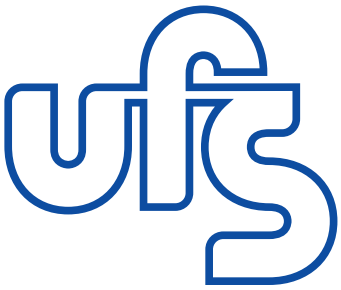


UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

SEEduca: Sistema de Gestão Acadêmica para as Escolas Públicas

Trabalho de Conclusão de Curso

Matheus Cardoso Teixeira

São Cristóvão – Sergipe

2024

UNIVERSIDADE FEDERAL DE SERGIPE

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

DEPARTAMENTO DE COMPUTAÇÃO

Matheus Cardoso Teixeira

SEEduca: Sistema de Gestão Acadêmica para as Escolas Públicas

Trabalho de Conclusão de Curso submetido ao Departamento de Computação da Universidade Federal de Sergipe como requisito parcial para a obtenção do título de Bacharel em Sistemas da Informação.

Orientador(a): Prof. Dr. Edward David Moreno Ordonez

São Cristóvão – Sergipe

2024

Querida família,

Hoje, enquanto finalizo este importante capítulo da minha jornada acadêmica, não posso deixar de expressar a imensa gratidão que sinto por cada um de vocês. Luciene, minha esposa dedicada, sua incansável compreensão, apoio e amor foram a âncora que me sustentou durante os momentos desafiadores desta empreitada. Sua presença constante e seu encorajamento foram a luz que iluminou meu caminho em direção à realização deste sonho.

A você, João Victor, e a você, Micaella, meus filhos amados, agradeço por serem minha inspiração diária. Suas risadas contagiantes, seus abraços afetuosos e sua compreensão juvenil me deram forças nos momentos em que a jornada acadêmica parecia árdua demais. Vocês são meu orgulho e minha maior motivação para buscar sempre o melhor em tudo o que faço.

À nossa família como um todo, expresso minha profunda gratidão por todo o apoio inabalável, compreensão e amor incondicional ao longo deste caminho. Cada palavra de incentivo, cada gesto de carinho, foi como um vento gentil impulsionando-me em direção aos meus objetivos.

Hoje, ao concluir esta monografia, sei que não estou apenas celebrando uma conquista pessoal, mas também uma conquista de todos nós. Este trabalho é dedicado a vocês, minha família amada, cujo apoio e amor são a fundação sobre a qual construímos nossos sonhos e aspirações.

Com amor e gratidão,

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

Resumo

Este trabalho descreve o desenvolvimento do sistema de gestão escolar SEEDuca, voltado para as escolas públicas estaduais, com o objetivo de proporcionar uma solução eficiente e abrangente que atenda às necessidades específicas das instituições de ensino do estado. Para definir os requisitos do sistema, foi realizada uma pesquisa minuciosa. Inicialmente, conduzimos uma análise aprofundada em plataformas de trabalho científico, como o INPI, Scopus e Web of Science, para identificar soluções similares e melhores práticas em sistemas de gestão escolar. Além disso, realizamos uma pesquisa nas principais lojas de aplicativos, como a AppStore e PlayStore, para entender as preferências dos usuários e as funcionalidades valorizadas em aplicativos relacionados à educação. A pesquisa de requisitos foi complementada por um questionário aplicado aos principais envolvidos na gestão escolar das escolas públicas estaduais, a fim de capturar as necessidades específicas do sistema SEEDuca. Com base nessas informações, definimos a arquitetura do sistema, garantindo escalabilidade, segurança e integração com outros sistemas utilizados nas escolas públicas estaduais. Este projeto visa contribuir significativamente para a qualidade da educação e melhorar a experiência de pais, alunos e professores nas escolas públicas estaduais.

Palavras-chave: Sistema de gestão escolar, escolas públicas estaduais, SEEDuca, educação, arquitetura do sistema, qualidade da educação.

Abstract

This work describes the development of the SEEDuca school management system, aimed at state public schools, with the aim of providing an efficient and comprehensive solution that meets the specific needs of state educational institutions. To define the system's requirements, a thorough research was conducted. Initially, we conducted an in-depth analysis on scientific work platforms such as INPI, Scopus, and Web of Science to identify similar solutions and best practices in school management systems. Additionally, we conducted research in major app stores such as the AppStore and PlayStore to understand user preferences and valued features in education-related applications. The requirements research was complemented by a questionnaire applied to key individuals involved in the management of state public schools to capture the specific needs of the SEEDuca system. Based on this information, we defined the system's architecture, ensuring scalability, security, and integration with other systems used in state public schools. This project aims to significantly contribute to the quality of education and improve the experience of parents, students, and teachers in state public schools.

Keywords: School management system, state public schools, SEEDuca, education, system architecture, quality of education.

Lista de ilustrações

Figura 1 – Percentual de alunos em escolas públicas e privadas em Sergipe. 23
Figura 2 – Imagem capturada do site https://www.altexsoft.com/. 25
Figura 3 – Telas do app Minha Escola SP 32
Figura 4 – Telas do app Escola Paraná 33
Figura 5 – Telas do app Escola RS 34
Figura 6 – Telas do app NetEscola 35
Figura 7 – Telas do app Estudante SC 35
Figura 8 – Diagrama de casos de uso 49
Figura 9 – Diagrama de classes 50
Figura 10 – Diagrama de sequência - Professor cadastra atividade 51
Figura 11 – Diagrama de sequência - Aluno consulta atividade de casa 52
Figura 12 – Diagrama de sequência - Coordenador gerencia cardápio da merenda escolar 53
Figura 13 – Telas iniciais do SEEduca 54
Figura 14 – Tela Home do SEEduca 55
Figura 15 – Telas com as informações da Escola e do Aluno 55
Figura 16 – Telas da Agenda Escolar, Frequência e Notas do aluno 56
Figura 17 – Telas do Mural de Avisos, Informações do Transporte e Recados 56
Figura 18 – Tela da Carteirinha Estudantil 57
Figura 19 – Tela de Login 58
Figura 20 – Tela de Redefinição de Senha 58
Figura 21 – Dashboard 59
Figura 22 – Visualizar Dados da Escola 60
Figura 23 – Lista de Turmas da Sala 001 60
Figura 24 – Agenda da Turma 61
Figura 25 – Agenda da Turma 62
Figura 26 – Gerenciar Itens do Cardápio 62
Figura 27 – Cadastrar Novos Alunos 63
Figura 28 – Registro de Presença 64
Figura 29 – Gerenciar Notas do Boletim 64
Figura 30 – Agenda de Recados 65
Figura 31 – Cadastrar Novo Funcionário 66
Figura 32 – Visualizar Dados do Professor 66

Lista de quadros

Quadro 1 – Aplicativos selecionados 32
Quadro 2 – Características observadas nos aplicativos 36
Quadro 3 – Tabela comparativa de funcionalidades 37
Quadro 4 – Resultado da busca do INPI 38
Quadro 5 – Tabela comparativa de funcionalidades na base de patentes 38
Quadro 6 – Quadro comparativo das strings de busca nas bases de artigos 39
Quadro 7 – Resultado da pesquisa na base *Web Of Science* 40
Quadro 8 – Resuldado da pesquisa na base Scopus 41
Quadro 9 – Requisitos funcionais elicitados 47
Quadro 10 – Requisitos não funcionais elicitados 48

Lista de tabelas

Tabela 1 – Pesquisa de *Apps* 31

Lista de códigos

Código 1 – Trecho1 do arquivo settings.py . . . 68
Código 2 – Trecho2 do arquivo settings.py . . . 69
Código 3 – escola_model.py . . . 70
Código 4 – escola_api_view.py . . . 71
Código 5 – escola_serializer.py . . . 71
Código 6 – escola_serializer.py . . . 72
Código 7 – aluno_model.py . . . 73
Código 8 – aluno_api_view.py . . . 74
Código 9 – aluno_permission.py . . . 74
Código 10 – turma_model.py . . . 75
Código 11 – turma_model.py . . . 76
Código 12 – boletim_model.py . . . 76
Código 13 – boletim_model.py . . . 77
Código 14 – boletim_model.py . . . 78
Código 15 – boletim_model.py . . . 79
Código 16 – agenda_escolar_api_view.py . . . 80
Código 17 – escolas - urls.py . . . 80
Código 18 – urls.py . . . 81
Código 19 – escolas/index.js . . . 82
Código 20 – escolas/index.js . . . 83
Código 21 – escolas/index.js . . . 84
Código 22 – escolas/add/index.js . . . 86
Código 23 – escolas/add/index.js . . . 87
Código 24 – escolas/components/Menu/index.js . . . 88
Código 25 – routes.js . . . 90
Código 26 – card . . . 90
Código 27 – datatable . . . 91
Código 28 – MDBox . . . 91
Código 29 – Biblioteca Axios . . . 92
Código 30 – Biblioteca Axios . . . 92
Código 31 – Response.data . . . 93
Código 32 – Rotas do SimpleJWT . . . 94
Código 33 – Acessando a rota ME . . . 95
Código 34 – Dashboard . . . 96
Código 35 – Dashboard . . . 97
Código 36 – Dashboard . . . 98

Código 37 – Dashboard 99
Código 38 – Consumindo a api no front mobile 100
Código 39 – Código para obtenção de tokens de acesso 101
Código 40 – Código para realizar o login na aplicação 102
Código 41 – Provider do contexto envolvendo a aplicação 102
Código 42 – Código para realizar o logout na aplicação 103
Código 43 – Routes 104

Lista de gráficos

Gráfico 1 – Gráfico dos artigos publicados sobre o tema do sistema 39

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
abnTeX	ABsurdas Normas para TeX
DCOMP	Departamento de Computação
UFS	Universidade Federal de Sergipe
SEDUC	Secretaria de Estado da Educação
SEEduca	Sistema de Gestão Acadêmica para as Escolas Públicas
APP	Application
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
JSON	JavaScript Object Notation
REST	Representational State Transfer
JWT	JSON Web Token
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
JS	JavaScript
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDE	Itegrated Development Environment
SQL	Structured Query Language
URL	Uniform Resource Locator

Sumário

1 Introdução 17
1.1 Objetivos 18
1.1.1 Objetivo Geral 18
1.1.2 Objetivos Específicos 18
1.2 Metodologia 18
1.3 Estrutura do Documento 20

2 Fundamentação Teórica 22
2.1 Educação e o Papel das Escolas Públicas 22
2.2 Sistemas de Gestão Escolar 23
2.2.1 Funções e Características dos Sistemas de Gestão Escolar 23
2.2.2 Vantagens dos Sistemas de Gestão Escolar 24
2.2.3 Desafios na Implementação de Sistemas de Gestão Escolar 25
2.3 API como Solução para Sistemas Multiplataformas 25
2.3.1 O que é uma API? 25
2.3.2 Vantagens do Uso de APIs em Sistemas Multiplataformas 26
2.3.3 Desafios na Implementação de APIs em Sistemas Multiplataformas . . 26
2.4 Tecnologias Necessárias para o Desenvolvimento do Sistema 27
2.5 Tecnologias e Ferramentas 28
2.5.1 Django *Framework* 28
2.5.2 Django Rest *Framework* 28
2.5.3 Pycharm 28
2.5.4 *Visual Studio Code* 29
2.5.5 *React Native* 29
2.5.6 Draw.io 29
2.5.7 Figma 29

3 Estudo de Soluções Similares 31
3.1 Pesquisa em Lojas de Apps 31
3.1.1 Minha Escola SP 32
3.1.2 Escola Paraná 33
3.1.3 Escola RS 33
3.1.4 NetEscola 34
3.1.5 Estudante SC 35
3.2 Análise das Soluções Encontradas 36
3.3 Pesquisa em Base de Patentes 37

3.4 Pesquisa em Artigos Científicos 38

4 Desenvolvimento do Produto de *Software* 42
4.1 Pesquisa Sobre Gestão Escolar 42
4.1.1 Perfil dos Pesquisados 43
4.1.1.1 Perfil dos membros das escolas 43
4.1.1.2 Perfil dos alunos e responsáveis 43
4.1.2 Resultados da Pesquisa 43
4.1.2.1 Resultados da pesquisa de membros das escolas 44
4.1.2.2 Resultados de pesquisa dos alunos e responsáveis 45
4.1.3 Conclusão da pesquisa 46
4.2 Requisitos de *Software* 46
4.2.1 Elicitação dos Requisitos Funcionais e Não Funcionais 46
4.3 Diagrama de Casos de Uso 48
4.4 Diagrama de Classes 50
4.5 Diagramas de Sequência 50
4.5.1 Professor cadastra atividade 51
4.5.2 Aluno consulta atividade de casa 51
4.5.3 Coordenador insere cardápio da merenda escolar 52
4.6 Protótipo de Telas 53
4.6.1 Protótipo do APP SEEduca 54
4.6.1.1 Tela inicial, Login e Recuperação de Senha 54
4.6.1.2 Tela *Home* 54
4.6.1.3 Informações da escola e do aluno 55
4.6.1.4 Telas da Agenda Escolar, Frequência e Notas do aluno 55
4.6.1.5 Telas do Mural de Avisos, Informações de Transporte e Recados 56
4.6.1.6 Tela da Carteirinha Estudantil 57
4.6.2 Protótipo do Sistema *WEB* SEEduca 57
4.6.2.1 Tela de *Login* 57
4.6.2.2 Tela de Redefinição de Senha 58
4.6.2.3 *Dashboard* 59
4.6.2.4 Informações da Escola 59
4.6.2.5 Listar Turmas 60
4.6.2.6 Agenda da Turma 61
4.6.2.7 Lista de Disciplinas 61
4.6.2.8 Gerenciar Itens do Cardápio 62
4.6.2.9 Cadastrar Novos Alunos 63
4.6.2.10 Registrar Nova Presença 63
4.6.2.11 Gerenciar Notas do Boletim 64
4.6.2.12 Agenda de Recados 65

4.6.2.13 Cadastrar Novo Funcionario . . . 65
4.6.2.14 Visualizar Dados do Professor . . . 66

5 Implementação do Sistema de Gestão SEEDUCA . . . 67
5.1 *Backend* . . . 67
5.1.1 Principais Funcionalidades . . . 69
5.1.1.1 Gestão de Escolas . . . 69
5.1.1.2 Gestão de Pessoas . . . 73
5.1.2 Modelos de Dados . . . 75
5.1.2.1 App *escolas* . . . 75
5.1.2.2 App *pessoas* . . . 76
5.1.3 APIs RESTful . . . 79
5.1.4 Integração com o *Frontend* . . . 81
5.2 *Frontend WEB* . . . 82
5.2.1 Organização das Telas . . . 82
5.2.2 Componentes Reutilizáveis . . . 88
5.2.3 Componentes e Estilos . . . 89
5.2.3.1 Principais Componentes Implementados . . . 89
5.2.3.2 Estilos e Componentes do *Material Dashboard* 2 React . . . 89
5.2.4 Integração com o *Backend* . . . 91
5.2.4.1 Requisições com Axios . . . 91
5.2.4.2 Requisições com Fetch . . . 92
5.2.4.3 Manipulação dos Dados Recebidos . . . 92
5.2.5 Implementação da Autenticação e Controle de Acesso . . . 93
5.2.5.1 Contexto de Autenticação no *Frontend* . . . 94
5.2.5.2 Controle de Acesso com React *Router* DOM . . . 94
5.3 *Frontend Mobile* . . . 95
5.3.1 Organização das Telas . . . 95
5.3.2 Componentes e Navegação . . . 95
5.3.3 Consumo de APIs RESTful . . . 100
5.3.4 Contexto de Autenticação . . . 101
5.3.5 Controle de Acesso com React *Navigation* . . . 103

6 Testes do Sistema . . . 105
6.1 Testes Unitários . . . 105
6.2 Testes de Integração e Sistema . . . 106
6.3 Resultados dos Testes . . . 107
6.4 Limitação dos Testes . . . 107

7 Documentação do Código . . . 108

7.1 *Backend* Django 108
7.2 *Frontend Web* React JS 109
7.3 *Frontend Mobile* React Native 109

8 Conclusões 110
8.1 Desafios e Limitações 110
8.2 Dívidas Técnicas 110
8.3 Perspectivas Futuras 111

9 Considerações Finais 112

Referências 113

1 Introdução

A gestão escolar nas instituições de ensino público do estado desempenha um papel fundamental na promoção de uma educação de qualidade e no desenvolvimento dos alunos. No entanto, os sistemas tradicionais de gestão enfrentam desafios que impactam a eficiência e a integração entre os diversos *stakeholders* envolvidos(LüCK, 2009).

Nesse contexto, o desenvolvimento de um sistema de gestão escolar voltado especificamente para as escolas públicas do estado surge como uma solução crucial. A utilização de tecnologia e ferramentas digitais pode revolucionar a forma como essas escolas são administradas, aprimorando a eficiência, a transparência e a comunicação entre pais, alunos, professores e gestores(SANTOS; ALMEIDA, 2020; ARAúJO; COSTA, 2018).

Um sistema de gestão escolar abrangente para as escolas públicas pode englobar uma ampla gama de funcionalidades, desde o gerenciamento de matrículas e registros de alunos até a automação de processos de avaliação, comunicação de notas e faltas, gestão de horários e planejamento de atividades. Além disso, a implementação de uma plataforma web e/ou móvel proporciona acessibilidade e conveniência aos usuários, permitindo o acesso a informações relevantes e a interação em tempo real(SANTOS; ALMEIDA, 2020; ARAúJO; COSTA, 2018).

Ao adotar um sistema de gestão escolar direcionado às escolas públicas do estado, espera-se alcançar benefícios como o aumento da eficiência administrativa, o fortalecimento da comunicação entre a escola, a comunidade e os órgãos governamentais responsáveis, além de embasar as decisões com base em dados concretos. Os pais poderão acompanhar o desempenho acadêmico de seus filhos, receber comunicados importantes e se envolver ativamente na vida escolar. Os professores poderão registrar notas e faltas de forma ágil, acessar recursos didáticos relevantes e interagir de maneira mais eficiente com os alunos. Já os gestores terão acesso a insights valiosos para aprimorar os processos internos e promover o crescimento das escolas públicas do estado(SANTOS; ALMEIDA, 2020; ARAúJO; COSTA, 2018).

Diante desse cenário, neste trabalho foi desenvolvido um sistema de gestão escolar

específico para as escolas públicas de Sergipe, atendendo às suas necessidades particulares, promovendo uma administração eficiente e uma comunicação integrada. Por meio de uma abordagem tecnológica e inovadora, busca-se otimizar o ambiente educacional, facilitando o acesso às informações e contribuindo para o sucesso dos alunos, a satisfação dos pais e o progresso institucional nas escolas públicas do estado.

1.1 Objetivos

1.1.1 Objetivo Geral

Desenvolver um sistema de gestão escolar específico para as escolas públicas do estado, com o objetivo de criar uma plataforma que facilite a gestão escolar. Além disso, a plataforma visa melhorar a comunicação entre a escola e a comunidade, substituindo métodos arcaicos de interação como agendas, boletins e murais, tornando o processo de aprendizado mais eficiente.

1.1.2 Objetivos Específicos

- Realizar uma pesquisa e análise de sistemas de gestão escolar existentes, se possível de outras escolas públicas, identificando suas funcionalidades e diferenciais.
- Definir os requisitos necessários para o desenvolvimento do sistema de gestão escolar, levando em consideração as necessidades específicas das instituições de ensino.
- Desenvolver um sistema intuitivo e de fácil utilização, que atenda aos requisitos elicitados e proporcione uma experiência positiva para os usuários, como pais, alunos, professores e gestores.
- Implementar funcionalidades-chave, como gerenciamento de matrículas, controle de presenças, registro de notas e avaliações, gestão de horários e turmas, entre outras, de acordo com as necessidades da instituição de ensino.
- Realizar testes do sistema, garantindo sua eficiência, confiabilidade e segurança. Validar o sistema por meio de *feedback* e avaliação dos usuários, a fim de realizar melhorias e correções necessárias.

1.2 Metodologia

Para o desenvolvimento do sistema de gestão escolar, foi adotada uma metodologia que visa garantir a eficiência, qualidade e adequação às necessidades específicas das instituições de ensino e da Secretaria de Educação do Estado. Os passos a seguir descrevem a abordagem metodológica adotada:

- **Estudo do contexto educacional**: Realização de um estudo aprofundado sobre o ambiente educacional, compreendendo a estrutura e funcionamento das escolas, os processos administrativos e pedagógicos, bem como as demandas e desafios enfrentados pelos diferentes *stakeholders*.

- **Revisão de mercado e análise de soluções existentes**: Realização de pesquisas e revisões em busca de soluções de gestão escolar já disponíveis no mercado, tentando dar enfase a sistemas de escolas públicas. Essa análise permite identificar funcionalidades e melhores práticas utilizadas em sistemas já estabelecidos, com o objetivo de aproveitar ideias bem-sucedidas e inovadoras.

- **Levantamento de requisitos**: Realização de entrevistas, questionários e reuniões com os principais envolvidos na gestão escolar, incluindo diretores, professores, funcionários administrativos e pais. O objetivo é entender suas necessidades, expectativas e desejos em relação ao sistema de gestão escolar, identificando os requisitos funcionais e não funcionais que devem ser atendidos.

- **Definição da arquitetura do sistema**: Com base nos requisitos levantados, é elaborada a arquitetura do sistema de gestão escolar, considerando aspectos como escalabilidade, segurança, integração com outros sistemas e flexibilidade para futuras expansões e personalizações.

- **Desenvolvimento do sistema**: Com a arquitetura definida, inicia-se a etapa de desenvolvimento do sistema de gestão escolar. Utilizando as melhores práticas de programação e seguindo os padrões de desenvolvimento estabelecidos, a equipe de desenvolvimento implementa as funcionalidades identificadas e realiza testes para garantir a qualidade do sistema.

- **Validação e avaliação**: O sistema desenvolvido passa por um processo de validação, no qual é testado em ambiente real com a participação de usuários-chave. *Feedbacks* e sugestões são coletados para possíveis melhorias e ajustes finais antes da implantação definitiva.

- **Implantação e treinamento**: O sistema é implantado nas escolas, e um programa de treinamento é conduzido para capacitar os usuários na utilização do sistema. Essa etapa é fundamental para garantir que todos os envolvidos estejam aptos a aproveitar ao máximo as funcionalidades oferecidas pelo sistema de gestão escolar.

- **Monitoramento e manutenção**: Após a implantação, o sistema é monitorado constantemente para identificar possíveis problemas ou necessidades de atualizações. Manutenções regulares são realizadas para corrigir falhas, adicionar novas funcionalidades e garantir a segurança e estabilidade do sistema.

Ao seguir essa metodologia, busca-se desenvolver um sistema de gestão escolar eficiente, que atenda às necessidades específicas das instituições de ensino do estado, promovendo uma administração mais eficaz, comunicação aprimorada e melhor experiência para pais, alunos, professores e gestores.

1.3 Estrutura do Documento

Este documento está estruturado em oito capítulos, conforme descrito a seguir:

- **Capítulo 01 - Introdução**: Este capítulo apresenta uma visão geral sobre a importância da gestão escolar nas instituições de ensino público do estado e a necessidade de desenvolvimento de um sistema de gestão escolar específico para essas escolas. Além disso, são apresentados os objetivos gerais e específicos do trabalho, bem como a metodologia adotada para o desenvolvimento do sistema.
- **Capítulo 02 - Fundamentação Teórica**: Neste capítulo, são apresentados os principais conceitos teóricos relacionados à gestão escolar e aos sistemas de informação, fornecendo uma base teórica para o desenvolvimento do sistema proposto.
- **Capítulo 03 - Estudo de Soluções Similares**: Aqui são revisadas e analisadas soluções de gestão escolar existentes, com ênfase em sistemas utilizados por outras escolas públicas. Esta análise busca identificar funcionalidades, melhores práticas e desafios encontrados em sistemas similares.
- **Capítulo 04 - Desenvolvimento do Projeto**: Neste capítulo, é detalhado o processo de desenvolvimento do sistema de gestão escolar, desde a definição dos requisitos até a implementação das funcionalidades-chave. Serão abordadas as etapas de concepção, arquitetura, desenvolvimento e testes do sistema.
- **Capítulo 05 - Implementação do Sistema**: Aqui são apresentados os detalhes da implementação do sistema desenvolvido, incluindo as tecnologias utilizadas, a estrutura de código, os desafios enfrentados e as soluções adotadas durante o processo de implementação.
- **Capítulo 06 - Testes do Sistema**: Este capítulo descreve os testes realizados no sistema de gestão escolar, incluindo os tipos de testes, os resultados obtidos e as correções ou melhorias realizadas com base nos testes.
- **Capítulo 07 - Documentação do Código**: Aqui é apresentada a documentação do código fonte do sistema desenvolvido, fornecendo informações detalhadas sobre a estrutura do código, as funcionalidades implementadas, os padrões de codificação adotados e outras informações relevantes para o entendimento e manutenção do sistema.

- **Capítulo 08 - Conclusões**: Este capítulo apresenta as conclusões do trabalho, incluindo uma síntese dos resultados obtidos, as contribuições do sistema desenvolvido, as limitações encontradas e sugestões para trabalhos futuros.

Essa estrutura foi elaborada de forma a proporcionar uma organização lógica e sequencial do conteúdo, permitindo uma compreensão abrangente do processo de desenvolvimento do sistema de gestão escolar específico para as escolas públicas do estado.

2

Fundamentação Teórica

No presente capítulo, serão abordados os fundamentos essenciais para proporcionar uma compreensão mais aprofundada dos conceitos empregados neste trabalho, além de expor a motivação que impulsionou o desenvolvimento desta pesquisa. O embasamento teórico e a contextualização são essenciais para a compreensão do projeto em questão.

2.1 Educação e o Papel das Escolas Públicas

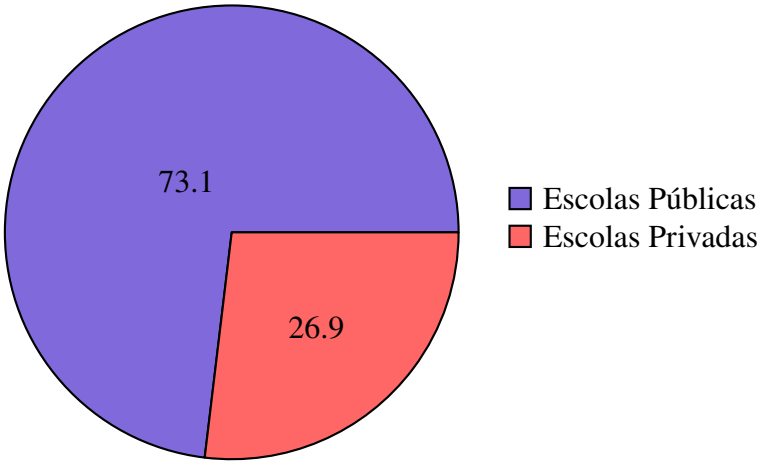
A educação é um dos pilares fundamentais para o desenvolvimento e progresso de uma sociedade. Ela desempenha um papel essencial na formação e preparação dos indivíduos para o exercício da cidadania plena e para a inserção no mercado de trabalho. Além disso, a educação contribui para a construção de uma sociedade mais justa, igualitária e consciente de suas responsabilidades(TEDESCO, 2007; FAUSTO, 2012).

