UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

COLEGIADO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Sophie Dilhon Gama

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Next.js

Vitória, ES

2024

Sophie Dilhon Gama

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Next.js

Monografia apresentada ao Curso de Ciência da Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Colegiado do Curso de Ciência da Computação

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2024

Sophie Dilhon Gama
Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Next.js/ Sophie Dilhon Gama. – Vitória, ES, 2024-
52 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vítor E. Silva Souza

Monografia (PG) – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Colegiado do Curso de Ciência da Computação, 2024.

1. Engenharia Web. 2. FrameWeb. 3. Next.js. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Next.js

CDU 02:141:005.7

Sophie Dilhon Gama

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Next.js

Monografia apresentada ao Curso de Ciência da Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Vitória, ES, 05 de Julho de 2024:

Prof. Dr. Vítor E. Silva Souza
Orientador

Prof. Dra. Patrícia Dockhorn Costa
Universidade Federal do Espírito Santo

Clarete Aparecida Diniz Gomes
Universidade Federal do Espírito Santo

Vitória, ES
2024

Resumo

Com o rápido crescimento da Internet, a demanda por sistemas *Web* complexos e robustos também aumentou, destacando a necessidade de métodos e modelos especializados para desenvolver, implementar e manter esses sistemas. A Engenharia *Web* surgiu como uma resposta a essa demanda, funcionando como uma forma de adaptação dos conhecimentos da Engenharia de Software para o desenvolvimento de sistemas *Web*.

O método FrameWeb (*Framework-based Design Method for Web Engineering*) foi proposto nesse contexto, oferecendo uma abordagem de modelagem que voltada ao desenvolvimento utilizando *frameworks Web*, com o intuito de tornar esse processo mais rápido e eficiente. O FrameWeb define uma arquitetura padrão para facilitar a integração com *frameworks* de desenvolvimento *Web*, e por isso tem sido utilizado em diversos trabalhos com o foco em avaliá-lo com os diferentes *frameworks* disponíveis no mercado.

Este trabalho tem como objetivo aplicar o método FrameWeb novamente no desenvolvimento do SCAP (Sistema de Controle de Afastamentos de Professores), porém utilizando desta vez o *framework* Next.js. De modo a contribuir com o estudo que tem sido feito, e avaliar desta vez a robustez do modelo quando utilizado um *framework* SPA (*Single Page Application*).

Palavras-chaves: Engenharia Web. FrameWeb. Next.js. Typescript.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

FrameWeb	Framework-based Design Method for Web Engineering
UML	Unified Modeling Language
SCAP	Sistema de Controle de Afastamento de Professores
SPA	Single Page Application
DOM	Document Object Model
ORM	Object Relational Mapping
MVC	Model-View-Controller
OO	Orientado a Objeto

Sumário

1 Introdução

O advento da Internet e da *World Wide Web* (WWW), na década de 80, trouxe uma nova forma de comunicação e interação entre as pessoas. O crescimento da *Web* ocorreu de forma tão rápida que, em pouco tempo, ela se tornou uma plataforma essencial para os negócios, comércios e outros setores da sociedade. Nesse contexto, emerge na *Web* uma variedade de aplicações complexas e de grande porte, os chamados *WebApps*, que no entanto, não eram desenvolvidas com o apoio de metodologias e processos bem definidos (MURUGESAN et al., 2001).

Surge então a necessidade de adaptar métodos da Engenharia de Software ao desenvolvimento das *WebApps*, de forma a construir soluções eficazes e garantir a qualidade e a manutenibilidade dos sistemas (BEDER, 2017). A Engenharia *Web* pode ser definida como o uso de princípios científicos, de engenharia, de gerência e abordagens sistemáticas para o desenvolvimento, implantação e manutenção de aplicações *Web* de alta qualidade (MURUGESAN et al., 2001).

Ao final dos anos 90 a ideia de *frameworks Web* começou a ser popularizada, reunindo várias bibliotecas úteis para desenvolvimento *Web* em uma única *stack* de *software* para os desenvolvedores utilizarem e agilizarem o processo de desenvolvimento. Muitos dos *frameworks Web* são baseados na arquitetura Model View Controller (MVC), alguns exemplos são o Ruby on Rails,[1] Django,[2] Laravel,[3] Spring MVC,[4] entre outros.

Apesar do crescimento no uso dos *frameworks Web*, não havia um método da Engenharia *Web* voltado exclusivamente para o desenvolvimento de sistemas que os utilizassem, nesse contexto, em sua tese de mestrado, Souza (2007) propôs o FrameWeb, método para o projeto de sistemas de informação *Web* baseado em *frameworks*. Objetivando o aumento da produtividade da equipe de desenvolvimento.

O método FrameWeb vem sendo evoluído nos últimos anos e, como parte de sua avaliação, diferentes implementações de um mesmo sistema de informação Web chamado SCAP (Sistema de Controle de Afastamento de Professores) — uma *WebApp* que auxilia o controle de afastamento de professores do Departamento de Informática (DI) pela Internet — foram desenvolvidas, aplicando-se o método. Neste trabalho, será aplicado o método FrameWeb em uma nova implementação do SCAP, utilizando a linguagem *TypeScript* e o *framework* Next.js que, diferente da grande maioria dos trabalhos anteriores, não é um *framework* baseado em MVC.

[1] <https://rubyonrails.org>
[2] <https://www.djangoproject.com>
[3] <https://laravel.com>
[4] <https://spring.io>

1.1 Motivação e Justificativa

Em seu trabalho de conclusão de curso, Duarte (2014) desenvolveu a primeira implementação do SCAP, utilizando o método FrameWeb e os *frameworks* da plataforma Java EE 7, com o objetivo de avaliar o método com uma gama maior de *frameworks* e tecnologias, em relação à proposta original do método (SOUZA, 2007). Em trabalhos seguintes, o método foi avaliado com outros diferentes *frameworks*, como o VRaptor 4 (PRADO, 2015), Symfony (BERGER, 2021), Angular (GOMES, 2022), etc.

Essas avaliações contribuem com a evolução do método FrameWeb de várias maneiras. Elas permitem identificar limitações e desafios específicos ao utilizar diferentes *frameworks*, o que pode levar à proposição de melhorias ou adaptações no método. Por exemplo, em sua implementação, Duarte (2014) identificou que o método não apresentava o nível de granularidade necessário em relação às páginas web e sugeriu a criação de novos estereótipos. Já Dalapicola (2021) e Gomes (2022) identificaram que o método precisava de alterações nos modelos de persistência e apresentação, quando utilizada uma linguagem sem tipagem, como o *JavaScript*.

Seguindo essa mesma linha, este trabalho tem como motivação avaliar o método FrameWeb com um novo *framework*, o Next.js, que, assim como Angular, é um *framework* SPA (*Single Page Application*), com crescente popularidade no mercado. Através dessa avaliação, pretende-se verificar a compatibilidade do FrameWeb com as características específicas do Next.js. Isso pode revelar novos *insights* sobre a adequação do método e contribuir com sugestões de aprimoramento que atendam melhor às necessidades atuais do desenvolvimento *web*.

1.2 Objetivos

Este trabalho tem como objetivo geral aplicar o método FrameWeb (SOUZA, 2007) em uma nova implementação do SCAP (Sistema de Controle de Afastamento de Professores), baseando-se nos requisitos levantados por Duarte (2014) e Prado (2015) e utilizando o *framework* Next.js, de forma a contribuir com a análise do método FrameWeb, assim como em sua evolução.

Para isso, faz-se necessária a definição de objetivos específicos que, juntos, auxiliam na conclusão do objetivo geral, sendo eles:

- Compreender o método FrameWeb;
- Analisar os requisitos da aplicação SCAP;
- Implementar a aplicação SCAP utilizando o *framework* Next.js e o método FrameWeb.

- Avaliar a aplicação do método FrameWeb no desenvolvimento do SCAP, e propor melhorias caso necessário.

1.3 Método de Desenvolvimento do Trabalho

Para que os objetivos apresentados na seção anterior sejam satisfeitos, os seguintes passos devem ser seguidos:

- Conduzir estudo abrangente da bibliografia disponível sobre o FrameWeb (SOUZA, 2007; SOUZA, 2020);
- Revisar e compreender os requisitos da aplicação SCAP levantados por Duarte (2014) e Prado (2015);
- Estudar padrões de arquitetura, em específico MVC e MVVM, assim como os *frameworks* baseados em tais padrões, e *frameworks* SPA;
- Elaborar os modelos FrameWeb e gerar o Documento de Projeto, considerando o *framework* escolhido para a implementação;
- Implementar o sistema SCAP, a partir dos requisitos já levantados e do Documento de Projeto gerado, utilizando o *framework* Next.js;
- Redigir a monografia utilizando o *template* abnTeX[5] para a escrita em LaTeX[6] seguindo os requisitos das normas da ABNT (Associação Brasileira de Normas Técnicas).

1.4 Organização da Monografia

Além desta introdução, esta monografia é composta por outros cinco capítulos:

- O Capítulo 2 apresenta os aspectos relativos ao conteúdo teórico relevante para o trabalho;
- No Capítulo 3 é descrito o sistema SCAP e seus requisitos;
- O Capítulo 4 apresenta os modelos FrameWeb desenvolvidos e as tecnologias utilizadas na implementação, além de mostrar os resultados obtidos;
- O Capítulo 5, por fim, discute os resultados obtidos e apresenta as conclusões do trabalho.

[5] <https://www.abntex.net.br/>
[6] <https://www.latex-project.org/>

2 Referencial Teórico e Tecnologias Utilizadas

Neste capítulo serão apresentados os referenciais teóricos utilizados para o desenvolvimento deste trabalho. A Seção 2.1 aborda os conceitos da Engenharia de Software, em seguida, o método FrameWeb é explicado na Seção 2.2 e, por fim, as categorias de *frameworks* suportadas pelo método, assim como o *framework* utilizado neste trabalho são apresentados na Seção 2.3.

2.1 Engenharia de Software

A Engenharia de Software é uma área da computação que se preocupa com todo o ciclo de vida do software, desde a especificação, desenvolvimento e até manutenção de sistemas de software (SOMMERVILLE, 2011). Por meio da aplicação de métodos e ferramentas que possibilitam a construção de sistemas complexos, tem como objetivo gerar produtos dentro do prazo e com qualidade (PRESSMAN, 2011). Segundo Sommerville (2011), a Engenharia de Software define quatro atividades essenciais para todos os processos de software, são elas:

1. **Especificação de software**: são definidas as funcionalidades do sistema, a partir da comunicação cliente e engenheiro;
2. **Projeto e implementação de software**: são definidos os modelos arquiteturais e como o sistema é implementado;
3. **Validação de software**: o software deve ser validado para garantir que os requisitos estejam contemplados;
4. **Evolução de software**: são feitas adaptações e evoluções no sistema para atender as necessidades do cliente.

2.1.1 Engenharia de Requisitos

Como visto anteriormente, é na fase de Especificação de Software que as funcionalidades e restrições do sistema são definidas, e é fundamental que os softwares contemplem os requisitos estabelecidos para garantir um desempenho satisfatório no suporte aos processos de negócios. Dessa forma, uma importante tarefa no desenvolvimento de software é a identificação dos requisitos dos negócios que os sistemas vão apoiar (FALBO, 2017). É neste contexto que entra a Engenharia de Requisitos, ação de Engenharia de Software que ocorre durante as atividades de comunicação e modelagem (PRESSMAN, 2011), responsável por

descobrir, analisar, documentar e verificar esses serviços e restrições (SOMMERVILLE, 2011).

Sommerville (2011) classifica requisitos em duas categorias, sendo elas:

- **Requisitos funcionais**: descrevem as funcionalidades que o sistema deve fornecer, ou seja, como o sistema deve se comportar para determinadas entradas;
- **Requisitos não funcionais**: descrevem restrições sobre os serviços ou funções oferecidas pelo sistema, como por exemplo, desempenho, confiabilidade e disponibilidade.

Além dos requisitos funcionais e não funcionais, é importante dar destaque às Regras de Negócio, requisitos provenientes do domínio de aplicação do sistema que refletem características e restrições do mesmo (SOMMERVILLE, 2011; FALBO, 2014).

2.1.2 Projeto de Sistemas

A fase de Projeto de Sistemas ocorre após a Especificação de Requisitos e tem como objetivo incorporar a tecnologia aos requisitos funcionais e não funcionais definidos anteriormente, projetando o que será construído na implementação. Assim, é necessário conhecer as tecnologias disponíveis e as facilidades do ambiente de software no qual o sistema será implementado, uma vez que essas informações não eram consideradas na etapa anterior (FALBO, 2014; PRESSMAN, 2011).

Inicialmente, o projeto é representado em um nível alto de abstração e, à medida que o trabalho avança, os refinamentos conduzem as representações de menores níveis de abstração (FALBO, 2014). Dessa forma, o resultado do processo de projeto é um modelo de arquitetura que descreve como o sistema está organizado em um conjunto de componentes de comunicação (SOMMERVILLE, 2011).

2.1.3 Engenharia *Web*

Com o crescimento da Internet, foi significativo o impacto que a mesma causou em diversos setores da economia, comércios utilizando sites para realizar suas vendas, indústrias com sistemas para gerenciar seus processos e até mesmo em nossas vidas pessoais (MURUGESAN et al., 2001). A rápida necessidade de sistemas complexos deixa de lado a preocupação por qualidade a longo prazo, surgindo a chamada "Crise *Web*" (MURUGESAN et al., 2001), uma variação potencialmente mais séria da conhecida "Crise de Software" (GIBBS, 1994). É importante ressaltar que, embora os sistemas *Web* sejam softwares, eles possuem características e requisitos exclusivos (PRESSMAN, 2011), o que trouxe a necessidade de uma nova área da Engenharia de Software, a Engenharia *Web*.

A Engenharia *Web* pode então ser descrita como a aplicação de princípios e métodos da Engenharia de Software, adaptados ao desenvolvimento de sistemas *Web* e suas características específicas (BEDER, 2017; MURUGESAN et al., 2001), com o objetivo de garantir a qualidade dos sistemas *Web*. Olsina, Lafuente e Rossi (2001) definem um conjunto de atributos técnicos que levam a qualidade de um sistema *Web*, são eles:

- **Usabilidade**: o sistema deve ser fácil de usar, com uma interface intuitiva e que atenda as necessidades do usuário;
- **Funcionabilidade**: o sistema deve funcionar corretamente, atendendo às características do domínio;
- **Eficiência**: o sistema deve fornecer respostas rápidas e precisas;
- **Confiabilidade**: o sistema deve ser confiável, e capaz de se recuperar de erros;
- **Manutenibilidade**: o sistema deve ser fácil de ser corrigido, adaptado e melhorado.

A Engenharia *Web* se baseia em dois conceitos para atingir a qualidade dos sistemas *Web*: Agilidade e Arcabouço de Processo. A agilidade é uma abordagem de desenvolvimento de software que se baseia em ciclos curtos de desenvolvimento, já o Arcabouço de Processo é um conjunto de atividades que devem ser realizadas ao longo de todo o processo de desenvolvimento do sistema, independente do seu tamanho e complexidade (BEDER, 2017).

2.2 FrameWeb

Com o uso de *frameworks* se tornando estado-da-prática no desenvolvimento de sistemas de informação *Web* (*Web-based Information Systems*, ou WIS), surge a necessidade de um método de projeto que trate diretamente os aspectos relacionados a esses *frameworks*. Nesse contexto, Souza (2007) propõe o FrameWeb, método de projeto para construção de WIS baseado em *frameworks* que possui como principal objetivo agilizar a fase de desenvolvimento.

