UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

COLEGIADO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Lucas Ribeiro Mendes Silva

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação com ASP.NET Core

Vitória, ES

2021

Lucas Ribeiro Mendes Silva

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação com ASP.NET Core

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

2021

Lucas Ribeiro Mendes Silva
Aplicação do método FrameWeb no desenvolvimento de um sistema de informação com ASP.NET Core/ Lucas Ribeiro Mendes Silva. – Vitória, ES, 2021-
99 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, 2021.

1. FrameWeb. 2. ASP.NET Core. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. Aplicação do método FrameWeb no desenvolvimento de um sistema de informação com ASP.NET Core
CDU 02:141:005.7

Lucas Ribeiro Mendes Silva

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação com ASP.NET Core

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, (dia) de (mês) de (ano):

Prof. Dr. Vítor E. Silva Souza
Orientador

Prof. Dr. João Paulo Andrade Almeida
Universidade Federal do Espírito Santo

Bruno Borlini Duarte
Universidade Federal do Espírito Santo

Vitória, ES
2021

Dedico este trabalho aos meus pais, por todo amor e suporte que me deram ao longo da minha trajetória acadêmica.

Agradecimentos

A minha família, por todo amor e apoio aos estudos que me deram desde cedo. Também aos meus amigos, que sempre estiveram comigo nos melhores e piores momentos da vida e do curso, fazendo com que eu pudesse chegar até este momento com um sorriso no rosto e um sentimento de dever cumprido.

Ao meu orientador Vítor E. Souza, pelos muitos ensinamentos, dedicação nas revisões da monografia e momentos de descontração, mesmo diante de um cenário atípico de pandemia a qual vivemos na data corrente deste trabalho.

A todos os professores do curso de Ciência da Computação da UFES, que sempre se mostraram dispostos para me atender, e colaboraram com meu crescimento como pessoa e profissional. Com destaque para a professora Jordana Salamon, pelas diversas vezes que me atendeu fora de horário e até mesmo nos finais de semana, tirando dúvidas e fornecendo dicas para melhor elaboração deste trabalho.

Especialmente ao professor Ricardo de Almeida Falbo (in memorian), que não está mais entre nós, mas a quem tenho grande respeito e admiração como pessoa e profissional. Falbo foi uma fonte de conhecimento e inspiração para mim, me incentivando a não desistir no momento que mais duvidei da minha capacidade como profissional.

"Há um grande desejo em mim de sempre melhorar. Melhorar. É o que me faz feliz."
(Ayrton Senna)

Resumo

Com a ascenção da Internet como um dos principais meios de comunicação existentes, primordialmente a *World Wide Web* como plataforma para implantação de Sistemas de Informação Web (*Web Information Systems* ou WISs), surgiu a demanda por *softwares* de qualidade que atendessem critérios de segurança, usabilidade, portabilidade, desempenho e outros. Através de conceitos de Engenharia de Software, foram criados métodos e ferramentas para auxiliarem no processo de desenvolvimento, implantação e manutenção desses *softwares*, assim nascendo a Engenharia Web.

Foi proposto, então, o FrameWeb, um método baseado em *frameworks* para desenvolvimento de WISs. FrameWeb define uma arquitetura padrão para facilitar a integração com *frameworks* de desenvolvimento Web, além de propor uma linguagem para construção de um conjunto de modelos, que trazem para o projeto arquitetural do sistema conceitos inerentes a estes *frameworks*. O método ainda conta com o apoio de ferramentas próprias para diagramação e geração de código dos modelos propostos, nomeadas como FrameWeb *Tools*.

Este trabalho tem como objetivo desenvolver uma nova revisão e implementação do Sistema de Controle de Afastamentos de Professores (SCAP), aplicando o método FrameWeb, apoiado por seu conjunto de ferramentas, de modo a avaliar sua eficiência no desenvolvimento de um WIS na plataforma ASP.NET Core.

Palavras-chaves: Engenharia Web. FrameWeb. ASP.NET Core.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UML	Unified Modeling Language
SCAP	Sistema de Controle de Afastamento de Professores
DAO	Domain Access Object
MVC	Model-View-Controller
DI	Dependency Injection
WIS	Web Information System
MDD	Model-Driven Development
ORM	Object-Relational Mapping
SGBDR	Sistemas de Gerenciamento de Banco de Dados Relacionais
EF	Entity Framework

Sumário

1 Introdução

Com a ascenção da Internet como um dos principais meios de comunicação existentes no mundo, em especial a *World Wide Web* como plataforma para implantação de Sistemas de Informação Web (*Web Information Systems* ou WISs), surgiu a demanda por *softwares* de qualidade que atendam requisitos como segurança, usabilidade, portabilidade, desempenho e outros. Consequentemente, através de conceitos de Engenharia de Software, foram criados métodos e ferramentas para auxiliarem no processo de desenvolvimento, implantação e manutenção desses *softwares*, inaugurando, então, a Engenharia Web (SOUZA; FALBO, 2005).

Durante o processo de desenvolvimento de WISs, é comum a utilização de *frameworks*, artefatos de código que proveem componentes prontos para reuso. Porém, apesar de terem influência direta na arquitetura dos sistemas, é comum que não sejam considerados até a fase de implementação. Foi proposto, então, o FrameWeb (SOUZA; FALBO; GUIZZARDI, 2009), um método baseado em *frameworks* para desenvolvimento de WISs. FrameWeb define uma arquitetura padrão para facilitar a integração com *frameworks* de desenvolvimento Web, além de propor uma linguagem (MARTINS; SOUZA, 2015) para construção de um conjunto de modelos, que trazem para o projeto arquitetural do sistema conceitos inerentes a estes *frameworks*.

Foram desenvolvidas duas ferramentas de apoio ao método, sendo elas um editor de diagramas e um gerador de código. O *FrameWeb Editor* (CAMPOS; SOUZA, 2017) permite criar diagramas que atendam aos modelos propostos pelo método, de forma gráfica. Já o gerador de código (ALMEIDA; CAMPOS; SOUZA, 2017; ZUPELI; SOUZA, 2018) é capaz realizar transformações dos modelos criados no *FrameWeb Editor* em código fonte.

O uso de técnicas MDD (*Model-Driven Development*) adotadas pelas ferramentas associadas ao FrameWeb permitem que seus modelos utilizem conceitos de categorias de *frameworks* (ex.: mapeamento objeto/relacional) sem que se comprometam com *frameworks* específicos (ex.: Hibernate, Eloquent, Sequelize, etc.) até a fase de implementação. O gerador de código, por sua vez, pode ser estendido através de *templates*. Estes servem como base para a geração de código em diferentes linguagens e *frameworks*, que refletem os modelos FrameWeb construídos no editor.

Neste trabalho será desenvolvida uma nova revisão e implementação do Sistema de Controle de Afastamentos de Professores (SCAP), assim como foi feito em (DUARTE, 2014) e (PRADO, 2015), utilizando o método FrameWeb, apoiado por seu conjunto de ferramentas, mas atendendo à plataforma ASP.NET Core (MICROSOFT, 2019), de modo a avaliar a viabilidade e aplicabilidade do método e seu ferramental a esta plataforma.

1.1 Motivação e Justificativa

Para avaliar a aplicabilidade do FrameWeb em diversas plataformas e conjuntos de *frameworks*, foram modelados e desenvolvidos WISs para estas plataformas. Em particular, Duarte (2014) e Prado (2015) realizaram o levantamento e análise de requisitos do SCAP, desenvolvendo-o através do método FrameWeb, utilizando a plataforma Java EE 7 e o *framework* VRaptor 4, respectivamente.

Sucessivamente, outros trabalhos reimplementaram o SCAP utilizando os *frameworks* Java Spring MVC e Vaadin (MATOS, 2017), Ninja (AVELAR, 2018), Wicket e Tapestry (FERREIRA, 2018) e Play (GUTERRES, 2019) na plataforma Java, bem como os *frameworks* Laravel (PINHEIRO, 2017), Code Igniter e Node.JS (MEIRELLES, 2019) em outras plataformas. Estes trabalhos trazem análises do método FrameWeb através do desenvolvimento do SCAP utilizando os *frameworks* escolhidos.

ASP.NET Core (MICROSOFT, 2019) vem ganhando destaque e crescendo cada vez mais no mercado de WISs.[1] Porém, até a data corrente deste trabalho, não foram feitos experimentos do método FrameWeb nesta plataforma, de modo a avaliar o suporte por seu ferramental. Sendo assim, este trabalho tem como motivação incluir o *framework* da Microsoft na lista de compatibilidade do editor e do gerador de código FrameWeb, por meio da criação de *templates* apropriados para a plataforma e os testando a partir do desenvolvimento do SCAP. Todas as linguagens de programação e *frameworks* suportados pelo FrameWeb Editor estão disponíveis no repositório oficial do projeto.[2]

1.2 Objetivos

Diferente de trabalhos passados, feitos majoritariamente em Java e sem o apoio das ferramentas FrameWeb, este trabalho tem como objetivo geral aplicar o método FrameWeb, com apoio de seu conjunto de ferramentas, em uma nova implementação do SCAP, mas desta vez feita em C#, utilizando a plataforma ASP.NET Core. Baseando-se nos requisitos levantados por Duarte (2014) e Prado (2015), a documentação do SCAP será revisada, quando necessário, de modo a torná-la mais consistente e detalhada.

Os objetivos principais deste trabalhos são: prover análises dos resultados alcançados nas etapas de modelagem e geração de código até a total implementação do SCAP, utilizando ASP.NET Core, indicando as vantagens e desvantagens do uso do método FrameWeb e suas ferramentas no desenvolvimento de um WIS, propondo melhorias e novas funcionalidades para trabalhos futuros.

[1] <https://insights.stackoverflow.com/survey/2020/#most-popular-technologies>
[2] <https://github.com/nemo-ufes/FrameWeb>

1.3 Método de Desenvolvimento do Trabalho

Para alcançar os objetivos propostos, foram desenvolvidas as seguintes atividades:

- **Atividade 1**

  Estudo do método FrameWeb (SOUZA; FALBO; GUIZZARDI, 2009) e de seu ferramental (MARTINS; SOUZA, 2015; CAMPOS; SOUZA, 2017; ZUPELI; SOUZA, 2018), revisão dos padrões e técnicas de Engenharia de Software presentes nas notas de aula do professor doutor Ricardo de A. Falbo (2014, 2017, 2018), estudo dos trabalhos de experimentação do método FrameWeb, em especial os resultados e documentações levantadas por Duarte (2014) e Prado (2015).

- **Atividade 2**

  Elaboração dos documentos de Definição e Especificação de Requisitos do SCAP. Ambos documentos foram baseados nos trabalhos de Duarte (2014) e Prado (2015), sendo revisadas as partes que apresentavam incompletudes ou inconsistências.

- **Atividade 3**

  Elaboração dos *templates* FrameWeb responsáveis pela transformação M2T (*Model to Text*) dos diagramas feitos no *FrameWeb Editor* para código fonte em C#. Os códigos gerados foram analisados e executados na plataforma ASP.NET Core para fins de comprovação de funcionamento e equivalência com os modelos de teste.

- **Atividade 4**

  Através do FrameWeb Editor, foram feitos os modelos FrameWeb referentes ao SCAP, obedecendo ao máximo a documentação da aplicação.

- **Atividade 5**

  Com o código gerado, o ambiente de desenvolvimento foi preparado para que o sistema pudesse ser implementado. Enquanto a aplicação estava em processo de desenvolvimento, foram feitas análises de eficiência da utilização do método FrameWeb e de seu ferramental.

- **Atividade 6**

  Elaboração da Monografia e da apresentação final, baseadas nos conhecimentos adquiridos durante o curso de Ciência da Computação e do Projeto de Graduação.

- **Atividade 7**

  Apresentação do trabalho para a banca.

- **Observações**

Para o desenvolvimento e organização do trabalho, foi empregado o uso do Kanban, técnica que consiste num sistema de controle e gestão do fluxo de trabalho por meio do uso de cartões. Subdividindo as tarefas descritas anteriormente em tarefas menores, estas foram alocadas num quadro *To-do*, indicando que deveriam ser feitas. Quando iniciada uma tarefa, esta era realocada para o quadro *In progress*, indicando que estava sendo desenvolvida. Uma vez finalizada, a tarefa tinha como destino o quadro *Done*, assinalando seu término. O Trello (TRELLO, 2011) foi escolhido como ferramenta de auxílio na manutenção do quadro Kanban.

1.4 Organização da Monografia

Esta monografia foi particionada em 5 diferentes partes, incluindo este capítulo de Introdução. O Capítulo 2 traz informações que dizem respeito ao referencial teórico utilizado como base para a elaboração deste trabalho.

O Capítulo 3 detém informações sucintas sobre a documentação de requisitos do WIS desenvolvido neste trabalho, documentação esta que passou por revisões e aprimoramentos para melhor atender os objetivos do trabalho.

O Capítulo 4 aborda o projeto arquitetural e as etapas de implementação do SCAP, contendo também informações que dizem respeito às diferentes tecnologias utilizadas para melhor alcançar os resultados esperados, assim como o processo de modelagem e geração de código utilizando o FrameWeb e suas ferramentas.

O Capítulo 5 conclui este trabalho, trazendo as impressões sobre a experiência de desenvolvimento de um WIS, intermediada pelo método FrameWeb, falando sobre os resultados alcançados e problemas encontrados, assim como propostas de melhorias para o aprimoramento do método e seu ferramental.

2 Referencial Teórico

Neste capítulo são apresentados os referenciais teóricos estudados para o desenvolvimento do Projeto de Graduação. A Seção 2.1 apresenta a Engenharia Web, vertente *Web* da Engenharia de Software. A Seção 2.2 aborda os tipos de *frameworks* utilizados para o desenvolvimento de aplicações com o método FrameWeb e que são suportados por suas ferramentas, abordadas na Seção 2.3.

2.1 Engenharia Web

Aspectos existentes no ambiente da Web, tais como: concorrência, carga imprevisível, disponibilidade, dinamicidade, segurança, estética e etc. (PRESSMAN, 2005) trouxeram novos desafios aos processos já existentes no âmbito de desenvolvimento de *software* para *desktops*, originando uma nova sub-área da Engenharia de Software, chamada de Engenharia Web (SOUZA, 2007). A Figura 1 apresenta a árvore de critérios de qualidade que um WIS deve possuir, segundo Pressman (2005).

A Engenharia Web adota um modelo de projeto divido em diferentes fases, seguindo uma abordagem iterativa. No final de cada etapa, temos sempre artefatos importantes para a construção do WIS e, assim que terminado o projeto, temos o *software* implementado, testado, implantado e atendendo às expectativas do cliente. Um projeto de *software*, por sua vez, pode ainda passar por situações que necessitem de manutenção ou extensão de suas funcionalidades.

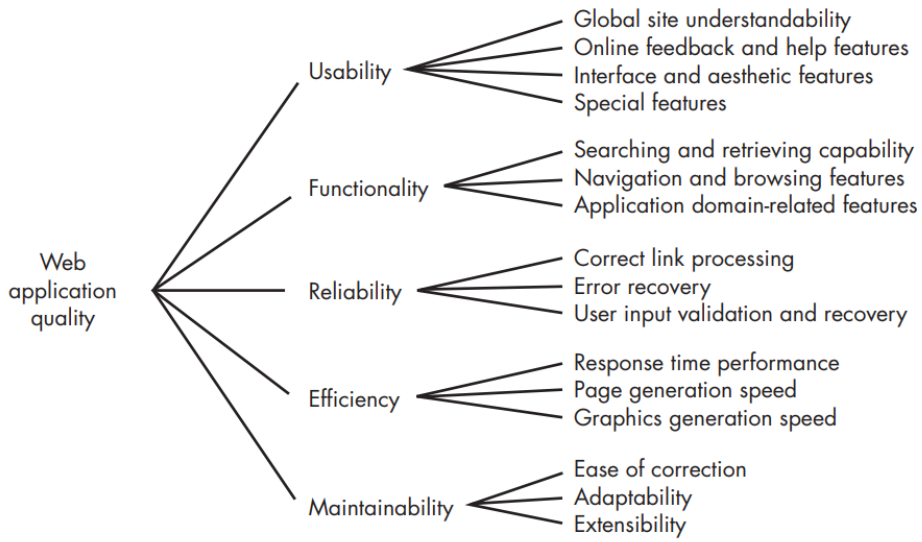

Figura 1 – Qualidades necessárias para uma aplicação Web (PRESSMAN, 2005).

A primeira etapa para o desenvolvimento de um software é entender quais são as necessidades do cliente, e para alcançar este objetivo é realizada a fase de levantamento de requisitos, junto às partes interessadas, são buscadas todas as informações relevantes possíveis a respeito das funcionalidades que o sistema a ser desenvolvido deve executar (requisitos funcionais), assim como as restrições as quais deve operar (requisitos não funcionais). O produto principal dessa fase é o Documento de Definição de Requisitos (FALBO, 2017).

Requisitos Funcionais referem-se às funcionalidades que o sistema deve prover para que atenda às necessidades do cliente. Já os Requisitos Não Funcionais descrevem as exigências técnicas ou restrições as quais o sistema deve operar (Ex: segurança, disponibilidade, portabilidade, etc.). As Regras de Negócio são um conjunto de premissas e restrições encorporadas às funcionalidades do sistema, de forma a satisfazer as políticas do negócio (Ex: leis, regulamentações, padrões de produção, etc.). Há ainda os Requisitos Negativos, responsáveis por registrar particularidades que não fazem parte do escopo do sistema, portanto não devem ser consideradas no desenvolvimento (FALBO, 2017).

Identificados os requisitos, inicia-se a fase de análise de requisitos, fase esta que resulta no refinamento dos requisitos levantados. Durante a etapa de análise de requisitos, são construídos modelos gráficos para que haja melhor compreenssão do domínio e dos processos de negócio do sistema em desenvolvimento. Essa fase serve de base para a geração do Documento de Especificação de Requisitos (FALBO, 2017).

Sucessivamente, temos a fase de implementação. Aqui o *software* é verdadeiramente programado utilizando a linguagem de programação, *frameworks* e demais tecnologias escolhidas durante a fase de análise de requisitos. A fase de testes, que deve ser feita durante todo o ciclo de implementação do WIS, visa garantir a qualidade do *software*, dando enfoque a características como usabilidade, segurança, eficiência, interoperabilidade e outros, assim como validar se o mesmo foi ou está sendo construído de acordo com as necessidades do cliente. Satisfeitos os requisitos, o WIS deve ser implantado num *web server* e disponibilizado para as partes interessadas.

2.2 Frameworks

Frameworks são conjuntos de artefatos de código para reúso, que, uma vez combinados, podem formar uma infraestrutura de uma aplicação semi-completa. Mediante configurações, composições e heranças, estes frameworks podem ser utilizados como base para o desenvolvimento de aplicações Web que resolvam problemas dos mais diversos domínios (SOUZA, 2007). Nas subseções que se seguem, são apresentados os tipos de *frameworks* que foram utilizados para o desenvolvimento deste trabalho.

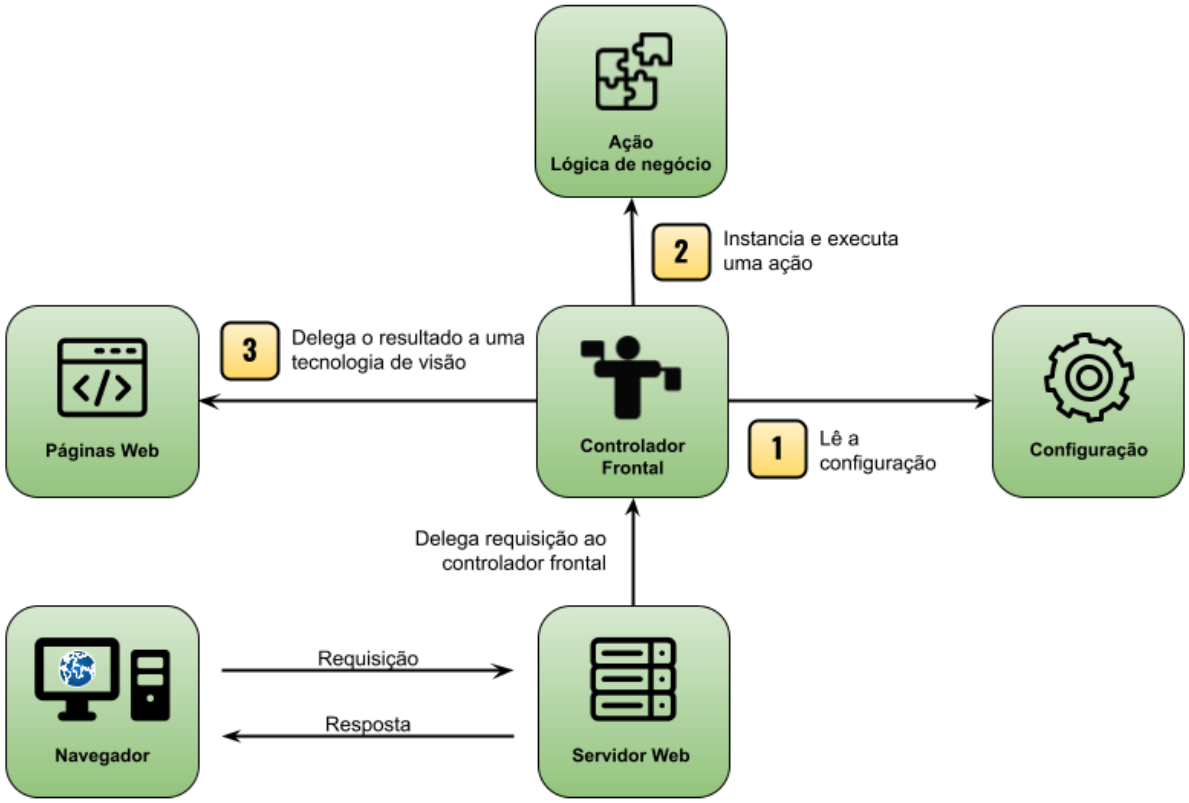

Figura 2 – Funcionamento do padrão Modelo-Visão-Controlador.

2.2.1 Frameworks MVC (*Model-View-Controller*)

O padrão arquitetural MVC (Model-View-Controller ou Modelo-Visão-Controlador) consiste na divisão da aplicação em 3 camadas, cada qual com diferentes competências. A Figura 2 apresenta o fluxo de eventos do padrão MVC.

"Elementos da visão representam informações de modelo e as exibem ao usuário, que pode enviar, por meio deles, estímulos ao sistema, que são tratados pelo controlador. Este último altera o estado dos elementos do modelo e pode, se apropriado, selecionar outros elementos de visão a serem exibidos ao usuário. O modelo, por sua vez, notifica alterações em sua estrutura à visão para que esta consulte novamente os dados e se atualize automaticamente" (SOUZA, 2007).

A camada de visão se encontra do lado do cliente, representada por páginas Web, enquanto que a camada de modelo está situada do lado de servidor, onde a aplicação Web é sustentada. Desta forma, para que a comunicação entre visão e modelo seja possível, é necessário um agente intermediador, neste caso, a camada de controle. Um *framework* MVC é aquele que disponibiliza um serviço para receber e responder requisições HTTP (*Hypertext Transfer Protocol*), possibilitando a comunicação cliente-servidor. Este serviço, por sua vez, pode ser estendido e especializado de acordo com as necessidades da aplicação e, comunentemente, oferece uma série de facilidades como: definição de rotas customizadas, conversão automática de tipos, validação de dados de entrada e etc. (SOUZA, 2007).

Frameworks MVC trazem como vantagem uma melhor modularização, coesão e manutebilidade do código da aplicação, evitando que o código da lógica de negócio esteja entremeado diretamente nas páginas Web, fazendo com que cada uma das partes cumpra apenas com as responsabilidades que lhe dizem respeito.

2.2.2 Frameworks ORM (*Object/Relational Mapping*)

Sistemas de Gerenciamento de Banco de Dados Relacionais (SGBDR) são comunentemente utilizados para armazenamento de informações na indústria de *software*. Isso se dá pelo fato de terem uma base teórica bem consolidada no campo da álgebra relacional, garantindo-lhes uma maior confiabilidade. Porém, mesmo aplicações desenvolvidas sobre o paradigma de programação orientada a objetos (POO), fazem utilização de SGBDRs para persistência de dados. Esse "choque" entre paradigmas gera um problema de compatibilidade na conversação entre as duas vertentes, conhecido como *paradigm mismatch* (ou incompatibilidade de paradigmas), visto que os bancos de dados que trabalham com o conceito de álgebra relacional, estruturam dados em tabelas, enquanto que aplicações que utilizam o paradigma POO, armazenam informações em grafos de objetos interconectados e referenciados em memória (SOUZA, 2007).

Para solucionar este problema, surgiu na década de 1980 a ideia do Mapeamento Objeto/Relacional (*Object/Relational Mapping* ou apenas ORM), que consiste na persistência de objetos em tabelas de SGBDRs de forma transparente, se dando através da utilização de meta-dados que descrevem o mapeamento entre o mundo dos objetos e o mundo relacional (BAUER; KING, 2005; SOUZA, 2007). A Figura 3 apresenta a ideia de funcionamento de um *framework* ORM.

De modo geral, ao invés do desenvolvedor interagir com o SGBDR utilizando SQL (*Structured Query Language*, ou Linguagem de Consulta Estruturada) e ter o trabalho de mapear objetos e tabelas manualmente, cabe a ele apenas informar ao *framework* ORM quais são as regras de mapeamento a serem utilizadas nos objetos (geralmente feitas com *Data Annotations* ou *Fluent API*), tornando possível a transformação deles (objetos) para o formato de tabelas e vice-versa. Além disso, as interações com o banco de dados são feitas através da execução de métodos, como *add()*, *find()*, *update()* e *remove()*, característicos de operações básicas de CRUD (*Create, Read, Update, Delete*).

2.2.3 Frameworks de Injeção de Dependência

Uma vez que um WIS se encontra particionado em diferentes camadas, cada qual com suas responsabilidades, surge a necessidade de integrá-las para que se comuniquem e o sistema funcione como esperado. Segundo Fowler (2004), uma vez que uma classe instanciada depende diretamente de serviços providos por outra classe, é interessante que

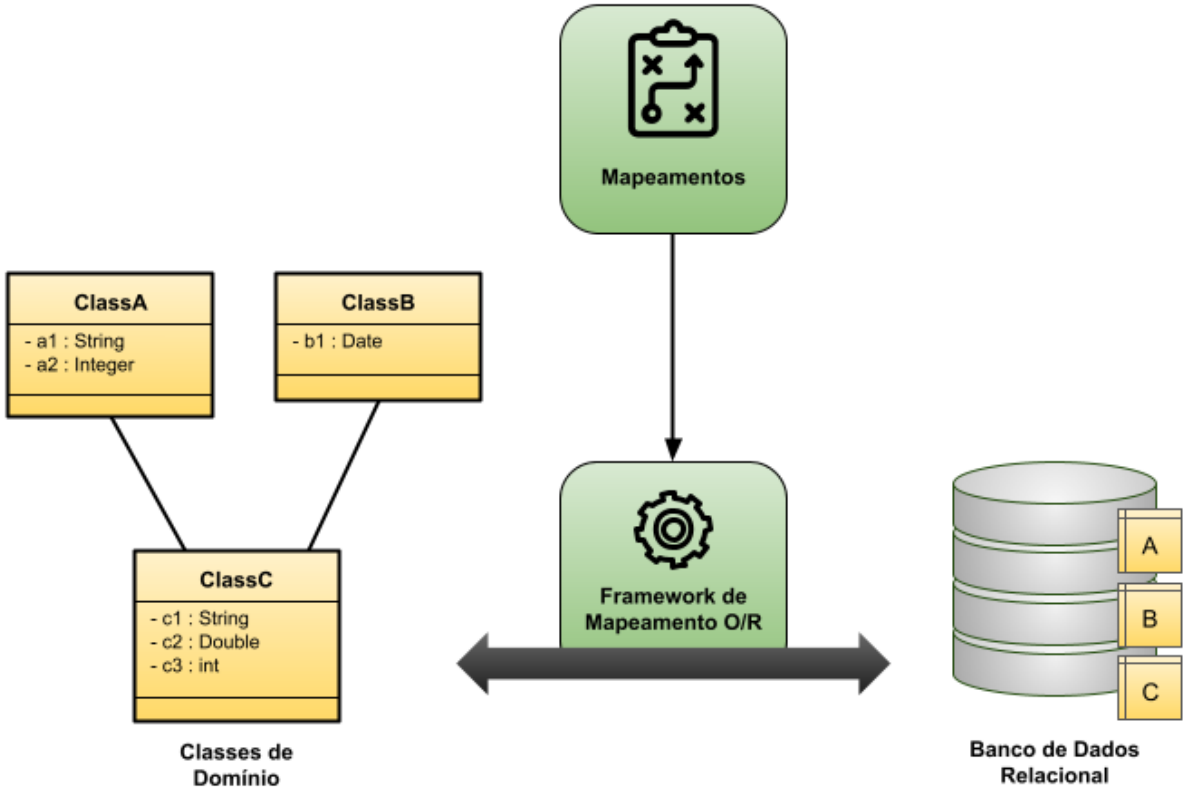

Figura 3 – Funcionamento de um *framework* ORM.

a primeira faça utilização da segunda através de um contrato, ou seja, uma interface. Isso faz com que a classe dependente não precise ter conhecimento a respeito da implementação do serviço requisitado, mas apenas dos métodos os quais a interface do serviço oferece.

Seguindo essa linha de raciocínio, surgiram os *frameworks* de Injeção de Dependência (*Dependency Injection*, ou DI), que têm como função satisfazer as dependências entre as classes do sistema, especificadas pelo próprio desenvolvedor através de configurações externas. A Figura 4 apresenta o comportamento comum de um *framework* DI.

Uma vez requisitado um contrato de serviço, seja por construtor ou via *data annotation*, cabe ao *framework* DI injetar alguma instância de classe que atende este contrato, definida previamente pelo desenvolvedor. Essa abordagem retira da classe solicitante a responsabilidade de buscar ou criar um objeto de interesse, sendo também conhecida como Inversão de Controle (*Inversion of Control*, ou IoC), fazendo com que o controle de busca e criação de objetos seja feito por um elemento externo, que neste caso é próprio *framework* DI (SOUZA, 2007).

