Luiz Meirelles

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Codeigniter e NodeJS.

Vitória, ES

2019

Luiz Meirelles

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Codeigniter e NodeJS.

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

2019

Luiz Meirelles

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Codeigniter e NodeJS./ Luiz Meirelles. – Vitória, ES, 2019-

49 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, 2019.

1. Palavra-chave1. 2. Palavra-chave2. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Codeigniter e NodeJS.

CDU 02:141:005.7

Luiz Meirelles

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Codeigniter e NodeJS.

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, 13 de dezembro de 2019:

Prof. Dr. Vítor E. Silva Souza
Orientador

Prof. Dr. Davidson Cury
Universidade Federal do Espírito Santo

César Henrique Bernabé
Universidade Federal do Espírito Santo

Vitória, ES
2019

A minha família que me deu todo suporte.

Agradecimentos

Em primeiro lugar quero agradecer a Deus, pois, ele me deu forças e sabedoria nos momentos mais difíceis. Nem escrevendo um livro eu poderia descrever o sentimento de dificuldade que tive, só Deus sabe o que passei.

Em segundo lugar quero agradecer a minha família, minha namorada e a família de minha namorada. Todos tiveram de suportar minhas angústias, ausências, vontade de sumir e compromissos esquecidos. Todos de alguma forma nessa caminhada colaboraram para esse dia, desde uma simples conversa até uma janta pronta me esperando. Sou agradecido a todos, pois, sem casa, comida, transporte e as coisas básicas que uma família pode fornecer as coisas teriam sido muito mais difíceis.

Em terceiro lugar quero agradecer aos meus colegas que me acompanharam nessa jornada. Cada um me ajudou de alguma forma, desde uma conversa motivadora até com trabalhos a serem feitos. Nessa vida nunca sabemos como lidar com tudo e numa amizade a gente pode aprender com o outro a como lidar melhor com uma situação.

Em quarto lugar quero agradecer a UFES como um todo. Todo o corpo docente e administrativo que fez parte dessa caminhada. Aprendi muito com meus professores e pessoas do administrativo. Com os professores não só em termos de matéria mas novas formas de enxergar e pensar. Com pessoas do administrativos, via as dificuldades encontradas para realizarem seus trabalhos e o famoso jogo de cintura para fazer as coisas andarem. Estudar na UFES foi um privilégio que inicialmente admito não reconhecer, mas, com o tempo aprendi a dar valor, ver colegas que faziam estágio e ainda tinham de pagar para estudarem é complicado. A UFES como qualquer instituição é formada por seres humanos e como todos sabem seres humanos não são perfeitos.

"A Paz que você procura, está no silêncio que você não faz."
(Autor Desconhecido)

Resumo

Com o crescimento e desenvolvimento da Internet, as aplicações foram ficando mais complexas. Inicialmente as primeiras aplicações foram implementadas de maneira *ad-hoc* e sem o emprego de alguma metodologia de desenvolvimento. Logo surgiu a necessidade do uso de conceitos de Engenharia de Software aplicados ao desenvolvimento de sistemas e aplicações Web. Surge então a Engenharia Web (*Web Engineering* ou WebE), que contém vários métodos propostos para análise, projeto e desenvolvimento de Sistemas de Informação baseados na Web (*Web-based Information System* – WISs).

Nesse cenário nasce o método FrameWeb (*Framework-based Design Method for Web Engineering*), que sugere a utilização de vários *frameworks*, com objetivo de ajudar o desenvolvimento e tornar mais rápida e eficiente a construção de sistemas de informação para a Web. O FrameWeb dá suporte a algumas categorias de *frameworks* e, para cada uma, existem diversos *frameworks* que podem ser utilizados no desenvolvimento de uma aplicação. Surge, então, a necessidade de verificar se o método se aplica bem aos vários *frameworks* existentes dentro de cada categoria à qual FrameWeb dá suporte.

O objetivo deste trabalho é reimplementar parte de um sistema já existente, o SCAP: Sistema de Controle de Afastamento de Professores, testando o método FrameWeb com os *frameworks* Codeigniter (em PHP) e NodeJS (em JavaScript), complementando experimentos anteriores do método, que utilizaram outros *frameworks*. O SCAP é um sistema que permite o controle dos afastamentos que os professores do Departamento de Informática da UFES, visando auxiliar os professores no processo de solicitação de afastamento.

Palavras-chaves: Engenharia Web. FrameWeb. Codeigniter. JavaScript. NodeJS.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UML	Unified Modeling Language
CRUD	Create, Read, Update and Delete.
MVC	Model, View and Controller.

Sumário

1 Introdução

Cada vez mais acessamos a Internet e este acesso vem ficando cada dia mais rápido, disponível e estável. Está cada vez mais presente em nosso dia a dia, seja para trabalho ou entretenimento. Com isso, as aplicações estão se tornando mais complexas de forma a atender um determinado público para um determinado fim. Surge a demanda por software de alta qualidade para suprir requisitos como segurança, velocidade, compatibilidade, usabilidade e etc.

Com o passar do tempo o modo *ad hoc* para desenvolvimento se torna insuficiente. É necessário então aplicar conceitos de Engenharia de Software no desenvolvimento dos então chamados *WebApps* (aplicações para a Web), conjunto de páginas Web que interagem com o visitante, provendo, armazenando e processando informações (SOUZA, 2007). Passou-se, então, a empregar conceitos de Engenharia de Software e ferramentas para auxiliar no desenvolvimento de sistemas e aplicações que funcionem na Web. Neste contexto foi criada a Engenharia Web (PRESSMAN, 2011), ou WebE (*Web Engineering*).

FrameWeb (*Framework-based Design Method for Web Engineering*) é um método que visa a construção rápida e eficiente de WebApps, por meio da utilização de *frameworks*. O método foca em Sistemas de Informação baseados na Web (*Web-based Information Systems* – WISs) que são como sistemas de informação tradicionais, porém disponíveis na Internet, ou em uma Intranet, como é o caso de lojas virtuais, ambientes cooperativos, sistemas de gerência empresarial, dentre muitos outros (SOUZA, 2007).

O FrameWeb apresenta, um perfil UML (extensão da linguagem UML) com quatro tipos de modelos para a fase de projeto. Tais modelos representam conceitos utilizados por algumas categorias de *frameworks*, facilitando, desta forma, a comunicação entre as equipes de projetistas e a de programadores durante o desenvolvimento da aplicação.

O FrameWeb dá suporte a certas categorias de *frameworks* e, dentro de cada categoria, há várias opções de *frameworks* para serem utilizados na implementação. Na proposta original foram feitos experimentos com apenas um *framework* de cada categoria, de modo que não é possível saber se o método é adequado a um conjunto diferente de *frameworks* dentro das mesmas categorias.

Em seu trabalho de conclusão de curso, Duarte (2014) desenvolveu uma WebApp — o Sistema de Controle de Afastamentos de Professores, ou SCAP — com o método FrameWeb, utilizando um conjunto diferente de *frameworks*, baseado na plataforma Java EE 7 (DEMICHIEL; SHANNON, 2013). O trabalho serve de base para uma análise do método com vistas a uma futura adequação a uma gama maior de *frameworks*. Por exemplo, Prado (2015) desenvolveu o SCAP com o *framework* controlador VRaptor, Ferreira (2018)

com os *frameworks* controladores Tapestry e Wicket, Avelar (2018) com o *framework fullstack* Ninja, dentre outros.

O objetivo deste trabalho é, a partir dos requisitos levantados em (DUARTE, 2014) e refinados em (PRADO, 2015), realizar uma nova implementação do SCAP aplicando o FrameWeb, porém utilizando o *framework* CodeIgniter, na plataforma PHP, bem como com JavaScript puro interpretado pelo NodeJS.

1.1 Objetivos

O objetivo deste trabalho é aplicar o método FrameWeb (SOUZA, 2007) na implementação da aplicação SCAP (Sistema de Controle de Afastamento de Professores) a partir de requisitos levantados por Duarte (2014) e Prado (2015), utilizando dois novos *frameworks*, CodeIgniter (PHP) e NodeJS (JavaScript), contribuindo, assim, para a evolução do FrameWeb com propostas de melhorias.

1.2 Método

Para o desenvolvimento deste trabalho, foram realizadas as seguintes atividades:

- Pesquisa: busca de informações sobre o método FrameWeb e os diversos *frameworks* para os quais ele foi inicialmente elaborado e o desenvolvimento de Aplicações Web na mesma. Os requisitos do SCAP foram pesquisados em projetos de graduação anteriores (DUARTE, 2014; PRADO, 2015);
- Projeto: utilização do método e dos modelos propostos pelo FrameWeb para projetar o SCAP nas plataformas desejadas;
- Desenvolvimento: implementação das funcionalidades do SCAP utilizando os *frameworks* CodeIgniter e NodeJS;
- Finalização: realização de testes finais, além de pequenos ajustes e melhorias;
- Apresentação: apresentação final do trabalho, com a entrega da monografia e dos sistemas desenvolvidos.

1.3 Organização da Monografia

Esta monografia é dividida em cinco capítulos incluindo a introdução.

No Capitulo 2 realiza-se um levantamento dos principais temas abordados ao longo deste trabalho que são: FrameWeb, *frameworks* e a Engenharia Web.

No Capítulo 3 é descrito brevemente como foi feita a especificação e análise dos requisitos do SCAP (Sistema de Controle de Afastamento de Professores), apresentando seus objetivos e funcionalidades.

No Capítulo 4 apresenta-se o resultado da aplicação do FrameWeb no projeto, podem ser vistos os modelos propostos e a implementação dos mesmos com as tecnologias utilizadas.

No Capítulo 5 tem-se as conclusões retiradas deste trabalho juntamente com as melhorias propostas para o método FrameWeb.

2 Engenharia Web e o Método FrameWeb

A Web modificou-se muito com os passar do tempo. Saiu de uma forma simples com páginas estáticas para páginas com comportamento dinâmico, oferecendo diversos serviços que estão presentes em nossa rotina. Agregou valor às nossas vidas, transformando a maneira de pensar, relacionar e viver. Nos últimos anos a Web tornou-se presente em muitas situações na vida de milhares de pessoas em todo mundo, ultrapassando muitos desenvolvimentos tecnológicos presentes na história (GINICE; MURUGESAN, 2001).

Presente em quase todo lugar, cada vez mais rápida e de fácil acesso, ela fornece serviços que se tornaram indispensáveis para nossa vida. E cada vez mais os sistemas antes desenvolvidos para um meio fechado e legados são migrados para a Web para se tornarem acessíveis. Diversos setores de negócios hoje em dia solicitam serviços que antes necessitavam de uma infraestrutura presente nas empresas, podendo hoje ser terceirizados, ex.: gerenciamento de e-mails, redes, máquinas virtuais, banco de dados e até serviço de desenvolvimento de aplicações Web (como *chatbots*). Uma vasta abrangência de novas e complexas aplicações corporativas está emergindo na Web (LI; SHI; YANG, 2000).

Tornou-se essencial a aplicação de conceitos existentes na Engenharia de Software, adaptados para essa nova plataforma, no desenvolvimento das aplicações Web. Características do ambiente Web, como concorrência, carga imprevisível, disponibilidade, sensibilidade ao conteúdo, evolução dinâmica, imediatismo, segurança e estética (PRESSMAN, 2011) deram origem a uma nova sub-área da Engenharia de Software, a Engenharia Web.

As aplicações estão se tornando cada vez mais complexas e mais presente nos dispositivos usuais, televisão, smartphones, relógios, carros, etc. Antigamente o usuário necessitava de ter a aplicação instalada localmente, hoje já é possível executá-la e ter o serviço necessário sem instalar nada, até mesmo sem ter um *hardware* específico. Chamaremos aqui tais sistemas de WIS (*Web-based Information Systems*) e ao conjunto amplo de todas as aplicações Web como WebApps (abreviação de *Web Applications*). Tais aplicações executam em um servidor conectado à World Wide Web (WWW) e podem ser acessadas de qualquer computador ou dispositivo conectado à Internet.

Neste capítulo será abordada a adaptação da Engenharia de Software para satisfazer o desenvolvimento de WISs para, em seguida, demonstrar a aplicação do método FrameWeb para as WebApps e o benefício do uso de *frameworks* para um rápido desenvolvimento.

2.1 Engenharia Web

A Engenharia Web (*Web Engineering* – WebE) é a Engenharia de Software voltada para o desenvolvimento de aplicações Web (PRESSMAN, 2005). A Engenharia Web trata de abordagens disciplinadas e sistemáticas para o desenvolvimento, implantação e manutenção de sistemas baseados na Web (DESHPANDE et al., 2001).

