

Vinícius Berger

**Aplicação do método FrameWeb no
desenvolvimento de um sistema de informação
utilizando os frameworks Yii e Symfony**

Vitória, ES

2021

Vinícius Berger

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Yii e Symfony

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

Vinícius Berger

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Yii e Symfony/ Vinícius Berger. – Vitória, ES, 2021-

103 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. Yii. 2. Symfony. 3. FrameWeb. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. III. Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Yii e Symfony

CDU 02:141:005.7

Vinícius Berger

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Yii e Symfony

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^a. Dr^a. Monalessa Perini Barcellos
Universidade Federal do Espírito Santo

César Henrique Bernabé
Leiden University Medical Center

Vitória, ES
2021

Agradecimentos

Agradeço primeiramente a Deus por ter ouvido e respondido as minhas orações. Por ter me concedido saúde e determinação para não desanimar durante a realização deste trabalho e por me ajudar a ultrapassar todos os obstáculos encontrados ao longo dos anos de estudo nesta Universidade. Principalmente por ter colocado pessoas extraordinárias em meu caminho.

Gratidão aos meus pais, Valdecir e Nicélia, por sua presença e amor incondicional. Por terem sempre sido uma base firme na qual eu pude me apoiar para realizar os meus sonhos. Esta monografia é a prova de que os esforços deles não foram em vão. Ao meu irmão, Nicolás, pela amizade e por toda ajuda e força que me deu, principalmente no início dessa jornada. Sou grato à minha família — avôs e avós, tios e tias, primos e primas — pelo apoio que sempre me deram.

Agradeço à minha namorada Emily, por ter estado ao meu lado praticamente durante todo o percurso acadêmico e por ter compreendido a minha ausência em muitos momentos enquanto eu me dedicava à realização desse sonho. Agradeço também a toda sua família, que sempre me acolheu de braços abertos.

Um agradecimento muito especial aos meus amigos Gustavo e Léo, com os quais eu passei a maior parte do tempo nesta Universidade. Nunca vou esquecer o que vivemos juntos: as conversas, os cafés, os grupos de estudo, os passeios; as competições que participamos e os prêmios que ganhamos; as reprovações e suadas aprovações. Dizem que somos a média das cinco pessoas com as quais mais convivemos e eu sou grato por ter esses dois influenciando positivamente a minha média. Agradeço também a todos os meus amigos e colegas do curso de graduação que compartilharam dos inúmeros desafios, sempre com o espírito colaborativo. Juntos conseguimos avançar e ultrapassar todos os obstáculos.

A todos os professores que contribuíram com a minha formação acadêmica e profissional, especialmente ao meu orientador, Vítor, que apesar da intensa rotina de sua vida acadêmica, aceitou me orientar nesta monografia. As suas correções e indicações foram muito valiosas e fizeram toda a diferença.

Também agradeço ao meu chefe, mentor e amigo, Alexandre, por todo conhecimento que me passou nesses últimos anos e por me incentivar diariamente a ir além, sempre buscando a excelência.

Agradeço a todos que direta ou indiretamente fizeram parte da minha formação.

*Princípio de modelagem:
os modelos não estão certos ou errados;
eles são mais ou menos úteis.
(Martin Fowler)*

Resumo

A ascensão da Internet como um dos principais meios de comunicação, em meados dos anos 90, fez surgir uma crescente demanda por sistemas comerciais implantados nesse ambiente. A complexidade desses sistemas foi crescendo com o passar do tempo e as formas tradicionais de desenvolvimento passaram a não atender as necessidades do mercado, principalmente as relacionadas a prazo e qualidade. Começou-se então a utilizar conceitos da Engenharia de Software no desenvolvimento de aplicações para a Web, dando origem à Engenharia Web.

Nesse contexto surgiu o FrameWeb, um método de projeto para sistemas Web que permite realizar a modelagem do sistema considerando o uso de *frameworks*. Dessa forma, ele permite que o projetista especifique melhor o que deve ser implementado pelo programador.

Desde 2014 o FrameWeb tem sido utilizado por estudantes finalistas em seus projetos de conclusão de curso, a fim de testar a eficácia do método com diferentes *frameworks*, linguagens de programação e padrões de projeto.

De forma semelhante, neste trabalho foram desenvolvidas duas versões de um sistema Web (já especificado e analisado) chamado SCAP — Sistema de Controle de Afastamentos de Professores — utilizando o FrameWeb na etapa de projeto e os *frameworks* Yii e Symfony na etapa de implementação.

Palavras-chaves: Engenharia Web. FrameWeb. Yii. Symfony. SCAP.

Lista de ilustrações

Figura 1 – Arquitetura padrão proposta pelo FrameWeb. Adaptado de (SOUZA, 2007).	21
Figura 2 – Funcionamento do padrão MVC. Adaptado de (SOUZA, 2007).	23
Figura 3 – Funcionamento do Controlador Frontal. Adaptado de (FOWLER, 2002).	23
Figura 4 – Padrão Active Record. Adaptado de (FOWLER, 2002).	25
Figura 5 – Padrão Data Mapper. Adaptado de (FOWLER, 2002).	27
Figura 6 – Diagrama de Casos de Uso do SCAP.	30
Figura 7 – Diagrama de Classes do SCAP.	34
Figura 8 – Tipos enumerados do SCAP.	34
Figura 9 – Modelo de Entidades.	41
Figura 10 – Diagrama de Estados para instâncias da classe Afastamento.	42
Figura 11 – Modelo de Navegação: Caso de uso “Solicitar Afastamento”.	44
Figura 12 – Modelo de Navegação: Caso de uso “Cadastrar usuário (Secretário)”.	44
Figura 13 – Modelo de Navegação: Caso de uso “Cadastrar usuário (Professor)”.	45
Figura 14 – Modelo de Navegação: Caso de uso “Cadastrar chefe do departamento”.	45
Figura 15 – Modelo de Navegação: Caso de uso “Encaminhar Afastamento”.	46
Figura 16 – Modelo de Navegação: Caso de uso “Manifestar-se Contra Afastamento”.	46
Figura 17 – Modelo de Navegação: Caso de uso “Registrar decisão CT”.	47
Figura 18 – Modelo de Navegação: Caso de uso “Cancelar Afastamento”.	47
Figura 19 – Modelo de Navegação: Caso de uso “Registrar parecer relator”.	48
Figura 20 – Modelo de Navegação: Caso de uso “Consultar Afastamento”.	48
Figura 21 – Modelo de Aplicação (Symfony).	49
Figura 22 – Modelo de Aplicação (Yii).	50
Figura 23 – Modelo de Persistência (Symfony).	52
Figura 24 – Login.	53
Figura 25 – Listagem de Afastamentos.	56
Figura 26 – Consultar Afastamento.	56
Figura 27 – Solicitar Afastamento.	57
Figura 28 – Visualizar Afastamento (1).	58
Figura 29 – Visualizar Afastamento (2).	58
Figura 30 – Encaminhar Afastamento (1).	59
Figura 31 – Encaminhar Afastamento (2).	59
Figura 32 – Email enviado ao relator.	60
Figura 33 – Manifestar-se contra Afastamento.	60
Figura 34 – Registrar decisão do DI.	61
Figura 35 – Mensagem de sucesso.	61

Figura 36 – Mensagem de erro.	62
Figura 37 – Listagem de Professores.	62
Figura 38 – Cadastrar Professor (1).	63
Figura 39 – Cadastrar Professor (2).	63

Lista de tabelas

Tabela 1 – Atores do SCAP.	29
Tabela 2 – Caso de uso “Registrar decisão DI”.	32
Tabela 3 – Implementações anteriores do SCAP.	36
Tabela 4 – Tecnologias utilizadas no desenvolvimento do projeto	37
Tabela 5 – Softwares de apoio ao desenvolvimento do projeto	39
Tabela 6 – Mapeamentos objeto/relacionais utilizados explicitamente no Modelo de Entidades do SCAP (fragmento da tabela disponível em (SOUZA, 2007)).	40

Lista de abreviaturas e siglas

CEPE	Conselho de Ensino, Pesquisa e Extensão da Universidade Federal do Espírito Santo
CRUD	Criação, Leitura, Atualização e Exclusão, do inglês <i>Create, Read, Update and Delete</i>
CT	Centro Tecnológico
DI	Departamento de Informática da Universidade Federal do Espírito Santo
DI	Injeção de Dependência, do inglês <i>Dependency Injection</i>
EUA	Estados Unidos da América
FrameWeb	<i>Framework-based Design Method for Web Engineering</i>
MEC	Ministério da Educação
NoSQL	Não Somente SQL, do inglês <i>Not Only SQL</i>
OO	Paradigma de programação Orientado a Objetos
ORM	Mapeamento Objeto/Relacional, do inglês <i>Object/Relational Mapping</i>
PRPPG	Pró-Reitoria de Pesquisa e Pós-Graduação
SCAP	Sistema de Controle de Afastamentos de Professores
SGBDR	Sistema de Gerenciamento de Bancos de Dados Relacionais
SGDP/ME	Secretaria de Gestão e Desempenho de Pessoal / Ministério da Economia
SQL	Linguagem de Consulta Estruturada, do inglês <i>Structured Query Language</i>
UFES	Universidade Federal do Espírito Santo
UML	Linguagem de Modelagem Unificada, do inglês <i>Unified Modeling Language</i>

Sumário

1	INTRODUÇÃO	13
1.1	Objetivos	14
1.2	Método	15
1.3	Organização do trabalho	15
2	REFERENCIAL TEÓRICO	17
2.1	Engenharia Web	17
2.2	Processo de Software	18
2.3	O Método FrameWeb	19
2.4	<i>Frameworks</i>	21
2.4.1	<i>Frameworks MVC</i>	22
2.4.2	<i>Frameworks ORM</i>	24
2.4.2.1	Padrão <i>Active Record</i>	25
2.4.2.2	Padrão <i>Data Mapper</i>	26
3	ESPECIFICAÇÃO DE REQUISITOS E ANÁLISE	28
3.1	Descrição do Escopo	28
3.2	Modelo de Casos de Uso	29
3.3	Modelo de Classes	32
3.4	Restrições de Integridade	34
4	PROJETO E IMPLEMENTAÇÃO	36
4.1	Plataforma de desenvolvimento	36
4.1.1	Yii	38
4.1.2	Symfony	38
4.2	Arquitetura	39
4.3	Modelo de Entidades	40
4.4	Modelo de Navegação	43
4.5	Modelo de Aplicação	49
4.6	Modelo de Persistência	50
4.7	Implementação	53
5	CONSIDERAÇÕES FINAIS	64
5.1	Discussão	65
5.1.1	Limitações	65
5.1.2	Contribuições	66

5.2	Trabalhos Futuros	66
	REFERÊNCIAS	68
	APÊNDICES	71

1 Introdução

Após alguns anos desde a liberação da Internet para uso comercial, que ocorreu no final dos anos 80, surgiram as primeiras empresas provedoras de acesso à Internet nos EUA (SILVA, 2001). Ao mesmo tempo, o Laboratório Europeu de Física de Partículas (CERN) inventou a *World Wide Web*, que começou a ser utilizada para colocar informações ao alcance de qualquer usuário da Internet (SILVA, 2001).

“Nos primórdios da *World Wide Web* (por volta de 1990 a 1995), os sites eram formados por nada mais que um conjunto de arquivos de hipertexto linkados que apresentavam informações usando texto e gráficos limitados” (PRESSMAN, 2011, p. 37). Em pouco tempo, a Web tomou grandes proporções. Devido ao grande número de usuários envolvidos, investimento em pesquisas e o surgimento de tecnologias auxiliares, grandes sistemas comerciais começaram a ser desenvolvidos e implantados nesse ambiente. A evolução e manutenção desses sistemas, com o tempo, começou a exigir esforços demasiados dos programadores e as formas tradicionais de desenvolvimento passaram a não suprir as necessidades do mercado (PRADO, 2015). Surgiram então os primeiros *frameworks*, que, segundo Gabrieli et al. (2007), são infra-estruturas de aplicações projetadas para serem reutilizadas. Portanto, um *framework* é uma arquitetura semi-pronta de aplicação e representa o nível mais alto de reutilização (CUNHA; HERBERT, 2003). A redução da manutenção e a maximização do reuso são dois dos ganhos conseguidos com a utilização dos *frameworks* (GAMMA et al., 2000), o que resulta em redução do tempo de desenvolvimento.

De acordo com Souza (2007), este contexto foi o que motivou o início das pesquisas que deram origem ao FrameWeb, um método para Engenharia Web que promove a modelagem dos componentes dos *frameworks* nos diagramas de projeto com vistas a fazer com que estes modelos estejam mais próximos de implementação em código. Dessa forma, a definição da arquitetura fica a cargo do projetista do sistema e não mais dos programadores, o que incentiva a utilização de uma arquitetura robusta e o aumento da produtividade no desenvolvimento de sistemas de informação para a Web (SOUZA, 2007).

Duarte (2014) aplicou o método FrameWeb no processo de desenvolvimento de um sistema nomeado SCAP — Sistema de Controle de Afastamentos de Professores — para avaliar sua aplicabilidade e propor melhorias para os modelos que são definidos pelo método. O SCAP é um sistema que foi especificado para gerenciar as solicitações de afastamento de professores que tramitam dentro do Departamento de Informática da UFES (DUARTE, 2014). Essas solicitações podem ser fundamentadas em diversos instrumentos jurídicos. Por exemplo: os afastamentos para participação em eventos e

outras atividades que ocorrem em solo brasileiro são disciplinados pelas Portarias MEC nº 204/2020 e UFES nº 90 de 10/02/2020 (PRPPG, c2013c). Já os afastamentos para participação em eventos e outras atividades que ocorrem fora do Brasil são disciplinados pelos Decretos Presidenciais nº 91.800 de 18/10/1985, nº 1.387 de 07/02/1995, bem como pelas Portarias MEC nº 204/2020 e UFES nº 90 de 10/02/2020 (PRPPG, c2013a). Há ainda os afastamentos para capacitação regidos pela Lei 8.112/1990, Decreto nº 9.991/2019, Instrução Normativa nº 201/19-SGDP/ME e Resolução CEPE 31/2012 (PRPPG, c2013b).

O SCAP já foi projetado e implementado por vários estudantes de graduação da UFES em seus trabalhos de conclusão de curso, cada um utilizando *frameworks* diferentes, a saber: Java EE 7 por Duarte (2014), VRaptor 4 por Prado (2015), Laravel e Vue.js por Pinheiro (2017), Spring MVC e Vaadin por Matos (2017), Ninja por Avelar (2018), Wicket e Tapestry por Ferreira (2018), Play por Guterres (2019), Code Igniter e Node.js por Meirelles (2019). Vale ressaltar que os requisitos do SCAP, levantados e analisados por Duarte (2014), foram complementados e materializados oficialmente em “Documento de Requisitos” e “Documento de Especificação de Requisitos” por Prado (2015). Os demais trabalhos utilizaram estes documentos como base para gerar, cada um, seu próprio “Documento de Projeto” e sua própria implementação do SCAP. Não houve nenhuma atualização no “Documento de Requisitos” nem no “Documento de Especificação de Requisitos” desde Prado (2015), apesar de Guterres (2019) ter feito alterações muito interessantes, no âmbito de seu trabalho, que poderiam ser incorporadas oficialmente em um trabalho futuro.

De forma semelhante, duas novas implementações desse sistema são realizadas neste trabalho, uma utilizando o *framework* Yii e outra utilizando o *framework* Symfony, seguindo o método FrameWeb proposto por Souza (2007). Espera-se, dessa forma, contribuir com a validação do método FrameWeb em um cenário novo e eventualmente sugerir adaptações.

1.1 Objetivos

O presente trabalho tem por objetivo geral realizar duas novas implementações do SCAP, uma utilizando o *framework* Yii e outra utilizando o *framework* Symfony, ambas utilizando o método FrameWeb proposto por Souza (2007) na etapa de Projeto. Em seguida, realizar uma análise da aplicação do método FrameWeb em ambos os projetos.

Os objetivos específicos a seguir serão a base para o alcance do objetivo geral:

- Compreensão dos requisitos do SCAP;
- Uso do método FrameWeb proposto por Souza (2007) para os projetos arquiteturais das duas novas implementações do sistema;

- Capacitação nos *frameworks* Yii e Symfony, selecionados para o desenvolvimento;
- Desenvolvimento das duas novas implementações do sistema, incluindo documentação das arquiteturas e projetos.

1.2 Método

Para atingir o objetivo geral apresentado na seção anterior, os seguintes passos serão realizados:

1. Revisão bibliográfica: leitura de materiais que forneçam uma revisão dos conceitos já estudados e materiais que apresentem conteúdos novos e necessários para o desenvolvimento do sistema, sendo o método FrameWeb proposto por [Souza \(2007\)](#) o principal deles;
2. Análise do escopo do sistema e dos requisitos levantados: leitura dos trabalhos anteriores, com ênfase nos trabalhos realizados por [Duarte \(2014\)](#) e [Prado \(2015\)](#), onde consta todo o levantamento e análise de requisitos do sistema a ser implementado;
3. Estudo das tecnologias: leitura da documentação oficial dos dois *frameworks* MVC utilizados para desenvolver o sistema, a saber: Yii e Symfony. Além disso, uma pequena revisão sobre as tecnologias relacionadas ao desenvolvimento Web (HTML, CSS, JavaScript e PHP) e banco de dados relacional;
4. Elaboração das arquiteturas das duas novas implementações do sistema: com base no método FrameWeb e considerando as tecnologias estudadas, elaborar as duas arquiteturas do sistema gerando os Documentos de Projeto (um para a implementação feita com o Yii e outra para a implementação feita com o Symfony);
5. Implementação do sistema: codificação das duas novas implementações do sistema utilizando as tecnologias definidas;

1.3 Organização do trabalho

Este trabalho está dividido em cinco capítulos, incluindo a Introdução. O conteúdo dos próximos capítulos é descrito a seguir.

No Capítulo 2 — Referencial Teórico — são reunidos conceitos e explicações de outros autores a respeito dos assuntos relacionados a este trabalho, formando assim uma base de conhecimento que servirá de alicerce para a argumentação ao longo dos demais capítulos.

No Capítulo 3 — Especificação de Requisitos e Análise — são apresentados o levantamento e análise dos requisitos do SCAP (Sistema de Controle de Afastamentos de Professores) realizados em Duarte (2014) e Prado (2015).

No Capítulo 4 — Projeto e Implementação — é apresentada a aplicação do método FrameWeb na construção das arquiteturas do SCAP. Para as implementações do sistema foram escolhidos dois *frameworks* MVC (Yii e Symfony) ambos desenvolvidos em PHP. Ao longo do capítulo serão apresentados trechos dos códigos referente às duas implementações do SCAP.

No Capítulo 5 — Considerações Finais — são apresentados os resultados obtidos da aplicação do método FrameWeb em termos de compatibilidade com os frameworks Yii e Symfony.

2 Referencial Teórico

Devido ao grande número de usuários, investimento em pesquisas e o conseqüente avanço da tecnologia computacional no início dos anos 90, grandes sistemas comerciais começaram a ser desenvolvidos para o ambiente Web. [Pressman \(2011\)](#) afirma que, com o passar do tempo, tais sistemas começaram não apenas a oferecer funções especializadas aos usuários, como também foram integrados aos bancos de dados corporativos e às aplicações de negócio, tornando-os ferramentas computacionais sofisticadas.

Porém, na maioria dos casos, o desenvolvimento desses sistemas ocorreu sem uma abordagem sistemática e sem procedimentos de controle e garantia de qualidade, o que gerou uma crescente e legítima preocupação sobre a maneira como os sistemas baseados na Web eram desenvolvidos e sua qualidade e integridade ([MURUGESAN et al., 2001](#)).

Considerando que pessoas, negócios e governos dependem cada vez mais de software para tomada de decisões estratégicas e táticas e que a falha de um software pode gerar falhas catastróficas para esses indivíduos ([PRESSMAN, 2011](#)), é imprescindível que tais aplicações sejam construídas de forma a garantir o máximo de qualidade do produto final.

[Murugesan et al. \(2001\)](#) afirma que para obter maior sucesso no desenvolvimento de aplicações e sistemas baseados na Web, há a necessidade de abordagens disciplinadas e uso de novos métodos e ferramentas (para desenvolvimento, implantação e avaliação de sistemas), que sejam específicos para essa plataforma.

Algumas características do ambiente Web citadas por [Pressman \(2011, p. 37\)](#), como concorrência, carga imprevisível, disponibilidade, sensibilidade ao conteúdo, evolução dinâmica, imediatismo, segurança e estética, segundo [Souza \(2007, p. 19\)](#) “imprimiram uma nova dinâmica aos processos já existentes, dando origem a uma nova sub-área da Engenharia de Software, a Engenharia Web”.

2.1 Engenharia Web

A [IEEE \(1990, p. 67\)](#) define Engenharia de Software como sendo “(1) A aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, na operação e na manutenção de software; isto é, a aplicação de engenharia ao software. (2) O estudo de abordagens como definido em (1)”.

A Engenharia Web, para [Pressman \(2011\)](#), é uma versão adaptada da Engenharia de Software. [Murugesan et al. \(2001\)](#) afirma que os princípios da Engenharia de Software são incorporados pela Engenharia Web, a qual aplica-os para a natureza mais aberta e flexível da Web, considerando também outros elementos específicos desse ambiente.

A Engenharia Web se preocupa com o estabelecimento e uso de princípios científicos, de engenharia [...] e abordagens disciplinadas e sistemáticas para o desenvolvimento, implantação e manutenção bem-sucedidos de sistemas e aplicativos de alta qualidade baseados na Web (MURUGESAN et al., 2001, p. 4, tradução nossa).

É importante observar que as definições convergem para o fato de que a Engenharia Web possui uma abordagem disciplinada para o desenvolvimento de um software profissional. “Disciplina” para Fernandes, Luft e Guimarães (1996, p. 225) significa, entre outras coisas, a “observância de normas ou preceitos”.

Essa abordagem disciplinada, para Pressman (2011), ocorre quando há a adoção de um **processo**, um conjunto de **métodos** (práticas) e um leque de **ferramentas** que possibilitem o desenvolvimento de software com alto padrão de qualidade.

Nas próximas seções serão apresentadas as etapas de um processo de software genérico descrito por Pressman (2011), um método específico para etapa de Modelagem chamado FrameWeb, proposto por Souza (2007) e um tipo de ferramenta utilizada para auxiliar a etapa de Contrução: os *frameworks*.

2.2 Processo de Software

“Processo é um conjunto de atividades, ações e tarefas realizadas na criação de algum produto de trabalho” (PRESSMAN, 2011).

Quando se fala de Processo, no contexto da Engenharia de Software, Pressman (2011) afirma que o mesmo não deve ser considerado uma prescrição rígida de como desenvolver um software, mas sim uma abordagem adaptativa que possibilita à equipe envolvida a escolha de um conjunto de tarefas que sejam mais apropriadas à sua realidade, sempre lembrando que o objetivo é entregar software de qualidade e dentro do prazo.

Uma metodologia de processo estabelece o alicerce para um processo de engenharia de software completo, por meio da identificação de um pequeno número de atividades estruturais aplicáveis a todos os projetos de software, independentemente de tamanho ou complexidade (PRESSMAN, 2011, p. 40).

Segundo Pressman (2011), uma metodologia de processo genérica para engenharia de software compreende cinco atividades:

1. **Comunicação:** também chamada de “Especificação de Requisitos”, é a atividade onde se procura compreender os objetivos das partes interessadas para com o projeto e fazer o levantamento das necessidades que elucidarão as funções e características do software.

2. **Planejamento:** também chamada de “Análise”, nesta etapa, com base no que foi levantado na etapa anterior, cria-se um “mapa” que norteará as ações da equipe. Esse mapa — denominado plano de projeto de software — descreve as tarefas técnicas a serem conduzidas, os riscos prováveis, os recursos que serão necessários, os produtos resultantes a serem produzidos e um cronograma de trabalho.
3. **Modelagem:** também chamada de “Projeto”, nesta etapa cria-se um “esboço” do sistema, de modo que se possa ter uma ideia do todo. São produzidos modelos para melhor entender as necessidades do software e o projeto que irá atender a essas necessidades.
4. **Construção:** também chamada de “Implementação”, essa atividade combina geração de código (manual ou automatizada) e testes necessários para revelar erros na codificação.
5. **Emprego:** também chamada de “Implantação”, é onde o software (como uma entidade completa ou como um incremento parcialmente efetivado) é entregue ao cliente, que avaliará o produto e fornecerá *feedback*.

Segundo [Pressman \(2011\)](#), as atividades descritas acima podem ser aplicadas iterativamente, conforme o projeto se desenvolve, produzindo, a cada iteração, um incremento no software. Assim, cada incremento disponibilizará uma parte dos recursos e funcionalidades, deixando o software cada vez mais completo.

2.3 O Método FrameWeb

“FrameWeb é um método de projeto para construção de sistemas de informação Web (*Web Information Systems – WISs*) baseado em *frameworks*” ([SOUZA, 2007](#), p. 71).

De acordo com [Souza \(2007\)](#), as principais propostas do método são a definição de uma arquitetura lógica baseada no padrão arquitetônico *Service Layer* (Camada de Serviço) (Figura 1) e a utilização de uma linguagem de modelagem específica para a produção de quatro tipos de modelos de projeto:

- **Modelo de Entidades:** inicialmente chamado de “Modelo de Domínio”, o Modelo de Entidades “é um diagrama de classes da UML que representa os objetos de domínio do problema e seu mapeamento para a persistência em banco de dados relacional” ([SOUZA, 2007](#), p. 78). Ele ajusta o modelo de classes (construído na etapa anterior do processo) para a plataforma de implementação escolhida, indicando os tipos de dados de cada atributo e a navegabilidade das associações. Além disso, utiliza mecanismos leves de extensão da UML (como estereótipos e restrições) para

adicionar meta-dados às classes de domínio (mapeamentos de persistência) que permitam aos *frameworks* ORM (Seção 2.4.2) transformar objetos que estão na memória em linhas de tabelas no banco de dados relacional e vice-versa. “Apesar de tais configurações serem relacionadas mais à persistência do que ao domínio, elas são representadas no Modelo de Domínio [Entidades] porque as classes que são mapeadas e seus atributos são exibidas neste diagrama” (SOUZA, 2007). O Modelo de Entidades serve de base para a implementação das classes residentes no pacote “Domínio” do padrão arquitetônico sugerido pelo FrameWeb (Figura 1).

- **Modelo de Persistência:** “o Modelo de Persistência é um diagrama de classes da UML que representa as classes DAO existentes, responsáveis pela persistência das instâncias das classes de domínio” (SOUZA, 2007, p. 81). Ele apresenta uma interface e uma classe concreta (que implementa a interface) para cada classe de domínio que precisa de lógica de acesso a dados. A interface define os métodos de persistência básicos e as operações de consultas específicas necessárias para atender aos casos de uso do sistema. Pode-se criar uma interface e uma classe concreta base (com métodos comuns à todas as outras) para que o modelo não fique “poluído” com métodos repetidos. “Automaticamente, 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” (SOUZA, 2007). Como pôde ser observado, o método FrameWeb indica a utilização do padrão de projeto DAO para a construção da camada de acesso a dados.
- **Modelo de Navegação:** “o Modelo de Navegação é um diagrama de classes da UML que representa os diferentes componentes que formam a camada de Lógica de Apresentação, como páginas Web, formulários HTML e classes de ação do *framework Front Controller*” (SOUZA, 2007, p. 84). A classe de ação é o principal componente do modelo e suas associações de dependência regem o fluxo da aplicação, permitindo identificar como os dados são submetidos e quais resultados são esperados a partir de uma certa entrada de dados. O projetista pode criar uma classe de ação para cada caso de uso ou agrupar casos de uso em uma mesma classe de ação. Também fica a cargo do projetista definir se deve representar várias ações em um mesmo diagrama ou se deve ter um diagrama para cada ação (SOUZA, 2007).
- **Modelo de Aplicação:** “o Modelo de Aplicação é um diagrama de classes da UML que representa as classes de serviço, que são responsáveis pela codificação dos casos de uso, e suas dependências” (SOUZA, 2007, p. 89). A granularidade das classes de serviço deve ser definida pelo projetista com base nos casos de uso modelados na fase anterior do processo. Assim como no Modelo de Persistência, deve-se usar as regras de programação por interfaces: cada classe de serviço deve ter

uma interface (com a definição dos métodos que executam a lógica de negócio) e uma implementação concreta dessa interface (SOUZA, 2007). Esse diagrama mostra também as associações entre as classes de serviço e os DAOs necessários para a execução dos casos de uso.

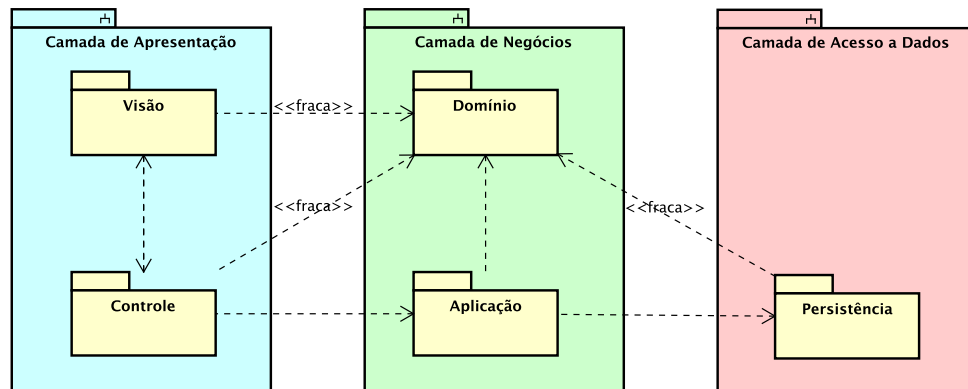


Figura 1 – Arquitetura padrão proposta pelo FrameWeb. Adaptado de (SOUZA, 2007).

Esses quatro modelos, segundo Souza (2007), trazem conceitos utilizados pelos *frameworks* para a fase de Projeto por meio da criação de um perfil UML (a linguagem de modelagem define extensões leves ao meta-modelo da UML para representar componentes típicos da plataforma Web e dos *frameworks* utilizados) que faz com que os diagramas fiquem mais próximos da implementação. Dessa forma, “a fase de codificação é facilitada pelo uso dos *frameworks*, especialmente porque os modelos de projeto exibem componentes relacionados a eles” (SOUZA, 2007, p. 72).

Em uma publicação mais recente, Souza (2020) resume os avanços incorporados ao método FrameWeb desde sua proposta inicial, detalhada em (SOUZA, 2007).

2.4 Frameworks

De acordo com Souza (2007, p. 42), “um *framework* é visto como um artefato de código que provê componentes prontos que podem ser reutilizados mediante configuração, composição ou herança”. Souza (2007) ainda afirma que o uso combinado de alguns desses *frameworks* permite a construção de sistemas Web de grande porte sem muito esforço de codificação.

Segundo Gabrieli et al. (2007), *frameworks* são infra-estruturas de aplicações projetadas para serem reutilizadas. Portanto, um *framework* é uma arquitetura semi-pronta de aplicação e representa o nível mais alto de reutilização (CUNHA; HERBERT, 2003). A redução da manutenção e a maximização do reuso são dois dos ganhos conseguidos com a utilização dos *frameworks* (GAMMA et al., 2000).

“As **ferramentas** da engenharia de software fornecem suporte automatizado ou semiautomatizado para o processo e para os métodos” (PRESSMAN, 2011, p. 40). Considerando a metodologia de processo genérica descrita na Seção 2.2 e as definições anteriores, pode-se dizer que os *frameworks* são **ferramentas** que auxiliam a atividade de Construção.

2.4.1 Frameworks MVC

MVC é a abreviatura de Modelo-Visão-Controlador (*Model-View-Controller*). O padrão MVC, desenvolvido no final dos anos 70 para uso na plataforma Smalltalk, “divide a interação da interface com o usuário em três papéis distintos” (FOWLER, 2002, p. 330, *tradução nossa*).

De acordo com Fowler (2002), o Modelo é um objeto que representa alguma informação sobre o domínio do problema, contendo todos os dados e comportamentos que não estejam relacionados com a interface de usuário. A Visão representa a exibição do Modelo na interface com o usuário, sendo este seu papel específico: exibição de informações; qualquer necessidade de alteração nos dados do Modelo é tratada pelo terceiro membro da tríade MVC: o Controlador.

