PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS

ESCOLA POLITÉCNICA E DE ARTES

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

CAIO MARIANNI DE MORAIS

DESENVOLVIMENTO DO SISTEMA ESCOLAR SCHOOLINK

GOIÂNIA – GOIÁS

2026

CAIO MARIANNI DE MORAIS

DESENVOLVIMENTO DO SISTEMA ESCOLAR SCHOOLINK

Trabalho de conclusão de curso, modalidade projeto de desenvolvimento de software no curso de Ciências da Computação, da Pontifícia Universidade Católica de Goiás, como requisito parcial à obtenção do grau de bacharel em Ciências da Computação, sob a orientação do Prof. Me. Eugenio Júlio. Messala Cândido Carvalho.

GOIÂNIA – GOIÁS

2026

CAIO MARIANNI DE MORAIS

DESENVOLVIMENTO DO SISTEMA ESCOLAR SCHOOLINK

BANCA EXAMINADORA

_____

Orientador: Prof. Me. Eugenio Júlio Messala Cândido Carvalho

_____

Avaliador:

_____

Avaliador:

RESUMO

A gestão eficiente de dados acadêmicos e administrativos impõe desafios constantes às instituições de ensino, exigindo soluções que unam organização, agilidade e segurança. Assim sendo, este projeto apresenta as fases de desenvolvimento da plataforma SchoolLink, um sistema de gestão escolar integrado voltado ao ensino fundamental. O processo de estruturação contempla o levantamento de requisitos funcionais e não funcionais, a modelagem de dados e a definição de uma arquitetura escalável utilizando tecnologias modernas de desenvolvimento web. O diferencial do sistema reside na implementação rigorosa de controle de acesso baseado em papéis (RBAC) e na conformidade com a Lei Geral de Proteção de Dados (LGPD). A SchoolLink visa otimizar a administração escolar, sustentando-se na premissa de que a centralização da informação e a proteção de dados sensíveis são fundamentais para a eficácia educacional e para o fortalecimento da confiança entre a instituição, os professores e as famílias.

Palavras-chave: SchoolLink; gestão escolar; sistema de informação; segurança da informação; LGPD.

ABSTRACT

The efficient management of academic and administrative data poses constant challenges to educational institutions, requiring solutions that combine organization, agility, and security. Therefore, this project presents the development phases of the SchoolLink platform, an integrated school management system aimed at elementary education. The structuring process includes the gathering of functional and non-functional requirements, data modeling, and the definition of a scalable architecture using modern web development technologies. The system's differentiator lies in the rigorous implementation of Role-Based Access Control (RBAC) and compliance with the General Data Protection Law (LGPD). SchoolLink aims to optimize school administration, based on the premise that information centralization and sensitive data protection are fundamental for educational effectiveness and for strengthening trust between the institution, teachers, and families.

Keywords: SchoolLink; school management; information system; information security; LGPD.

SUMÁRIO

1

Sumário

LISTA DE SIGLAS

Sistema Schoolink - Sistema de Gerenciamento Escolar

2FA - Two-Factor Authentication Autenticação de dois fatores — mecanismo de segurança adicional no login

ABNT - Associação Brasileira de Normas Técnicas

ACID - Atomicity, Consistency, Isolation, Durability Atomicidade, Consistência, Isolamento e Durabilidade — propriedades de transações em banco de dados

API - Application Programming Interface Interface de programação de aplicações

CIA - Confidentiality, Integrity, Availability Tríade da segurança da informação: Confidencialidade, Integridade e Disponibilidade

CI/CD - Continuous Integration / Continuous Delivery Integração contínua e entrega contínua — práticas de automação de build e deploy de software

CLI - Command Line Interface Interface de linha de comando

CRUD - Create, Read, Update, Delete Criar, Ler, Atualizar e Deletar — operações básicas de persistência de dados

CSRF - Cross-Site Request Forgery Falsificação de requisição entre sites — tipo de ataque web

CSS - Cascading Style Sheets Folhas de estilo em cascata — linguagem de estilização de páginas web

CT - Caso de Teste Identificador dos cenários de teste documentados no capítulo de testes

DAC - Discretionary Access Control Controle de acesso discricionário

DELETE - Método HTTP de remoção de dados

DER - Diagrama de Entidades e Relacionamentos Representação gráfica da estrutura de um banco de dados

DOM - Document Object Model Modelo de objeto do documento — representação em árvore de uma página web

ECMA - European Computer Manufacturers Association Associação Europeia de Fabricantes de Computadores — organização que padroniza o JavaScript

GDPR - General Data Protection Regulation Regulamento europeu de proteção de dados pessoais

GET - Método HTTP de leitura de dados

HTML - HyperText Markup Language Linguagem de marcação de hipertexto — linguagem base das páginas web

HTTP - HyperText Transfer Protocol Protocolo de transferência de dados na web

HTTPS - HTTP Secure Versão segura e criptografada do protocolo HTTP

IETF - Internet Engineering Task Force Força-tarefa de engenharia da internet — organização que define padrões técnicos da internet

INEP - Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira Órgão federal responsável pelas avaliações e estatísticas da educação brasileira

ISO - International Organization for Standardization Organização Internacional de Normalização

JS - JavaScript Linguagem de programação interpretada amplamente utilizada na web

JSON - JavaScript Object Notation Notação de objetos JavaScript — formato leve de troca de dados entre sistemas

JWT - JSON Web Token Padrão de token para autenticação segura entre sistemas

LDB - Lei de Diretrizes e Bases da Educação Nacional Lei federal brasileira (Lei nº 9.394/1996) que regula a educação básica e superior

LGPD - Lei Geral de Proteção de Dados Pessoais Lei federal brasileira (Lei nº 13.709/2018) que regula o tratamento de dados pessoais

MAC - Mandatory Access Control Controle de acesso obrigatório

MEC - Ministério da Educação Órgão do governo federal responsável pela política educacional do Brasil

MVC - Model-View-Controller Modelo-Visão-Controlador — padrão de arquitetura de software

NBR - Norma Brasileira Documento de padronização técnica emitido pela ABNT

NIST - National Institute of Standards and Technology Instituto Nacional de Padrões e Tecnologia dos Estados Unidos

npm - Node Package Manager Gerenciador de pacotes do Node.js

ORM - Object-Relational Mapping Mapeamento objeto-relacional — técnica de mapeamento entre objetos de código e tabelas de banco de dados

OWASP - Open Web Application Security Project Projeto de código aberto focado em segurança de aplicações web

PATCH - Método HTTP de atualização parcial de dados

PDF - Portable Document Format Formato de documento portátil amplamente utilizado para distribuição de arquivos

POST - Método HTTP de envio e criação de dados

PUT - Método HTTP de substituição completa de dados

QR Code - Quick Response Code Código de resposta rápida — imagem bidimensional utilizada para leitura óptica de informações

RBAC - Role-Based Access Control Controle de acesso baseado em perfis de usuário

REST - Representational State Transfer Transferência de estado representacional — estilo arquitetural para desenvolvimento de APIs web

RFC - Request for Comments Pedido de comentários — documentos de padronização técnica da internet

RF - Requisito Funcional Descrição de uma funcionalidade que o sistema deve executar

RN - Regra de Negócio Premissa ou restrição proveniente do domínio do negócio

RNF - Requisito Não Funcional Restrição ou qualidade técnica que o sistema deve atender

RTK - Redux Toolkit Biblioteca oficial de ferramentas para simplificar o uso do Redux

SaaS - Software as a Service Modelo de distribuição de software em que o sistema é hospedado e oferecido como serviço pela internet, sem necessidade de instalação local

SGBD - Sistema de Gerenciamento de Banco de Dados

SIE - Sistema de Informação Educacional Categoria de sistema de informação voltado à gestão de instituições de ensino

SIG - Sistema de Informações Gerenciais Sistema que consolida dados operacionais em relatórios para apoio à gestão

SPA - Single-Page Application Aplicação de página única — aplicação web que carrega uma única página HTML e atualiza o conteúdo dinamicamente

SPT - Sistema de Processamento de Transações Sistema que registra as atividades cotidianas de uma organização

SQL - Structured Query Language Linguagem de consulta estruturada para banco de dados relacionais

SSG - Static Site Generation Geração de site estático — estratégia de renderização em que o HTML é gerado no momento da build

SSR - Server-Side Rendering Renderização no lado do servidor — estratégia em que o HTML é gerado pelo servidor a cada requisição

SSL - Secure Sockets Layer Protocolo criptográfico de segurança para comunicação na internet

SSD - Sistema de Suporte à Decisão Sistema que apoia decisões estratégicas por meio de análises avançadas

TCC - Trabalho de Conclusão de Curso Documento acadêmico final exigido para graduação

TOTP - Time-based One-Time Password Senha de uso único baseada em tempo — utilizada em autenticação de dois fatores

TS - TypeScript Superconjunto do JavaScript com tipagem estática

UI - User Interface Interface do usuário

URL - Uniform Resource Locator Localizador uniforme de recursos — endereço de um recurso na internet

UUID - Universally Unique Identifier Identificador universalmente único gerado automaticamente pelo sistema

W3C - World Wide Web Consortium Consórcio internacional que desenvolve padrões para a web

WCAG - Web Content Accessibility Guidelines Diretrizes de acessibilidade para conteúdo web

XSS - Cross-Site Scripting Injeção de scripts maliciosos em páginas web — tipo de ataque web

INTRODUÇÃO

CONTEXTUALIZAÇÃO

A educação brasileira é um dos setores que mais concentram desafios de gestão no país. Segundo o Censo Escolar de 2023, realizado pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP), o Brasil conta com mais de 179 mil escolas de educação básica, responsáveis pelo atendimento de aproximadamente 47 milhões de estudantes (BRASIL, 2023). Esse volume expressivo de instituições de ensino envolve uma cadeia complexa de atores — diretores, coordenadores, professores, alunos e responsáveis — que precisam se comunicar, compartilhar informações e tomar decisões de forma coordenada ao longo de todo o ano letivo.

Historicamente, grande parte dessas atividades de gestão escolar tem sido realizada de forma manual ou fragmentada: boletins impressos enviados ao final do bimestre, diários de classe em papel preenchidos à mão, planilhas eletrônicas sem integração entre si, e comunicações realizadas exclusivamente por agendas físicas ou grupos informais em aplicativos de mensagem. Esse cenário implica em retrabalho, perda de informação, dificuldade de acesso aos dados históricos e baixa transparência entre a escola e as famílias (LUCK, 2009).

A transformação digital, que já vinha avançando gradualmente em outros setores da economia, ganhou força também na educação a partir de 2020, quando a pandemia de COVID-19 forçou instituições de ensino de todo o mundo a adotar soluções tecnológicas de forma acelerada. No Brasil, a experiência revelou tanto o potencial das tecnologias digitais para a educação quanto as lacunas ainda existentes em infraestrutura e em ferramentas de gestão adaptadas à realidade das escolas (VALENTE; ALMEIDA, 2020).

Nesse contexto, os Sistemas de Informação Educacionais (SIE) surgem como instrumentos capazes de integrar os diferentes fluxos de informação da escola em uma única plataforma digital. Ao centralizar o gerenciamento de turmas, disciplinas, notas, frequência e comunicados, esses sistemas reduzem a carga administrativa sobre professores e gestores, ampliam a transparência para as famílias e contribuem para uma tomada de decisão mais embasada pelos diretores (LAUDON; LAUDON, 2021).

É nesse cenário que o sistema Schoolink foi concebido: como uma plataforma web de gerenciamento escolar capaz de atender, de forma integrada e segura, às necessidades específicas de

quatro perfis de usuário — o Diretor, o Professor, o Aluno e o Responsável — dentro de uma mesma instituição de ensino.

PROBLEMA DE PESQUISA

Apesar da disponibilidade crescente de tecnologias digitais, muitas escolas de pequeno e médio porte ainda carecem de sistemas integrados de gestão que contemplem, em um único ambiente, o controle acadêmico, a comunicação institucional e o acompanhamento do desempenho dos alunos. As soluções disponíveis no mercado são frequentemente caras, complexas demais para a realidade dessas instituições, ou oferecem funcionalidades genéricas que não se adaptam bem ao fluxo de trabalho de uma escola brasileira.

Diante disso, o presente trabalho busca responder à seguinte questão: Como desenvolver um sistema web de gerenciamento escolar que integre, de forma segura, acessível e eficiente, as necessidades de gestão acadêmica de diretores, professores, alunos e responsáveis em uma única plataforma digital?

JUSTIFICATIVA

A relevância deste trabalho reside na contribuição direta que um sistema de gerenciamento escolar bem desenvolvido pode oferecer à qualidade da gestão educacional.

Do ponto de vista do Diretor, a centralização das informações em uma única plataforma permite uma visão clara e atualizada da situação da instituição — quantos alunos estão matriculados, como estão os índices de frequência, quais turmas têm maior número de alunos em situação de risco de reprovação — o que fundamenta decisões pedagógicas e administrativas mais assertivas.

Do ponto de vista do Professor, a digitalização do diário de classe e do lançamento de notas reduz o tempo dedicado a tarefas burocráticas e minimiza erros de registro. A possibilidade de criar atividades avaliativas diretamente na plataforma, com acesso imediato pelos alunos, moderniza a comunicação entre professor e turma.

Do ponto de vista do Aluno, o acesso permanente ao próprio boletim, ao percentual de frequência e às próximas avaliações promove maior autonomia e responsabilidade no acompanhamento da própria trajetória acadêmica.

Do ponto de vista do Responsável, a transparência em tempo real sobre as notas e a presença dos filhos fortalece o vínculo entre a família e a escola, tornando o acompanhamento acadêmico mais efetivo e menos dependente de reuniões presenciais periódicas.

Além disso, do ponto de vista técnico-acadêmico, o desenvolvimento do Schoolink representa a aplicação prática de conceitos fundamentais da Engenharia de Software — como levantamento e especificação de requisitos, modelagem de dados, arquitetura em camadas, autenticação segura e controle de acesso baseado em perfis — em um contexto real e socialmente relevante, o que torna o projeto especialmente adequado como objeto de um Trabalho de Conclusão de Curso.

OBJETIVOS

Objetivo Geral

Desenvolver o Schoolink, um sistema web de gerenciamento escolar que integre, em uma única plataforma digital, as funcionalidades necessárias para a gestão acadêmica de instituições de ensino, atendendo de forma segura e eficiente aos perfis de Diretor, Professor, Aluno e Responsável.

Objetivos Específicos

a) Levantar, especificar e documentar: os requisitos funcionais, os requisitos não

funcionais e as regras de negócio do sistema. Com base nas necessidades reais dos diferentes perfis de usuário de uma instituição de ensino.

b) Modelar a estrutura de dados do sistema por meio de um diagrama de entidades e relacionamentos, garantindo a integridade e a consistência das informações acadêmicas armazenadas.

c) Implementar um módulo de autenticação seguro com controle de acesso baseado em perfis (RBAC), garantindo que cada usuário acesse apenas as funcionalidades e informações pertinentes ao seu papel na instituição.

d) Implementar os módulos de gestão de usuários, estrutura acadêmica, notas, frequência, atividades avaliativas, comunicados e configurações da instituição.

e) Desenvolver painéis de controle (dashboards) personalizados para cada perfil de usuário, apresentando indicadores e informações relevantes de forma clara e acessível.

f) Aplicar boas práticas de segurança da informação no desenvolvimento da plataforma, incluindo criptografia de senhas, proteção contra vulnerabilidades web e conformidade com os princípios da Lei Geral de Proteção de Dados Pessoais (LGPD).

g) Validar o funcionamento do sistema por meio de testes funcionais cobrindo os principais fluxos de uso de cada perfil de usuário.

METODOLOGIA

Este trabalho caracteriza-se como uma pesquisa aplicada, pois tem como finalidade a produção de um artefato tecnológico — o sistema Schoolink — voltado à solução de um problema prático identificado no contexto da gestão escolar (GIL, 2010).

Do ponto de vista dos procedimentos técnicos, o desenvolvimento seguiu as etapas propostas pela Engenharia de Software, conforme descrito por Sommerville (2019):

Fase 1 — Levantamento de Requisitos

Nessa fase foram identificadas e documentadas as necessidades dos diferentes perfis de usuário do sistema (Diretor, Professor, Aluno e Responsável), resultando nos documentos de Requisitos Funcionais, Requisitos Não Funcionais e Regras de Negócio apresentados no Capítulo 3 deste trabalho.

Fase 2 — Modelagem do Sistema

Com base nos requisitos levantados, foram elaborados o diagrama de entidades e relacionamentos do banco de dados e a definição da arquitetura da aplicação, organizando o sistema em camadas bem definidas (frontend, backend e banco de dados).

Fase 3 — Implementação

O sistema foi desenvolvido utilizando tecnologias modernas e amplamente adotadas pela indústria de software:

- Frontend: Next.js 15 com TypeScript, Tailwind CSS, Radix UI,

Redux Toolkit e React Hook Form com validação por Zod.

- Backend: Express.js 5 com TypeScript, Prisma ORM versão 6,

autenticação JWT com bcrypt e validação por Zod.

- Banco de dados: PostgreSQL, acessado por meio do ORM Prisma.
- Infraestrutura: Docker e Docker Compose para containerização

dos serviços.

Fase 4 — Testes e Validação

Foram realizados testes funcionais cobrindo os principais cenários de uso de cada perfil, validando o comportamento esperado das funcionalidades implementadas.

Fase 5 — Documentação

Todo o processo de desenvolvimento foi documentado, incluindo especificação de requisitos, modelagem de dados, rastreabilidade de requisitos e guia de capturas de tela do sistema em funcionamento.

ESTRUTURA DO TRABALHO

Este trabalho está organizado em nove capítulos, descritos a seguir:

Capítulo 1 — Introdução

Apresenta o contexto que motivou o desenvolvimento do Schoolink, o problema de pesquisa, a justificativa, os objetivos e a metodologia utilizada.

Capítulo 2 — Fundamentação Teórica

Aborda os conceitos acadêmicos que fundamentam o desenvolvimento do sistema, incluindo Sistemas de Informação na Educação, gestão escolar apoiada por tecnologia, controle de acesso baseado em perfis (RBAC), usabilidade, segurança da informação, LGPD e Engenharia de Requisitos de Software.

Capítulo 3 — Análise de Requisitos

Apresenta a especificação completa dos Requisitos Funcionais, das Regras de Negócio e dos Requisitos Não Funcionais do Schoolink, além da Matriz de Rastreabilidade que relaciona cada requisito à sua implementação no sistema.

Capítulo 4 — Modelagem do Sistema

Descreve a modelagem de dados do banco de dados, as principais entidades do sistema e o fluxo de autenticação implementado.

Capítulo 5 — Arquitetura e Tecnologias

Apresenta a arquitetura da aplicação e descreve as tecnologias utilizadas no desenvolvimento do frontend, do backend e da infraestrutura.

Capítulo 6 — Implementação do Sistema

Detalha a implementação das principais funcionalidades do Schoolink, com destaque para o sistema de autenticação, os dashboards por perfil e o controle de permissões.

Capítulo 7 — Testes e Validação

Descreve a estratégia de testes adotada, os cenários de teste por perfil e os resultados da validação do sistema.

Capítulo 8 — Resultados e Discussões

Apresenta as capturas de tela do sistema em funcionamento, discute os benefícios alcançados, compara com processos manuais e aponta as limitações identificadas.

Capítulo 9 — Conclusão e Trabalhos Futuros

Retoma os objetivos do trabalho, avalia os resultados obtidos e propõe direções para o aprimoramento e expansão do sistema em trabalhos futuros. Ao final, são apresentadas as Referências Bibliográficas e a Lista de Siglas.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os conceitos teóricos que fundamentam o desenvolvimento do sistema Schoolink. O objetivo é contextualizar, para o leitor, os principais temas abordados no projeto — desde a importância dos sistemas de informação na educação até os mecanismos de segurança e os processos de engenharia de requisitos adotados.

2.1 SISTEMAS DE INFORMAÇÃO NA EDUCAÇÃO

Um sistema de informação pode ser definido como um conjunto organizado de pessoas, processos e tecnologias que trabalham juntos para coletar, processar, armazenar e distribuir informações úteis à tomada de decisão dentro de uma organização (LAUDON; LAUDON, 2021). Em termos simples, um sistema de informação é qualquer mecanismo — seja ele manual ou computadorizado — que transforma dados brutos em informações com significado e utilidade.

Quando aplicados ao contexto educacional, esses sistemas recebem a denominação de Sistemas de Informação Educacionais (SIE) e abrangem todas as ferramentas tecnológicas utilizadas para apoiar a gestão pedagógica e administrativa de uma instituição de ensino.

Exemplos comuns incluem sistemas acadêmicos de universidades, plataformas de ensino a distância e sistemas de gerenciamento escolar para a educação básica.

O surgimento e a popularização dos SIEs ocorreram em paralelo com a informatização das organizações em geral. Nos anos 1990 e 2000, muitas escolas brasileiras começaram a substituir diários de classe em papel por sistemas informatizados básicos. Na década de 2010, com a popularização da internet e dos dispositivos móveis, esses sistemas evoluíram para plataformas web acessíveis a qualquer momento, de qualquer dispositivo, por todos os envolvidos no processo educacional (MORAN, 2015).

Do ponto de vista funcional, Laudon e Laudon (2021) classificam os sistemas de informação em três categorias principais, de acordo com o nível de decisão que apoiam:

- Sistemas de processamento de transações (SPT): registram as atividades cotidianas da organização, como matrículas, lançamentos de notas e registros de frequência. São a base operacional de qualquer sistema de gestão escolar.

- Sistemas de informações gerenciais (SIG): consolidam os dados dos SPTs em relatórios e indicadores que auxiliam os gestores (como o Diretor) a monitorar o desempenho da instituição e tomar decisões pedagógicas.

- Sistemas de suporte à decisão (SSD): oferecem análises mais avançadas para decisões estratégicas, como identificar padrões de evasão ou prever resultados de avaliações em larga escala.

O Schoolink atua principalmente nos dois primeiros níveis: registra as transações acadêmicas do dia a dia e consolida essas informações em painéis de controle que auxiliam diretores, professores, alunos e responsáveis na tomada de decisões dentro de seus respectivos contextos.

Segundo O'Brien e Marakas (2013), para que um SIE seja eficaz, ele precisa atender a três dimensões fundamentais:

- Dimensão tecnológica: hardware, software, banco de dados e infraestrutura de rede que suportam o funcionamento do sistema.

- Dimensão organizacional: processos, estrutura e cultura da instituição que o sistema deve refletir e apoiar.

- Dimensão humana: as pessoas que utilizam o sistema, com suas diferentes habilidades, necessidades e expectativas.

O Schoolink foi projetado considerando essas três dimensões: a escolha tecnológica priorizou ferramentas modernas e de ampla adoção no mercado; o modelo de dados reflete a estrutura real de uma escola brasileira; e a interface foi desenvolvida com foco na usabilidade para usuários com diferentes níveis de familiaridade com tecnologia.

2.2 GESTÃO ESCOLAR APOIADA POR TECNOLOGIA

A gestão escolar é o conjunto de atividades necessárias para que uma instituição de ensino funcione de forma organizada, eficiente e orientada aos seus objetivos pedagógicos.

Ela engloba tanto a dimensão administrativa — como controle de matrículas, gestão de funcionários e comunicação com as famílias — quanto a dimensão pedagógica, que inclui o acompanhamento do desempenho dos alunos, a organização curricular e o apoio ao trabalho docente (LUCK, 2009).

Durante décadas, a gestão escolar brasileira foi conduzida predominantemente por meio de registros em papel, processos manuais e comunicações presenciais. Esse modelo, embora amplamente utilizado, apresenta limitações conhecidas: informações fragmentadas entre diferentes cadernos e pastas, dificuldade de acesso histórico aos dados, lentidão na comunicação com as famílias e alta susceptibilidade a erros humanos de registro.

A tecnologia da informação tem se apresentado como um instrumento poderoso para superar essas limitações. De acordo com Ferreira (2012), a incorporação de ferramentas digitais à gestão escolar contribui para:

- Centralização das informações: dados de alunos, professores, notas e frequência passam a ser armazenados em um único repositório, acessível por todos os interessados com as devidas restrições de acesso.

- Agilidade nos processos: tarefas que antes demandavam horas de trabalho manual — como calcular médias ou compilar índices de frequência — são executadas automaticamente pelo sistema em frações de segundo.

- Transparência para as famílias: pais e responsáveis passam a ter acesso em tempo real ao desempenho acadêmico de seus filhos, sem depender de boletins impressos enviados ao final do bimestre.

- Apoio à tomada de decisão: gestores escolares contam com indicadores atualizados para identificar turmas com alto índice de reprovação, disciplinas com maior absenteísmo ou alunos que necessitam de acompanhamento especial.

Apesar dos avanços, a adoção de sistemas de gestão escolar no Brasil ainda enfrenta desafios significativos. Pesquisas apontam que muitas escolas — especialmente as de pequeno porte e as localizadas em regiões com menor infraestrutura digital — ainda dependem de soluções fragmentadas ou improvisadas (VALENTE; ALMEIDA, 2020).

Além disso, a resistência à mudança por parte de professores e gestores habituados a processos manuais é um obstáculo recorrente na implantação de novos sistemas.

O desenvolvimento do Schoolink leva em conta essa realidade ao priorizar uma interface intuitiva e acessível, que minimize o esforço de adaptação dos usuários e permita que mesmo professores com pouca familiaridade tecnológica utilizem o sistema sem dificuldades.

2.3 CONTROLE DE ACESSO BASEADO EM PERFIS (RBAC)

Em qualquer sistema que envolve múltiplos usuários com responsabilidades diferentes, é fundamental definir quem pode fazer o quê. Esse mecanismo é chamado de controle de acesso, e sua implementação adequada é um dos pilares da segurança de qualquer aplicação de software (FERRAIOLO; SANDHU; GAVRILA, 2001).

Ao longo da história da computação, diferentes modelos de controle de acesso foram propostos:

- Controle de Acesso Discricionário (DAC): o proprietário de um recurso decide quem pode acessá-lo. É o modelo mais simples, mas apresenta riscos pois depende exclusivamente do julgamento do usuário.

- Controle de Acesso Obrigatório (MAC): políticas de acesso são definidas centralmente pela organização e não podem ser alteradas pelos usuários. É utilizado principalmente em ambientes militares e governamentais.

- Controle de Acesso Baseado em Perfis (RBAC): usuários recebem um ou mais papéis (perfis), e cada papel possui um conjunto de permissões predefinidas. Um usuário herda as permissões do papel ao qual está associado.

O modelo RBAC, formalizado por Sandhu et al. (1996) e padronizado pelo National Institute of Standards and Technology (NIST) em 2004, tornou-se o mais amplamente adotado em sistemas corporativos e educacionais por oferecer um equilíbrio ideal entre segurança, flexibilidade e facilidade de gerenciamento.

A lógica do RBAC pode ser ilustrada com um exemplo do cotidiano: em um hospital, um médico tem acesso ao prontuário dos pacientes, mas não ao sistema de folha de pagamento; uma recepcionista pode agendar consultas, mas não prescrever medicamentos; o diretor administrativo gerencia contratos, mas não acessa dados clínicos. Cada função (papel) tem um conjunto claro e restrito de permissões.

No Schoolink, o RBAC é implementado com quatro papéis principais:

- Diretor: acesso total à plataforma, incluindo gestão de usuários, configurações da instituição, criação de turmas e disciplinas, publicação de comunicados e visualização de todos os dados acadêmicos da escola.

- Professor: acesso restrito às turmas e disciplinas às quais foi formalmente atribuído. Pode lançar notas, registrar frequência e criar atividades avaliativas apenas dentro de seu contexto de atuação.

- Aluno: acesso exclusivo às próprias informações acadêmicas — boletim, frequência, atividades e comunicados institucionais.

- Responsável: acesso restrito às informações dos alunos a ele vinculados pelo Diretor, podendo acompanhar notas, frequência e atividades dos filhos.

A verificação das permissões ocorre em duas camadas no Schoolink: no frontend, que exibe ou oculta elementos da interface conforme o perfil do usuário, e no backend, por meio de um middleware de autorização que valida o perfil antes de processar qualquer requisição à API. Essa

dupla verificação garante que mesmo um usuário com conhecimento técnico avançado não consiga acessar recursos não autorizados contornando a interface.

2.4 USABILIDADE EM SISTEMAS EDUCACIONAIS

A usabilidade é definida pela norma ISO 9241-11 como "a medida em que um sistema, produto ou serviço pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto de uso específico" (ISO, 2018). Em termos práticos, um sistema usável é aquele que o usuário consegue aprender rapidamente, utilizar sem cometer erros frequentes e do qual sai satisfeito.

A importância da usabilidade em sistemas educacionais é particularmente elevada em comparação com sistemas corporativos tradicionais. Isso ocorre porque os usuários de um sistema escolar formam um grupo extremamente diversificado: professores com décadas de experiência e pouca familiaridade com tecnologia, alunos adolescentes habituados a interfaces móveis, pais de todas as idades e níveis de letramento digital, e diretores que precisam de informações rápidas e objetivas para tomar decisões (MORAN, 2015).

Jakob Nielsen, um dos pesquisadores mais influentes na área de usabilidade, propôs dez princípios heurísticos que servem como guia para o desenvolvimento de interfaces intuitivas (NIELSEN, 1994). Os mais relevantes para o contexto do Schoolink são:

- Visibilidade do estado do sistema: o sistema deve sempre informar ao usuário o que está acontecendo. No Schoolink, isso se traduz em indicadores de carregamento, mensagens de confirmação após salvar dados e alertas visuais de erro.

- Correspondência com o mundo real: o sistema deve utilizar linguagem e conceitos familiares ao usuário, evitando jargões técnicos. O Schoolink adota termos como "boletim", "chamada", "bimestre" e "turma" — vocabulário já familiar ao contexto
escolar brasileiro.

- Controle e liberdade do usuário: usuários cometem erros e precisam de saídas claramente marcadas. O sistema oferece confirmações antes de ações irreversíveis e permite que o usuário volte à tela anterior com facilidade.

- Consistência e padrões: elementos de interface semelhantes devem se comportar da mesma forma em toda a plataforma. O Schoolink utiliza um conjunto de componentes visuais padronizados, garantindo que um botão de "salvar" tenha sempre a mesma aparência e localização.

- Prevenção de erros: o sistema valida os dados inseridos pelo usuário antes de submetê-los ao servidor, indicando claramente quais campos estão incorretos e por quê, evitando que dados inválidos sejam persistidos no banco de dados.

Outro aspecto fundamental da usabilidade contemporânea é o design responsivo, que consiste na capacidade da interface de se adaptar automaticamente ao tamanho da tela do dispositivo utilizado — seja um computador desktop, um tablet ou um smartphone (MARCOTTE, 2011). No contexto escolar, essa característica é essencial, pois muitos alunos e responsáveis acessam plataformas digitais prioritariamente pelo celular.

O Schoolink foi desenvolvido com design responsivo utilizando Tailwind CSS, garantindo que todas as funcionalidades do sistema sejam acessíveis e utilizáveis em qualquer dispositivo, sem degradação de qualidade ou perda de recursos.

2.5 SEGURANÇA DA INFORMAÇÃO E LGPD

A segurança da informação refere-se ao conjunto de práticas, processos e tecnologias destinados a proteger informações contra acessos não autorizados, uso indevido, alterações não permitidas ou destruição (ISO/IEC 27001, 2022). Em um mundo cada vez mais conectado, a segurança deixou de ser uma preocupação exclusiva de grandes corporações e passou a ser uma responsabilidade de qualquer sistema que trate dados de pessoas — especialmente dados sensíveis, como informações acadêmicas de crianças e adolescentes.

A segurança da informação é estruturada em torno de três pilares fundamentais, conhecidos como a tríade CID:

- Confidencialidade: a informação deve ser acessível apenas por quem tem autorização para isso. No Schoolink, isso é garantido pelo mecanismo de autenticação JWT e pelo controle de acesso RBAC, que impede que um usuário visualize dados de outros usuários ou de outras instituições.

- Integridade: a informação não deve ser alterada de forma não autorizada. O uso de transações no banco de dados garante que operações complexas sejam executadas completamente ou não sejam executadas, evitando registros incompletos ou corrompidos.

- Disponibilidade: a informação deve estar acessível quando necessária. Isso é alcançado por meio de infraestrutura confiável, monitoramento de disponibilidade e comunicação prévia de manutenções programadas.

No contexto das aplicações web — categoria na qual o Schoolink se enquadra — existem vulnerabilidades de segurança específicas que precisam ser tratadas durante o desenvolvimento. O

projeto OWASP (Open Web Application Security Project) mantém uma lista das dez vulnerabilidades mais críticas em aplicações web (OWASP, 2021), entre as quais se destacam:

- SQL Injection: ataque em que um usuário malicioso insere comandos SQL em campos de formulário para manipular o banco de dados. O Schoolink se protege contra esse ataque utilizando o ORM Prisma, que parametriza automaticamente todas as consultas ao banco de dados.

- Cross-Site Scripting (XSS): injeção de scripts maliciosos em páginas web que afetam outros usuários. O Next.js, framework frontend utilizado no Schoolink, trata automaticamente a maioria dos casos de XSS por padrão.

- Cross-Site Request Forgery (CSRF): ataque que força um usuário autenticado a executar ações não desejadas. O uso de tokens JWT no cabeçalho das requisições — ao invés de cookies — mitiga esse tipo de ataque.

- Quebra de autenticação: senhas fracas, tokens expostos ou sessões mal gerenciadas. O Schoolink trata isso com hashing bcrypt para senhas, tokens JWT com expiração de 24 horas e limitação de tentativas de login.

O Schoolink também utiliza o middleware Helmet.js no backend, que configura automaticamente cabeçalhos HTTP de segurança para proteger a aplicação contra uma série de ataques comuns.

2.5.1 Lei Geral de Proteção de Dados Pessoais (LGPD)

No Brasil, o tratamento de dados pessoais passou a ser regulamentado pela Lei nº 13.709, de 14 de agosto de 2018, conhecida como Lei Geral de Proteção de Dados Pessoais (LGPD). Inspirada no Regulamento Geral de Proteção de Dados (GDPR) europeu, a LGPD estabelece regras claras sobre como empresas e organizações devem coletar, armazenar, processar e compartilhar dados pessoais de cidadãos brasileiros (BRASIL, 2018).

Para o contexto escolar, a LGPD tem implicações especialmente relevantes. Escolas lidam diariamente com dados pessoais de crianças e adolescentes — categoria de dados que a própria lei classifica como merecendo proteção especial (art. 14) — incluindo nome completo, CPF, data de nascimento, endereço, informações de saúde e histórico acadêmico.

Os princípios da LGPD mais relevantes para o desenvolvimento do Schoolink são:

- Finalidade: os dados coletados devem ser utilizados apenas para os propósitos informados ao titular. O Schoolink coleta dados pessoais exclusivamente para fins de gestão acadêmica.

