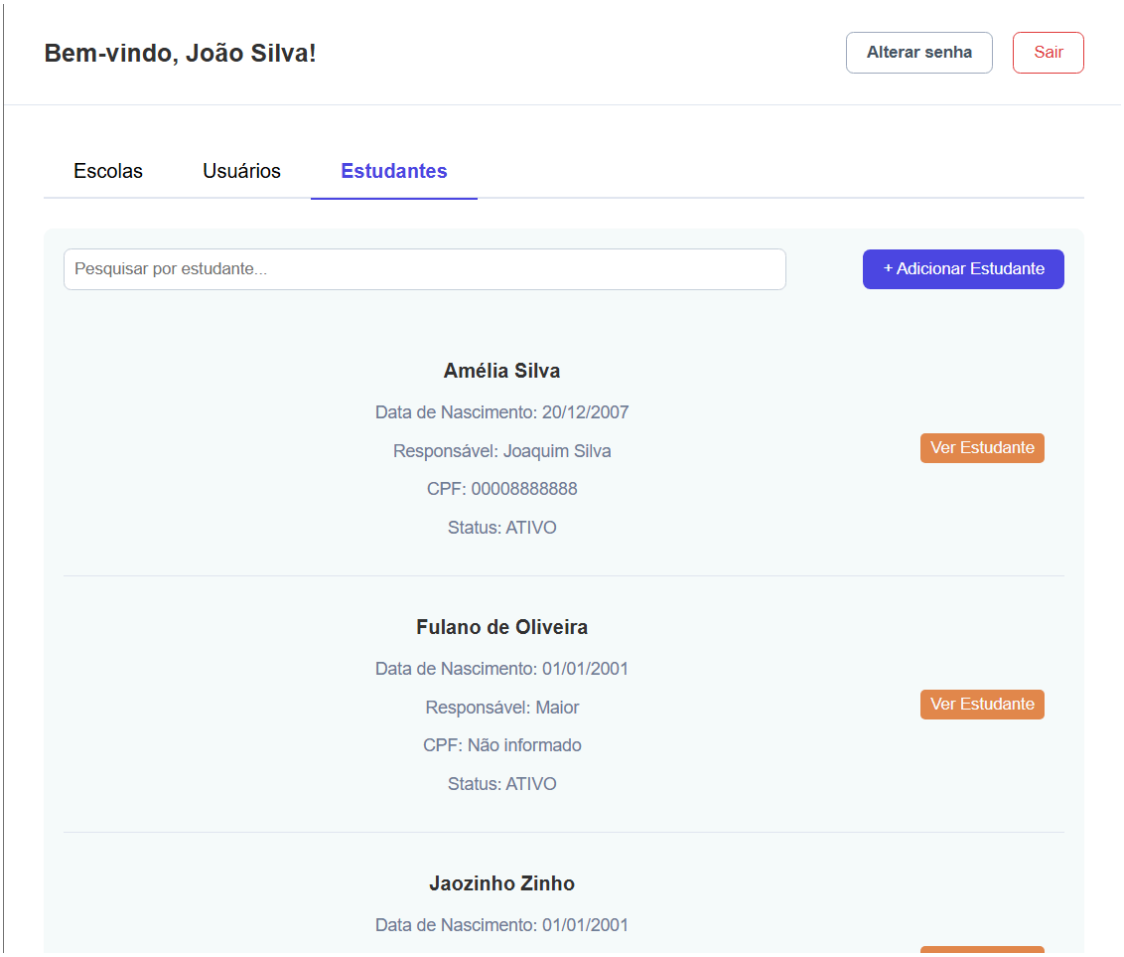

Figura 8: Menu “Estudantes” de um usuário com cargo administrador. Fonte: Autor, 2025.

A parte central do menu “Estudantes” exibe a lista de todos os estudantes cadastrados no sistema, apresentando informações importantes como nome do estudante, data de nascimento, responsável, CPF e status do estudante.

Para cada estudante, é apresentado um botão “Ver Estudante”, que redireciona o usuário para uma página com o perfil completo do aluno. A Figura 9 apresenta esta tela, que é dividida em um cabeçalho e três seções principais.

