Leonardo Nascimento dos Santos

Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando o framework Grails

Vitória, ES

2021

Leonardo Nascimento dos Santos

Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando o framework Grails

Monografia apresentada ao Curso de Ciência da Computação do Departamento de Informática 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

Departamento de Informática

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2021

Leonardo Nascimento dos Santos
Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando o framework Grails/ Leonardo Nascimento dos Santos. – Vitória, ES, 2021-
76 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vítor E. Silva Souza

Monografia (PG) – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Departamento de Informática, 2021.

1. FrameWeb. 2. Grails. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando o framework Grails

CDU 02:141:005.7

Leonardo Nascimento dos Santos

Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando o framework Grails

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

Trabalho aprovado. Vitória, ES, 14 de maio de 2021:

Prof. Dr. Vítor E. Silva Souza
Orientador

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

Pedro Henrique Brunoro Hoppe
Universidade Federal do Espírito Santo

Vitória, ES
2021

Dedico a Deus e aos meus pais que são tudo em minha vida.

Agradecimentos

Agradeço a Deus por ser o primeiro em minha vida. Sem Deus eu não teria forças para superar todas as barreiras enfrentadas durante esses anos.

Aos meus pais, Neilson Nascimento dos Santos e Angela Maria Nascimento dos Santos, que me deram todo o amor incondicional que um filho poderia ter. Meus pais nunca deixaram que nada me faltasse e me deram total apoio nas decisões que tomei. Por esse motivo, espero conseguir retribuir da mesma maneira para que nada falte para eles também.

Também agradeço aos meus "irmãos", Vinícius Berger e Gustavo Costa Duarte, pois não é preciso ter irmãos de sangue para se construir uma amizade verdadeira. Eles foram peças fundamentais para que eu conseguisse chegar nesse momento tão importante.

Agradeço aos meus amigos, Igor de Moura Ventorim, Gustavo Epichin Monjardim e Eduardo Gorayeb Dalapicola, por sempre me ajudarem durante a minha graduação. Obrigado a todos os meus colegas e todas as pessoas boas que conheci na universidade.

Obrigado a todos os professores que contribuíram para a minha formação, em especial Rosane, Eduardo, Ricardo (In Memoriam), Davidson, Maria Cristina e principalmente o meu orientador Vítor, que me transmitiu um conhecimento inestimável.

"Buscai primeiro o Reino de Deus, e a sua justiça,
e as demais coisas vos serão acrescentadas."
(Jesus Cristo)

Resumo

No desenvolvimento das primeiras aplicações para *Web*, a preocupação em seguir a metodologia proposta pela Engenharia de Software não era muito relevante, fazendo com que estas aplicações fossem criadas de maneira *ad-hoc*. Com a evolução e o crescimento da Internet, a construção de *softwares* voltados para esse tipo de plataforma foi se tornando cada vez mais complexa, sendo necessário a adoção de técnicas que favorecessem a criação e adaptação das aplicações. Para esse fim, foi criada a Engenharia *Web* (*Web Engineering* ou *WebE*).

Com o surgimento da proposta do método FrameWeb (*Framework-based Design Method for Web Engineering*), diversas vantagens puderam ser aproveitadas durante a utilização de *frameworks* na fase de projetos, fazendo com que o desenvolvimento de Sistemas de Informação baseados na Web (*Web-based Information System – WISs*) se tornasse mais eficiente, proporcionando maior agilidade aos desenvolvedores.

Na criação de aplicações para *Web*, existem inúmeras categorias de *frameworks* que podem ser escolhidas para a execução das implementações dos projetos. O método FrameWeb se propõe a dar suporte a quatro dessas categorias (Controlador Frontal, Injeção de Dependências, Mapeamento Objeto/Relacional e Segurança) e, por esse motivo, existe a necessidade de realizar a verificação da eficácia deste suporte, testando *frameworks* que fazem parte dessas categorias e verificando os resultados da aplicação do método.

A proposta do método vem sendo analisada por meio da aplicação *Web* SCAP (Sistema de Controle de Afastamento de Professores) e já foram utilizado alguns *frameworks* para esse fim. Este trabalho tem como objetivo testar a eficácia do método FrameWeb, realizando uma nova implementação do SCAP, utilizando o *framework* Grails, dentro da categoria de Controlador Frontal.

Palavras-chaves: FrameWeb. Web. Engenharia Web. Grails.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UML	Linguagem de Modelagem Unificada, do inglês *Unified Modeling Language*
SCAP	Sistema de Controle de Afastamento de Professores
WIS	*Web-based Information Systems*
WWW	*World Wide Web*
ABNT	Associação Brasileira de Normas Técnicas
DI	Departamento de Informática
CT	Centro Tecnológico
PRPPG	Pró-Reitoria de Pesquisa e Pós-Graduação
DAO	*Data Access Object*
URL	*Uniform Resource Locator*
GORM	*Grails Object Relational Mapping*
API	*Application Programming Interface*
SGBD	Sistema Gerenciador de Banco de Dados
CoC	*Convention Over Configuration*
GSP	*Groovy Server Pages*
CASE	*Computer-Aided Software Engineering*

Sumário

1 Introdução

Antigamente, no início da *World Wide Web* (WWW), os sites eram constituídos por diversos arquivos de hipertexto ligados que apresentavam as informações por meio da utilização de gráficos limitados e de textos. Com o passar do tempo e com o surgimento de ferramentas de desenvolvimento, os engenheiros da Internet puderam oferecer mais informações por meio da capacidade computacional. Nascem, assim, os sistemas e aplicações baseadas na *Web (WebApp)*. Nos dias atuais, as *WebApps* se tornaram ferramentas computacionais sofisticadas que foram integradas às aplicações de negócio e aos bancos de dados corporativos, oferecendo funções especializadas aos usuários finais (PRESSMAN, 2011).

Com a criação de diversas *WebApps* diferentes, é importante ressaltar que uma aplicação *Web* pode se enquadrar em mais de uma categoria, assim como realizar uma troca de categoria ao longo do seu tempo de vida (BEDER, 2012). Desta maneira, o foco deste trabalho serão os Sistemas de Informação Baseados na *Web (Web-based Information Systems – WISs)*, que são enquadrados em uma categoria específica de *WebApps*. Esses sistemas são caracterizados como sistemas de informação tradicionais, mas estão disponíveis na Internet.

Quando falamos sobre o desenvolvimento de *WebApps* atuais, em particular os WISs, temos que entender que utilizar a Engenharia de Software é uma tarefa fundamental. Os aspectos relacionados ao estabelecimento de técnicas, processos, métodos, ferramentas e ambientes de suporte ao desenvolvimento de software são tratados de forma clara pela Engenharia de Software (FALBO, 2014). Trazendo alguns benefícios como, por exemplo, a compatibilidade entre plataformas e a facilidade de gerenciamento, as *WebApps*, por meio dos servidores, se tornaram indispensáveis para manterem os serviços e as aplicações disponíveis em qualquer parte do mundo através da Internet.

No meio de tantas criações e adaptações tecnológicas, surge o uso de *frameworks* para *WebApps*. Se tornando uma das ferramentas mais importantes para o desenvolvimento de *WebApps*, os *frameworks* passaram a auxiliar no encapsulamento das funcionalidades de alto nível com maior eficiência e agilidade, fazendo com que a maior parte do tempo e do trabalho fossem economizados. Visando propor uma abordagem diferenciada para a construção de sistemas para *Web*, surge então o método FrameWeb (*Framework-based Design Method for Web Engineering*) (SOUZA, 2007; SOUZA, 2020).

Baseado na linguagem de modelagem UML (*Unifield Modeling Language*) (BOOCH; RUMBAUGH; JACOBSON, 2006), o método FrameWeb propõe quatro diagramas para a fase de projeto de software que incorporam os conceitos trazidos pelos *frameworks* utilizados

no desenvolvimento Web, de modo a facilitar a comunicação entre desenvolvedores, agilizar o desenvolvimento por meio de geração de código, dentre outras vantagens. Originalmente, o método se propunha a dar suporte a três categorias de *frameworks*: controladores frontais, injeção de dependências e mapeamento objeto/relacional.

Existem, no entanto, inúmeros *frameworks* existentes dentro de cada categoria. Sendo assim, se faz necessário experimentar o método com *frameworks* diversos, avaliando sua adequação. Neste contexto, Duarte (2014) desenvolveu uma *WebApp* — um Sistema de Controle de Afastamento de Professores (SCAP) — em seu trabalho de conclusão de curso. Esse sistema foi criado para apoiar um departamento de universidade a realizar um controle das solicitações de afastamento de seus professores efetivos. Utilizando os requisitos que foram levantados por Duarte (2014) e posteriormente analisados por Prado (2015), este trabalho tem com tarefa fundamental a implementação do SCAP utilizando outro *framework Web*, para que seja possível realizar a verificação da eficiência do método FrameWeb.

O *framework* experimentado neste trabalho será o Grails. O *framework* Grails utiliza as linguagens de programação Java e Groovy, tendo como característica a minimização da complexidade da criação de *WebApps* e podendo ser integrado com qualquer biblioteca Java por meio de *plugins*.

1.1 Justificativa e Motivação

Desenvolver um sistema com o método FrameWeb, representa uma grande busca entre as diversas fontes de conhecimento envolvendo o desenvolvimento *Web* e as inúmeras áreas dentro da computação. O conhecimento adquirido entre essas áreas, quando somado, estabelece um processo de aprendizagem que agrega um valor inestimável durante as fases do desenvolvimento do projeto.

O projeto também contribui no aprimoramento e no desenvolvimento do método FrameWeb, uma vez que ao experimentá-lo utilizando um novo tipo de *framework*, possibilita realizar a atualização da lista de suas aplicações. Ainda neste contexto, este projeto é responsável por auxiliar o Departamento de Informática, administrando o processo de afastamento de professores, que por sua vez, pode se tornar complexo sem um meio de automatização.

Agilizar a execução deste processo possibilita que tanto os secretários quanto os professores do Departamento de Informática possam controlar as requisições e avaliações de afastamentos, diminuindo o tempo entre as etapas e anulando possíveis erros.

1.2 Objetivos

Este trabalho tem como objetivo geral a aplicação do método FrameWeb (SOUZA, 2007; SOUZA, 2020), realizando uma nova implementação do SCAP, utilizando o *framework* Grails. Uma vez que os requisitos foram levantados por Duarte (2014) e reformulados por Prado (2015), é possível verificar como o método se comporta, apresentando a devida evolução.

O objetivo geral pode ser subdividido nos seguintes objetivos específicos:

- Utilizar o *framework* Grails para realizar uma nova implementação do SCAP, aproveitando os seus requisitos. Nesse objetivo será possível aplicar os conceitos de Engenharia de Requisitos, Projeto de Sistemas de Software e Engenharia de Software;
- Aplicar o método FrameWeb para definir a documentação da arquitetura do projeto de sistema para o *framework*;
- Realizar o estudo dos requisitos já levantados.

1.3 Método de Trabalho

De acordo com a seção anterior, para que os objetivos apresentados fossem alcançados, foi necessário realizar os seguintes passos:

1. Revisão bibliográfica e pesquisa: leitura dos padrões de Projeto de Sistemas de Software (FALBO, 2018), análise dos temas de Engenharia de Software (FALBO, 2014), Engenharia de Requisitos (FALBO, 2017) e entendimento do método FrameWeb (SOUZA, 2007) para realizar a aplicação do *framework* Grails;
2. Estudo do sistema SCAP: organização das informações referentes aos requisitos levantados por Duarte (2014) e Prado (2015);
3. Definição da documentação do projeto: por meio do uso do método FrameWeb, elaboração da arquitetura do projeto para o *framework* utilizado;
4. Desenvolvimento da implementação: geração de uma nova implementação do SCAP por meio do uso do *framework* descrito no projeto;
5. Redação da monografia: utilização do template abnTeX[1] para a escrita da monografia em LaTeX[2] seguindo os requisitos das normas da ABNT (Associação Brasileira de Normas Técnicas);

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

6. Apresentação do Projeto: apresentação final da monografia e demonstração do sistema SCAP com o *framework* proposto.

1.4 Organização da Monografia

A organização desta monografia foi construída e dividida da seguinte maneira:

- **Capítulo 1: Introdução**

  Esse capítulo descreve a metodologia utilizada, os objetivos e as questões relacionadas ao contexto do projeto.

- **Capítulo 2: Referencial Teórico**

  Esse capítulo descreve o que foi estudado com relação aos principais temas abordados ao longo deste trabalho, ou seja: o método FrameWeb, Engenharia *Web* e *Frameworks*.

- **Capítulo 3: Especificação de Requisitos**

  Esse capítulo traz uma breve descrição das funcionalidades e objetivos do SCAP, assim como a análise dos requisitos e especificação.

- **Capítulo 4: Projeto Arquitetural e Implementação**

  Esse capítulo exibe o resultado da execução do método FrameWeb no projeto, trazendo os modelos e mostrando as tecnologias utilizadas na implementação.

- **Capítulo 5: Considerações Finais**

  Esse capítulo apresenta as expectativas relacionadas a trabalhos futuros e mostra as conclusões referentes à nova implementação do SCAP.

2 Referencial Teórico

No decorrer deste capítulo serão apresentados conceitos usados como base para o desenvolvimento deste trabalho: um breve resumo sobre Engenharia de Requisitos (Seção 2.1), uma descrição sobre a Engenharia *Web*, alguns de seus atributos técnicos de qualidade e as fases seguidas por uma abordagem iterativa (Seção 2.2); o método FrameWeb, suas propostas, suas divisões através de camadas e a utilização de uma linguagem específica de modelagem (Seção 2.3). O capítulo também aborda uma visão sobre as categorias dos *frameworks*, uma descrição sobre o *framework* Grails utilizado no projeto e a linguagem Groovy que é utilizada por ele (Seção 2.4).

2.1 Engenharia de Requisitos

Considerados um fator determinante para o fracasso ou sucesso de um projeto de *software*, os requisitos desempenham um papel central no processo de *software*. De modo geral, é possível dizer que os requisitos de um sistema incluem restrições sob as quais ele deve operar, restrições que devem ser satisfeitas no seu processo de desenvolvimento, especificações dos serviços que o sistema deve prover e propriedades gerais do sistema (FALBO, 2017).

Os requisitos podem ser definidos como as descrições das restrições operacionais e dos serviços que devem ser providos pelo sistema (SOMMERVILLE, 2007). Com relação ao tipo de informação documentada por um requisito, uma classificação é amplamente aceita e de acordo com Sommerville (2007), os **requisitos funcionais** são declarações de serviços que o sistema deve prover, descrevendo o que o sistema deve fazer. Já os **requisitos não funcionais**, descrevem restrições sobre os serviços ou funções oferecidas pelo sistema. Neste contexto, ainda existem as regras de negócio, que definem requerimentos e restrições do negócio, sendo particulares para cada cliente.

Uma tarefa bem útil é representar os requisitos em níveis diferentes de descrição, pois os desenvolvedores, os clientes que contratam o desenvolvimento do sistema e os usuários finais são muito interessados em requisitos, mas possuem expectativas distintas. Assim, Sommerville (2007) realiza a descrição de dois níveis de requisitos:

- **Requisitos de Usuário ou de Cliente:** devem ser de fácil entendimento para clientes e usuários do sistema que não possuem conhecimento técnico. São diagramas intuitivos das restrições e dos serviços esperados do sistema, todos por meio da utilização de linguagem natural.

- **Requisitos de Sistema:** especificam em detalhes as restrições, serviços e funções do sistema, produzindo uma versão melhorada dos requisitos de clientes que os desenvolvedores utilizam para implementar, projetar e testar o sistema.

Em seguida, as funcionalidades que um sistema deve prover podem ser capturadas e descritas pelos modelos de casos de uso. Na maioria das vezes, um sistema atende a vários atores e por este motivo, analisar a funcionalidade que ele provê como uma única unidade pode ser uma tarefa complicada. O conceito de caso de uso tem a utilidade de dividir essa funcionalidade em partes mais agradáveis e menores (OLIVÉ, 2007).

Então, com a responsabilidade de relacionar as necessidades e as exigências do cliente, a fase de análise de requisitos se segue e estabelece atividades para que seus objetivos sejam atingidos por meio de soluções que deverão ser implementadas e entregues em um produto final de *software*. Desta maneira, a análise de requisitos engloba a compreensão dos estudos das necessidades e das solicitações do usuário para que as soluções sejam desenvolvidas. Resumidamente, a análise de requisitos envolve o entendimento do problema, sua avaliação, uma estratégia para a solução, sua modelagem, a especificação dos requisitos e a devida validação (AMUI, 2015).

Diferentes objetos desempenham um mesmo papel no mundo real, compartilhando um mesmo comportamento e uma mesma estrutura. Não existe a necessidade de realizar a modelagem de cada objeto individualmente, pois é mais vantajoso reunir em apenas um lugar, um modelo que descreve o comportamento e a estrutura desses objetos. Esse modelo passa a se tornar uma classe. Uma classe descreve um conjunto de objetos com os mesmos atributos, associações, operações e a mesma semântica. Portanto, a modelagem orientada a objetos é constituída pela definição de classes (FALBO, 2017).

Além de organizar as classes em um diagrama, esta fase também identifica restrições de integridade, que são limitações às possíveis soluções no desenvolvimento referente ao produto a ser entregue e não ao projeto em si. Elas não representam os requisitos de forma direta, mas induzem à definição de requisitos específicos. As restrições afetam a construção, o desenho da solução, a validação, os testes e a implantação do *software*, não podendo sofrer alterações. Por este motivo, elas devem ser muito bem validadas (VAZQUEZ; SIMÕES, 2016).

Após especificados os requisitos, se seguem atividades de projeto e implementação de *softwares*, que são invariavelmente intercaladas. O projeto de *software* é uma atividade criativa aonde são identificados os componentes de *software* e seus relacionamentos de acordo com os requisitos do cliente. A implementação estabelece o processo de concretização do projeto como um programa. Sempre existe um processo de projeto, pois um projeto trata de como resolver um problema. O projeto e a implementação estão intimamente ligados e, ao elaborar um projeto, é necessário levar em consideração os problemas de

implementação que serão enfrentados (SOMMERVILLE, 2011).

2.2 Engenharia *Web*

A *Web* é uma ferramenta que dispensa apresentações, pois ela já está familiarizada entre a maioria das pessoas, se encontra presente no dia a dia e em quase todas as áreas, podendo ser acessada por meio de muitos *hardwares* diferentes. Inicialmente, o conteúdo de *websites* era apenas textual, estático e não existia a presença de animações, sons, imagens ou conteúdo gerado de maneira dinâmica para cada tipo de usuário. A preocupação dos desenvolvedores permanecia em torno da simplicidade de apenas visualizar as informações sem complexidades.

Com a evolução de forma acelerada da *web*, surgiu a necessidade de mudanças significativas na maneira como os *websites* eram criados. Os *websites* passaram a englobar diversos conteúdos e funções complexas, adicionando centenas ou milhares de objetos em seu contexto. Sendo assim, quando toda a adição desses conteúdos começou a gerar um impacto direto no sucesso dos negócios, os projetos *web* deixaram de ser tratados de maneira superficial (PRESSMAN, 2011).

Acompanhando as evoluções do mundo, diversos setores onde não se imaginava uma maneira de como a *web* poderia ser utilizada, foram obrigados a adotar essa tecnologia para poderem se manter no mercado, se equiparando com a concorrência existente. Se antes era preciso concentrar e controlar todos os sistemas de forma interna e não unificada, atualmente já é possível realizar a terceirização de serviços, como por exemplo: o controle de banco de dados, o gerenciamento de e-mails, o controle de inúmeros *hardwares* de maneira remota, entre outros.

Algumas aplicações passaram a desempenhar um papel muito importante nas organizações, como por exemplo: as aplicações de instituições financeiras, que não toleram nenhum tipo de erro em sua utilização. Assim, os problemas encontrados nas aplicações *Desktop*, também passaram a ser visualizados nas aplicações para a *web*. Alguns fatores como a falta de qualificação e a falta de experiência dos desenvolvedores, a não utilização de modelos de processo, a não utilização de métricas para estimativas, se somavam para encadear os problemas encontrados. Além disso, o planejamento incoerente, os métodos obsoletos e inadequados, o não cumprimento de custos e prazos, a falta de documentação e o não cumprimento dos requisitos, dificultavam muito o controle da qualidade das aplicações (PERUCH, 2007).

Neste contexto de atualizações globais, à medida que a complexidade dos *websites* foi aumentando, eles passaram a ser considerados verdadeiras aplicações na *web*, sendo necessário utilizar os fundamentos da Engenharia *Web*, que pode ser definida como a utilização de conceitos, princípios e métodos da Engenharia de Software, de modo que

estabeleçam uma maneira de realizar adaptações referentes às características das aplicações *web* (BEDER, 2012).

Existem atributos técnicos de qualidade que são utilizados na Engenharia de Software. A usabilidade visa a facilidade de utilização da aplicação, independente do tipo de usuário. A funcionalidade faz referência ao comportamento do sistema, buscando operações e informações corretamente. A eficiência é voltada para o tempo de resposta, retornando as informações em uma velocidade satisfatória. A confiabilidade deve garantir a recuperação de erros e validação das informações e a manutenibilidade deve garantir a fácil atualização das operações existentes na aplicação. Todos esses atributos podem ser utilizados no desenvolvimento de aplicações *web*. De acordo com Offutt (2002), os principais atributos técnicos de qualidade podem ser estendidos por meio de outros atributos:

- **Segurança:** existem inúmeras informações confidenciais que são armazenadas e extraídas por meio das *WebApps*, além de existir a integração com bancos de dados governamentais e corporativos. Por estes motivos, assim como outros, em inúmeras situações, a segurança da *WebApp* deve ser tratada com prioridade. Para estabelecer o atributo de segurança, a *WebApp* deve possuir a habilidade de se defender contra ataques maldosos e bloquear solicitações que não possuem acesso autorizado;
- **Disponibilidade:** uma *WebApp* indisponível não tem serventia nenhuma para os usuários, mesmo que ela seja de extrema qualidade. A disponibilidade é definida pelo percentual de tempo que a aplicação fica disponível para uso. Os usuários sempre esperam que as *WebApps* fiquem disponíveis a todo o momento, no entanto, a disponibilidade também está relacionada com os tipos de plataformas diferentes que as *WebApps* são compatíveis;
- **Escalabilidade:** os servidores e as *WebApps* não podem ser projetados para um número fixo de usuários. A capacidade de volume e a capacidade de resposta devem ser levadas em consideração durante a construção, ou seja, variações significativas podem ocorrer a qualquer momento. Um número bem grande de usuários devem ser esperados no futuro;
- **Tempo de inserção no mercado:** do ponto de vista comercial, é uma boa medida de qualidade. Geralmente, um número variável de usuários são atraídos pelas primeiras *WebApps* que atendem um segmento específico de mercado. O usuário fica responsável por realizar a avaliação da qualidade da *WebApp*;

O desenvolvimento de uma *WebApp* é uma atividade com características variadas, envolvendo questões organizacionais, técnicas, gerenciais, artísticas e sociais. Reunindo um conjunto de atividades aplicadas, o objetivo é gerar uma aplicação de qualidade que atenda às características esperadas de forma eficiente.

Ao iniciar um projeto de uma *WebApp*, a Engenharia *Web* estabelece algumas fases para que uma abordagem iterativa seja seguida. Essas fases são aplicadas de acordo com que o projeto se desenvolve. Se repetindo quantas vezes forem as iterações do projeto, as fases de comunicação, planejamento, modelagem, construção e emprego, produzem um incremento de *software* a cada iteração, disponibilizando uma parte das funcionalidade e dos recursos do *software*, se tornando bem mais completo (PRESSMAN, 2011).

2.2.1 Comunicação

De início temos a fase de **comunicação**, onde é necessário compreender os objetivos das partes interessadas, conhecendo as restrições e necessidades do *software*, documentando o registro da análise e realizando a verificação, validação e gerenciamento dos requisitos. Além disso, os requisitos de qualidade, interface de usuário, ambiente de sistema e conteúdo também devem ser tratados, assim como requisitos não-funcionais.

2.2.2 Planejamento

Em sequência, é descrita a fase de **planejamento**, em que as tarefas técnicas a serem conduzidas devem ser descritas, assim como os recursos que serão utilizados, um cronograma de trabalho, os riscos prováveis e os resultados esperados dos produtos. Por meio dos requisitos especificados, os elementos da arquitetura são definidos mediante a uma transmissão entre a análise de contexto e o desenvolvimento do sistema.