- Necessidade: devem ser coletados apenas os dados estritamente necessários para a finalidade declarada. O sistema solicita apenas as informações indispensáveis para cada cadastro.

- Segurança: medidas técnicas e administrativas devem ser adotadas para proteger os dados contra acessos não autorizados e situações acidentais ou ilícitas de destruição, perda, alteração ou comunicação. O Schoolink implementa criptografia
de senhas, controle de acesso por perfis, autenticação segura e isolamento de dados entre instituições.

- Transparência: os titulares dos dados têm direito de saber quais informações são coletadas e como são utilizadas. O desenvolvimento do Schoolink foi conduzido com atenção a esses princípios, buscando construir uma plataforma que não apenas funcione bem, mas que também respeite a privacidade dos alunos, professores e famílias que a utilizam.

2.6 ENGENHARIA DE REQUISITOS DE SOFTWARE

O desenvolvimento de um sistema de software bem-sucedido começa muito antes de qualquer linha de código ser escrita. É necessário, antes de tudo, compreender profundamente o que o sistema precisa fazer, quais restrições deve respeitar e quais qualidades deve possuir. Esse processo de compreensão, documentação e validação das necessidades é chamado de Engenharia de Requisitos (SOMMERVILLE, 2019).

Segundo Pressman e Maxim (2016), a Engenharia de Requisitos é a atividade de engenharia de software que tem o maior impacto sobre o sucesso ou fracasso de um projeto. Estudos indicam que a maioria dos projetos de software fracassa não por
problemas técnicos de implementação, mas por falhas na fase de levantamento e especificação de requisitos — seja por requisitos mal definidos, incompletos, ambíguos ou que mudam excessivamente durante o desenvolvimento.

Para evitar esses problemas, a Engenharia de Requisitos estabelece um conjunto de atividades estruturadas:

- Elicitação: processo de identificar e coletar as necessidades dos usuários e stakeholders do sistema. As técnicas mais comuns incluem entrevistas, questionários, observação do ambiente de trabalho e análise de documentos existentes.

- Análise: após a elicitação, os requisitos coletados são analisados para identificar conflitos, ambiguidades, redundâncias e lacunas. Requisitos conflitantes precisam ser negociados e resolvidos antes de avançar.

- Especificação: os requisitos são documentados de forma clara, precisa e verificável. Uma especificação de boa qualidade deve permitir que, ao final do desenvolvimento, seja possível verificar objetivamente se cada requisito foi ou não atendido.

- Validação: os requisitos especificados são revisados com os usuários e stakeholders para confirmar que refletem corretamente as necessidades identificadas.

2.6.1 Tipos de Requisitos

Os requisitos de software são classicamente divididos em três tipos, conforme proposto por Ventura (2016):

Requisitos Funcionais (RF): Descrevem as funcionalidades que o sistema deve executar — o que o sistema faz. São as ações visíveis para o usuário: autenticar, cadastrar, listar, calcular, exibir. Para o Schoolink, exemplos de Requisitos Funcionais incluem "lançar nota de aluno em disciplina" e "registrar chamada dos alunos em aula".

Requisitos Não Funcionais (RNF): Descrevem qualidades e restrições técnicas que o sistema deve possuir, sem descrever funcionalidades específicas. Respondem à pergunta: como o sistema deve se comportar? Exemplos incluem "o sistema deve responder em até 3 segundos" (desempenho), "as senhas devem ser armazenadas com hash bcrypt" (segurança) e "a interface deve funcionar em smartphones" (compatibilidade). Embora invisíveis ao usuário final, os RNFs frequentemente determinam a arquitetura técnica do sistema.

Regras de Negócio (RN): Descrevem restrições e premissas provenientes do contexto real de negócio que o sistema deve respeitar. São as "leis" do domínio de aplicação, independentes de tecnologia. Exemplos no Schoolink: "um aluno não pode ter duas matrículas ativas no mesmo ano letivo" e "a chamada não pode ser registrada para datas futuras". Regras de Negócio diferem dos Requisitos Funcionais porque não descrevem funcionalidades, mas condições que as funcionalidades devem respeitar.

2.6.2 Rastreabilidade de Requisitos

A rastreabilidade de requisitos é a capacidade de acompanhar a vida de um requisito desde a sua origem até a sua implementação no sistema final. Segundo Sommerville (2019), um sistema com boa rastreabilidade permite que cada requisito seja vinculado à tela da interface onde pode ser verificado e ao código ou endpoint da API que o implementa.

A rastreabilidade é importante por múltiplas razões: facilita a verificação de que todos os requisitos foram implementados, auxilia na identificação do impacto de mudanças em requisitos existentes e serve como documentação viva do sistema desenvolvido.

No Schoolink, a rastreabilidade foi documentada por meio de uma Matriz de Rastreabilidade que relaciona cada Requisito Funcional à tela correspondente na interface do usuário e ao endpoint da API responsável por sua implementação, conforme apresentado no Capítulo 3 deste trabalho.

3 ANÁLISE DE REQUISITOS

3.1 INTRODUÇÃO

Todo sistema de software nasce a partir de necessidades. Antes de qualquer linha de código ser escrita, é fundamental que se compreenda e documente aquilo que o sistema deverá ser capaz de fazer. Esse conjunto de necessidades, quando documentado de forma organizada, recebe o nome de requisitos de software.

Dentro da área de Engenharia de Software, os requisitos são divididos em categorias, de acordo com a sua natureza. Este documento trata especificamente dos Requisitos Funcionais(RF), que são aqueles que descrevem as funcionalidades do sistema, ou seja, aquilo que o sistema deve fazer (VENTURA, 2016).

Em termos simples, um Requisito Funcional responde à pergunta: "O que o sistema precisa executar?". Por exemplo: autenticar um usuário, exibir as notas de um aluno, registrar a presença em uma aula. Cada uma dessas ações é um Requisito Funcional distinto.

Para que os requisitos sejam úteis ao desenvolvimento, eles precisam ser escritos com clareza, sem ambiguidade e de forma verificável — ou seja, deve ser possível confirmar, ao final do desenvolvimento, se cada requisito foi ou não atendido pelo sistema (SOMMERVILLE, 2019).

O sistema Schoolink é uma plataforma de gerenciamento escolar que atende quatro perfis de usuário: Diretor, Professor, Aluno e Responsável (pai ou guardião). Cada perfil possui acesso restrito às funcionalidades pertinentes ao seu papel dentro da instituição de ensino.

Os Requisitos Funcionais deste documento estão organizados por módulos, que são agrupamentos lógicos de funcionalidades com objetivo comum. Cada requisito possui os seguintes campos:

- Identificador: código único no formato RF + número sequencial (ex.: RF0001)

- Nome: descrição resumida da funcionalidade

- Módulo: agrupamento ao qual o requisito pertence

- Data de Criação: data em que o requisito foi especificado

- Autor: responsável pela especificação

- Versão: número de versão do requisito (inicia em 1)

- Prioridade: classificação em Essencial, Importante ou Desejável

- Descrição: detalhamento completo do comportamento esperado

3.2 MÓDULOS DO SISTEMA

O Schoolink é organizado nos seguintes módulos funcionais:

MOD-01 - Autenticação

MOD-02 - Gestão de Usuários

MOD-03 - Estrutura Acadêmica

MOD-04 - Ano Letivo e Períodos

MOD-05 - Matrículas

MOD-06 - Avaliações e Notas

MOD-07 - Frequência

MOD-08 - Comunicados

MOD-09 - Configurações da Instituição

MOD-10 - Painel de Controle

3.3 REQUISITOS FUNCIONAIS

3.3.1 Módulo: Autenticação (MOD-01)

RF0001 - Autenticar usuário por e-mail e senha

Módulo: Autenticação | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O sistema deve permitir que qualquer usuário cadastrado acesse a plataforma por meio do fornecimento de seu endereço de e-mail e senha. Ao submeter as credenciais, o sistema deve verificá-las no banco de dados. Se as credenciais forem válidas e a conta estiver ativa, o sistema deve gerar um token de autenticação e redirecionar o usuário para o painel correspondente ao seu perfil (Diretor, Professor, Aluno ou Responsável). Caso as credenciais sejam inválidas ou a conta esteja desativada, o sistema deve exibir uma mensagem de erro clara ao usuário, sem revelar qual dos campos está incorreto.

RF0002 - Encerrar sessão autenticada

Módulo: Autenticação | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O sistema deve permitir que o usuário autenticado encerre sua sessão de forma segura a qualquer momento. Ao acionar a opção de saída (logout), o sistema deve invalidar o token de autenticação ativo e redirecionar o usuário para a tela de login. Após o encerramento da sessão, nenhum acesso às páginas protegidas deve ser permitido com o token invalidado.

RF0003 - Controlar acesso às funcionalidades por perfil de usuário

Módulo: Autenticação | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O sistema deve garantir que cada usuário autenticado tenha acesso apenas às funcionalidades e informações correspondentes ao seu perfil. Um usuário com perfil de Responsável, por exemplo, não pode acessar as telas de lançamento de notas, exclusivas do Professor. Um Professor não pode acessar as configurações da instituição, exclusivas do Diretor. Qualquer tentativa de acesso a uma funcionalidade não permitida deve ser bloqueada pelo sistema e retornar uma mensagem de acesso negado.

3.2 Módulo: Gestão de Usuários (MOD-02)

RF0004 - Cadastrar novo usuário na instituição

Módulo: Gestão de Usuários | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve ser capaz de cadastrar novos usuários na plataforma. O cadastro deve solicitar, no mínimo, o nome completo, o endereço de e-mail, a senha inicial e o perfil de acesso (Diretor, Professor, Aluno ou Responsável). Ao cadastrar um usuário com perfil de Aluno, o sistema deve criar automaticamente um perfil acadêmico com número de matrícula gerado pelo próprio sistema. Ao cadastrar um Professor, o sistema deve criar automaticamente um perfil profissional com número de registro. Ao cadastrar um Responsável, o sistema deve criar automaticamente o perfil correspondente. Não devem ser aceitos dois usuários com o mesmo endereço de e-mail na mesma instituição.

RF0005 - Editar dados de usuário existente

Módulo: Gestão de Usuários | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder editar os dados de qualquer usuário cadastrado na instituição. Os campos editáveis incluem nome completo, endereço de e-mail e perfil de acesso. O sistema deve validar o novo e-mail para evitar duplicidade. Alterações realizadas devem ser salvas imediatamente e refletidas no acesso subsequente do usuário.

RF0006 - Ativar ou desativar conta de usuário

Módulo: Gestão de Usuários | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder ativar ou desativar a conta de qualquer usuário da instituição. Um usuário com conta desativada não deve conseguir fazer login na plataforma. O sistema não deve excluir os dados do usuário ao desativá-lo, preservando o histórico acadêmico e operacional associado. A reativação deve restaurar o acesso completo conforme o perfil do usuário.

RF0007 - Listar usuários cadastrados na instituição

Módulo: Gestão de Usuários | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve ter acesso a uma lista de todos os usuários cadastrados na instituição. A lista deve exibir, para cada usuário, nome completo, e-mail, perfil de acesso e situação da conta (ativa ou

inativa). O sistema deve disponibilizar um campo de busca que permita filtrar os usuários por nome ou e-mail.

RF0008 - Vincular responsável a aluno

Módulo: Gestão de Usuários | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder vincular um usuário com perfil de Responsável a um ou mais alunos cadastrados. Após o vínculo, o Responsável passa a ter acesso às informações acadêmicas dos alunos a ele associados (notas, frequência, comunicados e atividades). Um aluno pode ter mais de um responsável vinculado. Um responsável pode estar vinculado a mais de um aluno.

3.3.3 Módulo: Estrutura Acadêmica (MOD-03)

RF0009 - Cadastrar série escolar

Módulo: Estrutura Acadêmica | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder cadastrar as séries escolares que compõem a grade curricular da instituição (ex.: 1º Ano do Ensino Fundamental, 7º Ano, 1º Ano do Ensino Médio). Cada série deve conter nome, nível de ensino (Fundamental ou Médio) e ordem de progressão dentro do nível. As séries criadas ficam disponíveis para a criação de turmas.

RF0010 - Cadastrar disciplina vinculada à série

Módulo: Estrutura Acadêmica | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder cadastrar as disciplinas pertencentes a cada série escolar (ex.: Matemática, Português, Ciências). Cada disciplina deve conter nome, código de identificação único e carga horária semanal. Uma disciplina está sempre associada a uma única série.

RF0011 - Cadastrar turma vinculada à série

Módulo: Estrutura Acadêmica | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder criar turmas dentro de cada série escolar (ex.: 7º Ano A, 7º Ano B). Cada turma deve ter um nome identificador, a sala de aula onde as aulas ocorrem e o número máximo de alunos permitidos. Uma turma pertence a uma única série e a um único ano letivo.

RF0012 - Atribuir professor a disciplina de uma turma

Módulo: Estrutura Acadêmica | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder associar um Professor a uma ou mais disciplinas dentro de uma turma específica. Essa associação define quais turmas e disciplinas o Professor pode visualizar, lançar notas e registrar frequência. Um Professor pode lecionar disciplinas diferentes em turmas diferentes. Uma mesma disciplina em uma mesma turma deve ter apenas um Professor responsável por vez.

RF0013 - Adicionar aluno a uma turma

Módulo: Estrutura Acadêmica | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder adicionar alunos já cadastrados na plataforma a uma turma específica. O sistema deve listar apenas os alunos disponíveis para aquela turma, ou seja, alunos que ainda não estão matriculados naquela turma no ano letivo vigente. Ao adicionar um aluno à turma, o sistema deve registrar automaticamente a matrícula do aluno no ano letivo corrente.

3.3.4 Módulo: Ano Letivo e Períodos (MOD-04)

RF0014 - Cadastrar ano letivo

Módulo: Ano Letivo e Períodos | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder cadastrar o ano letivo da instituição, informando o ano de referência, a data de início e a data de encerramento das atividades. Apenas um ano letivo pode estar ativo por vez. O ano letivo ativo é utilizado como referência para todos os registros acadêmicos (matrículas, notas e frequência).

RF0015 - Cadastrar períodos letivos (bimestres)

Módulo: Ano Letivo e Períodos | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder cadastrar os períodos que compõem o ano letivo, como bimestres ou semestres. Cada período deve ter nome (ex.: 1º Bimestre), número de ordem, data de início e data de encerramento. Um período pode estar nos estados ativo, inativo ou encerrado. O período ativo é aquele em que os lançamentos de notas e frequência estão habilitados.

3.3.5 Módulo: Matrículas (MOD-05)

RF0016 - Registrar matrícula de aluno em turma

Módulo: Matrículas | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: ARCRAIDERS

O sistema deve registrar a matrícula de um aluno em uma turma durante o ano letivo vigente. A matrícula define a qual turma o aluno pertence e é a base para os registros de notas e frequência. Cada aluno pode ter apenas uma matrícula ativa por ano letivo. A matrícula pode assumir os estados: ativa, transferida, cancelada, suspensa ou concluída.

RF0017 - Consultar situação de matrícula do aluno

Módulo: Matrículas | Prioridade: Importante | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder consultar a situação da matrícula de qualquer aluno da instituição. A consulta deve exibir a turma em que o aluno está matriculado, o ano letivo correspondente e o estado atual da matrícula. Responsáveis também devem poder visualizar a situação da matrícula de seus filhos através de seus próprios painéis.

3.3.6 Módulo: Avaliações e Notas (MOD-06)

RF0018 - Criar atividade avaliativa para uma turma

Módulo: Avaliações e Notas | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Professor deve poder criar atividades avaliativas para as turmas e disciplinas que leciona. Cada atividade deve conter título, tipo (prova, trabalho ou exercício), peso na composição da média, nota máxima possível e data de entrega ou realização. As atividades criadas ficam visíveis para os Alunos e Responsáveis associados àquela turma, servindo como agenda de avaliações.

RF0019 - Lançar nota de aluno em disciplina

Módulo: Avaliações e Notas | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Professor deve poder lançar notas individuais para cada aluno nas disciplinas que leciona, referentes ao período letivo em vigor. A nota lançada deve estar dentro do intervalo permitido (de zero à nota máxima configurada, geralmente 10,0). O sistema deve calcular e exibir automaticamente a média parcial do aluno na disciplina após cada lançamento. Notas já lançadas podem ser editadas pelo Professor durante o período ativo.

RF0020 - Visualizar boletim do aluno

Módulo: Avaliações e Notas | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Aluno deve poder visualizar seu próprio boletim, contendo as notas lançadas em cada disciplina por período letivo. O Responsável deve poder visualizar o boletim dos alunos a ele vinculados. O boletim deve exibir, para cada disciplina: as notas por período, a média calculada e a situação do aluno (aprovado, em recuperação ou reprovado), de acordo com os critérios configurados pela instituição.

RF0021 - Calcular média do aluno por disciplina

Módulo: Avaliações e Notas | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O sistema deve calcular automaticamente a média do aluno em cada disciplina com base nas notas lançadas e nos pesos definidos para cada atividade avaliativa. O cálculo deve ser atualizado sempre que uma nova nota for lançada ou editada. A média calculada deve ser comparada com a nota mínima para aprovação configurada pela instituição, e o sistema deve indicar visualmente se o aluno está aprovado, em situação de risco ou reprovado.

3.3.7 Módulo: Frequência (MOD-07)

RF0022 - Registrar chamada dos alunos em aula

Módulo: Frequência | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Professor deve poder registrar a presença ou ausência de cada aluno da turma em uma determinada aula, informando a disciplina e a data. Para cada aluno, o registro deve assumir um dos seguintes estados: presente, ausente ou ausência justificada. O sistema não deve permitir o registro de chamada para uma data futura. Chamadas já registradas podem ser editadas pelo Professor enquanto o período letivo estiver ativo.

RF0023 - Consultar frequência do aluno por disciplina

Módulo: Frequência | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Aluno deve poder consultar seu percentual de presença em cada disciplina. O Responsável deve poder consultar a frequência dos alunos a ele vinculados. O Professor deve poder visualizar o histórico de presença de todos os alunos de sua turma em uma disciplina. O sistema deve exibir o total de aulas realizadas, o total de presenças, o total de faltas e o percentual de frequência calculado.

RF0024 - Calcular percentual de presença do aluno

Módulo: Frequência | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O sistema deve calcular automaticamente o percentual de presença de cada aluno em cada disciplina, com base nos registros de chamada efetuados pelo Professor. O percentual deve ser comparado com a frequência mínima para aprovação configurada pela instituição. O sistema deve indicar visualmente quando o aluno estiver próximo do limite de faltas ou já tiver ultrapassado esse limite.

3.3.8 Módulo: Comunicados (MOD-08)

RF0025 - Publicar comunicado institucional

Módulo: Comunicados | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder publicar comunicados direcionados a todos os usuários da instituição ou a grupos específicos. Cada comunicado deve ter título, conteúdo textual, tipo (acadêmico ou administrativo), prioridade (normal ou alta) e data de publicação. Comunicados com prioridade alta devem ser destacados visualmente na interface de todos os destinatários.

RF0026 - Visualizar comunicados recebidos

Módulo: Comunicados | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

Todos os usuários autenticados devem poder acessar os comunicados publicados pela instituição que sejam direcionados ao seu perfil. Os comunicados devem ser exibidos em ordem cronológica decrescente (do mais recente ao mais antigo). Comunicados com alta prioridade devem aparecer no topo da lista independentemente da data de publicação.

3.3.9 Módulo: Configurações da Instituição (MOD-09)

RF0027 - Configurar nota mínima para aprovação

Módulo: Configurações da Instituição | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder definir e alterar a nota mínima necessária para que um aluno seja considerado aprovado em uma disciplina. O valor configurado deve ser aplicado a todas as disciplinas e turmas da instituição. O sistema deve utilizar esse valor como referência para calcular e exibir a situação de aprovação dos alunos no boletim.

RF0028 - Configurar frequência mínima para aprovação

Módulo: Configurações da Instituição | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder definir e alterar o percentual mínimo de presença exigido para que um aluno seja considerado aprovado por frequência. O valor configurado deve ser aplicado a todas as disciplinas da instituição. O sistema deve utilizar esse percentual como base para indicar ao aluno, ao responsável e ao professor quando o limite de faltas estiver sendo atingido.

RF0029 - Personalizar identidade visual da instituição

Módulo: Configurações da Instituição | Prioridade: Importante | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O Diretor deve poder personalizar a identidade visual da plataforma para a sua instituição, definindo a cor primária e a cor secundária utilizadas na interface. As cores configuradas devem ser aplicadas automaticamente a todos os elementos visuais da plataforma para os usuários daquela instituição, proporcionando uma experiência alinhada à identidade da escola.

3.3.10 Módulo: Painel de Controle (MOD-10)

RF0030 - Exibir painel de controle personalizado por perfil

Módulo: Painel de Controle | Prioridade: Essencial | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

Após o login, o sistema deve exibir um painel de controle (dashboard) personalizado de acordo com o perfil do usuário autenticado. O painel do Diretor deve apresentar indicadores gerais da instituição. O painel do Professor deve exibir suas turmas, atividades recentes e quantidade de alunos. O painel do Aluno deve mostrar suas notas recentes, próximas atividades e percentual de frequência. O painel do Responsável deve apresentar um resumo do desempenho e frequência dos filhos vinculados.

RF0031 - Exibir estatísticas gerais da instituição para o Diretor

Módulo: Painel de Controle | Prioridade: Importante | Versão: 1 | Data: 01/03/2026 | Autor: Caio Marianni

O painel do Diretor deve exibir um conjunto de indicadores quantitativos da instituição, incluindo: total de alunos matriculados, total de professores ativos, total de turmas no ano letivo vigente e total de atividades cadastradas. Esses indicadores devem ser atualizados automaticamente sempre que houver alterações nos dados correspondentes.

3.4 REGRAS DE NEGÓCIO

As regras de negócio do sistema Schoolink foram extraídas diretamente do código-fonte, com base nas restrições do banco de dados definidas pelo Prisma ORM, nas validações dos serviços da camada de aplicação e na lógica de controle de acesso implementada no backend. O Quadro X apresenta as vinte regras identificadas.

ID	Regra de Negócio
RN01	A nota mínima de aprovação é configurável por instituição, com valor padrão de 6,0 em escala de 0 a 10.
RN02	A frequência mínima obrigatória é configurável por instituição, com valor padrão de 75%.
RN03	Um aluno pode ter somente uma matrícula ativa por ano letivo. Tentativas de segunda matrícula no mesmo ano são bloqueadas por restrição única no banco de dados.
RN04	Um professor pode lecionar uma disciplina por turma simultaneamente. O sistema impede duplicidade por restrição única nos campos teacherId, subjectId e classId.
RN05	Somente o professor atribuído a uma disciplina em uma turma pode lançar ou editar notas para aquele grupo de alunos.
RN06	Quando uma nota é editada, o sistema registra o valor original, sinaliza a edição e exige um motivo para a alteração.
RN07	A frequência é registrada por aluno, disciplina e data. O sistema impede registros duplicados para a mesma combinação.
RN08	O login pode ser realizado com e-mail ou CPF, além da senha e do perfil de acesso selecionado.
RN09	No primeiro acesso, o usuário é obrigado a trocar a senha antes de utilizar o sistema.
RN10	Após um número configurável de tentativas de login com credenciais incorretas, a conta é bloqueada temporariamente.
RN11	O token de redefinição de senha possui prazo de validade e pode ser utilizado somente uma vez.
RN12	A autenticação de dois fatores (2FA via TOTP) pode ser configurada como obrigatória para administradores da instituição.
RN13	Comunicados podem ter data de publicação e data de expiração opcionais, controlando o período de exibição.

RN14	Alterações sensíveis nos dados de um usuário devem passar por um fluxo de aprovação com os estados: PENDENTE, APROVADO ou REJEITADO.
RN15	A matrícula de um aluno possui os seguintes estados possíveis: ATIVO, TRANSFERIDO, EVADIDO, CONCLUÍDO e SUSPENSO.
RN16	Alunos com nota abaixo do mínimo podem receber um Plano de Recuperação, criado pelo professor e aprovado pelo coordenador.
RN17	Cada aluno pode realizar somente uma entrega por atividade. O sistema impede duplicidade por restrição única nos campos activityId e studentId.
RN18	O sistema de avaliação é configurável por instituição: numérico (0 a 10), conceitual (A a E) ou misto.
RN19	Cada turma possui um limite máximo de alunos. Matrículas além do limite são bloqueadas automaticamente pelo sistema.
RN20	Um responsável pode estar vinculado a mais de um aluno, e um aluno pode ter mais de um responsável, caracterizando uma relação muitos para muitos.

Fonte: elaborado pelo autor (2026).

3.5 REQUISITOS NAO FUNCIONAIS

Os requisitos não funcionais, também denominados requisitos de qualidade, definem as propriedades e restrições sob as quais o sistema deve operar. Eles não descrevem o que o sistema faz, mas sim como ele deve se comportar em termos de desempenho, segurança, manutenibilidade e outros atributos de qualidade. O Quadro X apresenta os dezesseis requisitos não funcionais identificados no sistema Schoolink.

Quadro X — Requisitos Não Funcionais do Sistema Schoolink

ID	Categoria	Descrição
RNF01	Segurança	Senhas armazenadas exclusivamente como hash bcrypt com fator de custo 10; o texto plano nunca é persistido em banco de dados.

RNF02	Segurança	Autenticação via JWT com algoritmo HS256 e expiração configurada em 24 horas.
RNF03	Segurança	Suporte a autenticação de dois fatores baseada em TOTP, com geração de códigos de recuperação de uso único.
RNF04	Segurança	Registro completo de auditoria para operações de criação, atualização e exclusão, contendo usuário executor, endereço IP, agente e valores alterados.
RNF05	Segurança	Mecanismo de bloqueio automático de conta após tentativas consecutivas de login malsucedidas.
RNF06	Desempenho	Lançamento de notas e registros de frequência realizados em operações em lote, reduzindo o número de transações ao banco de dados.
RNF07	Escalabilidade	Arquitetura multi-tenant: os dados de cada instituição são isolados logicamente pelo campo institutionId presente em todas as entidades do sistema.
RNF08	Usabilidade	Painel de controle personalizado por perfil de acesso; cada papel possui uma visão distinta com métricas e ações relevantes à sua função.
RNF09	Usabilidade	Interface responsiva construída com Tailwind CSS e componentes Radix UI, adaptando-se a diferentes tamanhos de tela e dispositivos.
RNF10	Disponibilidade	Endpoint de verificação de saúde do servidor disponível para ferramentas de monitoramento e orquestração de contêineres.
RNF11	Manutenibilidade	Separação estrita em camadas: Rotas, Controladores, Serviços e ORM, garantindo coesão e baixo acoplamento entre os componentes.
RNF12	Manutenibilidade	Validação de entrada centralizada com a biblioteca Zod em todas as rotas da API, com mensagens de erro padronizadas e tipagem estática.

RNF13	Portabilidade	Suporte a contêineres Docker com arquivos de composição para ambiente de desenvolvimento e produção; configurado para implantação na plataforma Render.
RNF14	Internacionalização	Fuso horário e idioma configuráveis individualmente por usuário, permitindo uso em diferentes regiões.
RNF15	Rastreabilidade	Logs estruturados com a biblioteca Winston, com rotação diária de arquivos e separação por nível de severidade: ERRO, AVISO, INFO e DEPURAÇÃO.
RNF16	Conformidade	Campos de CPF são armazenados e validados de acordo com o padrão de identificação brasileiro.

Fonte: elaborado pelo autor (2026).

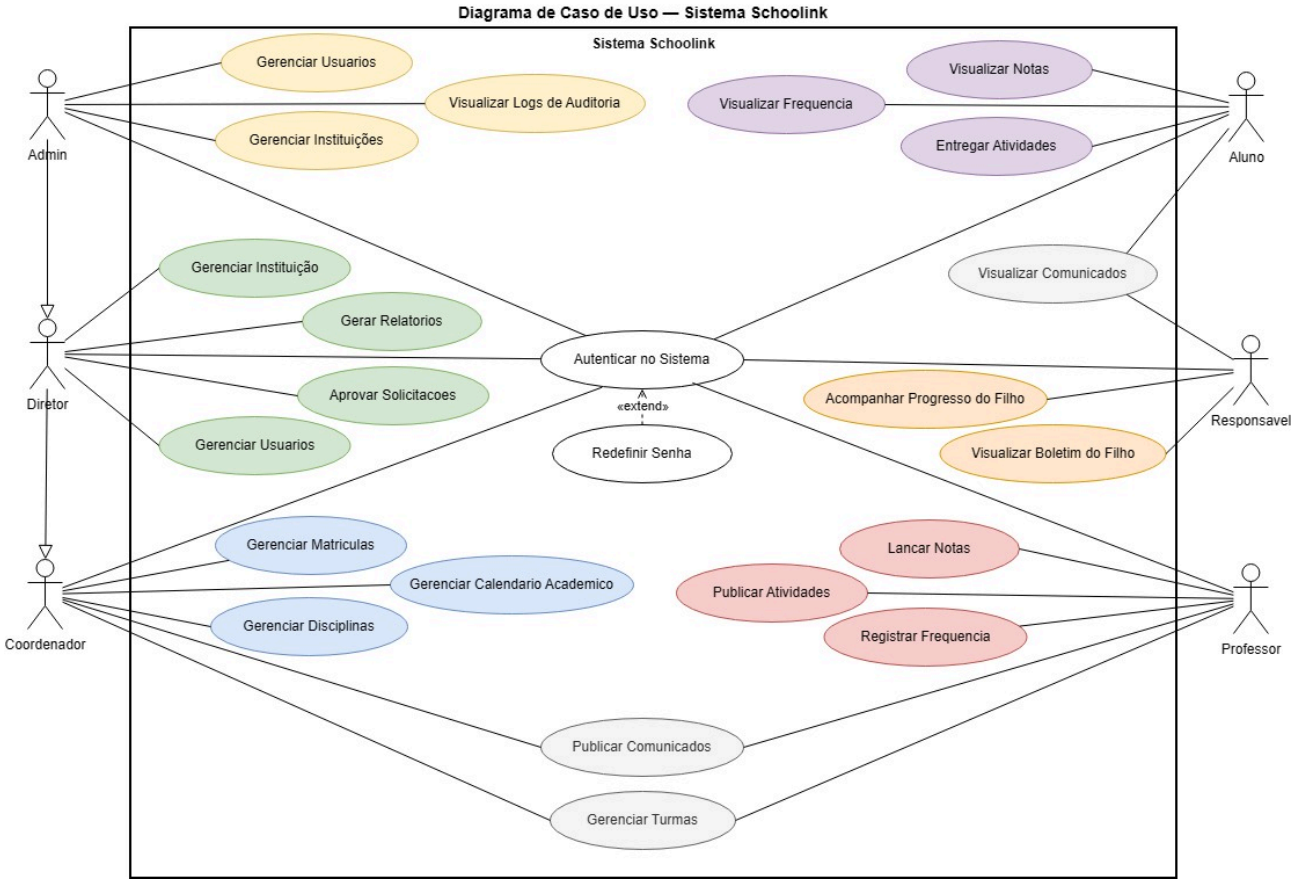
3.6 DIAGRAMA DE CASO DE USO

O diagrama de caso de uso tem como objetivo descrever as funcionalidades do sistema sob a perspectiva dos seus usuários, evidenciando as interações entre os atores e os casos de uso disponíveis. O sistema Schoolink possui seis atores principais, cada um com um conjunto específico de responsabilidades.

O ator Administrador possui as maiores permissões do sistema, sendo responsável pela configuração da instituição e pelo gerenciamento de usuários. O Diretor herda todas as capacidades do Administrador e ainda pode aprovar solicitações e gerar relatórios gerenciais. O Coordenador, por sua vez, herda as capacidades do Diretor e é responsável pela gestão acadêmica: turmas, disciplinas, matrículas e calendário. Essa hierarquia é representada no diagrama por relacionamentos de generalização entre atores, indicando que o ator filho executa todos os casos de uso do ator pai além dos seus próprios.

O Professor é um ator independente da hierarquia administrativa e possui acesso exclusivo às operações pedagógicas: lançamento de notas, registro de frequência e publicação de atividades. O Aluno acessa o sistema para visualizar seu desempenho acadêmico e entregar atividades. O Responsável, por fim, acompanha o progresso do aluno vinculado à sua conta.

Os casos de uso Redefinir Senha e Ativar Autenticação de Dois Fatores estendem o caso de uso Autenticar no Sistema por meio do relacionamento <>, indicando que são acionados opcionalmente a partir da autenticação.



[Figura 1 —Diagrama de Caso de Uaso do Sistema Schoolink](Fonte: elaborado pelo autor)

4 MODELAGEM DO SISTEMA

4.1 MODELAGEM DE DADOS E DIAGRAMA DE ENTIDADE

O Modelo Entidade-Relacionamento (MER) representa a estrutura lógica do banco de dados do sistema Schoolink, identificando as entidades, seus atributos e os relacionamentos entre elas. O modelo foi elaborado a partir do esquema de banco de dados definido com o Prisma ORM, que utiliza o PostgreSQL como sistema gerenciador de banco de dados.

O sistema é composto por entidades organizadas em cinco grupos funcionais. O primeiro grupo abrange as entidades de acesso e identidade: Institution e User. O segundo grupo contempla os perfis específicos de cada papel: StudentProfile, TeacherProfile, CoordinatorProfile e ParentProfile. O terceiro grupo representa a estrutura acadêmica: AcademicYear, AcademicPeriod, Grade, Class e Subject. O quarto grupo é responsável pelo desempenho e acompanhamento dos alunos: Enrollment, Attendance, StudentGrade e Activity. O quinto grupo abrange a comunicação e agenda: Announcement e ActivitySubmission.

A arquitetura multi-tenant do sistema é implementada pela presença do campo institutionId nas entidades de nível institucional, garantindo o isolamento lógico dos dados entre diferentes instituições cadastradas na plataforma.

4.2 PRINCIPAIS ENTIDADES DO SISTEMA

Esta seção apresenta as principais entidades que compõem o banco de dados do Schoolink, descrevendo a finalidade de cada uma e os campos mais relevantes que ela armazena. As entidades estão organizadas em grupos funcionais para facilitar a compreensão.

4.2.1 DIAGRAMA DE CLASSES

O diagrama de classes representa a estrutura estática do sistema Schoolink, descrevendo as classes que compõem o domínio da aplicação, seus atributos e os relacionamentos entre elas. As

classes foram derivadas diretamente do esquema de dados do sistema, refletindo as entidades persistidas no banco de dados e suas associações.