A educação é um ato político, e é fundamental para a transformação da sociedade. Ela deve buscar a conscientização dos indivíduos e promover uma visão crítica do mundo, permitindo que as pessoas se tornem agentes ativos de mudança e não apenas espectadores passivos(FREIRE, 1992). No contexto das escolas públicas, sua relevância é ainda mais significativa, pois essas instituições têm o papel fundamental de garantir o acesso à educação de qualidade para toda a população, independentemente de sua origem social ou econômica. A educação pública deve ser inclusiva e proporcionar oportunidades iguais para todos os estudantes, visando a promoção da equidade(HADDAD, 2009).

As escolas públicas são responsáveis por oferecer um currículo diversificado e adequado às necessidades locais, promovendo o desenvolvimento intelectual, social e emocional dos alunos. Além disso, elas desempenham um papel crucial na formação de valores e no estímulo ao pensamento crítico, preparando os jovens para serem cidadãos ativos, conscientes e participativos na sociedade(FREIRE, 1996; LIBâNEO, 2013; SAVIANI, 2003). Na Figura 1 podemos visualizar

a importância das escolas públicas para a educação brasileira, pois 73,1 % dos alunos matriculados, são da escola pública, entre municipais, estaduais e federais, contra 26,9 % das escolas particulares. Dados do censo escolar 2023.

Figura 1 – Percentual de alunos em escolas públicas e privadas em Sergipe.



Fonte: Autor (2024).

Uma gestão eficiente requer uma administração participativa, em que professores, funcionários, alunos e comunidade escolar estejam envolvidos nas decisões e no acompanhamento do desempenho da instituição. A gestão democrática contribui para a construção de uma escola mais aberta, transparente e responsiva às necessidades dos estudantes(DEWEY, 2000).

A gestão das escolas públicas é um desafio complexo e multifacetado, que envolve aspectos administrativos, pedagógicos e financeiros. É fundamental que as escolas sejam geridas de forma eficiente e transparente, garantindo a alocação adequada de recursos, a valorização dos professores e demais profissionais da educação, bem como a participação ativa da comunidade escolar nas decisões e no acompanhamento do desempenho da instituição(LüCK, 2009; PARO, 2001; VEIGA, 2008).

2.2 Sistemas de Gestão Escolar

Os sistemas de gestão escolar desempenham um papel crucial na administração e organização das atividades educacionais em instituições de ensino. Essas plataformas tecnológicas são projetadas para facilitar e otimizar o gerenciamento das escolas, abrangendo desde aspectos administrativos até pedagógicos.

2.2.1 Funções e Características dos Sistemas de Gestão Escolar

Os sistemas de gestão escolar oferecem uma variedade de funcionalidades para simplificar e aprimorar a rotina das escolas. Algumas das principais características incluem:

1. **Cadastro de Alunos e Professores:** Permitem o registro e acompanhamento dos dados dos estudantes e professores, facilitando o controle de matrículas e o acesso às informações pessoais e acadêmicas.

2. **Controle de Frequência e Notas:** Os sistemas de gestão escolar permitem o lançamento e monitoramento das frequências dos alunos, bem como o registro das notas e avaliações.

3. **Emissão de Boletins e Relatórios:** Possibilitam a geração automatizada de boletins, relatórios de desempenho dos alunos e outros documentos importantes para o acompanhamento do progresso educacional.

4. **Comunicação Escola-Família:** Facilitam a comunicação entre a escola, alunos e responsáveis, por meio de ferramentas de comunicação integradas, como mensagens, notificações e boletins online.

5. **Planejamento Pedagógico:** Auxiliam na elaboração de planos de aula, grade curricular, cronogramas e outras atividades pedagógicas.

6. **Controle de Biblioteca:** Gerenciam o acervo bibliográfico da escola, possibilitando o cadastro de livros, empréstimos e reservas.

2.2.2 Vantagens dos Sistemas de Gestão Escolar

A adoção de sistemas de gestão escolar oferece diversos benefícios para as instituições de ensino, tais como:

1. **Eficiência Administrativa:** Com a automatização de processos, a gestão escolar torna-se mais ágil e eficiente, reduzindo o tempo gasto em atividades burocráticas.

2. **Melhoria da Comunicação:** Por meio do uso das ferramentas de comunicação integradas, a escola pode estabelecer uma comunicação mais efetiva com alunos, pais e professores, facilitando a troca de informações e o acompanhamento do desempenho dos estudantes.

3. **Transparência e Acesso à Informação:** Os sistemas de gestão escolar permitem maior transparência nas informações educacionais, tornando-as acessíveis para toda a comunidade escolar.

4. **Tomada de Decisões Embasadas em Dados:** Com o registro e análise de dados, a escola pode embasar suas decisões em informações concretas, possibilitando a implementação de ações mais assertivas para a melhoria da qualidade do ensino.

5. **Organização e Segurança dos Dados:** Os sistemas de gestão escolar garantem a segurança e integridade dos dados, evitando perdas ou extravios de informações importantes.

2.2.3 Desafios na Implementação de Sistemas de Gestão Escolar

Apesar das vantagens, a implementação de sistemas de gestão escolar também enfrenta alguns desafios, como a capacitação dos profissionais envolvidos, a integração de sistemas pré-existentes, a garantia da acessibilidade e a adaptação à realidade específica de cada instituição.

No contexto das escolas públicas estaduais, é fundamental considerar as necessidades e particularidades dessa rede de ensino, garantindo que o sistema de gestão escolar atenda às demandas específicas dessas instituições e contribua efetivamente para a melhoria da qualidade da educação pública. Nesse sentido, a adoção de tecnologias e metodologias adequadas, aliada a um planejamento bem estruturado, é essencial para o sucesso da implementação desses sistemas nas escolas públicas estaduais.

2.3 API como Solução para Sistemas Multiplataformas

As APIs (Interfaces de Programação de Aplicativos) desempenham um papel crucial no desenvolvimento de sistemas multiplataformas, permitindo a interação e comunicação entre diferentes *softwares* e plataformas de maneira eficiente e padronizada.

Tannembaum (2014) afirma que as APIs são essenciais para a construção de sistemas operacionais modernos e são projetadas para facilitar a integração entre aplicativos e o núcleo do sistema, proporcionando uma interface clara e segura para a programação de *software*.

2.3.1 O que é uma API?

Uma API é um conjunto de regras e protocolos que permite a interação entre diferentes *softwares* e aplicativos. Ela atua como uma ponte de comunicação, permitindo que sistemas distintos troquem informações, solicitem serviços e executem operações específicas sem a necessidade de compartilhar todo o código-fonte.

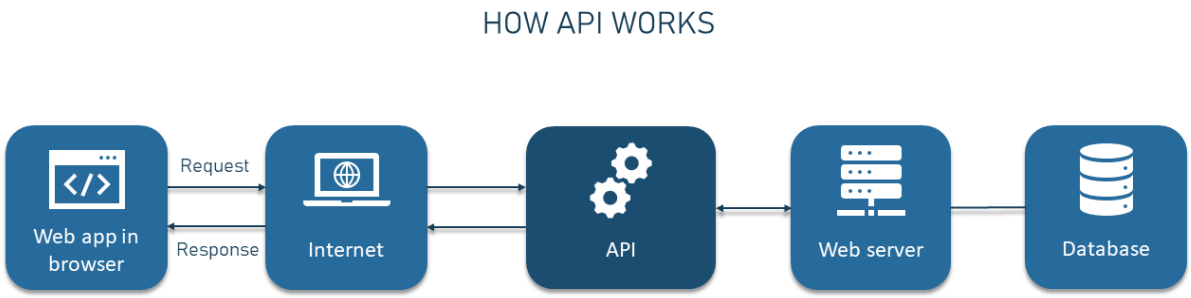


Figura 2 – Imagem capturada do site https://www.altexsoft.com/.

Musser (2011) ressalta que as APIs são fundamentais para a construção de ecossistemas digitais e para a integração de sistemas de forma flexível e escalável. Elas possibilitam a exposição controlada de funcionalidades e dados, permitindo que desenvolvedores criem novos serviços e aplicações sobre uma base já existente, estimulando a inovação e o crescimento tecnológico.

2.3.2 Vantagens do Uso de APIs em Sistemas Multiplataformas

Ao desenvolver um sistema multiplataforma, como o SEEduca, a adoção de APIs oferece inúmeras vantagens:

1. **Integração Simples e Eficiente:** As APIs facilitam a integração de diferentes sistemas e tecnologias, permitindo que o sistema de gestão escolar se comunique com outras plataformas, como aplicativos móveis, serviços web e bancos de dados externos.
2. **Reaproveitamento de Código:** Com o uso de APIs, é possível reaproveitar componentes e funcionalidades de outros sistemas, acelerando o desenvolvimento e reduzindo custos.
3. **Manutenção Simplificada:** Utilizando APIs, as atualizações e melhorias em um sistema não afetam necessariamente os demais sistemas que se conectam a ele, garantindo maior estabilidade e facilidade na manutenção.
4. **Acesso a Recursos Externos:** As APIs possibilitam o acesso a recursos externos, como dados de clima, mapas, sistemas de pagamento, entre outros, enriquecendo o sistema de gestão escolar com informações relevantes e atualizadas.

2.3.3 Desafios na Implementação de APIs em Sistemas Multiplataformas

Embora as APIs ofereçam inúmeras vantagens, sua implementação também pode enfrentar desafios, como:

1. **Padronização e Compatibilidade:** É essencial garantir que as APIs sejam padronizadas e compatíveis com os diferentes sistemas e plataformas que precisam interagir, para evitar conflitos e dificuldades de comunicação.
2. **Segurança:** As APIs podem representar pontos de vulnerabilidade no sistema, sendo necessário adotar medidas robustas de segurança para proteger os dados e informações sensíveis.
3. **Documentação Adequada:** Uma boa documentação da API é fundamental para que os desenvolvedores possam compreender sua estrutura e funcionalidades, facilitando a integração e o uso adequado.

4. **Gerenciamento de Versões:** Com o tempo, é possível que as APIs passem por atualizações e melhorias, sendo importante gerenciar adequadamente as versões para garantir a compatibilidade com sistemas já integrados.

A utilização de APIs no desenvolvimento do sistema de gestão escolar para escolas públicas estaduais permitiu uma maior flexibilidade, adaptabilidade e interconexão com outros sistemas, enriquecendo a experiência dos usuários e contribuindo para uma gestão educacional mais eficiente e moderna.

2.4 Tecnologias Necessárias para o Desenvolvimento do Sistema

No desenvolvimento do SEEDuca, foram utilizadas tecnologias modernas e eficientes. As principais escolhas para o desenvolvimento do sistema foram:

- **Linguagem de Programação:** Python como a linguagem de programação principal, devido à sua facilidade de aprendizado, legibilidade e ampla comunidade de desenvolvedores.
- ***Framework Web:*** Django o *framework web* escolhido, pois fornece uma estrutura robusta e escalável para o desenvolvimento de aplicações *web* com *Python*.
- ***Framework Front-end:*** *React* para o desenvolvimento do *front-end web*, permitindo a criação de interfaces interativas e responsivas.
- **Desenvolvimento *Mobile*:** Para o desenvolvimento da versão *mobile* do sistema, foi utilizado o *React Native*, que possibilita a criação de aplicativos para *Android* e iOS de forma eficiente.
- **Banco de Dados:** O banco de dados escolhido foi o MariaDB, uma opção robusta e de código aberto, que oferece bom desempenho e segurança.
- **Comunicação entre Serviços:** As APIs RESTful foram utilizadas para a comunicação entre os diferentes serviços do sistema, permitindo uma arquitetura de serviços web eficiente e flexível.

A escolha dessas tecnologias visou garantir a eficiência, escalabilidade e segurança do sistema de SEEDuca, proporcionando uma experiência de uso agradável tanto para os administradores quanto para os usuários finais. O uso dessas tecnologias também facilitará a manutenção e evolução contínua do sistema, permitindo que ele acompanhe as demandas em constante mudança da área de educação.

2.5 Tecnologias e Ferramentas

Nesta seção foram apresentadas as tecnologias e ferramentas que desempenharam um papel fundamental no desenvolvimento do trabalho. Essa seção forneceu insights sobre as escolhas técnicas, plataformas, linguagens de programação e o uso específico das ferramentas que impulsionaram o projeto, permitindo aos leitores entender o contexto técnico do trabalho e como ele foi realizado com sucesso.

2.5.1 Django *Framework*

O Framework Django é amplamente reconhecido por sua eficiência e poder na criação de aplicativos da web robustos(SAITO, 2021). O Django faz com que seja mais fácil construir aplicativos web de qualidade, com menos código, e de forma mais rápida do que nunca(FOUNDATION, 2023). Ele fornece uma estrutura sólida para o desenvolvimento web, incluindo um sistema de ORM (Object-Relational Mapping) para interação com bancos de dados, um sistema de templates para renderização de páginas HTML e uma arquitetura baseada em MVT (Model-View-Template) que promove uma organização limpa e escalável do código-fonte(BAE, 2019).

2.5.2 Django Rest *Framework*

O Django Rest *Framework* (DRF) é uma poderosa extensão do Django que simplifica o desenvolvimento de APIs RESTful em aplicativos web(SANTOS; MONTEIRO, 2020). Ele oferece funcionalidades abrangentes para serialização de dados, autenticação, autorização, visualização da API e manipulação de solicitações HTTP, tornando o processo de criação de APIs robustas e escaláveis uma tarefa eficiente(DUFRESNE; SOBOLEV, 2018). Além disso, o DRF integra-se perfeitamente com o Django, permitindo que os desenvolvedores aproveitem os recursos do Django enquanto criam APIs completas(DUFRESNE; SOBOLEV, 2018).

2.5.3 Pycharm

O PyCharm é um ambiente de desenvolvimento integrado (IDE) voltado para programação em Python, conhecido por sua eficiência e recursos avançados(JOHNSON, 2019). O PyCharm é projetado por desenvolvedores, para desenvolvedores, para fornecer todas as ferramentas necessárias para um desenvolvimento produtivo e eficiente em Python (JETBRAINS, 2023). Ele oferece recursos como realce de sintaxe, depuração, análise de código, refatoração inteligente, integração com sistemas de controle de versão e suporte a várias estruturas e bibliotecas Python. Além disso, sua interface de usuário intuitiva e personalizável torna o desenvolvimento Python mais agradável e produtivo(SMITH; BROWN, 2021).

2.5.4 *Visual Studio Code*

O *Visual Studio Code* (VSCode) é um editor de código desenvolvido pela *Microsoft*. O *Visual Studio Code* é um editor de código-fonte que combina a simplicidade de um editor de texto com recursos poderosos de desenvolvimento, como depuração, controle de código-fonte integrado e suporte a extensões (MICROSOFT, 2023). O VSCode é amplamente reconhecido por sua eficiência, flexibilidade e uma comunidade de extensões ativa que oferece suporte a várias linguagens de programação e tecnologias. Sua interface de usuário limpa e personalizável, juntamente com recursos avançados como depuração interativa e integração com sistemas de controle de versão, tornam-no uma escolha popular entre desenvolvedores para uma ampla gama de tarefas de desenvolvimento de *software*(WILSON; THOMPSON, 2020).

2.5.5 *React Native*

O *React Native* é um *framework* de desenvolvimento de aplicativos móveis que permite aos desenvolvedores criar aplicativos nativos para iOS e *Android* usando JavaScript e React. O React Native permite que você crie aplicativos móveis usando apenas JavaScript e React. Ele utiliza os mesmos blocos de construção de UI usados em aplicativos nativos de iOS e *Android*, o que resulta em desempenho e aparência nativos (FACEBOOK, 2023). Ele compartilha grande parte do código entre as plataformas, oferece uma ampla gama de componentes predefinidos e suporta personalização. Reconhecido por sua eficiência e velocidade de desenvolvimento, o *React Native* é favorecido por sua comunidade ativa e bibliotecas de terceiros, tornando-o uma escolha popular para criar aplicativos móveis de alta qualidade de maneira eficaz e econômica(BROWN; WHITE, 2020).

2.5.6 Draw.io

O Draw.io é uma ferramenta de diagramação online que permite criar diagramas e gráficos facilmente. Draw.io é uma ferramenta de diagramação on-line gratuita e de código aberto que pode ser usada para criar diagramas de fluxo, diagramas de processo, organogramas, wireframes e muito mais (JGRAPH, 2023). Essa plataforma é conhecida por sua interface simples e intuitiva, que torna a criação de diagramas fácil para usuários de todos os níveis de habilidade. Draw.io oferece recursos como arrastar e soltar elementos, uma ampla variedade de formas e ícones, bem como a capacidade de exportar e compartilhar seus diagramas em vários formatos, tornando-o uma escolha popular para a criação de diagramas profissionais e colaborativos(HARRISON; PATEL, 2020).

2.5.7 Figma

O Figma é uma ferramenta de design de interface de usuário (UI) e prototipagem colaborativa baseada na web que ganhou reconhecimento significativo na indústria de design(BAXTER;

CHAN, 2021). Ele oferece uma plataforma completa para criar designs de alta qualidade, protótipos interativos e colaboração em tempo real entre membros da equipe. O Figma é elogiado por sua facilidade de uso, recursos avançados de design, capacidade de trabalhar em múltiplos dispositivos e plataformas, bem como por sua capacidade de permitir que equipes remotas colaborem de maneira eficaz. Além disso, a natureza baseada na web do Figma facilita o acesso e a colaboração em projetos de design em qualquer lugar do mundo(CHAN; PATEL, 2020).

[1]

[1] Alguns trechos desse capítulo foram escritos com o auxílio de uma IA generativa, especificamente o ChatGPT da OpenAI.

3

Estudo de Soluções Similares

Neste capítulo, serão apresentados os resultados da busca por soluções semelhantes em lojas de aplicativos, artigos científicos e bases de patentes, além de realizar um comparativo entre as soluções encontradas e a proposta desenvolvida, proporcionando embasamento e fundamentação para a validação da solução proposta neste trabalho de desenvolvimento do sistema de gestão escolar.

3.1 Pesquisa em Lojas de Apps

Esta seção apresenta a etapa de pesquisa de soluções semelhantes em lojas de aplicativos. As pesquisas foram realizadas nas lojas de aplicativos da *Play Store* e *App Store* e evido a grande quantidade de resultados foi definido o critério de selecionar soluções que possuam uma avaliação acima de 4 (quatro) estrelas, que além de reduzir a quantidade de aplicativos garante que as soluções utilizadas como referência sejam mais eficientes, possuam características alinhadas com sistema em desenvolvimento, e ajudem a produzir um produto com mais qualidade. As pesquisas realizadas foram computadas e os resultados inseridos na Tabela 1:

Tabela 1 – Pesquisa de *Apps*

Termo de busca	***Play Store***	***App Store***	**Total**
Gestão escolar	25	14	39
Escola pública	11	1	12
Agenda escolar	23	10	33
Total	59	25	84

Fonte: Autor (2024).

Da tabela de apps pesquisados (Tabela 1) foram selecionados 5 (cinco) aplicativos que

serão utilizados como referência no desenvolvimento deste projeto. Esses aplicativos selecionados possuem características alinhadas com as necessidades desejadas para a aplicação de gestão escolar proposta, pois todos são para gestão de escolas públicas. O Quadro 1 exibe as informações dos aplicativos.

Quadro 1 – Aplicativos selecionados

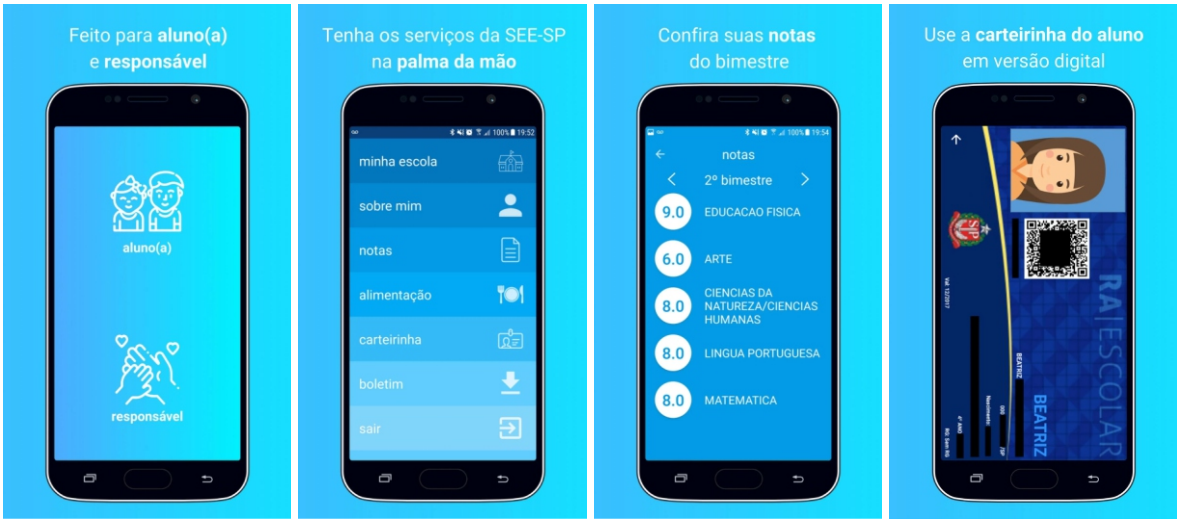
Nome	Link de Acesso
Minha Escola SP	https://play.google.com/store/apps/details?id=br.gov.sp.educacao.minhaescola
Escola Paraná	https://play.google.com/store/apps/details?id=br.gov.pr.celepar.escolapr
Escola RS	https://play.google.com/store/apps/details?id=br.com.procergs.estudante
NetEscola	https://play.google.com/store/apps/details?id=br.gov.go.seduc.netescola
Estudante SC	https://play.google.com/store/apps/details?id=br.sc.gov.ciasc.sed

Fonte: Autor (2024).

3.1.1 Minha Escola SP

O aplicativo Minha Escola SP, desenvolvido pela Secretaria da Educação do Estado de São Paulo, tem como propósito promover o engajamento dos alunos e incentivar a participação das famílias na vida escolar. Com diversas funcionalidades, como informações de contato com a escola, consulta ao histórico de matrículas, acesso ao boletim com visualização simplificada das notas, carteirinha do aluno e avaliação da alimentação escolar, o aplicativo busca facilitar o acesso às informações relevantes e promover uma maior interação entre a escola, os alunos e seus responsáveis.

Figura 3 – Telas do app Minha Escola SP

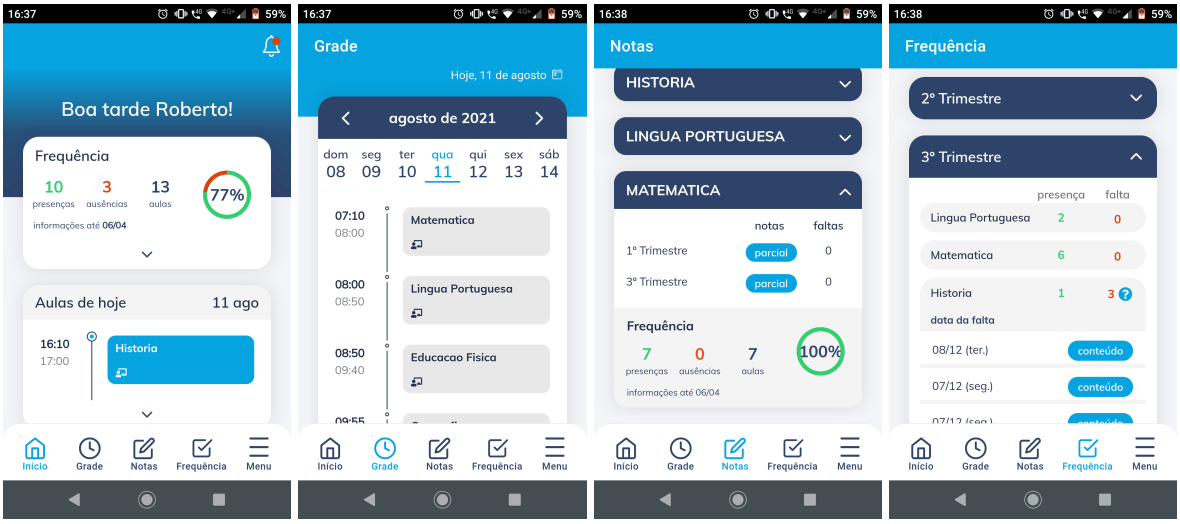


Fonte: Minha Escola SP (2018).

3.1.2 Escola Paraná

O aplicativo desenvolvido para os estudantes e responsáveis da rede de ensino do Estado do Paraná tem como objetivo facilitar o acesso às informações escolares, proporcionando aos alunos uma melhor organização e acompanhamento de suas atividades e envolvendo os responsáveis de forma virtual na sala de aula. O aplicativo oferece diversas funcionalidades, como o acompanhamento da frequência diária por disciplina e período, o acesso às notas finais e parciais, a disponibilidade da grade de aulas com conteúdo aplicado, planejamento e material de apoio de cada disciplina, o calendário de atividades escolares e datas importantes, avisos relevantes, notícias da Secretaria da Educação e um canal de atendimento para os responsáveis.