2.2.4 Frameworks de Segurança

Uma característica comum em WISs são mecanismos de autenticação e autorização de usuários. Autenticar um usuário consiste em verificar se uma chave de acesso (geralmente

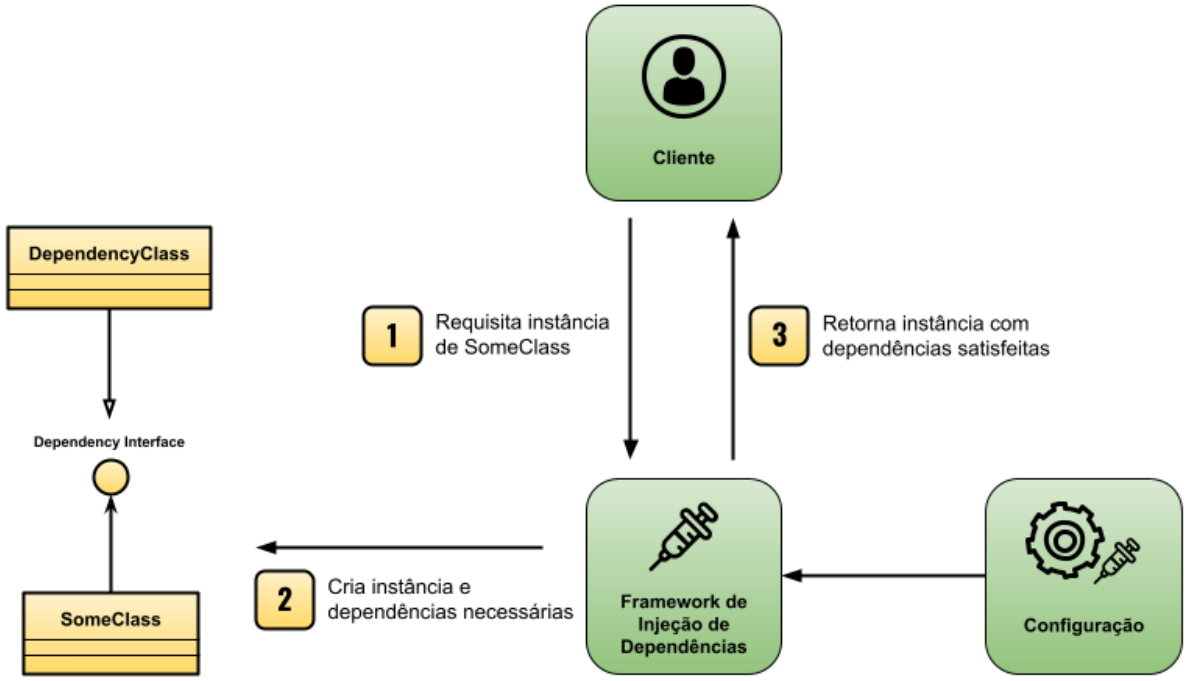

Figura 4 – Funcionamento de um *framework* DI.

um par de login e senha ou um *token*) é válida para acessar o sistema. Autorizar, por sua vez, consiste em verificar qual é o nível de acesso (ou *roles*) do usuário autenticado, permitindo que este use somente funcionalidades que lhe dizem respeito (SOUZA, 2007).

Existem *frameworks* de Segurança que proveem artifícios prontos para a utilização destes mecanismos, bastando usar a solução padrão, que geralmente não requer customizações, ou que uma classe de domínio, que represente um usuário, herde a classe de usuário disponibilizada pelo *framework*. Abaixo estão listados alguns *frameworks* de segurança para diferentes plataformas:

- Identity, para C# (ASP.NET Core);
- Spring Security, para Java (Spring);
- Laravel Auth, para PHP (Laravel).

2.3 FrameWeb

FrameWeb trata-se de um método de desenvolvimento de projetos de software, mais especificamente WISs (*Web Information Systems*). Este, por sua vez, define uma arquitetura padrão a qual, em linhas gerais, incorpora *frameworks* de diferentes naturezas, de maneira a utilizá-los concomitantemente com um conjunto de modelos de projeto propostos, fazendo com que tais modelos tornem-se mais ricos e próximos da fase de implementação (SOUZA, 2020).

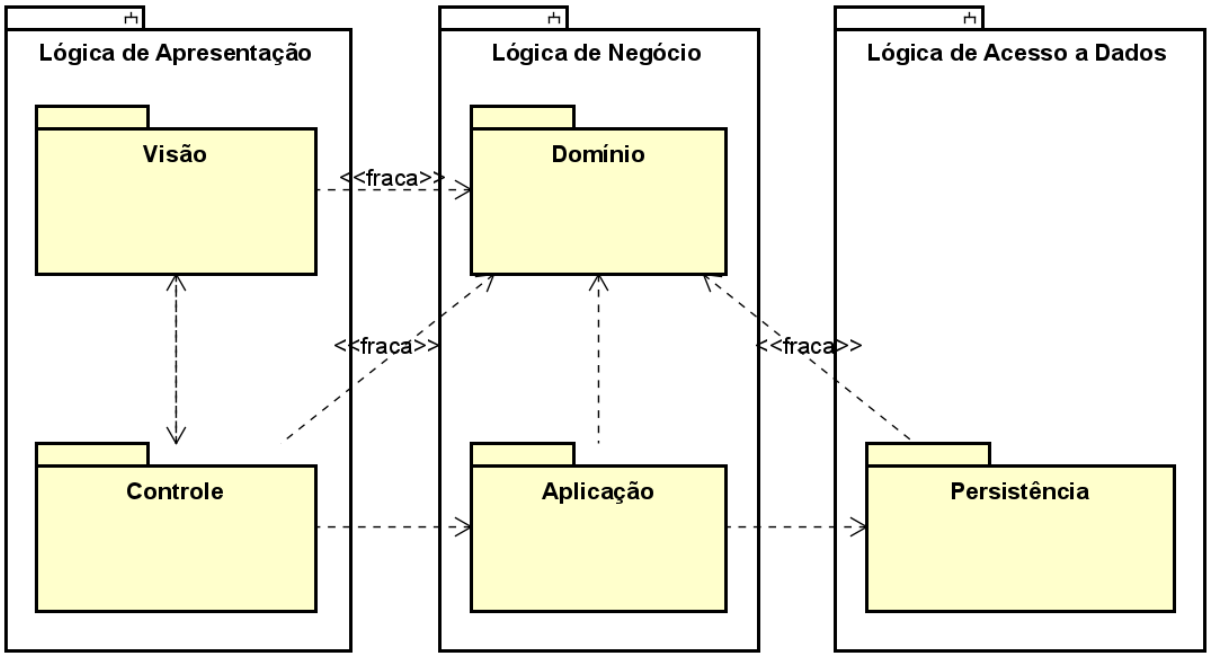

Figura 5 – Padrão arquitetônico *Service Layer* (SOUZA, 2007).

2.3.1 Arquitetura de Software do FrameWeb

A principal proposta do método FrameWeb concentra-se na fase de Projeto, tendo uma arquitetura lógica que divide o sistema em diferentes camadas, denominada padrão arquitetônico *Service Layer* (FOWLER, 2002b), apresentado na Figura 5.

A primeira camada é denominada **Lógica de Apresentação**. Nesta, teremos os pacotes Visão e Controle. O pacote Visão é composto por páginas *Web*, folhas de estilo, conteúdos multimídia e scripts, sendo disponibilizados do lado do cliente para interação com o usuário. O pacote Controle é composto por classes de ação, incorporadas por um *framework Front Controller* (ou Controlador Frontal) (ALUR; MALKS; CRUPI, 2003, p. 166), responsável por estabelecer a comunicação entre serviços e cliente, e isso se dá através de estímulos provocados pelo usuário, que refletem em alguma ação definida em uma classe de controle.

A segunda camada é denominada como **Lógica de Negócio**. Nesta, teremos os pacotes Domínio e Aplicação. O primeiro é composto por classes que representam o domínio do problema, sendo as mesmas geralmente definidas no diagrama de classes criado durante a fase de Especificação de Requisitos. O segundo é composto por classes de serviço, responsáveis por prover funcionalidades, geralmente definidas nos modelos de Casos de Uso criados durante a fase de Especificação de Requisitos.

O pacote Controle depende do pacote Aplicação para que as funcionalidades do sistema possam ser providas ao usuário, da mesma forma que o pacote Aplicação depende do pacote Domínio para manipular suas informações. Os pacotes Controle e Visão possuem uma relação de dependência com o pacote Domínio, todavia esteriotipada como **«fraca»**,

representando um baixo acomplamento entre eles. Isso se dá pelo fato desses pacotes não modificarem objetos das classes de domínio (entidade) de forma persistente, tendo sempre o pacote Aplicação como intermediador dessa interação.

A terceira e última camada é denominada como **Lógica de Acesso a Dados**. Nesta, há apenas o pacote Persistência, que é composto por classes DAO (*Data Access Object*) (ALUR; MALKS; CRUPI, 2003) que utilizam algum *framework* ORM (*Object/Relational Mapping*) responsável por prover uma camada de abstração para manipulação de objetos no banco de dados relacional.

O pacote Aplicação depende do pacote de Persistência para que os objetos de entidade possam ser gravados, recuperados, modificados e excluídos do banco de dados. Há também uma relação esteriotipada como **«fraca»** entre o pacote Persistência e Domínio, visto que o primeiro precisa conhecer as classes de entidade para persistir seus objetos.

2.3.2 Modelos FrameWeb

Além da arquitetura apresentada na Seção 2.3.1, devemos modelar os artefatos a serem implementados na fase seguinte (implementação), sendo assim, o método propõe quatro diferentes tipos de modelos, todos baseados no diagrama de classes da UML:

- O **Modelo de Entidades** representa o domínio do problema através de um diagrama de classes da UML. Por sua vez, cada classe contém configurações que dizem respeito ao seu mapeamento para o banco de dados. A partir deste modelo são implementadas as classes do pacote Entidade na fase de implementação, pertencente a camada de Lógica de Negócio (SOUZA, 2020);
- O **Modelo de Persistência** representa as classes DAO, se dando através de um diagrama de classes da UML. Cada DAO encarrega-se por persistir instâncias das classes de entidade no banco de dados. Estas classes fazem parte do pacote de Persistência, incluso na camada de Lógica de Acesso a Dados (SOUZA, 2020);
- O **Modelo de Navegação** representa os componentes da camada de Lógica de Apresentação, se dando através de um diagrama de classes UML, são elas: páginas Web e formulários HTML, pertencentes ao pacote Visão; além das classes de ação (ou controle), que compõem o pacote Controle (SOUZA, 2020);
- O **Modelo de Aplicação** é um diagrama de classes da UML responsável por representar as classes de serviço, que codificam os casos de uso, e suas dependências. Este diagrama guia a implementação das classes do pacote Aplicação e a configuração das dependências entre os pacotes Controle, Aplicação e Persistência, especificando quais classes de ação dependem de quais classes de serviço e quais DAOs atendem as classes de serviço (SOUZA, 2020);

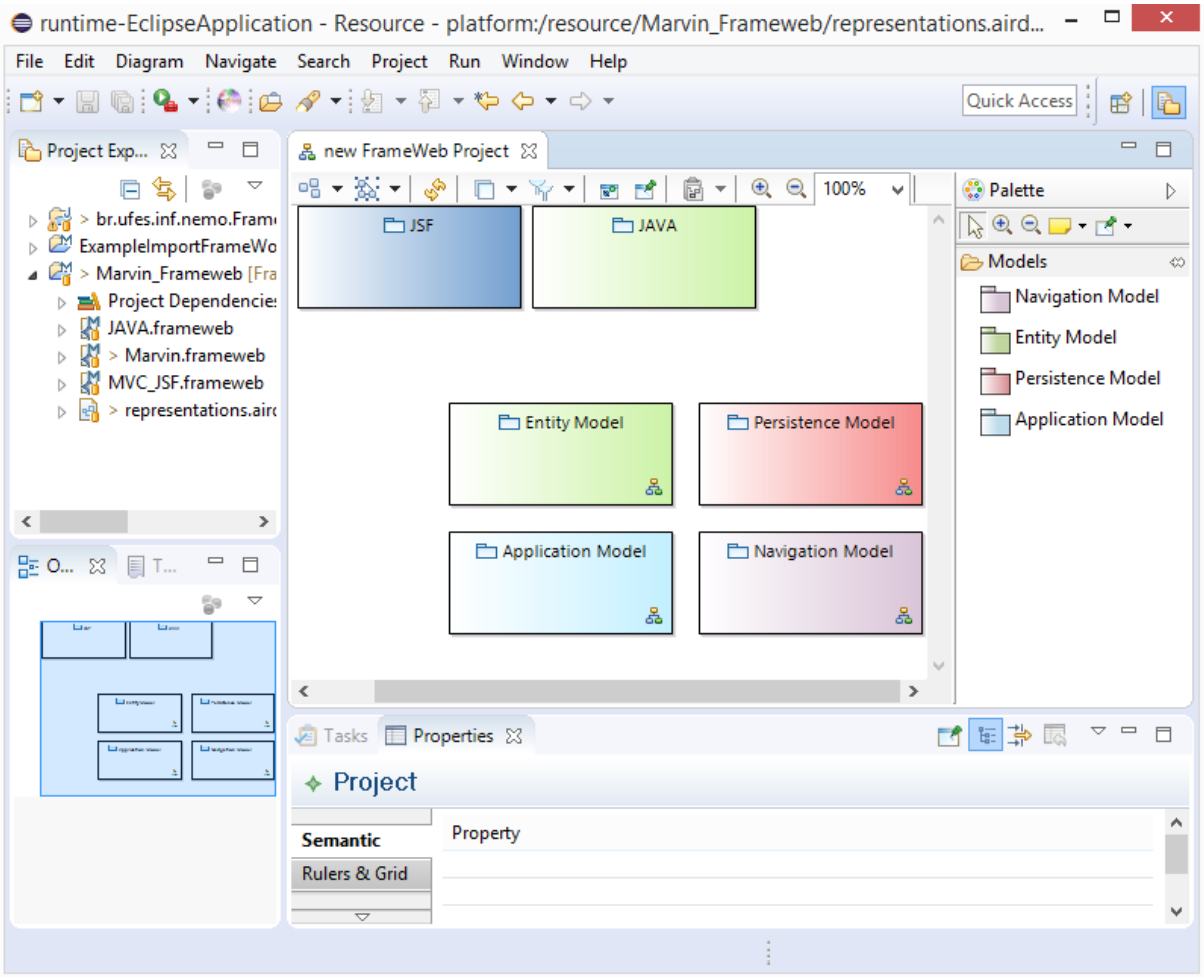

Figura 6 – Visão geral da primeira versão do FrameWeb Editor (CAMPOS; SOUZA, 2017).

2.3.3 Ferramentas FrameWeb

Em razão dos modelos FrameWeb serem baseados na UML, qualquer editor UML pode ser utilizado para construí-los. No entanto, para garantir que os modelos fossem elaborados utilizando apenas construtos válidos para o FrameWeb, surgiu a necessidade de desenvolver uma linguagem própria, especificada formalmente através de metamodelos (MARTINS; SOUZA, 2015), utilizando uma abordagem MDD (*Model-driven Development* ou Desenvolvimento Orientado a Modelos).

A partir do metamodelo do FrameWeb, foi construído o FrameWeb Editor (CAMPOS; SOUZA, 2017), ferramenta que oferece um ambiente gráfico baseado na plataforma Eclipse RCP (*Rich Client Platform*) a qual viabiliza a criação dos diagramas dos modelos FrameWeb. A Figura 6 nos dá uma visão geral do editor, tendo, à esquerda, o *Model Explorer*, utilizado para visualização de arquivos e diretórios do projeto. No centro estão as representações visuais dos modelos, *frameworks* e configuração do projeto. À direita há uma paleta de construtos para serem utilizados nos modelos e, porfim, na parte inferior são disponibilizadas as opções de configuração dos elementos que compõe os modelos.

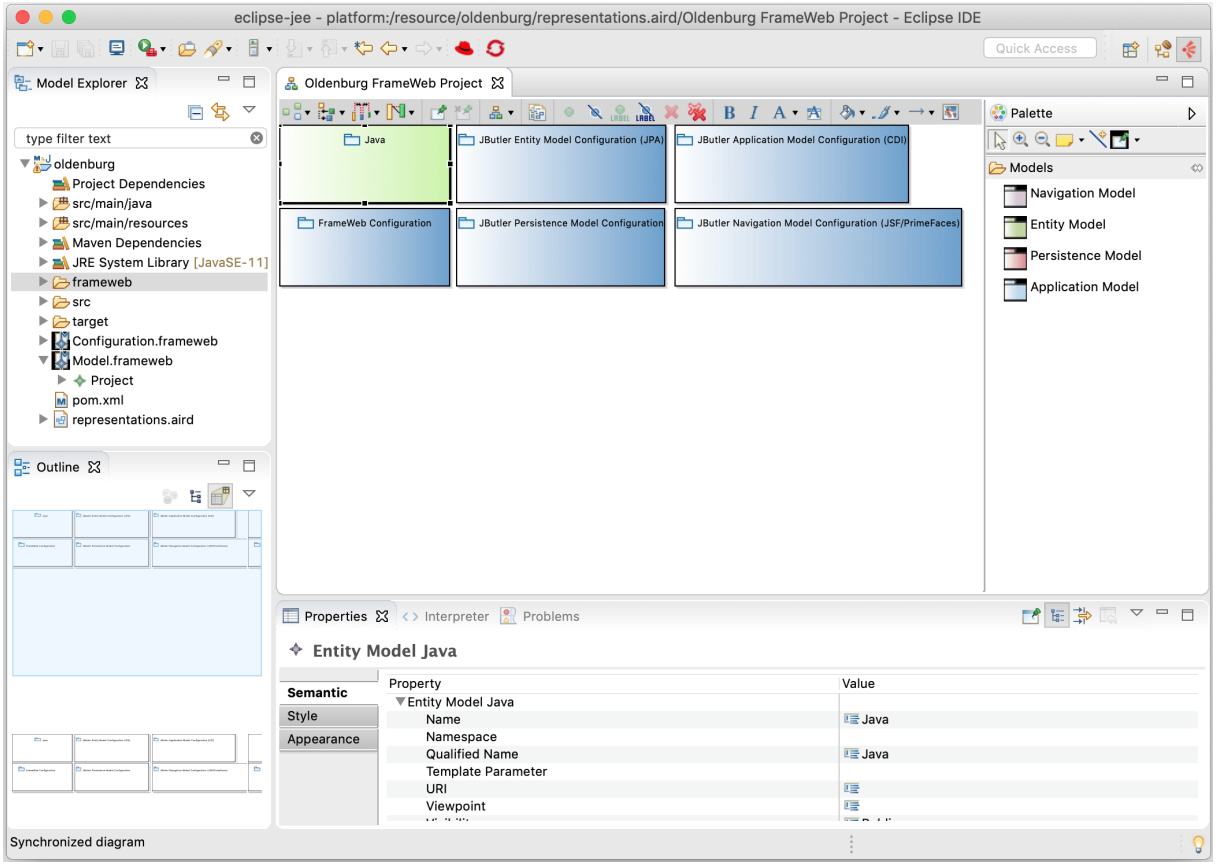

Figura 7 – Projeto FrameWeb com a faceta ativada (SOUZA, 2020).

Para desenvolver aplicações com o FrameWeb *Editor*, algumas ferramentas precisam ser instaladas. Primeiramente, é necessário ter instalado algum *runtime* Java a partir da versão 8; Em seguida, instalar o *Eclipse IDE for Java EE Developers*. Uma vez instalado o Eclipse, é preciso adicionar os *plugins*: Sirius, através do *Eclipse Marketplace*, e FrameWeb Tools, através da opção *Help*, seguida de *Install New Software*, aplicando o *endpoint* <http://dev.nemo.inf.ufes.br/framewebplugin/>. Mais detalhes do processo de preparação do ambiente FrameWeb estão disponíveis no ToolsTutorial01.[1]

Com os pré-requisitos concluídos, podemos criar um projeto Web (*Dynamic Web Project*) no Eclipse e adicionar a faceta *FrameWeb Tools*. Assim que a faceta for adicionada, será criado um modelo FrameWeb em branco, contendo apenas o componente FrameWeb Configuration. A Figura 7 apresenta uma visão geral do editor após a adição da faceta.

O elemento nomeado como "FrameWeb Configuration" está atrelado ao arquivo Configuration.frameweb, que nos permite adicionar algumas configurações globais no projeto, são elas: *Class Extension* e *Page Extension* — extensões dos arquivos de classes e páginas Web (ex: .cs e .html); *Framework Definition Path* — caminho para o diretório de templates; *Src Path* e *View Path* — caminhos para os diretórios onde classes e páginas Web serão geradas, respectivamente.

[1] <https://github.com/nemo-ufes/FrameWeb/wiki/ToolsTutorial01>

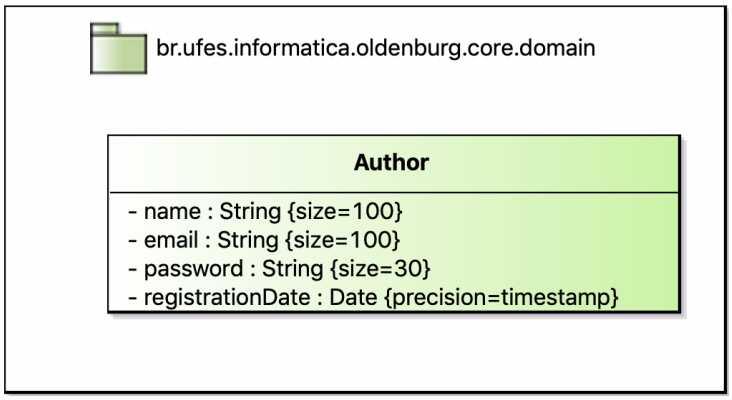

Figura 8 – Modelo de Entidades do Oldenburg (SOUZA, 2020).

Os demais elementos disponíveis no quadro principal do editor referem-se a definição de linguagem (*Language Definition Files*), que, na prática, inclui uma lista de tipos primitivos e classes de uma determinada linguagem (ex: Java possui tipos int, char, String, etc.); e *frameworks* (*Framework Definition File*), associados a templates, que, no caso do JSF (*framework* MVC), inclui também um conjunto de tags referentes aos componentes visuais da biblioteca de decoração (PrimeFaces tem inputText, commandButton, etc.), que serão utilizadas para geração de código do Modelo de Navegação. Estas definições estão disponíveis para *download* no repositório oficial da ferramenta[2] e foram importadas para o projeto.

Como exemplo, apresentaremos o Oldenburg, WIS para registro de autores de conferências, apresentado na *wiki* do FrameWeb Editor.[3] A Figura 8 apresenta o Modelo de Entidades do Oldenburg, contendo a classe **Author** e seu mapeamento objeto-relacional. Neste modelo estão definidos os atributos da entidade Author, cada qual com sua visibilidade, nome, tipo de dado e regras de mapeamento (ou *constraints*). Não há especificado um atributo ID para a classe **Author**, pois as entidades do Oldenburg herdam a classe abstrata **PersistentObjectSupport**, pertencente ao pacote de utilitários do JButler.[4]

A persistência de **Author**, apresentada no Modelo de Persistência na Figura 9, é tratada pela interface **AuthorDAO**, que é implementada pela classe **AuthorDAOJPA**. Este DAO define o método *retrieveByEmail*, retornando um autor (Author) a partir do email do mesmo. No Oldenburg, todas as interfaces DAO herdam **BaseDAO** e todas as classes DAO herdam **BaseJPADAO**, os quais fornecem métodos básicos de CRUD. Ambos encontram-se disponíveis no pacote de utilitários do JButler.

As telas e classe de controle que dão acesso às funcionalidade de cadastro de autores são representadas pelo Modelo de Navegação, apresentado na Figura 10. Presente no diretório "*~/core/registration*", os componentes *Page*, assinados com o esteriótipo

[2] <https://github.com/nemo-ufes/FrameWeb>
[3] <https://github.com/nemo-ufes/FrameWeb/wiki/ToolsTutorial02>
[4] <https://github.com/dwws-ufes/jbutler>

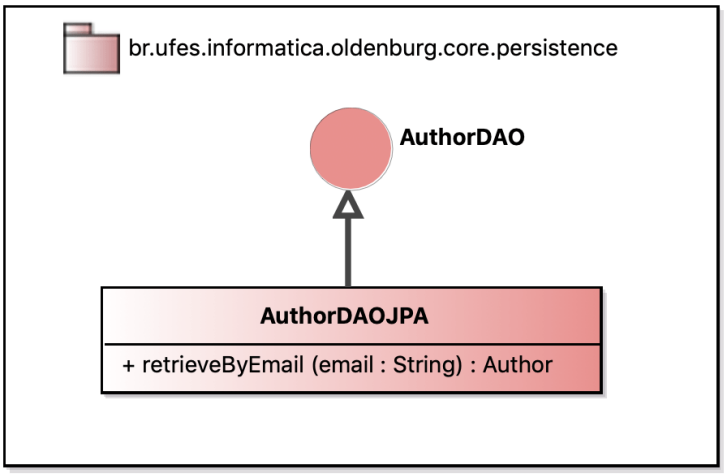

Figura 9 – Modelo de Persistência do Oldenburg (SOUZA, 2020).

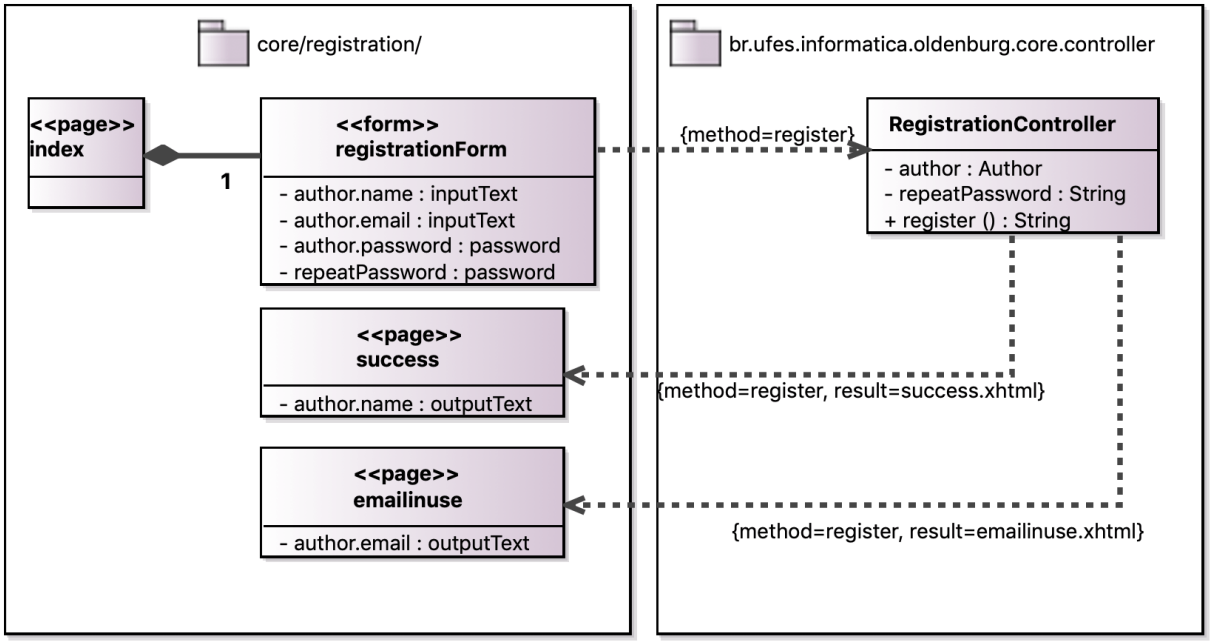

Figura 10 – Modelo de Navegação do Oldenburg (SOUZA, 2020).

«page», representam páginas Web, sendo ***index*** a principal página da funcionalidade de cadastro de autor. A página ***index*** é composta pelo formulário *registrationForm*, assinado com o esteriótipo **«form»**, o qual contém campos de texto (*inputText*) para nome e email do autor, assim como campos para preenchimento e confirmação de senha (*password*). Este formulário se encontra associado à classe de controle **RegistrationController** e, assim que submetido, o Controlador Frontal irá copiar os valores dos campos recebidos pelo corpo da requisição para o atributo *author* (note que os campos do formulário submetido possuem sufixo de mesmo nome do atributo de destino), presente na própria *controller*, assim executando o método *register*. A depender do resultado, o usuário será redirecionado para as páginas ***success*** ou ***emailinuse***, devolvendo algumas informações úteis para montar a página de destino (*author.name* ou *author.email*).

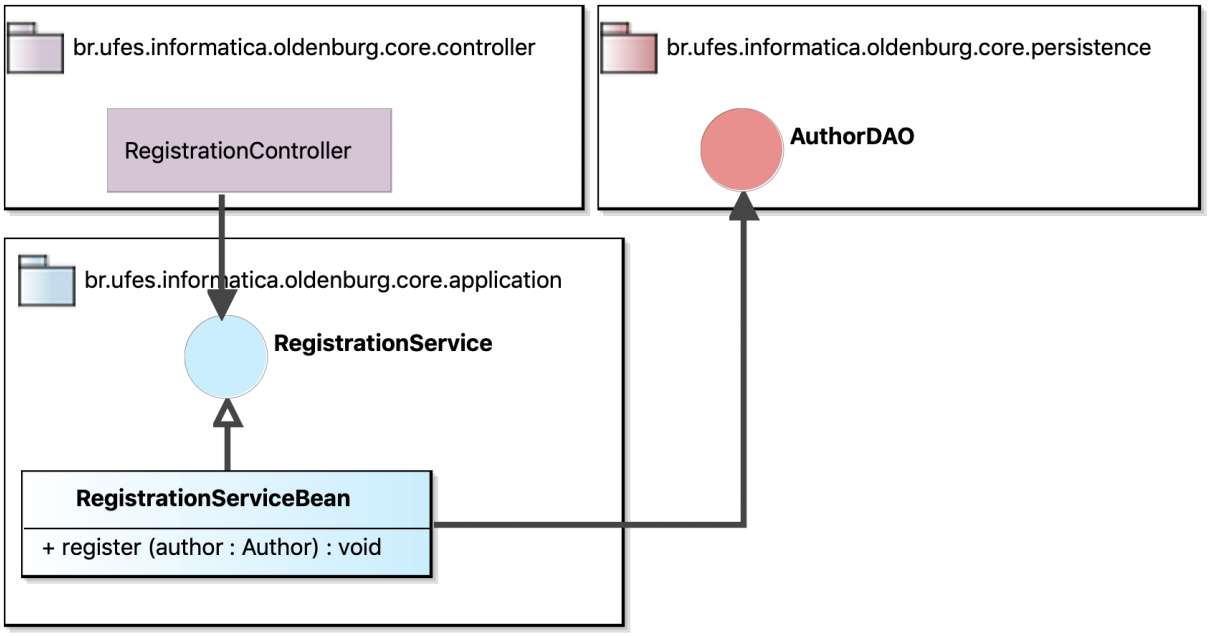

Figura 11 – Modelo de Aplicação do Oldenburg (SOUZA, 2020).

Para representar as interações entre as diferentes partes do WIS, utiliza-se o Modelo de Aplicação, apresentado na Figura 11. Assim como as interfaces e classes DAO mostradas anteriomente, os serviços também são divididos entre interface e implementação, a dizer, **RegistrationService** e **RegistrationServiceBean**, respectivamente. A controladora **RegistrationController** se encontra associada à interface **RegistrationService**, indicando uma relação de dependência que implicará na injeção do serviço solicitado na classe de controle. Este mesmo comportamento acontece a partir da relação de dependência entre **RegistrationServiceBean** e **AuthorDAO**, onde o DAO é injetado no serviço.

De modo geral, uma vez associadas as diferentes partes do WIS, assim que usuário submeter o formulário de cadastro de autor na camada de visão, este será capturado pela *controller* **RegistrationController**, que irá invocar o serviço **RegistrationService** e executar o método *register*. Este método, por sua vez, utiliza **AuthorDAO** para performar as operações necessárias no banco de dados e prover uma resposta para o usuário.

Criados os modelos FrameWeb, pode-se então gerar o código do WIS clicando com o botão direito em algum espaço em branco do quadro principal do editor, então selecionando a opção "Generate Source Code". Os arquivos referentes às interfaces, classes e páginas Web serão criados em seus respectivos diretórios dentro do projeto Eclipse.