2.2.3 Modelagem

Na fase de **modelagem**, os padrões e as normas são definidas para realizar a organização do desenvolvimento do sistema, acrescentando detalhes para que o problema e a solução proposta sejam compreendidos da melhor maneira possível. As transições entre as fases e os métodos de realização também são definidos, abordando todas as questões globais do projeto.

2.2.4 Construção

Posteriormente, na fase de **construção**, é preciso transformar toda a lógica, operações e o controle do sistema em código-fonte, utilizando uma linguagem de programação determinada. O conteúdo da aplicação, o aspecto visual e os elementos de navegação são definidos e testes são realizados para revelar possíveis erros na implementação do código.

2.2.5 Emprego

Por fim, na fase de **emprego**, a aplicação é entregue ao cliente, em partes que serão incrementadas ou completa, para que ele realize a avaliação do produto. A aplicação deve ser mantida sempre atualizada e permitir uma manutenibilidade fácil para garantir que esteja sempre disponível e seja funcional. Nesta fase ainda podem ocorrer mudanças estruturadas ou não estruturadas.

2.3 O Método FrameWeb

Utilizando *frameworks* como base, o FrameWeb (*Framework-based Design Method for Web Engineering*) (SOUZA, 2007; SOUZA, 2020) é um método de projeto voltado para o desenvolvimento de sistemas de informação *Web* (*Web Information Systems* - WISs). Ao se desenvolver aplicações distribuídas, especialmente as baseadas na plataforma *Web*, o uso de *frameworks* se padronizou, passando a existir inúmeras propostas para a Engenharia *Web*. Mesmo com muitos *frameworks* existentes, assim como métodos e metodologias, não havia nada que englobasse diretamente as características dos *frameworks* utilizados no desenvolvimento de WISs. O método FrameWeb surgiu com o intuito de propor modelos de projetos que se aproximam da implementação do sistema, definindo uma arquitetura básica para o WIS e assumindo que determinados tipos de *frameworks* serão utilizados no decorrer da construção da aplicação.

Voltado para a fase de projeto arquitetural, o FrameWeb visa deixar as organizações e os desenvolvedores com a opção de adotar as técnicas mais adequadas para as demais etapas do processo. Nesta fase, as principais propostas do método são estabelecidas:

- Divisão do sistema em camadas, através de uma arquitetura padrão, de modo a realizar uma integração com os *frameworks* utilizados;
- Construção de modelos de projeto que reúnem conceitos utilizados pelos *frameworks*, através de um perfil UML que traga os diagramas para mais perto da implementação.

O FrameWeb propõe o uso de uma arquitetura lógica do sistema, que seja estabelecida pelo padrão arquitetônico *Service Layer* (Camada de Serviço) (FOWLER, 2002), representado na Figura 1. O sistema é dividido em três camadas grandes que são subdivididas internamente por pacotes:

- **Lógica de Apresentação:** possui a responsabilidade de realizar a interação entre o sistema e o usuário, exibindo as informações e interpretando os comandos em ações da persistência de dados e da lógica de negócio.

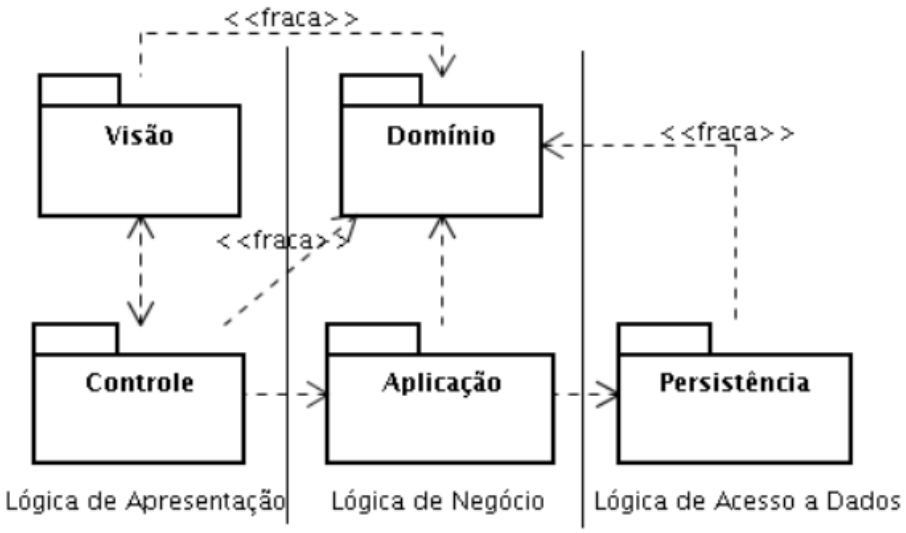

Figura 1 – Arquitetura padrão para WIS baseada no padrão arquitetônico *Service Layer* (FOWLER, 2002).

– Pacote de **Visão:** realiza a iteração humano-computador, definindo o formato de relatórios, formulários e janelas, sendo que a construção de protótipos é muito útil para facilitar o desenvolvimento dos mecanismos que serão utilizados.

– Pacote de **Controle:** define as classes que serão responsáveis por controlar a iteração, enviando requisições para os objetos da Lógica de Negócio.

- **Lógica de Negócio:** engloba as funcionalidades que dão suporte aos processos de negócio, concentrando as regras de negócio, conceitos do domínio, cálculos e processamentos.

  – Pacote de **Domínio:** reúne o relacionamento direto entre os diagramas de classes produzidos na fase de análise e os conceitos do domínio do problema.

  – Pacote de **Aplicação:** está ligado ao modelo de casos de uso e é projetado de maneira independente da interface com o usuário.

- **Lógica de Acesso a Dados:** estabelece o acesso a dados, gerenciando requisições e cuidando da sincronização de elementos de dados.

  – Pacote de **Persistência:** contém as classes que são responsáveis por gravar os objetos de domínio que necessitam ser persistidos pelo sistema em um banco de dados. O método FrameWeb sugere que as classes sigam o padrão de projeto *Data Access Object* (DAO) (ALUR; CRUPI; MALKS, 2003). Este padrão determina uma interface de operações de persistência, incluindo métodos para a criação, recuperação, alteração e exclusão, agrupando o código relacionado à entidade persistente (BAUER; KING, 2007).

Com o objetivo de representar de maneira direta os conceitos existentes nos *frameworks* que podem ser integrados, o uso de uma linguagem específica de modelagem se tornou necessária (MARTINS; SOUZA, 2015), pois os artefatos que serão codificados pelos desenvolvedores após a fase de projeto também devem ser modelados. O FrameWeb apresenta uma linguagem de modelagem que estende o meta-modelo UML, representando os componentes relacionados aos *frameworks* e os componentes mais utilizados no desenvolvimento *Web*. Um perfil UML é utilizado para a construção de quatro tipos de diagramas:

- **Modelo de Entidades:** partindo do modelo de classes desenvolvido na fase de análise, os objetos de domínio do problema que serão persistidos no banco de dados são representados por um diagrama de classes da UML. Além disso, os mapeamentos que guiam a persistência dos objetos destas classes são adicionados;
- **Modelo de Persistência:** um diagrama de classes da UML representa a implementação das classes DAO existentes no sistema e que serão persistidas, exibindo todas as implementações das interfaces, assim como os seus respectivos métodos;
- **Modelo de Navegação:** demonstra as páginas *Web* do sistema, seus atributos e suas iterações, exibindo o funcionamento dos inúmeros componentes que formam a camada de Lógica de Apresentação. Além disso, ele estabelece a maneira como cada categoria de usuário irá navegar de um elemento da *WebApp* para outro. A mecânica de navegação é definida como parte do projeto, sendo necessário observar a definição e se concentrar nos requisitos gerais de navegação (PRESSMAN, 2011).
- **Modelo de Aplicação:** exibe por meio de um diagrama de classes da UML, as classes de serviço, que implementam os casos de uso e as suas dependências. Esse diagrama é utilizado para auxiliar os desenvolvedores na implementação das classes do pacote Aplicação e na configuração das dependências entre os pacotes Controle, Aplicação e Persistência. Assim, é possível definir quais DAOs são necessários para que as classes de serviço alcancem seus objetivos e quais classes de ação dependem de quais classes de serviço (SOUZA, 2007).

  De acordo com Souza (2007), se existirem classes de ação que dependem de uma classe de serviço exibida, elas também devem estar presentes no modelo, realizando a representação por meio da indicação da dependência por meio de uma associação direcionada à interface da classe de serviço e de espaços de nomes que pertencem a outro pacote. Portanto, quando a classe de serviço depende de algum DAO, a interface deste deve aparecer no diagrama e estar associada à classe de serviço em questão.

A definição da linguagem de modelagem permite, ainda, a criação de ferramentas de apoio ao método, como um editor gráfico (CAMPOS; SOUZA, 2017) e um gerador de código (ALMEIDA; CAMPOS; SOUZA, 2017). O FrameWeb Editor estabelece um ambiente gráfico para a criação de modelos FrameWeb, apresentando suas funcionalidades e aspectos relevantes de sua implementação.

2.4 Frameworks

Para realizar a implementação de um WIS, através da Engenharia *Web*, o que mais se pode encontrar são ferramentas, propostas de metodologias e inúmeras linguagens que facilitam o processo realizado pelos desenvolvedores. Após a construção dos primeiros sistemas, verificou-se que os WISs possuíam uma infraestrutura arquitetônica bem parecida. Desta maneira, diversos *frameworks* foram criados para generalizar essa infraestrutura e para facilitar a realização do desenvolvimento de novas aplicações.

Assim, um *framework* pode ser considerado um design reutilizável de uma parte ou de todo o sistema, sendo representado por um conjunto de classes abstratas e concretas que demonstram a forma como suas instâncias interagem com um grande potencial de especialização (MATTSSON; BOSCH, 1999). Com a utilização das funcionalidades que os *frameworks* oferecem, a implementação dos códigos fontes das aplicações se tornam bem mais simples, minimizando os custos e o tempo utilizado.

Diversos *frameworks* foram criados para a plataforma Java e após a construção de inúmeros WISs, Souza (2007) estabeleceu uma organização para os *frameworks* em seis categorias diferentes:

- *Frameworks* MVC (Controladores Frontais);
- *Frameworks* Decoradores;
- *Frameworks* de Mapeamento Objeto/Relacional;
- *Frameworks* de Injeção de Dependência (Inversão de Controle);
- *Frameworks* para Programação Orientada a Aspectos (AOP);
- *Frameworks* para Autenticação e Autorização.

O *framework* Grails utilizado neste trabalho, pertence à categoria de *frameworks* MVC (Controladores Frontais) e será descrito nas próximas subseções.

2.4.1 Controladores Frontais: Frameworks MVC

O padrão MVC, abreviatura de Modelo-Visão-Controlador (*Model-View-Controller*), possibilita a divisão do projeto em camadas muito bem definidas. Correspondendo a objetos da camada de Lógica de Negócio, o **(modelo)** faz referência aos objetos que descrevem as informações sobre o negócio. A **(visão)** cuida da exibição e da entrada de informações na interface do usuário. O **(controlador)** trata das requisições, envia para a camada de Lógica de Negócio e após receber as respostas, realiza a solicitação para que as informações sejam atualizadas pela **(visão)**. A Figura 2 demonstra como funciona esse padrão arquitetural.

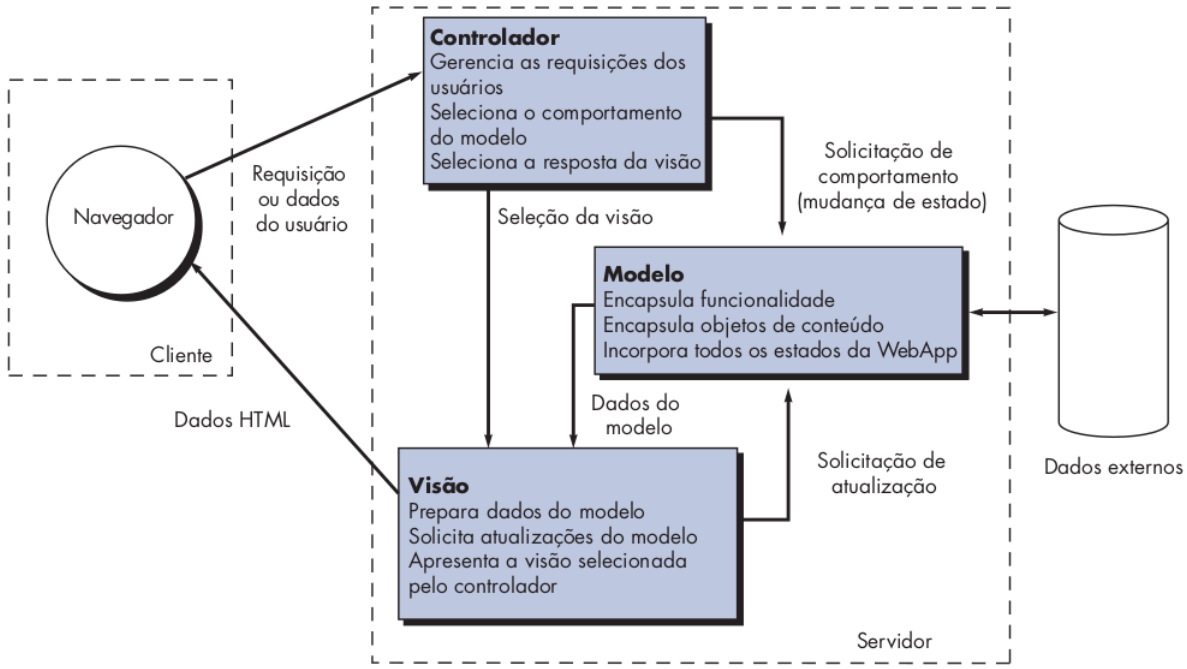

Figura 2 – Arquitetura MVC através de uma representação esquemática (PRESSMAN, 2011).

O **(modelo)** apenas tem conhecimento sobre a lógica e os dados que estão armazenados no sistema por meio de bancos de dados ou arquivos. Ele é considerado o núcleo da aplicação e é onde as operações de CRUD podem ser realizadas. A **(visão)** não possui lógica de negócio e fica responsável por controlar a entrada dos dados fornecidos pelo usuário, devolvendo a saída das informações assim que elas forem repassadas pelo **(controlador)**. O **(controlador)** interpreta as ações do usuário e realiza um mapeamento dessas ações em comandos que selecionam o comportamento do **(modelo)** ou a resposta da **(visão)**.

O padrão MVC aborda dois tipos de separação. No primeiro tipo, existe uma separação entre a apresentação **(visão)** e a lógica de negócio **(modelo)**. No segundo tipo, o **(controlador)** é separado da **(visão)**, sendo este segundo tipo menos importante do que o primeiro. Como inúmeros sistemas possuem um único controlador por visão, o segundo

tipo quase não é utilizado. Entretanto, o segundo tipo é mais comum em interfaces *Web*, pois a parte de **(visão)** *front end* é naturalmente separada do **(controlador)** (FOWLER, 2002).

De acordo com Souza (2007), a arquitetura MVC necessita de algumas alterações para atender as necessidades dos aplicativos *Web*. O **(modelo)**, situado no servidor *Web*, não consegue enviar as notificações das alterações para a **(visão)**, já que se encontra no navegador do lado do cliente e a comunicação é sempre iniciada pelo cliente. Então, apesar de MVC ser um nome bem conceituado, o nome correto para esse padrão arquitetônico, quando aplicado à *Web*, seria "Controlador Frontal" (*Front Controller*) (ALUR; CRUPI; MALKS, 2003). O padrão Controlador Frontal pode ser visualizado na Figura 3.

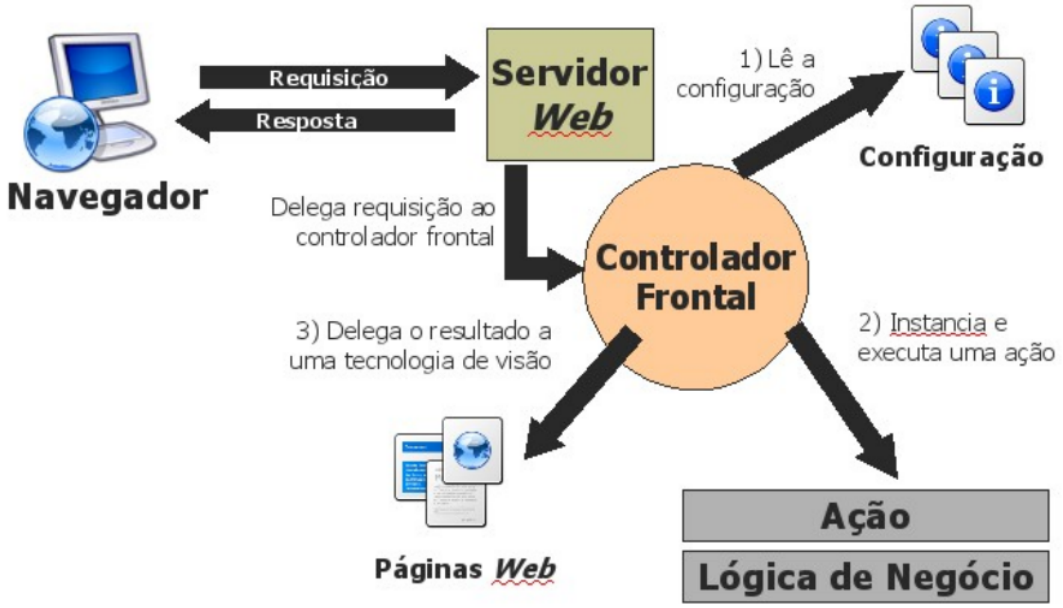

Figura 3 – Funcionamento do padrão arquitetônico Controlador Frontal na *Web* (SOUZA, 2007).

2.4.2 Framework Grails

O Grails é um *framework* voltado para o desenvolvimento de aplicações *web*, baseado em MVC e que utiliza a linguagem Groovy. Com o objetivo de facilitar a vida dos desenvolvedores, a partir da utilização do Grails, a implementação da lógica de negócio passou a ser realizada de maneira imediata e sem a preocupação com a integração de inúmeras bibliotecas, como o Hibernate para a persistência, Spring e várias outras.

Visualizar de maneira instantânea o resultado das alterações em tempo de execução, também é uma característica do Grails. Algumas outras particularidades também podem ser mencionadas: as camadas de visualização e controle baseadas nas classes de domínio podem ser geradas automaticamente, a inversão de controle e a injeção de dependências são baseadas em Spring, os principais *frameworks* e bibliotecas Java podem ser integrados

facilmente e o *framework* para persistência GORM fica responsável por controlar a validação dos dados, tirando a preocupação dos desenvolvedores com arquivos de mapeamento ou com anotações.

Apesar do *framework* Grails fornecer a opção de geração automática das camadas de visualização e controle, cabe ao desenvolvedor escolher a melhor alternativa para a utilização no projeto. É possível gerar as camadas automaticamente e realizar as alterações necessárias contidas nos modelos criados nas outras etapas do projeto ou simplesmente optar por seguir a modelagem, implementando todas as funcionalidades.

Após o projeto utilizando o *framework* Grails ser finalizado, é possível obter como resultado uma aplicação Java EE completa, possuindo escalabilidade, robustez, desempenho e alta disponibilidade. Além disso, Grails é altamente portável entre servidores de aplicação (WEISSMANN, 2015).

2.4.3 Linguagem Groovy

Alterar o comportamento do código enquanto ele é executado pode ser bem simples utilizando Groovy. Por se tratar de uma linguagem dinâmica e orientada a objetos, ela tem como vantagem a execução de tarefas em tempo de execução, se destacando com relação às outras linguagens que só executam essas tarefas no momento da compilação ou por meio de padrões de projeto ou soluções arquiteturais que nem sempre são as mais amigáveis. A linguagem Groovy é altamente compatível com os códigos desenvolvidos por meio da linguagem Java, que podem ser acessados transparentemente pelos códigos implementados em Groovy e vice-versa.

Com relação às variáveis que são utilizadas na linguagem, a definição dos tipos pode acontecer de modo estático ou dinamicamente, ou seja, a definição dos tipos das variáveis utilizadas no Groovy não é obrigatório. Os métodos são bem mais simples de serem definidos, pois não é preciso explicitar o tipo de retorno e nem utilizar a instrução return para retornar um valor.

No decorrer do Capítulo 4 serão abordados mais alguns detalhes referentes a linguagem Groovy, que serão exemplificados de acordo como a demonstração da implementação do projeto.

3 Especificação de Requisitos e Análise do SCAP

Este capítulo apresenta uma descrição de escopo referente ao SCAP (Sistema de Controle de Afastamento de Professores), assim como o modelo de casos de uso e o diagrama de classes que foi levantado anteriormente por Duarte (2014) e posteriormente analisado por Prado (2015).

3.1 Descrição do Escopo

O SCAP surgiu com o objetivo de auxiliar o Departamento de Informática (DI) da UFES no controle e no registro de solicitações de afastamento do seus professores, para que eles possam participar de eventos que acontecem no Brasil e no exterior. Essas solicitações de afastamento necessitam passar por uma série de avaliações para que sejam aprovadas. Elas são avaliadas pelos professores do DI e dependendo do caso, também devem ser avaliadas pelo Centro Tecnológico (CT) e pela Pró-Reitoria de Pesquisa e Pós-Graduação (PRPPG). Nestes casos, somente após receber a aprovação de todas as instâncias, o afastamento é publicado no Diário Oficial da União e o professor recebe a autorização para participar do evento.

A Câmara Departamental (composta pelos representantes discentes e pelos funcionários do departamento) fica responsável por avaliar e aprovar as solicitações de afastamento para eventos no Brasil. O chefe do departamento (cargo ocupado por um professor do DI por meio de um mandato temporário) recebe a solicitação de afastamento pelo email e após dez dias, se nenhum membro da Câmara Departamental for contra ao pedido, o afastamento é aprovado. Assim, para eventos nacionais, o processo permanece dentro do DI.

Para pedidos de afastamento referente a eventos internacionais, um professor (sem parentesco com o solicitante) é escolhido para se tornar relator do pedido. Assim que o relator manda o parecer, o pedido passa por avaliação para aprovação como no caso descrito com eventos que são realizados no Brasil. Para que o pedido seja publicado no Diário Oficial da União, ele deve receber a aprovação do CT e da PRPPG. Entretanto, o SCAP não possui uma integração com os processos do CT e da PRPPG, fazendo com que o controle das tramitações permaneça dentro do DI, restringindo o escopo do sistema.

Com o intuito de automatizar as tramitações das solicitações de afastamento, o SCAP auxilia os professores e secretários do DI, facilitando o processo desde a criação até a aprovação e armazenamento. Com o envio de e-mails automáticos para os envolvidos e

com a utilização de formulários para a criação dos documentos necessários, o sistema pode ser considerado fundamental para esse processo.

3.2 Modelo de Casos de Uso

Após o levantamento de requisitos e da definição do escopo, os atores identificados no sistema SCAP são apresentados na Tabela 1.

Tabela 1 – Atores do SCAP (DUARTE, 2014).

Ator	**Descrição**
Professor	Professores efetivos do DI/UFES.
Chefe do Departamento	Professores do DI/UFES que estão realizando a função administrativa de chefe e subchefe do departamento.
Secretário	Secretário do DI/UFES.

A parte administrativa do sistema fica por conta dos **secretários**. Eles possuem a responsabilidade de realizar o cadastro dos mandatos dos chefes do departamento, realizar o cadastro dos professores e dos seus respectivos parentescos. Quando surgem pareceres de fora do DI e quando os pedidos de afastamento são finalizados, os **secretários** também ficam responsáveis pelo controle dessas tarefas.

O SCAP fornece algumas funcionalidades para os **professores**. Eles podem realizar o cadastro das solicitações de afastamento e realizar uma manifestação contra o afastamento de outro professor, caso ainda esteja dentro do prazo. Além disso, o **professor** fica responsável por recomendar a aprovação ou não de um afastamento no exterior, caso ele seja adicionado como relator do mesmo.

Se um **professor** se tornar **chefe do departamento** por meio de um mandato, ele deve realizar o encaminhamento de solicitações aos relatores que farão o deferimento de pareceres com relação aos afastamentos internacionais.

O diagrama de casos de uso do SCAP pode ser visualizado por meio da Figura 4. Uma pequena descrição dos casos de uso será apresentada nos próximos parágrafos e uma versão mais completa dessa descrição pode ser encontrada em (DUARTE, 2014; PRADO, 2015).