Figura 4 – Telas do app Escola Paraná

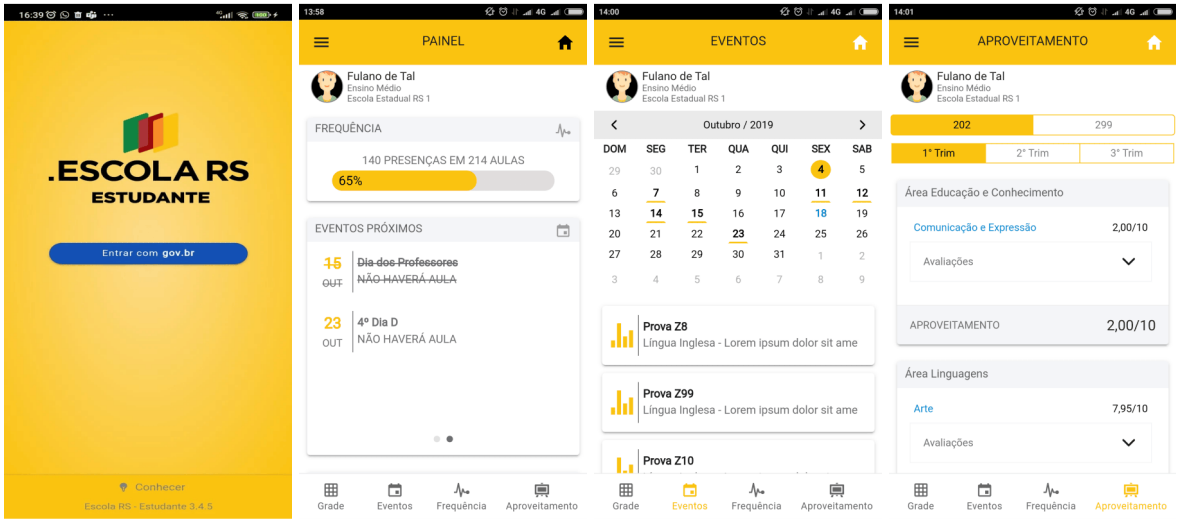


Fonte: Escola Paraná (2017).

3.1.3 Escola RS

O aplicativo "Escola RS"foi desenvolvido para atender aos pais e alunos da rede pública estadual de ensino do estado do Rio Grande do Sul. Ele oferece diversas funcionalidades que permitem acompanhar a vida escolar do estudante, consultar a grade horária, monitorar a frequência, visualizar o calendário com eventos importantes como avaliações e dias não letivos, além de verificar o aproveitamento nas avaliações. Além disso, o aplicativo está integrado ao ISE (Sistema de Controle e Gestão da rede estadual de ensino do RS), garantindo uma maior eficiência e sincronização de dados entre as escolas e os usuários do aplicativo.

Figura 5 – Telas do app Escola RS



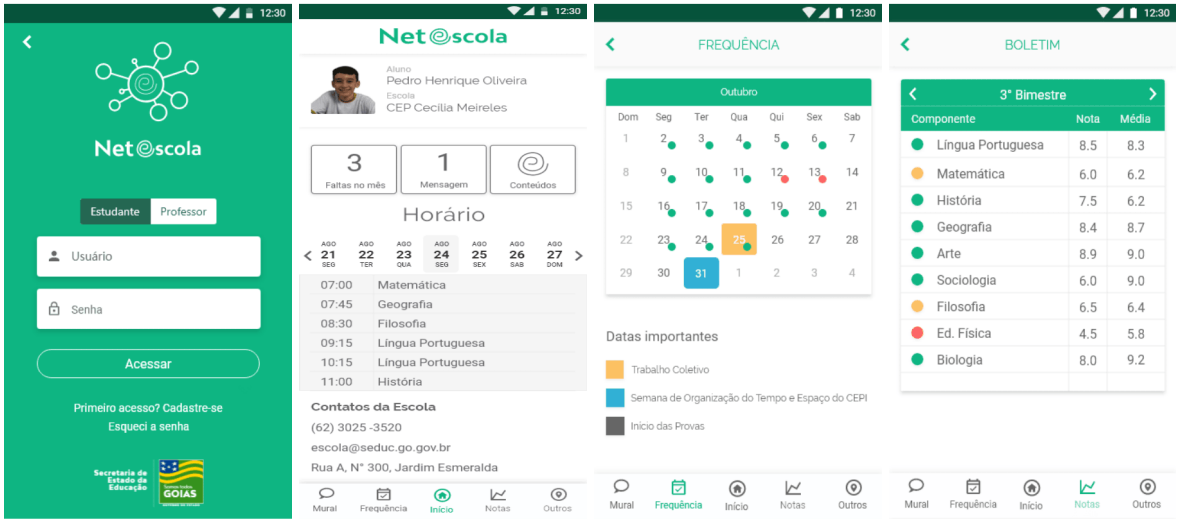
Fonte: Escola RS (2019).

3.1.4 NetEscola

O aplicativo NetEscola, desenvolvido pela Secretaria de Estado da Educação de Goiás, oferece suporte aos alunos por meio de conteúdos complementares para as aulas presenciais e, principalmente, para as aulas não presenciais. Com material disponível para todas as etapas de ensino, o aplicativo permite o acesso a qualquer hora e em qualquer lugar, proporcionando um aprendizado sem fronteiras.

Em breve, uma nova versão do aplicativo estará disponível, oferecendo funcionalidades adicionais para pais e alunos. Essas funcionalidades incluem o acesso às notas, registro de faltas, mural de recados, calendário escolar e quadro de horários. Essas ferramentas adicionais visam promover uma maior interação e transparência entre a escola, os alunos e seus responsáveis.

Figura 6 – Telas do app NetEscola

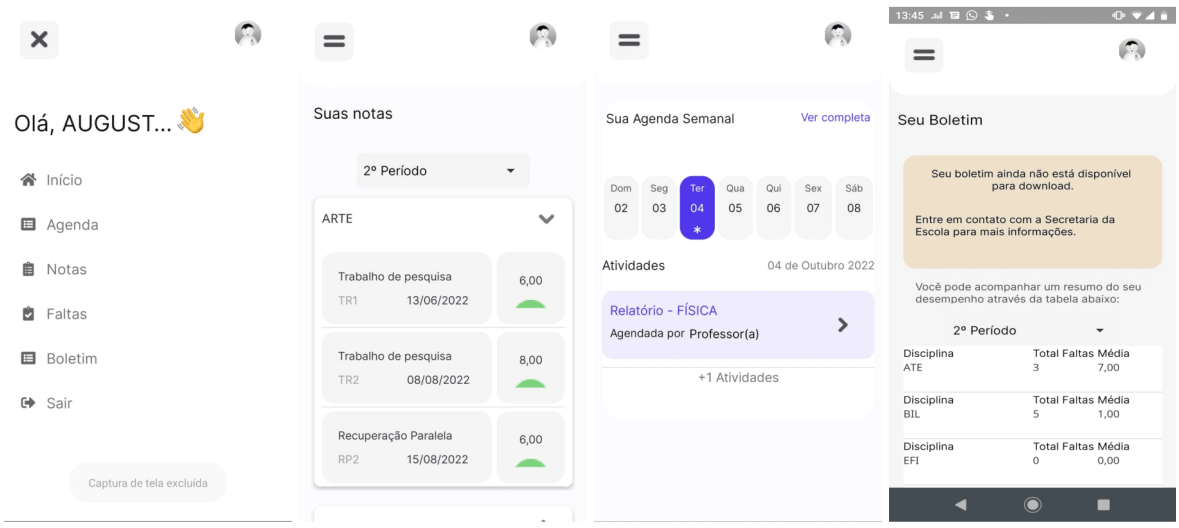


Fonte: NetEscola (2020).

3.1.5 Estudante SC

O aplicativo EstudanteSC foi desenvolvido para possibilitar o acompanhamento escolar dos estudantes da rede estadual de ensino do Estado de Santa Catarina. Por meio dessa ferramenta, os estudantes têm a facilidade de conferir suas notas, visualizar boletins, verificar a infrequência, além de acessar as agendas de provas, trabalhos e tarefas. Com o EstudanteSC, busca-se proporcionar aos estudantes uma forma prática e eficiente de acompanhar seu desempenho acadêmico e manter-se atualizado com as atividades escolares.

Figura 7 – Telas do app Estudante SC



Fonte: Estudante SC (2017).

3.2 Análise das Soluções Encontradas

Considerando que o objetivo das soluções pesquisadas é o mesmo, que é auxiliar na gestão acadêmica de escolas públicas, é esperado que algumas delas possuam funcionalidades similares, mesmo que implementadas de formas diferentes. Com base nessa premissa, foram identificadas e selecionadas as principais funções encontradas na seção 3.1, apresentadas no Quadro 2. Essas funcionalidades representam características relevantes que serão analisadas em relação ao desenvolvimento do sistema de gestão escolar em estudo.

Quadro 2 – Características observadas nos aplicativos

Código	**Característica**
C01	Dados da escola
C02	Dados pessoais
C03	Agenda
C04	Frequência
C05	Notas
C06	Merenda
C07	Mural de avisos
C08	Recado aluno
C09	Histórico de matrículas
C10	Histórico de matrículas
C11	Informações do transporte
C12	Tarefas da escola
C13	Tarefas de casa

Fonte: Autor (2024).

Após a seleção dos aplicativos listados no Quadro 1, foi realizado um comparativo entre suas funcionalidades e um levantamento das funcionalidades desejadas para a solução proposta, levando em consideração as características identificadas no Quadro 2. As funcionalidades desejadas podem ser observadas no Quadro 3, representando os requisitos para o desenvolvimento da solução proposta.

Quadro 3 – Tabela comparativa de funcionalidades

	M. Escola SP	**Escola Paraná**	**Escola RS**	**NetEscola**	**Escola SC**	**SEEDuca**
C01	X			X		X
C02	X	X	X	X	X	X
C03	X	X	X	X	X	X
C04	X	X	X	X	X	X
C05	X	X	X	X	X	X
C06	X					X
C07		X		X		X
C08		X				X
C09	X					X
C10	X					X
C11						X
C12						X
C13						X

Fonte: Autor (2024).

Após realizar uma pesquisa abrangente sobre os aplicativos de gestão escolar desenvolvidos por outras secretarias de educação, foi possível identificar um conjunto significativo de funcionalidades importantes já disponíveis nesses aplicativos. Percebeu-se a necessidade de agregar essas funcionalidades em um único aplicativo, especificamente voltado para atender às demandas das escolas públicas do estado, além de agregar novas funcionalidades relevantes ao atual contexto escolar.

Essa abordagem permitirá oferecer uma solução completa e integrada, que atenda às necessidades específicas das escolas públicas e contribua para uma gestão mais eficiente e eficaz do ambiente educacional. Ao reunir as funcionalidades existentes e adicionar novos recursos, busca-se oferecer um aplicativo que seja adaptável, intuitivo e que proporcione uma experiência de uso aprimorada para os usuários envolvidos no processo educacional.

3.3 Pesquisa em Base de Patentes

Foi realizada uma busca específica por registros de *software* na base de patentes do INPI, desta vez relacionados ao tema de sistemas de gestão escolar. Os termos de busca utilizados foram os mesmos empregados na pesquisa em lojas de apps, conforme apresentados na Tabela 1. No total, foram encontrados 28 resultados relacionados, porém nenhum resultado foi encontrado pesquisando com termo "escola pública", justamente o termo dos apps predominantes na análise do Quadro 2 e alvo da solução proposta nesse trabalho. Dentre esses resultados, foram selecionados os que mais se alinham com o escopo deste trabalho e que contêm informações relevantes, sendo detalhados no Quadro 4 para referência e análise posterior.

Quadro 4 – Resultado da busca do INPI

Pedido	**Título**
BR 51 2020 001012 1	Education1 sistema gestão escolar para instituições de ensino
BR 51 2018 000115 7	Portal Educanet *Web* - Sistema de Gestão Escolar

Fonte: Autor (2024).

Dentre os resultados encontrados foram listados o Education1, que se trata de um sistema de gestão escolar completo para instituições particulares de ensino e o Educanet Web que é um sistema de gestão escolar utilizado por algumas escolas municipais do estado de Minhas Gerais. O Quadro 5 exibe o comparativo das funcionalidades encontradas nos sistemas encontrados com as características relacionadas no Quadro 2.

Quadro 5 – Tabela comparativa de funcionalidades na base de patentes

	Education1	**Educanet *Web***
C01	X	X
C02		X
C03	X	X
C04	X	X
C05	X	X
C06		X
C07	X	
C08		
C09		X
C10		
C11		X
C12	X	
C13	X	

Fonte: Autor (2024).

3.4 Pesquisa em Artigos Científicos

Esta seção apresenta os resultados da busca por artigos científicos relacionados à gestão escolar. Para realizar essa busca, foram utilizadas as bases de dados *Scopus* e *Web of Science*, empregando strings de busca que se alinham com a temática deste trabalho. As strings de busca utilizadas estão disponíveis no Quadro 6 para referência.

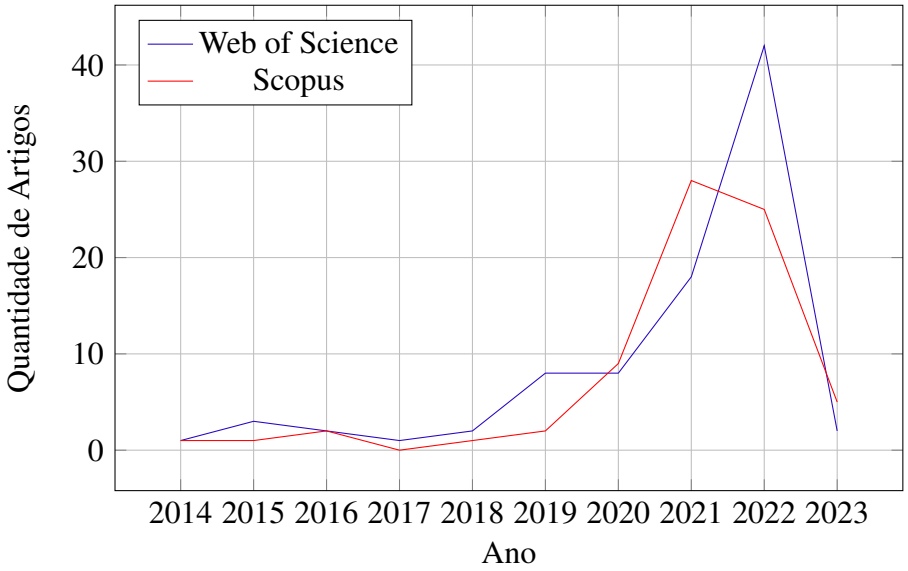
Quadro 6 – Quadro comparativo das strings de busca nas bases de artigos

Base de dados	**String de busca**
Web of Sciences	Tópico = ((*software OR app OR application*) *AND* (*management OR administration*) *AND school*)
Scopus	*TITLE-ABS-KEY*((*software OR app OR application*) *AND* (*management OR administration*) *AND school*)

Fonte: Autor (2024).

Por se tratar de um tem abrangente a busca de anterioridade retornou um número grande de resultados onde foi necessário realizar um refinamento maior na pesquisa. Na base da *"Web of Science"* além da string de busca relatada no Quadro 6 foi realizado um refinamento por **tipo de documento: artigo** e **categoria: Ciências da Computação Sistemas da Informação**, que nos retornou um resultado mais enxuto e direcionado na pesquisa com 87 resultados nos últimos 10 anos. Já na base *"Scopus"* o refinamento foi realizado o refinamento por **tipo: artigo**, **área: Ciencias da Computação** e **palavras chave: (computação educacional, sistemas de gerenciamento, tecnologia educacional e sistemas de administração)**, que resultaram em 74 artigos durante o mesmo intervalo de tempo.

Gráfico 1 – Gráfico dos artigos publicados sobre o tema do sistema



Fonte: Autor (2024).

Com o objetivo de aprofundamento, foram selecionados os artigos mais relevantes de cada base de dados para a análise. A partir desses artigos, foi feita uma leitura detalhada para identificar as funcionalidades relacionadas ao escopo do sistema de gestão escolar, principalmente com base no Quadro 2. Os resultados obtidos da base Web of Science estão resumidos no Quadro 7.

Quadro 7 – Resultado da pesquisa na base *Web Of Science*

Título	**Resumo**	**Referência**
Design of Smart Classroom System Based on Internet of Things Technology and Smart Classroom	O artigo aborda o ensino inteligente em sala de aula com apoio da tecnologia, como a Internet das Coisas. O estudo foca no sistema de sala de aula inteligente e como a coleta de dados é realizada para otimizar o ambiente. Os resultados demonstram que essa abordagem pode fornecer referência para a construção de salas de aula inteligentes em outras universidades.	(ZHANG; LI, 2021)
Design of the Student Attendance Management System Based on the Internet of Things Technology	O artigo explora o uso da IoT na gestão de frequência estudantil, com foco na precisão e coleta eficiente de dados. Identifica a influência da personalidade dos alunos na taxa de frequência e destaca a importância da melhoria contínua do sistema.	(XUN; GOSTARIA; DUAN, 2022)

Fonte: Autor (2024).

Foram analisados vários sistemas, mas apenas dois se assemelharam ao escopo do projeto, conforme visto no Quadro 7. O primeiro busca otimizar o ambiente educacional e melhorar a comunicação entre a instituição e os alunos, enquanto o segundo visa gerenciar a frequência estudantil. Essas características já estão contempladas no projeto, não sendo necessário acrescentá-las. Além disso, ambos os artigos introduzem a Tecnologia de Internet das Coisas (IoT) para a coleta de dados, mas essa funcionalidade não será integrada devido às limitações de recursos, uma vez que o sistema é voltado para escolas públicas.

Agora no Quadro 8 é possível visualizar os resultados da análise da base de dados Scopus.

Quadro 8 – Resuldado da pesquisa na base Scopus

Título	**Resumo**	**Referência**
Wearable mobile internet devices involved in big data solution for education	Desenvolvimento de um modelo de big data na educação, incluindo a coleta de dados de atividades por meio de dispositivos móveis vestíveis e a verificação do modelo com aplicativos de otimização escolar.	*Sun et al. (2016)*
Research on construction of normative ideological instruction teaching management system based on ZigBee wireless sensor network	Desenvolvimento de um sistema de gestão educacional eficiente, baseado em ZigBee e tecnologia *Go-Ahead*, visando otimizar o ensino, monitorar remotamente a instrução, e melhorar a gestão de professores e alunos.	Pan (2014)

Fonte: Autor (2024).

Durante a pesquisa na base *Scopus*, foram identificados dois artigos alinhados com o escopo do projeto, exibidos no Quadro 8. O primeiro aborda o uso de big data para análise de dados coletados por dispositivos móveis vestíveis. O segundo emprega ZigBee e tecnologia *Go-Ahead* para coleta de dados, otimização do ensino e aprimoramento da gestão escolar. Similarmente à pesquisa na base Web of Science, não foram encontradas características adicionais relevantes para o projeto. Em nenhum dos artigos, em ambas as pesquisas, possível testar as soluções e as características foram levantadas a partir das informações dos artigos.

4

Desenvolvimento do Produto de *Software*

Neste capítulo, abordamos de forma detalhada o processo de desenvolvimento do nosso sistema de gestão escolar, focalizando principalmente na fase inicial de coleta de informações. Iniciamos com uma pesquisa abrangente, realizada com potenciais usuários e partes interessadas, a fim de compreender suas necessidades, desafios e expectativas em relação ao *software*. Essa pesquisa não apenas permitiu a obtenção de insights valiosos, mas também serviu como base sólida para a etapa subsequente: a elicitação de requisitos de *software*. Ao explorar as informações coletadas, identificamos os principais requisitos funcionais e não funcionais que o sistema deve incorporar para atender de forma eficaz e abrangente às demandas do ambiente escolar. Além disso, neste capítulo, também introduzimos visualizações essenciais que enriquecem a compreensão do sistema proposto, incluindo o diagrama de casos de uso, o diagrama entidade-relacionamento e o diagrama de classes. Essas representações gráficas fornecem uma visão panorâmica das interações entre os usuários e o sistema, da estrutura dos dados subjacentes e das relações entre os diversos componentes do *software*, estabelecendo assim uma base sólida para o processo de desenvolvimento subsequente.

4.1 Pesquisa Sobre Gestão Escolar

Com o intuito de aprofundar nossa compreensão das necessidades específicas dentro do contexto de um sistema de gestão escolar, realizamos um estudo minucioso que se concentrava nas áreas chave da administração educacional. Esse estudo foi conduzido por meio de um questionário eletrônico hospedado no *Google Forms*, permitindo a coleta de dados significativos para a identificação das dificuldades enfrentadas pelos principais atores das instituições de ensino. Visando garantir a confidencialidade das respostas e incentivar a participação aberta, todas as perguntas foram formuladas de maneira anônima, ressaltando que o objetivo primordial era adquirir um conhecimento aprofundado do tópico e mapear funcionalidades de vital importância. A pesquisa foi divulda em grupos de aplicativos como WhatsApp e plataformas como o Facebook,

com o intuito de alcançar uma ampla gama de profissionais e usuários do setor. Os detalhes das perguntas empregadas estão disponíveis no Apêndice A para referência.

4.1.1 Perfil dos Pesquisados

Os formulários de pesquisa receberam um total de 41 respostas, distribuídas entre 14 (34,1%) respostas para o formulário direcionado aos membros das escolas e 27 (65,9%) respostas para o formulário destinado aos pais e responsáveis. Portanto, é necessário analisar os resultados separadamente para cada grupo.

4.1.1.1 Perfil dos membros das escolas

Entre os participantes que responderam ao formulário destinado aos membros das escolas, 13 (92,9%) são originários do estado de Sergipe, enquanto 1 (7,1%) é do estado de Alagoas. Esses resultados eram esperados, dado que a pesquisa foi amplamente divulgada em Sergipe. Quanto aos papéis desempenhados por esses membros, 2 (14,3%) são diretores, 3 (21,4%) são coordenadores e 9 (64,3%) são professores. Esses números estão de acordo com o esperado, levando em consideração o perfil dos respondentes e suas funções na escola. No que diz respeito ao tipo de escola em que trabalham, 7 (50%) atuam em escolas particulares, 2 (14,3%) em escolas municipais, 4 (28,6%) em escolas estaduais e 1 (7,1%) em instituições de ensino federal. Isso indica que metade dos participantes faz parte do público-alvo deste projeto, uma vez que trabalham em escolas públicas.

4.1.1.2 Perfil dos alunos e responsáveis

No grupo que respondeu ao formulário destinado aos alunos e responsáveis, 1 (3,7%) é proveniente do estado do Mato Grosso do Sul, 8 (29,6%) do estado de Alagoas e 18 (66,7%) do estado de Sergipe. Essa distribuição também estava de acordo com as expectativas, considerando a ampla divulgação da pesquisa em Sergipe. Em relação ao tipo de usuário, 4 (14,8%) se identificaram como alunos, 18 (66,7%) como responsáveis por alunos e 5 (18,5%) como outros. Quanto ao perfil das escolas onde os alunos estudam, 2 (7,4%) afirmaram estudar em escolas federais, 6 (22,2%) em escolas estaduais, 18 (66,7%) em escolas particulares e 1 (3,7%) selecionou a opção "outros". A última resposta indica que a pesquisa não atingiu completamente o perfil do público-alvo deste projeto, uma vez que mais da metade dos respondentes pertence a escolas particulares.

4.1.2 Resultados da Pesquisa

A análise detalhada dos resultados da pesquisa se faz tão fundamental quanto a avaliação do perfil dos participantes, requerendo uma abordagem distinta entre os membros das escolas e os alunos e responsáveis.

4.1.2.1 Resultados da pesquisa de membros das escolas

Quando questionados sobre a forma como gerenciam as disciplinas que estão lecionando, os participantes apresentaram as seguintes respostas: 2 (14,3%) utilizam documentos impressos ou arquivos digitais, 3 (21,4%) acessam via sistema web/local, 4 (28,6%) usam o aplicativo da escola, 4 (28,6%) optam pela planilha Excel, e 1 (7,1%) selecionou a opção "outros".

Em relação à consulta das turmas que estão ministrando aulas, as respostas foram distribuídas da seguinte forma: 1 (7,1%) utiliza sistema web/local, 3 (21,4%) optam pela planilha Excel, 4 (28,6%) recorrem a documentos impressos ou arquivos digitais, 5 (35,7%) utilizam o aplicativo da escola e 1 (7,1%) selecionou a opção "outros".

Quanto à gestão da agenda escolar das turmas, as preferências foram as seguintes: 2 (14,3%) utilizam sistema web/local, 3 (21,4%) escolhem a planilha Excel, 4 (28,6%) optam pelo aplicativo da escola, 4 (28,6%) preferem documentos impressos ou arquivos digitais, e 1 (7,1%) selecionou a opção "outros".

No que diz respeito à administração das atividades em sala de aula, as respostas foram distribuídas da seguinte forma: 1 (7,1%) fazem uso do sistema web/local, 3 (21,4%) utilizam o aplicativo da escola, 8 (57,1%) recorrem a documentos impressos ou arquivos digitais, e 2 (14,3%) selecionaram a opção "outros".

Em relação à supervisão das atividades de casa dos alunos, as preferências foram as seguintes: 1 (7,1%) opta pelo sistema *web*/local, 2 (14,3%) utilizam documentos impressos ou arquivos digitais, 3 (21,4%) recorrem ao aplicativo da escola, e 8 (57,1%) fazem anotações em cadernos/agendas.

Quando se trata da organização e registro das notas dos alunos, as escolhas foram as seguintes: 2 (14,3%) preferem o sistema *web*/local, 2 (14,3%) utilizam documentos impressos ou arquivos digitais, 3 (21,4%) optam pela planilha Excel, 3 (21,4%) fazem anotações em cadernos/agendas, e 4 (28,6%) utilizam o aplicativo da escola.

Em relação ao controle de frequência dos alunos, as respostas se distribuem da seguinte forma: 1 (7,1%) fazem uso da planilha Excel, 2 (14,3%) utilizam o sistema *web*/local, 2 (14,3%) optam por documentos impressos ou arquivos digitais, 3 (21,4%) fazem anotações em cadernos/agendas, e 6 (42,8%) preferem o aplicativo da escola.

A partir deste ponto, as respostas diminuem, visto que as questões se voltam para coordenadores e diretores, excluindo professores. Quando questionados sobre como gerenciam informações dos alunos da escola, 1 (20%) respondeu utilizando o sistema *web*/local, enquanto 4 (80%) armazenam pastas/fichários com documentos dos professores.

No que se refere ao gerenciamento das informações dos professores da escola, 1 (20%) respondeu utilizando o sistema *web*/local e 4 (80%) armazenam pastas/fichários com documentos dos professores.

Quando abordados sobre como gerenciam as turmas da escola, 1 (20%) utiliza o sistema web/local, 1 (20%) opta pela planilha Excel e 3 (60%) fazem anotações em cadernos/agendas.

No tocante ao gerenciamento do material escolar e uniformes, 3 (60%) optam por pastas/fichários com informações da escola e 2 (40%) fazem anotações em cadernos/agendas.

No que diz respeito a recursos adicionais considerados importantes em um sistema de gestão escolar, 3 (60%) mencionaram a gestão de dados do transporte escolar, 1 (20%) a gestão de informações da merenda e 1 (20%) a possibilidade de enviar mensagens para alunos/responsáveis.