A Listagem 2.1 (extraída do repositório oficial do FrameWeb[5]) apresenta o template utilizado para a geração de código das classes Front Controller do JButler para o Oldenburg. A notação **{{ var }}** demarca o local a qual será inserido o valor de alguma variável ou atributo extraídos do Modelo de Navegação, enquanto **{% if | for | endif | ... %}**

[5] <https://github.com/nemo-ufes/FrameWeb/blob/master/frameworks/jbutler/templates>

indica o início ou término de *statements* de fluxo de controle. No repositório oficial do editor há uma tabela que aponta quais variáveis estão disponíveis para utilização em cada modelo/*template*.[6] Todas as *tags*, filtros e funções do Twig (versão 1.x) encontram-se disponíveis na documentação oficial da linguagem.[7]

[6] <https://github.com/nemo-ufes/FrameWeb/tree/master/frameworks/utils>
[7] <https://twig.symfony.com/doc/1.x/>

Listagem 2.1 – Template para classes do tipo Front Controller

```
1  package {{ package.Name }};
2
3  import javax.ejb.EJB;
4  import javax.enterprise.inject.*;
5  import br.ufes.inf.nemo.jbutler.ejb.controller.JSFController;
6
7  /** TODO: generated by FrameWeb. Should be documented. */
8  @Model
9  public class {{ class.Name }} extends JSFController {
10   /** Serialization id (using default value, change if necessary). */
11   private static final long serialVersionUID = 1L;
12
13   {% for association in associations %}
14   /** TODO: generated by FrameWeb. Should be documented. */
15   @EJB
16   private {{ association.TargetMember.Type.Name }} {{ association.TargetMember.
         Type.Name | lower_first }};
17   {% endfor %}
18
19   {% for attribute in attributes %}
20   /** TODO: generated by FrameWeb. Should be documented. */
21   private {{ attribute.Type.Name }} {{ attribute.Name }};
22   {% endfor %}
23
24   {% for method in methods %}
25   /** TODO: generated by FrameWeb. Should be documented. */
26   {{ method.Visibility.Name }} {% if method.MethodType is null %}void{% else %}{{
         method.MethodType.Name }}{% endif %} {{ method.Name }}({% for parameter in
        method.OwnedParameters %}{{ parameter.Type.Name }} {{ parameter.Name }}{% if
         loop.last == false %}, {% endif %}{% endfor %}) {
27     // FIXME: auto-generated method stub
28     return{% if method.MethodType is not null %} null{% endif %};
29   }
30   {% endfor %}
31
32   {% for attribute in attributes %}
33   /** Getter for {{ attribute.Name }}. */
34   public {{ attribute.Type.Name }} get{{ attribute.Name | capitalize }}() {
35     return {{ attribute.Name }};
36   }
37
38   /** Setter for {{ attribute.Name }}. */
39   public void set{{ attribute.Name | capitalize }}({{ attribute.Type.Name }} {{
         attribute.Name }}) {
40     this.{{ attribute.Name }} = {{ attribute.Name }};
41   }
42   {% endfor %}
43 }
```

3 Definição e Especificação de Requisitos

Este capítulo contém informações sucintas que dizem respeito à documentação dos requisitos do SCAP, elaborada por Duarte (2014) e Prado (2015), mas revisada e aprimorada neste trabalho para atender às necessidades do sistema e melhor orientar seu processo de implementação. A documentação completa pode ser encontrada nos apêndices deste trabalho.

3.1 Minimundo

No Departamento de Informática (DI) da UFES, professores podem solicitar pedidos de afastamento referentes a eventos no Brasil ou no exterior. Mediante avaliações, estes podem ou não ser aprovados.

Os secretários são responsáveis pela parte administrativa do sistema, cabendo a eles as funções de cadastrar usuários e mandatos dos chefes de departamento. Também são responsáveis pelo cadastramento dos pareceres de fora do DI e também pelo arquivamento de pedidos já finalizados.

Ao cadastrar um usuário, os secretários devem informar: tipo de usuário, nome, matrícula, e-mail, telefone e parentescos. Já para cadastrar um chefe de departamento, um mandato temporário deve ser criado e atribuído a um professor já cadastrado no sistema. De um mandato devem ser informados: tipo de mandato, professor, data de início e data de fim.

Professores podem submeter pedidos de afastamento. No ato da submissão devem ser informados: data de início do afastamento, data de fim do afastamento, tipo de afastamento, motivo do afastamento, tipo de ônus, data de início do evento, data de fim do evento, nome do evento e os documentos necessários. Para um evento no Brasil, o pedido deve ser avaliado pelos membros do departamento por meio da isenção de manifestações contrárias ao pedido de afastamento, isto é, caso nenhum professor se oponha, este será aceito. Havendo alguma oposição, será feita uma reunião entre os professores para decidir se o pedido deve ser aceito ou não, cabendo aos secretários atualizarem o status do pedido de acordo com a decisão alcançada durante a reunião.

Para um evento no exterior, o pedido deve primeiro passar pelo chefe de departamento do DI, que irá selecionar um professor do DI para ser o relator responsável pelo pedido de afastamento, desde que o escolhido não seja o próprio solicitante, nem um parente daquele que fez o pedido e nem o próprio chefe de departamento. Uma vez que o relator tenha submetido um parecer favorável e nenhum membro do departamento se manifeste

contra o parecer, os secretários encaminharão o pedido para o Centro Tecnológico (CT) e posteriormente para a Pró-reitoria de Pesquisa e Pós-Graduação (PRPPG), registrando seus pareceres. Se totalmente aprovado, o afastamento será publicado no Diário Oficial da União. Caso haja rejeição por qualquer uma das entidades responsáveis pela avaliação do pedido, este será rejeitado.

Para todos os processos que ocorrem durante o ciclo de vida de um pedido de afastamento, os envolvidos do DI serão notificados através de e-mails automáticos enviados pelo sistema. Não cabe ao SCAP se responsabilizar pelas tramitações do CT e da PRPPG, ou seja, o sistema não se responsabiliza pelos processos que ocorrem fora do DI, sendo assim, os pareceres externos devem ser registrados pelos próprios secretários.

3.2 Casos de Uso

Com os requisitos identificados e o escopo (minimundo) definido, seguimos com a identificação dos atores do sistema, apresentados na Tabela 1, assim como a elaboração dos diagramas de Casos de Uso, que foram divididos em diferentes subsistemas e são apresentados nas subseções que se seguem.

Tabela 1 – Atores do SCAP.

Ator	**Descrição**
Usuário	Usuário credenciado no SCAP.
Professor	Professor efetivo do DI/UFES.
Chefe do Departamento	Professor do DI/UFES que está realizando a função administrativa de chefe/subchefe do departamento.
Secretário	Secretário do DI/UFES.
Scheduler	Processo responsável por disparar tarefas agendadas.

O ator **Usuário** refere-se a todo e qualquer usuário do sistema em sua forma mais genérica. Um **Professor** é aquele com permissão para solicitar afastamentos e emitir pareceres. O **Chefe de Departamento** é também um **Professor**, porém este também se encarrega por encaminhar pedidos de afastamento para relatores. O **Secretário** é responsável pela parte administrativa do sistema, ficando encarregado por gerenciar informações de usuários e mandatos, mas também orquestrar todo o processo de um pedido de afastamento. O **Scheduler** é identificado como um ator, porém este nada mais é do que um processo (de software) executando em *background* que tem como função aprovar, arquivar ou cancelar pedidos de afastamento, a depender de sua situação.

3.2.1 Subsistema Gerenciamento de Usuário

Este subsistema tem como função operar no gerenciamento de informações e autenticação de usuários do sistema, mas também no gerenciamento de mandatos de chefes

e subchefes do Departamento de Informática. A Figura 12 apresenta o diagrama de Casos de Uso do Subsistema Gerenciamento de Usuário.

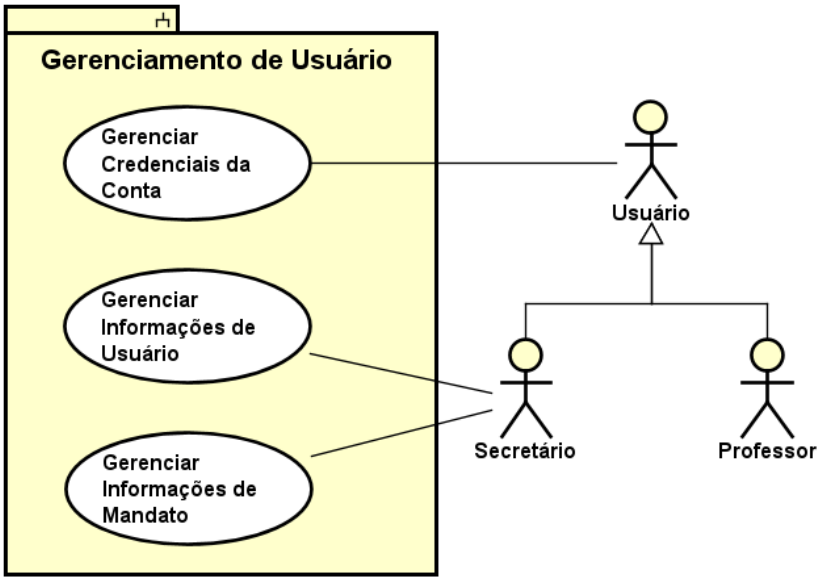

Figura 12 – Diagrama de Casos de Uso do Subsistema Gerenciamento de Usuário.

O caso de uso **Gerenciar Credenciais da Conta** permite que o usuário configure os dados de sua própria conta, tais como mudança de nome, senha, email, etc. Já **Gerenciar Informações de Usuário** permite que o secretário edite informações de um usuário. Um secretário tem poder para inserir novos usuários no sistema, assim como editar suas informações não credenciais e parentescos.

Com permissão de uso exclusiva para os secretários, o caso de uso **Gerenciar Informações de Mandato** permite que secretários criem mandatos de chefe ou subchefe de departamento e os atribua a determinados professores, com limite máximo de um professor por tipo de mandato, dentro de um intervalo estipulado de tempo. Secretários também podem excluir mandatos agendados ou finalizar mandatos em andamento, em virtude de desistência ou qualquer outro fator que venha a invalidar a posição de chefe ou subchefe do departamento.

3.2.2 Subsistema Gerenciamento de Afastamento

Como maior e principal módulo do SCAP, este subsistema é responsável por todo o ciclo de vida dos pedidos de afastamento. A Figura 13 apresenta o diagrama de casos de uso do subsistema Gerenciamento de Afastamento.

O caso de uso **Solicitar Afastamento** permite que um professor faça um pedido de afastamento no sistema, enquanto o caso de uso **Cancelar Afastamento** dá ao professor solicitante o direito de desistir de um pedido em andamento a qualquer momento.

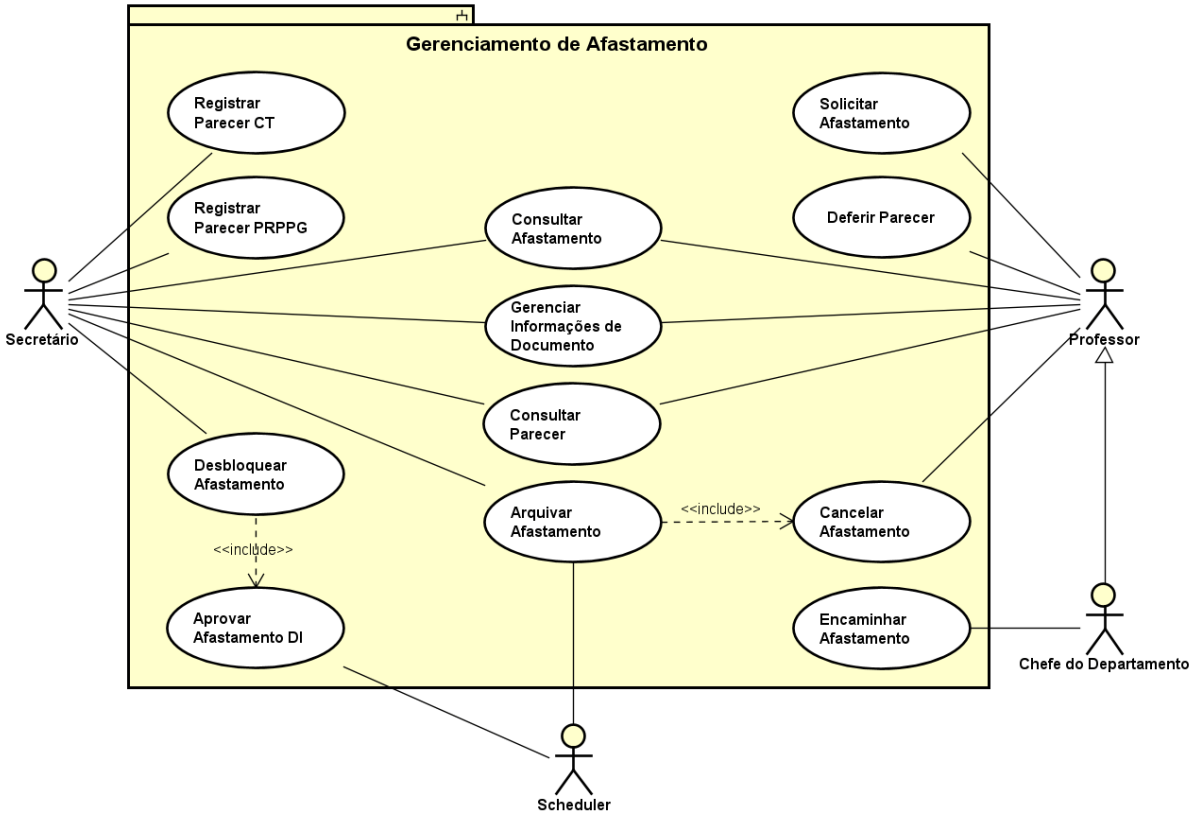

Figura 13 – Diagrama de Casos de Uso do Subsistema Gerenciamento de Afastamento.

O caso de uso **Deferir Parecer** permite que professores se manifestem contra ou a favor de um pedido de afastamento. No cenário onde um pedido atende a um evento internacional, primeiro é necessário que o relator, escolhido pelo chefe de departamento no caso de uso **Encaminhar Afastamento**, tenha emitido seu parecer. Posteriormente, todos os professores podem se manifestar. Quando ao menos um parecer desfavorável é deferido, o pedido é bloqueado e sua aprovação fica a critério da decisão tomada durante a reunião do DI, cabendo aos secretários executar a ação de desbloquear o pedido de afastamento (caso de uso **Desbloquear Afastamento**). Se a decisão da reunião for desfavorável ao pedido, este é reprovado, mas caso todos estejam de acordo, é incluído o caso de uso **Aprovar Afastamento DI**, responsável por aprovar a solicitação de afastamento e disponibilizar a ata de aprovação do pedido pelo Departamento de Informática.

O caso de uso **Consultar Afastamento** dá, aos professores e secretários, o direito de consultarem todos os pedidos de afastamento cadastrados no sistema. Enquanto **Gerenciar Informações de Documento** permite que secretários e exclusivamente o professor solicitante possam adicionar ou remover documentos anexados ao pedido de afastamento, mas todos têm o direito de consultá-los. O caso de uso **Consultar Parecer** permite que secretários e professores visualizem pareceres já deferidos contra ou a favor de um pedido de afastamento.

Os casos de uso **Registrar Parecer CT** e **Registrar Parecer PRPPG** têm como função fazer com que secretários possam emitir os pareceres do CT e da PRPPG, respectivamente, para pedidos de afastamento internacionais e já aprovados pelo DI. Caso alguma dessas entidades tenham emitido um parecer negativo, o pedido é reprovado.

O caso de uso **Aprovar Afastamento DI**, quando executado pelo scheduler, faz com que, 10 dias após a solicitação de afastamento ter sido feita (com os documentos necessários já submetidos e ao menos o parecer do relator emitido, caso necessário), o pedido possa ser automaticamente aprovado. Não havendo pareceres contrários e estando em situação "Liberado", o pedido é aprovado, caso contrário, nada é feito.

O caso de uso **Arquivar Afastamento** permite que secretários alterem a situação de um pedido de afastamento aprovado para arquivado, porém, esta ação só será necessária caso o secretário se antecipe em arquivar o pedido, pois assim que um afastamento alcança a data de fim, o scheduler se encarrega de arquivá-lo no dia seguinte, no mesmo horário que a solicitação foi feita. Num cenário onde o scheduler tente arquivar um afastamento, mas este por acaso se encontra numa situação inválida, ele será imediatamente cancelado, portanto aqui é incluído o caso de uso **Cancelar Afastamento**.

Durante o ciclo de vida de uma solicitação de afastamento, todos os envolvidos são notificados a respeito das mudanças de estado de um pedido de afastamento, via email.

3.3 Diagrama de Classes

O diagrama de classes tem como objetivo prover ao observador (cliente ou desenvolvedor) uma visão estática da estrutura do sistema. Isso se dá através da modelagem de um conjunto de classes de domínio e seus relacionamentos (FALBO, 2017). A Figura 14 apresenta o diagrama de classes da fase de análise de requisitos do SCAP.

A classe **Pessoa** representa uma pessoa e um usuário genérico do sistema. Esta possui informações pessoais como nome, sobrenome, email, telefone e matrícula, assim como o atributo "usuário ativo?", que indica se um usuário pode ter acesso ao sistema. Este atributo, quando falso, tem como efeito impedir o acesso de um usuário que não atue mais no departamento ou que esteja incapacitado de exercer sua função.

As classes **Secretario** e **Professor** herdam de **Pessoa**, portanto compartilham algumas informações em comum, sendo que um professor pode estar associado a vários **Mandato**s, garantindo-lhe a posição de chefe ou subchefe do departamento, a depender do tipo de mandato. Para cada mandato, há um registro sobre a data de início e fim do mesmo, assim como um atributo "foi interrompido?" que serve para determinar se um mandato foi finalizado precocemente. Um professor pode ainda estar associado a **Parentesco**s, que por sua vez está associado a outro **Professor**.

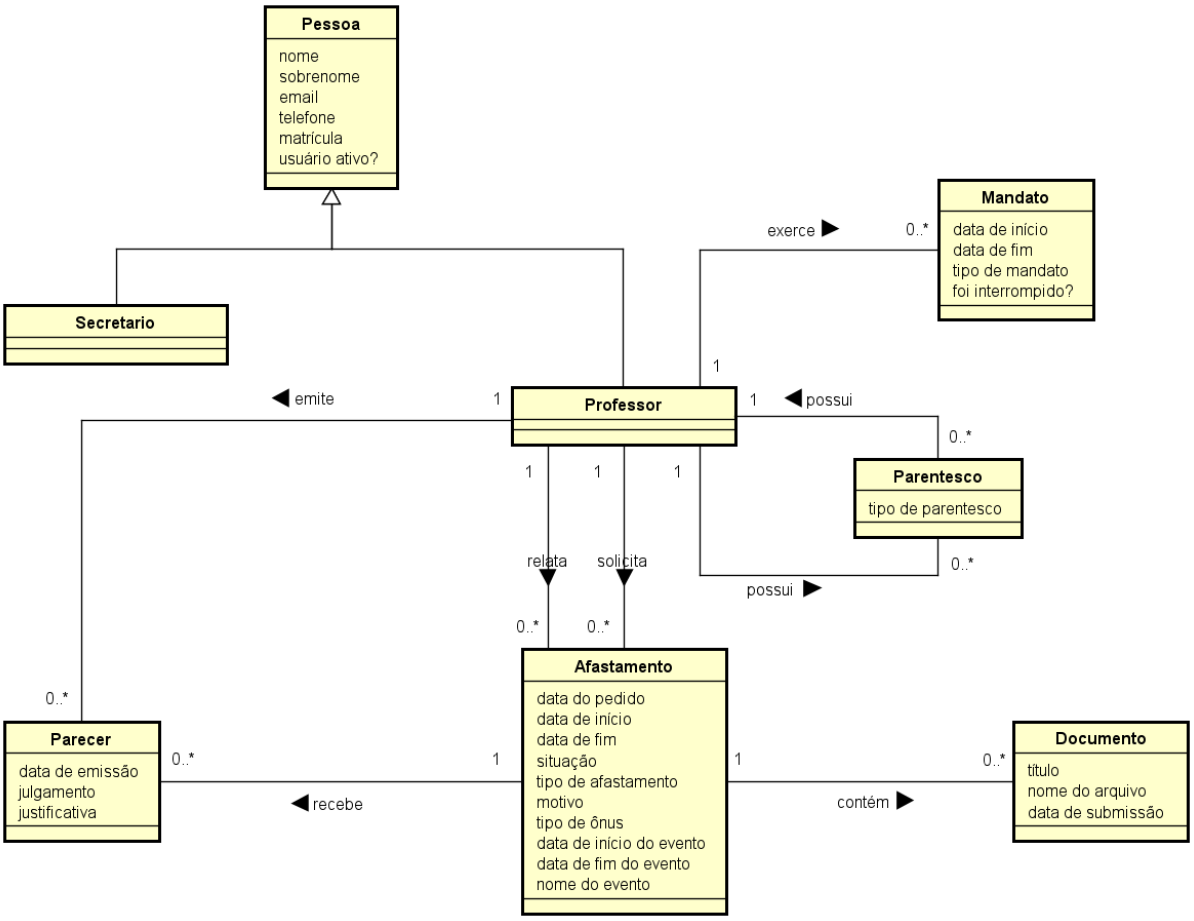

Figura 14 – Diagrama de Classes (fase de análise de requisitos).

Um **Professor** pode solicitar **Afastamento**s, assim como pode ser relator de pedidos solicitados por outros professores, uma vez escolhido pelo chefe de departamento. A classe **Afastamento** carrega informações que dizem respeito a condição de um pedido de afastamento durante seu trâmite de aprovação. Esta por sua vez está associada a **Documento**s, os quais mantêm arquivos submetidos pelas partes envolvidas. Já a classe **Parecer** comporta as informações de um parecer submetido por um **Professor**, tendo como alvo um determinado **Afastamento**.

3.4 Restrições de Integridade

As Restrições de Integridade são regras de negócio, geralmente escritas em linguagem natural, que têm como objetivo garantir a integridade do modelo conceitual, visto que em determinadas situações os recursos providos pela UML são insuficientes para capturar tais restrições ou são de difícil compreenssão para clientes e usuários (FALBO, 2017).

Abaixo estão listadas todas as restrições de integridade que foram identificadas durante o processo de desenvolvimento do SCAP:

- O solicitante do pedido de afastamento não pode ser relator do próprio pedido;
- O solicitante do pedido de afastamento não pode deferir pareceres ao próprio pedido;
- O relator não pode ser parente do solicitante do pedido de afastamento;
- A data de fim do afastamento deve ser maior ou igual a data de início do mesmo;
- A data de fim do evento deve ser maior ou igual a data de início do mesmo;
- Os intervalos das datas de início e fim de um afastamento e do seu evento alvo devem se intersectar;
- Deve haver no máximo um chefe e um subchefe de departamento simultaneamente;
- Secretários não devem solicitar afastamentos.

4 Projeto Arquitetural e Implementação

As seções que se seguem apresentam o processo de planejamento e desenvolvimeto do SCAP. Na Seção 4.1 é apresentada a arquitetura utilizada no sistema, assim como os *frameworks* utilizados. A Seção 4.2 aponta as bibliotecas utilizadas para dar suporte à implementação do sistema. Na Seção 4.3 são apresentados os modelos FrameWeb desenvolvidos dentro do FrameWeb Editor e, sucessivamente, na Seção 4.4 são apresentados os códigos gerados pelo gerador de código do FrameWeb e toda implementação aplicada sobre eles.

4.1 Arquitetura do Sistema

A arquitetura do SCAP, apresentada na Figura 15, segue a proposta do método FrameWeb, sendo definida pelo padrão arquitetônico *Service Layer*, que é dividido em três camadas: Lógica de Apresentação (*Presentation Tier*), Lógica de Negócio (*Business Tier*), Lógica de Acesso a Dados (*Data Access Tier*).

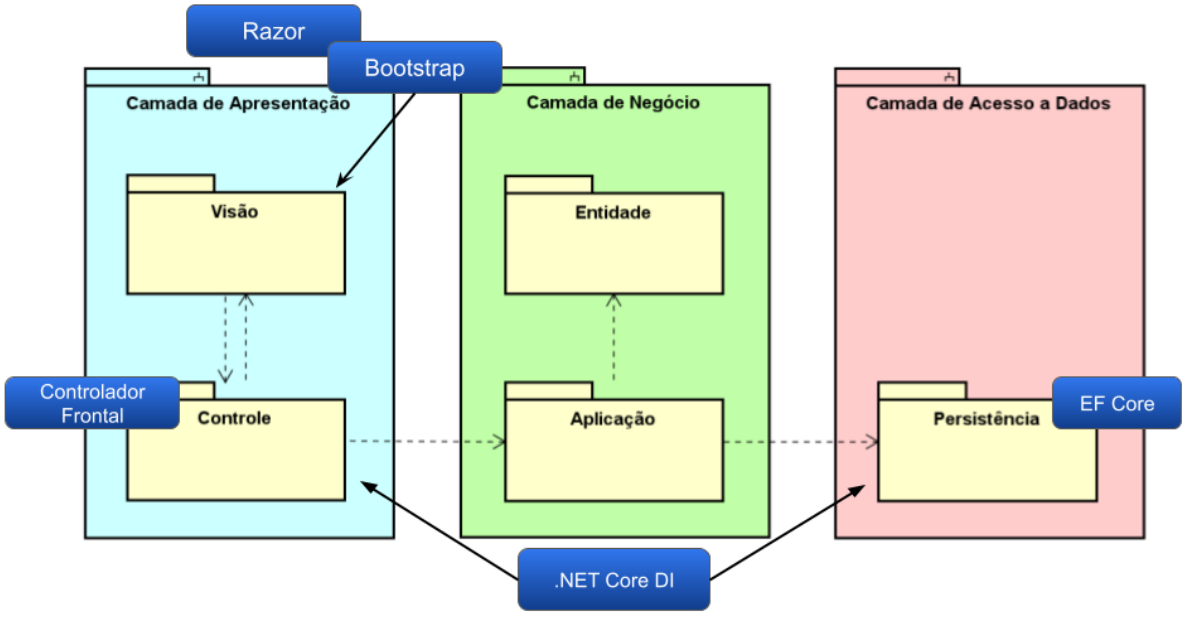

Figura 15 – Arquitetura de Sistema do SCAP.

A Camada de Apresentação contém os pacotes Visão e Controle, sendo o primeiro encarregado por comportar as páginas Web, criadas utilizando o *template engine* Razor e decoradas pelo *framework front-end* Bootstrap. Já o segundo engloba as classes que utilizam o *framework* Controlador Frontal ASP.NET Core MVC. Ambos pacotes comunicam-se

entre si, permitindo a interação cliente-servidor. Há ainda o *framework* de Segurança Identity, aplicado sobre o pacote Controle, permitindo que os diferentes usuários do sistema tenham acesso somente às funcionalidades que lhes dizem respeito. Esta camada é melhor apresentada na Seção 4.4, por meio de exemplos práticos com o código gerado.

A Camada de Negócio possui os pacotes Entidade e Aplicação, sendo o primeiro responsável por acomodar as classes de Entidade, que são mapeadas para o banco de dados pelo *framework* de ORM Entity Framework Core. Este mapeamento se dá através de configurações feitas com *data annotations* (ou atributos, como são chamadas em C#), permitindo que o ORM entenda como os objetos das classes de Entidade devem ser transformados para tabelas no banco de dados. Enquanto que o pacote de Aplicação comporta as classes de Serviço, responsáveis pela codificação dos Casos de Uso do sistema.

A Camada de Acesso a Dados possui o pacote Persistência, o qual contém classes Repository. Estas classes são responsáveis por fornecerem métodos para interação com o banco de dados, se dando de forma transparente, dispensando a necessidade de escrever *queries* em SQL e fazer o mapeamento para os objetos equivalentes de forma manual. As classes Repository utilizam um contexto de banco de dados (**DbContext**) provido pelo Entity Framework Core, permitindo-as acessar o banco de dados utilizando (nativamente) o padrão DAO, abordado na Seção 2.3.1.

O *framework* de injeção de dependências é padrão da plataforma .NET Core, assumindo a responsabilidade de configurar dependências entre as classes das diferentes camadas do sistema, possibilitando a comunicação entre todas as partes. O pacote Controle se comunica com o pacote Aplicação para consumir os serviços do sistema, enquanto que o pacote Persistência é consumido pelo pacote Aplicação, provendo o acesso às informações contidas no banco de dados.

A plataforma .NET Core trabalha com o *design pattern* de injeção de dependências nativamente, tendo as dependências de uma determinada classe satisfeitas através da passagem de parâmetros via construtor, configuradas através da interface **IServiceCollection**, disponibilizada pelo método *ConfigureServices* na classe **Startup**, onde todas as configurações do sistema são definidas quando iniciado. Aqui são configuradas as classes que vão atender determinadas interfaces, assim como seus ciclos de vida, são eles:

- ***Transient*** — Uma instância de classe é gerada toda vez que demandada;
- ***Scoped*** — Uma instância de classe é gerada por escopo, sendo compartilhada por todos que a demandam durante a requisição;
- ***Singleton*** — Uma única instância de classe é gerada e compartilhada por todos que a demandam na aplicação.

4.2 Tecnologias e Bibliotecas Utilizadas

Nas Subseções que se seguem são apresentadas as tecnologias e bibliotecas de apoio utilizadas durante todo o processo de desenvolvimento do SCAP.

4.2.1 MySQL

Para armazenamento de informações do SCAP, foi utilizado o MySQL 5.7, Sistema Gerenciador de Banco de Dados Relacional *open-source* e gratuito. O MySQL possui uma integração completa e bem consolidada com o EF Core, sendo uma solução implementada e disponibilizada pela Pomelo Foundation.[1]

4.2.2 Docker

Docker[2] é um serviço para virtualização de sistemas operacionais através do uso de *containers*. Um *container* é uma unidade padrão de *software* responsável por empacotar o código de uma aplicação e suas dependências, permitindo que esta seja distribuída e executada em diferentes ambientes de computação. Uma imagem de *container* é um pacote *standalone* que torna-se um *container* quando executado pela *Docker Engine*.

No SCAP foram utilizados dois diferentes *containers* para execução do WIS e banco de dados, conectados entre si por meio de configurações feitas via *docker-compose*, ferramenta mais básica para orquestração de *containers*.

4.2.3 AutoMapper

AutoMapper[3] é uma biblioteca para mapear um objeto para outro (*object-object mapping*). A abordagem *object-object mapping* consiste em transformar um objeto em um objeto de outro tipo, requerendo pouca ou nenhuma configuração adicional, uma vez que este trabalha através de convenções de nomes.

Neste trabalho, o AutoMapper foi utilizado nas *controllers* para transformar instâncias de classes de entidade em objetos de classes *ViewModel* e vice-versa, que funcionam como DTOs (*Data Transfer Object* ou Objeto de Transferência de Dados). Um DTO tem como objetivo agrupar um conjunto de informações numa classe simples, de maneira a otimizar a comunicação entre diferentes partes de um sistema (FOWLER, 2002a), enquanto que as *ViewModels* têm o mesmo propósito, porém são utilizadas para comunicação cliente-servidor e auxiliam na construção das páginas Web.