O FrameWeb concentra-se na fase de Projeto de Sistema, que ocorre após o levantamento de requisitos, e define uma arquitetura lógica padrão para WISs baseada no padrão arquitetônico Camada de Serviço (FOWLER, 2002), apresentada na Figura 1. A arquitetura é dividida em três camadas, sendo elas:

- **Lógica de Apresentação**: tem o objetivo de prover a interface gráfica ao usuário e é dividida em dois pacotes. O primeiro é o pacote **Visão**, que contém as páginas *Web*, folhas de estilo, imagens, scripts que executam do lado do cliente, e outros arquivos

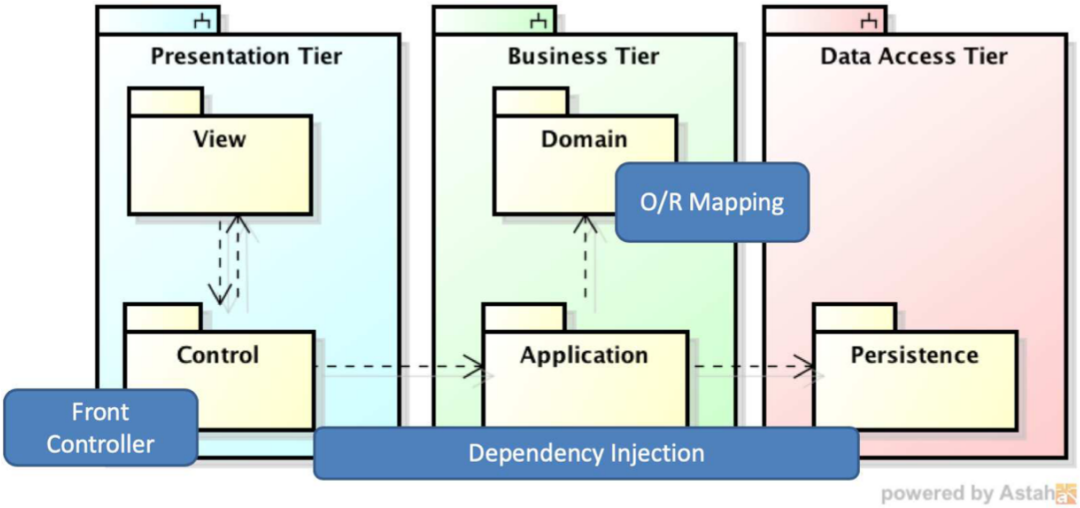

Figura 1 – Arquitetura sugerida pelo método FrameWeb (SOUZA, 2020).

relacionados exclusivamente com a exibição de informações ao usuário. O segundo é o pacote **Controle**, que envolve classes de ação que lidam com as requisições feitas pelos componentes do pacote **Visão**, utilizando a infraestrutura do *framework* Controlador Frontal e chamando serviços oferecidos pelo pacote **Aplicação**;

- **Lógica de Negócio**: tem o objetivo de prover os serviços do sistema e é dividida em dois pacotes. O primeiro é o pacote **Aplicação**, que implementa os casos de uso definidos na especificação de requisitos. O segundo é o pacote **Domínio**, que contém as classes de domínio do sistema, que representam conceitos do domínio do problema;

- **Lógica de Acesso a Dados**: tem o objetivo de prover a persistência dos dados e possui um único pacote, chamado **Persistência**. Esse pacote é responsável pelo armazenamento dos objetos em mídias de longa duração, como bancos de dados. O FrameWeb propõe ainda o uso de um *framework* ORM (*Object-Relational Mapping*) com o padrão DAO (*Data Access Object*) (ALUR; CRUPI; MALKS, 2003) para adicionar uma camada de abstração a mais na manipulação de objetos no banco de dados relacional.

Além das arquiteturas lógicas e físicas do sistema, é também na fase de Projeto de Sistema que são modelados os artefatos que serão implementados na próxima etapa. O FrameWeb define quatro modelos baseados no diagrama de classes UML (SOUZA, 2007; SOUZA, 2020):

- **Modelo de Entidade**: diagrama de classes UML representando o domínio do problema e o mapeamento dos objetos que serão persistidos. Esse modelo tem um forte papel para a implementação da Camada de Domínio;

- **Modelo de Persistência**: diagrama de classes UML que representa as classes DAO. Para cada uma das classes de domínio que será persistida, é criada uma classe DAO que define os métodos de persistência para uma dada classe, guiando assim a construção da Camada de Persistência;
- **Modelo de Navegação**: diagrama de classes UML que representa as páginas *Web* e outros elementos da Camada de Apresentação. Esse modelo auxilia o desenvolvedor a implementar os pacotes Visão e Controle uma vez que todos os atributos e fluxos de navegação baseados nos estímulos enviados pelo usuário são definidos nele;
- **Modelo de Aplicação**: diagrama de classes UML que representa as classes responsáveis pela implementação dos casos de uso, além das dependências dos pacotes Controle, Domínio e Persistência.

A proposta inicial do método FrameWeb foi feita pensando em *frameworks* MVC (SOUZA, 2007), no entanto, com a popularidade dos *frameworks* SPA, o método foi adaptado para suportar esse tipo de *framework*, que se diferencia do primeiro principalmente pelo fato de possuir elementos que se comportam tanto como *controller* quanto como *view* e por focar na interação destes (HOPPE; SOUZA, 2023).

2.3 Frameworks

Schmidt, Gokhale e Natarajan (2004) definem *frameworks* como um conjunto de artefatos de software, sejam eles classes, objetos ou componentes, que colaboram entre si para prover uma arquitetura reutilizável para aplicações de uma certa família. Dessa forma, os *frameworks* visam o reuso de soluções, o que pode reduzir o tempo de desenvolvimento e deixar a manutenção do software mais fácil (GAMMA et al., 2000).

Frameworks podem ser classificados de diversas formas, como *frameworks* MVC, *frameworks* SPA, *frameworks* ORM, entre outros. Nesta seção, serão abordados com detalhes os *frameworks* ORM e SPA, e por fim, o *framework* utilizado neste trabalho, que se encaixa nesta categoria.

2.3.1 Frameworks SPA

Frameworks SPA (*Single Page Application* ou Aplicações de Página Única) são *frameworks* que implementam aplicações *Web* executadas por uma única página, ou seja, a página é carregada apenas uma vez ao iniciar a aplicação (SCOTT, 2015). Toda a responsabilidade de renderizar a aplicação é transferida ao *browser* do cliente, o chamado *Client-Side Rendering* (CSR), ou Renderização do Lado do Cliente, e então o DOM (*Document Object Model* ou Modelo de Documento por Objetos) é manipulado via *JavaScript*

para alterar a visualização da página (SCOTT, 2015; KONSHIN, 2018). Isso faz com que, ao interagir com as páginas, não seja necessário carregar a página inteira novamente, apenas os dados e componentes que foram alterados, tornando a experiência de usuário mais fluida ao navegar pelo sistema.

Devido a essa característica, os *frameworks* SPA se tornaram muito populares em aplicações Web, dentre os mais utilizados estão o Angular[1] e o Vue,[2] além do React,[3] que, apesar de ser uma biblioteca, também implementa aplicações SPA.

No entanto, este tipo de aplicação possui também problemas, sendo eles o alto tempo de carregamento inicial, pois é necessário carregar todo o código da aplicação de uma vez, e a dificuldade de indexação por mecanismos de busca (SEO), uma vez que estes tendem a classificar melhor páginas que carregam mais rápido (KONSHIN, 2018). Isso é uma grande desvantagem para um sistema de vendas, por exemplo, que precisa ser indexado por mecanismos de busca para que o site seja mais fácilmente encontrado por possíveis consumidores.

2.3.2 Frameworks ORM

Grande parte dos sistemas precisam que seus dados sejam armazenados de alguma forma e, dentre os diversos tipos de sistemas de armazenamento, os bancos de dados relacionais são os mais utilizados. Esses sistemas possuem uma estrutura de dados organizada em tabelas, e a partir de um sistema de gerenciamento de banco de dados relacional (SGBDR), é possível manipular esses dados por meio da linguagem SQL (*Structured Query Language* ou Linguagem Estruturada de Consulta) (SILBERSCHATZ; KORTH; SUDARSHAN, 2019).

Por outro lado, muitas aplicações utilizam linguagens de programação orientadas a objetos (OO), o que pode tornar a manipulação de dados em um banco de dados relacional mais complexa, uma vez que as SGBDR representam os dados como tabelas, enquanto as linguagens OO os representam como um grafo de objetos interconectados. Essa é a chamada incompatibilidade de paradigmas (*paradigm mismatch*) (HIBERNATE, 2010; BAUER; KING, 2005).

Na década de 80, surgiram os *frameworks* ORM (Mapeamento Objeto/Relacional), que têm como objetivo mapear os objetos da aplicação para as tabelas do banco de dados relacional. O uso desses *frameworks* facilita a manipulação de dados para o desenvolvedor, que precisa apenas informar ao *framework* como transformar objetos e seus atributos em tabelas e colunas e chamar métodos simples, sem precisar aprender uma nova linguagem como SQL.

[1] Angular, <https://angular.io>
[2] Vue, <https://vuejs.org>
[3] React, <https://react.dev>

Alguns exemplos de *frameworks* ORM são o Hibernate para Java,[4] Entity Framework para .NET,[5] e o Prisma para TypeScript.[6]

2.3.3 Next.js

O Next.js é um *framework Web* construído em cima do React, biblioteca JavaScript *front-end open-source* baseada em componentes, voltada para a construção de interfaces de usuário e mantida pela Meta.[7] Por ser uma biblioteca, o React não implementa técnicas de roteamento, o que faz com que exista a dependência do uso de outras bibliotecas, como o React Router.[8] Já o Next.js faz isso de forma nativa.

O *framework* destaca-se também por suas técnicas de renderização. Além do CSR, o Next.js implementa os métodos *Server-Side Rendering* (SSR), ou Renderização do Lado do Servidor, e *Static Site Generation* (SSG), ou Geração Estática de Site, que permitem que o código da aplicação seja executado no servidor, e resolvem os problemas de SEO e tempo de carregamento inicial, comuns em aplicações SPA. Um detalhe que tem feito com que esse *framework* ganhe popularidade, é o fato de que o desenvolvedor pode escolher qual método de renderização utilizar para cada página do sistema (VERCEL, 2023b).

Outro ponto alto do Next.js é a facilidade de se fazer o *deploy* da aplicação, que pode ser realizado de forma otimizada com o provedor de *hosting* da Vercel,[9] plataforma que desenvolveu o *framework* (VERCEL, 2023a).

Em termos de arquitetura, o Next.js não segue o padrão Model-View-Controller (MVC). Em vez disso, ele adota um modelo de execução baseado em páginas, em que cada arquivo dentro da pasta *pages* é mapeado para uma rota na aplicação. Dessa forma, quando uma requisição é feita, o Next.js determina qual página deve ser renderizada e aplica a técnica de renderização adequada (CSR, SSR ou SSG) conforme configurado pelo desenvolvedor. Por exemplo, se criarmos um arquivo *pages/index.js*, ele será acessível via a rota /. No entanto, fora essa convenção, o Next.js não especifica outros padrões de arquitetura, deixando a cargo do desenvolvedor a organização do código (VERCEL, 2023a).

Apesar do Next.js ser construído em cima da biblioteca React, isso não significa que este seja um *framework* exclusivo para o desenvolvimento *frontend*. O Next.js pode ser utilizado como um *framework fullstack*, permitindo o desenvolvimento tanto do *frontend* quanto do *backend*, em uma única aplicação. Para isso, o Next.js utiliza a pasta reservada *api*, dentro do diretório *pages*, cada arquivo dentro dessa pasta é mapeado para uma rota de API,

[4] Hibernate, <https://hibernate.org>
[5] Entity Framework, <https://docs.microsoft.com/pt-br/ef>
[6] Prisma, <https://www.prisma.io>
[7] Meta, <https://about.meta.com>
[8] React Router, <https://reactrouter.com>
[9] Vercel, <https://vercel.com>

permitindo que os desenvolvedores criem *endpoints* que podem ser acessados diretamente pelo cliente. Essas rotas de API são executadas no lado do servidor, proporcionando a capacidade de manipular dados, realizar autenticações, conectar a bancos de dados, entre outras funcionalidades comuns de *backend* (VERCEL, 2023a). Dessa forma, o Next.js oferece uma solução completa e integrada para o desenvolvimento de aplicações *Web* modernas. Ainda assim, o *framework* é mais comumente utilizado unicamente como *frontend*, com o *backend* sendo implementado separadamente (HARRIS, 2024).

JavaScript é uma linguagem leve e interpretada, que é utilizada para adicionar interatividade e dinamicidade a páginas *Web*. Por ser uma linguagem de tipagem fraca e dinâmica, ela pode ser propensa a erros em tempo de execução, o que, em sistemas de porte elevado, pode se tornar um grande problema (TYPESCRIPT, 2024). Nesse ambiente, surge o *TypeScript*, linguagem de programação de alto nível desenvolvida e mantida pela Microsoft.[10] O *TypeScript* é convertido em código *JavaScript*, e portanto fornece todas as características desta, além de adicionar suporte a tipos, permitindo a detecção de erros em tempo de compilação, e outras funcionalidades, como interfaces, *generics* e *unions* (TYPESCRIPT, 2024).

O Next.js suporta o uso de *TypeScript* em suas aplicações, o que pode ser uma grande vantagem para desenvolvedores que desejam ter um código mais seguro e menos propenso a erros.

[10] Microsoft, <https://www.microsoft.com>

3 Especificação de Requisitos

Este capítulo tem como objetivo apresentar a especificação de requisitos do SCAP (Sistema de Controle de Afastamento de Professores), assim como os modelos de casos de uso e de classes levantados por Duarte (2014) e Prado (2015), que serão utilizados como base para a implementação do sistema.

3.1 Descrição do Escopo

O SCAP é um sistema que visa auxiliar o gerenciamento das solicitações de afastamento de professores do Departamento de Informática (DI) da Universidade Federal do Espírito Santo (UFES) para participação em eventos, sejam eles no Brasil ou no exterior. Nesses casos, o professor deve submeter uma solicitação de afastamento temporário que será avaliada pelos professores do DI, e em alguns casos, pelo Conselho do Centro Tecnológico (CT) e Pró-Reitoria de Pesquisa e Pós-Graduação (PRPPG) da UFES. Ao ser aprovada em todas as instâncias, o afastamento temporário é autorizado.

Para eventos no Brasil, as solicitações tramitam apenas no DI, e devem ser aprovadas pela Câmara Departamental (formada pelos funcionários do Departamento e representantes discentes). Dessa forma, o professor envia seu pedido para a lista de e-mails dos funcionários do DI, endereçado ao Chefe de Departamento, cargo exercido temporariamente por um professor do DI. Caso nenhum membro da Câmara Departamental se oponha ao pedido em até dez dias, o afastamento é autorizado.

Para eventos no exterior, um professor, que não tenha parentesco com o solicitante, é escolhido para ser o relator do pedido. Após emitido o parecer do relator, o processo deve ser aprovado pela Câmara Departamental, como no caso de eventos no Brasil. Além disso, a solicitação deve ser aprovada pelo Conselho Departamental do CT e pela PRPPG. Depois de ser aprovado por todos os envolvidos, o afastamento é autorizado e o pedido é publicado no Diário Oficial da União. No entanto, o SCAP é responsável por assistir o processo apenas dentro do DI, não havendo uma integração com os processos no CT e na PRPPG.

O objetivo do SCAP é agilizar e simplificar o processo para os professores e secretários do DI, automatizando o envio de e-mails aos membros envolvidos e utilizando formulários para a criação dos documentos necessários.

3.2 Modelos de Casos de Uso

Na Tabela 1 são apresentados os atores do SCAP, identificados por Duarte (2014) no levantamento de requisitos do sistema.

Tabela 1 – Atores do Sistema

Atores	**Descrição**
Professor	Professor efetivo do DI
Secretário	Secretário do DI
Chefe de Departamento	Professor do DI que está realizando a função administrativa de chefe ou subchefe do departamento

O secretário é responsável pela administração do sistema. Sendo assim, ele deve cadastrar os professores, e seus parentescos, além dos mandatos de chefes e subchefes do departamento. Outras tarefas incluem, cadastrar as decisões do CT e PRPPG (em casos de eventos no exterior), e arquivar processos que forem concluídos.

O professor pode cadastrar solicitações de afastamento e também se manifestar contra pedidos de outros professores, caso esteja dentro do prazo para tal. Se for adicionado como relator de um pedido de afastamento no exterior, o professor deve emitir um parecer sobre o mesmo, e assim decidir se o DI aprova ou não tal solicitação.

O chefe de departamento é um professor que foi eleito para exercer a função administrativa durante um período de tempo. Assim, além das funcionalidades comuns a um professor, ele também é responsável por nomear relatores para pedidos de afastamento no exterior.

O SCAP foi dividido em dois módulos: Cadastral e Núcleo. O primeiro é responsável pela parte cadastral, ou seja, casos de uso dos secretários. Já o segundo, abrange as funcionalidades de professores e chefes de departamento. Uma versão mais completa e detalhada pode ser vista em (DUARTE, 2014; PRADO, 2015). As Figuras 2 e 3 apresentam os diagramas de casos de uso do SCAP.

Em **Cadastrar Usuário**, o secretário cadastra um novo usuário no sistema, sendo este um professor ou um secretário, junto de seus dados. Já em **Cadastrar chefe de departamento**, o secretário cadastra o mandato do chefe ou subchefe do departamento, assim como suas datas de início e fim.

Os casos de uso **Registrar parecer do PRPPG** e **Registrar parecer do CT**, ocorrem apenas em casos de afastamento no exterior. O secretário é responsável por registrar as decisões da PRPPG e do CT, respectivamente, quando estes aprovam ou não um pedido de afastamento.