Nas últimas duas questões, o perfil dos respondentes muda novamente, sendo composto exclusivamente por diretores de escola. Quando questionados sobre como são administradas as informações da escola, todos os participantes responderam que utilizam pastas/fichários com informações da escola.

Em relação ao gerenciamento das informações dos coordenadores, 1 (50%) respondeu que é feito por meio de pastas/fichários com informações da escola, enquanto 1 (50%) utiliza anotações em cadernos/agendas

4.1.2.2 Resultados de pesquisa dos alunos e responsáveis

Quando indagados sobre como acessam a agenda na escola, as respostas variaram da seguinte forma: 2 (7,4%) optam por uma folha impressa entregue aos alunos, 4 (14,8%) utilizam o sistema *web*, 5 (18,5%) recorrem ao aplicativo da escola, 15 (55,6%) fazem anotações em cadernos/agendas, e 1 (3,7%) escolheu a opção "outros".

No que se refere à comunicação de avisos na escola, as escolhas foram distribuídas da seguinte maneira: 2 (7,4%) mencionaram o mural da escola, 3 (11,1%) preferem o sistema *web*, 4 (14,8%) recebem avisos por meio de folhas impressas entregues aos alunos, 5 (18,5%) utilizam o aplicativo da escola, 9 (33,3%) fazem anotações em cadernos/agendas, e 4 (14,8%) optaram por outra alternativa.

Quanto à apresentação das atividades de casa aos alunos, as respostas se distribuíram da seguinte forma: 1 (3,7%) mencionou o uso do aplicativo da escola, 3 (11,1%) preferem o sistema *web*, 8 (29,6%) recebem atividades em folhas impressas entregues aos alunos, 12 (44,4%) fazem anotações em cadernos/agendas, e 3 (11,1%) escolheram a opção "outros".

No que diz respeito à consulta das notas, 1 (3,7%) utiliza anotações em caderno/agenda, 3 (11,1%) acessam pelo aplicativo da escola, 6 (22,2%) preferem o sistema *web*, 9 (33,3%) têm acesso ao boletim impresso entregue aos alunos, e 8 (29,6%) optaram por outra alternativa.

Em relação à verificação da frequência escolar pelos alunos ou responsáveis, 3 (11,1%) mencionaram o uso do aplicativo da escola, 5 (18,5%) preferem o sistema *web*, 6 (22,2%) têm acesso ao boletim impresso entregue aos alunos, 6 (22,2%) fazem anotações em cadernos/agendas, e 7 (25,9%) escolheram a opção "outros".

Quando questionados sobre recursos ou funcionalidades adicionais que considerariam importantes em um sistema de gestão escolar, 3 (11,1%) destacaram a importância de visualizar dados do transporte, 3 (11,1%) desejam acessar informações sobre a merenda, 17 (63%) gostariam de ter a opção de enviar mensagens para a escola/professores, e 4 (14,8%) escolheram outra alternativa.

4.1.3 Conclusão da pesquisa

Os resultados desta pesquisa sobre gestão escolar apontam para a importância do desenvolvimento do sistema SEEduca. As respostas variadas dos membros das escolas, alunos e responsáveis indicam a necessidade de uma abordagem flexível para atender às diferentes preferências e práticas em instituições de ensino. Além disso, as demandas por recursos adicionais, como informações de transporte e comunicação eficaz, ressaltam a importância de oferecer funcionalidades abrangentes. Portanto, o sistema em desenvolvimento tem o potencial de atender a essas necessidades específicas e melhorar a gestão escolar, promovendo uma experiência mais eficiente e integrada para todos os envolvidos.

4.2 Requisitos de *Software*

Os requisitos de um *software* representam as especificações das funcionalidades desejadas, dos serviços a serem providos ou das limitações impostas ao seu funcionamento (PRESSMAN; MAXIM, 2020). Dado o papel crítico da elicitação de requisitos ao longo do ciclo de vida do projeto de *software*, os requisitos foram estabelecidos como um meio essencial para a organização e a compreensão eficaz do escopo e das metas do sistema em desenvolvimento.

4.2.1 Elicitação dos Requisitos Funcionais e Não Funcionais

A organização dos requisitos é uma etapa crucial no desenvolvimento de *software* e geralmente envolve a categorização em requisitos funcionais e não funcionais (PRESSMAN; MAXIM, 2020). Os requisitos funcionais descrevem as funcionalidades e os serviços que o sistema deve oferecer, delineando como o sistema deve responder a entradas específicas e comportar-se em cenários definidos. Já os requisitos não funcionais estabelecem as restrições e condições que devem ser atendidas para garantir o funcionamento adequado do sistema como um todo. No contexto deste projeto, os requisitos funcionais e não funcionais estão detalhadamente apresentados nos Quadro 9 e Quadro 10, respectivamente.

Quadro 9 – Requisitos funcionais elicitados

Código	**Título**	**Classificação**	**Atores**
RF01	Os alunos poderão acessar o sistema para visualizar suas informações pessoais	Essencial	Aluno
RF02	Os alunos poderão acessar a agenda escolar	Essencial	Aluno
RF03	Os alunos poderão acessar os avisos da escola	Essencial	Aluno
RF04	Os alunos poderão acessar as tarefas de casa passadas pelos professores	Essencial	Aluno
RF05	Os alunos poderão visualizar suas notas	Essencial	Aluno
RF06	Os alunos poderão gerar e imprimir o boletim escolar	Essencial	Aluno
RF07	Os alunos poderão acessar sua frequência	Essencial	Aluno
RF08	Os alunos poderão acessar os dados do transporte escolar	Importante	Aluno
RF09	Os alunos poderão visualizar o cardápio da merenda	Importante	Aluno
RF10	Os alunos poderão enviar mensagens à escola	Desejável	Aluno
RF11	Os professores poderão consultar suas disciplinas	Essencial	Professor
RF12	Os professores poderão consultar sua turmas	Essencial	Professor
RF13	Os professores poderão gerenciar a agenda escolar de suas turmas	Essencial	Coordenador, Professor, Diretor
RF14	Os professores poderão gerenciar as atividades de casa e sala de aula	Essencial	Coordenador, Professor, Diretor
RF15	Os professores poderão gerenciar as notas dos alunos	Essencial	Coordenador, Professor, Diretor
RF16	Os professores poderão gerenciar a frequência dos alunos	Essencial	Coordenador, Professor, Diretor
RF17	Os coordenadores poderão gerenciar as informações dos alunos da escola	Essencial	Coordenador, Diretor
RF18	Os coordenadores poderão gerenciar as informações dos professores	Essencial	Coordenador, Diretor
RF19	Os coordenadores poderão gerenciar as disciplinas	Essencial	Coordenador, Diretor
RF20	Os coordenadores poderão gerenciar as turmas da escola	Essencial	Coordenador, Diretor
RF21	Os coordenadores poderão gerenciar os Materiais Escolares e Uniformes da escola	Essencial	Coordenador, Diretor
RF22	Os coordenadores poderão gerenciar o cardápio da merenda da escola	Importante	Coordenador, Diretor
RF23	Os diretores poderão gerenciar as informações da escola	Essencial	Diretor
RF24	Os diretores poderão gerenciar as informações dos coordenadores	Essencial	Diretor

Fonte: Autor (2024).

Quadro 10 – Requisitos não funcionais elicitados

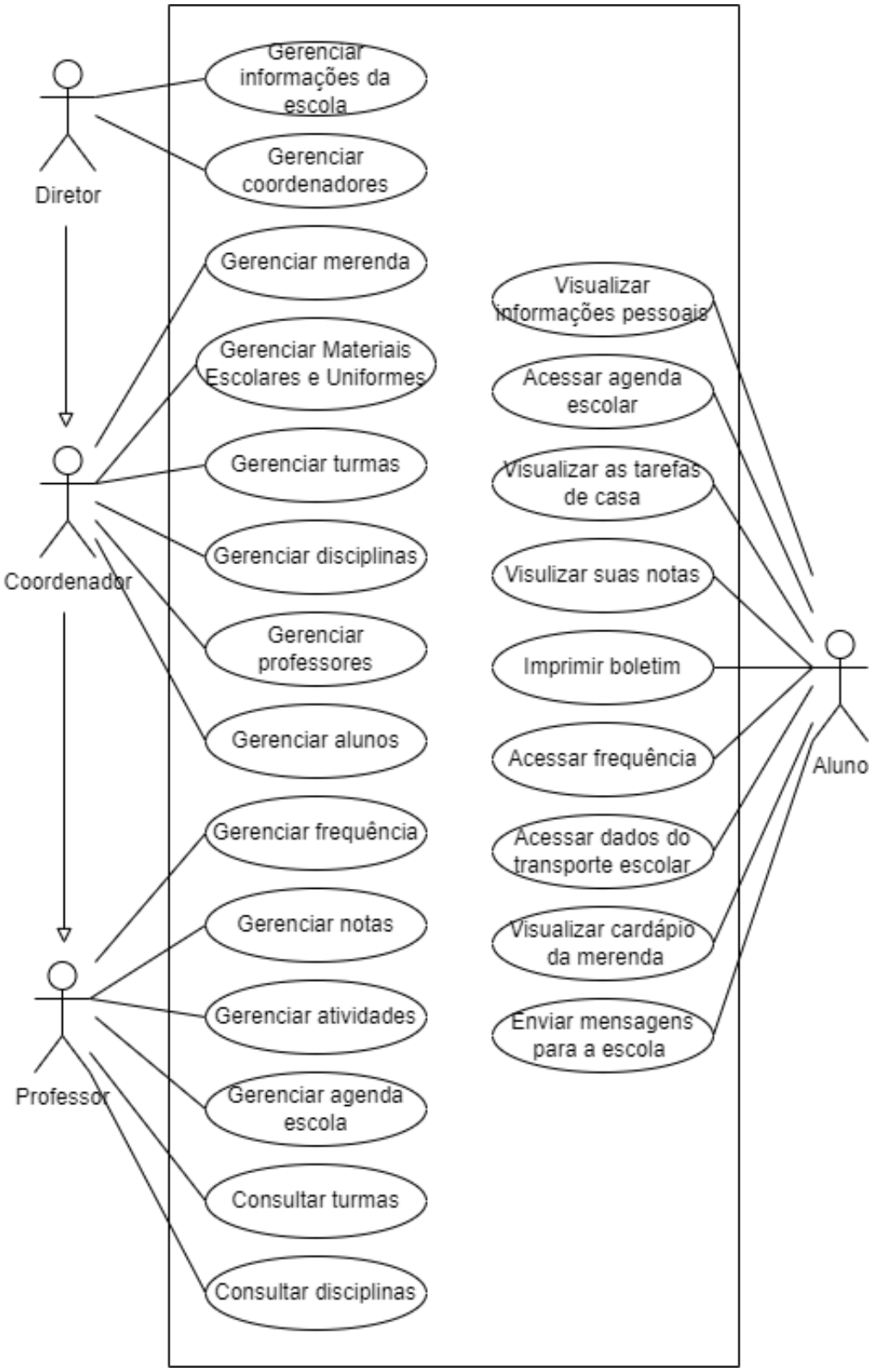
Código	**Título**	**Classificação**	**Tipo**
RNF01	O *software* será uma aplicação web acessível a partir de diversos dispositivos, como computadores, *tablets* e *smartphones*	Essencial	Usabilidade
RNF02	A interface do sistema deve ser responsiva, adaptando-se a diferentes tamanhos de tela e dispositivos móveis	Essencial	Usabilidade
RNF03	O *backend* do sistema será desenvolvido em Python com o auxílio do *framework* Django	Essencial	Implementação
RNF04	O sistema se comunicará com o banco de dados MariaDB para armazenar e recuperar informações	Essencial	Implementação
RNF05	A arquitetura do *software* deve ser modular e escalável, permitindo a adição de novos módulos e funcionalidades no futuro	Essencial	Arquitetura
RNF06	A interface mobile será desenvolvida com a linguagem javascript utilizando o React Native	Essencial	Implementação

Fonte: Autor (2024).

4.3 Diagrama de Casos de Uso

Os requisitos de *software* são frequentemente capturados com a técnica de casos de uso, que descreve as interações entre os usuários e o sistema, proporcionando uma visão narrativa do uso pretendido (PRESSMAN; MAXIM, 2020). Na Figura 8, encontra-se o diagrama de casos de uso que ilustra essas interações.

Figura 8 – Diagrama de casos de uso

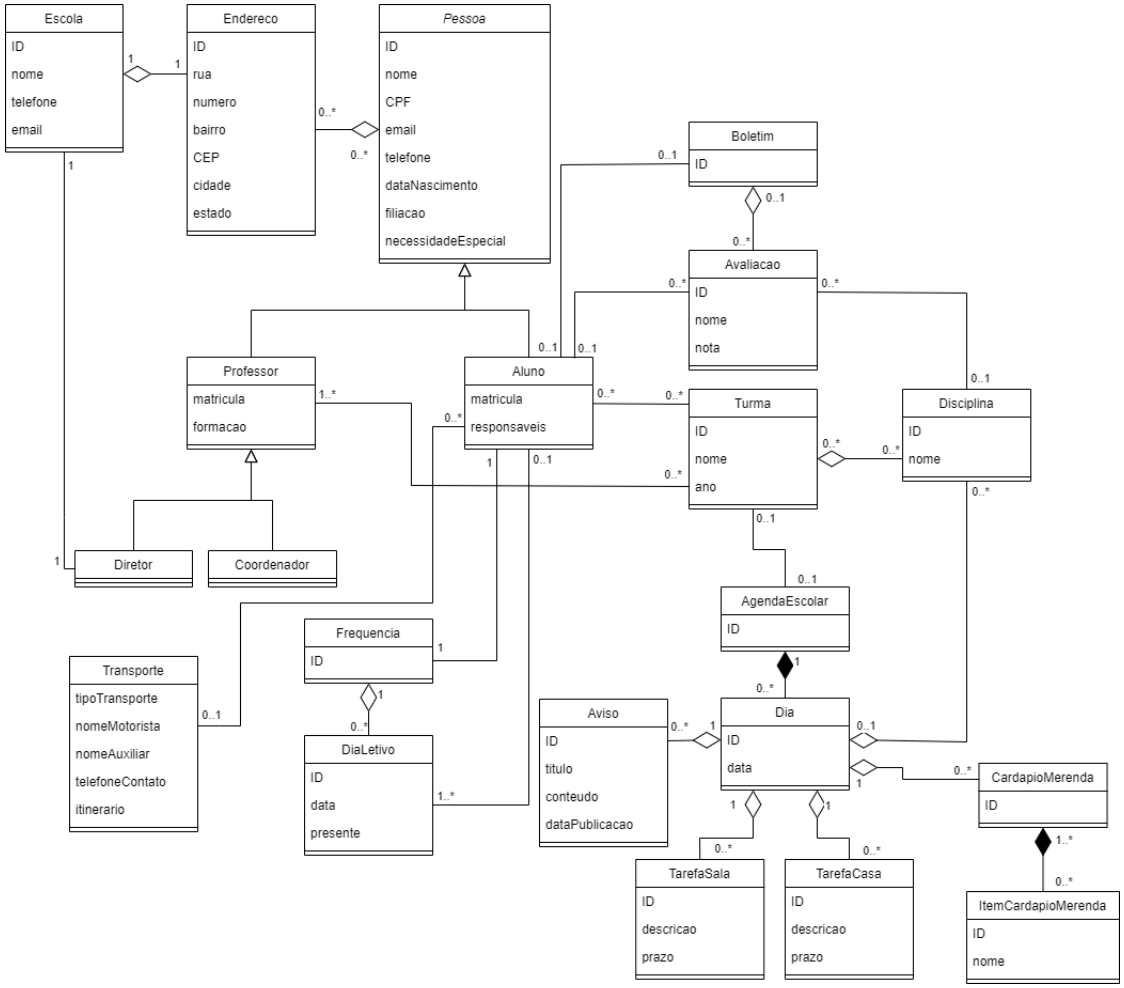


Fonte: Autor (2024).

4.4 Diagrama de Classes

Os diagramas de classe, uma ferramenta crucial na modelagem de sistemas orientados a objetos, são amplamente utilizados para representar as classes de um sistema e suas relações com outras classes. Essas conexões, denominadas associações, servem para demonstrar como essas classes estão interligadas, podendo requerer ou não conhecimento mútuo de suas contrapartes (SOMMERVILLE, 2011). Na Figura 9, vemos a representação das classes do sistema e as suas interações.

Figura 9 – Diagrama de classes



Fonte: Autor (2024).

4.5 Diagramas de Sequência

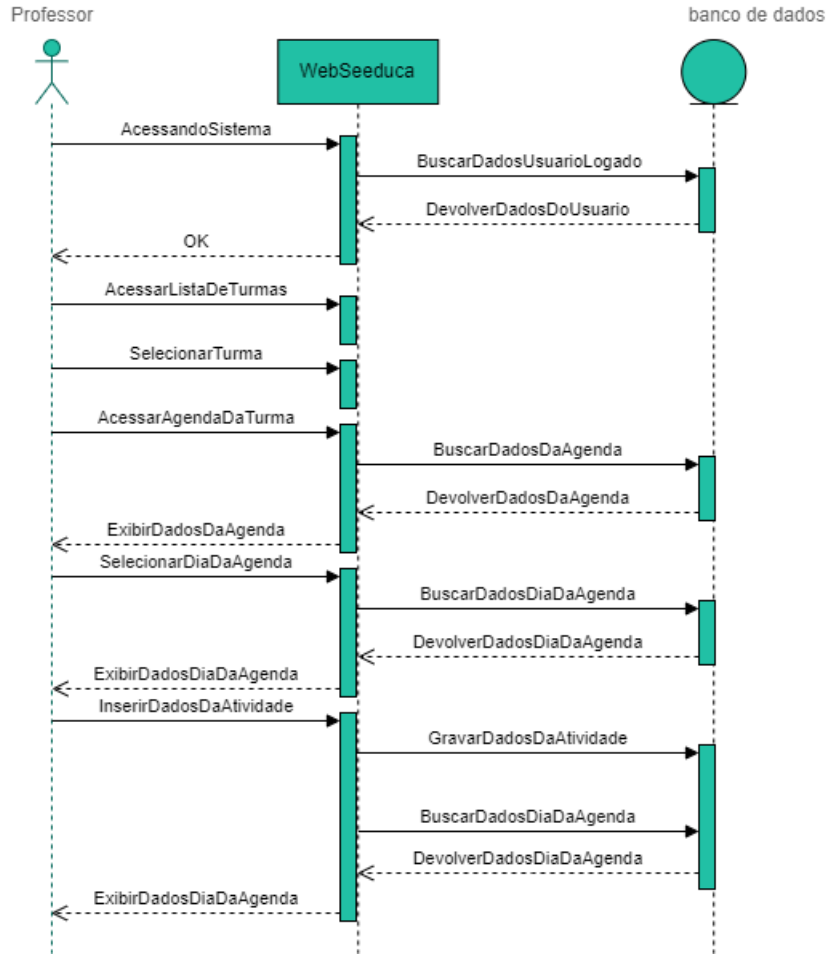
Os diagramas de sequência, uma ferramenta fundamental na modelagem de sistemas orientados a objetos, são utilizados para representar a interação entre objetos em um cenário específico de um sistema. Esses diagramas capturam o comportamento dinâmico do sistema, mostrando como diferentes objetos colaboram ao longo do tempo para realizar uma funcionali-

dade*(GAMMA et al., 1994)*. Em um diagrama de sequência, os objetos são dispostos verticalmente e as mensagens entre eles são representadas por setas horizontais, permitindo uma visualização clara do fluxo de interações.

4.5.1 Professor cadastra atividade

Na Figura 10 é possível visualizar o fluxo de interações entre usuário e sistema no caso de uso onde o ator Professor cadastra uma atividade na agenda de uma de suas turmas.

Figura 10 – Diagrama de sequência - Professor cadastra atividade

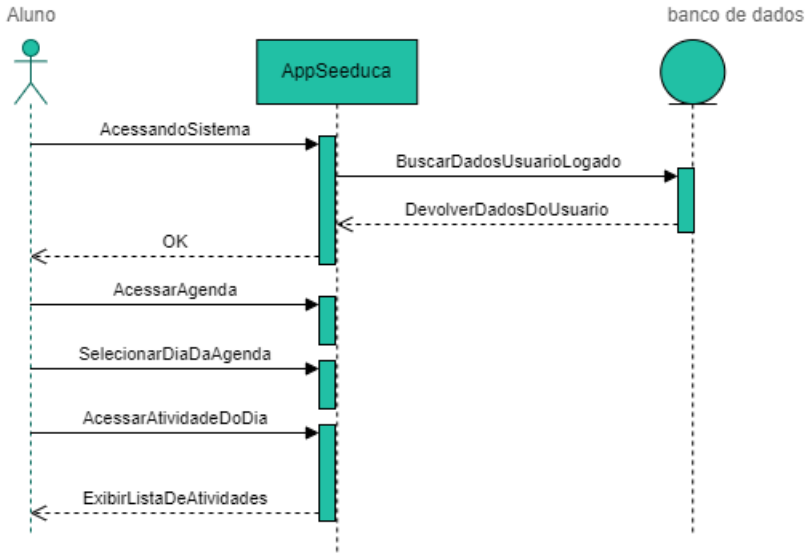


Fonte: Autor (2024).

4.5.2 Aluno consulta atividade de casa

Já na Figura 11 é possível ver o outro lado do sistema onde o ator Aluno consulta as atividades cadastradas em um determinado dia da agenda. A partir desses dois diagramas podemos ver a diferença principal entre os sistemas *web* e *mobile*, onde o primeiro realiza muito mais interações com o banco de dados que o segundo que só faz consulta no momento da autenticação.

Figura 11 – Diagrama de sequência - Aluno consulta atividade de casa



Fonte: Autor (2024).

4.5.3 Coordenador insere cardápio da merenda escolar

A Figura 12 exibe o diagrama de sequência do caso de uso Coordenador insere cardápio de merenda escolar. Como acontece no diagrama da Figura 10 o sistema realiza muitas consultas no banco de dados durante a execução, ao contrário do aplicativo.

Figura 12 – Diagrama de sequência - Coordenador gerencia cardápio da merenda escolar

Fonte: Autor (2024).

4.6 Protótipo de Telas

Com o propósito de aprimorar a visualização da ferramenta e enriquecer sua compreensão, foram desenvolvidos dois protótipos de telas utilizando a plataforma Figma. Estes protótipos foram concebidos para atender às necessidades tanto dos alunos quanto dos integrantes da escola. Eles desempenham um papel crucial ao oferecer uma representação visual das interfaces, tornando mais claro o funcionamento das respectivas funcionalidades, contribuindo assim para a

qualidade e eficácia do projeto.

4.6.1 Protótipo do APP SEEduca

O primeiro protótipo concebido é o APP SEEduca, uma plataforma *mobile* voltada para alunos e responsáveis. A intenção é otimizar sua acessibilidade ao máximo, e para isso, está sendo planejada a implementação em React Native, visando a compatibilidade com uma ampla gama de dispositivos móveis.

4.6.1.1 Tela inicial, Login e Recuperação de Senha

Abrindo o aplicativo o usuário visualizará a tela inicial e após alguns segundo será direcionado à tela de *login*. Caso ele tenha esquecido a senha poderá entrar na tela de recuperação de senha. O aplicativo não permite cadastro no sistema, pois por regra de negócio a primeira senha do usuário será gerada pela secretaria da escola, a partir do sistema de gestão. A Figura 13 apresenta as telas descritas.

Figura 13 – Telas iniciais do SEEduca

Fonte: Autor (2024).

4.6.1.2 Tela *Home*

Essa é a tela onde se tornam acessíveis todas as funcionalidades disponíveis ao usuário aluno como: acessar dados da escola, acessar dados pessoais, informações sobre notas, frequência, acessar cardápio da merenda, informações sobre o transporte escolar, etc... A Figura 14 exibe como ficou seu *design*.

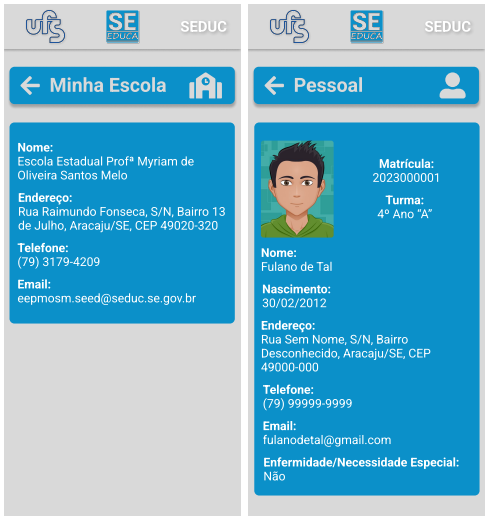
Figura 14 – Tela Home do SEEduca

Fonte: Autor (2024).

4.6.1.3 Informações da escola e do aluno

Nas duas telas ilustradas abaixo Figura 15 são exibidas as informações pertinentes à escola e os dados pessoais relevantes do aluno.

Figura 15 – Telas com as informações da Escola e do Aluno



Fonte: Autor (2024).

4.6.1.4 Telas da Agenda Escolar, Frequência e Notas do aluno

Nas telas seguinte o usuário consegue visualizar a agenda escolar diária, com as tarefas de sala de aula e da escola, a tela de frequência escolar e a tela com as notas, como podemos ver

na Figura 16.

Figura 16 – Telas da Agenda Escolar, Frequência e Notas do aluno

Fonte: Autor (2024),

4.6.1.5 Telas do Mural de Avisos, Informações de Transporte e Recados

Nas telas ilustradas na Figura 17 foram concebidos o Mural de Avisos onde a escola divulga de forma prática informações para toda a escola, a tela com as informações do transporte escolar do aluno e a tela de recados, onde a escola e os alunos/responáveis poderão mandar recados de forma eficiente.

Figura 17 – Telas do Mural de Avisos, Informações do Transporte e Recados

Fonte: Autor (2024).

4.6.1.6 Tela da Carteirinha Estudantil

E por fim temos a tela da emissão da carteira estudantil que, além das informações do aluno, apresenta um *QR Code* para validação das informações junto ao emissor. Podemos visualizar uma representação dessa tela na Figura 18.

Figura 18 – Tela da Carteirinha Estudantil

Fonte: Autor (2024).