O funcionamento do padrão MVC é apresentado na Figura 2. O Controlador recebe a entrada do usuário por meio de um estímulo vindo da Visão, manipula o Modelo e, se necessário, seleciona um elemento de Visão para exibir o resultado. O Modelo notifica à Visão as alterações que ocorreram em sua estrutura. Dessa forma, a Visão fica ciente da necessidade de consultar novamente o estado do Modelo (SOUZA, 2007). Segundo Fowler (2002) é provável que existam diversas Visões de um Modelo em uma mesma tela em interfaces de usuário mais complexas ou “ricas”. Dessa forma, se o usuário fizer uma alteração no Modelo por meio de uma Visão, as outras precisam refletir essa mudança (para manter a consistência das informações apresentadas). Segundo Fowler (2002), para se fazer isso sem criar uma dependência, geralmente é implementado o padrão *Observer*, onde a Visão é o *listener* (ouvinte) dos eventos emitidos pelo Modelo.

A arquitetura MVC veio ao encontro das necessidades dos aplicativos Web (SOUZA, 2007). Porém, um observador mais rigoroso notará que ela não se encaixa perfeitamente nessa plataforma, pois o Modelo, situado no servidor Web, não consegue notificar a Visão sobre alterações, já que esta última se encontra no navegador do cliente e a navegação sempre é realizada no sentido cliente-servidor (SOUZA, 2007). Dessa forma, o nome correto para esse padrão arquitetônico quando aplicado à Web, é “Controlador Frontal” (*Front Controller*) (SOUZA, 2007, p. 43 apud ALUR et al., 2003).

De acordo com Fowler (2002, p. 344), Controlador Frontal é “um controlador que lida com todas as solicitações de um site”. Seu funcionamento é ilustrado na Figura 3: o servidor Web entrega toda requisição vinda do navegador do cliente para o Controlador

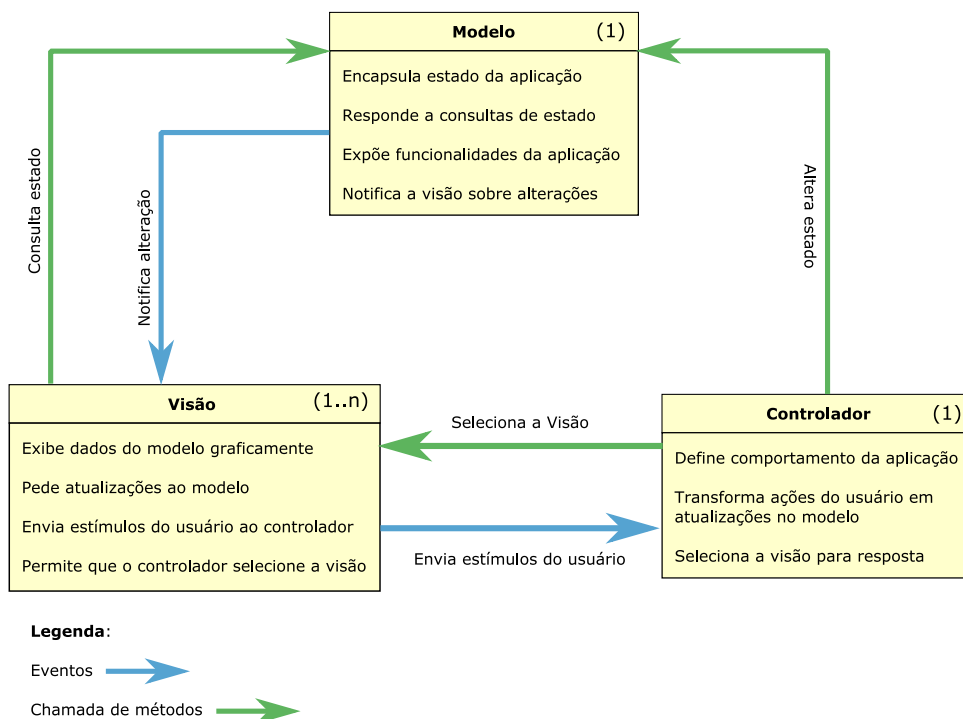


Figura 2 – Funcionamento do padrão MVC. Adaptado de (SOUZA, 2007).

Frontal, o qual analisa a solicitação para decidir que tipo de ação iniciar, cria uma instância de uma classe de ação específica e, em seguida, delega a esse objeto a execução do serviço. Por fim, o Controlador Frontal recebe o resultado do processamento e decide que tipo de resposta enviar para o navegador, que pode ser a construção de uma página, redirecionamento a outra página, montagem e exibição de um relatório, dentre várias possibilidades (SOUZA, 2007).

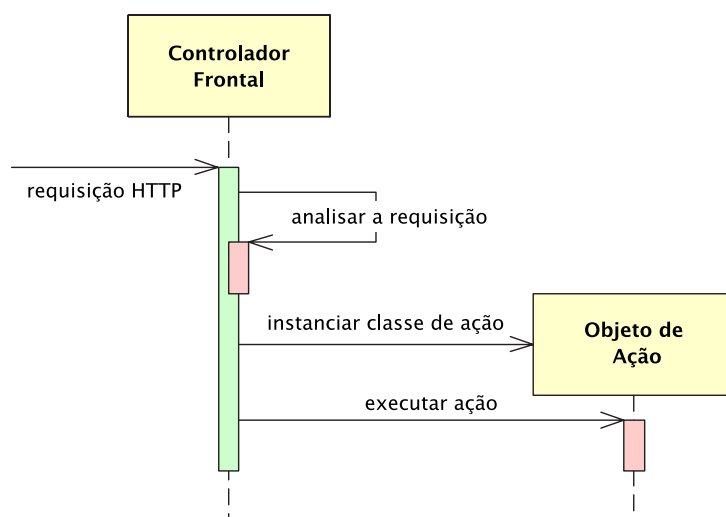


Figura 3 – Funcionamento do Controlador Frontal. Adaptado de (FOWLER, 2002).

Frameworks MVC implementam exatamente essa arquitetura, deixando a cargo do desenvolvedor a tarefa de configurar o *framework*, escrever as classes de ação e as páginas da Web que serão retornadas como resultado. Além disso, esse tipo de *framework* também

fornece uma série de outras facilidades, como conversão automática de tipos, validação de dados de entrada e internacionalização de páginas Web (SOUZA, 2007).

2.4.2 Frameworks ORM

Segundo Souza (2007), Sistemas de Bancos de Dados Relacionais (SGBDRs) há décadas têm sido o padrão de fato para persistência de dados e não há indícios de que este cenário vá mudar tão cedo. Bauer e King (2005) dizem que a tecnologia relacional é profundamente conhecida e isso por si só é motivo suficiente para que muitas organizações a escolham. “Os bancos de dados relacionais estão tão arraigados não é por acidente, mas porque são uma abordagem incrivelmente flexível e robusta para gerenciamento de dados” (BAUER; KING, 2005, p. 3). Conseqüentemente, “mesmo aplicações desenvolvidas no paradigma orientado a objetos (OO) têm utilizado bancos de dados relacionais para persistência de seus objetos” (SOUZA, 2007, p. 46).

Porém existe um pequeno problema ao unir o paradigma de programação orientado a objetos com o mundo dos bancos de dados relacionais: as operações SQL (linguagem utilizada para manipular os dados em um banco de dados relacional), como projeção e junção, sempre resultam em uma representação tabular dos dados, o que é bem diferente do grafo de objetos interconectados usado para executar a lógica de negócios em um aplicativo que foi construído usando OO (BAUER; KING, 2005). “Esses são modelos fundamentalmente diferentes, não apenas maneiras diferentes de visualizar o mesmo modelo” (BAUER; KING, 2005, p. 7, *tradução nossa*). Existe, portanto, uma incompatibilidade de paradigmas.

Para solucionar esse problema surgiu o “Mapeamento Objeto/Relacional”, ou ORM. Bauer e King (2005) utilizam a barra entre as palavras “Objeto” e “Relacional” para enfatizar o problema de incompatibilidade que ocorre quando os dois mundos colidem. Os mesmos autores definem ORM como sendo a persistência automatizada (e transparente) de objetos em um aplicativo (construído usando OO) para as tabelas em um banco de dados relacional, usando metadados que descrevem o mapeamento entre os objetos e o banco de dados. Assim, pode-se dizer que, em essência, o ORM funciona transformando dados de uma representação para outra.

Os *frameworks* ORM, que são responsáveis por trazer, de forma simplificada, o benefício do mapeamento Objeto/Relacional para dentro do projeto, esperam que o programador informe, por meio de metadados, como deve ser feita a transformação dos objetos e seus atributos em tabelas e colunas do SGBDR. Em contrapartida, disponibilizam métodos simples como salvar, excluir e recuperar, além de uma linguagem de consulta, similar à SQL, porém orientada a objetos, para que consultas mais elaboradas possam ser realizadas com igual facilidade (SOUZA, 2007).

Fowler (2002), no capítulo “*Data Source Architectural Patterns*” (Padrões de Arquitetura de Fonte de Dados) explica alguns padrões comumente utilizados por ferramentas de persistência de objetos em ORM. Segundo o autor, em um projeto inicial para um Modelo de Domínio, a escolha principal recai entre dois padrões: o *Active Record* e o *Data Mapper*, os quais serão apresentados a seguir.

2.4.2.1 Padrão *Active Record*

Segundo Fowler (2002), no padrão *Active Record* as classes de domínio correspondem muito de perto à estrutura de registro do banco de dados, ou seja, cada classe do domínio representa uma tabela (ou uma *view*) do banco, cujas colunas são representadas por propriedades na classe. Uma instância de uma classe (objeto) representa um único registro em uma tabela.

Ainda segundo Fowler (2002), esse padrão coloca a lógica de acesso a dados nas classes do domínio (elas podem herdar os métodos de uma classe *Active Record* se for mais conveniente) de forma que cada objeto sabe como ler e gravar seus próprios dados no banco de dados. Além disso, cada classe do domínio é responsável por qualquer lógica de negócio que atue sobre os dados.

A Figura 4 foi adaptada de Fowler (2002) e exemplifica o padrão *Active Record*. É possível ver uma classe “Pessoa” com três atributos persistentes (nome, sobrenome e numeroDeDependentes), três métodos de lógica de acesso a dados (inserir, atualizar e excluir) e três métodos da lógica de negócios (recuperarIsencao, estaMarcadoParaAuditoria e recuperarRendimentosTributaveis) em um relacionamento de dependência com o banco de dados.

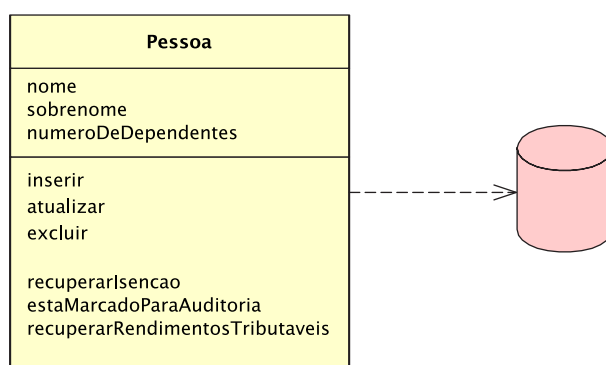


Figura 4 – Padrão *Active Record*. Adaptado de (FOWLER, 2002).

A classe *Active Record* normalmente tem métodos que fazem o seguinte (FOWLER, 2002, p. 161, tradução nossa):

- Construir uma instância do *Active Record* a partir de uma linha de conjunto de resultados SQL;

- Construir uma nova instância para posterior inserção na tabela;
- Métodos de localização estáticos para envolver consultas SQL comumente usadas e retornar objetos *Active Record*;
- Atualizar o banco de dados e inserir nele os dados do *Active Record*;
- Obter e definir os campos;
- Implementar algumas peças da lógica de negócios.

Segundo Fowler (2002), o padrão *Active Record* é uma boa escolha para a lógica de domínio que não seja muito complexa, como um sistema que seja fortemente baseado em CRUD (operações básicas de cadastro, do inglês *Create, Retrieve, Update, Delete*). A principal vantagem é a simplicidade de implementação e compreensão do padrão.

O principal problema relacionado ao seu uso, segundo Fowler (2002), é que esse padrão funciona bem somente se os objetos do tipo *Active Record* corresponderem diretamente às tabelas no banco de dados em um esquema isomórfico (se há um mapeamento bijetivo entre elas, ou seja, cada propriedade em um objeto tem sua respectiva coluna na tabela e vice versa).

Outro argumento citado por Fowler (2002) contra o uso do *Active Record* é o fato de que ele acopla o projeto dos objetos ao projeto do banco de dados, tornando mais difícil refatorar o sistema ao longo do tempo.

2.4.2.2 Padrão *Data Mapper*

Como já foi mencionado no início da Seção 2.4, objetos e bancos de dados relacionais usam mecanismos diferentes para estruturar os dados, de forma que alguns conceitos do mundo dos objetos (como coleções e herança) não existem nesses bancos. Porém, o uso desses conceitos é muito valioso para organizar os dados em aplicações cuja lógica de negócios é extensa (FOWLER, 2002). O padrão *Active Record*, explicado anteriormente, funciona bem quando há uma correspondência direta entre os atributos dos objetos e as colunas das tabelas relacionadas, porém quando se quer evoluir o modelo dos objetos de forma independente do esquema do banco de dados (e vice versa), o padrão *Data Mapper* é mais indicado (FOWLER, 2002).

O padrão *Data Mapper* cria uma camada de “Mapeadores” que movem os dados entre os objetos e o banco de dados, mantendo-os independentes um do outro (FOWLER, 2002). “Com o *Data Mapper*, os objetos na memória não precisam nem mesmo saber que há um banco de dados presente. Eles não precisam conter comandos SQL e certamente não precisam ter nenhum conhecimento do esquema do banco de dados” (FOWLER, 2002, p. 165, tradução nossa). Fowler (2002) ainda afirma que isso ajuda na etapa de codificação

pois é possível trabalhar com os objetos do domínio sem ter que entender como eles estão armazenados no banco de dados.

A Figura 5 foi adaptada de Fowler (2002) e exemplifica o padrão *Data Mapper*. Comparando-a com a Figura 4 pode-se ver que as operações referentes à lógica de acesso a dados foram movidas para uma camada intermediária (Mapeador Pessoa) a qual tem relacionamentos de dependência com o banco de dados e com a classe de domínio em substituição àquele relacionamento direto que existia entre estes dois.

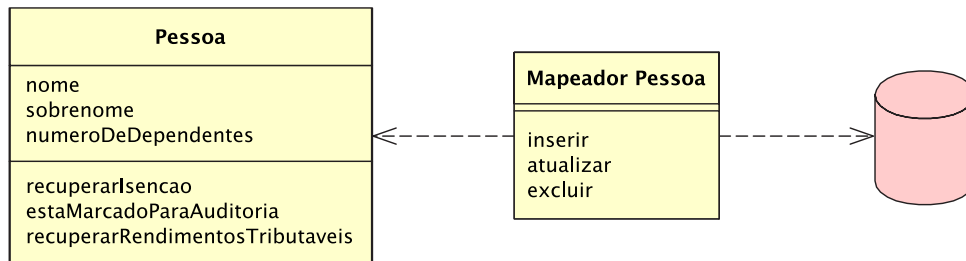


Figura 5 – Padrão Data Mapper. Adaptado de (FOWLER, 2002).

Segundo Fowler (2002), o “preço” que se paga pelo uso do *Data Mapper* em relação ao *Active Record* é a implementação dessa camada adicional, de forma que optar por um ou outro padrão vai depender da complexidade da lógica de negócio. “Se o modelo do domínio for bastante simples e o banco de dados estiver sob o controle do desenvolvedor do modelo do domínio, então é razoável que os objetos do domínio acessem diretamente o banco de dados por meio do *Active Record*” (FOWLER, 2002, p. 170, tradução nossa).

3 Especificação de Requisitos e Análise

Como já foi mencionado na Introdução (Capítulo 1), o objetivo deste trabalho é realizar um novo projeto e uma nova implementação do SCAP, um sistema Web criado para gerenciar o afastamento dos professores do Departamento de Informática (DI) da UFES. Duarte (2014) e Prado (2015) já realizaram todo o levantamento dos requisitos e análise do SCAP, os quais serão apresentados neste capítulo, juntamente com algumas propostas de melhorias.

3.1 Descrição do Escopo

As solicitações de afastamento gerenciadas pelo SCAP podem ser para participação em eventos (cursos, palestras, *workshops*, seminários, entre outros) realizados no Brasil ou no exterior. De forma resumida, a tramitação das solicitações feitas por professores do DI ocorre da seguinte forma:

- Para os eventos realizados **em solo nacional**, a solicitação de afastamento precisa ser aprovada pelo DI. Assim, a solicitação deve ser enviada a todos os professores do DI via email. Caso ninguém se manifeste contra o afastamento dentro de dez dias, a solicitação é aprovada. Se houver uma manifestação contrária ao afastamento, o caso é discutido em uma reunião do DI, na qual decide-se pelo deferimento ou indeferimento. Dessa forma, o processo ocorre inteiramente dentro do departamento.
- Para os eventos realizados **fora do Brasil**, a solicitação de afastamento precisa ser aprovada pelo DI, pelo Centro Tecnológico (CT) e pela Pró-Reitoria de Pesquisa e Pós-Graduação (PRPPG). Neste caso, o Chefe do Departamento, ao ter ciência de uma nova solicitação de afastamento, escolhe um professor (que não tenha parentesco com o solicitante) para ser o relator da solicitação, o qual deve emitir um parecer recomendando a aprovação ou reprovação do afastamento no âmbito do DI. Independentemente da recomendação do relator, o processo deve aguardar dez dias, que é o prazo para que outros professores se manifestem contra ele, exatamente como no caso anterior. Se houver uma manifestação contrária ao afastamento, o caso é discutido em uma reunião do DI, na qual decide-se pelo deferimento ou indeferimento da solicitação no âmbito do DI. Se não houver manifestação contrária, a solicitação é considerada aprovada pelo DI e é então encaminhada ao CT e posteriormente à PRPPG. O afastamento só é deferido se a solicitação for aprovada em todas as instâncias.

O SCAP foi idealizado com o objetivo de assistir os professores e secretários do DI na tramitação das solicitações de afastamento, tornando mais simples todo o processo, tanto a criação da solicitação quanto a análise e armazenamento da mesma. O sistema atinge esse objetivo através do envio automático de emails aos envolvidos e da utilização de formulários para inserção e atualização das informações.

O escopo do sistema é limitado aos procedimentos realizados no DI. Não existe integração com os processos do CT e da PRPPG. Dessa forma, os pareceres do CT e da PRPPG são inseridos manualmente no SCAP.

3.2 Modelo de Casos de Uso

A Tabela 1 lista e descreve os atores identificados por Duarte (2014) no levantamento de requisitos.

Tabela 1 – Atores do SCAP.

Ator	Descrição
Professor	Professores efetivos do DI
Chefe do Departamento	Professores do DI que estão exercendo a função administrativa de “Chefe do Departamento” ou “Subchefe do Departamento”
Secretário	Secretários do DI

O **Secretário** lida com a parte administrativa do sistema, sendo sua responsabilidade cadastrar novos professores e manter esses cadastros atualizados, principalmente os relacionamentos de parentesco entre professores e os mandatos daqueles que forem (ou tiverem sido) chefe ou subchefe do departamento. Além disso, é responsável por cadastrar as decisões do CT e da PRPPG (quando for um afastamento internacional) e por arquivar os processos de solicitação de afastamento já finalizados.

O **Professor** é o ator principal do sistema. Ele realiza solicitações de afastamento e, se julgar necessário e ainda houver tempo hábil, pode se manifestar contra um afastamento de outro professor. Se for escolhido como relator de um afastamento internacional, deve recomendar ao DI aprovar ou reprovar tal afastamento.

O **Chefe do Departamento** tem a responsabilidade de escolher um relator para cada afastamento internacional que for cadastrado no sistema.

A Figura 6 apresenta todos os casos de uso do SCAP. É possível ver os três atores citados anteriormente, cada qual relacionado com os casos de uso de sua competência. Em seguida é apresentada uma listagem com uma breve descrição de cada caso de uso. Vale ressaltar que originalmente o SCAP foi dividido em dois subsistemas. Segundo Prado (2015), tal divisão foi realizada para facilitar a implementação do sistema e não devido a

natureza das classes criadas. Neste trabalho, no entanto, optou-se por não utilizar esta subdivisão, pois não foi identificada tal facilidade neste contexto.

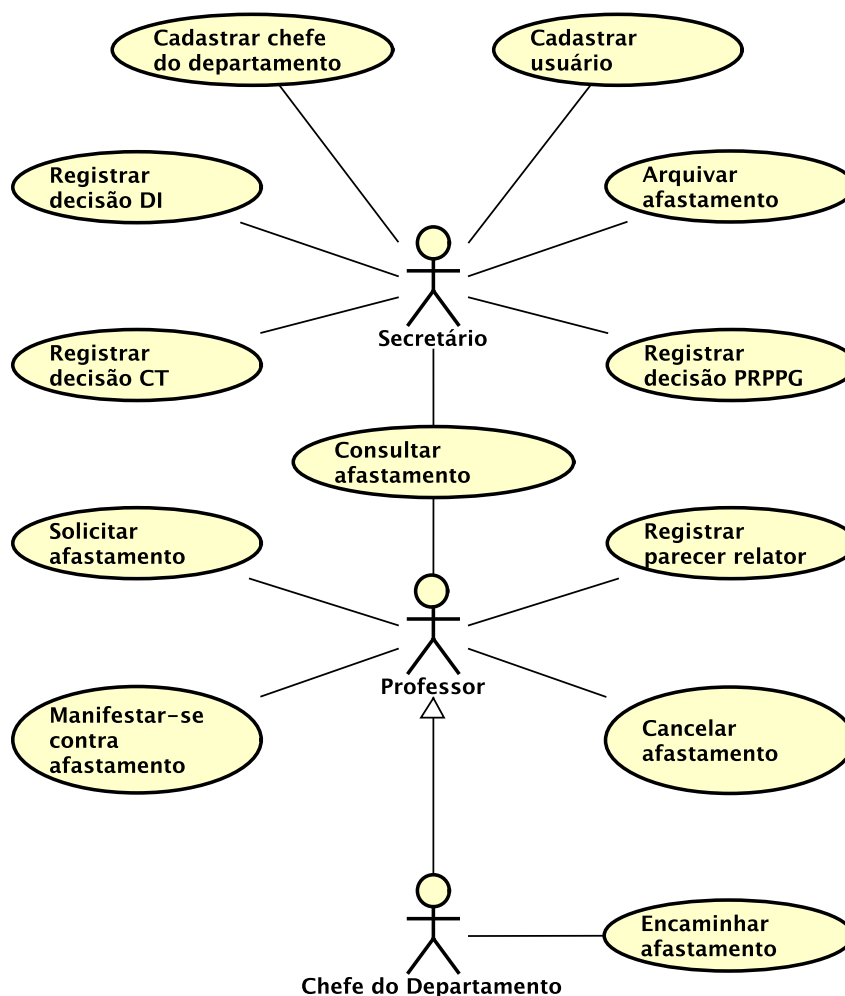


Figura 6 – Diagrama de Casos de Uso do SCAP.

- **Solicitar afastamento:** um professor realiza uma solicitação de afastamento no sistema, informando todos os dados necessários para a tramitação da mesma;
- **Cancelar afastamento:** o professor cancela uma solicitação de afastamento da qual seja o solicitante;
- **Encaminhar afastamento:** o chefe do departamento, após analisar um pedido de afastamento internacional recém-cadastrado, escolhe um professor (que não tenha relação de parentesco com o solicitante) como relator do afastamento;
- **Deferir parecer Registrar parecer relator:** o professor relator de um afastamento internacional cadastra seu parecer sobre o mesmo;
- **Consultar afastamento:** um professor ou um secretário consulta uma lista de afastamentos cadastrados ou fornece características (dados) de um certo afastamento a fim de localizá-lo no sistema;

- **Manifestar-se contra afastamento:** um professor cadastra um parecer desfavorável a um afastamento nacional. Nesse caso, o deferimento ou indeferimento do afastamento é decidido em uma reunião do DI;
- **Cadastrar usuário:** um secretário cadastra um novo professor ou um novo secretário no sistema, informando os dados pessoais necessários;
- **Cadastrar chefe do departamento:** um secretário insere um mandato no cadastro do professor correspondente (que é o chefe do departamento naquele mandato), informando as datas de início e fim do mesmo;
- **Registrar decisão DI:** um secretário registra a decisão do Departamento de Informática sobre um pedido de afastamento, após a realização de uma reunião, nos casos em que um professor se manifesta contra o afastamento de outro;
- ~~Registrar parecer CT~~ **Registrar decisão CT:** um secretário registra a decisão do Centro Tecnológico sobre um pedido de afastamento internacional;
- ~~Registrar parecer PRPPG~~ **Registrar decisão PRPPG:** um secretário registra a decisão da Pró-Reitoria de Pesquisa e Pós-Graduação sobre um pedido de afastamento internacional;
- **Arquivar afastamento:** um secretário arquiva uma solicitação de afastamento que já está encerrada, ou seja, já está deferida ou indeferida.

O caso de uso “Deferir parecer” foi renomeado para “Registrar parecer relator”. Isso porque o termo “deferir” significa, entre outras coisas, “conceder; atender; despachar (um requerimento) em conformidade com o que nele se pede; ter acatamento” (FERNANDES; LUFT; GUIMARÃES, 1996). No meio jurídico, quando se está falando de processos, o termo “deferido” é usado por certas autoridades nos requerimentos que lhes são dirigidos, quando concedem o que lhes foi pedido (FERNANDES; LUFT; GUIMARÃES, 1996). Considerando que a descrição do caso de uso em Duarte (2014) indica que o parecer, nesse caso, também pode ser desfavorável, usar o termo “deferir” pode gerar certa confusão. Assim, para eliminar qualquer ambiguidade, será usada a expressão “Registrar parecer relator”, mantendo o mesmo padrão de nomes de outros casos de uso e deixando bem claro, inclusive, que esse caso de uso trata exclusivamente de um parecer de um relator.

Os casos de uso “Registrar parecer CT” e “Registrar parecer PRPPG” foram renomeados respectivamente para “Registrar decisão CT” e “Registrar decisão PRPPG” para que não haja o falso entendimento de que esses casos de uso geram Pareceres no sistema. O que de fato ocorre é a mudança da situação do Afastamento e a inclusão (*upload*) de um Documento (um arquivo contendo a decisão do CT ou da PRPPG).

Além disso, foi acrescentado um novo caso de uso para o ator “Secretário”: “Registrar decisão DI”. De acordo com o Documento de Especificação de Requisitos produzido por Duarte (2014) e complementado Prado (2015), quando um professor se manifesta contra uma solicitação de afastamento nacional, o deferimento ou indeferimento da mesma depende de uma reunião do DI. Após essa reunião um secretário deve registrar a decisão do DI no sistema. Porém, essa ação do secretário não foi listada como um caso de uso separado, mas foi embutida no caso de uso “Manifestar-se contra afastamento”, que é de responsabilidade de um professor. Para deixar explícita essa ação do secretário, criou-se o caso de uso “Registrar decisão DI”. A descrição do caso de uso e suas precondições são exibidas na Tabela 2.

Tabela 2 – Caso de uso “Registrar decisão DI”.

Caso de uso: Registrar decisão DI. Descrição sucinta: Um secretário registra a decisão do DI quando houve uma manifestação contrária à um afastamento.		
Nome do fluxo de eventos normal	Precondições	Descrição
Registrar decisão DI	1 - Uma solicitação de afastamento nacional deve ter sido criada. 2 - A situação do afastamento deve ser “Aguardando decisão DI”. 3 - Algum professor deve ter se manifestado contra o afastamento.	1 - Um secretário, após a reunião do DI, muda a situação do afastamento e faz <i>upload</i> do documento contendo a decisão do DI (geralmente um trecho da ata da reunião).
Requisitos relacionados: RF09. Classes relacionadas: Secretário, Afastamento, Parecer.		

3.3 Modelo de Classes

O Diagrama de Classes do SCAP é apresentado na Figura 7, onde é possível ver todas as classes identificadas por Duarte (2014) e Prado (2015) a fim de adaptar as entidades do mundo real para o paradigma OO:

- A classe **Afastamento** representa uma solicitação de afastamento feita por algum professor. Seu atributo “situacao” indica em qual estágio da tramitação o afastamento se encontra. O tipo enumerado “Situacao”, exibido na Figura 8, lista os possíveis valores para este atributo. Já o atributo “tipo_afastamento” indica se o afastamento

é nacional ou internacional. Seus possíveis valores são indicados no tipo enumerado “TipoAfastamento” da Figura 8.

- A classe **Documento** representa os documentos que podem ser anexados pelo solicitante (para comprovar a veracidade das informações) ou por um secretário, ao cadastrar as decisões do DI / CT / PRPPG.
- A classe **Parecer** representa um parecer emitido por um professor que se manifestou contra o afastamento ou pelo professor relator do afastamento. O atributo “tipo_parecer” indica se o parecer é favorável ou desfavorável. Seus valores possíveis estão na Figura 8, no tipo enumerado “TipoParecer”.
- As classes **Professor** e **Secretário** representam, respectivamente, professores e secretários. Ambas são subclasses da classe **Pessoa**. Um professor pode ter relações de parentesco com outros professores, as quais são representadas pelo autorrelacionamento na classe “Professor”.
- A classe **Mandato** representa um período no qual um professor é (ou foi) o chefe ou subchefe do departamento. No mundo real, o subchefe do departamento é o substituto imediato do chefe, atuando somente quando este estiver impossibilitado por algum motivo. Porém, como o escopo deste sistema é reduzido, não há diferenciação entre eles: permite-se que o subchefe atue em conjunto com o chefe (há uma restrição de integridade, na Seção 3.4, para garantir que apenas dois professores atuem ao mesmo tempo). Dessa forma, ambos são tratados como “chefe do departamento” e não há a necessidade de um atributo na classe “Mandato” para diferenciá-los.

Uma modificação que foi feita neste trabalho em relação aos anteriores é a seguinte:

- **Remoção do atributo “parentesco” da classe “Parentesco”:** Prado (2015), quando acrescentou a classe associativa “Parentesco” ao modelo de classes do SCAP, definiu um único atributo nomeado “parentesco” que poderia assumir os valores “sanguineo” ou “matrimonial”. Porém não há tratamento diferenciado entre esses dois tipos de parentesco: tudo que vale para um vale para o outro. Sendo assim, como esse dado não tem utilidade dentro do escopo do sistema, ele foi removido. Conseqüentemente, a representação da classe associativa “Parentesco” deixou de ser necessária no diagrama de classes.

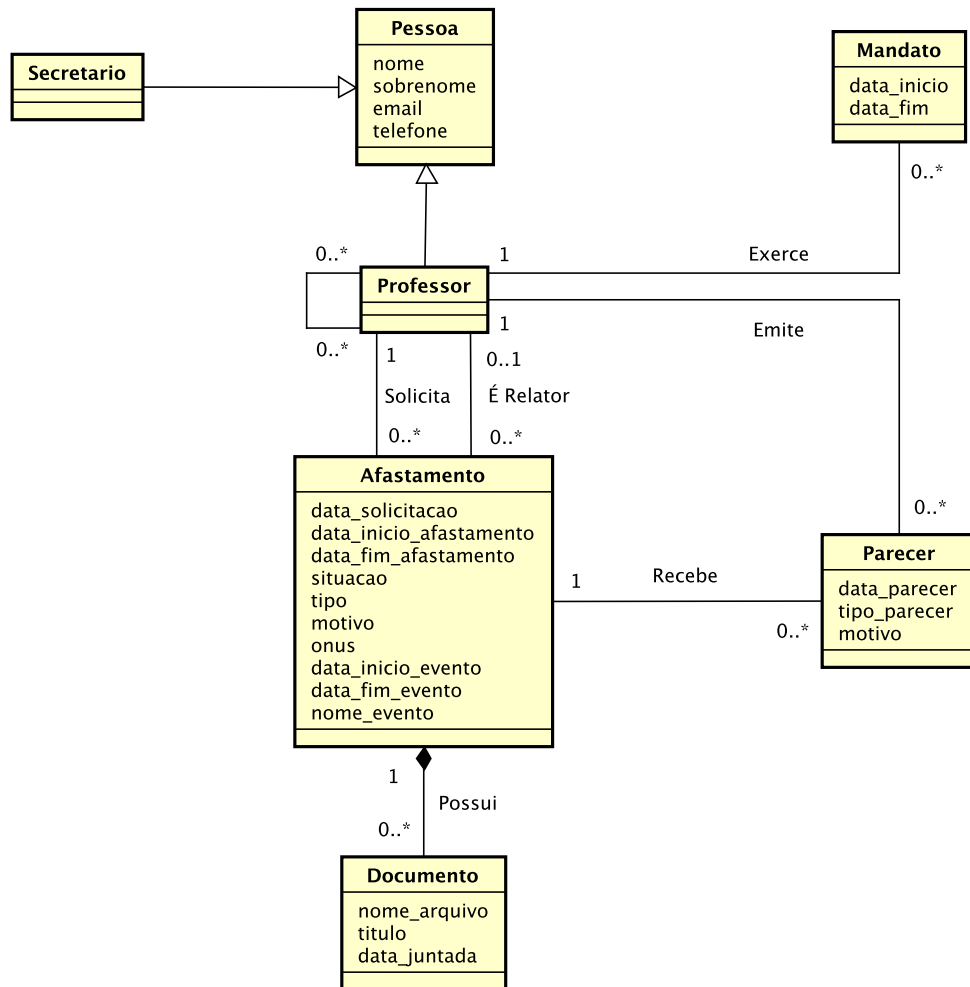


Figura 7 – Diagrama de Classes do SCAP.

