Rafael dos Anjos Avelar

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Ninja

Vitória, ES

2018

Rafael dos Anjos Avelar

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Ninja

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

Coorientador: Me. Bruno Borlini Duarte

Vitória, ES

2018

Rafael dos Anjos Avelar

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Ninja/ Rafael dos Anjos Avelar. – Vitória, ES, 2018-

51 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, 2018.

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

CDU 02:141:005.7

Rafael dos Anjos Avelar

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Ninja

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, 25 de setembro de 2014:

Prof. Dr. Vítor E. Silva Souza
Orientador

Professor
Convidado 1

Professor
Convidado 2

Vitória, ES
2018

Dedico à Deus pela vida que me foi dada. Ao meu pai, Paulo Maurício Fulgêncio de Avelar(In Memorian) e meu irmão Saulo dos Anjos Avelar(In Memorian), que não poderei abraçá-los para comemorar o fim desta etapa. Seus ensinamentos e quão importante ambos são para mim, estarão sempre comigo.

Agradecimentos

Agradeço a Deus, por tudo que ele faz por todos nós, mesmo quando não percebemos e por colocar pessoas maravilhosas em minha vida.

À a minha mãe, não só pelos cuidados e o carinho, mas também, como exemplo de força e determinação para superar momentos difíceis.

Agradeço aos meus irmãos Marcela, Paulinho e Bruno, os melhores irmãos que um filho caçula poderia querer.

Agradeço aos professores do Curso de Ciência da Computação da UFES, pelo conhecimento a mim compartilhado.

Aos meus amigos, que vivem comigo nessa era são grandes companheiros.

E, especialmente, agradeço à Jacqueline Freire Rigatto, por todo amor, carinho, compreensão e pelo futuro que desejamos ter juntos.

“Alguns acreditam que apenas um grande poder é capaz de manter o mal sob controle.
Mas não é isso que eu tenho visto.
Eu descobri que são as pequenas ações cotidianas de pessoas comuns que mantêm a escuridão distante.
Pequenos gestos de gentileza e amor."
(J.R.R Tolkien, O Hobbit)

Resumo

O desenvolvimento de aplicações para a Internet é um ramo muito importante, não apenas pela base de usuários, mas sim pela grande quantidade de possibilidades de soluções que ela pode prover. Foram criadas muitas tecnologias para que WebApps sejam criados com cada vez mais agilidade, confiabilidade e facilidade de uso. A área da ciência da computação não está concentrada só na criação de tecnologias, mas também em aplica-las da melhor maneira possível. Com a intenção de estudar como aplicações Web podem ser melhor desenvolvidas, surgiu a Engenharia Web.

Os artefatos tecnológicos conhecidos como Frameworks de desenvolvimento Web são o foco de estudo da metodologia FrameWeb(FrameWork-based Desing Method for Wev Engeneering), que divide tais frameworks em categorias e sugeri o uso de modelos, facilitando a utilização de tais tecnologias. O método já foi utilizando anteriormente com frameworks na plataforma Java EE, porém mais usos do método em conjunto com diferentes frameworks agregam valor ao FrameWeb.

Para testar o método com frameworks diferentes foi desenvolvida uma Web App - Sitema de Controle de Afastamento de Professores- ou SCAP. Frameworks baseados na plataforma Java EE 7 foram escolhidos, mais especificamente, o Ninja Framework, e outros que o compõem. O objetivo deste trabalho foi implementar uma nova versão do SCAP com o framework fullstack Ninja.

Palavras-chaves: Ninja. FrameWeb. SCAP.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

WIS Web-based Information System

DI Departamento de Informática

CT Centro Tecnológico

PRPPG Pró-reitoria de Pesquisa e Pós-Graduação

OO Orientado à Objetos

UML Unified Modeling Language

SCAP Sistema de Controle de Afastamento de Professores

API Application Programming Interface

FTL FreeMarker Template Language

HTTP hyperText Transfer Protocol

IDE Integrated Development Enviroment

SGBD Sistema Gerenciador de Banco de Dados

GUI Graphical User Interface

REST Representational State Transfer

POJO Plain Old java Object

CASE Computer-Aided Software Engineering

JSON JavaScript Object Notation

Sumário

1 Introdução

A medida que a Internet passou a ser uma ferramenta usada por inúmeras corporações ao redor do globo, foram popularizando-se sistemas de Informação Web (*Web Information Systems* ou WISs). Uma das ferramentas criadas para ajudar no desenvolvimento de sistemas foram os *frameworks de aplicação* ou simplesmente *frameworks* (FAYAD; SCHMIDT, 1997).

Com o intuito de desenvolver estes sistemas de maneira rápida, eficiente e a fim de conter padrões de qualidade oriundos da Engenharia de Software (PRESSMAN, 2011), Souza (2007) propõe o método FrameWeb. O método define uma arquitetura padrão para facilitar a integração com *frameworks* comumente utilizados no desenvolvimento de aplicações para Web, além de propor modelos, baseados na UML, que trazem para o projeto arquitetural do sistema conceitos inerentes a estes *frameworks*.

O FrameWeb já foi utilizado em outros trabalhos acadêmicos como o SAE - Sistema de Acompanhamento de Egressos (NASCIMENTO, 2016) e o AlocaWeb (SALVATORE, 2016). Sempre que possível, novas tecnologias e frameworks são utilizados, para que estes trabalhos sirvam como casos de exemplo de uso do FrameWeb, sugerindo alterações para que o método se adapte estas tecnologias utilizadas. Para dar foco na mudança dentre frameworks, alguns destes trabalhos se propõe a desenvolver um mesmo sistema — a saber, o Sistema de Controle de Afastamento de Professores (SCAP) (DUARTE, 2014; PRADO, 2015) — com frameworks diferentes.

Visando contribuir neste sentido, o objetivo deste trabalho é realizar uma nova implementação do SCAP -Sistema de Cadastramento de Afastamento de Professores, considerando os requisitos levantados por Duarte (2014) e revisados por Prado (2015), aplicando-se o FrameWeb, porém utilizando o framework Ninja.[1] A partir dessa implementação, deseja-se avaliar os impactos que o novo framework ao seguir o método FrameWeb.

1.1 Objetivos

O desenvolvimento deste projeto de graduação tem como principal objetivo aplicar o método FrameWeb (SOUZA, 2007), utilizando o framework Ninja e utilizando o documento de requisitos de sistema do SCAP (Sistema de Cadastro de Afastamento de Profesores), elaborado por (DUARTE, 2014) e revisado por (PRADO, 2015).

O objetivo acima pode ser decomposto nos seguintes sub-objetivos:

[1] http://www.ninjaframework.org

- Estudar as documentações do SCAP feitas por Duarte (2014) e Prado (2015), e o método FrameWeb.
- Implementar o SCAP com o framework Ninja e os documentos de projetos criados com o uso do FrameWeb.
- Analisar o resultado da implementação, esclarecendo as semelhanças e diferenças que o novo framework traz ao SCAP e consequentemente ao FrameWeb.

1.2 Metodologia

A produção deste trabalho de graduação foi dividido nos seguintes passos:

- Estudo e Pesquisa: o método FrameWeb, tecnologias e os frameworks envolvidos no desenvolvimento do projeto, foram estudados.
- Estudo da Documentação de Requisitos: a documentação de requisitos revisada por (PRADO, 2015) foi a base para o desenvolvimento da aplicação.
- Desenvolvimento da aplicação: implementação da API do WIS e de elementos para a visualização das páginas, levando em consideração a documentação de requisitos e o método FrameWeb.
- Análise sobre impacto do novo framework ao sistema: observar o funcionamento das funcionalidades do sistema, a documentação de requisitos, e identificar as mudanças que houveram, em decorrência do uso de uma nova tecnologia.
- Redação da monografia: etapa correspondente à escrita deste documento.
- Apresentação final dos resultados.

1.3 Organização do texto

Esse trabalho é composto por cinco partes, contando com essa introdução, as demais estão descritas a seguir:

- **Capítulo 2** – Apresentação do conteúdo: resenha dos conhecimentos adquiridos e essenciais à compreensão composição desse trabalho;
- **Capítulo 3** – SCAP: Descrição do Escopo do Problema e apresentação da análise de Requisitos;
- **Capítulo 4** – Projeto e Implementação: apresentação do sistema e do *Framework* usado no desenvolvimento do projeto;

- **Capítulo 5** – Considerações Finais: análise feita sobre o trabalho desenvolvido onde são apresentadas as conclusões, limitações e oportunidades de trabalhos futuros.

2 Referencial Teórico

A Internet, enquanto apenas um conjunto de páginas estáticas sem a capacidade de acessar outras informações além das contidas em seus arquivos HTML, era pouco interessante fora do campo acadêmico. Durante sua evolução, surgiu a possibilidade de integrar-se com sistemas de informação, permitindo-lhe acesso à tecnologias como banco de dados e outras aplicações já existentes, como por exemplo, os sistemas bancários. Isso despertou o interesse principalmente de empresas que viram seu potencial e passaram a exigir o desenvolvimento de aplicações capazes de prover facilidades para processos já existentes, e novas funcionalidades. Dessa maneira surgiu o conceito de "aplicações Web" (*Web Applications* ou apenas *WebApps*).

Estas novas aplicações eram desenvolvidas de maneira *ad hoc*, ou seja, sem uma padronização ou metodologia adequada, isto inviabilizava o uso de práticas de desenvolvimento comuns à outros tipos de sistemas, como: gerenciamento de riscos, métricas e de controle sobre o projeto de software (PRESSMAN, 2011). Essa falta de metodologia fez com que os conceitos oriundos da engenharia de software tradicional fossem adaptados às novas necessidades. Este novo ramo da engenharia de software é conhecida hoje em dia como Engenharia Web.