Como as aplicações se tornam cada vez mais complexas, necessitam de qualidade de desenvolvimento, por isso abordagens sistemáticas e disciplinadas são necessárias para manter um certo padrão de qualidade na construção da aplicação. Algumas aplicações são tão importantes para a organização que representam a fonte de renda da mesma, não podendo ficar indisponíveis como, por exemplo, lojas virtuais. Com toda essa complexidade, os problemas antes presentes em aplicações locais, também começam a aparecer nas aplicações para Web. Dificuldades como: inexperiência e formação inadequada dos desenvolvedores, falta (de uso) de métricas para estimativas, falta (de uso) de modelos de processo, métodos inadequados e obsoletos, planejamento incorreto, prazos e custos excedidos, falta de documentação, dificuldades de implementação e manutenção, WebApps mal definidas e projetadas, *layout* inadequado (comunicação visual), mudança contínua dos requisitos, da tecnologia e da arquitetura, falta de rigor no processo de desenvolvimento, dentre outros (PERUCH, 2007). Como podemos observar, a Engenharia Web está presente em todas as fases do desenvolvimento da aplicação.

Os mesmos conceitos ou atributos de qualidade (OLSINA; LAFUENTE; ROSSI, 2001) empregados na Engenharia de Software também são necessários para o desenvolvimento da aplicação:

- Usabilidade: o software deve ter uma característica inteligível, ou seja, fácil uso a todos os tipos de pessoas (inclusive deficientes), com pouco ou muito conhecimento técnico;
- Funcionalidade: o software deve se comportar bem atendendo todas as demandas para qual foi construído, buscando todas as informações corretamente e as operações funcionando;
- Eficiência: o tempo de resposta para determinada funcionalidade deve ser o mínimo possível. Atualmente também significa em como o sistema apresenta funcionalidades principais e enquanto busca as informações para os usuários. Por exemplo, ao invés de fazer o usuário esperar carregar o sistema inteiro, já carrega algumas funcionalidades mais utilizadas, permitindo-lhe ir direto ao ponto e apresentando o andamento do carregamento;
- Confiabilidade: o sistema deve se recuperar de erros, os links devem ser direcionados para os locais corretos, a validação das informações dos usuários devem ser checadas;

- Manutenibilidade: o sistema deve prover fácil manutenção para aprimorar funcionalidades e inserir novas ou retirar. Com novas atualizações, novas informações disponíveis no mercado, mudança nas regras de negócio e evolução das tecnologias, o sistema deve ser construído de forma fácil e intuitiva permitindo sua evolução ou adaptação.

Vários outros trabalhos foram e têm sido apresentados na literatura sobre como e o que deve ser avaliado em aplicações Web. Alguns desses trabalhos podem ser vistos em (BEVAN, 1998; OLSINA; ROSSI, 1999; LOWE, 1999). Para se garantir esses atributos de qualidade um processo de desenvolvimento deve ser estabelecido e, conforme já mencionado anteriormente, a Engenharia Web já utiliza de métodos propostos pela Engenharia de Software (DUARTE, 2014).

A Engenharia Web propõe uma abordagem incremental e iterativa, onde no final de cada iteração se realizam atividades de análise, projeto, implementação e testes numa parte do sistema, avaliando se o sistema corresponde às expectativas dos *stakeholders*. A primeira fase da Engenharia Web consiste na análise de requisitos e contém atividades necessárias para modelagem do problema a ser resolvido, como coleta de informações sobre todas as funcionalidades que o sistema deve prover (requisitos funcionais) e restrições de operação (requisitos não funcionais). Requisitos de um sistema são descrições dos serviços que devem ser fornecidos por esse sistema e as suas restrições operacionais (SOMMERVILLE, 2007).

São quatro os tipos de análises propostas: **de conteúdo**, que concentra informações sobre como a aplicação deve apresentar o conteúdo, **de interação**, que analisa a interação do usuário com aplicação, **funcional**, que observa como o conteúdo é afetado ou criado pela aplicação e a **de configuração**, que reúne informações da infraestrutura na qual a aplicação se encontra disponível.

Após levantadas, reunidas e organizadas as informações necessárias na modelagem de análise, a Engenharia Web propõe elaborar o modelo de projeto. Com as informações obtidas anteriormente no modelo de análise, o projeto é organizado em seis tópicos principais: projeto de conteúdo, projeto arquitetural, projeto de estética, projeto de navegação, projeto de interface e projeto de componentes (PRESSMAN, 2005). Existem alguns métodos que podem ser aplicados no desenvolvimento de aplicações Web, como WAE (CONALLEN, 2002), OOWS (FONS et al., 2003), OOHDM (SCHWABE; ROSSI, 1998) e o próprio FrameWeb (vide Seção 2.3).

Em seguida vem a fase de implementação, na qual o sistema é construído. Utiliza-se uma linguagem de programação de acordo com os requisitos dos sistema e características necessárias de operação da aplicação, como escalabilidade, tempo de resposta com servidor, organização dos dados, etc. Por último temos a fase de testes, responsável por garantir que a aplicação esteja cumprindo os requisitos levantados. Tais testes garantem conteúdo correto,

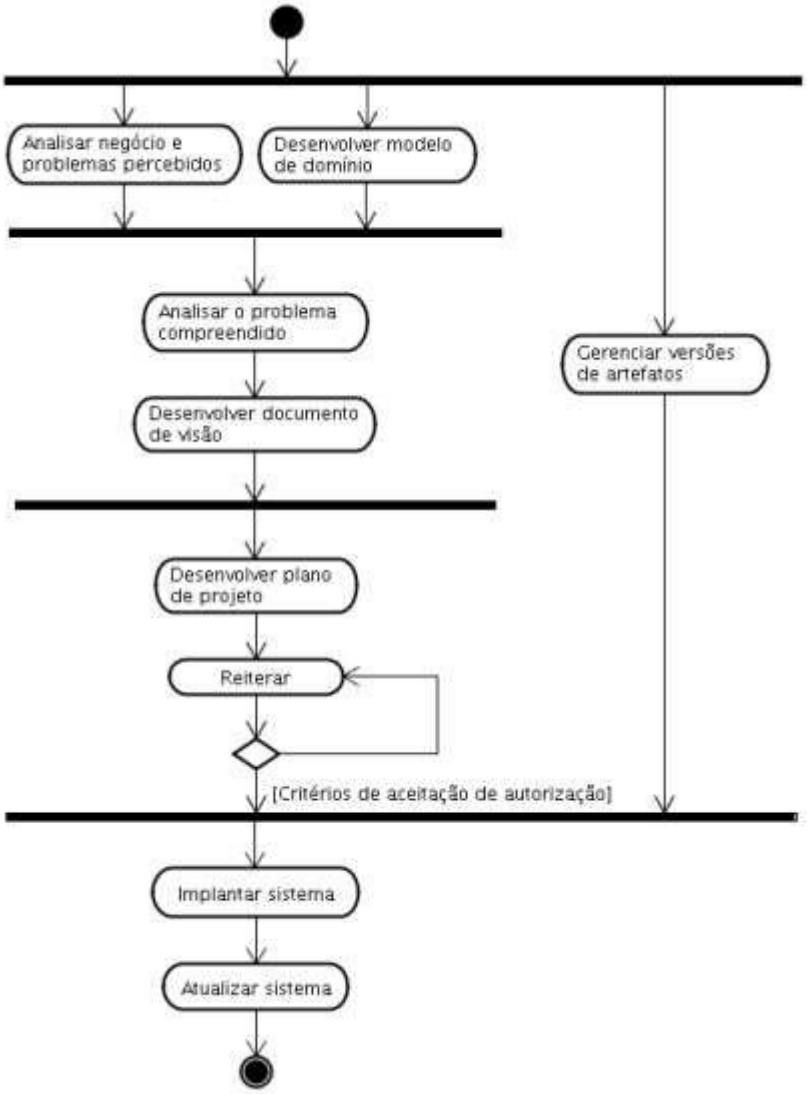

Figura 1 – Processo de desenvolvimento de Software proposto por (CONALLEN, 2002).

navegabilidade, segurança, carga, eficiência e interoperabilidade (diversos navegadores Web). Como exemplo dessa abordagem temos a metodologia proposta por Conallen (2002), que descreve um processo de desenvolvimento de software iterativo, incremental e centrado em casos de uso. As atividades do processo são mostradas na Figura 1. O sub-processo "Reiterar" inicia com a revisão e refinamento do plano da iteração. A as atividades são executadas em paralelo: levantamento de requisitos, análise, projeto da arquitetura, projeto detalhado, implementação, testes, gerência de alterações, desenvolvimento do ambiente e desenvolvimento do projeto. Após uma iteração, apresentam-se os modelos construídos aos *stakeholders*.

2.2 Frameworks

Foi observado que a maioria dos sistemas de informação desenvolvidos para a Web (WIS) possuíam uma infraestrutura muito similar. Desenvolveram-se então *frameworks* que continham generalizações do desenvolvimento desta infraestrutura.

Desta forma, um *framework* pode ser considerado como uma aplicação reutilizável e semi-completa que pode ser especializada para produzir aplicações personalizadas (HUSTED et al., 2004). O uso de *frameworks* permite a construção da aplicação sem muito esforço de codificação, pois utiliza-se seu código em conjunto com o código da aplicação, criando um código necessário para atender tal funcionalidade.

Um exemplo é a autenticação de usuário, presente em várias aplicações. Alguns *frameworks* já disponibilizam esse código de forma independente das demais partes do sistema, permitindo assim a personalização para a aplicação em construção. Souza (2007) organiza a maioria dos *frameworks* de WebApps em seis categorias diferentes:

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

Nas próximas subseções são descritas as categorias de *frameworks* MVC (Controlador Frontal), Mapeamento Objeto/Relacional e de Injeção de Dependência. Em seguida, apresentamos os *frameworks* utilizados neste trabalho, a saber: CodeIgniter e NodeJS.

2.2.1 Frameworks MVC

Segundo Souza (2007), o MVC (*Model-View-Controller*) ou Modelo-Visão-Controlador é um arquitetura baseada em camadas desenvolvida pelo Centro de Pesquisas da Xerox de Palo Alto (Xerox PARC) para a linguagem Smalltalk em 1979 (REENSKAUG, 1979).

Com objetivo de favorecer a manutenibilidade da aplicação, o *framework* MVC faz uma separação distinta entre Dados (*Model*) e o *Layout* (*View*). Assim, em caso de mudanças necessárias no *layout* da aplicação, estas não impliquem em mudanças na camada de dados. As funções de busca e recuperação de dados continuam preservadas. Entre essas duas camadas situa-se a camada de Controle (*Controller*), que é responsável pela comunicação entre elas. Nela está presente a lógica de acesso aos dados, de negócio, de

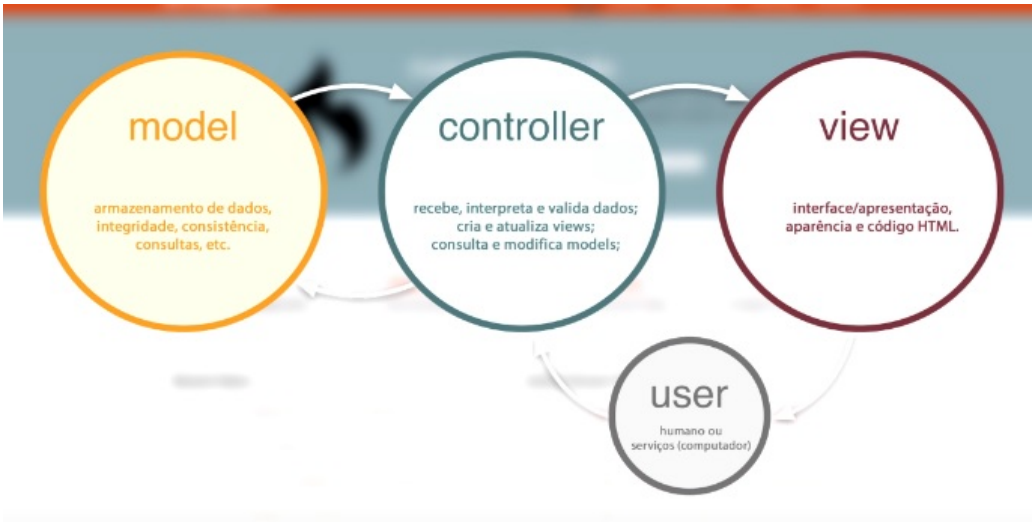

Figura 2 – Diagrama representando a arquitetura MVC. Retirado do vídeo do Curso de CodeIgniter para iniciantes da RBTech.

apresentação e de interação com o usuário. A Figura 2 exemplifica a relação entre *Model*, *View*, *Controller* e usuários.

Vamos supor uma aplicação financeira que realiza cálculo de juros. É fornecida uma interface gráfica com campos para inserir valores, escolher o tipo de cálculo (juros simples ou composto) e um botão para envio dos valores para execução do cálculo. O que se vê na interface gráfica foi codificado na camada *View*.

Ao clicar no botão há uma requisição com os valores digitados no formulário e o tipo de cálculo selecionado. O evento do botão é como um pedido a um intermediador que prepara as informações para então enviá-las para o cálculo. Este intermediador nós chamamos de *Controller*. O controlador é o único no sistema que conhece o responsável pela execução do cálculo, neste caso a camada que contém as regras de negócios. Esta operação matemática será realizada pelo *Model* assim que ele receber um pedido do *Controller*.