[1] <https://github.com/PomeloFoundation/Pomelo.EntityFrameworkCore.MySql>
[2] <https://www.docker.com/>
[3] <https://automapper.org/>

4.2.4 Fluent Validation

Para validações na camada de lógica de negócio, especificamente nos serviços, foi utilizada a biblioteca Fluent Validation.[4] Esta biblioteca tem como função criar regras de validação fortemente tipadas para as entidades do sistema, podendo emitir mensagens customizadas para diferentes erros.

4.2.5 Hangfire

Hangfire[5] é uma biblioteca para execução de processos (de *software*) em *background* em aplicações .NET, sendo independente de sistema operacional. Neste trabalho, o Hangfire foi utilizado para realizar as ações automatizadas do *scheduler*, ator citado na Seção 3.2, contando com um painel para acompanhamento do ciclo de vida dos processos em execução, agendados ou arquivados.

4.2.6 KissLog

KissLog[6] é uma biblioteca para *logging* e monitoramente de tráfego de aplicações .NET. Neste trabalho o KissLog foi utilizado para registrar exceções que venham a ocorrer na camada de Serviço. Suas informações são disponibilizadas online através de uma interface interativa e amigável.

4.2.7 Rotativa

O Rotativa[7] é uma biblioteca para geração de documentos em PDF nas plataformas .NET como se fossem páginas Web, sendo renderizados dinamicamente e disponibilizados com maior facilidade. Esta, por sua vez, acaba sendo dependente de sistema operacional, mas para o SCAP foram disponibilizados binários para Windows e Linux, permitindo que o *deploy* do WIS seja feito num destes dois sistemas operacionais.

4.3 Modelos FrameWeb

Nesta Seção são apresentados os modelos FrameWeb do SCAP, desenvolvidos através do FrameWeb Editor e utilizados para fins de documentação e geração de código.

[4] <https://fluentvalidation.net/>
[5] <https://www.hangfire.io/>
[6] <http://kisslog.net/>
[7] <https://github.com/webgio/Rotativa.AspNetCore>

4.3.1 Modelo de Entidades

A Figura 16 apresenta o Modelo de Entidades da fase de projeto do SCAP, feito no FrameWeb Editor e baseado no modelo de entidades da fase de análise de requisitos, apresentado na Figura 14.

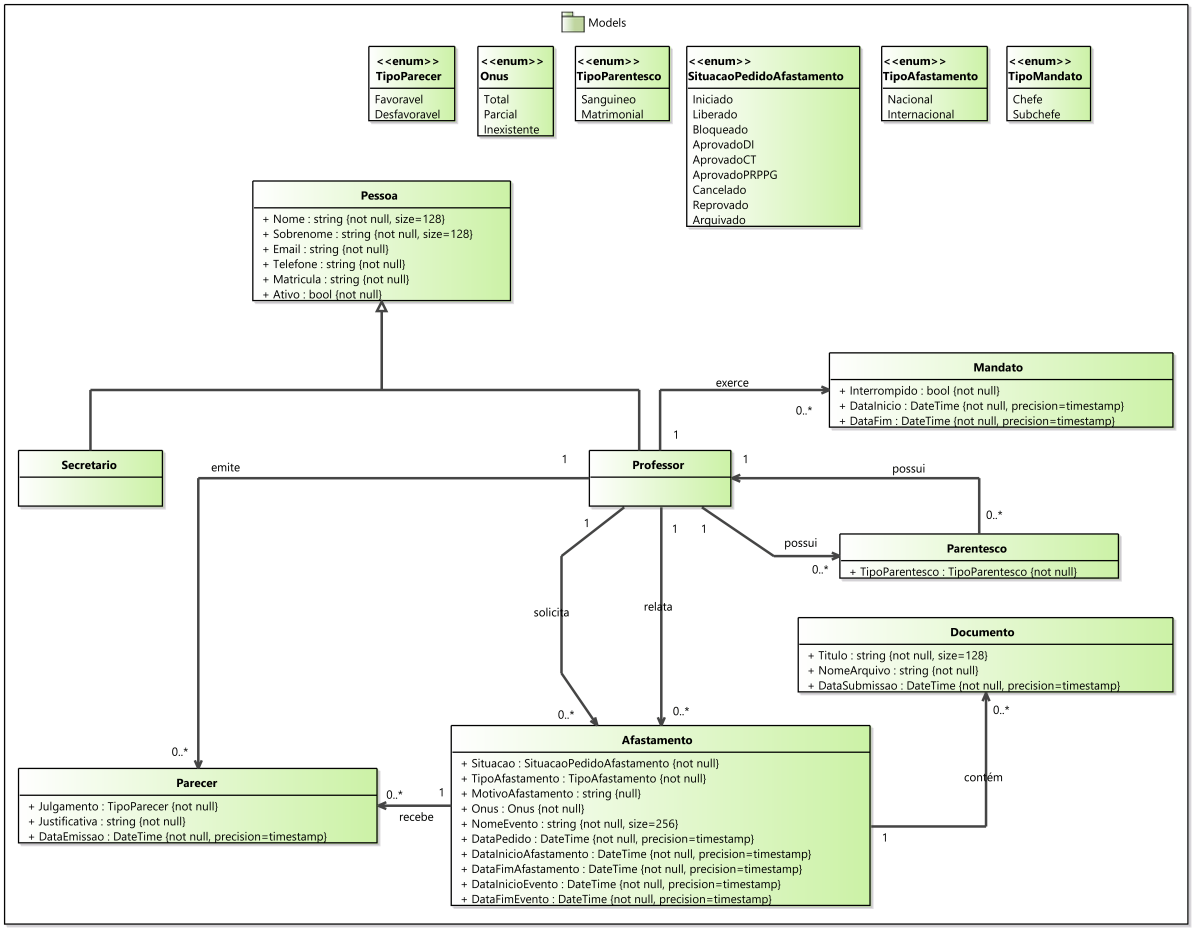

Figura 16 – Modelo de Entidades do SCAP (fase de projeto).

Diferente do modelo anterior, neste, todas as classes de entidade concentram-se no pacote "Models", com seus atributos devidamente tipados e acompanhados de regras (ou *constraints*) necessárias para que o *framework* de ORM (EF Core) faça o mapeamento das entidades para o banco de dados da forma esperada.

Como exemplo de *constraint*, podemos notar que todos os atributos possuem uma anotação **{null}** ou **{not null}**, indicando se o atributo poderá assumir ou não um valor nulo no banco de dados, respectivamente. Há também a regra **{size=«max»}**, responsável por indicar o tamanho máximo (representado pela variável **«max»**) de uma cadeia de caracteres (tipo **string**). Caso não seja definido um tamanho máximo, fica a cargo do *framework* de ORM ou da *engine* de banco de dados definir um limite padrão.

Se tratando de datas, o FrameWeb Editor nos permite escolher a precisão da informação, portanto, todo atributo do tipo **DateTime** é acompanhado de uma *constraint*

{precision=«date|time|timestamp»}, sendo **«date»** um valor com precisão de data, **«time»** com precisão de tempo e **«timestamp»** com precisão de data e tempo. No SCAP, todas as datas possuem precisão **«timestamp»** para validações mais apuradas e melhor controle de eventos do sistema (ações do *scheduler*).

Diferente do método FrameWeb, onde os as classes de domínio possuem atributos privados e métodos *getters* e *setters*, para atender às convenções da linguagem C#, todos os atributos foram definidos com visibilidade pública, representada pelo símbolo +, a esquerda de cada atributo. Essa adaptação foi necessária porque na linguagem escolhida para este trabalho, a definição de *getters* e *setters* adota uma sintaxe diferente da que foi idealizada para a criação do método FrameWeb, pois utiliza-se o conceito de propriedades ao invés de atributos, assim como mostrado num trecho de código da classe Documento na Listagem 4.1, onde as propriedades são definidas com visibilidade pública.

Listagem 4.1 – Convenção de *getters* e *setters* em C#

```
1  namespace SCAP.Models
2  {
3    public class Documento : Entity
4    {
5      [Required]
6      [StringLength(128)]
7      public string Titulo { get; set; }
8
9      [Required]
10     public string NomeArquivo { get; set; }
11
12     // ...
13   }
14 }
```

Neste modelo há definições de navegabilidade para cada uma das associações entre as entidades, assim como verbos que dão significado para os relacionamentos, mantendo as cardinalidades apresentadas no modelo da fase de análise de requisitos.

Por fim, temos também tipos enumerados, que, por sua vez, são definidos como classes esteriotipadas com a notação **«enum»**. Uma classe enumerada contém um conjunto finito de literais (identificadores) definidos pelo modelador ou programador, associados explicitamente ou não a valores inteiros e distintos entre si. Os tipos enumerados do SCAP são apresentados na Tabela 2.

A classe **Pessoa** herda de **IdentityUser**, classe esta provida pelo *framework* de segurança Identity. Esta herança será tratada apenas a nível de implementação, visto que a versão do FrameWeb Editor utilizada neste trabalho não possui suporte para modelar aspectos que dizem respeito a autenticação e autorização de usuários.

As classes **Mandato**, **Parentesco**, **Documento**, **Afastamento** e **Parecer** herdam a classe abstrata **Entity**, escondida do modelo através de recursos do editor, em prol

Tabela 2 – Tipos Enumerados do SCAP.

TipoParecer	
Favoravel	O parecer é favorável ao pedido de afastamento.
Desfavoravel	O parecer é desfavorável ao pedido de afastamento.
Onus	
Total	A universidade assumirá os custos da viagem para o evento.
Parcial	A universidade não assumirá os custos da viagem para o evento, mas manterá a remuneração normal do docente.
Inexistente	A universidade suspenderá a remuneração do docente.
TipoParentesco	
Sanguineo	Parentesco consanguíneo.
Matrimonial	Parentesco civil.
SituacaoPedidoAfastamento	
Iniciado	Situação dada ao pedido de afastamento quando criado.
Liberado	O pedido de afastamento está liberado para receber pareceres.
Bloqueado	O pedido de afastamento encontra-se bloqueado do fluxo normal de aprovação.
AprovadoDI	O pedido de afastamento foi aprovado pelo Departamento de Informática.
AprovadoCT	O pedido de afastamento foi aprovado pelo Centro Tecnológico.
AprovadoPRPPG	O pedido de afastamento foi aprovado pela Pró-Reitoria de Pesquisa e Pós-Graduação.
Cancelado	O solicitante desistiu do pedido de afastamento.
Reprovado	O pedido de afastamento foi reprovado.
Arquivado	O pedido de afastamento foi arquivado.
TipoAfastamento	
Nacional	O evento do afastamento é nacional.
Internacional	O evento do afastamento é internacional.
TipoMandato	
Chefe	O mandato é de chefe do departamento.
Subchefe	O mandato é de subchefe do departamento.

da organização e legibilidade. Esta classe tem como função prover um identificador único para todas as entidades do sistema, assim como para aproveitar recursos de tipos genéricos durante a fase de implementação. A classe **Entity** é apresentada na Listagem 4.2.

Listagem 4.2 – Classe Entity

```
1  namespace SCAP.Models
2  {
3    public abstract class Entity
4    {
5      [Key]
6      public Guid Id { get; set; }
7
8      public Entity()
9      {
10       Id = Guid.NewGuid();
```

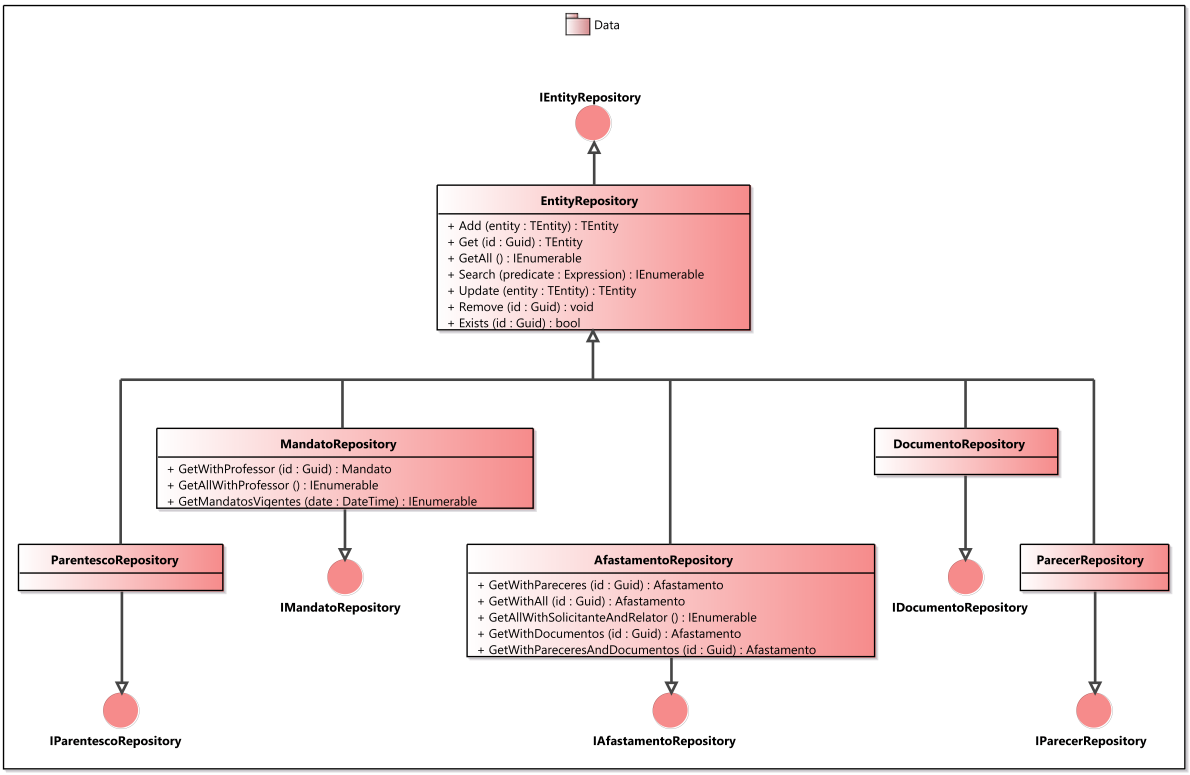

Figura 17 – Modelo de Persistência do SCAP (Entity).

```
11        }
12    }
13 }
```

4.3.2 Modelo de Persistência

As Figuras 17 e 18 apresentam o Modelo de Persistência do SCAP, particionado para fins de organização da monografia.

O Entity Framework Core, *framework* de ORM utilizado neste trabalho, possui uma camada de acesso a dados nativa, camada esta que satisfaz o padrão DAO. Dito isto, para não tornar o modelo redundante ou dispensável, foi utilizado o padrão **Repository**, que também é satisfeito pelo Entity Framework Core, porém com baixo reaproveitamento de código para consultas mais complexas ao banco de dados. Todas as interfaces e classes Repository estão situadas no pacote "Data".

O padrão Repository faz mediação entre o domínio do problema e a camada de mapeamento de dados, agindo como uma coleção de objetos de domínio em memória (FOWLER, 2002b). Este, por sua vez, está mais próximo do domínio do problema se comparado ao padrão DAO, elevando ainda mais o nível de abstração de acesso às informações persistidas no banco de dados.

O método FrameWeb sugere que as interfaces do modelo de persistência utilizem

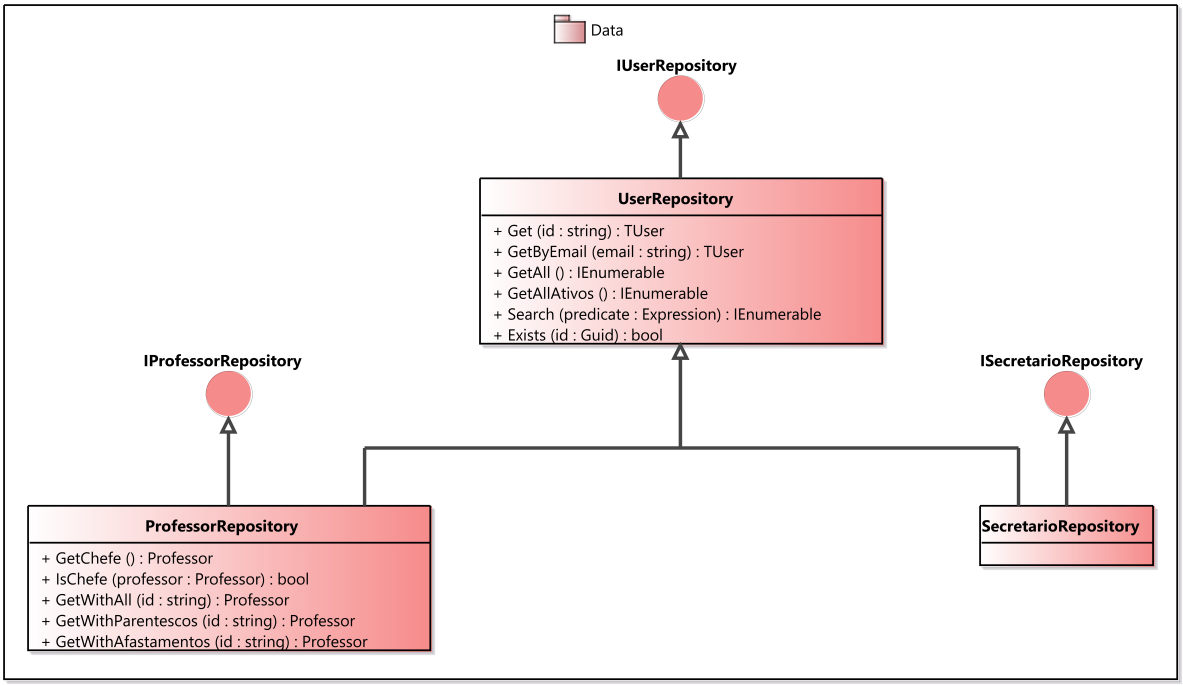

Figura 18 – Modelo de Persistência do SCAP (Pessoa/IdentityUser).

nomes que satisfaçam a lei de formação: "**«NomeEntidade»DAO**" (ex: AfastamentoDAO), enquanto que as classes que implementam essas interfaces tenham nome igual a: "**«NomeEntidade»DAO«NomeORM»**" (ex: AfastamentoDAOEF). Porém, como este trabalho não utiliza diretamente o padrão DAO e as sugestões de nomes do método FrameWeb destoam das convenções utilizadas em C# (interfaces começam com "I"), não faz sentido manter estas nomenclatura. Portanto, as interfaces das classes de persistência do SCAP foram chamadas de "**I«NomeEntidade»Repository**" (ex: IAfastamentoRepository), enquanto as classes que implementam as interfaces foram chamadas de "**«NomeEntidade»Repository**" (ex: AfastamentoRepository). Por decisão de projeto, o nome do *framework* de ORM foi desconsiderado da nomenclatura das classes de persistência, pois em caso de mudança de *framework*, basta que o template de geração de código das classes de persistência seja alterado ao invés do modelo.

A interface **IEntityRepository** tem como objetivo criar um contrato de métodos CRUD genéricos para efetuar operações no banco de dados, métodos estes implementados originalmente pela classe **EntityRepository**, tendo como alvo todas as classes de domínio que herdam **Entity**. As classes de persistência que herdam **EntityRepository** carregam também as implementações de seus métodos CRUD, assim como suas interfaces deveriam herdar a interface **IEntityRepository**, porém, por limitações do FrameWeb Editor, não é possível criar uma generalização entre duas interfaces, sendo feita a nível de código.

Como a classe **Pessoa** não herda Entity, mas sim **IdentityUser**, esta possui um Repository a parte, sendo representado pela interface **IUserRepository**, implementada

Tabela 3 – Métodos de EntityRepository.

EntityRepository	
Add	Adiciona uma entidade TEntity no banco de dados.
Get	Resgata uma entidade TEntity do banco de dados através de seu ID.
GetAll	Resgata todas as entidades TEntity do banco de dados.
Search	Busca entidades TEntity no banco de dados a partir de um predicado ou função anônima, utilizando a operação *WHERE*.
Update	Atualiza uma entidade TEntity no banco de dados.
Remove	Remove uma entidade TEntity do banco de dados a partir de seu ID.
Exists	Verifica se uma entidade TEntity existe no banco de dados através do seu ID.

Tabela 4 – Métodos específicos de AfastamentoRepository.

AfastamentoRepository	
GetWithPareceres	Resgata uma entidade Afastamento do banco de dados através de seu ID, carregando também os Pareceres associados.
GetWithAll	Resgata uma entidade Afastamento do banco de dados através de seu ID, carregando todas as suas entidades relacionadas.
GetAllWithSolicitanteAndRelator	Resgata todas as entidades Afastamento do banco de dados, carregando também as associações de Solicitante e Relator, para cada uma delas.
GetWithDocumentos	Resgata uma entidade Afastamento do banco de dados através de seu ID, carregando também os Documentos associados.
GetWithPareceresAndDocumentos	Resgata todas as entidades Afastamento do banco de dados, carregando também as associações de Pareceres e Documentos para cada uma delas.

pela classe **UserRepository**, contendo apenas métodos genéricos para operações de de leitura, uma vez que o Identity provê recursos nativos para operações de escrita, considerados apenas a nível de implementação. Semelhante ao caso do parágrafo anterior, as classes que herdam **UserRepository**, herdam também seus métodos de leitura, mas suas interfaces não herdam **IUserRepository** por limitações do editor.

A Tabela 3 detalha os métodos de **EntityRepository** que são herdados pelos Repositories que têm um descendente de Entity como alvo, o qual chamaremos de **TEntity**. Enquanto que a Tabela 4 detalha os métodos específicos de **AfastamentoRepository**, que, assim como os demais Repositories, tiveram seus nomes inspirados na convenção de nomes dos *query methods* do *framework* Spring Data JPA.[8]

[8] <https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#reference>

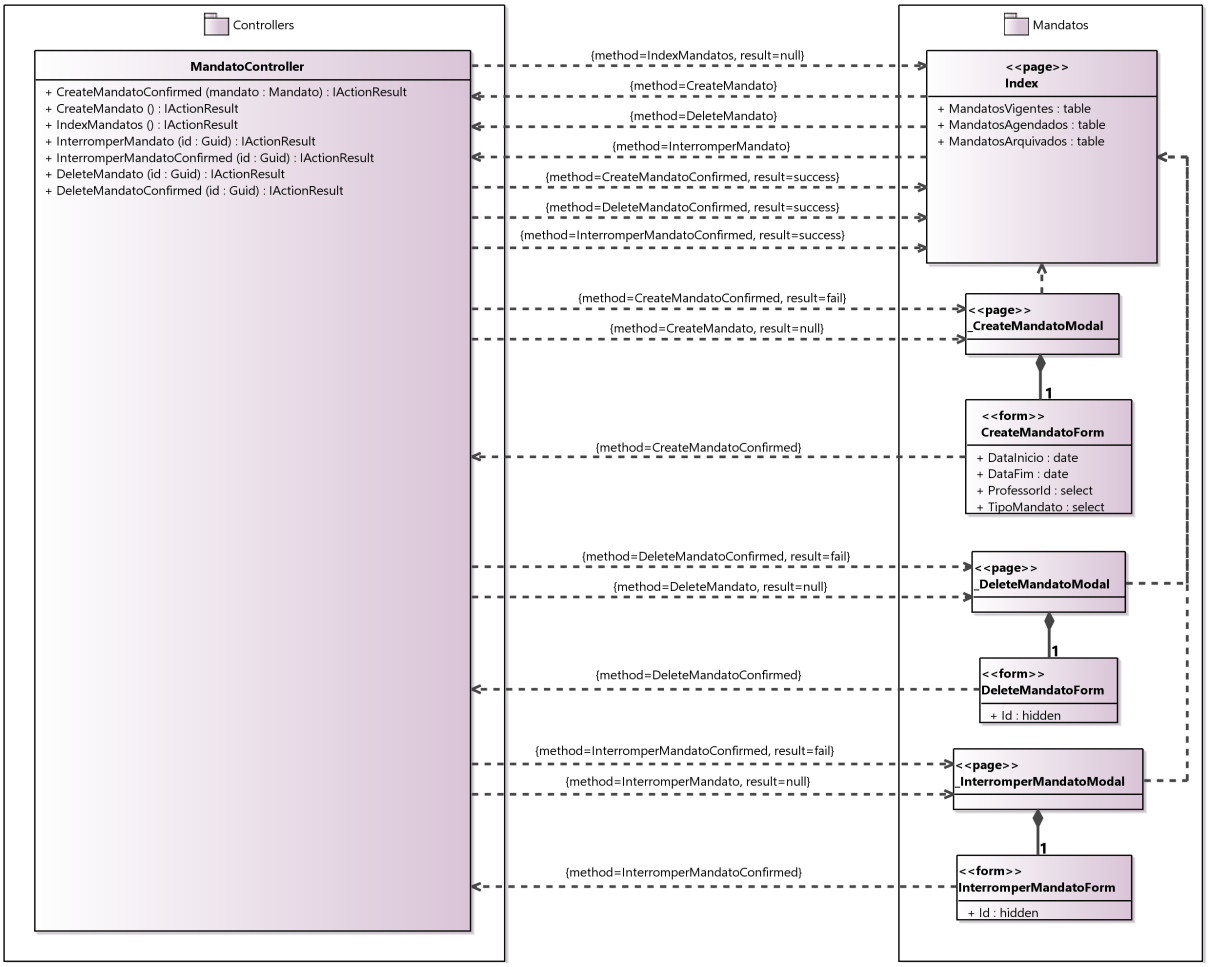

Figura 19 – Modelo de Navegação do SCAP (contexto de mandatos).

4.3.3 Modelo de Navegação

Tratando-se do SCAP, o Modelo de Navegação tomou grandes proporções, inviabilizando a exibição completa do diagrama na monografia. Portanto, a Figura 19 apresenta um fragmento deste modelo limitado ao contexto de mandatos. O Modelo de Navegação completo encontra-se disponível no repositório oficial deste trabalho,[9] requerendo o FrameWeb Editor para sua visualização.

Na proposta original, uma *controller* possui um atributo referente a uma entidade, tendo seus campos preenchidos de acordo com os valores recebidos pelo corpo da requisição HTTP, assim como explicado na Seção 2.3.2. Neste trabalho, os valores vindos da rota ou do corpo da requisição são mapeados para objetos de mesmo nome, definidos como parâmetro de uma ação ao invés de um atributo de classe. Como iremos trabalhar com *View Models*, explicadas na Seção 4.2.3, podemos ter mais de uma classe que represente uma entidade, portanto é interessante que estas estejam limitadas ao escopo dos métodos ao invés da classe Controladora em si. Todas as entidades utilizadas como parâmetro nas

[9] <https://bitbucket.org/vitorsouza-ufes/2018-lucas-silva/src/master/Project/Code/br.ufes.inf.nemo.scap.frameweb/>

ações das *controllers* do Modelo de Navegação do SCAP foram substituídas por *View Models* equivalentes na fase de implementação. Cada página Web do ASP.NET Core MVC, quando feita em Razor, pode ter definida qual será a *View Model* que aquela página atenderá. Portanto, não há necessidade de especificar o nome da propriedade a ser preenchida na *controller* como prefixo dos *inputs* dos formulários.

Para melhor dinamicidade da aplicação, ao invés de termos uma página para cada ação efetuada, foi utilizado em todo sistema uma estrutura conhecida como *modal*, uma pequena janela que abre sobre a página contendo informações de interesse (ex: formulário, notificação, etc.). Originalmente o método FrameWeb não possui suporte para este formato de UI (*User Interface*), porém Duarte (2014) propôs uma adaptação ao Modelo de Navegação, permitindo a utilização de Ajax (*Asynchronous JavaScript And XML*), fazendo com que as informações de um formulário pudessem ser submetidas e o resultado recebido sem que a página fosse recarregada. Porém, a adaptação em questão encontra-se acoplada ao modelo *Web Components*, adotado na proposta original do método FrameWeb (SOUZA, 2007), utilizando por exemplo a biblioteca *PrimeFaces*.[10] Contudo, nenhuma biblioteca equivalente foi encontrada para ASP.NET Core MVC, portanto foi necessária uma nova adaptação ao Modelo de Navegação, flexibilizando-o para que tenha como retorno fragmentos de uma página HTML (ou *Partial View*), oriundos das *controllers*.

Uma página Web com nome de prefixo "_" (ex:_CreateMandatoModal), é na verdade uma *Partial View* ao invés de uma página completa, estando ligado a sua página mãe através de uma associação *Page Dependency*, em destaque (marcada em ciano para ajudar na visualização) na Figura 20, com a qual assume um papel de pertencimento. Podemos ver os eventos que ocorrem entre as páginas Web do pacote Mandatos e a *controller* **MandatoController**, enumerados e marcados em vermelho.

1. Ao acessar a rota referente a página principal de mandatos (*/mandatos*), a requisição é atendida por uma instância de **MandatoController**, a qual através do método *IndexMandatos*, retorna a página **Index** para o cliente.

2. Na página **Index**, é requisitada à *controller* uma *Partial View* referente ao *modal* de criação de mandato, ação feita através do método *CreateMandato*.

3. É retornado o *modal* **_CreateMandatoModal**, que é composto pelo formulário **CreateMandatoForm**, contendo os *inputs* necessários para criar um mandato.

4. O formulário é submetido e através do método *CreateMandatoConfirmed* há uma tentativa de criar um novo mandato.

5. a) Caso o mandato seja criado com sucesso, o *modal* é fechado e a página **Index** mantida para novas interações.