Um dos conceitos adaptados para a Engenharia Web foi o uso de *frameworks*. Não foram encontradas metodologias adequadas especificamente para esta nova categoria de aplicações de desenvolvimento Web. Com o intuito de interagir com esses *frameworks*, que (SOUZA, 2007) desenvolveu o método FrameWeb, um conjuntos de modelos intermediários, usando linguagem *UML*, baseados nos documentos de projeto de sistemas, porém mais específico para o conjunto de *frameworks* escolhidos para implementar a solução do problema.

Neste capítulo, serão abordados os temas sobre a Engenharia de Software utilizadas ao desenvolvimento de aplicações Web (Engenharia de Software Web, ou EngWeb), dos conceitos e divisões dos frameworks utilizados para criar um sistema de informação através do método FrameWeb.

2.1 Engenharia Web

A Engenharia Web, *Web Engineering – WebE* é o estudo para a criação, documentação, implantação e manutenção de aplicações Web. Uma aplicação feita segundo a *WebE* é projetada seguindo uma série de metodologias, padronizações, definições de conceitos. Todas essas abordagens tem por objetivo garantir que atributos de qualidade e que os

stakeholders fiquem satisfeitos. Olsina, Lafuente e Rossi (2001) define tais atributos de qualidade:

- **Usabilidade:** a aplicação precisa de uma interface legível, que possa atender diversos tipos de usuários, incluindo casos mais especiais (portadores de alguma deficiência), seja responsiva, ou seja, que informe aos usuários quando seus comandos foram reconhecidos pelo sistema e se foram bem sucedidos ou não.
- **Funcionalidade:** ser capaz de executar a função para a qual o sistema foi desenvolvido. Nos sistemas de informação web (ou *WIS: Web-based Information System*), isto geralmente envolve a apresentação de páginas e links.
- **Eficiência:** páginas de internet, hoje em dia, precisam ser montadas (ou renderizadas) a partir de informações provenientes de outras partes do sistema. Este processo precisa levar tempo inferior a 10 segundos, para que a página não seja carregada sem tais informações.
- **Confiabilidade:** O sistema precisa ser capaz de manter consistência mesmo após falhas, revertendo estados inconsistentes, além de saber lidar com falhas cometidas pelos usuários.
- **Manutenibilidade:** O sistema precisa ser capaz de adquirir novas funcionalidades e ser capaz de se adaptar a novas tecnologias que venham a surgir na Web.

Para que tais atributos de qualidade sejam alcançados e as expectativas do cliente satisfeitas, a WebE divide o desenvolvimento seja feito por etapas. Cada uma delas produzindo um artefato, até que finalmente, obtêm-se o sistema devidamente implementado, testado e implantado. Essas etapas são iterativas, ou seja, fases tendem há ocorrer mais de uma vez, de maneira cíclica, com o intuito de aprimoramento dos artefatos gerados em cada etapa (BOURQUE; FAIRLEY et al., 2014).

A primeira fase, chamada de levantamento de requisitos, é onde identifica-se as capacidades almejadas para o sistema. Define-se tais necessidades como problemas a serem resolvidos, e em seguida divide-os em problemas cada vez menores, ao ponto que possam ser definidas de maneira simples e sucinta, chamamos estas definições de requisitos funcionais. Enquanto que, as restrições observadas no ambiente do problema, são denominadas requisitos não-funcionais.

Ao fim desta etapa, tem-se três objetivos a serem alcançados: descrever o pedido do cliente, estabelecer uma base para um projeto de software e definir um conjunto de requisitos que possam ser usados para validar o software futuramente (PRESSMAN, 2011).

Segundo (BOURQUE; FAIRLEY et al., 2014), a fase conhecida como análise de requisitos é responsável por:

- Detectar e resolver conflitos entre requisitos;
- identificar as limitações do software e como ele deve interagir com a organização e o ambiente inserido;
- Elaborar uma documentação de requisitos do sistema capaz originar requisitos de software.

Após a fase de análise, inicia-se o projeto de software, e como dito por Falbo (2017): "o objetivo da fase de projeto (ou design) é produzir uma solução para o problema identificado e modelado nas fases de levantamento e análise de requisitos, incorporando a tecnologia aos requisitos e projetando o que será construído na implementação".

O projeto da arquitetura envolve, dentre outras, questões relativas à organização e estrutura geral do sistema, seleção de alternativas de projeto, atribuição de funcionalidades a elementos de projeto e atendimento a atributos de qualidade (requisitos não funcionais). (MENDES, 2002)

O Projeto de Arquitetura pode ser dividido em objetivos:

- Primeiro: definir como os dados irão interagir. No FrameWeb, este aspecto tem maior evidência nos modelos de entidades e navegação;
- Segundo: consiste determinar como as soluções tecnológicas trabalham e interagem umas com as outras, levando em consideração as estruturas, padronizações e limitações de cada *framework* ou tecnologia envolvida. Este aspecto é mostrado em todos os modelos do FrameWeb, visto que essa é justamente a proposta do método, aproximar a fase de projeto à fase de implementação, ao descrever os modelos com mais detalhes das tecnologias utilizadas.

A arquitetura de sistema escolhida por Souza (2007) para o desenvolvimento do FrameWeb foi é a *Model-View-Controller*(FOWLER, 2002), abreviada como MVC. Desenvolvida a partir dos trabalhos de Wojciechowski et al. (2004). Sua principal premissa, como pode ser vista na figura 1, é a divisão do sistema em três módulos: modelo, responsável por representar os dados do domínio ou da lógica de negócio; visão, responsável por recolher as entradas do usuário e apresentação de dados; e controlador, responsável pelo envio das requisições do usuário controle de fluxo do sistema.

Uma vez que são finalizadas as primeiras versões dos modelos da fase de projeto, pode-se inciar as fases de implementação, que consiste na tradução destes modelos em códigos-fonte de acordo com a linguagem de programação definida previamente, e começa-se à testar as funcionalidades da ferramenta.

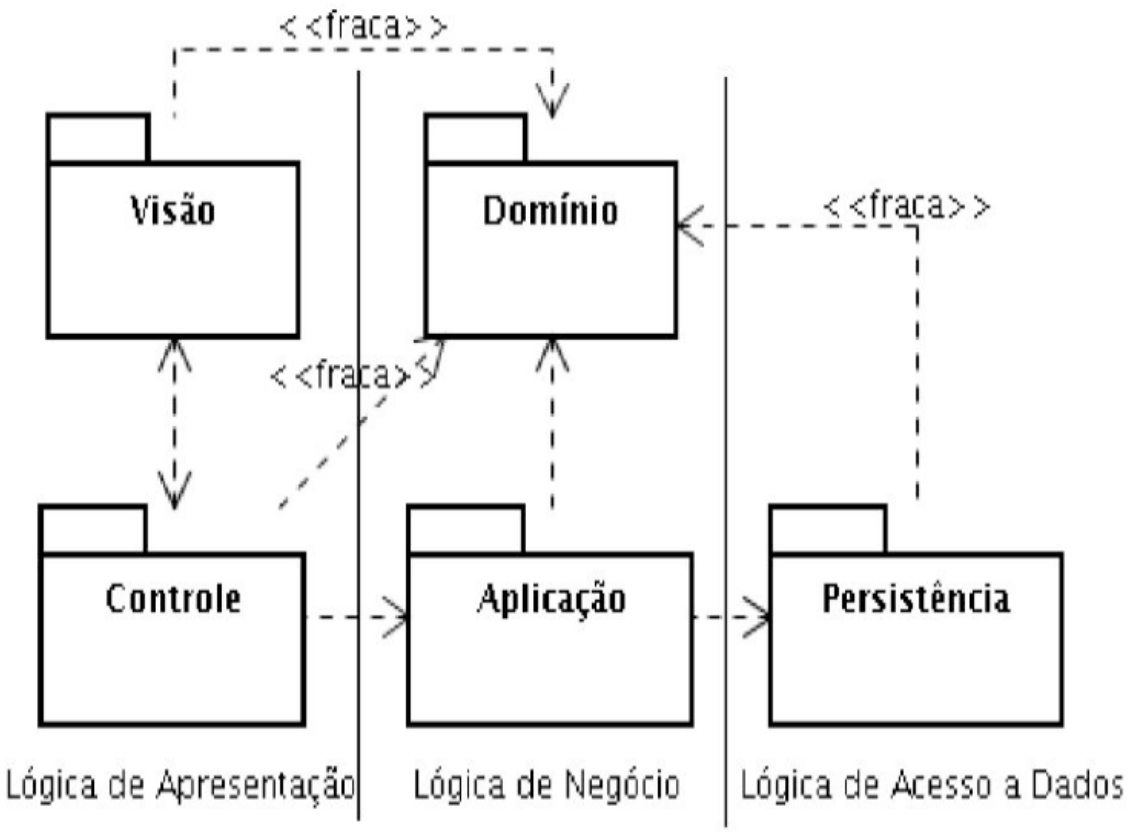

Figura 1 – Arquitetura padrão de camadas de serviço (FOWLER, 2002).

2.2 FrameWorks

Após o surgimento de *WISs* mais complexos, percebendo as necessidades semelhantes dentro dessa classe de sistemas, foram desenvolvidos os primeiros *frameworks* para aplicações Web. Frameworks podem ser definidos como aplicações semi-completas e reutilizáveis, capazes de serem utilizadas para o desenvolvimento de novas aplicações.

Normalmente, na criação de um *WIS*, mais de um *framework* é utilizado em seu desenvolvimento, uma vez que existem diferentes camadas dentro da aplicação. Os frameworks para desenvolvimento Web exercem o papel de, pelo menos, uma das seis categorias descritas por Souza (2007):

- 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.

Os frameworks podem ser usados de acordo com a necessidade, sendo assim, não é obrigatório que uma aplicação Web esteja usando frameworks para todas as finalidades apresentadas. Dentre os tipos de frameworks apresentados, os que foram utilizados nesta implementação do SCAP foram: frameworks MVC, decorador, de mapeamento objeto/relacional e de injeção de dependência.