O *Model* realiza a operação matemática e retorna o valor calculado para o *Controller*, que também é o único que possui conhecimento da existência da camada de visualização. Tendo o valor "em mãos", o intermediador o repassa para a interface gráfica que exibirá para o usuário. Caso esta operação deva ser registrada em uma base de dados, o *Model* se encarrega também desta tarefa.

Porém, na plataforma Web a arquitetura MVC precisa ser levemente adaptada, visto que o *Model*, situado no servidor Web, não pode notificar a *View* sobre alterações, já que esta encontra-se no navegador do lado do cliente e a comunicação é sempre iniciada pelo cliente. Portanto o nome correto para esse padrão arquitetônico, quando aplicado à Web, seria "Controlador Frontal" (*Front Controller*) (CRUPI; ALUR; MALKS, 2003; SOUZA, 2007). A Figura 3 representa o funcionamento de um Controlador Frontal na Web.

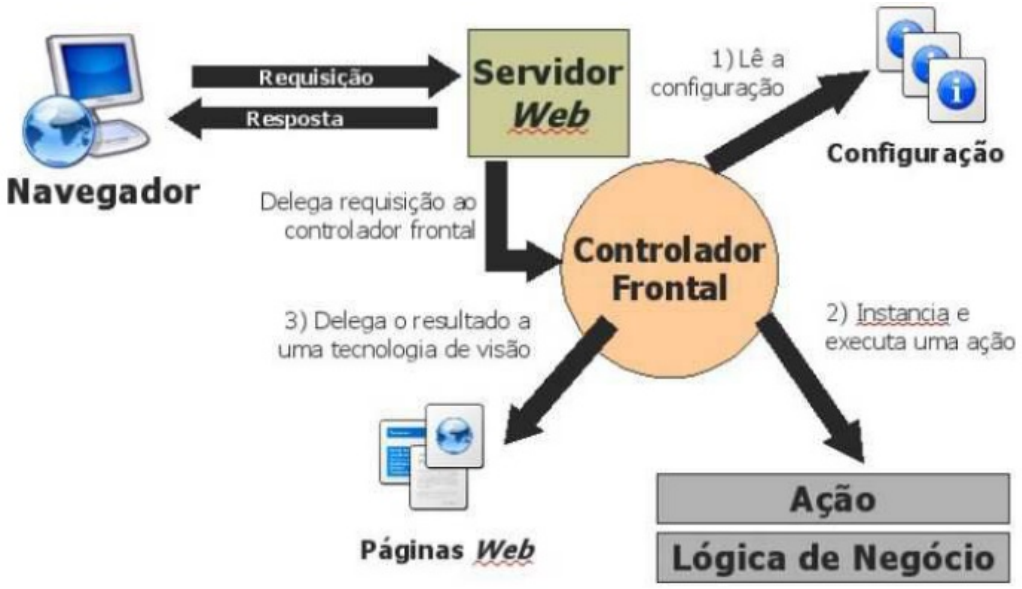

Figura 3 – Representação de um Controlador Frontal na Web (SOUZA, 2007).

2.2.2 Frameworks de Mapeamento Objeto/Relacional

Atualmente a maioria das aplicações usa o modelo de banco de dados relacional, porém grande parte da implementação é orientada a objetos, por ser um paradigma que permite melhor manutenibilidade e reuso do código. Temos então uma "incompatibilidade de paradigmas" (*paradigm mismatch*) (SOUZA, 2007).

Isso se dá porque "operações de SQL [linguagem estruturada de consultas para SGBDRs] como projeção sempre unem resultados em uma representação em tabelas de dados resultantes. Isso é bem diferente do grafo de objetos interconectados usados para executar uma lógica de negócio em um aplicativo Java!" (BAUER; KING, 2005).

Surge então, na década de 80, o Mapeamento Objeto/Relacional (ORM – *Object Relational Mapping*), que consiste na representação das tabelas por meio de classes e os registros de cada tabela são representados como instâncias das classes correspondentes. Dessa forma o programador não precisa se preocupar com os comandos de manipulação dos dados, pois estará presente uma interface que dispõe de métodos para manipulação dos dados. É preciso apenas que o programador informe ao *framework* utilizado como mapear esses objetos em forma de tabelas. Sendo assim, o programador não precisa realizar o mapeamento de forma manual, o que tomaria muito tempo e poderia gerar inconsistências no banco de dados.

2.2.3 Frameworks de Injeção de Dependências

O objetivo desses *frameworks* é prover baixo acoplamento entre as classes, tirando a responsabilidade de algumas classes criarem instâncias de classes das quais elas dependem para realizarem suas funções. O baixo acoplamento melhora a manutenibilidade do

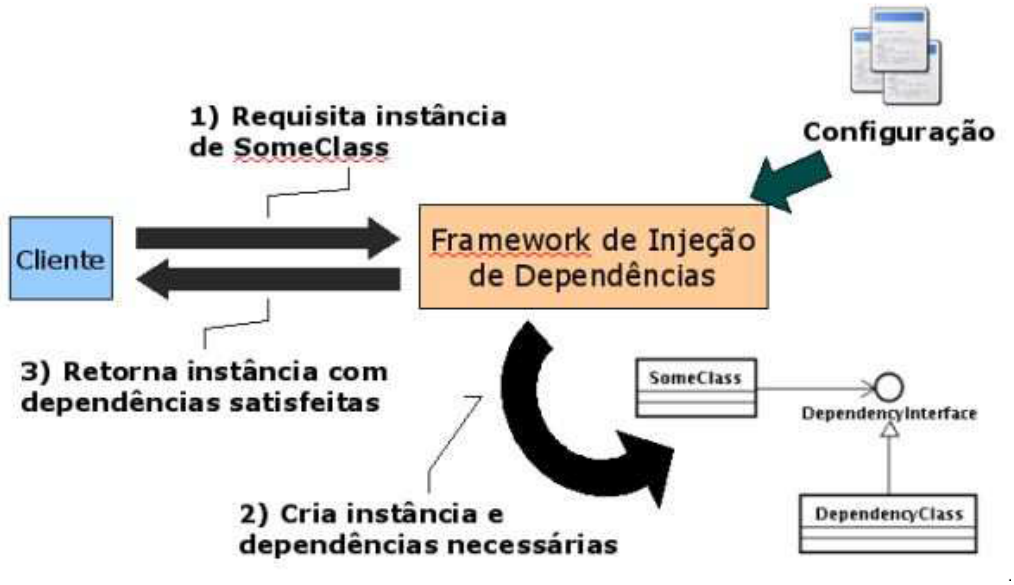

Figura 4 – Funcionamento de um framework de Injeção de Dependências (SOUZA, 2007).

código. O conceito de Injeção de Dependências (DI - *Dependency Injection*) diz que a responsabilidade de configurar dependências entre classes é assumida por um terceiro (no caso, o *framework*). A Figura 4 representa o funcionamento de um *framework* de Injeção de Dependências.

2.2.4 CodeIgniter

O CodeIgniter é um *framework* de desenvolvimento ágil de aplicações em PHP. Por padrão a organização arquitetural é MVC. Seu objetivo principal é possibilitar que o desenvolvedor desenvolva projetos mais rapidamente do que se estivesse codificando do zero. Ele contém algumas funções nativas que ajudam o desenvolvimento e legibilidade do código, por exemplo:

- *Helpers*: são como grupo de funções que podem ser utilizadas e chamadas em qualquer parte do código. Você também pode criar um *helper* e ele fica disponível para ser chamado em qualquer parte do sistema;
- Funções de banco de dados: *queries* relacionadas a CRUD (acrônimo para *Create*, *Retrieve*, *Update* e *Delete*, operações básicas de cadastro), podem ser feitas por comandos nativos. O CodeIgniter fornece comandos como: *insert*, *update*, *delete*. Essas funções são ótimas para o desenvolvimento do Model;
- Estrutura fácil: o *framework* apresenta uma boa organização de pastas e arquivos. Por padrão já existem pastas chamadas *Models*, *Views* e *Controllers*. Há uma pasta separada só para comunicação com o banco de dados. Esta estrutura pode ser observada na Figura 5.

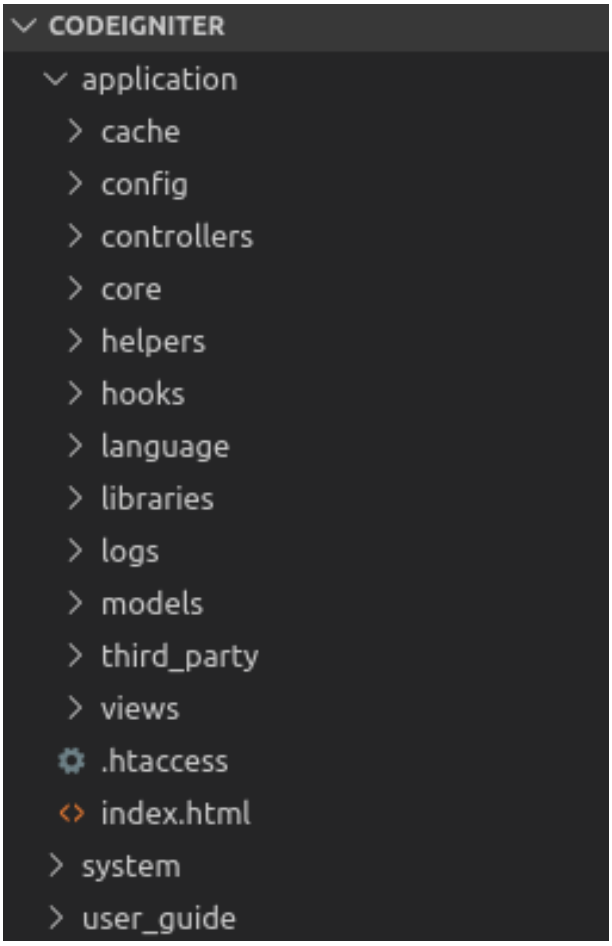

Figura 5 – Estrutura básica do CodeIgniter.

2.2.5 NodeJS

O NodeJS é um interpretador assíncrono de código JavaScript construído a partir do motor do navegador Chrome. É conhecido por ser rápido, pois foi construído focado em migrar a programação do JavaScript do cliente (*front-end*) para os servidores, criando aplicações de alta escalabilidade (como um servidor Web), manipulando milhares de conexões/eventos simultâneas em tempo real numa única máquina física. Logo a linguagem utilizada para *front-end* e *back-end* se torna a mesma, o JavaScript.

A instalação do NodeJS vem acompanhada do NPM. O NPM é um gerenciador de pacotes para adição de novas funcionalidades e reúso de código.

Neste trabalho utilizou-se o EJS que é uma *view engine* para escrever código JavaScript e o resultado é um código estático HTML. Assim, a aplicação se torna mais dinâmica.

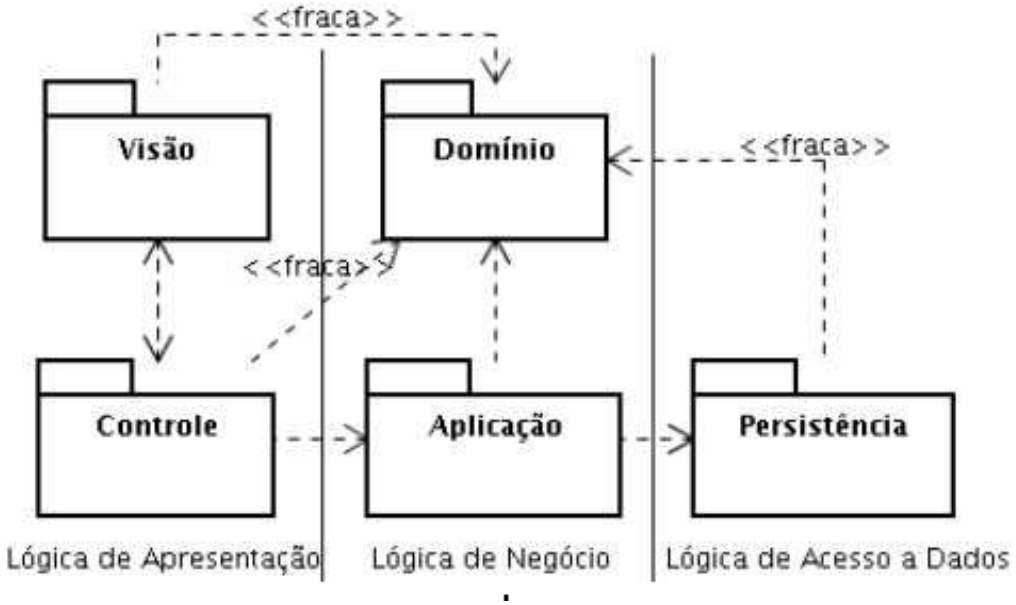

Figura 6 – Arquitetura padrão para WIS baseada no padrão arquitetônico Service Layer (FOWLER et al., 2002).