No sistema, um professor realiza o cadastro de um pedido de afastamento por meio do caso de uso **Solicitar Afastamento**, fornecendo todos os dados que são necessários para realizar a tramitação. Um professor pode cancelar uma solicitação de afastamento utilizando o caso de uso **Cancelar Afastamento**, realizando a alteração do status para cancelado.

Após escolher um relator para um pedido de afastamento internacional, o Chefe do Departamento pode executar o caso de uso **Encaminhar Afastamento**. Quando um

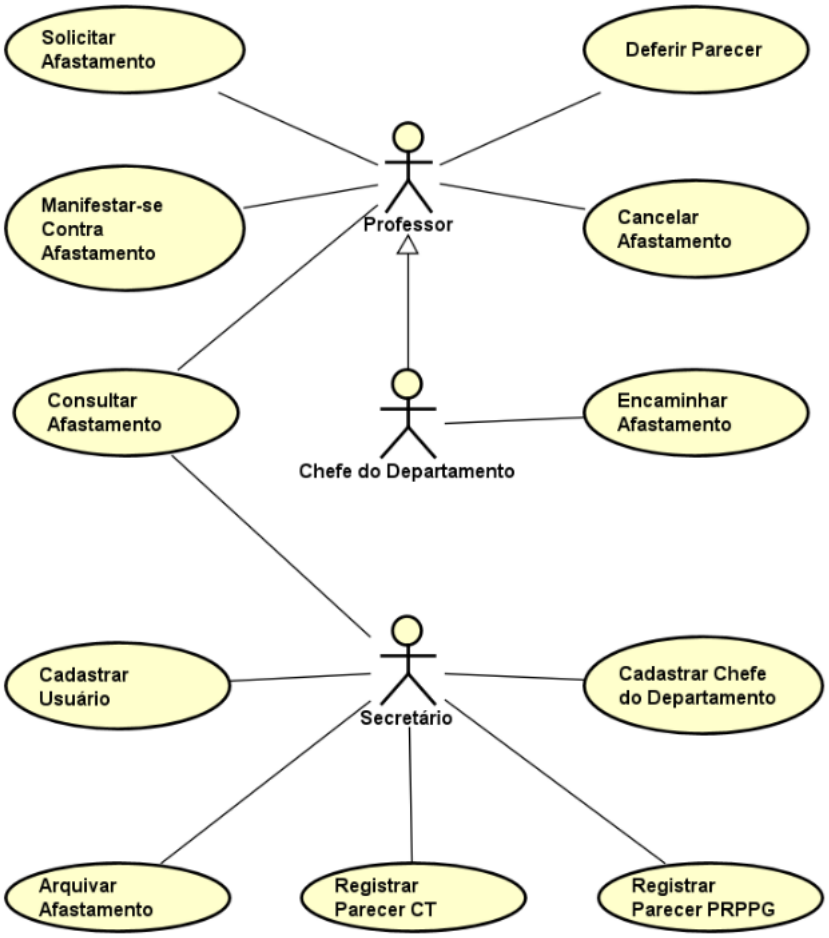

Figura 4 – Diagrama de Casos de Uso do SCAP.

professor que se tornou relator por meio da indicação do Chefe do Departamento realizar o cadastro do seu parecer sobre o afastamento, o caso de uso **Deferir Parecer** pode ser utilizado.

Um professor, um chefe de departamento ou um secretário podem utilizar o caso de uso **Consultar Afastamento** assim que necessitarem obter informações sobre uma solicitação de afastamento. Se um professor for contra a um pedido de afastamento, o caso de uso **Manifestar-se Contra Afastamento** pode ser utilizado e após o motivo ser cadastrado, uma reunião é agendada para decidir a aprovação ou reprovação da solicitação.

Um secretário realiza o cadastramento de novos professores ou secretários por meio do caso de uso **Cadastrar Usuário**, onde são informados todos os dados necessários. Um secretário também utiliza o caso de uso **Cadastrar Chefe do Departamento** para especificar o período do mandato do novo chefe do departamento.

Quando existe uma solicitação de afastamento internacional, um secretário realiza o cadastro do parecer do Centro Tecnológico e da Pró-Reitoria de Pesquisa e Pós-Graduação por meio dos casos de uso **Registrar Parecer CT** e **Registrar Parecer PRPPG**.

O caso de uso **Arquivar Afastamento** é executado após a tramitação de uma

solicitação de afastamento ser realizada, fazendo com que um secretário realize a alteração do status para "Arquivado".

3.3 Análise do SCAP

O diagrama de classes do SCAP pode ser visualizado por meio da Figura 5. As instâncias das classes são especificadas de acordo com o comportamento exercido pelos objetos que seguem as especificações das classes.

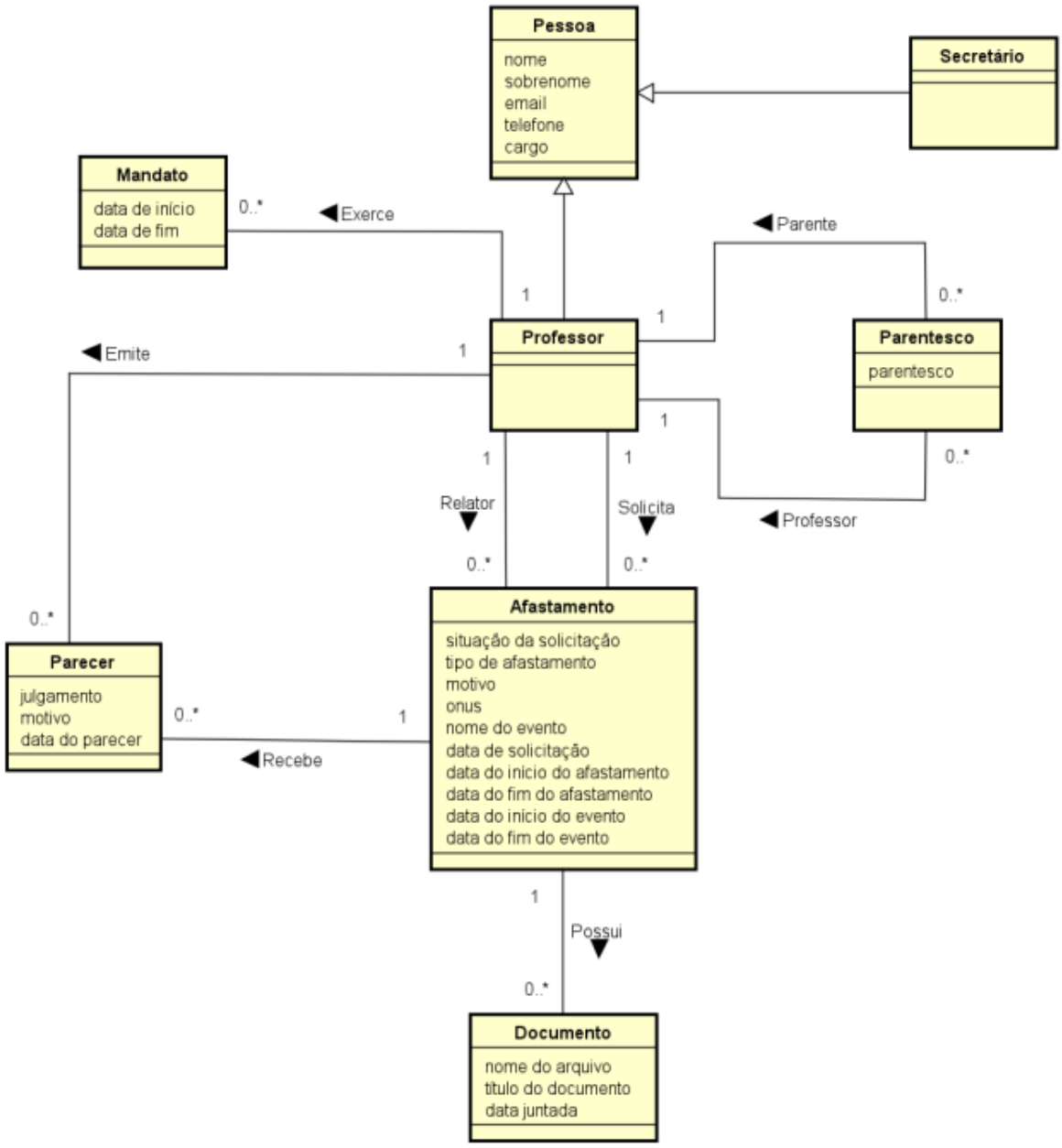

Figura 5 – Diagrama de Classes do SCAP.

Herdando todos os atributos da classe **Pessoa**, os professores são representados pela classe **Professor** e os secretários são representados pela classe **Secretário**. Os professores

podem possuir relações de parentesco, seja ela matrimonial ou sanguínea. Essas relações são representadas pela classe **Parentesco**. Um professor pode ocupar o cargo de chefe ou subchefe do departamento, sendo que o tempo de permanência no cargo é representado pela classe **Mandato**.

Os pedidos de afastamento solicitados por professores são representados pela classe **Afastamento**. Nela se encontram todas as informações necessárias para realizar as tramitações. Esta classe pode conter documentos que são visualizados por meio da classe **Documento**. Quando um pedido de afastamento se trata de um evento internacional, a relação **Relator** deve ser utilizada informando o professor que foi indicado para isso.

A classe **Parecer** é utilizada quando um professor efetua a emissão de um parecer com relação a um afastamento. Um professor pode se tornar relator e criar pareceres de vários afastamentos diferentes.

Restrições de integridade do sistema foram identificadas por Duarte (2014) e, após concluir a implementação de uma nova versão, Prado (2015) verificou que existia a necessidade de adicionar novas restrições. As restrições de integridade do SCAP que foram atualizadas estão descritas a seguir:

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

4 Projeto Arquitetural e Implementação

De acordo com o que já foi mencionado anteriormente, o método FrameWeb oferece suporte a quatro categorias de *frameworks*: Controlador Frontal, Injeção de Dependências, Mapeamento Objeto/Relacional e Segurança. Por este motivo, a aplicação do método na fase de projeto vem sendo testada por meio da implementação do sistema SCAP, utilizando *frameworks* que fazem parte destas categorias.

Como o objetivo principal deste trabalho era verificar os resultados da aplicação e testar a eficácia do método FrameWeb, a ferramenta SCAP não necessitou ser entregue englobando todas as suas funcionalidades. Assim, algumas restrições de integridade não foram implementadas, a ferramenta não possui a funcionalidade de enviar emails, gerar atas de reuniões e nem anexar documentos.

Neste capítulo, por meio da Seção 4.1, é descrito como o *framework* Grails organiza os diretórios do projeto. A Seção 4.2 apresenta as tecnologias presentes no projeto mediante o uso do *framework* Grails. A Seção 4.3 demonstra os modelos gerados por meio da aplicação do método FrameWeb e a Seção 4.4, exibe as telas do sistema SCAP que foi implementado por meio do *framework* Grails.

4.1 Arquitetura do Sistema

De uma maneira simplificada, a arquitetura do sistema é a organização ou a estrutura de componentes de programa, a maneira como esses componentes realizam iterações e a estrutura de dados que são utilizadas por estes componentes. De uma forma mais abrangente, entretanto, os componentes podem ser generalizados para representar as suas iterações e os principais elementos de um sistema (PRESSMAN, 2011).

A Figura 6 demonstra a estrutura da hierarquia dos diretórios e arquivos que são criados durante a utilização do *framework* Grails.

Por meio dos relatos apresentados por Beder (2012), a hierarquia de diretórios e arquivos do *framework* Grails segue o paradigma ***Convention Over Configuration*** **(CoC)**. O uso do **CoC** tem como objetivo diminuir a quantidade de decisões tomadas pelos desenvolvedores, adotando como "padrão" algo que é utilizado de forma comum, uma convenção. Seguindo as convenções, os desenvolvedores já sabem a priori onde se encontram todos os elementos que compõem a aplicação em desenvolvimento.

Na lista abaixo são apresentados de maneira geral os diretórios criados por meio do *framework* Grails. As abas dos diretórios estão representadas em negrito:

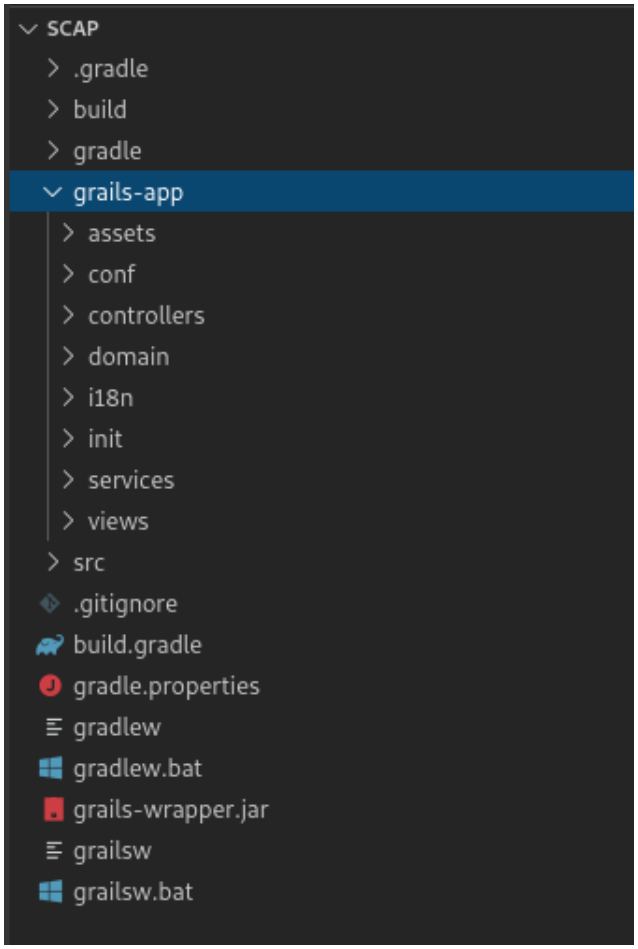

Figura 6 – Estrutura criada pelo *framework* Grails.

- **"grails-app/assets":** local onde se encontram os arquivos de imagens, javascript e *stylesheets*;
- **"grails-app/conf":** local onde se encontram as configurações da aplicação, tais como a configuração do banco, as configurações de inicialização, entre outros;
- **"grail-app/controllers":** local onde ficam localizados todos os controladores criados;
- **"grails-app/domain":** local onde se encontram os modelos ou classes de domínio;
- **"grails-app/i18n":** local onde são armazenados os pacotes de mensagens relacionados à internacionalização;
- **"grails-app/services":** local onde se encontram as classes utilizadas na camada de serviços (*Web Services*);
- **"grails-app/views":** local onde estão localizadas as visões, ou seja, os arquivos GSP (*Groovy Server Pages*), que são responsáveis por renderizar as páginas utilizadas pela aplicação.

Dentro dos diretórios, ainda é possível incluir arquivos referentes a bibliotecas terceirizadas que podem ser utilizadas no projeto, assim como códigos fontes escritos na linguagem Java ou na linguagem Groovy. Estes códigos podem ser reaproveitados pela aplicação.

4.2 Tecnologias Presentes no Projeto

Utilizando o *framework* Grails, esta versão do SCAP foi implementada na linguagem Apache Groovy. Os dados armazenados no banco de dados necessitavam estar de acordo com as especificações dos modelos utilizados no projeto. Para que essa tarefa fosse monitorada, foi utilizado a ferramenta phpMyAdmin, que facilitou a visualização e conferência dos dados.

Descrevendo mais sobre a linguagem Groovy, ela também pode ser utilizada como uma linguagem de *script*, não sendo necessário realizar a geração de arquivos executáveis e nem a sua compilação. O Groovy simplifica a implementação, adicionando dinamicamente às suas classes os métodos de acesso (*gets* e *sets*), economizando esforço e tempo. Com o objetivo de simplificar a sintaxe da linguagem Java, o Groovy representa comportamentos dinâmicos, como escritas e leituras, consulta a banco de dados e geração de objetos em tempo de execução ao invés de compilação (KÖNIG et al., 2007).

Nas subseções abaixo estão descritas mais algumas tecnologias que foram utilizadas no projeto por meio do *framework* Grails.

4.2.1 Scaffolding

A abordagem Scaffolding é um termo que foi adotado pelo *framework* Grails para realizar a geração dos controladores e das visões relacionadas às classes de domínio. Com a sua utilização, é possível gerar todo o código responsável por fornecer um CRUD essencial para o sistema, permitindo incluir, ler, atualizar e deletar os registros armazenados no banco de dados. Ainda é possível criar códigos de autenticação/autorização, testes unitários, entre outras operações (BEDER, 2012).

De acordo com Beder (2012), o Scaffolding pode ser estático ou dinâmico. O Scaffolding estático produz, através da utilização de *templates*, o código relacionado as visões e aos controladores que podem receber personalizações das equipes de desenvolvedores *web*. Já o Scaffolding dinâmico pode servir a vários propósitos, como na criação de interfaces simples, sem o intuito de realizar muitas personalizações nas visões que são geradas em tempo de execução. Para a implementação desta versão do SCAP, foi utilizado o Scaffolding dinâmico.

4.2.2 Gradle

O Gradle é uma ferramenta de construção de *build* bastante poderosa que pode ser utilizada na gerência de projetos, gestão de dependências, padronização de diretórios, construção, ciclo de vida, entre outras funcionalidades. O Gradle ainda possui uma quantidade imensa de *plug-ins* que podem ser muito aproveitosos nos projetos (WEISSMANN, 2015).

Com a adoção do Gradle pelo *framework* Grails, foi necessário realizar algumas mudanças na localização de alguns diretórios e de alguns arquivos com relação às versões anteriores. Isso facilitou o entendimento da estrutura e a configuração dos arquivos existentes.

4.2.3 GORM

O GORM (*Grails Object Relational Mapping*) é uma API (*Application Programming Interface*) que facilita muito a execução de tarefas relacionadas à persistência de objetos. Nas versões iniciais do *framework* Grails, o GORM era basicamente uma fina camada de código Groovy sobre o Hibernate, que fornecia uma interface de programação onde era possível aproveitar as características dinâmicas da linguagem (WEISSMANN, 2015).

De acordo com Weissmann (2015), hoje em dia, o GORM passou a ser visto como uma API de persistência Groovy real, onde é possível desenvolver sistemas em Grails que usem qualquer tipo de SGBD (Sistema Gerenciador de Banco de Dados), relacional ou não. Para isto, é necessário que seja escrita uma implementação do GORM para o SGDB que será utilizado. Uma observação interessante é que o GORM também pode ser utilizado com sucesso independentemente do Grails.

4.3 Modelos FrameWeb

Nas subseções a seguir são descritos e apresentados os modelos referentes à aplicação do método FrameWeb que foi realizada durante a fase de projeto arquitetural do SCAP. A ferramenta FrameWeb Editor (CAMPOS; SOUZA, 2017) foi utilizada para realizar a criação dos quatro tipos básicos de modelos FrameWeb, pois ela oferece um ferramental bem conhecido em softwares de modelagem contendo características e propriedades próprias da linguagem.

4.3.1 Modelo de Navegação

Os Modelos de Navegação foram gerados de acordo com os casos de uso do sistema. A Figura 7 e a Figura 8 representam o caso de uso “Cadastrar Usuário”, que é realizado por um secretário. O caso de uso “Cadastrar Chefe do Departamento” também é utilizado por um secretário e foi representado por meio da Figura 9.

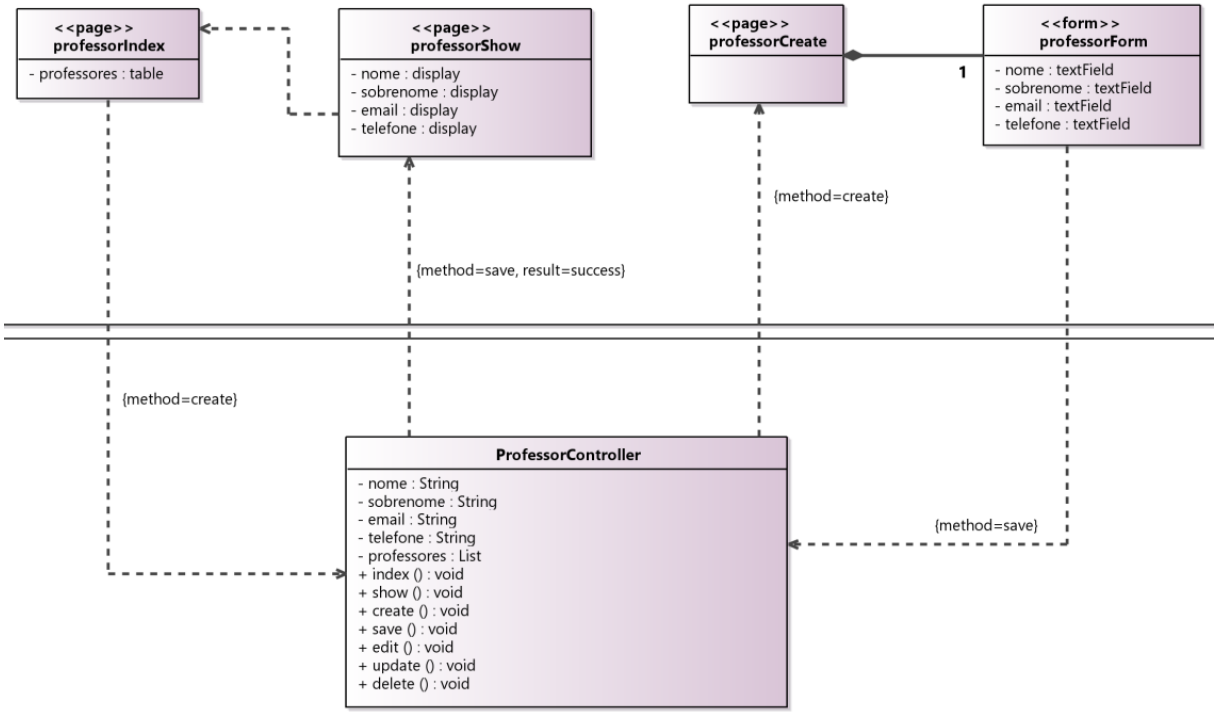

Figura 7 – Modelo de Navegação do Caso de Uso: Cadastrar Usuário - Professor.

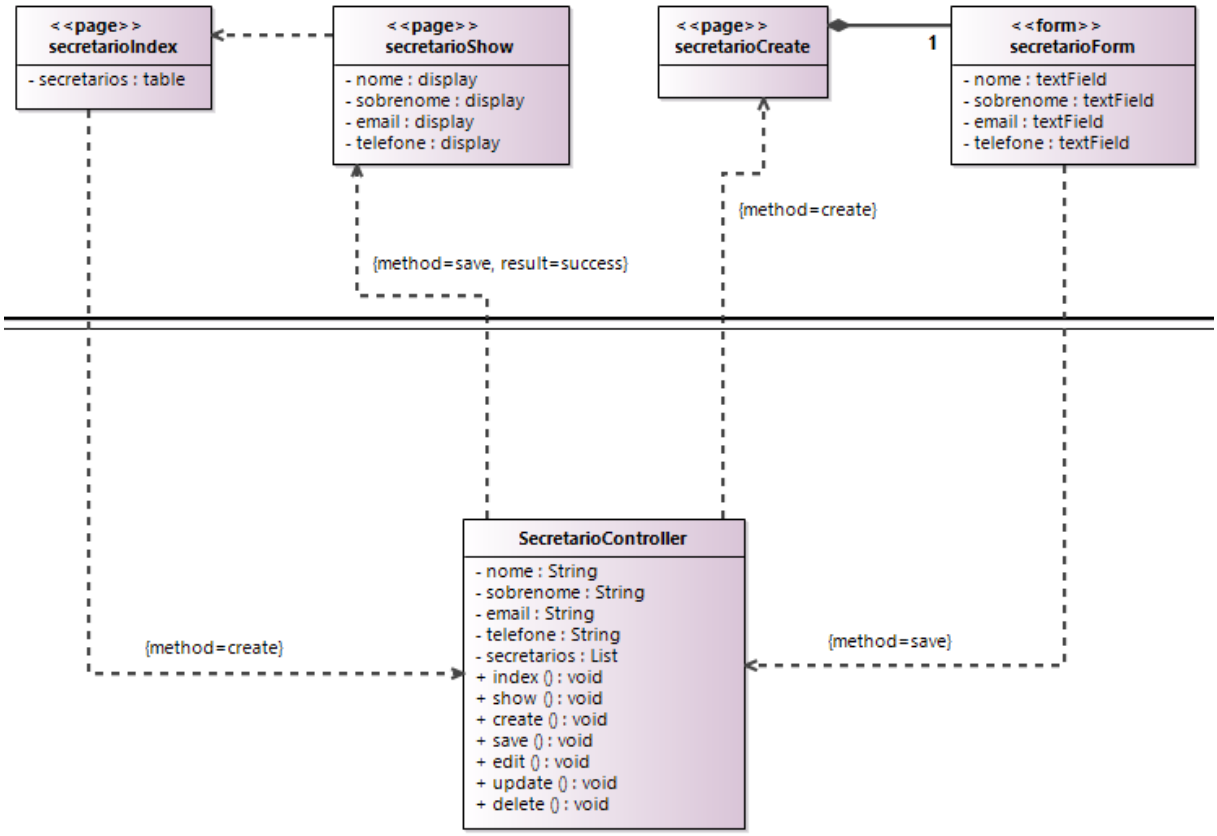

Figura 8 – Modelo de Navegação do Caso de Uso: Cadastrar Usuário - Secretário.