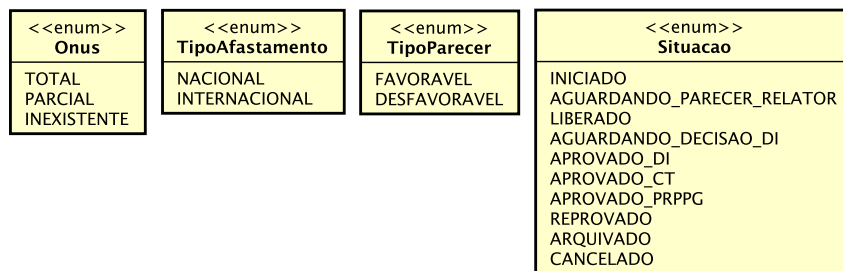


Figura 8 – Tipos enumerados do SCAP.

3.4 Restrições de Integridade

A seguir são listadas as restrições de integridade observadas por Duarte (2014) e complementadas por Prado (2015):

1. Um professor não pode ser solicitado a dar um parecer sobre sua própria solicitação de afastamento.

2. A data de início de um afastamento não pode ser posterior a data de fim do mesmo afastamento.
3. A data de início de um mandato de professor não pode ser posterior a data de fim do mesmo mandato.
4. Não pode haver mais de dois professores (chefe e subchefe de departamento) exercendo um mandato ao mesmo tempo.
5. Um secretário não pode criar uma solicitação de afastamento.
6. Um professor não pode ser relator de um afastamento solicitado por um parente.
7. Uma pessoa deve ser ou secretário ou professor (não pode ser ambos).

4 Projeto e Implementação

O SCAP já foi implementado anteriormente por vários estudantes de graduação em seus trabalhos de conclusão de curso, sempre utilizando o método FrameWeb na fase de Projeto. A Tabela 3 lista todas as implementações do SCAP feitas em trabalhos anteriores e as duas implementações realizadas neste trabalho, indicando o *framework* MCV utilizado com a linguagem de programação, as respectivas tecnologias ORM com o padrão adotado pelas mesmas e as respectivas tecnologias DI.

Tabela 3 – Implementações anteriores do SCAP.

Trabalho	<i>Framework</i> MVC	Tecnologia ORM	Tecnologia DI
Duarte (2014)	JSF (Java)	Hibernate ¹	CDI
Prado (2015)	VRaptor (Java)	Hibernate ¹	CDI
Pinheiro (2017)	Laravel (PHP)	Eloquent ²	Laravel
Matos (2017)	Spring (Java)	Hibernate ¹	Spring
Matos (2017)	Vaadin (Java)	Hibernate ¹	Spring
Avelar (2018)	Ninja (Java)	Hibernate ¹	Google Guice
Ferreira (2018)	Wicket (Java)	Hibernate ¹	CDI
Ferreira (2018)	Tapestry (Java)	Hibernate ¹	Tapestry
Meirelles (2019)	CodeIgniter (PHP)	CodeIgniter ²	Não utilizou
Meirelles (2019)	NodeJS (JavaScript)	Não utilizou	Não utilizou
Guterres (2019)	Play (Scala)	Slick ³	Google Guice
<i>Este trabalho</i>	Yii (PHP)	Yii ²	Yii
<i>Este trabalho</i>	Symfony (PHP)	Doctrine ¹	Symfony

¹ Padrão *Data Mapper*

² Padrão *Active Record*

³ Padrão não identificado

No decorrer deste capítulo será apresentada a plataforma de desenvolvimento, incluindo as tecnologias e os softwares de apoio utilizados nas duas novas implementações do SCAP e o projeto do sistema, materializado nos diagramas propostos pelo FrameWeb, além de detalhes relevantes da implementação.

4.1 Plataforma de desenvolvimento

A Tabela 4 lista e descreve as tecnologias usadas no projeto e nas implementações do SCAP. Logo em seguida, as seções 4.1.1 e 4.1.2 apresentam brevemente os *frameworks* MVC utilizados. Por último, a Tabela 5 lista e descreve os softwares que apoiaram o

desenvolvimento do projeto, tanto na etapa de Projeto quanto na Implementação.

Tabela 4 – Tecnologias utilizadas no desenvolvimento do projeto

Tecnologia	Versão	Descrição	Propósito	Y ¹	S ²
PHP	7.3.1	Linguagem de programação de código-fonte aberto, interpretada no lado do servidor, especialmente adequada para o desenvolvimento Web.	Escrita do código-fonte das classes que compõem o sistema.	x	x
Apache HTTP Server	2.4.37	Servidor HTTP de código fonte aberto.	Hospedagem da aplicação Web, dando acesso aos usuários via HTTP.	x	x
MariaDB Server	10.1.37	Sistema Gerenciador de Banco de Dados Relacional gratuito.	Armazenamento dos dados manipulados pela ferramenta.	x	x
Bootstrap	3.4.1	Framework Web, de código-fonte aberto, que disponibiliza componentes de interface baseados em HTML, CSS e JavaScript.	Reutilização de componentes visuais Web de alto nível.	x	x
Yii PHP framework	2.0	Conjunto de implementações, em PHP, de padrões de projeto e componentes comumente utilizados no ambiente Web.	Redução da complexidade no desenvolvimento de aplicações Web a partir de seus componentes prontos para o uso.	x	
Symfony PHP framework	5.2	Conjunto de implementações, em PHP, de padrões de projeto e componentes comumente utilizados no ambiente Web.	Redução da complexidade no desenvolvimento de aplicações Web a partir de seus componentes prontos para o uso.		x
Doctrine	2.7	API para persistência de dados por meio de mapeamento objeto/relacional.	Persistência dos objetos de domínio sem necessidade de escrita dos comandos SQL.		x

¹ Foi utilizada na implementação feita com o Yii?

² Foi utilizada na implementação feita com o Symfony?

4.1.1 Yii

O Yii é um *framework* PHP de alta performance baseado em componentes para desenvolvimento rápido de aplicações web modernas. O nome Yii (pronunciado ii) significa “simples e evolutivo” em chinês. Ele também pode ser considerado um acrônimo de *Yes, It Is* (Sim, ele é) (YII, c2008).

O Yii é um *framework* bastante genérico, o que significa que ele pode ser usado para o desenvolvimento de todo tipo de aplicações Web usando PHP, como por exemplo portais, fóruns, sistemas de gerenciamento de conteúdo (CMS), projetos de *e-commerce*, Web *services* RESTful e assim por diante (YII, c2008).

Como a maioria dos *frameworks* PHP, o Yii implementa o padrão de arquitetura MVC (Modelo-Visão-Controlador) e promove a organização do código baseada nesse padrão. Sua principal filosofia é de que o código deve ser escrito de uma maneira simples, porém elegante (YII, c2008).

Atualmente, o Yii tem duas versões principais disponíveis: a 1.1 e a 2.0. A versão 1.1 é a antiga geração e agora está em modo de manutenção. A versão 2.0 representa a geração atual do *framework* e é uma reescrita completa do Yii, adotando as tecnologias e protocolos mais recentes, incluindo Composer, PSR, *namespaces*, *traits*, e assim por diante (YII, c2008).

A escolha deste *framework* para o desenvolvimento de uma das novas implementações do SCAP está baseada em dois pontos principais: o primeiro é que ele ainda não foi testado oficialmente com o método FrameWeb; o segundo é que o padrão de projeto utilizado pela tecnologia ORM do Yii (*Active Record*) foi pouco testada em trabalhos anteriores, como pôde ser observado na Tabela 3.

4.1.2 Symfony

Symfony é um dos projetos PHP de maior sucesso. É um *framework full-stack* completo de código aberto para desenvolvimento de aplicativos da Web e um conjunto popular de componentes reutilizáveis (SYMFONY, s.d.a). Foi originalmente concebido em 2005 pela agência interativa SensioLabs para o desenvolvimento de sites para seus próprios clientes (SYMFONY, s.d.b).

Milhares de sites e aplicativos dependem do Symfony como a base de seus serviços da Web. A maioria dos principais projetos PHP, como Drupal e Laravel (e até mesmo o Yii), usam componentes Symfony para construir seus aplicativos (SYMFONY, s.d.b).

Symfony é um ambiente de desenvolvimento estável e reconhecido internacionalmente. É fácil de usar graças à integração de soluções criadas em outros ambientes, como injeção de dependências (baseadas no Java) e soluções desenvolvidas especificamente para

o *framework*, como a *Web Debug Toolbar* e a *Web Profiler*. O *Symfony* não confina o desenvolvedor ao seu ambiente, mas permite que ele escolha os componentes de software que deseja usar ([SYMFONY, s.d.c](#)).

O *Symfony* foi escolhido para desenvolver uma das duas implementações do SCAP deste trabalho pela sua importância na comunidade PHP e pelo fato de ainda não ter sido testado com o *FrameWeb* em trabalhos anteriores.

Tabela 5 – Softwares de apoio ao desenvolvimento do projeto

Tecnologia	Versão	Descrição	Propósito
Astah UML	8.0	Ferramenta de diagramação UML.	Criação dos modelos de Entidades, Aplicação, Persistência e Navegação.
TeX Live	2017	Implementação do \LaTeX .	Documentação do projeto arquitetural do sistema.
TeXstudio	2.12	Editor de texto \LaTeX .	Escrita da documentação do sistema, sendo usado o <i>template abnTeX</i> .
JetBrains PhpStorm	2018.3	Ambiente de desenvolvimento (IDE) com suporte ao desenvolvimento PHP.	Implementação (codificação) da aplicação Web.
Composer	1.6.3	Ferramenta de gerência/construção de projetos de software.	Obtenção e integração das dependências do projeto.

4.2 Arquitetura

O *FrameWeb* ([SOUZA, 2007](#)) indica a utilização de uma arquitetura baseada no padrão Camada de Serviço. A Figura 1 ilustrou a arquitetura proposta pelo *FrameWeb*.

No projeto feito com o *Yii*, no entanto, essa arquitetura não se encaixa perfeitamente pelo fato de a tecnologia ORM utilizada (incorporada no *framework* *Yii*) ser baseada no padrão *Active Record*. Uma das características do *Active Record*, como foi descrito no Capítulo 2, é que os *models* são fortemente acoplados à camada de acesso aos dados, diferentemente do que acontece no padrão Camada de Serviço. Dessa forma, como desejasse experimentar a aplicação do método *FrameWeb* em novas plataformas, utilizando as características e funcionalidades que essas novas plataformas oferecem, abriu-se mão da arquitetura padrão indicada pelo *FrameWeb* (unicamente pela incompatibilidade mencionada) no projeto feito com o *Yii*. Já no projeto feito com o *Symfony* a arquitetura seguiu a indicação do *FrameWeb*.

4.3 Modelo de Entidades

Seguindo as etapas indicadas pelo método FrameWeb para a construção do Modelo de Entidades, utilizou-se como base o diagrama de classes do SCAP (Figura 7) e foi-se ajustando-o para a plataforma de implementação escolhida. A Figura 9 apresenta o Modelo de Entidades finalizado. É possível ver que foram adicionados os mapeamentos de persistência aos atributos — como por exemplo as restrições de não-nulo (*not null*), precisão das datas e horas (*precision*) e número máximo de caracteres (*size*) — e às classes — como o estereótipo «*mapped*» na classe Pessoa.

A Tabela 6 descreve os mapeamentos objeto/relacionais utilizados explicitamente no Modelo de Entidades do SCAP. Há muitos outros mapeamentos possíveis propostos pelo FrameWeb, os quais podem ser encontrados em (SOUZA, 2007).

Tabela 6 – Mapeamentos objeto/relacionais utilizados explicitamente no Modelo de Entidades do SCAP (fragmento da tabela disponível em (SOUZA, 2007)).

Mapeamento	Extensão	Valores Possíveis
Se uma classe é persistente, transiente ou mapeada (a classe não é persistente, mas suas propriedades serão se alguém herdá-las)	Estereótipo de classe	«persistent» «transient» «mapped»
Se um atributo pode ser nulo ou não	Restrição de atributo	null not null
Precisão do tipo “data e hora”: armazenar somente a data, somente a hora ou ambos	Restrição de atributo	precision=(date time timestamp)
Tamanho da coluna na qual o atributo será armazenado (para os tipos que isso se aplica)	Restrição de atributo	size=value

Como o FrameWeb define valores padrão de mapeamentos (que são os mais utilizados, marcados em negrito na Tabela 6) não foi necessário indicar todos os tipos de mapeamento possíveis. Por exemplo: o mapeamento que indica se uma classe é persistente, transiente ou mapeada possui o valor «*persistent*» como padrão, permitindo que seja indicado explicitamente apenas nas classes que possuam um mapeamento diferente deste, que nesse caso é apenas a classe Pessoa.

Este mesmo Modelo de Entidades é utilizado nas duas implementações do SCAP feitas neste trabalho. Não houve nenhuma incompatibilidade entre esse diagrama e os *frameworks* MVC utilizados, pois o modelo descreve basicamente os objetos de domínio do problema e seus mapeamentos para a estrutura do banco de dados relacional, sendo que os *frameworks* MCV tem pouca influência nesse diagrama. Uma prova disso é que em todas as implementações anteriores do SCAP não foi necessário fazer modificações

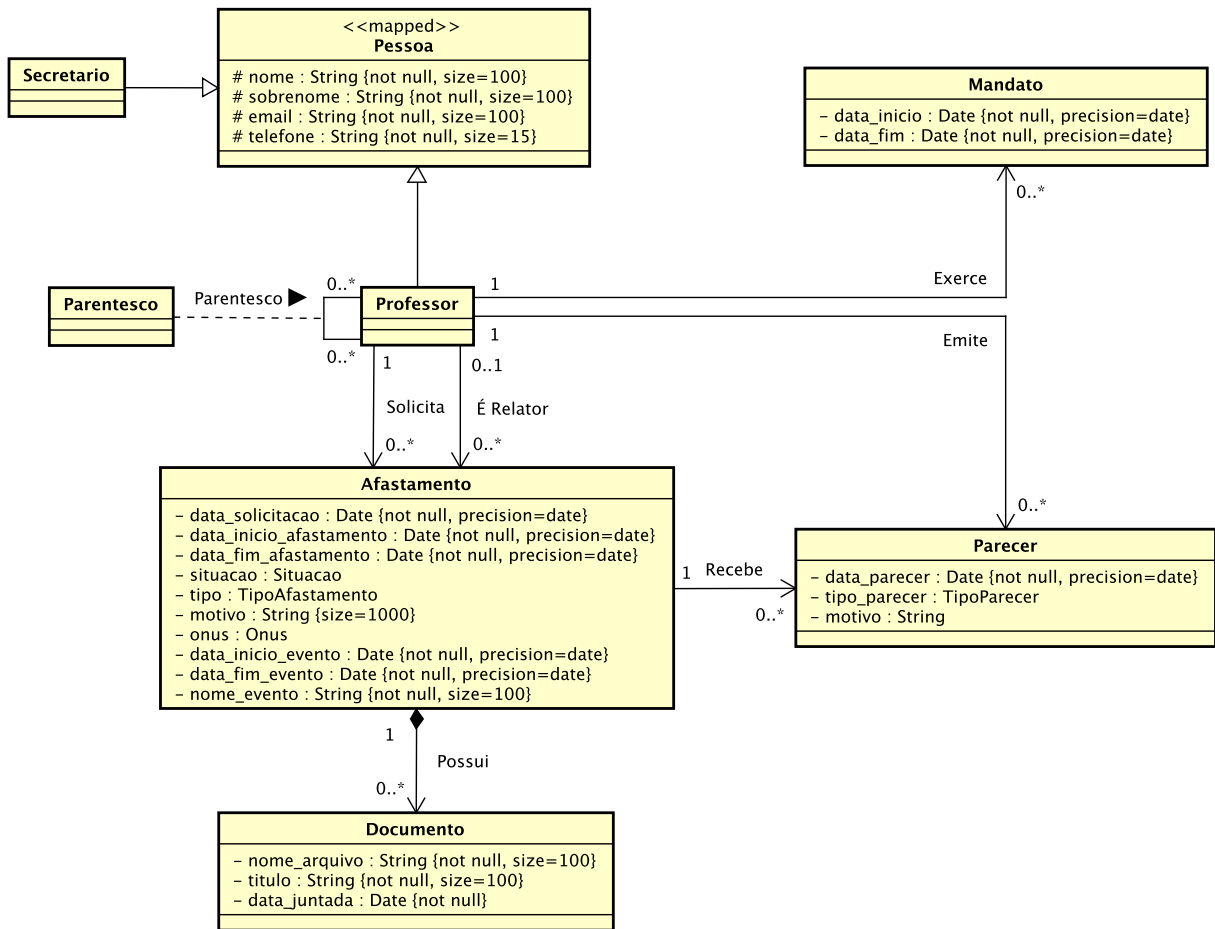


Figura 9 – Modelo de Entidades.

neste digrama por causa do *framework* utilizado (as poucas mudanças que foram feitas até então estão relacionadas ao domínio do problema).

A Figura 10 apresenta um diagrama de estados para as instâncias da classe Afastamento. Cada possível valor do atributo “situacao” (tipo enumerado “Situacao” da Figura 8) gerou um estado no diagrama. Para que o diagrama não ficasse muito poluído, omitiu-se o estado “CANCELADO”, o qual é alcançado por todos os outros estados (menos pelo “REPROVADO” e “ARQUIVADO”) por meio do evento (Caso de Uso) “Cancelar afastamento”.

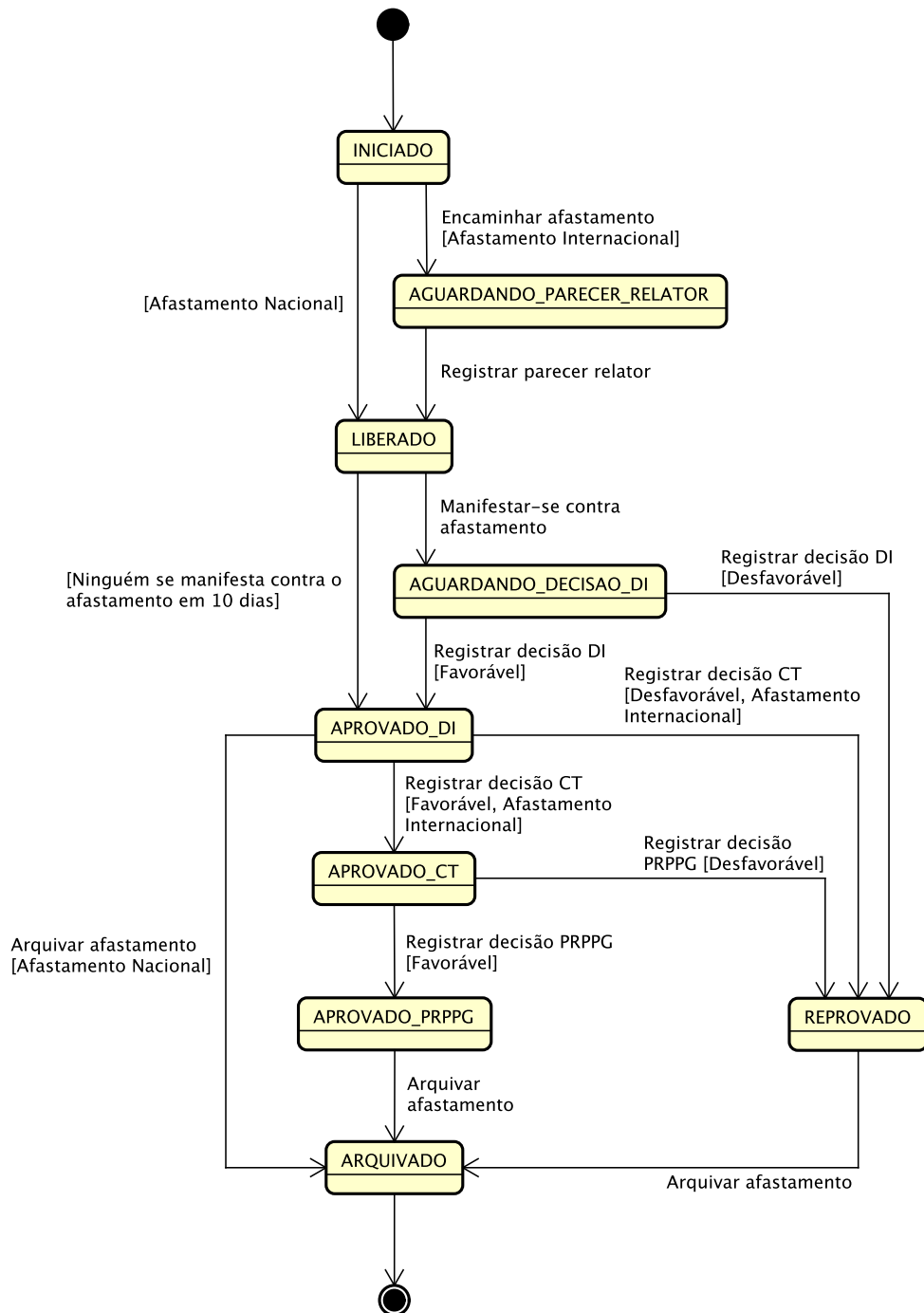


Figura 10 – Diagrama de Estados para instâncias da classe Afastamento.

4.4 Modelo de Navegação

Para guiar a implementação da camada de apresentação, o FrameWeb propõe a construção dos modelos de navegação. Neste trabalho, cada caso de uso originou um modelo de navegação, os quais serão utilizados nas duas implementações do SCAP.

Os modelos de navegação referentes aos casos de uso “Registrar decisão PRPPG” e “Registrar decisão DI” são muito parecidos com o modelo de navegação do caso de uso “Registrar decisão CT” (muda apenas o nome do método e o nome do formulário) e por isso foram omitidos. Já o modelo referente ao caso de uso “Arquivar Afastamento” é muito parecido com o do caso de uso “Cancelar Afastamento” e por isso também foi omitido.

Algumas classes de ação (controladores) aparecem em mais de um modelo de navegação com uma lista de atributos diferentes. Isso foi feito para enfatizar os atributos que importam naquele caso de uso, conforme proposto pelo método FrameWeb: “Os atributos da classe de ação representam parâmetros de entrada e saída relevantes àquela ação” (SOUZA, 2007). A mesma estratégia foi adotada com os métodos da classe, exibindo somente aqueles que são relevantes para o caso de uso em questão. A implementação de cada controlador, no entanto, contém todos os atributos e métodos que aparecem em todos os modelos.

Nos modelos onde há duas saídas possíveis (resultado) para o mesmo método — figuras 11, 12, 13, 14 — foi fundamental indicar qual resultado acionará cada saída. No caso específico destes modelos, o resultado “*success*” leva o usuário de volta à página que deu início ao caso de uso, e os resultados “*error*” ou “*null*” (quando não há resultado) levam o usuário para a página onde há o formulário para criação ou atualização do registro. Como não existe no FrameWeb uma notação que permita especificar múltiplos resultados em uma mesma saída, a seguinte notação foi utilizada: **{result=[error || null]}**. Assim, a saída que possui essa restrição será acionada na primeira chamada ao método, quando não há resultado (*null*) e nas chamadas subsequentes, quando há o envio dos dados do formulário, se houver algum erro (*error*).

Nos modelos onde a classe de ação (controlador) possui apenas uma entrada (evocação de método) e uma única saída (resultado) para o mesmo método — figuras 15, 16, 17, 18, 19, 20 — não foi indicada a restrição **result** pois o usuário sempre será levado de volta à página que deu início ao caso de uso, independentemente do resultado. Assim, quando o usuário é redirecionado para essa página após a execução de uma ação, a página exibe inicialmente uma notificação indicando o resultado (*feedback*).

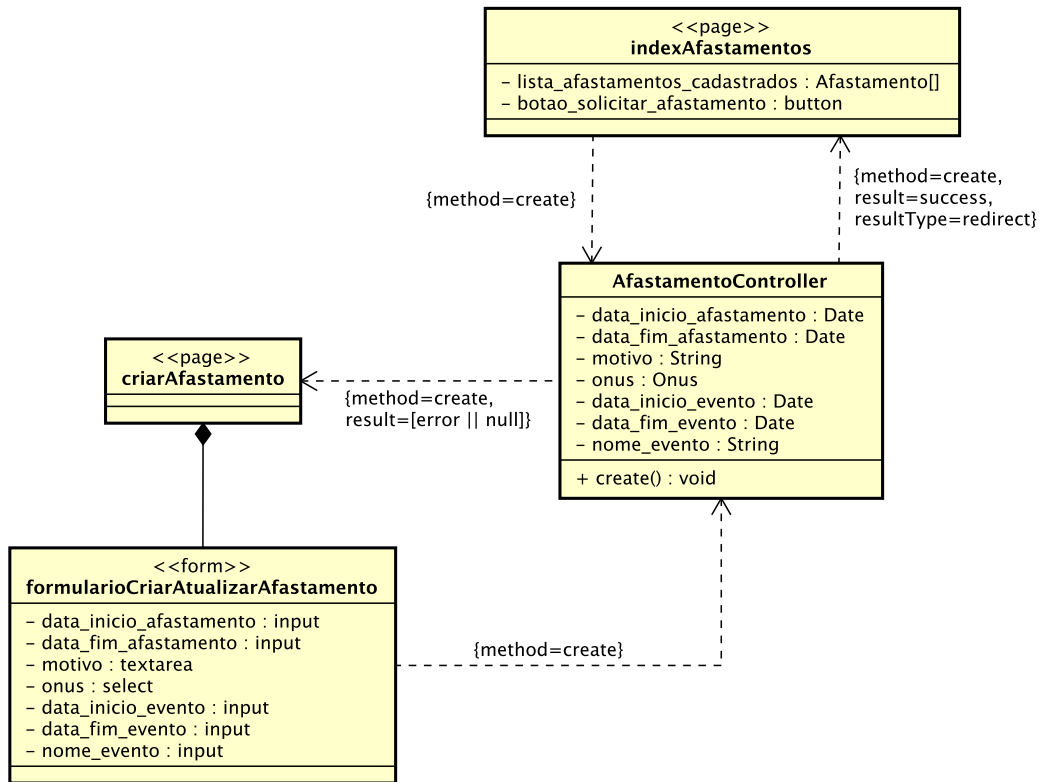


Figura 11 – Modelo de Navegação: Caso de uso “Solicitar Afastamento”.

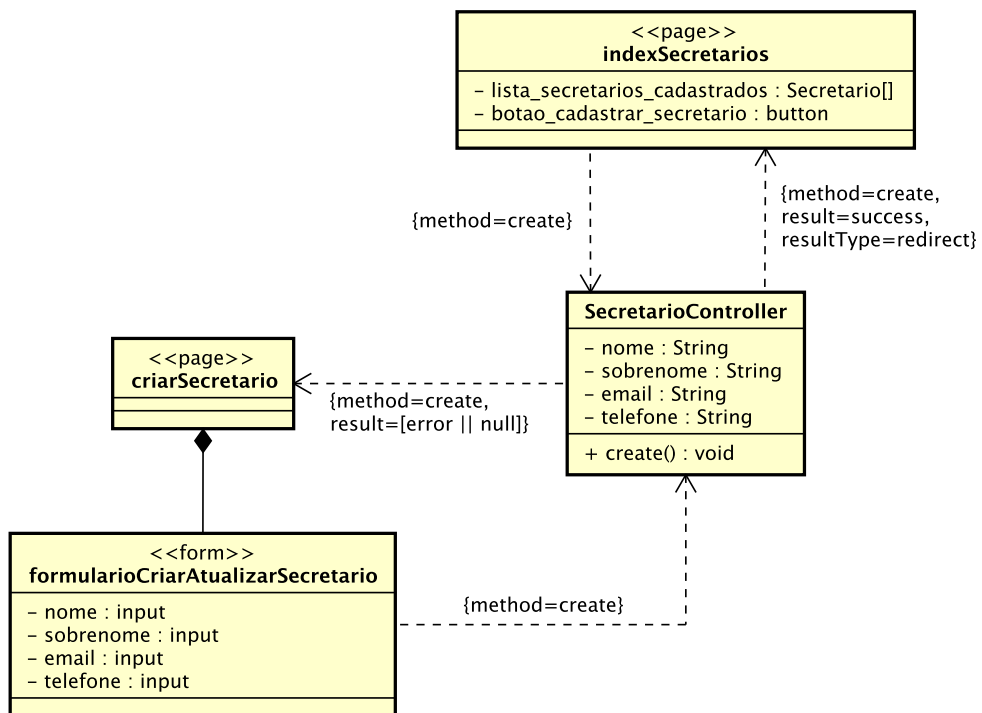


Figura 12 – Modelo de Navegação: Caso de uso “Cadastrar usuário (Secretário)”.

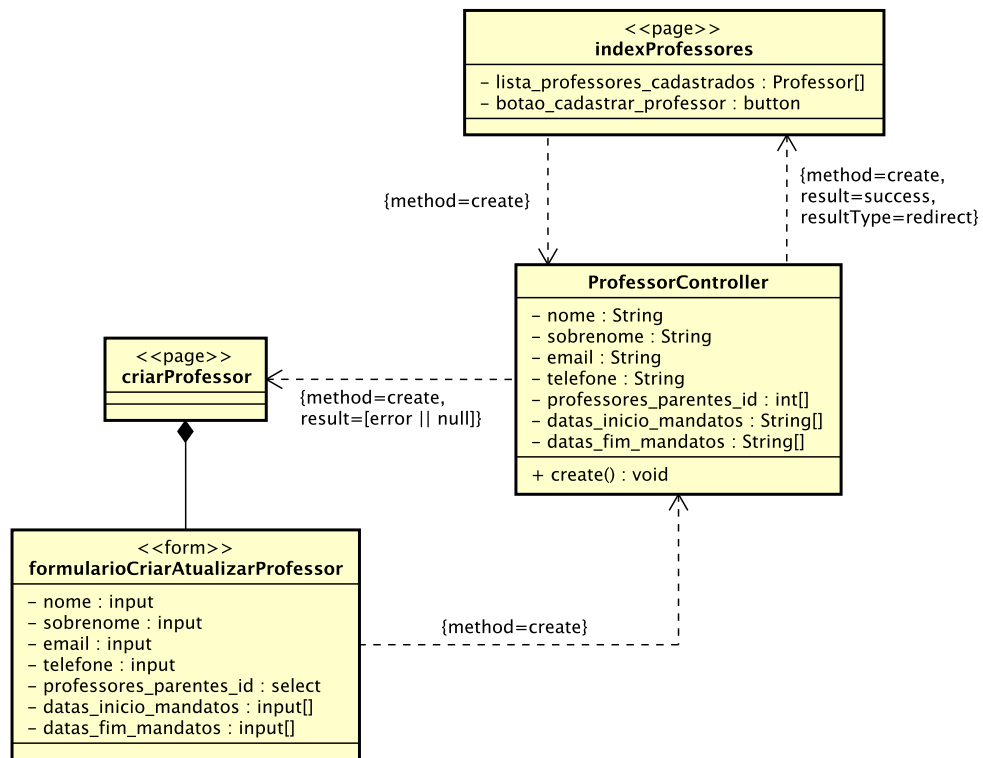


Figura 13 – Modelo de Navegação: Caso de uso “Cadastrar usuário (Professor)”.