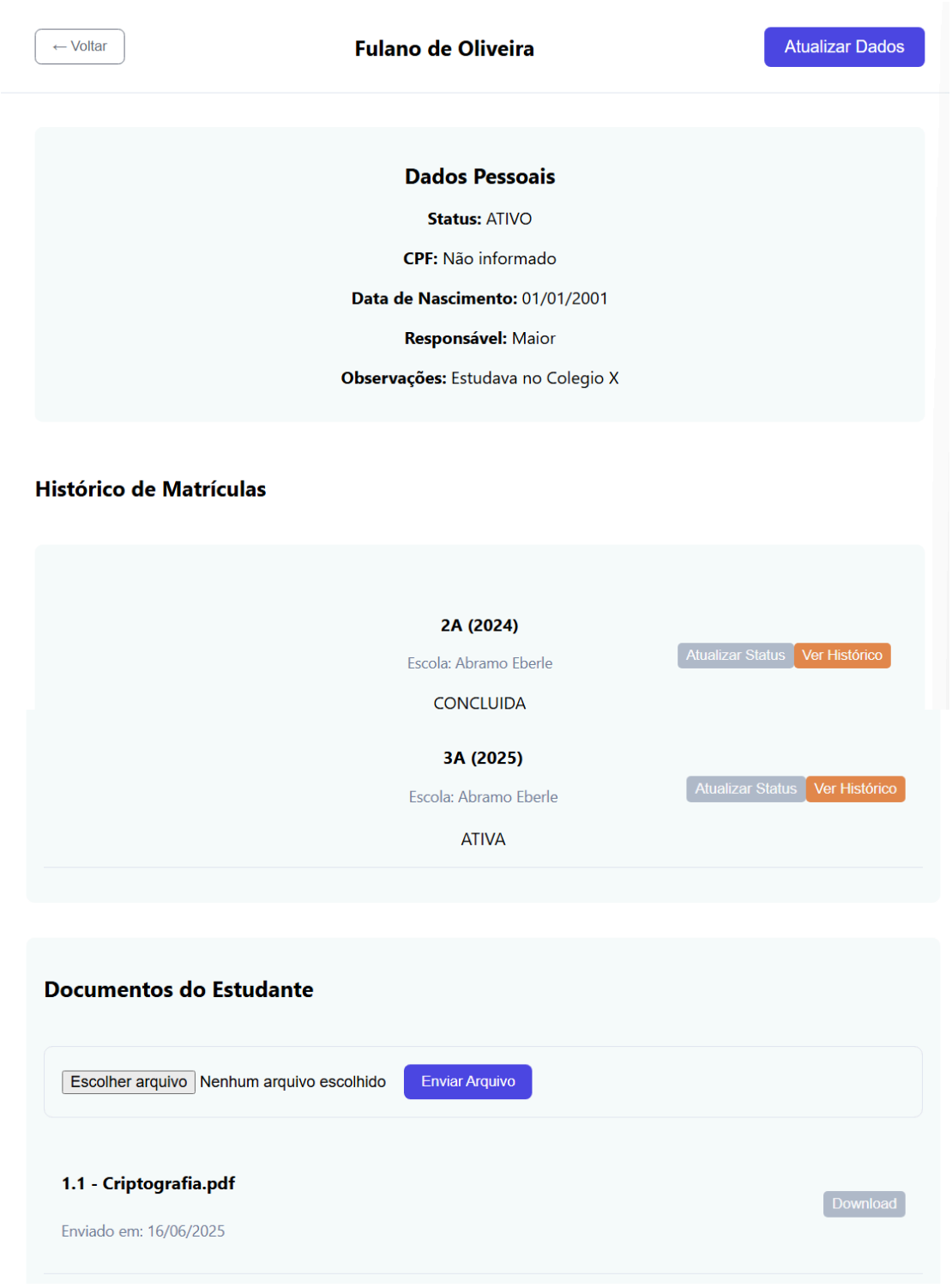

Figura 9: Página com perfil completo do estudante. Fonte: Autor, 2025.

O cabeçalho da página apresenta os elementos de navegação, contendo um botão para voltar ao menu inicial “Estudantes”, apresentado na Figura 8, o nome completo do discente e um botão para atualizar os dados do estudante. Por meio dessa atualização é possível inserir um cpf para os alunos que não informaram no momento da matrícula, alterar o nome do responsável ou o status do estudante. Para alterar o status de “ATIVO” para “TRANSFERIDO” ou “CONCLUIDO”, o estudante não pode ter nenhuma

matrícula com status “ATIVA”.

A primeira seção abaixo do cabeçalho apresenta os dados pessoais do estudante. Em seguida, na segunda seção, é listado o histórico do discente. Nesta parte são apresentadas as turmas nas quais o estudante esteve ou está matriculado, informando a escola, o status dessa matrícula e botões de ação para, respectivamente, atualizar este status ou visualizar o histórico detalhado do aluno na turma.

É importante ressaltar que o sistema impõe uma regra para a atualização de status, visando garantir a consistência do histórico acadêmico. Enquanto o estudante possuir uma matrícula “ATIVA”, só poderá ser aplicada alguma alteração a esta matrícula vigente. Apenas na ausência de uma matrícula ativa será permitido a edição do status de registros anteriores.

Na terceira parte desta página está a área “Documentos do Estudante”, referente a gestão de arquivos. Composta por um campo de seleção de arquivo e um botão para enviar. Logo abaixo, são listados os documentos vinculados ao aluno com a respectiva data de envio e um botão para realizar o *download* de cada um.

Ao selecionar a opção “Ver Histórico” em uma das matrículas listadas na Figura 9, o usuário é redirecionado para a página com as informações detalhadas sobre o desempenho do estudante naquela turma, conforme apresentado na Figura 10.