[10] <https://www.primefaces.org/>

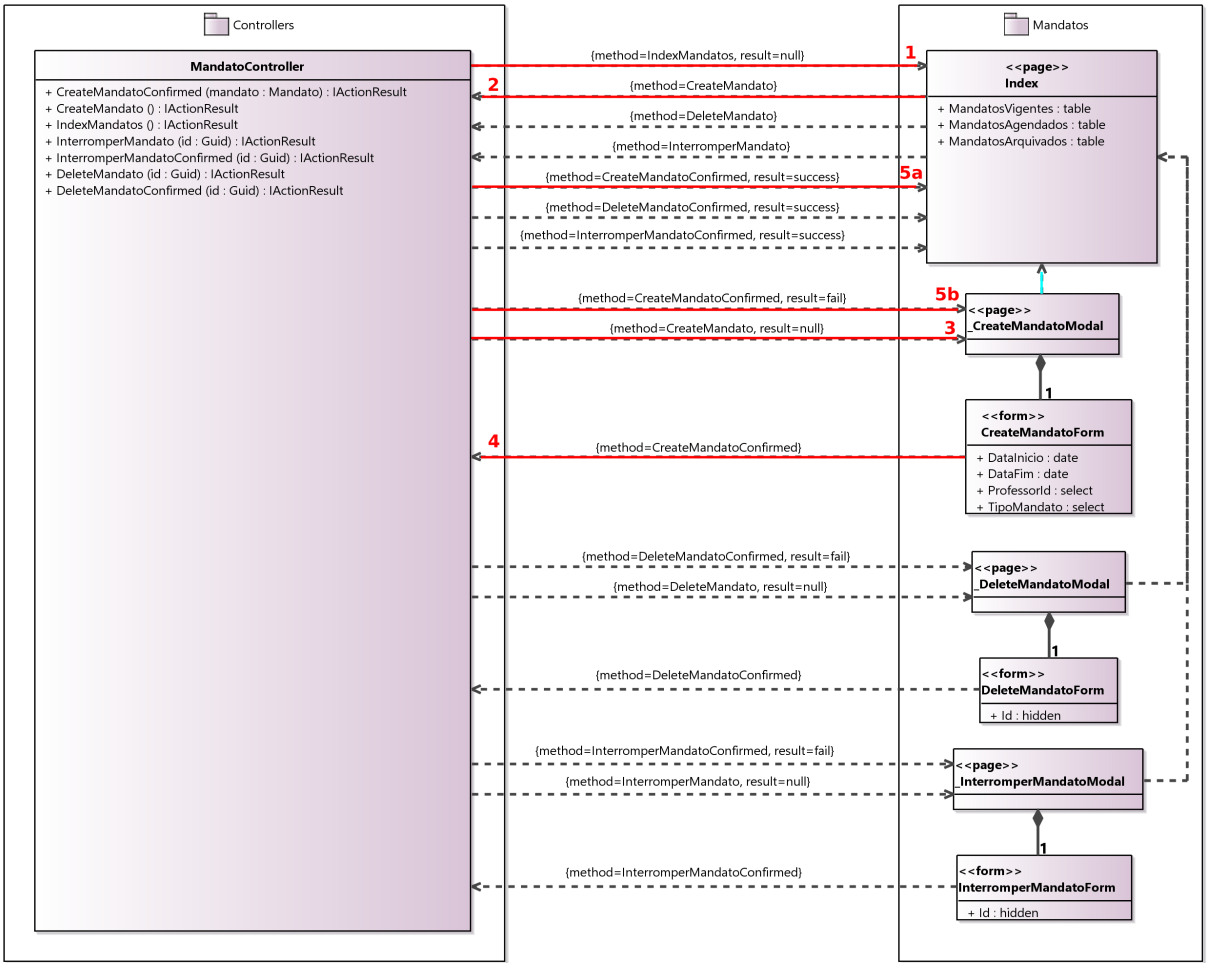

Figura 20 – Fluxo de requisições do Modelo de Navegação do SCAP orientado por cores e enumerações.

b) Caso haja alguma falha durante a criação do mandato, é retornada uma nova *Partial View* contendo o mesmo *modal* de criação de mandato, porém com mensagens indicando quais erros de validação ocorreram, dando ao usuário uma nova oportunidade de criar um mandato.

4.3.4 Modelo de Aplicação

A Figura 21 apresenta o Modelo de Aplicação do SCAP limitado ao contexto de afastamento. Esta redução de escopo fez-se necessária para a apresentação deste trabalho devido às dimensões tomadas pelo diagrama final. O Modelo de Aplicação completo encontra-se disponível no repositório oficial deste trabalho,[11] requerendo o FrameWeb Editor para sua visualização.

Análogo ao Modelo de Persistência apresentado na Seção 4.3.2, o Modelo de Aplicação possui uma classe abstrata **EntityService** a qual provê implementações dos métodos

[11] <https://bitbucket.org/vitorsouza-ufes/2018-lucas-silva/src/master/Project/Code/br.ufes.inf.nemo.scap.frameweb/>

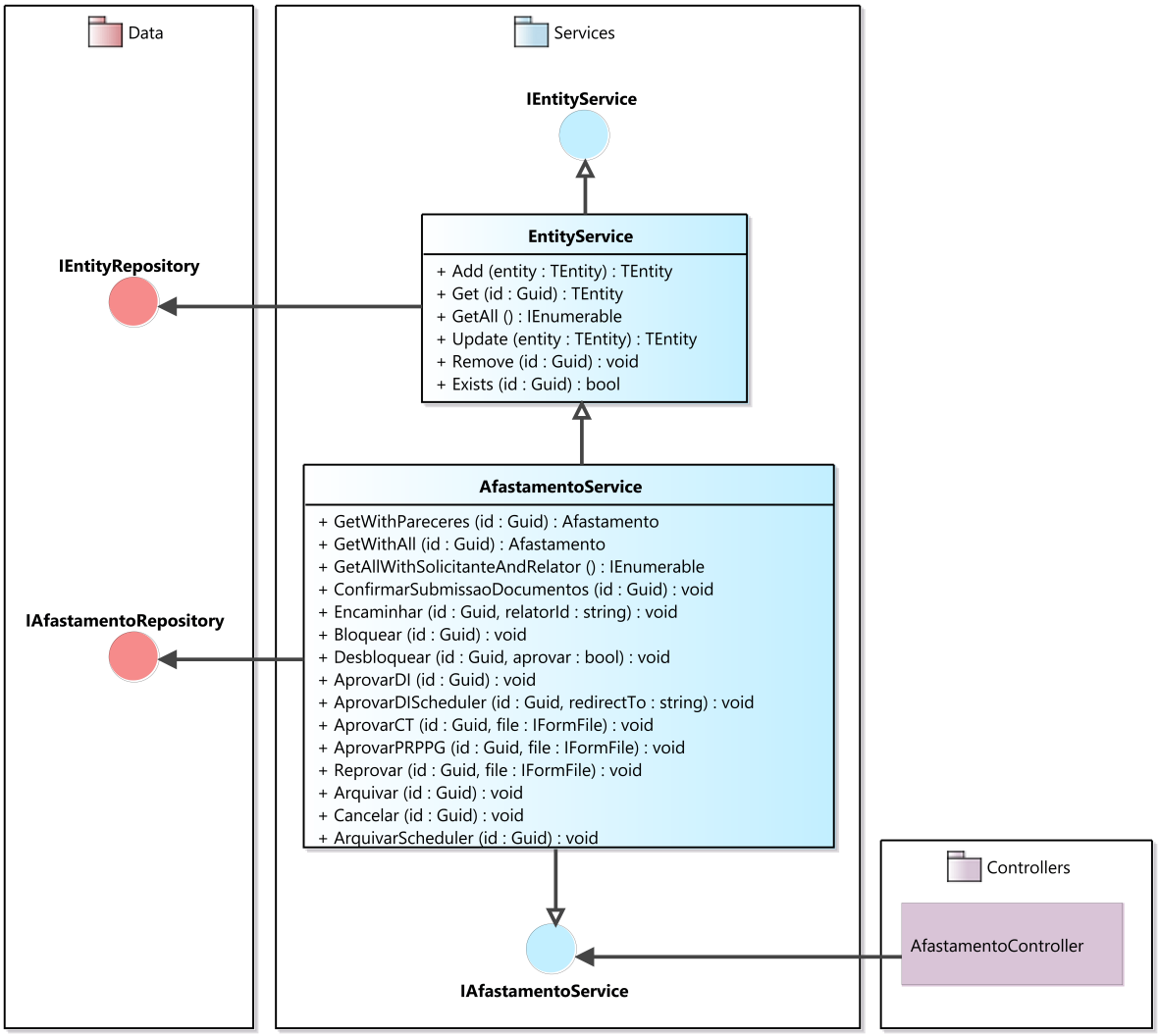

Figura 21 – Modelo de Aplicação do SCAP (contexto de afastamento).

de **IEntityService** para os serviços que têm como alvo uma entidade herdeira de **Entity**. O mesmo se aplica para a classe concreta **UserService** e sua interface **IUserService**, mas estas atendem aos serviços que têm como alvo uma entidade herdeira de **Pessoa**.

Novamente, devido a limitações do FrameWeb Editor, não foi possível criar generalizações entre as interfaces de serviço. Sendo assim, **IAfastamentoService**, que deveria herdar **IEntityService**, só pôde ter sua herança consolidada durante a fase de implementação. O mesmo se aplica para os demais serviços.

Neste modelo, **AfastamentoController** possui uma dependência com a interface **IAfastamentoService**, denonimada *Service Controller Association*, fazendo com que uma instância de **AfastamentoService** seja injetada em **AfastamentoController** através da interface da qual a mesma depende.

AfastamentoService depende da interface **IAfastamentoRepository**, associação denominada *DAO Service Association*. Porém, neste trabalho foi adotado o padrão *Unit*

of Work (ou Unidade de Trabalho), onde todos os Repositories se concentram dentro de uma única classe, que por sua vez é instanciada e injetada numa interface **IUnitOfWork** presente em cada um dos serviços que herdam **IEntityService** ou **IUserService**, ou seja, todos os serviços têm acesso a todos os Repositories.

Além de acesso aos Repositories, uma Unidade de Trabalho provê métodos que fazem controle de transações e pontos de persistência de alterações na base de dados (FOWLER, 2002c). Originalmente, o FrameWeb espera que transações de banco de dados sejam feitas através de métodos anotados com *@Transaction*, *data annotation* oriunda do *framework* de ORM JPA. Porém, não existe uma abordagem equivalente para ASP.NET Core, tornando necessária a adesão do padrão *Unit of Work* para impedir que os serviços se comuniquem diretamente com o contexto do banco de dados e acabe ferindo os princípios do método FrameWeb. A Listagem 4.3 apresenta o código da interface **IUnitOfWork**.

Listagem 4.3 – Interface IUnitOfWork.

```
1  using System;
2  using Microsoft.EntityFrameworkCore.Storage;
3  using SCAP.Models;
4
5  namespace SCAP.Data.Interfaces
6  {
7    public interface IUnitOfWork : IDisposable
8    {
9      IAfastamentoRepository Afastamentos { get; }
10     IDocumentoRepository Documentos { get; }
11     IMandatoRepository Mandatos { get; }
12     IParecerRepository Pareceres { get; }
13     IParentescoRepository Parentescos { get; }
14     IProfessorRepository Professores { get; }
15     ISecretarioRepository Secretarios { get; }
16
17     int SaveChanges();
18     IDbContextTransaction BeginTransaction();
19
20     IEntityRepository<TEntity> Repository<TEntity>() where TEntity : Entity;
21     IUserRepository<TUser> UserRepository<TUser>() where TUser : Pessoa;
22   }
23 }
```

4.4 Geração de Código e Implementação

Uma vez modelada a aplicação no FrameWeb Editor, para usufruir da *feature* de geração de código, foram desenvolvidos *templates* em Twig, voltados para ASP.NET Core e adaptados para o contexto do SCAP quando necessário. A Listagem 4.4 apresenta o *template* para geração de código das *controllers* utilizadas neste trabalho.

Listagem 4.4 – Template para geração de código das classes de Controle do ASP.NET Core MVC, adaptados para o contexto do SCAP.

```
using [...]

{% set entityName = (class.Name | replace({'Controller' : ''})) %}

namespace SCAP.{{ package.Name }}
{
  [Authorize, ActiveUser]
  public class {{ class.Name }} : BaseController
  {
    {% for association in associations %}
      private readonly {{ association.TargetMember.Type.Name }} _{{ association.
          TargetMember.Type.Name | slice(1, association.TargetMember.Type.Name.
          length()) | lower_first }};
    {% endfor %}

    public {{ class.Name }}({% for association in associations %}{{ association.
        TargetMember.Type.Name }} {{ association.TargetMember.Type.Name | slice(1,
         association.TargetMember.Type.Name.length()) | lower_first }}{% if loop.
        last == false %}, {% endif %}{% endfor %}IUserService<Pessoa> userService,
         IMapper mapper, INotificator notificator)
      : base(userService, mapper, notificator)
    {
      {% for association in associations %}
        _{{ association.TargetMember.Type.Name | slice(1, association.
            TargetMember.Type.Name.length()) | lower_first }} = {{ association.
            TargetMember.Type.Name | slice(1, association.TargetMember.Type.Name.
            length()) | lower_first }};
      {% endfor %}
    }

    {% set domainClassName = class.Name | replace({'Controller': ''}) %}

    {% for method in methods %}
      {% if method.RequestMethod == "POST" %}
        [ValidateAntiForgeryToken]
        [HttpPost, ActionName("{{ method.Name | replace({'Confirmed':''}) }})"]
      {% endif %}
      public IActionResult {{ method.Name }}({% for parameter in method.
          OwnedParameters %}{{ parameter.Type.Name }} {{ parameter.Name }}{% if
          loop.last == false %}, {% endif %}{% endfor %})
      {
        throw new NotImplementedException();
      }
    {% endfor %}
  }
}
```

Implementada e desenvolvida manualmente, a classe **BaseController** é herdada por todas as classes de controle do sistema, compartilhando métodos utilitários entre elas, visando evitar repetição de código para operações comuns. Além disso, todas as classes de

controle são anotadas com os filtros[12] *Authorize* e *ActiveUser*. A primeira trata-se de uma anotação da própria plataforma, que é implementada pelo Identity e força que apenas usuários logados acessem as ações da *controller*, enquanto *ActiveUser*, filtro implementado especificamente para o SCAP, permite que apenas usuários (entidade **Pessoa**) que estejam com a *flag* "Ativo" habilitada, acessem as funcionalidades do sistema.

Há também as definições das associações feitas no Modelo de Aplicação, onde são injetadas as dependências dos serviços via construtor. O construtor possui obrigatoriamente 3 parâmetros, são eles: **IUserService<Pessoa>** — serviço de usuário a nível de **Pessoa**; **IMapper** — recurso para operações *object-object mapping* com o AutoMapper, citado na Seção 4.2.3; **INotificator** — recurso implementado para compartilhamento de mensagens de erro entre as camadas de Navegação e Lógica de Negócio, capturadas durante os processos de validação feitos com o Fluent Validation, citado na Seção 4.2.4.

Por fim, são definidas as ações da *controller*, anotadas com *HttpPost* quando o método utiliza o verbo *post* nas requisições HTTP. A anotação *ActionName* dá ao método um *alias* para ser invocado, ou serve apenas para evitar conflitos entre métodos de mesmo nome e parâmetros, enquanto *ValidateAntiForgeryToken* é responsável por validar um *token* registrado nos formulários das páginas Web, tendo como intuito impedir ataques CSRF (*Cross-site request forgery*). A partir do *template* definido, conseguimos gerar a classe **ParentescoController**, apresentada na Listagem 4.5.

Listagem 4.5 – Resultado alcançado com a classe ParentescoController através da geração de código do Modelo de Navegação.

```
1  using [...]
2
3  namespace SCAP.Controllers
4  {
5    [Authorize, ActiveUser]
6    public class ParentescoController : BaseController
7    {
8      private readonly IParentescoService _parentescoService
9      private readonly IProfessorService _professorService;
10
11     public ParentescoController(IParentescoService parentescoService,
           IProfessorService professorService, IUserService<Pessoa> userService,
           IMapper mapper, INotificator notificator)
12       : base(userService, mapper, notificator)
13     {
14       _parentescoService = parentescoService;
15       _professorService = professorService;
16     }
17
18     public IActionResult CreateParentesco(string professorId)
19     {
20       throw new NotImplementedException();
```

[12] <https://docs.microsoft.com/aspnet/core/mvc/controllers/filters>

```
21      }
22
23      [ValidateAntiForgeryToken]
24      [HttpPost, ActionName("CreateParentesco)"]
25      public IActionResult CreateParentescoConfirmed(string professorId, Parentesco
            parentescso)
26      {
27        throw new NotImplementedException();
28      }
29
30      public IActionResult DeleteParentesco(Guid id)
31      {
32        throw new NotImplementedException();
33      }
34
35      [ValidateAntiForgeryToken]
36      [HttpPost, ActionName("DeleteParentesco)"]
37      public IActionResult DeleteParentescoConfirmed(Guid id)
38      {
39        throw new NotImplementedException();
40      }
41    }
42 }
```

Para tornar funcional a classe **ParentescoController** que foi gerada, esta precisou passar por algumas adaptações que vão além da implementação de seus métodos. Seu resultado final é apresentado pela Listagem 4.6.

Listagem 4.6 – Classe ParentescoController após a implementação sobre o código gerado.

```
1  using [...]
2
3  namespace SCAP.Controllers
4  {
5    [Authorize(Roles = "Secretario"), ActiveUser]
6    public class ParentescosController : BaseController
7    {
8      private readonly IProfessorService _professorService;
9      private readonly IParentescoService _parentescoService;
10
11     public ParentescosController(IProfessorService professorService,
           IParentescoService parentescoService,
12       IUserService<Pessoa> userService, IMapper mapper, INotificator notificator)
13       : base(userService, mapper, notificator)
14     {
15       _professorService = professorService;
16       _parentescoService = parentescoService;
17     }
18
19     [Route("usuarios/{professorId:guid}/parentescos/novo")]
20     public IActionResult Create(string professorId)
21     {
22       var professor = _professorService.GetWithParentescos(professorId);
23
24       if (professor == null)
25         return NotFound();
```

```
26
27        return ParentescoModal(professor);
28      }
29
30      [HttpPost]
31      [ValidateAntiForgeryToken]
32      [Route("usuarios/{professorId:guid}/parentescos/novo")]
33      public IActionResult Create(string professorId, ParentescoViewModel
            parentescoViewModel)
34      {
35        var professor = _professorService.GetWithParentescos(professorId);
36
37        if (professor == null)
38          return NotFound();
39
40        if (!ModelState.IsValid)
41          return ParentescoModal(professor);
42
43        _parentescoService.Add(Mapper.Map<Parentesco>(parentescoViewModel));
44
45        if (ErrorsOccurred())
46          return ParentescoModal(professor);
47
48        TempData["Success"] = "Parentesco criado com sucesso!";
49
50        return JsonOk(Url.Action("Details", "Users", new { id = professor.Id }));
51      }
52
53      private IActionResult ParentescoModal(Professor professor)
54      {
55        var except = professor.Parentescos
56          .Select(p => p.Parente.Id)
57          .Concat(new List<string> { professor.Id });
58
59        var professores = _professorService.GetAll()
60          .Where(p => !except.Contains(p.Id))
61          .Select(p => new
62          {
63            p.Id,
64            Label = $"{p.UserName} - {p.Nome} {p.Sobrenome}"
65          });
66
67        ViewBag.ParenteId = new SelectList(professores, "Id", "Label");
68
69        return PartialView("Modals/_CreateParentescoModal", new ParentescoViewModel
              { ParenteDeId = professor.Id });
70      }
71
72      [Route("parentescos/{id:guid}/excluir")]
73      public IActionResult Delete(Guid id)
74      {
75        var parentesco = _parentescoService.Get(id);
76
77        if (parentesco == null)
78          return NotFound();
79
80        return PartialView("Modals/_DeleteParentescoModal", Mapper.Map<
```

```
            ParentescoViewModel>(parentesco));
81      }
82
83      [ValidateAntiForgeryToken]
84      [HttpPost, ActionName("Delete")]
85      [Route("parentescos/{id:guid}/excluir")]
86      public IActionResult DeleteConfirmed(Guid id)
87      {
88        var parentesco = _parentescoService.Get(id);
89
90        if (parentesco == null)
91          return NotFound();
92
93        _parentescoService.Remove(id);
94
95        if (ErrorsOccurred())
96          return PartialView("Modals/_DeleteParentescoModal", Mapper.Map<
              ParentescoViewModel>(parentesco));
97
98        TempData["Success"] = "Parentesco excluído com sucesso!";
99
100       return JsonOk(Url.Action("Details", "Users", new { id = parentesco.
              ParenteDeId }));
101     }
102   }
103 }
```

No produto final de **ParentescoController**, o filtro *Authorize* recebeu o parâmetro *Roles="Secretario"*, permitindo que apenas secretários utilizem as ações da *controller*. Além disso, cada um dos métodos públicos passou a ser anotado com o atributo *Route*, parametrizado com uma *string* indicando qual a rota (URL) de acesso àquela ação. Os métodos foram renomeados, removendo o nome da entidade da assinatura dos mesmos (ex: CreateParentesco virou Create e CreateParentescoConfirmed virou CreateConfirmed). As figuras 22, 23 e 24 apresentam os resultados das telas de detalhes de usuário e a ação de adicionar parentescos funcionando.

A implementação completa deste trabalho encontra-se disponível em: <https://bitbucket.org/vitorsouza-ufes/2018-lucas-silva/src/master/Project/Code/>.

4.5 Demonstração das telas do SCAP

Esta seção tem como intuito apresentar algumas das principais telas do SCAP, demonstrando o fluxo de eventos na visão de diferentes usuários do sistema, como secretário, professor solicitante e professor não solicitante de um pedido de afastamento. Algumas telas e passos foram omitidos, pois tratam-se de processos repetitivos ou muito semelhantes aos que já serão apresentados.

A Figura 25 apresenta o *modal* contendo o formulário de solicitação de afastamento na visão do professor solicitante. Após a submissão do formulário de solicitação, o professor

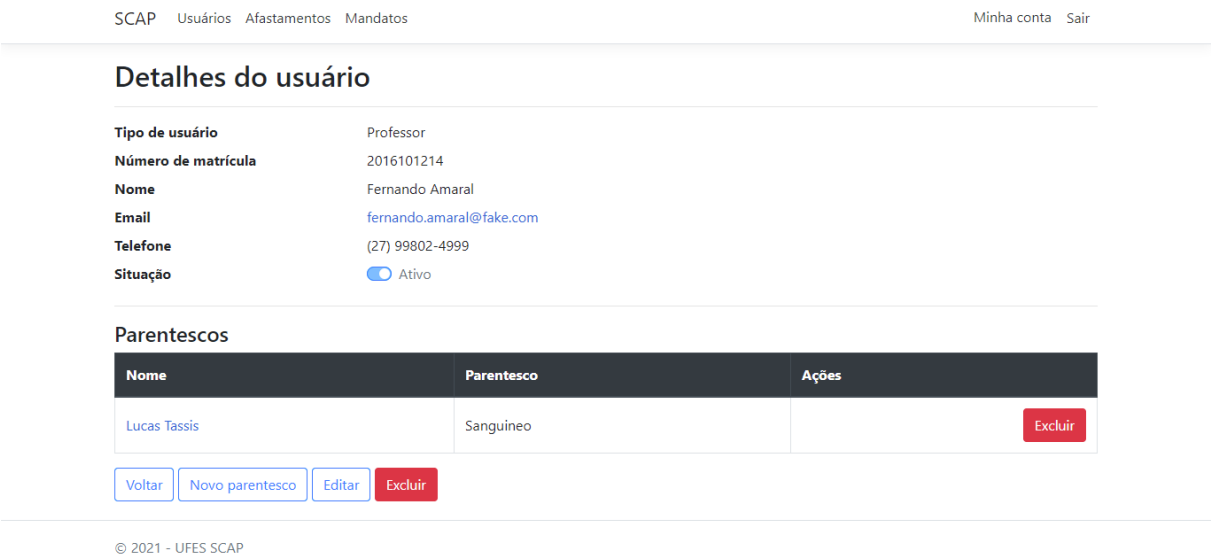

Figura 22 – Tela de detalhes do usuário.

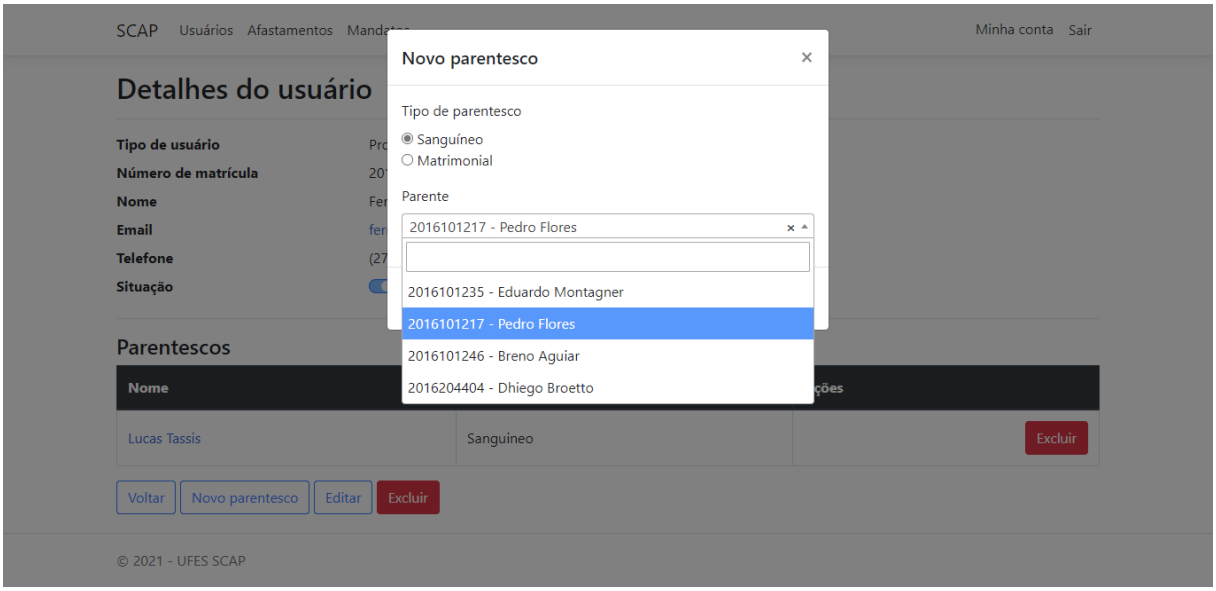

Figura 23 – Tela de detalhes do usuário com *modal* para adicionar um parentesco aberto.

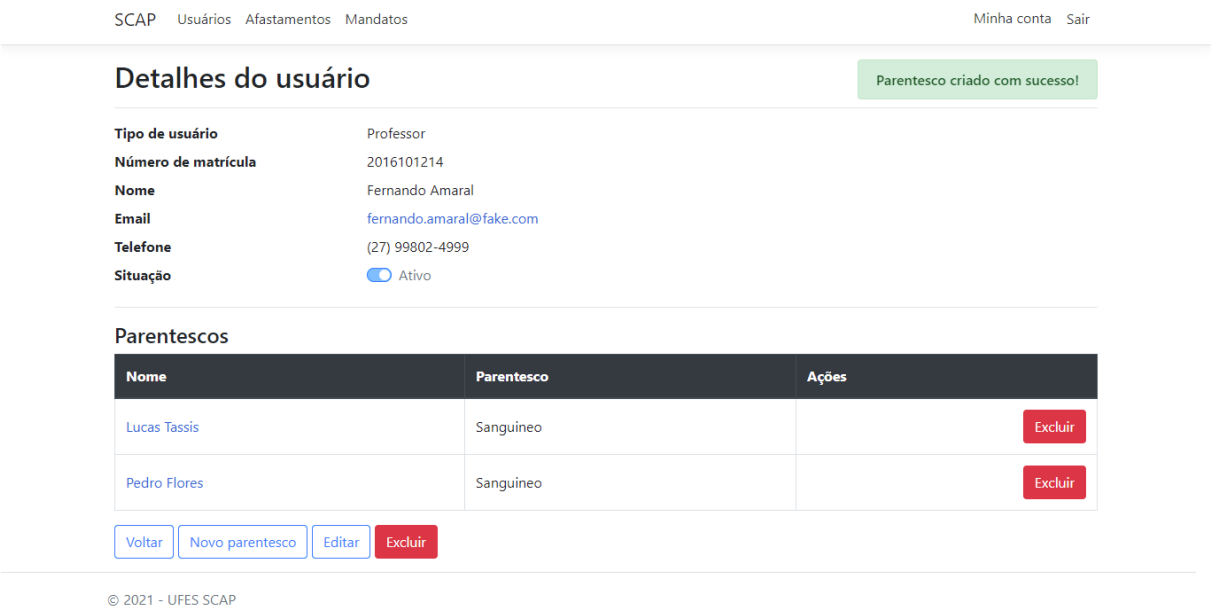

Figura 24 – Tela de detalhes do usuário após a criação de um novo parentesco, atualizada e com mensagem de *feedback* da ação.

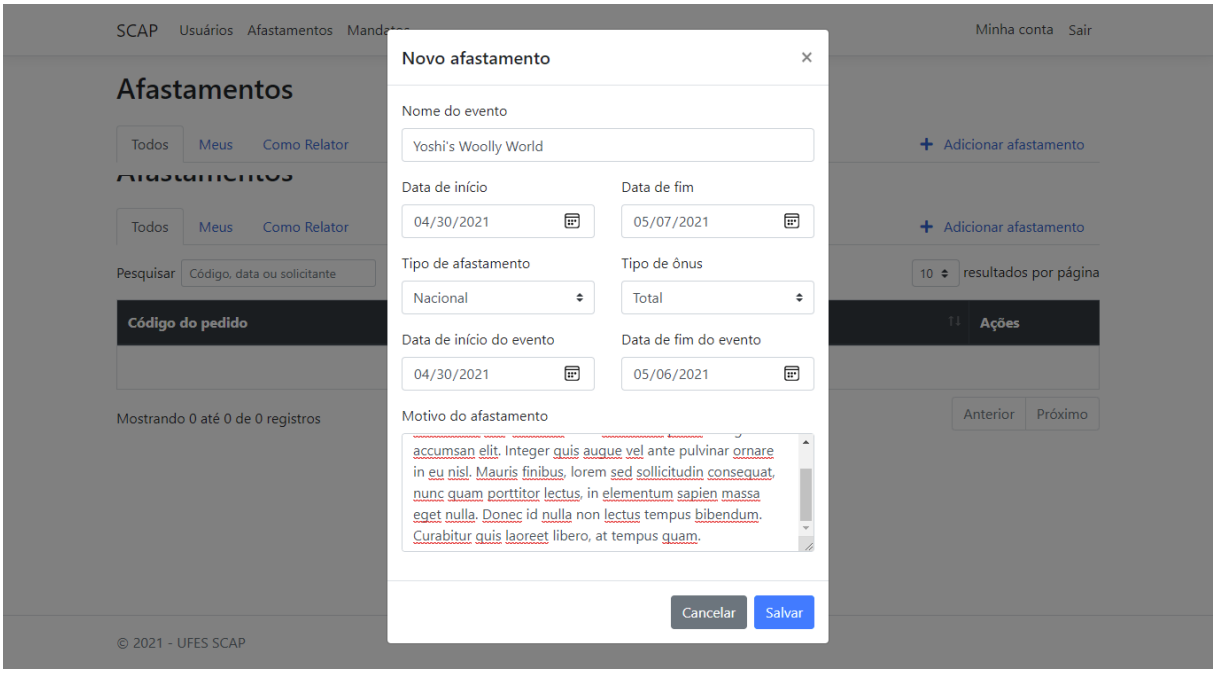

Figura 25 – Formulário de solicitação de afastamento.

se depara com a tela de detalhes do seu pedido, apresentada na Figura 26. Aqui ele irá adicionar os documentos necessários e confirmar a conclusão do pedido.