Rafael: não sei se o FreeMarker chega a ser considerado um Framework decorador, acredito que sim pois ele é capaz de montar páginas incluindo, inserção de rodapés .

Rafael: está comentado um trecho descrevendo os 3 principais tipos de frameworks utilizados no trabalho, fiquei na duvida sobre tentar finalizar de escrever ou não.

2.2.1 FrameWorks MVC

Baseado no padrão de projeto (*desing pattern*) de Fowler (2002). Também conhecidos como frameworks controladores frontais, um framework deste tipo é o intermediário dentre duas camadas da aplicação : visão e lógica de negócio.

Estes frameworks, como descrito na figura 2, precisam gerenciar as conexões, protocolos como o HTTP entre o servidor de aplicação e os clientes. Instanciam (ou possuem injeções de) classes de ação que farão a execução da lógica de negócio. E são capazes de traduzir as informações em formatos de dados próprios da Web em dados compreensível para a camada de aplicação, e vice-versa.

2.2.2 FrameWorks Injeção de Dependência

Também conhecidos como Frameworks de inversão de controle (*Inversion of Control - IoC*), um framework dessa classe não reside numa camada específica do modelo criado por (FOWLER, 2002). Seu uso é muito importante pois, permite um baixo nível de acoplamento entre objetos de camadas distintas, algo extremamente desejável quando possuí uma arquitetura onde existem camadas independentes que precisam se comunicar.

Ao invés de um objeto criar suas dependências, ele a "injeta", isto é, faz uma chamada à outros objetos que são gerenciados por terceiros. Tal chamada é feita através de anotações(*annotations*), essas anotações também são utilizadas durante a implementação das classes para indicar o controle de instanciação.

2.2.3 Frameworks Decoradores

Esta classe de frameworks estão localizadas no pacote de visão e são responsáveis pela montagem de páginas, mantendo um padrão de *Design*. Possui diversos elementos

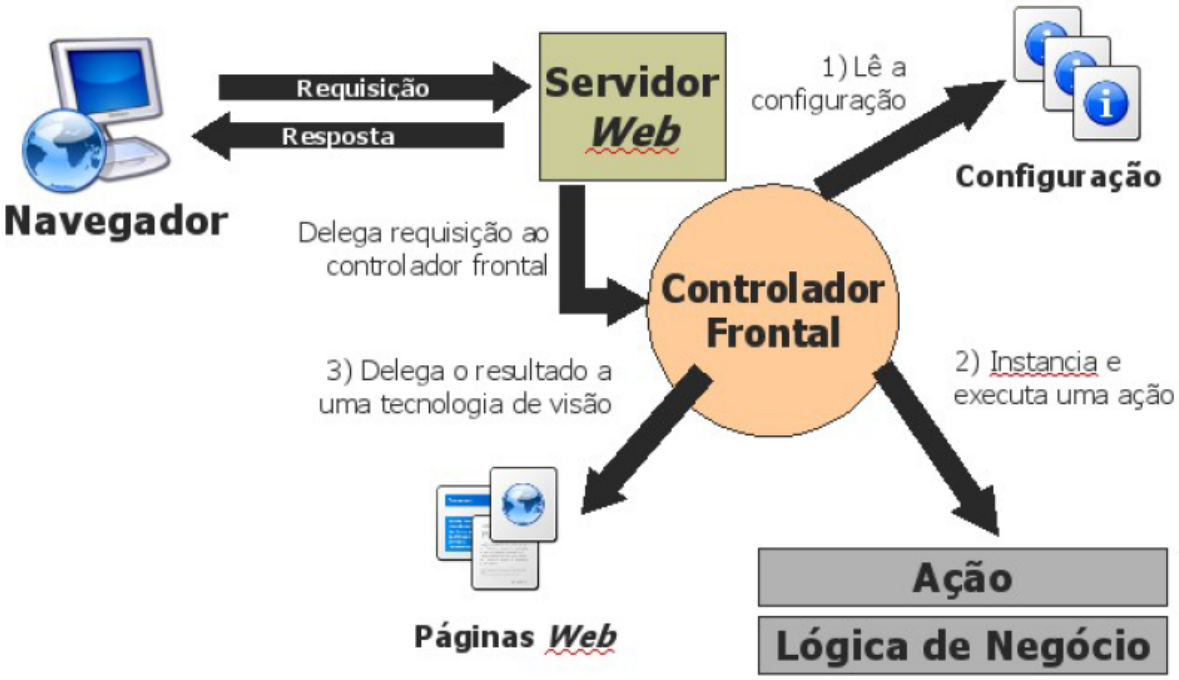

Figura 2 – Funcionamento de um frameWork MVC (SOUZA, 2007).

gráficos como: *layouts*, esquemas de cores, barras de navegação, cabeçalhos e rodapés. Tudo com o intuito de prover usabilidade ao usuário.

2.2.4 FrameWorks de mapeamento Objeto/Relacional

Conhecidos em inglês como *ORM (Obejct/Relacional Mapping)*, essa clase de frameworks é utilizada devido a incompatibilidade entre paradigmas, da classes provenientes do paradigma Orientado à Objetos(OO), e das estrutura de SGBDs relacionais (apesar de haver bancos de Dados orientados à objetos, estes não alcançaram grande uso comercialmente).

2.2.5 Ninja FrameWork

"No ano de 2012, ainda não havia uma única ferramenta que integrava essa lista de conceitos e tecnologias[1]*:"*

- *Web Friendly* e de arquitetura REST(FIELDING; TAYLOR, 2000);
- Java *vanilla*("puro") com injeção de dependências e fácil integração com IDEs;
- Rápidos ciclos de desenvolvimento;
- Facilidade e agilidade em testes;

[1] http://www.ninjaframework.org/introduction.html

- Tecnologias para *mocking tests* (MACKINNON; FREEMAN; CRAIG, 2000);
- Uso simplificado de JSON [2];
- Estruturas prontas para autentificação de usuários;
- Renderização de páginas HTML, validação de formulário;
- suporte à *Google App Engine(GAE)*[3].

A principal proposta do Ninja é ser um conjunto de frameworks para aplicações Web completo, ou seja, sem a necessidade de configurações extras que o desenvolvedor teria de fazer para integrar múltiplos frameworks. Para este objetivo, ele incorpora frameworks bem estabelecidos para cada da aplicação, entrega-os pré-configurados, através de arquétipos do Maven[4].

Por estar presente em todas as camadas (*stacks*) de uma aplicação, este framework é chamado de *Fullsatck*, ("toda a pilha") em tradução livre.

Os frameworks impactam diretamente no desenvolvimento de aplicações, conferindo eficiência e confiabilidade, uma vez que trata-se de códigos-fonte prontos e validados por seus criadores e comunidade de usuários. No entanto, não havia uma arquitetura que baseava-se em frameworks de desenvolvimento Web. Isso incentivou Souza (2007) a desenvolver o método FrameWeb, sendo melhor explicado na seção 2.3.

2.3 O Método FrameWeb

FrameWeb é um método para o projeto de Sistemas de Informação Web que assume que determinados tipos de frameworks serão utilizados durante a construção da aplicação, define uma arquitetura básica para o WIS e propõe modelos de projeto que se aproximam da implementação do sistema usando esses frameworks (SOUZA, 2007). Como o FrameWeb se baseou no modelo MVC Fowler (2002), a descrição feita anteriormente na seção 2.1 se mantém, assim como a figura 1 é também, uma representação desse método.

Os estudos da engenharia de software envolvem muitas camadas de abstrações, olhando um sistema de diferentes formas: modelos de caso de uso, onde funcionalidades são descritas como ações de um tipo de usuário (ator) sobre o sistema; diagramas de classes, que tratam os dados do problema com base no paradigma orientado à objeto. Contudo, não haviam alternativas em desenvolvimento Web que personalizavam os modelos de fase de projeto aos *frameworks* escolhidos (SOUZA, 2007). A especificação do método define quatro modelos:

[2] https://www.json.org/
[3] http://code.google.com/appengine/
[4] https://maven.apache.org/

- Modelo de Entidades (antigo Modelo de Domínio): Representa o pacote de "Domínio"na figura 1 e é baseado no diagrama de classes da fase de análise, mas desta vez, especificado de acordo com os *frameworks* e demais tecnologias utilizadas, como os tipos dos dados e instruções para o mapeamento objeto/relacional;

- Modelo de Aplicação: representa as classes de serviço do sistema implementadas no pacote "Aplicação"da figura citada acima, é usado para guiar a implementação das mesmas e demostrar suas dependências;

- Modelo de Navegação: é um diagrama que representa as classes pertencentes ao pacote "controle", junto de elementos visuais da aplicação, como páginas Web, formulários e demais elementos que componham o pacote "Visão", e da interação entre os pacotes. Basicamente demostra o funcionamento da camada de Lógica de Apresentação;

- Modelo de Persistência: representa a implementação das classes do sistema referentes à camada de acesso à dados, mostrando como as interfaces conectam as classes de serviço às implementações contidas no pacote "Persistência", e apresentam os métodos específicos de cada implementação. Para o acesso a dados, Souza (2007) sugere o padrão de projeto *Data Access Object*, ou DAO (ALUR et al., 2003).

3 SCAP

Neste capítulo é descrito o escopo do SCAP (Sistema de Controle de Afastamentos de Professores), bem como são apresentados os modelos de casos de uso que foram observados a partir dos requisitos previamente levantados por Duarte (2014), aprimorados por Prado (2015) e utilizados neste trabalho.

3.1 Descrição do Escopo

O Departamento de Informática (DI), do Centro Tecnológico (CT) do Campus de Goiabeiras da Universidade Federal do Espírito Santo (UFES) pretende manter um registro sobre as solicitações, do corpo docente, referentes aos afastamentos de suas atividades acadêmicas para participações em eventos. Também pretende facilitar o cadastro e análise destas solicitações, tanto nacionais quanto internacionais. Estas serão feitas através de formulários nas páginas Web do SCAP, e este deve ser capaz de notificar, via e-mails automáticos, para os membros do departamento (professores e secretários) cadastrados no sistema.