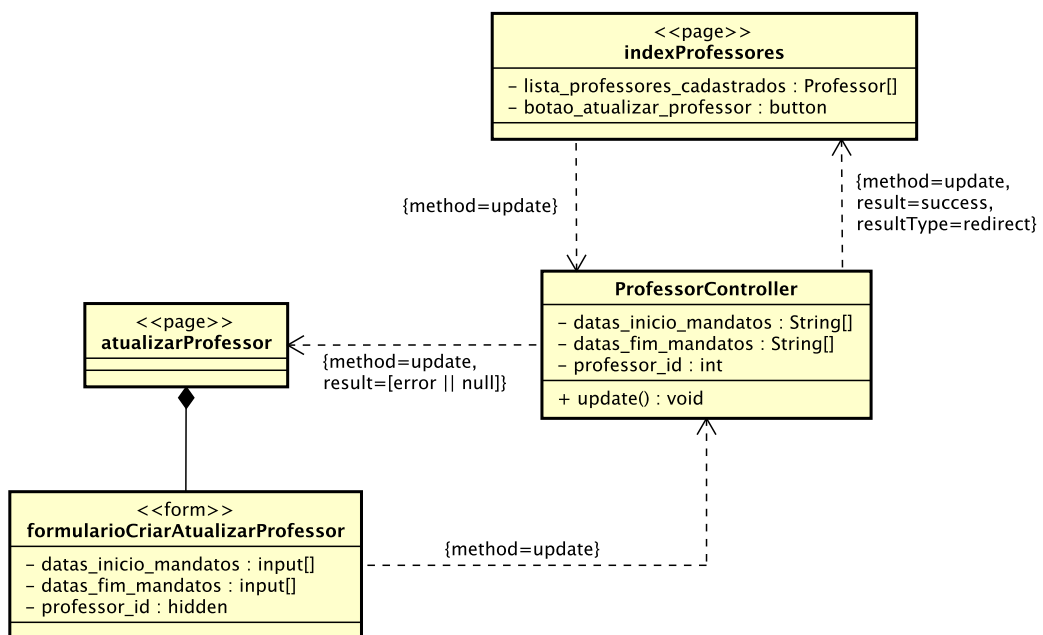


Figura 14 – Modelo de Navegação: Caso de uso “Cadastrar chefe do departamento”.

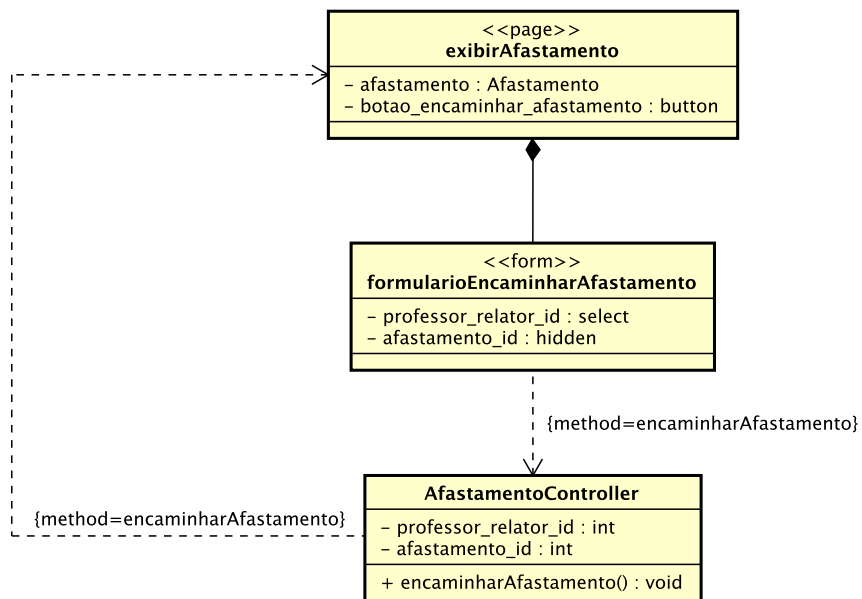


Figura 15 – Modelo de Navegação: Caso de uso “Encaminhar Afastamento”.

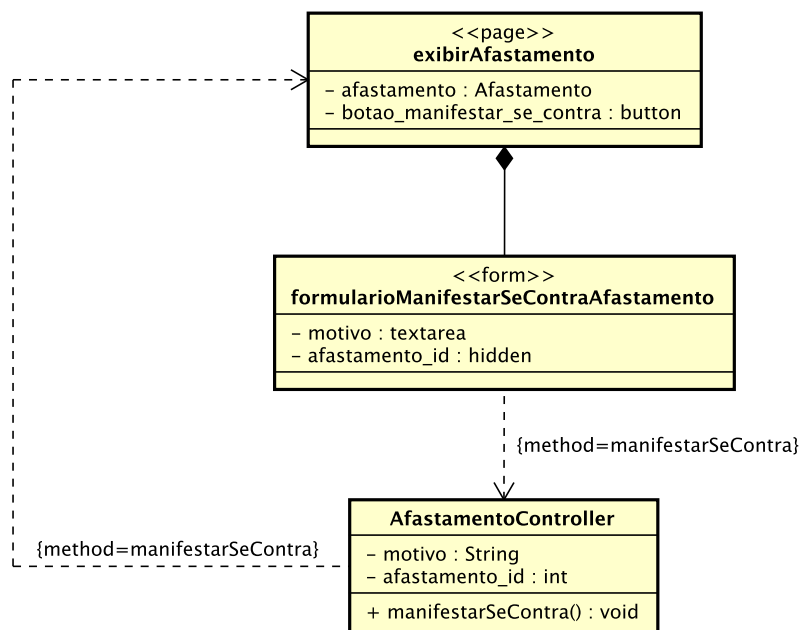


Figura 16 – Modelo de Navegação: Caso de uso “Manifestar-se Contra Afastamento”.

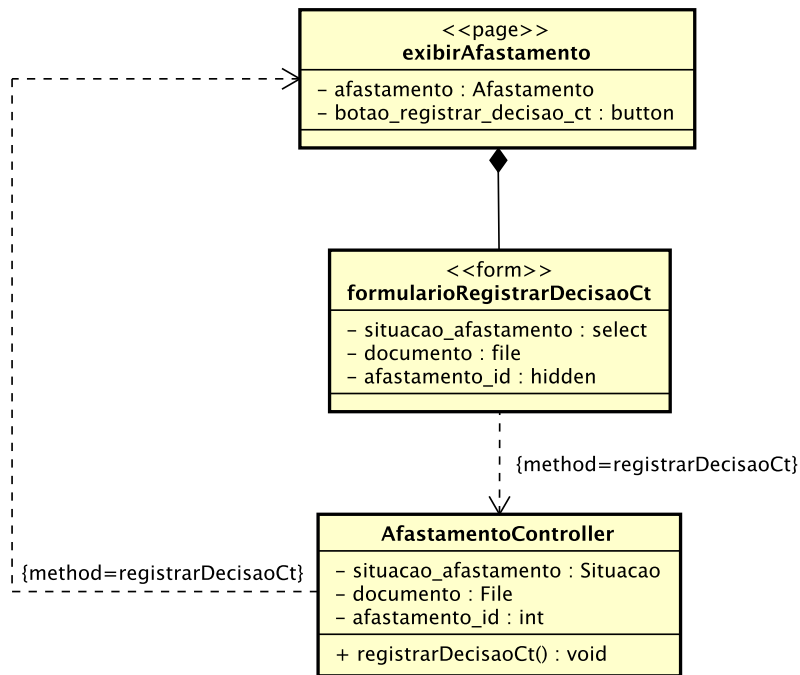


Figura 17 – Modelo de Navegação: Caso de uso “Registrar decisão CT”.

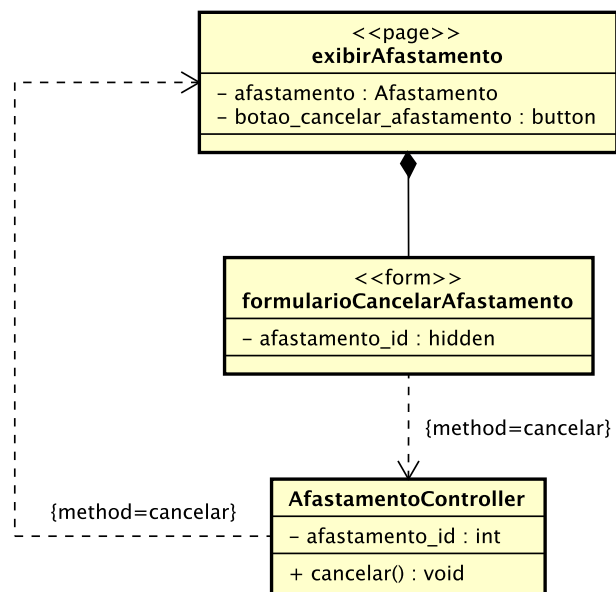


Figura 18 – Modelo de Navegação: Caso de uso “Cancelar Afastamento”.

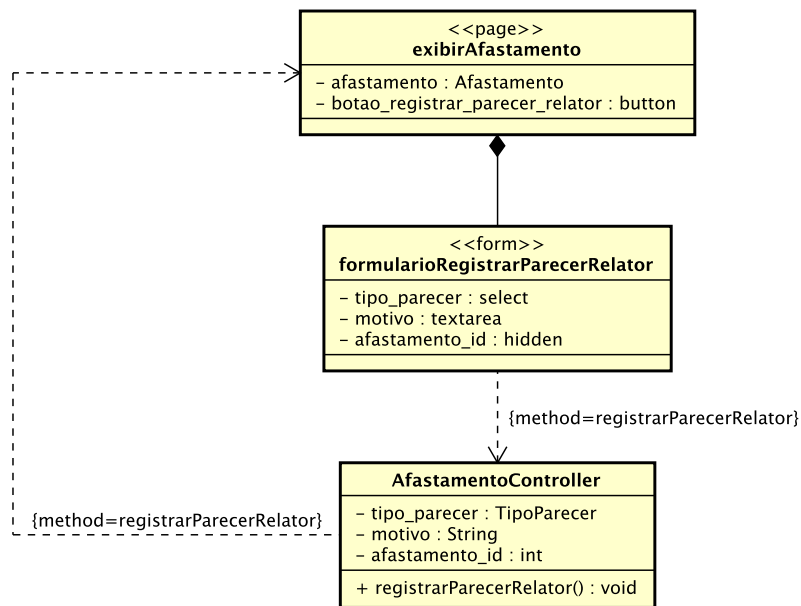


Figura 19 – Modelo de Navegação: Caso de uso “Registrar parecer relator”.

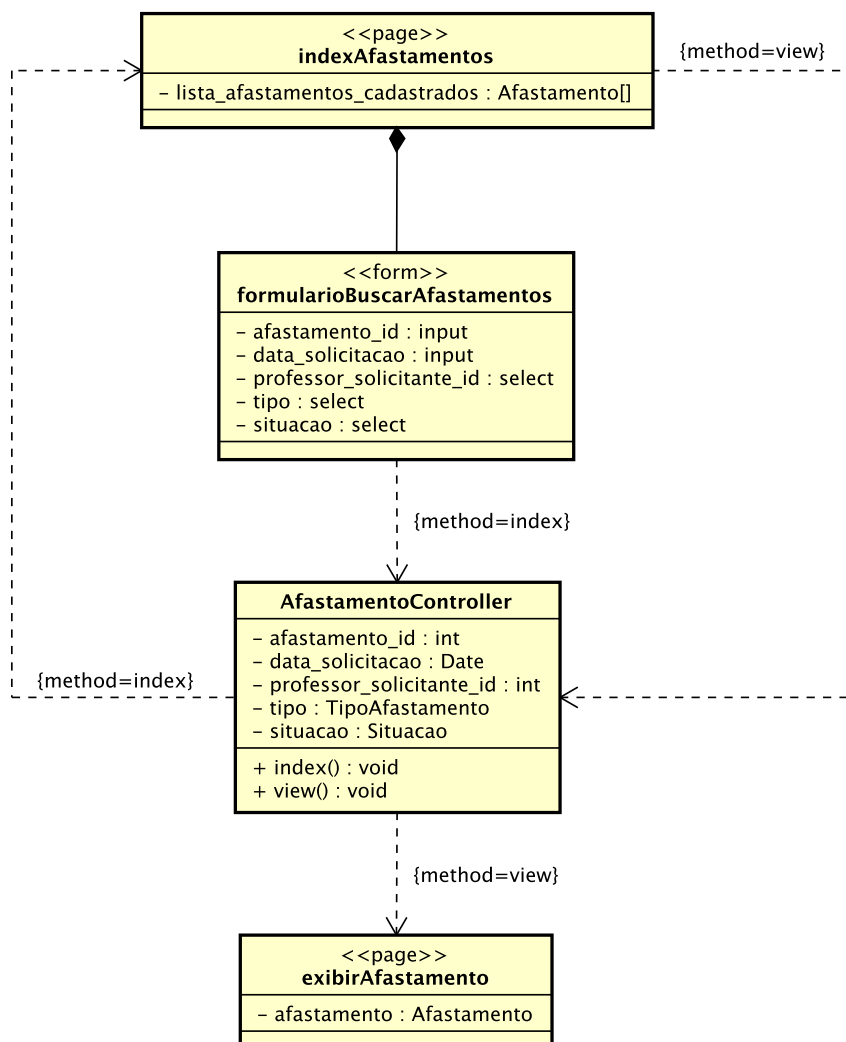


Figura 20 – Modelo de Navegação: Caso de uso “Consultar Afastamento”.

4.5 Modelo de Aplicação

O FrameWeb propõe a construção do modelo de aplicação com o objetivo de auxiliar a implementação das classes presentes no pacote “Aplicação”, além de permitir a visualização e configuração das dependências entre os pacotes “Controle”, “Aplicação” e “Persistência”.

A Figura 21 apresenta o modelo de aplicação para o projeto feito com o Symfony. É possível ver os três controladores principais do sistema se conectando às três classes de aplicação por meio de interfaces (optou-se por criar uma classe de aplicação para cada controlador). Olhando o diagrama fica claro quais classes de aplicação são utilizadas em cada controlador. Além disso, pode-se ver quais métodos devem ser implementados e quais interfaces DAO são necessárias em cada classe de aplicação. A administração dessas dependências é realizada pelo próprio Symfony.

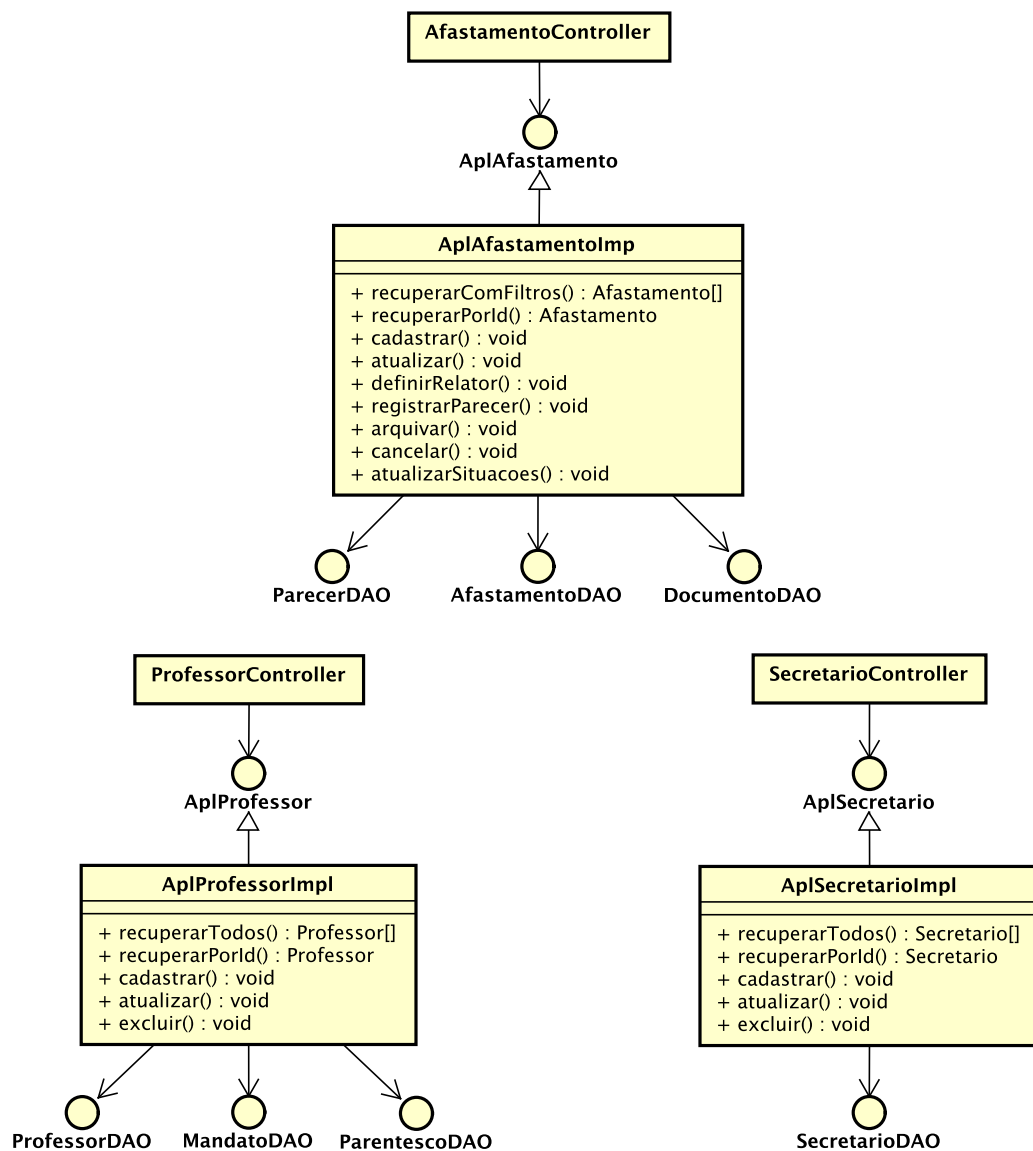


Figura 21 – Modelo de Aplicação (Symfony).

Para exemplificar a visualização das dependências no diagrama da Figura 21, considere o caso de uso “Cadastrar chefe do departamento”, onde o secretário localiza o cadastro do Professor (Chefe do Departamento) e nele adiciona um Mandato. Esse caso de uso é responsabilidade do controlador “ProfessorController”, como mostrado na Figura 14. Para realizá-lo, “ProfessorController” depende do método “atualizar()” da classe “AplProfessorImpl”, que por sua vez depende dos DAOs de Professor e Mandato.

A Figura 22 apresenta o modelo de aplicação para o projeto feito com o Yii. Nesse caso, os métodos que executam a lógica de negócio e a lógica de acesso a dados ficam nas classes do pacote domínio, como indica o padrão *Active Record*. Considerando isso, houve a necessidade de ajustar um pouco o modelo de aplicação sugerido pelo FrameWeb. Primeiramente, todos os métodos que estariam em classes de serviço agora estão nas classes de domínio (*models*). Além disso, as associações com as interfaces DAO já não são necessárias pois a lógica de acesso a dados está no próprio objeto de domínio. Dessa forma, os três controladores principais dependem das suas respectivas classes de domínio para executar os casos de uso.

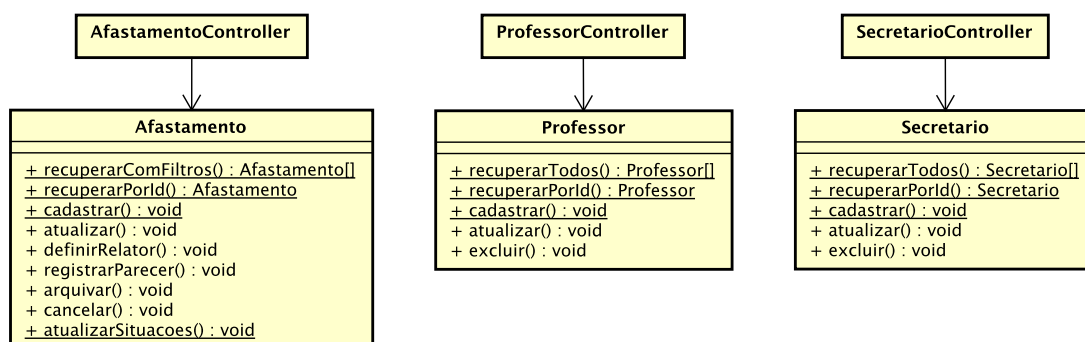


Figura 22 – Modelo de Aplicação (Yii).

O ajuste mencionado anteriormente foi baseado na estratégia adotada por Meirelles (2019) na tentativa de criar o Modelo de Aplicação proposto pelo FrameWeb com uma tecnologia ORM baseada no padrão *Active Record*.

4.6 Modelo de Persistência

Para guiar a implementação das classes DAO existentes o FrameWeb propõe a construção do modelo de persistência, o qual exibe uma interface DAO e uma implementação dessa interface para cada classe do domínio que precisa persistir suas instâncias. As interfaces definem os métodos que devem ser implementados na classe DAO.

A grande vantagem de usar interfaces é que o sistema não fica dependente de uma tecnologia de persistência específica, ou seja, basta trocar a implementação da interface para que o sistema já esteja pronto para trabalhar com a nova tecnologia.

O modelo de persistência, tal como é proposto pelo FrameWeb, foi melhor aproveitado na versão do SCAP feita com o Symfony, pois este utiliza um *framework* ORM baseado no padrão *Data Mapper* e, como foi explicado no Referencial Teórico (Capítulo 2), nesse padrão as operações da lógica de acesso a dados ficam separadas dos objetos de domínio, exatamente como sugerido pelo FrameWeb.

A Figura 23 apresenta o modelo de persistência do SCAP para o projeto feito com o Symfony. De acordo com Souza (2007), para não repetir operações comuns (por exemplo: salvar, atualizar, excluir...) em todas as interfaces DAO, é permitido apresentar uma interface base que declara esses métodos e uma implementação concreta dessa interface de forma que todas as outras interfaces herdam as definições da interface base e todas as demais classes herdam da implementação concreta base, sem que isso precise estar explícito no diagrama. Assim, na Figura 23 é possível ver a interface “DAOBase” e a implementação concreta “DAOBaseDoctrine” que são responsáveis por definir e implementar os métodos comuns. Nas classes de persistência específicas são implementados somente os métodos mais específicos de acesso a dados exigidos pela aplicação. Nesse projeto foi necessário adicionar métodos específicos apenas nas classes “AfastamentoDAODoctrine” e “ProfessorDAODoctrine”.

Vale mencionar que o nome das classes já indica qual tecnologia de persistência foi utilizada. No projeto feito com o Symfony, por exemplo, foi utilizado o Doctrine, que já vem pré-instalado com o *framework*. Assim, o nome das classes na Figura 23 termina com “Doctrine”, como em “AfastamentoDAODoctrine”. Esse sistema de nomenclatura é uma sugestão do FrameWeb.

Já no projeto feito com o Yii a lógica de acesso a dados fica no próprio objeto de domínio, de acordo com o padrão *Active Record*. Dessa forma, não serão necessárias a criação de interfaces e implementações de DAOs nesse projeto. Portanto não será apresentado o modelo de persistência para o projeto feito com o Yii.

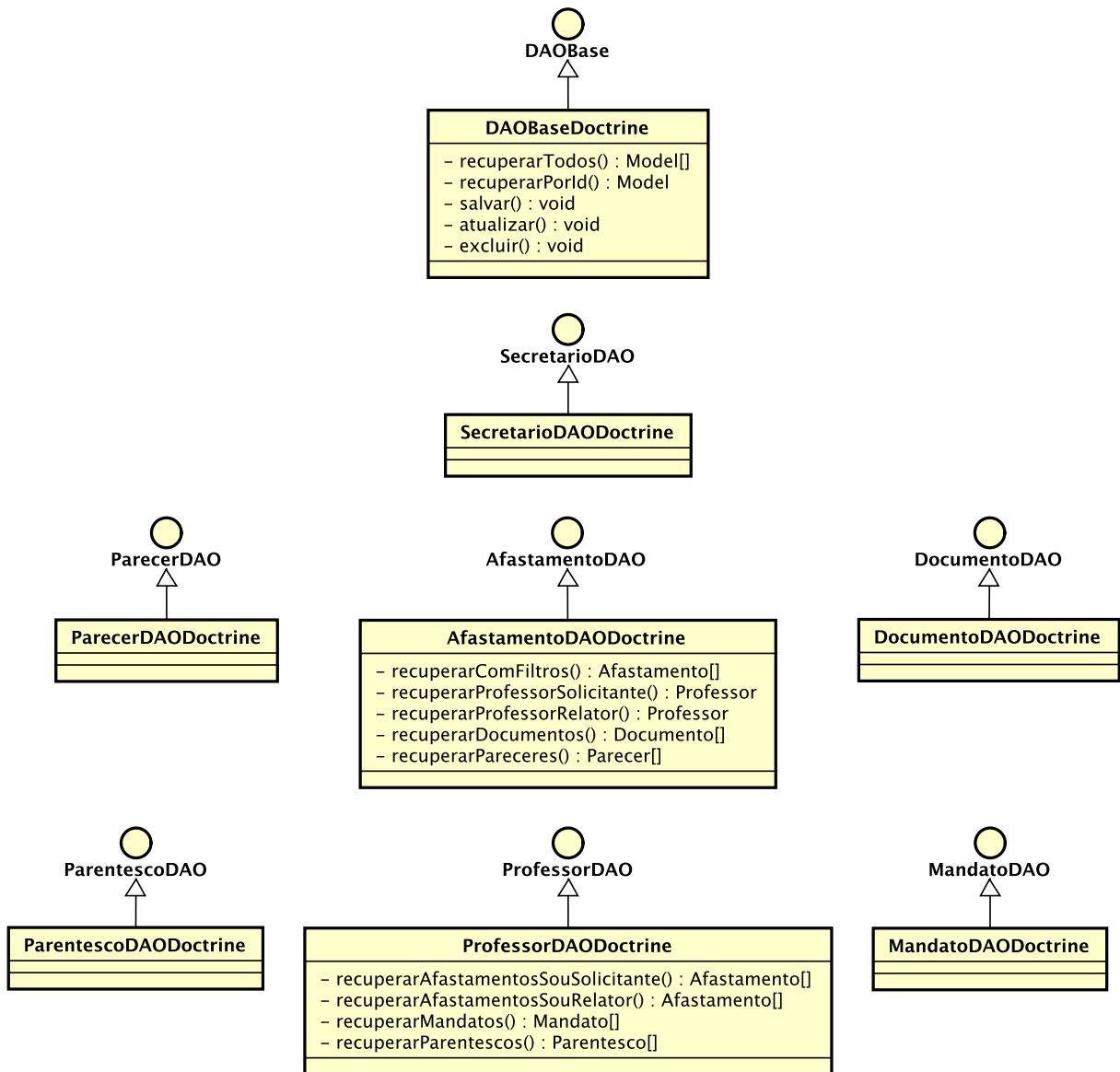


Figura 23 – Modelo de Persistência (Symfony).

4.7 Implementação

Esta seção apresentará a implementação do SCAP por meio de imagens das telas do sistema e trechos do código-fonte. Apesar de terem sido desenvolvidas duas implementações (uma com o Yii e outra com o Symfony), a parte visual de ambas ficou muito semelhante pelo fato de ter-se usado um *template* (Material Admin 2.0).

A Figura 24 apresenta a tela de login do SCAP. Para realizar a autenticação, o usuário informa o email cadastrado e a senha.

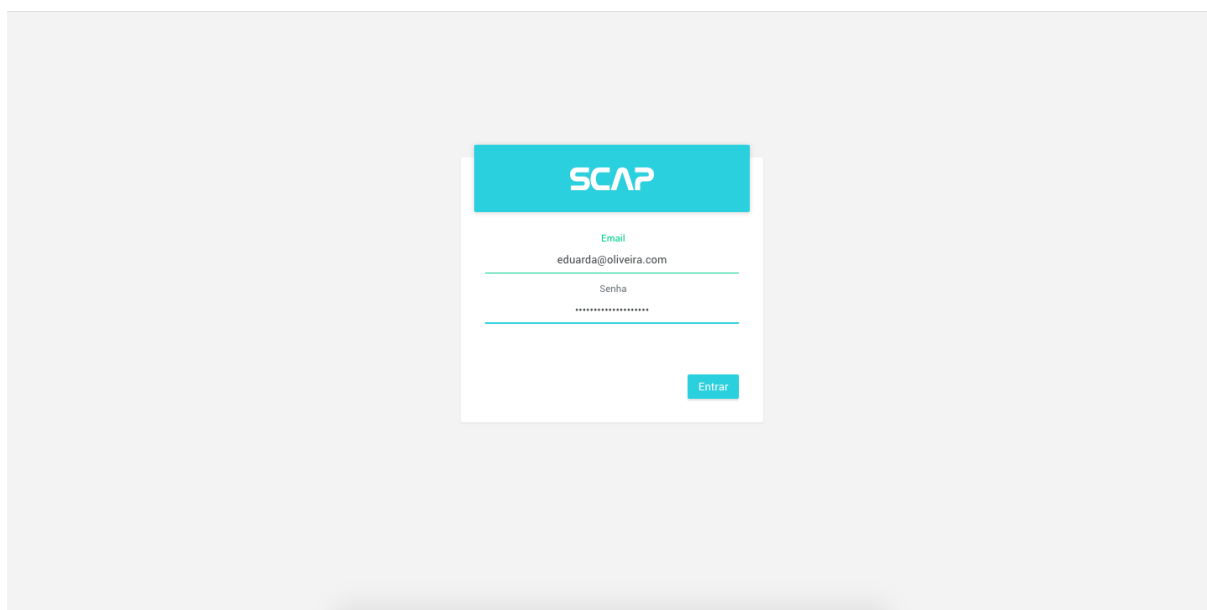


Figura 24 – Login.

Tanto o Yii quanto o Symfony fornecem estruturas de segurança que dão suporte à autenticação e autorização dos usuários. Para usar esta estrutura no Yii foi necessário configurar o componente de aplicativo *user* (como mostrado na Listagem 4.1) e criar uma classe que implementa a interface “yii\web\IdentityInterface” (neste projeto foi criada a classe *Usuario*, que se relaciona com a classe *Pessoa* de forma biunívoca, ou seja, um para um).

Listagem 4.1 – Yii – Configuração do componente *user*.

```
1 <?php
2 $config = [
3     'components' => [
4         'user' => [
5             'identityClass' => 'app\models\Usuario',
6         ],
7         // outros componentes ...
8     ],
9     // outras configurações ...
10 ];
```

Dessa forma, o componente *user* fica disponível para uso em qualquer parte do código, e pode ser usado para autorizar ou negar a execução de determinada ação. A Listagem 4.2 apresenta um exemplo de uso do componente *user*. Trata-se do trecho inicial (a verificação de autorização do usuário) do método do controlador de Afastamentos que gerencia o caso de uso “Registrar parecer relator”. É possível ver nas linhas 20 e 21 que o componente *user* está sendo usado para verificar se o usuário logado é um professor e se o mesmo é o relator do afastamento em questão.

Listagem 4.2 – Yii – Exemplo de autorização de usuário.

```

1 <?php
2 /**
3  * @param $id
4  * @return Response
5  * @throws NotFoundHttpException
6  * @throws Exception
7  */
8 public function actionRegistrarParecerRelator($id)
9 {
10     $model = $this->findModel($id);
11
12     // CONDIÇÕES
13     // - Afastamento é internacional
14     // - Situação é Aguardando parecer relator
15     // - O usuário logado deve ser um professor
16     // - O usuário logado deve ser o relator
17     if (
18         $model->tipo !== Afastamento::TIPO_INTERNACIONAL ||
19         $model->situacao !== Afastamento::SITUACAO_AGUARDANDO_PARECER_RELATOR ||
20         empty(Yii::$app->user->identity->professor) ||
21         $model->professorRelator->id !== Yii::$app->user->identity->professor->id
22     ) {
23         throw new Exception('Ação não permitida');
24     }
25
26     // executa caso de uso ...
27 }

```

No Symfony o processo foi semelhante: foi criada uma classe *Usuario* que implementa a interface “Symfony\Component\Security\Core\User\UserInterface” e a mesma foi indicada nas configurações do *framework* como responsável pela gestão da autenticação e autorização dos usuários.