Finalizado o pedido de afastamento por parte do solicitante, os demais professores do departamento irão receber por e-mail uma notificação informando que há um novo pedido de afastamento passível de receber pareceres, assim como mostra a Figura 27. Após clicar no *link* do e-mail, o professor não solicitante será redirecionado para a página de

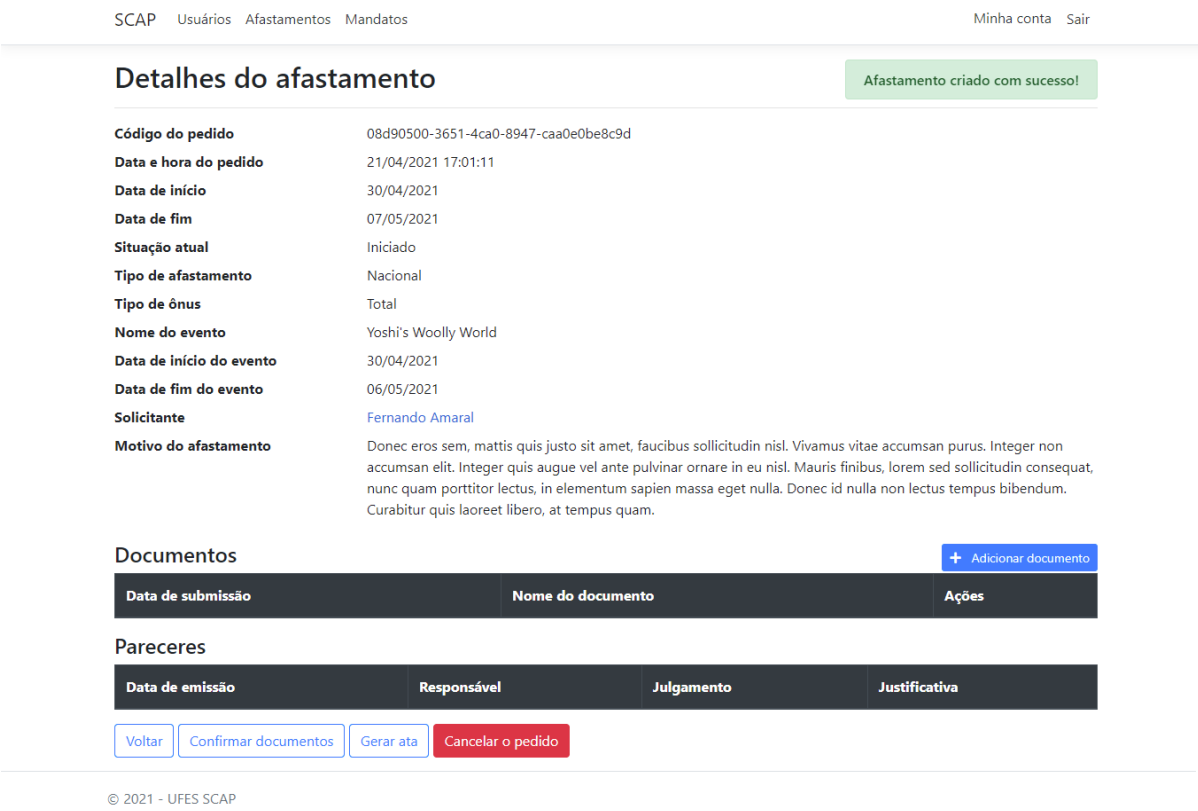

Figura 26 – Tela de detalhes de um pedido de afastamento na visão do solicitante.

detalhes do pedido, podendo então submeter seu parecer, assim como mostra a Figura 28.

Submetido o parecer desfavorável, o solicitante será notificado sobre o bloqueio do seu pedido, cabendo a ele aguardar a reunião semanal do departamento para que alguma decisão seja tomada. A Figura 29 mostra a visão de um secretário aprovando o pedido de afastamento até então bloqueado, considerando que houve concordância de todos os membros do departamento durante a reunião.

Aprovado o pedido de afastamento por parte do DI, tanto o professor solicitante quanto os secretários serão notificados sobre a aprovação do pedido, sendo disponibilizada uma ata de aprovação em nome do DI, para que assim os secretários possam dar continuidade aos trâmites fora do departamento. A ata é apresentada na Figura 30.

A Figura 31 apresenta o formulário de criação de mandato de chefe ou subchefe de departamento na visão de secretário, único usuário que tem permissão para isso. Já a Figura 32 apresenta o painel de configuração da conta do usuário logado.

As Figuras 33 e 34 apresentam as páginas principais de pedidos de afastamento e usuários, respectivamente. Contendo recursos de busca e paginação, disponíveis para todos os usuários, limitando a criação de pedidos de afastamento para professores e a criação de usuários para secretários.

Para ter acesso a qualquer conteúdo do SCAP, é necessário que o usuário esteja

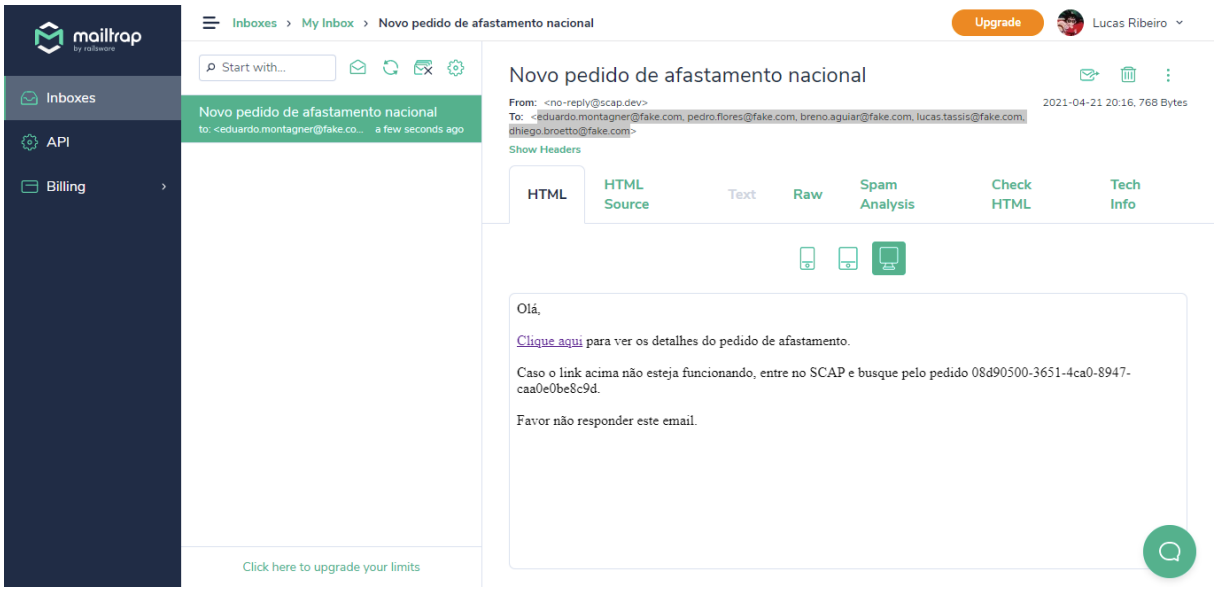

Figura 27 – E-mail recebido por professores não solicitantes, informando sobre um novo pedido de afastamento.

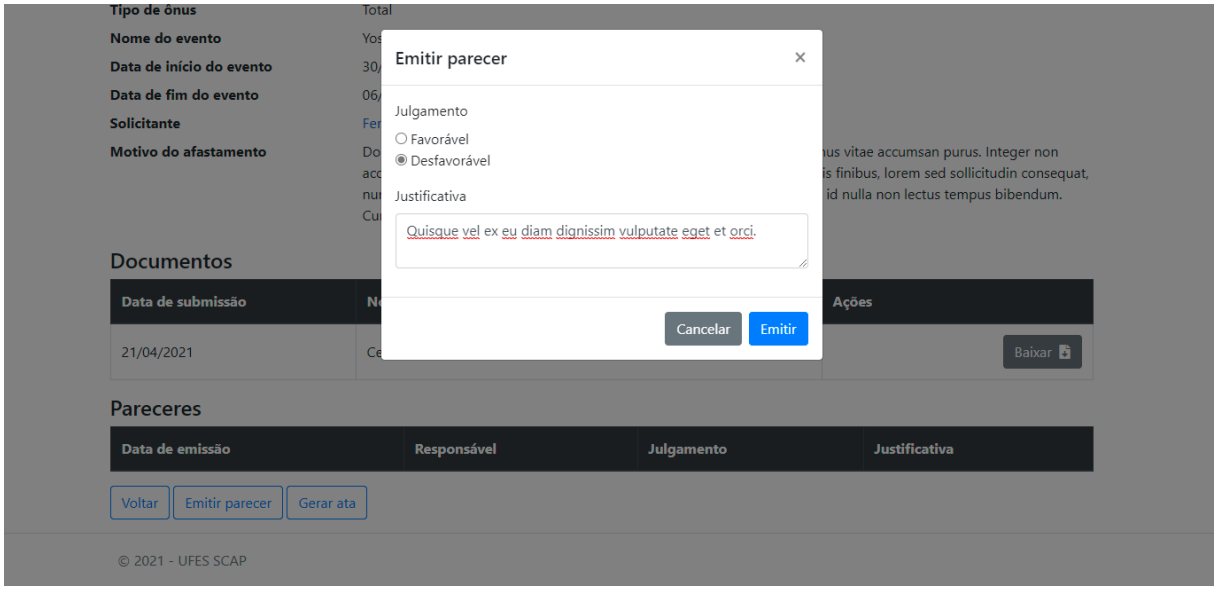

Figura 28 – Formulário de submissão de um parecer desfavorável na visão do professor não solicitante.

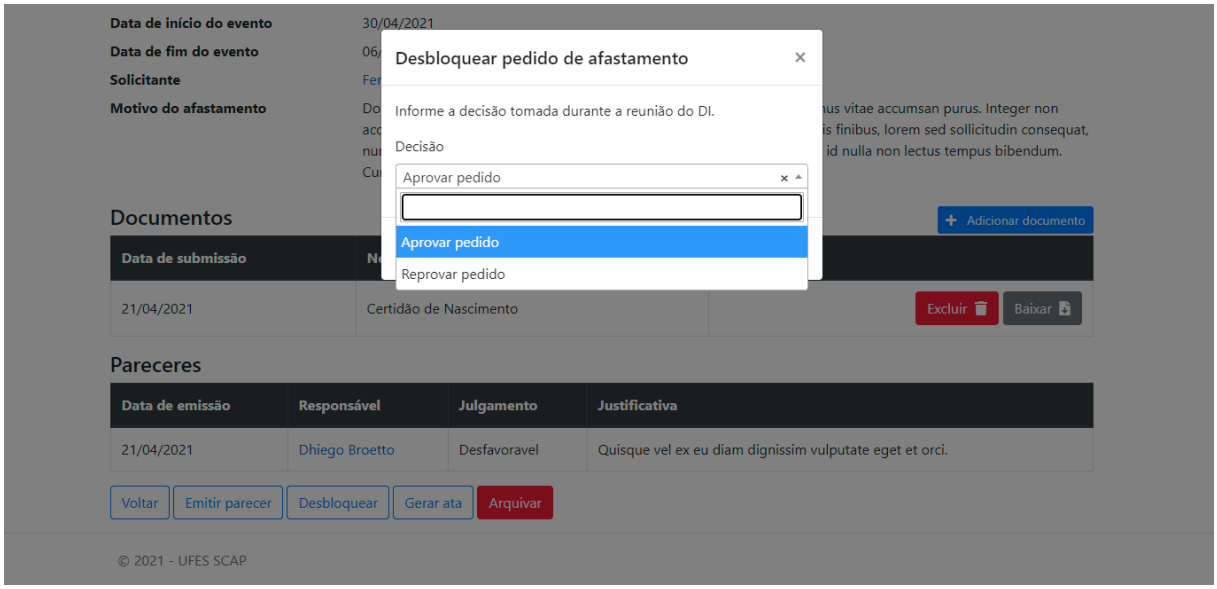

Figura 29 – Formulário de desbloqueio de um pedido afastamento na visão de secretário.

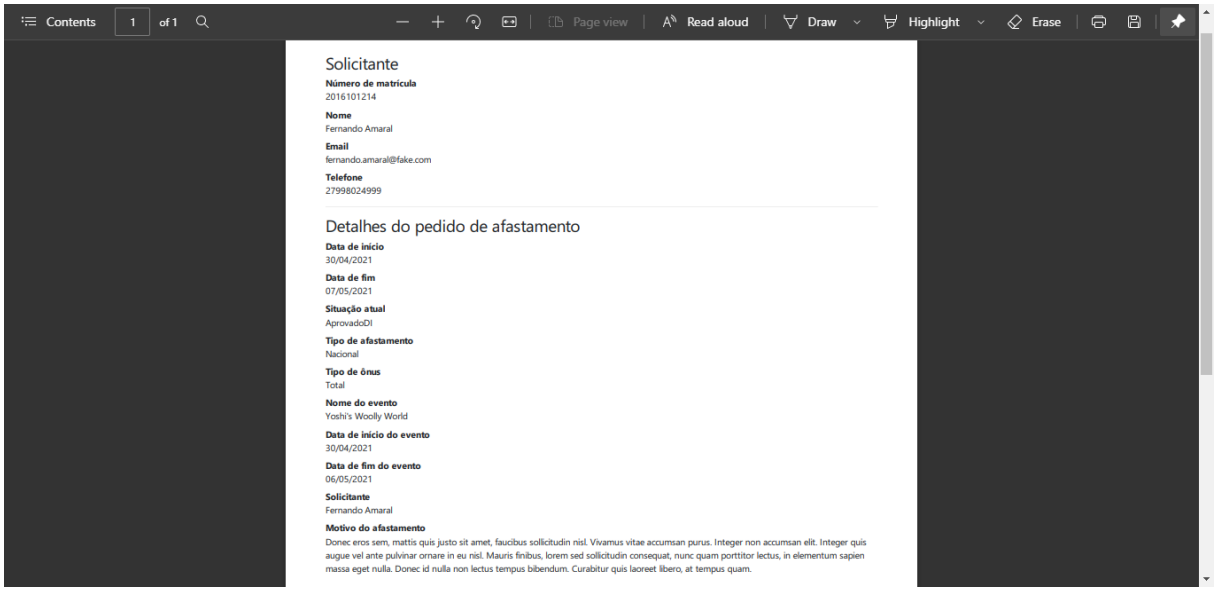

Figura 30 – Ata de aprovação do DI.

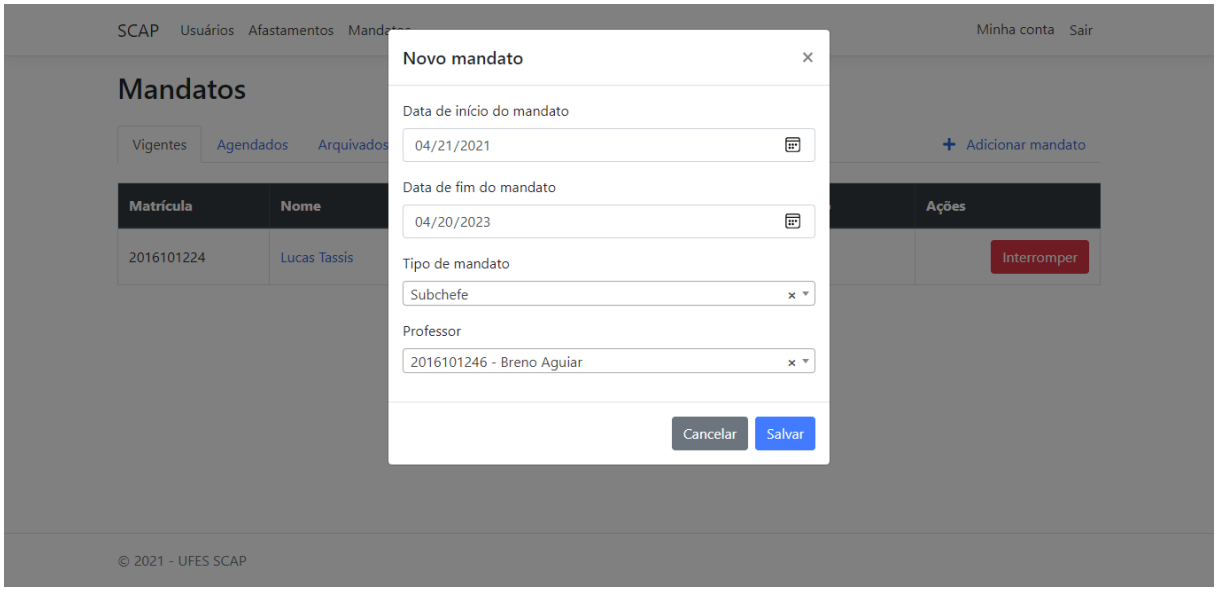

Figura 31 – Formulário de criação de mandato de chefe ou subchefe do departamento na visão de secretário.

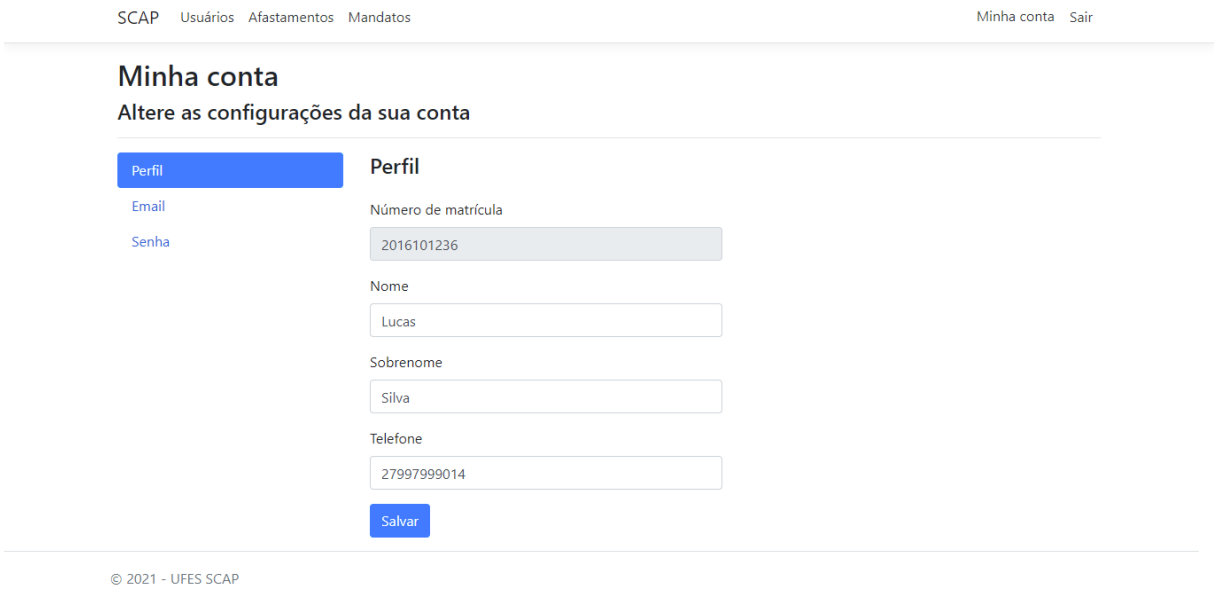

Figura 32 – Página de configuração da conta de usuário.

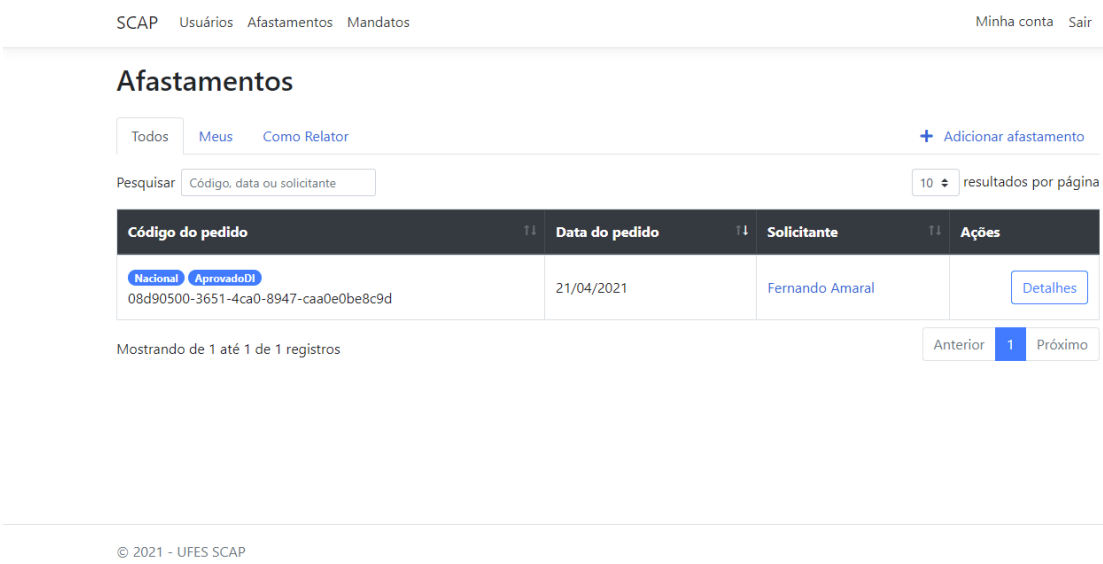

Figura 33 – Página com a lista de pedidos de afastamento na visão de professor.

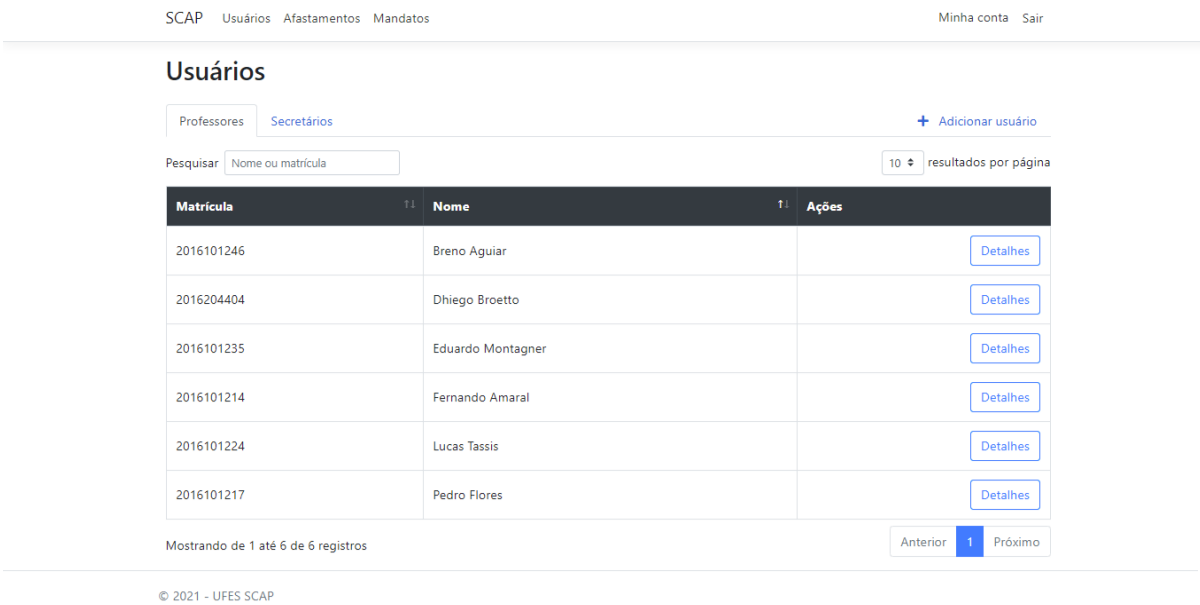

Figura 34 – Página com a lista de usuários na visão de secretário.

Figura 35 – Página de *login*.

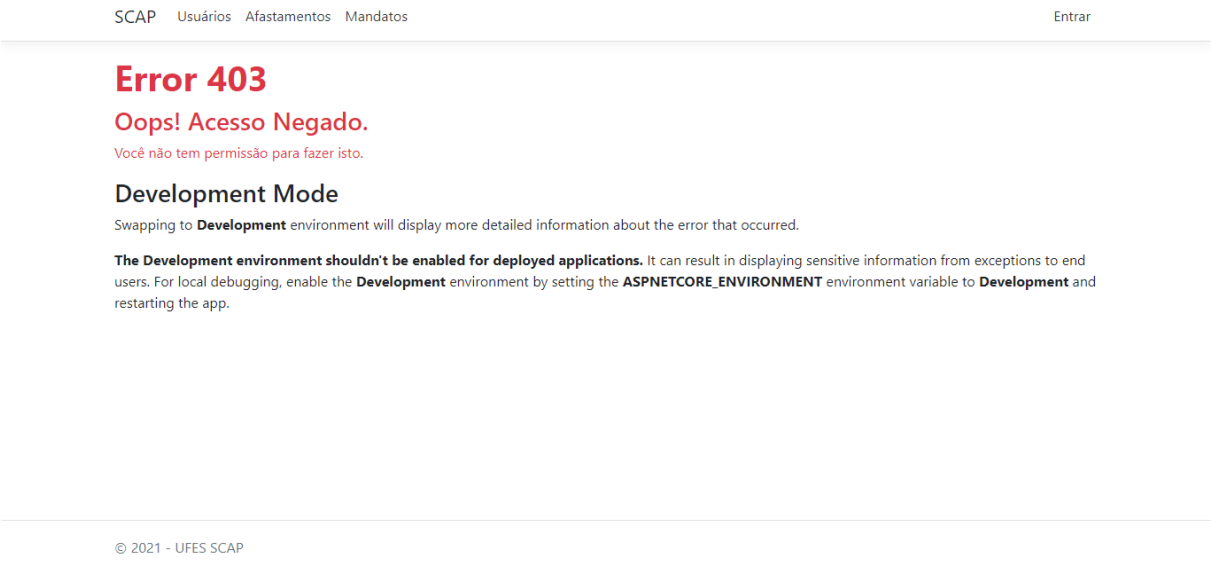

Figura 36 – Página genérica para erros (em modo de desenvolvimento).

devidamente logado, portanto a Figura 35 apresenta a página de *login* do sistema. Caso haja algum acesso indevido que fuja das condições esperadas e tratadas pelo sistema, uma página genérica de erro será exibida, assim como mostra a Figura 36.

5 Considerações Finais

Neste trabalho, todos os objetivos planejados foram plenamente alcançados, sendo o principal deles a aplicação do método FrameWeb no desenvolvimento do SCAP, utilizando ASP.NET Core. Mediante adaptações do método, citadas nas seções 4.3 e 4.4 e, com apoio das ferramentas FrameWeb, o SCAP foi integralmente implementado, seguindo rigorosamente a documentação revisada e especificada para o mesmo. Esta, por sua vez, encontra-se disponível nos apêndices deste trabalho.

As seções que se seguem abordam os demais objetivos citados na Introdução. A Seção 5.1 traz as boas impressões e análises dos resultados obtidos durante a utilização do método FrameWeb e seu ferramental. Já a Seção 5.2 discute as desvantagens e problemas encontrados com a utilização do FrameWeb neste trabalho. Por fim, a Seção 5.3 sugere melhorias e novas funcionalidades tanto para o método quanto para suas ferramentas, através de recomendações para trabalhos futuros.

5.1 Discussão e Apresentação de Resultados

A arquitetura proposta pelo método FrameWeb, combinada ao conjunto de *frameworks* definidos na Seção 2.2, fez com que o desenvolvimento do SCAP em ASP.NET Core fosse facilitado, desenvolvendo-o sobre um padrão bem definido, favorecendo o reaproveitamento de código e modularização das diferentes partes do sistema, cabendo a cada uma delas lidar com suas respectivas responsabilidades.

Os modelos propostos pelo método desempenharam um papel fundamental no que diz respeito a documentação do WIS, e por seus diagramas serem originados a partir da já conhecida UML, a curva de aprendizagem foi consideravelmente baixa. Por terem sido desenvolvidos no FrameWeb Editor, onde foram restritos apenas aos construtos válidos da linguagem, diminuiu-se a suscetibilidade a falhas e fugas das restrições da linguagem.

Ainda que utilizando tecnologias que fogem do ecossistema Java, plataforma a qual o método FrameWeb foi idealizado, inicialmente, este apresentou bons resultados, sendo responsável pela geração de um código bem estruturado, de fácil manutenção e passível de evolução. Apesar das adaptações necessárias, o produto final não destoou muito daquilo que foi proposto originalmente pelo método. Dessa forma, podemos considerar que o método FrameWeb traz uma boa abordagem para o desenvolvimento de WISs em ASP.NET Core, tendo potencial para tornar-se uma proposta verdadeiramente genérica.

Mas a principal vantagem da utilização do método atualmente, além da arquitetura baseada em *frameworks*, é a geração de código sobre os modelos feitos no FrameWeb

Tabela 5 – Taxas de aproveitamento de código das entidades.

Arquivo	Linhas geradas	Linhas cód. final	Aproveitamento (%)
Models/Afastamento.cs	136	58	88.03
Models/Documento.cs	56	24	80.42
Models/Entity.cs	31	20	74.41
Models/Mandato.cs	56	31	78.08
Models/Onus.cs	7	15	53.88
Models/Parecer.cs	61	27	79.91
Models/Parentesco.cs	43	23	78.39
Models/Pessoa.cs	82	32	64.09
Models/Professor.cs	56	28	89.93
Models/Secretario.cs	20	13	90.61
Models/SituacaoPedidoAfastamento.cs	7	21	69.44
Models/TipoAfastamento.cs	7	14	58.37
Models/TipoMandato.cs	7	14	53.59
Models/TipoParecer.cs	7	14	56.89
Models/TipoParentesco.cs	7	14	57.64

Editor. A ferramenta de geração de código traz ao desenvolvedor a possibilidade de fazer da documentação também um artefato funcional para a fase de implementação, gerando quase todo o "esqueleto" do sistema, o que acaba por poupar o desenvolvedor de despender tempo reescrevendo diversas estruturas similares (*boilerplate code*) e garantindo fidelidade à documentação, e estas vantagens foram todas aproveitadas no desenvolvimento do SCAP.

As definições de *frameworks* e *templates* criados neste trabalho foram facilitados graças a similaridade entre as linguagens Java e C#, pois estas foram em maior parte adaptadas do JButler (SOUZA, 2020). Todos os artefatos de código gerados puderam ser aproveitados durante a fase de implementação. Alguns servindo apenas como ponto de partida e outros aproveitados quase que por completo, como por exemplo as interfaces ou classes que não necessitaram de nenhuma implementação além daquelas vindas das classes de base que foram herdadas.

As tabelas 5, 6, 7, 8 e 9 apresentam, para cada arquivo, o número de linhas geradas e o número de linhas após a implementação dos métodos, assim como o percentual de similaridade entre os códigos gerados e implementados, que foram calculados utilizando a técnica *Dice-coefficient*,[1] aplicada através da biblioteca *string-similarity*, feita em Node.js.

Em linhas gerais, tivemos uma média de aproveitamento de 60.52%, sendo um resultado razoável se comparado aos 68∼94% de Souza (2020), mas também levando em consideração as adaptações necessárias aos modelos do método e aos artefatos de código gerados, discorridas nas Seções 4.3 e 4.4, respectivamente. Poderíamos ter obtido resultados ainda melhores, porém estaríamos sendo tendenciosos quanto ao *overfitting* dos *templates* voltados para o SCAP, o que acabaria fugindo da proposta original do método.

[1] <https://en.wikipedia.org/wiki/S%C3%B8rensen%E2%80%93Dice_coefficient>

Tabela 6 – Taxas de aproveitamento de código dos serviços.