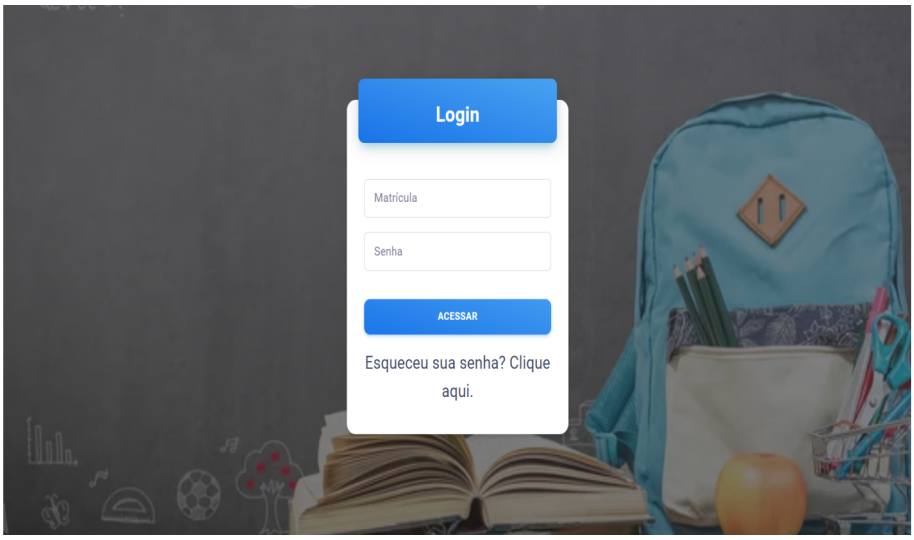
4.6.2 Protótipo do Sistema *WEB* SEEduca

O segundo protótipo é os Sistema *WEB* de gestão SEEduca, que é voltado para os integrantes da escola. Nesse sistema é onde as informações serão alimentadas pelos professores, coordenadores e diretores da escola. Devido a grande quantidade de telas apresentarei apenas algumas telas mais importantes.

4.6.2.1 Tela de *Login*

A primeira tela é a tela de Login, onde poderão acessar os Diretores, Coordenadores e Professores. Administradores, ou seja, superusuários também podem acessar a partir dessa tela, porém algumas funções, como visualizar perfil, não ficará visível. Alunos não conseguirão logar a partir dessa tela, apenas no App.

Figura 19 – Tela de Login

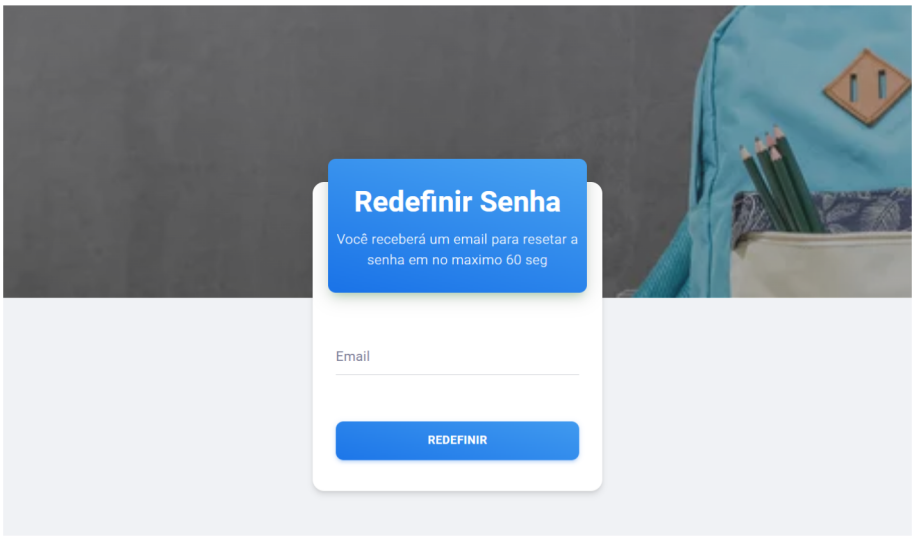


Fonte: Autor (2024).

4.6.2.2 Tela de Redefinição de Senha

A próxima tela é a tela de redefinição de senha, que pode ser acessada clicando no link "Esqueceu a senha? Clique aqui."da tela de Login. Para a recuperação basta inserir o email principal do usuário e clicar em recuperar, dessa forma um email com instruções para a recuperação será enviado.

Figura 20 – Tela de Redefinição de Senha



Fonte: Autor (2024).

4.6.2.3 *Dashboard*

Após efetuar o *login* o usuário será encaminhado para o Dashboard da aplicação, onde algumas informações e gráficos sobre o sistema serão exibidos na tela. A partir dessa tela também ficará acessivel o menu da aplicação, que será diferente dependendo do tipo de usuário. Diretores e Coordenadores terão o mesmo menu, no entanto os professores terão um menu mais simplificado, de acordo com suas permissões de acesso no sistema.

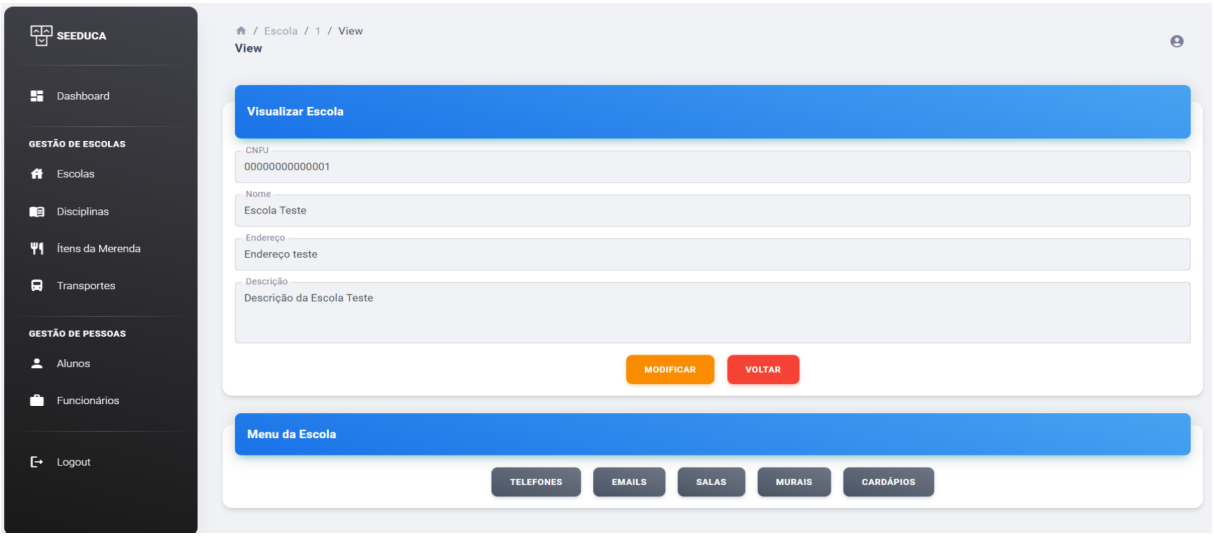
Figura 21 – Dashboard

Fonte: Autor (2024).

4.6.2.4 Informações da Escola

Nesta tela vemos as informações da escola selecionada, onde podemos abrir a tela de modificação dos dados, voltar ou acessar as opções da escola como, cadastro de emails, telefones, murais de avisos, cardápios da merenda e as salas da escola.

Figura 22 – Visualizar Dados da Escola

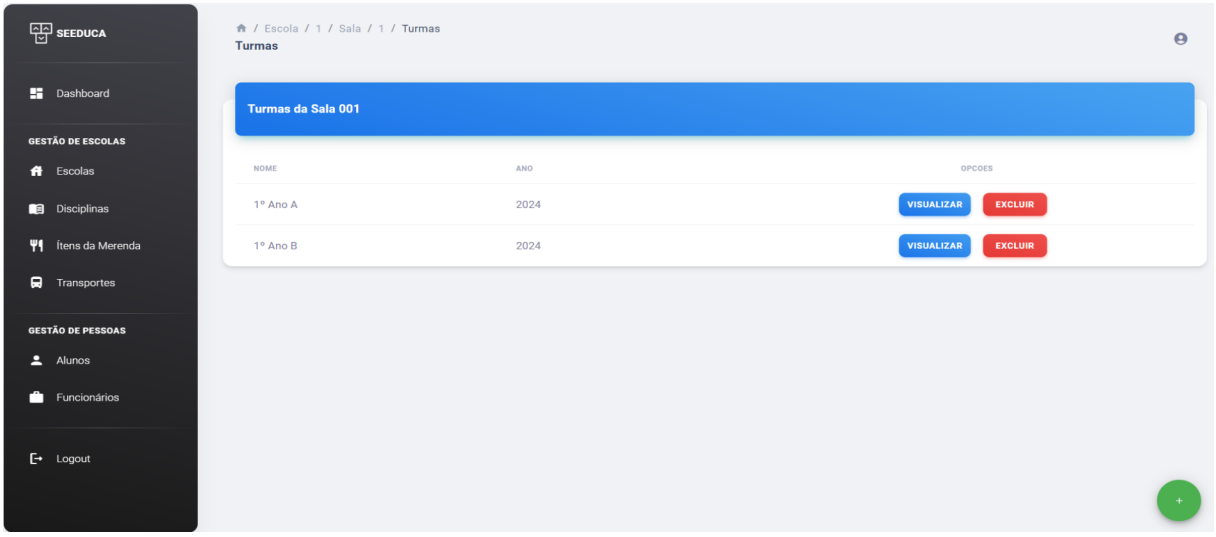


Fonte: Autor (2024).

4.6.2.5 Listar Turmas

Na tela exibida na Figura 23 temos a lista das turmas que foram ou estão sendo ministradas na Sala 001 da escola. Seguindo o padrão das outras telas vemos uma tabela com a lista de turmas, os respectivos botões de visualizar e excluir turma e um botão flutuante que redireciona para a tela de cadastro de novas turmas nessa sala.

Figura 23 – Lista de Turmas da Sala 001

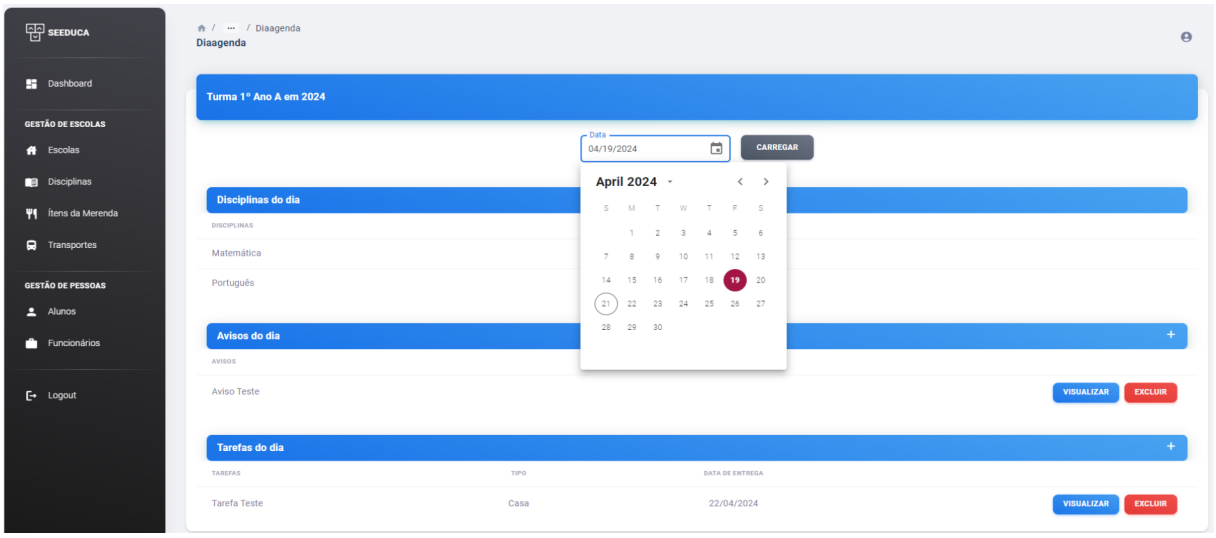


Fonte: Autor (2024).

4.6.2.6 Agenda da Turma

A imagem abaixo mostra a tela onde podemos acessar a Agenda da Turma. Nela temos um seletor de data e o botão para carregar as informações da agenda. Nela também podem ser cadastrados os avisos da turma e as tarefas, sejam elas de casa ou sala de aula.

Figura 24 – Agenda da Turma

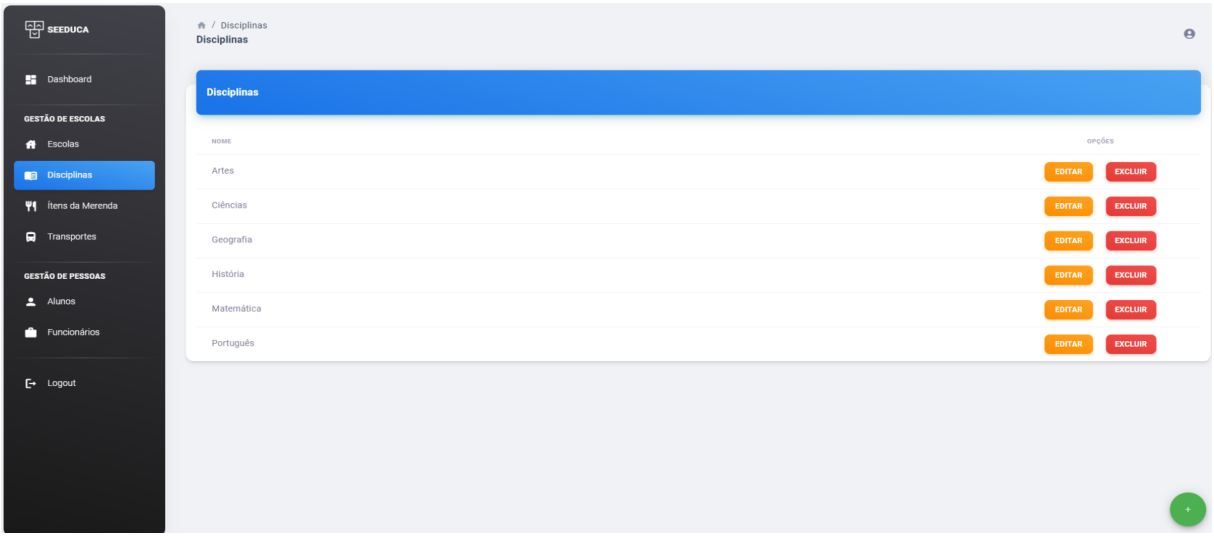


Fonte: Autor (2024).

4.6.2.7 Lista de Disciplinas

Na próxima tela vemos a lista de disciplinas cadastradas no sistema, assim como as opções de cadastro de novas disciplinas, editar e excluir disciplinas já cadastradas.

Figura 25 – Agenda da Turma

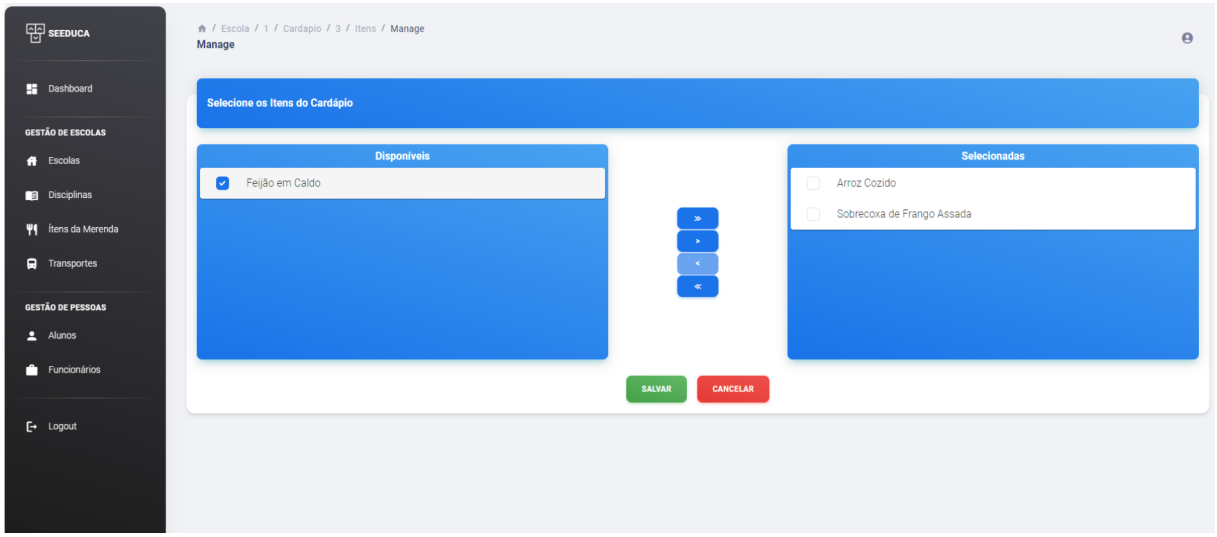


Fonte: Autor (2024).

4.6.2.8 Gerenciar Itens do Cardápio

Já na tela da Figura 26 vemos a opção de gerenciamento dos itens de um cardápio selecionado. Nela podemos ver duas listas: a da esquerda que ficam os itens disponíveis e a da direita com os itens selecionados para o cardápio do dia.

Figura 26 – Gerenciar Itens do Cardápio

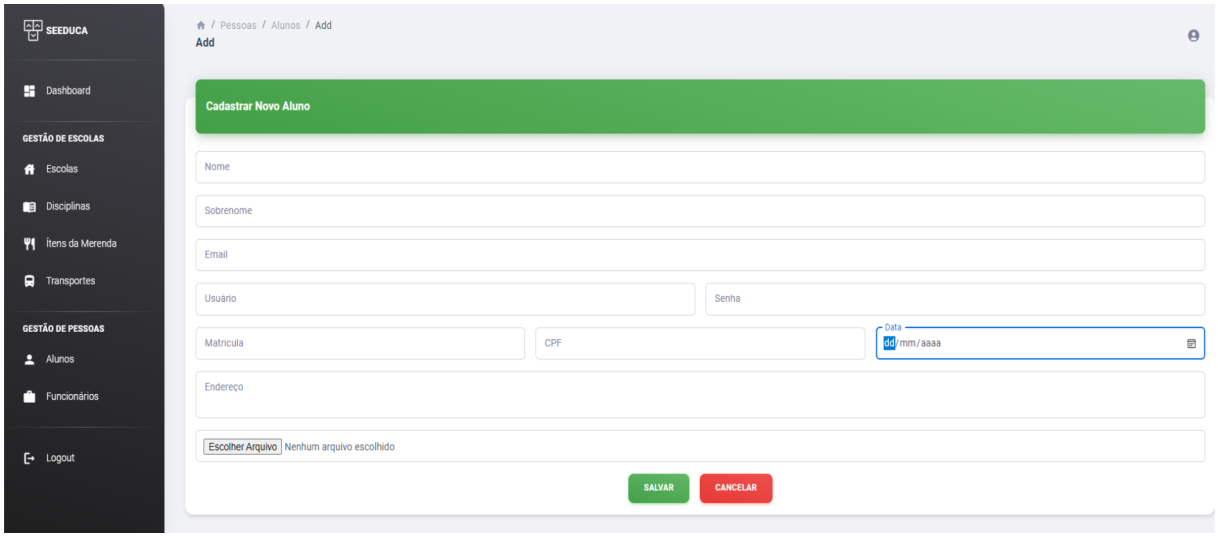


Fonte: Autor (2024).

4.6.2.9 Cadastrar Novos Alunos

Na imagem abaixo vemos a tela de cadastro de novos alunos onde inserimos todos os dados do aluno exceto os responsáveis, telefones e emails secundários, que foram alocados no menu de opções do aluno, disponíveis na tela de visualização dos dados do aluno.

Figura 27 – Cadastrar Novos Alunos

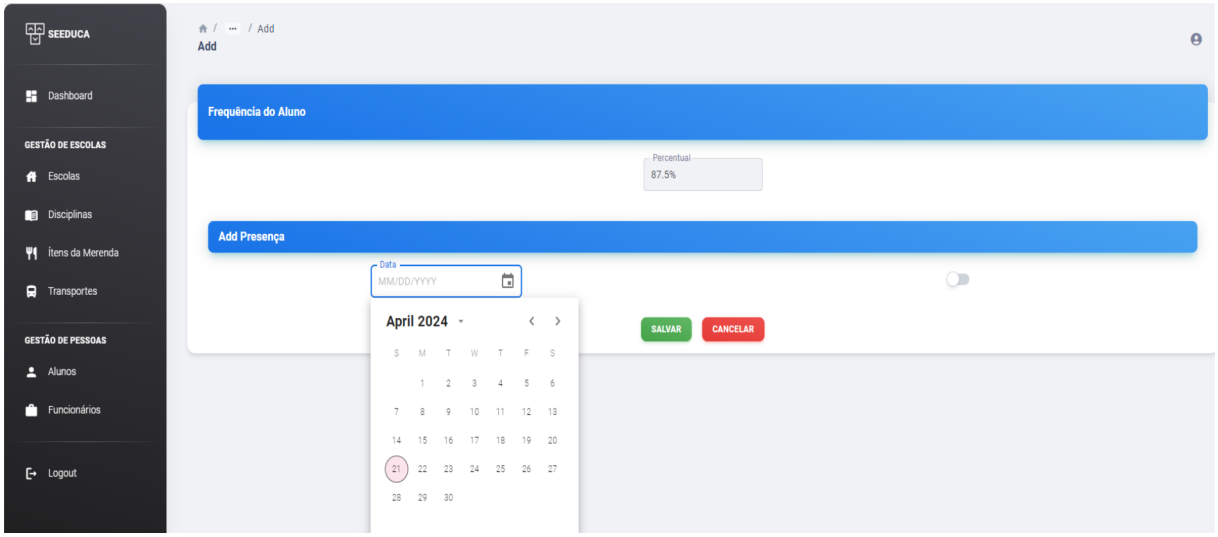


Fonte: Autor (2024).

4.6.2.10 Registrar Nova Presença

A próxima imagem veremos a tela de registro de uma nova presença do aluno na turma. Nesta tela selecionamos uma data e marcamos o switch de presença como verdadeiro ou falso. Após salvar o valor do percentual de frequência é atualizado automaticamente.

Figura 28 – Registro de Presença

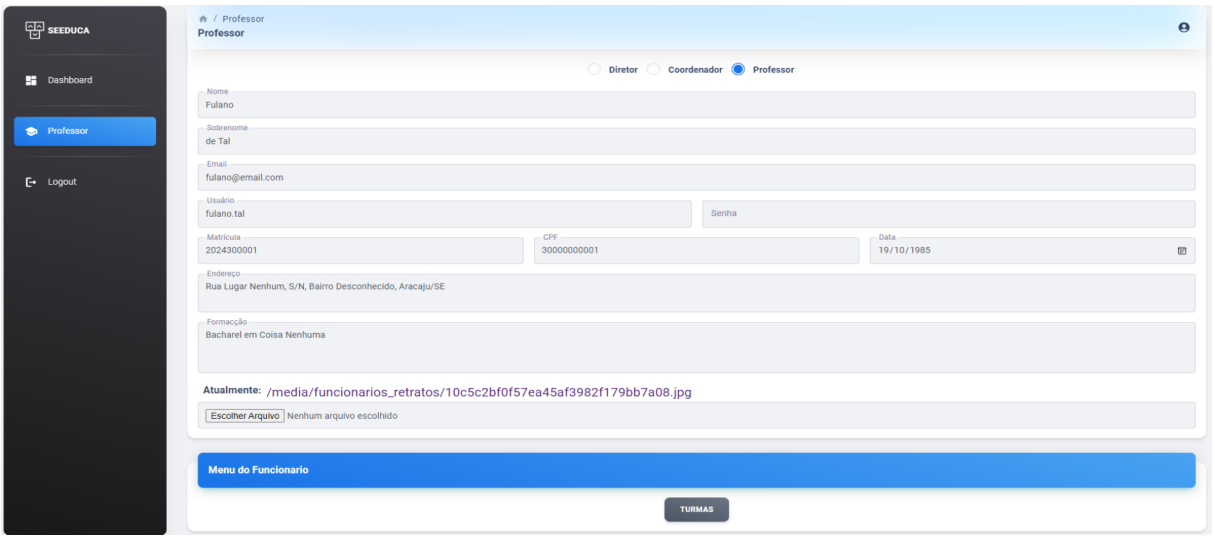


Fonte: Autor (2024).

4.6.2.11 Gerenciar Notas do Boletim

Logo abaixo veremos a tela onde é possível gerenciar as notas do aluno. Ela consiste em umma tabela com todas as unidades, inclusive, recuperações 1 e 2, além das médias semestrais e média geral. Abaixo das unidades vemos o botão que permite alterar os valores das notas.

Figura 29 – Gerenciar Notas do Boletim



Fonte: Autor (2024).

4.6.2.12 Agenda de Recados

Na próxima tela temos uma Agenda de Recados digital, onde responsáveis e a escola podem trocar mensagens relacionadas ao aluno. Ela é bem semelhante a um chat, porém as mensagens não serão necessariamente sincronas e trocadas em tempo real.

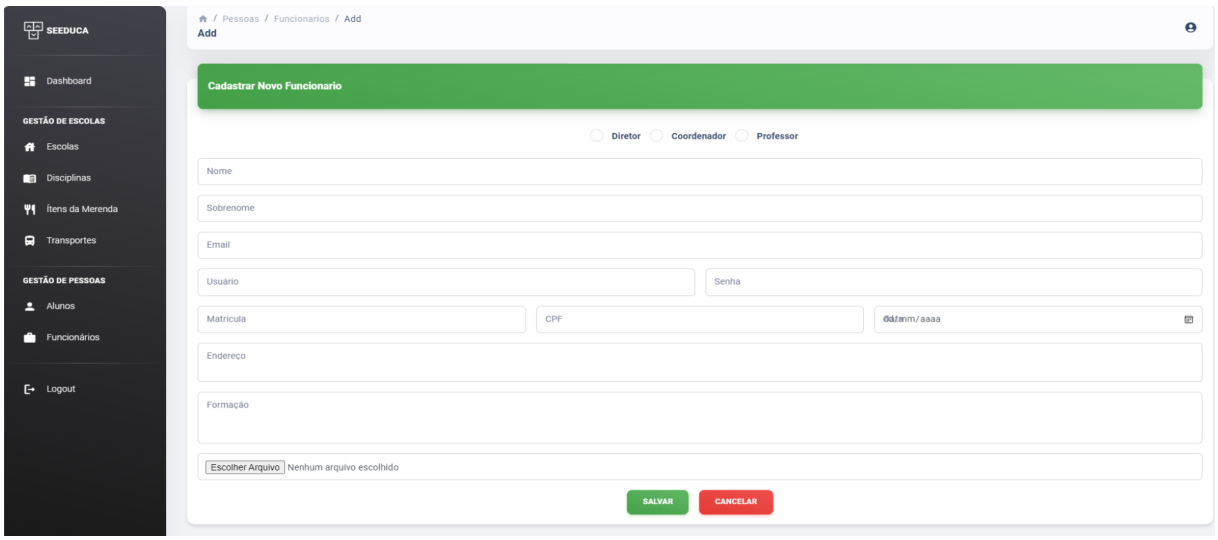
Figura 30 – Agenda de Recados

Fonte: Autor (2024).