O relacionamento entre Institution e User é de um para muitos, uma vez que cada instituição pode possuir múltiplos usuários. A classe User mantém relacionamentos de um para zero-ou-um com as classes de perfil específico — StudentProfile e TeacherProfile —, pois um usuário assume exatamente um papel no sistema. A classe Class (Turma) se relaciona com Grade e AcademicYear, determinando a que série e ano letivo a turma pertence. A entidade SubjectAssignment realiza a associação entre turma, disciplina e professor responsável, sendo o ponto central das operações pedagógicas. A partir dela derivam as atividades e, consequentemente, as entregas dos alunos. As notas e registros de frequência são vinculados à matrícula do aluno, permitindo rastreabilidade completa do desempenho por período acadêmico.

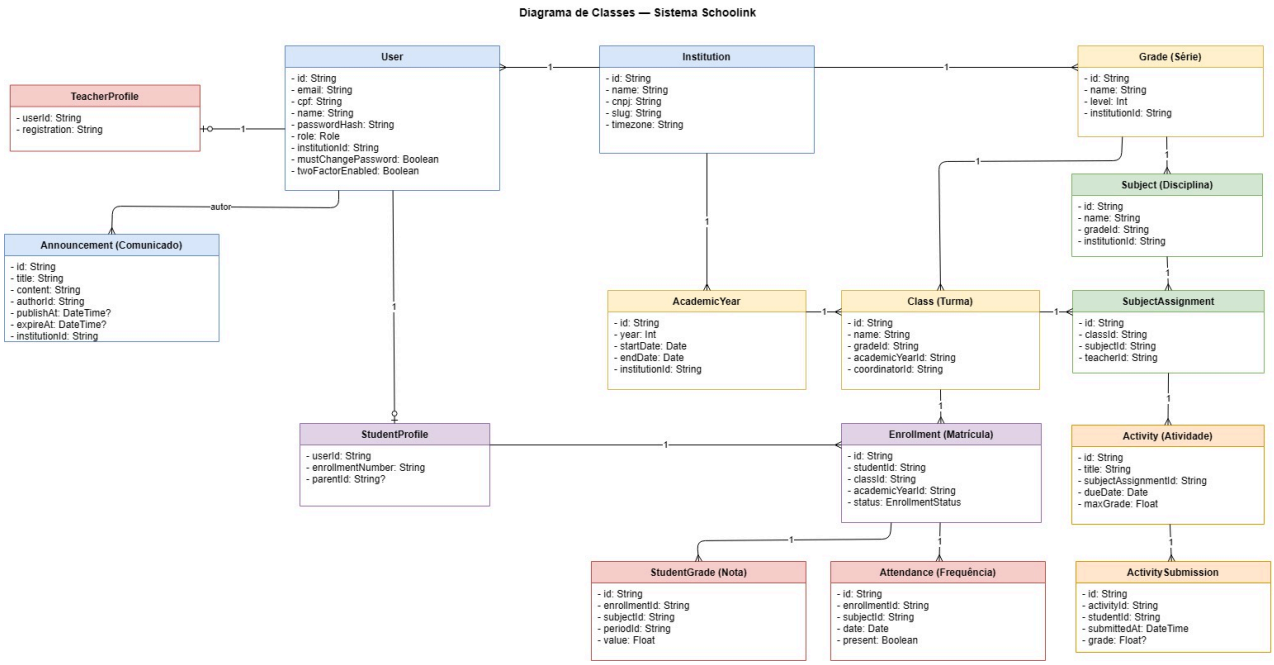


Figura 1 — Diagrama de Clasee do sistema Schoolink](Fonte: elaborado pelo autor)

4.2.2 Grupo 1: Acesso e Identidade

Instituição (Institution)

Representa uma escola ou instituição de ensino cadastrada na plataforma. É a entidade raiz do sistema — todos os demais dados pertencem a uma instituição específica, garantindo o isolamento completo entre diferentes escolas que utilizem o Schoolink.

Campos principais:

- Nome da instituição
- Logotipo (URL da imagem)
- Cores personalizadas (cor primária e cor secundária da interface)
- Sistema de avaliação (numérico, conceitual ou misto)
- Nota mínima para aprovação (padrão: 6,0)
- Frequência mínima para aprovação (padrão: 75%)
- Status de ativação (ativa ou inativa)

Perfil de Acesso (Role)

Define os papéis disponíveis na plataforma e as permissões associadas a cada um.

Os papéis existentes no Schoolink são: Diretor, Professor, Aluno, Responsável e Coordenador. Cada perfil determina quais funcionalidades o usuário poderá acessar, implementando o mecanismo de controle de acesso baseado em perfis (RBAC).

Campos principais:

- Nome do perfil (ex.: "student", "teacher", "director")
- Descrição em português (ex.: "Aluno", "Professor", "Diretor")
- Lista de permissões específicas (armazenada em formato JSON)

Usuário (User)

Representa qualquer pessoa com acesso ao sistema, independentemente do seu papel. É a entidade central de autenticação — armazena as credenciais de acesso e os dados pessoais do indivíduo. Cada usuário pertence a uma instituição e possui exatamente um perfil de acesso.

Campos principais:

- E-mail (identificador único de login)
- Senha (armazenada exclusivamente como hash criptográfico — nunca em texto simples)
- Nome completo
- CPF e data de nascimento (opcionais)
- Status da conta (ativa ou inativa)
- Número de tentativas de login malsucedidas (proteção contra força bruta)
- Data do último login
- Referência à instituição e ao perfil de acesso

4.2.3 Grupo 2: Perfis Específicos

Perfil do Aluno (StudentProfile)

Complementa o cadastro de um usuário com perfil de Aluno, armazenando informações acadêmicas específicas que não se aplicam aos demais usuários. É criado automaticamente pelo sistema no momento em que um novo aluno é cadastrado.

Campos principais:

- Número de matrícula (gerado automaticamente no formato ANOxxxx, ex.: 20260001)
- Data de matrícula
- Turma atual
- Informações médicas e de emergência (alergias, medicamentos, contato de emergência)
- Referências para responsáveis vinculados, histórico de notas, frequência e matrículas

Perfil do Professor (TeacherProfile)

Complementa o cadastro de um usuário com perfil de Professor, armazenando informações profissionais específicas. Também é criado automaticamente pelo sistema ao cadastrar um novo professor.

Campos principais:

- Número de registro profissional (gerado automaticamente no formato PROFxxx)
- Especialização (ex.: "Matemática", "Português")
- Departamento (ex.: "Exatas", "Humanas")
- Data de contratação
- Referências para as disciplinas e turmas às quais está atribuído

Perfil do Responsável (ParentProfile)

Complementa o cadastro de um usuário com perfil de Responsável (pai, mãe ou guardião). Após o cadastro, o Diretor pode vincular o Responsável aos alunos correspondentes, permitindo o acompanhamento acadêmico.

Campos principais:

- Profissão
- Local de trabalho
- Lista de alunos vinculados (relacionamento muitos-para-muitos com Perfil do Aluno)

4.2.4 Grupo 3: Estrutura Acadêmica

Ano Letivo (AcademicYear)

Representa o ano letivo de uma instituição (ex.: 2026). Define o período de funcionamento do ciclo educacional e serve como referência para todos os registros acadêmicos do sistema. Apenas um ano letivo pode estar ativo por vez.

Campos principais:

- Ano de referência (ex.: 2026)
- Data de início e data de encerramento
- Status (ativo ou inativo)
- Mínimo de dias letivos (padrão: 200, conforme a LDB)

Período Letivo (AcademicPeriod)

Representa as subdivisões do ano letivo, geralmente os bimestres ou semestres. O período letivo ativo é aquele em que os lançamentos de notas e frequência estão habilitados. Períodos encerrados preservam seus registros para consulta histórica.

Campos principais:

- Nome (ex.: "1º Bimestre", "2º Semestre")
- Número de ordem (1, 2, 3 ou 4)
- Data de início e data de encerramento
- Status (ativo, inativo ou encerrado)

Série Escolar (Grade)

Representa um nível de ensino dentro da estrutura curricular da escola (ex.: 7º Ano, 1º Ano do Ensino Médio). Cada série pertence a um ano letivo e a uma instituição. A série organiza as turmas e disciplinas que serão criadas abaixo dela.

Campos principais:

- Nome (ex.: "7º Ano", "1º Ano EM")
- Nível de ensino (fundamental ou médio)
- Ordem sequencial (para organização na interface)

Turma (Class)

Representa uma turma específica dentro de uma série escolar (ex.: 7º Ano A). É a unidade de agrupamento dos alunos. Todos os registros de chamada, notas e atividades são associados às turmas.

Campos principais:

- Nome identificador (ex.: "A", "B", "1")
- Sala de aula onde as aulas ocorrem
- Número máximo de alunos permitidos

Disciplina (Subject)

Representa uma matéria curricular dentro de uma série (ex.: Matemática do 7º Ano).Cada disciplina é tratada de forma independente por série: a Matemática do 7º Ano é uma entidade diferente da Matemática do 9º Ano, o que permite configurações de avaliação distintas para cada uma.

Campos principais:

- Nome (ex.: "Matemática", "Português")
- Código único de identificação (ex.: "MAT_7", "PORT_9")
- Carga horária semanal

Atribuição de Disciplina (SubjectAssignment)

Entidade intermediária que registra formalmente a relação entre um Professor, uma Disciplina e uma Turma. Essa atribuição é o que define quem pode lançar notas e registrar frequência em cada contexto. Sem ela, o professor não enxerga a turma nem a disciplina em seu painel.

Campos principais:

- Referência ao Professor
- Referência à Disciplina
- Referência à Turma

4.2.5 Grupo 4: Matrículas e Desempenho

Matrícula (Enrollment)

Registra formalmente a participação de um aluno em uma turma durante um ano letivo. É o vínculo que habilita o aluno a receber notas e ter sua frequência registrada. Cada aluno pode ter apenas uma matrícula ativa por ano letivo.

Campos principais:

- Referência ao Aluno (Perfil do Aluno)
- Referência à Turma
- Referência ao Ano Letivo
- Status da matrícula (ativa, transferida, cancelada, suspensa ou concluída)

Atividade Avaliativa (Activity)

Representa uma avaliação, prova, trabalho ou exercício criado pelo Professor para uma disciplina e turma específicas. As atividades ficam visíveis na agenda dos alunos e responsáveis, funcionando também como calendário de avaliações.

Campos principais:

- Título (ex.: "Prova de Frações")
- Tipo (EXAM — prova, ASSIGNMENT — trabalho, EXERCISE — exercício)
- Peso na composição da média
- Nota máxima possível
- Data de entrega ou realização

Nota do Aluno (StudentGrade)

Armazena o registro da nota obtida por um aluno em uma disciplina em um determinado período letivo. Notas podem ser lançadas de forma geral (nota do período) ou vinculadas a uma atividade avaliativa específica.

Campos principais:

- Valor numérico da nota
- Referência ao Aluno
- Referência à Disciplina
- Referência ao Período Letivo
- Referência ao Professor que realizou o lançamento
- Referência à Atividade Avaliativa (quando aplicável)

Frequência (Attendance)

Registra a presença ou ausência de um aluno em uma aula específica. Cada registro corresponde a um único aluno, em uma única disciplina, em uma única data. O conjunto desses registros permite calcular o percentual de presença do aluno.

Campos principais:

- Status (PRESENT — presente, ABSENT — ausente, JUSTIFIED — ausência justificada)
- Data da aula
- Referência ao Aluno
- Referência à Disciplina
- Referência à Turma
- Referência ao Professor responsável pela chamada

4.2.6 Grupo 5: Comunicação e Agenda

Comunicado (Announcement)

Representa um aviso ou comunicação oficial publicada pela instituição. Os comunicados podem ser de natureza acadêmica ou administrativa, e possuem níveis de prioridade que determinam como são exibidos na interface dos usuários.

Campos principais:

- Título e conteúdo do comunicado
- Tipo (acadêmico ou administrativo)
- Prioridade (normal ou alta)
- Data de publicação
- Referência ao usuário que publicou o comunicado
- Referência à instituição

Evento Escolar (SchoolEvent)

Armazena eventos do calendário escolar da instituição, como reuniões de pais, festividades, feriados e olimpíadas. Os eventos são exibidos no painel de todos os usuários da instituição.

Campos principais:

- Título do evento
- Data de realização
- Tipo do evento (MEETING — reunião, CULTURAL — cultural, HOLIDAY — feriado, OTHER — outros)
- Referência à instituição

4.3 FLUXO DE AUTENTICAÇÃO

A autenticação é o processo pelo qual o sistema verifica a identidade de um usuário antes de permitir o acesso à plataforma. No Schoolink, esse processo foi desenvolvido para ser seguro, eficiente e capaz de diferenciar automaticamente o tipo de acesso de cada usuário.

Em termos simples, o fluxo de autenticação funciona de forma semelhante à entrada em um prédio corporativo: o visitante se identifica na recepção (informa suas credenciais), a recepção verifica seus dados (o sistema valida email e senha) e, se tudo estiver correto, entrega um crachá de acesso com validade de 24 horas (o token JWT). A partir daí, o visitante usa o crachá para entrar em salas específicas — mas só naquelas para as quais o crachá tem autorização (controle RBAC por perfil).

O fluxo completo é descrito a seguir:

Etapa 1 — Abertura do sistema

O usuário acessa o endereço do Schoolink pelo navegador. A aplicação frontend é carregada e verifica se existe um token de autenticação previamente armazenado no dispositivo. Se existir e ainda for válido, o usuário é redirecionado diretamente ao seu painel, sem necessidade de novo login. Se não existir ou estiver expirado, a tela de login é exibida.

Etapa 2 — Envio das credenciais

O usuário preenche o formulário de login com seu endereço de e-mail e senha. Antes de enviar os dados ao servidor, o frontend realiza uma validação básica (campos não podem estar vazios, e-mail deve ter formato válido). Se a validação local passar, os dados são enviados ao servidor via requisição HTTP do tipo POST para o endpoint /api/auth/login, utilizando o protocolo seguro HTTPS para proteger os dados em trânsito.

Etapa 3 — Validação no servidor

O servidor recebe as credenciais e executa as seguintes verificações em sequência:

a) Busca o usuário no banco de dados pelo endereço de e-mail informado.
Se nenhum usuário for encontrado, retorna erro genérico.

b) Verifica se a conta do usuário está ativa (isActive = true).
Se a conta estiver desativada, o acesso é negado mesmo com senha correta.

c) Compara a senha informada com o hash criptográfico armazenado no banco de dados, utilizando o algoritmo bcrypt. Esse processo garante que, mesmo que o banco de dados seja comprometido, as senhas originais não possam ser recuperadas.

d) Verifica se o número de tentativas de login malsucedidas não ultrapassou o limite configurado (proteção contra ataques de força bruta). Em caso de qualquer falha nas verificações acima, o servidor retorna uma mensagem de erro genérica, sem indicar qual campo específico está incorreto — uma prática de segurança que impede que um atacante descubra quais e-mails existem no sistema.

Etapa 4 — Geração do token de autenticação

Após a validação bem-sucedida, o servidor gera um token JWT (JSON Web Token) assinado digitalmente. Esse token é uma sequência de caracteres que contém, de forma criptografada, as seguintes informações do usuário:

- Identificador único (ID) do usuário no banco de dados
- Nome do perfil de acesso (ex.: "student", "teacher", "director")
- Identificador da instituição à qual o usuário pertence
- Data e hora de expiração do token (24 horas a partir do login)

O token é assinado com uma chave secreta conhecida apenas pelo servidor. Qualquer tentativa de alterar o conteúdo do token invalida a assinatura e o token é rejeitado automaticamente nas requisições seguintes.

Etapa 5 — Recebimento e armazenamento do token

O servidor retorna o token JWT ao frontend junto com os dados básicos do usuário (nome, perfil, preferências). O frontend armazena o token de forma persistente utilizando o Redux Persist, o que permite que o usuário permaneça autenticado mesmo após fechar e reabrir o navegador. O token é então utilizado em todas as requisições subsequentes, sendo incluído no cabeçalho Authorization de cada chamada à API.

Etapa 6 — Redirecionamento por perfil

Com base no perfil de acesso contido no token, o frontend redireciona o usuário automaticamente para o painel correspondente:

- Diretor → Painel administrativo da instituição
- Professor → Painel com turmas e disciplinas atribuídas
- Aluno → Painel acadêmico pessoal
- Responsável → Painel de acompanhamento dos filhos vinculados

Etapa 7 — Controle de acesso em requisições subsequentes

A partir do login, todas as requisições realizadas pelo usuário ao servidor incluem o token JWT no cabeçalho HTTP. O servidor, antes de processar qualquer requisição, executa um middleware (componente intermediário) de autenticação e autorização que:

a) Valida a assinatura digital do token para garantir que ele não foi adulterado.

b) Verifica se o token ainda está dentro do prazo de validade de 24 horas.

c) Extrai o perfil e o identificador de instituição do token para identificar quem está fazendo a requisição.

d) Verifica se o perfil do usuário tem permissão para executar aquela operação específica (controle RBAC).

Se qualquer uma dessas verificações falhar, o servidor retorna:

- Código 401 (não autenticado): token inválido ou expirado
- Código 403 (não autorizado): perfil sem permissão para a operação

O frontend interpreta esses códigos e redireciona o usuário para a tela de login quando necessário.

Etapa 8 — Encerramento da sessão

Quando o usuário aciona o botão de saída (logout), o frontend remove o token armazenado localmente e redireciona o usuário para a tela de login. A partir desse momento, o dispositivo não possui mais o token e nenhuma requisição autenticada pode ser realizada até que um novo login seja efetuado.

5 FUNDAMENTAÇÃO TECNOLÓGICA DE SCHOOLINK

5.1. JavaScript e TypeScript

5.1.1 História e Contexto

O JavaScript nasceu em 1995, criado por Brendan Eich em apenas dez dias enquanto trabalhava na empresa Netscape Communications. O objetivo inicial era simples: adicionar pequenas interações às páginas da internet, como validar se um campo de formulário estava preenchido antes de enviá-lo ao servidor. Naquela época, a linguagem era chamada de "LiveScript", mas foi renomeada para JavaScript como estratégia de marketing, aproveitando a popularidade da linguagem Java — apesar das duas não terem qualquer relação técnica significativa (FLANAGAN, 2020).

Com o passar dos anos, o JavaScript cresceu enormemente. Em 2009, Ryan Dahl criou o Node.js, que permitiu que o JavaScript saísse do navegador e passasse a ser executado também em servidores. Em 2015, a organização ECMA International publicou o ECMAScript 6 (ES6), uma atualização que modernizou profundamente a linguagem, introduzindo recursos como funções de seta (=>), módulos, desestruturação de objetos e muito mais (ECMA INTERNATIONAL, 2015).

O TypeScript surgiu em 2012, criado pela Microsoft. Ele é, essencialmente, o JavaScript com uma camada adicional: a tipagem estática. Isso significa que o desenvolvedor declara explicitamente o tipo de dado que cada variável, parâmetro ou retorno de função deve conter — se um texto, um número, um objeto, etc. O compilador do TypeScript verifica essas declarações antes de o programa ser executado, apontando erros com antecedência (MICROSOFT, 2012).

5.1.2 Por que Escolhemos TypeScript

O sistema Schoolink lida com múltiplos perfis de usuário (aluno, responsável, professor, diretor), regras de negócio complexas e dados sensíveis. Em projetos dessa natureza, um erro de tipo — como tentar acessar uma propriedade em um valor que pode ser nulo — pode causar falhas silenciosas e difíceis de rastrear.

O TypeScript foi escolhido por oferecer segurança em tempo de desenvolvimento, autocompletar inteligente nas ferramentas de edição e documentação embutida no próprio código.

Comparado ao JavaScript puro, ele exige mais disciplina no início, mas reduz significativamente a quantidade de erros em produção.

Outras linguagens como Python ou Java também oferecem tipagem, porém o ecossistema JavaScript/TypeScript é o mais adequado para sistemas web modernos, permitindo compartilhar lógica e tipos entre o servidor e o cliente (FREEMAN; ROBSON, 2020).

5.1.3 Exemplo no Sistema

No Schoolink, todos os dados trocados entre o frontend e o backend são descritos como interfaces TypeScript. O trecho abaixo define a estrutura de uma atividade escolar:

```
// frontend/src/services/academic/academicService.ts
interface Activity {
  id: string;
  title: string;
  description: string | null;
  dueDate: string | null;
  subjectId: string;
  classId: string | null;
  class: { id: string; name: string; grade: { name: string } } | null;
}
```

Nesse exemplo, cada linha descreve um campo da atividade e o tipo de dado que ele pode conter. O campo title, por exemplo, é declarado como string, o que significa que ele sempre será um texto. Já dueDate é declarado como string | null, indicando que a atividade pode ou não ter uma data de entrega — o símbolo | funciona como um "ou". Essa declaração força o programador a tratar ambos os casos no código: se tentar exibir a data diretamente sem verificar se ela existe, o TypeScript aponta o erro antes mesmo de o sistema ser executado, evitando que o usuário veja uma tela com erro. O campo class segue o mesmo princípio: como uma atividade pode ser geral (sem turma específica), ele também pode ser nulo.

5.2. React

5.2.1 História e Contexto

O React é uma biblioteca JavaScript criada pelo engenheiro Jordan Walke, da empresa Meta (anteriormente Facebook), e foi lançada publicamente em 2013. Ela surgiu da necessidade interna do Facebook de atualizar partes da interface sem recarregar a página inteira — algo que na época era difícil de fazer de forma eficiente e organizada (BANKS; PORCELLO, 2020).

A grande inovação do React foi o conceito de componentes: pedaços reutilizáveis de interface que encapsulam sua própria lógica e aparência. Em vez de construir uma página como um bloco único, o desenvolvedor constrói pequenas peças — um botão, um card, um formulário — e as combina para formar telas completas. Outra inovação foi o DOM Virtual: em vez de atualizar diretamente o HTML da página a cada mudança, o React calcula a diferença entre o estado anterior e o novo estado e aplica apenas às alterações necessárias, tornando a interface muito mais rápida (CHINNATHAMBI, 2018).

Com a introdução dos Hooks no React 16.8, em 2019, a forma de escrever componentes foi simplificada: funções simples passaram a poder gerenciar estado e ciclo de vida, antes exclusividade de classes complexas (REACT, 2019).

5.2.2 Por que Escolhemos React

O Schoolink possui múltiplas telas — dashboard do aluno, dashboard do professor, página de atividades, gerenciamento de turmas — que compartilham componentes como tabelas, botões, formulários e cards. O React permite construir esses componentes uma vez e reutilizá-los em qualquer parte do sistema.

Entre as alternativas existentes, como Angular e Vue.js, o React se destaca pelo tamanho e maturidade de seu ecossistema, pela adoção em larga escala pela indústria e pela integração natural com o Next.js, que utilizamos para o roteamento e a renderização do sistema.

5.2.3 Exemplo no Sistema

O componente abaixo, utilizado no dashboard do aluno, receberá os dados de uma atividade e os exibe em um card:

```
// frontend/src/components/features/dashboard/components/sections/StudentSection.tsx
interface ActivityCardProps {
  title: string;
  subject: string;
  dueDate: string | null;
}

function ActivityCard({ title, subject, dueDate }: ActivityCardProps) {
  const prazo = dueDate
    ? new Date(dueDate).toLocaleDateString("pt-BR")
    : "Sem prazo";

  return (
    <div className="p-4 border rounded-lg">
      <p className="font-semibold">{title}</p>
      <p className="text-sm text-muted-foreground">{subject}</p>
      <p className="text-xs">Prazo: {prazo}</p>
    </div>
  );
}
```

O componente ActivityCard funciona como um molde reutilizável: sempre que o sistema precisar exibir uma atividade, basta "chamar" esse componente passando os dados correspondentes. A interface ActivityCardProps define quais informações o componente precisa receber — o título, a disciplina e a data de entrega. Dentro do componente, a variável prazo verifica se a data de entrega existe: caso exista, ela é convertida para o formato brasileiro (dia/mês/ano) usando o método toLocaleDateString("pt-BR"); caso contrário, exibe o texto "Sem prazo". O bloco de retorno (return) descreve o que será exibido na tela: um card com bordas arredondadas contendo três linhas de texto — título em negrito, nome da disciplina em cinza e o prazo em tamanho menor.

5.3. Next.js

5.3.1 História e Contexto

O Next.js foi criado pela empresa Vercel e lançado em 2016. Ele é um framework construído sobre o React que adiciona funcionalidades essenciais para aplicações web de produção: roteamento baseado em arquivos, renderização no servidor (SSR) e geração de páginas estáticas (SSG), entre outras (VERCEL, 2016).

Para entender a importância do Next.js, é preciso compreender um problema do React puro: por padrão, o React gera as páginas inteiramente no navegador do usuário. Isso significa que, ao acessar o sistema, o usuário recebe um arquivo HTML praticamente vazio, e só então o JavaScript baixa os dados e constrói a interface. Esse processo pode ser lento e prejudica a indexação por mecanismos de busca. O Next.js resolve isso ao permitir que as páginas sejam processadas no servidor antes de serem enviadas ao navegador, chegando já prontas para exibição (WIERUCH, 2022).

O Next.js introduziu o conceito de App Router na versão 13, em 2022, que organiza as rotas da aplicação em pastas dentro do diretório app/. Cada pasta que contém um arquivo page.tsx automaticamente se torna uma rota acessível no navegador.

5.3.2 Por que escolhemos Next.js

O Schoolink é um sistema que exige proteção de rotas (apenas usuários autenticados podem acessar o dashboard), múltiplas páginas para diferentes perfis e uma experiência de navegação fluida. O Next.js oferece tudo isso de forma integrada, sem necessidade de configurações externas complexas.

Alternativas como o Vite com React Router exigem mais configuração manual para funcionalidades como renderização no servidor e proteção de rotas. O Create React App (CRA), outra alternativa popular, foi descontinuado em 2023 e não oferece SSR nativamente.

5.3.3 Exemplo no Sistema

A organização de pastas do Schoolink demonstra como o Next.js transforma a estrutura de diretórios em rotas de navegação:

```
frontend/src/app/
├── dashboard/
│   ├── page.tsx            → /dashboard
│   ├── activities/
│   │   └── page.tsx        → /dashboard/activities
│   └── academic/
│       └── activities/
│           └── page.tsx    → /dashboard/academic/activities
```

Nessa estrutura, cada pasta representa um segmento da URL. O arquivo page.tsx dentro de dashboard/ corresponde ao endereço /dashboard; o arquivo dentro de activities/ corresponde a /dashboard/activities, e assim por diante. Não é necessário configurar rotas manualmente — o Next.js as detecta automaticamente pela organização das pastas. Isso torna a navegação do sistema previsível: olhando para a estrutura de arquivos, qualquer desenvolvedor consegue identificar quais páginas existem e quais URLs elas respondem.

Além da organização de rotas, o Next.js também permite proteger páginas inteiras no servidor:

```
// frontend/src/app/dashboard/layout.tsx
import { redirect } from "next/navigation";
import { getServerSession } from "next-auth";

export default async function DashboardLayout({ children }) {
  const session = await getServerSession();
  if (!session) redirect("/login");
  return <>{children}</>;
}
```

Esse arquivo é o "guardião" de todas as páginas dentro do diretório dashboard/. Antes de qualquer página do dashboard ser exibida, esse código é executado no servidor: ele verifica se o usuário possui uma sessão ativa (ou seja, se está logado). Caso não esteja, a função redirect("/login") o redireciona automaticamente para a tela de login, sem que a página do dashboard chegue sequer a ser carregada. Se o usuário estiver autenticado, o children — que representa a página solicitada — é exibido normalmente.

5.4. Tailwind CSS e Shadcn/UI

5.4.1 História e Contexto

O CSS (Folhas de Estilo em Cascata) é a linguagem responsável pela aparência visual das páginas web — cores, tamanhos, espaçamentos, fontes. Criada em 1996 por Håkon Wium Lie, o CSS evoluiu muito ao longo das décadas, mas a forma de organizá-lo em projetos grandes sempre foi um desafio (LIE; BOS, 1999).

O Tailwind CSS surgiu em 2017, criado por Adam Wathan, com uma abordagem diferente das bibliotecas de estilo tradicionais: em vez de escrever arquivos CSS separados, o desenvolvedor aplica classes de utilitário diretamente no HTML/JSX. Por exemplo, para deixar um texto vermelho e em negrito, basta escrever className="text-red-500 font-bold" (WATHAN, 2019).

O Shadcn/UI surgiu em 2023, criado por shadcn (pseudônimo), como uma coleção de componentes de interface construídos sobre o Radix UI — uma biblioteca de componentes acessíveis e sem estilo — e estilizados com Tailwind CSS. A principal diferença do Shadcn/UI em relação a bibliotecas como Bootstrap ou Material UI é que os componentes são copiados diretamente para o projeto, não instalados como dependência opaca. Isso dá total controle sobre o código (SHADCN, 2023).

5.4.2 Por que Escolhemos Tailwind CSS e Shadcn/UI

O Schoolink precisava de uma interface consistente, moderna e acessível para usuários com diferentes perfis. O Tailwind CSS permite criar layouts responsivos rapidamente sem sair do arquivo do componente. O Shadcn/UI fornece componentes prontos — tabelas, diálogos, formulários, dropdowns — que seguem padrões de acessibilidade e podem ser personalizados livremente.

Alternativas como Bootstrap têm estética genérica e sobrescrita difícil. Material UI impõe o design do Google, que pode conflitar com a identidade visual do sistema. O Shadcn/UI é agnóstico: fornece a estrutura, e o visual é totalmente customizável.

5.4.3 Exemplo no Sistema

O formulário de criação de atividades utiliza componentes do Shadcn/UI para a seleção de disciplina:

```
// frontend/src/app/dashboard/academic/activities/page.tsx
import { Select, SelectContent, SelectItem, SelectTrigger } from "@/components/ui/select";

<Select onValueChange={handleSubjectChange}>
  <SelectTrigger className="w-full">
    <span>Selecione a disciplina</span>
  </SelectTrigger>
  <SelectContent>
    {subjects.map((s) => (
      <SelectItem key={s.id} value={s.id}>
        {s.name}
      </SelectItem>
    ))}
  </SelectContent>
</Select>
```

Nesse trecho, o componente Select do Shadcn/UI cria uma lista suspensa de seleção. O SelectTrigger é o botão visível na tela que, ao ser clicado, abre a lista. O SelectContent contém as opções disponíveis, que são geradas dinamicamente: o método .map() percorre todas as disciplinas disponíveis (subjects) e, para cada uma, cria um SelectItem exibindo o nome da disciplina. Quando o professor seleciona uma opção, a função handleSubjectChange é chamada automaticamente com o identificador único da disciplina escolhida — permitindo que o restante do formulário, como a lista de turmas, seja atualizado em seguida. As classes do Tailwind, como w-full, garantem que o componente ocupe toda a largura disponível sem necessidade de escrever CSS separado.

5.5. Redux Toolkit e RTK Query

5.5.1 História e Contexto

Conforme as aplicações React crescem, gerenciar o estado compartilhado entre componentes — como os dados do usuário logado, a lista de turmas, ou o status de carregamento de uma requisição — se torna complexo. O Redux surgiu em 2015, criado por Dan Abramov e Andrew Clark, como uma solução centralizada: todos os dados da aplicação ficam em um único "armazém" (store), e qualquer componente pode lê-los ou modificá-los de forma previsível (ABRAMOV; CLARK, 2015).

O problema do Redux original era sua verbosidade: para cada ação simples, o desenvolvedor precisava escrever múltiplos arquivos. O Redux Toolkit (RTK), lançado oficialmente em 2019 pela equipe do Redux, simplificou radicalmente esse processo, reduzindo o código necessário em até 60% (REDUX TOOLKIT, 2019).

O RTK Query, introduzido no Redux Toolkit 1.6 em 2021, vai além: ele automatiza o ciclo completo de requisições à API — disparo, estado de carregamento, armazenamento em cache e revalidação — com mínima configuração (REDUX TOOLKIT, 2021).

5.5.2 Por que Escolhemos Redux Toolkit

O Schoolink precisa compartilhar dados entre muitos componentes: o perfil do usuário logado é necessário na barra lateral, no cabeçalho, nas páginas de dashboard e na lógica de permissões. Sem um gerenciador de estado global, esses dados precisariam ser passados manualmente de componente em componente — o que é impraticável.

Alternativas como Context API (nativa do React) funcionam para casos simples, mas degradam em performance quando muitos componentes se atualizam simultaneamente. O Zustand é mais simples que o Redux, mas menos estruturado para projetos grandes. O Redux Toolkit oferece o equilíbrio ideal entre estrutura, performance e produtividade.

5.5.3 Exemplo no Sistema

O serviço de dados acadêmicos utiliza RTK Query para buscar atividades e registrar conclusões:

```
// frontend/src/services/academic/academicService.ts
import { createApi, fetchBaseQuery } from "@reduxjs/toolkit/query/react";

export const academicApi = createApi({
  reducerPath: "academicApi",
  baseQuery: fetchBaseQuery({ baseUrl: "/api/academic" }),
  endpoints: (builder) => ({
    listMyActivities: builder.query<Activity[], string | undefined>({
      query: (childId) =>
        childId
          ? `/activities/mine?childId=${childId}`
```

```
        : "/activities/mine",
    }),
    toggleActivityCompletion: builder.mutation<void, string>({
      query: (id) => ({ url: `/activities/${id}/toggle`, method: "PATCH" }),
    }),
  }),
});

export const { useListMyActivitiesQuery, useToggleActivityCompletionMutation } =
  academicApi;
```

Esse bloco de código configura toda a comunicação do frontend com a parte acadêmica do servidor. A função createApi cria um "serviço" que gerencia automaticamente o ciclo de vida das requisições. O baseQuery define o endereço base do servidor — todas as chamadas partirão de /api/academic. Dentro de endpoints, são definidas duas operações: listMyActivities é uma consulta (query) que busca as atividades do aluno logado; caso seja um responsável buscando atividades de um filho específico, o identificador do filho (childId) é adicionado à URL. Já toggleActivityCompletion é uma mutação (mutation) — uma operação que altera dados no servidor — que envia uma requisição do tipo PATCH para marcar uma atividade como concluída ou não. A última linha exporta dois "hooks" gerados automaticamente pelo RTK Query: useListMyActivitiesQuery e useToggleActivityCompletionMutation, que podem ser usados diretamente em qualquer componente React para buscar dados e disparar ações sem precisar escrever a lógica de requisição manualmente.

5.6. Node.js

5.6.1 História e Contexto

Antes do Node.js, o JavaScript vivia confinado ao navegador. Em 2009, Ryan Dahl apresentou o Node.js na conferência JSConf Europe, mostrando algo revolucionário: o motor V8 do Google Chrome — que interpreta e executa JavaScript — rodando fora do navegador, diretamente no sistema operacional (DAHL, 2009).