Os professor do DI que deseja se ausentar de suas atividades acadêmicas, preenche um formulário com as informações necessárias: datas iniciais e finais dos períodos de sua ausência da faculdade; motivo desta ausência e se a universidade será onerada com a viagem, assim como informações relativas ao evento também precisam ser inseridas: datas de início e fim do evento, assim como seu nome e cidade sede. Pedidos que envolvam viagens nacionais são avaliados pelo Chefe de Departamento ou sub-chefe, que são professores do DI exercendo tais cargos no momento. Qualquer professores do departamento podem se manifestar, via preenchimento de formulários no SCAP, contra o pedido de afastamento de outro professor. Não havendo contestações e tendo o aval do Chefe do Departamento, em 10 dias, a solicitação é aprovada.

Nos casos de eventos internacionais, é necessário escolher outro docente, sem parentesco com o solicitante, para assumir o papel de Relator do pedido. Após parecer do relator, o pedido passa por aprovação no departamento como no caso acima. A solicitação ainda precisa ser encaminhada e aprovada, tanto pelo CT quando pela Pró-Reitoria de Pesquisa e Pós-Graduação (PRPPG) para, finalmente, serem publicadas no Diário Oficial da União. O envio e recebimento de documentações e pareceres de fora do DI não é responsabilidade do SCAP e foge do seu escopo de atuação. Em decorrência disso, o secretário do DI é responsável por registrar os pareceres vindos do CT e da PRPPG no sistema.

3.2 Modelos de Caso de Uso

A Figura 3 representa uma tabela com uma descrição sucinta de cada ator presente no sistema.

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

Figura 3 – Atores que interagem com o SCAP (DUARTE, 2014).

O SCAP foi divido por Prado (2015) em dois subsistemas, **núcleo** e **secretaria**. O primeiro, focado em atender os casos de uso do **Professor** e **Chefe de Departamento**. Já o segundo nas funcionalidades atribuídas ao **Secretário**. Portanto, o digrama de casos de uso foi divido de acordo com os atores envolvidos, nas figuras 4 e 5.

O **Professor** aparece como ator principal do sistema e as funcionalidades disponíveis a ele são: **Cadastrar** e **Cancelar** suas solicitações (diferenciadas caso sejam **nacionais** ou **internacionais**), receber notificações e **consultar** sobre os afastamentos de outros professores, **manifestar-se contra** caso deseje, e por fim, **deferir parecer** quando for apontado como relator. O chefe de departamento, sendo um cargo adicional, administrativo e temporário que um professor do DI assume, tem como função extra de **encaminhar as solicitação de afastamento** ao devido relator, nos casos de afastamentos internacionais. O diagrama de casos de uso do Professor e Chefe do departamento estão representados na Figura 4.

O **Secretário**, como funcionário unicamente administrativo, possui diversas funções neste âmbito, não tomando decisões sobre o andamento das solicitações. Suas atribuições resumem-se a: **Consultar Afastamento**, funcionalidade idêntica à do professor; **Cadastrar Usuário**, seja ele professor ou outro secretário; **Casdastrar Chefe de Departamento**, onde o período desse mandato é inserido no sistema; **Registrar Parecer CT** e **Registrar Parecer PRPPG**, quando os órgãos citados deferem sobre alguma solicitação internacional; e finalmente, **Arquivar afastamento**, quando um pedido passa por todos os trâmites necessários e o secretário muda seu status para “Arquivado”. Os casos explicados podem ser vistos na Figura 5.

3.3 Modelo Conceitual

A divisão do SCAP em dois subsistemas não afetou o diagrama de classes, que representa todo o sistema. Vale enfatizar que todas as classes de domínio estão implementadas

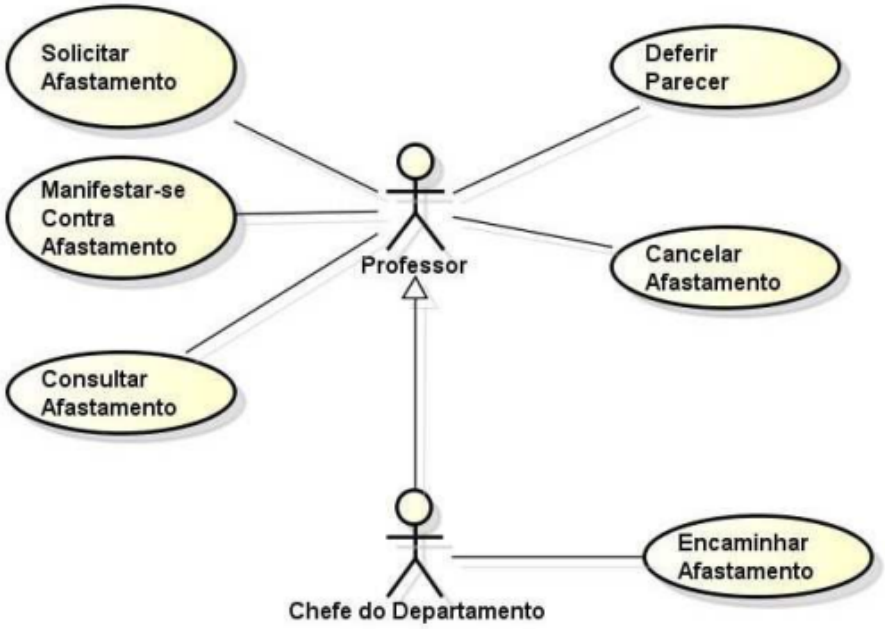

Figura 4 – Diagrama de Casos de Uso referente subsistema núcleo (PRADO, 2015)

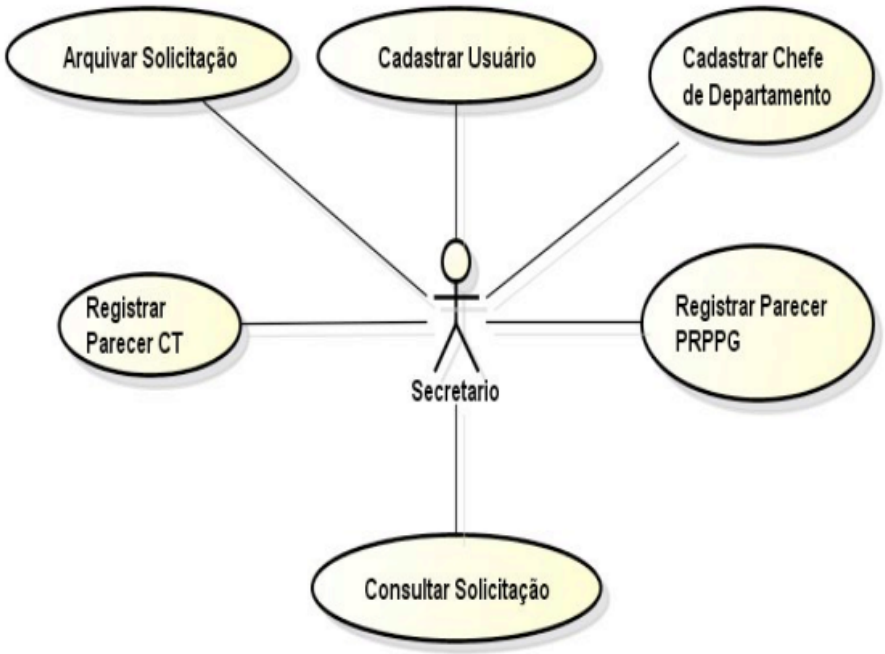

Figura 5 – Diagrama de Casos de Uso referente ao subsistema secretaria (PRADO, 2015)

no subsistema núcleo.

Para esta implementação do SCAP, foram implementadas apenas dois casos de uso, um condizente ao ator **Secretário** e outro ao ator **Professor**. Os casos de uso foram respectivamente : **Cadastrar Usuário** e **Cadastrar Afastamento**. As Funcionalidades de do caso de envio de e-mails e o upload de arquivos uso de Cadastro de usuário não

foram abordados neste trabalho, uma vez que demandariam muito tempo para na etapa de desenvolvimento e não contribuiriam para mudanças nos modelos FrameWeb na fase de projeto.

As classes **Professor** e **Secretário** representam os atores da fase de análise de requisitos de mesmo nome e, além disso, possuem os mesmos atributos, portanto herdam da classe **Pessoa**. Um professor pode ter relações de **Parentesco** com nenhum ou muitos outros professores. Por este tipo de associação ser de multiplicidade (*muitos para muitos*) que cria-se uma classe associativa para representar quando ocorre um parentesco (FALBO, 2014).

Para se determinar quem é chefe do departamento num período de tempo qualquer, usam-se objetos da classe **Mandato**, cada um relacionado a um professor e contendo as datas de início do mandato, e do fim uma vez que o mandato tenha terminado. Os atuais mandatos (do chefe e sub-chefe do departamento) terão as datas de fim, nulas.

Um objeto do tipo **Afastamento** representa uma única solicitação feita por um professor mas, seu atributo *tipo do afastamento* que classifica-o como internacional ou nacional. No primeiro caso, é necessário que outro professor assuma o papel de **Relator**. Em ambos os casos, podem conter um ou mais **Documentos** e múltiplos **Pareces** podem ser inseridos no decorrer do processo.

Mais detalhes sobre os requisitos do SCAP utilizados neste trabalho encontram-se em (PRADO, 2015).

4 Projeto Arquitetural e Implementação

O SCAP, especificamente, foi implementado múltiplas vezes através do uso do FrameWeb. Para essas implementações, foram utilizadas tecnologias diferentes em cada implementação. Contudo, há tecnologias semelhantes que, aliadas à documentação de Requisitos utilizada, implicam que haverão muitas similaridades nos modelos e implementações, algo até desejado ao buscar entender as nuâncias que um framework pode trazer ao método. Os frameworks utilizados nas três versões do SCAP podem ser vistos na tabela 1.