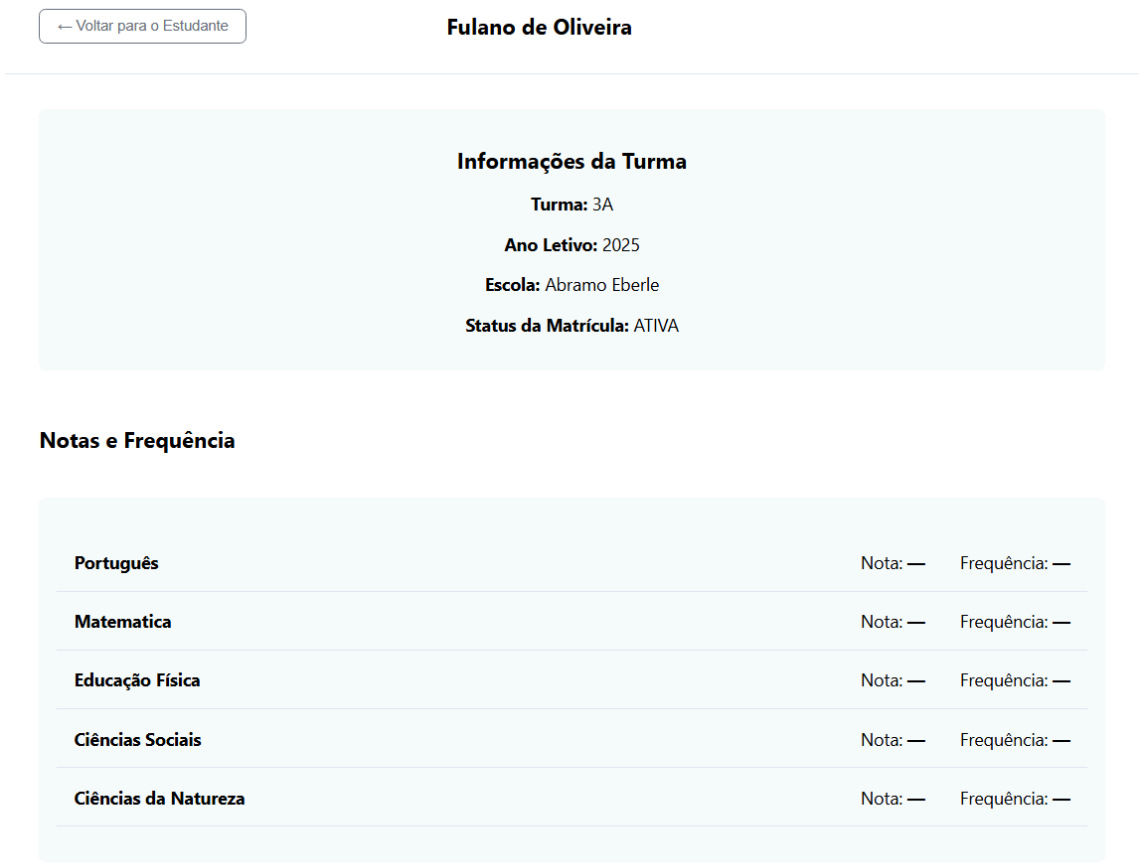

Figura 10: Página detalhada da matrícula do estudante. Fonte: Autor, 2025.

A página é dividida em duas seções. A primeira, “Informações da Turma”, exibe dados para contextualizar a matrícula, apresentando informações essenciais como identificação da turma, ano letivo, escola e o status atual da matrícula.

A segunda seção, intitulada “Notas e Frequência”, detalha o rendimento acadêmico do estudante. Nesta parte são listadas as disciplinas associadas à turma e para cada campo a respectiva nota e frequência. É importante ressaltar que o usuário administrador não pode alterar a nota e frequência do estudante, de acordo com os requisitos funcionais do sistema.

Retomando a tela de gerenciamento de escolas, apresentada na Figura 6, para cada instituição listada há um botão “Ver Escola”. Este botão serve como um ponto de acesso para uma visualização detalhada, redirecionando o administrador para a página de gerenciamento da escola, conforme apresentado na Figura 11.