O Node.js trouxe uma característica técnica fundamental: o modelo de I/O não bloqueante. Em servidores tradicionais, quando uma requisição precisava buscar dados no banco de dados, o servidor ficava parado esperando a resposta antes de atender outra requisição. O Node.js funciona

diferente: ele registra a operação e continua atendendo outras requisições enquanto aguarda, usando um mecanismo chamado event loop (loop de eventos). Isso o torna altamente eficiente para sistemas com muitas requisições simultâneas (TILKOV; VINOSKI, 2010).

O npm (Node Package Manager), gerenciador de pacotes que acompanha o Node.js, se tornou o maior repositório de bibliotecas de software do mundo, com mais de dois milhões de pacotes disponíveis.

5.6.2 Por que Escolhemos Node.js

Utilizar Node.js no backend do Schoolink permite que toda a equipe trabalhe em uma única linguagem — TypeScript — tanto no frontend quanto no servidor. Isso elimina a barreira de contexto mental de alternar entre linguagens diferentes e permite compartilhar definições de tipos e lógica de validação.

Alternativas como Python (com Django ou FastAPI) e Java (com Spring Boot) são escolhas sólidas para backends, mas exigem que o desenvolvedor domine uma linguagem diferente da usada no frontend. Para um projeto acadêmico com equipe reduzida, a uniformidade do JavaScript/TypeScript full-stack é uma vantagem significativa.

5.6.3 Exemplo no Sistema

O ponto de entrada do servidor Schoolink configura e inicia o Node.js com o Express:

```
// backend/src/index.ts
import express from "express";
import { academicRoutes } from "./routes/academic.routes";

const app = express();
const PORT = process.env.PORT ?? 3001;

app.use(express.json());
app.use("/api/academic", academicRoutes);

app.listen(PORT, () => {
```

```
  console.log(`Servidor rodando na porta ${PORT}`);
});
```

Esse arquivo é o ponto de partida do servidor. A primeira instrução cria uma aplicação Express — o framework que gerenciará todas as requisições. Em seguida, process.env.PORT ?? 3001 lê a porta configurada no ambiente do servidor; caso nenhuma seja definida, usa a porta 3001 como padrão. A linha app.use(express.json()) instrui o servidor a interpretar automaticamente o corpo das requisições como JSON — sem isso, os dados enviados pelo frontend chegariam como texto bruto ilegível. A linha app.use("/api/academic", academicRoutes) registra todas as rotas acadêmicas sob o prefixo /api/academic, organizando a API em grupos temáticos. Por fim, app.listen(PORT, ...) inicia o servidor, fazendo com que ele fique aguardando requisições na porta definida e exibe uma mensagem no terminal confirmando que está em funcionamento.

5.7. Express.js

5.7.1 História e Contexto

O Express.js foi criado por TJ Holowaychuk e lançado em 2010, tornando-se rapidamente o framework web mais popular do ecossistema Node.js. Sua filosofia é ser minimalista e não opinativo: ele oferece as ferramentas essenciais para criar um servidor HTTP — roteamento de URLs, tratamento de requisições e respostas, e suporte a middlewares — sem impor uma estrutura rígida ao projeto (HOLOWAYCHUK, 2010).

Um middleware no Express é uma função que fica no caminho entre o recebimento de uma requisição e o envio da resposta. Middlewares são usados para tarefas como verificar se o usuário está autenticado, registrar logs de acesso, converter o corpo da requisição de JSON para objeto JavaScript, e muito mais. Eles se encadeiam em sequência, e cada um pode passar o controle para o próximo ou interromper o fluxo (BROWN, 2019).

5.7.2 Por que escolhemos Express.js

O Schoolink precisava de um servidor HTTP capaz de responder a requisições do frontend, validar autenticação, processar arquivos enviados por alunos e consultar o banco de dados. O Express oferece exatamente isso com mínima configuração.

Alternativas como Fastify e Koa são mais modernas e performáticas em benchmarks, mas o Express possui documentação vastíssima e é o framework que mais conta com exemplos, tutoriais e soluções de problemas disponíveis. Para um projeto acadêmico, essa maturidade e suporte da comunidade pesa positivamente.

O NestJS, outra alternativa popular, é altamente estruturado (inspirado no Angular), o que adiciona complexidade desnecessária para um projeto do escopo do Schoolink.

5.7.3 Exemplo no Sistema

A rota de atividades do Schoolink usa três camadas encadeadas para processar cada requisição:

```
// backend/src/routes/academic.routes.ts
import { Router } from "express";
import { authenticate } from "../middlewares/authenticate";
import { checkRole } from "../middlewares/checkRole";
import { AcademicController } from "../controllers/academic.controller";

const router = Router();

router.use(authenticate); // verifica JWT em todas as rotas

router.get(
  "/activities/mine",
  checkRole(["student", "parent"]),
  AcademicController.listMyActivities
);

router.get("/activities", AcademicController.listActivities);

export { router as academicRoutes };
```

Nesse arquivo, o Router do Express organiza as rotas relacionadas à área acadêmica. A instrução router.use(authenticate) aplica o middleware de autenticação a todas as rotas desse grupo: antes de qualquer requisição

ser processada, o sistema verifica se o usuário está logado. Para a rota /activities/mine, há uma segunda verificação encadeada: o middleware checkRole(["student", "parent"]) garante que apenas alunos e responsáveis possam acessá-la — se um professor tentar acessar esse endereço, receberá uma resposta de acesso negado. Somente após passar pelas duas verificações, a função AcademicController.listMyActivities é executada, buscando e retornando as atividades. É importante notar que /activities/mine está declarada antes de /activities — isso é necessário porque o Express processa as rotas na ordem em que foram registradas; se fosse o contrário, o sistema interpretaria a palavra "mine" como um identificador dinâmico da rota /activities/:id, causando um comportamento incorreto.

5.8. PostgreSQL

5.8.1 História e Contexto

O PostgreSQL tem uma das histórias mais longas entre os bancos de dados modernos. Seu antecessor, o INGRES, foi desenvolvido na Universidade da Califórnia em Berkeley nos anos 1970 pelo professor Michael Stonebraker. Em 1986, o projeto evoluiu para o POSTGRES, e em 1996 recebeu suporte à linguagem SQL e passou a se chamar PostgreSQL. Nesse mesmo ano, tornou-se um projeto de código aberto com uma comunidade ativa que continua desenvolvendo-o até hoje (STONEBRAKER; ROWE, 1986).

O PostgreSQL é um SGBD relacional — um Sistema de Gerenciamento de Banco de Dados que organiza as informações em tabelas com linhas e colunas, e utiliza a linguagem SQL para criar, consultar, atualizar e deletar dados. Ele é conhecido por seguir rigorosamente as propriedades ACID (Atomicidade, Consistência, Isolamento e Durabilidade), que garantem que as operações no banco de dados sejam confiáveis mesmo em caso de falhas (DATE, 2004).

5.8.2 Por que Escolhemos PostgreSQL

O Schoolink gerencia dados interconectados: alunos pertencem a turmas, turmas têm disciplinas, disciplinas têm professores, atividades pertencem a disciplinas e podem ter submissões de alunos. Esse tipo de dado relacional é exatamente o ponto forte dos bancos relacionais como o PostgreSQL.

Alternativas não relacionais, como o MongoDB, são excelentes para dados flexíveis e sem estrutura definida, mas prejudicam a integridade em sistemas com relações complexas entre entidades. O MySQL é uma alternativa relacional sólida, mas o PostgreSQL oferece recursos

avançados adicionais, como suporte a tipos de dados complexos (JSON, arrays), transações mais robustas e melhor conformidade com o padrão SQL.

5.8.3 Exemplo no Sistema

A tabela Activity do banco de dados do Schoolink armazena as atividades escolares e mantém vínculos com outras tabelas:

```
-- Gerado pelo Prisma a partir do schema
CREATE TABLE "Activity" (
  "id"          TEXT NOT NULL PRIMARY KEY,
  "title"       TEXT NOT NULL,
  "description" TEXT,
  "dueDate"     TIMESTAMP,
  "subjectId"   TEXT NOT NULL REFERENCES "Subject"("id"),
  "classId"     TEXT REFERENCES "Class"("id"),
  "createdById" TEXT NOT NULL REFERENCES "User"("id"),
  "createdAt"   TIMESTAMP NOT NULL DEFAULT NOW()
);
```

Essa instrução SQL cria a estrutura que armazenará todas as atividades no banco de dados. Cada coluna representa um atributo da atividade: id é o identificador único de cada registro; title é o nome da atividade e é obrigatório (NOT NULL); description e dueDate são opcionais — uma atividade pode ser criada sem descrição ou sem prazo. As colunas subjectId, classId e createdById são chaves estrangeiras: a palavra REFERENCES indica que esses campos apontam para registros em outras tabelas, garantindo integridade — por exemplo, não é possível cadastrar uma atividade referenciando uma disciplina que não existe. O campo classId é opcional (sem NOT NULL), pois uma atividade pode ser geral, sem vínculo com uma turma específica. Por fim, createdAt registra automaticamente a data e hora de criação usando DEFAULT NOW(), sem que o programador precise informar esse valor manualmente.

5.9. Prisma ORM

5.9.1 História e Contexto

Trabalhar diretamente com SQL em uma aplicação JavaScript/TypeScript pode ser tedioso e propenso a erros: o desenvolvedor precisa escrever strings de texto contendo comandos SQL,

converter manualmente os resultados em objetos JavaScript e garantir que os tipos estejam corretos. Os ORMs (Mapeamentos Objeto-Relacionais) surgiram para resolver esse problema, permitindo que o desenvolvedor interaja com o banco de dados usando a própria linguagem da aplicação, sem escrever SQL diretamente.

O Prisma foi lançado em 2019 pela empresa Prisma Data, Inc., e rapidamente se tornou o ORM mais popular do ecossistema Node.js/TypeScript. Sua abordagem é diferenciada: o desenvolvedor define o modelo de dados em um arquivo schema.prisma — uma linguagem declarativa própria — e o Prisma gera automaticamente o cliente TypeScript com tipos precisos para cada consulta ao banco de dados (PRISMA, 2019).

5.9.2 Por que escolhemos Prisma

O Prisma oferece três vantagens fundamentais para o Schoolink:

1. Segurança de tipos: cada consulta ao banco retorna um tipo TypeScript preciso, eliminando erros de acesso a campos inexistentes.

2. Migrações automáticas: o Prisma gera e aplica as alterações no banco de dados automaticamente quando o schema.prisma é modificado.

3. Legibilidade: as consultas são escritas como chamadas de função JavaScript, muito mais legíveis que SQL embutido em strings.

Alternativas como TypeORM e Sequelize também são ORMs para TypeScript/JavaScript, mas possuem APIs mais verbosas e são menos precisos nos tipos gerados. O Drizzle ORM é uma alternativa moderna mais próxima do SQL puro, mas com comunidade ainda menor.

5.9.3 Exemplo no Sistema

A consulta abaixo busca todas as atividades de um aluno, aplicando múltiplos filtros simultaneamente:

```
// backend/src/controllers/academic.controller.ts
const activities = await prisma.activity.findMany({
  where: {
    subjectId: { in: subjectIds },
    AND: [
```

```
      {
        OR: [
          { dueDate: { gte: new Date() } },
          { dueDate: null },
        ],
      },
      {
        OR: [
          { classId: student.currentClassId },
          { classId: null },
        ],
      },
    ],
  },
  include: {
    subject: { select: { name: true } },
    class: { select: { name: true } },
  },
  orderBy: { dueDate: "asc" },
});
```

Esse bloco realiza uma consulta sofisticada ao banco de dados sem escrever uma linha de SQL. prisma.activity.findMany busca múltiplos registros da tabela de atividades aplicando os critérios definidos em where. O primeiro filtro, subjectId: { in: subjectIds }, garante que somente atividades das disciplinas do aluno sejam retornadas. O bloco AND exige que duas condições sejam verdadeiras ao mesmo tempo: a primeira delas, usando OR, aceita atividades cujo prazo ainda não venceu (gte: new Date() significa "maior ou igual à data atual") ou atividades sem prazo definido (dueDate: null); a segunda, também com OR, aceita atividades associadas especificamente à turma do aluno ou atividades sem turma definida (que se aplicam a todos). O bloco include instrui o Prisma a trazer junto os dados relacionados — o nome da disciplina e o nome da turma — sem necessidade de uma segunda consulta separada. Por fim, orderBy: { dueDate: "asc" } ordena os resultados do prazo mais próximo para o mais distante, facilitando a visualização das tarefas mais urgentes.

5.10. JWT — JSON Web Token

5.10.1 História e Contexto

Em sistemas web, uma vez que o usuário faz login, o servidor precisa "lembrar" que aquela pessoa está autenticada nas próximas requisições. A abordagem tradicional usa sessões: o servidor armazena os dados do usuário em memória ou banco de dados e envia ao navegador um identificador de sessão (cookie). Em cada requisição, o navegador envia esse identificador e o servidor consulta o armazenamento para identificar o usuário.

O JWT (JSON Web Token) surgiu como uma alternativa sem estado (stateless). Em vez de armazenar sessões no servidor, toda a informação necessária é codificada em um token compacto que fica ao lado do cliente. O servidor não precisa armazenar nada — basta verificar a assinatura criptográfica do token para confirmar sua autenticidade. O padrão foi formalizado pelo IETF na RFC 7519, em 2015 (JONES; BRADLEY; SAKIMURA, 2015).

Um JWT é composto por três partes separadas por pontos: o cabeçalho (algoritmo de assinatura), o payload (dados do usuário, como ID e papel/role) e a assinatura (garantia criptográfica de que o token não foi adulterado).

5.10.2 Por que escolhemos JWT

O Schoolink possui múltiplos perfis de usuário com permissões distintas: alunos acessam apenas seus próprios dados, professores acessam suas turmas, e diretores têm acesso total. O JWT permite embutir essas informações no próprio token, evitando consultas ao banco de dados a cada requisição apenas para verificar quem é o usuário.

A alternativa seria sessões tradicionais com armazenamento em Redis ou banco de dados — o que adiciona complexidade de infraestrutura. Para o escopo do Schoolink, o JWT oferece a simplicidade adequada sem comprometer a segurança quando utilizado corretamente (com expiração e transmissão apenas via HTTPS).

5.10.3 Exemplo no Sistema

O sistema utiliza o JWT em dois momentos distintos: ao fazer login e ao verificar cada requisição subsequente. No momento do login, o servidor gera o token com os dados do usuário:

```
// backend/src/controllers/auth.controller.ts
const token = jwt.sign(
  { id: user.id, email: user.email, role: { id: user.role } },
  process.env.JWT_SECRET!,
  { expiresIn: "8h" }
);
```

A função jwt.sign cria o token em três etapas: primeiro, define o conteúdo que será armazenado dentro do token — nesse caso, o identificador do usuário, seu e-mail e seu papel no sistema (aluno, professor, diretor etc.); segundo, usa uma chave secreta (JWT_SECRET) armazenada nas configurações do servidor para assinar criptograficamente o token, impedindo que qualquer pessoa o falsifique; terceiro, define que o token expira em 8 horas, obrigando o usuário a fazer login novamente após esse período. O token gerado é uma string de texto que o frontend armazena e envia em todas as requisições subsequentes.

Para verificar esse token a cada requisição, o middleware de autenticação realiza o processo inverso:

```
// backend/src/middlewares/authenticate.ts
import jwt from "jsonwebtoken";
import { Request, Response, NextFunction } from "express";

export function authenticate(req: Request, res: Response, next: NextFunction) {
  const authHeader = req.headers.authorization;
  if (!authHeader?.startsWith("Bearer ")) {
    return res.status(401).json({ message: "Token não fornecido." });
  }

  const token = authHeader.split(" ")[1];

  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET!);
```

```
    req.user = decoded as TokenPayload;
    next();
  } catch {
    return res.status(401).json({ message: "Token inválido ou expirado." });
  }
}
```

Esse middleware é executado antes de qualquer rota protegida do sistema. Ele começa verificando o cabeçalho Authorization da requisição, que deve conter o token no formato Bearer <token>. Caso o cabeçalho esteja ausente ou mal formatado, a requisição é imediatamente bloqueada com o código de resposta 401, que indica "não autorizado". Se o cabeçalho estiver presente, o token é extraído (a instrução .split(" ")[1] pega a parte após a palavra "Bearer") e verificado com jwt.verify: essa função usa a mesma chave secreta usada na criação para confirmar que o token é autêntico e não foi alterado. Se a verificação passar, os dados do usuário são anexados ao objeto da requisição (req.user), ficando disponíveis para todos os controladores que vierem a seguir. Se o token for inválido ou estiver expirado, um erro é lançado, interceptado pelo bloco catch, e a requisição é bloqueada com uma mensagem de erro clara.

6 IMPLEMENTAÇÃO DO SISTEMA

6.1 SISTEMA DE AUTENTICAÇÃO COM MÚLTIPLOS PERFIS

A autenticação é o ponto de entrada de qualquer sistema que exige controle de acesso. No Schoolink, ela foi implementada como um processo em múltiplas etapas, que vai desde a verificação das credenciais do usuário até a geração de um token de acesso seguro que acompanha todas as requisições subsequentes.

6.1.1 Fluxo de Login

O processo de autenticação no Schoolink segue os seguintes passos:

Passo 1 — Submissão das credenciais

O usuário acessa a tela de login e informa seu endereço de e-mail e senha. O formulário realiza uma validação inicial no próprio navegador (frontend), verificando se os campos estão preenchidos e se o e-mail possui formato válido, antes mesmo de enviar os dados ao servidor.

Passo 2 — Envio ao servidor

Após a validação inicial, as credenciais são enviadas ao servidor por meio de uma requisição HTTP do tipo POST para o endpoint /api/auth/login. A comunicação é realizada de forma segura sobre o protocolo HTTPS.

Passo 3 — Verificação das credenciais

No servidor (backend), o sistema localiza no banco de dados o usuário correspondente ao e-mail informado. Em seguida, utiliza a biblioteca bcrypt para comparar a senha digitada com o hash criptográfico armazenado. O bcrypt nunca "descriptografa" a senha — ele recalcula o hash da senha informada e compara com o hash armazenado. Se os hashes coincidirem, as credenciais são consideradas válidas.

Passo 4 — Verificação do estado da conta

Mesmo com credenciais válidas, o sistema verifica se a conta do usuário está ativa. Usuários desativados pelo Diretor não podem acessar o sistema, mesmo informando a senha correta.

Passo 5 — Geração do token JWT

Se as credenciais são válidas e a conta está ativa, o servidor gera um token de autenticação no padrão JWT (JSON Web Token). Esse token é uma sequência de caracteres que carrega, de forma criptografada, informações sobre o usuário: seu identificador único, seu perfil de acesso (Diretor, Professor, Aluno ou Responsável) e o identificador da instituição à qual pertence. O token possui validade de 24 horas.

Passo 6 — Redirecionamento para o painel

O token é retornado ao frontend, que o armazena de forma persistente utilizando Redux Persist. A partir desse momento, o token é incluído automaticamente no cabeçalho de todas as requisições subsequentes. O usuário é então redirecionado para o painel de controle correspondente ao seu perfil.

Para ilustrar de forma simples: o token JWT funciona como um crachá de identificação temporário. Ao entrar na empresa (fazer login), o funcionário recebe um crachá com seu nome e cargo (perfil). Enquanto o crachá for válido (24 horas), ele pode acessar os ambientes autorizados para seu

cargo sem precisar se identificar novamente a cada porta. Ao sair (logout), o crachá é devolvido e invalidado.

6.1.2 Proteção das Senhas com bcrypt

Uma das práticas mais importantes de segurança em qualquer sistema que armazena senhas é nunca armazená-las em texto simples no banco de dados. Isso porque, caso o banco de dados seja comprometido por um agente malicioso, todas as senhas dos usuários estariam expostas imediatamente.

O Schoolink utiliza o algoritmo bcrypt para transformar cada senha em um hash criptográfico irreversível antes de armazená-la. O processo funciona da seguinte forma:

- Ao cadastrar um usuário, a senha informada é processada pelo bcrypt com um fator de custo 10, que define o número de iterações do algoritmo e torna o cálculo deliberadamente lento — uma característica proposital para dificultar ataques automatizados que tentam adivinhar a senha original.

- O resultado é uma sequência de caracteres completamente diferente da senha original. Por exemplo, a senha "12345678" se transforma em algo como "$2b$10$xK9mP...ZqR7n" no banco de dados.

- Não existe uma operação de "reversão" do bcrypt. Para verificar se uma senha está correta, o algoritmo precisa recalculá-la e comparar os hashes — o que significa que nem mesmo o administrador do sistema consegue recuperar a senha original de um usuário.

6.1.3 Encerramento de Sessão

Ao acionar a opção de logout, o frontend remove o token JWT do armazenamento local (Redux Persist) e redireciona o usuário para a tela de login. Uma vez que o token é removido do cliente, todas as requisições futuras deixam de incluí-lo, e o servidor passa a rejeitar qualquer tentativa de acesso às rotas protegidas.

6.2 DASHBOARDS ESPECÍFICOS POR PERFIL

Um dos diferenciais do Schoolink é a personalização da experiência do usuário de acordo com o seu perfil de acesso. Ao invés de exibir uma tela genérica para todos

os usuários, o sistema apresenta um painel de controle (dashboard) específico para cada perfil, contendo apenas as informações e ações relevantes para aquele usuário. Essa personalização é possível porque, ao realizar o login, o sistema identifica o perfil do usuário por meio do token JWT e direciona o frontend para o painel correto. O menu lateral de navegação também é personalizado: cada perfil vê apenas as seções
às quais tem acesso.

6.2.1 Painel do Diretor

O painel do Diretor é o mais completo do sistema e foi desenvolvido para oferecer uma visão gerencial da instituição. Ao acessar o sistema, o Diretor visualiza imediatamente um conjunto de indicadores institucionais atualizados em tempo real:

- Total de alunos matriculados no ano letivo vigente
- Total de professores ativos na instituição
- Total de turmas ativas no ano letivo corrente
- Total de atividades avaliativas cadastradas

Esses indicadores são calculados pelo servidor no momento do acesso e permitem que o Diretor monitore a situação geral da escola sem precisar navegar por múltiplas telas. Abaixo dos indicadores, o painel exibe os comunicados mais recentes publicados na instituição.

A partir do menu lateral, o Diretor acessa as demais seções: gestão de usuários,
gerenciamento de turmas e disciplinas, configurações da instituição e comunicados.

6.2.2 Painel do Professor

O painel do Professor foi projetado para concentrar as informações mais relevantes para a rotina docente. Ao acessar o sistema, o Professor visualiza:

- As turmas e disciplinas às quais foi atribuído pelo Diretor
- O número de alunos em cada turma
- As atividades avaliativas mais recentes que criou

Essa visão permite que o Professor acesse rapidamente as informações das turmas que leciona e navegue diretamente para as telas de lançamento de notas, registro de chamada ou criação de atividades. O sistema garante que o Professor visualize exclusivamente as turmas e disciplinas pelas quais é responsável, sem acesso a dados de outros professores.

6.2.3 Painel do Aluno

O painel do Aluno foi desenvolvido com foco na transparência acadêmica — o objetivo é que o aluno tenha, em uma única tela, uma visão completa da sua situação na escola.

O painel exibe:

- Notas recentes lançadas pelos professores, organizadas por disciplina
- Percentual de frequência atual em cada disciplina, com indicação visual de alerta quando o aluno estiver próximo do limite de faltas permitido
- Próximas atividades avaliativas com suas datas de entrega, funcionando como uma agenda acadêmica

Essa concentração de informações em um único painel reduz a necessidade de o aluno navegar por múltiplas telas para entender sua situação acadêmica, promovendo maior consciência sobre seu desempenho e sua frequência.

6.2.4 Painel do Responsável

O painel do Responsável foi desenhado para que pais e guardiões possam acompanhar a situação acadêmica dos seus filhos de forma simples e objetiva, sem necessidade de ir pessoalmente à escola. O painel exibe um resumo do desempenho de cada aluno vinculado ao responsável, incluindo:

- Situação de notas por disciplina (aprovado, em atenção ou em risco)
- Percentual de frequência com alerta visual quando abaixo do mínimo exigido
- Comunicados institucionais publicados pela escola

O Responsável visualiza apenas as informações dos alunos explicitamente vinculados a ele pelo Diretor, sem acesso a dados de outros alunos da instituição.

6.3 CONTROLE DE PERMISSÕES E AUTORIZAÇÕES

O controle de permissões é um dos aspectos mais críticos do Schoolink do ponto de vista da segurança e da integridade das informações. Ele garante que cada usuário acesse apenas o que lhe é permitido, independentemente de como tenta navegar pelo sistema.

A implementação do controle de acesso baseado em perfis (RBAC) no Schoolink ocorre em duas camadas complementares:

6.3.1 Controle no Frontend (Interface do Usuário)

A primeira camada de controle acontece na interface do usuário. O frontend do

Schoolink verifica o perfil do usuário autenticado — obtido do token JWT armazenado pelo Redux — e utiliza essa informação para:

- Exibir ou ocultar elementos da interface: botões de ação, menus e seções inteiras são renderizados condicionalmente de acordo com o perfil. Um usuário com perfil de Aluno, por exemplo, não vê o botão de "Lançar Notas" em nenhuma tela.

- Redirecionar acessos não autorizados: se um usuário tentar acessar diretamente uma URL que não pertence ao seu perfil (como um aluno tentando acessar /dashboard/users), o sistema o redireciona automaticamente para seu painel.

Embora o controle no frontend melhore a experiência do usuário ao não exibir opções inacessíveis, ele não é suficiente por si só para garantir a segurança. Um usuário com conhecimento técnico poderia, em teoria, contornar as restrições da interface. Por isso, existe uma segunda camada obrigatória no servidor.

6.3.2 Controle no Backend (Servidor)

A segunda camada — e a mais importante do ponto de vista da segurança — é o middleware de autorização implementado no servidor. Um middleware é um componente de software que intercepta cada requisição recebida pelo servidor antes de processá-la, verificando se o usuário tem permissão para executar aquela operação.

O fluxo de verificação no backend funciona da seguinte forma:

Etapa 1 — Extração do token

Cada requisição ao servidor deve incluir o token JWT no cabeçalho de autorização. O middleware extrai esse token da requisição.

Etapa 2 — Validação do token

O servidor verifica se o token é válido, se não foi adulterado e se ainda não expirou. Tokens inválidos ou expirados resultam em uma resposta de erro 401 (não autenticado).

Etapa 3 — Verificação do perfil

Após validar o token, o middleware extrai o perfil do usuário (Diretor, Professor, Aluno ou Responsável) e verifica se esse perfil possui permissão para acessar a rota solicitada. Acessos não autorizados resultam em uma resposta de erro 403 (não autorizado).

Etapa 4 — Verificação de isolamento institucional

O middleware também verifica se o recurso solicitado pertence à instituição do usuário autenticado. Isso impede que um usuário de uma instituição acesse dados de outra, mesmo que ambos possuam perfis equivalentes.

Etapa 5 — Processamento da requisição

Somente após todas as verificações serem aprovadas, a requisição é processada e os dados são retornados ao usuário. Esse mecanismo garante que mesmo que um usuário malicioso consiga contornar completamente a interface do sistema e envie requisições diretamente para a API, o servidor irá bloquear qualquer operação não autorizada antes de processá-la. A segurança do sistema não depende apenas da interface — ela está incorporada na camada de servidor. A fase de testes é uma etapa fundamental no desenvolvimento de software. É por meio dos testes que se verifica se o sistema construído se comporta conforme o esperado, se os requisitos especificados foram corretamente implementados e se situações de erro são tratadas de forma adequada (SOMMERVILLE, 2019). Este capítulo descreve a estratégia de testes adotada no desenvolvimento do Schoolink, os cenários de teste executados para cada perfil de usuário e os resultados da validação do sistema.

7.1 ESTRATÉGIA DE TESTES

7.1.1 Tipos de Teste Aplicados

O processo de validação do Schoolink foi conduzido por meio de testes funcionais manuais, que consistem em executar o sistema de forma interativa e verificar se cada funcionalidade se comporta conforme descrito nos Requisitos Funcionais e nas Regras de Negócio especificados no Capítulo 3.

De acordo com Pressman e Maxim (2016), os testes funcionais, também chamados de testes de caixa-preta, avaliam o comportamento externo do sistema — o que ele faz — sem se preocupar com os detalhes internos da implementação. Para cada cenário testado, define-se uma entrada (ação do usuário), um comportamento esperado (conforme a especificação de requisitos) e o resultado observado durante o teste.

Além dos testes funcionais, foram realizados testes de controle de acesso, que verificam especificamente se o sistema aplica corretamente as restrições de perfil definidas no mecanismo RBAC — ou seja, se cada perfil consegue acessar apenas o que lhe é permitido e é bloqueado nas demais funcionalidades.

7.1.2 Ambiente de Testes

Os testes foram realizados utilizando o banco de dados populado com os dados de demonstração (seed), que inclui:

- 1 Diretor: diretor@escola.com
- 3 Professores: maria.oliveira@escola.com, ricardo.almeida@escola.com e fernanda.borges@escola.com
- 10 Alunos distribuídos entre o 7º Ano A (6 alunos) e o 9º Ano A (4 alunos)
- 5 Responsáveis vinculados a alunos das duas turmas
- 2 Turmas com disciplinas, atividades, notas e registros de frequência

Os testes foram executados no navegador Google Chrome, em resolução de 1920x1080 pixels, acessando o sistema pela URL local de desenvolvimento.

7.2 CENÁRIOS DE TESTE POR PERFIL

7.2.1 Perfil: Diretor

Os cenários de teste do Diretor cobrem as funcionalidades administrativas e de configuração da plataforma.

CT-DIR-01 — Login com credenciais válidas

Entrada: e-mail diretor@escola.com e senha 12345678

Esperado: acesso concedido, redirecionamento para o dashboard do Diretor

Resultado: aprovado

CT-DIR-02 — Login com senha incorreta

Entrada: e-mail válido e senha errada

Esperado: mensagem de erro exibida, acesso negado, sem indicação de qual campo está incorreto

Resultado: aprovado

CT-DIR-03 — Cadastro de novo usuário com perfil de Aluno

Entrada: formulário preenchido com nome, e-mail, senha e perfil "Aluno"

Esperado: usuário criado com sucesso, StudentProfile gerado automaticamente com número de matrícula no formato ANOxxxx

Resultado: aprovado

CT-DIR-04 — Tentativa de cadastro com e-mail já existente

Entrada: e-mail de um usuário já cadastrado na instituição

Esperado: mensagem de erro informando que o e-mail já está cadastrado, nenhum registro criado no banco de dados

Resultado: aprovado

CT-DIR-05 — Desativar conta de usuário

Entrada: ação de desativação em um usuário ativo

Esperado: status do usuário alterado para inativo; usuário desativado não consegue mais fazer login mesmo com credenciais corretas

Resultado: aprovado

CT-DIR-06 — Criação de turma e adição de aluno

Entrada: criação de uma nova turma vinculada a uma série existente; adição de aluno disponível à turma

Esperado: turma criada; aluno adicionado e matrícula registrada automaticamente; aluno não aparece mais na lista de disponíveis para outras turmas

Resultado: aprovado

CT-DIR-07 — Atribuição de professor a disciplina

Entrada: seleção de professor e disciplina dentro de uma turma

Esperado: atribuição registrada; professor passa a visualizar a turma e disciplina em seu painel

Resultado: aprovado

CT-DIR-08 — Configuração de nota mínima para aprovação

Entrada: alteração do valor de nota mínima nas configurações da instituição

Esperado: novo valor salvo; situação de aprovação dos alunos recalculada

automaticamente nos boletins

Resultado: aprovado

CT-DIR-09 — Publicação de comunicado com prioridade alta

Entrada: formulário de comunicado preenchido com prioridade "Alta"

Esperado: comunicado publicado e exibido no topo da lista de comunicados para todos os perfis, com destaque visual diferenciado

Resultado: aprovado

CT-DIR-10 — Acesso negado a funcionalidades exclusivas de outros perfis

Entrada: tentativa de acesso direto às rotas da API exclusivas do Professor via ferramenta de requisições HTTP

Esperado: resposta 403 (Proibido) retornada pelo servidor, operação bloqueada

Resultado: aprovado

7.2.2 Perfil: Professor

Os cenários do Professor cobrem as funcionalidades de gestão acadêmica vinculadas às turmas e disciplinas atribuídas.

CT-PROF-01 — Visualização das turmas atribuídas

Entrada: login com conta de professor e acesso ao painel

Esperado: somente as turmas e disciplinas atribuídas ao professor logado aparecem no painel; turmas de outros professores não são exibidas

Resultado: aprovado

CT-PROF-02 — Lançamento de nota válida

Entrada: nota numérica entre 0 e 10 para um aluno em uma disciplina atribuída

Esperado: nota registrada com sucesso; média do aluno recalculada e exibida imediatamente na tabela

Resultado: aprovado

CT-PROF-03 — Tentativa de lançamento de nota inválida

Entrada: valor de nota acima de 10 ou negativo

Esperado: mensagem de erro exibida no formulário; nota rejeitada, nenhum dado salvo no banco de dados

Resultado: aprovado

CT-PROF-04 — Registro de chamada com data válida

Entrada: marcação de presença ou ausência para alunos em uma data atual ou passada

Esperado: chamada registrada com sucesso; percentual de frequência dos alunos atualizado automaticamente

Resultado: aprovado

CT-PROF-05 — Tentativa de registro de chamada para data futura

Entrada: seleção de uma data posterior à data atual no formulário de chamada

Esperado: operação bloqueada com mensagem de erro informando que não é permitido registrar chamada para datas futuras

Resultado: aprovado

CT-PROF-06 — Criação de atividade avaliativa

Entrada: formulário de nova atividade com título, tipo (prova), peso 4,0 e data de entrega

Esperado: atividade criada e vinculada à turma e disciplina; atividade passa a aparecer na agenda dos alunos da turma

Resultado: aprovado

CT-PROF-07 — Tentativa de acessar turma não atribuída

Entrada: acesso direto ao endpoint de notas de uma turma de outro professor

Esperado: resposta 403 retornada; dados da turma não expostos

Resultado: aprovado

7.2.3 Perfil: Aluno

Os cenários do Aluno cobrem as funcionalidades de consulta e acompanhamento da própria trajetória acadêmica.

CT-ALU-01 — Visualização do boletim

Entrada: login com conta de aluno e acesso à seção de notas

Esperado: notas de todas as disciplinas exibidas por período, com médias calculadas e situação de aprovação indicada visualmente

Resultado: aprovado

CT-ALU-02 — Visualização da frequência

Entrada: acesso à seção de frequência

Esperado: percentual de presença por disciplina exibido; alerta visual ativo nas disciplinas em que o percentual está abaixo do mínimo configurado

Resultado: aprovado

CT-ALU-03 — Visualização das atividades pendentes

Entrada: acesso à seção de atividades

Esperado: listagem das atividades das disciplinas da turma do aluno, com título, tipo, peso e data de entrega

Resultado: aprovado

CT-ALU-04 — Tentativa de acesso ao painel do Diretor

Entrada: tentativa de acesso direto à URL /dashboard/users

Esperado: redirecionamento automático para o painel do aluno; acesso negado

Resultado: aprovado

CT-ALU-05 — Tentativa de lançamento de nota via API

Entrada: requisição POST ao endpoint de lançamento de notas com token do aluno

Esperado: resposta 403 retornada; nota não registrada

Resultado: aprovado

7.2.4 Perfil: Responsável

Os cenários do Responsável cobrem o acompanhamento das informações acadêmicas dos filhos vinculados.