4.6.2.13 Cadastrar Novo Funcionario

Em seguida vemos a tela de cadastro de funcionários da escola. Como é possível ver na tela temos quase todas as mesmas informações da tela de cadastro de alunos, porém nela temos a seleção do cargo do funcionário, que pode ser, Diretor, Coordenador ou Professor e também temos o campo pra inserir a informações de formação do Funcionário.

Figura 31 – Cadastrar Novo Funcionário

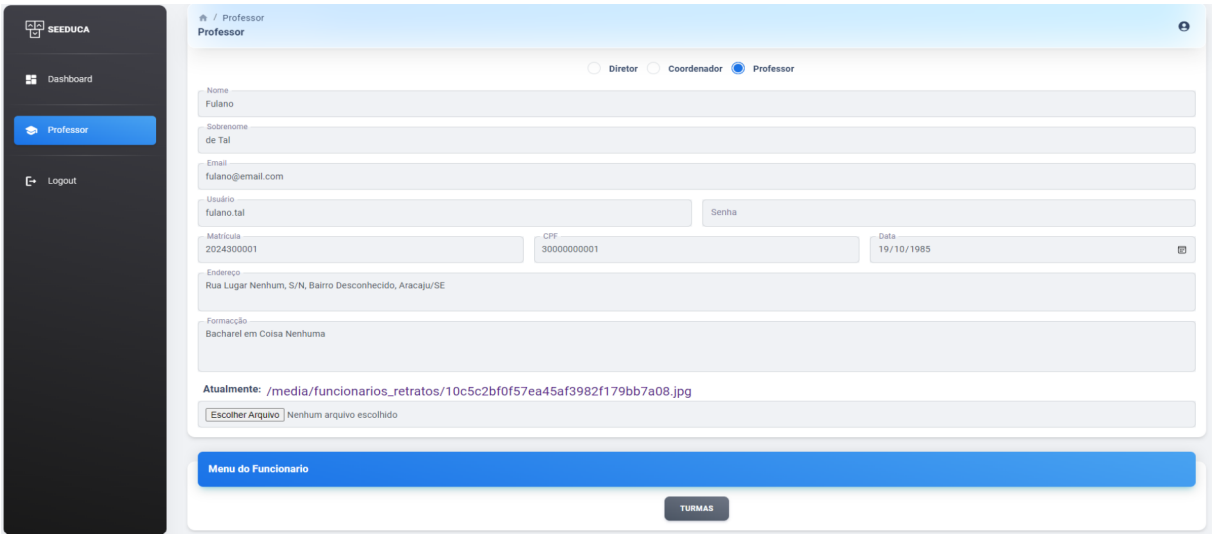


Fonte: Autor (2024).

4.6.2.14 Visualizar Dados do Professor

Por fim vemos a tela de visualização dos dados do Professor. Como é possível notar na imagem quando o usuário com a função de Professor logar no sistema, o menu de opções que aparecerá à esquerda terá bem menos opções. No menu de opções do professor apenas ficam disponíveis as turmas que este leciona.

Figura 32 – Visualizar Dados do Professor



Fonte: Autor (2024).

5

Implementação do Sistema de Gestão SEEDUCA

Este capítulo, fornece uma análise detalhada da implementação do sistema de gestão escolar SEEDUCA, explorando os aspectos técnicos e práticos das três partes principais: o *backend* desenvolvido com Django Rest *Framework*, uma poderosa estrutura para a construção de APIs em Python que oferece facilidade na criação de aplicativos *web* escaláveis e seguros; o *frontend web* construído com React, um *framework* JavaScript de código aberto mantido pelo *Facebook*, empregando o Material Dashboard 2 React para garantir uma interface elegante e responsiva; e o *frontend mobile* desenvolvido com React Native, uma estrutura que possibilita o desenvolvimento de aplicativos móveis multiplataforma utilizando JavaScript e React, permitindo a criação de uma experiência de usuário consistente tanto em dispositivos Android quanto iOS. Essas tecnologias combinadas oferecem uma solução abrangente e eficiente para a gestão escolar, proporcionando uma experiência integrada e de alto desempenho para administradores, professores, alunos e pais.

5.1 *Backend*

O sistema de gestão escolar foi meticulosamente planejado, com o backend concebido com base nos sólidos princípios da arquitetura RESTful. Isso significa que cada componente é estruturado de forma a facilitar a comunicação entre os diversos serviços que compõem o sistema. Para tornar isso uma realidade, optamos por utilizar o *framework* Django em conjunto com o Django Rest *Framework*, uma combinação poderosa que simplifica significativamente o desenvolvimento de APIs RESTful, garantindo ao mesmo tempo robustez e eficiência.

No coração do sistema, encontramos o banco de dados MariaDB, uma escolha deliberada devido à sua compatibilidade com o MySQL e sua comprovada confiabilidade em ambientes de produção. Essa escolha assegura que tenhamos uma base de dados sólida e escalável para suportar as necessidades em constante evolução do nosso sistema de gestão escolar.

Para garantir a segurança das interações do usuário com o sistema, implementamos um sistema de autenticação baseado em *tokens* JSON *Web Tokens* (JWT), utilizando o SimpleJWT. Essa abordagem moderna e robusta nos permite controlar com precisão o acesso aos recursos do sistema, mantendo a simplicidade de integração e a eficácia na autenticação dos usuários.

Código 1 – Trecho1 do arquivo settings.py

```
...

INSTALLED_APPS = [
    'corsheaders',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'rest_framework',
    'rest_framework_simplejwt',
    'escolas',
    'pessoas',
    'qrcode',
]

...

CORS_ALLOW_ALL_ORIGINS = True if
   os.environ.get('CORS_ALLOW_ALL_ORIGINS') == '1' else False

...

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'seeduca',
        'USER': os.environ.get('MARIADB_USER'),
        'PASSWORD': os.environ.get('MARIADB_PASSWORD'),
        'HOST': os.environ.get('MARIADB_HOST'),
        'PORT': '3306',
    }
}

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    )
}

...
```

Código 2 – Trecho2 do arquivo settings.py

```
...

SIMPLE_JWT = {
    "ACCESS_TOKEN_LIFETIME": timedelta(minutes=120),
    "REFRESH_TOKEN_LIFETIME": timedelta(days=1),
    "BLACKLIST_AFTER_ROTATION": False,
    "SIGNING_KEY": SECRET_KEY,
    "AUTH_HEADER_TYPES": ("Bearer",),
}

...
```

Nos trechos de código acima(**Códigos 1 e 2**), podemos observar as principais configurações do sistema no arquivo settings.py. Nele, estão definidos os aplicativos instalados, as configurações de CORS, bem como as configurações do banco de dados e dos tokens de autenticação utilizando SimpleJWT.

5.1.1 Principais Funcionalidades

O *backend* oferece uma ampla gama de funcionalidades para gerenciar escolas e pessoas associadas ao ambiente escolar.

5.1.1.1 Gestão de Escolas

- Cadastro e gerenciamento de escolas, salas de aula e turmas.
- Agenda escolar para programação de eventos e atividades.
- Mural de avisos para comunicação interna da escola.
- Cardápios de merenda para visualização dos alimentos servidos.
- Cadastro de disciplinas e organização do currículo escolar.
- Gestão do transporte escolar para controle de rotas e horários.

Abaixo no **Código 3**, apresentamos o modelo da escola juntamente com seus atributos. O modelo desempenha a função de estruturar a tabela correspondente à escola no banco de dados.

Código 3 – escola_model.py

```
from django.db import models


class Escola(models.Model):
    cnpj = models.CharField(
        max_length=14,
        unique=True,
    )
    nome = models.CharField(
        max_length=100,
    )
    endereco = models.CharField(
        max_length=255,
    )
    num_salas = models.IntegerField(
        default=0,
    )
    quantidade_alunos = models.IntegerField(
        default=0,
    )
    descricao = models.TextField(
        blank=True,
        null=True,
    )
    criado_em = models.DateTimeField(
        auto_now_add=True,
    )
    atualizado_em = models.DateTimeField(
        auto_now=True,
    )
    imagem = models.ImageField(
        upload_to='escolas_images/',
        blank=True,
        null=True,
        default='',
    )

    def __str__(self):
        return str(self.nome)
```

O **Código 4** abaixo implementa uma visualização (viewset) utilizando o Django REST *Framework* para manipular operações CRUD (criação, leitura, atualização e exclusão) em instâncias do modelo Escola. Ele utiliza o modelo Escola e seu respectivo serializador para definir as operações de serialização e desserialização dos dados. Além disso, define uma classe de permissão específica para diretores de escola, garantindo que apenas usuários com as devidas permissões possam acessar e manipular os dados das escolas.

Código 4 – escola_api_view.py

```
from rest_framework import viewsets

from escolas.models import Escola
from escolas.serializers import EscolaSerializer
from pessoas.permissions import DiretorEscolaPermission


class EscolaViewSet(viewsets.ModelViewSet):
    queryset = Escola.objects.all()
    serializer_class = EscolaSerializer
    permission_classes = [
        DiretorEscolaPermission,
    ]
```

Os **Códigos 5 6** é o serializador para o modelo Escola, utilizando o Django REST *Framework*. Ele especifica quais campos do modelo Escola devem ser incluídos na serialização e desserialização dos dados. Além disso, este serializador inclui campos relacionados à escola, como telefones, e-mails, salas, cardápios e murais de avisos, utilizando serializadores específicos para cada um desses relacionamentos. Isso permite que os dados relacionados sejam devidamente serializados e incluídos nas respostas da API de forma aninhada, facilitando a manipulação e a visualização dos dados relacionados. Os campos relacionados são marcados como somente leitura (*read_only=True*), indicando que eles não podem ser modificados por este serializador.

Código 5 – escola_serializer.py

```
from rest_framework import serializers

from escolas.models import Escola
from escolas.serializers.mural_avisos_serializer import
   MuralAvisosSerializer
from escolas.serializers.cardapio_serializer import
   CardapioMerendaSerializer
from escolas.serializers.email_escola_serializer import
   EmailEscolaSerializer
from escolas.serializers.sala_sem_objetos_serializer import
   SalaSemObjetosSerializer
from escolas.serializers.telefone_escola_serializer import
   TelefoneEscolaSerializer
```

Código 6 – escola_serializer.py

```
class EscolaSerializer(serializers.ModelSerializer):
    class Meta:
        model = Escola
        fields = [
            'id',
            'cnpj',
            'nome',
            'endereco',
            'num_salas',
            'quantidade_alunos',
            'descricao',
            'criado_em',
            'atualizado_em',
            'imagem',
            'objetos_telefones',
            'objetos_emails',
            'objetos_salas',
            'objetos_cardapios',
            'objetos_murais',
        ]
    objetos_telefones = TelefoneEscolaSerializer(
        many=True,
        source='escola_telefones',
        read_only=True,
    )
    objetos_emails = EmailEscolaSerializer(
        many=True,
        source='escola_emails',
        read_only=True,
    )
    objetos_salas = SalaSemObjetosSerializer(
        many=True,
        source='escola_salas',
        read_only=True,
    )
    objetos_cardapios = CardapioMerendaSerializer(
        many=True,
        source='escola_cardapios',
        read_only=True,
    )
    objetos_murais = MuralAvisosSerializer(
        many=True,
        source='escola_murais',
        read_only=True,
    )
```

Estes foram os códigos que configuram a API RESTful para o modelo de escola. Este é apenas um exemplo da API, todos os outros modelos no app escola seguem este mesmo padrão.

5.1.1.2 Gestão de Pessoas

- Cadastro e gerenciamento de funcionários, incluindo diretores, coordenadores e professores.
- Registro de alunos, com acompanhamento do boletim escolar, frequência e agenda de recados.
- Comunicação entre escola e responsáveis por meio da agenda de recados.
- Avaliação acadêmica por meio de registros de notas e médias.

Como podemos ver no **Código 7**, o modelo *Aluno* herda as características do modelo *Pessoa*. O modelo *Aluno* tem três campos adicionais: *eh_pcd* que é um campo booleano indicando se o aluno é uma pessoa com deficiência (PCD) ou não, *descricao_pcd* que é um campo de texto opcional para descrever a deficiência caso *eh_pcd* seja verdadeiro, e *retrato* que é um campo de imagem opcional para armazenar o retrato do aluno. O campo *retrato* é salvo na pasta *alunos_retratos/* dentro do sistema de arquivos.

Código 7 – aluno_model.py

```
1  from django.db import models
2
3  from pessoas.models.pessoa_model import Pessoa
4
5
6  class Aluno(Pessoa):
7      eh_pcd = models.BooleanField(
8          default=False,
9      )
10     descricao_pcd = models.TextField(
11         verbose_name='descricao da pcd',
12         blank=True,
13         null=True,
14     )
15     retrato = models.ImageField(
16         upload_to='alunos_retratos/',
17         blank=True,
18         null=True,
19         default='',
20     )
21
22     def __str__(self):
23         return str(self.usuario)
```

O **Código 8** abaixo, a exemplo do **Código 4**, implementa uma visualização (viewset) para manipular operações CRUD (criação, leitura, atualização e exclusão) em instâncias do modelo Aluno. Ele utiliza o modelo Aluno e seu respectivo serializador. Assim como o viewset

da escola, o AlunoViewSet define uma classe de permissão específica para garantir que apenas usuários com as devidas permissões possam acessar e manipular os dados dos Alunos, a classe AlunoPermission.

Código 8 – aluno_api_view.py

```
from rest_framework import viewsets

from pessoas.models import Aluno
from pessoas.permissions import AlunoPermission
from pessoas.serializers import AlunoSerializer


class AlunoViewSet(viewsets.ModelViewSet):
    queryset = Aluno.objects.all()
    serializer_class = AlunoSerializer
    permission_classes = [
        AlunoPermission,
    ]
```

O **Código 9** define a classe *AlunoPermission*, uma permissão personalizada no Django REST *Framework*. Ela controla o acesso às operações de uma visualização relacionada aos alunos com base em várias condições. Superusuários têm acesso irrestrito. Alunos (usuários não pertencentes a nenhum grupo) só podem visualizar detalhes de alunos se estiverem acessando seus próprios dados. Professores têm acesso de leitura. Coordenação e direção têm acesso completo. Essas regras garantem que diferentes tipos de usuários tenham permissões específicas ao interagir com os dados dos alunos.

Código 9 – aluno_permission.py

```
...

class AlunoPermission(permissions.BasePermission):
    def has_permission(self, request, view):
        if request.user.is_superuser:
            return True
        elif request.method == 'GET' and not
           request.user.groups.exists():
            aluno_id = view.kwargs.get('pk')
            return str(request.user.usuario_pessoa.id) ==
               str(aluno_id)
        elif request.method == 'GET' and
           request.user.groups.filter(name='Professor').exists():
            return True
        elif request.user.groups.filter(name__in=['Coordenador',
           'Diretor']).exists():
            return True
```

Assim como nas outras APIs RESTful foi desenvolvido um serializador presente no arquivo *aluno_serializer.py* responsável pera serialização e desserialização dos dados do Aluno. Da mesma forma foram criados Classes para todos os modelos presentes no app pessoas.

5.1.2 Modelos de Dados

Os modelos de dados foram organizados em dois aplicativos principais: escolas e pessoas. Cada aplicativo contém modelos específicos para suas respectivas funcionalidades.

5.1.2.1 App *escolas*

- **AgendaEscolar, AvisoEscola, AvisoAgenda, Cardapio, DiaAgenda, Disciplina, EmailEscola, Escola, ItemCardapio, MuralAvisos, Sala, TarefaAgenda, TelefoneEscola, Turma.**

Os **Códigos 10 e 11** é outro exemplo de um modelo do app escolas, o modelo Turma.

Código 10 – turma_model.py

```
1  from django.db import models
2
3  from escolas.models.base_model import YearField
4  from escolas.models.disciplina_model import Disciplina
5  from escolas.models.sala_model import Sala
6
7
8  class Turma(models.Model):
9      TURNO_CHOICES = [
10         ('M', 'Manh '),
11         ('T', 'Tarde'),
12         ('N', 'Noite'),
13     ]
14     nome = models.CharField(
15         max_length=100,
16     )
17     ano = YearField()
18     turno = models.CharField(
19         max_length=10,
20         choices=TURNO_CHOICES,
21     )
```

Código 11 – turma_model.py

```
criada_em = models.DateTimeField(
    auto_now_add=True,
)
atualizada_em = models.DateTimeField(
    auto_now=True,
)
sala = models.ForeignKey(
    Sala,
    on_delete=models.PROTECT,
    related_name='sala_turmas',
)
disciplinas = models.ManyToManyField(
    Disciplina,
    blank=True,
    related_name='disciplinas_turmas',
)

def __str__(self):
    return f'{str(self.nome)} em {self.ano}'

class Meta:
    unique_together = ['nome', 'ano', 'turno', 'sala', ]
```

5.1.2.2 App *pessoas*

- **AgendaRecados, Aluno, Avaliacao, Boletim, DiaLetivo, EmailPessoa, Frequencia, Funcionario, Media, Pessoa, Recado, ResponsavelAluno, Situacao, TelefonePessoa, TelefoneTransporte, Transporte.**

Já os **Códigos 12, 13, 14 e 15** é outro exemplo de um modelo do app pessoas, o Boletim.

Código 12 – boletim_model.py

```
from django.db import models
from django.core.exceptions import ValidationError

from escolas.models import Disciplina
from pessoas.models.aluno_model import Aluno
from pessoas.models.boletim_model import Boletim
```

Código 13 – boletim_model.py

```
class Avaliacao(models.Model):
    TIPO_AVALIACAO_CHOICES = [
        ('A1', '1   Avalia  o'),
        ('A2', '2   Avalia  o'),
        ('R1', '1   Recupera  o'),
        ('A3', '3   Avalia  o'),
        ('A4', '4   Avalia  o'),
        ('R2', '2   Recupera  o'),
    ]
    nome = models.CharField(
        max_length=100,
        choices=TIPO_AVALIACAO_CHOICES,
    )
    nota = models.FloatField(
        default=0,
    )
    confirmar = models.BooleanField(
        default=False,
    )
    criada_em = models.DateTimeField(
        auto_now_add=True,
    )
    atualizada_em = models.DateTimeField(
        auto_now=True,
    )
    aluno = models.ForeignKey(
        Aluno,
        on_delete=models.CASCADE,
        related_name='aluno_avaliacoes',
    )
    disciplina = models.ForeignKey(
        Disciplina,
        on_delete=models.CASCADE,
        related_name='disciplina_avaliacoes',
    )
    boletim = models.ForeignKey(
        Boletim,
        on_delete=models.CASCADE,
        related_name='boletim_avaliacoes',
    )

    def __str__(self):
        return f'{self.get_nome_display()} - {self.disciplina.nome}'

    class Meta:
        verbose_name = 'avalia  o'
        verbose_name_plural = 'avalia  es'
```

Código 14 – boletim_model.py

```
def save(self, *args, **kwargs):
    super_save = super().save(*args, **kwargs)
    if self.pk:
        if not self.boletim.encerrar:
            situacao = self.boletim.boletim_situacoes.filter(
                disciplina=self.disciplina).first()
            if situacao.situacao is None:
                avaliacoes = self.boletim.boletim_avaliacoes.filter(
                    disciplina=self.disciplina)
                a1 = avaliacoes.filter(nome='A1').first()
                a2 = avaliacoes.filter(nome='A2').first()
                r1 = avaliacoes.filter(nome='R1').first()

                if a1 and a2 and r1:  # Verifica se a1, a2 e r1 n o
                    s o None
                    media =
                        self.boletim.boletim_medias.filter(tipo='M1',
                        disciplina=self.disciplina.id).first()
                    if not a1.confirmar and not a2.confirmar and not
                        r1.confirmar:
                        media.valor = 0
                    elif a1.confirmar and not a2.confirmar and not
                        r1.confirmar:
                        media.valor = a1.nota
                    elif a1.confirmar and a2.confirmar and not
                        r1.confirmar:
                        media.valor = (a1.nota + a2.nota) / 2
                    elif a1.confirmar and a2.confirmar and
                        r1.confirmar:
                        if r1.nota > (a1.nota + a2.nota) / 2:
                            media.valor = ((a1.nota + a2.nota) + 2 *
                                r1.nota) / 4
                        else:
                            media.valor = (a1.nota + a2.nota) / 2
                    else:
                        raise ValidationError(
                            'Existem avalia  es dessa mat ria
                                pendentes de confirma  o no 1
                                semestre')

                    media.save()

                a3 = avaliacoes.filter(nome='A3').first()
                a4 = avaliacoes.filter(nome='A4').first()
                r2 = avaliacoes.filter(nome='R2').first()
```

Código 15 – boletim_model.py

```
if a3 and a4 and r2:  # Verifica se a3, a4 e r2 n o s o None
    media = self.boletim.boletim_medias.filter(tipo='M2',
       disciplina=self.disciplina.id).first()
    if not a3.confirmar and not a4.confirmar and not r2.confirmar:
        media.valor = 0
    elif a3.confirmar and not a4.confirmar and not r2.confirmar:
        media.valor = a3.nota
    elif a3.confirmar and a4.confirmar and not r2.confirmar:
        media.valor = (a3.nota + a4.nota) / 2
    elif a3.confirmar and a4.confirmar and r2.confirmar:
        if r2.nota > (a3.nota + a4.nota) / 2:
            media.valor = ((a3.nota + a4.nota) + 2 * r2.nota) / 4
        else:
            media.valor = (a3.nota + a4.nota) / 2
    else:
        raise ValidationError(
            'Existem avalia  es dessa mat ria pendentes de
               confirma  o no 2   semestre')

    media.save()

:
raise ValidationError('As notas de uma mat ria finalizada n o
   podem mais ser editadas!')

e ValidationError('Os boletins encerrados n o podem mais ser
   editados!')

_save
```

Os modelos foram estruturados de forma a refletir as relações e funcionalidades necessárias para o sistema de gestão escolar.

5.1.3 APIs RESTful

Para garantir uma implementação eficiente e padronizada das operações CRUD (criação, leitura, atualização e exclusão) para cada modelo de dados, optamos por utilizar os ViewSets fornecidos pelo Django Rest *Framework*. Esses ViewSets oferecem uma abordagem simplificada e poderosa para definir a lógica de manipulação de dados, permitindo que as APIs sejam desenvolvidas de forma rápida e consistente.

Outro exmplo de viewset pode ser visto no **Código 16**.

Código 16 – agenda_escolar_api_view.py

```
from rest_framework import viewsets

from escolas.models import AgendaEscolar
from escolas.serializers import AgendaEscolarSerializer
from pessoas.permissions import CoordenadorFullPermission


class AgendaEscolarViewSet(viewsets.ModelViewSet):
    queryset = AgendaEscolar.objects.all()
    serializer_class = AgendaEscolarSerializer
    permission_classes = [
        CoordenadorFullPermission,
    ]
```

Além disso, para organizar as rotas da API, empregamos o SimpleRouter em cada app. Essa abordagem não apenas facilita a configuração das URLs, mas também promove uma estrutura clara e coesa, facilitando a compreensão e manutenção do código no futuro. Ao combinar o poder dos ViewSets com a organização cuidadosa das rotas, conseguimos construir um sistema robusto e escalável para manipulação de dados com o uso de APIs RESTful.

No **Código 18** abaixo vemos trechos do código urls.py do app escolas, onde são declaradas as urls da viewset *AgendaEscolaViewSet*. A partir dessa declaração o SimpleRouter declara as rotas HTTP dos métodos GET, POST, PATCH e DELETE.

Código 17 – escolas - urls.py

```
from django.urls import path, include
from rest_framework.routers import SimpleRouter
from escolas import views

app_name = 'escolas'

escola_api_v1_router = SimpleRouter()
...

escola_api_v1_router.register(
    'sala/turma/agenda/api/v1',
    views.AgendaEscolarViewSet,
    basename='escola-sala-turma-agenda-api',
)
...

urlpatterns = [
    path('', include(escola_api_v1_router.urls))
]
```

5.1.4 Integração com o *Frontend*

A integração com os *frontends web* e *mobile* é uma parte crucial do sistema, e é alcançada utilizando um processo de comunicação assíncrona que utiliza o formato JSON para a troca de dados. Este método permite uma comunicação eficiente e ágil entre o *backend* e os *frontends*, facilitando a transferência de informações em tempo real. Para lidar com as requisições HTTP, os *frontends* contam com o suporte de bibliotecas bem estabelecidas, como o Axios ou o Fetch, que oferecem uma interface simplificada para lidar com solicitações e respostas da API. Em relação à segurança, a autenticação é um aspecto crucial. Nesse sentido, o uso do SimpleJWT é fundamental para garantir que apenas usuários autorizados possam acessar os recursos do sistema. Essa abordagem não só protege os dados sensíveis, mas também assegura a integridade das comunicações entre o *backend* e os *frontends*, proporcionando uma experiência segura e confiável para os usuários.

Abaixo vemos o urls.py principal onde foram configuradas as rotas SimpleJWT para autenticação.

Código 18 – urls.py

```
1  from django.conf.urls.static import static
2  from django.contrib import admin
3  from django.urls import path, include
4  from rest_framework_simplejwt.views import TokenObtainPairView,
       TokenRefreshView, TokenVerifyView
5
6  from appseeduca import settings
7
8  urlpatterns = [
9      path('admin/', admin.site.urls),
10     path('escolas/', include('escolas.urls')),
11     path('pessoas/', include('pessoas.urls')),
12     path('api/token/', TokenObtainPairView.as_view(),
           name='token_obtain_pair'),
13     path('api/token/refresh/', TokenRefreshView.as_view(),
           name='token_refresh'),
14     path('api/token/verify/', TokenVerifyView.as_view(),
           name='token_verify'),
15 ]
16
17 urlpatterns += static(settings.MEDIA_URL,
       document_root=settings.MEDIA_ROOT)
18 urlpatterns += static(settings.STATIC_URL,
       document_root=settings.STATIC_ROOT)
```

Para ter acesso ao código do *backend* completo basta acessar o repositório do github: https://github.com/matheusteixeiradcompufs/seeduca

5.2 *Frontend WEB*

O *frontend* do SEEduca foi desenvolvido utilizando o *framework* React, com a utilização do template *Material Dashboard* 2 React para fornecer uma interface de usuário moderna e responsiva. A estrutura do projeto é organizada de forma a facilitar a navegação e manutenção do código, com todas as telas dispostas na pasta src/layouts.