Em **Arquivar solicitação**, o secretário arquiva um processo que foi concluído, ou seja, que já passou por todas as instâncias e foi aprovado. **Consultar Solicitação** é

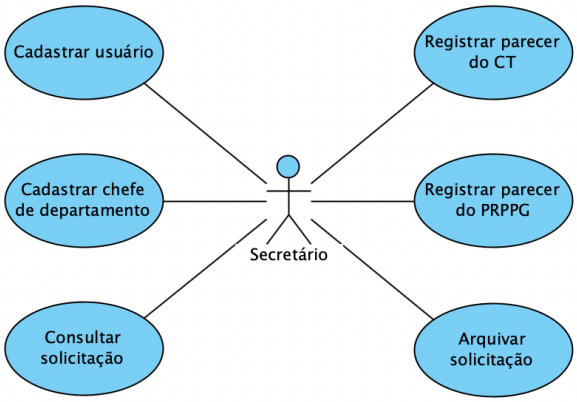

Figura 2 – Diagrama de Casos de Uso do subsistema Cadastral (PRADO, 2015).

um caso de uso comum aos professores e secretários, e permite que o usuário consulte o andamento e os dados de um pedido de afastamento.

Figura 3 – Diagrama de Casos de Uso do subsistema Núcleo (PRADO, 2015).

O caso de uso **Solicitar afastamento** é o principal caso de uso do sistema, e permite que um professor solicite um afastamento para participar de um evento. Para isso devem ser informadas as datas de início e fim, o nome do evento e se este é nacional ou internacional. Enquanto em **Cancelar solicitação**, o professor pode cancelar um pedido feito por ele próprio, alterando assim o estado do processo para cancelado.

O professor pode ainda **Manifestar-se contra** um pedido de afastamento de outro professor, caso esteja dentro do prazo para tal, registrando o motivo de sua opinião. Uma reunião deve então ser agendada para que os professores decidam se o afastamento será aprovado ou não. Por último, **Emitir parecer** é um caso de uso exclusivo de professores que foram nomeados como relator de um pedido de afastamento no exterior. O professor deve cadastrar um parecer sobre a solicitação.

O chefe de departamento, além das funcionalidades comuns a um professor, também

pode **Encaminhar afastamento**, ou seja, indicar um professor como relator para um pedido de afastamento no exterior.

3.3 Análise do SCAP

A Figura 4 apresenta o diagrama de classes do SCAP, adaptado do levantamento feito por Prado (2015), juntamente com o que havia sido levantado anteriormente por Duarte (2014).

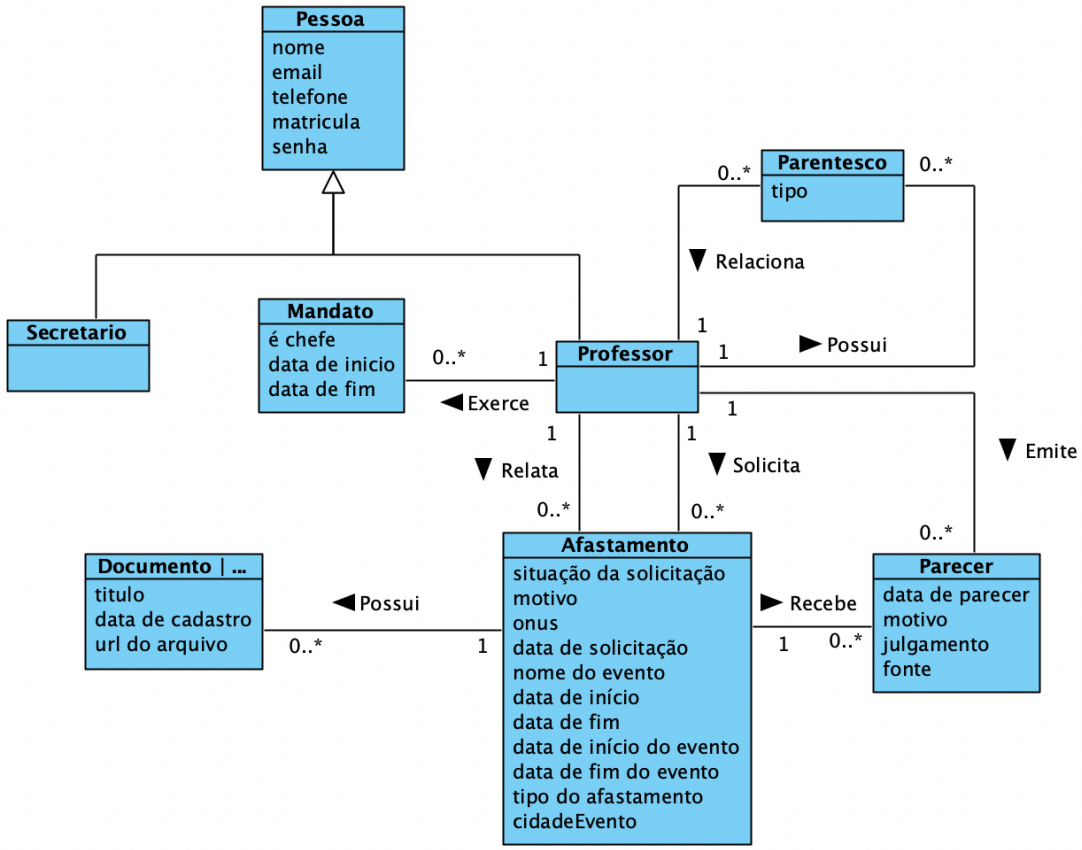

Figura 4 – Diagrama de Classes do SCAP, adaptado de (PRADO, 2015).

As classes **Professor** e **Secretario**, que representam os professores e secretários do DI, respectivamente, herdam as propriedades da classe **Pessoa**, que contém as informações pessoais do usuário. A classe **Parentesco** guarda relações de parentesco que podem existir entre professores do DI. A classe **Mandato** representa o tempo do mandato, de chefe ou subchefe do departamento, que o professor está exercendo.

A classe **Afastamento** guarda informações sobre os pedidos de afastamento feitos pelos professores, como as datas de início e fim, o nome do evento e etc. Este pode ou não possuir um **Documento** associado, e deve possuir um **Relator** caso seja um pedido de afastamento no exterior. Um professor pode ser relator de vários afastamentos, mas seu **Parecer** é único para cada solicitação de afastamento internacional.

Restrições de integridade complementam o modelos de classes, que por muitas vezes não são capazes de representá-las por notações gráficas. Abaixo estão listadas as restrições levantadas por Duarte (2014) e posteriormente ampliadas por Prado (2015):

- Um professor não pode ser nomeado como relator, nem emitir um parecer, para sua própria solicitação de afastamento;
- Um professor não pode ser relator de um afastamento solicitado por um parente;
- A data de início de um afastamento não pode ser posterior a data de fim do mesmo afastamento;
- A data de início de um mandato de professor não pode ser posterior a data de fim do mesmo mandato;
- Não pode haver mais de dois professores (chefe e subchefe de departamento) exercendo um mandato ao mesmo tempo;
- O secretário do departamento não pode abrir uma solicitação de afastamento.

4 Projeto Arquitetural e Implementação

Neste capítulo são apresentados os passos de projeto arquitetural e desenvolvimento da aplicação. Na Seção 4.1 estão as tecnologias utilizadas, enquanto a Seção 4.2 apresenta a arquitetura do sistema. São exemplificados, na Seção 4.3, os modelos FrameWeb desenvolvidos para auxilio na implementação do SCAP. A Seção 4.4 mostra a estrutura do sistema. E por fim, na Seção 4.5 são apresentadas capturas de tela da aplicação para ilustrar o desenvolvimento do SCAP.

4.1 Tecnologias Utilizadas

Neste projeto, utilizou-se a linguagem *TypeScript* unida ao *framework* Next.js, que foi utilizado como *framework fullstack*. A Tabela 2 apresenta as tecnologias e bibliotecas utilizadas, suas funções e suas respectivas versões. Dentre as tecnologias, destaca-se o MySQL, um sistema de gerenciamento de banco de dados relacional, e o Prisma, um ORM (*Object-Relational Mapping*) para Node.js, ferramenta utilizada para mapear classes para tabelas, além de facilitar a manipulação de dados com o banco.

Tabela 2 – Tecnologias e Bibliotecas Utilizadas.

Tecnologia	**Função**	**Versão**
TypeScript	Linguagem de programação	5
Next.js	*Framework* para desenvolvimento *Web*	14.1.3
Tailwind CSS	*Framework* CSS para estilizar componentes	3.3.0
Axios	Cliente HTTP baseado em *Promises*	1.6.8
Prisma	ORM (*Object-Relational Mapping*) para Node.js	5.11.0
MySQL	Sistema de Gerenciamento de Banco de Dados Relacional	8.3.0
React Dropzone	Componente para *upload* de arquivos	14.2.3
React Input Mask	Componente para máscaras de *inputs*	3.0.0
React Toastify	Componente para notificações	10.0.5
Luxon	Biblioteca para manipulação de datas	3.4.4
uuid	Biblioteca para geração de identificadores únicos	9.0.1
NPM	Gerenciador de pacotes para *Node.js*	10.5.0
Visual Paradigm	Ferramenta para modelagem de diagramas	17.1
VSCode	Editor de código-fonte	1.89.1

4.2 Arquitetura do Sistema

Após reunir os requisitos do sistema, e definir as tecnologias a serem utilizadas, entra a fase de definir a arquitetura a ser utilizada no projeto. A arquitetura define

elementos de software (ou módulos) e envolve informações sobre como eles se relacionam uns com os outros (FALBO, 2018).

Por ser um *framework* de *Single Page Application* (SPA), o Next.js funciona por meio de componentes, e possui um padrão de *design* diferente de projetos Java, por exemplo, que costumam utilizar padrões como o *Model-View-Controller* (MVC).

Baseado na arquitetura sugerida pelo FrameWeb, apresentada anteriormente na Figura 1, desenvolveu-se a arquitetura presente na Figura 5. A arquitetura é dividida em três camadas: Apresentação, Negócio e Persistência. Podemos separar logicamente os pacotes, assim a Camada de Apresentação representa o *frontend*, enquanto as camadas de Negócio e Persistência compõem o *backend*. Contudo, ambos estão presentes em um único servidor, adotando uma arquitetura monolítica.

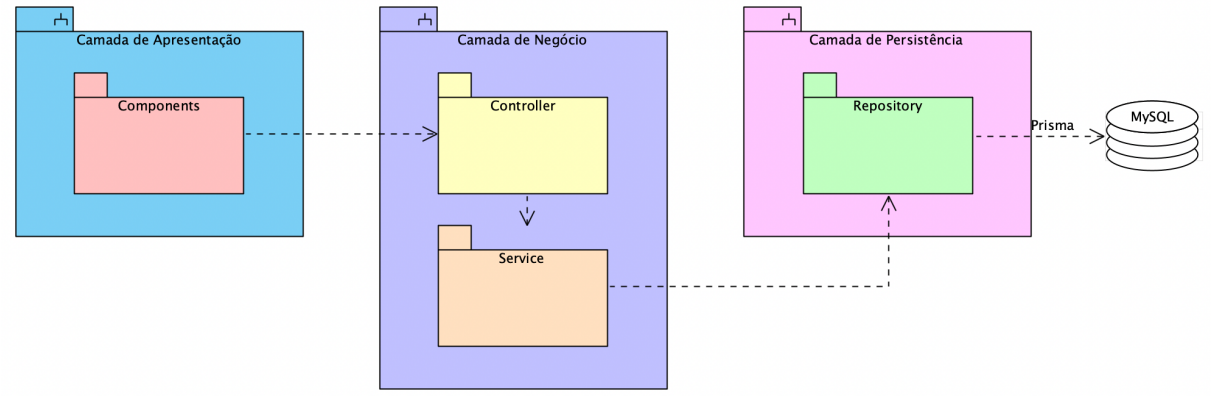

Figura 5 – Representação em pacotes da arquitetura do SCAP.

A primeira camada contém os componentes responsáveis por exibir as informações na tela e manejar as interações do usuário com o sistema. A comunicação entre as camadas de Apresentação e Negócio é mediada por uma API (*Application Programming Interface*), que funciona como uma ponte entre o *frontend* e o *backend*. A camada de Apresentação envia requisições HTTP para a API. Essas requisições são formatadas conforme os *endpoints* definidos pela API e podem incluir dados solicitados pelo usuário ou comandos para executar operações. Dentre os tipos de operações estão: *GET* para buscar dados, *POST* para criar novos registros, *PUT* para atualizar registros existentes e *DELETE* para remover registros.

A API, por sua vez, é implementada na camada de Negócio, que contém as classes de controle e serviço, a primeira é responsável por processar e responder as requisições, enquanto a segunda é responsável por executar as operações de negócio. Por fim, as classes de serviço acessam a camada de Persistência, que contém as classes de repositório, responsáveis por persistir os dados no banco, por meio de operações de leitura e escrita. Essas operações são realizadas utilizando o Prisma, *framework* ORM. As camadas de Negócio e Persistência implementam o padrão *Repository*, este propõe uma camada de

separação entre o domínio e o mapeamento de dados.

4.3 Modelos FrameWeb

Nessa seção são apresentados os modelos FrameWeb, construídos na fase de Projeto Arquitetural do SCAP, com o objetivo de guiar a fase de implementação do sistema.

4.3.1 Modelo de Entidades

O modelo de entidades de FrameWeb é um diagrama de classes UML que representam os objetos de domínio do problema e seu mapeamento para a persistência no banco de dados relacional. A partir dele são implementadas as classes da camada de domínio (SOUZA, 2007).

A Figura 6 apresenta o modelo de entidades do SCAP. O modelo foi feito a partir do diagrama de classes mostrado anteriormente, com adaptações para a plataforma escolhida para a implementação do sistema, dessa forma são apresentados os tipos de cada propriedade. A Figura 7 apresenta os tipos enumerados do modelo de entidades do SCAP.

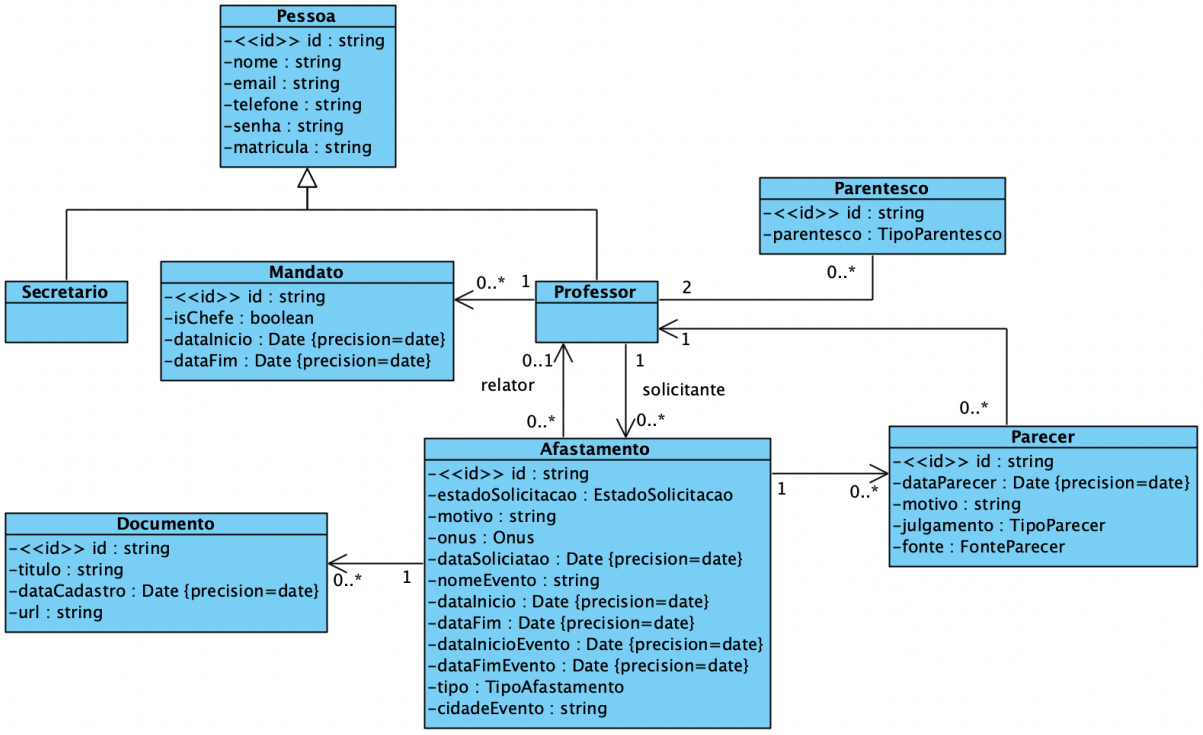

Figura 6 – Modelo de Entidades do SCAP.