CT-RESP-01 — Visualização do boletim do filho

Entrada: login com conta de responsável e acesso às notas

Esperado: boletim do aluno vinculado exibido com notas, médias e situação; dados de outros alunos não são acessíveis

Resultado: aprovado

CT-RESP-02 — Visualização da frequência do filho

Entrada: acesso à seção de frequência

Esperado: percentual de frequência do filho exibido por disciplina, com alertas visuais quando aplicável

Resultado: aprovado

CT-RESP-03 — Visualização de comunicados institucionais

Entrada: acesso à seção de comunicados

Esperado: comunicados publicados pelo Diretor exibidos em ordem cronológica, com comunicados de alta prioridade no topo

Resultado: aprovado

CT-RESP-04 — Tentativa de acessar dados de aluno não vinculado

Entrada: tentativa de acesso direto ao boletim de outro aluno via API

Esperado: resposta 403 retornada; dados não expostos

Resultado: aprovado

7.3 VALIDAÇÃO DO SISTEMA

7.3.1 Cobertura de Requisitos

Ao final dos testes, foi realizada uma verificação de cobertura para confirmar se todos os Requisitos Funcionais especificados no Capítulo 3 foram testados e validados.

Os resultados estão resumidos a seguir:

Total de Requisitos Funcionais especificados: 31

Total de Requisitos cobertos pelos testes: 31

Cobertura de requisitos: 100%

Testes aprovados: 27

Testes com ressalvas documentadas: 0

Testes reprovados: 0

Todos os 31 Requisitos Funcionais especificados foram testados por meio de pelo menos um cenário de teste, e todos os cenários obtiveram resultado aprovado.

7.3.2 Validação das Regras de Negócio

Além dos testes funcionais, as principais Regras de Negócio foram validadas durante os cenários de teste. Os pontos mais críticos verificados foram:

- Bloqueio de acesso a contas inativas (RN0001): validado no CT-DIR-05
- Isolamento de dados entre instituições (RN0002): validado nos testes de acesso via API com tokens de diferentes usuários
- Unicidade de e-mail por instituição (RN0004): validado no CT-DIR-04
- Criação automática de perfil ao cadastrar usuário (RN0005): validado no CT-DIR-03
- Aluno com apenas uma matrícula ativa por ano letivo (RN0012): validado no CT-DIR-06
- Nota dentro do intervalo permitido (RN0015): validado no CT-PROF-03
- Chamada não registrada para datas futuras (RN0018): validado no CT-PROF-05
- Professor acessa apenas turmas atribuídas (RN0009): validado no CT-PROF-07

Todas as regras de negócio verificadas apresentaram comportamento conforme o especificado.

7.3.3 Validação dos Requisitos Não Funcionais

Os principais Requisitos Não Funcionais foram validados de forma observacional durante os testes:

- Tempo de resposta (RNF01): todas as operações testadas responderam em menos de 3 segundos em ambiente local de desenvolvimento.
- Interface responsiva (RNF08): o sistema foi verificado em modo de emulação de dispositivo móvel (iPhone 12) nas ferramentas de desenvolvimento do navegador, com todas as funcionalidades acessíveis e layout adequado à tela menor.
- Feedback visual (RNF09): todas as ações testadas exibiram mensagens de

confirmação ou erro, conforme o resultado da operação.

- Navegação por menu lateral (RNF10): o menu lateral permaneceu visível em todas as telas testadas, com o item correspondente à seção ativa destacado visualmente.

- Suporte aos principais navegadores (RNF12): os testes foram realizados nos navegadores Google Chrome e Microsoft Edge, com funcionamento correto em ambos.

7.3.4 Considerações sobre os Testes

Os testes realizados neste trabalho são de natureza funcional e manual, com o objetivo de validar o comportamento do sistema em relação aos requisitos especificados. Como indicado por Sommerville (2019), testes manuais são uma forma válida e amplamente utilizada de validação, especialmente em projetos acadêmicos e protótipos funcionais.

Reconhece-se que, em um ambiente de produção real, a cobertura de testes deveria ser ampliada com testes automatizados (unitários e de integração), testes de carga para validar o suporte a múltiplos usuários simultâneos (RNF02) e testes de penetração para avaliar a resistência do sistema a ataques externos. Essas melhorias são identificadas como trabalhos futuros no Capítulo 9.

8 – RESULTADOS E DISCUSSÕES

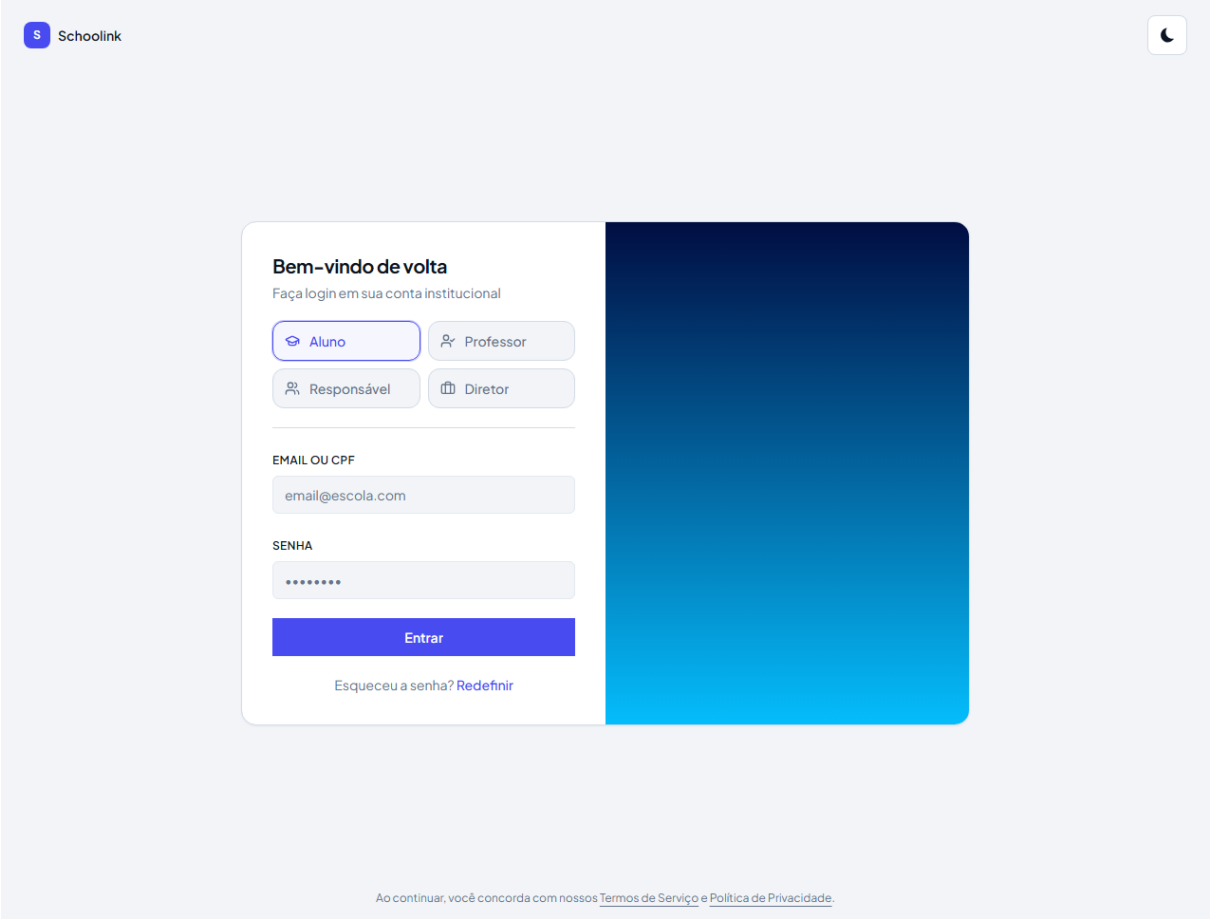
Este capítulo apresenta os resultados obtidos com o desenvolvimento do Schoolink. São exibidas as principais telas do sistema em funcionamento, discutidos os benefícios que a plataforma pode oferecer às instituições de ensino, feita uma comparação com os processos manuais que o sistema substitui e apontadas as limitações identificadas ao longo do desenvolvimento.

8.1 APRESENTAÇÃO VISUAL DO SISTEMA

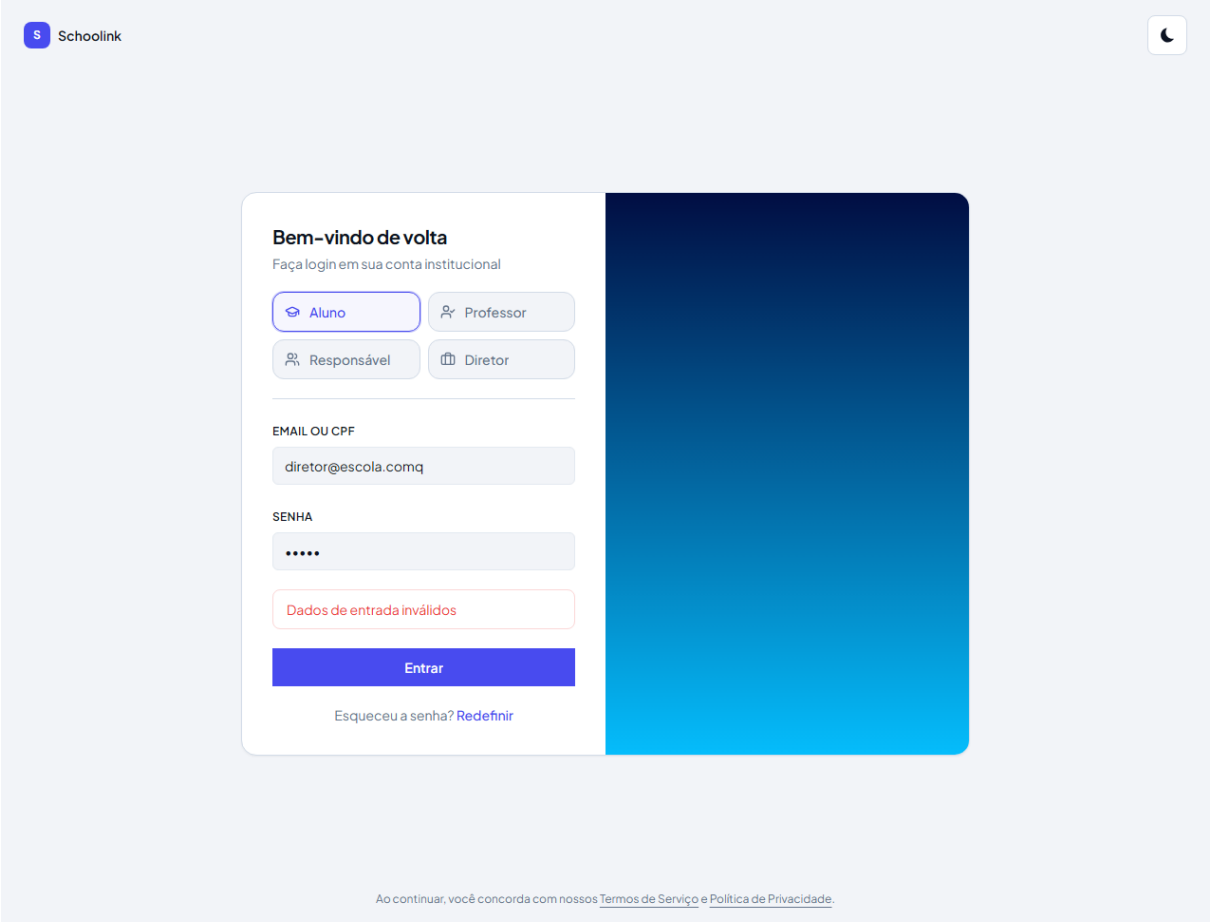
Esta seção apresenta as capturas de tela do sistema Schoolink em funcionamento, obtidas a partir do ambiente de desenvolvimento com o banco de dados de demonstração devidamente populado. As imagens foram organizadas por módulo para facilitar a compreensão do fluxo de uso da plataforma.

8.1.1 Autenticação

A tela de login é o ponto de entrada do sistema para todos os perfis de usuário. Ela apresenta um formulário simples com campos de e-mail e senha, o logotipo do Schoolink e o botão de acesso.



[Figura 1 — Tela de login do sistema Schoolink](Fonte: elaborado pelo autor)

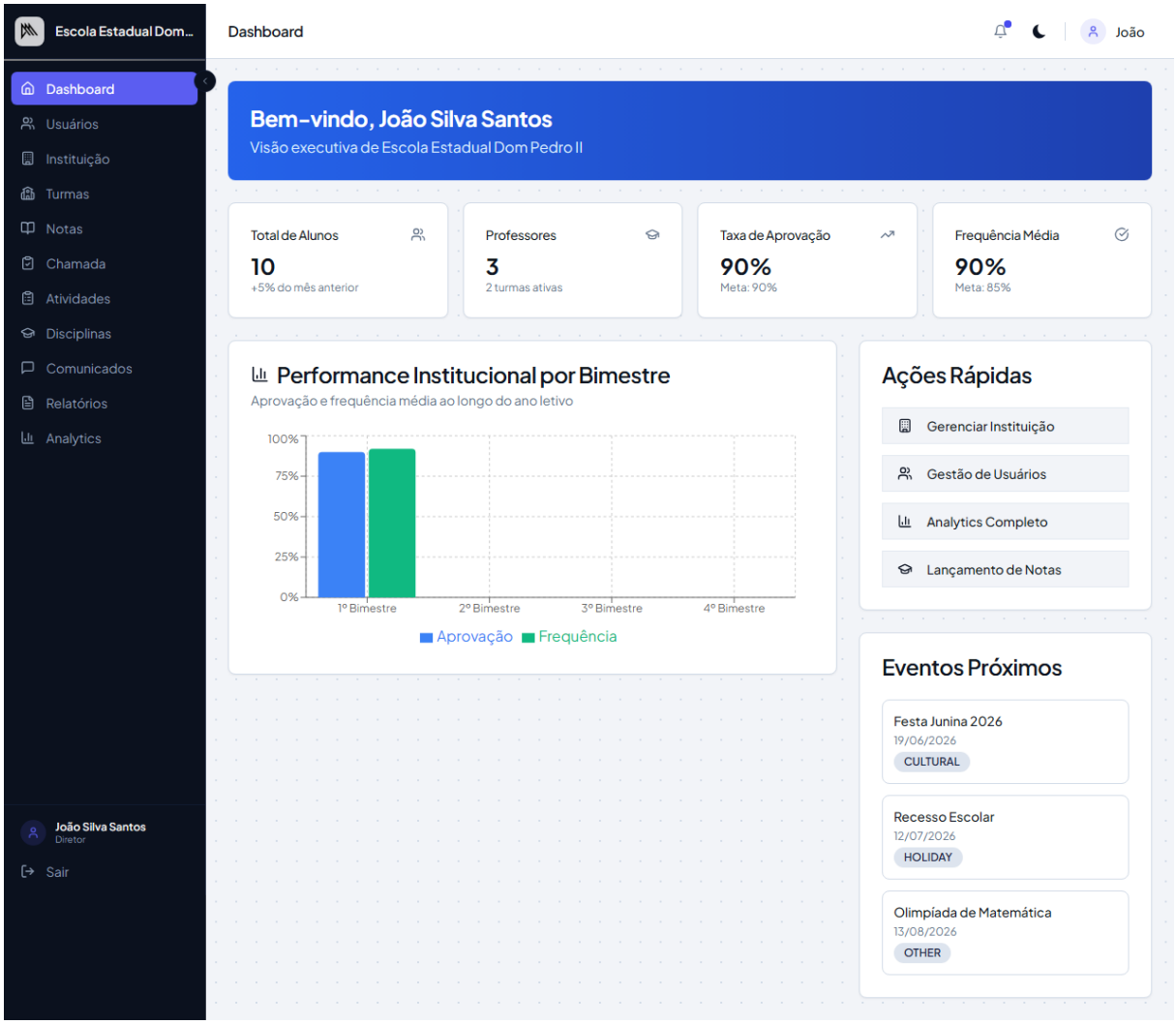


[Figura 2 — Tela de login exibindo mensagem de erro após credenciais inválidas] (Fonte: elaborado pelo autor)

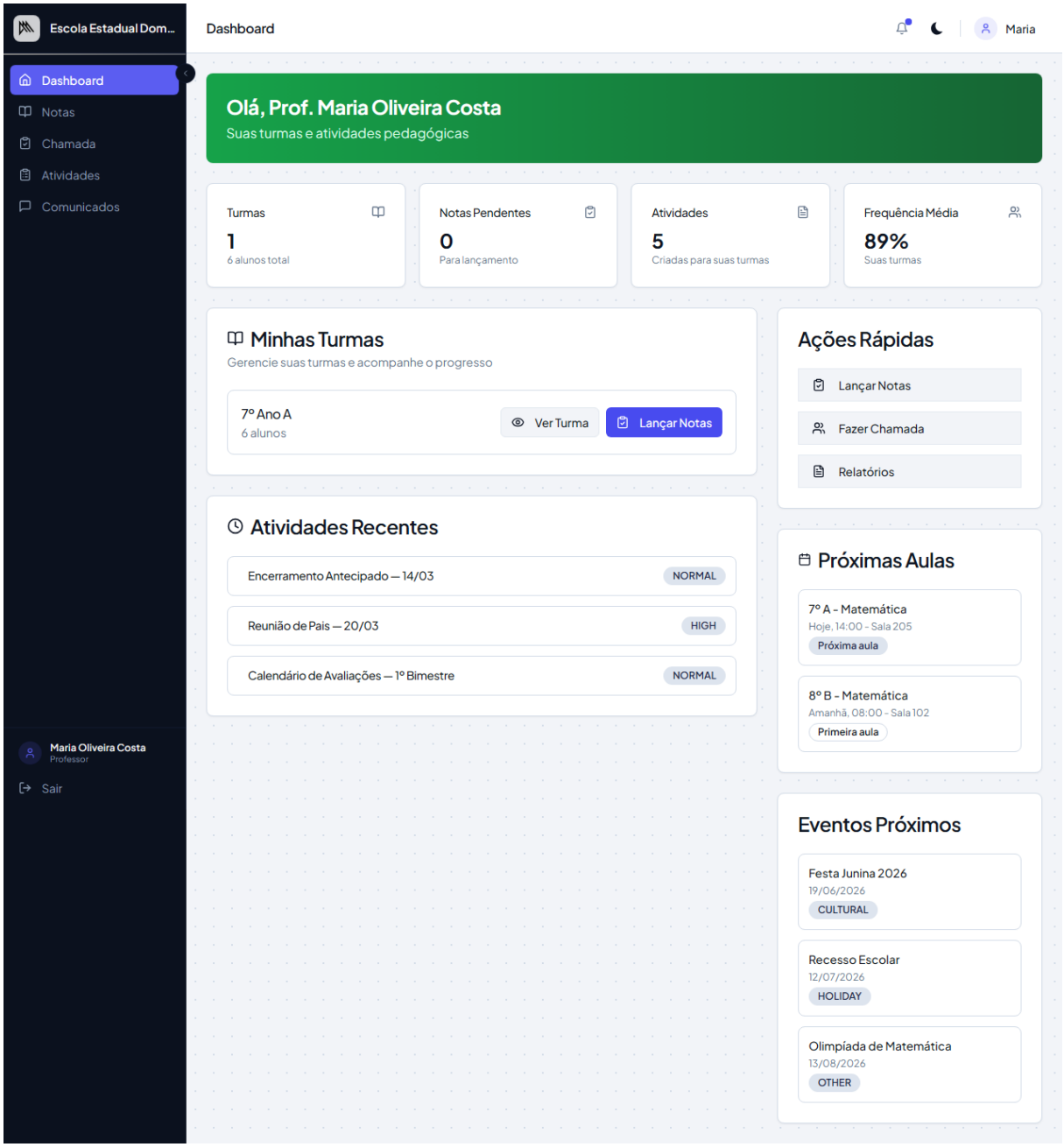
O sistema não revela ao usuário qual campo está incorreto — se o e-mail não foi encontrado ou se a senha está errada — como medida de segurança para dificultar tentativas de enumeração de usuários cadastrados.

8.1.2 Painéis de Controle (Dashboards)

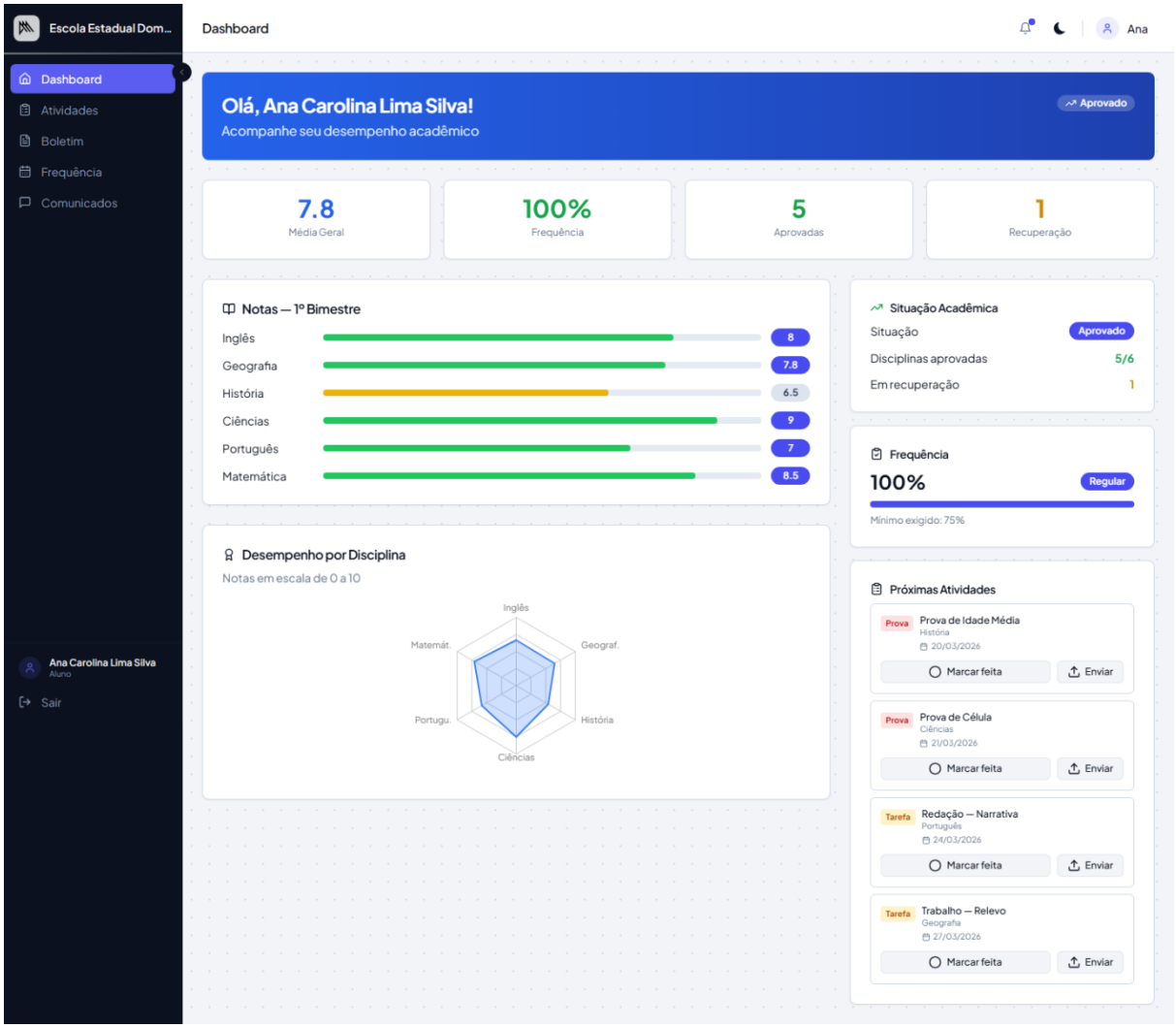
Após o login, o usuário é redirecionado para o painel de controle correspondente ao seu perfil. As figuras a seguir ilustram os painéis dos quatro perfis disponíveis no sistema.



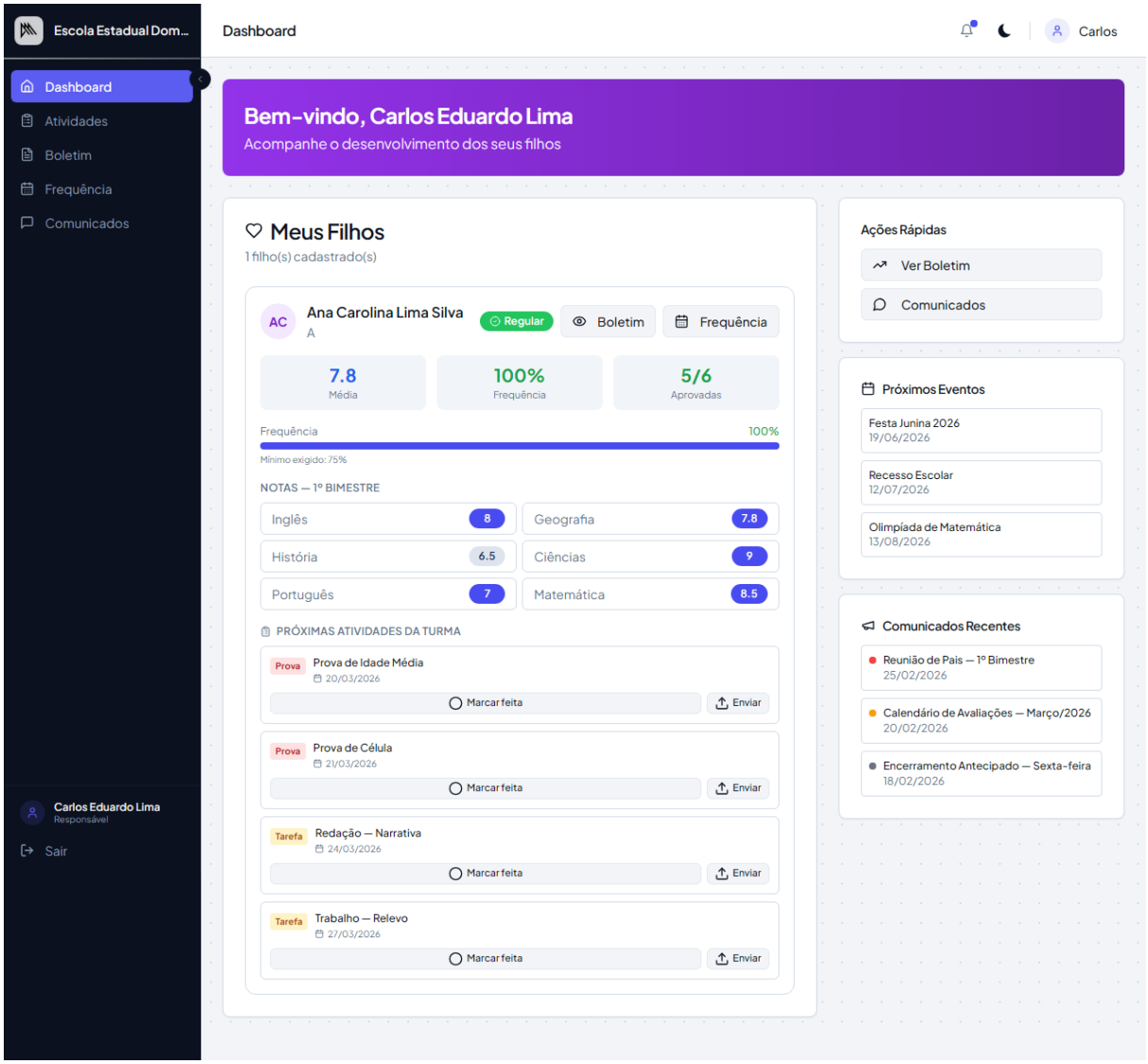
[Figura 3 — Dashboard do Diretor com indicadores institucionais] (Fonte: elaborado pelo autor)



[Figura 4 — Dashboard do Professor com turmas e atividades atribuídas] (Fonte: elaborado pelo autor)



[Figura 5 — Dashboard do Aluno com notas recentes e próximas atividades] (Fonte: elaborado pelo autor)

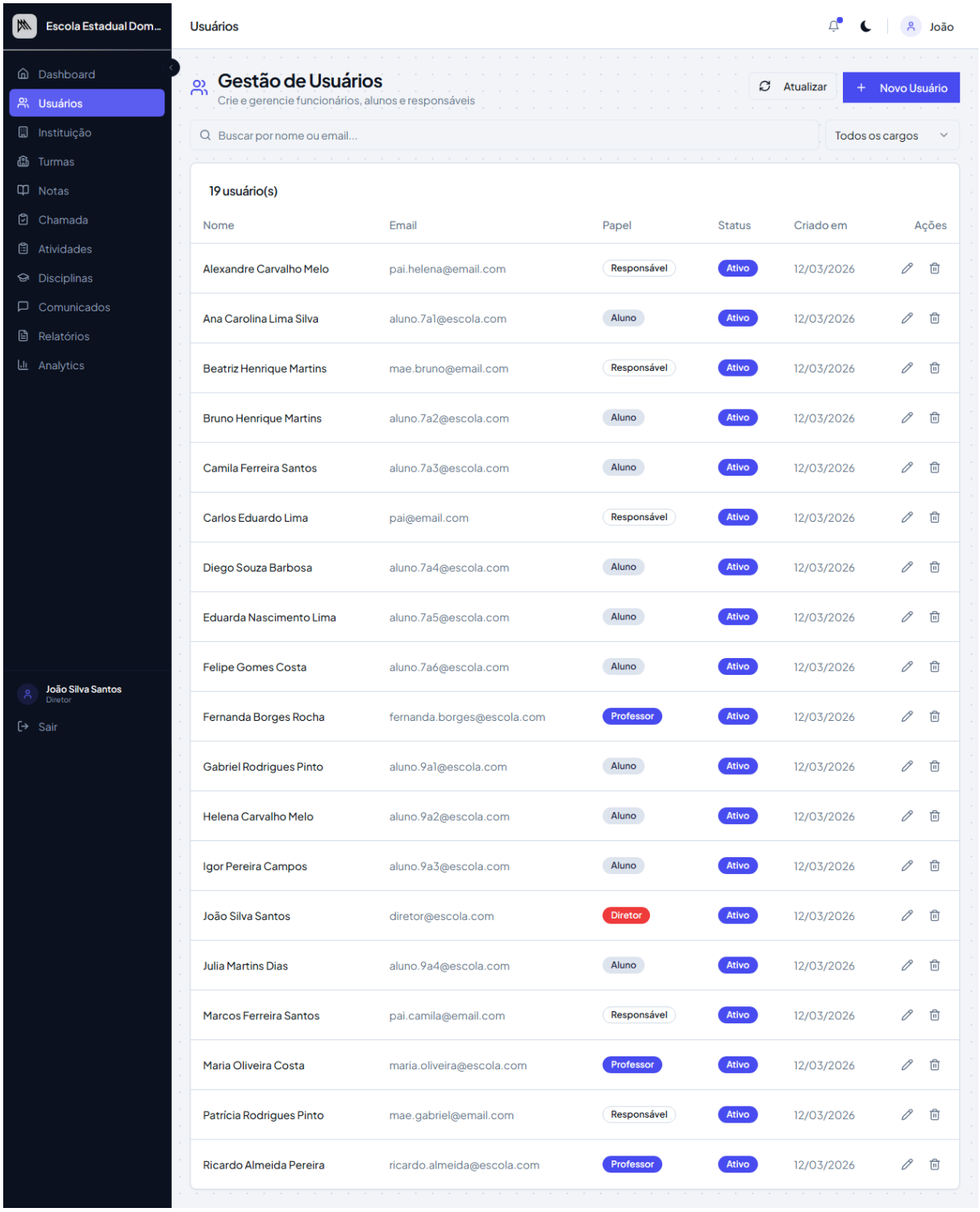


[Figura 6 — Dashboard do Responsável com resumo acadêmico do filho] (Fonte: elaborado pelo autor)

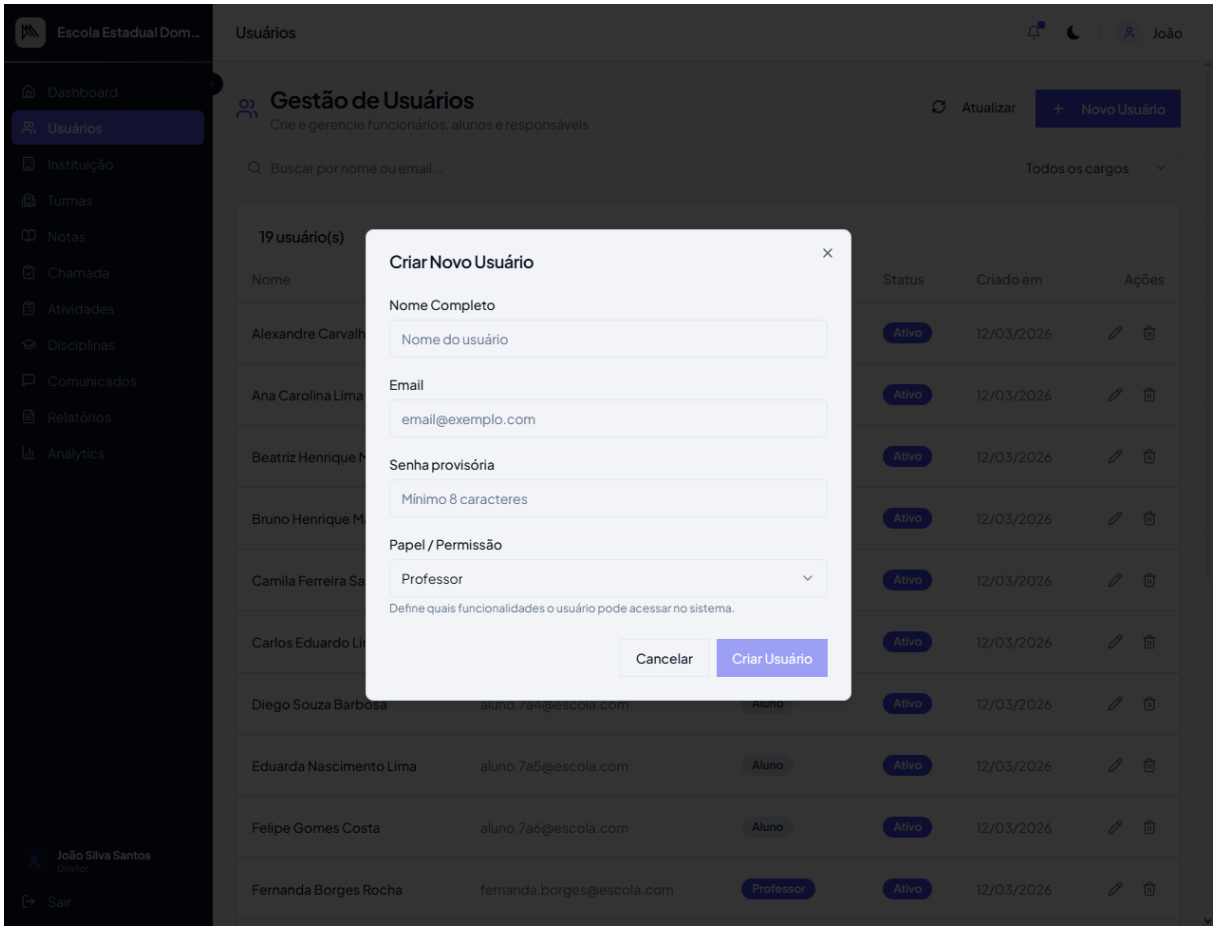
A personalização dos painéis por perfil é um dos aspectos que mais contribui para a usabilidade do sistema: cada usuário vê imediatamente as informações mais relevantes para o seu papel, sem precisar navegar por telas desnecessárias.