2.3 O Método FrameWeb

FrameWeb (SOUZA, 2007) é um método, baseado em *frameworks*, para o desenvolvimento de sistemas de informação Web (*Web Information Systems* – WISs). Mesmo existindo vários *frameworks* para o desenvolvimento de aplicações, não havia nada que apresentasse e reunisse as características desses *frameworks* para facilitar o desenvolvimento das aplicações na Engenharia Web (WebE).

Atualmente, o método FrameWeb concentra suas propostas na fase de projeto arquitetural, deixando desenvolvedores e organizações livres para adotar as técnicas que considerarem mais apropriadas para as demais fases do processo. O método propõe que a arquitetura de um WIS seja dividida em três camadas, podendo ser observado na Figura 6:

- Lógica de Apresentação: provê as interfaces gráficas para a comunicação da WebApp com o usuário, apresentando o conteúdo das classes de domínio, permitindo ao usuário fazer requisições e visualizar respostas. Esta camada contém os pacotes de Visão e Controle:
  - Visão: contém as páginas Web, imagens e scripts que são executados do lado do cliente e outros arquivos que relacionados à exibição de informações para o usuário. Os elementos presentes no pacote de Visão enviam estímulos causados pelo usuário (clique de um botão, preenchimento de um campo texto, acionamento de uma caixa de seleção) e esses estímulos devem ser devidamente tratados e respondidos pelas classes presentes no pacote Controle;
  - Controle: contém classes controladoras, que são responsáveis por tratar a

interação do usuário com os elementos presentes no pacote de Visão, tendo assim uma relação mútua. O pacote Controle também possui uma interdependência com o pacote Aplicação, que provê ao usuário acesso às principais funcionalidades do sistema.

- Lógica de Negócio: provê as regras de negócios e funcionalidades relacionadas. Contém os pacotes de Domínio e Aplicação:
  - Domínio: contém classes identificadas na fase de análise como sendo conceitos do domínio do problema;
  - Aplicação: contém a implementação dos casos de uso identificados na fase de análise de requisitos, provendo uma camada de serviços que deve ser independente da interface com o usuário. O pacote Aplicação possui duas dependências, uma com o pacote de Domínio, pois manipula objetos desse pacote e a outra com o pacote de Persistência, para que se possa recuperar, gravar, alterar e excluir objetos de domínio de acordo com a execução dos casos de uso.
- Lógica de Acesso a Dados: camada que provê acesso aos dados para o domínio da aplicação. Ela possui apenas o pacote de Persistência:
  - Persistência: contém classes que gravam as informações de domínio que precisam ser persistidas pelo sistema. O FrameWeb recomenda o uso do padrão de projeto *Data Access Object* (DAO) (ALUR; MALKS; CRUPI, 2003), que separa a tecnologia de persistência de dados da Lógica de Negócio, criando uma camada de abstração, tornando a aplicação independente do *framework* ORM (PRADO, 2015).

O FrameWeb segue a mesma abordagem de linguagens de modelagem, como WAE (CONALLEN, 2002) e UWE (KOCH et al., 2000), estendendo o metamodelo UML (MARTINS, 2016) para representar componentes mais utilizados da Web e os componentes relacionados aos *frameworks*. Possui quatro tipos de diagramas, com objetivo de facilitar a implementação das camadas explicadas na seção anterior (SOUZA, 2007; MARTINS, 2016):

- Modelo de Entidades: baseado no diagrama de classes geralmente criado na fase de análise, especifica os tipos de dados e o mapeamento das classes de domínio que serão persistidas;
- Modelo de Persistência: ajuda na implementação das classes DAO do sistema, mostrando todas as interfaces e suas implementações juntamente com os métodos de cada uma;

- Modelo de Navegação: é um diagrama de classes que representa as interações das páginas Web. Demonstra o funcionamento da camada Lógica de Apresentação;
- Modelo de Aplicação: representa as classes do pacote Aplicação e demonstra suas dependências.

Apenas os modelos de Navegação e Persistência estão presentes no desenvolvimento do SCAP realizado neste trabalho, conforme será apresentado no Capítulo 4. No capítulo seguinte, apresentamos os requisitos do sistema.

3 Especificação de Requisitos

A Engenharia de Requisitos tem com objetivo não somente levantar funcionalidades necessárias a aplicação, mas em entender o negócio, com objetivo final de entregar um produto que atenda às expectativas dos *stakeholders*.

Neste capítulo é feita uma descrição de escopo do SCAP (Sistema de Controle de Afastamentos de Professores), depois tem-se os modelos de casos de uso que foram observados a partir dos requisitos previamente levantados por Duarte (2014) e Prado (2015).

3.1 Descrição do Escopo

No SCAP devem estar todas as solicitações de afastamentos dos professores do Departamento de Informática (DI). As solicitações podem ser para eventos no Brasil ou no exterior. Essa solicitações precisam ser avaliadas pelos professores do DI e, em alguns casos, pelo Conselho Departamental do Centro Tecnológico (CT) e pela Pró-reitoria de Pesquisa e Pós-Graduação (PRPPG) para que sejam aprovadas e o professor possa realizar a viagem.

Os pedidos para afastamento no Brasil, são aprovados pela Câmara Departamental (formada pelos funcionários do departamento e representantes discentes). O pedido é feito por meio da lista de e-mails dos funcionários do DI e endereçado ao chefe do departamento, cargo este que é exercido por algum professor do DI durante mandato temporário. Caso ninguém se manifeste contrário à solicitação, o pedido é aprovado. Destacando que para viagem no Brasil, o procedimento é inteiramente realizado dentro do DI.

No caso de viagem para o exterior é escolhido um professor (que não tenha parentesco com o solicitante) para ser o relator do pedido. Após o parecer do relator, o pedido passa pela aprovação do DI, igual no caso acima. No entanto, ainda é necessário que o CT e a PRPPG aprovem o pedido e que o afastamento seja publicado no Diário Oficial da União, conforme art. 95 da lei 8.112 (<http://www.planalto.gov.br/ccivil_03/leis/l8112cons.htm>).

O SCAP só trata das tramitações que são realizadas dentro do DI. Ele não tem nenhuma comunicação com os sistemas do CT e da PRPPG, a forma como eles tramitam os processos não faz parte do escopo do sistema. Solicitações para viagem no exterior fazem parte do escopo enquanto se encontram dentro do DI.

O SCAP foi criado para agilizar e facilitar a solicitação de afastamento para o professor. O sistema atinge o objetivo enviando e-mails automáticos aos envolvidos e do

Tabela 1 – Atores do SCAP.

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

uso de formulários para ajudar na criação dos documentos necessários para se realizar uma solicitação. O sistema também permite que demais professores e secretários consultem dados sobre as solicitações.

3.2 Modelo de Casos de Uso

O modelo de casos de uso visa capturar e descrever as funcionalidades que um sistema deve prover para os atores que interagem com o mesmo, isto é, os usuários do sistema (SILVA, 2007). Alguns casos de uso podem ser mais críticos que outros para a lógica de negócio do sistema. Após a definição do escopo, o processo de Engenharia de Requisitos seguiu com a modelagem de atores, demonstrado na Tabela 1.

O Secretário lida com a parte cadastral do sistema. Cadastro de professores e seus parentes. É de sua responsabilidade atualizar o Chefe do Departamento quando um novo é eleito. Casos de Uso como Registrar Parecer CT e Registrar Parecer PRPPG também são sua responsabilidade. Ele também é responsável por Arquivar uma Solicitação de Afastamento quando a mesma já foi regularizada.

O Professor cadastra sua Solicitação. Pode se manifestar contra a Solicitação de outro Professor e pode ser o Relator de uma Solicitação. O Chefe de Departamento pode executar todos Casos de Uso de um Professor e ainda encaminhar uma Solicitação ao seu Relator.

O SCAP foi dividido em 2 subsistemas: Núcleo e Secretaria. O primeiro contempla os casos de usos dos professores e do chefe de departamento, enquanto o segundo envolve os casos de uso dos secretários.

As figuras 7 e 8 mostram os diagramas de casos de uso destes subsistemas. Nos parágrafos que se seguem, apresentamos uma descrição sucinta dos casos de uso levantados para o SCAP. Uma versão mais detalhada dessa descrição pode ser vista em (DUARTE, 2014; PRADO, 2015).

Em **Solicitar Afastamento**, um professor cadastra um pedido de afastamento no sistema, informando todos os dados necessários para a tramitação do mesmo. Em **Cancelar Afastamento** o professor cancela um pedido de afastamento e o seu status é alterado para cancelado, caso ele seja o solicitante.

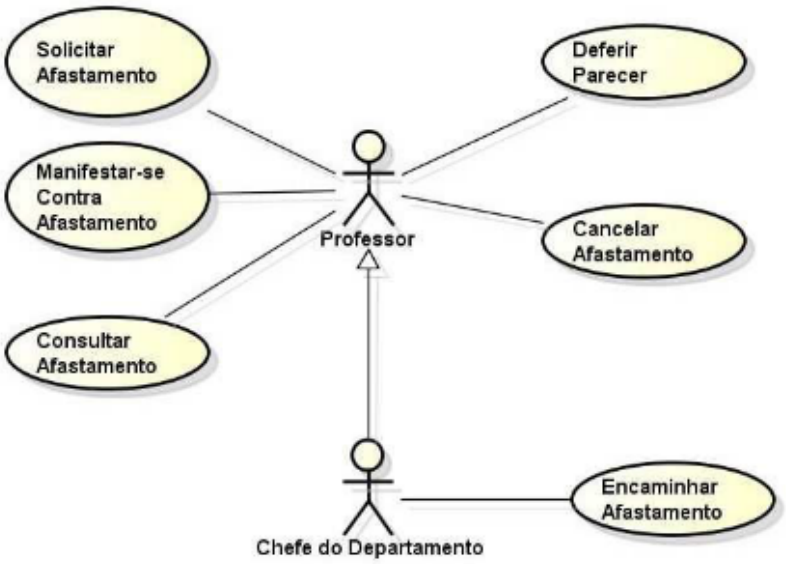

Figura 7 – Diagrama de Casos de Uso do subsistema Núcleo (PRADO, 2015).

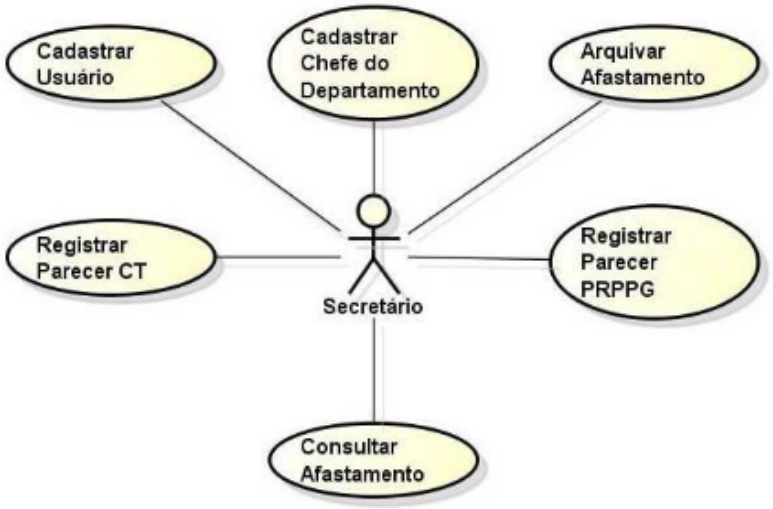

Figura 8 – Diagrama de Casos de Uso do subsistema Secretaria (PRADO, 2015).

O caso de uso **Encaminhar Afastamento** é realizado pelo Chefe do Departamento quando o mesmo analisa um pedido de afastamento internacional recém cadastrado e aponta um professor como relator deste afastamento. Já **Deferir Parecer** é utilizado quando um professor, que foi cadastrado pelo Chefe do Departamento como relator de um afastamento internacional, cadastra seu parecer sobre o afastamento.

Consultar Afastamento, como o nome já diz, consiste em uma busca que um professor realiza para ver os dados de um pedido de afastamento. Em **Manifestar-se Contra Afastamento** um professor que é contrário a um afastamento cadastra o motivo de sua opinião contrária, assim uma reunião é marcada e os professores decidem se o afastamento é aprovado ou não.

Em **Cadastrar Usuário** um secretário cadastra um novo professor ou secretário

no sistema, informando os dados pessoais necessários. **Em Cadastrar Chefe do Departamento** um secretário cadastra o mandato do Chefe de Departamento, informando as datas de início e fim do mesmo.

O casos de uso **Registrar Parecer CT** e **Registrar Parecer PRPPG** acontecem quando um secretário cadastra o parecer do Centro Tecnológico e da Pró-Reitoria de Pesquisa e Pós-Graduação, respectivamente, sobre um pedido de afastamento internacional.

Arquivar Afastamento acontece no final da tramitação de um pedido de afastamento quando o secretário muda o status do mesmo para "Arquivado".