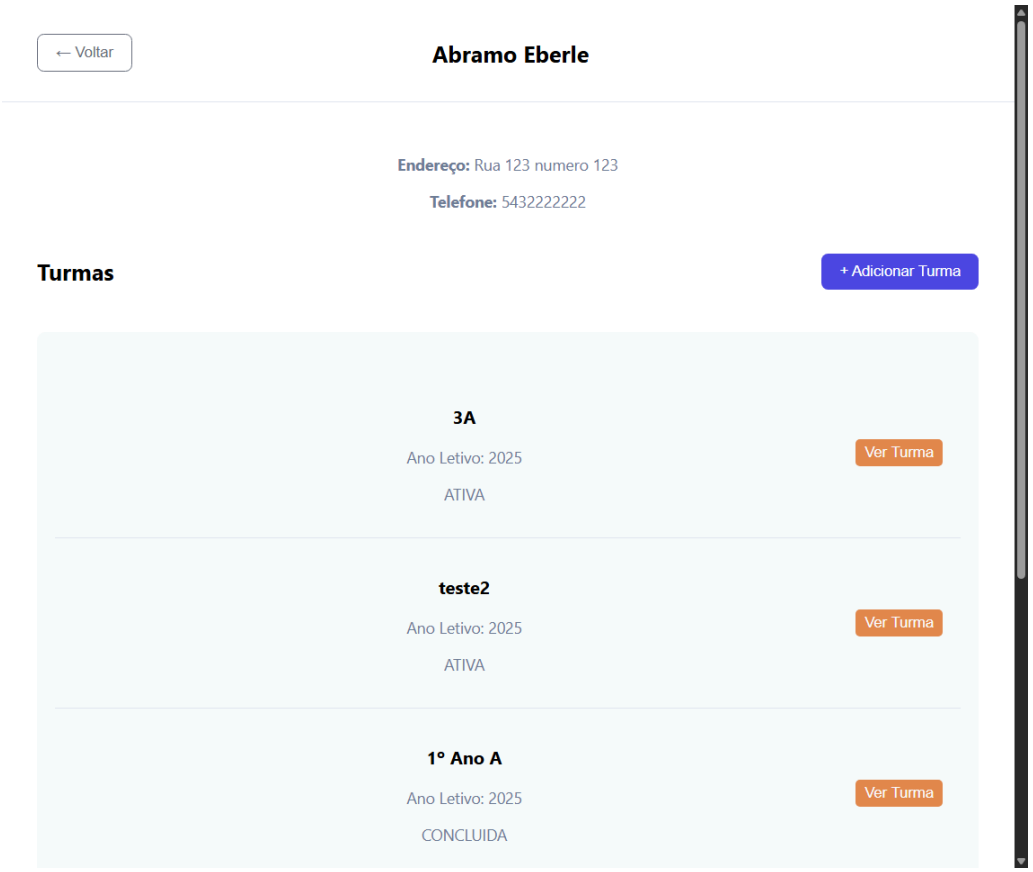

Figura 11: Página de gerenciamento da escola. Fonte: Autor, 2025.

No topo da página são exibidos os dados da instituição como nome, endereço e telefone, juntamente com um botão “Voltar” que permite retornar ao menu de gerenciamento de escolas. Logo abaixo, o botão “+Adicionar Turma” possibilita a inserção de uma nova turma na escola selecionada.

Em seguida é apresentada a listagem de todas as turmas cadastradas, com nome, ano letivo e status. Para facilitar a gestão, a exibição é organizada para mostrar primeiramente as turmas com status “ATIVA”. Cada item listado dispõe do botão “Ver Turma”, que leva o administrador a uma nova tela para consultar os detalhes e gerenciar

a turma, conforme apresentado na Figura 12.

