

UNIVERSIDADE PAULISTA

Danyllo Dias Martins

Wagner de Oliveira Martins

Pedro Henrique Oliveira Soares

Bruno Cerqueira De Andrade

Vitor de Oliveira Vieira

Projeto automatização gestão escolar

PIM IV

BRASÍLIA

2023

Danyllo Dias Martins
Wagner de Oliveira Martins
Pedro Henrique Oliveira Soares
Bruno Cerqueira De Andrade
Vitor de Oliveira Vieira

Projeto automatização gestão escolar
PIM IV

Projeto Integrado Multidisciplinar III do curso de análise e Desenvolvimento de Sistemas apresentado à Universidade Paulista – UNIP.

Orientador: Prof. MSc. Nathaniel Simch de Moraes

BRASÍLIA
2023

RESUMO

Este projeto tem como foco uma empresa que busca melhorar os serviços de gestão escolar. Procurando criar um ambiente educacional mais eficiente e de alta qualidade, com ênfase na automatização dos processos acadêmicos e administrativos. O principal objetivo é criar um sistema de gestão escolar abrangente, com isso o sistema compreende desde o cadastro de alunos até a geração de relatórios relacionados a atividades como matrícula, cursos, disciplinas, notas e faltas.

Ao longo desse projeto, foram documentadas todas as estruturas essenciais para abordar esses aspectos, isso inclui aspectos como design do banco de dados, a utilização de diagramas UML para representação de metodologias ágeis para um desenvolvimento flexível e eficaz, com escolhas feitas com critérios e fundamentadas nos motivos das linguagens de programação e a criação de protótipos de interfaces, com o objetivo de oferecer uma visualização clara e intuitiva do sistema.

Palavras-chave: TI; sistema escolar; análise de sistemas.

ABSTRACT

This project focuses on a company seeking to improve school management services. It seeks to create a more efficient and high-quality educational environment, with an emphasis on automating academic and administrative processes. The main objective is to create a comprehensive school management system, which covers everything from student registration to the generation of reports related to activities such as enrollment, courses, subjects, grades, and absences.

Throughout this project, all the essential structures to address these aspects have been documented, including aspects such as database design, the use of UML diagrams to represent agile methodologies for flexible and effective development, with choices made with criteria and based on the motives of the programming languages, and the creation of interface prototypes, with the aim of offering a clear and intuitive visualization of the system.

Key-words: IT; school system; system analysts.

SUMÁRIO

Sumário

1. INTRODUÇÃO	6
2. FUNDAMENTAÇÃO TEÓRICA	9
2.1 Banco de dados	9
2.1.1 Banco de Dados Relacional	9
2.1.2 Ferramentas de Banco de Dados	9
2.1.3 Linguagem SQL	9
2.1.4 Modelagem de Dados:	10
2.1.4.1 Diagrama ER	10
2.1.4.2 Atributos	10
2.1.4.3 Cardinalidade	10
2.1.4.4 Entidades	11
2.1.4.5 Relacionamentos	11
2.1.4.6 SQL Server	11
2.2 Diagrama UML	12
2.2.1 Diagrama de Casos de uso	12
2.2.2 Diagrama de Classes	12
2.3 Linguagem de Programação	13
2.3.1 Back-end	13
2.3.2 Front-end	13
2.3.3 Orientação a objetos	14
2.3.4 Framework	14
2.3.5 Linguagem JavaScript	15
2.3.5.1 Node	15
2.3.5.2 Npm	15
2.3.5.3 Electron	15
2.3.5.4 React	16
2.3.5.5 React Native	16
2.3.5.6 Vite	16
2.3.5.7 Tailwind CSS	17
2.3.5.8 Shadcn UI	17
2.3.5.9 Express	17
2.3.5.10 Prisma ORM	17
2.3.5.11 Typescript	17
2.3.5.12 Axios	18
2.3.5.13 Expo GO	18
2.4 Metodologia Ágil	19
2.5 LGPD (Lei Geral de Proteção de Dados Pessoais)	19
2.6 Prototipação	20
2.7 Scrum	20
2.8 Docker	21

3. DESENVOLVIMENTO	22
3.1 Banco de Dados	22
3.1.1 Diagrama ER	22
3.1.2 Modelo Lógico	23
3.1.3 Dicionário de Dados	24
3.1.4 Docker	29
3.1.5 Criação do banco de dados	29
3.2 Casos de Uso	29
3.2.1 Diagramas de Casos de Uso	31
3.2.2 Detalhamento de Casos de Uso	33
3.3 Cenário	52
3.4 Ciclo de vida do software	52
3.5 LGPD	53
3.6 Diagrama de Classes	53
3.7 Elicitação de Requisitos	53
3.8 Glossário do Sistema	55
3.9 Desenvolvimento do sistema	55
3.9.1 Linguagem de Programação	56
3.9.1.1 Linguagem JavaScript	56
3.9.1.2 Typescript	56
3.9.1.3 React	56
3.9.1.4 Electron	56
3.9.1.5 React Native	57
3.9.1.6 Expo GO	57
3.9.2 Criação da interface	57
3.9.2.2 Tailwind CSS	57
3.9.2.3 Shadcn UI	58
3.9.3 Criação do Back End	58
3.9.3.1 Prisma	58
3.9.3.2 Express	58
3.9.3.3 Axios	58
3.9.4 Código da aplicação	59
3.9.5 Protótipo	59
3.9.6 Guia de utilização da aplicação	62
4. CONCLUSÃO	68
Referências Bibliográficas	70
APÊNDICE	72
Apêndice A - Entrevista	72
Apêndice B - Diagrama de Classes	73
Apêndice C - Código de criação do banco de dados	74

1. INTRODUÇÃO

A gestão escolar desempenha um papel crucial na garantia de que os processos administrativos e acadêmicos sejam orientados de maneira eficaz e eficiente nas instituições de ensino. No entanto, essa tarefa complexa pode ser otimizada para liberar tempo para o que realmente importa: a entrega de uma educação de qualidade. Nesse contexto, a automação de serviços se apresenta como um imperativo, uma solução que visa acelerar operações, reduzir erros cometidos e permitir que educadores e administradores tenham foco para se concentrarem em proporcionar uma boa experiência educacional para todos.

Este projeto integrado multidisciplinar (PIM) se destina a explorar a precisão da automação de serviços na gestão escolar e seu impacto na eficiência das instituições de ensino. Para alcançar todo esse objetivo, concordamos em utilizar metodologias ágeis, especificamente o Scrum, que promovem a flexibilidade, a colaboração e a entrega iterativa de soluções. Além do mais, empregamos uma variedade de diagramas para representação visual do projeto, fornecendo uma compreensão mais simples de sua estrutura e usabilidade. Como parte dessa documentação abrangente, incluímos diagramas de entidade-relacionamento do banco de dados, diagramas de casos de uso para ilustrar interações entre os atores e o sistema, e diagramas de classes que detalham os atributos e métodos do conjunto de classes do sistema.

Já na parte de desenvolvimento foram aplicadas tecnologias e ferramentas de precisão para atingir os objetivos propostos. Com foco em proporcionar uma experiência de usuário eficiente e amigável, a linguagem de programação JavaScript e suas bibliotecas foram fundamentais para alcançar o objetivo. Na parte do front-end, recorreremos ao React que oferece um ambiente de desenvolvimento de interfaces de usuário interativas e responsivas. Isso assegura que os educadores, administradores e outros atores possam acessar o sistema de forma intuitiva e eficaz. Já na parte do back end, recorreremos ao Express e Prisma ORM. Para a aplicação da lógica do sistema e o controle de dados, foram utilizadas as tecnologias. Express é um framework web minimalista e altamente flexível que permitiu a construção de uma infraestrutura robusta para a parte de back end. O Prisma ORM, por sua vez, simplifica o gerenciamento e a persistência de dados,

garantindo uma base sólida para a integridade e a confiabilidade das informações escolares.

E com isso a segurança e a privacidade dos dados pessoais são importantes na gestão escolar. A Lei Geral de Proteção de Dados(LGPD) emerge como um marco regulatório que reforça a proteção dos dados pessoais dos alunos, professores e funcionários. Ela determina as diretrizes rígidas para a coleta, armazenamento, processamento e compartilhamento de dados pessoais, garantindo que os direitos fundamentais à privacidade e à transparência sejam preservados.

1.1 Objetivo geral

Este trabalho tem como objetivo desenvolver com abrangência um sistema de gestão escolar, com foco nas plataformas web, desktop e mobile, com o propósito na eficácia da gestão escolar. Este sistema tem o objetivo de gerar uma solução pronta para uso, Com base nas melhores práticas, que seja capaz de otimizar tanto os aspectos administrativos quanto acadêmico das escolas, o esforço visa estabelecer uma documentação abrangente e de fácil compreensão. destinada a servir como guia de referência indispensável para a implementação de sistemas de gestão escolar eficazes em instituições de ensino. O sistema e a documentação desenvolvidos neste trabalho têm o objetivo de estabelecer uma base segura para aprimorar os processos de gestão escolar. Com isso permitirá que educadores e administradores aloquem mais tempo para oferecer uma educação de qualidade.

1.2 Objetivos específicos

- Elaborar Diagramas de caso de uso;
- Elaborar Diagrama de classes;
- Elaborar Diagrama de entidade relacionamento;
- Elicitar requisitos para o sistema;
- Especificar a linguagem de programação no back end e no front end;
- Especificar o método utilizado para a construção do aplicativo mobile;
- Apresentar tecnologias para a criação de interfaces;

- Apresentar tecnologias para criação do back end;
- Apresentar a forma da criação do banco de dados
- Apresentar protótipos de interface para o projeto;
- Apresentar uma metodologia ágil para o projeto;

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Banco de dados

Segundo Banco de dados e sua definição (MANOVICH 2015) afirma que:

Banco de dados é definido como uma coleção estruturada de dados. Os dados armazenados em um banco de dados são organizados de forma a permitir agilidade na busca e na recuperação por um computador, ou seja, não há nada além de uma simples coleção de itens.

2.1.1 Banco de Dados Relacional

Um banco de dados relacional consiste em tabelas que se relacionam entre si e que por sua vez essas tabelas possuem colunas e linhas, conforme MACÁRIO et al. (2005, p.3). Ainda segundo MACÁRIO et al (2005, p.3) “Um banco de dados relacional é um conjunto de uma ou mais relações com nomes distintos. O esquema do banco de dados relacional é a coleção dos esquemas de cada relação que compõem o banco de dados.”.

2.1.2 Ferramentas de Banco de Dados

Também conhecida como SGBD (Sistema de Gerenciamento de Banco de Dados), uma coleção de programas que permitem aos usuários criarem e manipularem uma base de dados. Um SGBD facilita o processo de construir, definir e manipular dados. (MEIRA, 2005)

2.1.3 Linguagem SQL

SQL é a linguagem mais largamente aceita e utilizada para manipulação e acesso a bancos de dados. Tipicamente, uma unidade de um curso de BD pode ser organizada da seguinte forma: introdução e motivação; sintaxe do comando; lista de exemplos comentados; lista de exercícios (conceituais e práticos) e avaliação (DA SILVEIRA, 2010.)

2.1.4 Modelagem de Dados:

Segundo A Modelagem de Dados é um processo fundamental na construção de um Banco de Dados eficiente e eficaz. Os principais conceitos envolvidos na Modelagem de Dados são Entidade, Atributo, Relacionamento e Restrições.

2.1.4.1 Diagrama ER

Segundo Nogueira (1988), “Um diagrama entidade relacionamento (ER) é um tipo de fluxograma que ilustra como “entidades”, pessoas, objetos ou conceitos, se relacionam entre si dentro de um sistema.”.

2.1.4.2 Atributos

Um atributo pode ser: simples ou atômico, quando não pode ser dividido, como nome; composto, que é dividido em várias partes, como endereço, que é dividido em rua, número e CEP; de valor único, que tem um só valor para uma determinada entidade, como idade; multivalorado, que pode assumir vários valores para uma determinada entidade, como telefone; e derivado, que é um atributo que pode ser determinado a partir de outros atributos, como idade, que pode ser calculada a partir da data de nascimento e da data de hoje.(FRANCK et al, 2021).

2.1.4.3 Cardinalidade

Sobre cardinalidade e sua definição é possível afirmar:

A cardinalidade de um relacionamento especifica o número máximo de instâncias do relacionamento em que uma entidade pode participar. Por exemplo, o tipo de relacionamento entre empregado e departamento tem cardinalidade N:1, o que significa que cada empregado pode trabalhar em somente um departamento, enquanto que em cada departamento podem trabalhar inúmeros empregados. As possíveis cardinalidades são: 1:1 onde uma entidade está relacionada a apenas uma outra entidade, 1:N quando uma entidade está relacionada a várias outras no outro lado , N:1 Nessa cardinalidade, várias entidades em um lado do relacionamento estão associadas a uma única entidade no outro lado. e por fim M:N que é quando várias entidades em um lado do relacionamento estão associadas a várias entidades no outro lado. (MATOS, 2016)

2.1.4.4 Entidades

“Uma entidade representa um elemento do mundo real, como um funcionário ou uma disciplina.”(FRANCK et al., 2021), diante disso entidades são fundamentais para representar usuários, pessoas ou categorias, essas definidas através dos diagramas de entidade relacionamento.

2.1.4.5 Relacionamentos

Um relacionamento entre tabelas é uma associação entre duas ou mais tabelas de um banco de dados relacional. Os relacionamentos são usados para conectar dados de diferentes tabelas, permitindo que sejam recuperados e analisados de forma lógica.

Em UML, uma associação é definida como um relacionamento que descreve um conjunto de ligações, onde cada ligação é definida como uma conexão semântica entre os objetos, que são instâncias das classes participantes da associação. (MEDEIROS, 2004).

2.1.4.6 SQL Server

De um modo geral o SQL Server é um SGBD(Sistema de gerenciamento de banco de dados) poderoso e confiável, que oferece muitos recursos desejáveis em um bom SGBD. Dentre as principais características do SQL Server destacam-se a alta disponibilidade, desempenho e escalabilidade, segurança, gerenciamento e produtividade

Dos casos de sucesso de implantação do SQL Server observa-se com sucesso a atualização do sistema do banco Nossa Caixa no estado de São Paulo. O banco teve um aumento de cento e dezenove mil transações eletrônicas em 2002 para um milhão e setecentos mil transações em 2007. Para atender a este crescimento, uma operação de atualização do sistema foi iniciada em 2007 e como principais benefícios do novo sistema observa-se o aumento do número de

transações por segundo, melhor desempenho na estrutura, facilidade de gestão e aumento na segurança.

2.2 Diagrama UML

Segundo (Vargas, 2014) a UML (Unified Modeling Language ou Linguagem Unificada de Modelagem, em português) é "uma linguagem para especificação, documentação, visualização e desenvolvimento de sistemas orientados a objetos", auxiliando na descrição de projetos de software é apoiada por um modelo único; considerada uma das linguagens mais expressivas, pois facilita a comunicação entre todas as pessoas envolvidas no projeto. Nasceu da união de muitas linguagens gráficas orientadas a objetos.

2.2.1 Diagrama de Casos de uso

De acordo com (John Smith, 2023), especialista em análise e design de sistemas, comenta em seu artigo "Introdução aos Diagramas de Caso de Uso: Conceitos e Aplicações" que o diagrama de caso de uso é usado para modelar a interação entre os autores (usuários externos ou sistemas) e o sistema em questão.

Nesse artigo aborda também os principais elementos de um diagrama de caso de uso, como atores, casos de uso, associações, inclusões e extensões, sendo cada elemento essencial para a criação de um diagrama de caso de uso.

E no final o autor também enfatiza sobre a importância contínua dos diagramas de caso de uso como uma excelente ferramenta para a engenharia de software e destaca seu papel na criação de sistemas centrados no usuário.

2.2.2 Diagrama de Classes

Para (Larman, 2007), o Diagrama de Classes serve para ilustrar classes, interfaces e suas associações. Eles são usados para modelagem estática de objetos. O Diagrama de Classes é considerado por muitos autores como o mais importante e o mais utilizado diagrama da UML. Seu principal enfoque está em permitir a visualização das classes que irão compor o sistema com seus respectivos atributos e métodos, bem como demonstrar como as classes do sistema se relacionam, se complementam e transmitem informações entre si. Este diagrama

apresenta uma visão estática de como as classes estão organizadas, preocupando-se em definir a estrutura lógica das mesmas. O Diagrama de Classes serve como base para a construção da maior parte dos demais diagramas da UML.

2.3 Linguagem de Programação

De acordo com (Kanakadoss, 2005), a programação é um processo de transformação de um plano mental, que se encontra estruturado de uma forma familiar para a pessoa, numa outra forma que é compatível com a linguagem compreendida pelo computador. Deste modo, para Mayer (citado por Kanakadoss, 2005), um aluno deve conhecer em detalhe os processos a decorrer no computador para se tornar um programador de sucesso.