3.3 Análise do SCAP

A atividade de Análise dentro do paradigma da Orientação a Objetos tem por objetivo identificar objetos do mundo real, características destes objetos e relacionamentos entre eles que são relevantes para o problema a ser resolvido, especificando e modelando o problema de forma que seja possível criar um projeto orientado a objetos efetivo (PRESSMAN, 2005). Durante a Análise do SCAP, os conceitos do domínio do problema foram descritos como classes, com seus respectivos atributos, interações e relações com outras classes.

A Modelagem de Classes segue o modelo levantado previamente por Prado (2015) ao fazer a modelagem para implementação do SCAP com o *framework* VRaptor, revisando a modelagem feita anteriormente por Duarte (2014).

Apesar do SCAP ter sido dividido em dois subsistemas, as classes de domínio do problema, representadas no diagrama de classes da Figura 9, todas pertencem ao subsistema Núcleo. Isso se dá porque a divisão em subsistemas ocorreu com intuito de facilitar a implementação, e não devido a natureza das classes criadas (PRADO, 2015).

O Atributo "situação da solicitação" da classe Afastamento representa em qual estágio da tramitação um pedido de afastamento está, os status possíveis serão discutidos mais adiante. Já o atributo "tipo do afastamento" indica se o afastamento é necessário para atender a um evento localizado no Brasil ou no exterior.

As classes **Professor** e **Secretário** representam os professores de secretários do Departamento de Informática da UFES, respectivamente, ambas herdando os atributos da classe **Pessoa**. Professores podem possuir ou não uma relação de **Parentesco** com os demais professores. O chefe do departamento é representado por um **Professor** que possui um **Mandato** vigente.

Um **Afastamento** possui todas as informações necessárias para tramitação de um pedido de afastamento de um professor. A classe **Afastamento** pode ou não possuir mais de um **Documento** e requer um **Relator** somente quando o afastamento é devido a um

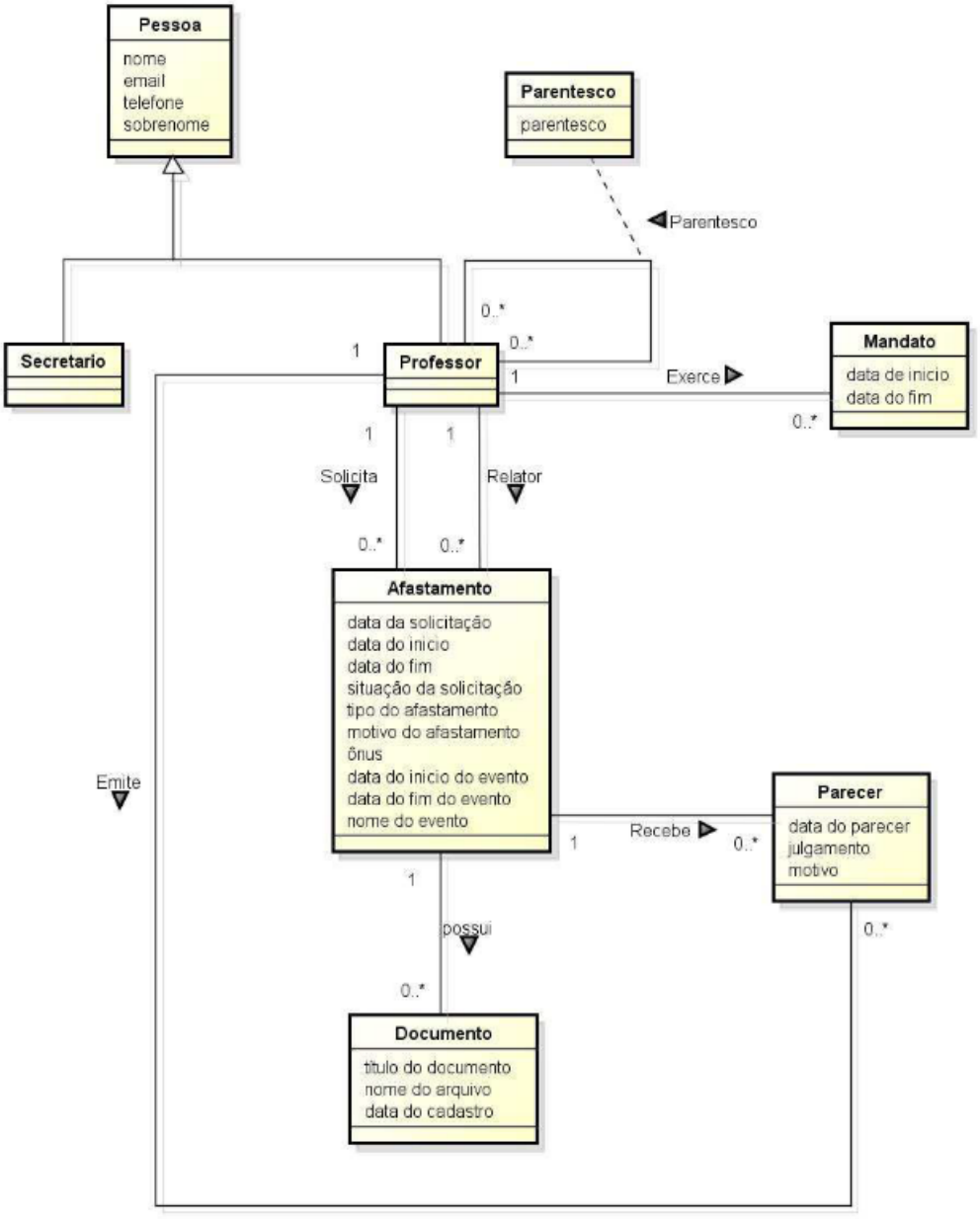

Figura 9 – Diagrama de Classes do SCAP (PRADO, 2015).

evento internacional.

Um **Parecer** é emitido por um professor e é referente a um único afastamento, porém um professor pode criar vários pareceres para afastamentos diferentes, assim como um professor pode ser relator de vários afastamentos.

Restrições de Integridade complementam as informações de um modelo deste tipo e capturam restrições relativas a relacionamentos entre elementos de um modelo que normalmente não são passíveis de serem capturadas pelas notações gráficas utilizadas na elaboração de modelos conceituais estruturais. Tais regras devem ser documentadas junto ao modelo conceitual estrutural do sistema (FALBO, 2011).

Listaremos aqui as restrições que foram observadas na primeira implementação do SCAP em (PRADO, 2015), juntamente com novas restrições referentes a funcionalidades que não foram contempladas no primeira implementação do SCAP por Duarte (2014):

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

4 Projeto Arquitetural e Implementação

Na fase de projeto deve-se produzir uma solução para o problema que foi identificado e modelado nas fases de levantamento e análise de requisitos, incorporando a tecnologia aos requisitos e projetando o que será construído na implementação. Sendo assim, é necessário conhecer a tecnologia disponível e os ambientes de hardware e software onde o sistema será desenvolvido e implantado. Durante o projeto, deve-se decidir como o problema será resolvido, começando em um alto nível de abstração, próximo da análise, e progredindo sucessivamente para níveis mais detalhados até se chegar a um nível de abstração próximo da implementação (FALBO, 2011).

Enquanto na fase de análise a tecnologia é imaginada como perfeita (capacidade ilimitada de armazenamento, custo zero e não passível de falha), a fase de projeto envolve a modelagem de como o sistema será implementado com a adição dos requisitos tecnológicos e de caráter não funcional (PRESSMAN, 2011).

O SCAP foi implementado múltiplas vezes utilizando-se FrameWeb na fase de projeto. Os *frameworks* utilizados em três versões do SCAP já implementadas em trabalhos anteriores podem ser vistos na Tabela 2. Este capítulo contém os dados técnicos das ferramentas utilizadas para a implementação SCAP e descreve a implementação com o *framework* MVC Codeigniter e MVC aplicado ao NodeJS.

O objetivo principal deste trabalho era a aplicação do método FrameWeb com diferentes *frameworks*, e não a entrega da ferramenta SCAP em si, logo alguns requisitos não foram contemplados na implementação. Em resumo foram implementadas funções respeitando as restrições de integridade para: Cadastro de Professor, Cadastro de Parentesco, Cadastro de Afastamento, Cadastro de Solicitações e Cadastro de Relatores. Ambas as implementações possuem a mesma decoração usando Bootstrap.

4.1 Comparação com Trabalhos Anteriores

Como pode ser observado, a Tabela 2 faz uma breve comparação deste trabalho com alguns trabalhos anteriores.

Nesse trabalho, para o funcionamento do CodeIgniter teve de ser instalado o servidor Apache. Para a implementação do NodeJS não é necessário a instalação de nenhum servidor, pois o próprio já é um servidor que, quando iniciado, começa a interpretar código JavaScript. Já Avelar (2018) usou Wildfly, servidor HTTP e Servlet Container de aplicações, implementado sobre a plataforma Java EE.

Diferentemente de Pinheiro (2017), não há uma aplicação para o lado do servidor

Tabela 2 – Tabela com os *frameworks* utilizados nas versões do SCAP feitas por Duarte (2014), Prado (2015) e Avelar (2018).

Categorias de *frameworks*	**Duarte (2014)**	**Prado (2015)**	**Avelar (2018)**	**Este trabalho**
Decorador	PrimeFaces	Nenhum	FreeMarker	Bootstrap
Controlador Frontal	JSF	VRaptor	Ninja	CodeIgniter e NodeJS
Injeção de Dependências	CDI	CDI	Google Guice	Não há
Mapeamento Objeto/Relacional	Hibernate	Hibernate	Hibernate	Não há

e outra para o lado do cliente. No subsistema referente ao lado do servidor, que foi desenvolvido para ser uma API, é usado o Laravel, que por sua vez é feito em cima do PHP. O *framework* ORM (*Object/Relational Mapping*) usado foi o Eloquent. Já do lado do cliente é usado como base o *framework* Quasar, que utiliza o Vue.js, que é feito em cima do JavaScript (PINHEIRO, 2017).

O motivo da criação de duas aplicações separadas é a disponibilização de uma aplicação rica para o usuário final, com todas as características de uma RIA (*Rich Internet Applications*) como descrito em (CASTELEYN et al., 2009). Para isso essa aplicação foi desenvolvida para ser um SPA (*Single-Page Applications*) e PWA (*Progressive Web App*) (PINHEIRO, 2017).

4.2 Arquitetura do Sistema

A definição da arquitetura do sistema inicia a fase de projeto de sistema, nela são mapeados os conceitos observados na fase de análise para a plataforma tecnológica de desenvolvimento. O nível de abstração da fase anterior é diminuído, pois busca-se alcançar maior proximidade com o nível da linguagem de programação (PERUCH, 2007).

No CodeIgniter, por padrão, a implementação de qualquer sistema é na estrutura MVC. No JavaScript foi seguida a estrutura MVC, mas não é por padrão da linguagem. O interpretador NodeJS interpreta o código em qualquer nível arquitetural, ficando a responsabilidade para o desenvolvedor. Em suma, ambas as implementações possuem a mesma arquitetura. A Figura 5, apresentada anteriormente no Capítulo 2, representa a estrutura padrão do CodeIgniter. A Figura 10 mostra a estrutura em que o MVC foi organizado na implementação com NodeJS.

Ambas as implementações do SCAP utilizam o mesmo banco de dados. O Sistema Geranciador de Banco de Dados utilizados foi o MySQL. Foram implementados dois *Models*:

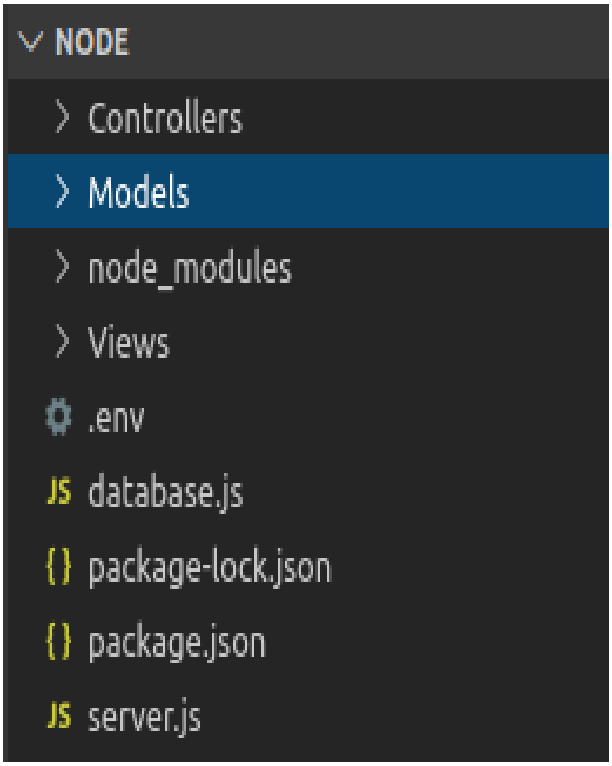

Figura 10 – Estrutura do NodeJS

- *Model* Professor;
- *Model* Afastamento.