A Figura 10 representa o caso de uso “Solicitar Afastamento”, que acontece quando um professor realiza o cadastro de um pedido de afastamento. Caso exista um pedido de

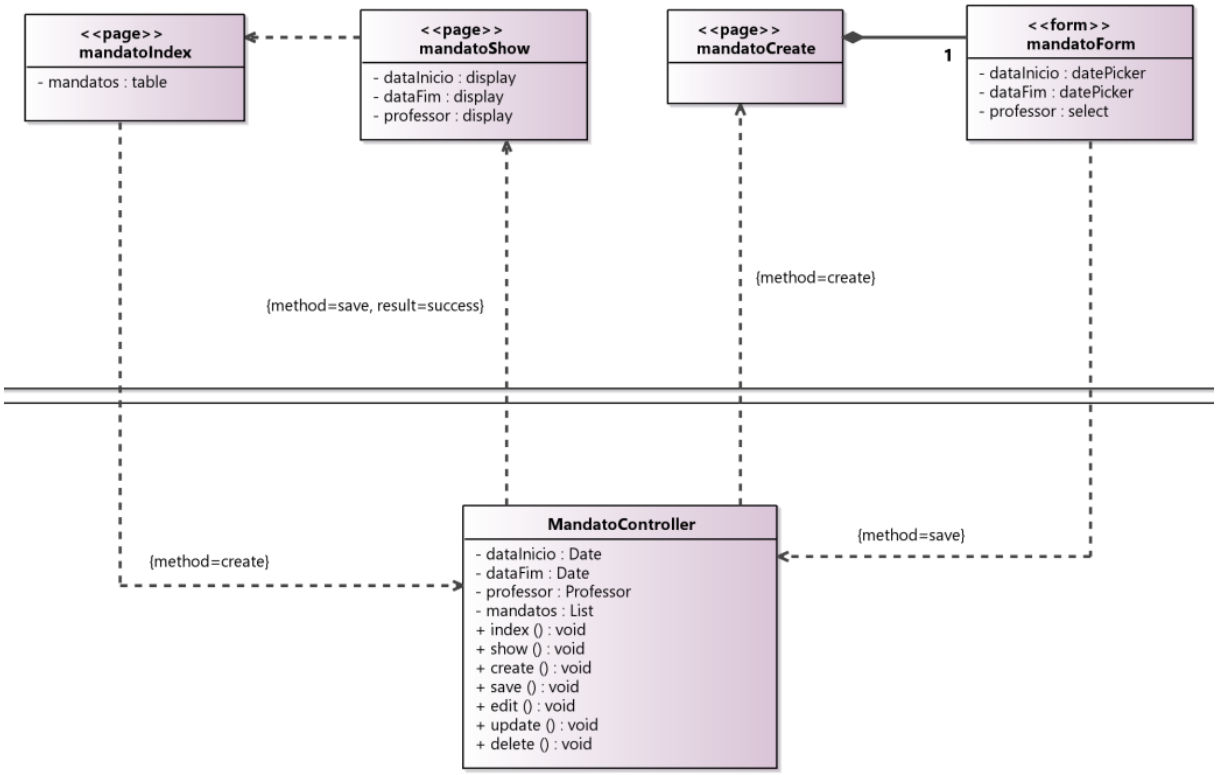

Figura 9 – Modelo de Navegação do Caso de Uso: Cadastrar Chefe do Departamento.

afastamento internacional, o chefe do departamento utiliza o caso de uso "Encaminhar Afastamento", que pode ser visualizado na Figura 11.

Os casos de uso "Deferir Parecer" e "Manifestar-se Contra Afastamento" são bem parecidos, pois é necessário realizar o cadastro de um parecer. Portanto, eles foram representados através da Figura 12.

Os casos de uso "Cancelar Afastamento", "Arquivar Afastamento", "Registrar Parecer CT" e "Registrar Parecer PRPPG" também são bem parecidos. Um professor cancela um pedido de afastamento realizando a alteração da situação da solicitação. Do mesmo modo, um secretário realiza a mudança da situação da solicitação, quando necessita arquivar um afastamento, registrar um parecer do CT ou registrar um parecer da PRPPG. Assim, esses casos de uso foram apresentados na Figura 13.

Todos os usuários do sistema podem utilizar o caso de uso "Consultar Afastamento". Para isso, é possível filtrar a lista de afastamentos através de quatro buscas diferentes. O caso de uso está representado através da Figura 14.

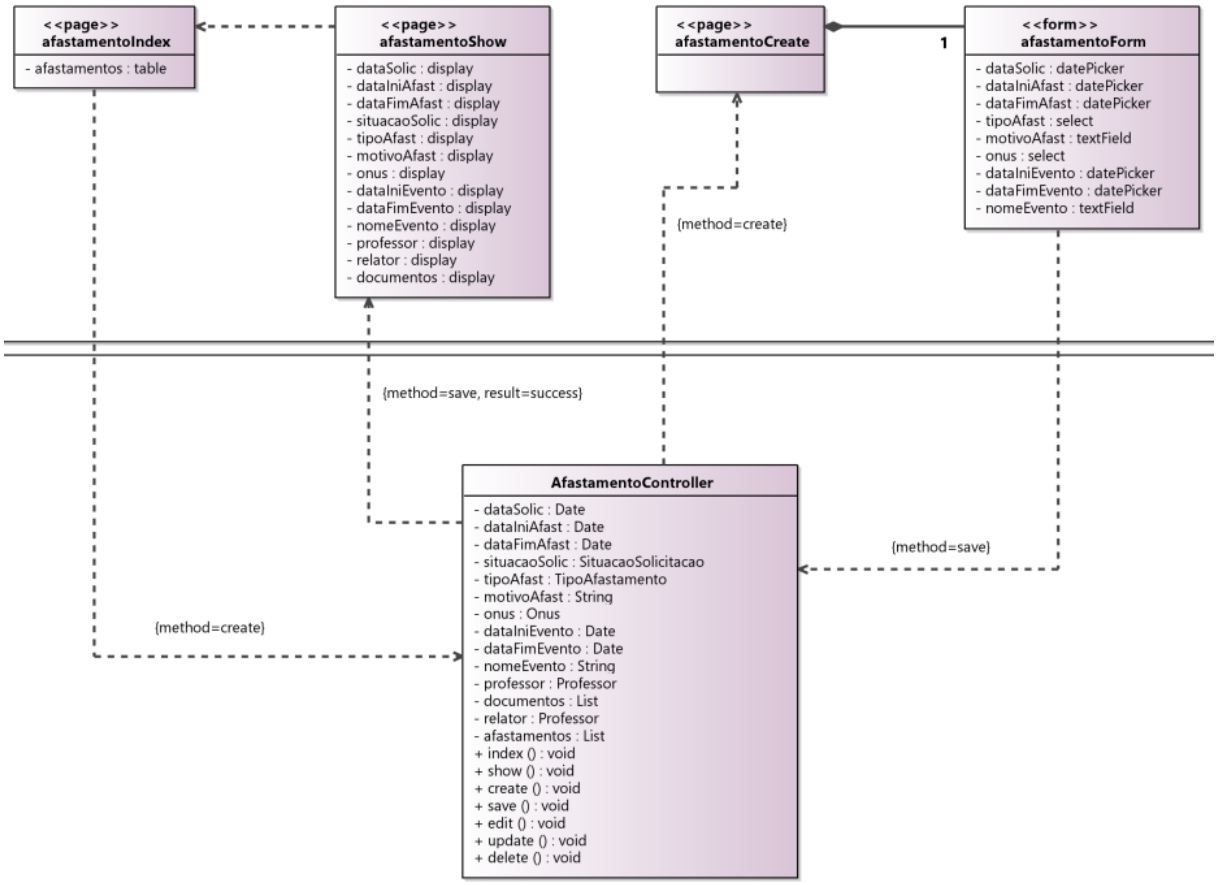

Figura 10 – Modelo de Navegação do Caso de Uso: Solicitar Afastamento.

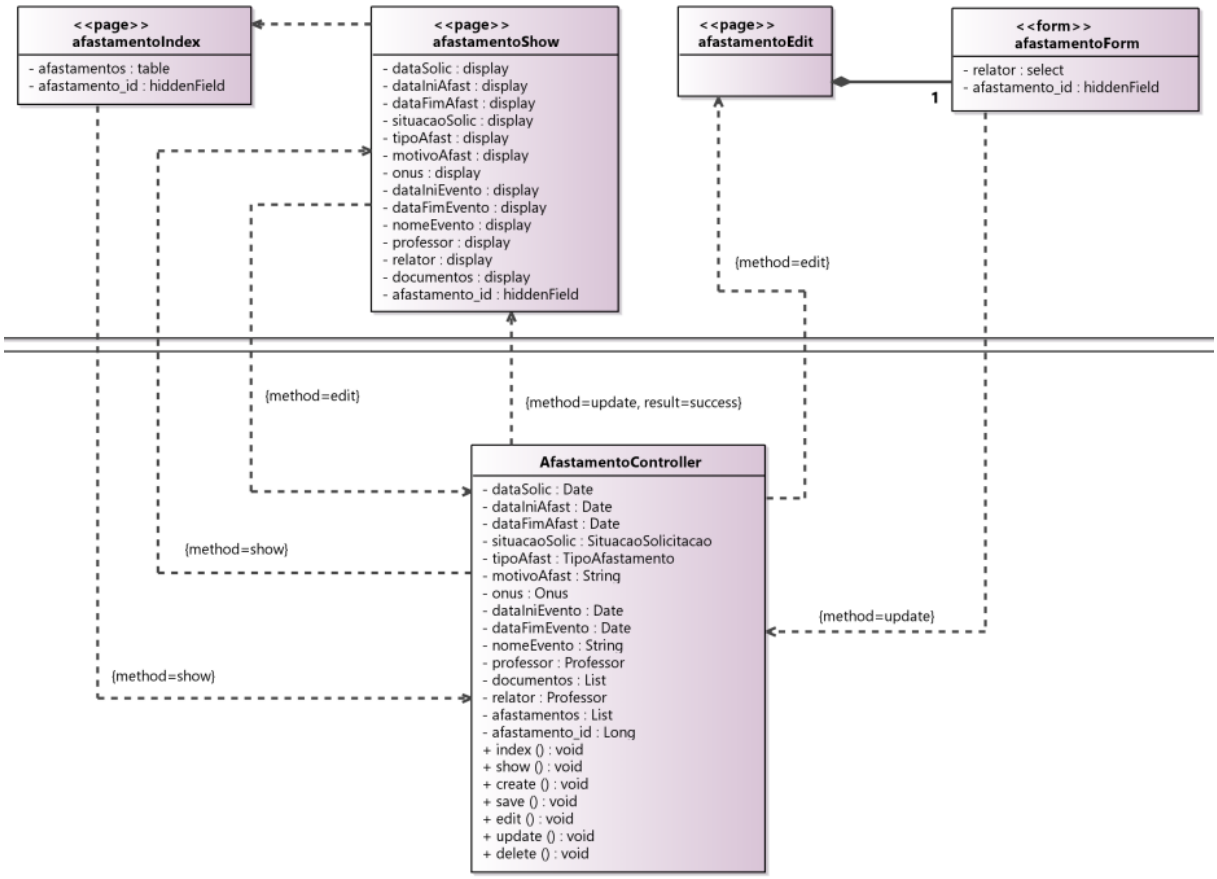

Figura 11 – Modelo de Navegação do Caso de Uso: Encaminhar Afastamento.

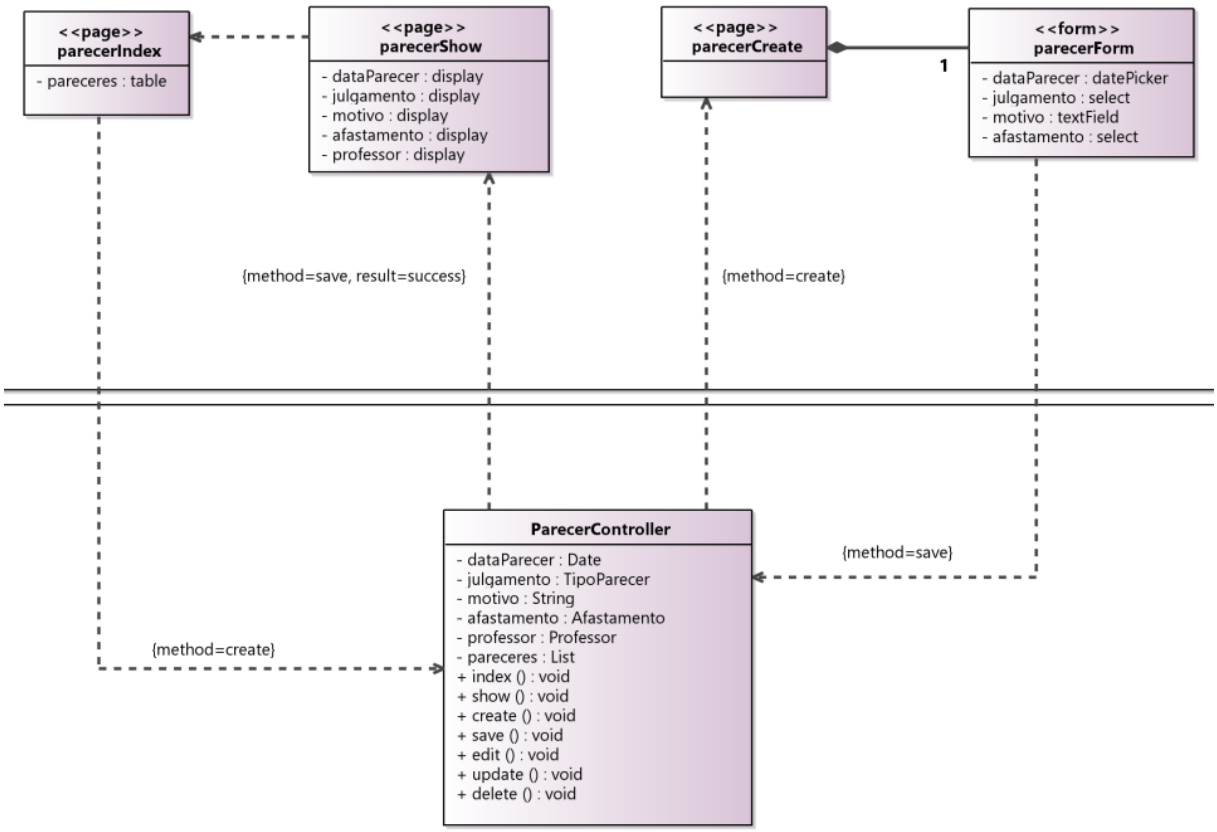

Figura 12 – Modelo de Navegação dos Casos de Uso: Deferir Parecer e Manifestar-se Contra Afastamento.

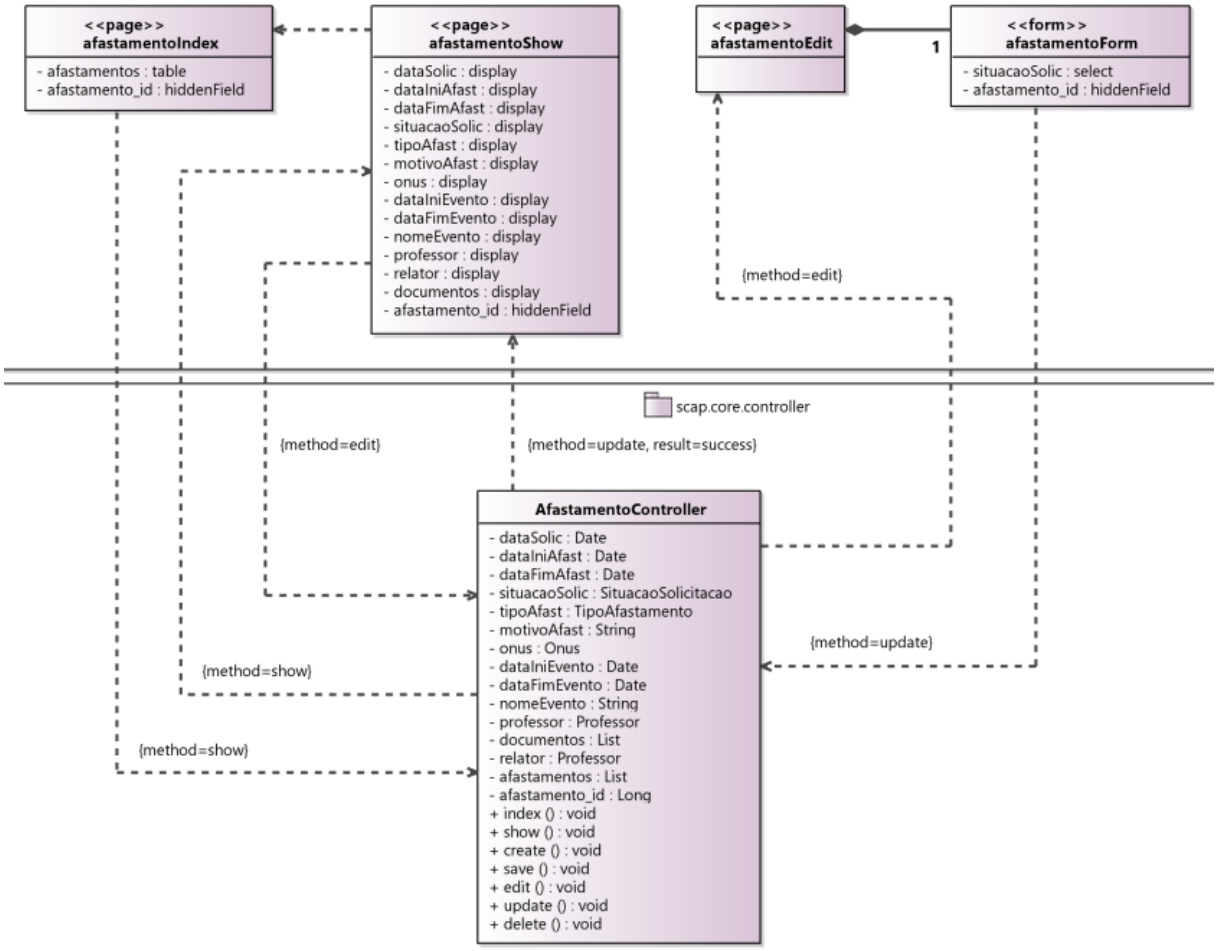

Figura 13 – Modelo de Navegação dos Casos de Uso: Cancelar Afastamento, Arquivar Afastamento, Registrar Parecer CT e Registrar Parecer PRPPG.

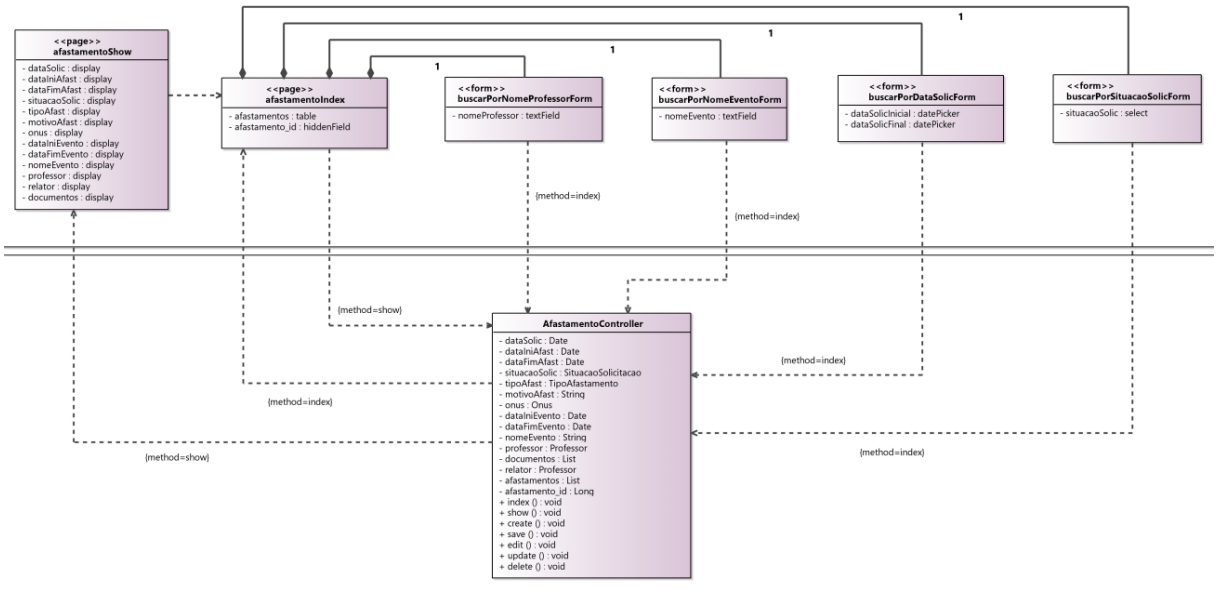

Figura 14 – Modelo de Navegação do Caso de Uso: Consultar Afastamento.

4.3.2 Modelo de Entidades

Utilizando o diagrama de classes que já foi apresentado na Figura 5, o Modelo de Entidades recebeu uma adequação de acordo com a plataforma de implementação utilizada. Foram especificados os tipos de dados de cada atributo, assim como as definições das navegabilidades das associações, mapeamentos de persistência, entre outras coisas. Esta adequação pode ser visualizada através da Figura 15, que representa o Modelo de Entidades do SCAP.

No modelo é possível observar o tipo referente a cada atributo, se ele pode ser inicializado com o valor nulo ou não, assim como alguns mapeamentos objeto/relacionais, como por exemplo: o valor "timestamp" que indica que um atributo do tipo "Date" pode armazenar a data e a hora. Ainda é possível visualizar os tipo de dados enumerados do sistema, que estão representados na Figura 16. Os tipos enumerados utilizados neste projeto foram os mesmo que estavam presentes na versão do SCAP que foi reformulada por Prado (2015), sendo necessário acrescentar o **TipoCargo**, que pode assumir os valores PROFESSOR ou SECRETARIO. Ambos os valores podem ser utilizados na classe pessoa que é persistida no banco de dados. O atributo **TipoCargo** foi utilizado apenas como uma alternativa de visualização no banco de dados, não sendo necessário no modelo. Por meio da navegação, as informações com relação às classes Professor e Secretario podem ser encontradas facilmente.