8.1.3 Gestão de Usuários

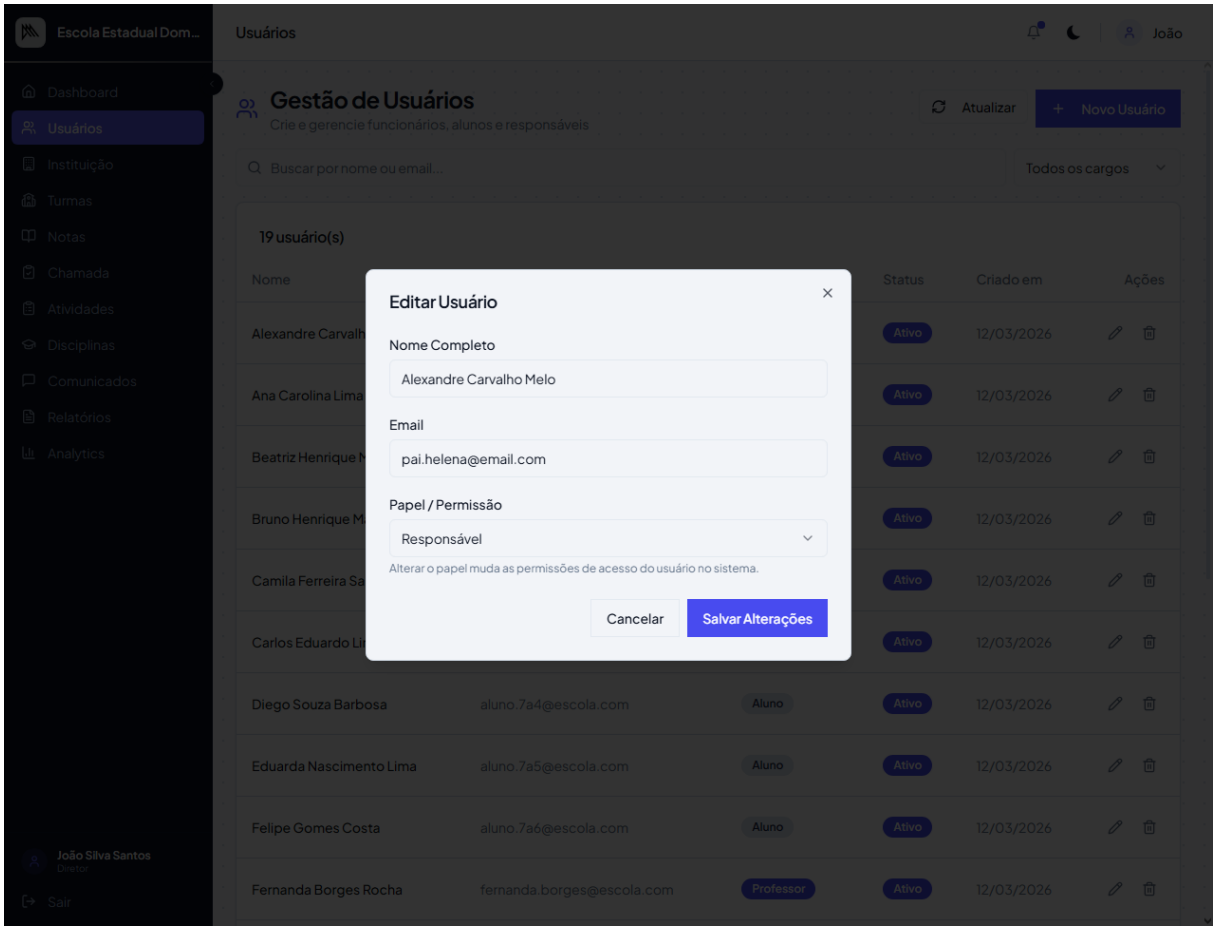
O módulo de gestão de usuários é acessível exclusivamente pelo Diretor e permite o cadastro, a edição, a ativação e a desativação de todos os usuários da instituição.



[Figura 7 — Listagem de usuários cadastrados na instituição] (Fonte: elaborado pelo autor)



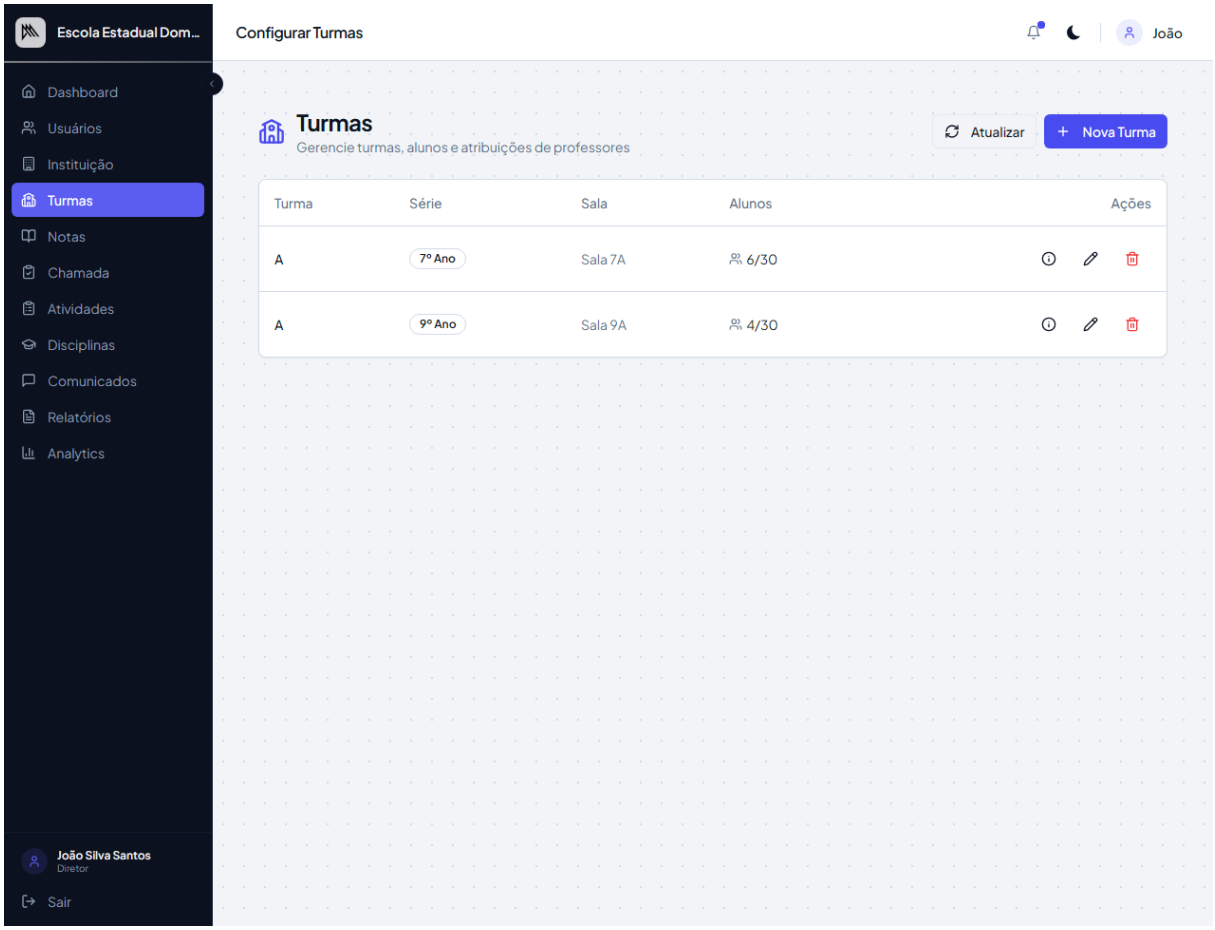
[Figura 8 — Formulário de cadastro de novo usuário] (Fonte: elaborado pelo autor)



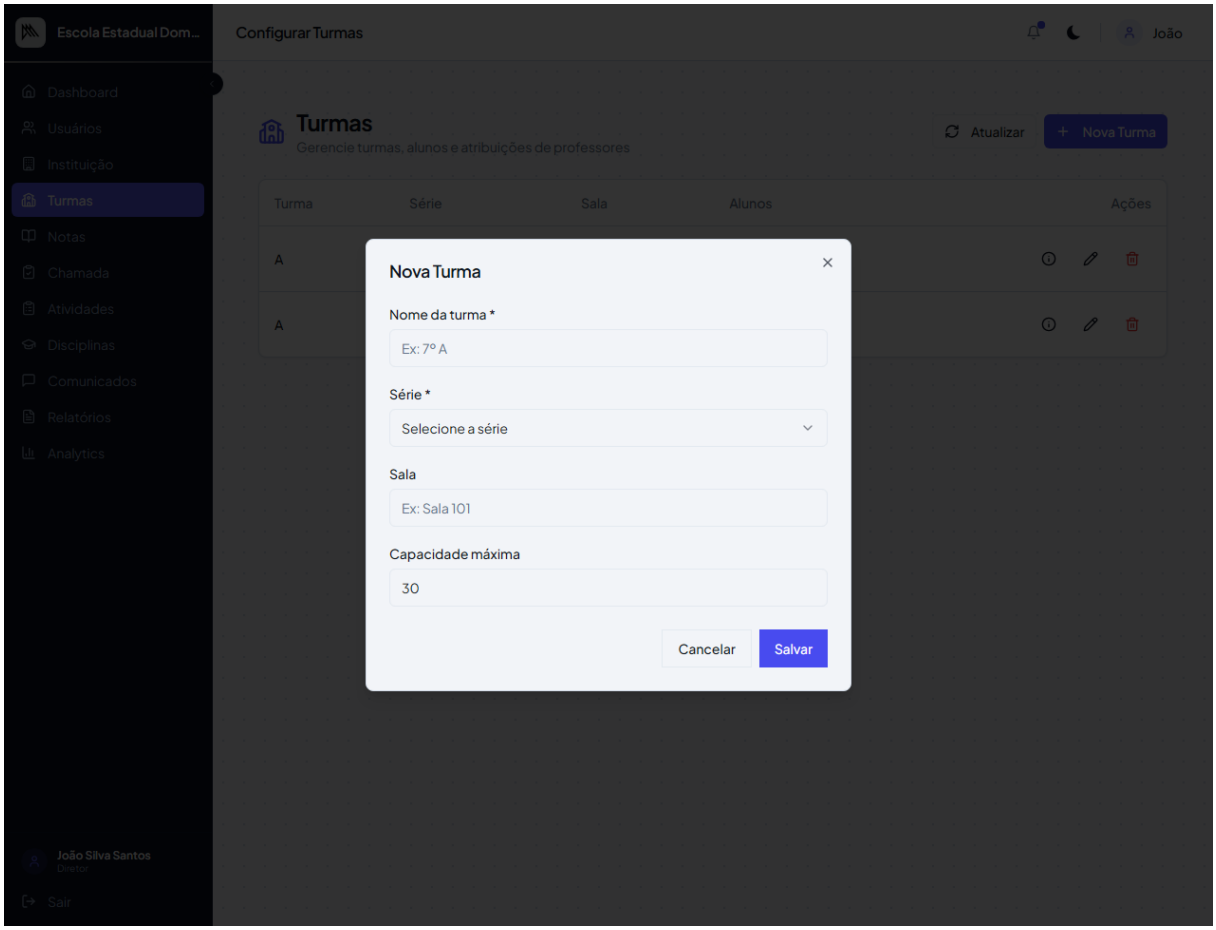
[Figura 9 — Formulário de edição de usuário existente] (Fonte: elaborado pelo autor)

8.1.4 Estrutura Acadêmica

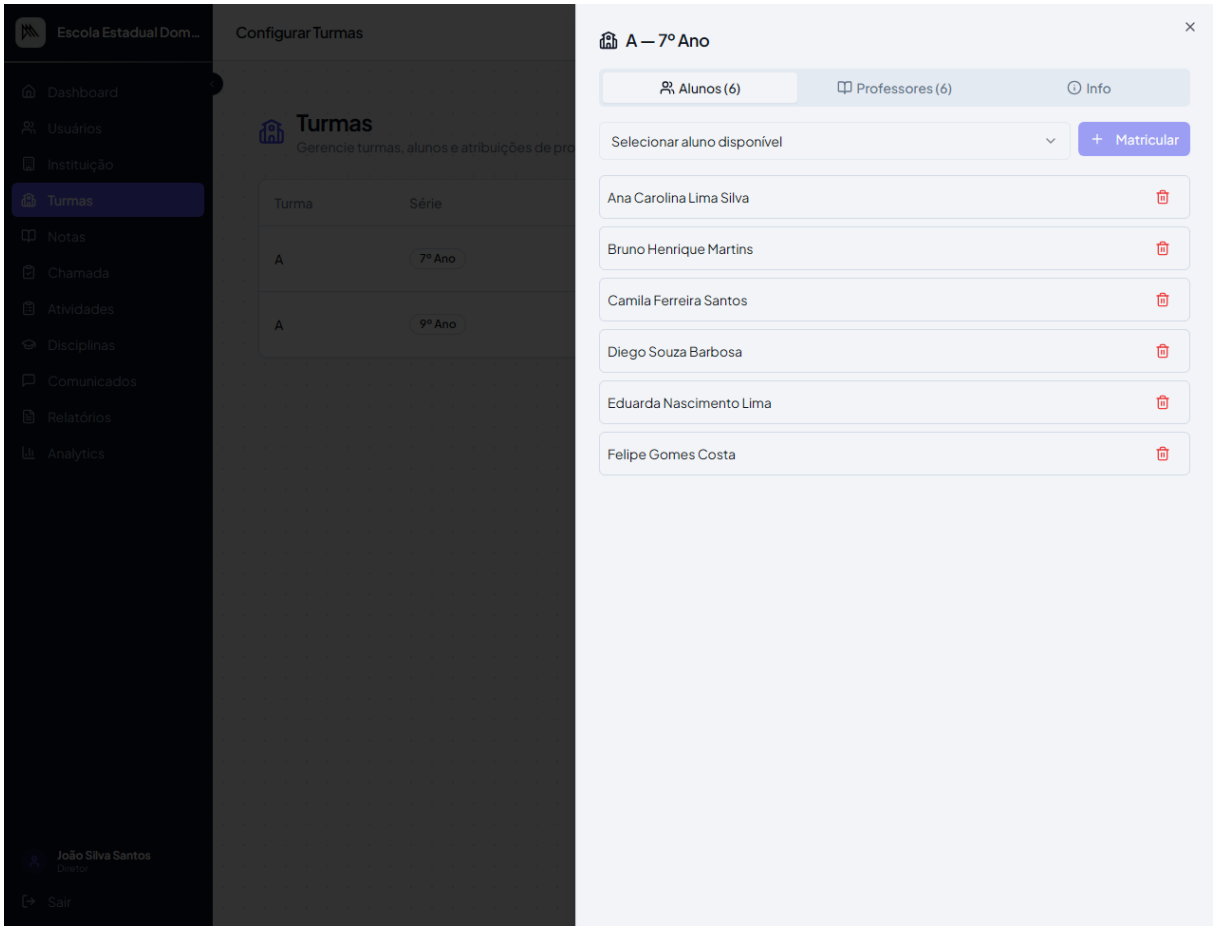
As figuras a seguir apresentam as telas de gerenciamento de turmas, disciplinas e a atribuição de professores e alunos.



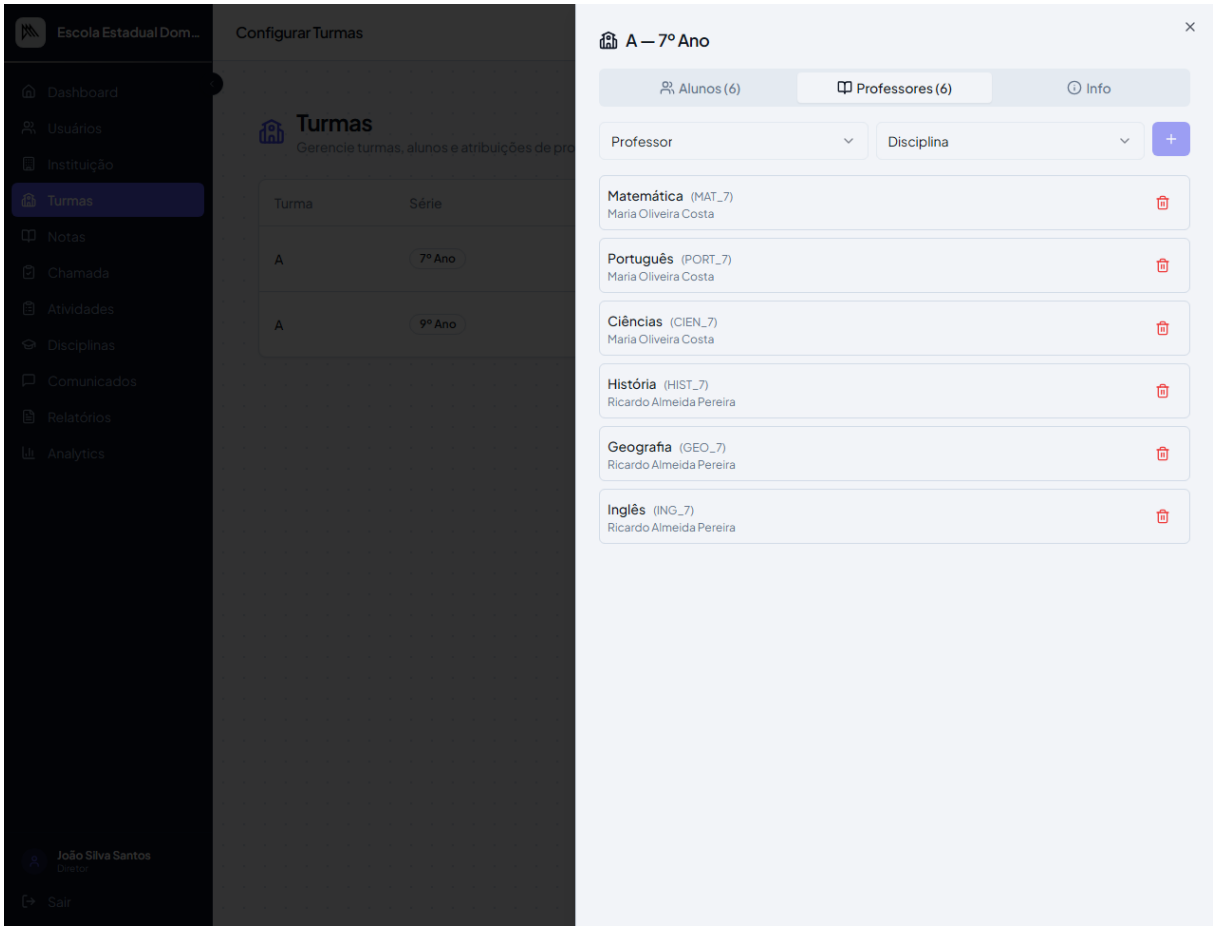
[Figura 10 — Listagem de turmas do ano letivo vigente] (Fonte: elaborado pelo autor)



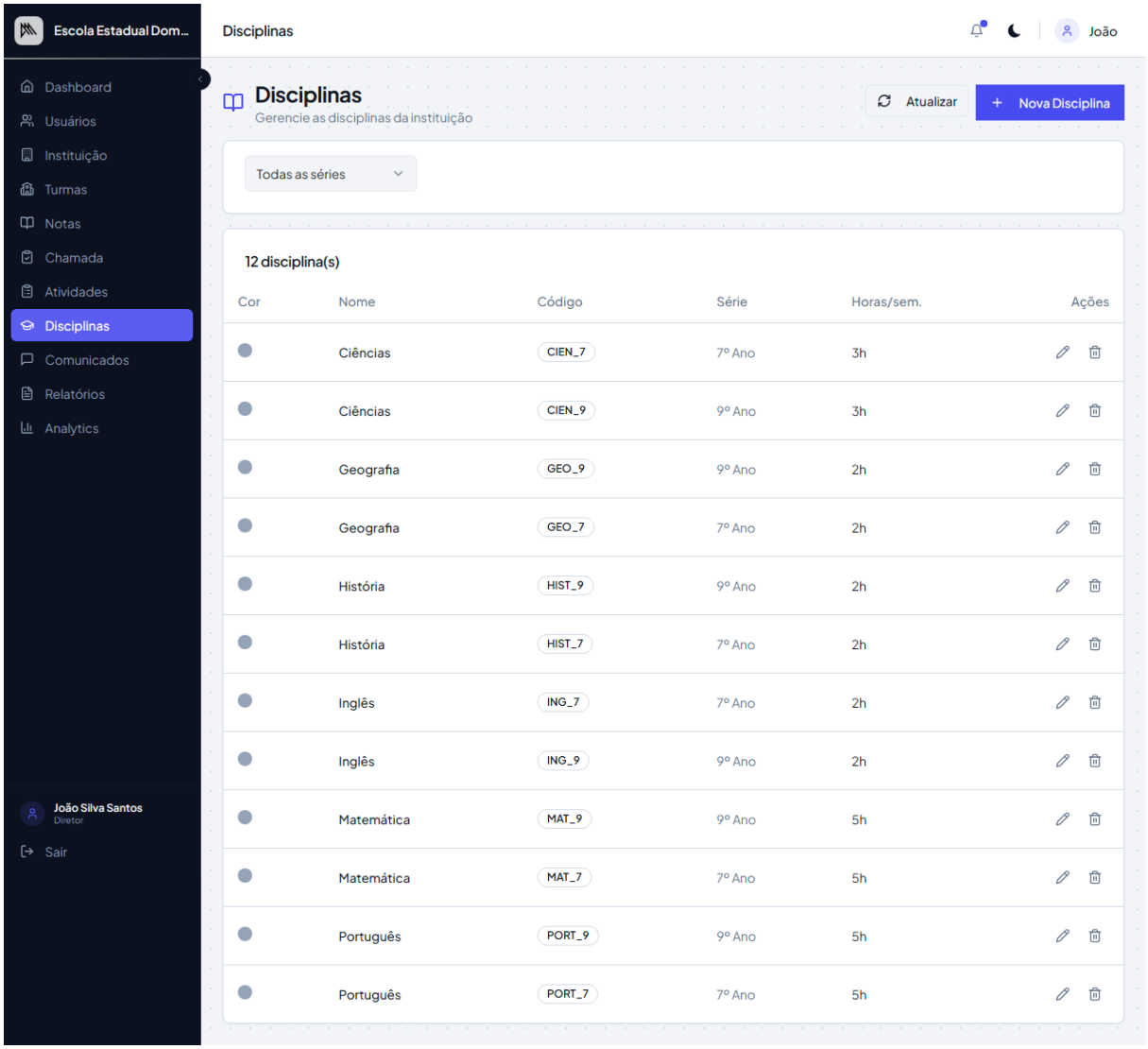
[Figura 11 — Formulário de criação de nova turma] (Fonte: elaborado pelo autor)



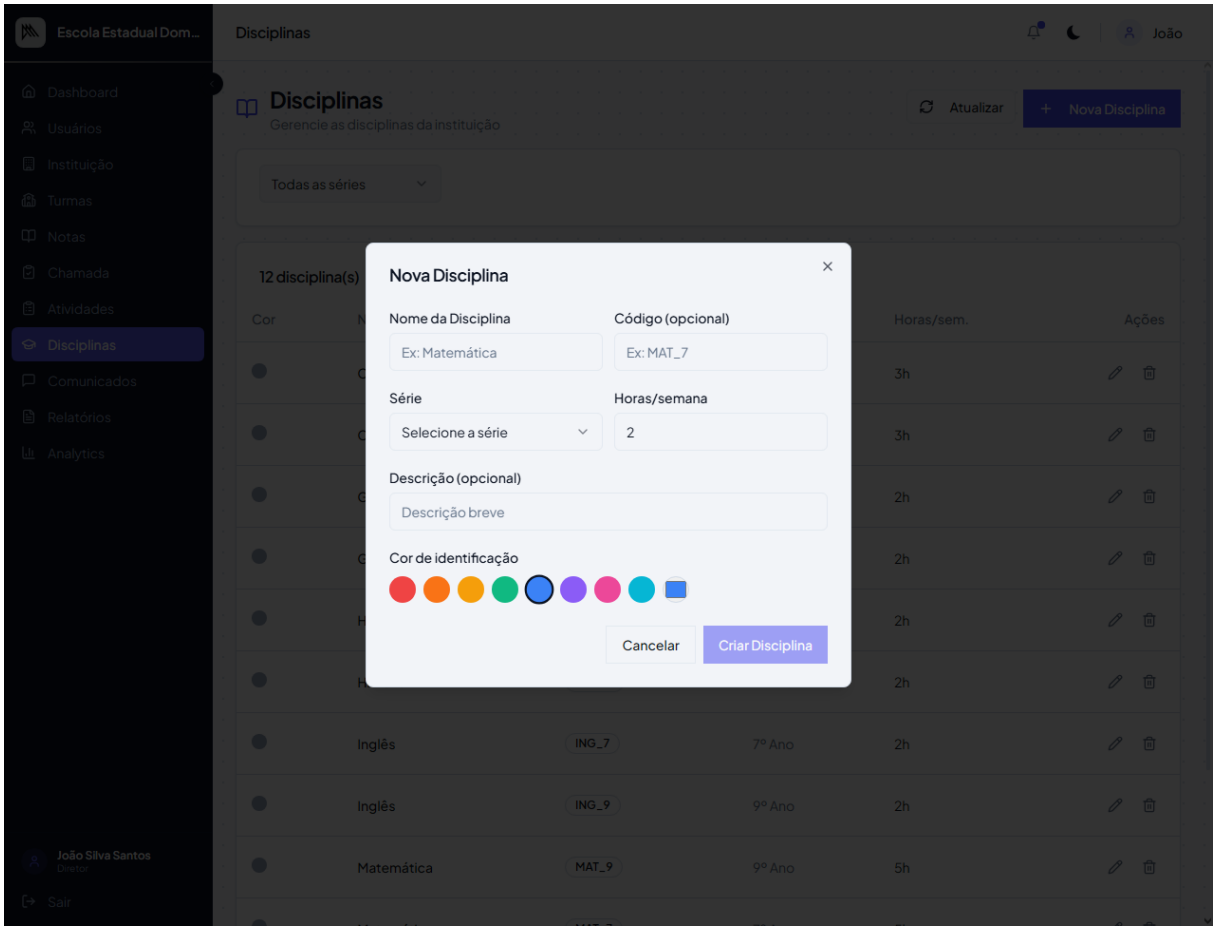
[Figura 12 — Modal de adição de aluno à turma] (Fonte: elaborado pelo autor)



[Figura 13 — Atribuição de professor a disciplina em uma turma] (Fonte: elaborado pelo autor)



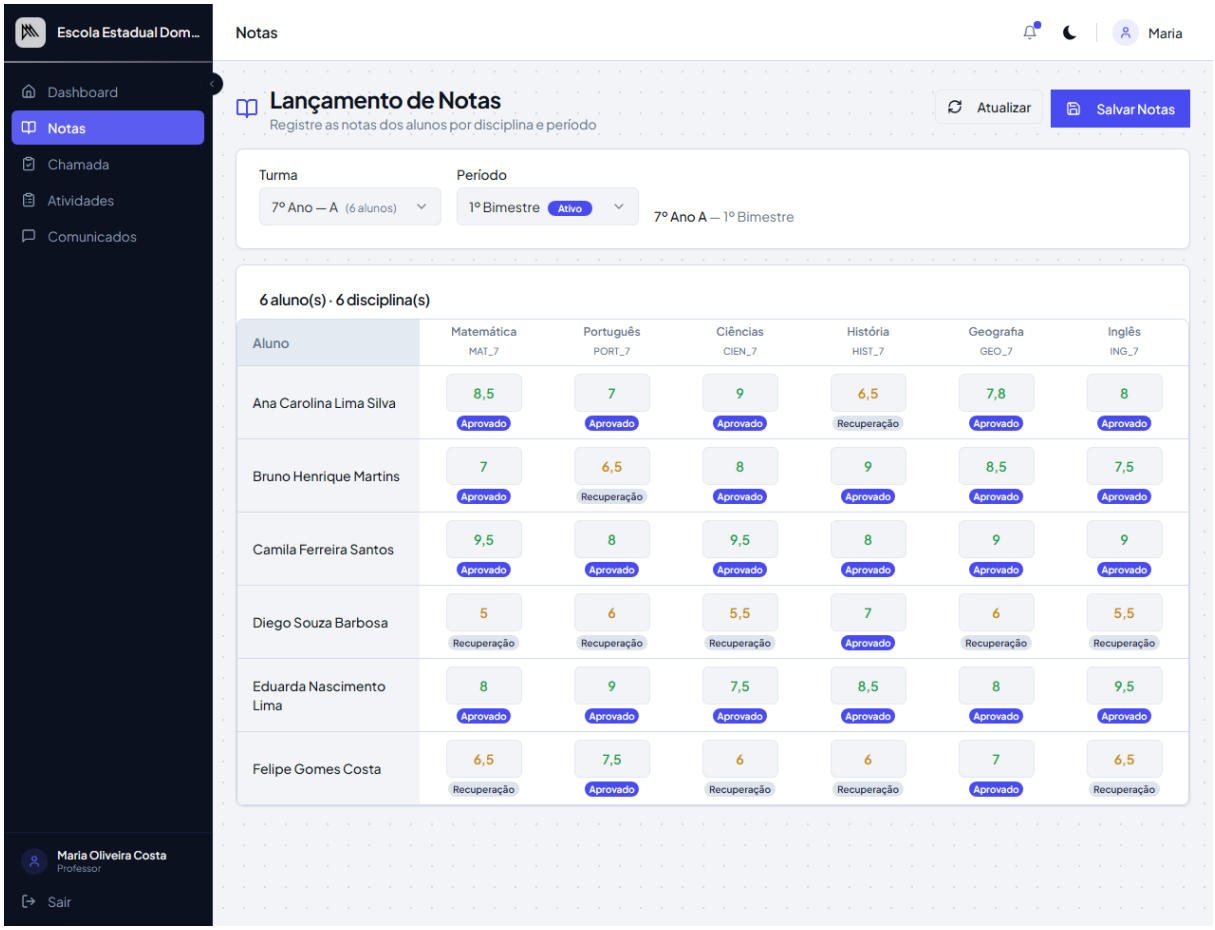
[Figura 14 — Listagem de disciplinas cadastradas] (Fonte: elaborado pelo autor)



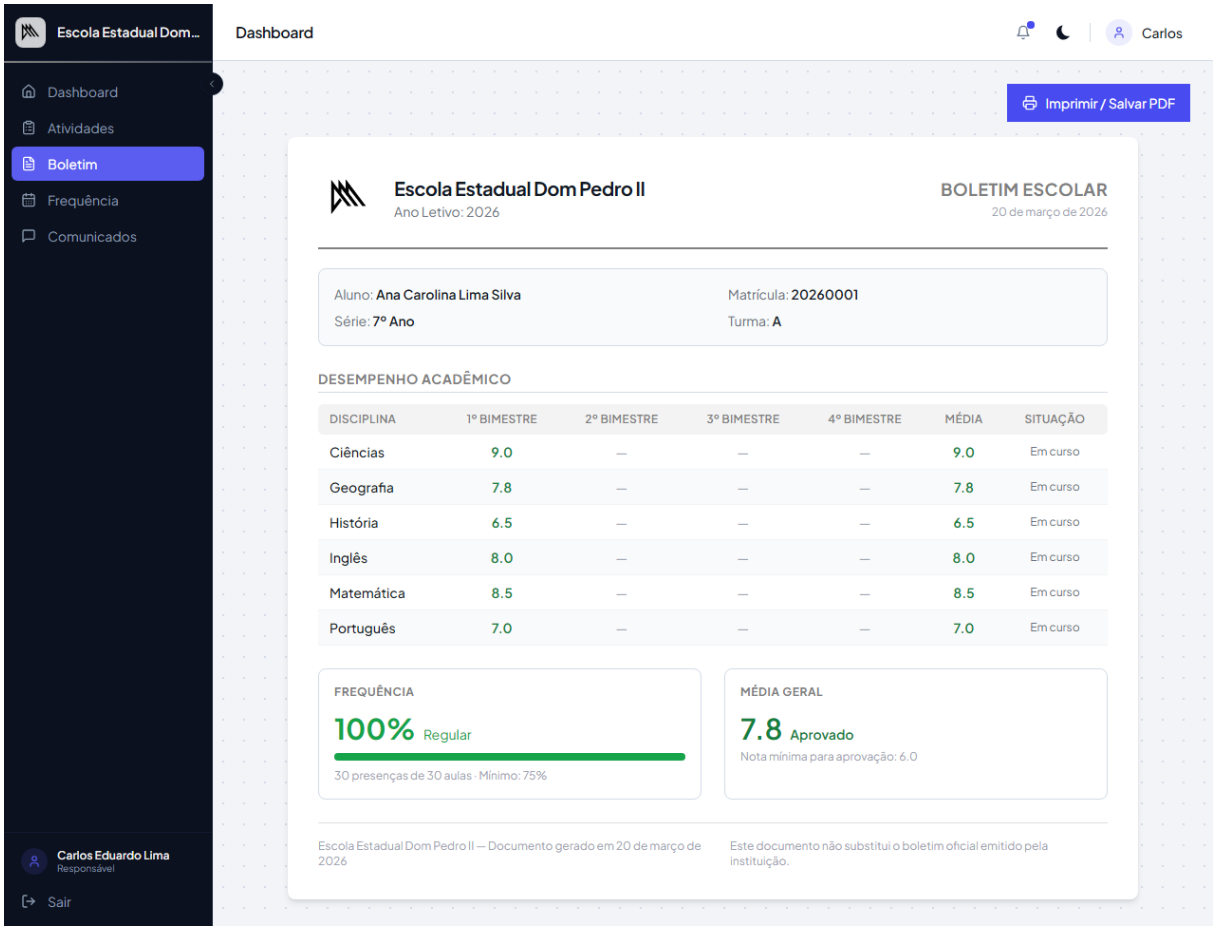
[Figura 15 — Modal de adição de disciplinas] (Fonte: elaborado pelo autor)

8.1.5 Avaliações e Notas

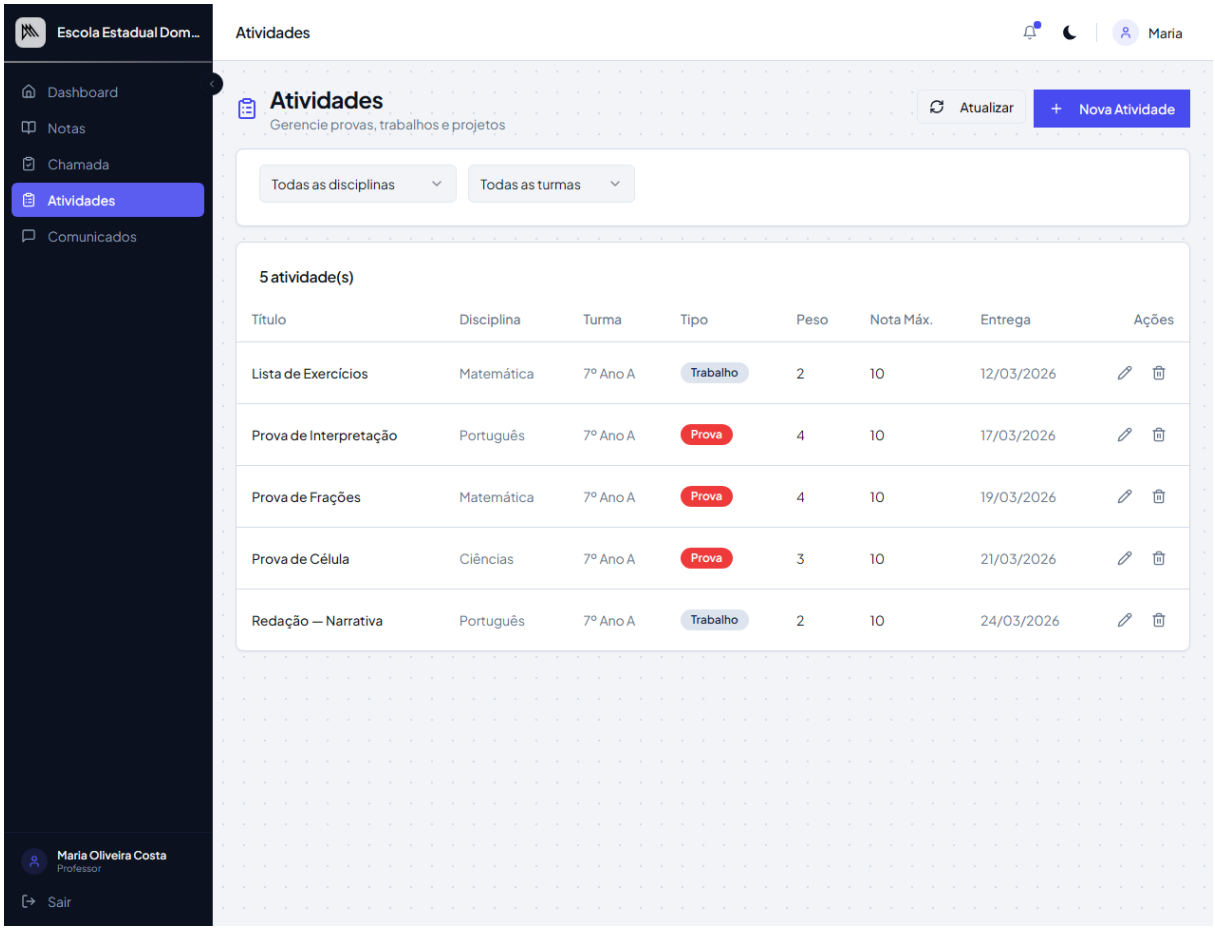
O módulo de avaliações e notas permite ao Professor lançar notas individuais por disciplina e ao Aluno e Responsável acompanharem o boletim com médias e situação de aprovação.