A Listagem 4.3 mostra um exemplo de como é feita a verificação de autorização de um usuário para execução de uma determinada ação no Symfony. Neste exemplo, por meio da anotação *@IsGranted* (linha 4) foi informado ao *framework* que a ação “Registrar parecer relator” só pode ser executada caso o usuário logado possua o papel “ROLE_PROFESSOR”, o qual é concedido a todos os usuários professores. Uma exceção é lançada automaticamente pelo *framework* caso o usuário não possua esse papel. Na linha 8 é possível ver como se obtém o usuário logado, o qual é usado na linha 18 para verificar se o mesmo é o relator do afastamento. Se qualquer condição não for satisfeita (linhas 16, 17 e 18), uma exceção é lançada.

Listagem 4.3 – Symfony – Exemplo de autorização de usuário.

```
1 <?php
2 /**
3  * @Route("/{id}/registrar-parecer-relator", name="...", methods={"POST"})
4  * @IsGranted("ROLE_PROFESSOR")
5  */
6 public function registrarParecerRelator(Request $request, Afastamento $afastamento)
7 {
8     $professorLogadoId = $this->getUser()->getProfessor()->getId();
9
10    // CONDIÇÕES
11    // - Afastamento é internacional
12    // - Situação é aguardando parecer relator
13    // - O usuário logado deve ser um professor (já garantido)
14    // - O usuário logado deve ser o relator
15    if (
16        $afastamento->getTipo() !== Afastamento::TIPO_INTERNACIONAL ||
17        $afastamento->getSituacao() !== Afastamento::SIT_AG_PARECER_RELATOR ||
18        $afastamento->getProfessorRelatorId() !== $professorLogadoId
19    ) {
20        throw new \Exception('Ação não permitida');
21    }
22
23    // executa caso de uso ...
24 }
```

Quando a autenticação é realizada com sucesso, o usuário é direcionado para a página que exibe a lista de todas as solicitações de afastamento cadastradas no sistema (Figura 25). A partir desta página o usuário logado consegue solicitar seu próprio afastamento (se for um professor), consultar afastamentos usando filtros (Figura 26) ou visualizar um afastamento.

O *layout* base da aplicação exibe um *header* em todas as páginas do sistema. Esse *header* mostra o nome do sistema (SCAP), o email do usuário logado e um menu com três opções: “Afastamentos”, “Professores” e “Secretários”. As opções “Professores” e “Secretários” levam o usuário para as listagens de professores e secretários cadastrados no sistema e só poderão ser acessadas se o usuário logado for um secretário. Por meio delas é possível realizar o gerenciamento dessas entidades (criação, visualização, atualização e exclusão), de acordo com os casos de uso “Cadastrar usuário” e “Cadastrar chefe do departamento”.

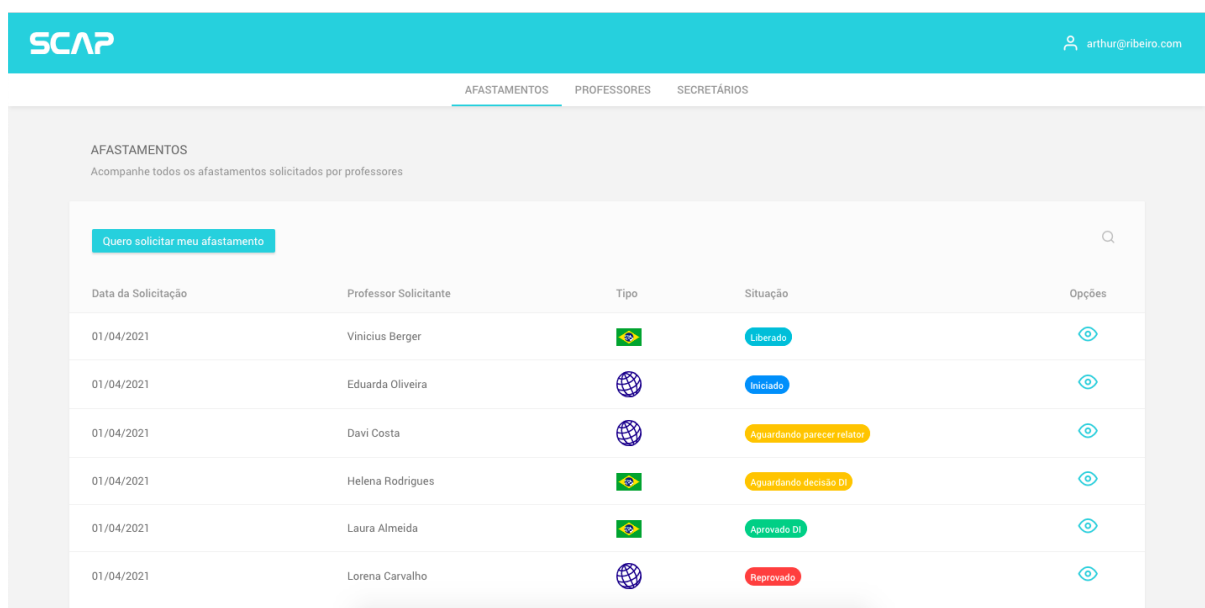


Figura 25 – Listagem de Afastamentos.

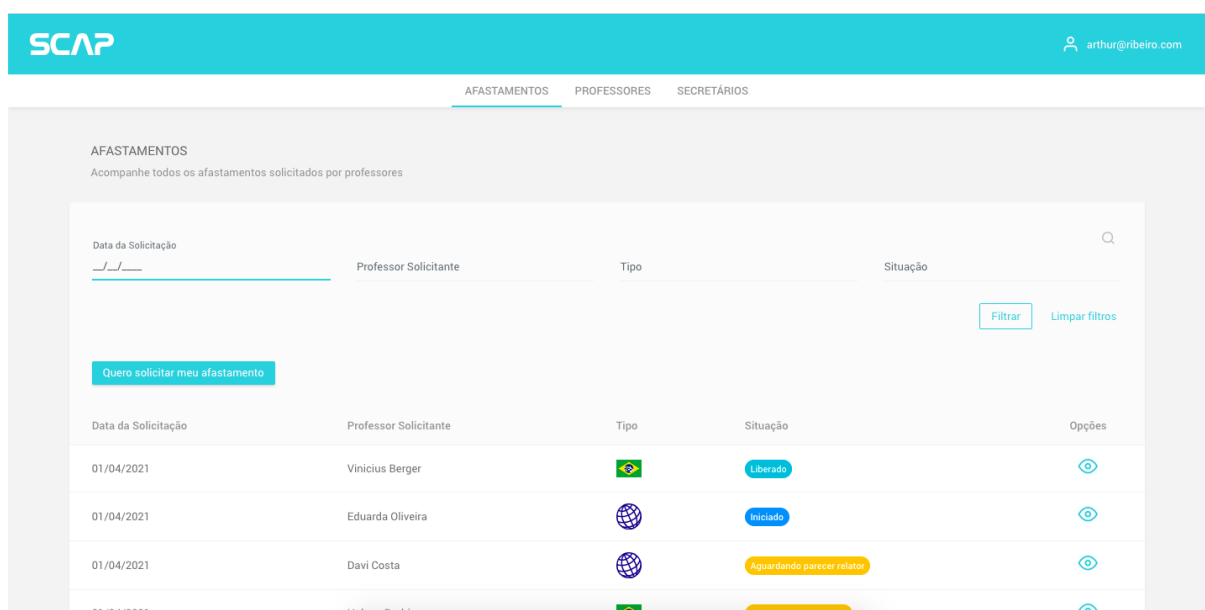


Figura 26 – Consultar Afastamento.

A Figura 27 mostra uma parte do formulário usado pelos professores para solicitar um afastamento. O formulário é dividido em três partes: “Informações básicas” — onde o professor deve informar o período, o tipo, o ônus e o motivo do afastamento — “Sobre o evento” — onde deve-se informar o nome do evento e as datas de início e término do mesmo — e “Documentos” — onde o professor pode anexar documentos que auxiliem a aprovação do afastamento e do auxílio financeiro (caso o ônus seja parcial ou total). Pode-se ver também uma regra de validação sendo aplicada no campo “Nome do Evento”, indicando que o mesmo não pode ser deixado em branco, conforme o mapeamento de persistência do Modelo de Entidades (Seção 4.3). Os demais mapeamentos de persistência (relacionados a atributos das classes de domínio) também possuem validações implementadas. Além disso, foram implementadas regras de validação para garantir as restrições de integridade do sistema (Seção 3.4).

SCAP vincius4@gmail.com

AFASTAMENTOS PROFESSORES SECRETÁRIOS

SOLICITAR MEU AFASTAMENTO

Informações Básicas
Qual é o período do afastamento e qual o motivo?

Data de Início do Afastamento: 01/06/2021
Data de Fim do Afastamento: 02/06/2021
Tipo: Nacional
Ônus: Parcial

Motivo: Participação em evento científico

Sobre o Evento
Que evento é este e quando vai ocorrer?

Nome do Evento: |
Data de Início do Evento: 01/06/2021
Data de Fim do Evento: 02/06/2021

Nome do Evento não pode ficar em branco.

Figura 27 – Solicitar Afastamento.

As figuras 28 e 29 mostram parcialmente a página de visualização de um afastamento, a qual exibe as seguintes seções: “Opções”, “Informações Básicas”, “Sobre o Evento”, “Professor Solicitante”, “Professor Relator”, “Documentos” e “Pareceres”.

Na Figura 28 é possível ver, logo no início da página, a seção “Opções” com oito botões, que representam todas as ações que podem ser executadas sobre um afastamento. Neste exemplo, os sete primeiros botões estão inativos pois são ações que não podem mais ser executadas sobre o afastamento em questão, devido à situação deste ser “Reprovado”. Assim, conforme o diagrama de estados da Figura 10, a única possibilidade de ação é o caso de uso “Arquivar afastamento”.

A Figura 29 mostra que o afastamento que está sendo visualizado possui quatro documentos anexados: — “Banner do evento”, “ATA da Reunião do DI”, “ATA da Reunião

do CT” e “ATA da Reunião da PRPPG” — e permite a visualização de cada um. Além disso, pode-se ver que existem dois pareceres cadastrados até o momento: um parecer favorável emitido pelo relator e um parecer desfavorável, emitido quando um professor se manifestou contra o afastamento.

SCAP

arthur@ribeiro.com

AFASTAMENTOS PROFESSORES SECRETÁRIOS

acompanhar afastamento

Opções
O que você deseja fazer com este afastamento?

Cancelar Encaminhar Registrar parecer relator Manifestar-se contra Registrar decisão DI Registrar decisão CT Registrar decisão PRPPG Arquivar

Informações Básicas
Qual é o período do afastamento e qual o motivo?

Reprovado

Data de Início do Afastamento	Data de Fim do Afastamento	Tipo	Ônus
01/07/2021	02/07/2021	Internacional	Inexistente

Motivo
Participação em evento científico

Figura 28 – Visualizar Afastamento (1).

SCAP

viniciush4@gmail.com

AFASTAMENTOS PROFESSORES SECRETÁRIOS

Vinicius Berger viniciush4@gmail.com (27) 99999-8888

Documentos
Quais documentos justificam a solicitação de afastamento?

01/04/2021 08:03:44 Banner do evento [Abrir em nova aba](#)

02/04/2021 16:35:06 ATA da Reunião do DI [Abrir em nova aba](#)

03/04/2021 09:15:06 ATA da Reunião do CT [Abrir em nova aba](#)

04/04/2021 10:06:12 ATA da Reunião da PRPPG [Abrir em nova aba](#)

Pareceres
Quais são os pareceres até o momento?

01/04/2021 Vinicius Berger (Relator) Informações fornecidas foram comprovadas

01/04/2021 Arthur Ribeiro Esse evento é fake

Figura 29 – Visualizar Afastamento (2).

As figuras 30 e 31 mostram a caixa de diálogo que é aberta quando o chefe do departamento clica na opção “Encaminhar” (uma das ações que pode ser executada sobre um afastamento, de acordo com o caso de uso “Encaminhar afastamento”). Essa caixa de diálogo permite que o chefe do departamento selecione um professor da lista para ser o relator do afastamento em questão. A lista de professores não inclui o professor que solicitou o afastamento. Além disso, como pode ser visto na Figura 31, os professores que são parentes do solicitante aparecem desabilitados e com a informação “(é parente do solicitante)” na frente do nome.

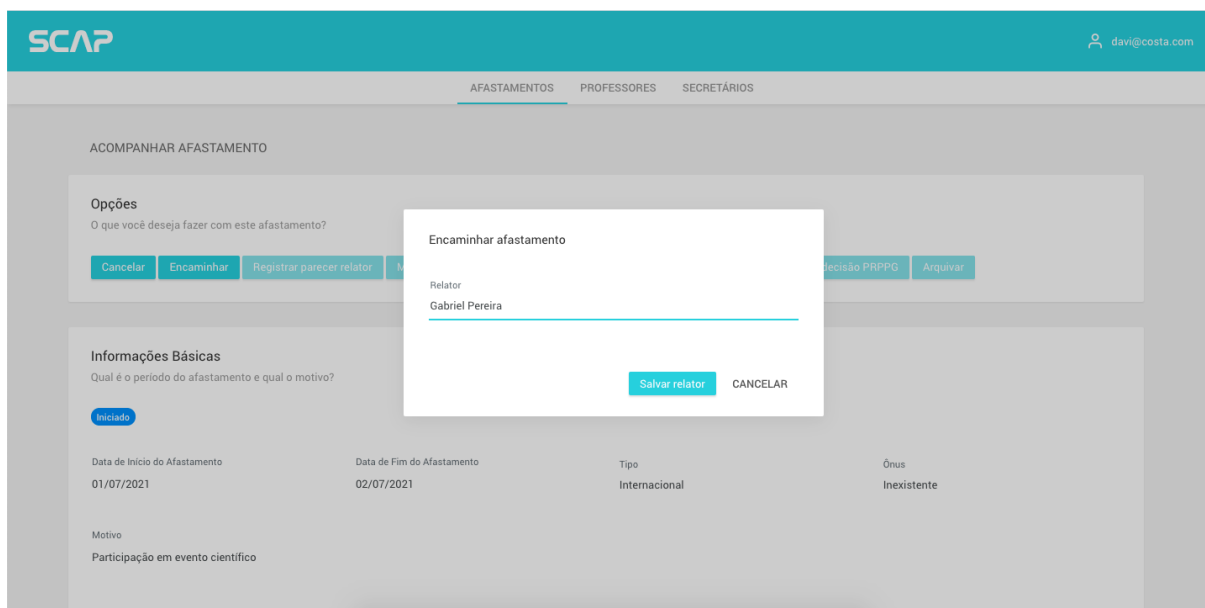


Figura 30 – Encaminhar Afastamento (1).

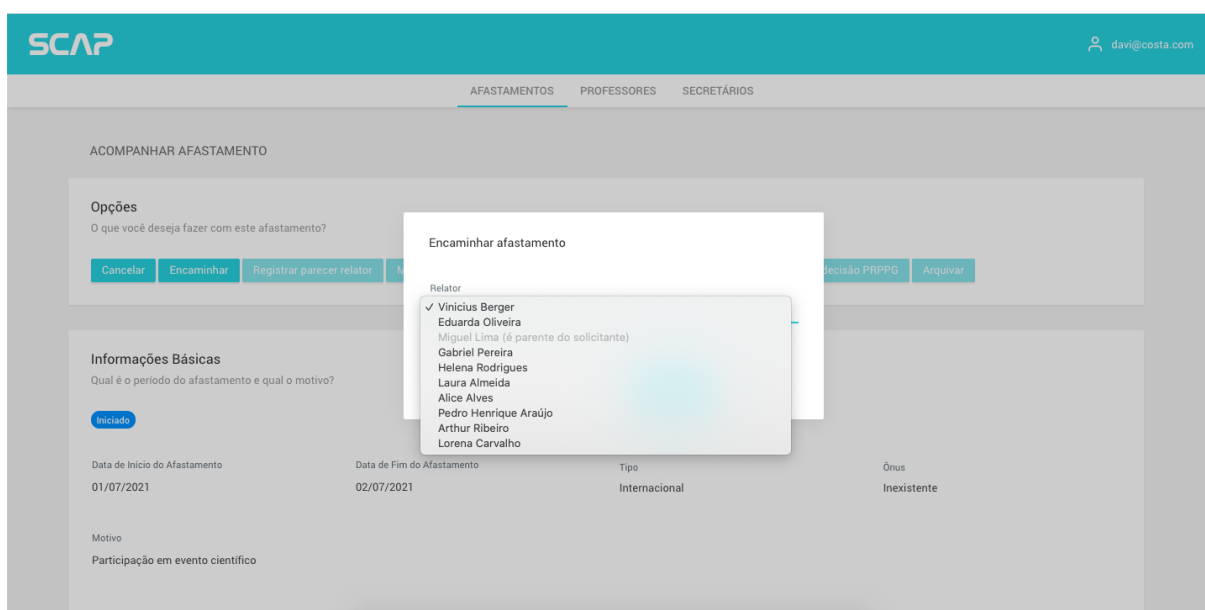


Figura 31 – Encaminhar Afastamento (2).

A Figura 32 mostra o email que é enviado ao professor após o mesmo ter sido escolhido como relator do afastamento.



Figura 32 – Email enviado ao relator.

A Figura 33 apresenta a caixa de diálogo que é aberta quando um professor pretende se manifestar contra um afastamento. Nesse caso é exibido como possibilidade de entrada de dados apenas um campo de texto onde o professor deve informar o motivo de ser contra o afastamento.

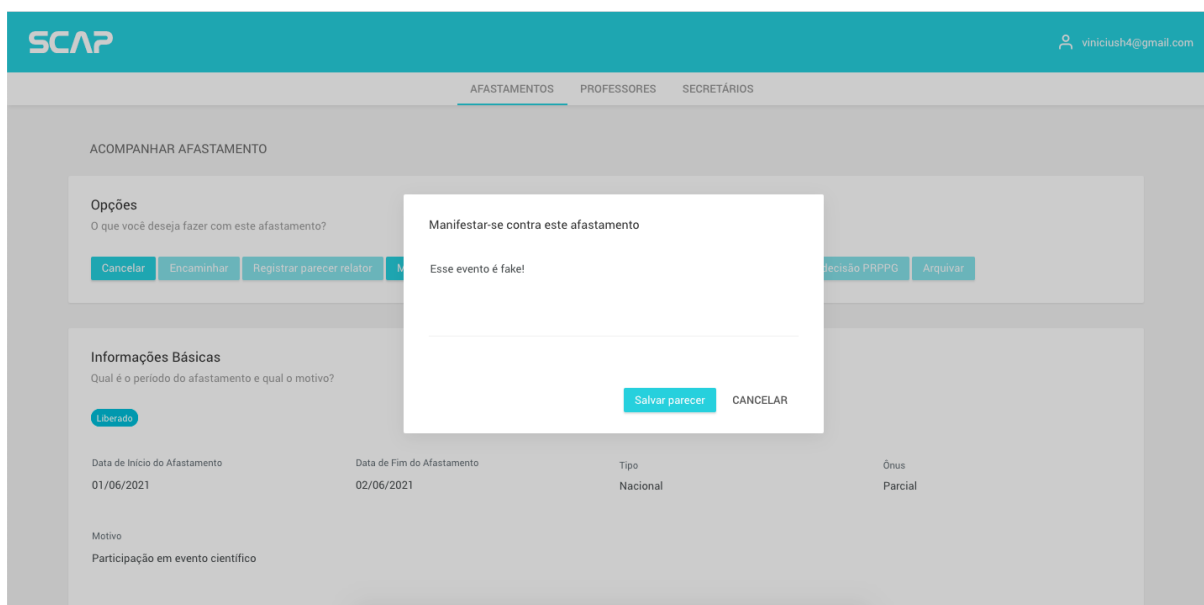


Figura 33 – Manifestar-se contra Afastamento.

A Figura 34 mostra a caixa de diálogo que é aberta quando um secretário vai registrar a decisão do DI. Nela existem três campos de entrada de dados: um para selecionar a decisão (favorável ou desfavorável), um para adicionar o título do documento anexado e outro para selecionar o arquivo a ser anexado (geralmente a ATA da reunião). Esse mesmo *layout* de caixa de diálogo é usado para registrar as decisões do CT e da PRPPG.

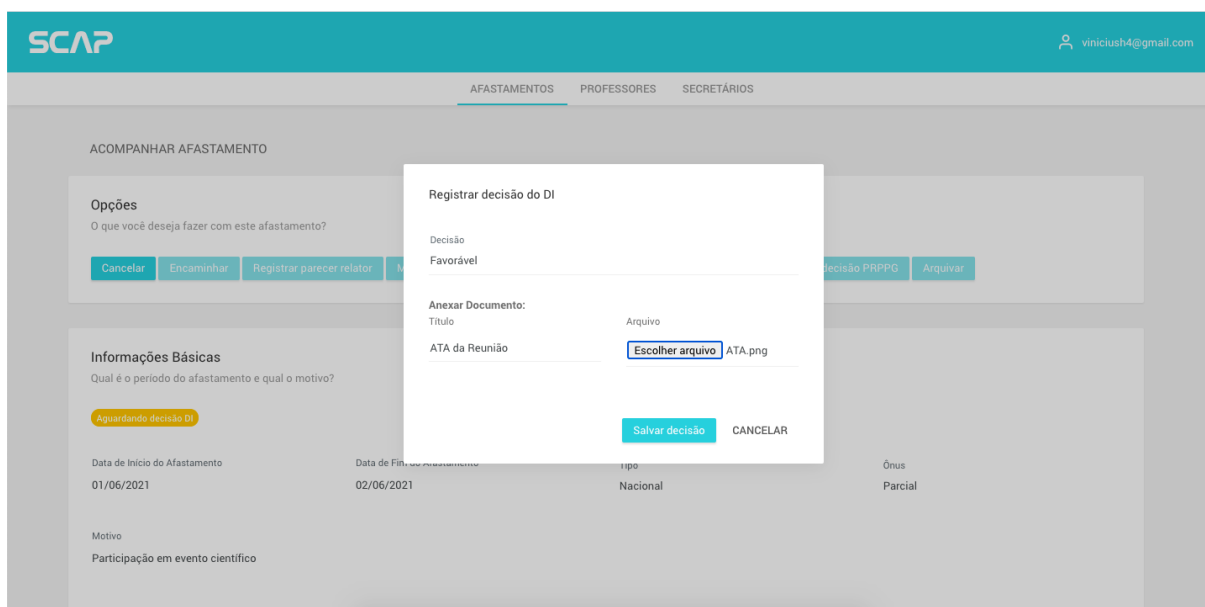


Figura 34 – Registrar decisão do DI.

As figuras 35 e 36 apresentam uma forma utilizada pelo sistema para dar *feedback* ao usuário sobre as ações executadas. Nesses exemplos a mensagem de sucesso informa que o afastamento foi criado com sucesso e a mensagem de erro indica que não foi possível registrar a decisão do CT.

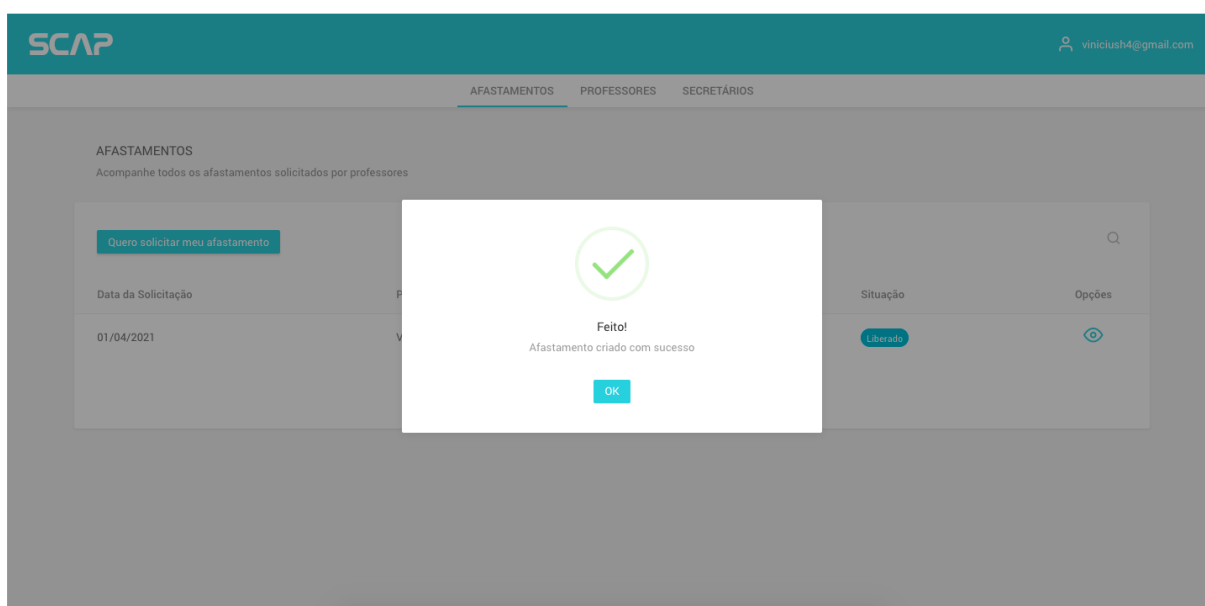


Figura 35 – Mensagem de sucesso.

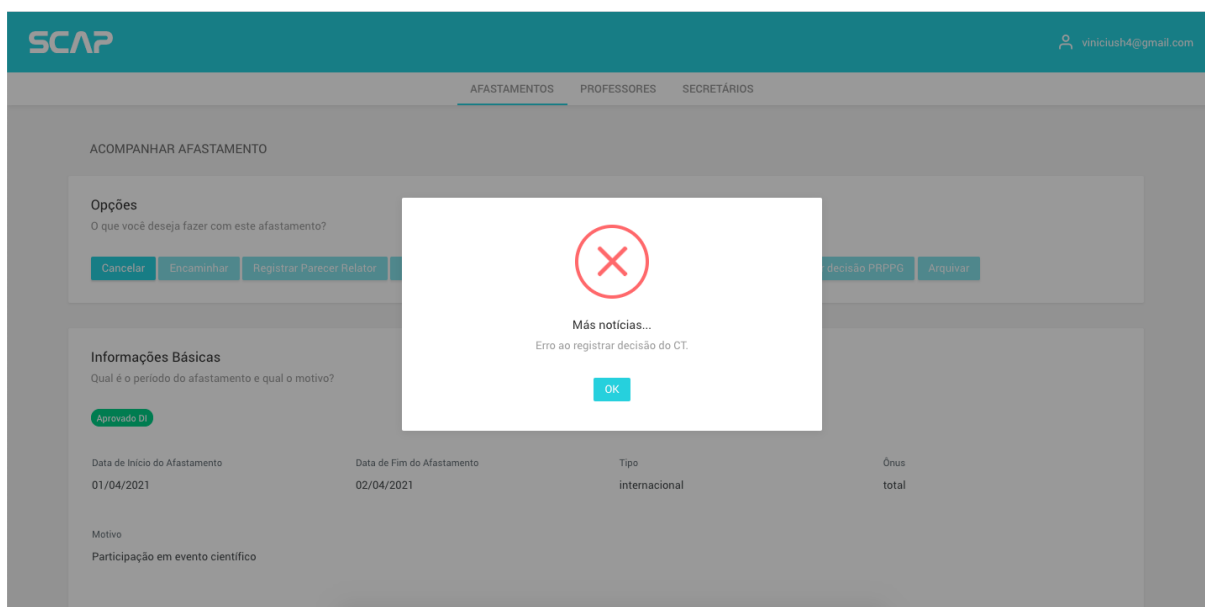


Figura 36 – Mensagem de erro.

A Figura 37 apresenta a listagem de professores cadastrados no sistema, a qual só pode ser acessada por secretários. Pode-se ver o nome completo, email e telefone de todos os professores. Além disso, o atual chefe do departamento aparece com uma “etiqueta” ao lado do nome, indicando esse cargo. A listagem de de secretários segue o mesmo layout desta página.

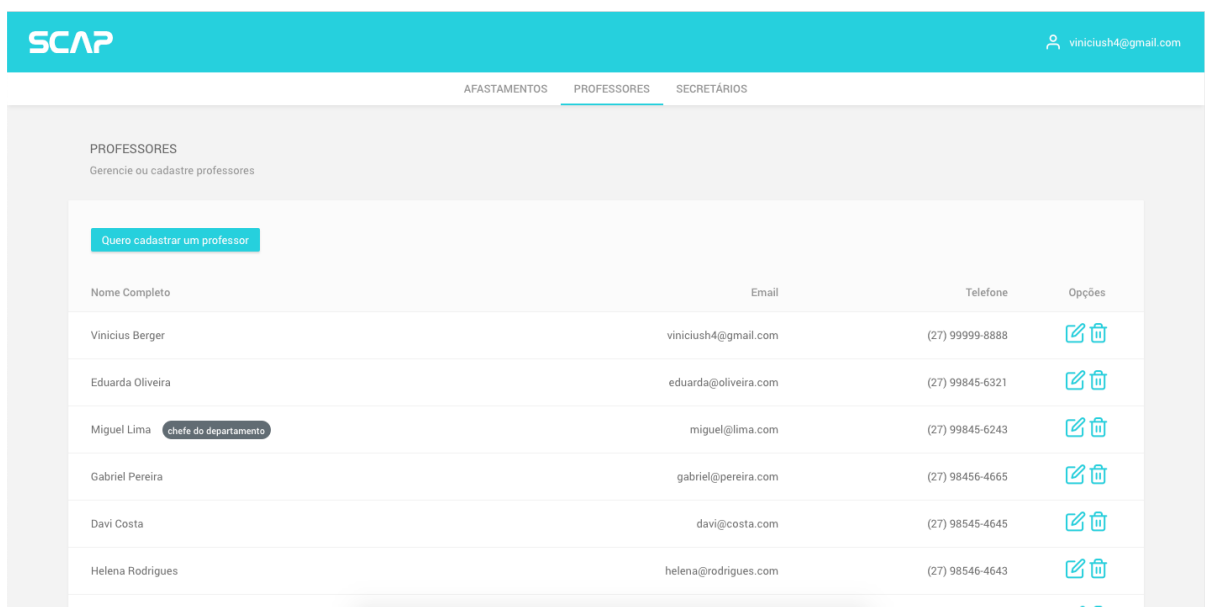
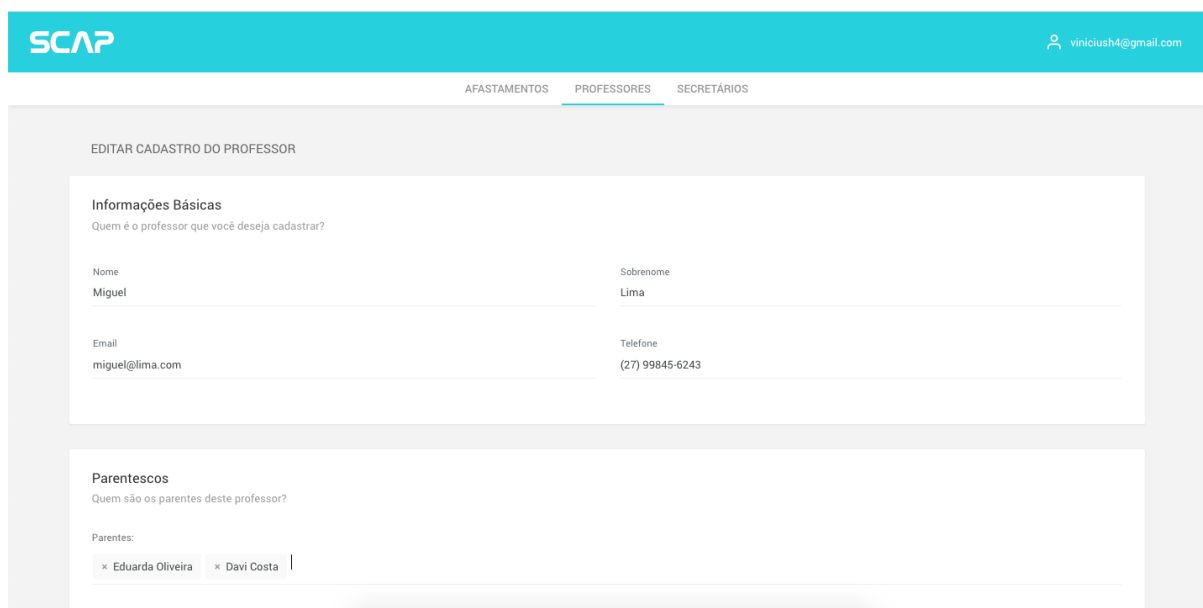


Figura 37 – Listagem de Professores.