Este capítulo contém os dados técnicos das ferramentas utilizadas para a implementação SCAP e expõe as razões para decisões tomadas durante a fase de implementação, além das alterações relevantes feitas aos modelos do SCAP.

4.1 Arquitetura do Sistema

O documento de projeto, divide o SCAP em dois sub-sistemas:

- Núcleo: sub-sistema que detém todas as classes de domínio responsável;
- Secretaria: sub-sistema responsável em tratar os casos de uso dos secretários do departamento.

4.2 Tecnologias

O setor industrial de desenvolvimento de aplicações web cresceu bastante em seu pouco tempo de existência. Uma das consequências desse crescimento é o aumento na complexidade dos sistemas. A implementação de novas aplicações deixaram de basear-se apenas em tecnologias como linguagens de programação e estruturas de dados, agora utilizam-se outras aplicações, especializadas, para a implantação desses sistemas maiores e mais complexos. Algumas das aplicações utilizadas para a implementação dessa versão do SCAP são:

Tabela 1 – Tabela com os frameworks utilizados nas versões do SCAP feitas por Duarte (2014) e Prado (2015) e na atual implementação.

Categorias de Web Frameworks segundo Souza (2007):	Versão de Duarte (2014)	Versão de Prado (2015)	Versão deste trabalho
Decorador	Prime Faces	Nenhum	FreeMarker
Controlador Frontal	JSF	VRaptor	Ninja
Injeção de Dependências	CDI	CDI	Google Guice
Mapeamento Objeto/Relacional	Hibernate	Hibernate	Hibernate

- Java EE: plataforma de desenvolvimento mais popular na atualidade. Vale ressaltar que o SCAP foi desenvolvido inicialmente por Duarte (2014), na plataforma Java EE 7, mas esta atualmente encontra-se em sua 8ª versão, a qual está sendo utilizada nesse projeto;

- Wildfly:[1] servidor HTTP e *Servlet Container* de aplicações, implementado sobre a plataforma Java EE;

- Eclipse:[2] IDE que possui versões para as principais plataformas de desenvolvimento, possui diversas ferramentas embutidas e diversos frameworks se integram ao ambiente através da instalação *plugins*;

- MySQL:[3] SGBD escolhido para prover as persistência dos dados do sistema;

- Maven:[4] incluso no Eclipse, o Maven é uma ferramenta de gerência de projetos que auxilia no download e instalação de dependências e bibliotecas. Inclusive, a documentação do Ninja incentiva seu uso, ao referenciá-lo em sua documentação, e ao sugerir o uso de "arquétipos" (*archetypes*) durante a criação de um projeto Ninja.

Além das tecnologias citadas anteriormente, uma ferramenta mais importante foi escolhida e coloca em foco durante o desenvolvimento do sistema, o framework Ninja.

4.2.1 Ninja Framework

O Ninja foi escolhido, inicialmente, apenas para o papel de controlador frontal porém, após estudos, nota-se sua abrangência como um *framework full stack*, que são muito utilizados na indústria devido a sua abrangência em mútiplas camadas de uma aplicação. Sua filosofia se inspira no conceito de convenção sobre padronização (*Convention over configuration - CoC*). O Ninja possuí diversos *frameworks* embutidos, citando apenas os relevantes ao projeto:

- Hibernate + JPA: Hibernate é um *framework* de Mapeamento Objeto/Relacional que visa a tradução dos objetos usados no sistema para as tabelas utilizadas no banco de dados. A Java Persistence API, utiliza a linguagem JPQL para a consultas ao banco de dados, uma linguagem com maior usabilidade para quem é acostumado com o paradigma Orientado à Objetos;

- FreeMarker:[5] um *template engine*, biblioteca Java capaz de gerar arquivos de texto

[1] http://wildfly.org/
[2] https://www.eclipse.org/
[3] https://www.mysql.com/
[4] https://maven.apache.org/
[5] https://freemarker.apache.org/

(HTML, código fonte, e-mails dentre outros), através do *templates* na extensão `.FTL` seguida da extensão desejada (`.ftl.html` no caso de arquivos HTML), possibilita, inclusive, a internacionalização de páginas, com o uso de variáveis e arquivos que armazenam as traduções correspondentes, para cada idioma que deseja-se dar suporte;

- Google Guice:[6] *framework* de injeção de dependências, sendo esta representada através do uso de anotações (*annotations*).

Apesar do Ninja possuir uma versão do Jetty[7] embutido, ou ser capaz funcionar como um serviço em sistemas operacionais Linux, foi utilizado o WildFly 10 pois, na fase de análise da tecnologia, não foi encontrado sinais de que essas tecnologias afetariam os modelos FrameWeb.

4.3 Modelos FrameWeb para o SCAP

Apesar dos modelos pertencerem à fase de projeto, fase anterior à implementação, a ordem entre a etapa de modelagem com FrameWeb e a desenvolvimento foram invertidas pois, haviam modelagens anteriores nos trabalhos já citados anteriormente, e buscou-se observar possíveis mudanças que o framework Ninja traria para os novos modelos.

Nesta seção, estão expostos modelos referentes à funcionalidades implementadas.

4.3.1 Modelo de Entidades

Originalmente chamado de modelo de domínio, o modelo de entidades do FrameWeb é um diagrama de classes da UML que representa os dados da solução, a partir dele são implementados os objetos para que sejam manipulados e persistidos pela camada de acesso a dados.

O modelo de entidades, representado na Figura 6, é criado a partir do diagrama de classes, da etapa de análise de requisitos. Suas cardinalidades continuam indicando os números mínimo e máximo dentre duas classes (FALBO, 2014). Este modelo pode ser reaproveitado do projeto de arquitetura de Prado (2015), uma vez que ambas implementações usam a combinação de Frameworks Hibernate mais JPA para a persistência de suas classes de domínio em banco de dados relacionais.

Os tipos dos atributos se adequam aos tipos suportados pela plataforma *Java* e pelas classes enumeradas. Tais tipos: **TipoParentesco, SituacaoSolic, TipoAfast, Onus e TipoParecer** foram implementados e modelados e podem ser observados na Figura 7. A restrição de atributo *não-nula*, comum às variáveis **dataSolic, dataIniAfast, nome,**

[6] https://github.com/google/guice
[7] https://www.eclipse.org/jetty/

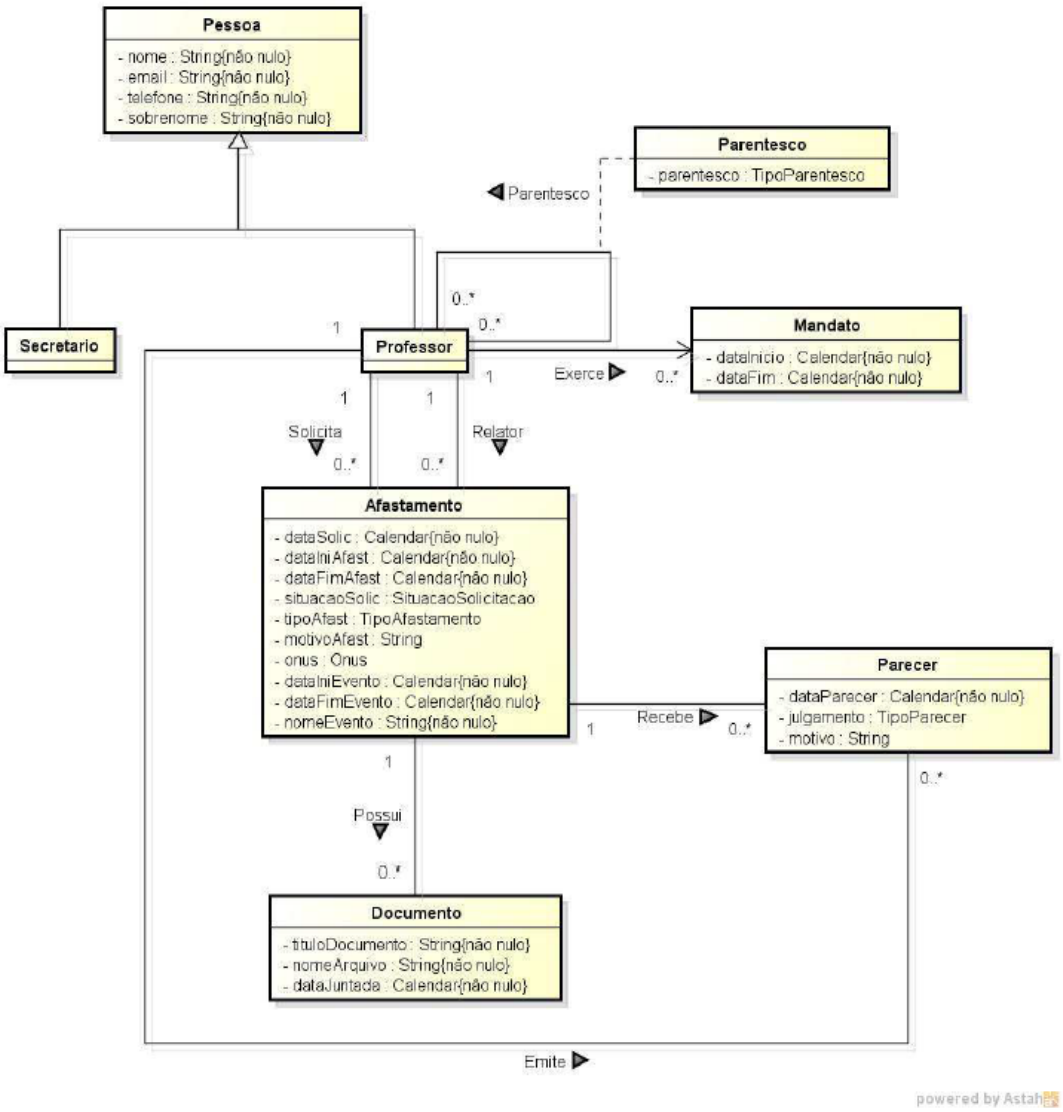

Figura 6 – Modelos de Entidades implementadas no SCAP(PRADO, 2015).

dataParacer, dentre outras, apenas não está sendo exibida, devido a uma configuração do FrameWeb Editor.