5.2.1 Organização das Telas

Cada tela é estruturada seguindo o padrão CRUD (Create, Read, Update, Delete). A pasta raiz de cada recurso (por exemplo, escolas, disciplinas, pessoas) contém um arquivo index.js responsável por renderizar a tela principal de listagem onde é possível visualizar os itens ou excluir. Para cada função adicional, como adicionar, editar ou visualizar detalhes de um item, são criadas pastas específicas dentro da pasta do recurso.

Código 19 – escolas/index.js

```
...

function Escolas() {

  ...

  useEffect(() => {
    const fetchEscolas = async () => {
      try {
        const response = await api.get("/escolas/api/v1/");
        setEscolas(response.data);
        setLoading(false);
      } catch (error) {
        if (error.response.status === 401) {
          await refreshToken();
          await fetchEscolas();
        } else {
          toast.error("Erro ao carregar escolas");
          console.error("Erro ao carregar escolas:", error);
        }
        setLoading(false);
      }
    };
    fetchEscolas();
  }, []);

  const handleView = (escolaid) => {
    setLoading(true);
    navigate(`/escola/${escolaid}/view`);
  };
```

Código 20 – escolas/index.js

```
const handleExcluir = async (escolaid) => {
  setLoading(true);
  try {
    await api.delete(`/escolas/api/v1/${escolaid}/`);
    const response = await api.get("/escolas/api/v1/");
    setEscolas(response.data);
    setLoading(false);
  } catch (error) {
    if (error.response.status === 401) {
      await refreshToken();
      await handleExcluir(escolaid);
    } else {
      toast.error("Erro ao excluir escola");
      console.log("Erro ao excluir escola", error);
    }
    setLoading(false);
  }
};

...

return (
  <DashboardLayout>
    ...
            <MDBox pt={3} px={2}>
              <DataTable
                table={{
                  columns: [
                    { Header: "cnpj", accessor: "cnpj", align:
                       "left" },
                    { Header: "nome", accessor: "nome", align:
                       "left" },
                    { Header: "num_salas", accessor: "num_salas",
                       align: "center" },
                    {
                      Header: "quantidade_alunos",
                      accessor: "quantidade_alunos",
                      align: "center",
                    },
                    { Header: "opcoes", accessor: "opcoes", align:
                       "center" },
                  ],
      ...
```

Código 21 – escolas/index.js

```
...
                rows: escolas.map((escola) => ({
                  cnpj: escola.cnpj,
                  nome: escola.nome,
                  num_salas: escola.num_salas,
                  quantidade_alunos: escola.quantidade_alunos,
                  opcoes: (
                    <Grid
                      container
                      spacing={2}
                      alignItems="center"
                      justifyContent="space-between"
                    >
                      <Grid item xs={12} sm={6} container>
                        <MDButton
                          variant="gradient"
                          color="info"
                          size="small"
                          onClick={() => handleView(escola.id)}
                        >
                          Visualizar
                        </MDButton>
                      </Grid>
                      <Grid item xs={12} sm={6} container>
                        <MDButton
                          variant="gradient"
                          color="error"
                          size="small"
                          onClick={() => handleExcluir(escola.id)}
                          disabled={!user.is_superuser}
                        >
                          Excluir
                        </MDButton>
                      </Grid>
                    </Grid>
                  ),
                })),
              }}
              isSorted={false}
              entriesPerPage={false}
              showTotalEntries={false}
              noEndBorder
            />
          </MDBox>
        </Card>
      </Grid>
      ...
    </DashboardLayout>
  );
}
export default Escolas;
```

Nos **Códigos 19, 20 e 21** acima exibe trechos do código index.js da tela escolas. **Escolas** é definida como um componente funcional que utiliza o *hook* **useState** para gerenciar o estado das escolas e o estado de carregamento. O hook **useEffect** é utilizado para buscar as escolas da API assim que o componente é montado. Se houver um erro de autorização (*status* 401), ele tenta renovar o token de autenticação e refazer a requisição. As funções **handleView** e **handleExcluir** são responsáveis por lidar com a visualização e exclusão de escolas, respectivamente. Se o estadode carregamento estiver ativo, é exibido um indicador visual de carregamento. Caso contrário, é renderizado o layout da lista de escolas, incluindo a tabela de dados, botões para visualização e exclusão de escolas, e um botão para adicionar novas escolas, este último disponível apenas para usuários com permissões de superusuário.

Por exemplo, para o recurso escolas, a estrutura de pastas fica da seguinte forma:

- **escolas**
  - **add**: Contém o formulário e funcionalidades para adicionar uma nova escola.
  - **editar**: Responsável pela edição de uma escola existente.
  - **view**: Mostra os detalhes de uma escola específica, incluindo suas salas, cardápios, murais, etc.
  - **salas**: Gerencia as salas de aula relacionadas a uma escola.
    - **add**: Permite adicionar uma nova sala.
    - **editar**: Permite editar os detalhes de uma sala existente.
    - **view**: Gerencia as turmas associadas a uma sala específica.
    - **turmas**: Gerencia as turmas associadas a uma sala específica.
      - **add**: Adiciona uma nova turma.
      - ...
  - **murais**
    - **add**: Permite adicionar um novo mural de avisos.
    - ...
  - ...

Os **Códigos 22 e 23**, logo abaixo, apresentam a tela AddEscolas, projetado para adicionar uma nova escola ao sistema. As funções **handleChange** atualizam os estados dos campos do formulário conforme o usuário interage. Quando o usuário solicita adicionar uma nova escola, a função **handleAdd** faz uma solicitação à API com os dados do formulário. Se ocorrer um erro de autorização, ela tenta renovar o token de autenticação antes de reenviar a solicitação. A função **handleCancelar** permite ao usuário cancelar a operação e retornar à lista de escolas. O formulário de adição de escola é renderizado com campos para CNPJ, nome, endereço, descrição e botões para salvar e cancelar.

Código 22 – escolas/add/index.js

```
...

function AddEscolas() {
  ...

  const handleAdd = async () => {
    setLoading(true);
    try {
      await api.post("/escolas/api/v1/", {
        cnpj: cnpj,
        nome: nome,
        endereco: endereco,
        descricao: descricao,
      });
      navigate("/escolas");
    } catch (error) {
      if (error.response.status === 401) {
        await refreshToken();
        await handleAdd();
      } else {
        toast.error("Erro ao cadastrar escola");
        console.log("Erro ao cadastrar escola", error);
      }
      setLoading(false);
    }
  };

  const handleCancelar = () => {
    setLoading(true);
    navigate("/escolas");
  };
...

  return (
    <DashboardLayout>
      <ToastContainer />
      <DashboardNavbar />
      <MDBox pt={6} mb={3}>
        <Grid container spacing={3}>
          <Grid item xs={12}>
            <Card>
              <MDBox
                mx={2}
                mt={-3}
                py={3}
                px={2}
                variant="gradient"
                bgColor="success"
                borderRadius="lg"
                coloredShadow="success"
              >
...
```

Código 23 – escolas/add/index.js

```
...
                <MDTypography variant="h6" color="white">
                  Cadastrar Nova Escola
                </MDTypography>
              </MDBox>
              <Grid container spacing={3} mb={2}>
                <Grid item xs={12} sm={12}>
                  <MDBox display="flex" justifyContent="center"
                     pt={2} px={2}>
                    <MDInput
                      type="number"
                      variant="outlined"
                      label="CNPJ"
                      value={cnpj}
                      onChange={handleChangeCnpj}
                      fullWidth
                    />
                  </MDBox>

...

                  <MDBox display="flex" justifyContent="center"
                     pt={2} px={2}>
                    <MDBox mr={1}>
                      <MDButton variant="contained" color="success"
                         onClick={handleAdd}>
                        Salvar
                      </MDButton>
                    </MDBox>
                    <MDBox ml={1}>
                      <MDButton variant="contained" color="error"
                         onClick={handleCancelar}>
                        Cancelar
                      </MDButton>
                    </MDBox>
                  </MDBox>
                </Grid>
              </Grid>
            </Card>
          </Grid>
        </Grid>
      </MDBox>
    </DashboardLayout>
  );
}
export default AddEscolas;
```

5.2.2 Componentes Reutilizáveis

Além das telas principais e suas funcionalidades específicas, o projeto também inclui uma pasta *components* em cada recurso, onde são armazenados componentes reutilizáveis. Isso promove a reutilização de código e a consistência visual em todo o aplicativo.

Por exemplo, a pasta *components* dentro de escolas pode conter componentes como Menu, que é compartilhado entre diferentes telas dentro do recurso de escolas.

Código 24 – escolas/components/Menu/index.js

```
...

function Menu({ escolaid }) {
  return (
    <Card>
    ...

        <MDTypography variant="h6" color="white">
          Menu da Escola
        </MDTypography>
      </MDBox>
      <Grid container spacing={3} mb={2}>
        <Grid item xs={12} sm={12}>
          <MDBox display="flex" justifyContent="center" pt={2}
             px={2}>
            ...
            <MDBox mr={1} ml={1}>
              <Link to={`/escola/${escolaid}/salas`}>
                <MDButton variant="gradient" color="secondary">
                  Salas
                </MDButton>
              </Link>
            </MDBox>
            <MDBox mr={1} ml={1}>
              <Link to={`/escola/${escolaid}/murais`}>
                <MDButton variant="gradient" color="secondary">
                  Murais
                </MDButton>
              </Link>
            </MDBox>
            ...
          </MDBox>
        </Grid>
      </Grid>
    </Card>
  );
}
...
export default Menu;
```

Essa organização cuidadosa do projeto *frontend* contribui para uma experiência de desenvolvimento mais eficiente e uma interface de usuário consistente e intuitiva para os usuários do sistema de gestão escolar.

5.2.3 Componentes e Estilos

O *frontend* do SEEDuca apresenta uma variedade de componentes de interface de usuário que foram implementados para fornecer uma subexperiência amigável e intuitiva para os usuários. O *Material Dashboard* 2 React foi utilizado como base para o *design* e os componentes da interface, oferecendo uma aparência moderna e consistente em todo o sistema.

5.2.3.1 Principais Componentes Implementados

Os principais componentes da interface incluem:

1. **Tela de *Login* e Recuperação de Senha**: Responsável por permitir que os usuários façam *login* no sistema e solicitem a recuperação de senha, caso necessário.
2. **Gerenciamento de Escolas, Salas e Turmas**: Telas dedicadas ao cadastro, edição e visualização de informações relacionadas a escolas, salas de aula e turmas.
3. **Gerenciamento de Agendas Escolares**: Funcionalidade para criar e gerenciar eventos e atividades na agenda escolar..
4. **Gerenciamento de Pessoas**: Permite o cadastro e gerenciamento de alunos, diretores, coordenadores e professores, incluindo a visualização de informações pessoais e histórico acadêmico.
5. **Registro de Presenças e Avaliações**: Telas para registrar a presença dos alunos em sala de aula e inserir avaliações acadêmicas.
6. **Comunicação via Agenda de Recados**: Funcionalidade que permite a comunicação entre escola e responsáveis por meio de mensagens na agenda de recados.

5.2.3.2 Estilos e Componentes do *Material Dashboard* 2 React

O *Material Dashboard* 2 React é composto por uma variedade de componentes específicos do *Material Design*, que são utilizados para construir a interface do usuário de forma coesasub e estilizada. Alguns dos componentes principais incluem:

- **Menu de Navegação**: Gerencia as rotas e fornece navegação entre as diferentes telas do sistema. Esse componente é renderizado a partir das rotas do tipo **collapse**.

Código 25 – routes.js

```
import Dashboard from "layouts/dashboard";
...
const routes = [
  //DASHBOARD
  {
    type: "collapse",
    name: "Dashboard",
    key: "dashboard",
    icon: <Icon fontSize="small">dashboard</Icon>,
    route: "/",
    component: <Dashboard />,
  },
...
];
export default routes;
```

- **Cards**: Utilizados para exibir informações de forma concisa e organizada em cartões distintos.

Código 26 – card

```
<Card>
  <MDBox
    mx={2}
    mt={-3}
    py={3}
    px={2}
    variant="gradient"
    bgColor="info"
    borderRadius="lg"
    coloredShadow="info"
  >
    <MDTypography variant="h6" color="white">
      Visualizar Turma
    </MDTypography>
  </MDBox>
...
</Card>
```

- **DataTable**: Componente para listar dados em tabelas, oferecendo recursos de ordenação, filtragem e paginação.

Código 27 – datatable

```
<DataTable
  table={{ columns, rows }}
  isSorted={false}
  entriesPerPage={false}
  showTotalEntries={false}
  noEndBorder
/>
```

- **MDBox, MDTypography, MDButton, MDInput e MDAvatar**: Componentes específicos do *Material Dashboard* 2 React, que fornecem estilos consistentes e elementos visuais alinhados com as diretrizes de design do *Material Design*.

Código 28 – MDBox

```
import MDBox from "components/MDBox";

<MDBox
  color="white"
  bgColor="info"
  variant="gradient"
  borderRadius="lg"
  shadow="lg"
  opacity={1}
  p={2}
>
  Box
</MDBox>
```

Para visualizar a documentação completa acesse esse endereço: https://www.creative-tim.com/learning-lab/react/overview/material-dashboard/. Esses componentes são combinados de forma a criar uma interface de usuário intuitiva, responsiva e esteticamente agradável para os usuários do sistema de gestão escolar.

5.2.4 Integração com o *Backend*

O *frontend* do SEEduca se integra ao *backend* por meio de requisições HTTP, permitindo a troca de dados entre o cliente e o servidor. Essa integração é realizada utilizando duas bibliotecas principais: Axios e Fetch, dependendo dos requisitos de autenticação.

5.2.4.1 Requisições com Axios

Para as operações que exigem autenticação, como todas as operações de CRUD e autenticação de usuários, é utilizada a biblioteca Axios. Esta biblioteca oferece suporte a

interceptadores de solicitação, permitindo a inclusão de tokens JWT de autenticação nos cabeçalhos das solicitações. As requisições são realizadas da seguinte forma:

Código 29 – Biblioteca Axios

```
axios.get('/api/escolas', {
  headers: {
    Authorization: `Bearer ${token}`
  }
})
  .then(response => {
    // Manipular os dados da resposta
  })
  .catch(error => {
    // Tratar erros
  });
```

5.2.4.2 Requisições com Fetch

Para operações que não requerem autenticação, como solicitar a redefinição de senha, confirmar o token para redefinição de senha e verificar o token lido pelo leitor de *QR code*, é utilizada a função Fetch do navegador. Essa função é nativa do JavaScript e é usada para fazer requisições HTTP assíncronas. Um exemplo de uso do Fetch é o seguinte:

Código 30 – Biblioteca Axios

```
fetch('/api/reset-password-confirm', {
  method: 'GET',
  headers: {
    'Content-Type': 'application/json',
  }
})
  .then(response => response.json())
  .then(data => {
    // Manipular os dados recebidos
  })
  .catch(error => {
    // Tratar erros
  });
```

5.2.4.3 Manipulação dos Dados Recebidos

Após receber uma resposta do servidor, os dados são recuperados da variável ***response*** (no caso do Axios, *response.data*, e no caso do Fetch, *response.json()*) e são atribuídos a estados (*states*) no *frontend*. Esses estados são então utilizados para renderizar a interface do usuário e manipular os dados conforme necessário.

Código 31 – Response.data

```
...

function EscolaSalaTurmas() {
  ...
  const [sala, setSala] = useState(null);

  useEffect(() => {
    const fetchSala = async () => {
      try {
        const response = await
           api.get(‘/escolas/sala/api/v1/${salaid}/‘);
        setSala(response.data);
        setLoading(false);
      } catch (error) {
        ...
      }
    };
    fetchSala();
  }, []);

...
```

Essa integração robusta com o *backend* permite que o *frontend* do sistema de gestão escolar interaja de forma eficiente e segura com os recursos fornecidos pelo servidor.

5.2.5 Implementação da Autenticação e Controle de Acesso

O sistema de autenticação do *frontend* se integra ao *backend* por meio da biblioteca SimpleJWT, que implementa três rotas padrão para autenticação: **api/token/**, **api/token/refresh/** e **api/token/verify/**. Essas rotas são configuradas no arquivo urls.py principal do *backend*:

Código 32 – Rotas do SimpleJWT

```
from rest_framework_simplejwt.views import TokenObtainPairView,
    TokenRefreshView, TokenVerifyView

urlpatterns = [
    ...

    path('api/token/', TokenObtainPairView.as_view(),
        name='token_obtain_pair'),
    path('api/token/refresh/', TokenRefreshView.as_view(),
        name='token_refresh'),
    path('api/token/verify/', TokenVerifyView.as_view(),
        name='token_verify'),
]

...
```

5.2.5.1 Contexto de Autenticação no *Frontend*

No *frontend*, o controle de autenticação é gerenciado pelo contexto de autenticação (**AuthContext**). Esse contexto utiliza os cookies para armazenar e gerenciar os *tokens* de acesso e atualização. As funções **obterTokens**, **refreshToken**, ***login*** e ***logout*** são responsáveis por realizar operações de autenticação e atualização dos *tokens*, conforme necessário.

5.2.5.2 Controle de Acesso com React *Router* DOM

O controle de acesso às rotas no *frontend* é gerenciado pelo React *Router* DOM. São implementados três arquivos de rotas:

- **AuthRoutes**: Contém as rotas acessíveis para usuários não autenticados, como a tela de *login* e recuperação de senha.
- **Routes**: Define as rotas acessíveis para diretores e coordenadores após a autenticação.
- **ProfessorRoutes**: Contém as rotas acessíveis para os professores após a autenticação.

Além disso, há uma rota **/pessoas/me/** que retorna os dados do usuário autenticado, permitindo que o sistema verifique a qual dos grupos o usuário pertence e controle o acesso de acordo. Essa abordagem garante que apenas usuários autenticados e autorizados tenham acesso às funcionalidades específicas do sistema de gestão escolar, proporcionando uma experiência de usuário segura e personalizada.

Código 33 – Acessando a rota ME

```
async function login({ username, password }) {
    ...
    try {
      const response = await api.post("/pessoas/me/", {
        username,
      });
      ...
    } catch (err) {
     ...
  }
```

Para mais detalhes sobre o código do *frontend web* do SEEduca acesse o repositório: https://github.com/matheusteixeiradcompufs/webSeeduca

5.3 *Frontend Mobile*

Agora, abordaremos detalhadamente a implementação do *frontend mobile* do SEEduca, desenvolvido com React Native. Exploraremos as principais características, funcionalidades e a integração com o *backend*, destacando as práticas adotadas no desenvolvimento e as tecnologias utilizadas.

5.3.1 Organização das Telas

O appSEEduca organiza todas as suas telas na pasta **src/pages**. Conforme as configurações de rotas e autenticação, apenas três telas estão disponíveis sem a necessidade de autenticação: *Splash, Login e Recover*. As demais podem ser acessadas após a autenticação, sendo a *Dashboard* a página principal imediatamente após o *login*. A partir da *Dashboard*, os usuários podem navegar para todas as outras telas do aplicativo.

5.3.2 Componentes e Navegação

A construção do *frontend mobile* do sistema adota os princípios de componentização e navegação para garantir uma interface de usuário intuitiva e eficiente. Os componentes da interface são encapsulados de forma modular, seguindo as melhores práticas de desenvolvimento em React Native. Cada parte da interface é representada por componentes reutilizáveis, como *TouchableOpacity, Text, ScrollView, View* e *Image*, permitindo uma fácil manutenção do código.

A navegação entre as diferentes telas do aplicativo é gerenciada pela biblioteca React *Navigation*, que oferece um conjunto de navegadores (*navigators*) para facilitar a transição entre os fluxos de tela. O aplicativo utiliza o NativeStackNavigation para organizar e controlar a navegação, garantindo uma experiência de usuário coesa e intuitiva.

As telas do aplicativo abrangem uma variedade de funcionalidades, incluindo informações sobre a escola do aluno, dados pessoais, agenda escolar, frequência, notas, cardápio da merenda, avisos da escola, transporte escolar, recados e carteira estudantil.

Nos **Códigos 34, 35, 36 e 37**, apresentamos trechos do *Dashboard*, que é o componente mais crucial do *frontend*, sendo o único que se comunica diretamente com o *backend*, com exceção da tela Agenda de Recados, que se comunica com o *backend* ao enviar ou receber mensagens. As demais telas recebem os dados por meio de parâmetros durante a navegação.

Código 34 – Dashboard

```
...
export default function Dashboard(){
    ...
    useEffect(() => {
        const loadUser = async () => {
            try{
                const response = await
                   api.post('/pessoas/me/aluno/', {
                    username: user.username
                });
                setAluno(response.data);
                console.log("Status da resposta antes do catch:",
                   response.status);
                setLoading(false);
            }catch(err: any){
                ...
            }
        };

        loadUser();
    }, []);

    useEffect(() => {
        const loadSala = async() => {
            setLoading(true);

            const alunoMatriculado =
               aluno?.objetos_boletins.some(objeto =>
               objeto.objeto_turma.ano === dataAtual.getFullYear());
            setMatriculado(alunoMatriculado);
            const boletins = aluno?.objetos_boletins.filter(objeto
               => objeto.objeto_turma.ano ===
               dataAtual.getFullYear());
            const boletim = boletins && boletins[boletins?.length-1];
            ...
```

Código 35 – Dashboard

```
...

        }

        loadSala();

    }, [aluno])

    const navigation =
       useNavigation<NativeStackNavigationProp<StackAppParamsList>>();

    ...

    async function openMinhaEscola() {
        matriculado ? navigation.navigate('MinhaEscola', {
           minhaEscola: minhaEscola }) : alert("Voc  ainda n o tem
           est  matriculado em nenhuma turma esse ano!");
    }

    async function openPessoal() {
        navigation.navigate('Pessoal', { aluno: aluno });
    }

    async function openCarteira() {
        matriculado ? navigation.navigate('Carteira', { aluno: aluno
           }) : alert('Voc  ainda n o est  matriculado em nehuma
           turma esse ano!');
    }

    async function openMerenda() {
        matriculado ? navigation.navigate('Merenda', { cardapios:
           cardapios }) : alert("Voc  ainda n o tem est
           matriculado em nenhuma turma esse ano!");
    }

    async function openMural() {
        matriculado ? navigation.navigate('Mural', { murais: murais
           }) : alert("Voc  ainda n o tem est  matriculado em
           nenhuma turma esse ano!");
    }

    async function openFrequencia() {
        matriculado ? navigation.navigate('Frequencia', {
           frequencia: frequencia }) : alert('Voc  ainda n o est
           matriculado em nehuma turma esse ano!');
    }
...
```

Código 36 – Dashboard

```
...
return(
    <View style={styles.container}>
        <View style={styles.header}>
            <Image
                style={styles.logoLeft}
                source={require('../../assets/logo_ufs.png')}
            />
            <Image
                style={styles.logo}
                source={require('../../assets/logo.png')}
            />
            <Image
                style={styles.logoRigth}
                source={require('../../assets/logo_seduc.png')}
            />
        </View>
        <ScrollView>
            <View style={styles.lineTextContainer}>
                <Text style={[styles.textId]}>Bom dia
                   {user.first_name}!</Text>
                <Text style={styles.textDate}>Hoje,
                   {formatDay(dataAtual.getDay())},
                   {dataAtual.getDate()} de
                   {formatMounth(dataAtual.getMonth())}</Text>
            </View>

            <View style={styles.lineContainer}>
                <TouchableOpacity
                    style={[styles.button, {marginRight: 20}]}
                    onPress={openMinhaEscola}
                >
                    <View>
                        <Text style={styles.textButton}>Minha</Text>
                        <Text style={styles.textButton}>Escola</Text>
                    </View>
                    <FontAwesome5 name="school" size={45}
                       color='#d9d9d9' />
                </TouchableOpacity>

                <TouchableOpacity
                    style={styles.button}
                    onPress={openPessoal}
                >
                    <Text style={styles.textButton}>Pessoal</Text>
                    <FontAwesome5 name="user-alt" size={45}
                       color='#d9d9d9' />
                </TouchableOpacity>
            </View>
...
```

Código 37 – Dashboard

```
...
                <View style={styles.lineContainer}>
                    <TouchableOpacity
                        style={[styles.button, {marginRight: 20}]}
                        onPress={openRecado}
                    >
                        <Text style={styles.textButton}>Recado</Text>
                        <FontAwesome5 name="comments" size={45}
                           color='#d9d9d9' />
                    </TouchableOpacity>

                    <TouchableOpacity
                        style={styles.button}
                        onPress={openCarteira}
                    >
                        <View>
                            <Text style={styles.textButton}>Carteira
                            </Text>
                            <Text
                               style={styles.textButton}>Estudantil
                            </Text>
                        </View>
                        <FontAwesome5 name="id-card" size={45}
                           color='#d9d9d9' />
                    </TouchableOpacity>
                </View>

                <View style={styles.lineContainer}>
                    <TouchableOpacity
                        style={[styles.button, {marginRight: 20}]}
                        onPress={openHistorico}
                    >
                        <Text style={styles.textButton}>Historico
                        </Text>
                        <FontAwesome5 name="history" size={45}
                           color='#d9d9d9' />
                    </TouchableOpacity>

                    <TouchableOpacity style={styles.button}
                       onPress={logout}>
                        <Text style={styles.textButton}>Sair</Text>
                        <FontAwesome5 name="sign-out-alt" size={45}
                           color='#d9d9d9' />
                    </TouchableOpacity>
                </View>
            </ScrollView>
        </View>
    )
}
```

5.3.3 Consumo de APIs RESTful

O *frontend mobile* se conecta ao *backend* exclusivamente para exibir as informações do usuário logado. Esses dados são obtidos com apenas uma chamada na API /pessoas/me/aluno/, utilizando principalmente o método HTTP *GET*. Essa abordagem simplifica o processo de obtenção de dados, garantindo eficiência e redução da sobrecarga de requisições. No entanto, em determinadas telas, como a de Agenda de Recados, são utilizados tanto o método *GET*, para receber novas mensagens, quanto o método *POST*, para enviar novas mensagens para a Agenda, além da tela de *Login*, que também utiliza o método *POST* para obter os *tokens* de acesso e *refresh*.