Os diagramas 8 e 9 apresentam os fluxos de estados para solicitações de afastamento do tipo nacional e internacional, respectivamente. Explicitando os estados que uma solicitação de afastamento pode assumir e as transições entre eles, de acordo com determinadas ações. Um exemplo de fluxo para um afastamento nacional é descrito abaixo:

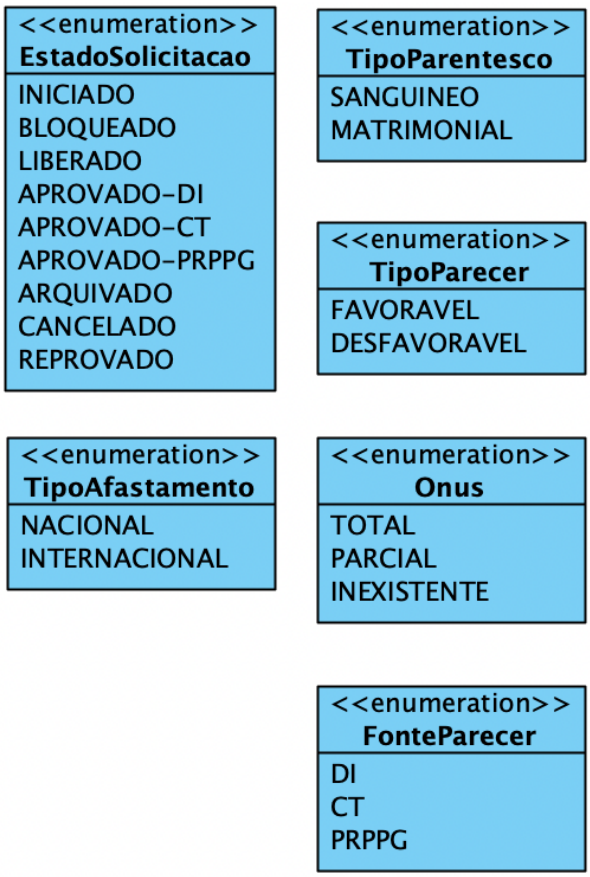

Figura 7 – Tipos Enumerados do Modelo de Entidades do SCAP.

1. O professor solicita um afastamento nacional, este é criado no estado **INICIADO**;
2. Um professor do DI manifesta-se contra a solicitação, o afastamento passa para o estado **BLOQUEADO**;
3. Após votação favorável do colegiado, o secretário registra a aprovação do afastamento que muda para o estado **APROVADO_DI**;
4. Por fim, o secretário arquiva a solicitação, modificando seu estado para **ARQUIVADO**.

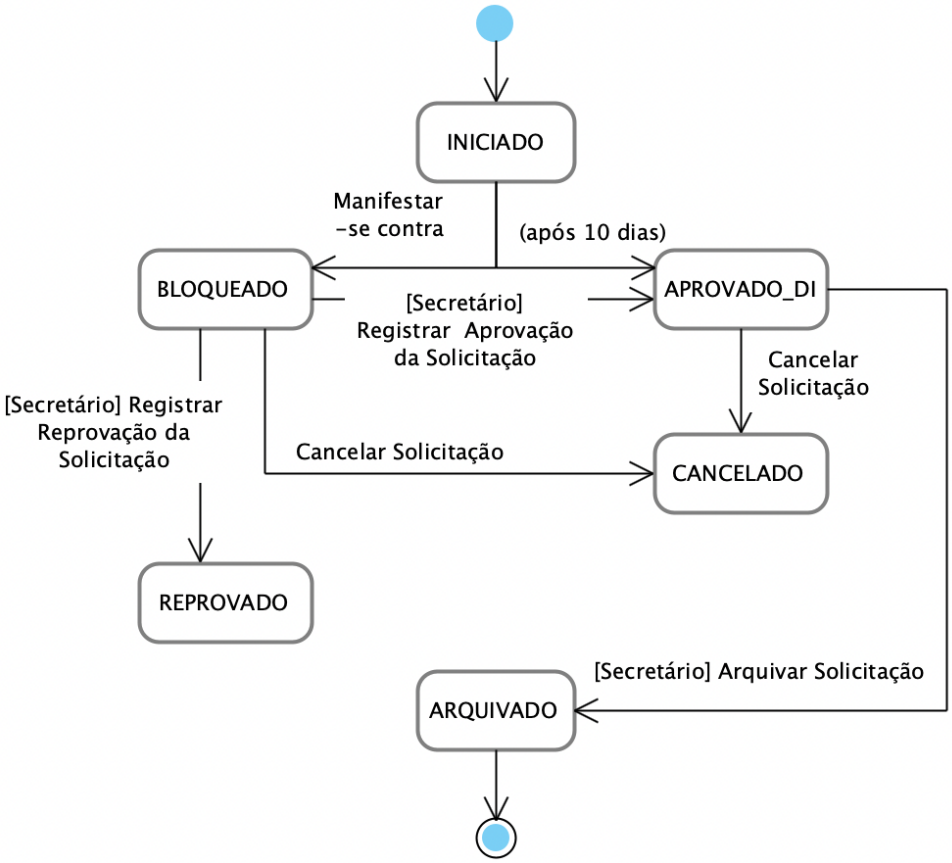

Figura 8 – Diagrama de Estados da Solicitação de Afastamento Nacional do SCAP.

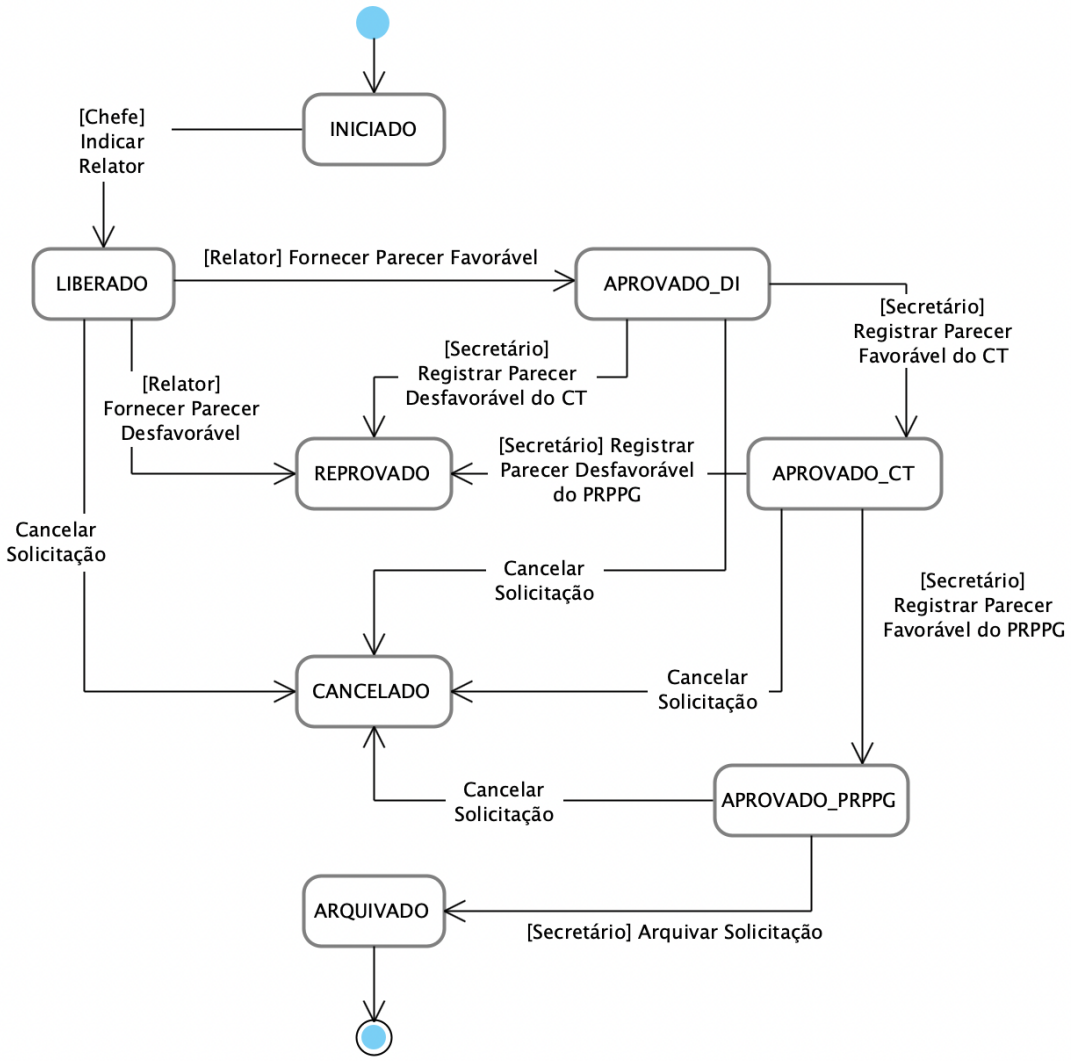

Figura 9 – Diagrama de Estados da Solicitação de Afastamento Internacional do SCAP.

4.3.2 Modelo de Persistência

O modelo de persistência de FrameWeb é um diagrama de classes UML que representa as classes responsáveis pela persistência das classes de domínio no banco de dados (SOUZA, 2007).

Para isso, foi criada uma interface base (**IBaseRepositoy**) que define os métodos comuns a todas as classes de persistência, sendo eles: *get* para buscar todos os elementos com base nos filtros passados como parâmetro, *post* para criar um novo elemento, *getById* para buscar um elemento a partir de seu *id* e *delete*. A Figura 10 apresenta essa interface e a classe *Repository* que a implementa. O tipo genérico T representa a classe de domínio sendo manipulada, ou seja, em **AfastamentoRepository** o método *getById* retorna um elemento da classe **Afastamento**.

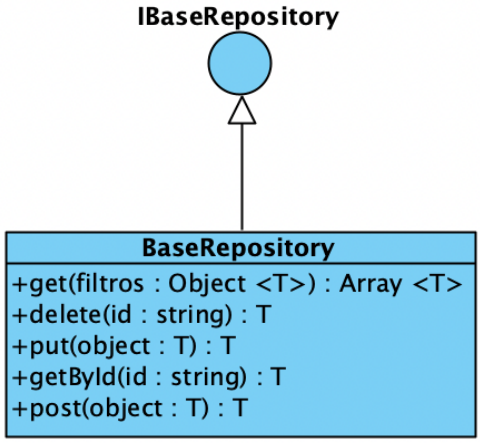

Figura 10 – Interface e Implementação do Repository Base.

As interfaces do modelo herdam **IBaseRepositoy**, enquanto as classes *Repository* estendem a classe **BaseRepository**. De modo que todas as classes de persistência possuam, além dos métodos específicos dessas, os métodos comuns definidos na interface base. A Figura 11 contém as classes de persistência, que juntas compõem o modelo de persistência do SCAP.

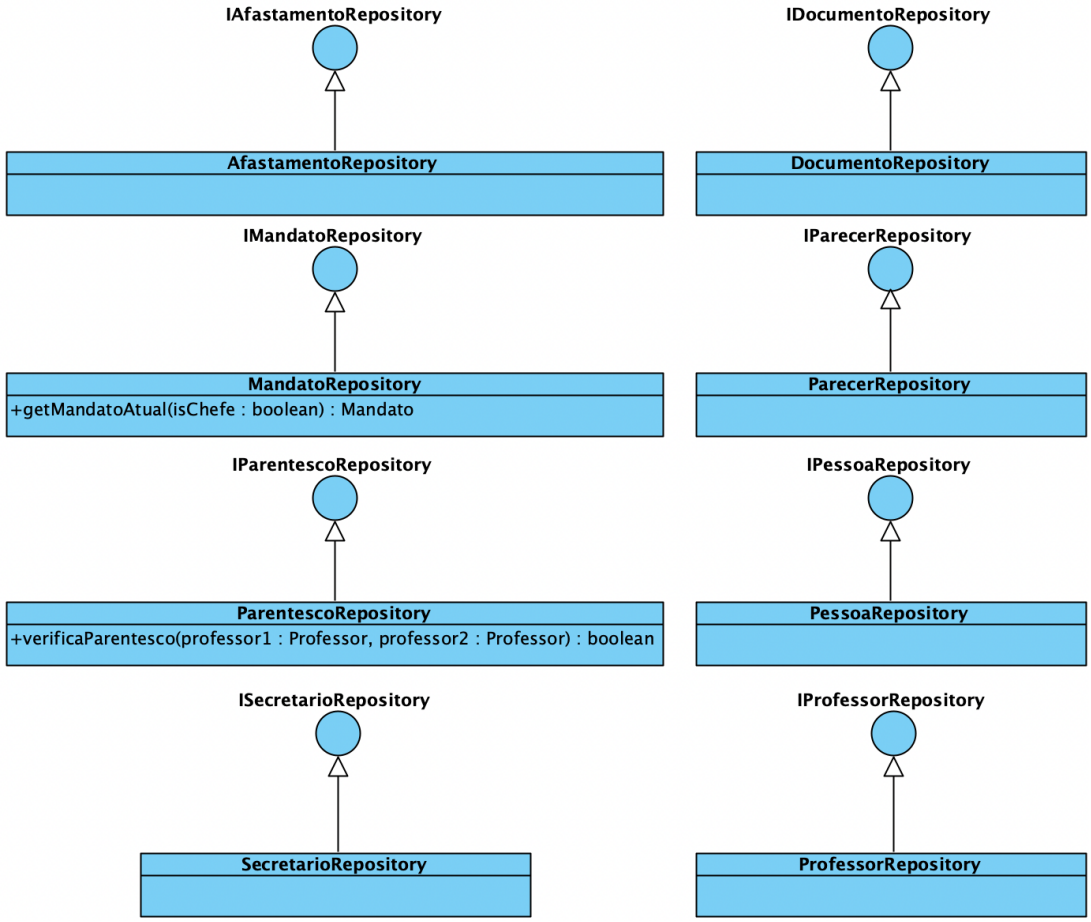

Figura 11 – Modelo de Persistência do SCAP.

4.3.3 Modelo de Navegação

O Modelo de Navegação é um diagrama de classe UML que representa os diferentes componentes que formam a camada de Lógica de Apresentação, como páginas Web, formulários HTML e classes de ação do *framework* específico (SOUZA, 2007). Esse modelo é utilizado para guiar a implementação dos pacotes Visão e Controle.

A Tabela 3 apresenta os estereótipos UML utilizados no modelo de navegação do SCAP, propostos por Hoppe e Souza (2023) como uma adaptação, para *frameworks* SPA, dos estereótipos propostos originalmente por Souza (2007).

Tabela 3 – Estereótipos UML para Modelos de Navegação (HOPPE; SOUZA, 2023).

Estereótipo	**Descrição**
(Nenhum)	Controladora de um framework Front Controller ou a parte controladora de um componente de um framework SPA.
«Page»	Página Web estática ou dinâmica.
«Partial»	Parte de uma página HTML que é gerada em tempo de execução por meio de AJAX.
«Form»	Formulário HTML

A Figura 12 apresenta o modelo de navegação do SCAP para os casos de uso

Cancelar Afastamento, **Cadastrar Afastamento** e **Consultar Afastamento**.

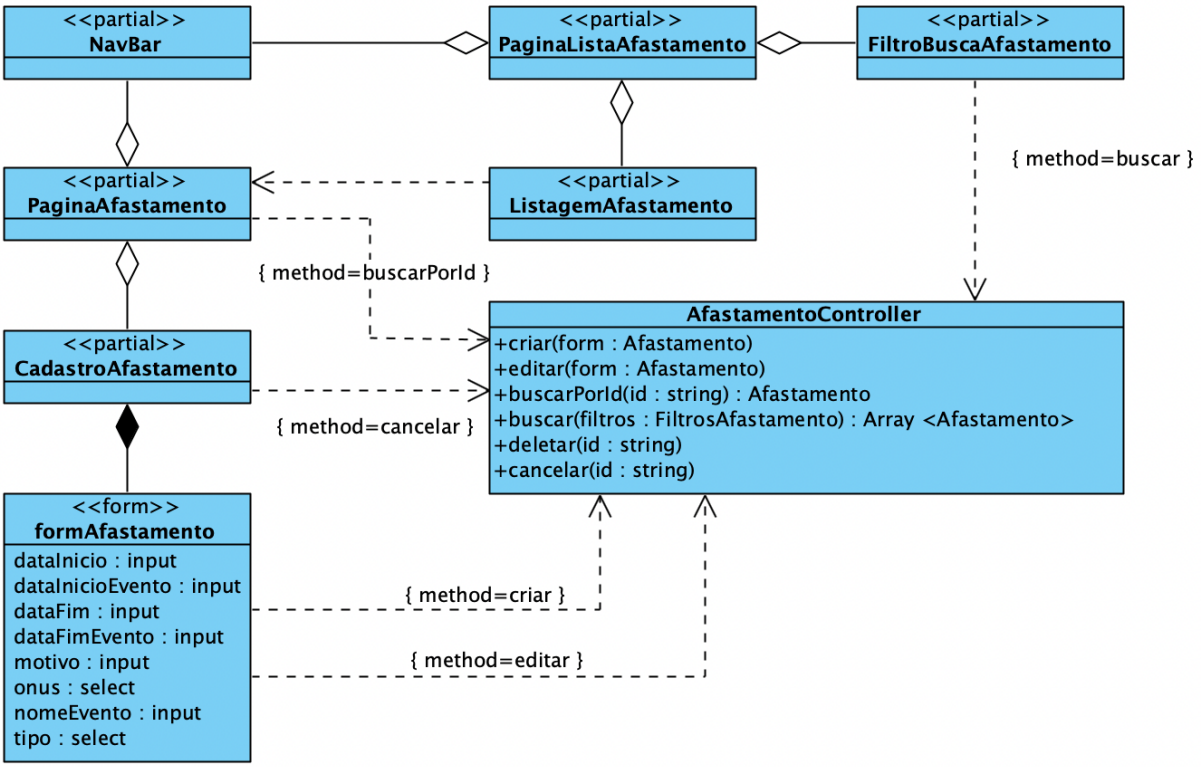

Figura 12 – Modelo de Navegação do SCAP.