2.3.1 Back-end

De acordo com (MARQUEZ - SOTO, 2022). O Back-end refere-se a parte de um aplicativo ou código de um programa que permite que ele funcione e que não pode ser acessado por um usuário. “Desenvolvedores Back End trabalham em uma camada que é abstraída do usuário”. A maioria dos dados e da sintaxe operacional é armazenada e acessada no back end de um sistema. Normalmente, o código é composto por uma ou mais linguagens de programação. O backend também é chamado de camada de acesso a dados de software ou hardware e inclui qualquer funcionalidade que precise ser acessada e navegada por meios digitais.

2.3.2 Front-end

No livro "Front-End Web Development: A Complete Guide" de (Gerardus Blokdyk, 2018), é falado sobre a definição do front end e qual é a sua parte no desenvolvimento de um software. Front end é a área da programação onde se tem a interação do usuário com o software, ou seja é a interface onde o usuário tem o conteúdo visual e funcional tanto de um site como de um aplicativo para celulares.

2.3.2.1 HTML

De acordo com (Brooks, 2007). HTML (acrônimo para HyperText Markup Language) é uma linguagem de marcação usada para especificar a estrutura

de um documento. Um navegador de internet (web browser) nada mais é do que um software que interpreta estas marcações de estrutura e, então, constrói uma página web com recursos de hipermídia com os quais o usuário pode interagir.

2.3.2.2 CSS

De acordo com (GRANNEL, 2007). CSS (acrônimo para Cascading Style Sheets) é uma linguagem de estilo usada para especificar a aparência (layout, cor e fonte) dos vários elementos de um documento que foi definido por uma linguagem de marcação (como a linguagem HTML). Ela foi criada com o objetivo de separar a estrutura do documento de sua aparência.

2.3.3 Orientação a objetos

De acordo com (Farinelli, 2007), A Orientação a Objetos é uma tecnologia que enxerga os sistemas como sendo coleção de objetos integrantes. Ela permite melhorar a reusabilidade e a extensibilidade dos softwares. A tecnologia orientada a objetos é fundamentada no que, coletivamente, chamamos de modelo de objetos, que engloba os princípios da abstração, hierarquização, encapsulamento, classificação, modularização, relacionamento, simultaneidade e persistência.

2.3.4 Framework

É uma técnica da orientação a objetos, voltada para a reutilização que se beneficia de três características das linguagens de programação orientadas a objetos: abstração, polimorfismo e herança. Ele também pode ser vislumbrado como o esqueleto - template - de algo que possa ser customizado pelo desenvolvedor e aplicado para um conjunto de aplicações de um mesmo domínio.

Para (Johnson, 1997) a visão geral de reutilização de software estava baseada em componentes. Mas mesmo que os frameworks tenham uma interface mais complexa, eles são demais customização que um componente. O autor faz a observação de que componentes e frameworks são técnicas diferentes, pois frameworks e componentes podem cooperar entre si, além de que frameworks podem facilitar a construção de novos componentes.

2.3.5 Linguagem JavaScript

Atualmente o nome oficial do JavaScript é ECMAScript pois é a junção das normas ECMA (European Computer Manufacturers Association) é a linguagem de programação script que é uma linguagem usada para manipular, personalizar e automatizar as funcionalidades de um sistema já existente.

O ECMAScript é uma linguagem orientada a objetos que realiza cálculos e manipula objetos computacionais de um ambiente de hospedagem.

Um exemplo de ambiente de hospedagem é um navegador web, pois ele fica do lado do cliente e hospeda objetos que representam janelas, menus, área de texto, histórico, cookies entre outros objetos.(Silva, 2020)

2.3.5.1 Node

Para (Tilkov, 2010) o node é um framework de interesse da comunidade de desenvolvedores JavaScript, este framework tem a função de permitir a execução de códigos fora do navegador web, é também uma de suas características mais interessantes e sua arquitetura assíncrona e orientada a eventos.

2.3.5.2 Npm

O npm (node package manager) é um gerenciador de pacotes node, no site oficial do npm¹ cita que o npm tem como utilidade a instalação de ferramentas, frameworks e vários outros tipos de pacotes para a facilitação do desenvolvimento ou para ter acesso a mais recursos para se utilizar dentro do JavaScript.

2.3.5.3 Electron

Electron é um framework JavaScript que permite a criação de aplicações desktop utilizando html, css, javascript e node. Com ele é possível criar toda interface como também toda lógica back end que acessa recursos do sistema

¹ <https://www.npmjs.com>

operacional, e para mais detalhes o site oficial do electron² tem toda a sua documentação

2.3.5.4 React

Em seu livro (Boduch, 2017) explora o React, sendo no início a explicação de o que é o React. O React nada mais é que uma biblioteca do JavaScript para criação de interfaces de usuário, essa biblioteca tem o foco em separar o código, eventos, vida útil e os dados em componentes, é só após compilado e transformado em código html, css e javascript onde esse componentes são unificados em um único código.

2.3.5.5 React Native³

Segundo a própria documentação oficial do React Native ela é uma biblioteca que é usada para criar aplicações multi plataformas com somente um único código para elas. Ou seja, é possível escrever um código que rodará em aparelhos iPhone e Android.

A base do React Native é React e JavaScript, além disso, os aplicativos criados com o React Native são rodados nativamente e possibilitam ser introduzidos no Google Play Store ou no Apple Store.

2.3.5.6 Vite

De acordo com sua documentação oficial⁴, no nível básico, desenvolver usando o Vite não é muito diferente de usar um servidor de arquivos estático. No entanto, o Vite fornece muitos aprimoramentos em relação às importações nativas do ESM para oferecer suporte a vários recursos que normalmente são vistos em configurações baseadas em empacotadores.

² <https://www.electronjs.org/pt/>

³ <https://reactnativeui.devkit.com/docs/>

⁴ <https://vitejs.dev/guide/>

2.3.5.7 Tailwind CSS

De acordo com sua documentação oficial⁵, Tailwind CSS é um framework desenvolvido para maximizar o potencial do bom e velho CSS e levá-lo ainda mais longe. De forma bastante simplificada e intuitiva, ele oferece responsividade, código enxuto, customização e integração com IDEs.

2.3.5.8 Shadcn UI

De acordo com sua documentação oficial⁶, é uma coleção de componentes reutilizáveis que você pode copiar e colar em seus aplicativos. Então basicamente você escolhe os componentes que você precisa. Copie e cole o código em seu projeto e personalize de acordo com suas necessidades.

2.3.5.9 Express

Express como explica seu site oficial⁷, é uma estrutura de aplicativo da web Node.js mínima e flexível que fornece um conjunto robusto de recursos para aplicativos da web e móveis. Com uma infinidade de métodos utilitários HTTP e middleware(cola de software) à sua disposição, criar uma API robusta é rápido e fácil de utilizar.

2.3.5.10 Prisma ORM

O Prisma trata-se de uma ferramenta ORM (Object Relational Mapper) também de código aberto. Seu objetivo é facilitar para o desenvolvedor, a comunicação entre os objetos TypeScript e os dados de banco que os representam, através de métodos que dispensam a necessidade de codificação de elaboradas consultas em SQL. Outra grande vantagem está no fato de que o Prisma conta com um sistema de migração para o banco (PRISMA, 2022).

2.3.5.11 Typescript

A despeito do seu sucesso, JavaScript tornou-se uma linguagem fraca para o desenvolvimento e manutenção de grandes aplicações. TypeScript é

⁵ <https://tailwindcss.com/>

⁶ <https://ui.shadcn.com/docs>

⁷ <https://expressjs.com/pt-br/>

uma extensão do JavaScript com intenção de suprir sua deficiência. Sintaticamente, TypeScript é um superset do EcmaScript, então todo programa JavaScript é um programa TypeScript. TypeScript agrega ao JavaScript um sistema de módulos, classes, interfaces, e um sistema de tipagem estática. Como o TypeScript visa prover uma leve assistência aos programadores, o sistema de módulos e sistema de tipos é fácil de usar. Em particular, ele suporta muitas práticas comuns ao JavaScript. Ele também possibilita o uso de código assistido por parte das IDEs³, o que sempre foi muito comum em linguagens como o C# e o Java. (Bierman, Gavin, Martín Abadi, Torgersen, 2014).

2.3.5.12 Axios

De acordo com sua documentação oficial.⁸ Axios é um cliente HTTP baseado em promessas para o node.js para o navegador. É isomórfico (pode rodar no navegador e no node.js com a mesma base de código). No lado do servidor usa o código nativo do node.js - o módulo http, enquanto no lado do cliente (navegador) usa XMLHttpRequests.

2.3.5.13 Expo GO

De acordo com sua documentação oficial.⁹ Expo Go é um sandbox gratuito e de código aberto para aprender e experimentar React Native em dispositivos Android e iOS. Você pode instalá-lo diretamente das lojas de aplicativos e colocá-lo em funcionamento em minutos sem necessidade de instalar um conjunto de ferramentas nativo e compilar um aplicativo.

Expo Go não é recomendado para criar aplicativos com qualidade de produção – em vez disso, use compilações de desenvolvimento. Também não é recomendado para implantação/distribuição de aplicativos. Dito isso, tudo o que você aprender enquanto trabalha no Expo Go será transferível para compilações de

⁸ <https://axios-http.com/ptbr/docs/intro>

⁹ <https://docs.expo.dev/get-started/expo-go/>

desenvolvimento, e é provável que você volte ao Expo Go ao longo de sua jornada quando quiser experimentar novas ideias em um novo sandbox de projeto rápido. É um ótimo lugar para começar!

2.4 Metodologia Ágil

"As metodologias ágeis para desenvolvimento de software são uma resposta às chamadas metodologias pesadas ou tradicionais. Mesmo com a evolução dos computadores, das técnicas e ferramentas nos últimos anos, a produção de software confiável, correto e entregue dentro dos prazos e custos estipulados ainda é muito difícil. Aproximadamente 31% dos projetos foram cancelados antes de estarem completos e 52,7% foram entregues, porém com prazos maiores, custos maiores ou com menos funcionalidades do que especificado no início do projeto. Dentre os projetos que não foram finalizados de acordo com os prazos e custos especificados, a média de atrasos foi de 222%, e a média de custo foi de 189% a mais do que o previsto. Considerando todos os projetos que foram entregues além do prazo e com custo maior, na média, apenas 61% das funcionalidades originais foram incluídas. Mesmo os projetos cuja entrega é feita respeitando os limites de prazo e custo possuem qualidade suspeita, uma vez que provavelmente foram feitos com muita pressão sobre os desenvolvedores, o que pode quadruplicar o número de erros de software, segundo a mesma pesquisa. As principais razões destas falhas estavam relacionadas com o processo em Cascata. A recomendação final foi que o desenvolvimento de software deveria ser baseado em modelos incrementais, o que poderia evitar muitas das falhas reportadas."(DOS SANTOS SOARES, 2004.)

2.5 LGPD (Lei Geral de Proteção de Dados Pessoais)

A internet tem realizado uma verdadeira revolução no cotidiano das pessoas. Boa parte disso ocorre devido ao aumento da conectividade entre os indivíduos (Oulasvirta et al. 2012). Conforme os dados do último censo realizado pelo IBGE, existem hoje cerca de 1,26 telefones celulares para cada habitante (IBGE 2017) Segundo o site (World IPv6 Launch 2018), o alicerce para este crescimento pode ser justificado pela Internet das Coisas (IOT).

Segundo (Diniz, 2006) Neste contexto aparecem tecnologias voltadas para IoT, onde a ideia por trás deste conceito teve origem em novas possibilidades originadas através de dispositivos inteligentes conectados a internet, que possibilitaram desta forma, a comunicação a partir de qualquer lugar a qualquer momento por um dispositivo. Como pode-se observar, são grandes os desafios do tema proposto devido a complexibilidade de aplicar requisitos de segurança a objetos com processamento, memória e largura de banda.

2.6 Prototipação

Segundo (Sommerville e Sawyer, 2003), um protótipo pode ser usado como meio de comunicação entre os diversos membros da equipe de desenvolvimento ou mesmo como meio de testar idéias. Quanto mais iterativo for o processo de desenvolvimento do protótipo, melhor será o sistema final.

2.7 Scrum

Scrum é uma metodologia ágil de gerenciamento de projetos e desenvolvimento de software, entre outros tipos de trabalho em equipe. Ele se concentra na entrega contínua de resultados para o cliente e nas adaptações às mudanças nas necessidades do projeto ao longo do tempo.

“Seu objetivo é fornecer um processo conveniente para projeto e desenvolvimento orientado a objeto. A Scrum apresenta uma abordagem empírica que aplica algumas idéias da teoria de controle de processos industriais para o desenvolvimento de softwares, reintroduzindo as idéias de flexibilidade, adaptabilidade e produtividade. O foco da metodologia é encontrar uma forma de trabalho dos membros da equipe para produzir o software de forma flexível e em um ambiente em constante mudança.

A ideia principal da Scrum é que o desenvolvimento de softwares envolve muitas variáveis técnicas e do ambiente, como requisitos, recursos e tecnologia, que podem mudar durante o processo. Isto torna o processo de desenvolvimento imprevisível e complexo, requerendo flexibilidade para acompanhar as mudanças. O resultado do processo deve ser um software que é realmente útil para o cliente.”(DOS SANTOS SOARES, 2004.)

2.8 Docker

Docker foi inicialmente apresentado ao mundo por Solomon Hykes, fundador e CEO da empresa dotCloud, em uma palestra relâmpago de cinco minutos na Python Developers Conference realizada em Santa Clara, na Califórnia, em 15 de março de 2013. Em poucas semanas após o anúncio, houve uma quantidade impressionante de notícias sobre o docker. O projeto foi rapidamente transformado em código aberto e disponibilizado publicamente, onde qualquer um poderia fazer o download e contribuir para o mesmo. (DA SILVA, 2016, p10).

O Docker é uma ferramenta que promete encapsular facilmente o processo de criação de um artefato distribuído para qualquer aplicação em escala em qualquer ambiente e racionalizar o fluxo de trabalho e a responsabilidade das organizações de software ágil. Inicialmente ele foi inventado para que a dotCloud pudesse suportar de forma mais simples a gerência de suas plataformas de serviços, onde desenvolvedores poderiam fazer deploy (lançamento) de suas aplicações, em vez de máquinas virtuais com grande volume de espaço, tudo rodaria em contêineres Linux. (TURNBULL, 2014, p 78).

3. DESENVOLVIMENTO

3.1 Banco de Dados

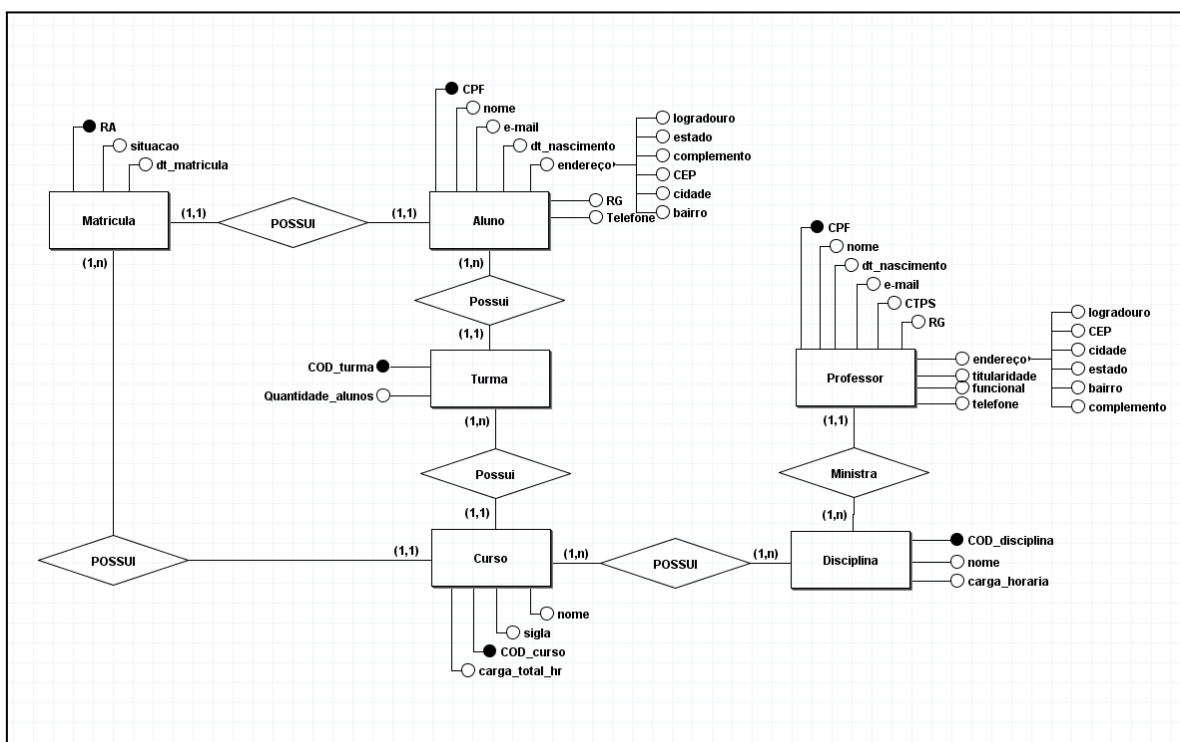
Nossa aplicação terá uma conexão com banco de dados relacional para o armazenamento dos dados dos usuários do sistema, e a ferramenta para o gerenciamento do banco de dados que iremos utilizar será o SQL Server, atualmente desenvolvido pela Microsoft e sua licença é gratuita. Após uma discussão com os envolvidos no projeto sobre qual gerenciador de banco de dados usar, foi visto que o SQL Server era a melhor escolha para o projeto, pois é o que se enquadra às necessidades do projeto.

Para a integração do back end com o banco de dados usaremos um pacote prisma onde conseguimos criar todo o esquema do banco de dados e ter acesso a interface gráfica do usuário para visualizar e editar dados contidos no banco de dados.

3.1.1 Diagrama ER

Na Figura 01 se apresenta o diagrama de entidade e relacionamento feito com base no projeto com todas as suas entidades, relacionamentos, atributos e cardinalidades.

Figura 01- Diagrama ER



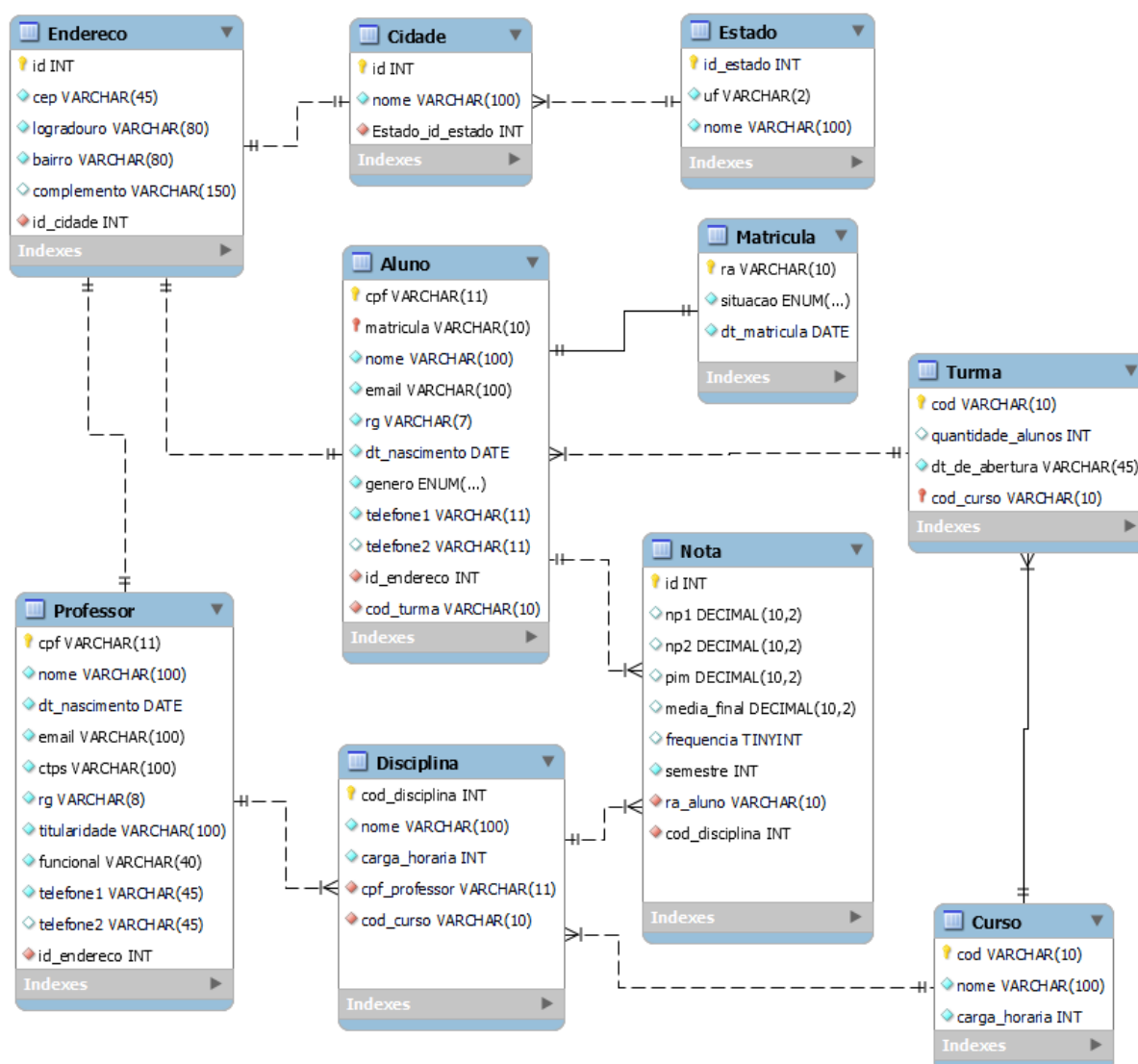
Fonte: Dos autores

3.1.2 Modelo Lógico

No modelo lógico visto na **Figura 2**, é introduzida as tabelas (entidades) e os seus atributos. É possível observar que cada atributo tem seu tipo de dados, como VARCHAR que armazena caracteres e INT que armazena números inteiros.

O modelo lógico estabelece a estrutura do banco de dados, podendo facilitar a visualização e a criação do mesmo.

Figura 2 - Modelo Lógico do banco de dados.



Fonte: Dos Autores.

3.1.3 Dicionário de Dados

O dicionário de dados documenta os dados e seus tipos que serão armazenados no banco de dado, como é visto a seguir nas tabelas de 01 a 10, e detalha as tabelas que foram apresentadas anteriormente no modelo lógico do banco de dados (**Figura 2**), as tabelas possuem os seguintes dados: Nome da Coluna, Tipo de Dados, Tamanho, Restrições e Descrição.

Tabela 1 - Dicionário de dados da tabela Aluno

Nome da tabela: Aluno				
Definição: Armazena dados dos alunos				
Nome da Coluna	Tipo de Dados	Tamanho	Restrições	Descrição
cpf	VARCHAR	11	PK, NOT NULL	
nome	VARCHAR	100	NOT NULL	
email	VARCHAR	100	NOT NULL	
rg	VARCHAR	7	NOT NULL	
dt_nascimento	DATE		NOT NULL	
genero	ENUM		NOT NULL	'Masculino', 'Feminino'
telefone1	VARCHAR	11	NOT NULL	
telefone2	VARCHAR	11		
id_endereco	INT		FK, NOT NULL	Chave estrangeira da tabela Endereço.
cod_turma	INT		FK, NOT NULL	Chave estrangeira da tabela Turma.
matricula	VARCHAR	10	PK, FK, NOT NULL	Chave estrangeira da tabela Matricula.
id_endereco	INT		FK, NOT NULL	Chave estrangeira da tabela Endereco
cod_turma	VARCHAR	10	FK, NOT NULL	Chave estrangeira da tabela Turma

Fonte: Dos Autores.

Tabela 2 - Dicionário de dados da tabela Matricula

Nome da tabela: Matricula				
Definição: Armazena dados da matrícula do aluno				
Nome da Coluna	Tipo de Dados	Tamanho	Restrições	Descrição
ra	VARCHAR	10	PK, NOT NULL	
situação	ENUM		NOT NULL	'Ativa', 'Inativa'.
dt_matricula	DATE		NOT NULL	Data da criação.

Fonte: Dos Autores.

Tabela 3 - Dicionário de dados da tabela Nota

Nome da tabela: Nota				
Definição: Armazena notas dos alunos				
Nome da Coluna	Tipo de Dados	Tamanho	Restrições	Descrição
id	INT		PK, NOT NULL, AI	
np1	DECIMAL	10,2		
np2	DECIMAL	10,2		
pim	DECIMAL	10,2		
media_final	DECIMAL	10,2		
frequencia	TINYINT			
semestre	INT			
ra_aluno	VARCHAR	10	FK, NOT NULL	Chave estrangeira da tabela Matricula.
cod_disciplina	INT		FK, NOT NULL	Chave estrangeira da tabela Disciplina.

Fonte: Dos Autores.

Tabela 4 - Dicionário de dados da tabela Turma.

Nome da tabela: Turma				
Definição: Armazena dados das turmas				
Nome da Coluna	Tipo de Dados	Tamanho	Restrições	Descrição
cod_turma	VARCHAR	10	PK, NOT NULL	
quantidade_alunos	INT			
dt_de_abertura	DATE		NOT NULL	Data da abertura da turma.
cod_curso	VARCHAR	10	FK, NOT NULL	Chave estrangeira da tabela Curso

Fonte: Dos Autores.

Tabela 5 - Dicionário de dados da tabela Curso

Nome da tabela: Curso				
Definição: Armazena dados dos cursos				
Nome da Coluna	Tipo de Dados	Tamanho	Restrições	Descrição
cod	VARCHAR	10	PK, NOT NULL	
nome	VARCHAR	100	NOT NULL	
carga_horaria	INT		NOT NULL	

Fonte: Dos Autores.

Tabela 6 - Dicionário de dados da tabela Disciplina

Nome da tabela: Disciplina				
Definição: Armazena dados das disciplinas				
Nome da Coluna	Tipo de Dados	Tamanho	Restrições	Descrição
cod_disciplina	INT		PK, NOT NULL, IA	Auto incremento
nome	VARCHAR	100	NOT NULL	
carga_horaria	INT		NOT NULL	
cpf_professor	VARCHAR	11	FK, NOT NULL	Chave estrangeira da tabela Professor
cod_curso	VARCHAR	10	NOT NULL	Chave estrangeira da tabela Curso

Fonte: Dos Autores.

Tabela 7 - Dicionário de dados da tabela Professor

Nome da tabela: Professor				
Definição: Armazena dados de professores				
Nome da Coluna	Tipo de Dados	Tamanho	Restrições	Descrição
cpf	VARCHAR	11	PK, NOT NULL	
nome	VARCHAR	100	NOT NULL	
dt_nascimento	DATE		NOT NULL	
email	VARCHAR	100	NOT NULL	
ctps	VARCHAR	100	NOT NULL	
rg	VARCHAR	8	NOT NULL	
titularidade	VARCHAR	100	NOT NULL	
funcional	VARCHAR	40	NOT NULL	
telefone1	VARCHAR	45	NOT NULL	
telefone2	VARCHAR	45		
id_endereco	INT		FK, NOT NULL	Chave estrangeira da tabela Endereco

Fonte: Dos Autores.

Tabela 8 - Dicionário de dados da tabela Endereco

Nome da tabela: Endereco				
Definição: Armazena dados de endereço dos usuários				
Nome da Coluna	Tipo de Dados	Tamanho	Restrições	Descrição
id	INT		PK, NOT NULL, AI	
cep	VARCHAR	45	NOT NULL	
logradouro	VARCHAR	80	NOT NULL	
bairro	VARCHAR	80	NOT NULL	
complemento	VARCHAR	150	NOT NULL	
cidade_id	INT		NOT NULL	Chave estrangeira da tabela Cidade

Fonte: Dos Autores.

Tabela 9 - Dicionário de dados da tabela Cidade

Nome da tabela: Cidade				
Definição: Armazena dados de Cidades				
Nome da Coluna	Tipo de Dados	Tamanho	Restrições	Descrição
id	INT		PK, NOT NULL, AI	
nome	VARCHAR	100	NOT NULL	
uf_estado	VARCHAR	2	FK, NOT NULL	Chave estrangeira da tabela Estado

Fonte: Dos Autores.

Tabela 10 - Dicionário de dados da tabela Estado

Nome da tabela: Estado				
Definição: Armazena dados de Estados do Brasil				
Nome da Coluna	Tipo de Dados	Tamanho	Restrições	Descrição
uf	VARCHAR	2	PK, NOT NULL	
nome	VARCHAR	100	NOT NULL	

Fonte: Dos Autores.

3.1.4 Docker

Utilizamos o Docker basicamente para alocar o Banco de Dados SQL Server, o Docker permite que seja mais fácil a transferência dos arquivos do banco entre as máquinas da equipe de desenvolvedores, ele também é mais seguro porque não tem uma interferência direta do sistema operacional.

3.1.5 Criação do banco de dados

Para a criação do banco de dados o prisma foi essencial, pois com ele foi criado apenas um esquema de dados e o resto do processo de implementar todo aquele esquema no banco de dados foi automatizado pelo prisma, onde se gerou um relatório dos comandos sql feitos por ele, como podem ser vistos no apêndice C.

3.2 Casos de Uso

Durante a elicitação de requisitos e funcionalidades os casos de uso são de grande importância no auxílio da modelagem de software. Em relação aos casos de uso utilizaremos o diagrama de casos de uso um dos principais da UML e o detalhamento de casos de uso em forma de tabelas. Foram levantados e numerados os casos de uso conforme as funcionalidades do sistema como pode ser visto na Tabela 1.

Tabela 11 - Casos de uso

Nº	Nome	Ator	Descrição
UC01	Efetuar Login	Aluno, Professor, Secretaria	Permitir que o usuário acesse o sistema com suas credenciais.
UC02	Efetuar Logout	Aluno, Professor, Secretaria	Permitir que o usuário se desconecte do sistema.
UC03	Manter Aluno	Secretaria	Permitir que o ator cadastre, consulte, edite e exclua um determinado aluno.
UC04	Manter Professor	RH	Permitir que o ator cadastre, consulte, edite e exclua um determinado professor.
UC05	Manter Curso	Secretaria	Permitir que o ator cadastre, consulte, edite

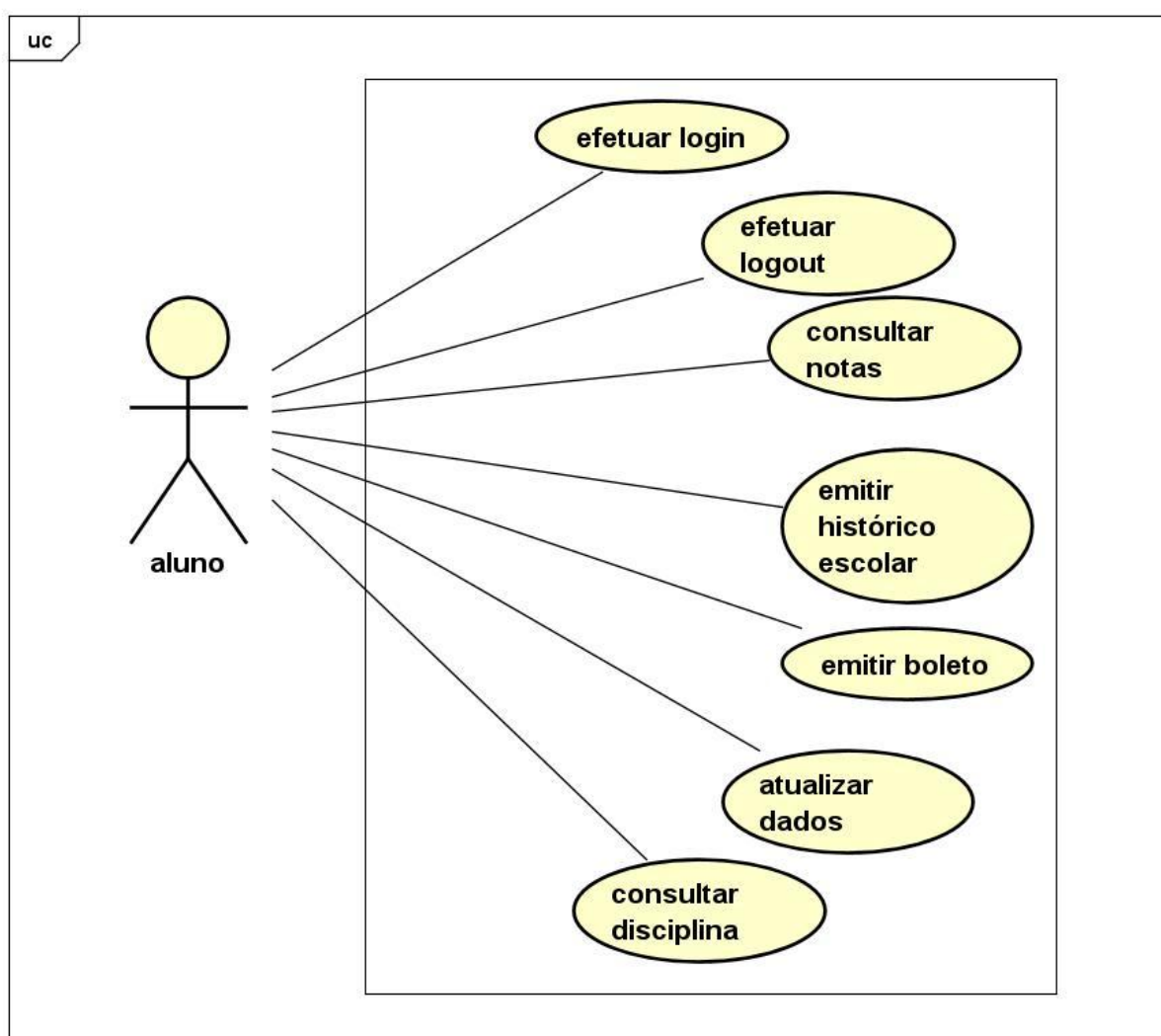
			e exclua um determinado curso.
UC06	Manter Disciplina	Secretaria	Permitir que o ator cadastre, consulte, edite e exclua uma determinada disciplina.
UC07	Manter Turma	Secretaria	Permitir que o ator cadastre, consulte, edite e exclua uma determinada turma.
UC08	Manter Notas	Professor, Secretaria	Permite que o ator lance e altere notas de um determinado aluno.
UC09	Manter Frequência	Professor, Secretaria	Permite que o ator lance faltas de um determinado aluno.
UC10	Manter Conteúdo Programático	Professor.	Permite que o ator lance e altere conteúdos programáticos no sistema
UC11	Consultar Notas	Aluno.	Permite que o ator visualize notas e médias de um aluno.
UC12	Emitir Mapa de Notas	Professor	Permite que o ator visualize o mapa de notas dos alunos.
UC13	Emitir Histórico escolar	Secretaria, Aluno	Permite que o ator emita o histórico escolar de um aluno.
UC14	Efetuar Matrícula	Secretaria	Permite que o ator efetue a matrícula de um aluno.
UC15	Atualizar Dados	Professor, Aluno	Permite que o ator altere alguns dados pessoais.
UC16	Emitir Boleto	Aluno	Permite que o ator emita o boleto de mensalidades.
UC17	Consultar Disciplina	Aluno	Permite que o ator consulte as disciplinas de sua grade curricular.
UC18	Emitir Relatório de matrículas	Secretaria	Permite que o ator emita relatórios de matrículas.

Fonte: Dos Autores.

3.2.1 Diagramas de Casos de Uso

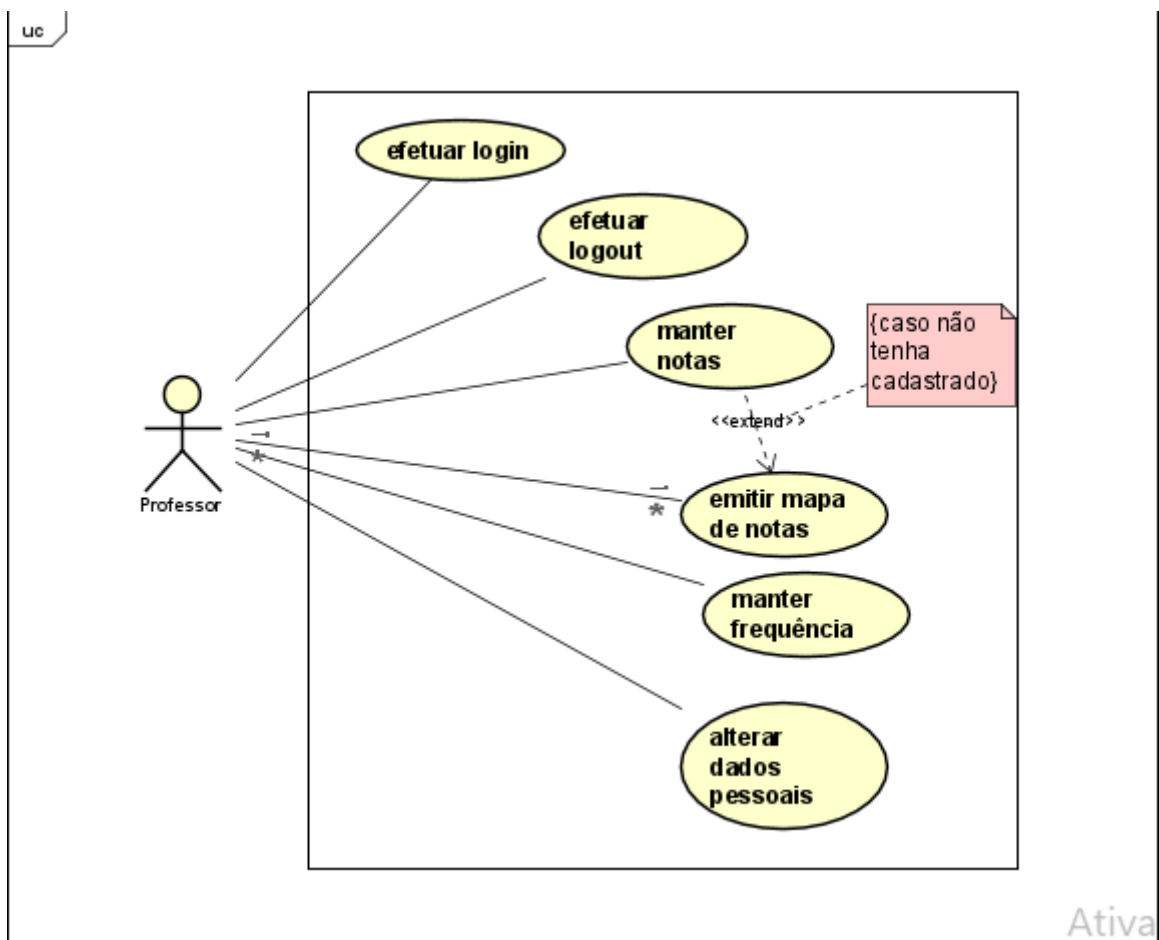
A seguir serão apresentados os diagramas de caso de uso da figura 03 até 05 feitos com base nas necessidades do projeto. Sendo eles o diagrama de aluno, professor e secretária e RH, eles descrevem as ações de cada ator e assim ajuda também na construção do diagrama de classes.

Figura 03- Diagrama de caso de uso do aluno



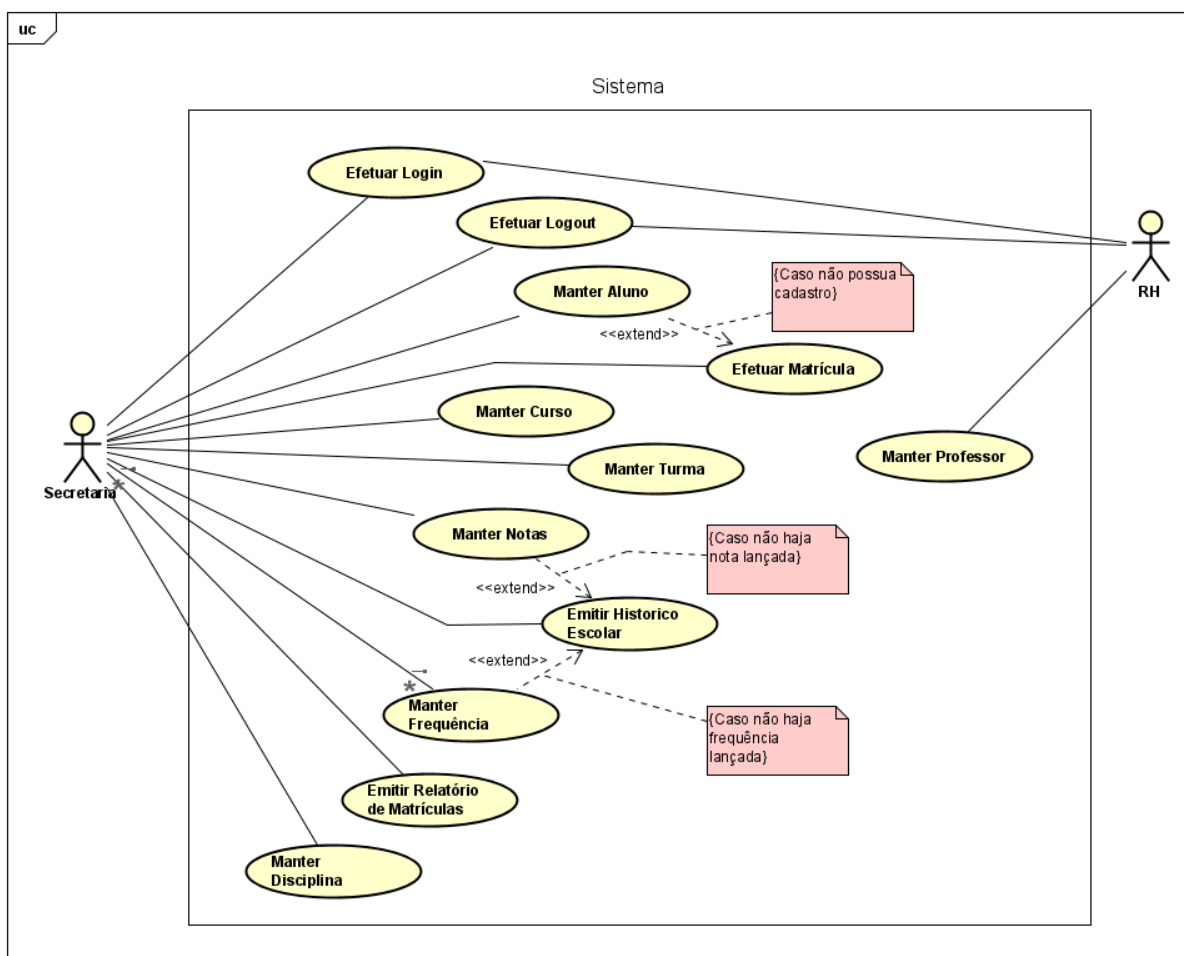
Fonte: Dos Autores

Figura 04- Diagrama de caso de uso do professor



Fonte: Dos Autores

Figura 05- Diagrama de caso de uso da Secretaria e RH



Fonte: Dos Autores

3.2.2 Detalhamento de Casos de Uso

Neste tópico será apresentado o detalhamento de caso de uso de cada funcionalidade solicitada para o sistema. Esse detalhamento é de grande importância pois documenta cada passo necessário para executar uma funcionalidade, detalhando seus atores, sendo eles principais e ou secundários, suas pré-condições e pós-condições caso possuam e seus fluxos com sua lista de ações. Veja esses detalhamentos nas tabelas a seguir.

Tabela 12 - Detalhamento de Caso de Uso “Efetuar Login”

UC01 - Efetuar Login	
Ator principal	Usuário
Atores secundários	Nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que um usuário acesse o sistema com suas credenciais realizando o caso de uso “Efetuar Login”.
Pré-condições	É necessário possuir credenciais pré-cadastradas no sistema.
Pós-condições	
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita a realização de login	
	2. O sistema apresenta a interface
3. O usuário informa as credenciais	
	4. O sistema valida a existência e tipo de credenciais inseridas, apresenta a interface correspondente às credenciais e o caso de uso termina.
Restrições/Validações	1. As credenciais precisam ser do tipo professor, aluno ou secretaria
Fluxo Alternativo I - Recuperar Senha	
Ações do ator	Ações do sistema
1. O usuário solicita a realização de login	
	2. O sistema apresenta a interface
3. O usuário informa as credenciais	
	4. O sistema invalida as senhas e apresenta a mensagem “Senha Incorreta”..
5. O usuário aciona a recuperação de senha	
	6. O sistema apresenta a interface

UC01 - Efetuar Login	
7. O usuário informa o email	
	8. O sistema valida e envia um código de recuperação de senha e solicita o código
9. O usuário inserir o código	
	9. O sistema valida, grava a nova senha, registrar a alteração e termina o caso de uso
Restrições/Validações	1. O email de recuperação precisa ser o mesmo cadastrado

Fonte: Dos Autores.

Tabela 13 - Detalhamento de Caso de Uso “Efetuar Logout”

C02 - Efetuar Logout	
Ator principal	Usuário
Atores secundários	Nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que um usuário desconecte do sistema efetuando o caso de uso “Efetuar Logout”.
Pré-condições	É necessário estar logado no sistema
Pós-condições	O usuário é desconectado do sistema, retornando para a área inicial de login.
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita a realização de logout	
	2. O sistema apresenta a mensagem de confirmação
3. O usuário confirma	
	4. O sistema desconecta o usuário
Restrições/Validações	Nenhuma.

Fonte: Dos Autores.

Tabela 14 - Detalhamento de Caso de Uso “Manter Aluno”

UC03 - Manter Aluno	
Ator principal	Secretaria
Atores secundários	Nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que a secretaria realize o caso de uso “Manter Aluno” que contém as funcionalidades: Cadastrar Aluno, Editar Aluno, Consultar Alunos e Excluir aluno.
Pré-condições	É preciso preencher todos os campos obrigatórios.
Pós-condições	Nenhuma.
Fluxo Principal	
Ações do ator	Ações do sistema
1 usuário solicita a inclusão de novo aluno	
	2.o sistema apresenta a interface de cadastro
3. O usuário informa os dados	
	4 o sistema valida e armazena os dados e o caso de uso termina.
Restrições/Validações	1. O aluno não pode possuir cadastro.
	2. É necessário ter completado o ensino médio.
Fluxo Alternativo I - Alterar aluno	
Ações do ator	Ações do sistema
1.O usuário solicita a alteração de um aluno	
	2. O sistema apresenta a interface
3. O usuário informa os novos dados	
	4. O sistema valida e armazena os dados e o caso de uso termina
Fluxo Alternativo I - Consultar Alunos	
Ações do ator	Ações do sistema
1. O usuário solicita a busca de alunos	
	2. O sistema apresenta a interface
3. O usuário informa os dados para busca	

UC03 - Manter Aluno	
	4. O sistema valida e imprime os dados e o caso de uso termina
Fluxo Alternativo I - Excluir Aluno	
Ações do ator	Ações do sistema
1. O usuário solicita a exclusão de um aluno	
	2. O sistema apresenta a interface
3. O usuário informa os dados de busca	
	4. O sistema valida e busca os dados e exclui os dados e o caso de uso termina

Fonte: Dos Autores.

Tabela 15 - Detalhamento de Caso de Uso “Manter Professor”

UC04 - Manter Professor	
Ator principal	Secretaria
Atores secundários	nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que a secretaria realize o caso de uso “Manter Professor” que contém as funcionalidades: Cadastrar, Editar, Consultar e Excluir professores.
Pré-condições	É preciso preencher todos os campos obrigatórios como o campo “Funcional” entre outros.
Pós-condições	
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita a inclusão de um novo professor	
	2. O sistema apresenta a interface
3. O usuário informa os dados	
	4. O sistema valida e armazena os dados e o

UC04 - Manter Professor	
	caso de uso termina
Restrições/Validações	
Fluxo Alternativo I - Alterar professor	
Ações do ator	Ações do sistema
1. O usuário solicita a alteração de um professor	
	2. O sistema apresenta a interface
3. O usuário informa novos os dados	
	4. O sistema valida e armazena os dados e o caso de uso termina
Fluxo Alternativo I - Consultar professores	
Ações do ator	Ações do sistema
1. O usuário solicita a consulta de professores	
	2. O sistema apresenta a interface
3. O usuário informa os dados para busca	
	4. O sistema valida e imprime os dados e o caso de uso termina
Fluxo Alternativo I - Excluir professor	
Ações do ator	Ações do sistema
1. O usuário solicita a exclusão de um professor	
	2. O sistema apresenta a interface
3. O usuário informa os dados	
	4. O sistema valida e busca os dados e exclui os dados e o caso de uso termina

Fonte: Dos Autores.

Tabela 16 - Detalhamento de Caso de Uso "Manter Curso"

UC05 - Manter Curso	
Ator principal	Secretaria
Atores secundários	Nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que a secretaria realize o caso de uso "Manter Curso" que contém as funcionalidades: Cadastrar, Editar, Consultar e Excluir Curso.
Pré-condições	
Pós-condições	
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita a inclusão de um novo curso	
	2. O sistema apresenta a interface
3. O usuário informa os dados	
	4. O sistema valida e grava os dados e o caso de uso termina
Restrições/Validações	
Fluxo Alternativo I - Alterar Curso	
Ações do ator	Ações do sistema
1. O usuário solicita a alteração de um professor	
	2. O sistema apresenta a interface
3. O usuário informa os novos dados	
	4. O sistema valida e armazena os dados e o caso de uso termina
Fluxo Alternativo I - Consultar Curso	
Ações do ator	Ações do sistema
1. O usuário solicita a consulta de cursos	
	2. O sistema apresenta a interface
3. O usuário informa os dados	

UC05 - Manter Curso	
	4. O sistema busca e imprime os dados e o caso de uso termina
Fluxo Alternativo I - Excluir Curso	
Ações do ator	Ações do sistema
1. O usuário solicita a exclusão de um curso	
	2. O sistema apresenta a interface
3. O usuário informa o dado de busca	
	3. O sistema valida e exclui os dados e o caso de uso termina

Fonte: Dos Autores.

Tabela 17 - Detalhamento de Caso de Uso “Manter Disciplina”

UC06 - Manter Disciplina	
Ator principal	Secretaria
Atores secundários	Nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que a secretaria realize o caso de uso “Manter Disciplina” que contém as funcionalidades: Cadastrar, Editar, Consultar e Excluir Disciplinas.
Pré-condições	
Pós-condições	
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita a inclusão de uma nova disciplina	
	2. O sistema apresenta a interface
3. O usuário informa os dados	
	4. O sistema valida e grava os dados e o caso de uso termina
Restrições/Validações	

UC06 - Manter Disciplina	
Fluxo Alternativo I - Alterar Curso	
Ações do ator	Ações do sistema
1. O usuário solicita a alteração de uma disciplina	
	2. O sistema apresenta a interface
3. O usuário informa os novos dados	
	4. O sistema valida e armazena os dados e o caso de uso termina
Fluxo Alternativo I - Consultar Disciplina	
Ações do ator	Ações do sistema
1. O usuário solicita a consulta de disciplinas	
	2. O sistema apresenta a interface
3. O usuário informa os dados	
	4. O sistema busca e imprime os dados e o caso de uso termina
Fluxo Alternativo I - Excluir Disciplina	
Ações do ator	Ações do sistema
1. O usuário solicita a exclusão de uma disciplina	
	2. O sistema apresenta a interface
3. O usuário informa o dado de busca	
	3. O sistema valida e exclui os dados e o caso de uso termina

Fonte: Dos Autores.

Tabela 18 - Detalhamento de Caso de Uso “Manter Turma”

UC07 - Manter Turma	
Ator principal	Secretaria
Atores secundários	Nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que a secretaria realize o caso de uso “Manter Turma” que contém as

UC07 - Manter Turma	
	funcionalidades: Cadastrar, Editar, Consultar e Excluir Disciplinas.
Pré-condições	
Pós-condições	
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita a inclusão de uma nova turma	
	2. O sistema apresenta a interface
3. O usuário informa os dados	
	4. O sistema valida e grava os dados e o caso de uso termina
Restrições/Validações	
Fluxo Alternativo I - Alterar Turma	
Ações do ator	Ações do sistema
1. O usuário solicita a alteração de uma turma	
	2. O sistema apresenta a interface
3. O usuário informa os novos dados	
	4. O sistema valida e armazena os dados e o caso de uso termina
Fluxo Alternativo I - Consultar Turma	
Ações do ator	Ações do sistema
1. O usuário solicita a consulta de turmas	
	2. O sistema apresenta a interface
3. O usuário informa os dados	
	4. O sistema busca e imprime os dados e o caso de uso termina
Fluxo Alternativo I - Excluir Disciplina	
Ações do ator	Ações do sistema

UC07 - Manter Turma	
1. O usuário solicita a exclusão de uma turma	
	2. O sistema apresenta a interface
3. O usuário informa o dado de busca	
	3. O sistema valida e exclui os dados e o caso de uso termina

Fonte: Dos Autores.

Tabela 19 - Detalhamento de Caso de Uso “Manter Notas”

UC08 - Manter Notas	
Ator principal	Professor
Atores secundários	Secretaria
Resumo	Este caso de uso descreve as etapas necessárias para que o sistema realize o caso de uso “Lançar Nota” que permite o armazenamento e alteração de notas no sistema.
Pré-condições	
Pós-condições	
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita lançar notas	
	2. O sistema apresenta a interface
3. O usuário informa as notas	
	4. O sistema armazena os dados e o caso de uso termina
Restrições/Validações	
Fluxo Alternativo I - Alterar Nota	
1. O usuário solicita a alteração de nota	
	2. O sistema apresenta a interface
3. O usuário informa as novas notas ou exclui.	
	4. O sistema armazena ou deleta os dados e o caso de uso termina

Fonte: Dos Autores.

Tabela 20 - Detalhamento de Caso de Uso "Manter Frequência"

UC09 - Manter Frequência	
Ator principal	Professor
Atores secundários	Secretaria
Resumo	Este caso de uso descreve as etapas necessárias para que o sistema realize o caso de uso "Lançar Nota" que permite o armazenamento e alteração de frequência de alunos no sistema.
Pré-condições	Nenhuma.
Pós-condições	Nenhuma.
Fluxo Principal	
Ações do ator	Ações do sistema
1. O solicita lançar faltas	
	2. O sistema apresenta a interface
3. O usuário informa o aluno	
	4. O sistema busca o aluno
5. O usuário informa os dados	
	6. O sistema armazena os dados e o caso de uso termina
Restrições/Validações	Nenhuma.
Fluxo Alternativo I - Alterar Frequência	
1. O usuário solicita a alteração de frequência	
	2. O sistema apresenta a interface
3. O usuário informa o aluno	
	4. O sistema busca o aluno
5. O usuário informa os novos dados ou os exclui	
	6. O sistema armazena os dados ou os exclui e o caso de uso termina
Restrições/Validações	Nenhuma.

Fonte: Dos Autores.

Tabela 21 - Detalhamento de Caso de Uso “Manter Conteúdo Programático”

UC10 - Manter Conteúdo Programático	
Ator principal	Professor
Atores secundários	Nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que um professor lance ou exclua conteúdos programáticos, efetuando o caso de uso “Manter Conteúdo Programático”
Pré-condições	O tipo de arquivo anexado deverá ser compatível com os arquivos aceitos pelo sistema.
Pós-condições	Os arquivos ficam visíveis para os alunos.
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita lançar conteúdo(s).	
	2. O sistema apresenta a interface.
3. O usuário anexa o conteúdo.	
	4. O sistema valida e armazena o conteúdo no sistema e o caso de uso termina.
Restrições/Validações	1. Somente o professor pode lançar conteúdos programáticos.
Fluxo Alternativo I - Alterar Conteúdo Programático	
Ações do ator	Ações do sistema
1. O usuário solicita alterar um conteúdo.	
	2. O sistema apresenta a interface.
3. O usuário altera ou exclui o conteúdo.	
	4. O sistema valida e armazena os dados ou os exclui e o caso de uso termina.
Restrições/Validações	1. Somente o professor pode alterar conteúdos programáticos.

Fonte: Dos Autores.

Tabela 22 - Detalhamento de Caso de Uso "Consultar Nota"

UC11 - Consultar Nota	
Ator principal	Aluno
Atores secundários	Nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que um aluno visualize suas notas e médias, efetuando o caso de uso "Consultar Nota"
Pré-condições	Somente é possível consultar notas caso haja notas previamente lançadas no sistema pelos professores ou secretaria.
Pós-condições	Nenhuma.
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita consultar suas notas.	
	2. O sistema apresenta a interface com as notas e médias do aluno e o caso de uso termina.
Restrições/Validações	1. Somente é possível consultar notas caso haja notas previamente lançadas no sistema pelos professores ou secretaria.
Fluxo Alternativo I - Consultar Média	
Ações do ator	Ações do sistema
1. O usuário solicita consultar sua média.	
	2. O sistema calcula a média e apresenta as médias do aluno e o caso de uso termina.
Restrições/Validações	Nenhuma.

Fonte: Dos Autores.

Tabela 23 - Detalhamento de Caso de Uso "Emitir Mapa de Notas"

UC12 - Emitir Mapa de Notas	
Ator principal	Professor
Atores secundários	Nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que um professor emita o mapa de notas dos alunos, efetuando o caso de

UC12 - Emitir Mapa de Notas	
	uso "Emitir Mapa de Notas"
Pré-condições	É necessário haver notas lançadas.
Pós-condições	O usuário poderá imprimir o mapa de notas.
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita emitir o mapa de notas.	
	2. O sistema apresenta a interface.
3. O usuário informa os dados	
	4. O sistema valida e emite o mapa de notas e o caso de uso termina.
Restrições/Validações	Nenhuma.

Fonte: Dos Autores.

Tabela 24 - Detalhamento de Caso de Uso "Emitir Histórico Escolar"

UC13 - Emitir Histórico Escolar	
Ator principal	Secretaria
Atores secundários	Aluno
Resumo	Este caso de uso descreve as etapas necessárias para que o usuário emita o histórico escolar, efetuando o caso de uso "Emitir Histórico Escolar"
Pré-condições	É preciso ter notas e frequência lançadas.
Pós-condições	O usuário poderá imprimir o histórico escolar.
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita a emissão do histórico escolar.	
	2. O sistema apresenta a interface.
3. Caso o usuário seja a secretaria ela informa os dados, caso seja o aluno essa ação é ignorada.	
	4. O sistema valida e emite o histórico escolar e

UC13 - Emitir Histórico Escolar	
	o caso de uso termina.
Restrições/Validações	1. Somente a secretaria pode realizar a busca do aluno que deseja emitir o histórico.

Fonte: Dos Autores.

Tabela 25 - Detalhamento de Caso de Uso "Efetuar Matrícula".

UC14 - Efetuar Matrícula	
Ator principal	Secretaria
Atores secundários	nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que o usuário efetue a matrícula de um aluno, efetuando o caso de uso "Efetuar Matrícula"
Pré-condições	É necessário que o aluno possua cadastro.
Pós-condições	Nenhuma
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita efetuar matrícula.	
	2. O sistema apresenta a interface.
3. O usuário informa os dados	
	4. O sistema valida e caso o aluno não possua cadastro o caso de uso "Manter Aluno" é solicitado. Caso contrário o gera a matrícula e armazena os dados e o caso de uso termina.
Restrições/Validações	1. Somente a secretaria pode efetuar matrículas.

Fonte: Dos Autores.

Tabela 26 - Detalhamento de Caso de Uso "Atualizar Dados".

UC15 - Atualizar Dados	
Ator principal	Professor
Atores secundários	Aluno
Resumo	Este caso de uso descreve as etapas

UC15 - Atualizar Dados	
	necessárias para que o usuário atualize alguns dados cadastrais, efetuando o caso de uso "Atualizar Dados"
Pré-condições	
Pós-condições	
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita a atualização de dados.	
	2. O sistema apresenta a interface.
3. O usuário informa os novos dados	
	4. O sistema valida os dados, registra a alteração e armazena os dados e o caso de uso termina.
Restrições/Validações	Nenhuma

Fonte: Dos Autores.

Tabela 27 - Detalhamento de Caso de Uso "Emitir Boleto".

UC16 - Emitir Boleto	
Ator principal	Aluno
Atores secundários	nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que o usuário emita o boleto de mensalidade, efetuando o caso de uso "Atualizar Dados"
Pré-condições	É necessário estar matriculado.
Pós-condições	O usuário poderá imprimir o boleto
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita a emissão de boleto.	
	2. O sistema apresenta a interface com o atual boleto e histórico de boletos passados e atrasados e o caso de uso termina.
Restrições/Validações	1. O sistema apenas apresentará boletos caso o

UC16 - Emitir Boleto	
	aluno não possua bolsa integral.

Fonte: Dos Autores.

Tabela 28 - Detalhamento de Caso de Uso "Consultar disciplina".

UC17 - Consultar Disciplina	
Ator principal	Aluno
Atores secundários	Nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que o usuário consulte suas disciplinas semestrais e grade curricular , efetuando o caso de uso "Consultar Disciplina"
Pré-condições	É necessário estar matriculado.
Pós-condições	nenhuma.
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita realizar a consulta de disciplinas.	
	2. O sistema apresenta as disciplinas do aluno e o caso de uso termina.
Restrições/Validações	
Fluxo alternativo I - Consultar Grade Curricular	
Ações do ator	Ações do sistema
1. O usuário solicita realizar a consulta da grade curricular completa.	
	2. O sistema apresenta a grade curricular do curso para o aluno e o caso de uso termina.
Restrições/Validações	Nenhuma.

Fonte: Dos Autores.

Tabela 29 - Detalhamento de Caso de Uso "Emitir Relatório de Matrículas".

UC18 - Emitir Relatório de Matrículas	
Ator principal	Secretaria
Atores secundários	nenhum
Resumo	Este caso de uso descreve as etapas necessárias para que o usuário emita o relatório de matrículas, efetuando o caso de uso "Emitir Relatório de Matrículas"
Pré-condições	É necessário selecionar um filtro ou tipo de relatório.
Pós-condições	É possível imprimir o relatório.
Fluxo Principal	
Ações do ator	Ações do sistema
1. O usuário solicita a emissão do relatório de matrículas.	
	2. O sistema apresenta a interface.
3. O usuário filtra o que deseja no relatório.	
	4. O sistema emite o relatório e o caso de uso termina.
Restrições/Validações	Nenhuma.

Fonte: Dos Autores.

3.3 Cenário

Uma empresa que presta serviços de gestão escolar decidiu integrar seus softwares para oferecer um serviço mais eficiente e de qualidade aos seus clientes visando a automação nos serviços escolares. Para isso, eles irão desenvolver um novo sistema automatizado de gestão escolar que substituirá os softwares atuais, onde deve ser capaz de controlar todos os aspectos relacionados à gestão escolar, incluindo cadastro de alunos, matrículas, cursos, disciplinas, notas e faltas, bem como a geração de relatórios associados a essas atividades.

3.4 Ciclo de vida do software

Para esse projeto de integração de softwares, será utilizado a metodologia ágil Scrum. É interessante trabalharmos com essa metodologia, pois ela se

concentra na entrega contínua de resultados para o cliente em ciclos curtos, chamados de sprints, muito bom destacar que essa boa prática faz com que tenhamos um relacionamento mais íntegro com o cliente e um nível de satisfação e qualidade mais eficaz. A equipe do projeto será composta por um Product Owner, um Scrum Master e um time de desenvolvimento que irá trabalhar em conjunto em todas as etapas do processo de desenvolvimento. Uma das vantagens de utilizar o Scrum, é que serão realizadas reuniões diárias para garantir a comunicação e o acompanhamento do progresso. Ao final de cada sprint, será realizada uma revisão para validação do trabalho concluído. Por fim, a equipe irá se reunir para planejar melhorias para o próximo ciclo.

3.5 LGPD

A LGPD no contexto do nosso software é de extrema importância, pois aspectos como a privacidade dos usuários e reduzir os riscos de violação de dados são objetivos fundamentais em nosso projeto. Além disso, promover transparência e o controle tornando a conformidade uma vantagem competitiva e um diferencial no mercado. Portanto, a LGPD é essencial para proteger a privacidade dos usuários e garantir a segurança de nossos dados.

3.6 Diagrama de Classes

Nesse tópico iremos abordar o diagrama UML de classes, que iremos usar no projeto para a implementação do sistema. Como podemos ver no apêndice B, seu maior objetivo é mostrar as classes que compõem o sistema e seus relacionamentos, como também descrever o que cada classe faz e os seus serviços.

3.7 Elicitação de Requisitos

Com base em uma entrevista feita com o cliente que pode ser vista no apêndice A foram elicitados vários requisitos que podem ser vistos na tabela 20 para a criação de um sistema de gestão escolar.

Tabela 30 - Requisitos do sistema

Código	Tipo	Descrição
RF01	Funcional	O sistema deve permitir o cadastro e alteração de informações de cursos.
RF02	Funcional	O sistema deve permitir o cadastro e alteração de informações de disciplinas.
RF03	Funcional	O sistema deve permitir o cadastro e alteração de informações de turmas.
RF04	Funcional	O sistema deve permitir o cadastro e alteração de informações de alunos e matrículas.
RF05	Funcional	O sistema deve permitir o cadastro e alteração de informações de professores.
RF06	Funcional	O sistema deve permitir a emissão de relatórios de notas e presenças das turmas e disciplinas ministradas.
RF07	Funcional	O sistema deve permitir a gestão do conteúdo de aula.
RF08	Funcional	O sistema deve permitir o acesso às turmas cadastradas.
RF09	Funcional	O sistema deve permitir a alteração de dados pessoais e atualização de dados.
RF10	Funcional	O sistema deve permitir o acesso às notas/médias, presença, conteúdo, matrícula, grade curricular e horária.
RF11	Funcional	O sistema deve permitir a emissão de boletim e certificado.
RNF01	Não Funcional	O sistema deve ser responsivo para as plataformas Web e Mobile.
RNF02	Não Funcional	O sistema web deve ser desenvolvido em Node.JS.
RNF03	Não Funcional	O sistema desktop deve ser desenvolvido em Javascript com o uso dos frameworks e bibliotecas adequadas.
RNF04	Não Funcional	O sistema mobile deve ser desenvolvido em Javascript com o uso dos frameworks e bibliotecas adequadas.
RN01	Regra de Negócio	Somente o usuário com perfil de "Secretaria" pode cadastrar, alterar ou excluir informações de cursos, disciplinas, turmas, alunos/matrículas e professores no sistema desktop
RN02	Regra de Negócio	O sistema desktop deve validar informações duplicadas ou inconsistentes antes de cadastrar ou alterar informações
RN03	Regra de Negócio	O sistema desktop deve permitir a associação de turmas com disciplinas e professores

RN04	Regra de Negócio	Somente o usuário com perfil de "Professor" pode visualizar relatórios de notas e presenças das turmas e disciplinas ministradas no sistema web
RN05	Regra de Negócio	O sistema web deve permitir a gestão do conteúdo de aula pelos professores
RN06	Regra de Negócio	O sistema web deve permitir o acesso dos professores às turmas cadastradas no sistema desktop
RN07	Regra de Negócio	Somente o usuário com perfil de "Aluno" pode visualizar suas informações pessoais, notas/médias, presenças, conteúdo, matrícula, grade curricular e horário no sistema mobile
RN08	Regra de Negócio	O sistema mobile deve permitir a alteração de informações pessoais dos alunos

Fonte: Dos Autores

3.8 Glossário do Sistema

Cadastro de Alunos: Permite o registro de informações dos alunos, como nome completo, data de nascimento, CPF, RG, endereço, telefone e e-mail.

Cursos: Permite o registro de informações sobre os cursos oferecidos pela escola, incluindo nome do curso, carga horária, valor, entre outras informações relevantes.

Disciplinas: Permite o registro de informações sobre as disciplinas oferecidas pela escola, incluindo nome da disciplina, carga horária, professor responsável, entre outras informações relevantes.

Faltas: Permite o registro de faltas dos alunos em aulas e atividades relacionadas às disciplinas cursadas.

Matrículas: Permite o registro de matrículas dos alunos em cursos, disciplinas e turmas.

Notas: Permite o registro de notas obtidas pelos alunos em avaliações e trabalhos relacionados às disciplinas cursadas.

Relatórios: Permite a geração de relatórios de alunos matriculados por turmas, mapa de notas, entre outros relatórios relacionados à gestão escolar.

3.9 Desenvolvimento do sistema

Nos tópicos subsequentes será abordado quais ferramentas utilizaremos no back end e front end para a construção da aplicação, e também de bibliotecas que serão utilizadas no JavaScript por meio do npm o gerenciador de pacotes node.

3.9.1 Linguagem de Programação

A linguagem de programação é o que desenvolve desde a parte da interação do usuário com o sistema, como a própria criação do sistema em si. É no nosso projeto haverá a elaboração de um programa voltado ao sistema de ensino escolar com uma linguagem de programação orientada a objetos que será abordado nos tópicos seguintes.

3.9.1.1 Linguagem JavaScript

Trata-se da linguagem de programação orientada a objetos que utilizamos para o desenvolvimento do programa nos dispositivos desktop e nas páginas web, como linguagem principal para o desenvolvimento tanto do back end quanto do front end do projeto. Além do JavaScript utilizaremos o framework Node para desenvolvê-lo diretamente no ambiente de desenvolvimento é para ter acesso ao npm para instalar bibliotecas auxiliares para a construção da aplicação.

3.9.1.2 Typescript

Com o super set Typescript iremos adicionar uma nova camada ao Javascript para um melhor desenvolvimento orientado a objetos é a adição da tipagem estática para que assim deixe o código solidamente tipado e com a presença da orientação a objetos.

3.9.1.3 React

O React como foi explicado antes utilize da técnica de componentização para criação de interfaces, juntando em um componente o html, css e javascript, por isso iremos utilizá-lo para construção de nossas interfaces.

3.9.1.4 Electron

Será o framework JavaScript que utilizaremos para criação do aplicativo desktop do nosso projeto, com ele poderemos utilizar das tecnologias utilizadas na

web como html, css e javascript e react para a criação também de nosso software desktop.

3.9.1.5 React Native

O React Native será a biblioteca JavaScript que utilizaremos para desenvolver o aplicativo mobile, é criar um ecossistema JavaScript onde a integração com back end é possível com o mobile, web e o desktop.

3.9.1.6 Expo GO

Olhando para a necessidade do projeto em atender o requisito do programa funcionar em celulares, pensamos no Expo GO, basicamente em nosso projeto ele vai simplificar muitos aspectos do processo de criação e desenvolvimento em relação a codificação, permitindo que os desenvolvedores criem aplicativos para iOS e Android usando uma única base de código React Native.

3.9.2 Criação da interface

Para a criação das interfaces do projeto utilizaremos o React tanto para a área web com a adição das bibliotecas ShadCn UI, Tailwind CSS como para área desktop com o Electron e para o mobile utilizaremos o React Native que conta com uma estrutura parecida com html mas utiliza-se de tags direcionadas aos aparelhos móveis.

3.9.2.2 Tailwind CSS

Com o Tailwind CSS conseguimos maximizar o potencial do CSS, oferecendo ao nosso projeto responsividade, código enxuto, customização e integração com IDEs.

3.9.2.3 Shadcn UI

Com o Shadcn UI conseguimos escolher os componentes do React pelo site oficial¹⁰ copiando e colando no código do projeto, personalizando o código de acordo com suas necessidades e deixando a programação mais otimizada em relação a codificação.

3.9.3 Criação do Back End

O back end é a parte "invisível" de um site ou aplicativo, onde os dados são processados, armazenados e as regras de funcionamento são definidas. Ele cuida do armazenamento de informações e também da segurança dessas informações, é importante destacar que o back end também é responsável sobre como o sistema responde às ações dos usuários.

3.9.3.1 Prisma

Com o Prisma facilitamos a interação entre as aplicações e o banco de dados. Ele permite realizarmos essa integração facilitando por exemplo o gerenciamento do banco de dados e acaba fornecendo segurança para nosso sistema, tornando o desenvolvimento do software mais eficiente e seguro.

3.9.3.2 Express

Havendo a necessidade de conexão entre o back end e o Prisma para gerenciamento dos dados pensamos no Express, ele será fundamental pois possui um sistema de roteamento flexível para direcionar solicitações HTTP para funções específicas e funções dentro do back end.

3.9.3.3 Axios

Pensamos no Axios para trabalhar diretamente nas solicitações HTTP no front end, o benefício que isso trás, é otimizar a rota entre todas as solicitações passando pelo front end e sendo alinhado diretamente com back end, garantindo integridade dos dados dentro do sistema.

¹⁰ <https://ui.shadcn.com/docs/components/accordion>

3.9.4 Código da aplicação

Para um melhor entendimento sobre a aplicação como um todo, é possível acessar o código de ambas as partes front end e back end, em nosso repositório do github¹¹, acessando através da nota de rodapé. Dentro do repositório contém as instruções de instalação do projeto em sua máquina pessoal, além de todo o código escrito.

3.9.5 Protótipo

A partir desse tópico será apresentado os protótipos de interface feitos para dar uma ideia de como é o esqueleto das interfaces como podemos ver na Figura 05 que exibe a tela inicial de nosso sistema, e da Figura 06 até Figura 10 é mostrado as outras interfaces como aluno, professor, curso e serviços.

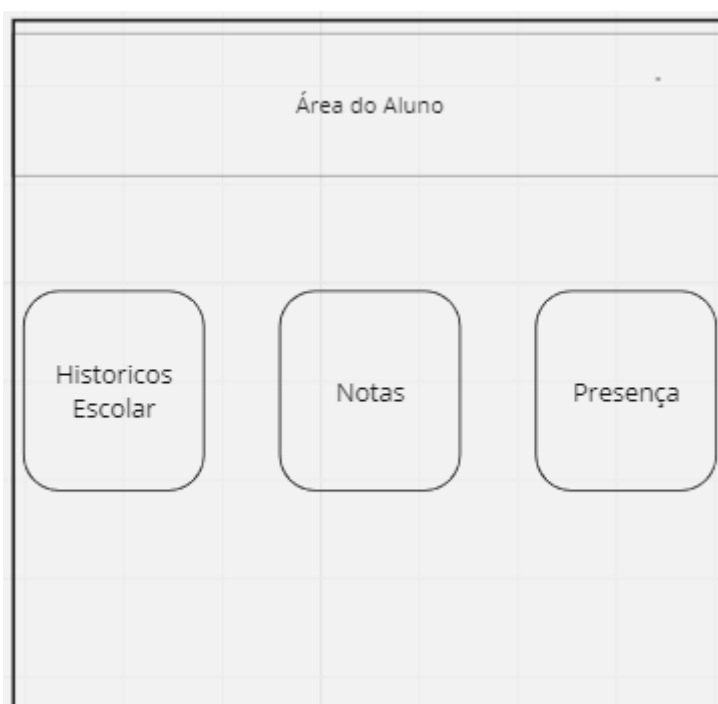
Figura 06 - Protótipo da interface de início



Fonte: Dos autores

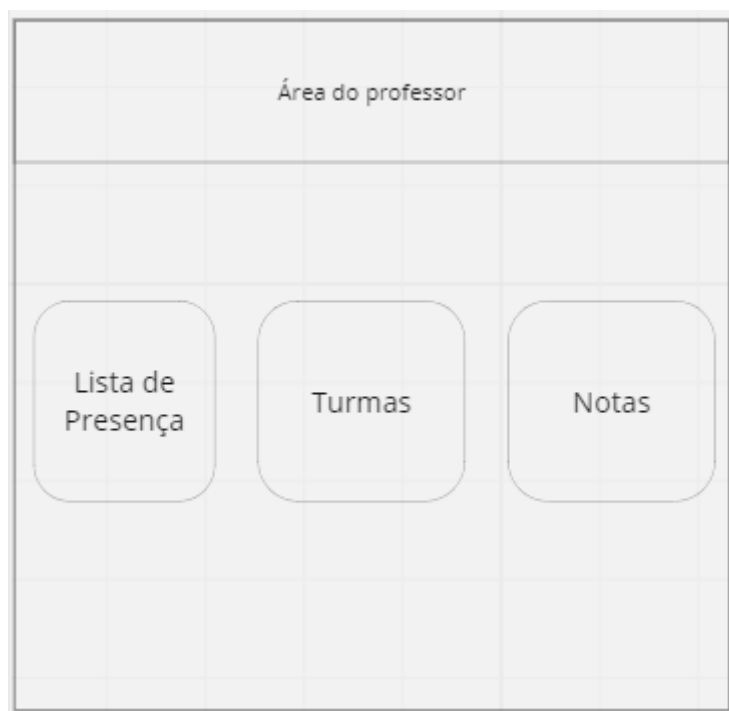
¹¹ <https://github.com/W-Wag/pim-4.3/>

Figura 07 - Protótipo da interface de Aluno



Fonte: Dos autores

Figura 08 - Protótipo da interface de Professor



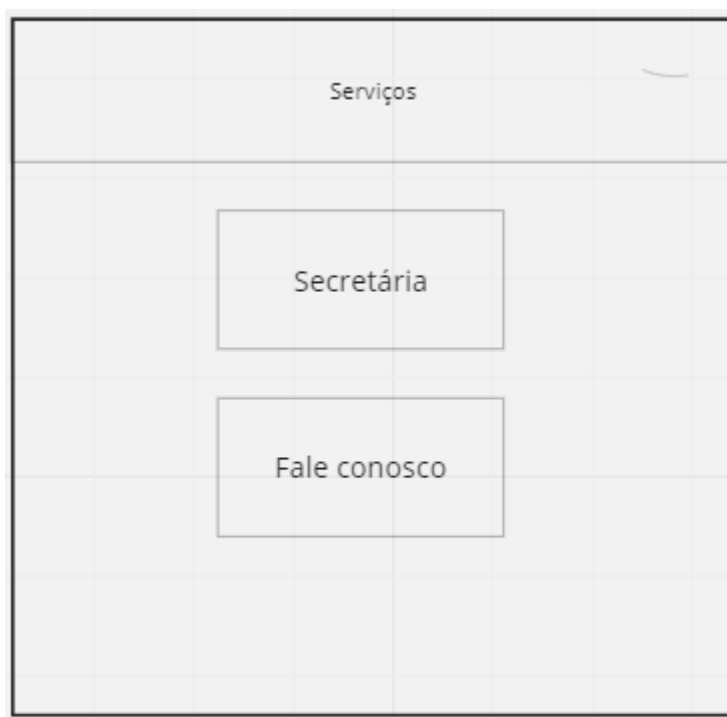
Fonte: Dos autores

Figura 09 - Protótipo da interface de Curso



Fonte: Dos autores

Figura 10 - Protótipo da interface de Serviços



Fonte: Dos autores

3.9.6 Guia de utilização da aplicação

Neste tópico iremos mostrar como o sistema está funcionando em relação ao acesso por parte dos alunos e professores.

O acesso ao perfil do aluno é descrito através da Figura 18, conseguimos visualizar os campos para verificar a lista de presença, especificado através da Figura 20, conseguimos também visualizar o histórico escolar visto na Figura 21 e por último conseguimos visualizar o boletim escolar demonstrado através da Figura 22.

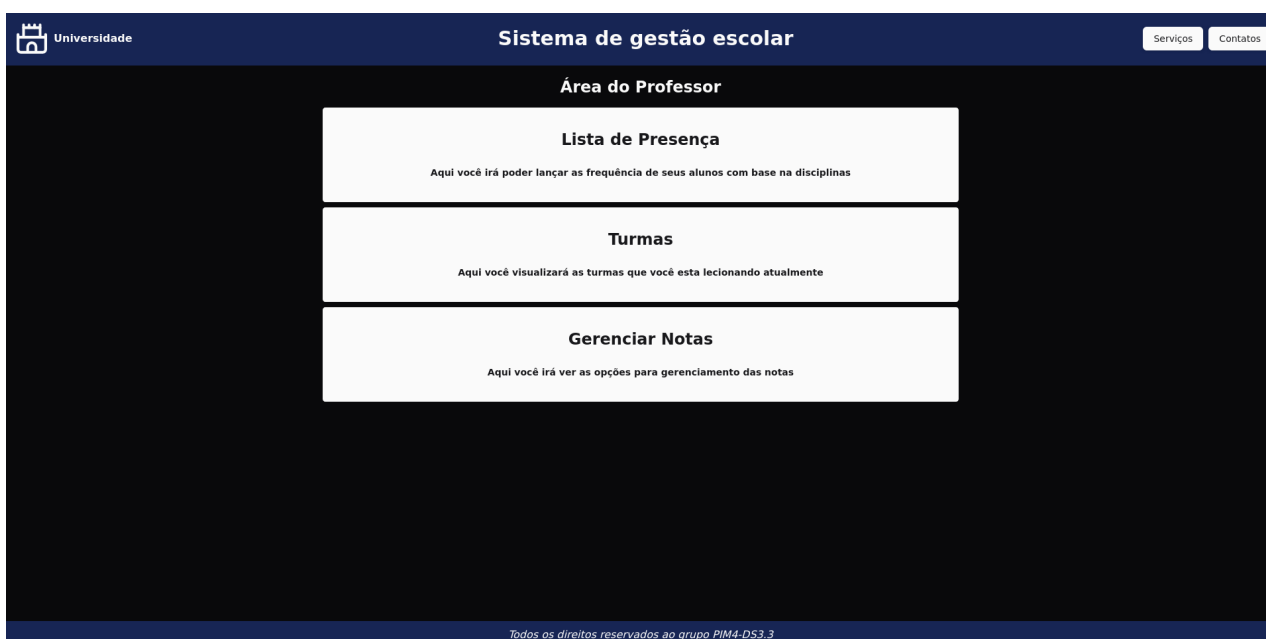
Para a utilização do sistema caso seja um professor, após iniciar o acesso a área do professor como mostra a Figura 11, a aplicação o redirecionará para a área do professor visto na Figura 12, tendo as escolhas de listar e gerenciar a presença de seus alunos, identificar a turma a qual leciona, e o gerenciamento de notas. Podemos observar como funciona o gerenciamento de presença na Figura 13, as turmas na Figura 14 e o gerenciamento de notas na Figura 15, onde a duas opções lançar notas onde e possível gerenciar as notas e trabalhos como é visto na Figura 16 e visualizar o mapa de notas apresentado na Figura 17.

Figura 11 - Apresentação do Sistema



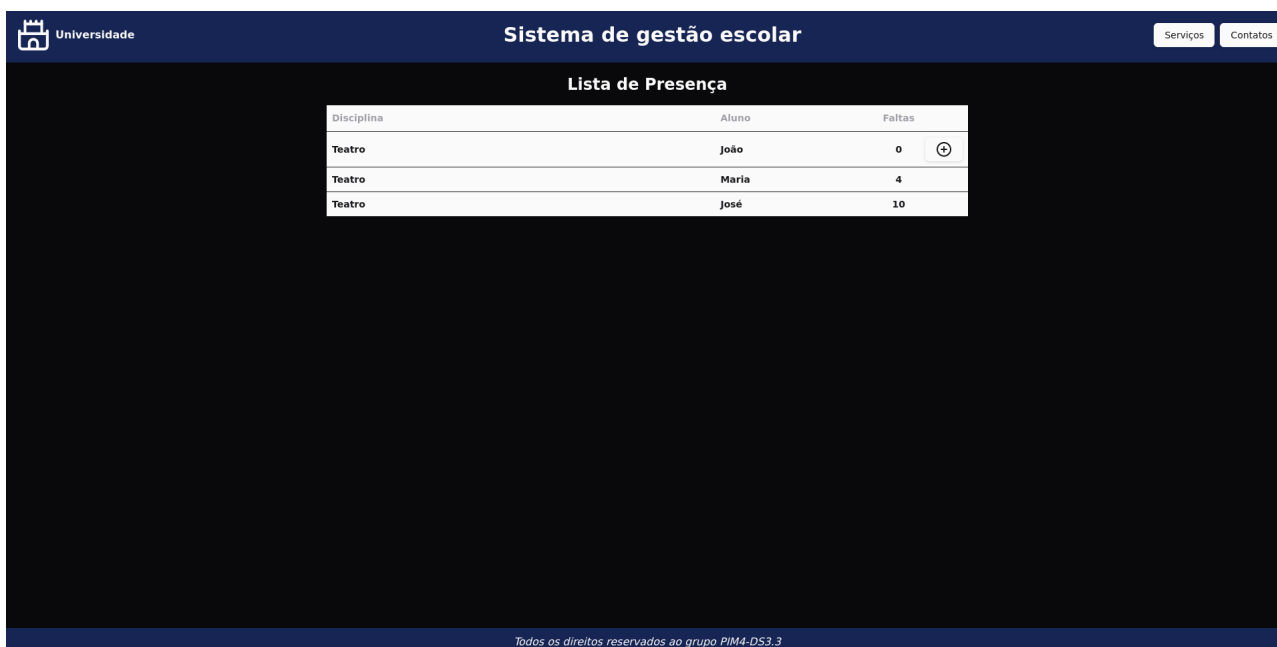
Fonte: Dos autores

Figura 12 - Área do Professor



Fonte: Dos autores

Figura 13 - Gerenciamento de presença



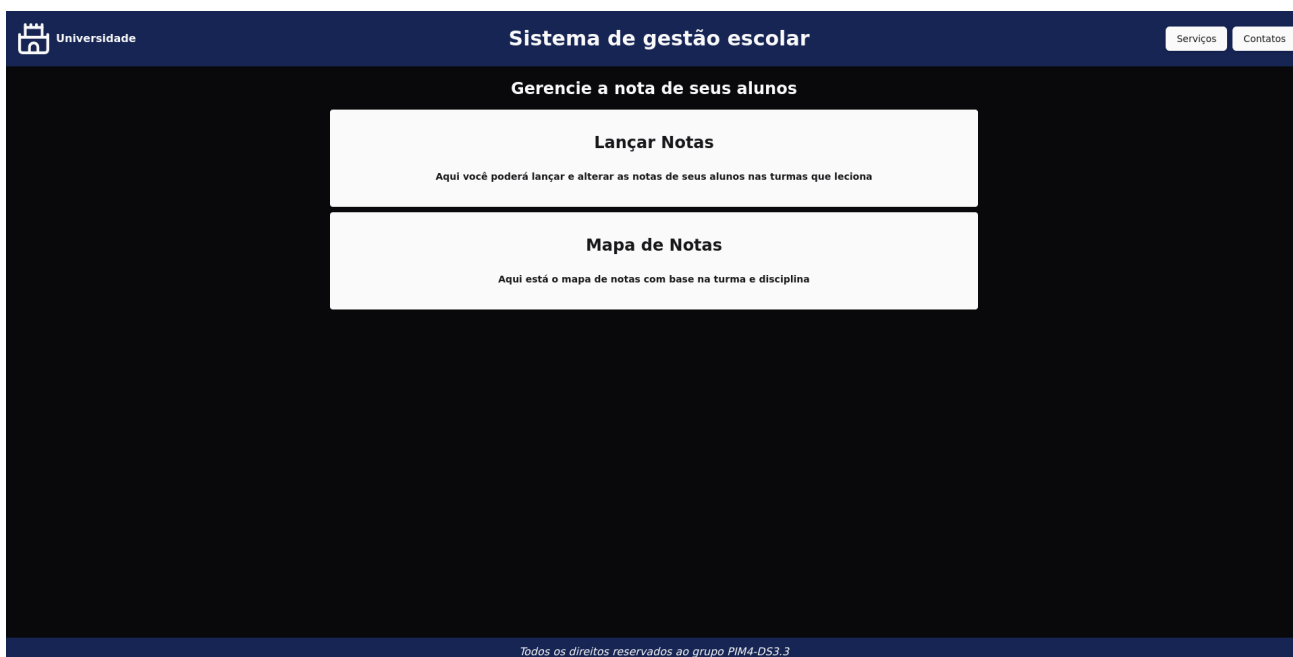
Fonte: Dos autores

Figura 14 - Turmas



Fonte: Dos autores

Figura 15 - Gerenciar notas do Alunos




Fonte: Dos autores

Figura 16 - Gerenciar notas das Provas e Trabalhos

The screenshot displays the 'Sistema de gestão escolar' interface. At the top, there is a header with the university logo and name, the system title, and links for 'Serviços' and 'Contatos'. Below the header, there are four identical forms arranged horizontally. Each form is titled 'Disciplina: Teatro' and 'Aluno: Jonas'. Inside each form, there are three input fields labeled 'NP1', 'NP2', and 'Trabalho', each followed by a small square icon. At the bottom of each form is an 'Alterar' button.

Fonte: Dos autores

Figura 17 - Mapa de notas



Universidade

Sistema de gestão escolar

Serviços

Contatos

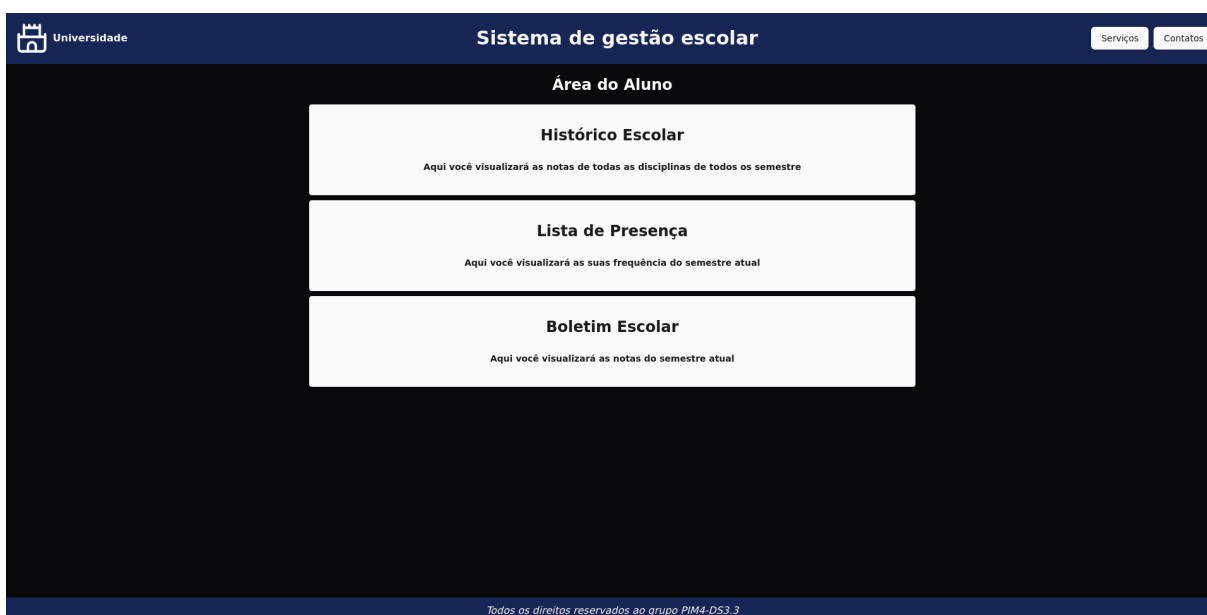
Mapa de notas

Turma: ADS 3/4	Disciplina: Artes cênicas		Semestre: 4		
Aluno	NP1	NP2	Trabalho	Média Final	Situação
Wilson	6.5	5.5	7.4	6.25	AP
Roberto	5	0	0	2	C

Todos os direitos reservados ao grupo PIM4-DS3.3

Fonte: Dos autores

Figura 18 - Área do Aluno



Fonte: Dos autores

Figura 19 - Histórico Escolar

Universidade **Sistema de gestão escolar** [Serviços](#) [Contatos](#)


Histórico escolar

Disciplina: Artes cênicas		Semestre: 1	
NP1	NP2	Trabalho	Média Final
6,5	5,5	7,4	6,25
Disciplina: Teatro		Semestre: 2	
NP1	NP2	Trabalho	Média Final
8	7	6	7,6
Disciplina: História da arte		Semestre: 3	
NP1	NP2	Trabalho	Média Final
6,5	5,5	7,4	6,25

Todos os direitos reservados ao grupo PIM4-DS3.3

Fonte: Dos autores

Figura 20 - Lista de Presença

Universidade

Sistema de gestão escolar

Serviços

Contatos


Lista de Presença

Disciplinas	Presença
Teatro	0
História da arte	12
Artes Cênicas	4

Todos os direitos reservados ao grupo PIM4-DS3.3

Fonte: Dos autores

Figura 21 - Boletim Escolar

Universidade

Sistema de gestão escolar

Serviços

Contatos

Boletim Escolar

Semestre atual: 1

Disciplina: Artes cênicas			
NP1	NP2	Trabalho	Média Final
6.5	5.5	7.4	6.25
Disciplina: Artes cênicas			
NP1	NP2	Trabalho	Média Final
6.5	5.5	7.4	6.25
Disciplina: Artes cênicas			
NP1	NP2	Trabalho	Média Final
6.5	5.5	7.4	6.25

Todos os direitos reservados ao grupo PIM4-DS3.3

Fonte: Dos autores

4. CONCLUSÃO

Concluimos que o desenvolvimento do projeto teve como principal foco aprimorar a gestão escolar, visando a eficiência e a qualidade dos serviços por meio da automação, especialmente na gestão de notas dos alunos e gerenciamento de dados da instituição.

O aprendizado que obtivemos proporcionou uma valiosa oportunidade de adquirir conhecimento prático em relação a gestão escolar e a implementação de processos de automação nesse contexto.

A abordagem ágil Scrum desempenhou um papel crucial no desenvolvimento do projeto, com reuniões diárias que asseguravam uma comunicação eficaz e monitoramento contínuo do progresso. Após cada reunião, realizamos uma revisão para avaliar a porcentagem de conclusão das tarefas, sempre buscando aprimoramentos para os próximos ciclos, em conformidade com o conceito de melhoria contínua.

Durante o desenvolvimento, criamos diversos diagramas para representar visualmente o projeto. Isso inclui diagramas de caso de uso, diagramas de entidade e relacionamento e diagramas de classes, todos concebidos para fornecer uma descrição precisa e detalhada de cada etapa necessária para executar as funcionalidades do sistema.

O projeto incorpora uma conexão com um banco de dados relacional, o qual será responsável por armazenar os dados dos usuários no sistema. Optamos pelo SQL Server devido à sua adequação às necessidades do projeto.

A codificação do sistema foi predominantemente baseada em JavaScript, com o auxílio do TypeScript para aplicar conceitos de programação orientada a objetos. Com ênfase em uma codificação bem documentada e organizada, visando facilitar o desenvolvimento contínuo do projeto e a colaboração eficaz entre os membros da equipe. Além disso, a parte visual do sistema (front-end) foi desenvolvida utilizando React, React Native, Electron, destacamos que utilizamos também o Expo GO para

realizar o desenvolvimento mobile. O back end foi desenvolvido utilizando Prisma, Express e Axios para a integração entre o back end e o front end.

É esperado que essas escolhas e abordagens conduzam a um sistema altamente organizado e eficiente, cumprindo os objetivos específicos do projeto.

No futuro, o planejado é expandir as funcionalidades para atender todas às necessidades da gestão escolar, melhorar a interface do usuário, reforçar a segurança, garantir a escalabilidade e o desempenho, criar documentação abrangente e fornecer treinamento. Por fim, o objetivo do projeto é criar uma solução de gestão escolar mais eficiente, adaptável e de alta qualidade para instituições educacionais, visando a automação de todos os processos escolares.

REFERÊNCIAS BIBLIOGRÁFICAS

- Araújo, M. A. P. "Modelagem de dados—teoria e prática." *Revista Saber Digital* 1.01 (2008): 27-64.
- BODUCH, Adam. *React and react native*. Packt Publishing Ltd, 2017.
- BROOKS, D. R. (2007). *An Introduction to HTML and JavaScript for Scientists and Engineers*. London: Springer-Verlag.
- BLOKDYK, Gerardus. *Front-End Web Development: A Complete Guide*. 2018. 274 p. Editora: Emereo Pty Limited.
- DA SILVEIRA, Marcos Cristiano; MONTEIRO, José Maria; DE SOUZA, Jefferson Teixeira. Um ambiente de mlearning para ensino da linguagem sql. *Simpósio Brasileiro de Informática na Educação (SBIE)*, 2010.
- DEITEL, Harvey M. et al. *C# como programar*. São Paulo: Pearson Education, 2003.
- DOS SANTOS SOARES, Michel. Metodologias ágeis extreme programming e scrum para o desenvolvimento de software. *Revista Eletrônica de Sistemas de Informação*, v. 3, n. 1, 2004. (<http://periodicosibepes.org.br/index.php/reinfo/article/view/146> - Acessado em 12/05/2023)
- de Carvalho Santos, Rodrigo, and Luís Augusto Mattos Mendes. "Estudo Comparativo dos Sistemas Gerenciadores de Bancos de Dados: Oracle, SQL Server e PostgreSQL."
- de Oliveira, Nairobi Spiecker, et al. "Segurança da informação para internet das coisas (iot): uma abordagem sobre a lei geral de proteção de dados (LGPD)." *Revista Eletrônica de Iniciação Científica em Computação* 17.4 (2019).
- de Oliveira Lopes, Jean. "PHP ou TypeScript: uma comparação de duas linguagens para web pelas suas características."
- Elmasri, Ramez. *Sistemas de banco de dados / Ramez Elmasri e Shamkant B. Navathe ; tradução Daniel Vieira ; revisão técnica Enzo Seraphim e Thatyana de Faria Piola Seraphim*. -- 6. ed. -- São Paulo : Pearson Addison Wesley, 2011.
- ESPOSITO, Dino. *Programming Microsoft ASP.NET 4*. Microsoft Press – Washington, 2010.
- Express. Disponível em: <https://expressjs.com/>. Acesso em: 19 de out. de 2023
- FARINELLI, Fernanda. *Conceitos básicos de programação orientada a objetos*. Instituto Federal Sudeste de Minas Gerais, 2007.
- FRANCK, K. M. .; PEREIRA, R. F. .; DANTAS FILHO, J. V. . Diagrama Entidade-Relacionamento: uma ferramenta para modelagem de dados conceituais em Engenharia de Software. *Research, Society and Development*, [S. l.], v. 10, n. 8, p. e49510817776, 2021. DOI: 10.33448/rsd-v10i8.17776. Disponível em: <https://rsdjournal.org/index.php/rsd/article/view/17776>. Acesso em: 21 maio. 2023.
- GRANNELL, C. (2007). *The Essential Guide to CSS and HTML Web Design*. New York: APress.

- I. Sommerville, e P. Sawyer, Requirements Engineering - A good practice guide - Wiley 1997.
- HORSTMANN, Cay. CORNELL, Gary. Core Java 2 – Fundamentals. The Sun Microsystems Press Java Series. Vol. I. 2003.
- JOHNSON, Ralph E. Frameworks = (Components + Patterns). Communications of the ACM, Vol. 40. No 10, Outubro 1997, p. 39 – 42.
- JUNIOR, FERNANDO COSTA SERRA, and BRUNO SEABRA NOGUEIRA MENDONÇA LIMA. "TECNOLOGIA DOCKER."
- Kanakadoss, S. (2005). SIMBUILDER: AN INVESTIGATION AND USABILITY STUDY OF NOVICE PROGRAMMING TECHNIQUES. Alabama: Auburn University.
- Larman, C. Utilizando UML e Padrões. Editora Bookman, (2007).
- MACÁRIO, Carla Geovana do N.; BALDO, Stefano Monteiro. O modelo relacional. Instituto de Computação Unicamp. Campinas, p. 1-15, 2005.
- MANOVICH, Lev. Banco de dados. Revista ECO-Pós, v. 18, n. 1, p. 7-26, 2015.
- MARQUEZ - SOTO. Pedro. Backend Developer in 30 Days: Acquire Skills on API Designing, Data Management, Application Testing, Deployment, Security and Performance Optimization (English Edition). BPB Publications, 2022.
- NATIVE, React. React Native. Disponível em: <https://reactnative.dev/>. [Último acesso: 25 de maio 2023], 2020.
- Praciano, F. D., de Sousa, J. F. L., & Machado, J. C. (2019, November). Uma análise experimental da utilização de diferentes tecnologias de armazenamento em um SGBD relacional. In Anais do XXXIV Simpósio Brasileiro de Banco de Dados (pp. 259-264). SBC.
- Prisma. Disponível em <https://riut.utfpr.edu.br/jspui/bitstream/1/32151/1/sistemawebgerenciamentogoluseimas.pdf>. Acesso em 24 de Out. 2023
- Silva, Maurício Samy. JavaScript-Guia do Programador: Guia completo das funcionalidades de linguagem JavaScript. Novatec Editora, 2020.
- shadcn ui. Disponível em <https://ui.shadcn.com/>. Acesso em: 19 de out. de 2023
- SMITH, John. Introdução aos Diagramas de Caso de Uso: Conceitos e Aplicações. TechTrends Journal, [S.l.], v. 10, n. 2, p. 45-62, mar. 2023.
- Tailwind. Disponível em: <https://codigofonte.com.br/artigos/por-que-usar-tailwind-css>. Acesso em: 12 de out. de 2023.
- Tilkov, Stefan, and Steve Vinoski. "Node. js: Using JavaScript to build high-performance network programs." IEEE Internet Computing 14.6 (2010): 80-83.
- VARGAS, Thânia Clair de Souza. A História de UML e seus Diagramas - Departamento de Informática e Estatística Universidade Federal de Santa Catarina (UFSC) – Florianópolis, SC – Brazil.
- MEDEIROS, Ernani Sales de. Desenvolvendo software com UML 2.0 definitivo. São Paulo, Pearson Makron Books, 2004.

APÊNDICE

Apêndice A - Entrevista

Analista: Qual o papel que você exerce?

Cliente: Eu excesso dois papéis! 1° de coordenação e 2° de professora

Analista: Qual seria a finalidade do sistema?

Cliente: Eu tenho uma universidade e eu preciso fazer a gestão dos meus alunos, cursos e turma.

Analista: O sistema que minha equipe vai desenvolver vai ser de acesso apenas para os operadores, ou seja administrativo e RH ou os alunos e professores terão acesso?

Cliente: Eu preciso de um projeto mobile, web e desktop. Então eu preciso que o mobile seja para o aluno, o sistema web para o professor e o desktop para secretaria.

Analista: Qual seria o acesso do professor?

Cliente: O professor vai ter algumas responsabilidades 1° lançar notas 2° lançar frequência 3° lançar conteúdo programático e isso para cada disciplina e para cada turma. Além desses lançamentos, o professor pode visualizar mapa de notas e imprimir. E ele pode visualizar e imprimir as listas de frequências em aula e frequência em prova.

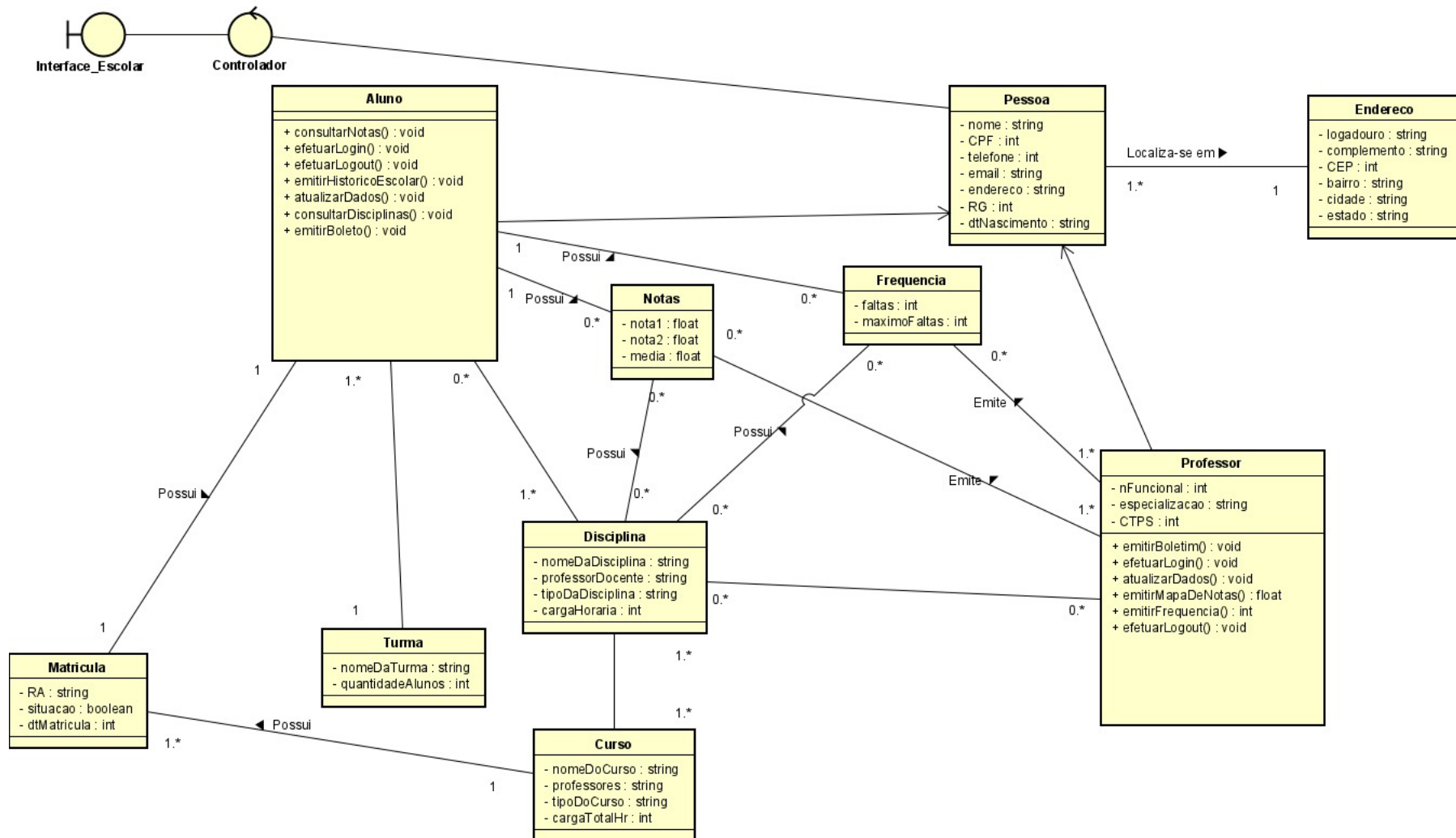
Analista: Qual vai ser o acesso do aluno?

Cliente: Ele vai poder visualizar nota, visualizar conteúdo, visualizar boletim escolar e visualizar o histórico escolar.

Analista: Qual seria a função da secretaria?

Cliente: A Secretaria vai cadastrar aluno, fazer matrícula do aluno. Esse cadastro pode ser alterado pela secretária e também o aluno pode gerir essas funcionalidades. E além de cadastrar aluno, a secretaria vai cadastrar turmas, cadastrar disciplinas, cadastrar salas, cadastrar blocos, faz também às matrículas e todas as responsabilidades do professor a secretaria faz também!

Apêndice B - Diagrama de Classes



Apêndice C - Código de criação do banco de dados

```
BEGIN TRY

BEGIN TRAN;

-- CreateTable
CREATE TABLE [dbo].[Aluno] (
[id] INT NOT NULL IDENTITY(1,1),
[cpf] NVARCHAR(1000) NOT NULL,
[nome] NVARCHAR(1000) NOT NULL,
[email] NVARCHAR(1000) NOT NULL,
[data_nascimento] DATETIME2 NOT NULL,
[rg] NVARCHAR(1000) NOT NULL,
[telefone] NVARCHAR(1000) NOT NULL,
[enderecoId] INT NOT NULL,
CONSTRAINT [Aluno_pkey] PRIMARY KEY CLUSTERED ([id]),
CONSTRAINT [Aluno_cpf_key] UNIQUE NONCLUSTERED ([cpf]),
CONSTRAINT [Aluno_email_key] UNIQUE NONCLUSTERED
(email)
);

-- CreateTable
CREATE TABLE [dbo].[Endereco] (
[id] INT NOT NULL IDENTITY(1,1),
[logradouro] NVARCHAR(1000) NOT NULL,
[estado] NVARCHAR(1000) NOT NULL,
[complemento] NVARCHAR(1000),
[cep] NVARCHAR(1000) NOT NULL,
[cidade] NVARCHAR(1000) NOT NULL,
[bairro] NVARCHAR(1000) NOT NULL,
[alunoId] INT NOT NULL,
```

```

CONSTRAINT [Endereco_pkey] PRIMARY KEY CLUSTERED ([id])
);

-- AddForeignKey
ALTER TABLE [dbo].[Endereco] ADD CONSTRAINT
[Endereco_alunoId_fkey] FOREIGN KEY ([alunoId]) REFERENCES
[dbo].[Aluno]([id]) ON DELETE NO ACTION ON UPDATE CASCADE;
-- DropForeignKey
ALTER TABLE [dbo].[Endereco] DROP CONSTRAINT
[Endereco_alunoId_fkey];

-- DropIndex
ALTER TABLE [dbo].[Aluno] DROP CONSTRAINT [Aluno_email_key];

-- AlterTable
ALTER TABLE [dbo].[Aluno] DROP CONSTRAINT [Aluno_pkey];
ALTER TABLE [dbo].[Aluno] DROP COLUMN [data_nascimento],
[id];
ALTER TABLE [dbo].[Aluno] ADD CONSTRAINT Aluno_pkey
PRIMARY KEY CLUSTERED ([cpf]);
ALTER TABLE [dbo].[Aluno] ADD [dt_nascimento] DATETIME2 NOT
NULL,
[genero] NVARCHAR(1000) NOT NULL,
[id_endereco] INT NOT NULL,
[ra] NVARCHAR(1000) NOT NULL,
[telefone2] NVARCHAR(1000);

-- AlterTable
ALTER TABLE [dbo].[Endereco] DROP COLUMN [alunoId],
[cidade];
ALTER TABLE [dbo].[Endereco] ADD [id_cidade] INT NOT NULL;

```

-- CreateTable

```
CREATE TABLE [dbo].[Professor] (
[cpf] NVARCHAR(1000) NOT NULL,
[nome] NVARCHAR(1000) NOT NULL,
[dt_nascimento] DATETIME2 NOT NULL,
[email] NVARCHAR(1000) NOT NULL,
[ctps] NVARCHAR(1000) NOT NULL,
[rg] NVARCHAR(1000) NOT NULL,
[titularidade] NVARCHAR(1000) NOT NULL,
[funcional] NVARCHAR(1000) NOT NULL,
[telefone] NVARCHAR(1000) NOT NULL,
[telefone2] NVARCHAR(1000),
[id_endereco] INT NOT NULL,
CONSTRAINT [Professor_pkey] PRIMARY KEY CLUSTERED ([cpf]),
CONSTRAINT [Professor_cpf_key] UNIQUE NONCLUSTERED ([cpf])
);
```

-- CreateTable

```
CREATE TABLE [dbo].[Matricula] (
[ra] NVARCHAR(1000) NOT NULL,
[situacao] NVARCHAR(1000) NOT NULL,
[data_Matricula] DATETIME2 NOT NULL CONSTRAINT
[Matricula_data_Matricula_df] DEFAULT CURRENT_TIMESTAMP,
CONSTRAINT [Matricula_pkey] PRIMARY KEY CLUSTERED ([ra]),
CONSTRAINT [Matricula_ra_key] UNIQUE NONCLUSTERED ([ra])
);
```

-- CreateTable

```
CREATE TABLE [dbo].[Turma] (
[cod] NVARCHAR(1000) NOT NULL,
[Quantidade_alunos] INT NOT NULL,
[Curso_cod] NVARCHAR(1000) NOT NULL,
```

```

CONSTRAINT [Turma_pkey] PRIMARY KEY CLUSTERED ([cod]),
CONSTRAINT [Turma_cod_key] UNIQUE NONCLUSTERED ([cod])
);

```

```
-- CreateTable
```

```

CREATE TABLE [dbo].[Curso] (
[cod] NVARCHAR(1000) NOT NULL,
[nome] NVARCHAR(1000) NOT NULL,
[carga_horaria] INT NOT NULL,
CONSTRAINT [Curso_pkey] PRIMARY KEY CLUSTERED ([cod]),
CONSTRAINT [Curso_cod_key] UNIQUE NONCLUSTERED ([cod])
);

```

```
-- CreateTable
```

```

CREATE TABLE [dbo].[Disciplina] (
[cod_disciplina] INT NOT NULL,
[nome] NVARCHAR(1000) NOT NULL,
[carga_horaria] INT NOT NULL,
[cpf_professor] NVARCHAR(1000) NOT NULL,
[cod_curso] NVARCHAR(1000) NOT NULL,
CONSTRAINT [Disciplina_pkey] PRIMARY KEY CLUSTERED
([cod_disciplina]),
CONSTRAINT [Disciplina_cod_disciplina_key] UNIQUE
NONCLUSTERED ([cod_disciplina])
);

```

```
-- CreateTable
```

```

CREATE TABLE [dbo].[Nota] (
[id] INT NOT NULL IDENTITY(1,1),
[np1] DECIMAL(32,16) NOT NULL,
[np2] DECIMAL(32,16) NOT NULL,
[pim] DECIMAL(32,16) NOT NULL,

```

```
[Semestre] INT NOT NULL,
[cpf_aluno] NVARCHAR(1000) NOT NULL,
CONSTRAINT [Nota_pkey] PRIMARY KEY CLUSTERED ([id])
);
```

```
-- CreateTable
```

```
CREATE TABLE [dbo].[Cidade] (
[id] INT NOT NULL IDENTITY(1,1),
[nome] NVARCHAR(1000) NOT NULL,
[uf_estado] NVARCHAR(1000) NOT NULL,
CONSTRAINT [Cidade_pkey] PRIMARY KEY CLUSTERED ([id])
);
```

```
-- CreateTable
```

```
CREATE TABLE [dbo].[Estado] (
[uf] NVARCHAR(1000) NOT NULL,
[nome] NVARCHAR(1000) NOT NULL,
CONSTRAINT [Estado_pkey] PRIMARY KEY CLUSTERED ([uf])
);
```

```
-- AddForeignKey
```

```
ALTER TABLE [dbo].[Aluno] ADD CONSTRAINT
[Aluno_id_endereco_fkey] FOREIGN KEY ([id_endereco]) REFERENCES
[dbo].[Endereco]([id]) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
-- AddForeignKey
```

```
ALTER TABLE [dbo].[Aluno] ADD CONSTRAINT [Aluno_ra_fkey]
FOREIGN KEY ([ra]) REFERENCES [dbo].[Matricula]([ra]) ON DELETE
CASCADE ON UPDATE CASCADE;
```

```
-- AddForeignKey
```

```

ALTER TABLE [dbo].[Professor] ADD CONSTRAINT
[Professor_id_endereco_fkey] FOREIGN KEY ([id_endereco])
REFERENCES [dbo].[Endereco]([id]) ON DELETE CASCADE ON UPDATE
CASCADE;

```

```

-- AddForeignKey
ALTER TABLE [dbo].[Turma] ADD CONSTRAINT
[Turma_Curso_cod_fkey] FOREIGN KEY ([Curso_cod]) REFERENCES
[dbo].[Curso]([cod]) ON DELETE CASCADE ON UPDATE CASCADE;

```

```

-- AddForeignKey
ALTER TABLE [dbo].[Disciplina] ADD CONSTRAINT
[Disciplina_cpf_professor_fkey] FOREIGN KEY ([cpf_professor])
REFERENCES [dbo].[Professor]([cpf]) ON DELETE CASCADE ON UPDATE
CASCADE;

```

```

-- AddForeignKey
ALTER TABLE [dbo].[Disciplina] ADD CONSTRAINT
[Disciplina_cod_curso_fkey] FOREIGN KEY ([cod_curso]) REFERENCES
[dbo].[Curso]([cod]) ON DELETE CASCADE ON UPDATE CASCADE;

```

```

-- AddForeignKey
ALTER TABLE [dbo].[Nota] ADD CONSTRAINT
[Nota_cpf_aluno_fkey] FOREIGN KEY ([cpf_aluno]) REFERENCES
[dbo].[Aluno]([cpf]) ON DELETE CASCADE ON UPDATE CASCADE;

```

```

-- AddForeignKey
ALTER TABLE [dbo].[Endereco] ADD CONSTRAINT
[Endereco_id_cidade_fkey] FOREIGN KEY ([id_cidade]) REFERENCES
[dbo].[Cidade]([id]) ON DELETE CASCADE ON UPDATE CASCADE;

```

```

-- AddForeignKey

```

```
ALTER TABLE [dbo].[Cidade] ADD CONSTRAINT  
[Cidade_uf_estado_fkey] FOREIGN KEY ([uf_estado]) REFERENCES  
[dbo].[Estado]([uf]) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
COMMIT TRAN;
```

```
END TRY
```

```
BEGIN CATCH
```

```
IF @@TRANCOUNT > 0
```

```
BEGIN
```

```
ROLLBACK TRAN;
```

```
END;
```

```
THROW
```

```
END CATCH
```