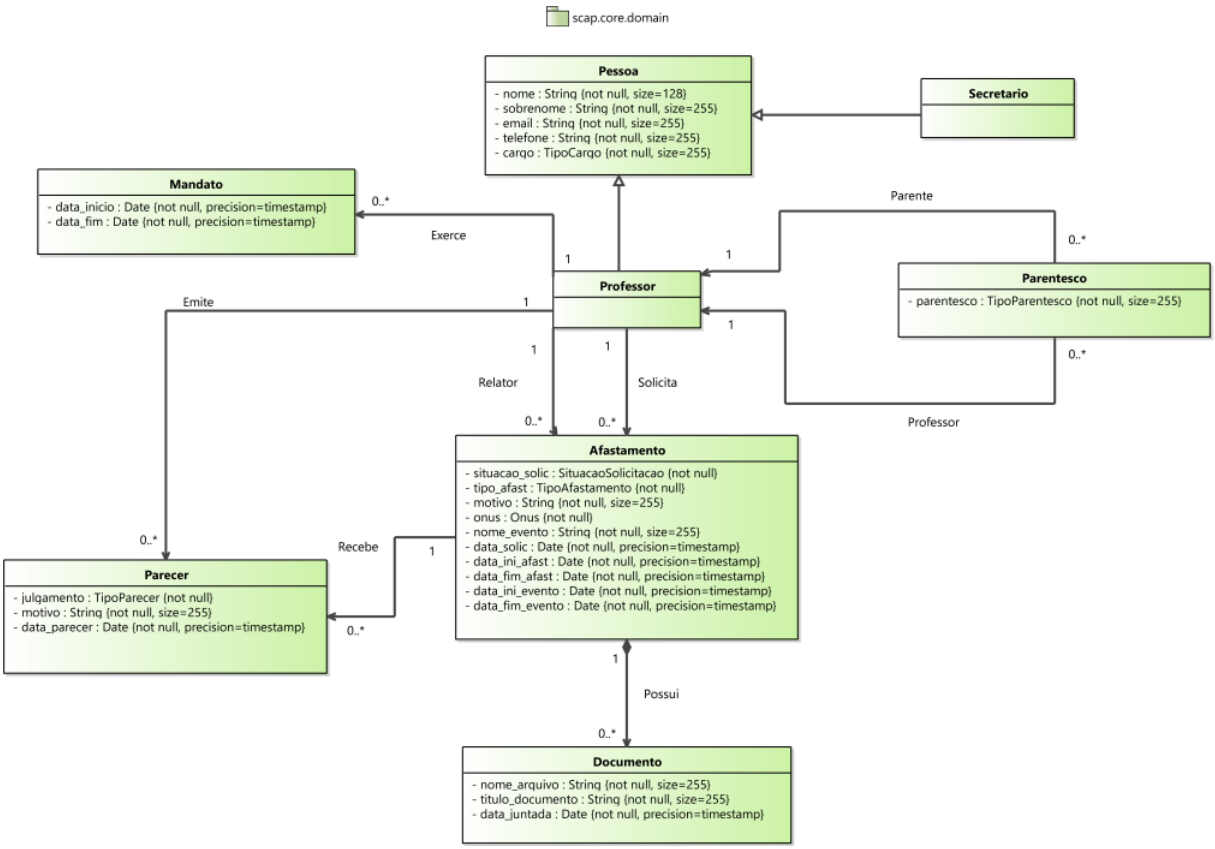

Figura 15 – Modelo de Entidades do SCAP.

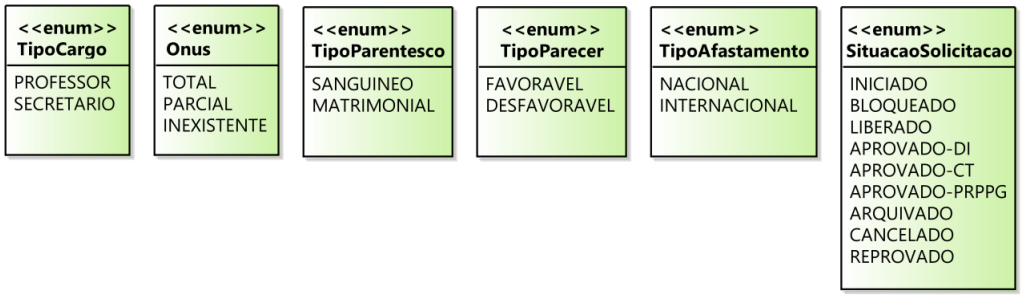

Figura 16 – Tipos Enumerados do SCAP.

4.3.3 Modelo de Aplicação

Neste projeto foi gerado um modelo de aplicação que representa quais classes de ação que dependem de quais classes de serviço e quais *Data Access Object* (DAO) foram utilizados. Portanto, este modelo pode ser visualizado por meio da Figura 17 e da Figura 18.

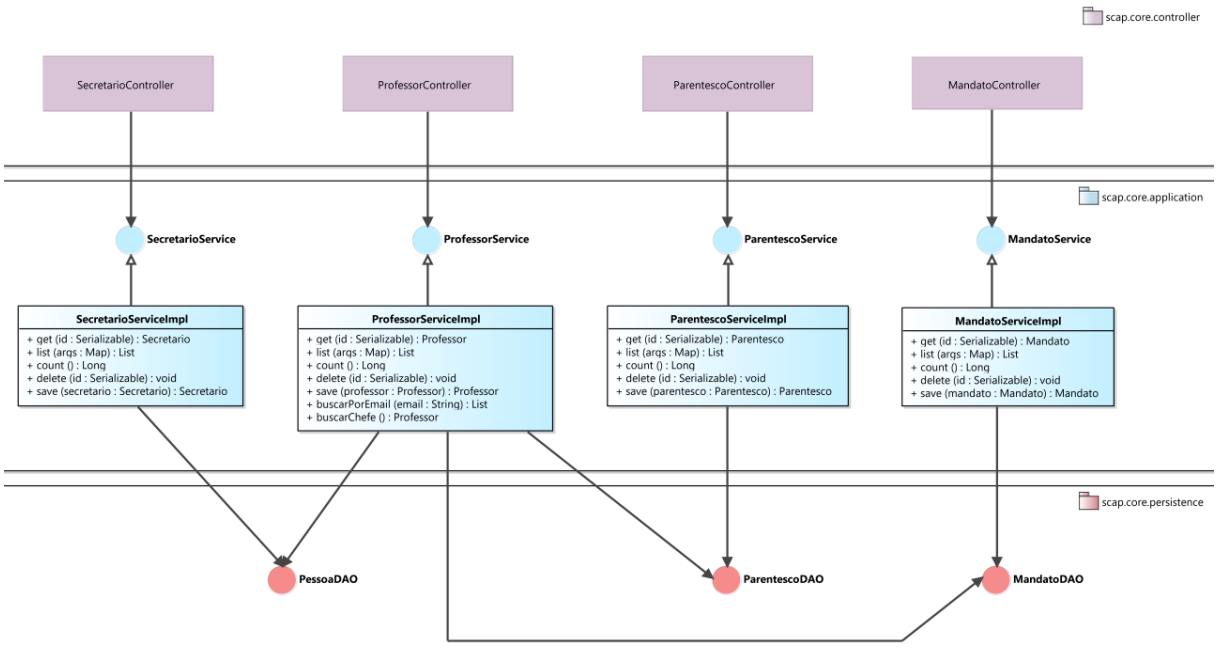

Figura 17 – Modelo de Aplicação - Parte 1.

4.3.4 Modelo de Persistência

As classes DAO que fazem parte do pacote Persistência, são construídas com o auxílio deste diagrama que pode ser visualizado por meio da Figura 19. Para cada classe de domínio que precisa de lógica de acesso a dados, foi criada uma interface e uma classe concreta DAO que realiza a implementação dessa interface. A partir da criação da interface DAO base, todas as outras interfaces herdaram as definições e as implementações concretas que foram definidas.

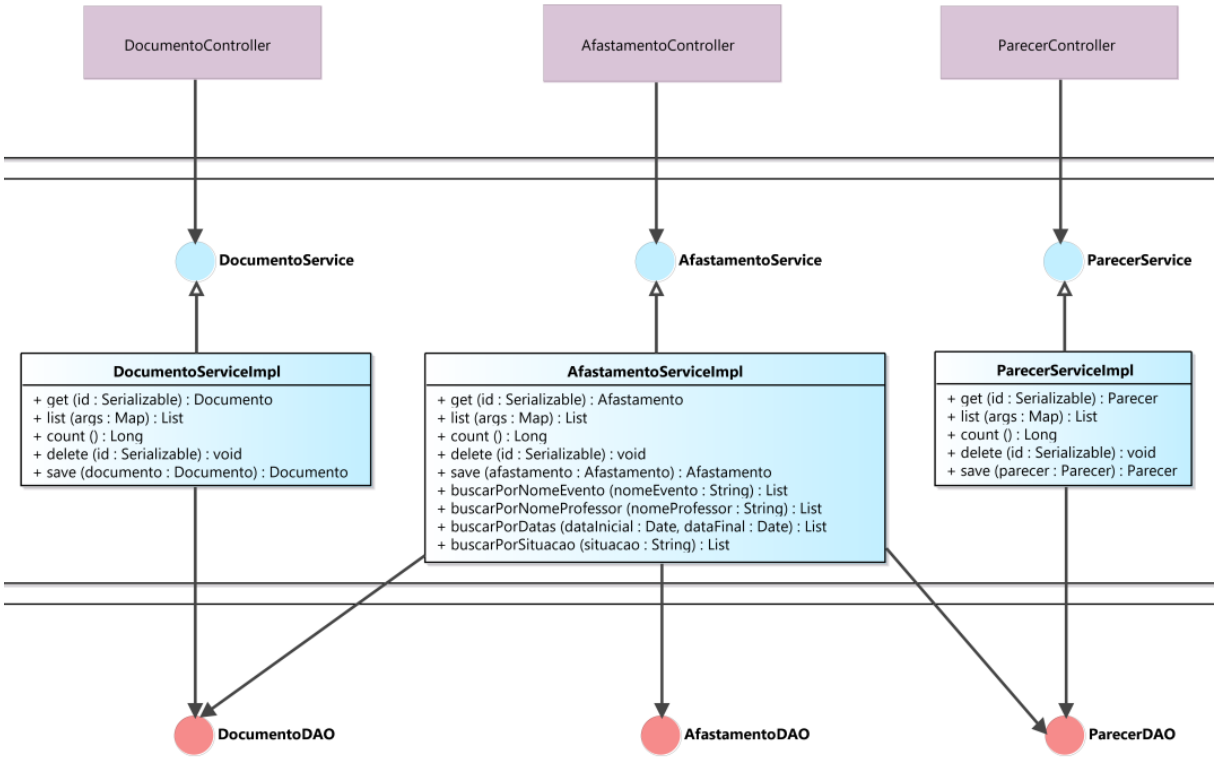

Figura 18 – Modelo de Aplicação - Parte 2.

De acordo com o método FrameWeb (SOUZA, 2007; SOUZA, 2020), é aconselhável seguir um padrão de nomes para as classes DAO, sendo que as intefaces devem seguir o padrão **<nome da classe de domínio>DAO** e as classes concretas devem seguir o padrão **<nome da interface><tecnologia de persistência>**. O *framework* Grails que foi utilizado neste projeto utiliza o **GORM** como tecnologia de persistência. A tecnologia **GORM** já foi descrita anteriormente na subseção 4.2.3.

Para evitar a poluição visual no Modelo de Persistência, não existe uma representação explícita da relação de herança entre o DAO base e os DAOs específicos. Por meio da definição de padrões, o foco é estabelecer regras para que a modelagem se torne mais simples e mais rápida.

4.4 Exibição do Sistema

O sistema possui um controle de segurança por meio da autenticação de usuários, não sendo possível acessar o conteúdo das páginas modificando a URL (*Uniform Resource Locator*) do navegador utilizado. Para ter acesso ao sistema, o usuário deve fornecer o login e a senha que são cadastrados pelos administradores do sistema. Na Figura 20 é possível visualizar a tela de login.

Por meio de regras de permissão, cada usuário visualiza a tela inicial de uma maneira modificada. Os secretários possuem a regra de permissão de administrador, pois

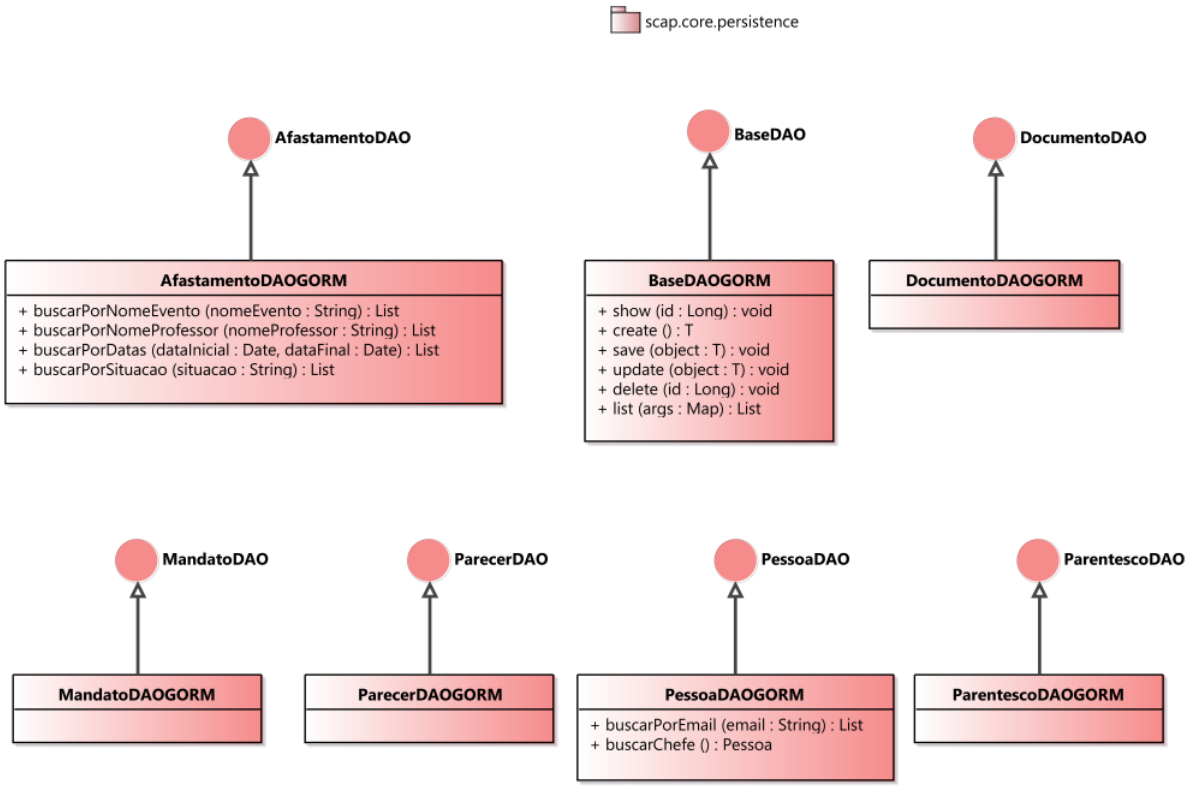

Figura 19 – Modelo de Persistência do SCAP.

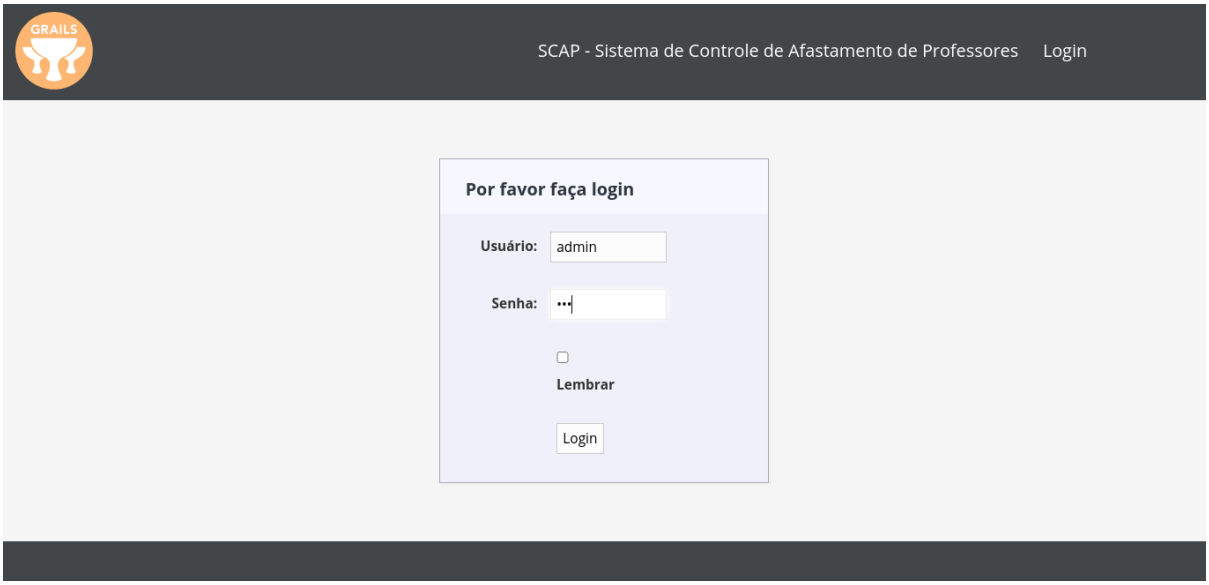

Figura 20 – Tela de Login do SCAP.

são responsáveis pelo controle da maioria das funcionalidades. Já os professores possuem a regra de permissão de usuário, podendo visualizar as funcionalidades correspondentes ao seu cargo. O professor que se tornar chefe ou subchefe do departamento, pode visualizar todas as funcionalidades referentes aos professores e ainda pode cadastrar relatores para afastamentos internacionais. A Figura 21 demonstra um usuário secretário que possui a regra de permissão de administrador. Na parte inferior da página do sistema, ficam

localizados os botões chamados de controladores. Através deles é possível realizar as ações disponíveis por cada funcionalidade, como a realização de cadastramentos e as suas devidas associações.

Figura 21 – Tela Inicial do Usuário Secretário do SCAP.

Ao clicar no botão "Professor Controller", o usuário secretário é redirecionado para a página de cadastramento de professores, pois cabe a ele cadastrar todos os professores e os parentescos entre eles. Para cadastrar os parentescos, o botão "Parentesco Controller" deve ser acionado, sendo necessário informar se o parentesco é sanguíneo ou matrimonial. Com a utilização do *scaffolding*, mencionado na Seção 4.2.1, alguns *links* foram criados. Esses *links* podem ser utilizados para a implementação de novas funcionalidades, execução de ações ou apenas ser removidos. As Figuras 22 e 23 demonstram essas funcionalidades.

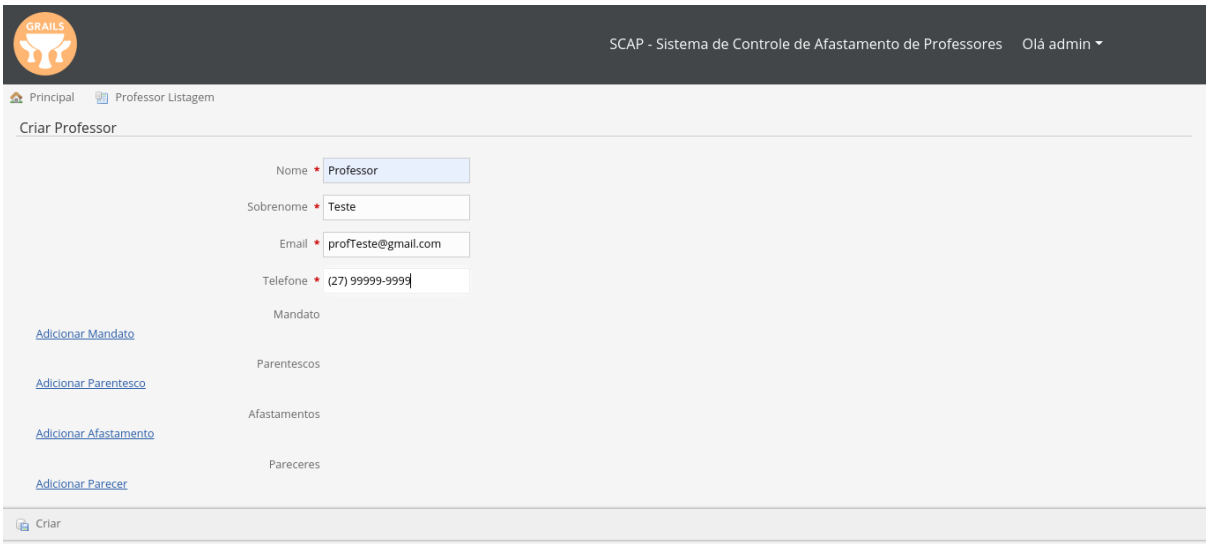

Figura 22 – Tela Cadastrar Professor do SCAP.

Figura 23 – Tela Cadastrar Parentesco do SCAP.

O usuário secretário também tem a responsabilidade de realizar o cadastramento do mandato referente ao chefe ou subchefe do departamento. Para aplicar esta ação, é necessário utilizar o botão "Mandato Controller", onde o secretário será redirecionado para a página de cadastro, sendo possível informar o período referente ao mandato e também selecionar o professor na lista de professores cadastrados. Este cadastramento pode ser visualizado através da Figura 24.

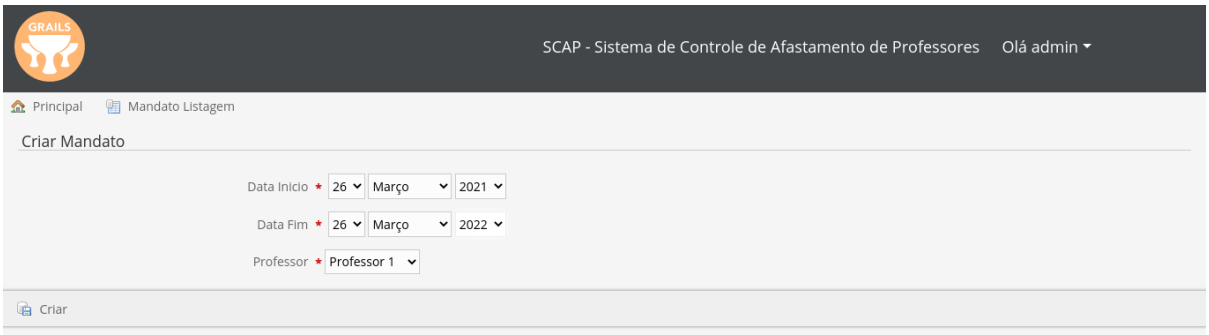

Figura 24 – Tela Cadastrar Mandato do Chefe de Departamento do SCAP.

Após o secretário realizar o cadastramento dos professores, o cadastramento dos parentescos e o cadastramento do mandato referente ao chefe ou subchefe do departamento, os professores recebem o acesso ao sistema. Por possuírem a regra de permissão de usuário, os professores visualizam a tela inicial com as funcionalidades reduzidas. A tela inicial do usuário professor pode ser visualizada através da Figura 25.

A Figura 26 demonstra a principal funcionalidade do sistema. Ao clicar no botão "Afastamento Controller" o professor pode preencher todas as informações necessárias para realizar uma solicitação de afastamento, iniciando todo o processo para que as tramitações sejam aplicadas. Através do botão "Documento Controller", é possível cadastrar documentos para que eles sejam associados ao afastamento. Está funcionalidade é demonstrada por meio da Figura 27.

O cadastro de um parecer é exemplificado pela Figura 28. Utilizando o botão "Parecer Controller", o professor que foi cadastrado pelo chefe do departamento como sendo relator de uma solicitação de afastamento internacional, pode preencher as informações

Figura 25 – Tela Inicial do Usuário Professor do SCAP.

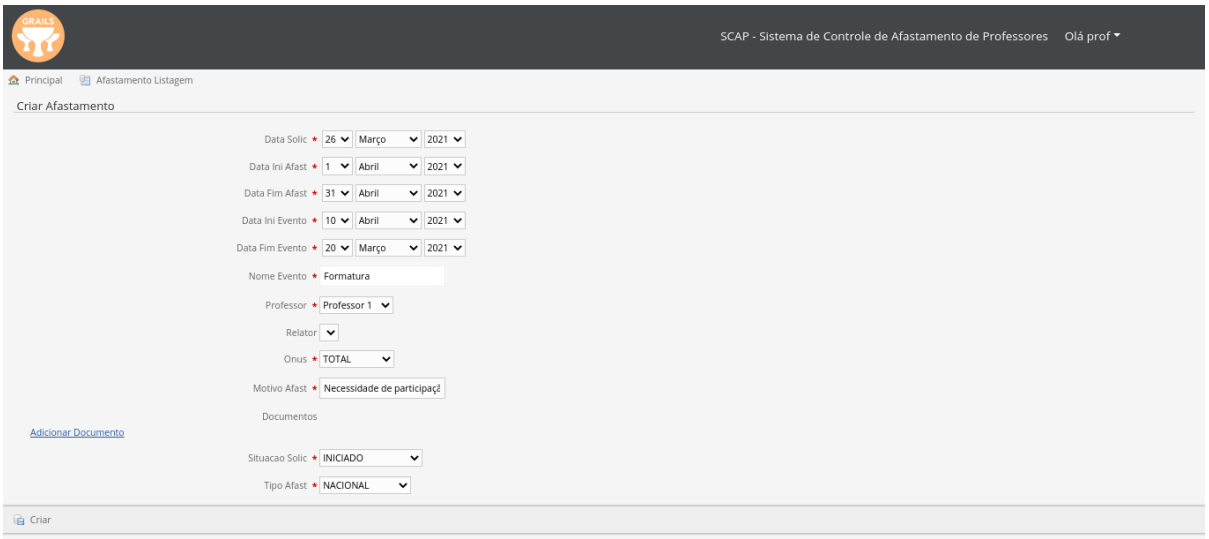

Figura 26 – Tela Cadastrar Afastamento do SCAP.