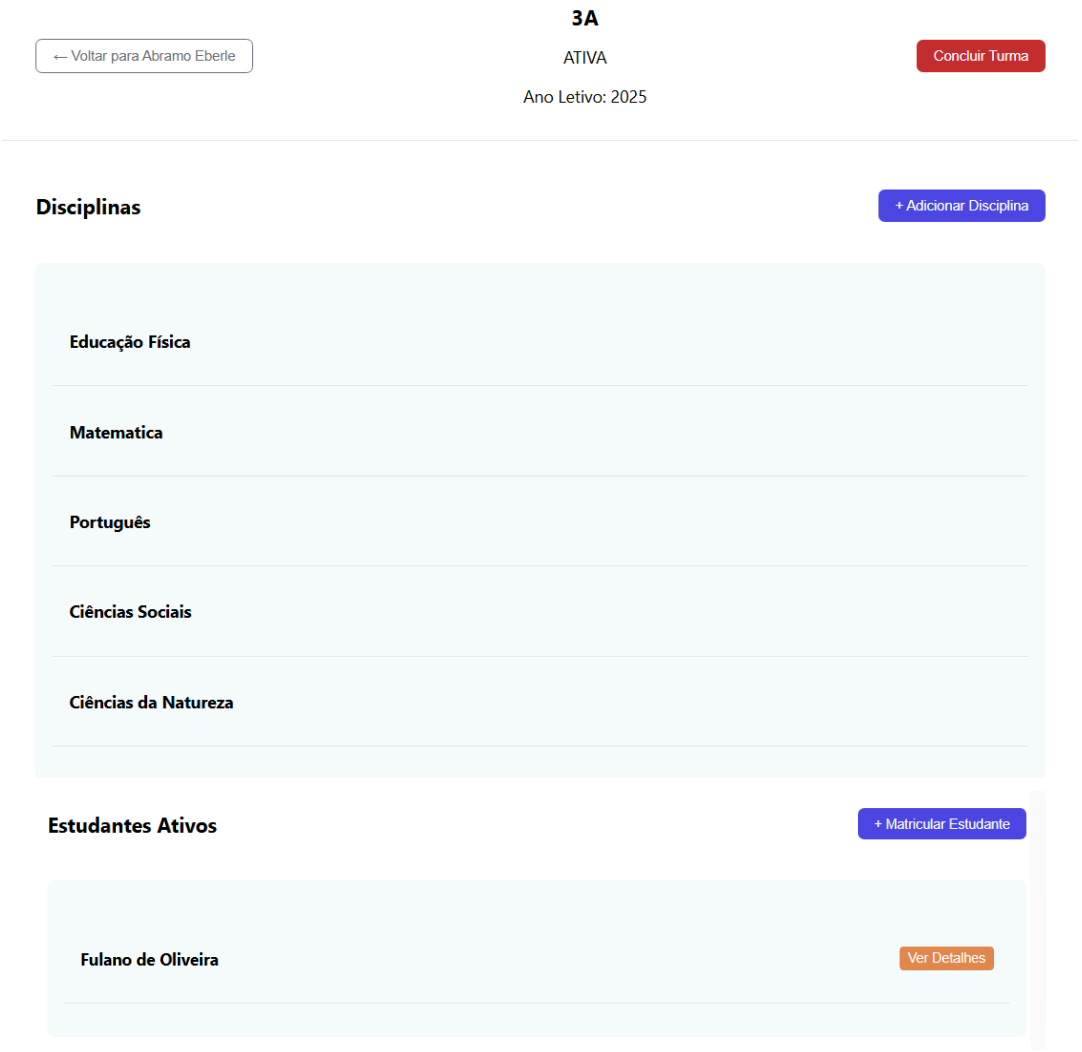

Figura 12: Página de gerenciamento da turma. Fonte: Autor, 2025.

O cabeçalho identifica a turma, o status e seu ano letivo. Além dessas informações, dois botões de ação são disponibilizados: um para voltar a página da escola que a turma está inserida e outro para "Concluir Turma", alterando o status de "ATIVA" para "CONCLUIDA". Essa ação, de acordo com as regras de negócio, somente é possível caso não existam estudantes com matrícula ativa na turma.

Abaixo, a tela é dividida em duas seções de gerenciamento. A primeira, "Disciplinas", lista os componentes curriculares daquela turma e oferece um botão para adicionar novos. Para evitar erros, o botão abre um modal com um menu listando as disciplinas cadastradas que não integram a matriz curricular da turma. A segunda seção, "Estudantes Ativos", é focada na gestão das matrículas dos alunos. A partir dela é possível matricular novos discentes por meio do botão "+Matricular Estudante" e visualizar a lista dos que pertencem à classe. Cada aluno na lista possui um botão "Ver detalhes", que redireciona o usuário ao perfil completo do estudante, apresentado na Figura 9.

O botão para matricular estudantes na turma abre um modal, apresentado na Figura 13, que permite ao usuário cadastrar um novo estudante ou pesquisar um estudante no sistema que está com status ativo e nenhuma matrícula ativa.