A **PaginaListaAfastamento** é responsável por listar os afastamentos cadastrados. Ela possui um componente filtro (**FiltroBuscaAfastamento**), em que é possível buscar os afastamentos por diferentes critérios, definidos pela interface mostrada na Figura 13. O componente **ListagemAfastamento** consiste de uma tabela que lista os afastamentos buscados, cada linha da tabela possui um botão que redireciona para a página de detalhes do afastamento (**PaginaAfastamento**).

Figura 13 – Interface Filtro Afastamento.

Se redirecionado para a **PaginaAfastamento** a partir do componente **ListagemAfastamento**, o *partial* possui o atributo **id** inicializado com o *id* do afastamento, e é

utilizado para fazer uma chamada ao método **buscarPorId** da *controller* **AfastamentoController**, por meio de uma requisição *GET* HTTP à API. Caso o usuário seja o solicitante do afastamento, ele pode cancelá-lo clicando em um botão que faz uma chamada ao método *DELETE* do protocolo HTTP, que é recebido pelo método **cancelar** da *controller* **AfastamentoController**.

Quando não possui um *id* inicializado, o *partial* é utilizado para criar um novo afastamento, o usuário deve então preencher os campos do formulário e clicar em um botão que faz uma chamada ao método **criar** da *controller* **AfastamentoController**, por meio de uma requisição do tipo *POST* à API.

A Figura 14 apresenta o modelo de navegação do SCAP para o caso de uso **Cadastrar Professor**. Na **PaginaProfessor** o secretário pode cadastrar um novo professor, para isso ele deve preencher os campos do formulário e clicar em um botão que faz uma chamada ao método *POST* HTTP, que é recebido pelo método **criar** da *controller* **PessoaController**.

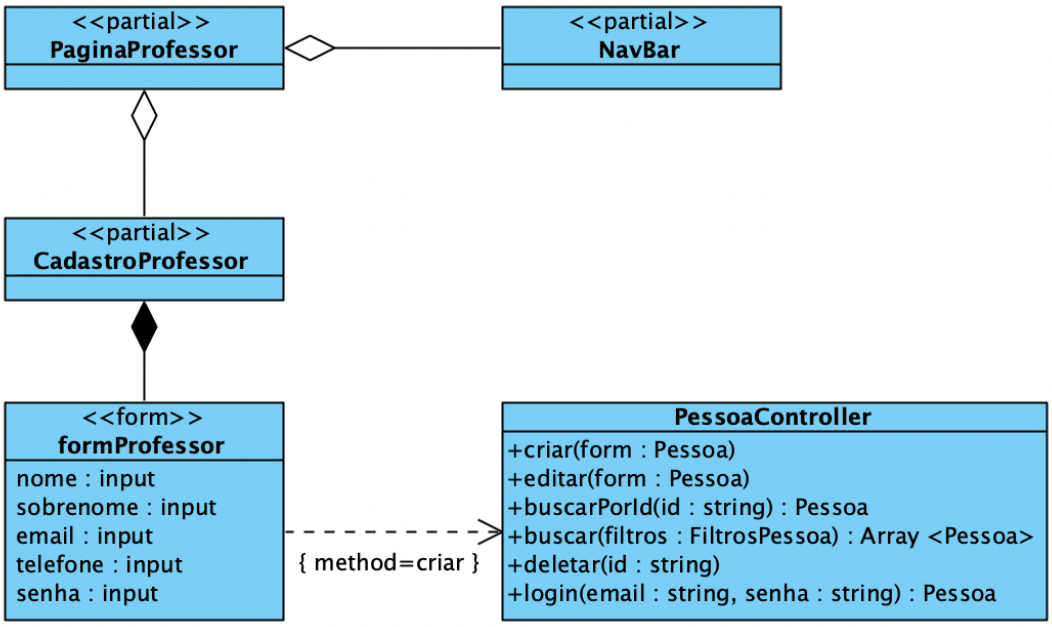

Figura 14 – Modelo de Navegação do SCAP do Caso de Uso Cadastrar Professor.

O componente **NavBar** presente em ambos os modelos de navegação é responsável por exibir no topo da tela uma barra de navegação com links para as páginas principais do sistema, como a página inicial, a página de listagem de afastamentos e a página de cadastro de professores. Ele é um componente comum a todas as páginas do sistema, exceto à tela de *login*.

4.3.4 Modelo de Aplicação

O Modelo de Aplicação é um diagrama de classes da UML que representa as classes de serviço, que são responsáveis pela codificação dos casos de uso, e suas dependências (SOUZA, 2007). Por ele pode-se visualizar a dependência entre os pacotes Controle (classes de ação), Aplicação (classes de serviço) e Persistência (interfaces *Repository*).

As figuras 15 e 16 contém os Modelos de Aplicação. As solicitações dos usuários são tratadas pelas classes de controle, que por sua vez chamam as classes de serviço para realizar as operações de negócio, que por fim acessam as classes de persistência para realizar as operações de leitura e escrita no banco de dados.

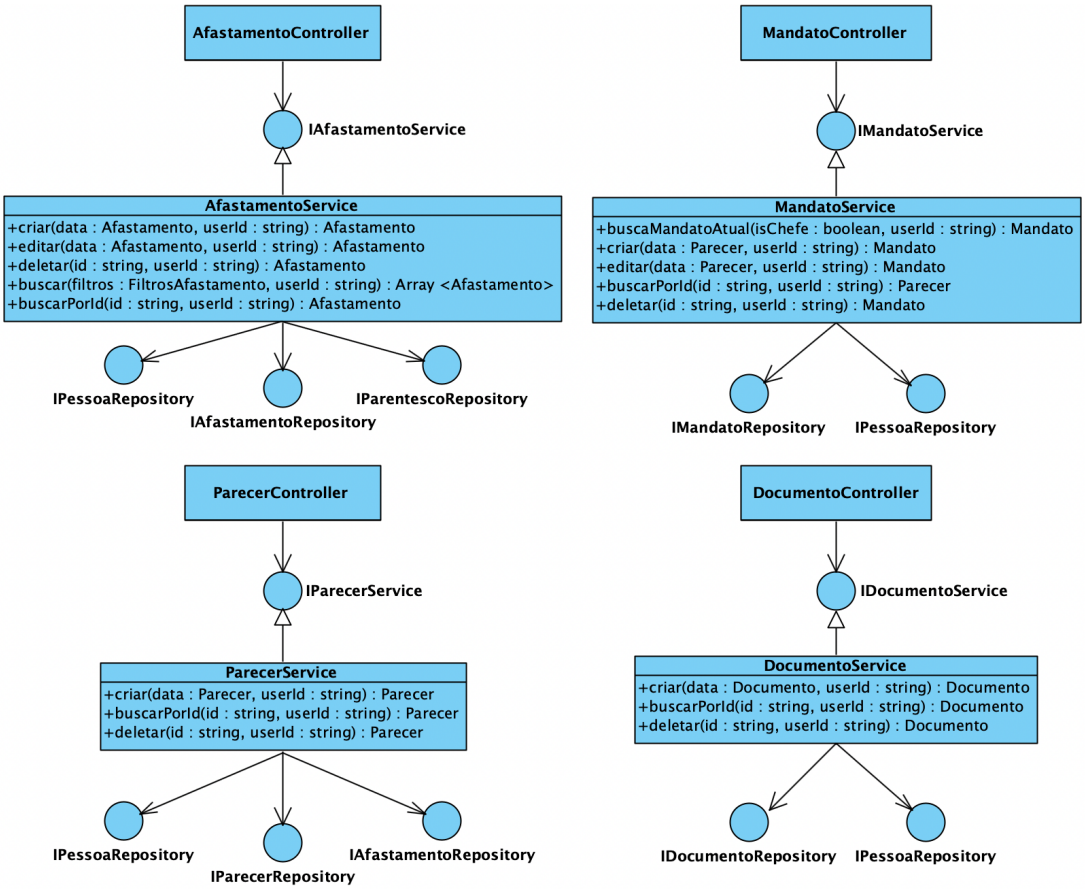

Figura 15 – Modelo de Aplicação do SCAP.

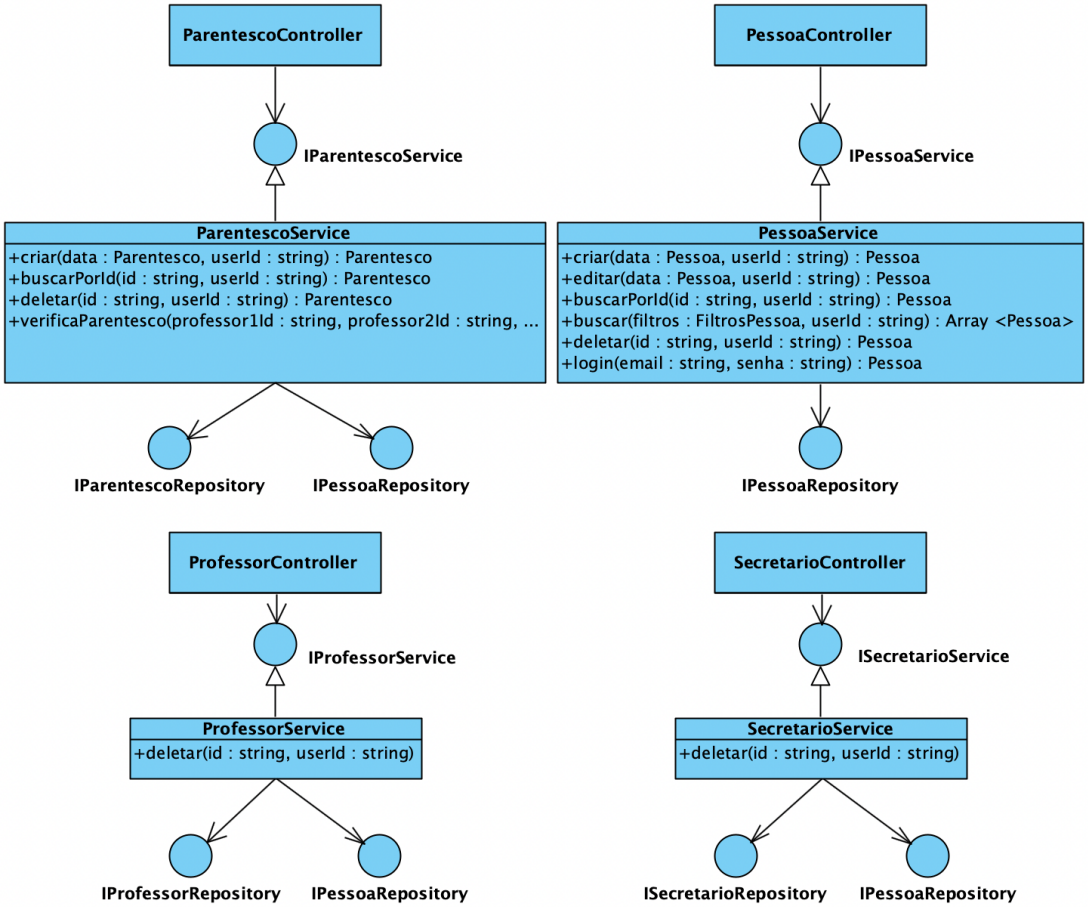

Figura 16 – Modelo de Aplicação do SCAP.

4.4 Estrutura do Sistema

A Figura 17 apresenta a organização de pastas da implementação do SCAP. Na pasta **components** estão os componentes reutilizáveis, como o componente *navbar* presente em todas as páginas do sistema. A pasta **src** é a principal, nela estão contidas algumas pastas especiais controladas pelo *framework* como **pages** e **api**. A primeira, **pages**, transforma tudo que está dentro dela em uma rota do sistema, ou seja, um arquivo *index.js* em uma pasta chamada **pessoa** estará disponível como a rota */pessoa*. De forma similar, a pasta **api** gera rotas de API a partir dos arquivos presentes nela.

Os arquivos presentes na pasta **api** possuem funções *handler* especiais que são responsáveis por lidar com as requisições feitas ao servidor. Estas funções utilizam as classes **controller** presente na pasta **lib** para manipular a requisição e retornar uma resposta. Esta, por consequência, utiliza as classes **service** para realizar as operações de negócio, que por fim acessam as classes **repository** para realizar operações de leitura e escrita no banco de dados. As classes foram criadas seguindo o padrão de projeto *Repository Pattern*. A Listagem 4.1 apresenta a função *handler* que é mapeada para a rota *afastamento/id*. Esta função é a responsável por encaminhar as requisições realizadas pelos componentes da Camada de Apresentação para a classe **AfastamentoController**, que então as processa e

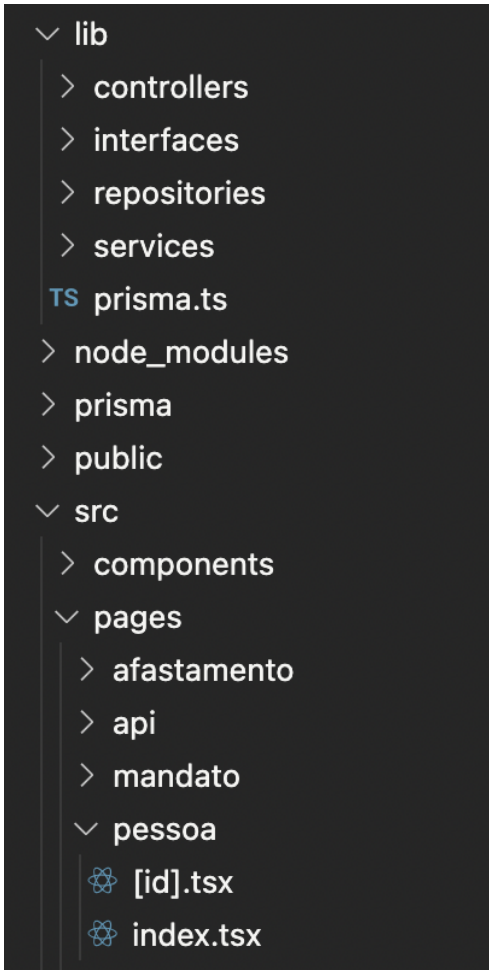

Figura 17 – Estrutura de pastas do SCAP.

retorna uma resposta.

No diretório **prisma** está presente o arquivo de configuração do Prisma, que contém o *schema* do banco de dados e as *migrations* geradas pelo Prisma. *Migrations* são arquivos que contém as alterações feitas no banco de dados, como a criação de tabelas ou a adição de colunas (PRISMA, 2020). O arquivo **prisma.schema** é responsável por definir as tabelas que serão criadas no banco, e as relações entre elas. Além disso, é a partir deste que o cliente *prisma* gera as classes das entidades que são utilizadas no sistema. A Figura 18 apresenta a definição do modelo Afastamento, no arquivo de configuração do Prisma, nele são definidos os tipos de cada atribulo além das relações com os modelos Professor, Documento e Parecer.

4.5 Apresentação de Resultados

Essa seção apresenta a implementação do SCAP, por meio de capturas de tela, e discute os resultados obtidos.