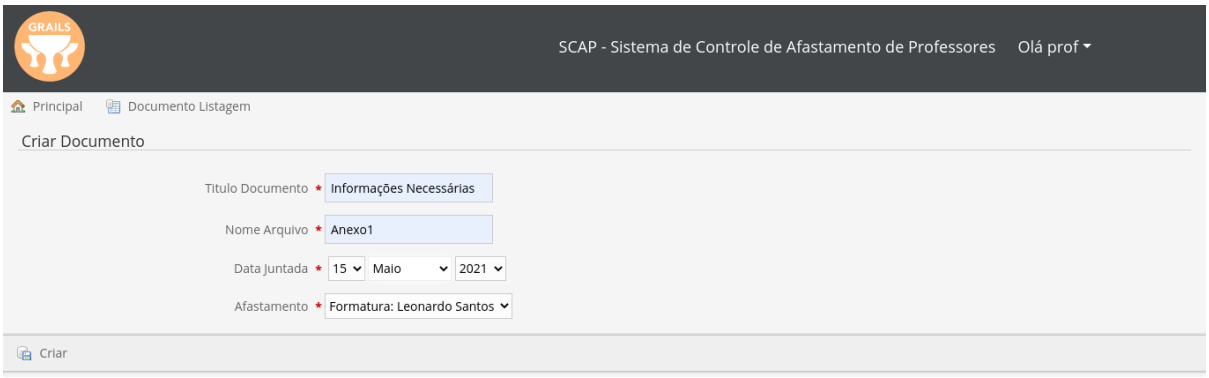

Figura 27 – Tela Cadastrar Documento de um Afastamento do SCAP.

justificando o motivo da sua decisão. Caso um professor seja contra a alguma solicitação

de afastamento, ele também pode cadastrar um parecer.

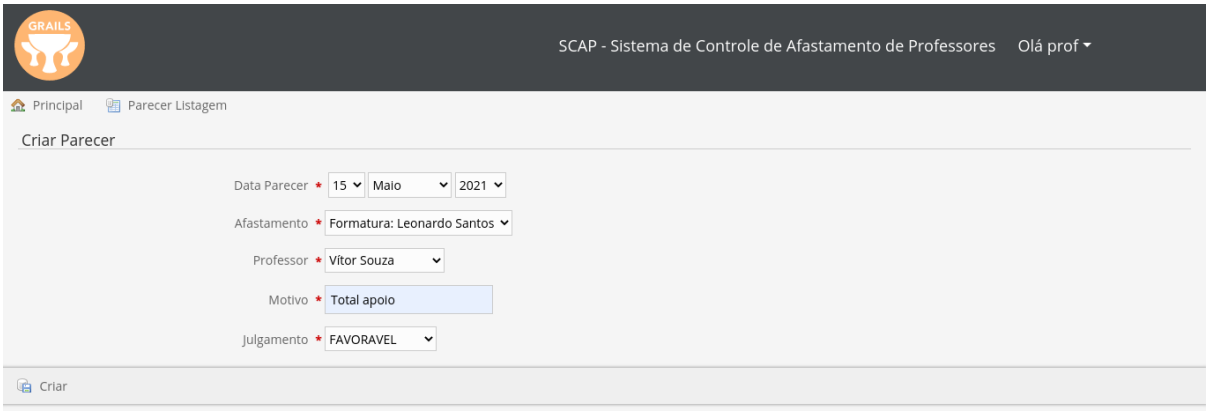

Figura 28 – Tela Cadastrar Parecer de um Afastamento do SCAP.

Todos os usuários do sistema podem realizar consultas para obterem informações sobre uma solicitação de afastamento. Os usuários podem realizar buscas pelo nome do professor, pelo nome do evento, pela situação do afastamento ou pelo período de data de solicitação. É possível visualizar uma lista com todos os afastamentos cadastrados no sistema e também obter detalhes de um afastamento específico. A Figura 29 apresenta a lista contendo todos os afastamentos e a Figura 30 exibe os detalhes do afastamento selecionado.

Figura 29 – Tela Listar Afastamentos do SCAP.

A Figura 31 demonstra mais uma funcionalidade disponível para o usuário secretário. Para melhorar o controle, é possível visualizar uma lista contendo todos os professores cadastrados, assim como os seus respectivos parentescos. Além disso, é possível saber de todas as solicitações de afastamentos realizadas por cada professor e realizar uma busca por email.

Após as tramitações de um afastamento serem realizada, o usuário secretário deve efetuar uma atualização no afastamento cadastrado, fazendo uma modificação no status do mesmo. Para isso, ele deve editar o afastamento, mudando o campo “Situacao Solic” para “Arquivado”. Por fim, a Figura 32 exemplifica esta ação.

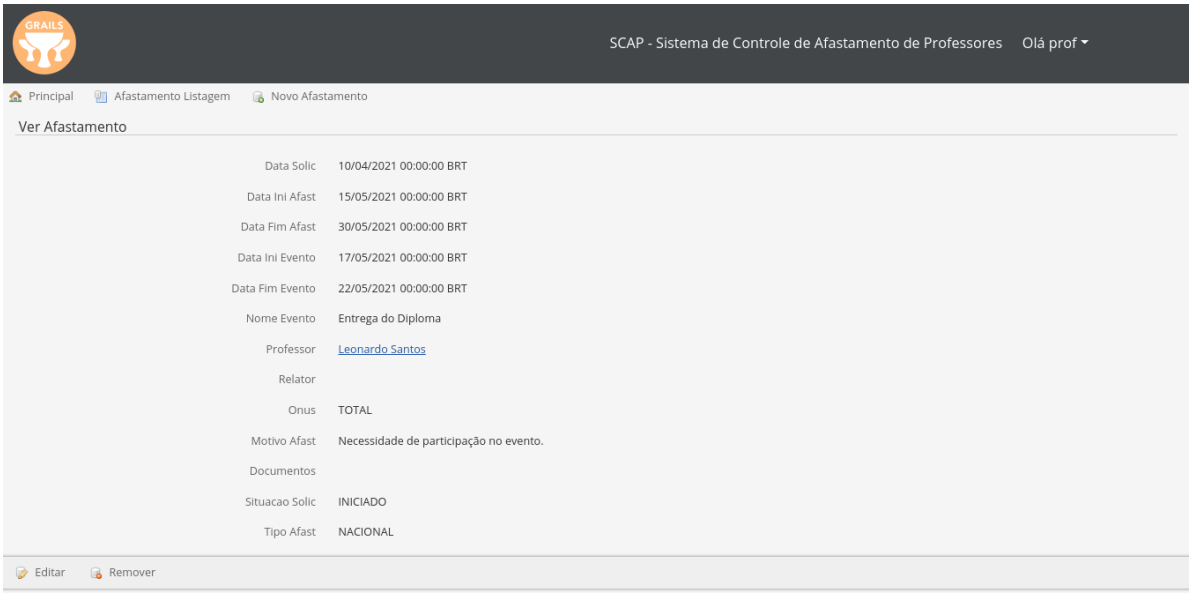

Figura 30 – Tela Visualizar Afastamento do SCAP.

Figura 31 – Tela Listar Professores do SCAP.

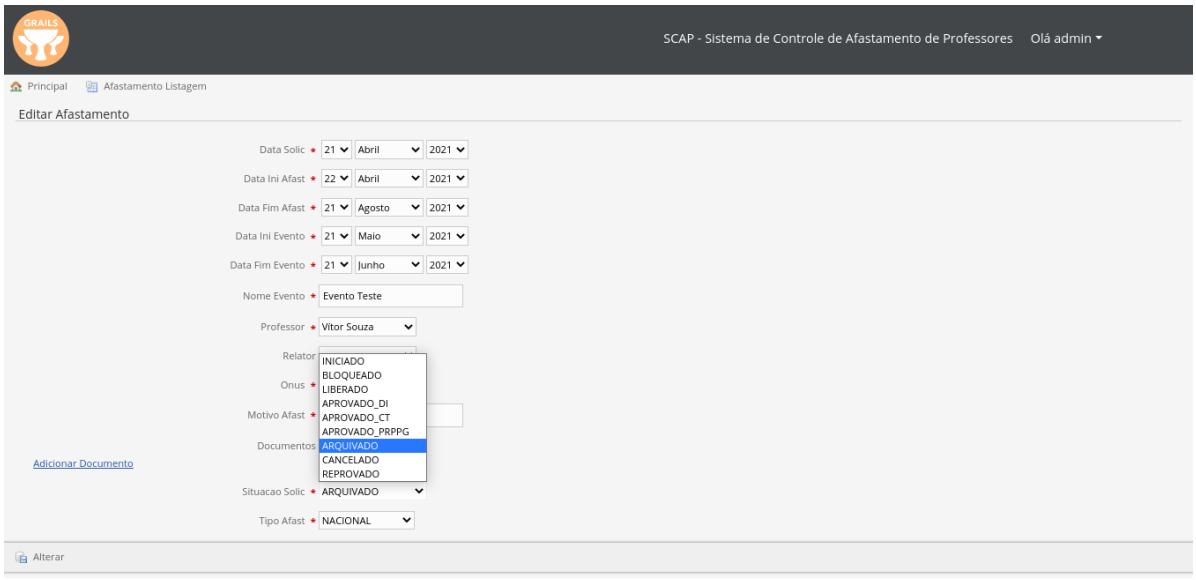

Figura 32 – Tela Editar Afastamento do SCAP.

O código-fonte implementado neste trabalho está disponível em <https://bitbucket.org/vitorsouza-students/pg-2019-leonardo-santos/src/master/code/>.

5 Considerações Finais

Este capítulo apresenta as conclusões que foram obtidas por meio da aplicação do método FrameWeb, o uso da ferramenta FrameWeb Editor (CAMPOS; SOUZA, 2017) para a geração dos modelos, a utilização do *framework* Grails, o aprendizado da linguagem Apache Groovy, os problemas enfrentados e, por fim, sugestões para trabalhos futuros.

5.1 Conclusões

Para que os objetivos fossem alcançados, foi preciso combinar todo o conhecimento adquirido durante o curso, reunindo o aprendizado obtido nas disciplinas de Banco de Dados, Linguagens de Programação, Engenharia de *Software*, Desenvolvimento *Web* e *Web* Semântica, entre outras. Por meio deste conhecimento, foi possível estabelecer a melhor estratégia para que os problemas enfrentados durante o projeto fossem sanados, gerando soluções que se tornaram bastante satisfatórias.

A versão do sistema SCAP descrita no decorrer deste projeto foi implementada utilizando a linguagem Apache Groovy, que ainda não tinha sido utilizada em versões anteriores. Por este motivo, é importante ressaltar que não foi utilizado nenhum código que estava presente nas versões passadas. Assim, foi necessário realizar o estudo e aprendizagem da linguagem Groovy e do *framework* Grails para que eles pudessem ser utilizados. O aprendizado de uma nova linguagem é um desafio que proporciona um retorno muito grande, agregando muita experiência que pode ser aplicada em diversas áreas.

Com a utilização de um *framework*, a implementação do sistema SCAP se tornou uma tarefa bem mais simples do que se fosse necessário implementar todo o código sem ter uma base de início. Além de fornecer a geração de boa parte do código necessário para o funcionamento do sistema, o uso de um *framework* ainda proporciona a disponibilização de diversas bibliotecas que podem ser incluídas no projeto a qualquer momento, diminuindo o tempo e as complexidades do desenvolvimento.

Durante o aprendizado para a utilização do *framework* Grails, algumas dificuldades foram enfrentadas. Apesar do *framework* disponibilizar uma rica documentação, alguns fatores são explicados de forma superficial, como por exemplo, a conexão com um banco de dados relacional. Para essa tarefa foi necessário recorrer aos fóruns da comunidade do *framework*, assim como vídeos tutoriais encontrados na Internet. Outra dificuldade enfrentada foi a incompatibilidade entre as versões do *framework*. Basta atualizar a versão e o projeto deixa de funcionar, sendo necessário realizar muitas alterações para que os erros sejam solucionados.

Depois que Souza (2007) estabeleceu uma organização para os *frameworks* em categorias diferentes, realizar a escolha de um *framework* que se enquadre na execução das propostas de um projeto se tornou uma tarefa bem mais fácil. Com a aplicação do método FrameWeb na fase de projeto arquitetural, a criação de modelos de projeto se aproxima muito da implementação do sistema, definindo uma arquitetura básica e deixando os desenvolvedores livres para adotarem as técnicas mais adequadas para as outras etapas do processo.

Para a criação dos modelos que são propostos pelo método FrameWeb, foi utilizada a ferramenta FrameWeb Editor (CAMPOS; SOUZA, 2017). Nesta etapa, foi necessário realizar a instalação e configuração da ferramenta, seguindo o tutorial (SOUZA, 2021) referente a ela. As configurações são um pouco extensas e devem ser realizadas com bastante atenção para evitar possíveis erros. No tutorial também são listadas algumas dicas para resolver erros que podem aparecer durante as configurações.

A utilização do FrameWeb Editor não é muito intuitivo no início, mas com relação às funcionalidades, a ferramenta cumpre o seu propósito. É possível realizar a criação dos quatro tipos básicos de modelos FrameWeb, fazendo verificações nos modelos e impedindo ações inválidas. Como toda ferramenta criada recentemente, ainda é necessário realizar correções em alguns erros, como por exemplo, o salvamento das associações entre os elementos contidos no modelo de navegação. Com o lançamento de novas versões da ferramenta, certamente esses erros serão solucionados.

Realizando a aplicação do método FrameWeb, claramente a sua eficiência e eficácia ficaram comprovadas. Por meio dos modelos que foram gerados, a implementação do código do sistema através do *framework* Grails foi agilizada, sendo necessário apenas seguir com as etapas do projeto. Com a divisão do sistema em camadas, a organização e a modularização tornaram os componentes do SCAP bem mais fáceis de serem aperfeiçoados ou substituídos.

Em termos gerais, pertencente a categoria de Controlador Frontal, o *framework* Grails foi considerado uma boa escolha para o desenvolvimento de aplicações *Web*. Assim, como contribuição para o FrameWeb, ele pode ser incluído na lista de *frameworks* que o método oferece suporte.

5.2 Trabalhos Futuros

Existem muitos cenários onde um aplicativo *mobile* traz melhor experiência para o usuário e satisfaz melhor suas expectativas, sobretudo com relação à usabilidade. Uma boa estratégia é começar sempre pela versão *Web*, garantindo o acesso universal e multi-plataforma. Conforme as necessidades forem surgindo, é possível investir em aplicativos específicos de plataformas com recursos e experiências nativas (LOPES, 2014).

De acordo com Lopes (2014), um aplicativo *mobile* tem acesso direto ao hardware do aparelho e a recursos do sistema operacional. Consegue se integrar com funções avançadas e a outros aplicativos. Pode manipular o funcionamento do aparelho e até substituir ou complementar funções nativas. Já uma *WebApp* roda enjaulada dentro do navegador e, por razões de segurança, não tem acesso direto à plataforma nativa.

Por estes motivos, como sugestão para trabalhos futuros, seria interessante realizar o desenvolvimento de um aplicativo *mobile* que representasse o sistema SCAP. Após a aplicação do método FrameWeb para a construção de novos modelos, o próximo *framework* utilizado ficaria responsável por controlar o *back-end* da aplicação. Reunindo todas as funcionalidades, a usabilidade certamente seria ampliada, gerando pontos positivos para os seus usuários.

A partir dos modelos FrameWeb construídos por meio do uso da ferramenta FrameWeb Editor, ainda é possível utilizar outro recurso, o gerador de código (ALMEIDA; CAMPOS; SOUZA, 2017). Para utilizar este recurso, é necessário que a arquitetura esteja definida, ou seja, é necessário realizar a importação do conjunto de arquivos que definem a arquitetura do projeto que está sendo modelado. Um arquivo de definição de linguagem e alguns arquivos de definição de *framework* devem ser importados para o projeto. Assim, por meio desses arquivos, o código pode ser gerado facilmente.

A definição de arquitetura só necessita ser realizada caso ela não seja atualmente suportada pelas ferramentas FrameWeb. Para o *framework* Grails, a definição da arquitetura começou a ser efetuada, mas devido as restrições de tempo para a conclusão desse trabalho, ela teve que ser descartada. Portanto, outra sugestão para trabalhos futuros, seria a criação de *templates* para o *framework* Grails e para a linguagem Groovy, realizando a geração do código e a comparação com a implementação atual.

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 2 vezes nas páginas 26 e 56.

ALUR, D.; CRUPI, J.; MALKS, D. *Core J2EE Patterns: Best Practices and Design Strategies*. 2. ed. [S.l.]: Pearson, 2003. 650 p. ISBN 0-131-42246-4. Citado 2 vezes nas páginas 24 e 28.

AMUI, S. F. *Processos de desenvolvimento de software*. [S.l.]: SESES, 2015. 176 p. ISBN 978-85-5548-040-9. Citado na página 19.

BAUER, C.; KING, G. *Java Persistence with Hibernate*. 1. ed. [S.l.]: Manning Publications, 2007. 841 p. ISBN 1-932-39488-5. Citado na página 24.

BEDER, D. M. *Engenharia Web: Uma abordagem sistemática para o desenvolvimento de aplicações web*. [S.l.]: UAB-UFSCar, 2012. 213 p. ISBN 978-85-7600-290-1. Citado 4 vezes nas páginas 14, 21, 35 e 37.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. *UML: Guia do Usuário*. [S.l.]: Elsevier Editora Ltda, 2006. 89 p. ISBN 13 978-85-352-1784-1. Citado na página 14.

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 26, 38, 54 e 55.

DUARTE, B. B. *Aplicação do Método FrameWeb no Desenvolvimento de um Sistema de Informação na Plataforma Java EE 7*. Vitória, ES, Brasil, 2014. Citado 6 vezes nas páginas 10, 15, 16, 30, 31 e 34.

FALBO, R. d. A. *Engenharia de Software*. [s.n.], 2014. 144 p. Disponível em: <https://inf.ufes.br/~falbo/files/ES/Notas_Aula_Engenharia_Software.pdf>. Citado 2 vezes nas páginas 14 e 16.

FALBO, R. d. A. *Engenharia de Requisitos*. [s.n.], 2017. 178 p. Disponível em: <https://www.inf.ufes.br/~falbo/files/ER/Notas_Aula_Engenharia_Requisitos.pdf>. Citado 3 vezes nas páginas 16, 18 e 19.

FALBO, R. d. A. *Projeto de Sistemas de Software*. [s.n.], 2018. 131 p. Disponível em: <https://www.inf.ufes.br/~falbo/files/PSS/Notas_Aula_Projeto_Sistemas_2018.pdf>. Citado na página 16.

FOWLER, M. *Patterns of Enterprise Application Architecture*. [S.l.]: Addison-Wesley Professional, 2002. 560 p. ISBN 0-321-12742-0. Citado 4 vezes nas páginas 8, 23, 24 e 28.

GUTERRES, C. S. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o framework Play*. Vitória, ES, Brasil, 2019. Nenhuma citação no texto.

KÖNIG, D. et al. *Groovy in Action*. 1. ed. [S.l.]: Manning, 2007. 659 p. ISBN 1-932-39484-2. Citado na página 37.

LOPES, S. *A Web Mobile: Design Responsivo e além para uma Web adaptada ao mundo mobile*. 1. ed. [S.l.]: Casa do Código, 2014. 308 p. ISBN 978-85-66250-23-7. Citado 2 vezes nas páginas 55 e 56.

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 na página 25.

MATTSSON, M.; BOSCH, J. Characterizing Stability in Evolving Frameworks. *Proceedings of the 29th International Conference on Technology of ObjectOriented Languages and Systems. doi: 10.1109/1999.779005*, TOOLS EUROPE, n. 12, p. 118–130, 1999. Citado na página 26.

OFFUTT, J. Quality Attributes of Web Software Applications. *Engineering Internet Software. doi: 10.1109/52.991329*, IEEE Software, n. 8, p. 25–32, 2002. Citado na página 21.

OLIVÉ, A. *Conceptual Modeling of Information Systems*. 7. ed. [S.l.]: Springer, 2007. 455 p. Citado na página 19.

PERUCH, L. A. *Aplicação e Análise do Método FrameWeb com Diferentes Frameworks Web*. Vitória, ES, Brasil, 2007. Citado na página 20.

PRADO, R. C. do. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework VRaptor 4*. Vitória, ES, Brasil, 2015. Citado 6 vezes nas páginas 15, 16, 30, 31, 34 e 45.

PRESSMAN, R. S. *Engenharia de Software: Uma Abordagem Profissional*. 7. ed. [S.l.]: AMGH Editora Ltda, 2011. 771 p. ISBN 978-85-8055-044-3. Citado 7 vezes nas páginas 8, 14, 20, 22, 25, 27 e 35.

SOMMERVILLE, I. *Engenharia de Software*. 8. ed. [S.l.]: Pearson Universidades, 2007. 568 p. ISBN 8-588-63928-9. Citado na página 18.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. [S.l.]: Pearson Universidades, 2011. 548 p. ISBN 8-579-36108-7. Citado na página 20.

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, 14, 16, 23, 25, 26, 28, 47 e 55.

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 4 vezes nas páginas 14, 16, 23 e 47.

SOUZA, V. E. S. *FrameWeb Tools Tutorial*. 2021. Disponível em: <https://github.com/nemo-ufes/FrameWeb/wiki/ToolsTutorial01>. Citado na página 55.

VAZQUEZ, C. E.; SIMÕES, G. S. *Engenharia de Requisitos: Software Orientado ao Negócio*. 1. ed. [S.l.]: Brasport, 2016. 328 p. ISBN 8-574-52790-4. Citado na página 19.

WEISSMANN, H. L. *Falando de Grails. Altíssima Produtividade no Desenvolvimento Web*. 1. ed. [S.l.]: Casa do Código, 2015. 411 p. ISBN 8-555-19025-8. Citado 2 vezes nas páginas 29 e 38.

Apêndices

Documento de Projeto de Sistema

SCAP — Sistema de Controle de Afastamento de Professores

Registro de Alterações:

Versão	Responsável	Data	Alterações
1.0	Leonardo Nascimento dos Santos	13/03/2019	Versão Inicial
1.1	Leonardo Nascimento dos Santos	21/03/2021	Modelos FrameWeb adicionados
1.2	Leonardo Nascimento dos Santos	21/04/2021	Correções dos modelos

Vitória, ES

2021

1 Introdução

Este documento apresenta o documento de projeto (*design*) arquitetural do sistema SCAP — Sistema de Controle de Afastamento de Professores. Este documento está organizado da seguinte forma: a Seção 2 apresenta a plataforma de software utilizada na implementação da ferramenta; a Seção 3 trata de táticas utilizadas para tratar requisitos não funcionais (atributos de qualidade); por fim, a Seção 4 apresenta o projeto da arquitetura de software e suas subseções explicam cada uma de suas camadas.

2 Plataforma de Desenvolvimento

Na Tabela 1 são listadas as tecnologias utilizadas no desenvolvimento da ferramenta, bem como o propósito de sua utilização.

Tabela 1 – Plataforma de Desenvolvimento e Tecnologias Utilizadas

Tecnologia	Versão	Descrição	Propósito
Apache Groovy	4.0	Linguagem de programação multifacetada, opcionalmente dinâmica e tipada criada para a plataforma Java.	Escrita do código-fonte das classes que compõem o sistema.
Hibernate	5.1.5	Ferramenta de persistência e consulta objeto/relacional de alta performance.	Mapeamento de classes para tabelas de banco de dados.
Spring *Framework*	4.3.9	*Framework* Java que possui o objetivo de facilitar o desenvolvimento de aplicações.	Injeção de dependências e inversão de controle.
Spring Boot	1.5.4	Ferramenta que agiliza o processo de configuração e publicação de aplicações baseadas em Spring.	Configuração e disponibilização de aplicações.
Gradle	3.5	Sistema de automatização de *builds*.	Geração de arquivos de *build* na linguagem Groovy.
Spock	1.1	*Framework* de testes e especificações para aplicativos.	Realização de testes.
Apache Tomcat	9.0.19	Servidor de Aplicações para Java EE.	Suporte a execução das tecnologias de *Servlets*, JDBC DataSourcesRealms, JNDI *Resources* e JSP, cobrindo a parte da especificação J2EE permitindo que o Java funcione no modo web.
GORM	7.0	Kit de ferramentas de acesso a dados.	Distribuição de conjunto de APIs para acessar dados relacionais e não relacionais.
Grails *Framework*	3.3.9	*Framework* para a construção de aplicações para a *web* através da linguagem de programação Groovy.	Fornecimento de alta produtividade no desenvolvimento de aplicações *Web*, através da utilização do paradigma da programação por convenção.