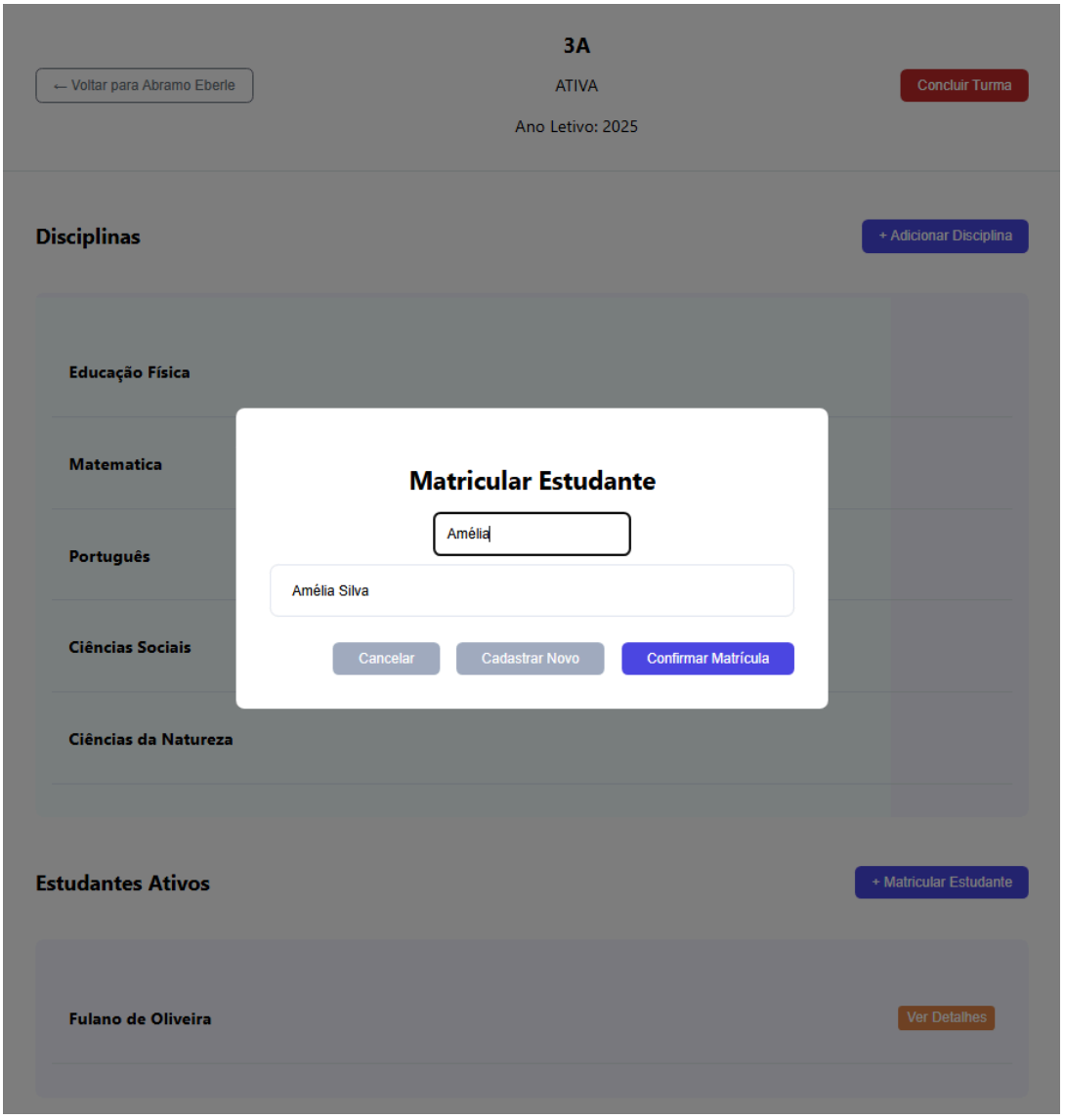

Figura 13: Matricular estudante a partir da página da turma. Fonte: Autor, 2025.

Conforme apresentado na Figura 13, o usuário pesquisou “Amélia” e o sistema retornou os resultados correspondentes, nesse caso, uma lista com uma única estudante, “Amélia Silva”. Para realizar o cadastro nessa turma é necessário pressionar o botão com o nome do estudante e confirmar a matrícula. O botão cancelar fecha o modal de “Matricular Estudante”, enquanto o botão “Cadastrar Novo” adapta a interface do modal para o formulário de cadastro de novo aluno, apresentando campos para inserir nome completo, data de nascimento, CPF, nome do responsável e observações.

5.3 Interfaces do Usuário Secretário

O usuário com cargo de secretário detém as condições necessárias para a maioria das operações diárias. Suas permissões são semelhantes a de um administrador, com

exceções em relação à adição e gerenciamento de escolas e usuários. A Figura 14 demonstra a página inicial de um usuário com cargo de secretário, sendo a única diferença em relação ao administrador.

Figura 14: Página inicial do usuário com cargo de secretário. Fonte: Autor, 2025.

É possível visualizar as diferenças entre a página inicial do secretário, apresentada na Figura 14, e do administrador, conforme Figura 6. Na aba "Escolas", embora o secretário possa pesquisar e visualizar as instituições, as ações de modificação foram suprimidas. O botão "+Adicionar Escola" e "Editar Telefone" não estão presentes. Além disso, a aba "Usuários" é removida completamente, impedindo que o secretário crie novas contas ou altere cargos.

Essas distinções na interface exemplificam a personalização do sistema, que se adapta de acordo com as funções que cada usuário deve exercer. Ao renderizar componentes e funcionalidades com base no nível de permissão associado ao perfil, o sistema otimiza a experiência do usuário, oferecendo uma ferramenta focada e alinhada às responsabilidades específicas.

5.4 Interfaces do Usuário Professor

O perfil de professor possui o escopo de permissões mais restrito do sistema. Sua função é estritamente acadêmica, não sendo permitido o gerenciamento de matrículas de estudantes ou modificações em dados de turmas e escolas. A interface da página inicial

do professor, apresentada na Figura 15, é projetada para ser um portal de acesso direto, removendo as funcionalidades administrativas que não são pertinentes ao seu cargo.

Figura 15: Página inicial do usuário com cargo de professor. Fonte: Autor, 2025.

A tela inicial não apresenta as abas “Usuários” e “Estudantes” como no cargo de administrador. A página contém apenas a aba “Escolas” que, por sua vez, é filtrada para exibir somente as instituições que o professor está vinculado. Semelhante ao perfil de secretário, não há botões para adicionar ou editar os dados das escolas, sendo a única ação possível a de visualizar mais detalhes sobre a escola.

A interface de gerenciamento de escolas, apresentada na Figura 11, é adaptada aos usuários com cargo de professor. A funcionalidade de adicionar turma não é exibida pois esta não é uma atribuição do docente. Desse modo, a tela serve como um portal de navegação, onde são listadas as turmas vinculadas à escola, com a única finalidade de permitir a consulta de seus detalhes.

Ao selecionar uma turma, o professor avança para a tela de detalhes, apresentada na Figura 12, que para este cargo funciona como um painel de consultas. Os botões de gerenciamento - “Concluir Turma”, “+Adicionar Disciplina” e “+Matricular Estudante”- são removidos da interface. Essa adaptação reforça a personalização do sistema de acordo com a função de cada usuário.

O propósito da tela de detalhes da turma para o professor é visualizar a composição da turma, tanto em termos de disciplina quanto de estudantes. A partir da lista de estudantes ativos, o professor pode visualizar detalhes sobre um aluno por meio do botão “Ver Detalhes”.

Ao acionar o botão para visualizar mais detalhes do estudante, o professor é redirecionado ao perfil completo do estudante, acessando uma versão da tela apresentada na Figura 9, porém adaptada para seu cargo. Nesta visualização, as funcionalidades de gerenciamento administrativo são omitidas: os botões para atualizar

os dados do aluno e os status das matrículas não são exibidos. O propósito dessa interface para o docente é permitir a visualização de detalhes sobre as matrículas para realizar a alteração das notas e frequência nas que estão ativas. Além disso, a gestão documental também é possível, permitindo tanto o *download* de arquivos existentes como a inserção de novos.