Listagem 4.1 – Função de gerenciamento da rota afastamento/id

```
const afastamentoRepository = new AfastamentoRepository();
const afastamentoService = new AfastamentoService(afastamentoRepository);
const afastamentoController = new AfastamentoController(afastamentoService);

export default async function handler(
    req: NextApiRequest,
    res: NextApiResponse
) {
    if (req.method === "GET") {
        await afastamentoController.buscarPorId(req, res);
    } else if (req.method === "DELETE") {
        await afastamentoController.deletar(req, res);
    } else if (req.method === "PUT") {
        await afastamentoController.editar(req, res);
    } else {
        res.status(405).json({ message: "Método não existente" });
    }
}
```

```
model Afastamento {
  id              String           @id @default(uuid())
  tipo            TipoAfastamento
  onus            Onus
  motivo          String
  estado          EstadoSolicitacao @default(INICIADO)
  solicitanteId   String
  relatorId       String?
  dataFim         DateTime
  dataFimEvento   DateTime
  dataInicio      DateTime
  dataInicioEvento DateTime
  dataSolicitacao DateTime?        @default(now())
  nomeEvento      String
  cidadeEvento    String
  relator         Professor?       @relation("relatorAfastamento", fields: [relatorId], references: [id])
  solicitante     Professor        @relation("solicitanteAfastamento", fields: [solicitanteId], references: [id])
  documentos      Documento[]
  pareceres       Parecer[]

  @@index([relatorId], map: "Afastamento_relatorId_fkey")
  @@index([solicitanteId], map: "Afastamento_solicitanteId_fkey")
}
```

Figura 18 – Trecho do arquivo de Configuração do Prisma.

4.5.1 Login

Para ter acesso ao SCAP, é necessário que o usuário realize o *login* fornecendo o *e-mail* e senha cadastrados no sistema. A Figura 19 apresenta a tela de *login* do SCAP. Por questões de segurança, sem o *login* o usuário não consegue acessar as outras páginas do sistema ou utilizar as rotas da API.

Como mencionado anteriormente, existem dois tipos de usuários: secretário e professor. Os secretários são responsáveis por gerenciar os usuários e os estados dos afastamentos. Já os professores podem solicitar afastamentos e visualizar o andamento das solicitações. Dessa forma, as telas mudam de acordo com o tipo de usuário que está logado.

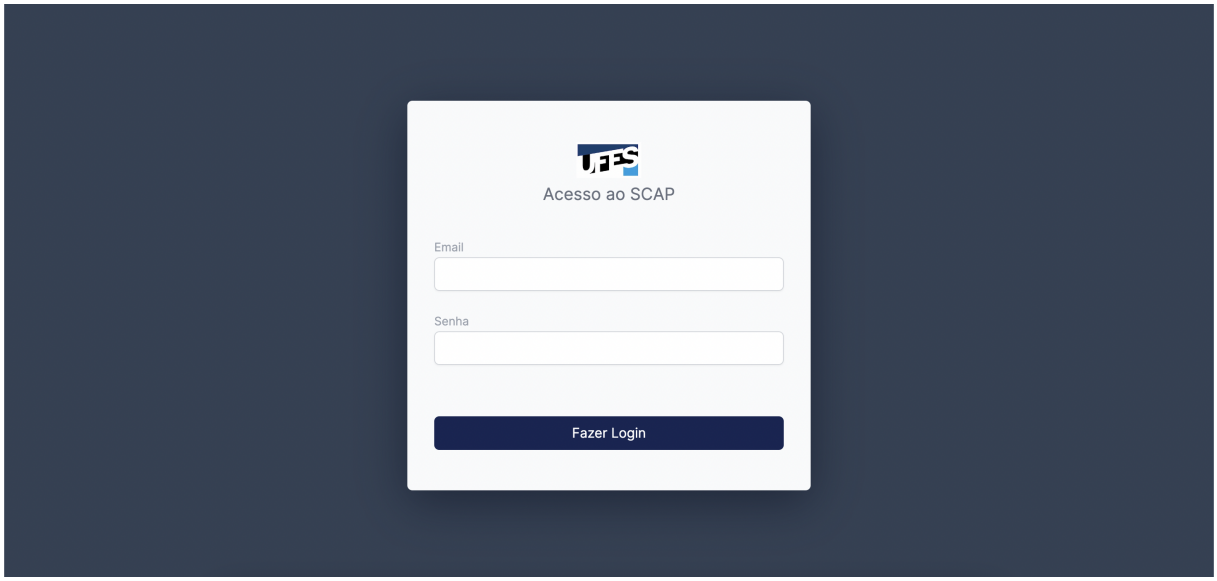

Figura 19 – Tela de Login do SCAP.

4.5.2 Listagem de Afastamentos

Ao completar o *login*, o usuário é redirecionado para a página inicial do sistema, onde é possível visualizar os afastamentos solicitados pelos professores. A Figura 20 apresenta a tela de listagem de afastamentos do SCAP. Nela é possível também buscar os afastamentos por diferentes critérios, como o nome do professor ou o estado do afastamento, a Figura 21 apresenta este filtro. Por fim, caso o usuário seja um professor, o botão de solicitar afastamento estará disponível, ao clicá-lo o usuário é redirecionado para a página de cadastro de afastamento.

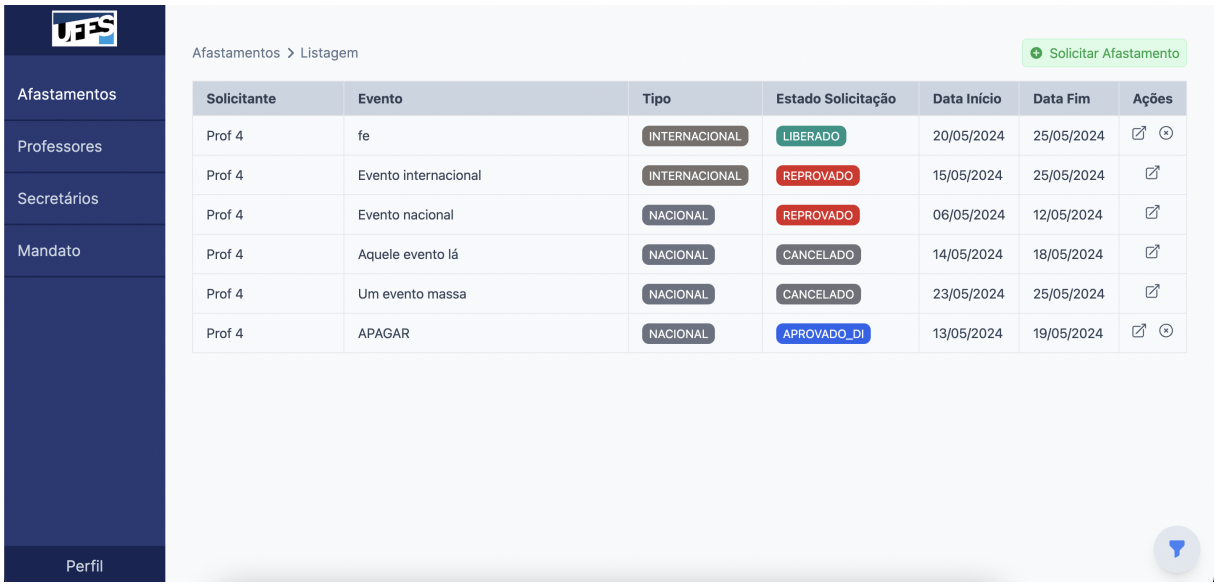

Figura 20 – Listagem de Afastamentos do SCAP.

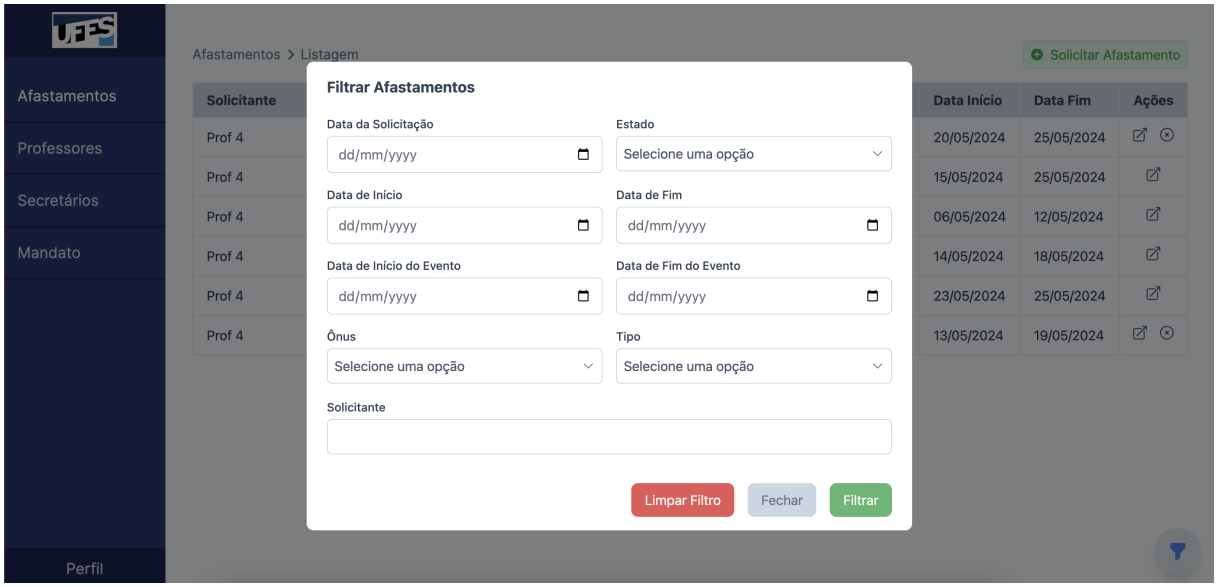

Figura 21 – Filtro de Busca de Afastamentos.

4.5.3 Cadastro de Afastamento

A Figura 22 apresenta a tela de cadastro de afastamento do SCAP. Esta rota é acessível apenas para professores, que podem solicitar um afastamento preenchendo o formulário, campos marcados com asterisco são obrigatórios.

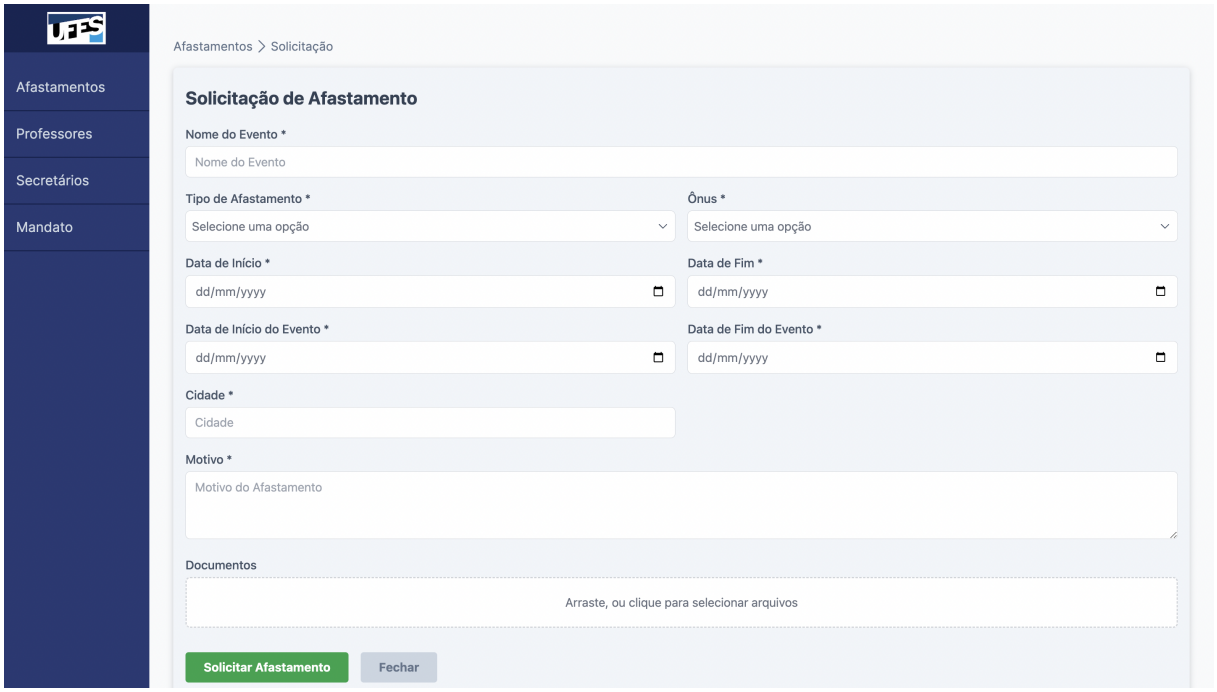

Figura 22 – Cadastro de Afastamento do SCAP.

A edição de um afastamento é feita de forma similar ao cadastro, porém os campos já estarão preenchidos com os valores do afastamento a ser editado. O mesmo componente é utilizado para ambas as ações, estando a diferença na rota acessada. Na edição, apenas o professor solicitante pode editar as informações do afastamento, exceto os campos: *estado*

e *relator*, que são de responsabilidade do secretário e professor chefe, respectivamente. Um exemplo é exibido na Figura 23.

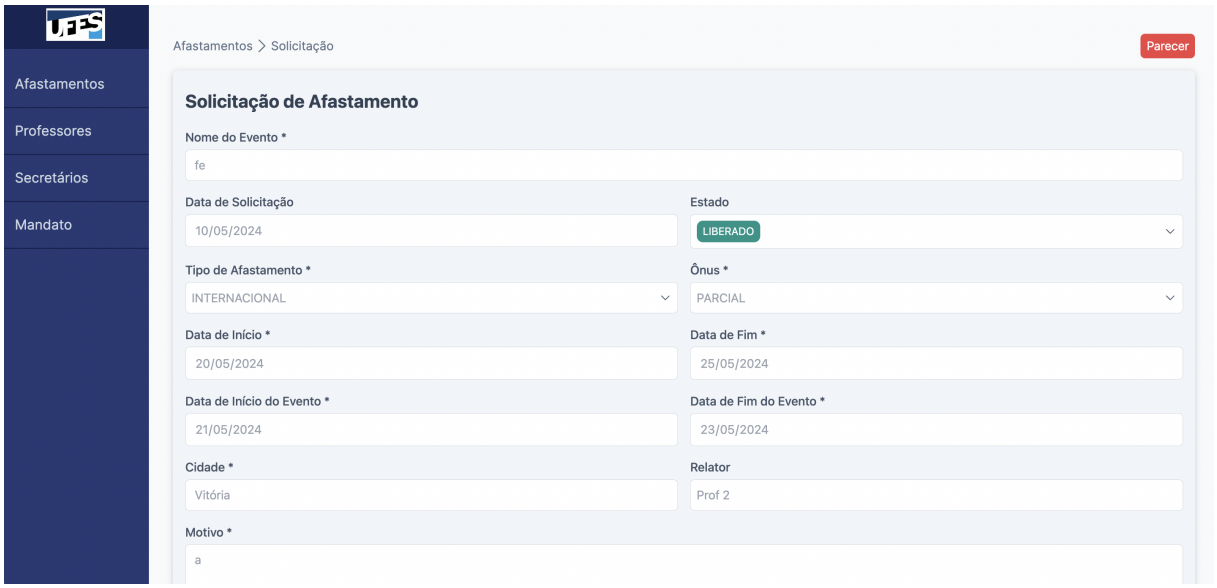

Figura 23 – Edição de Afastamento do SCAP.

Caso seja um professor sem parentesco com o solicitante do afastamento, o botão de **Parecer** estará disponível, abrindo um modal com o formulário para preenchimento do parecer. A Figura 24 apresenta este modal. Por serem funcionalidades semelhantes, o modal de parecer é utilizado também para pareceres externos (CT e PRPPG) e manifestações de professores contra o afastamento nacional.

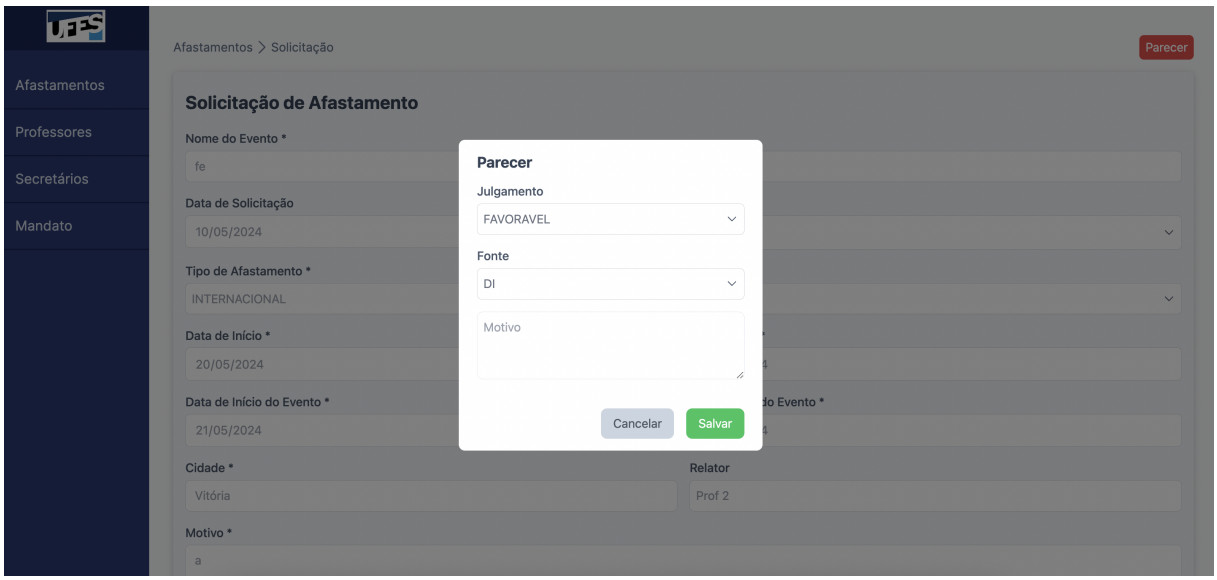

Figura 24 – Modal de Parecer do SCAP.