Apesar da mudança para o Ninja, não houve alteração nesse modelo uma vez que Prado (2015) também usou o framework *JPA*. Uma etapa muito comum para aplicações que utilizam o *JPA* como *framework* de mapeamento objeto/relacional, é a de configuração do **persistence.xml**. Nela, todas as classes de entidades precisam estar declaradas para que sejam criadas as devidas tabelas no banco de dados.

4.3.2 Modelo de Navegação

Este modelo, representado na Figura 8, tem por objetivo explicitar todos os componentes que formam a camada lógica da apresentação, tais como: páginas Web. formulários HTML e classes de ação dos controladores fontais. Partes do sistemas por

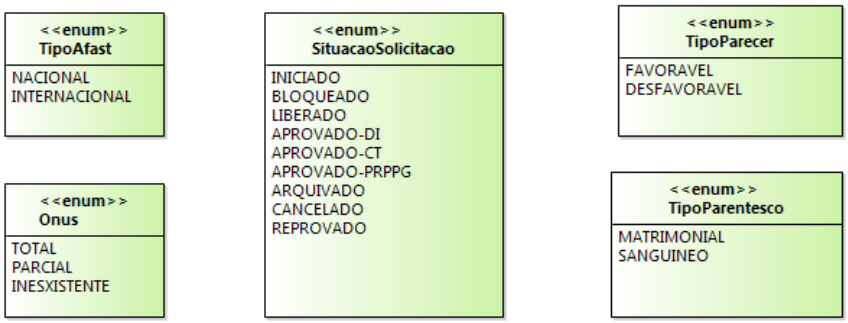

Figura 7 – Tipos Enumerados do SCAP

onde ocorre a interação com o usuário.

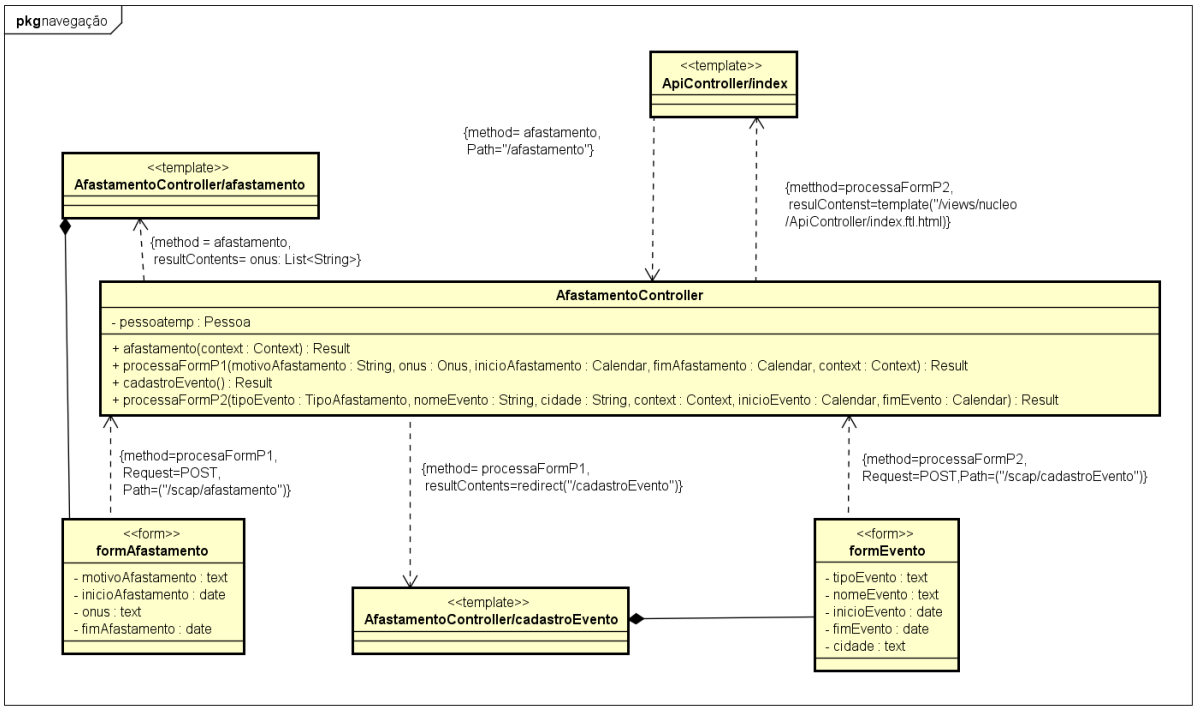

Figura 8 – Modelo de navegação referente ao caso de uso “Cadastras Afastamento”

O processo de solicitar um afastamento inicia-se quando o servidor recebe uma requisição do tipo (Request=*GET*) para o endereço "scap/afastamento", isso faz com que o arquivo de rotas (**Routes.java**, contida no pacote **conf** da figura 20), execute a rota mapeada nele, nesse caso, chamando o controlador e o método *AfastamentoController::afastamento*, para que execute o *template* e gere a página com o formulário. Por padrão, é definido que requisições do tipo **GET** é o valor padrão.

Após seu preenchimento, este envia junto de seus dados, dois guias para encontrar a rota referente ao método que irá processá-lo: a url (representada no modelo pelo elemento **Path**="/afastamento"da dependência) e o tipo da requisição (representado pelo elemento **Request=POST**). Ainda no caso em questão de uma rota ser "method"representa o

método,**processaFormP1**, que foi encontrado no arquivo **Routes.java**. Diferentemente do caso anterior, chama o *«template»* **cadastroEvento**. A página web, agora montada, faz uma requisição do tipo **POST** para que seu formulário seja processado. E finalmente, o método **processaFormP2** processa o formulário, e carrega a página inicial.

No início do formulário **cadastroAfastamento**, é indicado que os dados do formulário serão enviados para o mesmo endereço "scap/afastamento", agora porém, via requisição do tipo POST. Ao preencher e enviar o formulário, o arquivo de rotas, que diferencia cada rota pelo tipo de requisição, e apesar do mesmo endereço, e chama o método *processaFormP1()*. Este método como retorno, caso não haja desvios no fluxo do programa, *Results.redirect("/scap/cadastroevento");*, assim uma nova página é enviada ao navegador. O segundo formulário fará uma requisição para a mesma página ("/scap/cadastroevento") mas via POST, chamando o método *processaFormP2()*, e este redireciona para a página inicial.

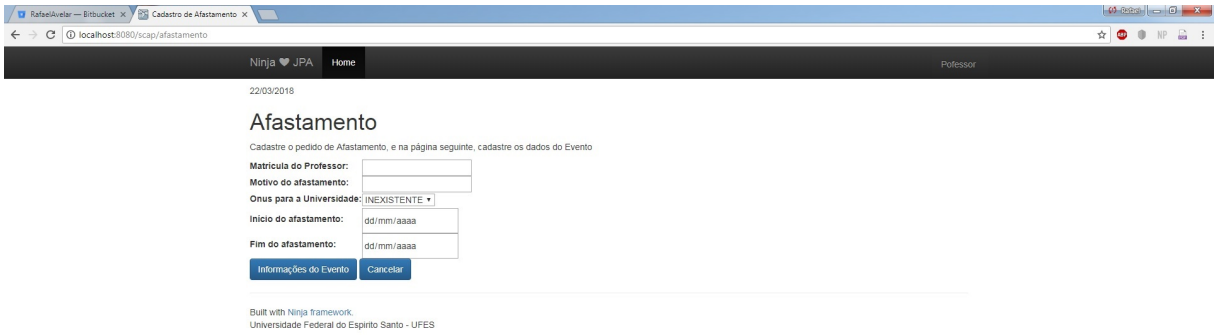

Figura 9 – Imagem da tela de cadastro de afastamento

Outro modelo apresentado, é referente ao caso de login de usuário, como pode ser visto na figura 12. Nela podemos observar a utilização da palavra reservada "result". Os tipo de result definidos para implementação estão descritos na figura 11. Foi estabelecida uma classe de ação para este modelo, o tipo *"result"*, sugerido inicialmente por (SOUZA, 2007). É importante não confundir *result* com *resultsContents*, uma vez que o primeiro representa uma sinalização do controlador à página, de capacidade limitada, usado geralmente como uma *flag*. Já o segundo, é o retorno propriamente dito de um método controlador, mais completo.

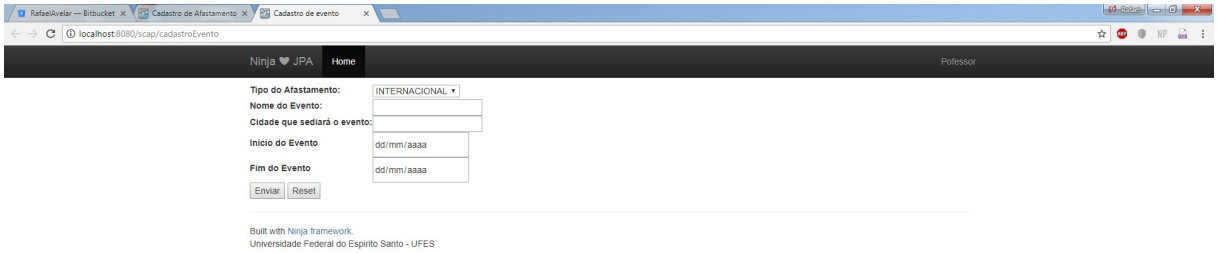

Figura 10 – Imagem da tela de cadastro das informções do evento

resultType	Descrição
sucess	sucesso na operação.
error	erro na operação.