A partir do perfil do estudante, o professor seleciona a opção "Ver Histórico" em uma das matrículas. O sistema realiza uma validação para averiguar se a matrícula está ativa. Caso não esteja, será apresentada uma tela semelhante a da Figura 10, sem a opção de alterar as notas e frequências. Caso contrário, o usuário será redirecionado para a interface de lançamento de notas e frequências, apresentada na Figura 16. Esta tela exibe as informações da turma para contextualização e a lista de disciplinas para avaliação.

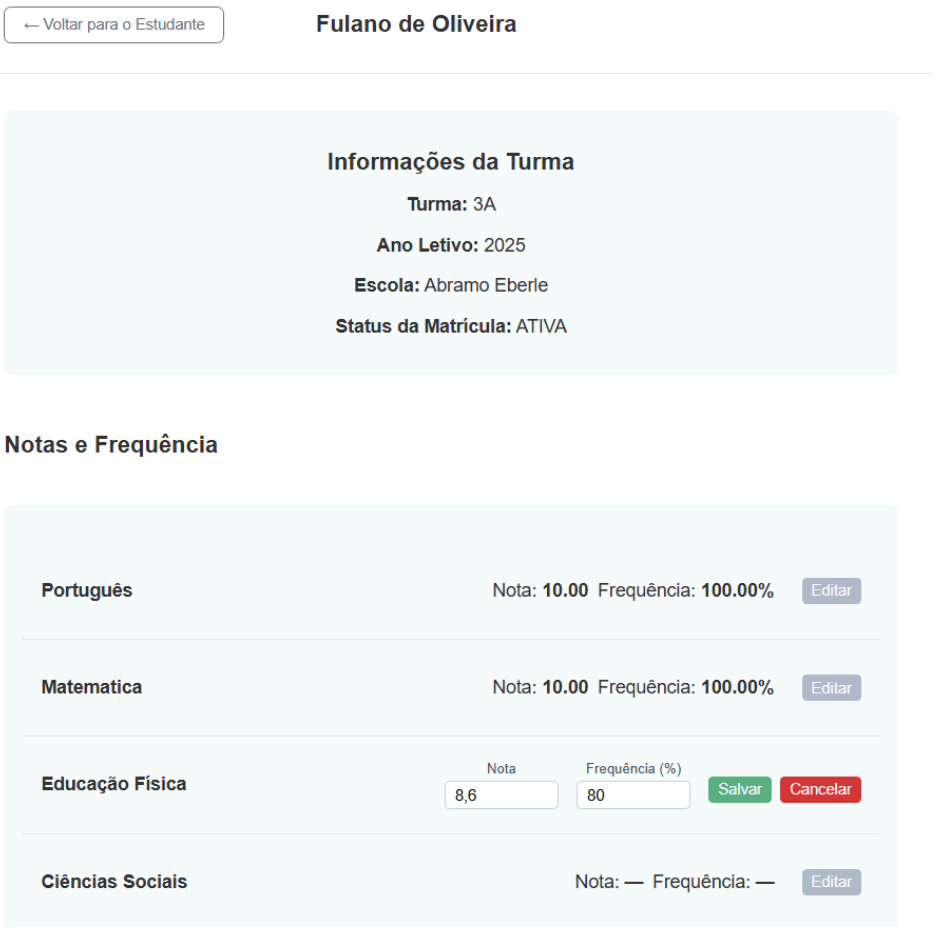

Figura 16: Página de edição de notas e frequências para professor. Fonte: Autor, 2025.

Na seção "Notas e Frequência", cada disciplina possui um botão "Editar". Ao pressioná-lo, o software habilita os campos para a entrada da nota e da frequência do estudante. Para confirmar a operação é necessário utilizar o botão "Salvar", ou pode descartar as alterações com o botão "Cancelar". Dessa forma é possível garantir que a inserção de dados seja um processo controlado, intuitivo e restrito apenas às matrículas ativas.

6. Considerações Finais

O presente trabalho documentou o processo de planejamento, desenvolvimento e implementação de um software *web* para o gerenciamento de dados e documentos de estudantes. A construção do sistema adotou uma arquitetura cliente-servidor, com o uso de API RESTful e interfaces intuitivas e personalizadas de acordo com as responsabilidades dos cargos dos usuários.

Considerando que o software resultante centraliza as informações, integra a gestão escolar e padroniza diversos processos, solucionando problemas como o gerenciamento manual dos dados, o extravio e a degradação de documentos, é possível considerar que os objetivos propostos foram alcançados. O sistema atende diretamente a demanda por uma ferramenta tecnológica, moderna e eficiente, capaz de otimizar a rotina de profissionais da educação.

Adicionalmente, a arquitetura modular estabelecida serve como uma base para futuras expansões internas. Funcionalidades como a automação na geração de documentos estudantis e a implementação de um módulo para a comunicação direta entre os usuários poderiam ser agregadas para otimizar os processos e a colaboração entre os diferentes setores da escola.