Por fim, o formulário de afastamento possui também um campo para *upload* de arquivos, onde o professor solicitante pode anexar o relatório de viagem e o secretário pode anexar atas ou documentos enviados pelo CT e PRPPG.

4.5.4 Cadastro de Usuários

Através do **NavBar** é possível navegar pela aplicação, clicando em **Professores** o usuário é redirecionado para a página de listagem de professores, onde é possível visualizar os professores cadastrados, assim como seus *e-mails* e telefones. A Figura 25 apresenta a tela de listagem de professores do SCAP.

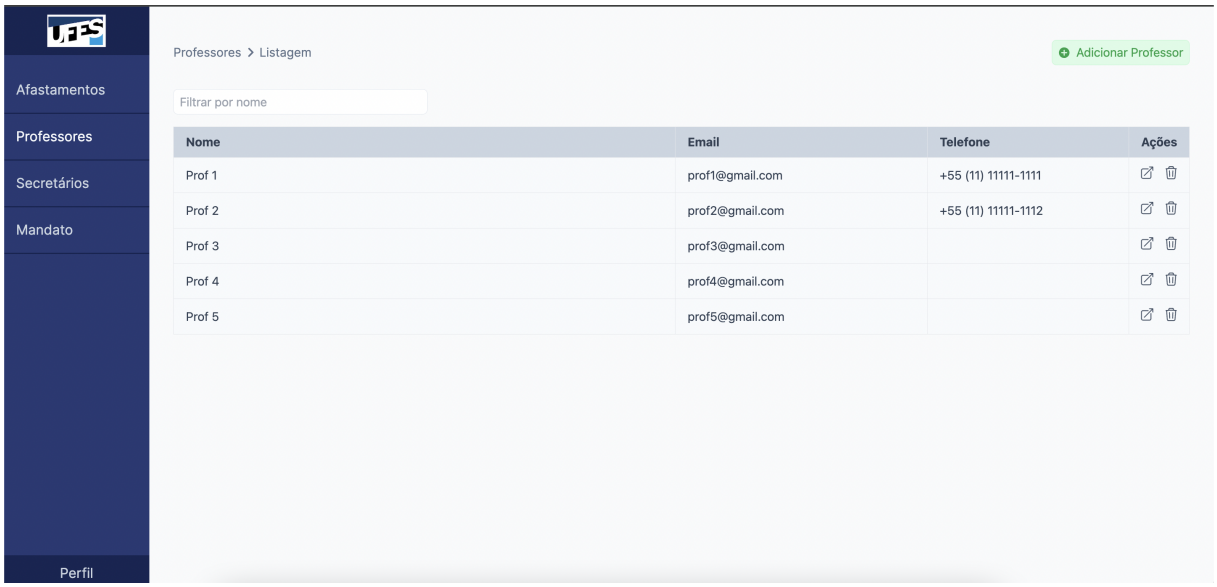

Figura 25 – Listagem de Professores do SCAP.

A Figura 26 apresenta a tela de listagem de secretários, que pode ser acessada da mesma forma, pelo **NavBar** clicando em **Secretários**.

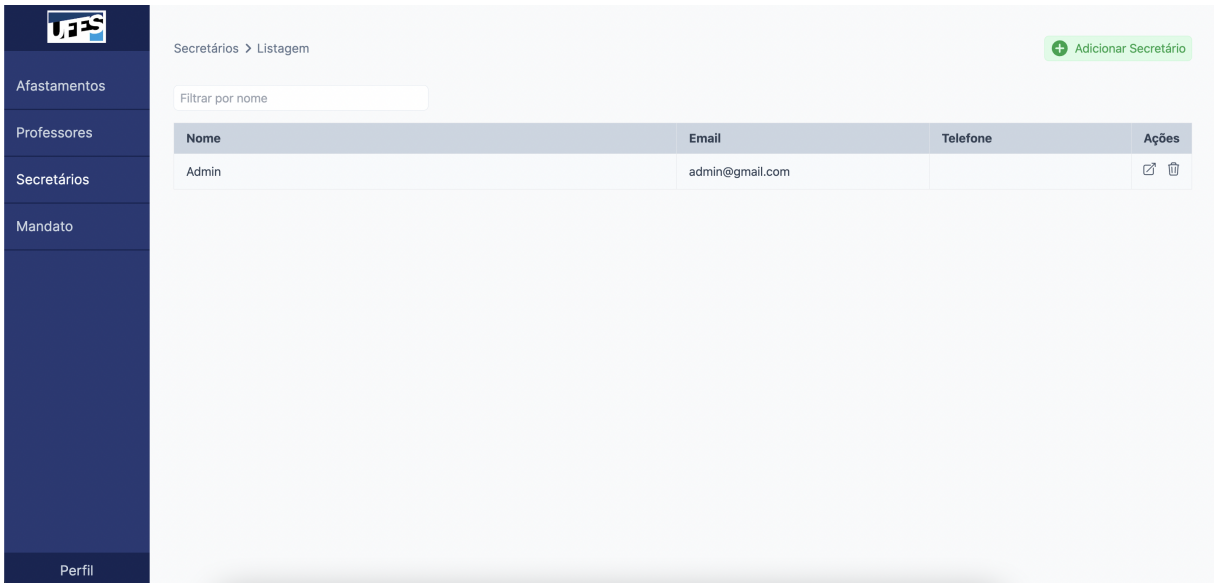

Figura 26 – Listagem de Secretários do SCAP.

Apenas o secretário pode deletar, visualizar e adicionar professores. A Figura 27 apresenta a tela de cadastro de usuários, por onde o secretário pode adicionar um novo

usuário preenchendo o formulário, seja ele um professor ou secretário. Campos marcados com asterisco são obrigatórios.

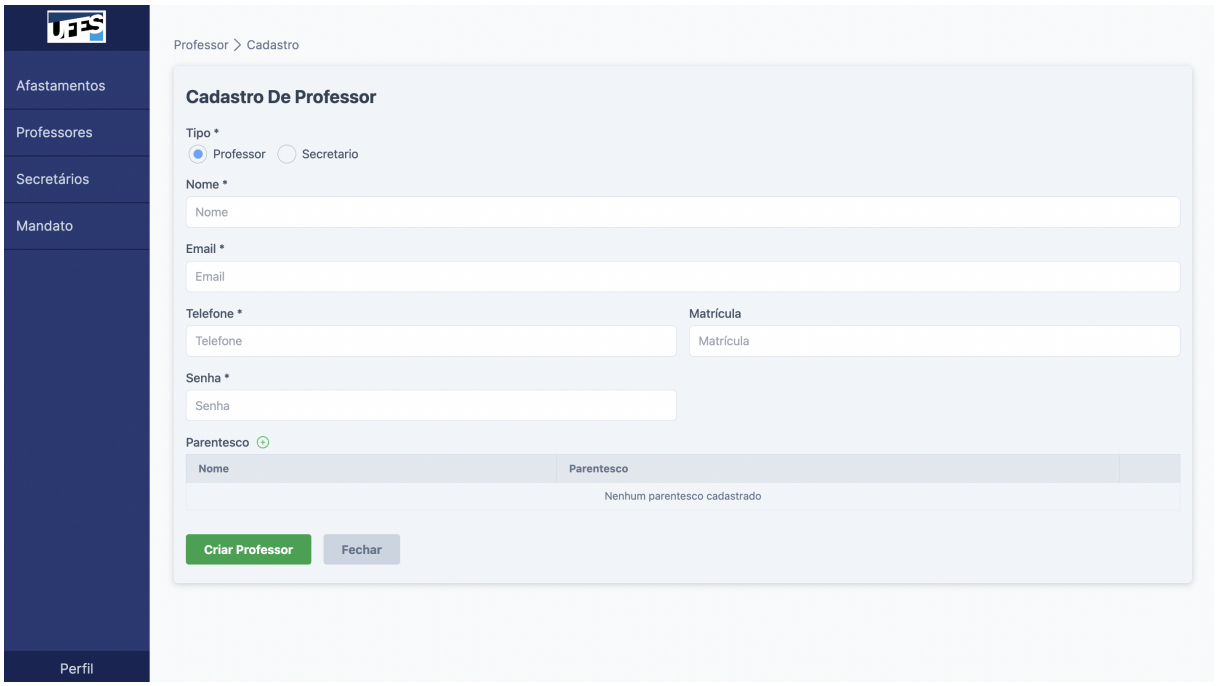

Figura 27 – Cadastro de Usuários do SCAP.

Caso o usuário a ser criado seja um professor, o secretário pode adicionar parentescos entre professores. Para isso, deve-se clicar no botão de adição que então abre um modal com um pequeno formulário. A Figura 28 apresenta este modal.

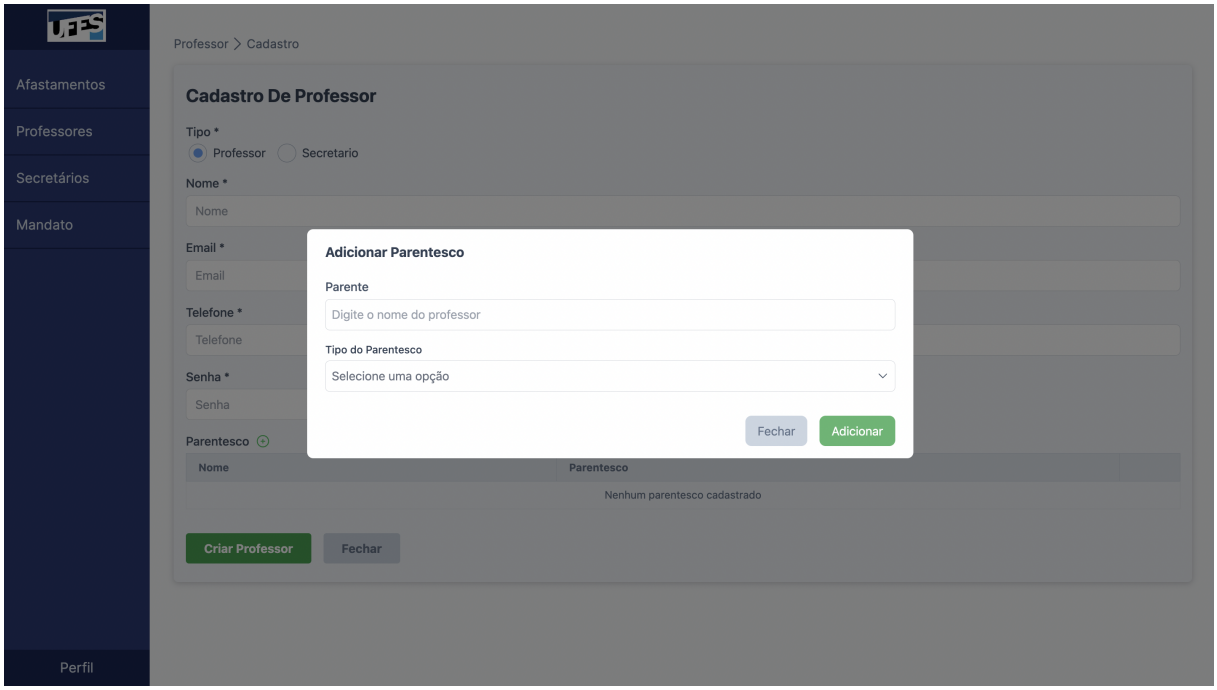

Figura 28 – Cadastro de Parentesco do SCAP.

O componente de cadastro de usuários é utilizado também para a edição de usuários, assim como foi feito com Afastamentos. A Figura 29 apresenta a página de usuário preenchida com os dados do professor logado no momento. É por essa página também que o usuário pode alterar sua senha ou realizar o *logout* do sistema.

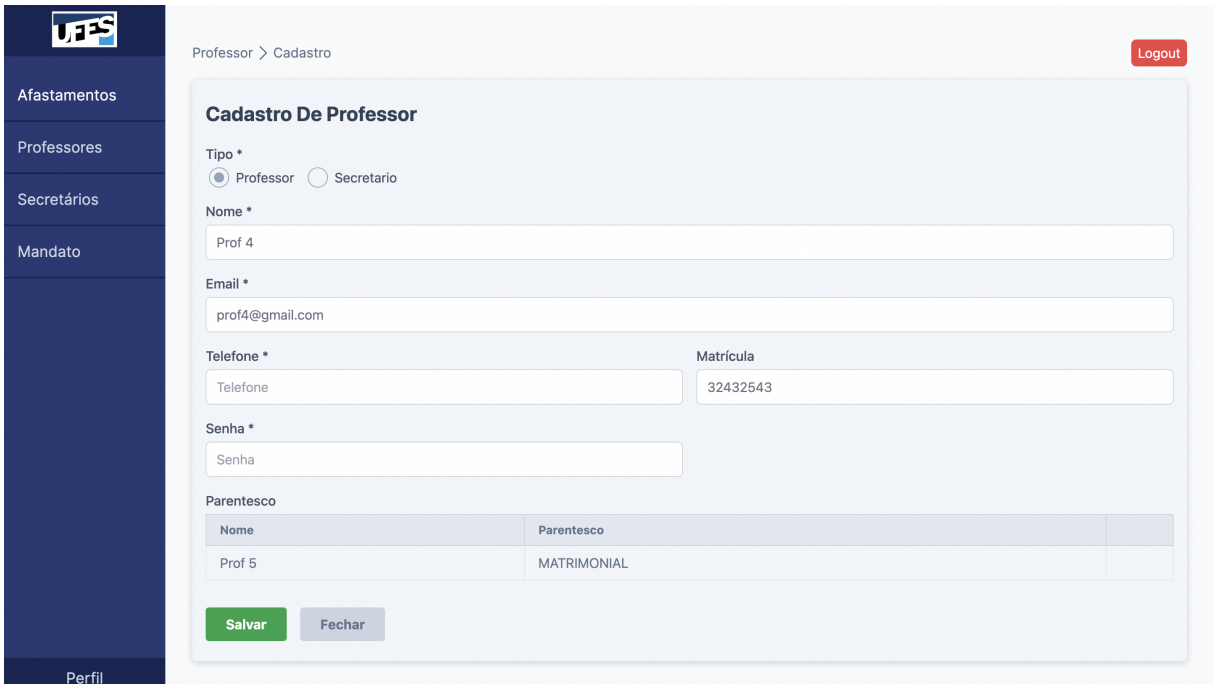

Figura 29 – Edição de Professor do SCAP.

4.5.5 Cadastro de Mandato

O secretário é responsável por gerenciar os mandatos dos professores, funções que estes exercem no departamento. A Figura 30 apresenta a tela de cadastro de mandatos do SCAP. Nela é possível visualizar os mandatos atuais, e para cadastrar um novo mandato deve-se finalizar o anterior. Para finalizar o mandato basta clicar no botão **Finalizar Mandato**, caso uma data de fim não seja enviada, será considerada a data de hoje. Professores também podem visualizar os mandatos atuais.