Na Tabela 2 vemos os softwares que apoiaram o desenvolvimento de documentos e também do código fonte.

Tabela 2 – Softwares de Apoio ao Desenvolvimento do Projeto

Tecnologia	Versão	Descrição	Propósito
FrameWeb Editor	1.0	Ferramenta CASE do método FrameWeb.	Criação dos modelos de Entidades, Aplicação, Persistência e Navegação.
TeX Live	2018	Implementação do LaTeX	Documentação do projeto arquitetural do sistema.
Texmaker	5.0.2	Editor de LaTeX.	Escrita da documentação do sistema, sendo usado o *template abnTeX*.[1]
phpMyAdmin	5.1.0	Ferramenta de software livre para administrar banco de dados.	Administração de banco de dados.

3 Atributos de Qualidade e Táticas

Na Tabela 3 são listados os atributos de qualidade considerados neste projeto, com uma indicação se os mesmos são condutores da arquitetura ou não e as táticas a serem utilizadas para tratá-los.

Tabela 3 – Atributos de Qualidade e Táticas Utilizadas

Categoria	Requisitos Não Funcionais	Condutor da Arquitetura	Tática
Facilidade de Aprendizado, Operabilidade	RNF02, RNF03	Sim	• Facilitar a compreensão do usuário para que os conceitos chaves sejam entendidos de forma intuitiva, possibilitando que os comandos do sistema sejam operados com mais eficiência. Para isso, o sistema deve possuir diálogos simples para que nenhuma dúvida seja gerada no momento da inserção das informações.
Segurança de Acesso	RNF01	Sim	• Garantir que os dados estão acessíveis apenas para aqueles que estão autorizados a acessá-los, evitando acesso não autorizado ou modificação dos dados. • Autorizar usuários, usando as classes de usuários definidas no projeto.
Manutenibilidade, Portabilidade	RNF04, RNF05, RNF06, RNF07	Sim	• Organizar a arquitetura da ferramenta segundo uma combinação de camadas e partições. • A camada de lógica de negócio deve ser organizada segundo o padrão Camada de Serviço. • A camada de gerência de dados deve ser organizada segundo o padrão DAO. • Separar a interface com o usuário do restante da aplicação, segundo o padrão MVC.
Reusabilidade	RNF07	Não	• Reutilizar componentes e frameworks existentes. • Desenvolver novos componentes para reuso, quando não houver componentes disponíveis e houver potencial para reuso.

4 Arquitetura de Software

A arquitetura de software do sistema SCAP — Sistema de Controle de Afastamento de Professores segue a arquitetura padrão sugerida pelo FrameWeb (SOUZA, 2007; SOUZA; FALBO; GUIZZARDI, 2009) baseada no padrão Camada de Serviço (FOWLER, 2002). A Figura 1 ilustra a arquitetura estabelecida pelo método FrameWeb.

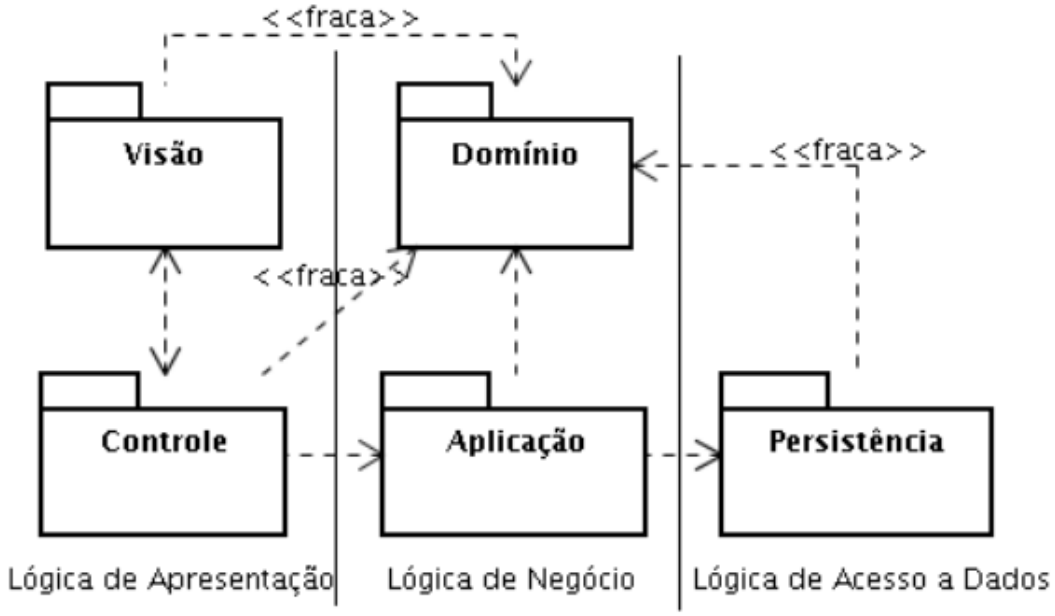

Figura 1 – Arquitetura padrão proposta pelo FrameWeb (SOUZA, 2007).

Nas próximas seções, serão apresentados diagramas FrameWeb relativos a cada uma das camadas da arquitetura do sistema.

4.1 Camada de Apresentação

Possui a responsabilidade de realizar a interação entre o sistema e o usuário, exibindo as informações e interpretando os comandos em ações da persistência de dados e da lógica de negócio. Apresenta os modelos de navegação, que auxiliam os desenvolvedores na implementação dos componentes e das classes dos pacotes Visão e Controle.

Os Modelos de Navegação foram gerados de acordo com os casos de uso do sistema. A Figura 2 e a Figura 3 representam o caso de uso “Cadastrar Usuário”, que é realizado por um secretário. O caso de uso “Cadastrar Chefe do Departamento” também é utilizado por um secretário e foi representado através da Figura 4.

A Figura 5 representa o caso de uso “Solicitar Afastamento”, que acontece quando um professor realiza o cadastro de um pedido de afastamento. Caso exista um pedido de

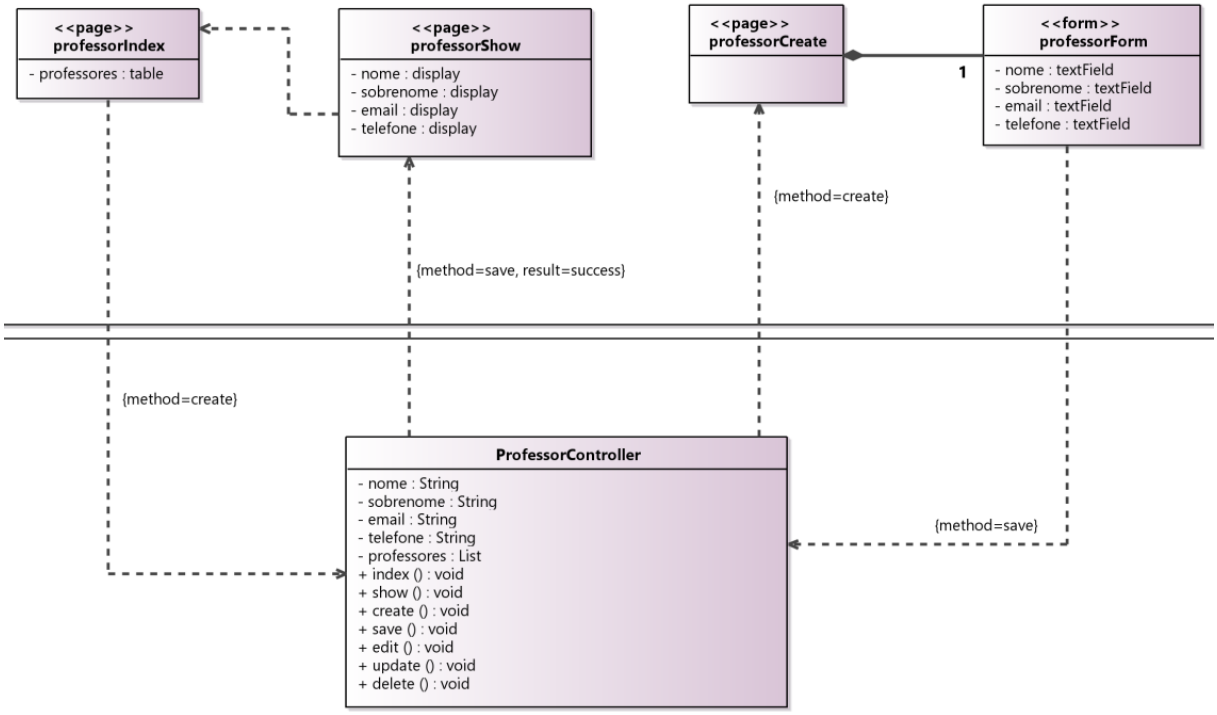

Figura 2 – Modelo de Navegação do Caso de Uso: Cadastrar Usuário - Professor.

afastamento internacional, o chefe do departamento utiliza o caso de uso “Encaminhar Afastamento”, que pode ser visualizado na Figura 6.

Os casos de uso “Deferir Parecer” e “Manifestar-se Contra Afastamento” são bem parecidos, pois é necessário realizar o cadastro de um parecer. Portanto, eles foram representados através da Figura 7.

Os casos de uso “Cancelar Afastamento”, “Arquivar Afastamento”, “Registrar Parecer CT” e “Registrar Parecer PRPPG” também são bem parecidos. Um professor cancela um pedido de afastamento realizando a alteração da situação da solicitação. Do mesmo modo, um secretário realiza a mudança da situação da solicitação, quando necessita arquivar um afastamento, registrar um parecer do CT ou registrar um parecer da PRPPG. Assim, esses casos de uso foram apresentados na Figura 8.

Todos os usuários do sistema podem utilizar o caso de uso “Consultar Afastamento”. Para isso, é possível filtrar a lista de afastamentos através de quatro buscas diferentes. O caso de uso está representado através da Figura 9.

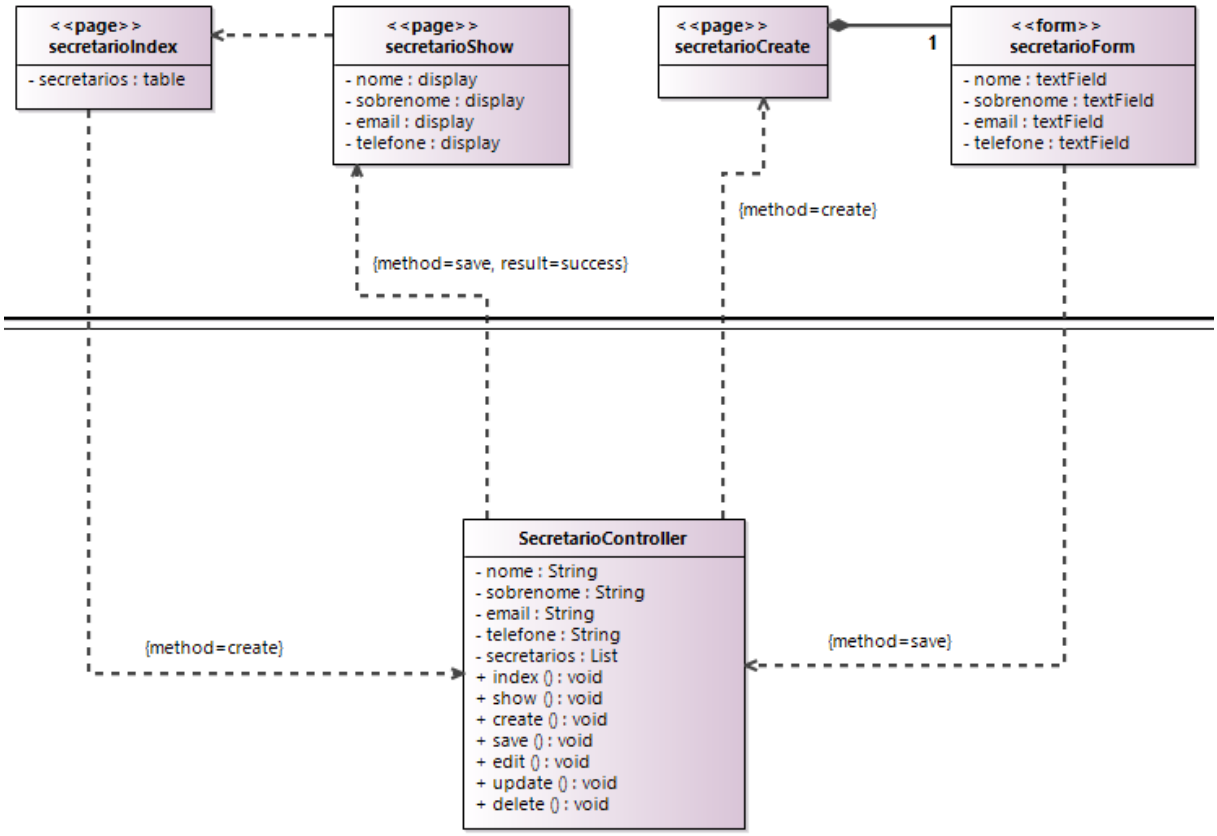

Figura 3 – Modelo de Navegação do Caso de Uso: Cadastrar Usuário - Secretário.

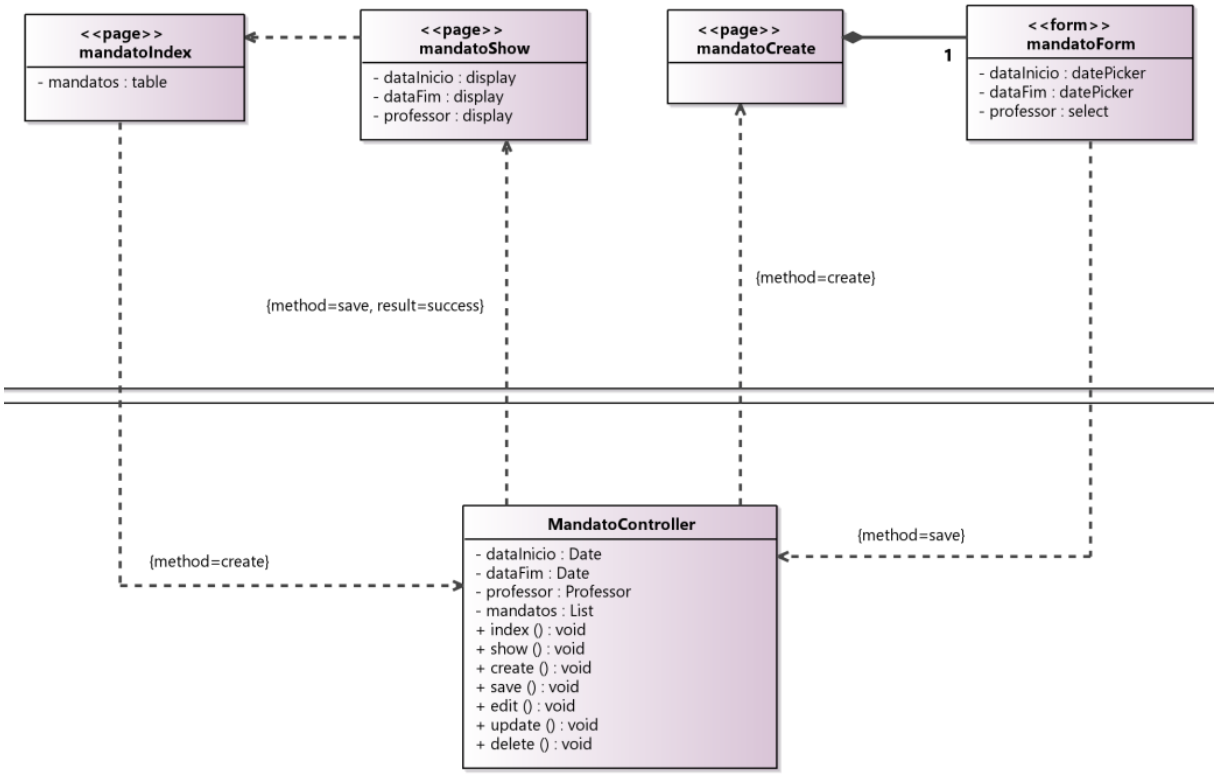

Figura 4 – Modelo de Navegação do Caso de Uso: Cadastrar Chefe do Departamento.

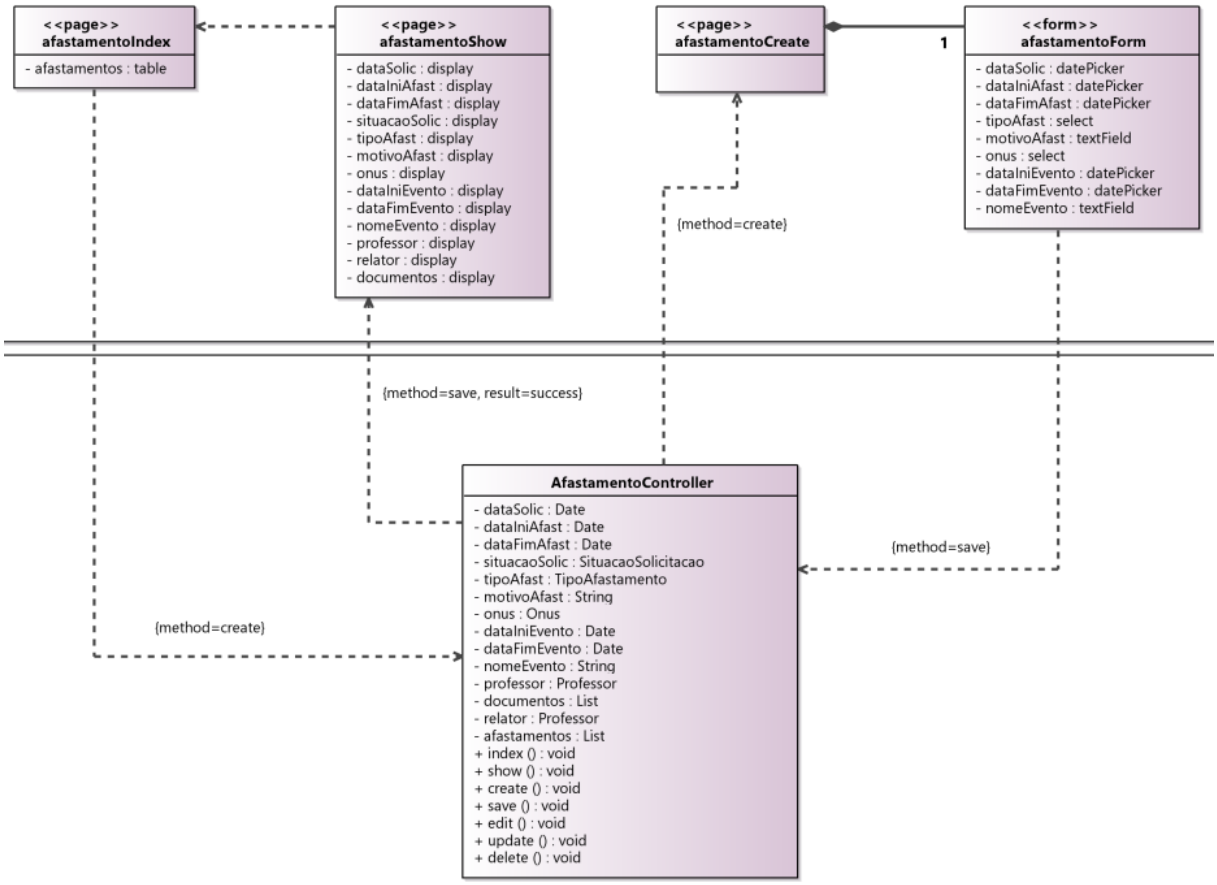

Figura 5 – Modelo de Navegação do Caso de Uso: Solicitar Afastamento.

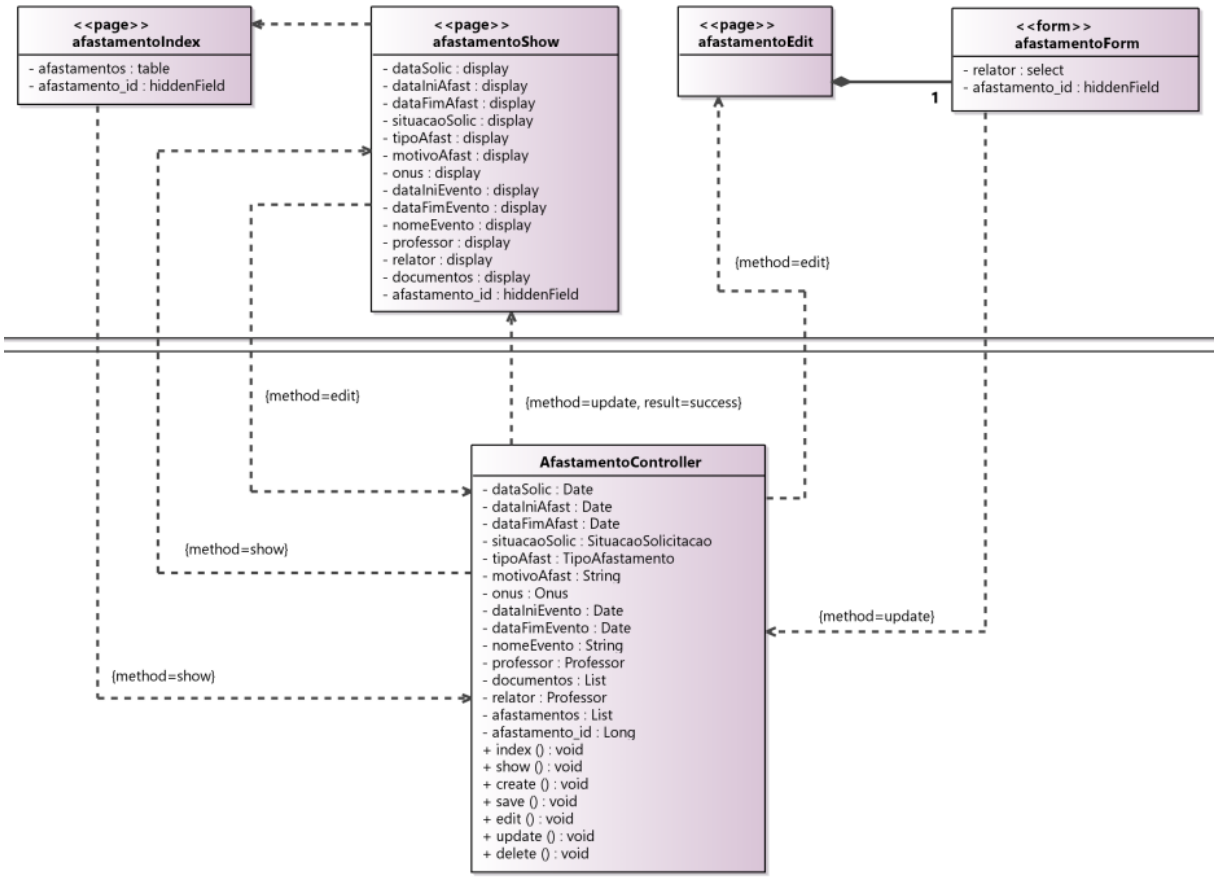

Figura 6 – Modelo de Navegação do Caso de Uso: Encaminhar Afastamento.