O *Model* Professor contém todas as funcionalidades relativas ao CRUD de um Professor e seu Parentesco, fazendo a comunicação com o banco de dados. O *Model* Afastamento contém todas as funcionalidades relativas a CRUD de um Afastamento, seu Solicitante e seu Relator, fazendo também a comunicação com o banco de dados.

Foram implementados dois *Controllers*:

- *Controller* Professor;
- *Controller* Afastamento.

O *Controller* Professor é responsável por receber as requisições e responder à *View* Cadastro Professor e à *View* Editar Cadastro Professor. O *Controller* Afastamento é responsável por receber as requisições e responder à *View* Cadastro Afastamento e à *View* Editar Cadastro Afasmento.

Foram implementadas quatro *Views*:

- Cadastro Professor;
- Editar Cadastro de Professor;

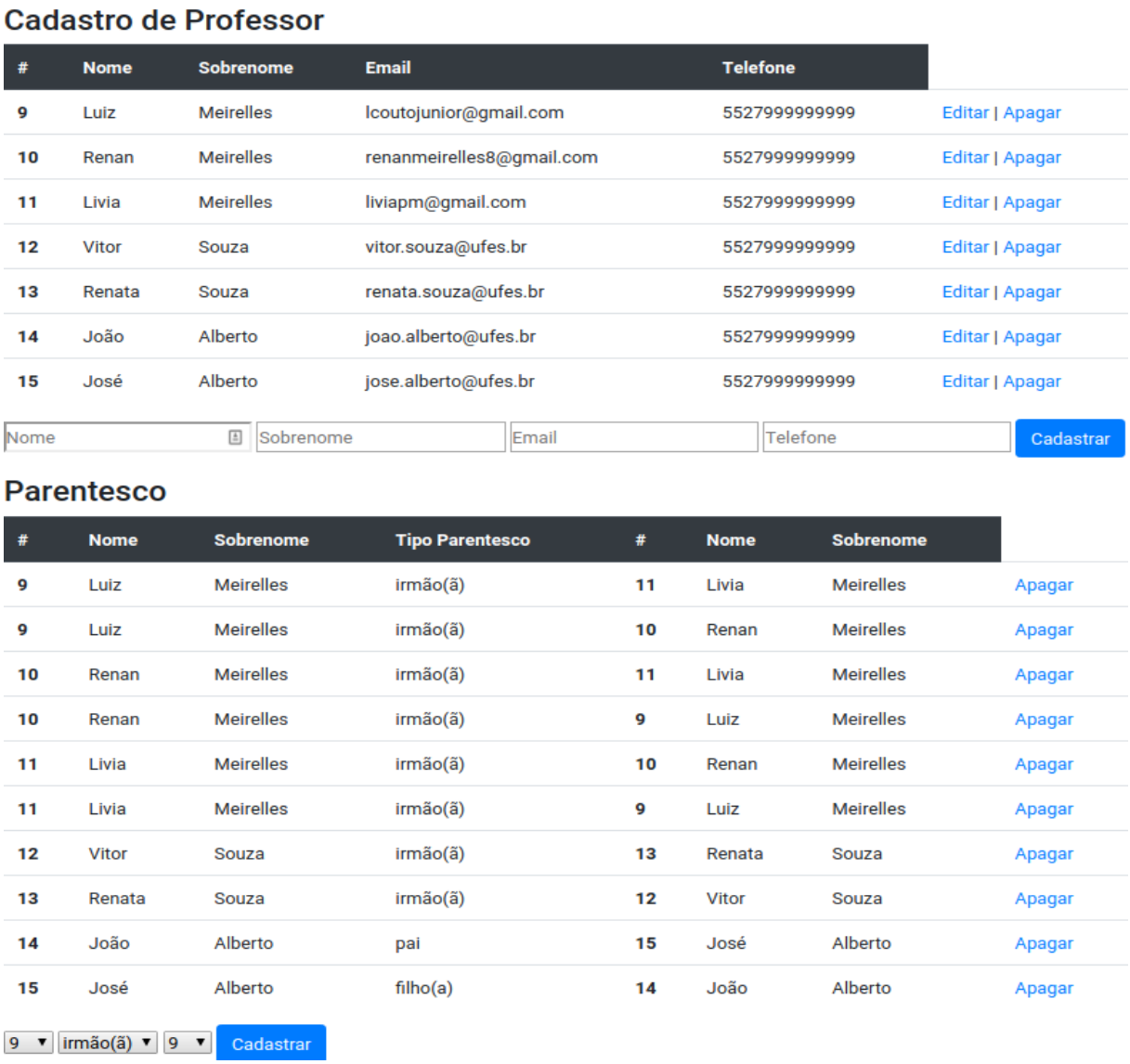

Figura 11 – View Cadastro Professor

- Cadastro Afastamento;
- Editar Cadastro Afastamento.

A *View* Cadastro Professor contém todas funcionalidades relativas ao Cadastro de um Professor, juntamente com cadastro de seu parentesco, sendo possível adicionar ou remover parentesco. A implementação não deixa um professor ser parente de si mesmo. Esta tela pode ser observada na Figura 11.

A *View* Editar Cadastro de Professor permite editar dados do próprio professor, como nome, sobrenome, e-mail e telefone. Pode ser observada na Figura 12.

A *View* Cadastro Afastamento contém todas funcionalidades relativas ao cadastro de um Afastamento, juntamente com o cadastro de uma Solicitação e de um Relator sendo possível adicionar ou remover uma solicitação e um relator. A implementação não deixa

Figura 12 – View Editar Cadastro Professor

um Afastamento ter a data de início menor que a final, um professor não pode ser relator do seu próprio afastamento, um afastamento só pode ser solicitado por um professor e um relator não pode ser parente do solicitante. Pode ser observada na Figura 13.

Por fim, a *View* Editar Cadastro Afastamento, permite editar dados daquele afastamento, como data da solicitação, data do início do afastamento, data do fim do afastamento, tipo de afastamento, motivo, ônus, data de início do evento, data de fim do evento, nome do evento. Esta interface pode ser observada na Figura 14.

Nesse trabalho, nenhum tratamento de dados foi feito na *View* mas sim nos *Controllers*. Por exemplo, ao preencher um formulário, caso a informação inserida esteja incorreta, automaticamente é o *Controller* que gerencia.

4.3 Modelos FrameWeb

Nas seções a seguir são apresentados os modelos FrameWeb construídos durante o projeto arquitetural do SCAP.

4.3.1 Modelo de Domínio

O modelo de domínio 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. A partir dele são implementadas as classes da camada de Domínio na atividade de implementação (SOUZA, 2007).

Nas duas implementações não há classes de domínio, portanto não há modelos. Isso se tornou uma escolha ao longo do projeto, pois o objetivo se tornou explorar os

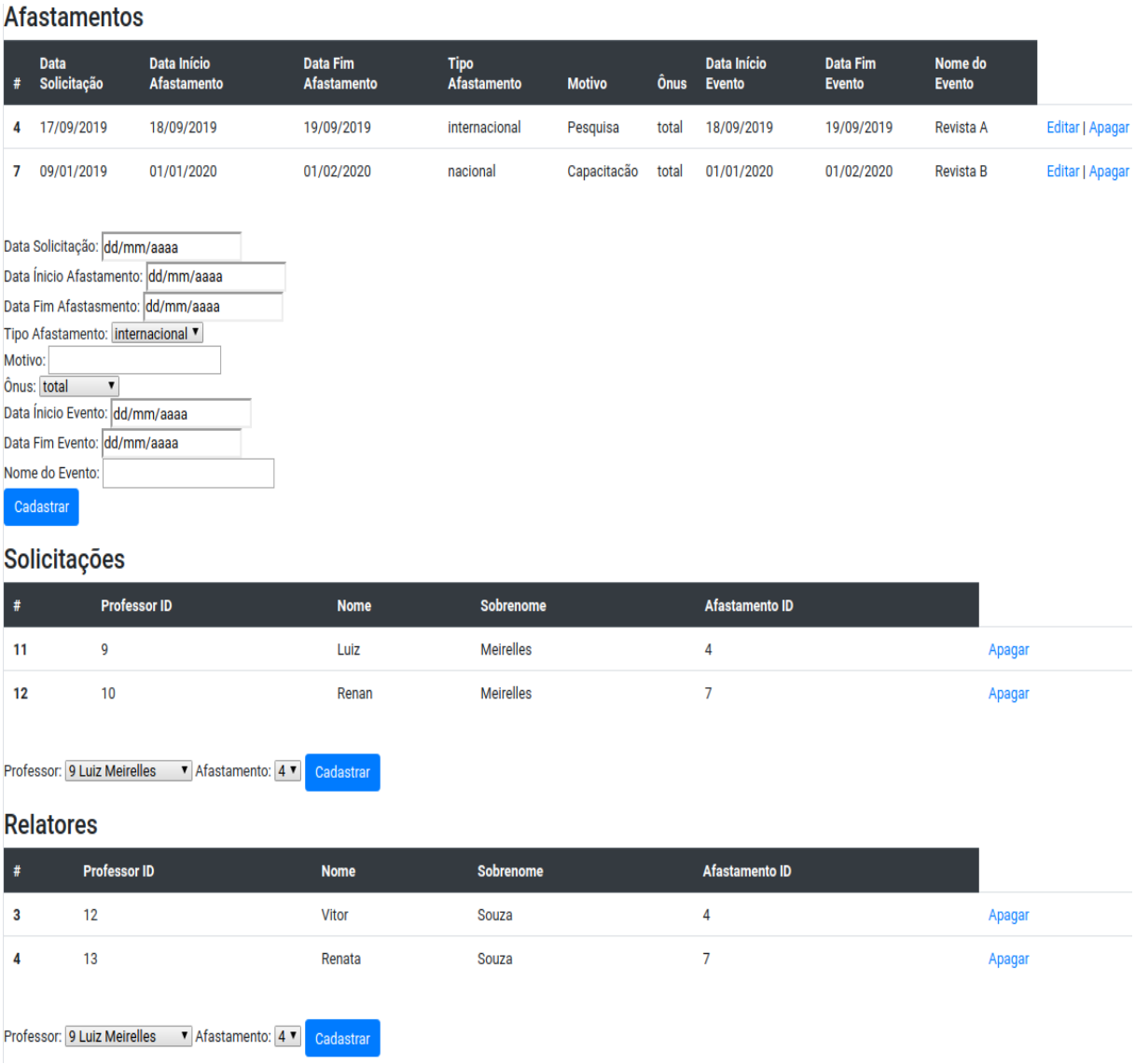

Figura 13 – View Cadastro Afastamento

frameworks CodeIgniter e NodeJS com suas limitações e em suas formas naturais. Houve, inicialmente, uma tentativa de implementar as classes de domínio, porém foi observado que a aplicação iria funcionar sem esta camada e a manutenibilidade não seria difícil. Seguindo o conceito de agilidade, simplesmente me concentrei em entregar algo de valor.

4.3.2 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. Esse modelo é utilizado pelos desenvolvedores para guiar a codificação das classes e componentes dos pacotes Visão e Controle (SOUZA, 2007).

O modelo de navegação para os casos de uso Cadastrar, Consultar, Editar e Remover

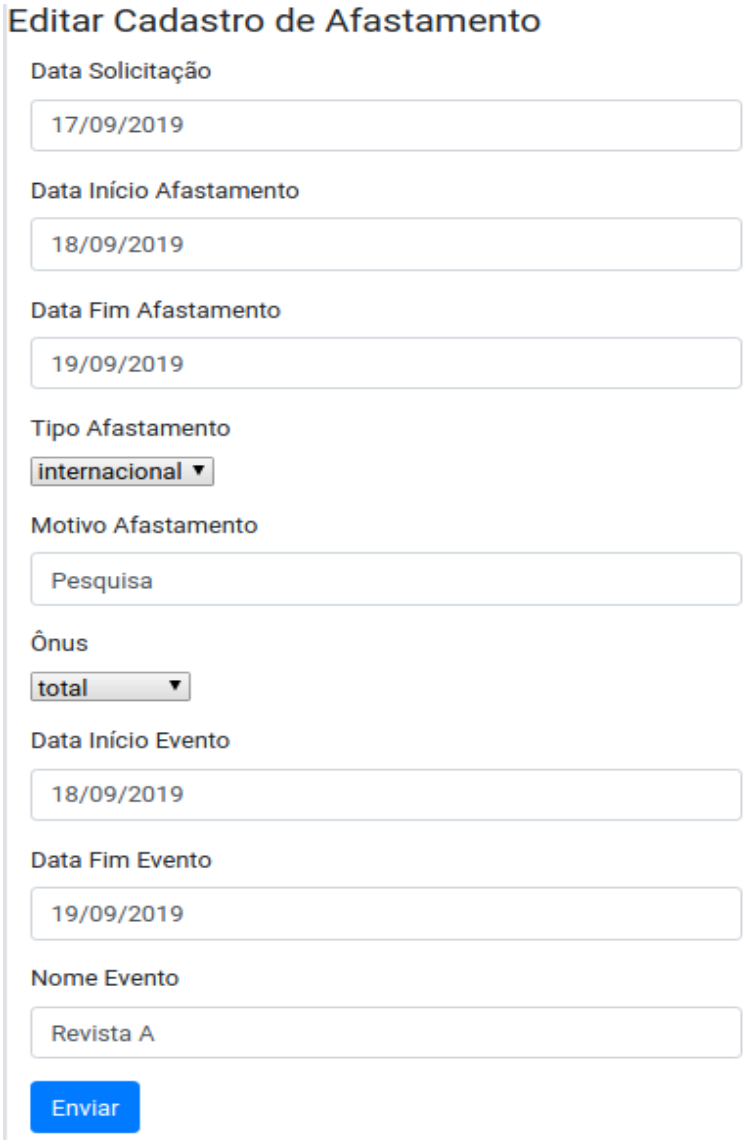

Figura 14 – View Editar Cadastro Afastamento

Professor podem ser observados na Figura 15. A *View* Cadastro Professor é composta por outra *View* Editar Cadastro Professor e ambas acionam o *Controller* Professor para desempenhar suas funcionalidades. A *View* Editar Cadastro Professor, é um formulário com os dados recuperados e tratados, podendo ser editados e submetidos ao *Controller* Professor.