Arquivo	Linhas geradas	Linhas cód. final	Aproveitamento (%)
Services/AfastamentoService.cs	188	604	17.09
Services/DocumentoService.cs	34	72	44.29
Services/EmailService.cs	56	114	34.78
Services/EntityService.cs	89	133	35.19
Services/MandatoService.cs	67	98	40.56
Services/ParecerService.cs	23	147	15.98
Services/ParentescoService.cs	23	90	30.67
Services/ProfessorService.cs	56	46	53.74
Services/SecretarioService.cs	23	16	72.07
Services/UserService.cs	155	216	34.03
Services/Interfaces/IAfastamentoService.cs	135	28	91.27
Services/Interfaces/IDocumentoService.cs	23	11	67.28
Services/Interfaces/IEmailService.cs	39	14	84.77
Services/Interfaces/IEntityService.cs	63	17	82.54
Services/Interfaces/IMandatoService.cs	47	17	94.21
Services/Interfaces/IParecerService.cs	15	10	59.21
Services/Interfaces/IParentescoService.cs	15	14	90.37
Services/Interfaces/IProfessorService.cs	39	16	91.54
Services/Interfaces/ISecretarioService.cs	15	15	83.56
Services/Interfaces/IUserService.cs	111	28	86.30

Tabela 7 – Taxas de aproveitamento de código dos *repositories*.

Arquivo	Linhas geradas	Linhas cód. final	Aproveitamento (%)
Data/AfastamentoRepository.cs	73	61	63.73
Data/ApplicationDbContext.cs	17	62	27.44
Data/DocumentoRepository.cs	18	19	92.42
Data/EntityRepository.cs	95	61	60.90
Data/MandatoRepository.cs	51	40	69.12
Data/ParecerRepository.cs	18	19	92.24
Data/ParentescoRepository.cs	18	18	87.84
Data/ProfessorRepository.cs	73	80	46.05
Data/SecretarioRepository.cs	18	21	87.59
Data/UserRepository.cs	84	58	58.93
Data/Interfaces/IAfastamentoRepository.cs	55	18	98.42
Data/Interfaces/IDocumentoRepository.cs	15	14	96.69
Data/Interfaces/IEntityRepository.cs	71	21	87.56
Data/Interfaces/IMandatoRepository.cs	39	16	97.97
Data/Interfaces/IParecerRepository.cs	15	14	96.62
Data/Interfaces/IParentescoRepository.cs	15	14	96.73
Data/Interfaces/IProfessorRepository.cs	55	18	96.76
Data/Interfaces/ISecretarioRepository.cs	15	14	94.18
Data/Interfaces/IUserRepository.cs	63	18	83.06

Tabela 8 – Taxas de aproveitamento de código das *controllers*.

Arquivo	Linhas geradas	Linhas cód. final	Aproveitamento (%)
Controllers/AfastamentosController.cs	146	394	38.79
Controllers/DocumentosController.cs	70	141	49.05
Controllers/MandatosController.cs	85	175	46.64
Controllers/PareceresController.cs	46	181	33.51
Controllers/ParentescosController.cs	64	132	51.13
Controllers/UsersController.cs	114	292	44.66

Tabela 9 – Taxas de aproveitamento de código das páginas Web.

Arquivo	Linhas geradas	Linhas cód. final	Aproveitamento (%)
Views/Afastamentos/Details.cshtml	18	286	8.28
Views/Afastamentos/Index.cshtml	19	195	23.09
Views/Afastamentos/Modals/_ArquivarAfastamentoModal.cshtml	10	20	32.41
Views/Afastamentos/Modals/_CancelarAfastamentoModal.cshtml	10	20	32.41
Views/Afastamentos/Modals/_ConfirmarSubmissaoDocumentosModal.cshtml	10	20	35.12
Views/Afastamentos/Modals/_CreateAfastamentoModal.cshtml	88	83	84.01
Views/Afastamentos/Modals/_DesbloquearAfastamentoModal.cshtml	25	31	60.24
Views/Afastamentos/Modals/_EncaminharAfastamentoModal.cshtml	31	28	65.29
Views/Documentos/Modals/_CreateDocumentoModal.cshtml	22	31	48.38
Views/Documentos/Modals/_DeleteDocumentoModal.cshtml	10	20	31.73
Views/Mandatos/Index.cshtml	19	207	13.78
Views/Mandatos/Modals/_CreateMandatoModal.cshtml	50	47	80.90
Views/Mandatos/Modals/_DeleteMandatoModal.cshtml	10	20	29.87
Views/Mandatos/Modals/_InterromperMandatoModal.cshtml	10	20	30.30
Views/Pareceres/Modals/_CreateParecerModal.cshtml	40	30	74.58
Views/Parentescos/Modals/_CreateParentescoModal.cshtml	42	34	73.49
Views/Parentescos/Modals/_DeleteParentescoModal.cshtml	10	20	31.95
Views/Users/DefinePassword.cshtml	40	44	76.07
Views/Users/DefinePasswordConfirmed.cshtml	1	11	0.00
Views/Users/Details.cshtml	6	148	1.75
Views/Users/Index.cshtml	13	123	16.52
Views/Users/Modals/_CreateUserModal.cshtml	61	51	81.71
Views/Users/Modals/_DeleteUserModal.cshtml	10	20	29.87
Views/Users/Modals/_EditUserModal.cshtml	53	49	81.21

5.2 Problemas do FrameWeb

Durante o processo de desenvolvimento do SCAP neste trabalho, foram identificados alguns problemas existentes no método FrameWeb e em suas ferramentas, são eles:

Método FrameWeb

- Apesar de não ser explicitamente proibido, não há no método FrameWeb a ideia de que um serviço possa fazer utilização de outro serviço, abrindo margem para o desperdício do reúso de uma lógica já implementada.

- A elaboração do Modelo de Navegação através de diagramas UML acaba tornando o processo de modelagem entre páginas Web, formulários e *controllers* um tanto quanto caótico, dadas as proporções que o mesmo tende a tomar devido a enorme quantidade de elementos necessários para representar o modelo, o que dificultou muito a visualização e entendimento do mesmo neste trabalho.

- Requisições assíncronas via Ajax amarradas ao formato *Web Components*, discutido na Seção 4.3.3, requerendo adaptações para o uso de *Partial Views*.

- Definição de métodos transacionais através de *data annotations*. Nem todos os *frameworks* possuem essa característica, requerendo adaptações como a do padrão *Unit*

of Work, apresentado na Seção 4.3.4, no qual as transações são feitas programaticamente.

- Na data corrente deste trabalho, o padrão DAO já é comum para *frameworks* de ORM como EF Core, Eloquent, Doctrine e etc., fazendo com que o Modelo de Persistência torne-se redundante caso não seja adaptado para um padrão como o Repository, discutido na Seção 4.3.2.

FrameWeb Editor

- Todos os problemas mencionados na lista de problemas do **Método FrameWeb** valem também para o editor, uma vez que os modelos feitos no FrameWeb Editor refletem diretamente nos modelos propostos originalmente pelo método FrameWeb.
- Não são permitidas heranças entre interfaces, o que acaba por prejudicar uma hierarquia que deveria ser bem definida e documentada.
- Não há distinção visual de qual classe um método pertence, havendo então conflitos entre métodos de mesmo nome, mesmo que de classes distintas. Há também situações onde, eventualmente, o editor duplica ou desaparece com os nomes dos métodos referenciados por algum elemento, impactando diretamente na geração de código.
- Desaparecimento das associações do Modelo de Navegação, reduzindo drasticamente a confiabilidade do editor como ferramenta de documentação.
- Dificuldade para manter os diagramas organizados, visto que qualquer pequena movimentação de um elemento no quadro acarreta numa completa desorganização dos diagramas, fazendo com que todo o processo de modelagem seja uma "briga" entre o modelador e o editor.
- Há um excesso de opções nas tabelas de configuração de elementos FrameWeb, sendo que a maioria das opções não são utilizadas ou possuem valor padrão contrário às necessidades mais comuns (ex: todo atributo de entidade começa com *unique=true*).
- Originalmente, o Twig não tem como objetivo executar transformações de modelo em texto, portanto os *templates* FrameWeb utilizados para geração de código acabam mostrando-se pobres e pouco legíveis, contendo muita repetição de código, manipulação de *strings* sobre nome de classes, problemas de espaçamento e quebras de linha, além de não contar com recursos para *debug* no editor. Problemas como estes tornam inviável a geração de código para linguagens como Python, que dependem obrigatoriamente de indentação, além de dar ao desenvolvedor o trabalho de ter que organizar cada arquivo gerado.

- O gerador de código carece de documentação técnica, tendo como únicas referências uma Wiki[2] e trabalhos passados, fazendo com que o desenvolvedor despenda muito tempo descobrindo como os *templates* funcionam.

- Dependência da IDE Eclipse e de um projeto Java Web para inicializar um projeto FrameWeb. O que acaba tornando o processo pouco prático para aqueles que não desejam utilizar a plataforma Java para desenvolver um WIS.

5.3 Trabalhos Futuros

Para trabalhos futuros que envolvam o método FrameWeb e seu conjunto de ferramentas de apoio, são sugeridas algumas mudanças, são elas:

- Maior flexibilização do uso da tecnologia *Ajax*, permitindo incluir fragmentos de páginas Web no retorno de chamadas assíncronas, requerendo adaptações como as feitas na Seção 4.3.3.

- Criação de uma linguagem textual, DSL (*Domain-specific language*), para melhor representação do Modelo de Navegação até então feito em UML.

- Modernização do método FrameWeb para dar suporte ao desenvolvimento de WISs que fazem utilização do modelo arquitetural REST.

- Correção dos problemas de usabilidade do editor, apresentados na Seção 5.2, visando melhorar a experiência de usuário ao modelar um WIS no editor.

- Integração com o Acceleo,[3] *template engine* que originalmente utiliza uma abordagem orientada a modelos e tem como principal função executar a transformação de modelo em texto. Essa proposta tem como intuito melhorar os resultados de código gerado e a experiência de usuário no desenvolvimento de novos *templates* através de uma linguagem apropriada, visando corrigir os problemas de usabilidade do gerador de código, apresentados na Seção 5.2.

- Portar o FrameWeb Editor para outra plataforma que não seja a IDE Eclipse, como por exemplo uma vertente Web.

[2] <https://github.com/nemo-ufes/FrameWeb/wiki/ToolsTutorial01>
[3] <https://www.eclipse.org/acceleo/>

Referências

ALMEIDA, N. V. de; CAMPOS, S. L.; SOUZA, V. E. S. A Model-Driven Approach for Code Generation forWeb-based Information Systems Built with Frameworks. In: *Proc. of the 23rd Brazilian Symposium on Multimedia and the Web (WebMedia 2017)*. Gramado, RS, Brazil: ACM, 2017. p. 245–252. Citado na página 14.

ALUR, D.; MALKS, D.; CRUPI, J. *Core J2EE Patterns: Best Practices and Design Strategies (2nd Edition)*. 2. ed. Prentice Hall, 2003. ISBN 0131422464. Disponível em: <http://www.corej2eepatterns.com/Patterns2ndEd/index.htm>. Citado 2 vezes nas páginas 24 e 25.

AVELAR, R. d. A. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Ninja*. Vitória, ES, Brasil, 2018. Citado na página 15.

BAUER, C.; KING, G. *Hibernate em ação*. [S.l.]: Ciência Moderna, 2005. ISBN 8573934042. Citado na página 21.

CAMPOS, S. L.; SOUZA, V. E. S. FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb. In: *Anais do 16º Workshop de Ferramentas e Aplicações, 23º Simpósio Brasileiro de Sistemas Multimedia e Web (WFA/WebMedia 2017)*. Gramado, RS, Brazil: SBC, 2017. p. 199–203. Citado 4 vezes nas páginas 8, 14, 16 e 26.

DUARTE, B. B. *Aplicação do Método FrameWeb no Desenvolvimento de um Sistema de Informação na Plataforma Java EE 7*. Vitória, ES, Brasil, 2014. Citado 5 vezes nas páginas 14, 15, 16, 33 e 51.

FALBO, R. A. *Notas de aula de Engenharia de Software*. 2014. Disponível em: <http://falbo.inf.ufes.br/files/ES/Notas_Aula_Engenharia_Software.pdf>. Citado na página 16.

FALBO, R. A. *Notas de aula de Engenharia de Requisitos*. 2017. Disponível em: <http://falbo.inf.ufes.br/files/ER/Notas_Aula_Engenharia_Requisitos.pdf>. Citado 4 vezes nas páginas 16, 19, 37 e 38.

FALBO, R. A. *Notas de aula de Projeto de Sistemas de Software*. 2018. Disponível em: <http://falbo.inf.ufes.br/files/PSS/Notas_Aula_Projeto_Sistemas_2018.pdf>. Citado na página 16.

FERREIRA, M. T. *Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Wicket e Tapestry*. Vitória, ES, Brasil, 2018. Citado na página 15.

FOWLER, M. *Data Transfer Object*. 2002. Disponível em: <https://martinfowler.com/eaaCatalog/dataTransferObject.html>. Citado na página 42.

FOWLER, M. *Patterns of Enterprise Application Architecture*. USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0321127420. Citado 2 vezes nas páginas 24 e 47.

FOWLER, M. *Unit of Work*. 2002. Disponível em: <https://martinfowler.com/eaaCatalog/unitOfWork.html>. Citado na página 54.

FOWLER, M. *Inversion of Control Containers and the Dependency Injection pattern*. 2004. Disponível em: <https://www.martinfowler.com/articles/injection.html>. Citado na página 21.

GUTERRES, C. S. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o framework Play*. Vitória, ES, Brasil, 2019. Citado na página 15.

MARTINS, B. F.; SOUZA, V. E. S. A Model-Driven Approach for the Design of Web Information Systems based on Frameworks. In: *Proc. of the 21st Brazilian Symposium on Multimedia and the Web (WebMedia 2015)*. Manaus, AM, Brazil: [s.n.], 2015. p. 41–48. Disponível em: <http://dl.acm.org/citation.cfm?id=2820439>. Citado 3 vezes nas páginas 14, 16 e 26.

MATOS, R. P. de. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando os frameworks Spring MVC e Vaadin*. Vitória, ES, Brasil, 2017. Citado na página 15.

MEIRELLES, L. C. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Codeigniter e NodeJS*. Vitória, ES, Brasil, 2019. Citado na página 15.

MICROSOFT. *ASP.NET Core 3.1*. 2019. Disponível em: <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1>. Citado 2 vezes nas páginas 14 e 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 15.

PRADO, R. C. d. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework VRaptor 4*. Vitória, ES, Brasil, 2015. Citado 4 vezes nas páginas 14, 15, 16 e 33.

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach*. 6. ed. [S.l.]: McGraw Hill, 2005. ISBN 007301933X. Citado 2 vezes nas páginas 8 e 18.

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 9 vezes nas páginas 8, 18, 19, 20, 21, 22, 23, 24 e 51.

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 8 vezes nas páginas 8, 23, 25, 27, 28, 29, 30 e 69.

SOUZA, V. E. S.; FALBO, R. A. An Agile Approach for Web Systems Engineering. In: *Proc. of the 11th Brazilian Symposium on Multimedia and the Web (WebMedia 2005)*. ACM, 2005. p. 1–3. Disponível em: <http://portal.acm.org/citation.cfm?doid=1114223.1114237>. Citado na página 14.

SOUZA, V. E. S.; FALBO, R. A.; GUIZZARDI, G. Designing Web Information Systems for a Framework-based Construction. In: HALPIN, T.; PROPER, E.; KROGSTIE, J. (Ed.). *Innovations in Information Systems Modeling: Methods and Best Practices*. 1. ed. [S.l.]: IGI Global, 2009. cap. 11, p. 203–237. Citado 2 vezes nas páginas 14 e 16.

TRELLO. *Kanban*. 2011. Disponível em: <https://trello.com/pt-BR>. Citado na página 17.

ZUPELI, B. L.; SOUZA, V. E. S. Integração de um Gerador de Código ao FrameWeb Editor. In: *Anais Estendidos do 24º Simpósio Brasileiro de Sistemas Multimedia e Web - Workshop de Ferramentas e Aplicações (WFA/WebMedia 2018)*. Salvador, BA, Brazil: SBC, 2018. p. 109–113. Citado 2 vezes nas páginas 14 e 16.

Apêndices

Documento de Definição de Requisitos

Projeto: SCAP - Sistema de Controle de Afastamento de Professores

Versão	Responsáveis	Data	Alterações
0.1	Rodolfo Costa, Rafael dos Anjos	15/09/2014	Versão parcial inicial
1.0	Lucas R. M. Silva	02/03/2021	Revisão geral

1. Introdução

Este documento apresenta os requisitos de usuário do sistema SCAP – Sistema de Controle de Afastamento de Professores e está organizado da seguinte forma: a Seção 2 contém uma descrição do propósito do sistema; a Seção 3 apresenta uma descrição do minimundo apresentando o problema; e a Seção 4 apresenta a lista de requisitos de usuário levantados junto aos membros do Departamento de Informática.

2. Descrição do Propósito do Sistema

O SCAP tem como propósito ser um sistema que visa apoiar o controle de afastamentos de professores do Departamento de Informática da UFES, buscando simplificar o processo de submissão, análise e armazenamento dos pedidos de afastamento.

3. Descrição do Minimundo

No Departamento de Informática (DI) da UFES, professores podem solicitar pedidos de afastamento referentes a eventos no Brasil ou no exterior. Mediante processos burocráticos e análises dos membros do departamento, estes podem ou não serem aprovados.

Os secretários são responsáveis pela parte administrativa do sistema, cabendo a eles as funções de cadastrar usuários e mandatos dos chefes de departamento. Também são responsáveis pelo cadastramento dos pareceres de fora do DI e também pelo arquivamento de pedidos já finalizados.

Ao cadastrar um usuário, os secretários devem informar: tipo de usuário, nome, matrícula, e-mail, telefone e parentescos. Já para cadastrar um chefe de departamento, um mandato temporário deve ser criado e atribuído a um professor já cadastrado no sistema. De um mandato devem ser informados: tipo de mandato, professor, data de início e data de fim.

Professores podem submeter pedidos de afastamento. No ato da submissão devem ser informados: data de início do afastamento, data de fim do afastamento, tipo de afastamento, motivo do afastamento, tipo de ônus, data de início do evento, data de fim do evento, nome do evento e os documentos necessários.

Para um evento no Brasil, o pedido deve ser avaliado unicamente pelos professores do DI por meio da isenção de manifestações contrárias ao pedido de afastamento, isto é, caso nenhum professor se oponha, este será aceito. Havendo alguma oposição, será feita uma reunião entre os professores para decidir se o pedido deve ser aceito ou não, cabendo aos secretários atualizarem o status do pedido de acordo com a decisão alcançada durante a reunião.

Para um evento no exterior, o pedido deve primeiro passar pelo chefe de departamento do DI, este, por sua vez, irá selecionar um professor do DI para ser o relator responsável pelo pedido de afastamento, desde que o escolhido não seja o próprio solicitante, nem um parente daquele que fez o pedido e nem o próprio chefe de departamento. Uma vez que o relator tenha submetido um parecer favorável e nenhum membro do departamento se manifeste contra o parecer, os secretários encaminharão o pedido para o Centro Tecnológico (CT) e posteriormente para a Pró-reitoria de Pesquisa e Pós-Graduação (PRPPG), registrando seus pareceres. Se totalmente aprovado, o afastamento será publicado no Diário Oficial da União. Caso haja rejeição por qualquer uma das entidades responsáveis pela avaliação do pedido, este será rejeitado.

Para todos os processos que ocorrem durante o ciclo de vida de um pedido de afastamento, os envolvidos do DI serão notificados através de e-mails automáticos enviados pelo sistema. Não cabe ao SCAP se responsabilizar pelas tramitações do CT e da PRPPG, ou seja, o sistema não se responsabiliza pelos processos que ocorrem fora do DI, sendo assim, os pareceres externos devem ser registrados pelos próprios secretários.

4. Requisitos de Usuário

Tomando por base o contexto do sistema, foram identificados os seguintes requisitos de usuário:

4.1 Requisitos Funcionais

Identificador	Descrição	Prioridade	Depende de
RF01	O sistema deve gerenciar informações de usuários.	Alta	-
RF02	O sistema deve gerenciar informações de pedidos de afastamento.	Alta	RF01
RF03	O sistema deve gerenciar informações de documentos.	Alta	RF01, RF02
RF04	O sistema deve gerenciar informações de pareceres.	Alta	RF01, RF02
RF05	O sistema deve gerenciar informações de parentescos.	Alta	RF01
RF06	O sistema deve gerenciar informações de mandatos.	Alta	RF01
RF07	O sistema deve permitir que usuários gerenciem suas próprias informações pessoais e credenciais.	Alta	RF01
RF08	O sistema deve permitir que secretários gerenciem informações de usuários.	Alta	RF01
RF09	O sistema deve permitir que secretários gerenciem informações de mandatos de chefe de departamento.	Alta	RF01, RF06
RF10	O sistema deve permitir que professores solicitem afastamento.	Alta	RF01, RF02, RF03
RF11	O sistema deve permitir que usuários consultem afastamentos.	Alta	RF01, RF02
RF12	O sistema deve permitir que professores cancelem seus pedidos de afastamento em andamento.	Alta	RF01, RF02, RF11
RF13	O sistema deve permitir que usuários gerenciem informações de documentos.	Alta	RF01, RF02, RF03, RF11
RF14	O sistema deve permitir que professores se manifestem contra afastamentos, bloqueando o pedido.	Alta	RF01, RF02, RF04, RF11
RF15	O sistema deve permitir que o chefe de departamento corrente selecione um professor para ser relator de um pedido de afastamento internacional.	Alta	RF01, RF02, RF05, RF06, RF11
RF16	O sistema deve permitir que relatores emitam um parecer para cada afastamento o qual ele foi indicado.	Alta	RF01, RF02, RF04, RF11
RF17	O sistema deve permitir que usuários consultem pareceres.	Alta	RF01, RF02, RF04, RF11
RF18	O sistema deve permitir que secretários arquivem pedidos de afastamento quando concluídos.	Alta	RF01, RF02, RF11
RF19	O sistema deve permitir que secretários registrem pareceres do CT	Alta	RF01, RF02,

	em seus respectivos afastamentos.		RF03, RF04, RF11
RF20	O sistema deve permitir que secretários registrem pareceres da PRPPG em seus respectivos afastamentos.	Alta	RF01, RF02, RF03, RF04, RF11
RF21	O sistema deve permitir que secretários desbloqueiem pedidos de afastamento que foram bloqueados devido à manifestação contrária de um colega.	Alta	RF01, RF02, RF11
RF22	O sistema deve agendar tarefas para aprovar pedidos de afastamento nacionais que não receberam manifestações contrárias.	Alta	RF01, RF02, RF04
RF23	O sistema deve ser capaz de enviar notificações automáticas via e-mail para membros do DI.	Alta	RF01, RF02, RF03, RF04
RF24	O sistema deve gerar um documento (ata) sobre um pedido de afastamento quando aprovado pelo DI.	Alta	RF01, RF02, RF03, RF04
RF25	O sistema deve agendar tarefas para arquivar ou cancelar pedidos de afastamento assim que atingirem a data de fim.	Alta	RF02

4.2 Requisitos Negativos

Identificador	Descrição	Prioridade	Depende de
RNeg01	O sistema não deve se responsabilizar pela tramitação fora do DI.	Alta	-
RNeg02	O sistema não deve se responsabilizar pela ausência mútua do chefe e do subchefe de departamento.	Alta	-
RNeg03	O sistema não deve se responsabilizar por ações antiéticas vindas dos usuários.	Alta	-

4.3 Requisitos Não Funcionais

Identificador	Descrição	Categoria	Escopo	Prioridade	Depende de
RNF01	O sistema deve prover controle de acesso às funcionalidades.	Segurança de Acesso	Sistema	Alta	-
RNF02	O sistema deve ser de fácil aprendizagem e operação, não requerendo treinamento.	Usabilidade	Sistema	Alta	-
RNF03	A persistência das informações deve ser implementada usando o SGBD (Sistema Gerenciador de Bancos de Dados) estabelecido na plataforma de implementação do Projeto. Contudo, o sistema deve ser projetado para, no futuro, ser possível utilizar outra tecnologia de bancos de dados.	Portabilidade	Sistema	Alta	-
RNF04	O sistema deve estar disponível como uma aplicação Web, acessível a partir dos principais navegadores disponíveis no mercado.	Portabilidade	Sistema	Alta	-
RNF05	O sistema deve ser extensível e de fácil manutenção.	Manutenibilidade	Sistema	Alta	-
RNF06	O desenvolvimento do sistema deve explorar o potencial de reutilização de componentes.	Reusabilidade	Sistema	Média	-
RNF07	O sistema deve ser desenvolvido utilizando a plataforma ASP.NET Core MVC. Seguindo o método FrameWeb, com auxílio de suas ferramentas.	Tecnologia	Sistema	Alta	-

Documento de Especificação de Requisitos

Projeto: SCAP – Sistema de Controle de Afastamento de Professores

1. Introdução

Este documento apresenta o resultado da análise dos requisitos do sistema SCAP. A atividade de análise de requisitos foi conduzida aplicando-se técnicas de modelagem de casos de uso, modelagem de classes e modelagem de comportamento dinâmico do sistema. Os modelos apresentados foram elaborados usando a UML.

Este documento está organizado da seguinte forma: a Seção 2 apresenta os subsistemas identificados, mostrando suas dependências na forma de um diagrama de pacotes; a Seção 3 apresenta o modelo de casos de uso, incluindo descrições de atores, os diagramas de casos de uso e descrições de casos de uso; a Seção 4 apresenta o modelo conceitual estrutural do sistema, na forma de diagramas de classes; a Seção 5 apresenta o modelo comportamental dinâmico do sistema, na forma de diagramas de estado; finalmente, a Seção 6 apresenta o glossário do projeto, contendo as definições das classes identificadas.

2. Identificação de Subsistemas

A **Figura 1** mostra os subsistemas identificados no contexto do presente projeto, os quais são descritos na tabela abaixo.

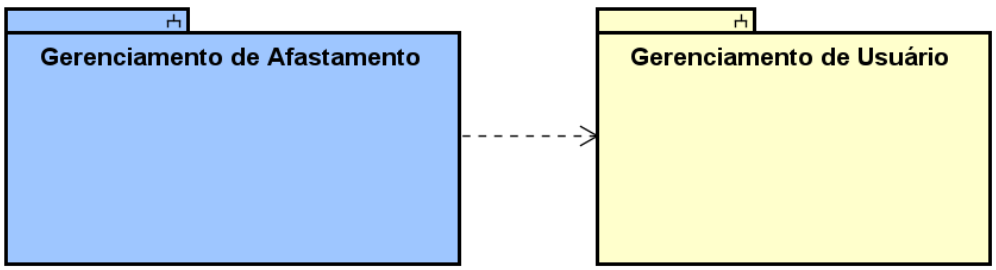

Figura 1 – Diagrama dos Subsistemas Identificados.

Tabela 1 – Subsistemas.

Subsistema	Descrição
Gerenciamento de Afastamento	Subsistema responsável pelas operações sobre pedidos de afastamento.
Gerenciamento de Usuário	Subsistema responsável pelo gerenciamento e autenticação de usuários do sistema e gerenciamento de mandatos.

3. Modelo de Casos de Uso

O modelo de casos de uso visa capturar e descrever as funcionalidades que um sistema deve prover para os atores que interagem com o mesmo. Os atores identificados no contexto deste projeto estão descritos na tabela abaixo.

Tabela 2 – Atores.

Ator	Descrição
Usuário	Usuário credenciado no SCAP.
Professor	Professor efetivo do DI/UFES.
Chefe do Departamento	Professor do DI/UFES que está realizando a função administrativa de chefe/subchefe do departamento.
Secretário	Secretário do DI/UFES.
Scheduler	Processo responsável por disparar tarefas agendadas.

A seguir, são apresentados os diagramas de casos de uso e descrições associadas, organizados por subsistema.

3.1 - Subsistema Gerenciamento de Usuário

A **Figura 2** apresenta o diagrama de casos de uso do subsistema Gerenciamento de Usuário.

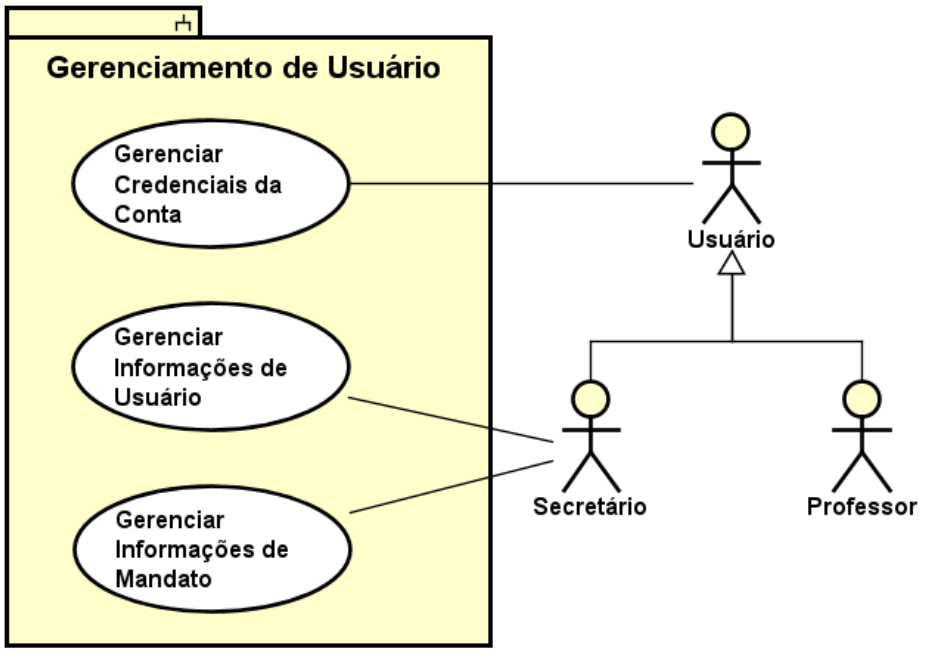

Figura 2 – Diagrama de Casos de Uso do Subsistema Gerenciamento de Usuário.

No contexto deste subsistema, as regras de negócio apresentadas na **Tabela 3** devem ser observadas.

Tabela 3 - Regras de Negócio do Subsistema Gerenciamento de Usuário.

Identificador	Descrição
RN01	A matrícula e tipo de usuário não podem ser alterados.
RN02	Caso o chefe esteja inativo, o subchefe deve assumir o posto de chefe.

A seguir, são apresentadas as descrições de cada um dos casos de uso identificados. Os casos de uso cadastrais de baixa complexidade, envolvendo inclusão, alteração, consulta e exclusão, são descritos na tabela abaixo, segundo o padrão da organização.

Tabela 4 – Casos de Uso Cadastrais do Subsistema Gerenciamento de Usuário.

Identificador	Caso de Uso	Ações Possíveis	Observações	Requisitos	Classes
CSU01	**Gerenciar Informações de Usuário**	I, A, C, E	Este caso de uso permite que o secretário edite informações de um usuário. [I] Informar: nome, sobrenome, e-mail, telefone, tipo de usuário e parentescos. Será enviado um link via e-mail para o usuário criar uma senha. [A] Não é permitido alterar matrícula, senha e tipo de usuário. Um usuário pode ser desativado por estar impossibilitado de exercer sua função ou não ser mais um membro do departamento. [E] Não é permitido excluir um professor que tenha afastamentos, mandatos ou pareceres associados.	RF01, RF05, RF08	Secretario, Professor, Parentesco
CSU02	**Gerenciar Credenciais da Conta**	A, C	Este caso de uso permite que o usuário configure os dados de sua própria conta. [A] Não é permitido alterar matrícula e tipo de usuário.	RF01, RF07	Secretario, Professor
CSU03	**Gerenciar Informações de Mandato**	I, C, E	[I] Informar: professor eleito, tipo de mandato, data de início e data de fim do mandato. [E] Só é permitido excluir um mandato que ainda não foi iniciado. Aqueles que já tenham começado serão interrompidos, mas não excluídos.	RF01,RF06, RF08,RF09	Professor, Mandato

3.2 - Subsistema Gerenciamento de Afastamento

A **Figura 3** apresenta o diagrama de casos de uso do subsistema Gerenciamento de Afastamento.