As figuras 38 e 39 apresentam o formulário para cadastro e atualização de dados de professores, o qual está dividido em três partes: “Informações Básicas”, “Parentescos” e “Mandatos”. Em “Informações Básicas” o secretário deve informar o nome, sobrenome, email e telefone do professor. Na seção “Parentescos” deve-se selecionar todos os professores

já cadastrados que são parentes deste. Em “Mandatos” deve-se informar as datas de início e término de cada mandato do professor.



SCAP

viniciush4@gmail.com

AFASTAMENTOS PROFESSORES SECRETÁRIOS

EDITAR CADASTRO DO PROFESSOR

Informações Básicas
Quem é o professor que você deseja cadastrar?

Nome: Miguel

Sobrenome: Lima

Email: miguel@lima.com

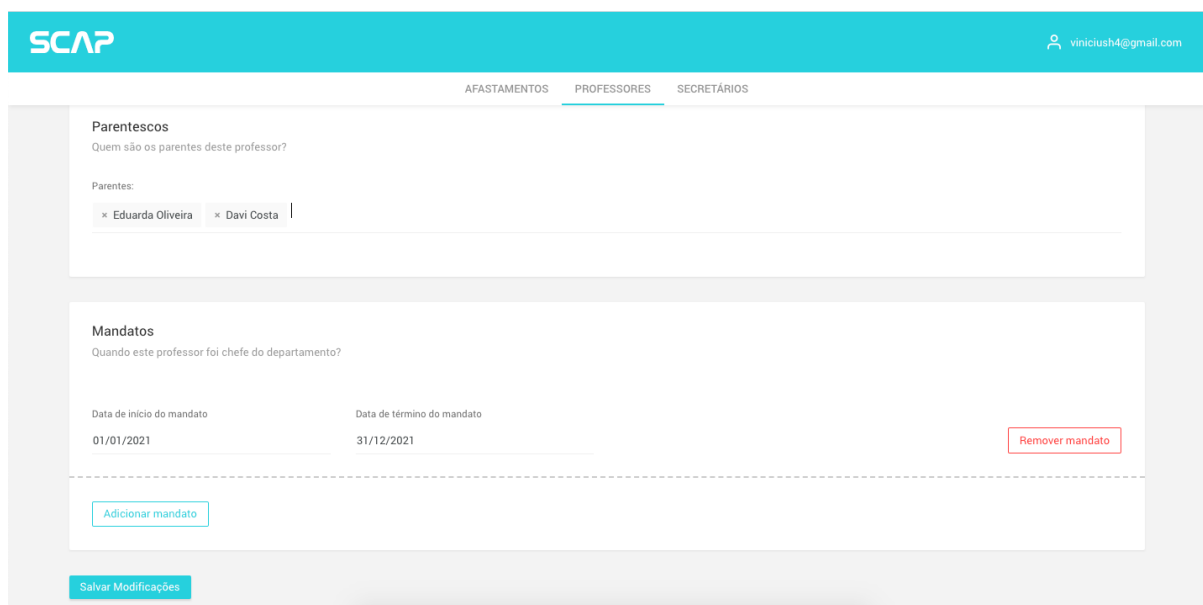
Telefone: (27) 99845-6243

Parentescos
Quem são os parentes deste professor?

Parentes:

- × Eduarda Oliveira
- × Davi Costa

Figura 38 – Cadastrar Professor (1).



SCAP

viniciush4@gmail.com

AFASTAMENTOS PROFESSORES SECRETÁRIOS

Parentescos
Quem são os parentes deste professor?

Parentes:

- × Eduarda Oliveira
- × Davi Costa

Mandatos
Quando este professor foi chefe do departamento?

Data de início do mandato: 01/01/2021

Data de término do mandato: 31/12/2021

Remover mandato

Adicionar mandato

Salvar Modificações

Figura 39 – Cadastrar Professor (2).

Os códigos-fonte implementados neste trabalho estão disponíveis em <https://bitbucket.org/vitorsouza-students/pg-2019-vinicius-berger/src/master/pg-codigo/>.

5 Considerações Finais

No Capítulo 2 foi evidenciada a importância de uma abordagem sistemática e disciplinada no desenvolvimento de software de qualidade e portanto, para atingir o objetivo geral deste trabalho (Seção 1.1), procurou-se seguir a indicação de [Pressman \(2011\)](#): adoção de um **processo** (com foco nas etapas de Projeto e Implementação, pois o levantamento de requisitos e a análise já tinham sido feitos por [Duarte \(2014\)](#) e [Prado \(2015\)](#)), um conjunto de **métodos** (neste caso foi utilizado apenas o método FrameWeb, proposto por [Souza \(2007\)](#)), e um leque de **ferramentas** (neste caso, os *frameworks* Yii e Symfony e as demais tecnologias citadas na Seção 4.1).

O conhecimento adquirido ao longo do curso de Ciência da Computação influenciou bastante a realização deste trabalho. Algumas disciplinas merecem destaque nesse sentido:

- **Programação III**, onde aprendeu-se a projetar e programar sistemas usando os conceitos de Programação Orientada a Objetos (OO);
- **Engenharia de Software**, onde estudou-se os principais conceitos e práticas que visam melhorar a qualidade dos produtos de software e aumentar a produtividade no processo de desenvolvimento;
- **Engenharia de Requisitos de Software**, onde aprendeu-se a levantar, analisar, modelar conceitualmente, documentar, avaliar e gerenciar requisitos de sistemas de software;
- **Projeto de Sistemas de Software**, onde estudou-se abordagens, padrões e métodos aplicáveis à fase de projeto de sistemas, a fim de se desenvolver a capacidade de observar aspectos relevantes a serem considerados nessa etapa do processo e a elaborar modelos de projeto de sistemas de software;
- **Banco de Dados**, onde foram apresentados os conceitos fundamentais de Sistemas de Gerência de Bancos de Dados, abordando problemas de projeto, uso e implementação;
- **Desenvolvimento Web e Web Semântica**, onde descobriu-se, em suma, o que são aplicações Web e como desenvolvê-las.

Conforme demonstrado ao longo desta monografia, os objetivos deste projeto de graduação foram plenamente alcançados. Neste capítulo, apresentamos uma discussão do uso de *frameworks* e de FrameWeb no desenvolvimento do projeto e apontamos algumas oportunidades de trabalhos futuros.

5.1 Discussão

Sem dúvida, o uso dos *frameworks* simplificou bastante a implementação do sistema, por meio de componentes prontos para uso que foram exaustivamente testados por programadores ao redor do mundo. A utilização foi facilitada pela vasta documentação e inúmeros exemplos de uso disponíveis na Web.

Os modelos propostos pelo FrameWeb ajudaram a visualizar o que deveria ser implementado, além de proporcionarem mais organização e aproveitamento de tempo na etapa de implementação. Uma vantagem observada no uso do FrameWeb, mais especificamente no Modelo de Entidades, foi a capacidade de se ter agrupadas, em um mesmo local, informações adicionais sobre o valor dos atributos de cada classe, como por exemplo o tipo, tamanho, precisão das datas e horas e se o mesmo pode ser nulo ou não (além de vários outros metadados suportados pelo FrameWeb). Essas informações, sem o uso do FrameWeb, estariam espalhadas pelo código do sistema (em forma de anotações, por exemplo) e no *schema* do banco de dados, fazendo com que o programador perdesse tempo quando houvesse necessidade de acessá-las (o que acontece bastante quando se está implementando o sistema).

O uso do FrameWeb, como é proposto originalmente, garante também que as camadas do sistema mantenham um nível baixo de acoplamento, facilitando a substituição e aperfeiçoamento dos componentes.

Após realizar as duas implementações do SCAP, uma com o *framework* Yii e outra com o *framework* Symfony, chegou-se à conclusão de que as propostas do FrameWeb são totalmente compatíveis com o Symfony, pois não houve nenhuma complicação na etapa de Projeto nem na Implementação do SCAP com este *framework*. O mesmo não pode ser dito sobre o *framework* Yii, que gerou uma limitação no uso do FrameWeb, a qual é discutida a seguir.

5.1.1 Limitações

A Tabela 3 mostrou que das onze implementações anteriores do SCAP, apenas duas utilizaram uma tecnologia ORM baseada no padrão *Active Record*. Como foi visto anteriormente, a adoção do padrão *Active Record* pela implementação ORM do Yii gerou uma pequena incompatibilidade com a arquitetura em camadas e com os modelos de Aplicação e Persistência propostos pelo FrameWeb. Essa incompatibilidade foi experimentada também por Meirelles (2019) em um dos trabalhos anteriores que utilizaram esse padrão. Apesar de não ter associado a dificuldade de criar os diagramas de Aplicação e Persistência ao uso do padrão *Active Record* ele diz “Não fora criado nenhuma classe de Serviço e classe DAO, porém os *Models* fazem o mesmo papel dos objetos DAO” (MEIRELLES, 2019). Isso mostra que a lógica de domínio e de persistência está nos objetos de domínio,

exatamente como proposto pelo *Active Record*. Assim, Meirelles (2019) precisou ajustar os modelos de Aplicação e Persistência, da mesma forma como foi feito neste trabalho.

Apesar de os modelos de Aplicação e Persistência propostos pelo FrameWeb não se encaixarem perfeitamente com o padrão de projeto adotado pela tecnologia ORM do Yii, como descrito na seção 4.2, os outros dois modelos foram muito bem aproveitados na implementação das classes de Domínio (modelo de Entidades) e nas classes dos pacotes Visão e Controle (modelos de Navegação). Já a versão do sistema feita com o Symfony aproveitou todos os diagramas do FrameWeb.

5.1.2 Contribuições

Algumas propostas de melhorias foram feitas para o SCAP, as quais estão detalhadas no Capítulo 3 e listadas a seguir:

- Ajustes nos nomes de três casos de uso;
- Inclusão do caso de uso “Registrar decisão DI”;
- Remoção do atributo “parentesco” da classe “Parentesco”.

Além disso, uma contribuição é sugerida para o método FrameWeb: como não existe até então uma notação, no modelo de Navegação, que permita especificar múltiplos resultados em uma mesma saída, sugere-se que seja utilizada a seguinte: **{result=[error || null]}**, onde os resultados “habilitados” a utilizar a saída estão dentro de colchetes, separados por duas barras verticais (neste exemplo, **error** e **null**). A vantagem de se utilizar esta notação é a simplificação do diagrama. Com ela, não é necessário informar uma associação para cada resultado possível do mesmo método que leve o fluxo da aplicação para o mesmo componente. Dessa forma, há uma redução de associações presentes no diagrama.

5.2 Trabalhos Futuros

Considerando as limitações experimentadas neste trabalho (Seção 5.1.1), sugere-se para trabalhos futuros que o SCAP seja implementado com *frameworks* que utilizem padrões de projeto diferentes dos que foram utilizados até o momento, a fim de identificar pontos de conflito que possam gerar melhorias para o FrameWeb. Sugere-se também o uso de linguagens de programação que ainda não foram utilizadas. Segue uma lista com sugestões de *frameworks* que podem ser explorados em trabalhos futuros:

- Django (Python);

- Ruby on Rails (Ruby);
- Asp .NET (C#);
- Phoenix (Elixir).

Outra sugestão interessante seria realizar uma compilação geral sobre o uso do FrameWeb na prática, considerando os resultados dos diferentes trabalhos realizados anteriormente. Além disso, poder-se-ia analisar, em um outro trabalho, se o FrameWeb é adequado para projetos ágeis e como ele poderia ser utilizado nesse contexto.

Pode-se também realizar um trabalho futuro com o objetivo de atualizar o “Documento de Requisitos” e o “Documento de Especificação de Requisitos” do SCAP, estendendo algumas funcionalidades e incluindo as alterações que foram feitas neste trabalho juntamente com outras alterações feitas em trabalhos anteriores (como por exemplo a alteração do nome da classe “Afastamento” para “SolicitacaoAfastamento”, realizada por [Guterres \(2019\)](#)), visando evoluir o sistema.

Referências

- AVELAR, R. d. A. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Ninja*. Vitória, ES, Brasil, 2018. Citado 2 vezes nas páginas 14 e 36.
- BAUER, C.; KING, G. *Hibernate in Action*. 1. ed. Greenwich: Manning, 2005. ISBN 1932394-15-X. Citado na página 24.
- CUNHA, G. E. d.; HERBERT, J. S. Proposta de Padrões de Software para a Reutilização Sistemática em Sistemas de Software Orientados a Objetos. *The Third Latin American Conference on Pattern Languages of Programming*, Porto de Galinhas, ago 2003. Disponível em: <http://www.cin.ufpe.br/~sugarloafplop/articles/wp/GabrielaUnisinos_novo.pdf>. Citado 2 vezes nas páginas 13 e 21.
- DUARTE, B. B. *Aplicação do Método FrameWeb no Desenvolvimento de um Sistema de Informação na Plataforma Java EE 7*. Vitória, ES, Brasil, 2014. Citado 11 vezes nas páginas 13, 14, 15, 16, 28, 29, 31, 32, 34, 36 e 64.
- FERNANDES, F.; LUFT, C. P.; GUIMARÃES, F. M. *Dicionário Brasileiro Globo*. 44. ed. São Paulo: Globo, 1996. v. 1. ISBN 85-250-0298-4. Citado 2 vezes nas páginas 18 e 31.
- FERREIRA, M. T. *Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Wicket e Tapestry*. Vitória, ES, Brasil, 2018. Citado 2 vezes nas páginas 14 e 36.
- FOWLER, M. *Patterns of Enterprise Application Architecture*. 1. ed. [S.l.]: Addison-Wesley, 2002. ISBN 0-321-12742-0. Citado 6 vezes nas páginas 7, 22, 23, 25, 26 e 27.
- GABRIELI, L. V. et al. Uso de um Framework no Desenvolvimento de Sistemas de Informação Web. *XXVII Encontro Nacional de Engenharia de Produção*, Foz do Iguaçu, out 2007. Citado 2 vezes nas páginas 13 e 21.
- GAMMA, E. et al. *Padrões de Projeto: soluções reutilizáveis de software orientado a objetos*. 1. ed. Porto Alegre: Bookman, 2000. v. 1. Tradução Luiz A. Meirelles Salgado. ISBN 978-85-7780-046-9. Citado 2 vezes nas páginas 13 e 21.
- GUTERRES, C. S. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o framework Play*. Vitória, ES, Brasil, 2019. Citado 3 vezes nas páginas 14, 36 e 67.
- IEEE. *Standard Glossary of Software Engineering Terminology*. [S.l.]: IEEE, 1990. IEEE Standard 610.12-1990. Citado na página 17.
- MATOS, R. P. de. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando os frameworks Spring MVC e Vaadin*. Vitória, ES, Brasil, 2017. Citado 2 vezes nas páginas 14 e 36.
- MEIRELLES, L. C. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Codeigniter e NodeJS*. Vitória, ES, Brasil, 2019. Citado 5 vezes nas páginas 14, 36, 50, 65 e 66.

- MURUGESAN, S. et al. Web Engineering: A New Discipline for Development of Web-Based Systems. In: MURUGESAN, S.; DESHPANDE, Y. (Ed.). *Web Engineering: Managing Diversity and Complexity of Web Application Development*. Berlin, Heidelberg: Springer, 2001, (Lecture Notes in Computer Science, v. 2016). p. 3–13. ISBN 978-3-540-42130-6. Disponível em: <https://doi.org/10.1007/3-540-45144-7_2>. Citado 2 vezes nas páginas 17 e 18.
- PINHEIRO, F. G. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando um framework PHP e um framework JavaScript*. Vitória, ES, Brasil, 2017. Citado 2 vezes nas páginas 14 e 36.
- 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 11 vezes nas páginas 13, 14, 15, 16, 28, 29, 32, 33, 34, 36 e 64.
- PRESSMAN, R. S. *Engenharia de Software: uma abordagem profissional*. 7. ed. Porto Alegre: Bookman, 2011. v. 1. Tradução Ariovaldo Griesi; revisão técnica Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade. ISBN 978-85-8055-044-3. Citado 6 vezes nas páginas 13, 17, 18, 19, 22 e 64.
- PRPPG. *Afastamento para Eventos e Outras Atividades no Exterior*. c2013. Disponível em: <<https://prppg.ufes.br/afastamento-para-eventos-e-outras-atividades-no-exterior>>. Citado na página 14.
- PRPPG. *Afastamentos Para Mestrado, Doutorado, Pós-Doutorado e Visitas Técnico-científicas/Intercâmbios Acadêmicos*. c2013. Disponível em: <<https://prppg.ufes.br/afastamentos-para-mestrado-doutorado-pos-doutorado-e-visitas-tecnico-cientificasintercambios>>. Citado na página 14.
- PRPPG. *Autorizações de Afastamentos para eventos no País*. c2013. Disponível em: <<https://prppg.ufes.br/autorizacoes-de-afastamentos-para-eventos-no-pais>>. Citado na página 14.
- SILVA, L. W. Internet foi criada em 1969 com o nome de "Arpanet" nos EUA. *Folha de São Paulo*, São Paulo, ago 2001. Disponível em: <<https://www1.folha.uol.com.br/folha/cotidiano/ult95u34809.shtml>>. Citado na página 13.
- 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, Vitória, 2007. Citado 18 vezes nas páginas 7, 9, 13, 14, 15, 17, 18, 19, 20, 21, 22, 23, 24, 39, 40, 43, 51 e 64.
- 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 na página 21.
- SYMFONY. *Do Que Se Trata?* s.d. Disponível em: <https://symfony.com/doc/5.0/the-fast-track/pt_BR/0-intro.html>. Citado na página 38.
- SYMFONY. *Sobre o Projeto Symfony*. s.d. Disponível em: <<https://symfony.com/about>>. Citado na página 38.

SYMFONY. *Symfony explicado a um desenvolvedor*. s.d. Disponível em: <<https://symfony.com/explained-to-a-developer>>. Citado na página 39.

YII. *O que é o Yii*. c2008. Disponível em: <<https://www.yiiframework.com/doc/guide/2.0/pt-br/intro-yii>>. Citado na página 38.

Apêndices



Documento de Projeto de Sistema

SCAP - Sistema de Controle de Afastamento de Professores

Registro de Alterações:

Versão	Responsável	Data	Alterações
1.0	Vinícius Berger	22/03/2019	Versão Inicial
1.1	Vinícius Berger	24/01/2021	Adicionados os modelos FrameWeb
1.2	Vinícius Berger	08/03/2021	Correções
1.3	Vinícius Berger	22/03/2021	Correções

Vitória, ES

2021

1 Introdução

Este documento apresenta o documento de projeto (*design*) arquitetural do sistema SCAP - Sistema de Controle de Afastamento de Professores. 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
PHP	7.3.1	Linguagem de programação de código-fonte aberto, interpretada no lado do servidor, especialmente adequada para o desenvolvimento Web.	Escrita do código-fonte das classes que compõem o sistema.
Apache HTTP Server	2.4.37	Servidor HTTP de código fonte aberto.	Hospedagem da aplicação Web, dando acesso aos usuários via HTTP.
MariaDB Server	10.1.37	Sistema Gerenciador de Banco de Dados Relacional gratuito.	Armazenamento dos dados manipulados pela ferramenta.
Bootstrap	3.4.1	Framework web, de código-fonte aberto, que disponibiliza componentes de interface baseados em HTML, CSS e JavaScript.	Reutilização de componentes visuais Web de alto nível.
Yii PHP framework	2.0	Conjunto de implementações, em PHP, de padrões de projeto e componentes comumente utilizados no ambiente Web.	Redução da complexidade no desenvolvimento de aplicações Web a partir de seus componentes prontos para o uso.

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
Astah UML	8.0	Ferramenta de diagramação UML.	Criação dos modelos de Entidades, Aplicação, Persistência e Navegação.
TeX Live	2017	Implementação do \LaTeX	Documentação do projeto arquitetural do sistema.
TeXstudio	2.12	Editor de LaTeX.	Escrita da documentação do sistema, sendo usado o <i>template abn-TeX</i> . ¹
JetBrains PhpStorm	2018.3	Ambiente de desenvolvimento (IDE) com suporte ao desenvolvimento PHP.	Implementação (codificação) da aplicação Web.
Composer	1.6.3	Ferramenta de gerência/-construção de projetos de software.	Obtenção e integração das dependências do projeto.

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 e Facilidade de Operação	RNF02, RNF03	Sim	Prover ao usuário a capacidade de entrar com comandos que permitam operar o sistema de modo mais eficiente. Para tal, as interfaces do sistema devem permitir, sempre que possível, a entrada de dados por meio de seleção ao invés da digitação. Fornecer mensagens de retorno para as ações do usuário. Fornecer, sempre que possível, dicas para preenchimento correto dos campos dos formulários.
Segurança	RNF01	Sim	Identificar usuários usando login e autenticá-los por meio de senha. Autorizar o acesso do usuário utilizando o padrão de projeto fornecido pelo framework.
Manutenibilidade e Portabilidade	RNF04, RNF05, RNF06, RNF07	Sim	Separar a interface com o usuário do restante da aplicação de acordo com o padrão MVC, facilitando assim a análise, modificação e testes de cada parte. Utilizar o padrão do <i>framework</i> para acesso a dados, o que facilitará a portabilidade para um outro SGBD (que seja suportado pelo <i>framework</i>) caso haja necessidade.
Reusabilidade	RNF07	Não	Reutilizar componentes existentes. Quando não existirem componentes disponíveis, mas houver potencial para reuso, devem ser desenvolvidos novos componentes de forma que possam ser reutilizados quando necessário.

4 Arquitetura de Software

O FrameWeb (SOUZA, 2007; SOUZA; FALBO; GUIZZARDI, 2009) indica a utilização de uma arquitetura baseada no padrão Camada de Serviço (FOWLER, 2002). A Figura 1 ilustra a arquitetura proposta pelo FrameWeb.

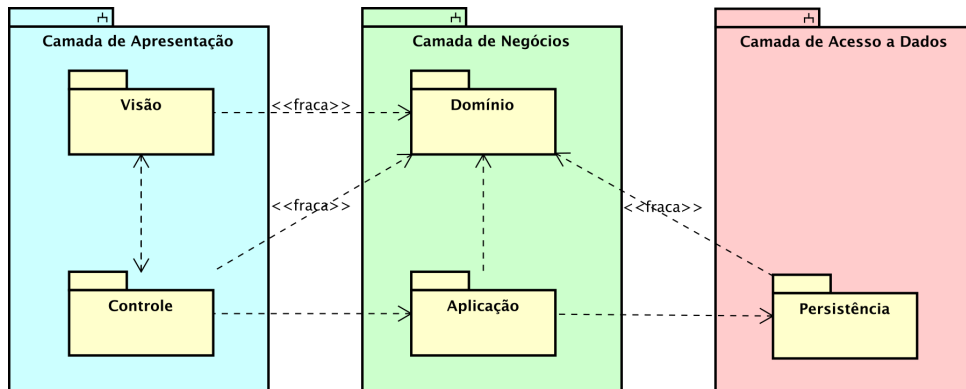


Figura 1 – Arquitetura padrão proposta pelo FrameWeb.

Neste projeto, no entanto, essa arquitetura não se encaixa perfeitamente pelo fato de a tecnologia ORM utilizada (incorporada no *framework* Yii) ser baseada no padrão *Active Record*. Uma das características do *Active Record*, segundo Flauzino (2017), é que “os *models* são fortemente acoplados à camada de acesso aos dados” diferentemente do que acontece no padrão Camada de Serviço (FOWLER, 2002) ilustrado na Figura 1. Dessa forma, como deseja-se experimentar a aplicação do método FrameWeb com uma nova plataforma, utilizando as características e funcionalidades que essa nova plataforma oferece, abriu-se mão da arquitetura padrão indicada pelo FrameWeb (unicamente pela incompatibilidade mencionada). Essa decisão terá reflexo nos diagramas das camadas de Negócios (Seção 4.2) e de Acesso a Dados (Seção 4.3).

4.1 Camada de Apresentação

Para guiar a implementação da camada de apresentação, o FrameWeb propõe a construção dos modelos de navegação. Neste projeto, cada caso de uso originou um modelo de navegação.

Os modelos de navegação referentes aos casos de uso “Registrar decisão PRPPG” e “Registrar decisão DI” são muito parecidos com o modelo de navegação do caso de uso “Registrar decisão CT” (muda apenas o nome do método e o nome do formulário) e por isso foram omitidos. Já o modelo referente ao caso de uso “Arquivar Afastamento” é muito parecido com o do caso de uso “Cancelar Afastamento” e por isso também foi omitido.

Algumas classes de ação (controladores) aparecem em mais de um modelo de navegação com uma lista de atributos diferentes. Isso foi feito para enfatizar os atributos que importam naquele caso de uso, conforme proposto pelo método FrameWeb: “Os atributos da classe de ação representam parâmetros de entrada e saída relevantes àquela ação” (SOUZA, 2007). A mesma estratégia foi adotada com os métodos da classe, exibindo somente aqueles que são relevantes para o caso de uso em questão. A implementação de cada controlador, no entanto, contém todos os atributos e métodos que aparecem em todos os modelos.

Nos modelos onde há duas saídas possíveis (resultado) para o mesmo método — Figuras 2, 3, 4, 5 — foi fundamental indicar qual resultado acionará cada saída. No caso específico destes modelos, o resultado “*success*” leva o usuário de volta à página que deu início ao caso de uso, e os resultados “*error*” ou “*null*” (quando não há resultado) levam o usuário para a página onde há o formulário para criação ou atualização do registro. Como não existe no FrameWeb uma notação que permita especificar múltiplos resultados em uma mesma saída, a seguinte notação foi utilizada: **{result=[error || null]}**. Assim, a saída que possui essa restrição será acionada na primeira chamada ao método, quando não há resultado (*null*) e nas chamadas subsequentes, quando há o envio dos dados do formulário, se houver algum erro (*error*).

Nos modelos onde a classe de ação (controlador) possui apenas uma entrada (evocação de método) e uma única saída (resultado) para o mesmo método — Figuras 6, 7, 8, 9, 10, 11 — não foi indicada a restrição **result** pois o usuário sempre será levado de volta à página que deu início ao caso de uso, independentemente do resultado. Assim, quando o usuário é redirecionado para essa página após a execução de uma ação, a página exibe inicialmente uma notificação indicando o resultado (*feedback*).

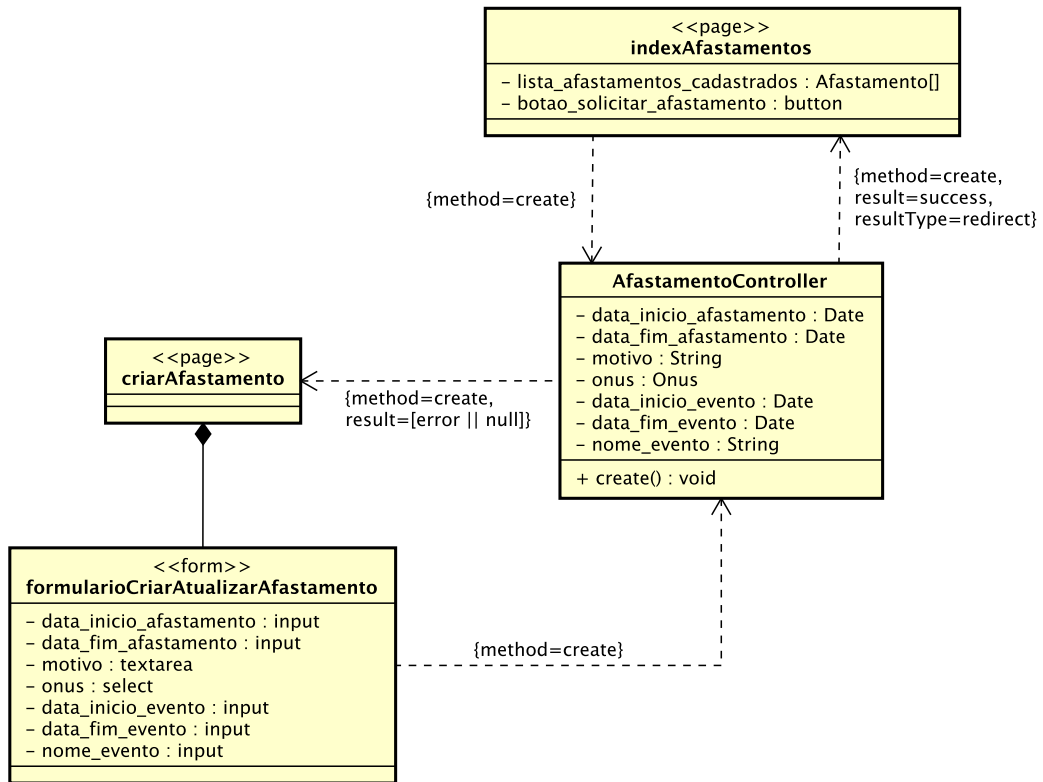


Figura 2 – Modelo de Navegação: Caso de uso - Solicitar Afastamento.

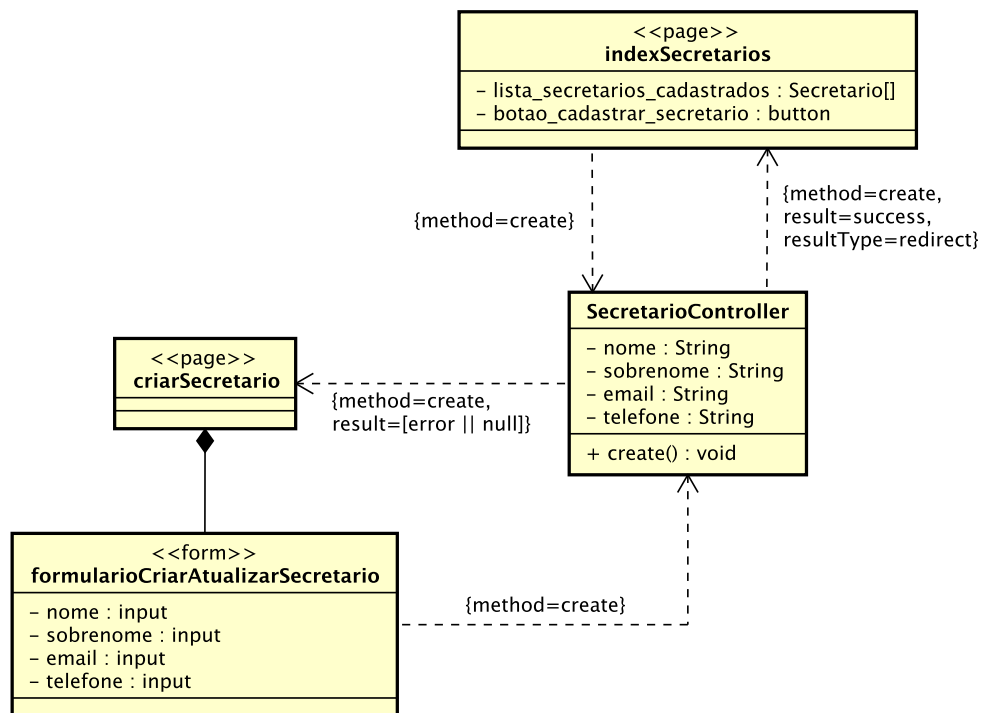


Figura 3 – Modelo de Navegação: Caso de uso - Cadastrar usuário (Secretário).

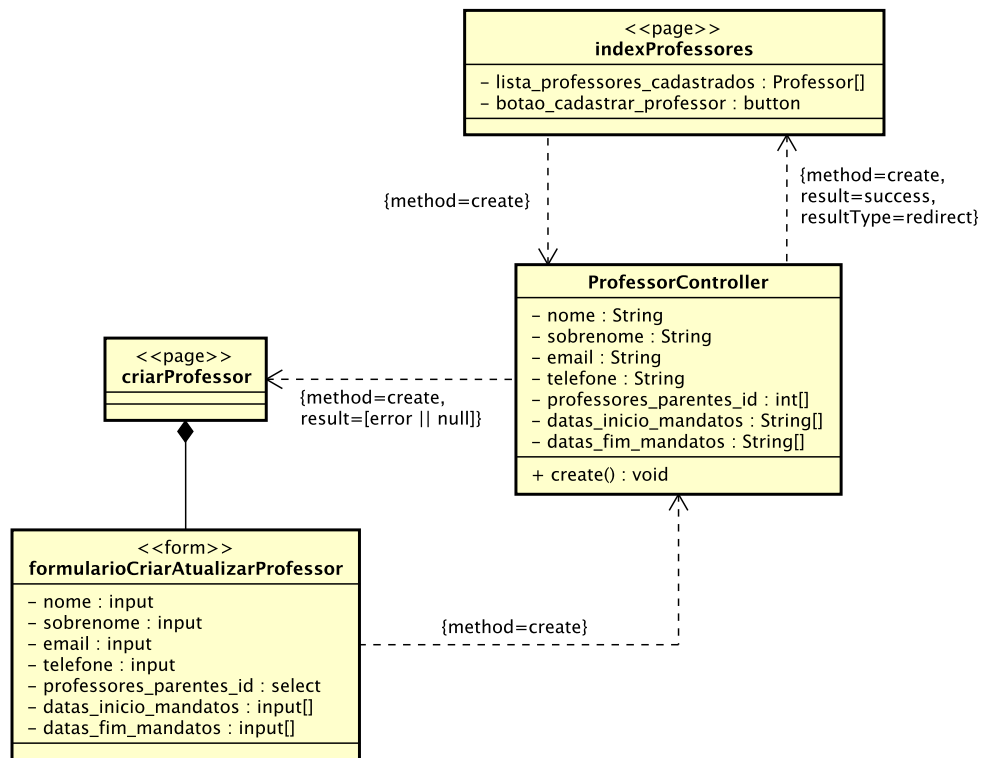


Figura 4 – Modelo de Navegação: Caso de uso - Cadastrar usuário (Professor).