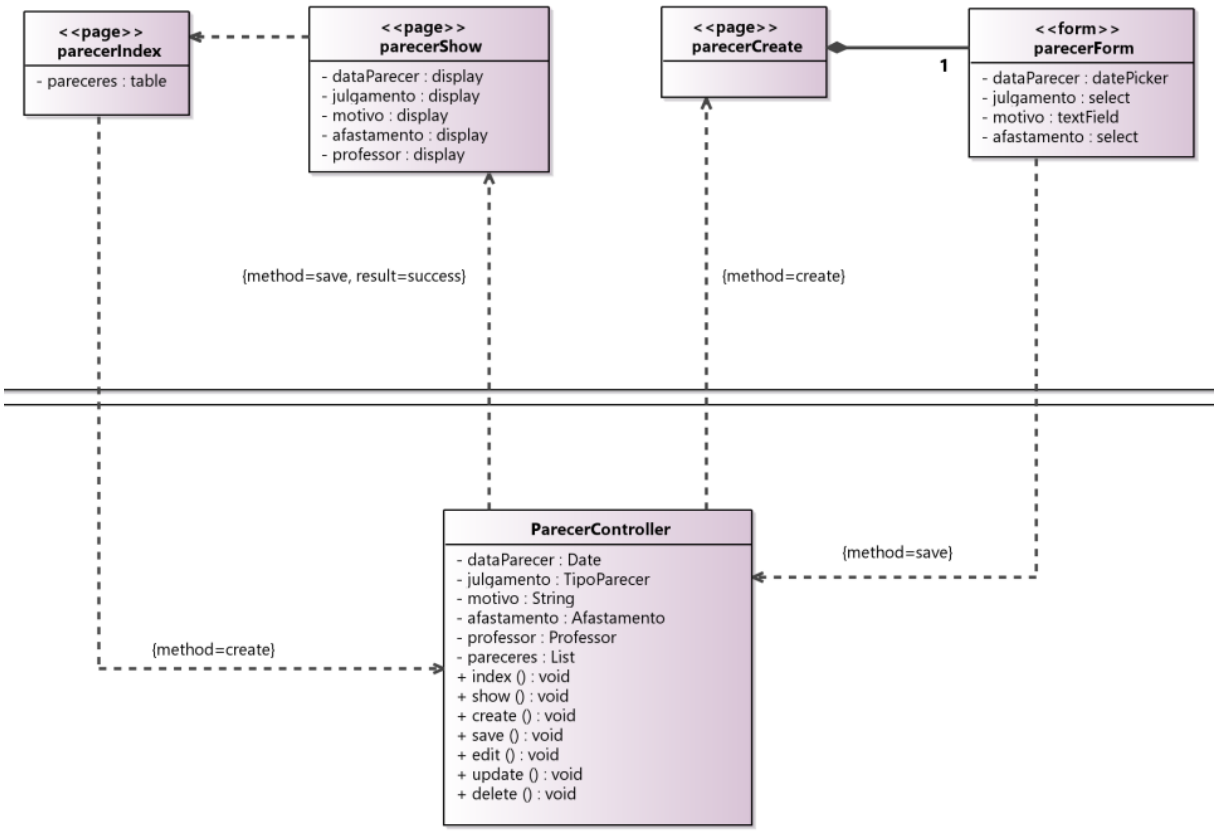

Figura 7 – Modelo de Navegação dos Casos de Uso: Deferir Parecer e Manifestar-se Contra Afastamento.

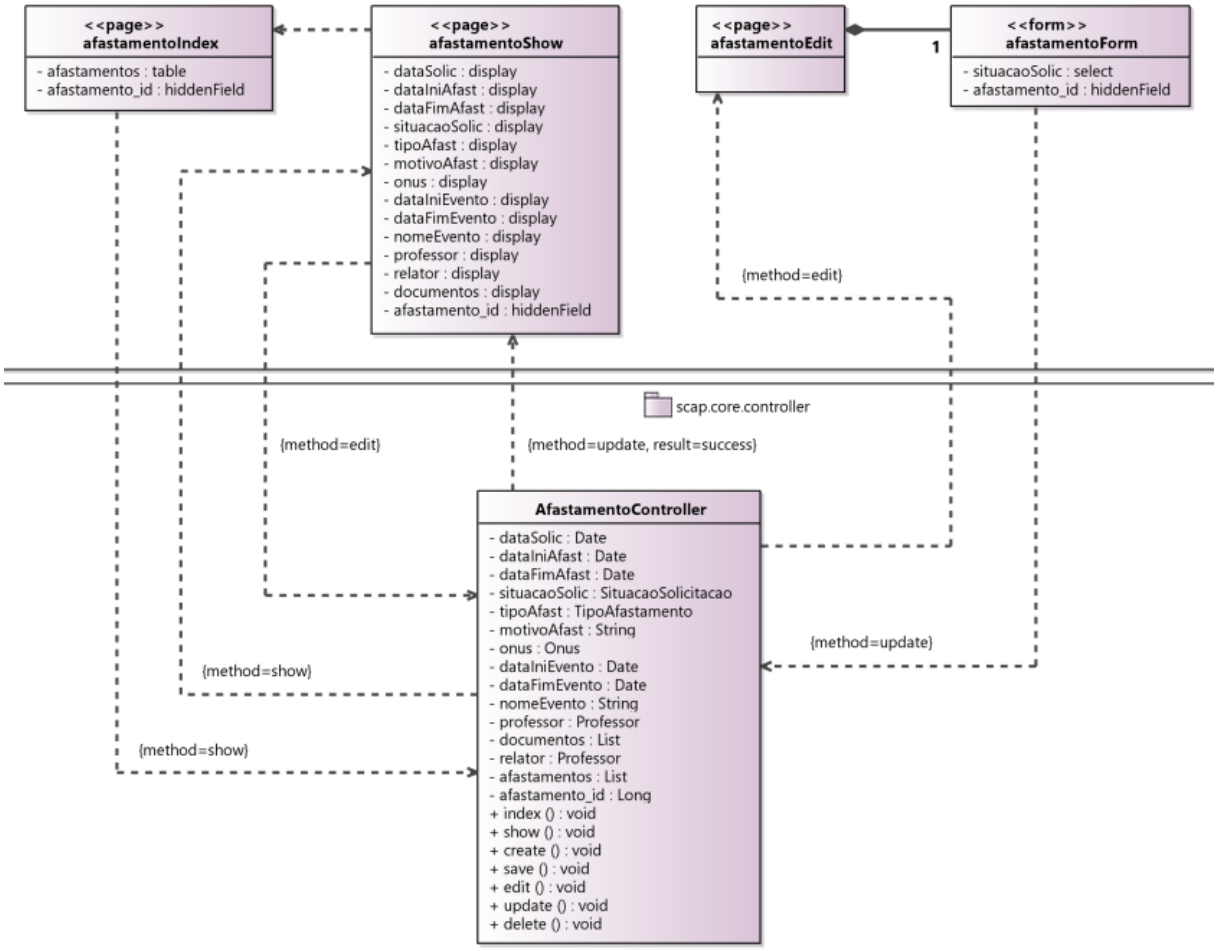

Figura 8 – Modelo de Navegação dos Casos de Uso: Cancelar Afastamento, Arquivar Afastamento, Registrar Parecer CT e Registrar Parecer PRPPG.

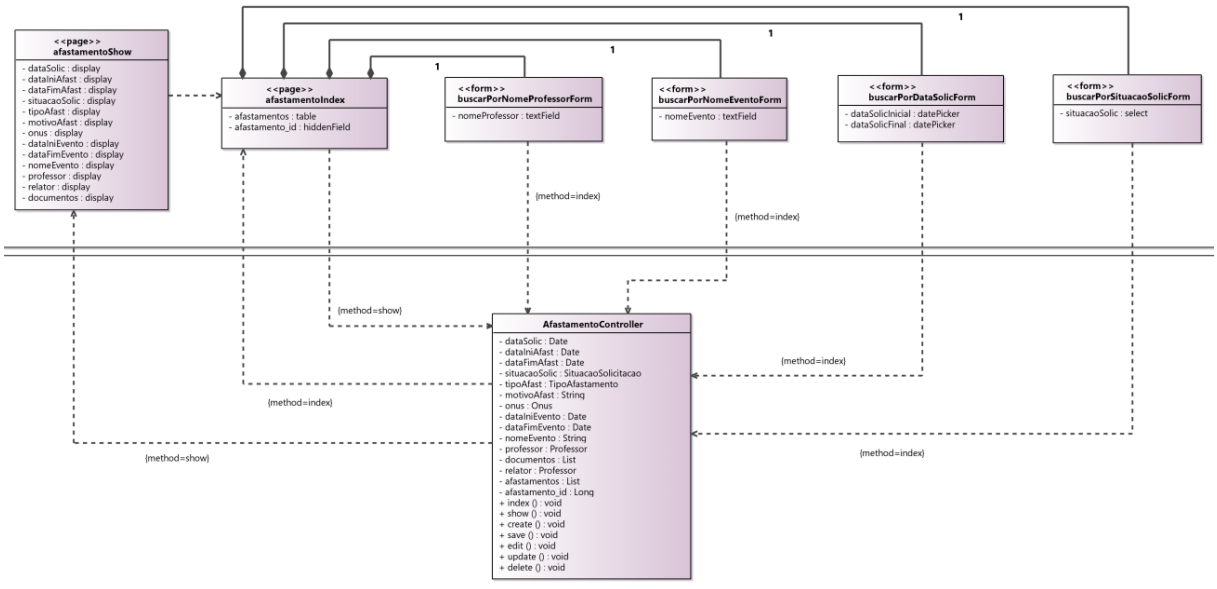

Figura 9 – Modelo de Navegação do Caso de Uso: Consultar Afastamento.

4.2 Camada de Negócio

Engloba as funcionalidades que dão suporte aos processos de negócio, concentrando as regras de negócio, conceitos do domínio, cálculos e processamentos. Representado por um diagrama de classes da UML, o Modelo de Entidades apresenta o mapeamento dos objetos de domínio do problema para a persistência em banco de dados relacional. Esse modelo pode ser visualizado na Figura 10 e na Figura 11.

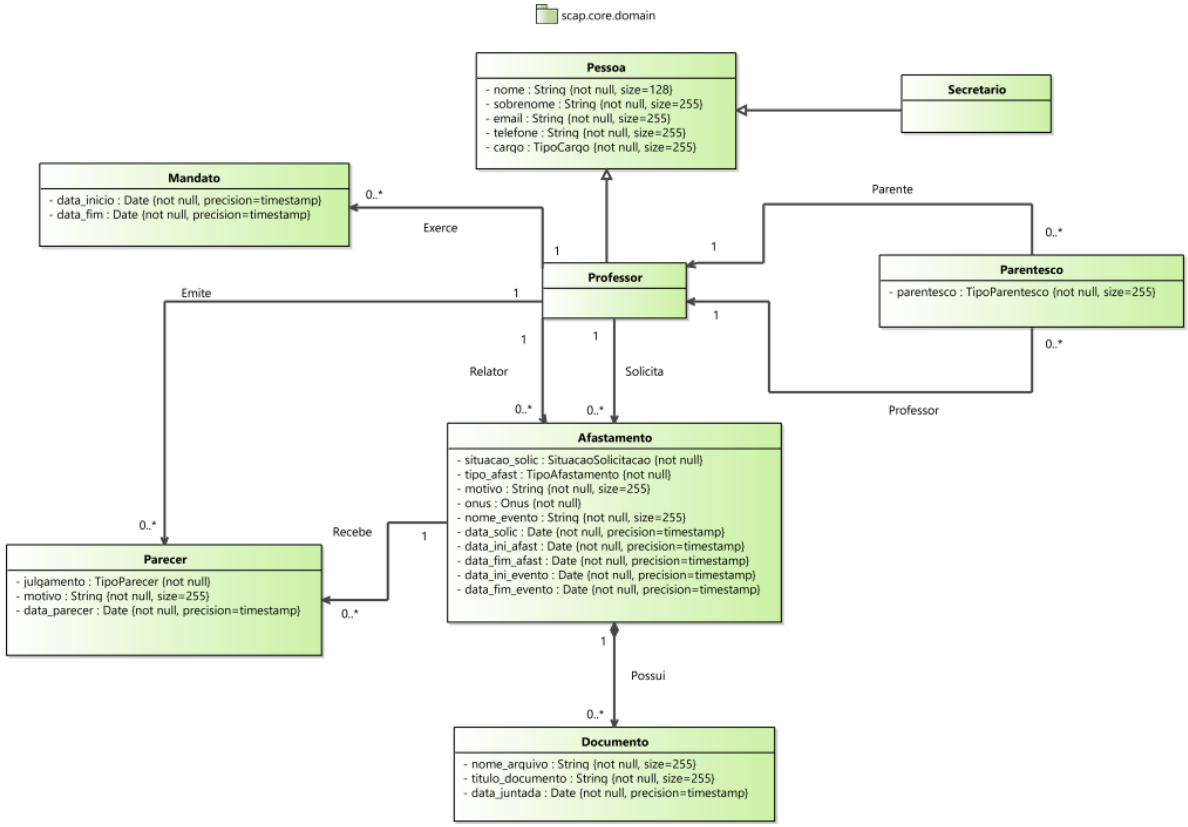

Figura 10 – Modelo de Entidades.

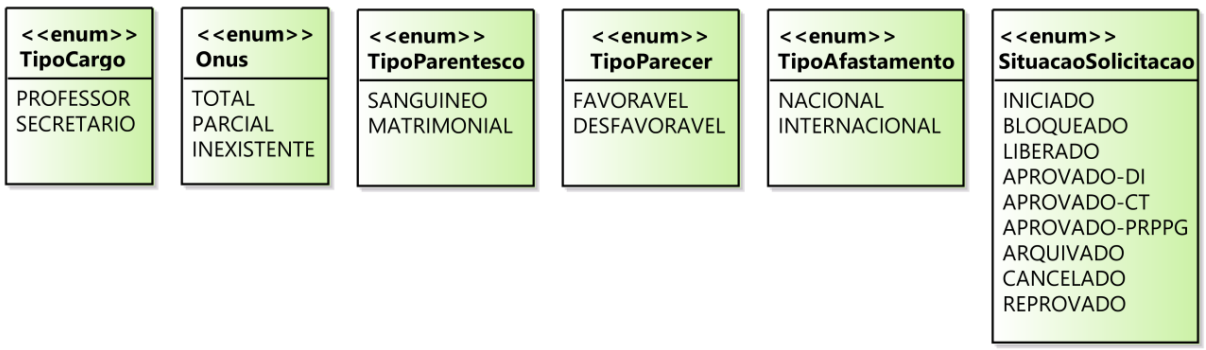

Figura 11 – Tipos Enumerados do SCAP.

O diagrama do Modelo de Aplicação é utilizado para auxiliar os desenvolvedores na implementação das classes do pacote Aplicação e na configuração das dependências entre os pacotes Controle, Aplicação e Persistência. Neste projeto foi gerado um modelo

de aplicação que representa quais classes de ação que dependem de quais classes de serviço e quais *Data Access Object* (DAO) foram utilizados. Portanto, este modelo pode ser visualizado através da Figura 12 e da Figura 13.

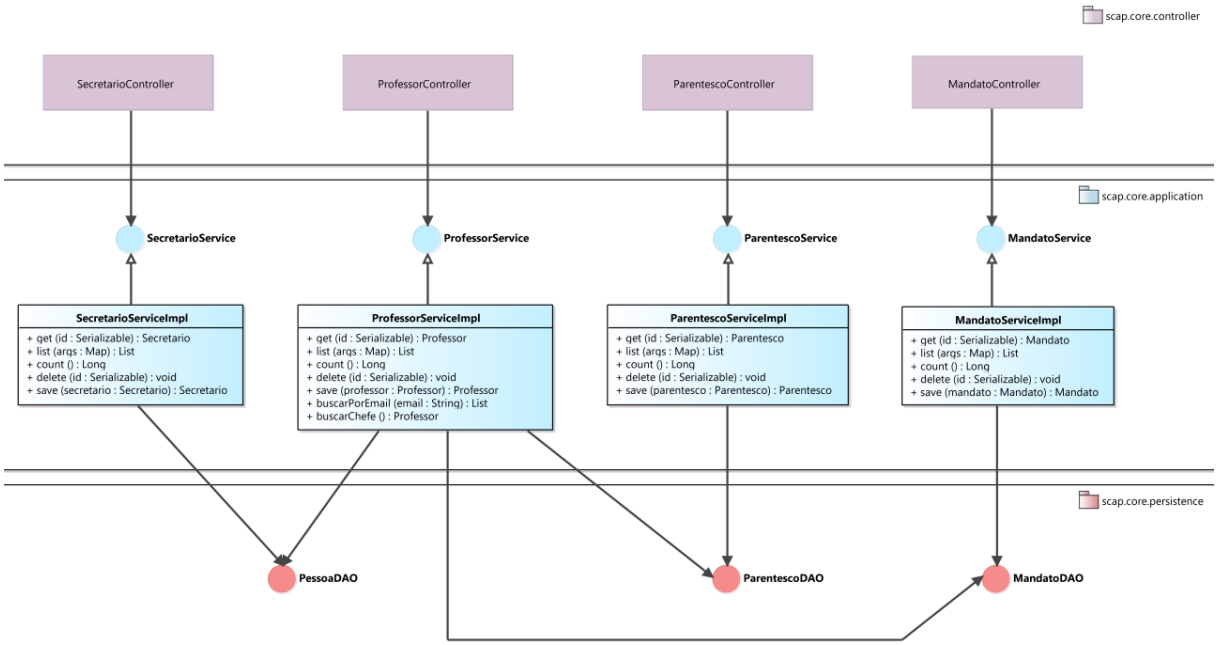

Figura 12 – Modelo de Aplicação - Parte 1.

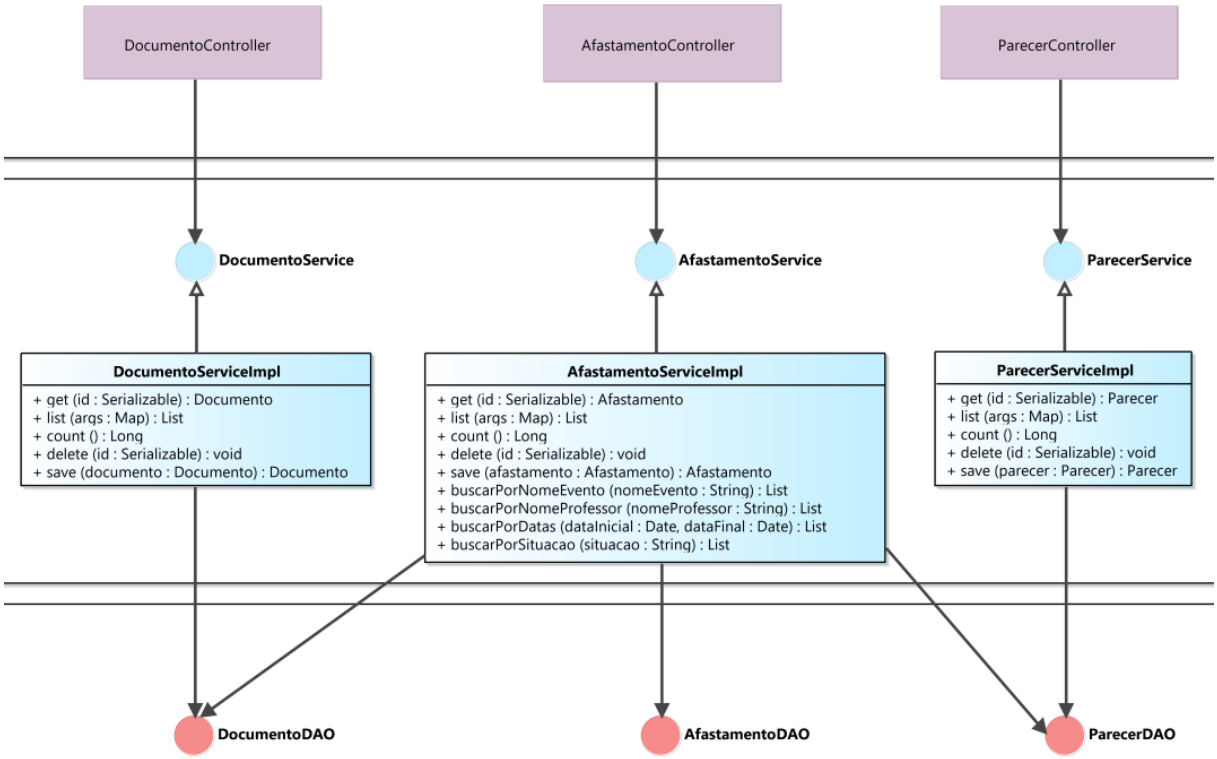

Figura 13 – Modelo de Aplicação - Parte 2.

4.3 Camada de Acesso a Dados

Estabelece o acesso a dados, gerenciando requisições e cuidando da sincronização de elementos de dados. Um diagrama de classes da UML representa as classes DAO existentes, que são responsáveis pela persistência das instância das classes de domínio. Esse diagrama pode ser visualizado na Figura 14.

Com o intuito de não causar poluição visual, é possível fazer com que os métodos que são comuns a todas a interfaces DAOs não sejam repetidos sem necessidade. Para isso, basta apresentar DAOs base que declaram esses métodos. De forma automática, todas as interfaces DAO de todos os diagramas herdam as definições da interface base, ocorrendo o mesmo com as implementações concretas de cada tecnologia de persistência, sem que isso precise estar explícito no diagrama. Ainda é possível declarar tanto a interface quanto a classe DAOBase usando tipos genéricos, deixando a cargo de suas sub-interfaces e sub-classes a especificação da classe gerenciada por cada DAO (SOUZA, 2007).

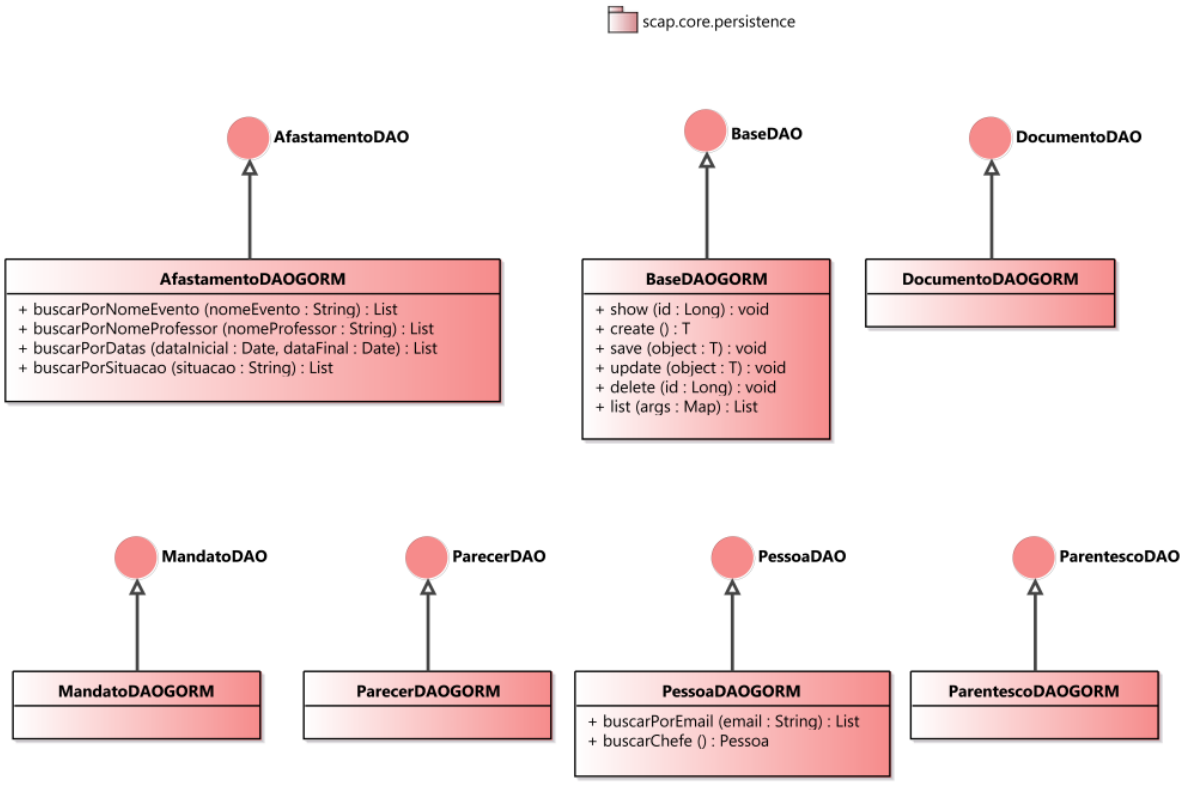

Figura 14 – Modelo de Persistência.

Referências

FOWLER, M. *Patterns of Enterprise Application Architecture*. 1. ed. [S.l.]: Addison-Wesley, 2002. ISBN 9780321127426. Citado na página 5.

SOUZA, V. E. S. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2007. Citado 2 vezes nas páginas 5 e 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. IGI Global, 2009. cap. 11, p. 203–237. ISBN 9781605662787. Disponível em: <http://www.igi-global.com/reference/details.asp?id=33232>. Citado na página 5.