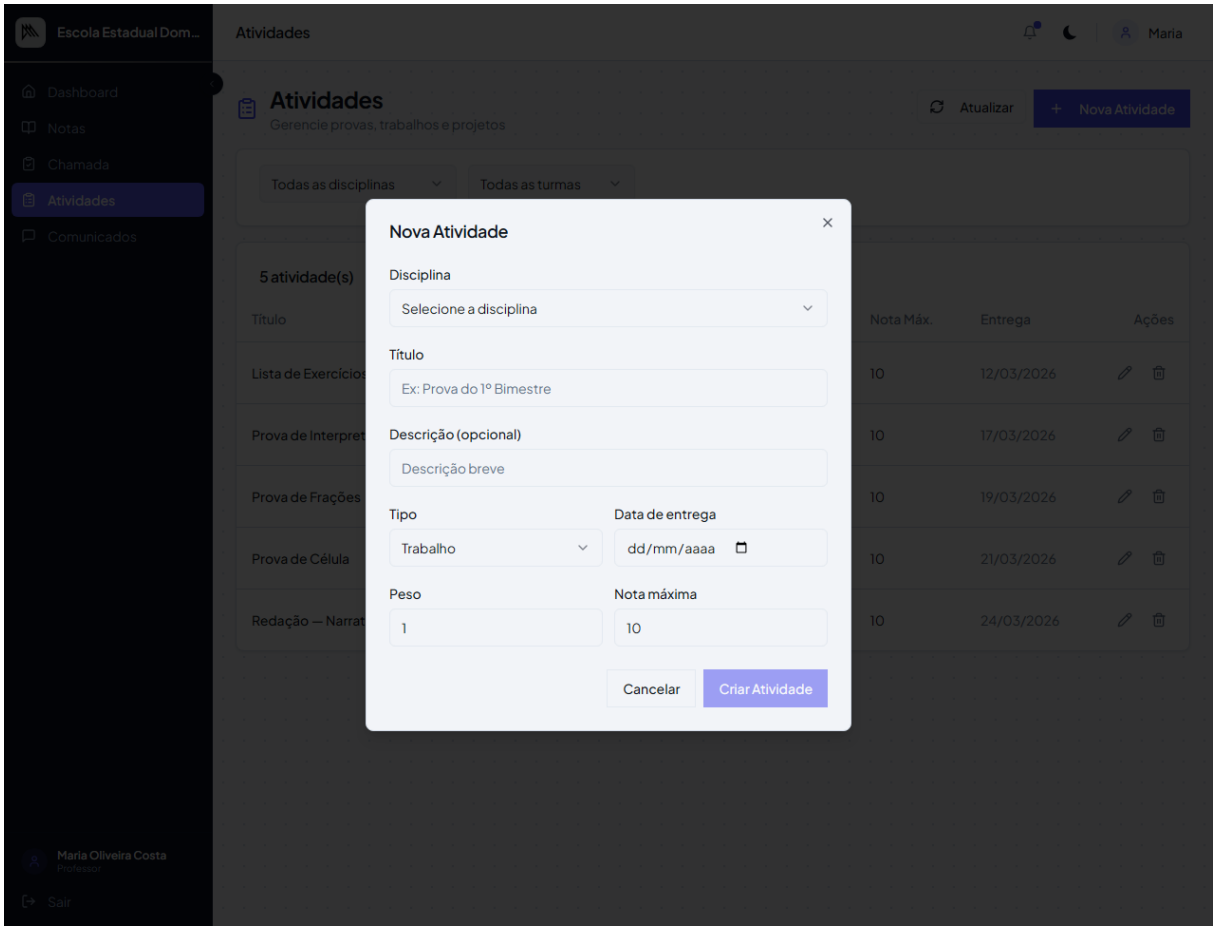
[Figura 16 — Tela de lançamento de notas pelo Professor] (Fonte: elaborado pelo autor)



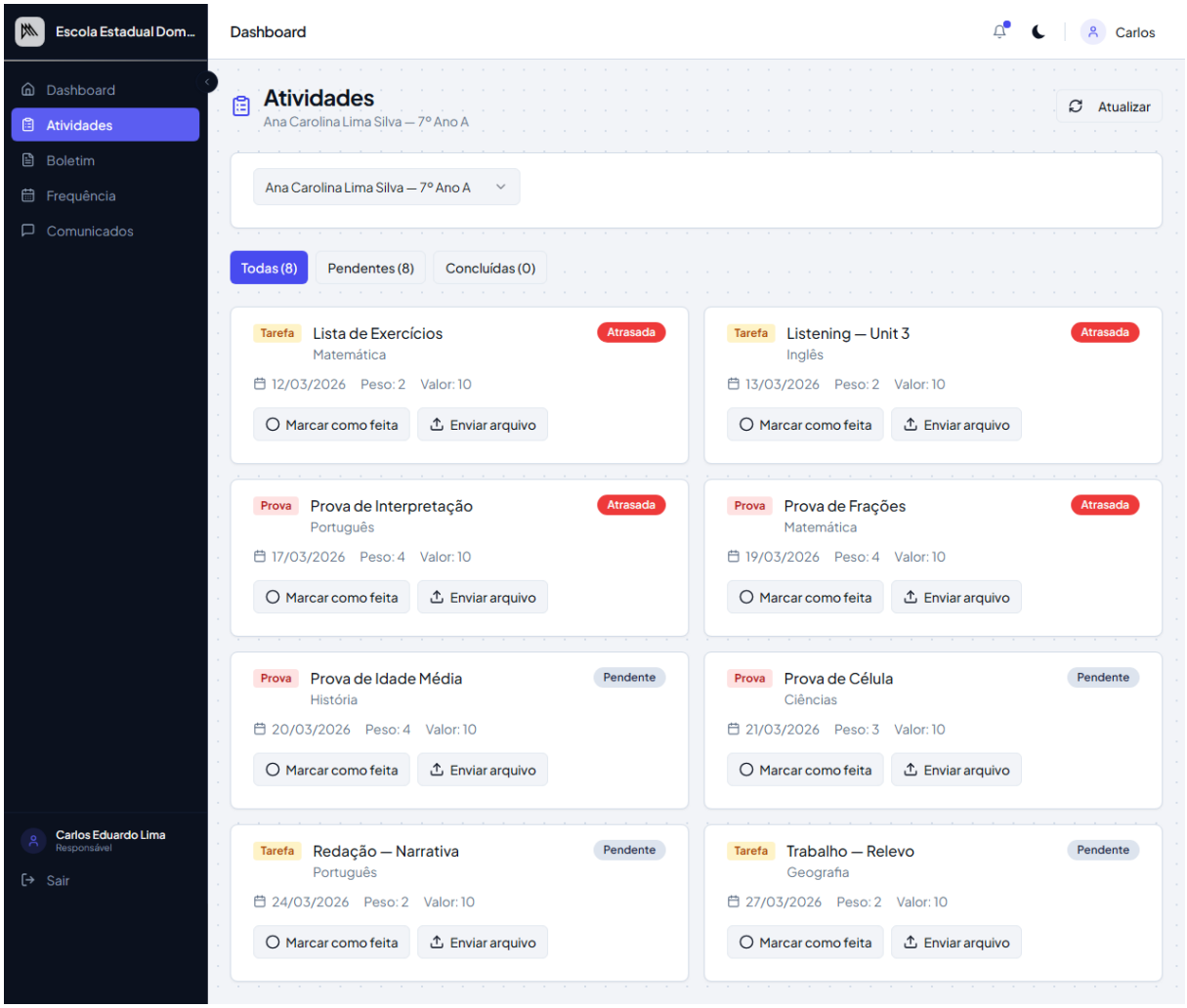
[Figura 17 — Boletim do Aluno com notas por disciplina e situação de aprovação] (Fonte: elaborado pelo autor)



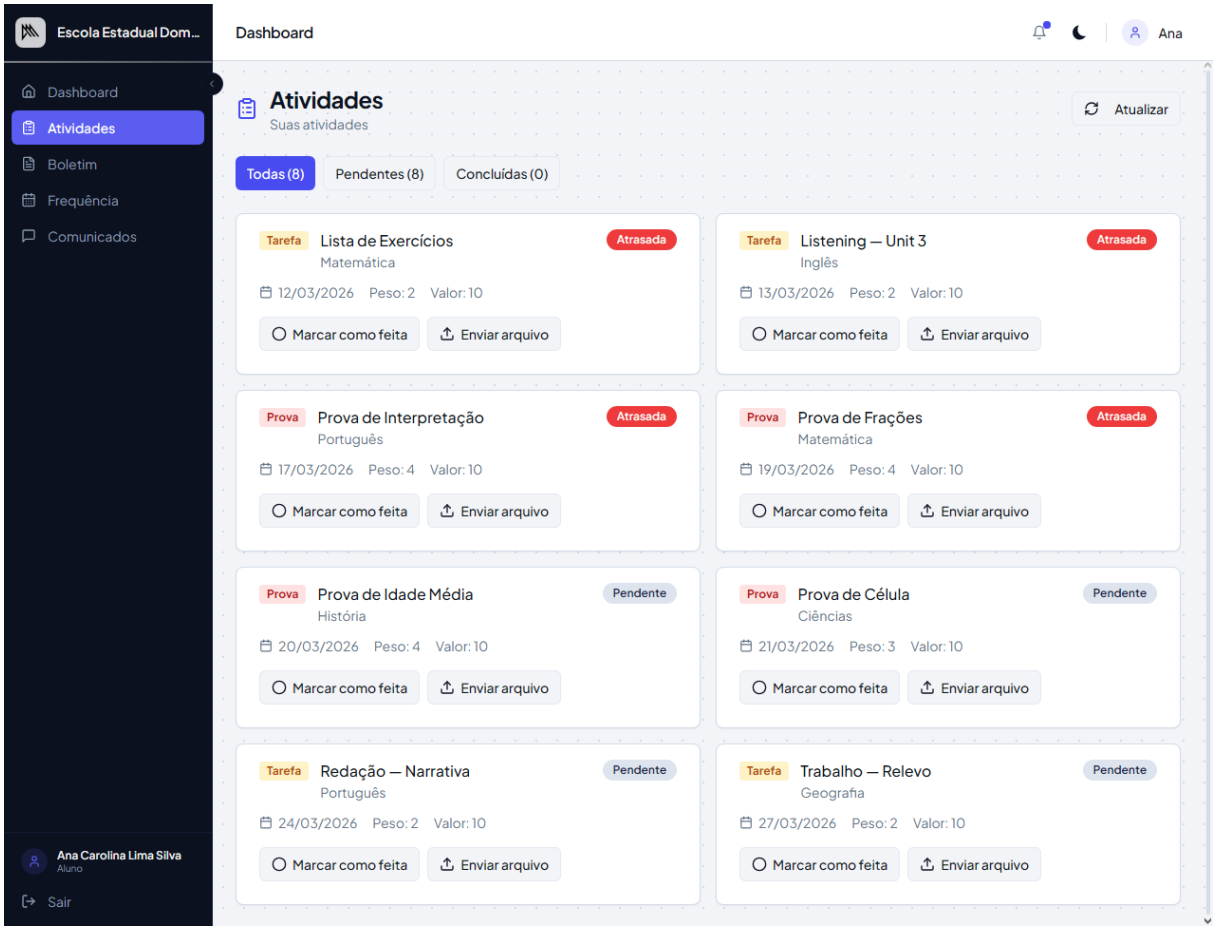
[Figura 18 — Listagem de atividades avaliativas criadas pelo Professor] (Fonte: elaborado pelo autor)



[Figura 19 — Formulário de criação de nova atividade avaliativa] (Fonte: elaborado pelo autor)



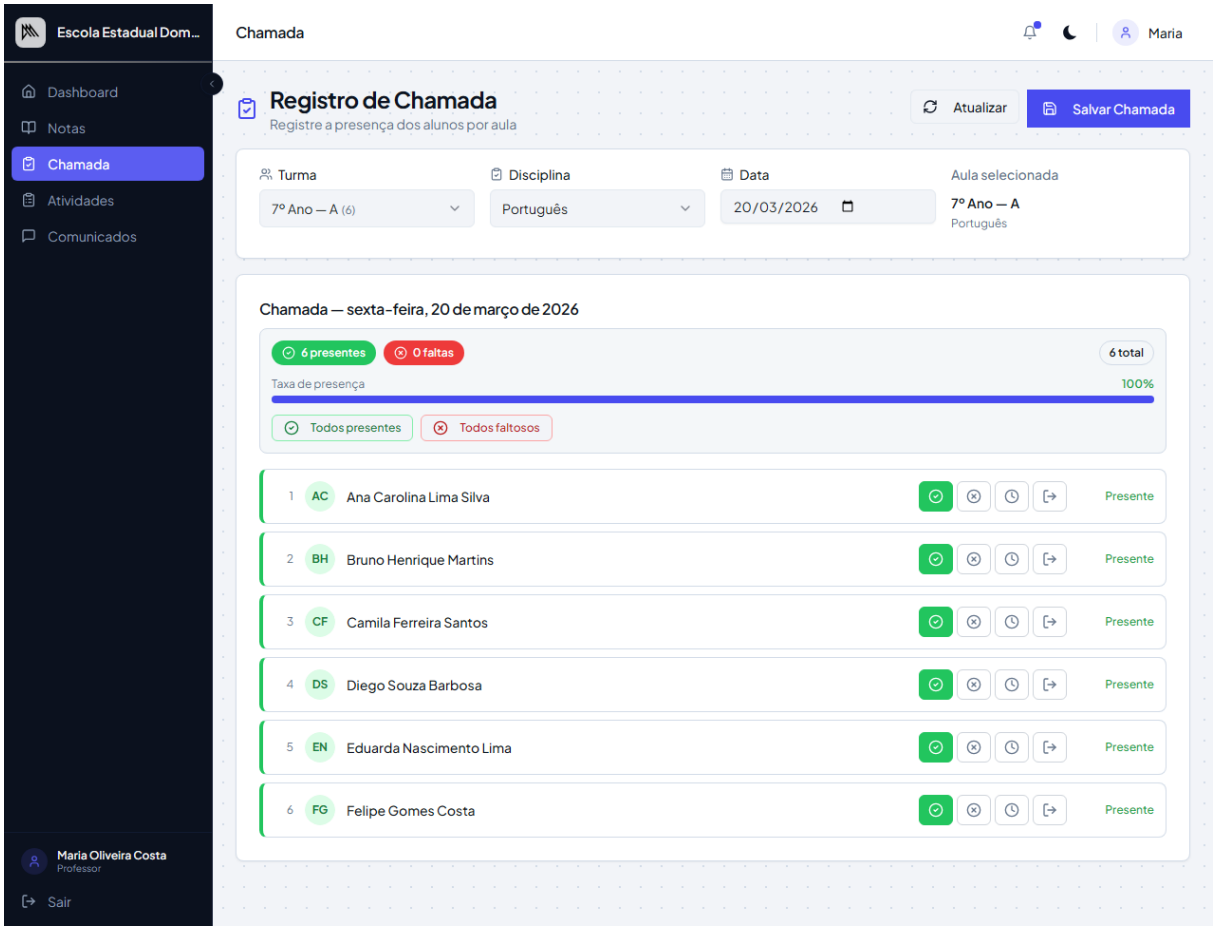
[Figura 20 — Listagem de atividades pendentes na visão do Responsável]



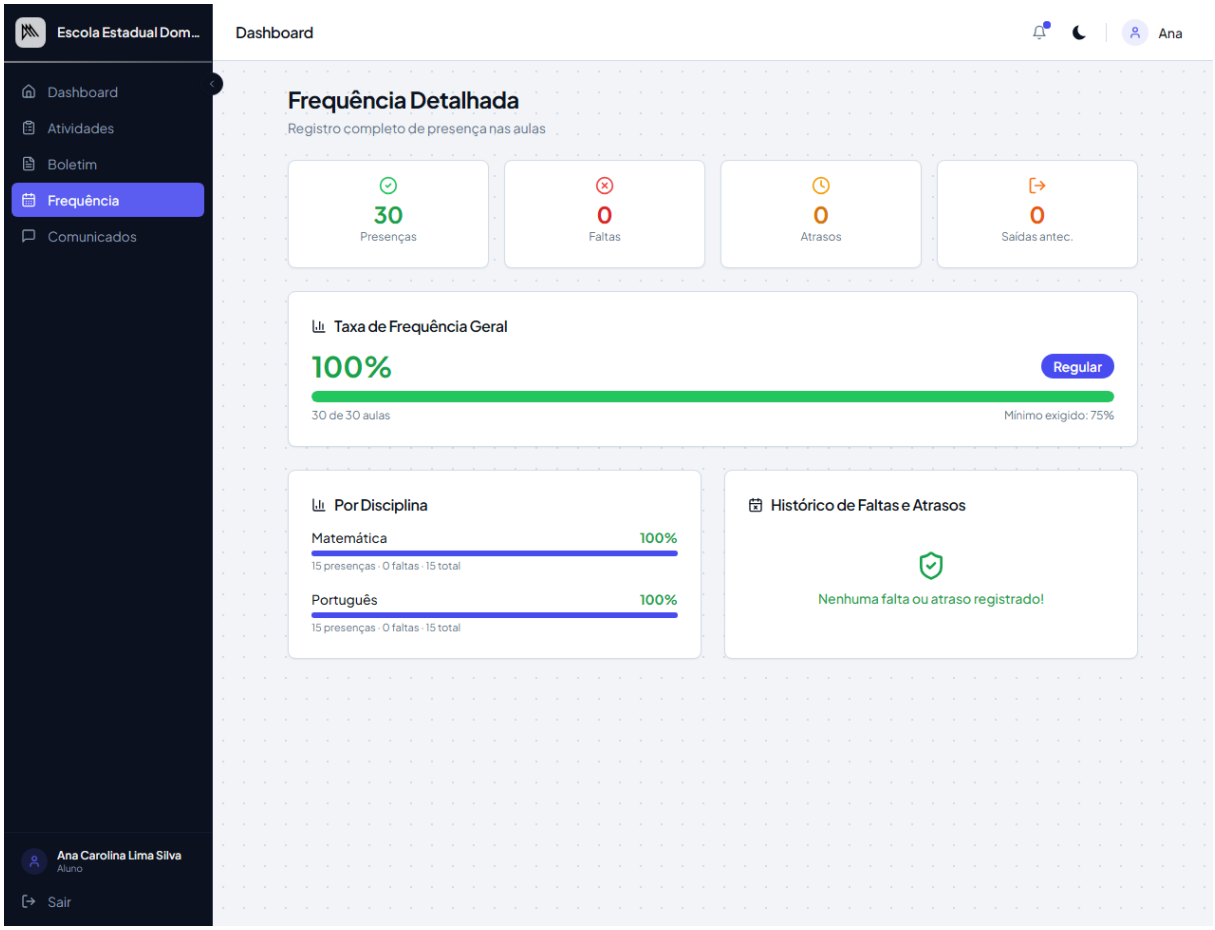
[Figura 21 — Listagem de atividades pendentes na visão do Aluno] (Fonte: elaborado pelo autor)

8.1.6 Frequência

O módulo de frequência permite ao Professor registrar a chamada diária e ao Aluno e Responsável acompanharem os percentuais de presença por disciplina.

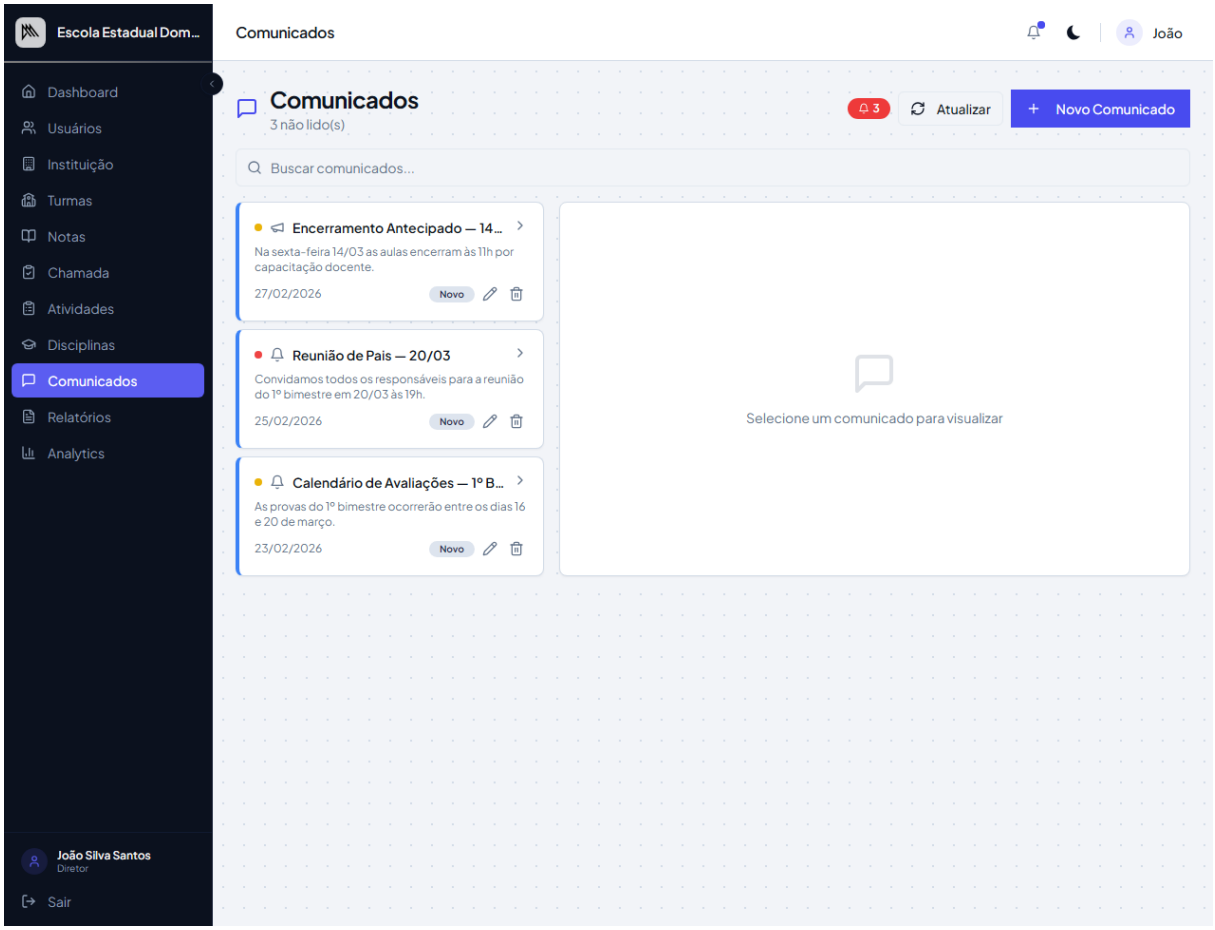


[Figura 22 — Tela de registro de chamada pelo Professor] (Fonte: elaborado pelo autor)

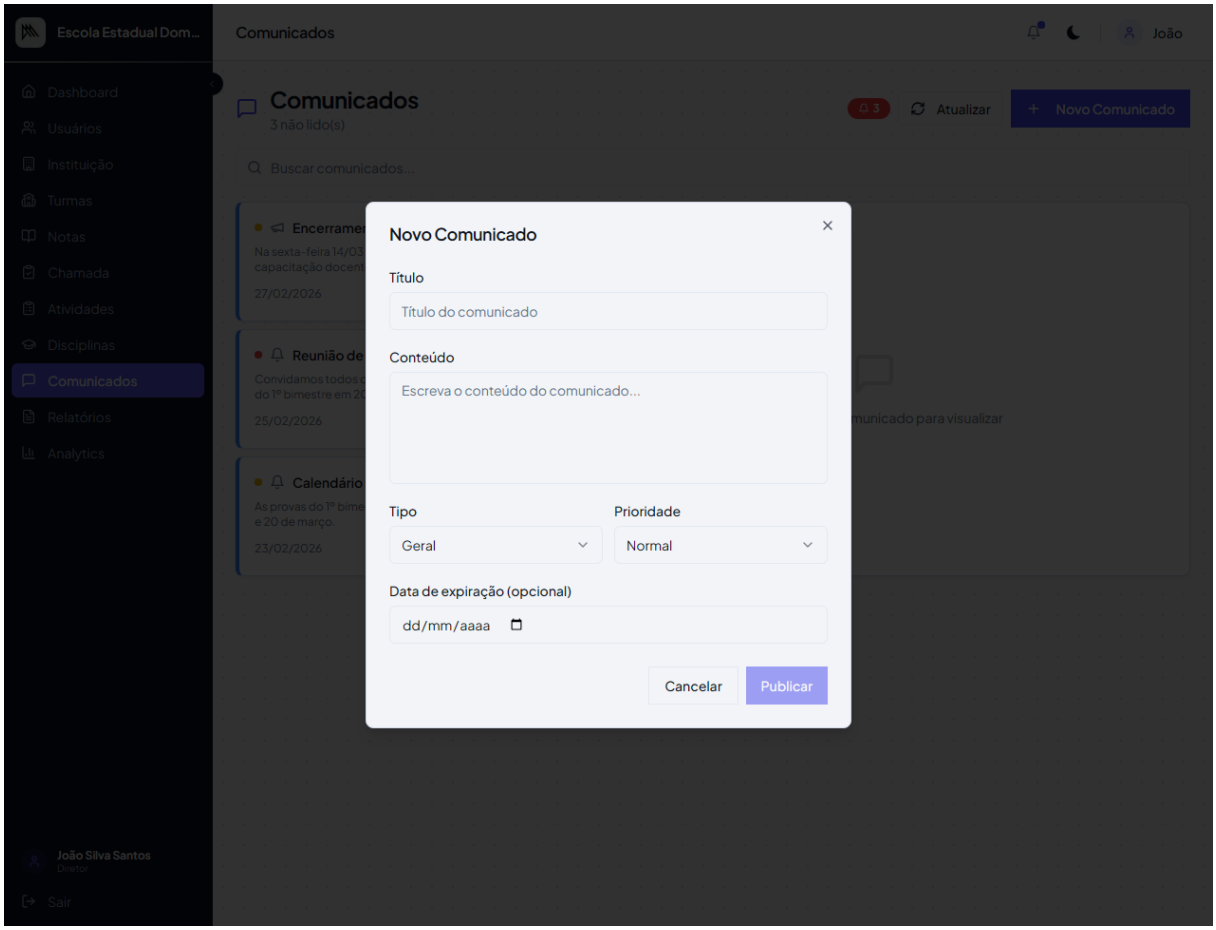


[Figura 23 — Consulta de frequência pelo Aluno com indicadores por disciplina] (Fonte: elaborado pelo autor)

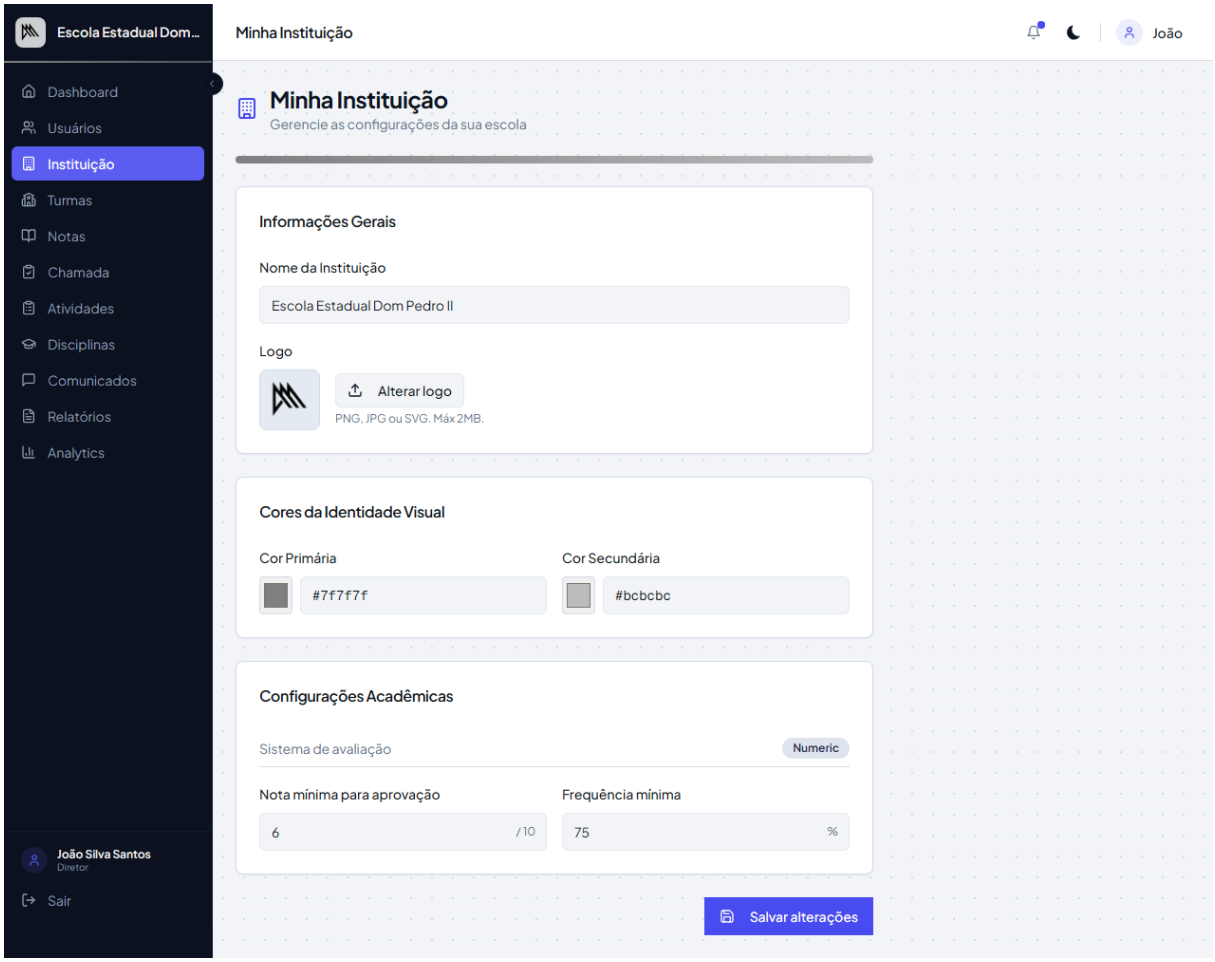
8.1.7 Comunicados e Configurações



[Figura 24 — Listagem de comunicados institucionais] (Fonte: elaborado pelo autor)