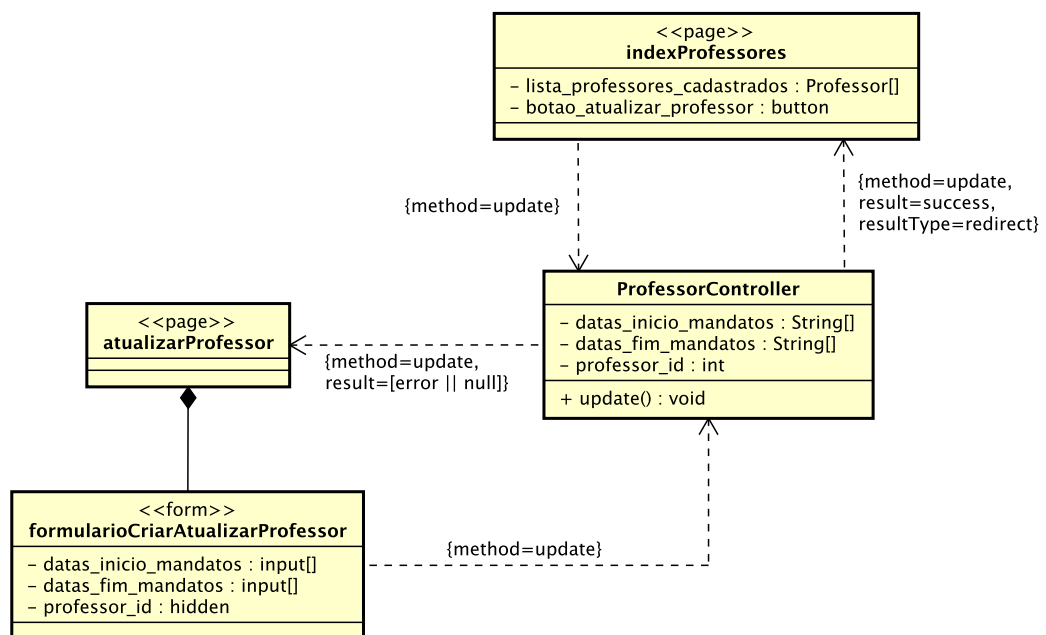


Figura 5 – Modelo de Navegação: Caso de uso - Cadastrar chefe do departamento.

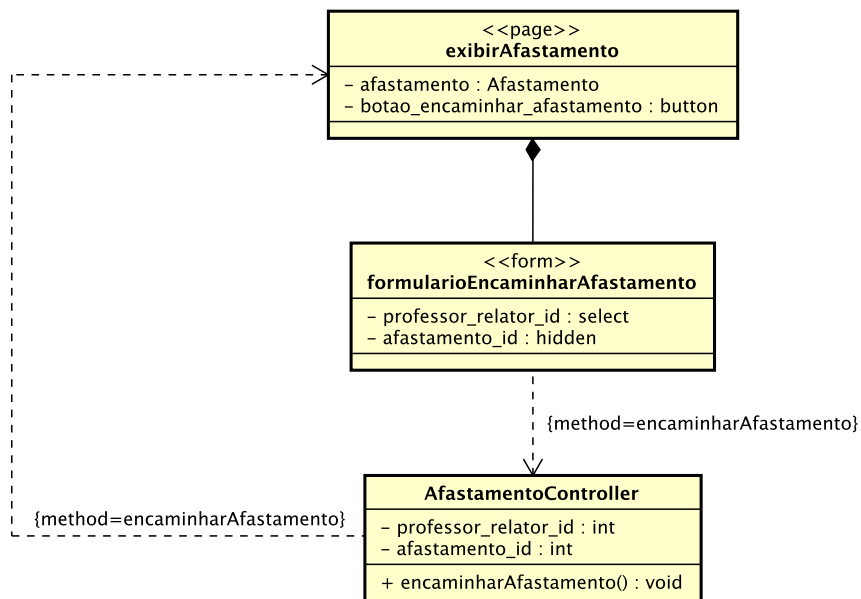


Figura 6 – Modelo de Navegação: Caso de uso - Encaminhar Afastamento.

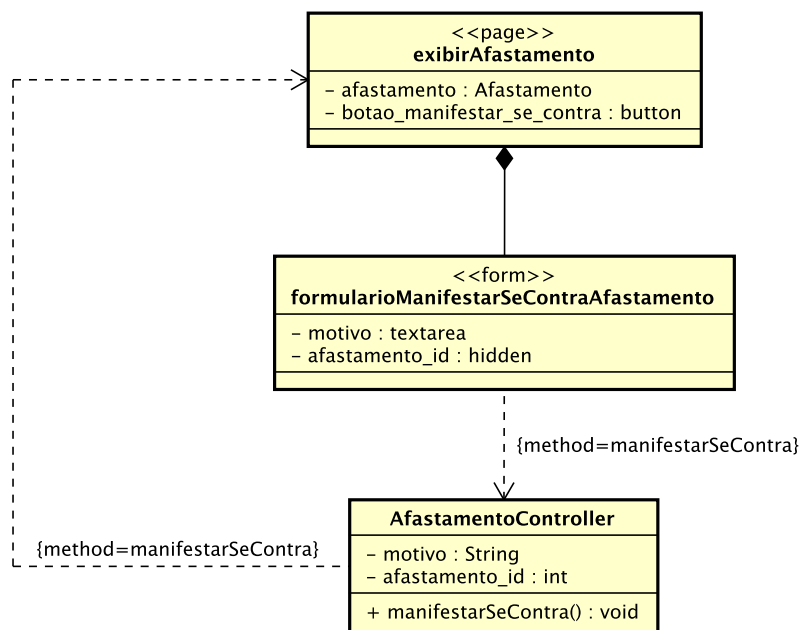


Figura 7 – Modelo de Navegação: Caso de uso - Manifestar-se Contra Afastamento.

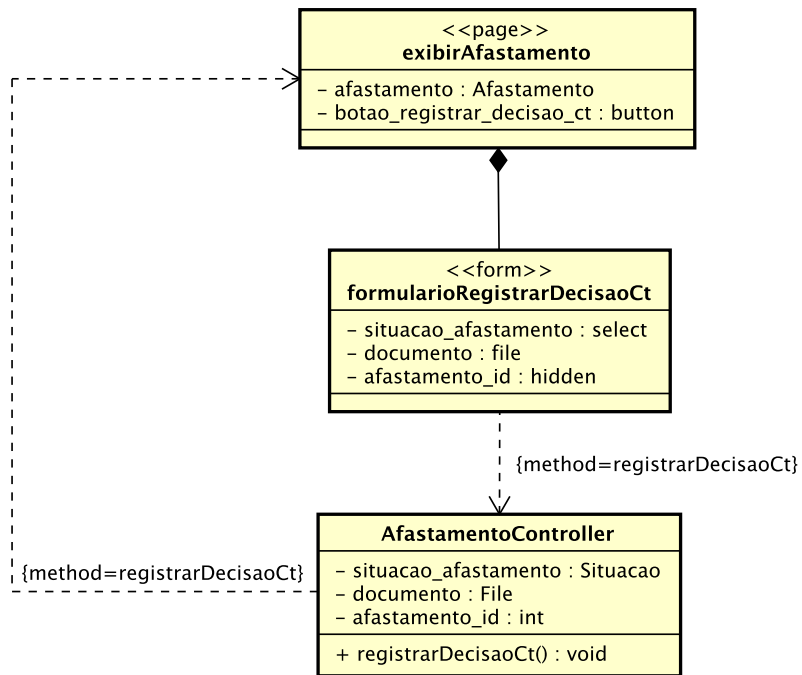


Figura 8 – Modelo de Navegação: Caso de uso - Registrar decisão CT.

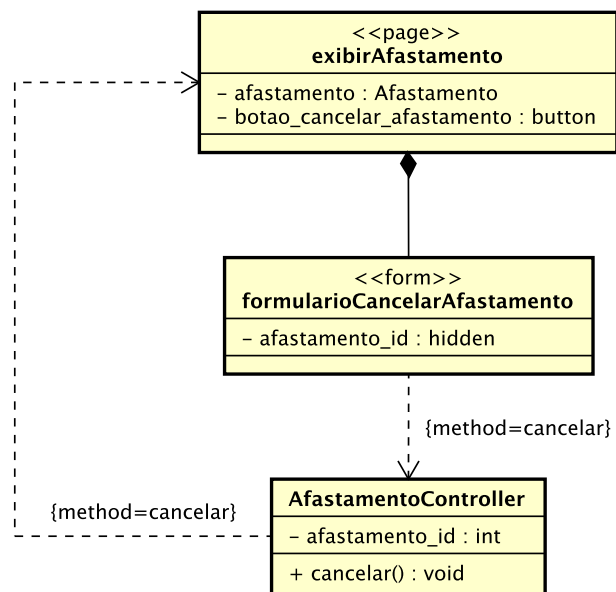


Figura 9 – Modelo de Navegação: Caso de uso - Cancelar Afastamento.

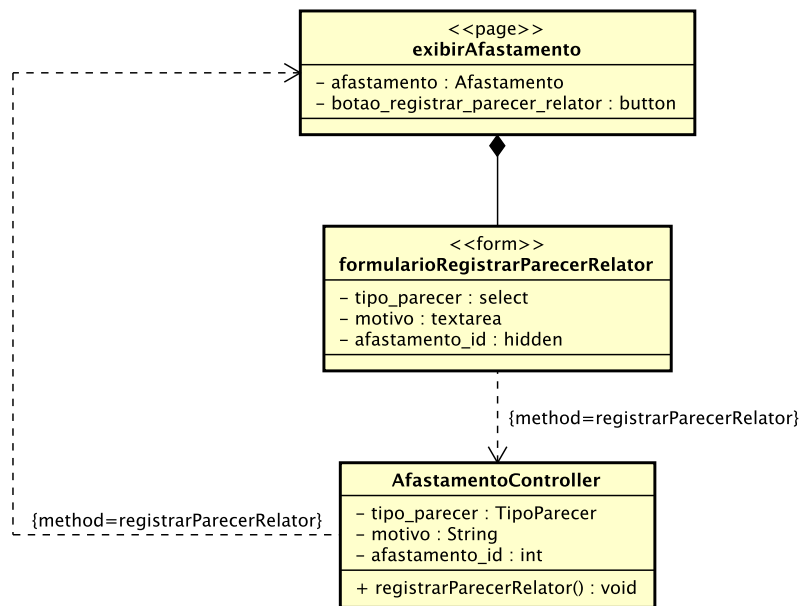


Figura 10 – Modelo de Navegação: Caso de uso - Registrar parecer relator.

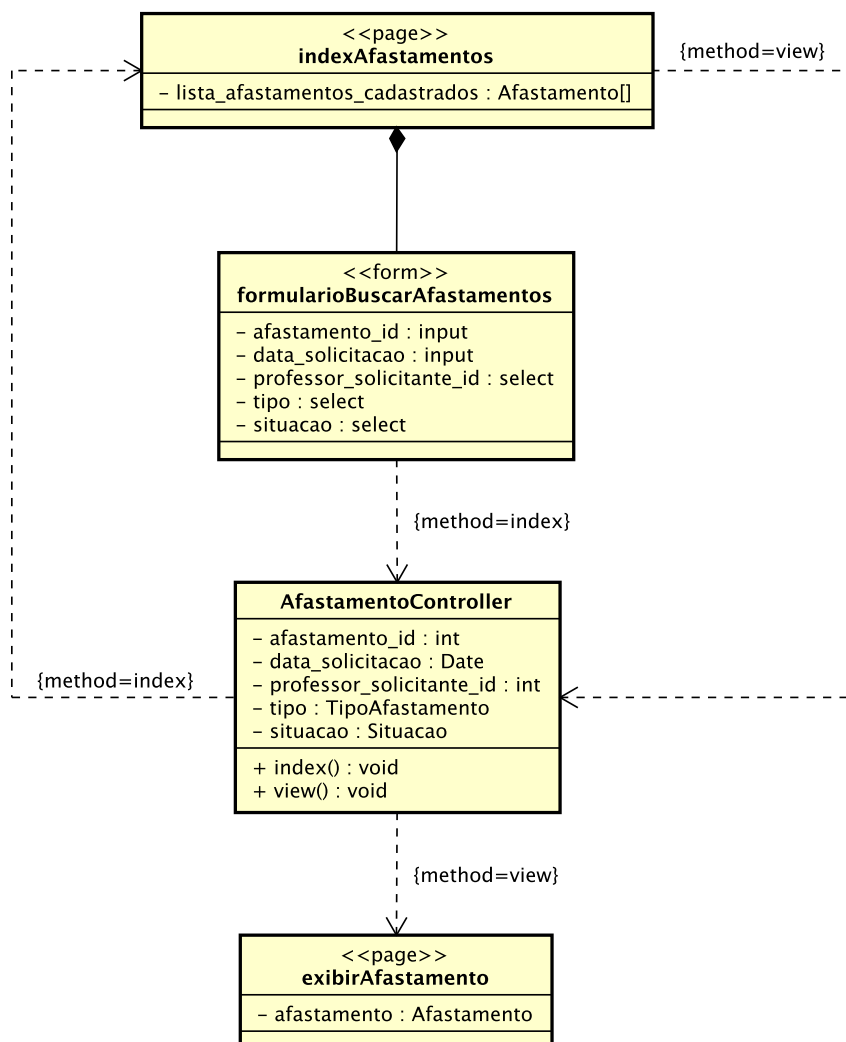


Figura 11 – Modelo de Navegação: Caso de uso - Consultar Afastamento.

4.2 Camada de Negócio

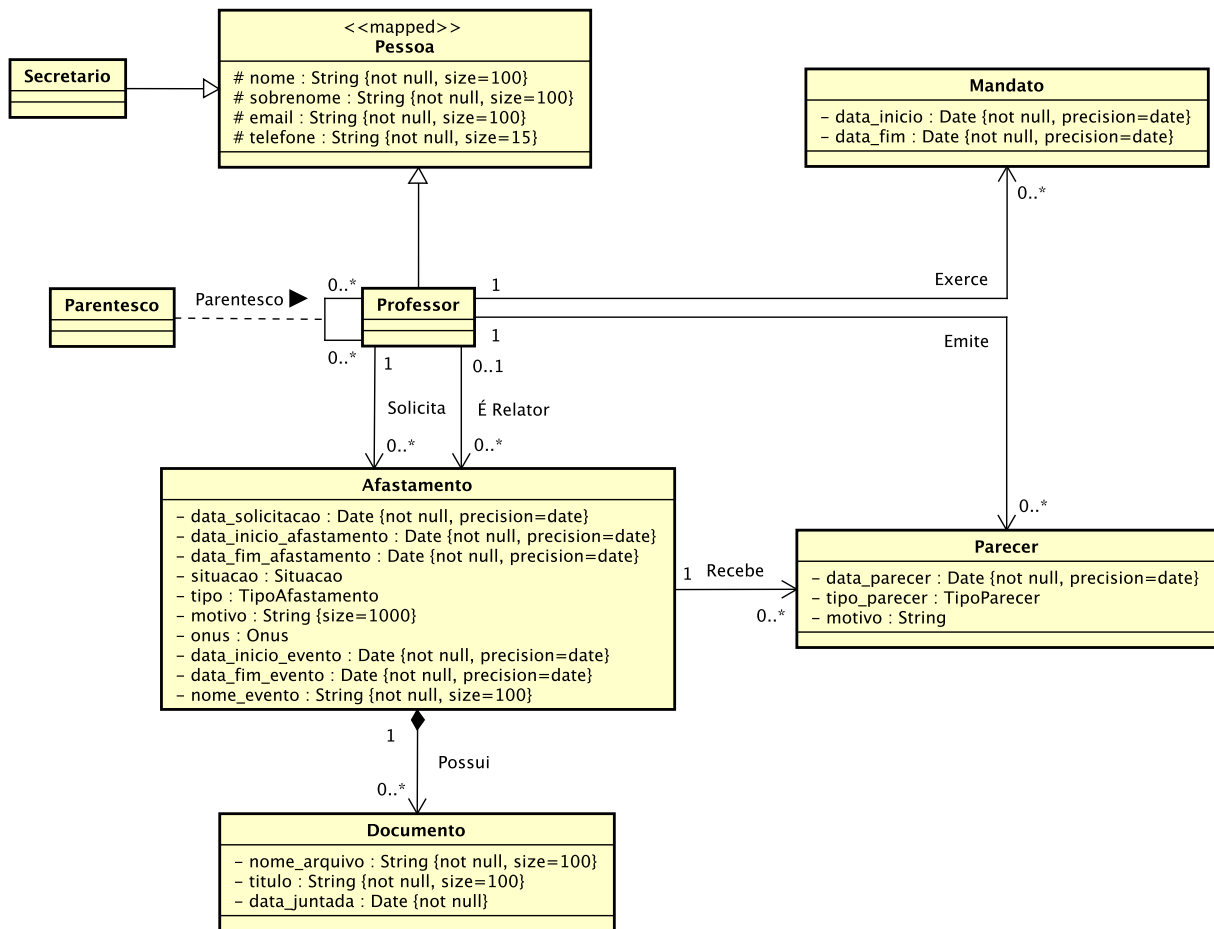


Figura 12 – Modelo de Entidades.

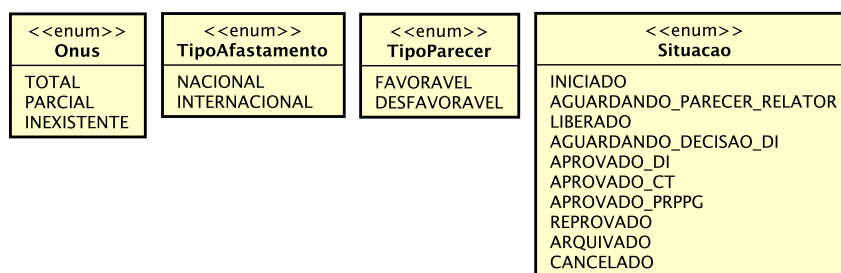


Figura 13 – Tipos enumerados.

Do modelo de Entidades é interessante mencionar que a classe Pessoa possui uma extensão (estereótipo de classe) com valor «*mapped*» indicando que a classe não é persistente, mas suas propriedades serão se alguém herdá-las, o que de fato ocorre.

O framework Yii já tem implementado um ORM que é baseado no padrão *Active Record*, ilustrado na Figura 14. Segundo Fowler (2002), o Active Record coloca a lógica de acesso a dados no objeto de domínio de forma que cada instância desse objeto sabe como ler e gravar seus dados no banco de dados.

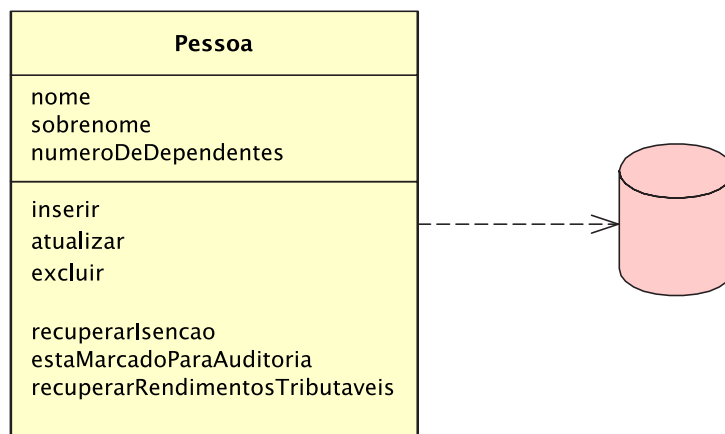


Figura 14 – Padrão Active Record. Adaptado de Fowler (2002).

Ainda segundo Fowler (2002), os objetos de domínio correspondem muito de perto à estrutura de registro do banco de dados, ou seja, cada atributo persistente do objeto de domínio corresponde a uma coluna na correspondente tabela do banco. Além disso, cada instância do objeto de domínio é responsável por qualquer lógica de domínio que atue sobre os dados.

Considerando isso, houve a necessidade de ajustar um pouco o modelo de aplicação sugerido pelo FrameWeb. Primeiramente, como os objetos de domínio são responsáveis pela lógica de domínio, todos os métodos que estariam em classes de serviço agora estão nas classes de domínio (models). Além disso, as associações com as interfaces DAO já não são necessárias pois a lógica de acesso a dados está no próprio objeto de domínio.

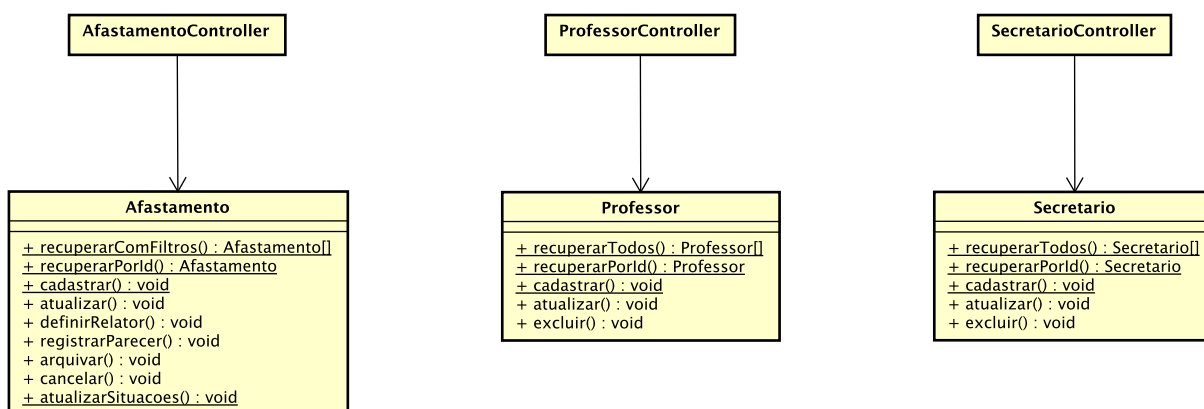


Figura 15 – Modelo de Aplicação.

4.3 Camada de Acesso a Dados

Como mencionado na seção anterior, a lógica de acesso a dados fica no próprio objeto de domínio, de acordo com o padrão *Active Record*. Dessa forma, não serão necessárias a criação de interfaces e implementações de DAOs nesse projeto. Portanto não será apresentado o modelo de persistência.

Referências

- FLAUZINO, M. Entenda ORM: Active Record e Data Mapper. Medium, nov 2017. Disponível em: <<https://medium.com/@matheusflauzino/entenda-orm-active-record-e-data-mapper-9be60da0e799>>. Citado na página 5.
- FOWLER, M. *Patterns of Enterprise Application Architecture*. 1. ed. [S.l.]: Addison-Wesley, 2002. ISBN 9780321127426. Citado 3 vezes nas páginas 5, 12 e 13.
- 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 6.
- 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.



Documento de Projeto de Sistema

SCAP - Sistema de Controle de Afastamento de Professores

Registro de Alterações:

Versão	Responsável	Data	Alterações
1.0	Vinícius Berger	22/03/2019	Versão Inicial
1.1	Vinícius Berger	24/01/2021	Adicionados os modelos FrameWeb
1.2	Vinícius Berger	08/03/2021	Correções
1.3	Vinícius Berger	22/03/2021	Correções

Vitória, ES

2021

1 Introdução

Este documento apresenta o documento de projeto (*design*) arquitetural do sistema SCAP - Sistema de Controle de Afastamento de Professores. 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
PHP	7.3.1	Linguagem de programação de código-fonte aberto, interpretada no lado do servidor, especialmente adequada para o desenvolvimento Web.	Escrita do código-fonte das classes que compõem o sistema.
Apache HTTP Server	2.4.37	Servidor HTTP de código fonte aberto.	Hospedagem da aplicação Web, dando acesso aos usuários via HTTP.
MariaDB Server	10.1.37	Sistema Gerenciador de Banco de Dados Relacional gratuito.	Armazenamento dos dados manipulados pela ferramenta.
Bootstrap	3.4.1	Framework web, de código-fonte aberto, que disponibiliza componentes de interface baseados em HTML, CSS e JavaScript.	Reutilização de componentes visuais Web de alto nível.
Symfony framework	5.2	Conjunto de implementações, em PHP, de padrões de projeto e componentes comumente utilizados no ambiente Web.	Redução da complexidade no desenvolvimento de aplicações Web a partir de seus componentes prontos para o uso.
Doctrine	2.7	API para persistência de dados por meio de mapeamento objeto/-relacional.	Persistência dos objetos de domínio sem necessidade de escrita dos comandos SQL.

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
Astah UML	8.0	Ferramenta de diagramação UML.	Criação dos modelos de Entidades, Aplicação, Persistência e Navegação.
TeX Live	2017	Implementação do \LaTeX	Documentação do projeto arquitetural do sistema.
TeXstudio	2.12	Editor de LaTeX.	Escrita da documentação do sistema, sendo usado o <i>template abn-TeX</i> . ¹
JetBrains PhpStorm	2018.3	Ambiente de desenvolvimento (IDE) com suporte ao desenvolvimento PHP.	Implementação (codificação) da aplicação Web.
Composer	1.6.3	Ferramenta de gerência/-construção de projetos de software.	Obtenção e integração das dependências do projeto.

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 e Facilidade de Operação	RNF02, RNF03	Sim	(i) Prover ao usuário a capacidade de entrar com comandos que permitam operar o sistema de modo mais eficiente. Para tal, as interfaces do sistema devem permitir, sempre que possível, a entrada de dados por meio de seleção ao invés da digitação. (ii) Fornecer mensagens de retorno para as ações do usuário. (iii) Fornecer, sempre que possível, dicas para preenchimento correto dos campos dos formulários.
Segurança	RNF01	Sim	Identificar usuários usando login e autenticá-los por meio de senha. Autorizar o acesso do usuário utilizando o padrão de projeto fornecido pelo framework.
Manutenibilidade e 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 de acordo com o padrão Camada de Serviço. A camada de gerência de dados deve ser organizada de acordo com o padrão DAO. Separar a interface com o usuário do restante da aplicação de acordo com o padrão MVC.
Reusabilidade	RNF07	Não	Reutilizar componentes existentes. Quando não existirem componentes disponíveis, mas houver potencial para reuso, devem ser desenvolvidos novos componentes de forma que possam ser reutilizados quando necessário.

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 proposta pelo FrameWeb.

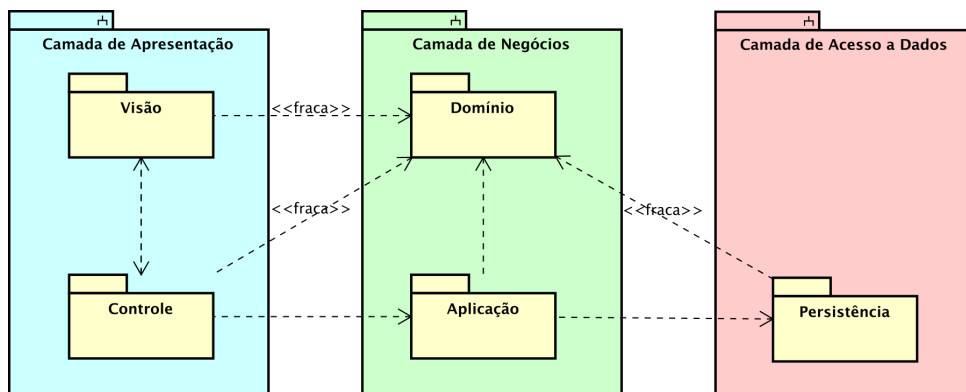


Figura 1 – Arquitetura padrão proposta pelo FrameWeb.

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

4.1 Camada de Apresentação

Para guiar a implementação da camada de apresentação, o FrameWeb propõe a construção dos modelos de navegação. Neste projeto, cada caso de uso originou um modelo de navegação.

Os modelos de navegação referentes aos casos de uso “Registrar decisão PRPPG” e “Registrar decisão DI” são muito parecidos com o modelo de navegação do caso de uso “Registrar decisão CT” (muda apenas o nome do método e o nome do formulário) e por isso foram omitidos. Já o modelo referente ao caso de uso “Arquivar Afastamento” é muito parecido com o do caso de uso “Cancelar Afastamento” e por isso também foi omitido.

Algumas classes de ação (controladores) aparecem em mais de um modelo de navegação com uma lista de atributos diferentes. Isso foi feito para enfatizar os atributos que importam naquele caso de uso, conforme proposto pelo método FrameWeb: “Os atributos da classe de ação representam parâmetros de entrada e saída relevantes àquela ação” (SOUZA, 2007). A mesma estratégia foi adotada com os métodos da classe, exibindo somente aqueles que são relevantes para o caso de uso em questão. A implementação de cada controlador, no entanto, contém todos os atributos e métodos que aparecem em todos

os modelos.

Nos modelos onde há duas saídas possíveis (resultado) para o mesmo método — Figuras 2, 3, 4, 5 — foi fundamental indicar qual resultado acionará cada saída. No caso específico destes modelos, o resultado “*success*” leva o usuário de volta à página que deu início ao caso de uso, e os resultados “*error*” ou “*null*” (quando não há resultado) levam o usuário para a página onde há o formulário para criação ou atualização do registro. Como não existe no FrameWeb uma notação que permita especificar múltiplos resultados em uma mesma saída, a seguinte notação foi utilizada: **{result=[error || null]}**. Assim, a saída que possui essa restrição será acionada na primeira chamada ao método, quando não há resultado (*null*) e nas chamadas subsequentes, quando há o envio dos dados do formulário, se houver algum erro (*error*).

Nos modelos onde a classe de ação (controlador) possui apenas uma entrada (evocação de método) e uma única saída (resultado) para o mesmo método — Figuras 6, 7, 8, 9, 10, 11 — não foi indicada a restrição **result** pois o usuário sempre será levado de volta à página que deu início ao caso de uso, independentemente do resultado. Assim, quando o usuário é redirecionado para essa página após a execução de uma ação, a página exibe inicialmente uma notificação indicando o resultado (*feedback*).

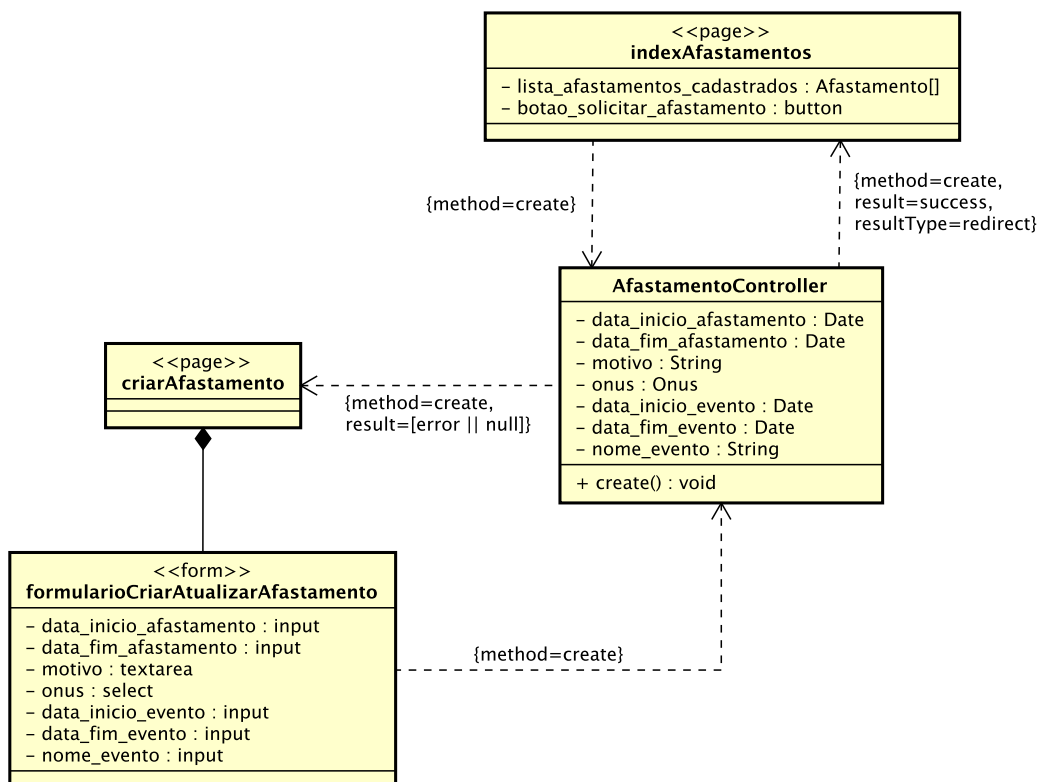


Figura 2 – Modelo de Navegação: Caso de uso - Solicitar Afastamento.