Código 38 – Consumindo a api no front mobile

```
...
useEffect(() => {
    const loadUser = async () => {
        try{
            const response = await api.post('/pessoas/me/aluno/', {
                username: user.username
            });

            setAluno(response.data);

            console.log("Status da resposta antes do catch:",
               response.status);
            setLoading(false);

        }catch(err: any){
            console.log('Mensagem de erro loadUser: ', err)
            try{
                await refreshToken();
                await loadUser();
            }catch(error){
                console.log('Mensagem de erro refreshToken: ', error)
                logout();
            }
            setLoading(false);
        }
    };

    loadUser();
}, []);
...
```

Ao estabelecer essa comunicação, o aplicativo *mobile* utiliza a biblioteca Axios, que oferece uma interface simplificada para realizar requisições HTTP. Essa escolha permite que o *frontend* interaja com as APIs RESTful fornecidas pelo *backend* de forma direta e eficiente. Além

disso, o código implementa tratamentos avançados de erros para lidar com possíveis falhas na obtenção de dados do servidor. Isso inclui a utilização de blocos *try-catch* para capturar exceções e a exibição de alertas visuais para fornecer *feedback* ao usuário, garantindo uma experiência de usuário robusta e confiável.

5.3.4 Contexto de Autenticação

Assim como no *frontend web*, o *frontend mobile* utiliza o arquivo AuthContext para consumir as APIs do **SimpleJWT** para gerenciar a autenticação da aplicação. O contexto de autenticação **AuthContext** fornece funcionalidades essenciais para lidar com o **login**, **logout** e atualização dos *tokens* de acesso, garantindo uma experiência segura e fluida para os usuários do aplicativo móvel.

Código 39 – Código para obtenção de tokens de acesso

```
async function obterTokens({ username, password } : LoginProps) {
    setLoadingAuth(true);
    try {
        const response = await api.post('/api/token/', {
            username,
            password,
        });
        const { access, refresh } = response.data;
        // Salva os tokens no AsyncStorage
        await AsyncStorage.setItem('@appseeduca:accessToken',
           access);
        await AsyncStorage.setItem('@appseeduca:refreshToken',
           refresh);
        api.defaults.headers['Authorization'] = `Bearer ${access}`;
        setUser(prevUser => ({
            ...prevUser,
            access,
            refresh,
        }));
        setLoadingAuth(false);
    } catch (error) {
        ...
    }
};
```

Ao utilizar o **SimpleJWT**, o aplicativo móvel pode realizar solicitações de autenticação para o *backend* de forma segura, obtendo tokens de acesso e atualizando-os conforme necessário. Além disso, o **AuthContext** armazena informações do usuário, como nome de usuário e informações de autenticação, proporcionando uma experiência personalizada e mantendo o estado de autenticação do usuário durante a sessão.

Código 40 – Código para realizar o login na aplicação

```
async function login({ username, password } : LoginProps) {
    setLoading(true);
    await obterTokens({ username, password });
    try{
        const response = await api.post('/pessoas/me/aluno/', {
            username
        });
        const data = response.data;
        const { objeto_usuario } = data;
        const { first_name } = objeto_usuario;
        await setUserAsync(username, first_name);
        await AsyncStorage.setItem('@appseeduca:username', username);
        await AsyncStorage.setItem('@appseeduca:firstName',
           first_name);
        console.log(`Usu rio ${user.first_name} logado`);
        setLoading(false);
    }catch(error){
        ...
    }

};
```

Dentro do **AuthContext**, são definidas funções como **login**, **logout** e **refreshToken**, que interagem com as APIs do **SimpleJWT** para lidar com a autenticação. O **AsyncStorage** é utilizado para armazenar os *tokens* de acesso e atualizá-los de acordo com as respostas do *backend*, garantindo a persistência dos dados de autenticação entre sessões.

Código 41 – Provider do contexto envolvendo a aplicação

```
...
return(
    <AuthContext.Provider
     value={{
        user,
        isAuthenticated,
        login,
        refreshToken,
        loadingAuth,
        loading,
        logout
        }}
    >
        {children}
    </AuthContext.Provider>
)
...
```

Dessa forma, o arquivo **AuthContext** desempenha um papel fundamental no *frontend mobile*, permitindo que o aplicativo realize operações de autenticação de forma eficiente e segura, proporcionando uma experiência de usuário satisfatória e protegendo os dados sensíveis da aplicação.

Código 42 – Código para realizar o logout na aplicação

```
async function logout() {
    setLoadingAuth(true);
    try {
        // Limpa os tokens no estado e no AsyncStorage
        setUser({
            username: '',
            first_name: '',
            access: '',
            refresh: '',
        });
        await AsyncStorage.removeItem('@appseeduca:username');
        await AsyncStorage.removeItem('@appseeduca:firstName');
        await AsyncStorage.removeItem('@appseeduca:accessToken');
        await AsyncStorage.removeItem('@appseeduca:refreshToken');
        setLoadingAuth(false);
        navigation.navigate('Login');
    } catch (error) {
        ...
    }
};
```

5.3.5 Controle de Acesso com React *Navigation*

Para implementar o controle de acesso com **React Navigation**, utilizamos os arquivos **auth.routes.tsx** e **app.routes.tsx**. No arquivo **auth.routes.tsx**, definimos as rotas acessíveis para usuários não autenticados, como a tela de *login* e recuperação de senha. Já no arquivo **app.routes.tsx**, especificamos as rotas acessíveis somente para alunos após a autenticação. Diretores, Coordenadores e Professores não tem autorização de acesso ao App.

Além disso, no contexto deste projeto, há uma rota específica **/pessoas/me/aluno/** que retorna os dados do usuário autenticado. Essa rota é fundamental para que o sistema identifique se o usuário é um aluno.

O **Código 43** se refere ao **index.js** do meu componente **Routes** que gerencia quais rotas são acessíveis quando o usuário está autenticado.

Código 43 – Routes

```
import React, { useContext } from "react";
import AppRoutes from "./app.routes";
import AuthRoutes from "./auth.routes";
import { ActivityIndicator, View } from "react-native";
import { AuthContext } from "../contexts/AuthContext";

function Routes(){
    const { isAuthenticated, loading } = useContext(AuthContext);

    if(loading){
        return(
            <View
                style={{
                    flex: 1,
                    backgroundColor: '#d9d9d9',
                    justifyContent: 'center',
                    alignItems: 'center',
                }}
            >
                <ActivityIndicator size={60} color='#02489a'/>
            </View>
        )
    }

    return(
        isAuthenticated ? <AppRoutes/> : <AuthRoutes/>
    )
}

export default Routes;
```

Para mais detalhes sobre o código do *frontend mobile* do SEEduca acesse o repositório: https://github.com/matheusteixeiradcompufs/appSeeduca

6

Testes do Sistema

Apesar da ausência de participação direta dos stakeholders no processo de validação do sistema SEEduca, foram realizados testes abrangentes para garantir sua funcionalidade e confiabilidade. Os testes foram conduzidos em diferentes fases do desenvolvimento, abordando aspectos específicos do sistema.

6.1 Testes Unitários

Para garantir a qualidade e robustez do sistema SEEduca, foram implementados testes unitários abrangentes no *backend*. Um total de 346 testes foram desenvolvidos por mim e aplicados meticulosamente, abrangendo todos os componentes e funcionalidades críticas do sistema. Esses testes estão organizados nas pastas **tests** de cada aplicativo, **pessoas** e **escolas**, seguindo as melhores práticas de desenvolvimento.

É importante destacar que, após a execução desses testes, o resultado foi satisfatório, alcançando uma cobertura de 100%. Isso significa que todas as linhas de código foram exercidas durante os testes, garantindo que não houvesse falhas ou lacunas na lógica de programação.

Os testes estão sendo rodados utilizando a ferramenta **Pytest**, um framework de testes muito utilizado na comunidade Python. O comando **coverage run -m pytest** está sendo utilizado em conjunto com a ferramenta *Coverage* para garantir uma cobertura abrangente do código-fonte.

Além disso, para visualizar de forma mais detalhada o relatório de cobertura, está sendo utilizado o comando **coverage html**, gerando um relatório em formato HTML que oferece uma visão clara e detalhada da cobertura de código alcançada.

A obtenção de uma cobertura de 100% e a aplicação dos testes unitários de forma organizada e automatizada demonstram o rigor e a eficácia do processo de teste implementado no desenvolvimento do SEEduca. Este resultado reflete o compromisso com a excelência e a

qualidade do produto final, proporcionando aos usuários uma experiência confiável e sem falhas. Além disso, a abordagem baseada em testes contribui significativamente para a manutenção do sistema a longo prazo, facilitando a identificação e correção de possíveis problemas à medida que o *software* evolui.

6.2 Testes de Integração e Sistema

Os testes de integração foram conduzidos meticulosamente, abrangendo uma série de cenários complexos para garantir o funcionamento fluido e confiável do SEEduca em todos os seus aspectos. Inicialmente, cada etapa foi minuciosamente planejada e executada, começando com a autenticação do superusuário para garantir a segurança e o controle de acesso ao sistema. Em seguida, os testes de CRUD foram realizados de maneira abrangente e detalhada.

No contexto dos testes de CRUD, foram examinados os seguintes aspectos:

- **Disciplinas**: Foram criadas, visualizadas, editadas e excluídas disciplinas, verificando-se a consistência e a precisão das operações.
- **Escolas, Salas e Turmas**: Para cada escola, foram adicionadas, visualizadas, editadas e excluídas salas e turmas, seguido pela inserção das disciplinas associadas às turmas. Posteriormente, uma Agenda Escolar foi criada para cada turma, e sua gestão semanal foi testada minuciosamente.
- **Funcionários**: Os testes de CRUD envolveram a criação de um diretor, um coordenador e um professor, seguidos pela atribuição do professor à turma previamente criada.
- **Alunos**: Após a criação dos alunos, estes foram matriculados nas turmas correspondentes, verificando-se a correta geração de boletins, controle de frequência e agenda de recados. As presenças foram registradas e notas foram inseridas, juntamente com o envio de mensagens na agenda de recados.

Além disso, foram realizados testes específicos para diferentes tipos de usuários:

- **Diretor**: O acesso e as funcionalidades específicas do diretor foram minuciosamente testados para garantir o controle e a administração adequados do sistema.
- **Coordenador**: Similarmente, as funcionalidades do coordenador foram testadas para garantir uma gestão eficaz das atividades acadêmicas.
- **Professor**: Os testes foram focados na visualização das turmas atribuídas ao professor e na verificação das funções relacionadas ao ensino e à avaliação.

- **Aluno**: O aplicativo foi acessado como aluno para verificar a exibição correta de todas as informações relevantes, incluindo notas, horários e eventos escolares.

Esses testes abrangentes foram essenciais para identificar e corrigir qualquer problema de integração ou funcionamento do sistema, garantindo sua confiabilidade e eficácia em um ambiente educacional dinâmico.

6.3 Resultados dos Testes

Os testes realizados no SEEduca evidenciaram sua capacidade de operar conforme o esperado em diversos cenários. Os resultados dos testes de unidade, integração e sistema atestam que o sistema satisfaz os requisitos funcionais e de desempenho estabelecidos desde a fase de análise e *design*.

6.4 Limitação dos Testes

É importante reconhecer que, devido à falta de participação ativa de *stakeholders*, não foi possível realizar testes de aceitação do usuário. Portanto, embora os testes técnicos tenham sido bem-sucedidos, não há garantia de que o SEEduca atenda completamente às necessidades e expectativas dos usuários finais. Recomenda-se que testes de aceitação do usuário sejam conduzidos no futuro, envolvendo os stakeholders para garantir uma validação mais abrangente do sistema.

7

Documentação do Código

A documentação do código é uma parte muito importante do desenvolvimento de um *software*, especialmente em projetos complexos como o SEEduca, um sistema de gestão escolar que engloba um *backend* Django, um *frontend web* React JS com *Material Dashboard* 2 React e um *frontend mobile* React Native. Neste capítulo, exploraremos as técnicas e ferramentas utilizadas para documentar o código-fonte do Seeduca em cada uma dessas camadas.

7.1 *Backend* Django

O *backend* do Seeduca foi desenvolvido utilizando o *framework* Django, conhecido por sua facilidade de desenvolvimento e robustez. Para documentar o código-fonte, foi utilizado o recurso nativo do Django chamado **admindocs**. O admindocs gera documentação automaticamente a partir das docstrings das views, models e funções de utilidade.

As *docstrings* foram cuidadosamente escritas em cada *view, model* e função para descrever sua funcionalidade, parâmetros e retorno. Isso permite que outros desenvolvedores entendam rapidamente o propósito de cada componente e como utilizá-lo.

Além disso, foram criados diagramas de modelo de dados utilizando a ferramenta **django-extensions**, fornecendo uma visão geral das relações entre os modelos do banco de dados. Esses diagramas ajudam na compreensão da estrutura do banco de dados e nas relações entre as entidades. Para acessar o diagrama basta acessar o arquivo no repositório https://github.com/matheusteixeiradcompufs/seeduca/blob/main/models_seeduca.png.

A documentação do *backend* pode ser acessada a partir da tela de administração do django, realizando o *login* como administrador. Na *navbar* acessando o *link* **Documentação** ou acessando diretamente a rota **/admin/doc/**.

7.2 *Frontend Web* React JS

O *frontend web* do Seeduca foi construído com **React JS**, uma biblioteca JavaScript popular para construção de interfaces de usuário. Para documentar o código-fonte, foi adotado o padrão **JSDoc**, uma convenção para escrever comentários em **JavaScript** que podem ser processados por ferramentas de documentação.

Cada componente React, função e método foram devidamente documentados com comentários **JSDoc**. Esses comentários descrevem o propósito de cada componente, os parâmetros que ele recebe e o que retorna. Isso facilita a compreensão do código e sua reutilização em outros lugares do projeto.

Além disso o **JSDoc** App é uma ferramenta para a geração da documentação. Com o comando **jsdoc** foi gerada a documentação do *frontend web* **SEEduca**, que está disponível para consulta no diretório docs do repositório. Para visualizar a documentação basta acessar o arquivo **index.html** da pasta docs do repositório https://github.com/matheusteixeiradcompufs/webSeeduca.

7.3 *Frontend Mobile* React Native

O *frontend mobile* do Seeduca foi desenvolvido com React Native, uma estrutura para construção de aplicativos móveis multiplataforma com JavaScript e React. Para documentar o código-fonte, foi utilizado o **Typedoc**, uma ferramenta para gerar documentação a partir de comentários **TypeScript**.

Cada componente, função e classe **TypeScript** foram documentados com comentários adequados, seguindo as convenções do **Typedoc**. Esses comentários descrevem a finalidade de cada componente, seus parâmetros e tipos, e o que é retornado.

Assim como no *frontend web*, o **Typedoc** fornece uma ferramenta de geração automática da documentação. Esse relatório também está disponível na pasta docs do repositório e pode ser acessado a partir do arquivo **index.html** do repositório https://github.com/matheusteixeiradcompufs/appSeeduca.

8
Conclusões

A elaboração do sistema SEEDuca foi um desafio gratificante que resultou em uma solução tecnológica robusta e abrangente para a gestão acadêmica das escolas públicas estaduais de Sergipe. Ao longo deste trabalho, busquei desenvolver um sistema que atendesse de forma eficiente às necessidades específicas dessas instituições de ensino, focando na escalabilidade, segurança e integração com outros sistemas já utilizados, além de proporcionar uma melhor comunicação e interação entre as escolas e os alunos, pais ou responsáveis, que sempre foi o objetivo principal.

8.1 Desafios e Limitações

Apesar dos avanços e benefícios alcançados, o desenvolvimento do SEEDuca não esteve isento de desafios. A definição dos requisitos do sistema exigiu uma pesquisa detalhada e a aplicação de questionários aos envolvidos na gestão escolar, o que demandou tempo e esforço significativo. Outro desafio, que se mostrou particularmente difícil, foi a necessidade de desenvolver uma interface *web* para o projeto. Inicialmente o projeto utilizaria o ***django admin*** como *frontend web* para os usuários das escolas, porém a interface se mostrou pouco intuitiva. Essa mudança tornou o escopo do projeto muito maior, tornando necessário o desenvolvimento de muito mais telas do que inicialmente foi concebido no início do projeto.

8.2 Dívidas Técnicas

Durante o desenvolvimento do SEEDuca, algumas dívidas técnicas foram acumuladas devido à falta de uma participação mais direta dos *stakeholders* e à ausência de um ambiente real para implantação e validações com os usuários. A participação limitada dos *stakeholders* resultou em lacunas na definição dos requisitos e funcionalidades, levando a decisões baseadas em suposições. Além disso, a falta de um ambiente real para testes dificultou a identificação

precoce de problemas de usabilidade, desempenho e integração com outros sistemas. Para mitigar essas dívidas, é necessário envolver continuamente os *stakeholders*, criar ambientes de teste mais realistas e adotar ciclos de desenvolvimento iterativos e ágeis.

8.3 Perspectivas Futuras

O SEEDuca representa um passo significativo na digitalização e melhoria da gestão escolar nas instituições públicas de Sergipe. No entanto, há sempre espaço para melhorias e inovações contínuas. Futuras versões do sistema podem incluir novas funcionalidades baseadas no *feedback* dos usuários, como módulos adicionais para gestão de eventos escolares, integração com plataformas de ensino a distância e melhorias na interface de usuário para torná-la ainda mais intuitiva e acessível. Outra nova melhoria que deve ser implementada é o acrescimo do ator pai ou responsável de alunos pois os mesmos poderiam trocar informações diretamente com a escola além de poder gerenciar mais de um aluno no mesmo acesso, no caso do mesmo possuir mais de um aluno matriculado.

9

Considerações Finais

O projeto SEEDuca demonstrou ser uma ferramenta poderosa para a modernização da gestão escolar. Por meio da implementação de funcionalidades como o registro de presença, gerenciamento de notas e agenda escolar, foi possível simplificar e agilizar processos administrativos, proporcionando uma experiência mais eficiente para professores, alunos e gestores. Além disso, a comunicação entre escola, alunos e responsáveis ficou mais eficiente com a utilização da agenda escolar, dos recados e do mural de avisos.

Sua implementação tem o potencial de impactar positivamente a qualidade da educação nas escolas públicas estaduais. Com um sistema de gestão escolar mais organizado e acessível, os professores podem focar mais no ensino, os alunos têm acesso facilitado às informações escolares, e os pais podem acompanhar de perto o desempenho e a frequência de seus filhos. Esse conjunto de melhorias contribui para um ambiente educacional mais transparente e eficiente, potencializando o desenvolvimento acadêmico dos alunos.

Em suma, o desenvolvimento do SEEDuca não só atendeu aos objetivos propostos inicialmente, mas também abriu portas para futuras inovações na gestão escolar. Este trabalho contribui de maneira significativa para a modernização do sistema educacional público de Sergipe, mostrando como a tecnologia pode ser uma aliada poderosa na melhoria da educação. A continuidade do projeto e o investimento em atualizações e novas funcionalidades serão essenciais para manter o sistema alinhado às necessidades e expectativas dos usuários, garantindo, assim, a sua relevância e eficiência a longo prazo.

Referências

ARAúJO, P.; COSTA, A. A revolução digital na gestão escolar: Estudo de caso em escolas públicas. In: *Anais do Congresso Brasileiro de Educação*. [S.l.: s.n.], 2018. v. 8, p. 122–138. Citado na página 17.

BAE, J. An empirical study on the use of django in web development. *Journal of Web Engineering*, v. 18, n. 2, p. 157–178, 2019. Citado na página 28.

BAXTER, M.; CHAN, K. A review of figma for collaborative ui design and prototyping. *International Journal of Human-Computer Interaction*, v. 37, n. 6, p. 567–580, 2021. Citado na página 30.

BROWN, M.; WHITE, L. A comprehensive review of react native for cross-platform mobile application development. *International Journal of Mobile Human Computer Interaction*, v. 12, n. 3, p. 45–58, 2020. Citado na página 29.

CHAN, A.; PATEL, S. The impact of figma on design collaboration and remote work. *Journal of Design and Communication*, v. 8, n. 2, p. 78–92, 2020. Citado na página 30.

DEWEY, J. *Democracia e Educação*. [S.l.]: Edições Pedagógicas, 2000. Citado na página 23.

DUFRESNE, K.; SOBOLEV, A. An analysis of django rest framework for web api development. *International Journal of Computer Science and Information Technologies*, v. 9, n. 3, p. 47–54, 2018. Citado na página 28.

Escola Paraná. *Escola Paraná*. 2017. Disponível em: <https://play.google.com/store/apps/details?id=br.gov.pr.celepar.escolapr>. Aplicativo para Android. Citado na página 33.

Escola RS. *Escola RS*. 2019. Disponível em: <https://play.google.com/store/apps/details?id=br.com.procergs.estudante>. Aplicativo para Android. Citado na página 34.

Estudante SC. *Estudante SC*. 2017. Disponível em: <https://play.google.com/store/apps/details?id=br.sc.gov.ciasc.sed>. Aplicativo para Android. Citado na página 35.

FACEBOOK. *React Native - Documentação Oficial*. 2023. Disponível em: <https://reactnative.dev/>. Acessado em 07 de setembro de 2023. Citado na página 29.

FAUSTO, B. *História Concisa do Brasil*. São Paulo: Edusp, 2012. Citado na página 22.

FOUNDATION, D. S. *Django Framework - Documentação Oficial*. 2023. Disponível em: <https://www.djangoproject.com/>. Acessado em 07 de setembro de 2023. Citado na página 28.

FREIRE, P. *Pedagogia da Esperança*. [S.l.]: Paz & Terra, 1992. Citado na página 22.

FREIRE, P. *Pedagogia da Autonomia: Saberes Necessários à Prática Educativa*. São Paulo: Paz e Terra, 1996. Citado na página 22.

GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. [S.l.]: Addison-Wesley, 1994. Citado na página 51.

HADDAD, S. *Educação e direitos humanos: a necessidade de uma agenda inclusiva*. São Paulo: Moderna, 2009. Citado na página 22.

HARRISON, J.; PATEL, R. A review of draw.io for diagram creation and collaboration. *Journal of Information Technology Research*, v. 13, n. 2, p. 45–58, 2020. Citado na página 29.

JETBRAINS. *Pycharm - Documentação Oficial*. 2023. Disponível em: <https://www.jetbrains.com/pycharm/>. Acessado em 07 de setembro de 2023. Citado na página 28.

JGRAPH. *Draw.io - Documentação Oficial*. 2023. Disponível em: <https://www.diagrams.net/>. Acessado em 07 de setembro de 2023. Citado na página 29.

JOHNSON, K. Evaluating the efficiency and advanced features of pycharm as a python ide. *Journal of Software Engineering and Applications*, v. 12, n. 4, p. 123–130, 2019. Citado na página 28.

LIBâNEO, J. C. *Didática*. São Paulo: Cortez, 2013. Citado na página 22.

LüCK, H. *Gestão escolar e qualidade do ensino: Formação e integração de equipes de trabalho.* [S.l.]: Vozes, 2009. Citado 2 vezes nas páginas 17 e 23.

MICROSOFT. *Visual Studio Code - Documentação Oficial*. 2023. Disponível em: <https://code.visualstudio.com/>. Acessado em 07 de setembro de 2023. Citado na página 29.

Minha Escola SP. *Minha Escola SP*. 2018. Disponível em: <https://play.google.com/store/apps/details?id=br.gov.sp.educacao.minhaescola>. Aplicativo para Android. Citado na página 32.

MUSSER, J. *APIs: A Strategy Guide*. [S.l.]: O'Reilly Media, 2011. Citado na página 26.

NetEscola. *NetEscola*. 2020. Disponível em: <https://play.google.com/store/apps/details?id=br.gov.go.seduc.netescola>. Aplicativo para Android. Citado na página 35.

PAN, W. Research on construction of normative ideological instruction teaching management system based on zigbee wireless sensor network. *Computer Modelling and New Technologies*, v. 18, n. 14075806, p. 95 – 99, 2014. Citado na página 41.

PARO, V. H. *Gestão Democrática da Escola Pública*. São Paulo: Cortez, 2001. Citado na página 23.

PRESSMAN, R. S.; MAXIM, B. R. *Engenharia de Software: Uma Abordagem Profissional*. [S.l.]: AMGH, 2020. Citado 2 vezes nas páginas 46 e 48.

SAITO, M. A comparative study of web application frameworks with a focus on django. *International Journal of Web Engineering and Technology*, v. 15, n. 1, p. 45–62, 2021. Citado na página 28.

SANTOS, A. C.; MONTEIRO, J. A. Building restful web apis with django rest framework. *Journal of Web Development and Web Design*, v. 5, n. 1, p. 34–48, 2020. Citado na página 28.

SANTOS, E.; ALMEIDA, G. Gestão escolar digital: Impactos e perspectivas. *Revista Brasileira de Educação*, v. 25, n. 79, p. 1–20, 2020. Citado na página 17.

SAVIANI, D. *Escola e Democracia*. Campinas, SP: Autores Associados, 2003. Citado na página 22.

SMITH, J.; BROWN, A. Comprehensive review of pycharm ide for python development. *International Journal of Advanced Computer Science and Applications*, v. 12, n. 5, p. 101–110, 2021. Citado na página 28.

SOMMERVILLE, I. *Engenharia de Software*. [S.l.]: Pearson, 2011. Citado na página 50.

SUN, A. et al. Wearable mobile internet devices involved in big data solution for education. *International Journal of Embedded Systems*, v. 8, n. 17411068, p. 293–299, 2016. Citado na página 41.

TANNEMBAUM, A. S. *Modern Operating Systems*. [S.l.]: Pearson Education, 2014. Citado na página 25.

TEDESCO, J. C. *Educação e sociedade nas Américas: desafios do século XXI*. São Paulo: Cortez, 2007. Citado na página 22.

VEIGA, I. P. A. *Gestão Educacional: Novos Olhares*. Campinas, SP: Papirus, 2008. Citado na página 23.

WILSON, P.; THOMPSON, R. Evaluating the efficiency and flexibility of visual studio code as a multilanguage development environment. *Journal of Software Engineering and Applications*, v. 13, n. 7, p. 299–310, 2020. Citado na página 29.

XUN, H.; GOSTARIA, B.; DUAN, C. Desenho do sistema de gestão de frequência estudantil baseado na tecnologia da internet das coisas. v. 2022, n. 1990530, 2022. Citado na página 40.

ZHANG, M.; LI, X. Projeto de sistema de sala de aula inteligente baseado em tecnologia de internet das coisas e sala de aula inteligente. v. 2021, n. 5438878, 2021. Citado na página 40.