Em uma evolução de maior impacto, o software poderia ser integrado com plataformas externas. Sistemas de gerenciamento de aprendizado como o *Moodle*, por exemplo, viabilizariam a sincronização de notas e materiais didáticos, enquanto a integração com serviços governamentais tornaria possível a automatização na submissão de relatórios obrigatórios, o que amplia o alcance e a relevância da ferramenta.

7. Referências

BARRETO, Raquel Goulart. Tecnologia e educação: trabalho e formação docente. In: Educação na Região Sudeste: conquistas históricas e retrocessos iminentes. Campinas, vol. 25, n. 89, p. 1181-1201, Set./Dez. 2004. Disponível em: https://www.scielo.br/j/es/a/6HmDSHGqC5VC3RSNtYWZmWS/?format=pdf&lang=pt. Acesso em: 03 dez. 2024.

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP). Censo Escolar da Educação Básica de 2023: Resumo Técnico. Brasília, 2023.

BRASIL. Lei Federal nº 8.159, de 08 de janeiro de 1991. Dispõe sobre a política nacional de arquivos públicos e privados e dá outras providências. Disponível em: https://www.planalto.gov.br/ccivil_03/leis/l8159.htm. Acesso em: 03 nov. 2024.

CALSEVERINI, Lineker Aleksander Guete; SILVA, Djalma Domingos. Schoolis: um projeto open source para informatizar o gerenciamento de escolas. 2024. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Informática para Negócios) – Faculdade de Tecnologia de São José do Rio Preto, São José do Rio Preto, 2024.

DIAGRAMS. Draw.io. Disponível em: https://app.diagrams.net/. Acesso em: 03 nov. 2024.

FERREIRA, Giselle Martins dos Santos; ROSADO, Luiz Alexandre da Silva;

CARVALHO, Jaciara de Sá. Educação e Tecnologia: questões críticas. Disponível em: https://osf.io/preprints/socarxiv/6hr5b. Acesso em: 03 nov. 2024.

FIGUEIREDO, Maria Joelma Kelly; NEGREIROS, Ana Cláudia Souza Vidal. Gestão do arquivo passivo escolar: um diagnóstico acerca da organização física de documentos em uma escola pública de Mossoró-RN. 2025.

HOLISTICS. Dbdiagram. Ferramenta para criação de diagramas de banco de dados. Disponível em: https://dbdiagram.io. Acesso em: 09 dez. 2024.

JETBRAINS. IntelliJ IDEA: The Leading Java and Kotlin IDE. Disponível em: https://www.jetbrains.com/idea/. Acesso em: 3 dez. 2024.

MENDES, Pedro Thomas Homem de Melo. Desenvolvimento de uma API RESTful para sistemas de automação residencial. 2024. Disponível em: https://repositorio.unesp.br/entities/publication/2c0d1fdf-fd9b-4cdb-8b43-b2c707b4748a. Acesso em 15 de junho de 2025.

MICROSOFT. Visual Studio Code. Disponível em: https://code.visualstudio.com/. Acesso em: 4 dez. 2024.

MOREIRA, Antonio Flávio Barbosa; KRAMER, Sonia. Contemporaneidade, educação e tecnologia. In: Educação na Região Sudeste: conquistas históricas e retrocessos iminentes. Campinas, vol. 28, n. 100 - Especial, p. 1037-1057, out. 2007. Disponível em: https://www.scielo.br/j/es/a/KS6FVdMKj4D9hzbGG9dfcps/?format=pdf&lang=pt. Acesso em: 03 dez. 2024.

ORACLE. MySQL. Disponível em: https://www.mysql.com/. Acesso em: 4 dez. 2024.

PANDOLFO, Eduardo Klein. Desenvolvimento de um web service rest para um protótipo de aplicativo no contexto pecuário. Disponível em: https://repositorio.unipampa.edu.br/handle/riu/7285. Acesso em: 04 dez. 2024.

PITA, Pedro Henrique Mendes. Guia Técnico para Apoio no Desenvolvimento de API RESTFul Seguras. Disponível em: https://repositorio.ipbeja.pt/server/api/core/bitstreams/0261cf94-5bfa-4e86-a068-82b50bec8c74/content. Acesso em 15 de junho de 2025.

PRESSMAN, Roger S. Software Engineering: A Practitioner's Approach. 5th ed. New York: McGraw-Hill, 2001.

REACT. React Documentation. Disponível em: https://react.dev. Acesso em: 3 dez. 2024.

SILVA, Iago Abner dos Santos. Estudo do impacto de ferramentas *backend* em sistemas de georreferenciamento com alta demanda. Disponível em: https://repositorio.ufrn.br/items/e82fee41-f9f3-4fcc-b15f-edf1eed7c14c. Acesso em: 04 dez. 2024.

SOMMERVILLE, Ian. Engenharia de Software. 9. ed. São Paulo: Editora Pearson Prentice Hall, 2011.

SPRING. Spring Framework Documentation. Disponível em:

https://spring.io/projects/spring-framework. Acesso em: 06 out. 2024.

VALENTE, Marco Tulio. Engenharia de Software Moderna: princípios e práticas para desenvolvimento de software com produtividade. Editora Independente, 2020.