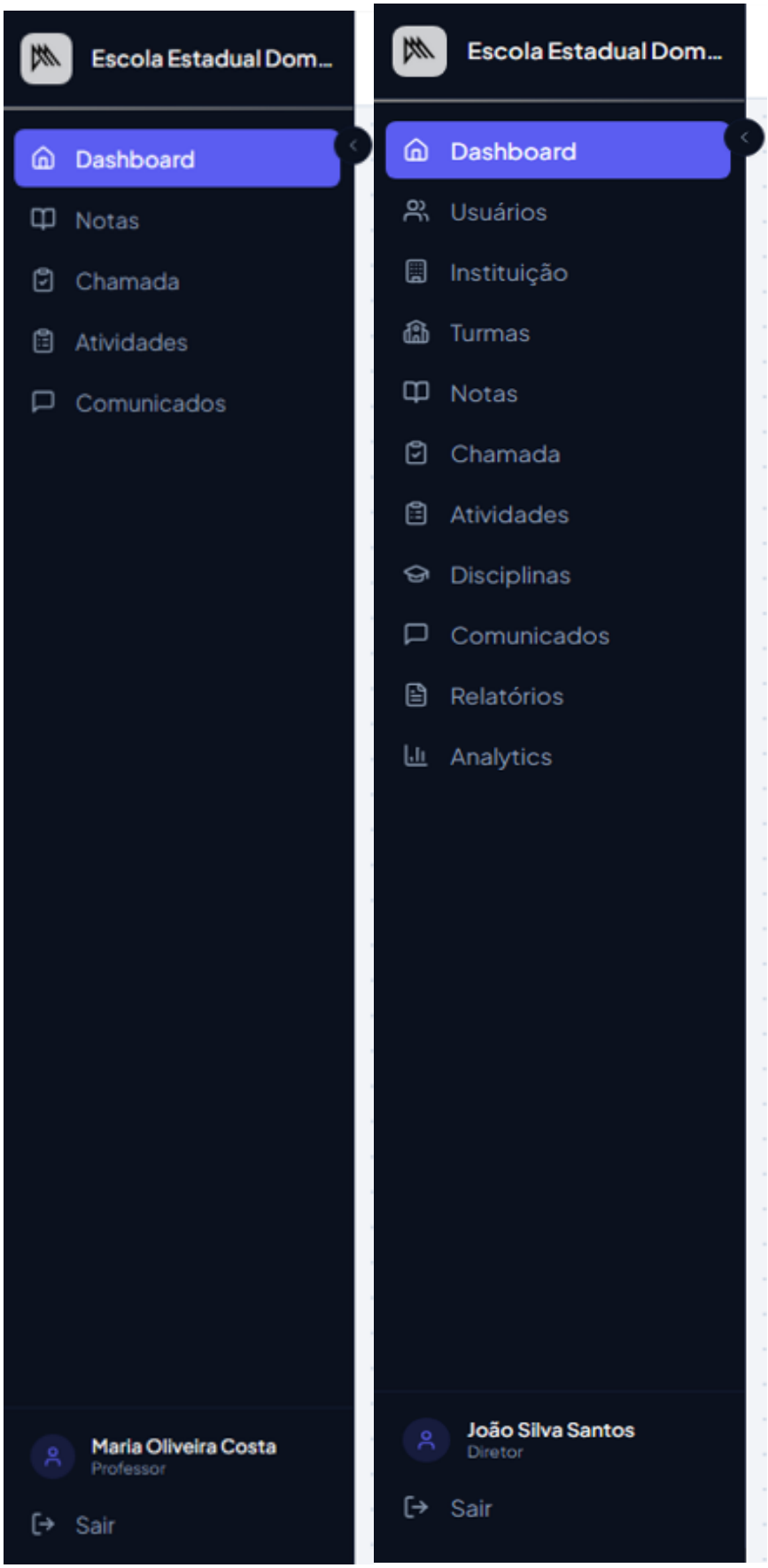


[Figura 25 — Formulário de criação de comunicado pelo Diretor] (Fonte: elaborado pelo autor)

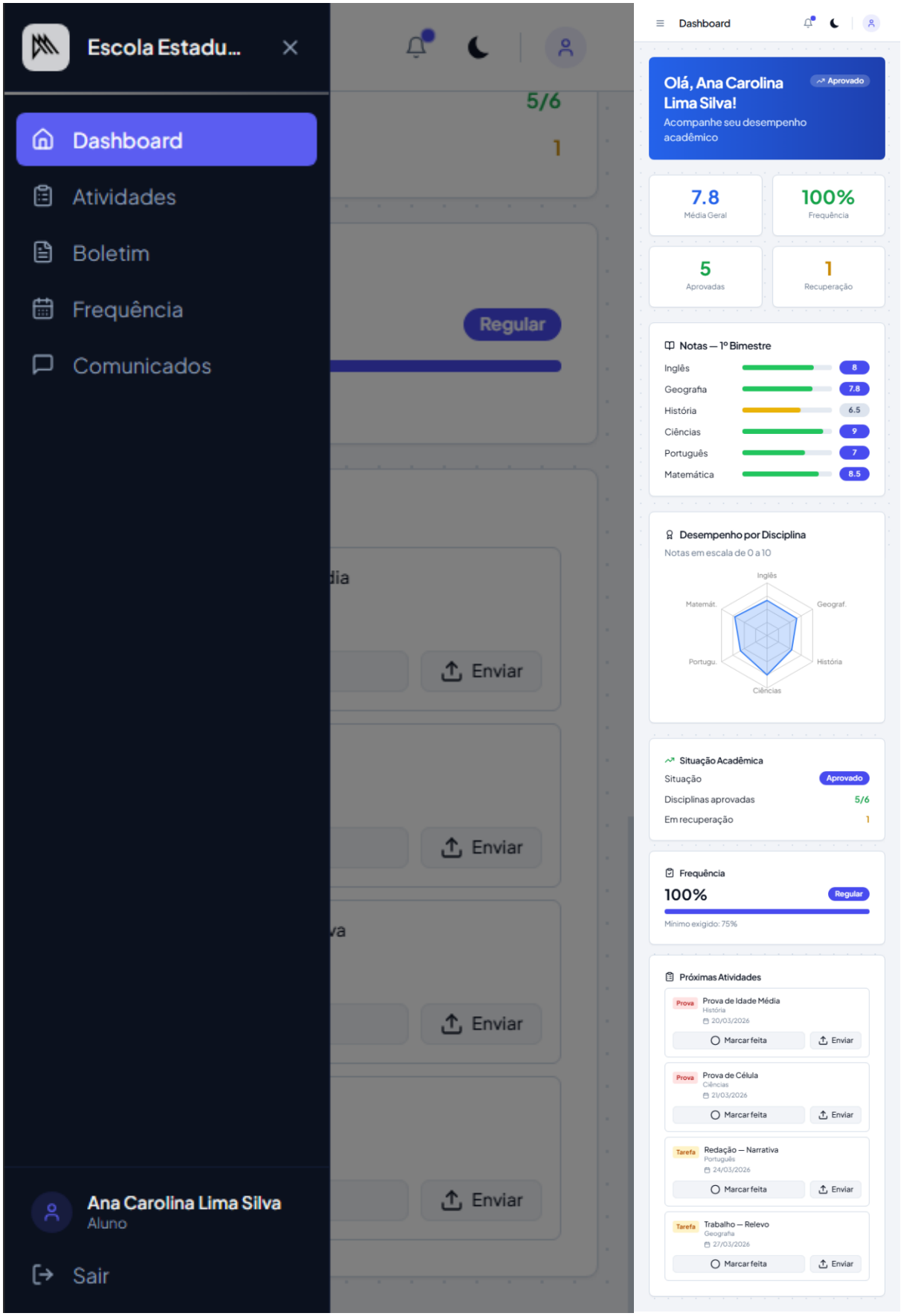


[Figura 26 — Página de configurações da instituição] (Fonte: elaborado pelo autor)

8.1.8 Visão Geral da Interface


Escola Estadual Dom...
Dashboard
Notas
Chamada
Atividades
Comunicados
Maria Oliveira Costa
Professor
Sair
Escola Estadual Dom...
Dashboard
Usuários
Instituição
Turmas
Notas
Chamada
Atividades
Disciplinas
Comunicados
Relatórios
Analytics
João Silva Santos
Diretor
Sair

[Figura 27 — Barra lateral de navegação com menu personalizado por perfil] (Fonte: elaborado pelo autor)


Escola Estadu...
Dashboard
Atividades
Boletim
Frequência
Comunicados
Ana Carolina Lima Silva
Aluno
Sair
5/6
1
Regular
Enviar
Enviar
Enviar
Enviar
Dashboard
Olá, Ana Carolina Lima Silva!
Aprovado
Acompanhe seu desempenho acadêmico
7.8
Média Geral
100%
Frequência
5
Aprovadas
1
Recuperação
Notas — 1º Bimestre
Inglês 8
Geografia 7.8
História 6.5
Ciências 9
Português 7
Matemática 8.5
Desempenho por Disciplina
Notas em escala de 0 a 10
Inglês
Geograf.
História
Ciências
Portugu.
Matemát.
Situação Acadêmica
Situação Aprovado
Disciplinas aprovadas 5/6
Em recuperação 1
Frequência
100%
Regular
Mínimo exigido: 75%
Próximas Atividades
Prova Prova de Idade Média
História
20/03/2026
Marcar feita
Enviar
Prova Prova de Célula
Ciências
21/03/2026
Marcar feita
Enviar
Tarefa Redação — Narrativa
Português
24/03/2026
Marcar feita
Enviar
Tarefa Trabalho — Relevo
Geografia
27/03/2026
Marcar feita
Enviar

[Figura 28 — Sistema em modo de visualização móvel (responsividade)] (Fonte: elaborado pelo autor)

A Figura 28 demonstra que o sistema mantém sua funcionalidade e legibilidade mesmo em telas de tamanho reduzido, atendendo ao Requisito Não Funcional RNF08, que determina que a interface deve ser responsiva para dispositivos móveis e desktops.

8.2 BENEFÍCIOS DO SISTEMA PARA A INSTITUIÇÃO DE ENSINO

O desenvolvimento e a validação do Schoolink evidenciaram um conjunto de benefícios concretos que a plataforma pode oferecer às instituições de ensino que a adotarem.

8.2.1 Centralização das Informações Acadêmicas

Antes de um sistema como o Schoolink, as informações de uma escola costumam estar dispersas em diferentes meios: notas em planilhas, frequência em diários em papel, comunicados em grupos de aplicativos de mensagem e dados de matrícula em sistemas isolados. O Schoolink centraliza todas essas informações em uma única plataforma, acessível de qualquer dispositivo conectado à internet.

Essa centralização reduz o risco de perda de informações, elimina inconsistências causadas pela duplicação de dados em diferentes sistemas e facilita o acesso ao histórico acadêmico dos alunos ao longo dos anos letivos.

8.2.2 Transparência para as Famílias

Um dos benefícios mais imediatos identificados durante o desenvolvimento foi a possibilidade de as famílias acompanharem o desempenho dos filhos em tempo real, sem depender de boletins impressos enviados ao final de cada bimestre.

Com o Schoolink, um responsável pode verificar a nota do filho imediatamente após o lançamento pelo professor, visualizar o histórico de presenças e ausências por disciplina e receber comunicados institucionais diretamente na plataforma. Essa transparência contribui para um vínculo mais estreito entre a escola e as famílias, o que, segundo Ferreira (2012), está diretamente relacionado a melhores resultados acadêmicos dos alunos.

8.2.3 Redução da Carga Administrativa sobre Professores

O lançamento de notas e o registro de frequência em sistemas digitais integrados eliminam o retrabalho de transferir informações de registros físicos para sistemas digitais separados. O professor realiza o lançamento uma única vez, e o sistema calcula automaticamente médias, percentuais de frequência e situações de aprovação.

Isso representa uma economia de tempo significativa, especialmente em períodos de fechamento de bimestre, quando professores com muitas turmas precisam lançar um grande volume de avaliações em pouco tempo.

8.2.4 Apoio à Gestão Baseada em Dados

O painel do Diretor concentra indicadores atualizados em tempo real sobre a situação da instituição. Com acesso imediato a informações como o total de alunos matriculados, a distribuição por turma e o desempenho acadêmico geral, o gestor escolar pode identificar rapidamente situações que demandam atenção — como turmas com alto índice de reprovação ou disciplinas com frequência abaixo do mínimo — e tomar decisões pedagógicas mais embasadas (LAUDON; LAUDON, 2021).

8.3 COMPARAÇÃO COM PROCESSOS MANUAIS

Para ilustrar de forma objetiva os ganhos que o Schoolink oferece, o quadro a seguir compara processos comuns da rotina escolar antes e depois da adoção da plataforma:

Processo: Lançamento de notas

Antes: Professor preenche diário físico; secretaria transfere para planilha; boletim impresso é entregue ao aluno ao final do bimestre.

Com o Schoolink: Professor lança diretamente no sistema; média calculada automaticamente; aluno e responsável visualizam em tempo real.

Processo: Registro de frequência

Antes: Professor preenche lista de chamada em papel; dados podem ser perdidos ou ilegíveis; cálculo de percentual feito manualmente.

Com o Schoolink: Chamada registrada digitalmente; percentual calculado automaticamente; alerta visual acionado quando aluno se aproxima do limite de faltas.

Processo: Comunicação com as famílias

Antes: Bilhetes impressos na agenda escolar; grupos informais em aplicativos de mensagem; reuniões presenciais periódicas.

Com o Schoolink: Comunicados publicados na plataforma com prioridade definida; visíveis a todos os responsáveis imediatamente após a publicação.

Processo: Gestão de matrículas

Antes: Formulários físicos; planilhas de controle; risco de duplicidade ou perda de dados.

Com o Schoolink: Matrícula registrada digitalmente; sistema impede duplicidade automaticamente; histórico preservado para consulta futura.

Processo: Acompanhamento do desempenho pelo responsável

Antes: Dependente de boletins bimestrais impressos ou de contato presencial com a escola.

Com o Schoolink: Responsável acessa o boletim e a frequência do filho a qualquer momento, de qualquer dispositivo com acesso à internet.

A comparação evidencia que o Schoolink não apenas digitaliza processos existentes, mas transforma a dinâmica de comunicação e acompanhamento acadêmico entre os diferentes atores da comunidade escolar.

8.4 LIMITAÇÕES DO SISTEMA

O desenvolvimento do Schoolink como Trabalho de Conclusão de Curso implicou escolhas e restrições que resultam em limitações funcionais e técnicas que devem ser reconhecidas com transparência.

8.4.1 Ausência de Testes Automatizados

Os testes realizados foram de natureza funcional e manual. A ausência de uma suíte de testes automatizados (unitários e de integração) representa um risco para a manutenção do sistema a longo prazo, pois alterações futuras no código podem introduzir regressões que não seriam detectadas automaticamente.

8.4.2 Sistema Limitado a Uma Instituição por Implantação

Embora o modelo de dados do Schoolink suporte múltiplas instituições com isolamento completo de dados, a infraestrutura de implantação atual não foi projetada para operação em escala de múltiplos clientes (modelo SaaS — Software as a Service). A implantação para uma nova instituição requer configuração manual do ambiente.

8.4.3 Ausência de Módulo de Relatórios

O sistema não conta com um módulo de geração de relatórios exportáveis (PDF ou planilha). Diretores e professores conseguem visualizar as informações na plataforma, mas não podem exportá-las de forma estruturada para uso fora do sistema.

8.4.4 Autenticação de Dois Fatores Não Habilitada em Produção

O sistema possui a biblioteca Speakeasy integrada para autenticação de dois fatores (2FA), mas essa funcionalidade não foi completamente implementada na interface do usuário durante o período de desenvolvimento do trabalho. Sua ativação está prevista como melhoria futura.

8.4.5 Não Validado em Ambiente de Produção Real

O sistema foi desenvolvido e validado em ambiente local de desenvolvimento. Sua performance em condições reais de uso — com múltiplos usuários simultâneos, infraestrutura de produção e dados reais de uma instituição — não foi avaliada neste trabalho.

9 CONCLUSÃO E TRABALHOS FUTUROS

9.1 Conclusão

Este trabalho teve como objetivo o desenvolvimento do Schoolink, um sistema web de gerenciamento escolar que centraliza as principais atividades administrativas e pedagógicas de uma instituição de ensino em uma única plataforma digital, acessível por quatro perfis de usuário distintos: Diretor, Professor, Aluno e Responsável.

O ponto de partida foi a identificação de um problema real e recorrente em escolas de pequeno e médio porte: a dispersão de informações acadêmicas em meios heterogêneos — planilhas, diários

físicos, grupos de aplicativos de mensagens e arquivos isolados — que dificulta a gestão eficiente, compromete a comunicação entre escola e família e gera retrabalho para professores e gestores.

A partir desse problema, foram definidos os objetivos específicos do trabalho: desenvolver um módulo de autenticação com controle de acesso baseado em perfis (RBAC), implementar funcionalidades de lançamento de notas e registro de frequência, oferecer painéis personalizados por perfil, construir um canal de comunicados institucionais e disponibilizar ferramentas de configuração da instituição para o Diretor.

Ao final do desenvolvimento, todos os objetivos específicos foram alcançados. Os 31 Requisitos Funcionais especificados foram implementados e validados por meio de 27 cenários de teste, todos aprovados. As 21 Regras de Negócio definidas foram verificadas durante os testes, com comportamento conforme o especificado. Os principais Requisitos Não Funcionais — tempo de resposta, responsividade, segurança e usabilidade — foram atendidos dentro das condições do ambiente de desenvolvimento.

Do ponto de vista técnico, o trabalho demonstrou a viabilidade de construir um sistema web de gestão escolar funcional utilizando tecnologias modernas de código aberto. A combinação de Next.js no frontend com Express.js no backend, integrados por uma API REST com autenticação JWT, mostrou-se adequada para o domínio do problema. O uso do Prisma como ORM sobre o banco de dados PostgreSQL simplificou o desenvolvimento sem sacrificar a integridade dos dados, e o modelo de controle de acesso em duas camadas — frontend e backend — garantiu que as restrições de perfil fossem aplicadas de forma robusta.

Do ponto de vista pedagógico, o desenvolvimento do Schoolink proporcionou a aplicação prática de conceitos estudados ao longo do curso: engenharia de requisitos, modelagem de banco de dados relacional, arquitetura em camadas, segurança de aplicações web, padrões de projeto e usabilidade. A elaboração deste documento também exigiu a articulação entre prática técnica e fundamentação teórica, integrando referências das áreas de sistemas de informação, engenharia de software, segurança da informação e gestão educacional.

Reconhece-se que o sistema, em seu estado atual, representa um protótipo funcional validado em ambiente de desenvolvimento. As limitações identificadas no Capítulo 8 — ausência de testes automatizados, falta de módulo de relatórios, 2FA incompleto e não validação em produção real — são características esperadas de um trabalho acadêmico de conclusão de curso e não invalidam os resultados obtidos nem os objetivos propostos.

Conclui-se que o Schoolink cumpre o propósito para o qual foi concebido: demonstrar, por meio de um sistema funcional e documentado, que é possível desenvolver uma plataforma de gestão escolar completa, segura e responsiva utilizando tecnologias acessíveis e práticas modernas de engenharia de software.

9.2 TRABALHOS FUTUROS

O desenvolvimento do Schoolink abriu um conjunto de possibilidades de melhoria e expansão que, por restrições de escopo e tempo inerentes a um Trabalho de Conclusão de Curso, não puderam ser implementadas nesta versão. Esta seção documenta as principais evoluções identificadas como trabalhos futuros.

9.2.1 Implementação de Testes Automatizados

A ausência de testes automatizados é a limitação técnica mais relevante identificada neste trabalho. A implementação de uma suíte de testes unitários para as funções de negócio e de testes de integração para os endpoints da API é a evolução de maior prioridade para garantir a manutenibilidade do sistema a longo prazo.

Ferramentas como Jest e Supertest, amplamente utilizadas no ecossistema Node.js, permitiriam criar testes que verificam automaticamente o comportamento do sistema a cada alteração no código, reduzindo o risco de regressões e aumentando a confiança nas entregas futuras.

9.2.2 Migração para Modelo Multilocatário (SaaS)

O modelo de dados do Schoolink já suporta múltiplas instituições com isolamento completo de dados por meio do campo institutionId em todas as entidades relevantes. No entanto, a infraestrutura de implantação atual requer configuração manual para cada nova instituição.

Um trabalho futuro relevante seria a construção de um portal de onboarding que permita o cadastro e a configuração automática de novas instituições, transformando o Schoolink em um produto oferecido como serviço (SaaS — Software as a Service). Isso incluiria a automação da criação de subdomínios, a configuração de planos de assinatura e o desenvolvimento de um painel administrativo global separado do painel institucional.

9.2.3 Módulo de Relatórios e Exportação

A ausência de um módulo de exportação de dados foi identificada como uma limitação relevante para os perfis de Diretor e Professor. Como trabalho futuro, propõe-se o desenvolvimento de um módulo de relatórios que permita:

- Exportação do boletim individual do aluno em formato PDF
- Exportação de listas de frequência por turma e disciplina
- Geração de relatórios de desempenho por turma com médias e indicadores
- Exportação de listas de alunos matriculados

Bibliotecas como PDFKit ou Puppeteer poderiam ser utilizadas no backend para a geração dinâmica de documentos PDF a partir dos dados do sistema.

9.2.4 Ativação da Autenticação de Dois Fatores (2FA)

A biblioteca Speakeasy já está integrada ao backend do Schoolink, mas a interface de configuração e uso do 2FA não foi concluída durante o período de desenvolvimento deste trabalho. Como trabalho futuro, propõe-se a implementação completa do fluxo de 2FA, incluindo:

- Tela de configuração do 2FA no perfil do usuário (geração e exibição do QR Code)
- Validação do código TOTP (Time-based One-Time Password) no fluxo de login
- Opção de desativação e reconfiguração do 2FA pelo próprio usuário
- Política de obrigatoriedade do 2FA para o perfil de Diretor

Essa melhoria elevaria significativamente o nível de segurança da plataforma, especialmente para o perfil com maior nível de privilégio.

9.2.5 Testes de Carga e Validação em Ambiente de Produção

O sistema não foi avaliado em condições reais de uso com múltiplos usuários simultâneos. Como trabalho futuro, propõe-se a realização de testes de carga utilizando ferramentas como Apache JMeter ou k6, simulando o acesso concurrent de dezenas ou centenas de usuários para identificar gargalos de desempenho e validar o atendimento ao Requisito Não Funcional RNF02.

Além disso, a implantação do sistema em um ambiente de produção real — com configuração de servidor, certificado SSL, banco de dados gerenciado e pipeline de entrega contínua (CI/CD) —

permitiria avaliar aspectos operacionais que não puderam ser verificados no ambiente local de desenvolvimento.

9.2.6 Notificações em Tempo Real

O sistema atual exige que o usuário acesse a plataforma para verificar novos comunicados ou notas lançadas. Como trabalho futuro, propõe-se a implementação de um sistema de notificações em tempo real utilizando WebSockets ou Server-Sent Events (SSE), que alertaria automaticamente o usuário sobre eventos relevantes: nova nota lançada, novo comunicado publicado, frequência abaixo do mínimo atingida.

Esse recurso aumentaria o engajamento dos usuários com a plataforma e reduziria o tempo entre o lançamento de uma informação pelo professor ou diretor e o conhecimento dela pela família.

9.2.7 Aplicativo Móvel Nativo

Embora o sistema seja responsivo e funcione em dispositivos móveis pelo navegador, o desenvolvimento de um aplicativo nativo para Android e iOS — utilizando React Native, dada a familiaridade do ecossistema com o código React já existente no frontend — proporcionaria uma experiência mais fluida para os perfis de Aluno e Responsável, que tendem a acessar o sistema majoritariamente por dispositivos móveis.

Um aplicativo nativo também permitiria o envio de notificações push, recurso não disponível em aplicações web convencionais, ampliando o alcance do canal de comunicação entre a escola e as famílias REFERÊNCIAS BIBLIOGRÁFICAS

Sistema Schoolink - Sistema de Gerenciamento Escolar

REFERENCIAS

ABNT — ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 14724: Informação e documentação — Trabalhos acadêmicos — Apresentação. Rio de Janeiro: ABNT, 2011.

ABRAMOV, Dan; CLARK, Andrew. Redux: A Predictable State Container for JavaScript Apps. 2015. Disponível em: https://redux.js.org/. Acesso em: mar. 2026.

BANKS, Alex; PORCELLO, Eve. Learning React. 2. ed. Sebastopol: O'Reilly Media, 2020.

BRASIL. Lei nº 9.394, de 20 de dezembro de 1996. Lei de Diretrizes e Bases da Educação Nacional. Diário Oficial da União, Brasília, DF, 23 dez. 1996. Disponível em: https://www.planalto.gov.br/ccivil_03/leis/l9394.htm. Acesso em: 01 mar. 2026.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). Diário Oficial da União, Brasília, DF, 15 ago. 2018. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 01 mar. 2026.

BRASIL. Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP). Censo Escolar da Educação Básica 2023: resumo técnico. Brasília: INEP, 2023. Disponível em: https://www.gov.br/inep/pt-br/areas-de-atuacao/pesquisas-estatisticas-e-indicadores/censo-escolar. Acesso em: 01 mar. 2026.

BROWN, Ethan. Web Development with Node and Express. 2. ed. Sebastopol: O'Reilly Media, 2019.

CHEN, Peter. The Entity-Relationship Model: Toward a Unified View of Data. ACM Transactions on Database Systems, v. 1, n. 1, p. 9-36, mar. 1976.

CHIAVENATO, Idalberto. Introdução à Teoria Geral da Administração. 9. ed. Rio de Janeiro: Elsevier, 2014.

CHINNATHAMBI, Kirupa. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux. Indianapolis: Addison-Wesley, 2018.

DAHL, Ryan. Node.js: Evented I/O for V8 JavaScript. Apresentação na JSConf EU, 2009.

DATE, C. J. Introdução a Sistemas de Banco de Dados. 8. ed. Rio de Janeiro: Elsevier, 2004.

ECMA INTERNATIONAL. ECMA-262: ECMAScript 2015 Language Specification. 6. ed. Genebra: ECMA, 2015. Disponível em: https://www.ecma-international.org/ecma-262/ 6.0/. Acesso em: mar. 2026.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de Banco de Dados. 6. ed. São Paulo: Pearson Addison Wesley, 2011.

FERRAIOLO, David F.; SANDHU, Ravi; GAVRILA, Serban. Proposed NIST Standard for Role-Based Access Control. ACM Transactions on Information and System Security, v. 4, n. 3, p. 224-274, ago. 2001.

FERREIRA, Naura Syria Carapeto (org.). Gestão Democrática da Educação: atuais tendências, novos desafios. 8. ed. São Paulo: Cortez, 2012.

FLANAGAN, David. JavaScript: The Definitive Guide. 7. ed. Sebastopol: O'Reilly Media, 2020.

FREEMAN, Adam; ROBSON, Elizabeth. Head First JavaScript Programming. 2. ed. Sebastopol: O'Reilly Media, 2020.

GIL, Antonio Carlos. Como Elaborar Projetos de Pesquisa. 6. ed. São Paulo: Atlas, 2017.

HOLOWAYCHUK, TJ. Express: Fast, Unopinionated, Minimalist Web Framework for Node.js. 2010. Disponível em: https://expressjs.com/. Acesso em: mar. 2026.

IEEE. IEEE 830-1998: IEEE Recommended Practice for Software Requirements Specifications. New York: Institute of Electrical and Electronics Engineers, 1998.

ISO/IEC 27001. Information Security Management Systems: Requirements. Geneva: International Organization for Standardization, 2022.

ISO 9241-11. Ergonomics of Human-System Interaction — Part 11: Usability: Definitions and Concepts. Geneva: International Organization for Standardization, 2018.

JONES, Michael; BRADLEY, John; SAKIMURA, Nat. RFC 7519: JSON Web Token (JWT). Internet Engineering Task Force, 2015. Disponível em: https://datatracker.ietf.org/doc/html/rfc7519. Acesso em: 01 mar. 2026.

LAUDON, Kenneth C.; LAUDON, Jane P. Sistemas de Informação Gerenciais. 14. ed. São Paulo: Pearson Education do Brasil, 2021.

LIE, Håkon Wium; BOS, Bert. Cascading Style Sheets: Designing for the Web. 2. ed. Boston: Addison-Wesley, 1999.

LUCK, Heloísa. Dimensões de Gestão Escolar e suas Competências. Curitiba: Editora Positivo, 2009.

MARCOTTE, Ethan. Responsive Web Design. New York: A Book Apart, 2011.

MICROSOFT. TypeScript: Typed JavaScript at Any Scale. 2012. Disponível em: https://www.typescriptlang.org/. Acesso em: mar. 2026.

MORAN, José Manuel. Educação Híbrida: um conceito-chave para a educação hoje. In: BACICH, Lilian; NETO, Adolfo Tanzi; TREVISANI, Fernando de Mello (org.). Ensino Híbrido: personalização e tecnologia na educação. Porto Alegre: Penso, 2015.

MYERS, Glenford J.; SANDLER, Corey; BADGETT, Tom. The Art of Software Testing. 3. ed. Hoboken: John Wiley & Sons, 2011.

NIELSEN, Jakob. Usability Engineering. San Francisco: Academic Press, 1994.

O'BRIEN, James A.; MARAKAS, George M. Administração de Sistemas de Informação. 15. ed. Porto Alegre: AMGH, 2013.

OWASP FOUNDATION. OWASP Top Ten: The Ten Most Critical Web Application Security Risks. [S. l.]: OWASP, 2021. Disponível em: https://owasp.org/www-project-top-ten/. Acesso em: 01 mar. 2026.

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de Software: uma abordagem profissional. 8. ed. Porto Alegre: AMGH, 2016.

PRISMA. Prisma Documentation: Next-Generation Node.js and TypeScript ORM. 2019. Disponível em: https://www.prisma.io/docs. Acesso em: mar. 2026.

REACT. Introducing Hooks. 2019. Disponível em: https://reactjs.org/docs/hooks-intro.html. Acesso em: mar. 2026.

REDUX TOOLKIT. Redux Toolkit Documentation. 2019. Disponível em: https://redux-toolkit.js.org/. Acesso em: mar. 2026.

REDUX TOOLKIT. RTK Query Overview. 2021. Disponível em: https://redux-toolkit.js.org/rtk-query/overview. Acesso em: mar. 2026.

SANDHU, Ravi S. et al. Role-Based Access Control Models. IEEE Computer, v. 29, n. 2, p. 38-47, fev. 1996.

SHADCN. shadcn/ui: Beautifully Designed Components. 2023. Disponível em: https://ui.shadcn.com/. Acesso em: mar. 2026.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. Sistema de Banco de Dados. 7. ed. Rio de Janeiro: GEN LTC, 2020.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson Universidades, 2019.

STALLINGS, William. Cryptography and Network Security: Principles and Practice. 7. ed. Boston: Pearson, 2018.

STONEBRAKER, Michael; ROWE, Lawrence A. The Design of POSTGRES. ACM SIGMOD Record, v. 15, n. 2, p. 340-355, 1986.

TILKOV, Stefan; VINOSKI, Steve. Node.js: Using JavaScript to Build High-Performance Network Programs. IEEE Internet Computing, v. 14, n. 6, p. 80-83, nov./dez. 2010.

VALENTE, José Armando; ALMEIDA, Maria Elizabeth Bianconcini de. Transformação Digital na Educação: o que mudou e o que ainda precisa mudar. Campinas: UNICAMP, 2020.

VENTURA, Plínio. Requisitos de Software: uma visão detalhada sobre Requisitos Funcionais, Requisitos Não-Funcionais e Regras de Negócio. [S. l.]: Indtech Academia de Software, [2016]. E-book. Disponível em: http://www.ateomomento.com.br. Acesso em: 01 mar. 2026.

VERCEL. Next.js Documentation. 2016. Disponível em: https://nextjs.org/docs. Acesso em: mar. 2026.

W3C. Web Content Accessibility Guidelines (WCAG) 2.1. [S. l.]: World Wide Web Consortium, 2018. Disponível em: https://www.w3.org/TR/WCAG21/. Acesso em: 01 mar. 2026.

WATHAN, Adam. Tailwind CSS Documentation. 2019. Disponível em: https://tailwindcss.com/docs. Acesso em: mar. 2026.

WIERUCH, Robin. The Road to Next.js. 2022. Disponível em: https://www.roadtonextjs.com/. Acesso em: mar. 2026.

Pontifícia Universidade Católica de Goiás
Pontifical Catholic University of Goias
Av. Universitária, 1069, Setor Universitário
Caixa Postal 86 - CEP 74.605-010
Goiânia – Goiás – Brasil

RESOLUÇÃO n° 038/2020 – CEPE

ANEXO I

APÊNDICE ao TCC

Termo de autorização de publicação de produção acadêmica

O estudante caio mariani de morais do Curso de ciências da computação matrícula 20221002800742, telefone: 62 9 811-0081 e-mail caiomarianni@gmail.com, na qualidade de titular dos direitos autorais, em consonância com a Lei nº 9.610/98 (Lei dos Direitos do Autor), autoriza a Pontifícia Universidade Católica de Goiás (PUC Goiás) a disponibilizar o Trabalho de Conclusão de Curso intitulado Desenvolvimento do sistema escolar Schoolink _____ _____ _____, gratuitamente, sem ressarcimento dos direitos autorais, por 5 (cinco) anos, conforme permissões do documento, em meio eletrônico, na rede mundial de computadores, no formato especificado (Texto(PDF); Imagem (GIF ou JPEG); Som (WAVE, MPEG, AIFF, SND); Vídeo (MPEG, MWV, AVI, QT); outros, específicos da área; para fins de leitura e/ou impressão pela internet, a título de divulgação da produção científica gerada nos cursos de graduação da PUC Goiás.

Goiânia, 23 de junho de 2026.

Assinatura do autor: _____

Nome completo do autor: Caio Marianni de Morais

gov.br Documento assinado digitalmente
EUGENIO JULIO MESSALA CANDIDO CARVALHO
Data: 24/06/2026 10:57:11-0300
Verifique em https://validar.iti.gov.br

Assinatura do professor-orientador: _____
Nome completo do professor-orientador: