REPÚBLICA FEDERATIVA DO BRASIL

MINISTÉRIO DA ECONOMIA

INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL

DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

Certificado de Registro de Programa de Computador

Processo Nº: **BR512022001664-8**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 31/03/2022, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: Pra Nadar Web

Data de publicação: 31/03/2022

Data de criação: 30/03/2022

Titular(es): FUNDAÇÃO UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE - FUERN

Autor(es): ADALBERTO VERONESE DA COSTA; SEBASTIÃO EMIDIO ALVES FILHO; EXLLEY CLEMENTE DOS SANTOS; TIAGO DA SILVA MORAIS; JEFTER ROBERTO MOTA TARGINO; FRANCISCO CLEMENTINO MAIA JÚNIOR; MATHEUS DIOGENES DA SILVA; HÉLIO VICTOR APOLINÁRIO SOARES

Linguagem: JAVA SCRIPT; OUTROS

Campo de aplicação: AD-01; IF-01; SD-01

Tipo de programa: AP-01; GI-01; SO-05

Algoritmo hash: SHA-512

Resumo digital hash:
5acd83dcc7e7f499d85a34aecede682f6ba90dc3abf02fcb74b658ad37ad1797a34c15d9a2045824ffa3ae57b511b63b284c15bb11c673ff1fb64427f5f0683c

Expedido em: 12/07/2022

Aprovado por:

Joelson Gomes Pequeno

Chefe Substituto da DIPTO - PORTARIA/INPI/DIRPA Nº 02, DE 10 DE FEVEREIRO DE 2021

UNIVERSIDADE DO ESTADO DO RIO GRANDE DO NORTE – UERN
FACULDADE DE CIÊNCIAS EXATAS E NATURAIS – FANAT
DEPARTAMENTO DE INFORMÁTICA – DI

HÉLIO VICTOR APOLINÁRIO SOARES

PRA NADAR: UMA APLICAÇÃO FRONT-END WEB PARA GERENCIAMENTO DO PROGRAMA PRA NADAR

MOSSORÓ - RN
2022

HÉLIO VICTOR APOLINÁRIO SOARES

PRA NADAR: UMA APLICAÇÃO FRONT-END WEB PARA GERENCIAMENTO DO PROGRAMA PRA NADAR

Relatório apresentado ao curso de Ciência da Computação da Universidade do Estado do Rio Grande no Norte como requisito da disciplina de Trabalho de Diplomação, sob a orientação do Prof. Dr. Sebastião Emidio Alves Filho e coorientação do Me. Exlley Clemente dos Santos.

MOSSORÓ - RN
2022

HÉLIO VITOR APOLINÁRIO SOARES

PRA NADAR: UMA APLICAÇÃO FRONT-END WEB PARA GERENCIAMENTO DO PROGRAMA PRA NADAR

Registro de software apresentado como pré-requisito para a obtenção do título de Bacharel em Ciência da Computação da Universidade do Estado do Rio Grande do Norte – UERN, submetida à aprovação da banca examinadora composta pelos seguintes membros:

Aprovada em: 20/09/2022

Banca Examinadora

Sebastião Emídio Alves Filho

Prof. Dr. Sebastião Emídio Alves Filho
Universidade do Estado do Rio Grande do Norte - UERN

Exlley Clemente dos Santos

Me. Exlley Clemente dos Santos
Universidade do Estado do Rio Grande do Norte - UERN

Ceres Germanna Braga Morais

Profa. Ms. Ceres Germanna Braga Morais
Universidade do Estado do Rio Grande do Norte - UERN

Prof. Dr. Adalberto Veronesse da Costa
Universidade do Estado do Rio Grande do Norte - UERN

SUMÁRIO

1 INTRODUÇÃO

O Programa Pra Nadar é um projeto de extensão fundado e dirigido pela Faculdade de Educação Física (FAEF) da Universidade do Estado do Rio Grande do Norte (UERN), oferecendo práticas de atividades aquáticas, como natação e hidroginástica. As práticas são oferecidas para a comunidade em geral, o que inclui alunos, professores e servidores da UERN, além da comunidade externa. O processo de matrícula dos praticantes passa pela necessidade da entrega de atestado médico, dando a devida condição para o exercício das atividades. O Pra Nadar funciona através das contribuições financeiras, provenientes da participação dos alunos matriculados no programa.

O programa inicialmente era organizado pelos secretários e professores da faculdade e não existia nenhum sistema de informática para contribuir na gestão dos dados que estavam sendo inseridos e tratados. O processo de gerenciamento de turmas, alunos, instrutores e secretários no programa era feito de forma manual, ocasionando dificuldade no gerenciamento.

Diante da dificuldade no gerenciamento, contando com problemas como complexidade no controle do número de alunos em cada turma e vagas disponíveis para matrículas, registro das contribuições feitas durante o decorrer do programa e controle das frequências dos alunos e instrutores. Com estas adversidades, a FAEF necessitava de um sistema de gerenciamento para ter melhor controle do programa. Desse modo, foi desenvolvido um sistema que visa administrar os dados do programa de forma rápida, intuitiva e dinâmica.

2 OBJETIVOS

O presente trabalho tem a finalidade de apresentar um sistema, denominado PraNadar Web, de gerenciamento de informações desenvolvido para auxiliar a equipe de alunos e professores da FAEF envolvida no programa Pra Nadar.

O projeto tem como objetivo implementar uma aplicação *front-end* para Web, sendo ela a parte visual e que vai ser utilizada diretamente pelo usuário, visando facilitar a forma com que os usuários exercem suas funções e atividades dentro do sistema.

3 METODOLOGIA

Este trabalho fundamentou-se nos conceitos da metodologia *Feature Driven Development (FDD,* em português, Desenvolvimento Dirigido por Funcionalidade*)* que segundo Rocha (2013): "… busca o desenvolvimento por funcionalidade, ou seja, por um requisito funcional do sistema". A FDD visa o desenvolvimento de funcionalidades em pequenos ciclos, normalmente entre dois e dez dias (COIMBRA, 2020). Para o desenvolver dessa aplicação ocorreram ciclos maiores, que tinham duração média de duas semanas, mas mantendo o foco na implementação de funcionalidades.

3.1 ETAPAS

Com base nisso, pode ser abordado mais sobre as etapas que foram implantadas durante o processo. Primeiramente, houve o levantamento de requisitos, com o intuito de reunir todos os requisitos que seriam necessários para concretizar o objetivo pretendido no projeto. Estes requisitos foram levantados baseados no que era realizado no programa Pra Nadar.

Segundo Pressman (1995),

> A análise de requisitos possibilita que o engenheiro de sistemas especifique a função e o desempenho do software, indique a interface do software com outros elementos do sistema e estabeleça quais são as restrições de projeto que o software deve enfrentar.

No Quadro 1, pode ser visto o requisito referente a autenticação de usuários dentro do sistema, permitindo que estes tenham acesso ao sistema baseado no seu usuário.

Quadro 1 - Requisito funcional 01

[RF-01] Login do Usuário
Descrição de caso de uso: Este caso de uso permite que o usuário seja validado e tenha acesso ao sistema.

Prioridade: [X] Essencial [ ] Importante [ ] Desejável
Entradas e pré-condições: Tipo do acesso, email e senha; Usuário possuir cadastro no sistema.
Saídas e pós-condições: Tipo da autenticação, token, tipo do acesso, id e nome; Usuário logado.
Fluxo básico: 1. Usuário desejar logar no sistema 2. Usuário escolher o tipo de acesso 3. Usuário preencher email e senha 4. Autenticação é bem sucedida 5. Usuário é logado no sistema
Fluxo alternativo: 1. Usuário desejar logar no sistema 2. Usuário escolher o tipo de acesso 3. Usuário preencher email e senha 4. Autenticação é mal sucedida 5. Retorna ao passo 3.

Fonte: Autoria Própria (2022)

Outrossim, antes de iniciar o desenvolvimento em si, foi feita a prototipagem de algumas telas da aplicação, e com isso, pôde ser visualizado de maneira rápida e simples, a estrutura das interfaces que vão ser usadas pelo usuário.

Na fase de desenvolvimento, foi implementado o código baseado no que foi planejado nas fases de levantamento de requisitos e na prototipagem. E, em cada iteração foram discutidas quais funcionalidades foram feitas de acordo com o que foi planejado e quais funcionalidades seriam entregues na próxima iteração. Na última fase, foi realizada a implantação - fase onde a aplicação se torna disponível para o usuário, sendo possível ser acessado por qualquer pessoa pelo link da aplicação[1].

[1] A aplicação está disponível em: http://pranadar.herokuapp.com/

4 ARQUITETURA DO SISTEMA

O sistema é composto por três componentes gerais, sendo eles a plataforma Web, a API e o banco de dados. Assim, a plataforma Web, consome a *Application Programming Interface (API,* em português, Interface de Programação de Aplicativos*)* que está conectada com o banco de dados, como pode ser visto no diagrama de componentes no Apêndice E.

Os usuários do sistema e suas ações foram baseadas nos diagramas de caso de uso que foram elaborados, considerando as necessidades que o cliente tinha em vista. Sendo assim, foi feito um caso de uso para cada usuário, como pode ser visto na Seção 2.

O sistema utiliza a arquitetura cliente-servidor, na qual o cliente faz a requisição dos dados para o servidor e o servidor retorna uma resposta. Levando isso em consideração, foi utilizado o *Hypertext Transfer Protocol* (*HTTP*) para a aplicação se comunicar com a *API.* O HTTP apresenta alguns métodos para realizar a comunicação entre o cliente e o servidor, permitindo a criação, visualização, atualização e exclusão de dados com uso dos métodos POST, GET, PUT e DELETE, respectivamente. Com base nisso, a biblioteca Axios foi utilizada para trabalhar com as requisições *HTTP* (CANAL TI, 2018).

5 DESCRIÇÃO DO PROJETO

Neste tópico, será abordado mais detalhadamente sobre a aplicação desenvolvida, falando sobre as tecnologias utilizadas, testes e a estrutura de arquivos da aplicação.

A aplicação apresenta quatro tipos de usuários: Administrador, Secretário, Instrutor e Aluno. Cada um deles tem interfaces para o gerenciamento de suas atividades específicas. Além disso, pode-se definir os objetivos individuais de cada usuário.

O administrador do sistema tem o direito de controlar todas as funcionalidades do sistema, como gerenciar pagamentos, vínculos, secretários, alunos, instrutores, turmas e alterar a senha dos usuários para a senha padrão. O diagrama de caso de uso do administrador pode ser visto no Apêndice A. Já o secretário tem um comportamento bem similar ao administrador, exceto realizar tarefas de alterações de senhas de outros usuários e gerenciamento de outros secretários. O diagrama de caso de uso de secretário está no Apêndice B. No diagrama de caso de uso de instrutor (Apêndice - C), é visto que o instrutor tem o controle de quais turmas está designado a ministrar aulas, e com isso, possuindo o direito de gerenciar os planos de aula e as frequências de cada plano. O diagrama de caso de uso de aluno (Apêndice - D), ilustra que até então, o aluno é responsável apenas por visualizar seu perfil e alterar sua senha.

5.1 TECNOLOGIAS UTILIZADAS

Levando em consideração que o projeto tem foco no *front-end,* a aplicação utiliza React, sendo ela uma biblioteca Javascript para criação de interfaces de usuário desenvolvida pelo Facebook. React utiliza o conceito de aplicação *single-page*, e, segundo a Meta Platforms, Inc. (2022a), “... é uma aplicação que carrega uma única página HTML e todos os assets (como JavaScript e CSS) necessários para a aplicação ser executada. Quaisquer interações com a página ou

páginas subsequentes não necessitam de outras requisições para o servidor, o que significa que a página não é recarregada".

A base da implementação foi criada usando *Create React App* (*CRA*) com o template da linguagem Typescript - linguagem de programação com tipagem forte construída em cima do Javascript (MICROSOFT, 2022a). Considerando isso, o projeto criado com *CRA* já vem com uma estrutura de pastas, sendo possível a execução imediata, sem a necessidade de configurações (META PLATFORMS, INC, 2022c). Outro detalhe a ser levado em conta, é que foi usado o gerenciador de pacotes Yarn. O Yarn é gerenciador de pacotes que instala os pacotes em paralelo de forma rápida e segura, além de ser possível compartilhar e usar códigos de todo mundo (YARN, 2022).

Em relação aos componentes personalizados, foram utilizadas as bibliotecas React Bootstrap e Styled Components. A React Bootstrap tem como papel principal prover uma variedade de componentes para serem customizados, visando a facilidade de não necessitar de outras ferramentas (REACT-BOOTSTRAP, 2022). Já a Styled Components, tem como objetivo personalizar componentes usando Javascript e Cascading Style Sheets (CSS) de forma simples, melhorando tanto a experiência do desenvolvedor quanto do usuário final (MADDERN et al., 20--).

Para realizar as requisições HTTP à API, foi utilizado o Axios. O Axios é um cliente HTTP baseado-em-promessas para o NodeJS e para o navegador. É isomórfico, podendo rodar no navegador e no NodeJS com a mesma base de código (AXIOS, 202-?). Em conjunto com o Axios, foi utilizada a React Query para realizar a busca de dados na API, armazenar os dados necessários em cache e manter os dados atualizados de maneira rápida e simples. Essa biblioteca fornece alguns hooks que contribuem para realizar as funcionalidades que foram citadas (LINSLEY, 2022).

O editor de código Visual Studio Code foi utilizado durante o desenvolvimento, tendo em vista que nele podem ser encontradas diversas extensões que ajudam tanto no desenvolvimento, quanto na produtividade (MICROSOFT, 202?b). Com

isso, uma das extensões que foram usadas no desenvolvimento, contribui com o uso do Git - sistema distribuído open-source para controle de versão de arquivos (SOFTWARE FREEDOM CONSERVANCY, 2022). Além disso, foi utilizada a plataforma Github para hospedar o código-fonte com os arquivos versionados com git (GITHUB, 2022).

5.2 TESTES

Para certificar que as funcionalidades foram desenvolvidas de forma correta e que estão ocorrendo de forma esperada para o usuário, foram realizados alguns testes manuais e testes unitários. Os testes manuais foram feitos de acordo com o que foi pedido pelo cliente, como por exemplo, realizar o cadastro de alguma turma dentro do sistema como um secretário.

Segundo Noleto (2020):

> ... teste unitário é uma verificação feita com uma pequena porção de código, uma unidade de um software. [...] No unitário, cada parte do sistema ganha uma atenção devida e detalhada, de modo a otimizar o processo de identificação de erros. O objetivo é ajudar a rastrear os bugs e impedir que eles retornem depois que alterações forem feitas no produto.

Para a realização desses testes, foi feito o uso de algumas bibliotecas que contribuem nesse quesito, sendo elas React Testing Library, Jest e Mock Service Worker. As descrições dessas bibliotecas podem ser vistas abaixo:

- Jest: é um framework de testes rápido e seguro, desenvolvido em Javascript e com foco na simplicidade. Nele podem ser gerados relatórios de cobertura de testes e analisar testes de forma intuitiva (JEST, 2022).
- React Testing Library: é uma biblioteca para a testagem de componentes React, sendo considerada uma solução leve. Ela é estruturada em cima da DOM (da sigla, *Document Object Model*) Testing Library - biblioteca usada para testagem de nós do DOM (MCCURDY, 2021).

- Mock Service Worker: É uma biblioteca de imitação de API que usa a Service Worker API para interceptar as requisições que estão sendo feitas (ZAKHARCHENKO, [entre 2018 e 2022]).

Com base nisso, pode ser visto na Figura 1 um teste unitário da tela de vinculação de usuários, utilizando React Testing Library. Neste código, verifica-se que inicialmente as importações necessárias; na linha 6 tem a descrição daquele escopo de teste; na linha 7 tem o ínicio do teste unitário e sua descrição; na linha 8 tem a renderização da página; na linha 10 ocorre o clique no botão de trocar visualização; e na linha 12, existe a asserção do que era esperado no teste, no caso, se o modo de visualização foi alterado.

Figura 1 - Teste da página de vinculação de usuários

```
1  import { screen } from '@testing-library/react';
2  import { renderApp } from '../../testUtils';
3  import userEvent from '@testing-library/user-event';
4  import BindUsersPage from '.';
5
6  describe('Bind users page', () => {
7      test('It changes view mode to bind instructor', async () => {
8          renderApp(<BindUsersPage />);
9
10         userEvent.click(await screen.findByText('Trocar'));
11
12         expect(await screen.findByText('Vincular instrutor')).toBeInTheDocument();
13     });
14 });
15
```

Fonte: Autoria própria (2022)

Para contribuir na confiabilidade, os campos de formulários foram validados de acordo com o tipo de dado que está sendo inserido. Para contribuir nesse processo, foram utilizadas expressões regulares para validar alguns campos, como CPF e RG.

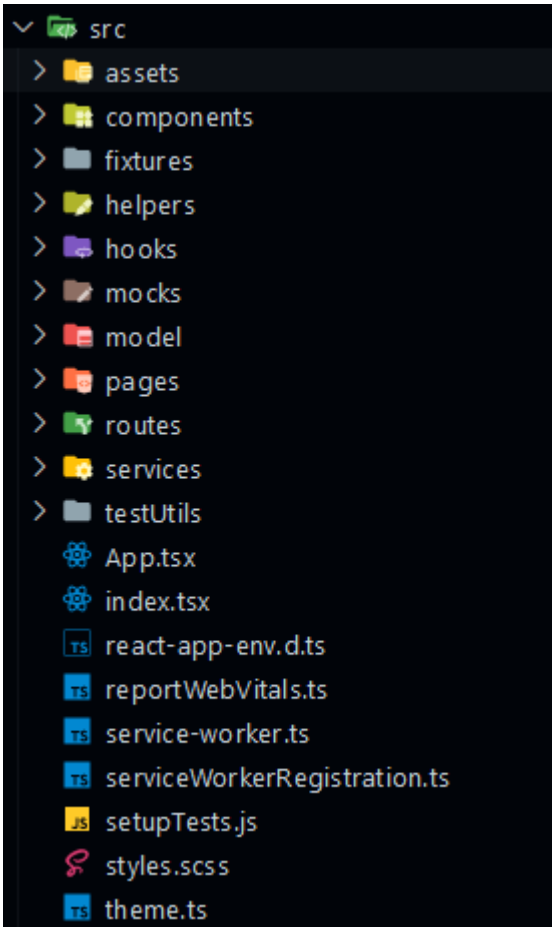
5.3 ESTRUTURA DE ARQUIVOS

Levando em consideração que o projeto já vem com uma estrutura inicial de pastas, foi organizado o código fonte dentro da pasta raiz do projeto (src) - diretório que armazena o código fonte, encontrado no diretório raiz. A estrutura de pasta utilizada é descrita da seguinte forma:

- Components**:** Separar componentes de códigos que podem se repetir dentro do código. Contribuindo assim, com a legibilidade e organização do projeto (META PLATFORMS, INC., 2022b);
- Models**:** Facilitar a compreensão e organização dos dados do sistema;
- Pages**:** Organizar a estrutura de páginas da aplicação. Uma página é um conjunto de componentes integrados para ser mostrado ao usuário;
- Hooks**:** Reunir os custom hooks que foram criados para serem utilizados na aplicação. Um custom hook é a utilização de um hook com a adição de novas funcionalidades que não existiam antes (META PLATFORMS, INC., 2022b);
- Routes**:** Reunir as rotas da aplicação. Para isso, foi utilizado a React Router, que é uma coleção de componentes de navegação, ou seja, com ela é possível manipular de diversas maneiras as rotas e a navegação da aplicação (REACT TRAINING, 2022);
- Services**:** Organizar as chamadas da API utilizando Axios. Nesses arquivos, eram separados em funções as chamadas de cada método HTTP, sendo uma função específica para cada rota da API;

Na Figura 2, é ilustrado a pasta raiz do projeto com a estrutura de diretórios citados anteriormente.

Figura 2 - Estrutura de diretórios da pasta src

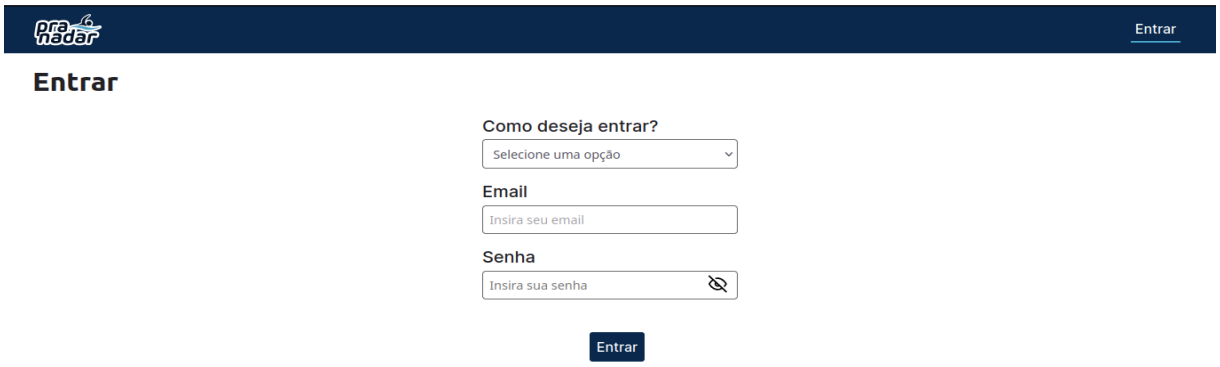


Fonte: Autoria própria (2022)

6 RESULTADOS

Na tela de *login* ilustrada na Figura 3, os usuários podem inserir seus dados (tipo de usuário, email e senha) e entrar no sistema.

Figura 3 - Tela de *login*



Fonte: Autoria própria (2022)

Na tela de visualização de alunos ilustrada na Figura 4, pode ser visto uma tabela com o campo de situação (o círculo verde diz que o usuário está ativo, o círculo vermelho significa inativo e o triângulo com exclamação sinaliza que aluno está com a contribuição atrasada), o nome do aluno, a categoria, se ele é contribuinte ou não, e os botões de ver detalhes e excluir aluno. Além disso, existe o botão de criar um novo aluno, que redireciona para uma nova página.

Na tela de detalhes do aluno ilustrada na Figura 5, são mostrados os dados de um aluno de acordo com aqueles que foram preenchidos na criação. Como o usuário que está acessando o sistema é o administrador, podem ser vistas opções como redefinir senha, pagamentos, vínculos, anexar arquivos e editar.

Na Figura 6, há a possibilidade de editar os dados do aluno em questão. Em alguns campos, como CPF e RG, foram usadas máscaras para melhorar o tratamento de dados e a usabilidade.

Figura 4 - Tela de visualização de alunos

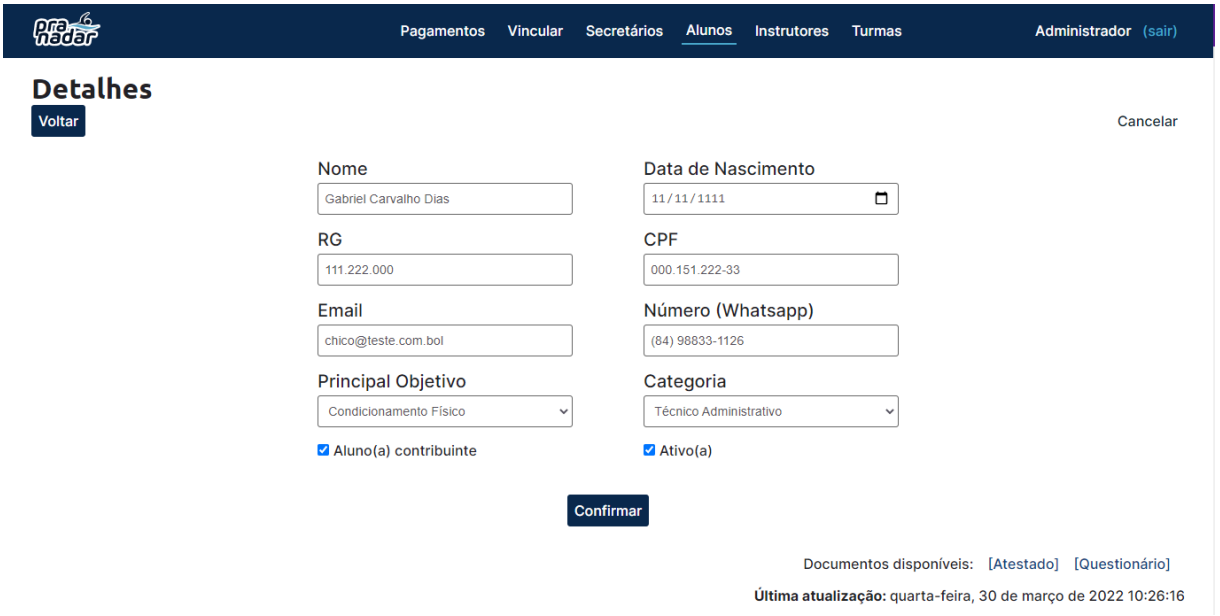
Fonte: Autoria própria (2022)

Figura 5 - Tela de detalhes do aluno

Fonte: Autoria própria (2022)

Na tela de visualização de turmas (Figura 7), pode ser observado a tabela que tem o campo de situação (ativo e inativo); o nome da turma; a modalidade da turma; dias da semana que ocorrem as aulas; os horários que ocorrem as aulas (horário de início e de término); e os botões de visualizar detalhes da turma e o de exclusão de turma.

Figura 6 - Tela de detalhes de aluno com edição ativa



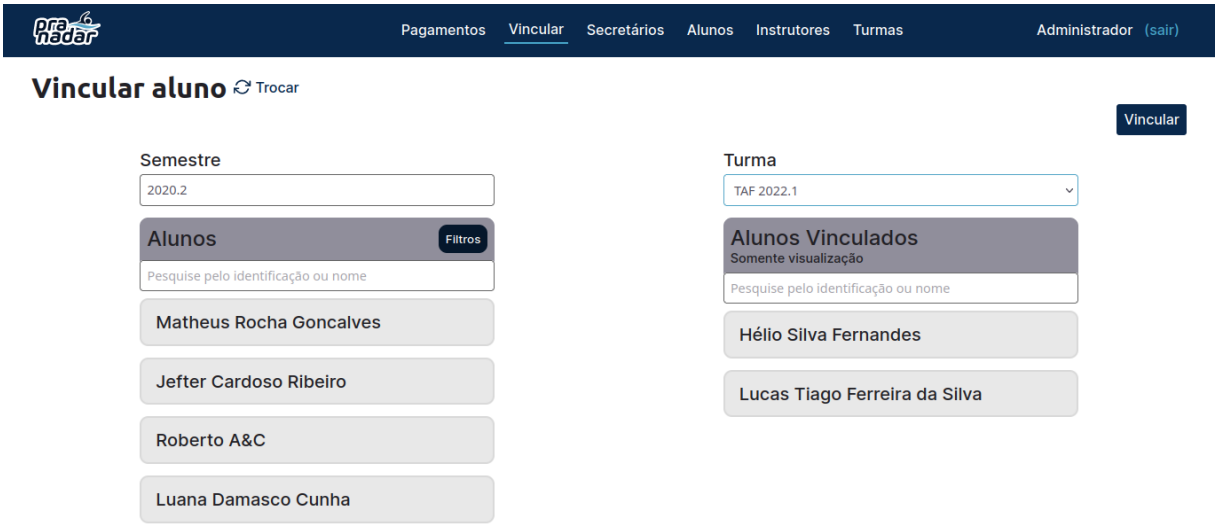
Fonte: Autoria própria (2022)

Figura 7 - Tela de visualização de turmas

Fonte: Autoria própria (2022)

Na tela de vinculação de usuários (Figura 8), é possível vincular alunos e instrutores, e para isso, utilizar o botão de trocar para alternar entre as visualizações. Para vincular um usuário, é necessário preencher o semestre daquele vínculo e qual o aluno e turma que vai ser vinculado. Além disso, quando tem uma turma selecionada, é possível ver os vínculos já existentes.

Figura 8 - Tela de vinculação de usuários



Fonte: Autoria própria (2022)

Na tela de planos de aula (Figura 9), possui uma tabela de listar os planos de aula, com os campos de data; objetivo; botão de detalhe e outro de exclusão. No canto superior direito, há botões que torna possível navegar para a tela de detalhes da turma que está relacionada com esses planos de aula e para o formulário de criação dos mesmos.

Figura 9 - Tela de planos de aula

Fonte: Autoria própria (2022)

Na tela de frequência (Figura 10), é listada a frequência dos alunos que estão vinculados com aquela turma específica. Sendo possível visualizar uma tabela com os campos de nome do aluno; a situação (presente ou ausente) e botão de alterar situação.

Figura 10 - Tela de frequência

Fonte: Autoria própria (2022)

7 CONCLUSÃO

Os sistemas de gestão de informações são usados em diversas áreas e situações para solucionar problemas recorrentes, contribuindo para facilitar o processo de gerenciamento de dados.

Como pôde ser observado, este trabalho teve como foco a aplicação *front-end* Web do sistema Pra Nadar, sendo este, destinado para a gestão do programa da FAEF. A aplicação apresenta fácil navegação entre as páginas, além de permitir, por exemplo, o gerenciamento das turmas, pagamentos, aulas e usuários. E com isso, conclui-se que o sistema automatizou as tarefas com sucesso, atendendo às expectativas. A partir de Abril de 2022, a versão de produção já está disponível.

8 PERSPECTIVAS FUTURAS

Como perspectivas futuras, podem ser feitas algumas melhorias de código visando a sua qualidade - permitindo maior legibilidade e manutenibilidade; adicionar novas funcionalidades como gerar relatório do sistema em arquivo e gerenciar as avaliações dos alunos; contribuir com o aumento da cobertura de testes unitários e melhorar a autenticação.

REFERÊNCIAS

AXIOS. **Introdução**: Cliente HTTP baseado em promessas para o navegador e Node.js. [*S. l.*], 202-?. Disponível em: https://axios-http.com/ptbr/docs/intro. Acesso em: 4 abr. 2022.

CANAL TI. **Arquitetura cliente-servidor**. [*S. l.*], 27 set. 2018. Disponível em: https://www.canalti.com.br/arquitetura-de-computadores/arquitetura-cliente-servidor/. Acesso em: 5 abr. 2022.

COIMBRA, ?. **Agile Methods: Feature Driven Development (FDD)**. [*S. l.*], 19 out. 2020. Disponível em: https://projetoseti.com.br/agie-methods-feature-driven-development-fdd/. Acesso em: 30 mar. 2022.

GITHUB. **About**. [*S. l.*], 2022. Disponível em: https://github.com/about. Acesso em: 4 de abr. 2022.

JEST. **Jest é um poderoso Framework de Testes em JavaScript com um foco na simplicidade**. [*S. l.*], 2022. Disponível em: https://jestjs.io/pt-BR/. Acesso em: 5 abr. 2022.

LINSLEY, Tanner. **Overview**. [*S. l.*], 2022. Disponível em: https://react-query.tanstack.com/overview. Acesso em: 5 abr. 2022.

MADDERN, Glen; JACOBS , Evan; PLUCKTHUN, Phil; STOIBER, Max. **Getting started**. [*S. l.*], 20--. Disponível em: https://styled-components.com/. Acesso em: 6 abr. 2022.

MCCURDY, Nick. **React Testing Library**. [*S. l.*], 21 jul. 2021. Disponível em: https://testing-library.com/docs/react-testing-library/intro/. Acesso em: 31 mar. 2022.

META PLATFORMS, INC. **Introdução**. [*S. l.*], 2022a. Disponível em: https://pt-br.reactjs.org/docs/getting-started.html. Acesso em: 1 abr. 2022.

META PLATFORMS, INC. **Glossário de Termos React**. [*S. l.*], 2022b. Disponível em: https://pt-br.reactjs.org/docs/glossary.html. Acesso em: 3 abr. 2022.

META PLATFORMS, INC. **Getting Started**. [*S. l.*], 2022c. Disponível em: https://create-react-app.dev/docs/getting-started/. Acesso em: 1 abr. 2022.

MICROSOFT. **TypeScript is JavaScript with syntax for types**. [*S. l.*], 2022a. Disponível em: https://www.typescriptlang.org/. Acesso em: 2 abr. 2022.

MICROSOFT. **Getting Started.** [S. l.], [202?b]. Disponível em: https://code.visualstudio.com/docs. Acesso em: 29 mar. 2022.

NOLETO, Caio. **Teste unitário: o que são, por que usar e por onde começar?**. [*S. l.*], 23 jul. 2020. Disponível em: https://blog.betrybe.com/tecnologia/testes-unitarios/. Acesso em: 4 abr. 2022.

PRESSMAN, Roger S. **Engenharia de Software**. 3. ed. rev. [*S. l.*]: Makron books, 1995.

REACT-BOOTSTRAP. **Introduction**. [*S. l.*], 202-?. Disponível em: https://react-bootstrap.github.io/getting-started/introduction. Acesso em: 5 abr. 2022.

REACT TRAINING. **React Router**: React Training Learn once, Route anywhere. [*S. l.*], 2022. Disponível em: https://v5.reactrouter.com/. Acesso em: 4 abr. 2022.

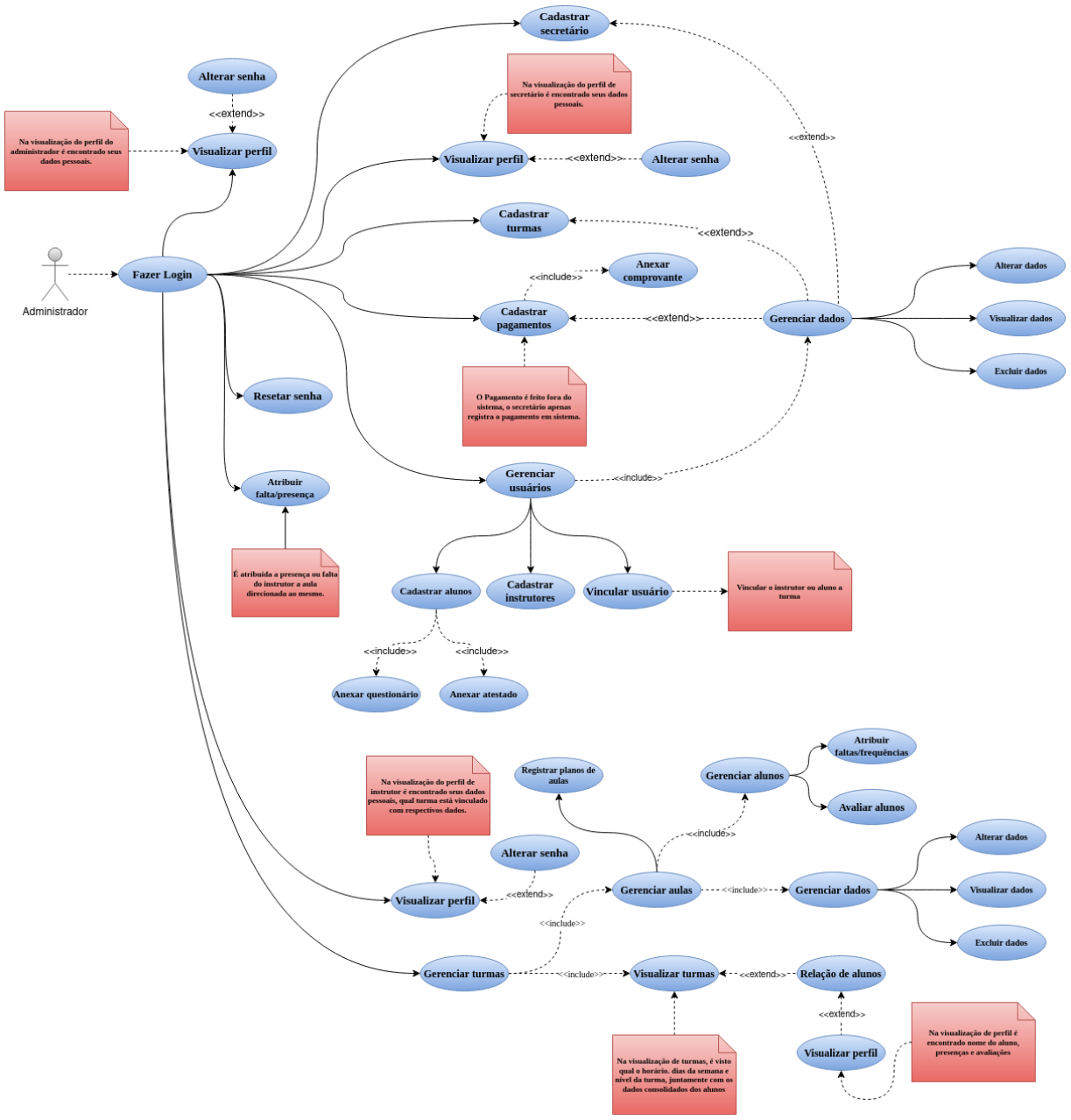
ROCHA, Fabio Gomes. **Introdução ao FDD - Feature Driven Development.** [S. l.], 2013. Disponível em: https://www.devmedia.com.br/introducao-ao-fdd-feature-driven-development/27971. Acesso em: 29 mar. 2022.

SOFTWARE FREEDOM CONSERVANCY. **About**. [*S. l.*], 2022. Disponível em: https://git-scm.com/about. Acesso em: 2 abr. 2022.

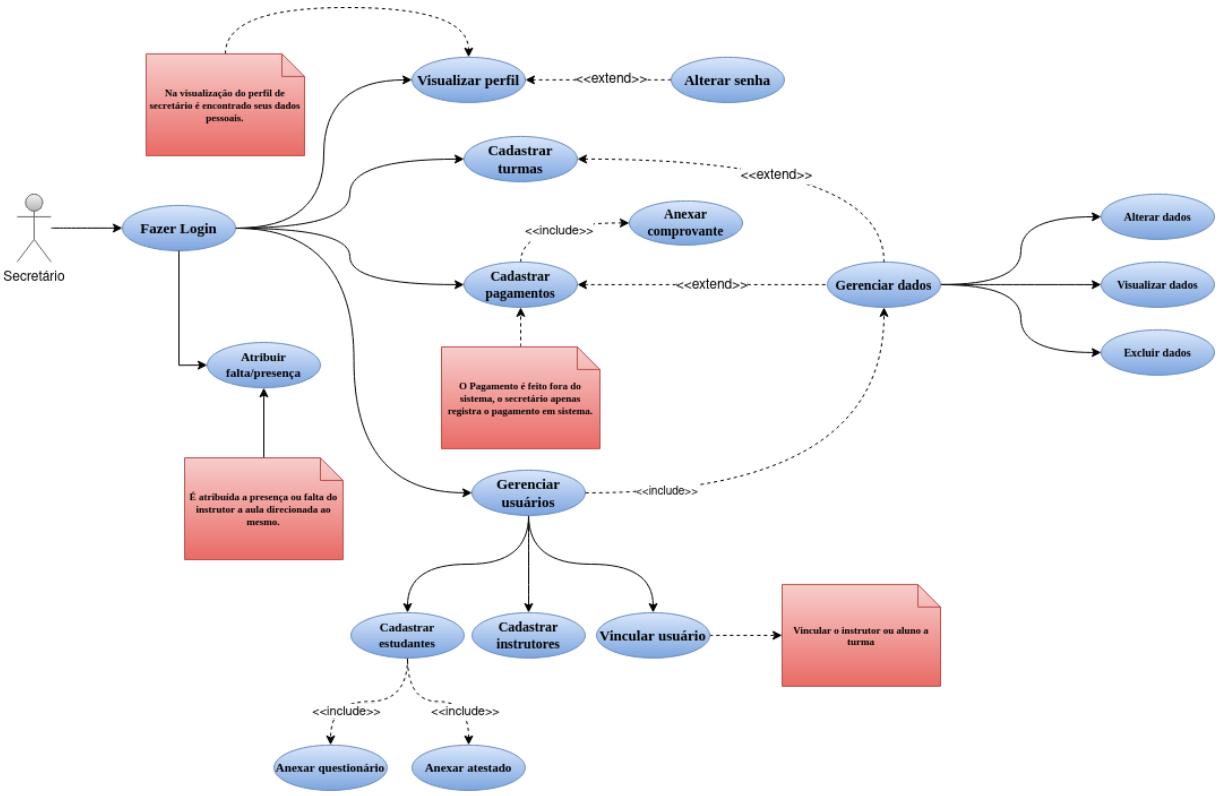
YARN. **Safe, stable, reproducible projects**. [*S. l.*], 2022. Disponível em: https://yarnpkg.com/. Acesso em: 4 abr. 2022.

ZAKHARCHENKO, Artem. **Introduction**. [*S. l.*], [entre 2018 e 2022]. Disponível em: https://mswjs.io/docs/. Acesso em: 31 mar. 2022.

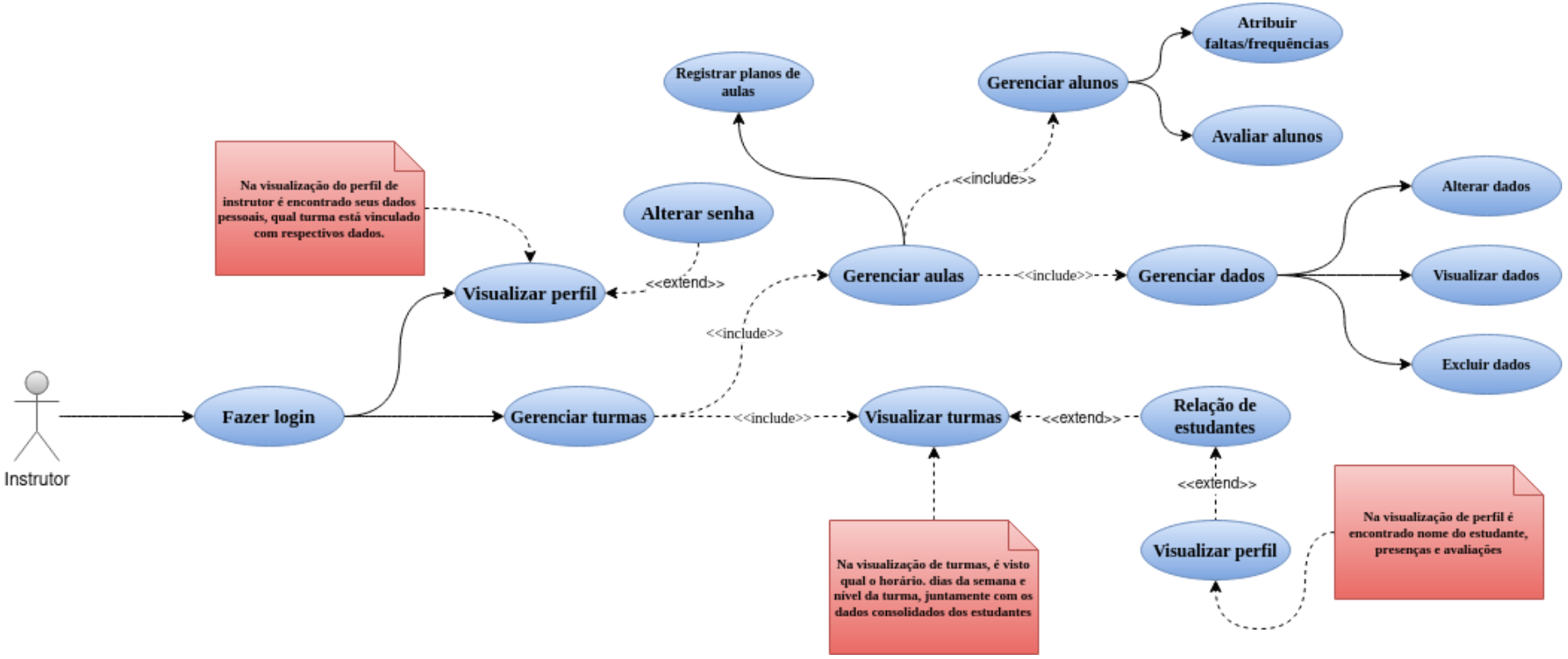
APÊNDICE A - CASO DE USO DE ADMINISTRADOR



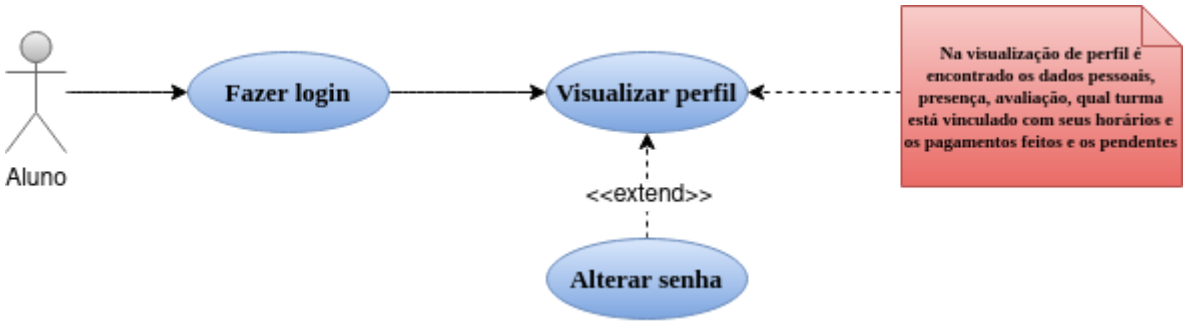
APÊNDICE B - CASO DE USO DE SECRETÁRIO



APÊNDICE C - CASO DE USO DE INSTRUTOR



APÊNDICE D - CASO DE USO DE ALUNO



APÊNDICE E - DIAGRAMA DE COMPONENTES