Figura 11 – tabela com os tipos de resultados possíveis.

4.3.3 Modelo de Aplicação

O modelo de aplicação é o responsável por reunir as classes que implementam a lógica de negócio do sistema, descrita nos casos de uso e suas devidas dependências com relação aos demais modelos do sistema (DUARTE, 2014). Essas dependências são gerenciadas pelo framework de injeção de dependências *Guice*, via anotações. As classes da camada de aplicação recebem nomes oriundos das entidades modeladas anteriormente, descritas como **<Apl><nome da classe de entidade>** e sua implementação recebe o segue o padrão **<Apl><nome da classe de entidade><Imp>**. Essa ligação entre implementações e interfaces precisa estar especificada no método **configure()**, do **Module.java**, como exemplificado na Figura 13.

Neste modelo, representado pela Figura 14, percebe-se uma dificuldade ao tentar manter duas ideias: manter clara a diferenciação entre funcionalidades voltadas para atender aos casos de uso de professores, e aos casos de uso dos secretários; e evitar a repetição de código, em função de atender o requisito de durabilidade, citado por Pressman

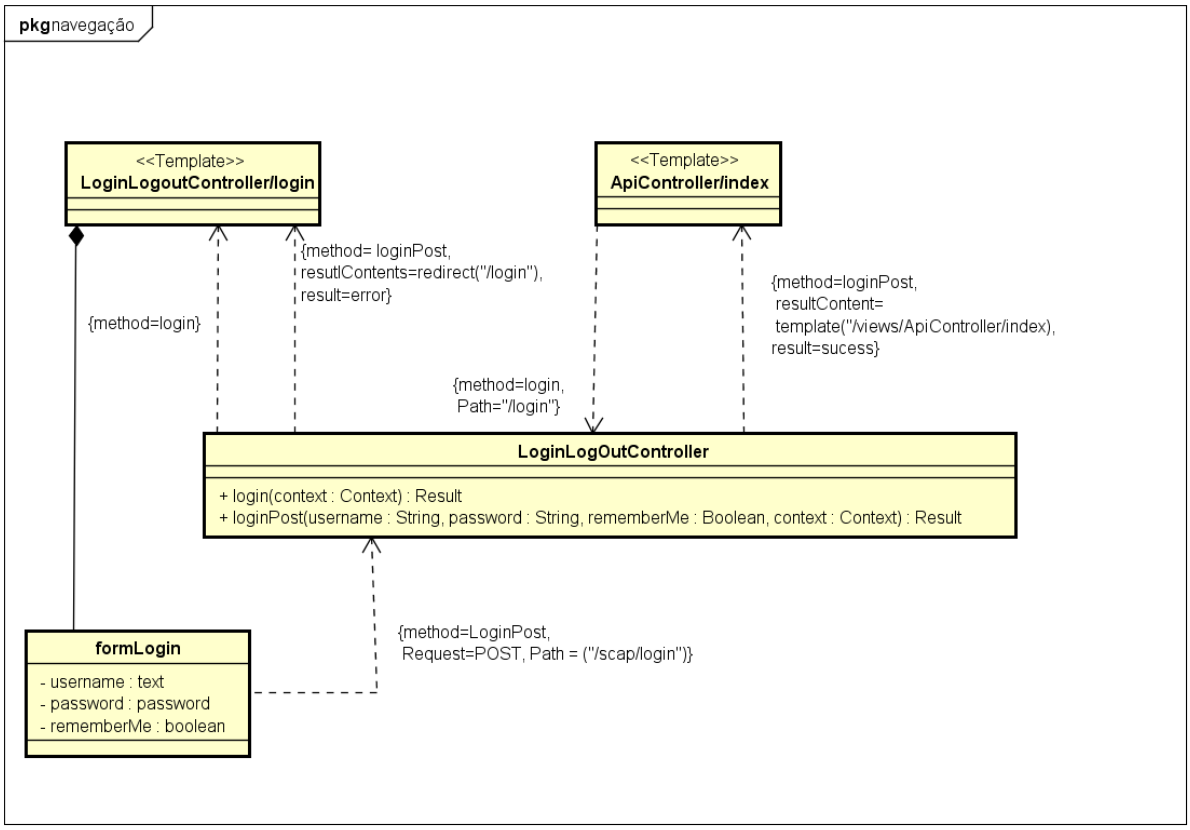

Figura 12 – Modelo de Navegação Referente ao Login de usuários.

(2011). Na imagem em questão foram modeladas apenas as dependências relacionadas ao caso de uso "Solicitar Afastamento", para que a imagem pudesse continuar legível.

4.3.4 Modelo de Persistência

O modelo de persistência, Figura 15, tem por objetivo indicar como a camada de acesso a dados é implementada, semelhante ao modelo de aplicação, uma vez que o foco dele é indicar a ligação entre interfaces e implementações, e como essas interfaces são utilizadas pela camada de aplicação. O modelo segue padrão de projeto DAO, impedindo que as classes da lógica de aplicação tenha acesso ao *frameworks ORM*, mantendo a possibilidade de substituição deste.

Apesar de precisar-se das entidades para a criação de tabelas, de acordo com a configuração no arquivo **persistence.xml**, o uso de *frameworks ORM*, deixa esta tarefa transparente ao desenvolvedor. Portanto, neste modelo não existe menção direta à classes do modelo de entidades, essa menção é feita indiretamente durante a nomenclatura das estruturas. As interfaces DAO recebem seus nomes de acordo com a lógica: **<nome da entidade><DAO>**, enquanto as classes de persistência seguem a lógica: **<tecnologia ORM><nome da entidade><DAO>**.

```
package conf;

import com.google.inject.AbstractModule;

@Singleton
public class Module extends AbstractModule {

    @Override
    protected void configure() {

        bind(StartupActions.class);

        bind(AplAfastamento.class).to(AplAfastamentoImp.class);
        bind(AfastamentoDAO.class).to(JPAafastamentoDAO.class);
        bind(AfastamentoDAO.class).to(JPAafastamentoDAO.class);
        bind(PessoaDAO.class).to(JPApessoaDAO.class);

    }

}
```

Figura 13 – Imagem referente à classe **Module**.

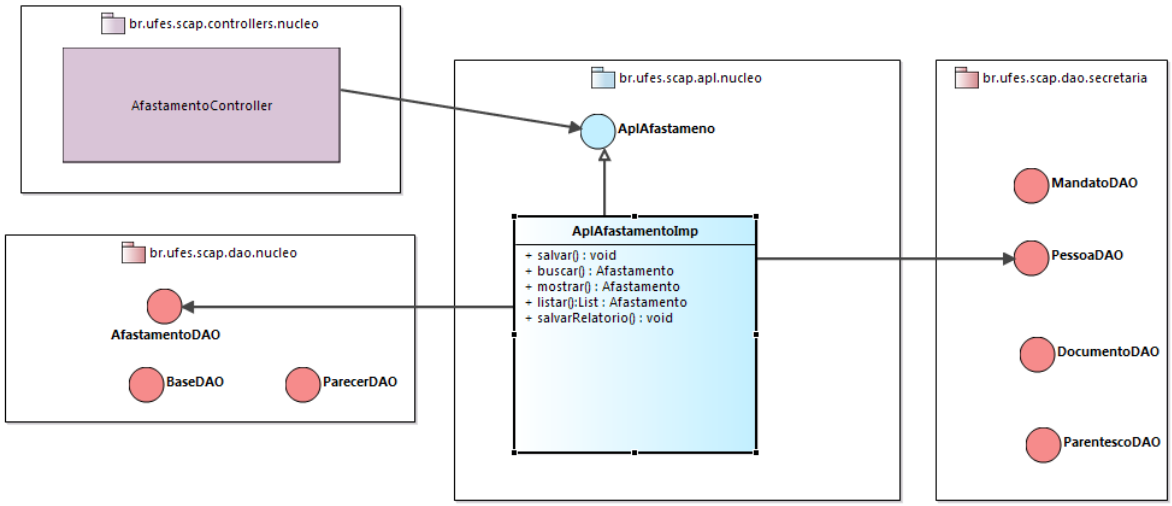

Figura 14 – Modelo de aplicação do caso de uso "Solicitar Afastamento".

4.4 Detalhes de Implementação

Na etapa de desenvolvimento de um sistema Web ocorre a produção efetiva da solução do problema. No caso do paradigma orientado a objetos, classes e métodos são implementados. Nela, também, percebe-se com maior facilidade as diferenças e semelhanças de um framework a outro, principalmente quando estes estão sobre uma tecnologia (seja ela: paradigma de linguagem, API e/ou plataforma de desenvolvimento). Umas das maneiras que frameworks se diferencia de outros é através de convenções: escolhas tomadas por