O modelo de navegação para os casos de uso Cadastrar, Consultar, Editar, Remover, Solicitar, Remover Solicitação, Ser Relator, Apagar Relator de Afastamento, podem ser observados na Figura 16. A *View* Cadastro Afastamento é composta por outra *View* Editar Cadastro Afastamento e ambas acionam o *Controller* Afastamento para desempenhar suas funcionalidades. A *View* Editar Cadastro Afastamento é um formulário com os dados recuperados e tratados, podendo ser editados e submetidos ao *Controller* Afastamento.

Note que os *Controllers* não possuem atributos, como sugerido pelo método Fra-

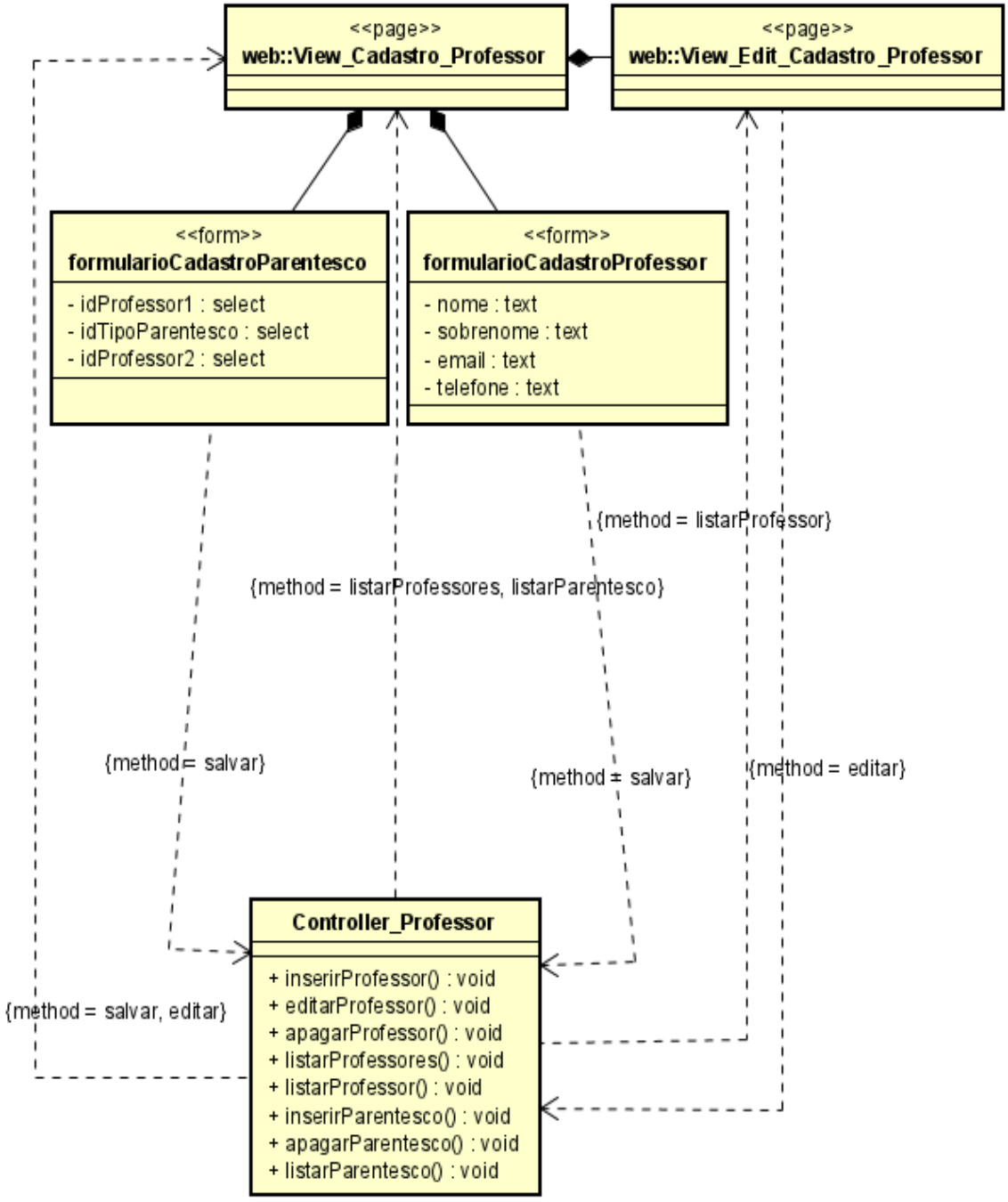

Figura 15 – Modelo de Navegação Professor.

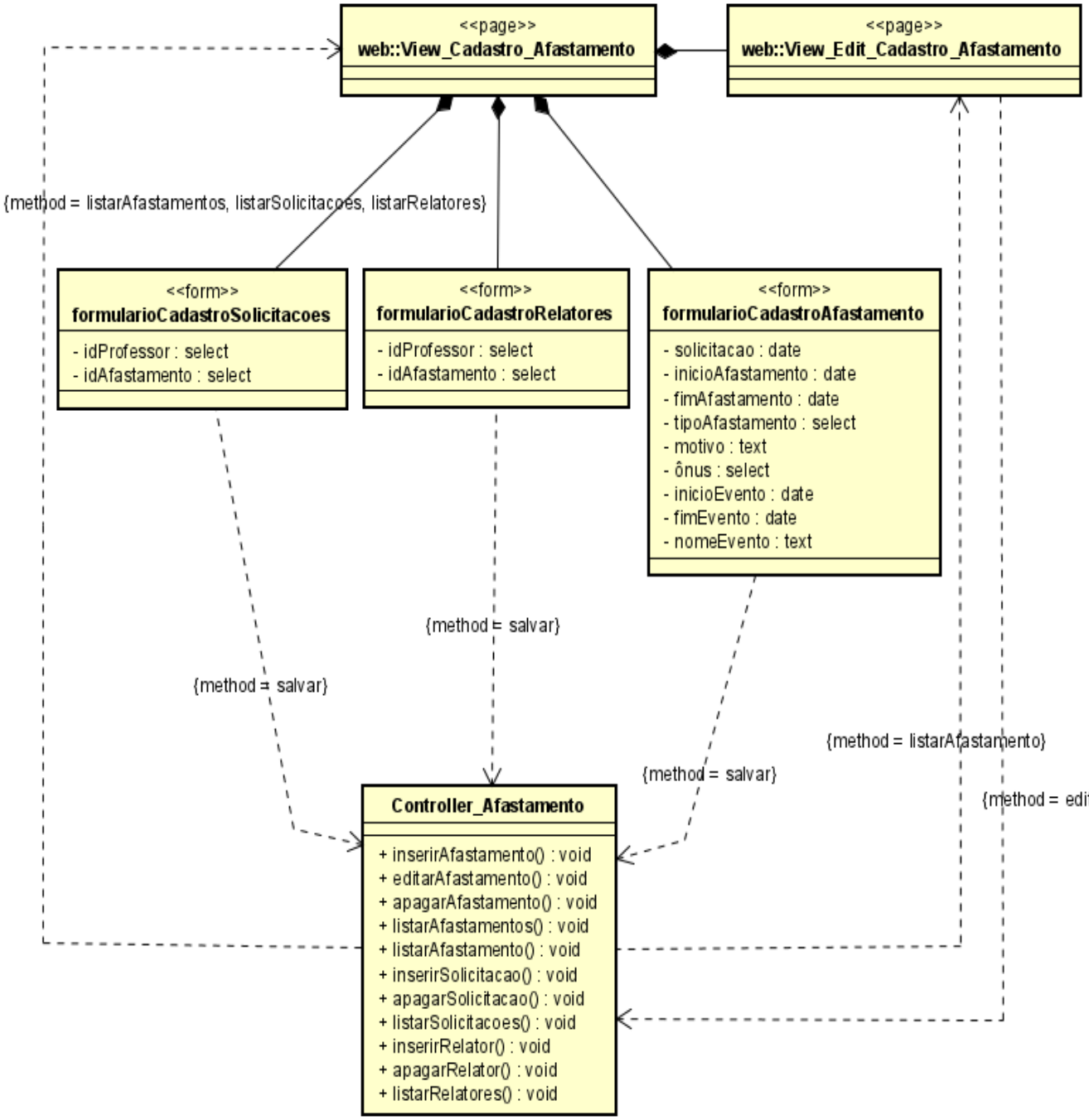

Figura 16 – Modelo de Navegação Afastamento.

meWeb. Os dados são passados da *View* diretamente para as funções dos *Controllers*. Os *Controllers* por sua vez acessam a requisição HTTP contendo os dados. Essa funções recebem os parâmetros fazem as verificações e são passadas para os *Models*. No CodeIgniter cada função disponível no *Controller* já é automaticamente uma rota. No NodeJS é necessário criar uma rota e chamar como *callback* uma função que recebe a requisição, faz verificações e posteriormente passa para os *Models*.

4.3.3 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. Esse diagrama guia a construção das classes DAO, que pertencem ao pacote Persistência (SOUZA, 2007).

Nas duas implementações do SCAP, não foram usados objetos DAOs. Não há também mapeamento objeto/relacional, pois meu objetivo foi explorar os dois *frameworks* de controle e não a preocupação para uma futura adaptação de um módulo ou pacote. Me concentrei na entrega das funcionalidades básicas do ponto de vista da agilidade.

O CodeIgniter por padrão fornece classes *Models*, uma espécie de classe que estende da Classe Model Principal, já tendo conexão com o Banco de Dados e fornecendo funções para o tratamento de dados.

Em NodeJS foi criada uma pasta chamada Model e os arquivos relacionados a *Models* foram implementados lá. Eles importam o conector e realizam toda a comunicação e tratamento com o Banco de Dados.

Na Figura 17 observamos a comunicação direta dos *Controllers* com seus Respectivos *Models*. Há uma fraca ligação entre o *Controller* Afastamento com o *Model* Professor, devido à necessidade de listar dados dos Professores na *View* Afastamento.

4.3.4 Modelo de Aplicação

O Modelo de Aplicação é um diagrama de classes da UML que representa as classes de serviço, que são responsáveis pela codificação dos casos de uso, e suas dependências. Esse diagrama é utilizado para guiar a implementação das classes do pacote Aplicação e a configuração das dependências entre os pacotes Controle, Aplicação e Persistência, ou seja, quais classes de ação dependem de quais classes de serviço e quais DAOs são necessários para que as classes de serviço alcancem seus objetivos (SOUZA, 2007).