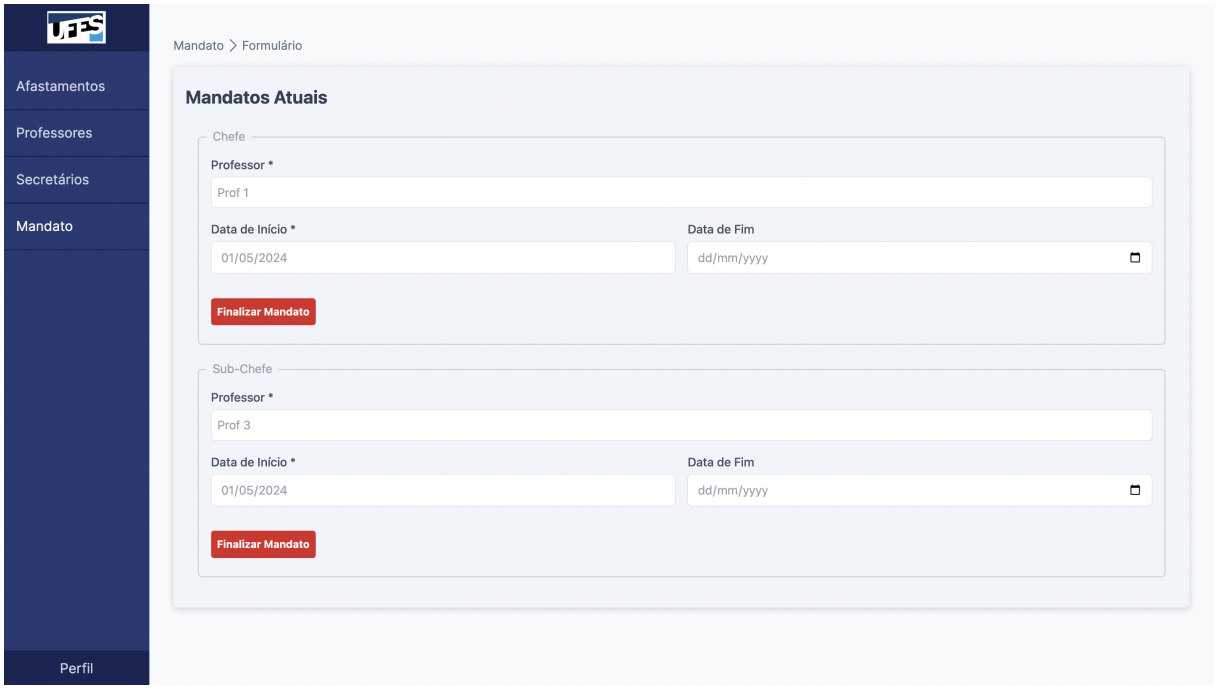

Figura 30 – Cadastro de Mandatos do SCAP.

5 Conclusão

Este capítulo apresenta as considerações finais do trabalho, dificuldades, e perspectivas para trabalhos futuros. Na Seção 5.1, são apresentadas as conclusões obtidas e suas relações com os objetivos definidos, enquanto na Seção 5.2 são apresentadas ideias e melhorias para trabalhos futuros.

5.1 Considerações Finais

Neste projeto uma nova implementação do SCAP aplicando o método FrameWeb foi realizada, dessa vez utilizando o *framework* Next.js. Os modelos FrameWeb produzidos foram essenciais para a implementação do SCAP, pois forneceram uma visão geral da aplicação, facilitando e agilizando o desenvolvimento do sistema. Mesmo assim, ao longo da implementação os modelos foram revisitados diversas vezes, e inconsistências tiveram de ser consertadas.

A aplicação do FrameWeb utilizando o Next.js demonstrou que o método é flexível e pode ser adaptado para atender às necessidades de *frameworks* modernos, proporcionando uma estrutura organizada e eficiente para o desenvolvimento tanto de aplicação monolíticas quanto de aplicações com microsserviços. De fato, utilizar uma linguagem tipada como TypeScript, juntamente com o Next.js, facilitou a modelagem dos diagramas, quando comparado com as implementações feitas em JavaScript (GOMES, 2022; DALAPICOLA, 2021).

Diversos trabalhos anteriores apontaram mudanças necessárias para o modelo de Navegação (DUARTE, 2014; PRADO, 2015; FERREIRA, 2018; AVELAR, 2018). Neste trabalho no entanto, foram utilizadas as adaptações realizadas por Hoppe e Souza (2023) ao modelo FrameWeb para *frameworks* SPA, o que facilitou a modelagem dos diagramas de navegação, principalmente. Assim, não foi necessário fazer nenhuma alteração aos modelos propostos pelo FrameWeb, explicitando a robustez do método.

Os objetivos definidos no Capítulo 1 estão listados abaixo.

- Compreender o método FrameWeb;
- Analisar os requisitos da aplicação SCAP;
- Implementar a aplicação SCAP utilizando o *framework* Next.js e o método FrameWeb.

Foi possível estudar e compreender o método FrameWeb para aplicá-lo na implementação do SCAP, considerando os requisitos levantados anteriormente. Com isso,

as disciplinas de Engenharia de Software foram revisitadas e colocadas em prática, e o conhecimento adquirido foi essencial para o andamento deste projeto. Além disso, o projeto permitiu o aprendizado de um novo e moderno *framework* de desenvolvimento, o Next.js, por meio da implementação do sistema SCAP. Assim, os objetivos foram totalmente satisfeitos.

Algumas dificuldades apareceram ao longo da implementação, isso pois foram encontradas poucas referências de projetos utilizando padrões, como *Repository Pattern* ou DAO, com o *framework* Next.js. Em sua maioria o Next.js é utilizado apenas como *frontend*, no entanto neste projeto ele foi utilizado como *fullstack*.

Tendo em vista que o foco do projeto era a aplicação e análise do método FrameWeb, duas funcionalidades não foram implementadas, sendo elas: o envio de *e-mails* e a geração de atas. Ambas funcionalidades são complexas e demandariam um tempo maior para serem implementadas.

Apesar do método FrameWeb utilizar UML, o que facilitaria a modelagem dos diagramas, a ferramenta utilizada para a modelagem, o *Visual Paradigm*, tem uma curva de aprendizado acentuada, o que inicialmente dificultou a modelagem dos diagramas. Além disso, a ferramenta possui suporte para a geração de código a partir dos diagramas, porém houve uma grande dificuldade de usá-la, o que fez com que a implementação fosse feita manualmente.

5.2 Trabalhos Futuros

Para trabalhos futuros, seria interessante analisar se os requisitos de fato contemplam as necessidades do departamento, implementar as funcionalidades que ficaram pendentes, e fazer o *deploy* da aplicação em um servidor.

Além disso, seria também interessante utilizar as ferramentas de modelagem, sejam elas o Editor FrameWeb (CAMPOS, 2017) ou o *Visual Paradigm* com o plugin desenvolvido (SILVA, 2023), com aplicações utilizando JavaScript, assim como testar o gerador de código.

A grande maioria das aplicações, dos trabalhos anteriores, apresentados na Tabela 4, foram feitas utilizando *frameworks* Java, JavaScript ou PHP. Portanto, para ampliar o campo da pesquisa, seria interessante implementar o sistema também com *frameworks* Python,[1] como Django, ou com *frameworks* Ruby,[2] como Ruby on Rails, linguagens essas muito utilizados no mercado atual. Por fim, para deixar o modelo mais robusto, é importante comparar todos os resultados obtidos ao longo de todas as diferentes implementações feitas.

[1] Python, <https://www.python.org>
[2] Ruby, <https://www.ruby-lang.org/en/>

Tabela 4 – Listagem de implementações do SCAP.

Trabalho	***Framework* MVC**	**Tecnologia ORM**	**Tecnologia DI**
Duarte (2014)	JSF (Java)	Hibernate (Data Mapper)	CDI
Prado (2015)	VRaptor (Java)	Hibernate (Data Mapper)	CDI
Pinheiro (2017)	Laravel (PHP)	Eloquent (Active Record)	Não utilizou
Matos (2017)	Spring (Java)	Hibernate (Data Mapper)	Embutido no Spring
Matos (2017)	Vaadin (Java)	Hibernate (Data Mapper)	Não utilizou
Avelar (2018)	Ninja (Java)	Hibernate (Data Mapper)	Google Guice
Ferreira (2018)	Wicket (Java)	Hibernate (Data Mapper)	CDI
Ferreira (2018)	Tapestry (Java)	Hibernate (Data Mapper)	Embutido no Tapestry
Meirelles (2019)	CodeIgniter (PHP)	Embutido no CodeIgniter (Active Record)	Não utilizou
Meirelles (2019)	NodeJS (JavaScript)	Não utilizou	Não utilizou
Guterres (2019)	Play (Scala)	Slick (Padrão não identificado)	Não utilizou
Dalapicola (2021)	AdonisJS e Quasar (Javascript)	Lucid (Active Record)	Embutido no AdonisJS
Berger (2021)	Yii (PHP)	Yii (Data Mapper)	Embutido no Yii
Berger (2021)	Symfony (PHP)	Doctrine (Active Record)	Embutido no Symfony
Gomes (2022)	Angular e NodeJS(JavaScript)	TypeORM (Data Mapper)	Embutido no Angular
Este	Next.js (TypeScript)	Prisma (Data Mapper)	Não utilizou

Referências

ALUR, D.; CRUPI, J.; MALKS, D. *Core J2EE patterns: best practices and design strategies*. [S.l.]: Prentice Hall Professional, 2003. Citado na página 16.

AVELAR, R. A. dos. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Ninja*. [S.l.], 2018. Vitória, ES, Brasil. Citado 2 vezes nas páginas 47 e 49.

BAUER, C.; KING, G. *Hibernate in Action*. Manning, 2005. (In Action Series). ISBN 9781932394153. Disponível em: <https://books.google.com/books?id=WCmSQgAACAAJ>. Citado na página 18.

BEDER, D. M. *Engenharia Web: uma abordagem sistemática para o desenvolvimento de aplicações web*. [s.n.], 2017. 215 p. Disponível em: <http://hdl.handle.net/123456789/2782>. Citado 2 vezes nas páginas 10 e 15.

BERGER, V. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Yii e Symfony*. [S.l.], 2021. Vitória, ES, Brasil. Citado 2 vezes nas páginas 11 e 49.

CAMPOS, S. L. *FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb*. Vitória, ES, Brasil, 2017. Citado na página 48.

DALAPICOLA, E. G. *Aplicação do método FrameWeb nodesenvolvimento do sistema SCAP utilizando os frameworks Quasar e AdonisJS*. [S.l.], 2021. Vitória, ES, Brasil. Citado 3 vezes nas páginas 11, 47 e 49.

DUARTE, B. B. *Aplicação do Método FrameWeb no Desenvolvimento de um Sistema de Informação na Plataforma Java EE 7*. [S.l.], 2014. Vitória, ES, Brasil. Citado 8 vezes nas páginas 11, 12, 21, 22, 24, 25, 47 e 49.

FALBO, R. d. A. *Engenharia de Software*. 2014. Disponível em: <http://www.inf.ufes.br/~jssalamon/wp-content/uploads/disciplinas/engsoft/Notas_Aula_Engenharia_Software_Falbo_2014.pdf>. Citado na página 14.

FALBO, R. d. A. *Engenharia de Requisitos*. [s.n.], 2017. 178 p. Disponível em: <http://www.inf.ufes.br/~vitorsouza/falbo/Notas_Aula_Engenharia_Requisitos_Falbo_2017.pdf>. Citado na página 13.

FALBO, R. de A. *Projeto de Sistema de Sofware*. 2018. Notas de Aula. Disponível em: <http://www.inf.ufes.br/~jssalamon/wp-content/uploads/disciplinas/projsistsoft/Notas_Aula_Projeto_Sistemas_2018.pdf>. Citado na página 27.

FERREIRA, M. T. *Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Wicket e Tapestry*. [S.l.], 2018. Vitória, ES, Brasil. Citado 2 vezes nas páginas 47 e 49.

FOWLER, M. *Patterns of Enterprise Application Architecture*. [S.l.]: Addison-Wesley Professional, 2002. 560 p. ISBN 0-321-12742-0. Citado na página 15.

GAMMA, E. et al. *Padrões de Projeto: soluções reutilizáveis de software orientado a objetos*. 1. ed. Porto Alegre: Bookman, 2000. v. 1. Tradução Luiz A. Meirelles Salgado. ISBN 978-85-7780-046-9. Citado na página 17.

GIBBS, W. Software's chronic crisis. *Scientific American - SCI AMER*, v. 271, p. 86–95, 09 1994. Citado na página 14.

GOMES, D. F. *Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando o framework Angular*. [S.l.], 2022. Vitória, ES, Brasil. Citado 3 vezes nas páginas 11, 47 e 49.

GUTERRES, C. S. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o framework Play*. [S.l.], 2019. Vitória, ES, Brasil. Citado na página 49.

HARRIS, C. *Microservices vs. Monolith architecture*. 2024. <https://www.atlassian.com/microservices/microservices-architecture/microservices-vs-monolith>. Acessado em 24/07/2024. Citado na página 20.

HIBERNATE. *What is Object/Relational Mapping?* 2010. <https://hibernate.org/orm/what-is-an-orm/>. Acessado em 25/02/2024. Citado na página 18.

HOPPE, P. H. B.; SOUZA, V. E. a. S. Support for single page application frameworks on frameweb. In: *Proceedings of the 29th Brazilian Symposium on Multimedia and the Web*. New York, NY, USA: Association for Computing Machinery, 2023. (WebMedia '23), p. 260–268. ISBN 9798400709081. Disponível em: <https://doi.org/10.1145/3617023.3617059>. Citado 4 vezes nas páginas 6, 17, 33 e 47.

KONSHIN, K. *Next.js Quick Start Guide: Server-side rendering done right*. Packt Publishing, 2018. ISBN 9781788995849. Disponível em: <https://books.google.com.br/books?id=-rBmDwAAQBAJ>. Citado na página 18.

MATOS, R. P. de. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando os frameworks Spring MVC e Vaadin*. [S.l.], 2017. Vitória, ES, Brasil. Citado na página 49.

MEIRELLES, L. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Codeigniter e NodeJS*. [S.l.], 2019. Vitória, ES, Brasil. Citado na página 49.

MURUGESAN, S. et al. Web engineering: A new discipline for development of web-based systems. In: . [S.l.: s.n.], 2001. p. 3–13. ISBN 978-3-540-42130-6. Citado 3 vezes nas páginas 10, 14 e 15.

OLSINA, L.; LAFUENTE, G.; ROSSI, G. Specifying quality characteristics and attributes for websites. In: _____. *Web Engineering: Managing Diversity and Complexity of Web Application Development*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 266–278. ISBN 978-3-540-45144-0. Disponível em: <https://doi.org/10.1007/3-540-45144-7_26>. Citado na página 15.

PINHEIRO, F. G. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando um framework PHP e um framework JavaScript*. Vitória, ES, Brasil, 2017. Citado na página 49.

PRADO, R. C. d. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework VRaptor 4*. [S.l.], 2015. Vitória, ES, Brasil. Citado 10 vezes nas páginas 5, 11, 12, 21, 22, 23, 24, 25, 47 e 49.

PRESSMAN, R. S. *Engenharia de Software - Uma Abordagem Profissional*. 8. ed. [S.l.]: McGraw-Hill, 2011. ISBN 9788580555349. Citado 2 vezes nas páginas 13 e 14.

PRISMA. *What are database migrations?* 2020. <https://www.prisma.io/dataguide/types/relational/what-are-database-migrations#what-are-database-migrations>. Acessado em 29/05/2024. Citado na página 38.

SCHMIDT, D. C.; GOKHALE, A.; NATARAJAN, B. Leveraging application frameworks: Why frameworks are important and how to apply them effectively. *Queue*, Association for Computing Machinery, New York, NY, USA, v. 2, n. 5, p. 66–75, jul 2004. ISSN 1542-7730. Disponível em: <https://doi.org/10.1145/1016998.1017005>. Citado na página 17.

SCOTT, E. *Design and Architecture: Understanding Single Page Web Applications*. 1. ed. Manning Publications Co., 2015. Disponível em: <https://livebook.manning.com/book/spa-design-and-architecture/chapter-1>. Citado 2 vezes nas páginas 17 e 18.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Database Systems Concepts*. 7th. ed. [S.l.]: McGraw-Hill Higher Education, 2019. ISBN 9780078022159. Citado na página 18.

SILVA, I. S. *Desenvolvimento de plugin para o Visual Paradigm com suporte ao método FrameWeb*. [S.l.], 2023. Vitória, ES, Brasil. Citado na página 48.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. [S.l.]: Pearson Education do Brasil, 2011. ISBN 9788579361081. Citado 2 vezes nas páginas 13 e 14.

SOUZA, V. E. S. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Dissertação (Mestrado), 2007. Disponível em: <http://portais.ufes.br/PRPPG/ext/mono.php?progpess=2032&curso=9&prog=30001013007P0>. Citado 10 vezes nas páginas 10, 11, 12, 15, 16, 17, 28, 32, 33 e 36.

SOUZA, V. E. S. The FrameWeb Approach to Web Engineering: Past, Present and Future. In: ALMEIDA, J. P. A.; GUIZZARDI, G. (Ed.). *Engineering Ontologies and Ontologies for Engineering*. 1. ed. Vitória, ES, Brazil: NEMO, 2020. cap. 8, p. 100–124. ISBN 9781393963035. Disponível em: <http://purl.org/nemo/celebratingfalbo>. Citado 3 vezes nas páginas 5, 12 e 16.

TYPESCRIPT. *The TypeScript Handbook*. 2024. <https://www.typescriptlang.org/docs/handbook/intro.html>. Acessado em 25/02/2024. Citado na página 20.

VERCEL. *Deploying*. 2023. <https://nextjs.org/learn/basics/deploying-nextjs-app>. Acessado em 10/10/2023. Citado 2 vezes nas páginas 19 e 20.

VERCEL. *How Next.js Works*. 2023. <https://nextjs.org/learn/foundations/how-nextjs-works/rendering>. Acessado em 25/09/2023. Citado na página 19.