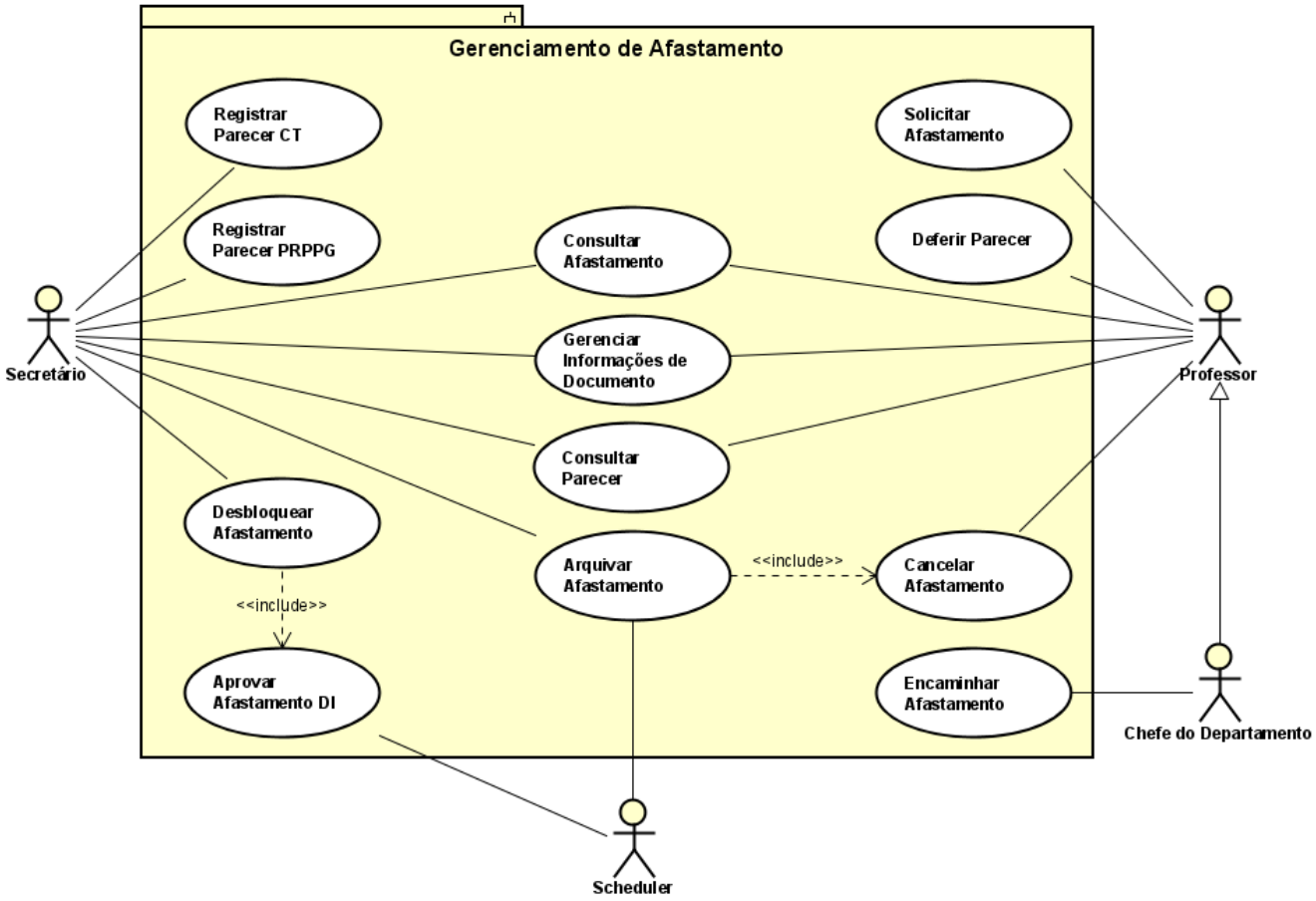

Figura 3 – Diagrama de Casos de Uso do Subsistema Gerenciamento de Afastamento.

No contexto deste subsistema, as regras de negócio apresentadas na **Tabela 5** devem ser observadas.

Tabela 5 - Regras de Negócio do Subsistema Gerenciamento de Afastamento.

Identificador	Descrição
RN03	Um professor não pode ser escolhido como relator da própria solicitação de afastamento.
RN04	Um professor não pode ser relator da solicitação de afastamento de algum parente.
RN05	Uma solicitação de afastamento só pode ser cancelada pelo professor que a criou.

A seguir, são apresentadas as descrições de cada um dos casos de uso identificados. Os casos de uso cadastrais de baixa complexidade, envolvendo inclusão, alteração, consulta e exclusão, são descritos na tabela abaixo, segundo o padrão da organização.

Tabela 6 – Casos de Uso Cadastrais do Subsistema Gerenciamento de Afastamento.

Identificador	Caso de Uso	Ações Possíveis	Observações	Requisitos	Classes
CSU07	**Encaminhar Afastamento**	A	[A] Chefe do departamento seleciona um relator responsável pelo pedido de afastamento internacional com status "Iniciado", não sendo o solicitante, um parente ou o próprio chefe de departamento. Um e-mail automático é enviado ao professor escolhido como relator, requerendo o parecer.	RF01,RF02, RF05,RF11, RF15,RF23	Professor, Mandato, Afastamento, Parentesco
CSU08	**Consultar Afastamento**	C	[C] Qualquer usuário pode consultar um afastamento específico através da lista de afastamentos, do código de um afastamento ou por um link enviado por e-mail.	RF02,RF11	Afastamento
CSU09	**Gerenciar Informações de Documento**	I, C, E	[I] Apenas Secretário e Professor solicitante podem adicionar documentos ao pedido. [C] Apenas Secretário e Professor solicitante podem consultar os documentos anexados no pedido. [E] Apenas Secretário e Professor solicitante podem excluir um documento que foi anexado no pedido.	RF02,RF03, RF11,RF13	Afastamento, Documento
CSU10	**Consultar Parecer**	C	[C] Todos os membros do DI podem ver os pareceres de um pedido de afastamento.	RF01,RF02, RF04,RF17	Professor, Afastamento, Parecer

Descrição de Caso de Uso

Subsistema: Gerenciamento de Afastamento
Caso de Uso: Solicitar Afastamento
Identificador do Caso de Uso: CSU04

Descrição Sucinta: Neste Caso de Uso um **Professor** do DI faz um pedido de afastamento no sistema.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Solicitar Afastamento		1. O professor preenche o formulário de solicitação de afastamento com as informações de: data de início do afastamento, data de fim do afastamento, tipo de afastamento, motivo do afastamento, tipo de ônus, data de início do evento, data de fim do evento e nome do evento. 2. O pedido de afastamento é cadastrado com situação "Iniciado" e o professor é redirecionado para a página do seu pedido. 3. O professor faz upload dos documentos necessários para se afastar e confirma que terminou a ação.

Fluxos de Eventos Variantes

Nome do Fluxo de Eventos Normal Relacionado	Variante	Descrição
Solicitar Afastamento	3 - O afastamento é nacional.	1. O pedido de afastamento fica passível de aprovação pelo Scheduler em nome do DI. 2. Um e-mail é enviado para todos os professores avisando que um novo pedido de afastamento nacional foi solicitado e está passível de manifestações.
Solicitar Afastamento	3 - O afastamento é internacional.	1. Um e-mail é enviado ao chefe do departamento avisando que um relator deve ser escolhido.

Requisitos Relacionados: RF01, RF02, RF03, RF10, RF23.
Classes Relacionadas: Professor, Mandato, Afastamento, Documento.

Descrição de Caso de Uso

Subsistema: Gerenciamento de Afastamento
Caso de Uso: Cancelar Afastamento
Identificador do Caso de Uso: CSU05

Descrição Sucinta: Neste Caso de Uso um **Professor** do DI cancela um pedido de afastamento solicitado por ele próprio.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Cancelar Afastamento	1. O afastamento deve estar em situação diferente de "Arquivado" e "Reprovado".	1. A situação do pedido é alterada para "Cancelado". 2. Um e-mail é enviado para os secretários, informando sobre o cancelamento do pedido.

Descrição Sucinta: Neste Caso de Uso, o **Scheduler** cancela um pedido de afastamento que se encontra em situação inconsistente na data de fim do afastamento.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Cancelar Afastamento	1. A data corrente ultrapassou a data de fim do afastamento. 2. O afastamento se encontra em uma situação diferente de "AprovadoDI", "AprovadoPRPPG", "Reprovado", "Cancelado", "Arquivado".	1. A situação do pedido é alterada para "Cancelado". 2. O Caso de Uso é finalizado.

Requisitos Relacionados: RF01, RF02, RF12, RF25.
Classes Relacionadas: Professor, Secretario, Afastamento.

Descrição de Caso de Uso

Subsistema: Gerenciamento de Afastamento
Caso de Uso: Deferir Parecer
Identificador do Caso de Uso: CSU06

Descrição Sucinta: Neste Caso de Uso, um **Professor** do DI cadastra seu parecer destinado a um pedido de afastamento ou parecer de um relator.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Deferir Parecer	1. O afastamento deve estar com status "Liberado".	1. O professor preenche e submete um formulário com seu parecer. 2. O parecer é cadastrado pelo sistema.

Fluxos de Eventos Variantes

Nome do Fluxo de Eventos Normal Relacionado	Variante	Descrição
Deferir Parecer	1 - O afastamento é internacional, o professor não é relator e o relator ainda não deferiu seu parecer.	1. O professor será alertado de que precisa esperar o parecer do relator para poder se manifestar. 2. O Caso de Uso é finalizado e nenhuma informação é persistida.
Deferir Parecer	2 - O afastamento é internacional e o professor não é relator.	1. O parecer será interpretado como contrário ao parecer do relator, alterando a situação do pedido de afastamento para "Bloqueado" até que uma decisão seja tomada na reunião do departamento. 2. Um e-mail é enviado ao solicitante do afastamento, informando a alteração da situação do pedido.
Deferir Parecer	2 - O afastamento é nacional e o parecer é favorável.	1. O pedido de afastamento se mantém passível de aprovação pelo Scheduler em nome do DI.
Deferir Parecer	2 - O afastamento é internacional, o professor é relator e o parecer é favorável.	1. O pedido de afastamento fica passível de aprovação pelo Scheduler em nome do DI. 2. Caso o responsável pelo parecer seja o relator, é liberada a permissão para os demais professores se manifestarem contra o parecer do relator, os notificando sobre a liberação por e-mail.
Deferir Parecer	2 - O parecer é desfavorável.	1. A situação do pedido é alterada para "Bloqueado" até que uma decisão seja tomada na reunião do departamento. 2. Um e-mail é enviado ao solicitante do afastamento, informando a alteração da situação do pedido.

Requisitos Relacionados: RF01, RF02, RF03, RF04, RF16, RF23, RF24.
Classes Relacionadas: Professor, Afastamento, Parecer.

Descrição de Caso de Uso

Subsistema: Gerenciamento de Afastamento
Caso de Uso: Arquivar Afastamento
Identificador do Caso de Uso: CSU11

Descrição Sucinta: Neste Caso de Uso, o **Secretário** arquiva um afastamento que foi finalizado.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Arquivar Afastamento	1. O afastamento deve ter alcançado sua data de fim. 2. O afastamento deve estar em situação igual a "AprovadoDI" ou "AprovadoPRPPG".	1. Uma vez finalizado o afastamento, o secretário altera a situação do pedido para "Arquivado"

Descrição Sucinta: Neste Caso de Uso, o **Scheduler** arquiva um afastamento que foi finalizado.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Arquivar Afastamento	1. O afastamento deve ter alcançado sua data de fim.	1. Um dia após a data de fim do afastamento, o Scheduler altera a situação do pedido para "Arquivado".

Fluxos de Eventos Variantes

Nome do Fluxo de Eventos Normal Relacionado	Variante	Descrição
Arquivar Afastamento	1 - O afastamento se encontra em uma situação diferente de "AprovadoDI", "AprovadoPRPPG", "Reprovado", "Cancelado", "Arquivado".	1. O Scheduler irá entender que o pedido se encontra em uma situação inconsistente. 2. É incluído aqui o caso de uso **CSU05** – Cancelar Afastamento.

Requisitos Relacionados: RF02, RF18, RF25.
Classes Relacionadas: Afastamento.

Descrição de Caso de Uso

Subsistema: Gerenciamento de Afastamento
Caso de Uso: Registrar Parecer CT
Identificador do Caso de Uso: CSU12

Descrição Sucinta: Neste Caso de Uso, o **Secretário** cadastra o parecer do CT no sistema.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Registrar Parecer CT	1. Afastamento deve atender um evento internacional. 2. Afastamento deve estar com status "AprovadoDI".	1. O secretário altera a situação do afastamento de acordo com o parecer do CT. 2. O secretário faz upload do documento enviado pelo CT contendo o parecer. 3. Um e-mail é enviado ao solicitante do afastamento, informando a alteração da situação do pedido.

Fluxos de Eventos Variantes

Nome do Fluxo de Eventos Normal Relacionado	Variante	Descrição
Registrar Parecer CT	1 - Parecer positivo.	1. A situação do afastamento é alterada para "AprovadoCT".
Registrar Parecer CT	1 - Parecer negativo.	1. A situação do afastamento é alterada para "Reprovado".

Requisitos Relacionados: RF01, RF02, RF03, RF04, RF13, RF19, RF23.
Classes Relacionadas: Professor, Secretário, Afastamento, Parecer, Documento.

Descrição de Caso de Uso

Subsistema: Gerenciamento de Afastamento
Caso de Uso: Registrar Parecer PRPPG
Identificador do Caso de Uso: CSU13

Descrição Sucinta: Neste Caso de Uso, o **Secretário** cadastra o parecer da PRPPG no sistema.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Registrar Parecer PRPPG	1. Afastamento deve atender um evento internacional. 2. Afastamento deve estar com status "AprovadoCT".	1. O secretário altera a situação do afastamento de acordo com o parecer da PRPPG. 2. O secretário faz upload do documento enviado pela PRPPG contendo o parecer. 3. Um e-mail é enviado ao solicitante do afastamento, informando a alteração da situação do pedido.

Fluxos de Eventos Variantes

Nome do Fluxo de Eventos Normal Relacionado	Variante	Descrição
Registrar Parecer PRPPG	1 - Parecer positivo.	1. A situação do afastamento é alterada para "AprovadoPRPPG".
Registrar Parecer PRPPG	1 - Parecer negativo.	1. A situação do afastamento é alterada para "Reprovado".

Requisitos Relacionados: RF01, RF02, RF03, RF04, RF13, RF20, RF23.
Classes Relacionadas: Professor, Secretário, Afastamento, Parecer, Documento.

Descrição de Caso de Uso

Subsistema: Gerenciamento de Afastamento
Caso de Uso: Desbloquear Afastamento
Identificador do Caso de Uso: CSU14

Descrição Sucinta: Neste Caso de Uso, o **Secretário** desbloqueia um pedido de afastamento que foi contrariado por algum membro do DI.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Desbloquear Afastamento	1. Afastamento deve estar com situação “Bloqueado".	1. Após a reunião do DI, o secretário altera o estado do pedido de afastamento. 2. Um e-mail é enviado ao solicitante do afastamento, informando a alteração da situação do pedido.

Fluxos de Eventos Variantes

Nome do Fluxo de Eventos Normal Relacionado	Variante	Descrição
Desbloquear Afastamento	1 - O pedido não foi aprovado pelos membros do DI.	1. A situação do pedido é alterada para “Reprovado”.
Desbloquear Afastamento	1 - O pedido foi aprovado pelos membros do DI.	1. É incluído aqui o caso de uso **CSU15** – Aprovar Afastamento DI.

Requisitos Relacionados: RF01, RF02, RF03, RF04, RF11, RF21, RF23, RF24.
Classes Relacionadas: Professor, Afastamento, Parecer, Documento.

Descrição de Caso de Uso

Subsistema: Gerenciamento de Afastamento
Caso de Uso: Aprovar Afastamento DI
Identificador do Caso de Uso: CSU15

Descrição Sucinta: Neste Caso de Uso, o **Scheduler** consulta e aprova pedidos de afastamento nacionais e internacionais (caso já tenha recebido o parecer do relator) que não tenham recebido parecer negativos em até 10 dias após ser liberado.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Aprovar Afastamento DI	1. O afastamento deve estar com situação "Liberado". 2. O afastamento não tenha recebido pareceres desfavoráveis por 10 dias.	1. O scheduler verifica se nenhum professor do DI se manifestou contra o pedido corrente. 2. Scheduler altera a situação do pedido para "AprovadoDI" e a ata de aprovação pelo DI fica disponível para download. 3. Um e-mail é enviado ao professor solicitante informando sobre a aprovação da sua solicitação de afastamento, e outro para os secretários para que possam dar continuidade aos trâmites fora do DI.

Descrição Sucinta: Neste Caso de Uso, o **Secretário** aprova um pedido pelo DI após a decisão tomada na reunião do departamento.

Fluxos de Eventos Normais

Nome do Fluxo de Eventos Normal	Precondição	Descrição
Aprovar Afastamento DI	1. O afastamento deve estar com situação "Bloqueado".	1. A situação do pedido é alterada para "AprovadoDI". 2. Um e-mail é enviado ao professor solicitante informando sobre a aprovação da sua solicitação de afastamento, e outro para os secretários para que possam dar continuidade aos trâmites fora do DI.

Requisitos Relacionados: RF01, RF02, RF03, RF04, RF22, RF23, RF24.
Classes Relacionadas: Professor, Afastamento, Parecer, Documento.

4. Modelo Estrutural

O modelo conceitual estrutural visa capturar e descrever as informações (classes, associações e atributos) que o sistema deve representar para prover as funcionalidades descritas na seção anterior. A seguir, são apresentados o diagrama de classe do subsistema Gerenciamento de Afastamento, que contém todas as classes do domínio do sistema.

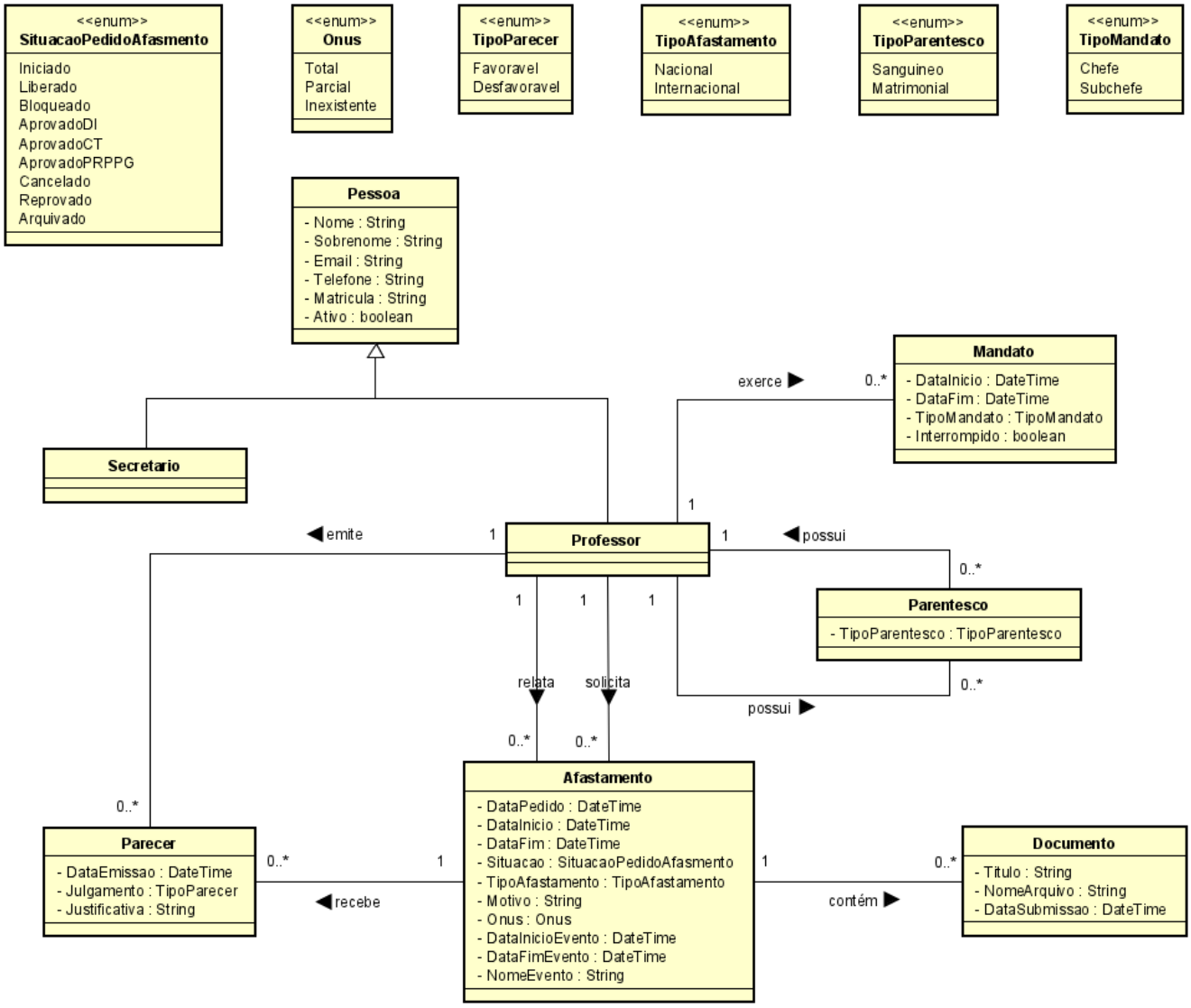

Figura 4 – Diagrama de Classes do Sistema.

- **Pessoa:** Qualquer professor ou Secretário usuário do sistema.
    - **Nome:** Primeiro Nome do usuário.
    - **Sobrenome:** Sobrenome do usuário.
    - **Email:** E-mail cadastrado do usuário.
    - **Telefone:** Telefone cadastrado do usuário.
    - **Matricula:** Matrícula do usuário na UFES.
    - **Ativo:** Indicativo se o usuário está impossibilitado de exercer sua função ou não é mais um membro do departamento.

- **Parentesco:** Representa um parentesco existente entre dois professores.
    - **TipoParentesco:** Tipo do parentesco entre dois professores.

- **Mandato:** Intervalo de tempo que um professor é Chefe ou Subchefe do Departamento.
    - **TipoMandato:** Tipo de mandato a ser cumprido (Chefe ou Subchefe).
    - **DataInicio:** Data de início do mandato.
    - **DataFim:** Data de fim do mandato.
    - **Interrompido:** Indicativo se o mandato foi finalizado fora do período planejado.

- **Afastamento:** Um pedido de afastamento realizado por um professor.
    - **DataPedido:** Dia que o pedido foi criado.
    - **DataInício:** Dia que o afastamento começa.
    - **DataFim:** Dia que o afastamento termina.
    - **Situacao:** Atual situação do pedido de afastamento, que varia entre: Iniciado, Liberado, Bloqueado, AprovadaDI, AprovadoCT, AprovadoPRPPG, Cancelado, Reprovado e Arquivado;
    - **TipoAfastamento:** Define se o afastamento é Nacional ou Internacional.
    - **Motivo:** Motivo para o qual o afastamento é necessário.
    - **Onus:** Tipo de ônus que o afastamento terá, podendo ser: Nenhum, Parcial ou Total.
    - **DataInicioEvento:** Data que o evento inicia.
    - **DataFimEvento:** Data que o evento termina.
    - **NomeEvento:** Nome do Evento.

- **Parecer:** Parecer de um professor sobre um Afastamento.
    - **DataEmissao:** Data que o parecer foi emitido.
    - **Julgamento:** Define se o parecer é positivo (Favorável) ou negativo (Desfavorável).
    - **Justificativa:** Define uma justificativa para o julgamento do parecer.

- **Documento:** Algum arquivo que é anexado a um documento.
    - **Titulo:** Título do documento anexado.
    - **NomeArquivo:** Nome do arquivo anexado.
    - **DataSubmissao:** Data que o arquivo foi anexado.

5. Modelo Dinâmico

O modelo dinâmico visa capturar o comportamento dinâmico do sistema. A seguir, são apresentados os diagramas de estados elaborados no contexto deste projeto.

As **Figuras 5** e **6** apresentam os diagramas de estados da classe Afastamento do subsistema Gerenciamento de Afastamento, para pedidos de afastamento nacionais e internacionais, respectivamente.

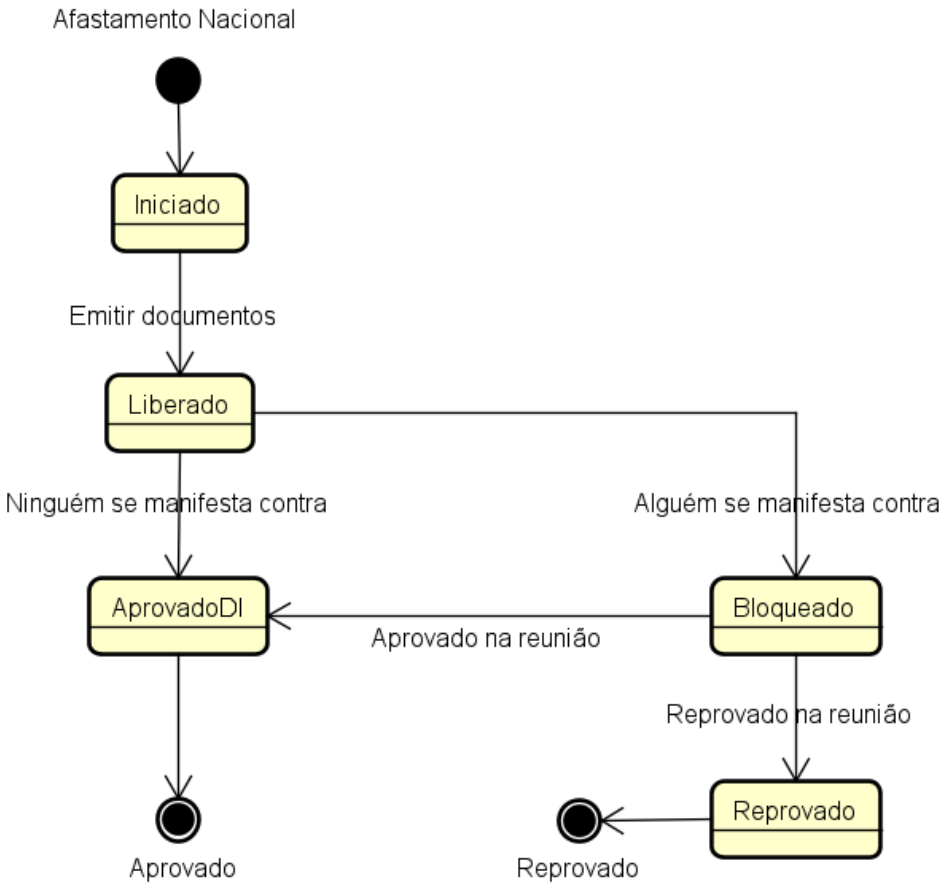

Figura 5 – Diagrama de Estados da Classe Afastamento (Nacional).

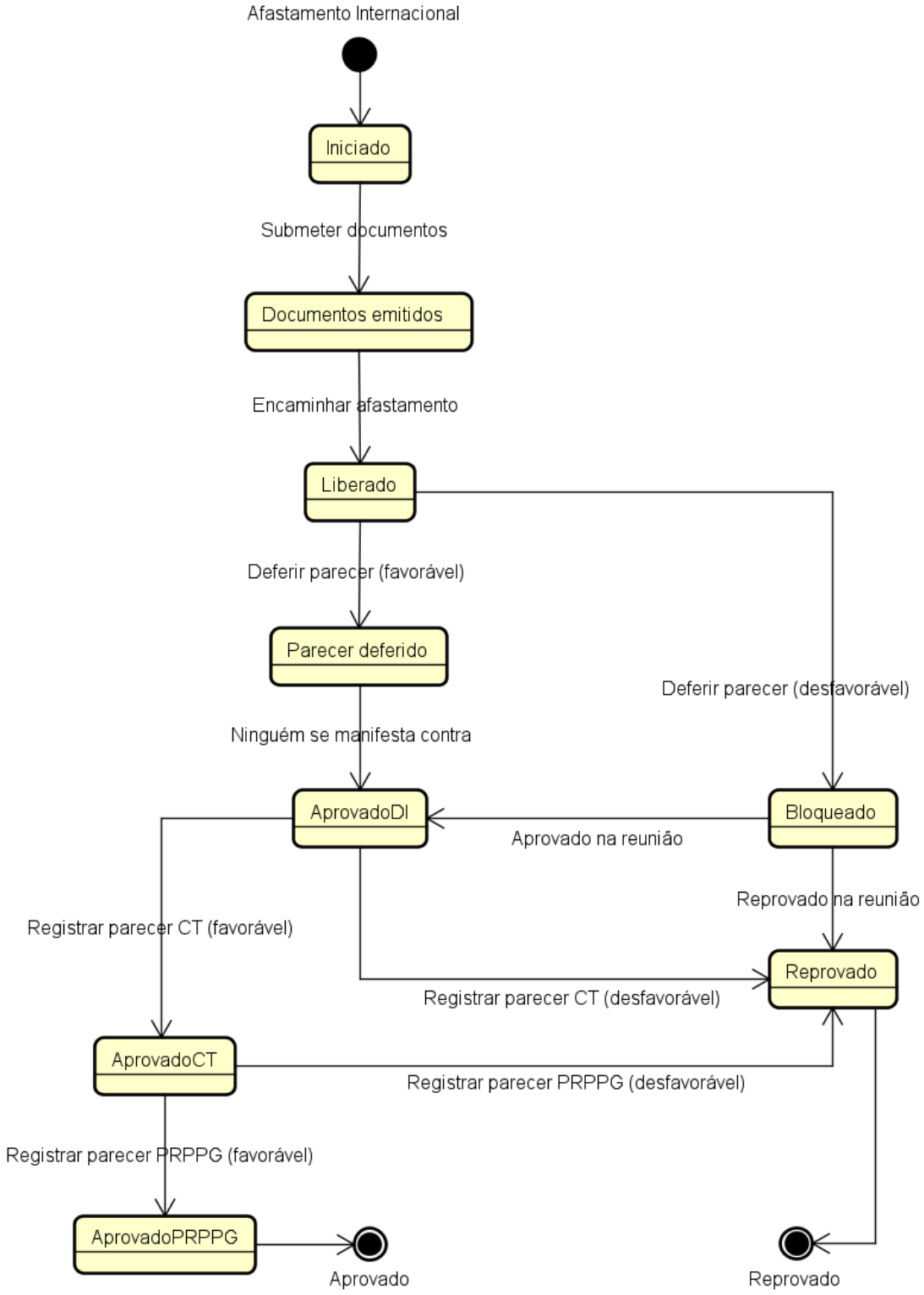

Figura 6 – Diagrama de Estados da Classe Afastamento (Internacional).