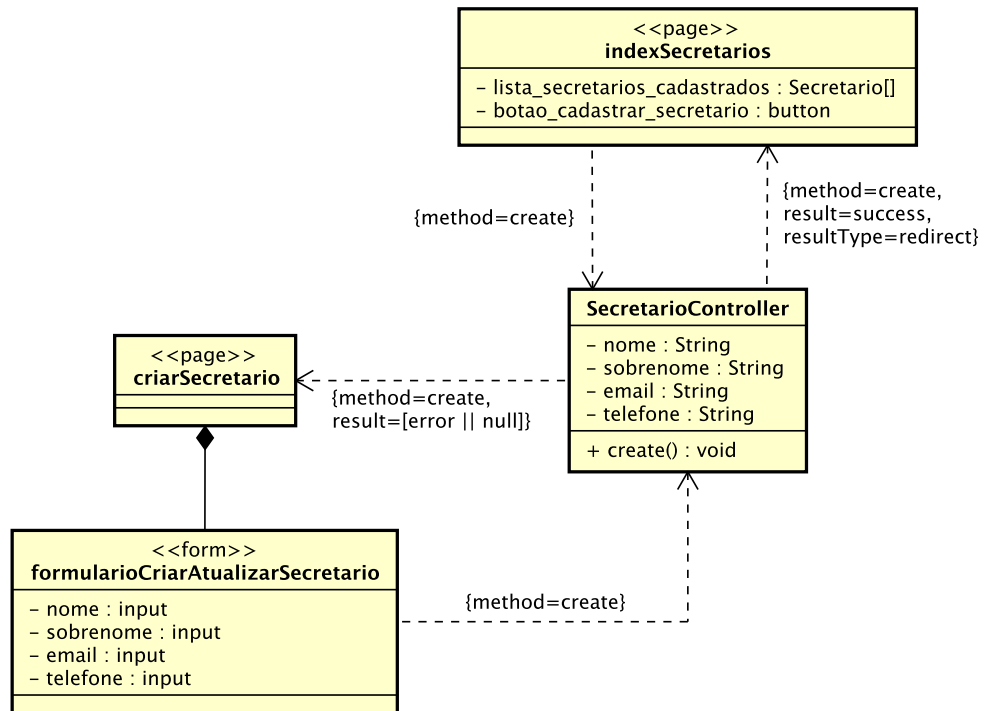


Figura 3 – Modelo de Navegação: Caso de uso - Cadastrar usuário (Secretário).

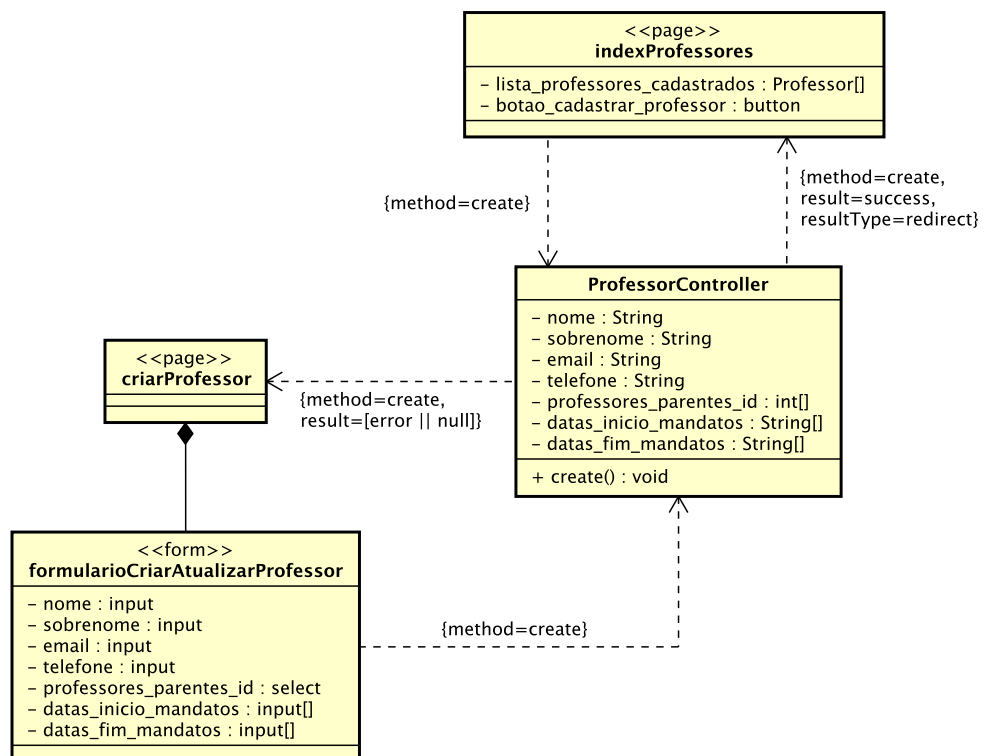


Figura 4 – Modelo de Navegação: Caso de uso - Cadastrar usuário (Professor).

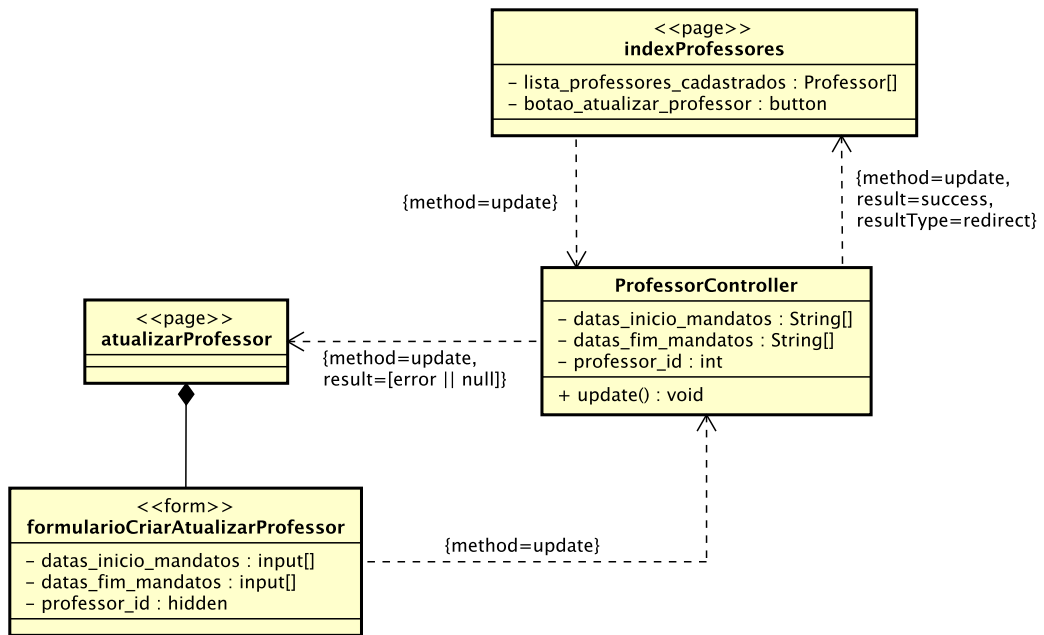


Figura 5 – Modelo de Navegação: Caso de uso - Cadastrar chefe do departamento.

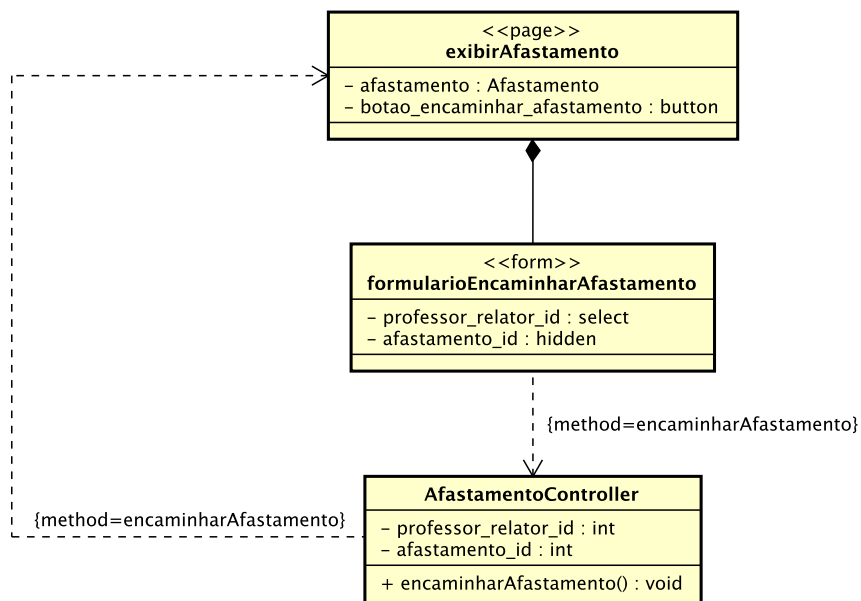


Figura 6 – Modelo de Navegação: Caso de uso - Encaminhar Afastamento.

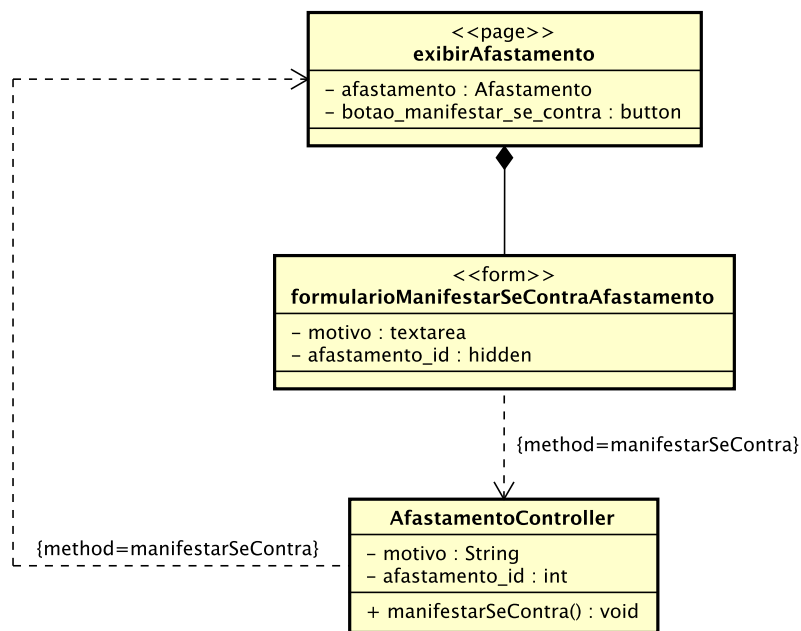


Figura 7 – Modelo de Navegação: Caso de uso - Manifestar-se Contra Afastamento.

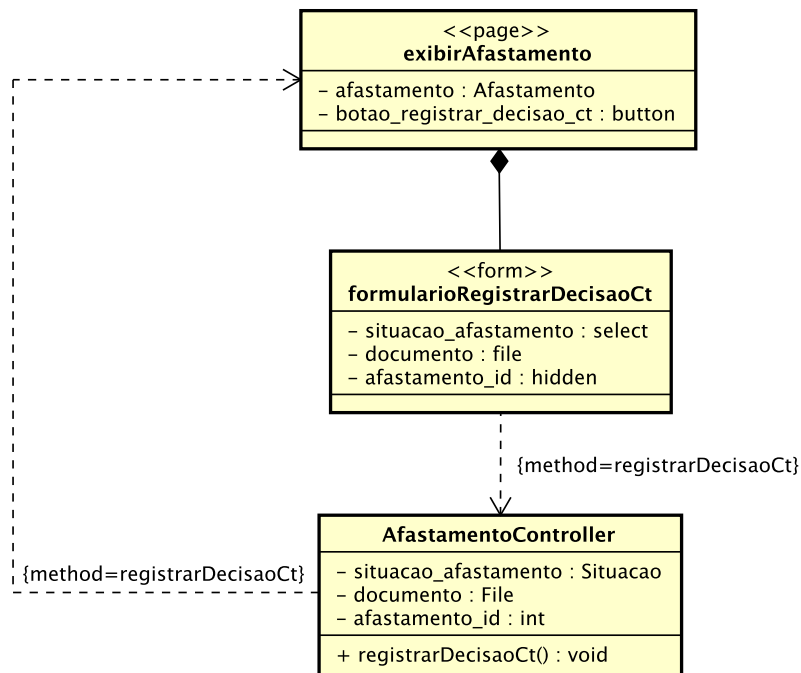


Figura 8 – Modelo de Navegação: Caso de uso - Registrar decisão CT.

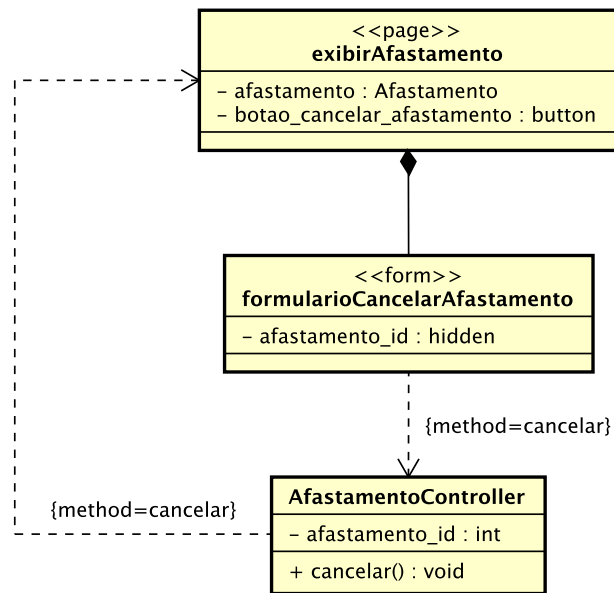


Figura 9 – Modelo de Navegação: Caso de uso - Cancelar Afastamento.

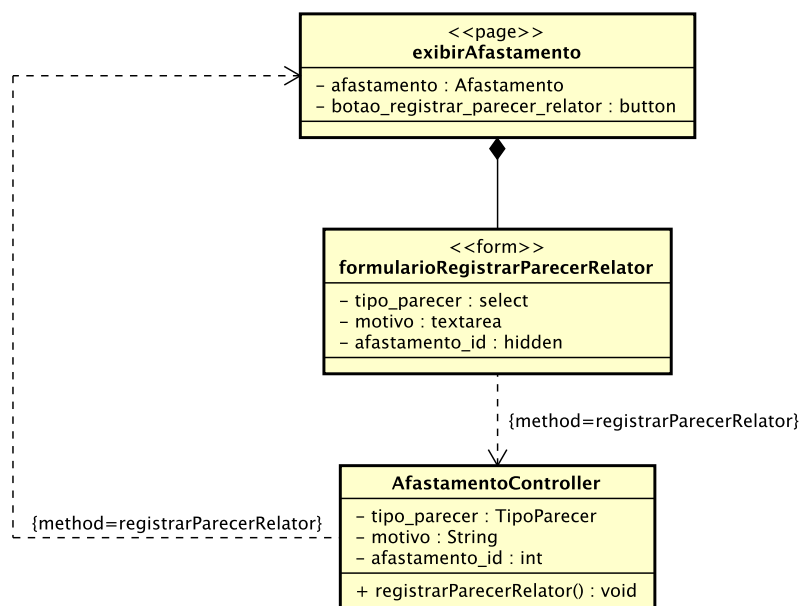


Figura 10 – Modelo de Navegação: Caso de uso - Registrar parecer relator.

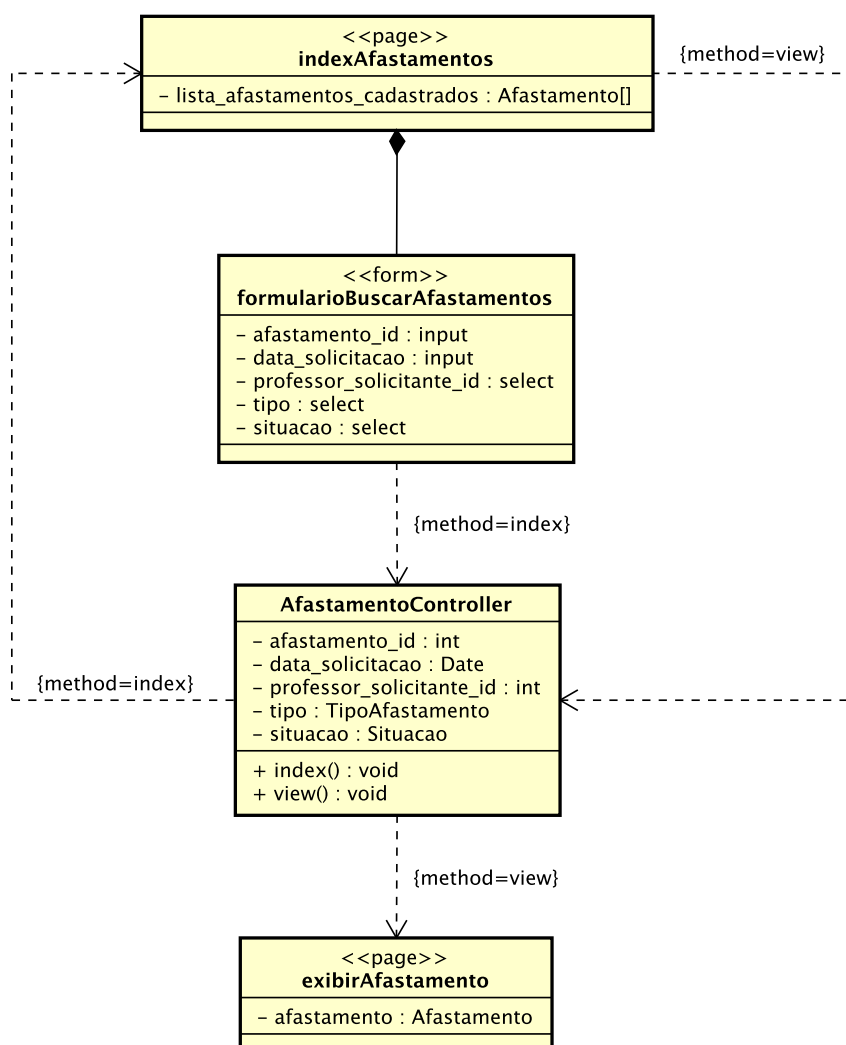


Figura 11 – Modelo de Navegação: Caso de uso - Consultar Afastamento.

4.2 Camada de Negócio

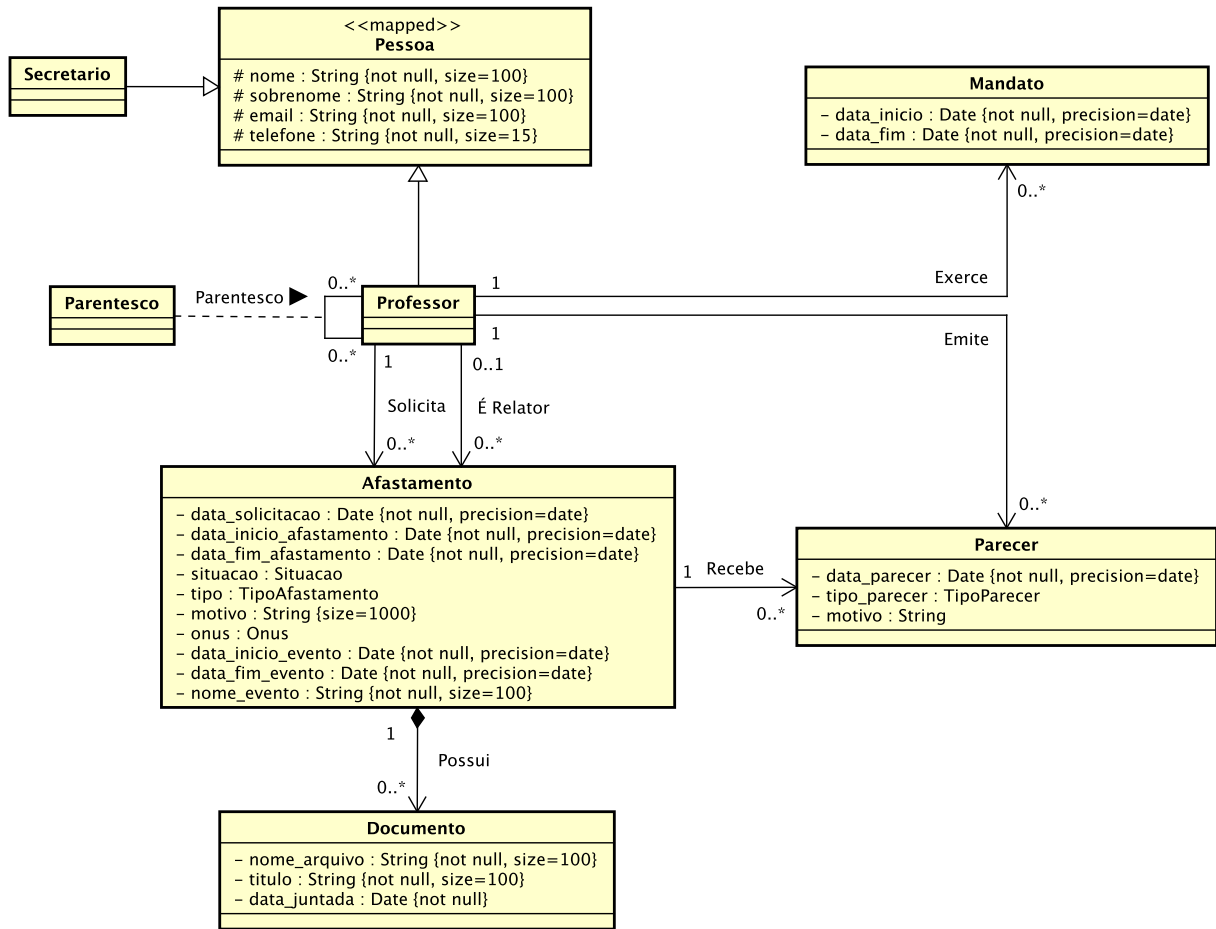


Figura 12 – Modelo de Entidades.

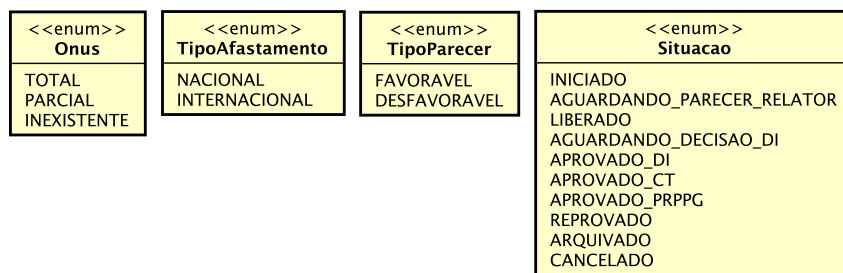


Figura 13 – Tipos enumerados.

Do modelo de Entidades é interessante mencionar que a classe Pessoa possui uma extensão (estereótipo de classe) com valor «mapped» indicando que a classe não é persistente, mas suas propriedades serão se alguém herdá-las, o que de fato ocorre.

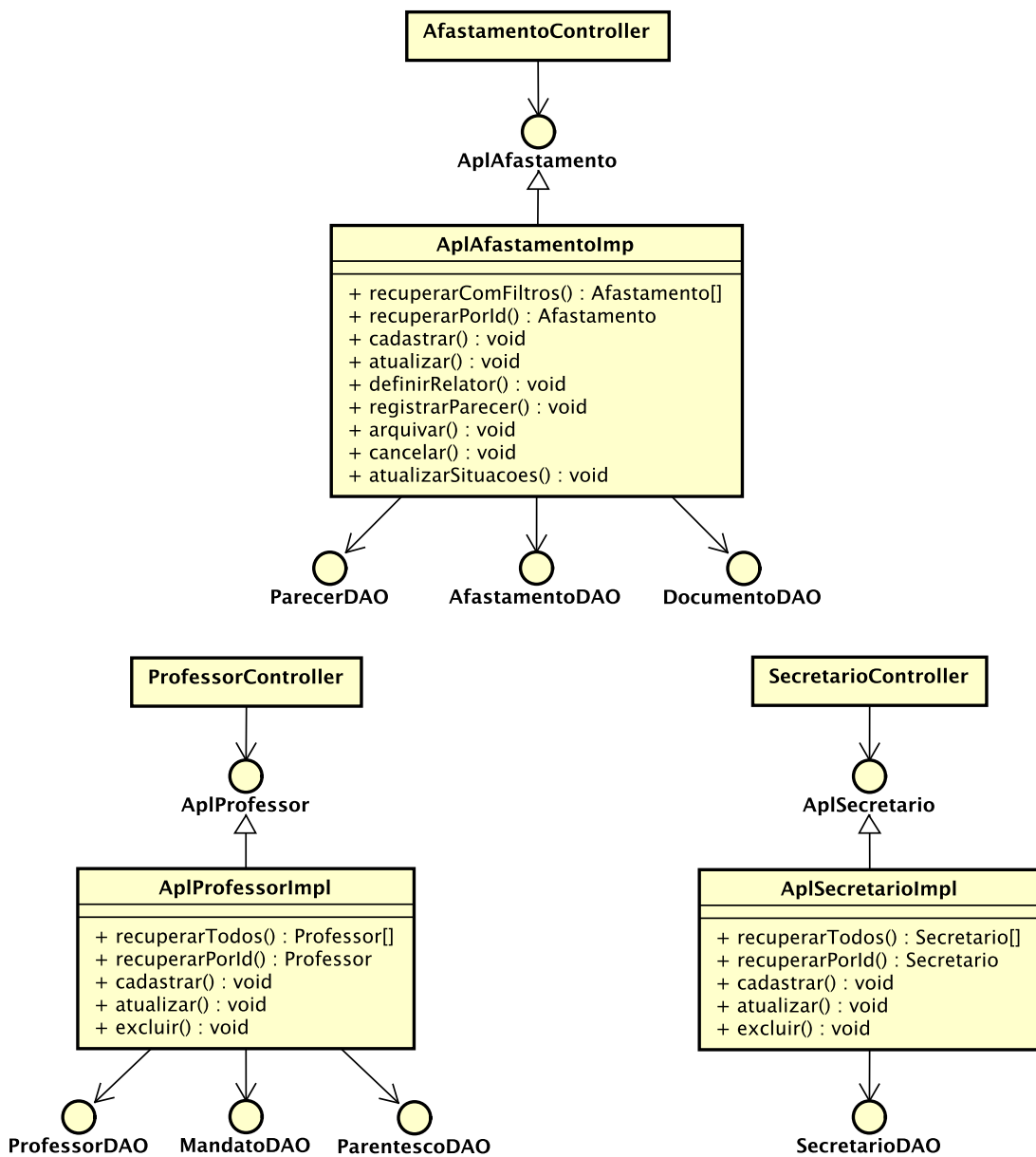


Figura 14 – Modelo de Aplicação.

4.3 Camada de Acesso a Dados

A Figura 15 apresenta o Modelo de Persistência do SCAP. De acordo com Souza (2007), para não repetir operações comuns em todas as interfaces DAO, é permitido apresentar uma interface base que declara esses métodos e uma implementação concreta dessa interface de forma que todas as outras interfaces herdam as definições da interface base e todas as demais classes herdam da implementação concreta base, sem que isso precise estar explícito no diagrama. Assim, na Figura 15 é possível ver a interface “DAOBase” e a implementação concreta “DAOBaseDoctrine” que são responsáveis por definir e implementar os métodos comuns.

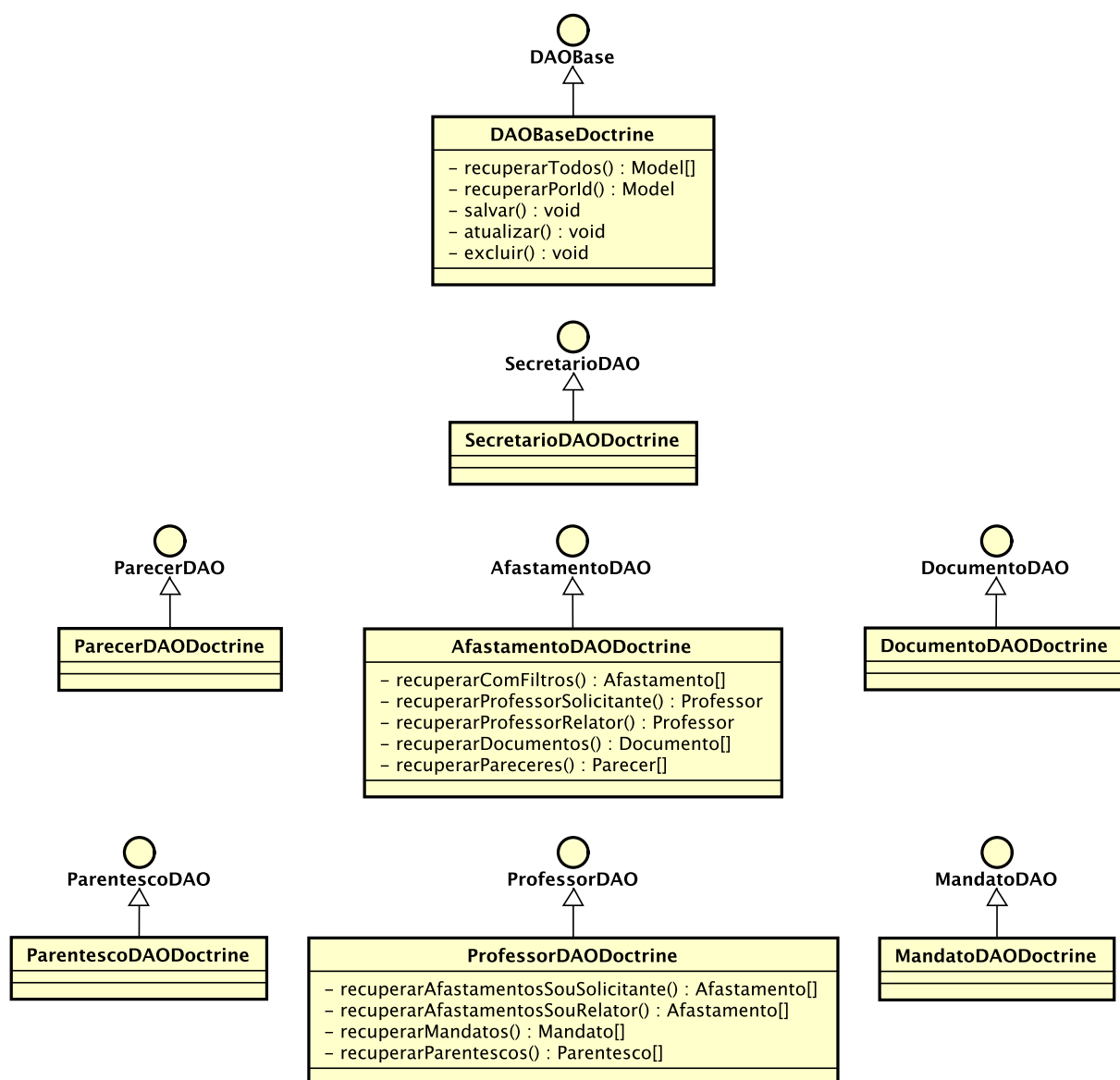


Figura 15 – Modelo de Persistência.

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.