Não fora criado nenhuma classe de Serviço e classe DAO, porém os *Models* fazem o mesmo papel dos objetos DAO. Eles proveem todas as funcionalidades para os Casos de Usos, a ligação é direta do *Controller* com o *Model* para desempenhar a funcionalidade, como pode ser observado na Figura 17. Esta figura pode ser considerada uma mistura

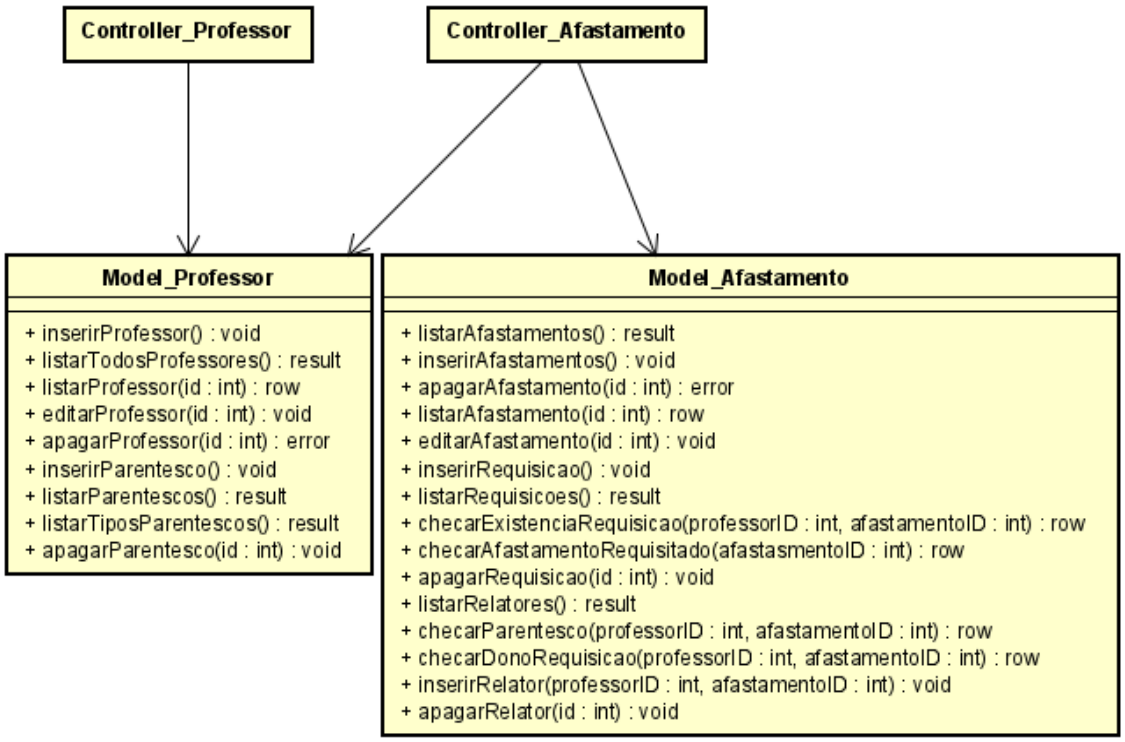

Figura 17 – Modelo de Persistência

entre Modelo de Persistência e Modelo de Aplicação, por mostrar as dependências entre *Controllers* e *Models*.

5 Considerações Finais

Em termos de aprendizado alcançou-se o objetivo dos estudos dos *frameworks* (CodeIgniter e NodeJS) e do método FrameWeb, além de aplicar e exercitar os conhecimentos obtidos durante o curso de graduação. Apesar de não terem sido feitos todos os conjuntos de modelos apresentados pelo FrameWeb, foi possível aproximar ao máximo a implementação do método trabalhando em conjunto com o conceito de agilidade. O objetivo de tentar desenvolver o SCAP totalmente não foi alcançado, pois lidar com dois *frameworks* de desenvolvimento tem seus desafios. Em NodeJS foi complicado o estabelecimento da conexão com o Banco de Dados, teve-se de criar um conector e chamá-lo a todo momento nos *Models*, diferentemente do *CodeIgniter* que já vem pronto e fica disponível para aplicação toda. Ressalto também a dificuldade em executar JavaScript dentro do HTML, para o qual foi necessário o estudo da *ViewEngine EJS*, diferentemente do PHP que pode ser chamado facilmente dentro do HTML. Além disso, houve dificuldade de lidar com uma linguagem assíncrona que é o JavaScript, sendo necessário o aprendizado de *Promises* que fazem a chamada de função esperar, por exemplo ao se fazer uma consulta ao Banco de Dados.

O Método FrameWeb é de fato rico em modelos para organização que facilitam o entendimento e desenvolvimento de WIS. Seu uso contribuiu significativamente para organizar e desenvolver nas duas linguagens/plataformas. O seu conhecimento ajuda na importância da modularização.

Porém, ao implementar com *frameworks* ágeis fica difícil tentar inserir todos os modelos. É notável a carência de estruturas que forneçam "regras" para se manter fielmente ao que é proposto pelo Método FrameWeb. Num mundo competitivo para desenvolvimento de serviços cada vez mais rápidos, simples, com um ligeiro grau de qualidade o FrameWeb se torna muito além do necessário. Para sistemas robustos, serviços já presentes no dia a dia das pessoas e uma equipe grande é sem dúvidas um método ideal e rico para unir a equipe pelo conhecimento.

5.1 Trabalhos Futuros

A Web Semântica (Semantic Web) é considerada o futuro da Internet, mais especificamente uma evolução da World Wide Web atual, referida pelo termo "Web Sintática". Nesta última, "os computadores fazem apenas a apresentação da informação, porém o processo de interpretação fica a cabo dos seres humanos mesmo" (BREITMAN, 2005). O objetivo da Web Semântica é permitir que softwares (agentes) instalados nos computadores sejam capazes de interpretar o conteúdo de páginas da Web para auxiliar humanos a

realizarem suas tarefas diárias na rede (SOUZA, 2007).

Para que agentes possam interpretar o conteúdo de páginas Web, é preciso explicitá-lo em estruturas formais de representação de conhecimento. Na área de Inteligência Artificial, diversas estruturas já foram propostas, como: facetas (*features*), dicionários, índices, taxonomias, thesaurus, redes semânticas etc. (BREITMAN, 2005).

Seria interessante fazer agentes que pudessem inserir, ler, remover e tomar decisões a partir da plataforma SCAP. Para isso é necessário fazer com que o SCAP fosse anotado e publicasse dados na Web Semântica a fim de que agentes pudessem interepretar suas funcionalidades e ao menos ler as informações publicadas dos dados. Vejo como um objetivo de explorar essa área e entender melhor para automatizar tarefas presentes no cotidiano. Com o atual cenário de serviços de Inteligência Artificial como Watson, Azure, Salesforce Einsten entre outros, está mais fácil a construção desses agentes. Considero como trabalho futuro publicar o SCAP na Web Semântica e construir um agente para desempenhar um papel autônomo.

Outra direção futura seria testar mais a aplicação SCAP para ver se atende o atual contexto do corpo docente do DI e refinar seus requisitos para adequar a ferramenta para o cenário atual, caso necessário, lançando, assim, novas versões.

Referências

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

AVELAR, R. d. A. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Ninja*. Vitória, ES, Brasil, 2018. Citado 4 vezes nas páginas 9, 13, 33 e 34.

BAUER, C.; KING, G. *Hibernate in Action*. Greenwich, CT: Manning, 2005. ISBN 1932394370 9781932394375. Disponível em: <http://www.amazon.com/Hibernate-Action-In-Christian-Bauer/dp/193239415X>. Citado na página 21.

BEVAN, N. Usability issues in web design. *Proceedings of Usability Professionals Association Conference, Washington DC, EUA.*, 06 1998. Citado na página 17.

BREITMAN, K. K. *Web semântica: a internet do futuro*. Rio de Janeiro: LTC, 2005. v. 1. 199 p. ISBN 85-216-1466-7. Citado 2 vezes nas páginas 44 e 45.

CASTELEYN, S. et al. *Engineering Web Applications*. 1. ed. [S.l.]: Springer Nature, 2009. Disponível em: https://www.springer.com/gp/book/9783540922001 - Último Acesso em: 15/12/2019. ISBN 8132205286. Citado na página 34.

CONALLEN, J. *Building Web Applications with UML*. 2. ed. [S.l.]: Addison-Wesley Professional, 2002. ISBN 9780201730388. Citado 4 vezes nas páginas 8, 17, 18 e 25.

CRUPI, J.; ALUR, D.; MALKS, D. *Core J2EE Patterns*. 2. ed. [S.l.]: Pearson, 2003. 650 p. ISBN 0131422464. Citado na página 20.

DEMICHIEL, L.; SHANNON, B. *JavaTM Platform, Enterprise Edition (Java EE) Specification, v7*. 2013. Citado na página 12.

DESHPANDE, Y. et al. Web engineering: A new discipline for development of web-based systems. *Springer Berlin Heidelberg*, n. 1, 2001. Disponível em http://www-itec.uniklu.ac.at/ harald/proseminar/web11.pdf - Último Acesso em: 08/04/2019. Citado na página 16.

DUARTE, B. B. *Aplicação do Método FrameWeb no Desenvolvimento de um Sistema de Informação na Plataforma Java EE 7*. Vitória, ES, Brasil, 2014. Citado 9 vezes nas páginas 9, 12, 13, 17, 27, 28, 30, 32 e 34.

FALBO, R. *Projeto de Sistemas de Software*. 2011. Notas de Aula. Disponível em: <http://www.inf.ufes.br/~falbo/files/Notas_Aula_Projeto_Sistemas.pdf>. Citado 2 vezes nas páginas 31 e 33.

FERREIRA, M. T. *Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Wicket e Tapestry*. Vitória, ES, Brasil, 2018. Citado na página 12.

FONS, J. et al. Oows: A method to develop web applications from web-oriented conceptual models. *International Workshop on Web Oriented Software Technology (IWWOST)*, 2003. Disponível em: https://pdfs.semanticscholar.org/6a95/ca5eca1d621ab82cd366596ea153e70ae4ae.pdf - Último acesso em 14/04/2019. Citado na página 17.

FOWLER, M. et al. *Patterns of Enterprise Application Architecture*. [S.l.]: Addison-Wesley Professional, 2002. Citado 2 vezes nas páginas 8 e 24.

GINICE; MURUGESAN. *Web Engineering: An Introduction. IEEE Multimedia. Volume: 8*. 2001. Issue 1. An optional note. Citado na página 15.

HUSTED, T. et al. *Struts em Ação*. 1. ed. [S.l.]: Ciência Moderna, 2004. ISBN 8573932996. Citado na página 19.

KOCH, N. et al. *Extending uml to model navigation and presentation in web applications*. 2000. Proceedings of Modelling Web Applications in the UML Workshop (UML'2000). Citado na página 25.

LI, H.; SHI, T.; YANG, S. An approach of sampling computing for wavelet analysis and its web-based implementation. In: EDITOR, T. (Ed.). *Signal Processing Proceedings*. [S.l.]: The publisher, 2000. v. 1. 5th International Conference. Citado na página 15.

LOWE, D. Website evaluation. In: *Proceedings of WebNet World Conference on the WWW and Internet 1999*. Virginia, EUA: [s.n.], 1999. Citado na página 17.

MARTINS, B. F. *Evolução do Método FrameWeb para o Projeto de Sistemas de Informação Web Utilizando uma Abordagem Dirigida a Modelos*. Vitória, ES, Brasil, 2016. Citado na página 25.

OLSINA, L.; LAFUENTE, G.; ROSSI, G. *Specifying Quality Characteristics and Attributes for Websites*. 2016. ed. [S.l.]: Springer, Berlin, Heidelberg, 2001. Disponível em: http://www.springerlink.com/content/nkwca8u2ljarttmx - Último Acesso em: 14/04/2019. ISBN 978-3-540-42130-6. Citado na página 16.

OLSINA, L.; ROSSI, G. Toward web-site quantitative evaluation: Defining quality characteristics and attributes. In: *Proceedings of WebNet World Conference on the WWW and Internet 1999*. Honolulu, Hawaii: Association for the Advancement of Computing in Education (AACE), 1999. p. 834–839. Disponível em: <https://www.learntechlib.org/p/7405>. Citado na página 17.

PERUCH, L. A. *Aplicação e Análise do Método FrameWeb com Diferentes Frameworks Web*. Vitória, ES, Brasil, 2007. Citado 2 vezes nas páginas 16 e 34.

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 33 e 34.

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 12 vezes nas páginas 8, 9, 12, 13, 25, 27, 28, 29, 30, 31, 32 e 34.

PRESSMAN, R. S. *Software Engineering: A Practitioner's Approach (Mcgraw-Hill Series*

in Computer Science). 6. ed. [S.l.]: McGraw-Hill Science/Engineering/Math, 2005. ISBN 978-0072853186. Citado 3 vezes nas páginas 16, 17 e 30.

PRESSMAN, R. S. *Engenharia de Software - Uma Abordagem Profissional*. 7. ed. [S.l.]: McGraw-Hill, 2011. ISBN 9788563308337. Citado 3 vezes nas páginas 12, 15 e 33.

REENSKAUG, T. M. H. *Thing-Model-View-Editor – an Example from a planningsystem*. 1979. Http://heim.ifi.uio.no/ trygver/1979/mvc-1/1979-05-MVC.pdf. Último Acesso em 14/04/2019. Citado na página 19.

SCHWABE, D.; ROSSI, G. An object oriented approach to web-based applications design. *Theory and Practice of Object Systems - Special issue objects, databases, and the WWW*, v. 4, 1998. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.57 - Último Acesso em 14/04/2019. Citado na página 17.

SILVA, R. P. *UML2 em Modelagem Orientada a Objetos*. 1. ed. VISUAL BOOKS, 2007. ISBN 9788575022054. Disponível em: <https://books.google.com.br/books?id=WBxhvgAACAAJ>. Citado na página 28.

SOMMERVILLE, I. *Engenharia de Software*. 8. ed. [S.l.]: Pearson–Addison Wesley, 2007. ISBN 9788588639287. Citado na página 17.

SOUZA, V. E. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Dissertação (Mestrado) — Programa de Pós-Graduação em Informática Universidade Federal do Espírito Santo, Brazil, 2007. Citado 13 vezes nas páginas 8, 12, 13, 19, 20, 21, 22, 24, 25, 37, 38, 42 e 45.

Apêndices