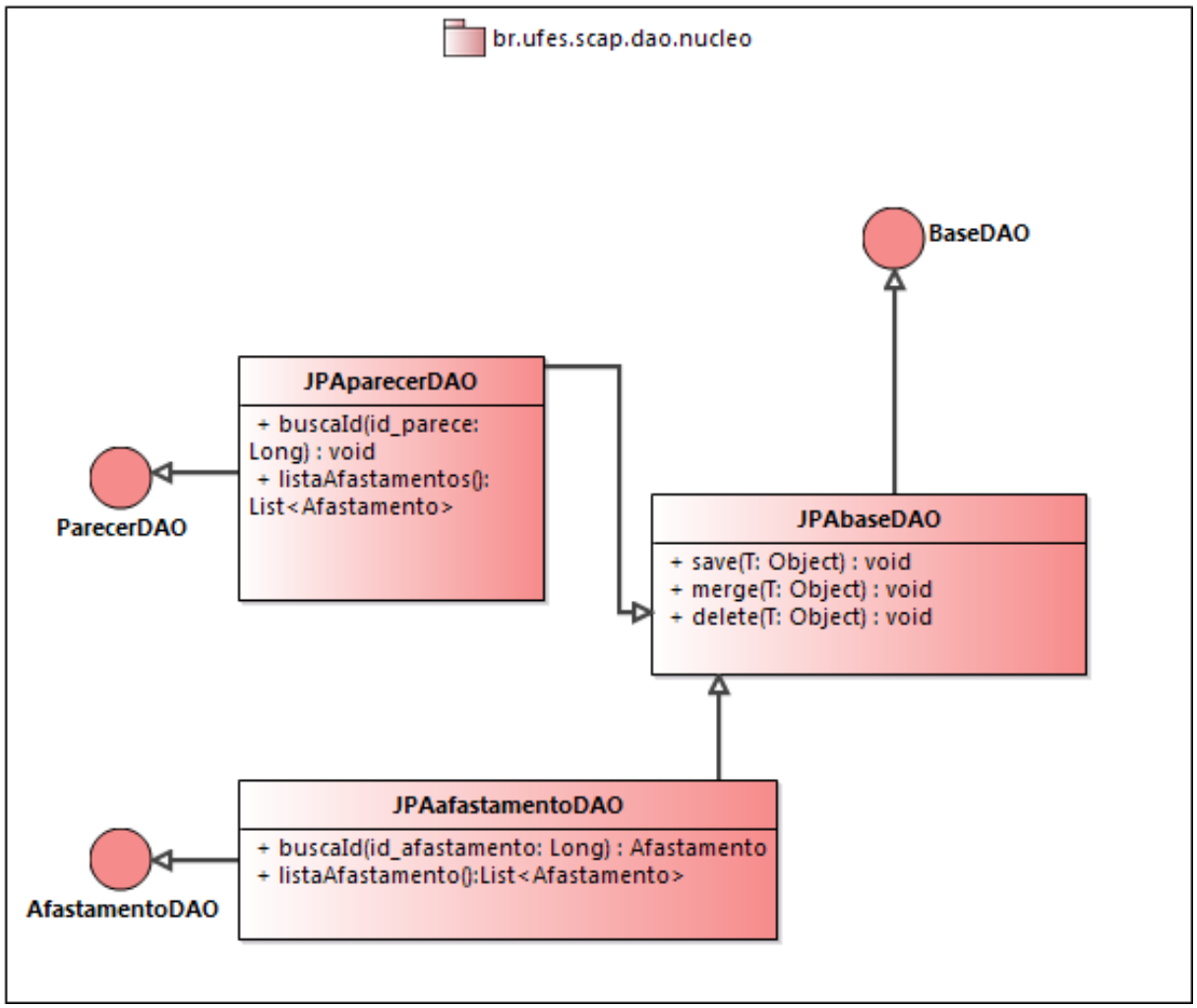

Figura 15 – Modelo de persistência do SCAP do pacote dao.nucleo

parte dos desenvolvedores de dado framework. Algumas das decisões tomadas pelos desenvolvedores do Ninja, e que foram observadas durante a fase de implementação, foram:

- O Ninja impõem que todos os métodos responsáveis pelo carregamento de páginas, possuam o retorno do tipo **Result**, nele estão os resultados do processamento dos métodos controladores. Podem conter envios de dados para a página via JSON(exemplificado na figura citada acima, quando é carregado o template "afastamento"), pode redirecionar para outras rotas (quando o template CadastroEvento é carregado), e até fugindo de uma própria convenção do Ninja, ao permitir carregar um template sem a necessidade de passar pelo arquivo de rotas (quando o se usa ao retorno da página index).

- A divisão do sistema em pacotes com nomes específicos: uma vez que o Ninja segue o padrão MVC. Na figura 16 representam os pacotes do sistema e onde os frameworks utilizados podem ser encontrados. Ele estabelece os nomes dos pacotes para cada camada de um sistema. **Controllers** designada para a localização das classes con-

troladoras frontais, **models** referente as classes mapeadas no diagrama de entidades (Figura 6) e **dao** que armazena a lógica de acesso ao banco de dados, representada pelo modelo de persistência (Figura 15). Curiosamente, não é estabelecido um pacote para classe de aplicação, o que pode-se observar nos exemplos em sua documentação online, são que as funcionalidades da classe de aplicação geralmente estão implementadas nos controladores, com estes, inclusive, acessando pacotes DAO mas, não há problemas ou configurações extras a serem feitas para a inclusão de novos pacotes;

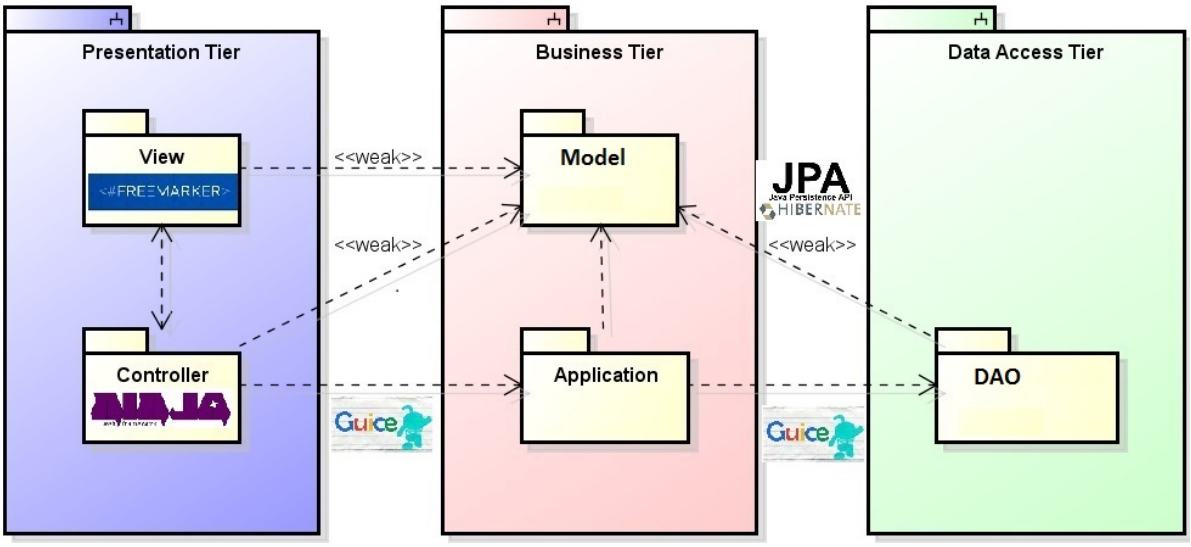

Figura 16 – Divisão das Camadas do SCAP com representações das localizações dos Frameworks.

- Como o Ninja segue o modelo *REST* de desenvolvimento de aplicações Web, ele requer que qualquer comunicação feita pelo navegador esteja mapeada na forma de rota. Essa rota deve conter uma das seguintes requisições: GET, POST, PUT, OPTIONS, HEAD, e DELETE. Entretanto, o Ninja permite que requisições adicionais sejam implementadas através de métodos HTTP. Essas configurações devem ser feitas no arquivo **Routes.java**, pertencente ao pacote **conf**.

- Outra funcionalidade que vale ser ressaltada ao utilizar Ninja, é sua capacidade de poder inserir os tipos enumerados nos parâmetros de seus métodos. Os controladores frontais aceitam esses tipos, enquanto que o formulário HTML envia um tipo **select** ao controlador. Tipos mais elaborados podem ser enviados para criação da página devido ao framework FreeMarker conseguir acessar dados dentro de objetos, contando que estes sejam enviados ao *template* pelo controlador, via **Result**.

A figura 17 mostra a tela de login de usuários. Podemos reparar na atuação do Framework decorador FreeMarker, mesmo que simples, com elementos como o rodapé e o cabeçalho, uma vez que eles são um mesmo arquivo que é reaproveitado por diferentes templates do sistema.

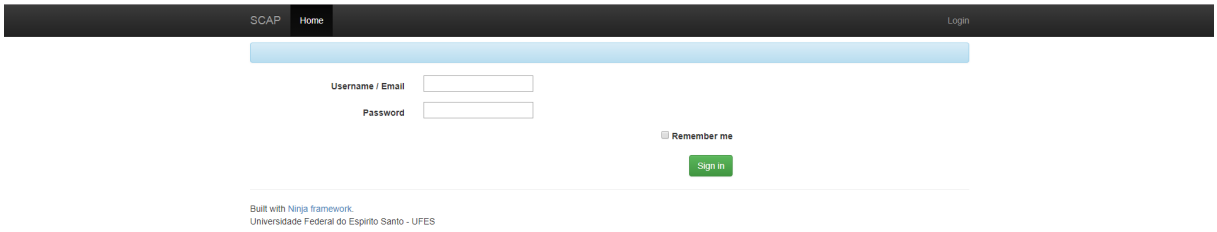

Figura 17 – figura referente à página de Login de usuário.

```
public class AfastamentoController{
    @Inject
    AplMandato aplMandato;
    @Inject
    AplAfastamento inserir_afastamento;
    @Inject
    AplPessoa aplPessoa;
```

Figura 18 – figura que representa trecho de código do *AfastamentoController*.

A figura 18 mostra as notações utilizadas pelo *Google Guice*. *@SessionScoped* define que a classe **CadastramentoController** é instanciada por sessão de usuário. Enquanto *@Inject* responsável por injetar objetos necessários, por exemplo, o objeto do tipo *AplAfastamento* será gerenciado pelo Ninja.

4.4.1 Divisão em Subsistemas

O SCAP foi divido por Prado (2015) em dois subsistemas, isso é representado através da hierarquia aplicada na divisão dos pacotes de seu projeto. Isso pode ser observado na Figura 19.

Em seu trabalho, Prado (2015) dividiu o SCAP nos subsistemas **br.ufes.scap.nucleo** e **br.ufes.scap.secretaria**, com o interesse de separar as funcionalidades que atendem

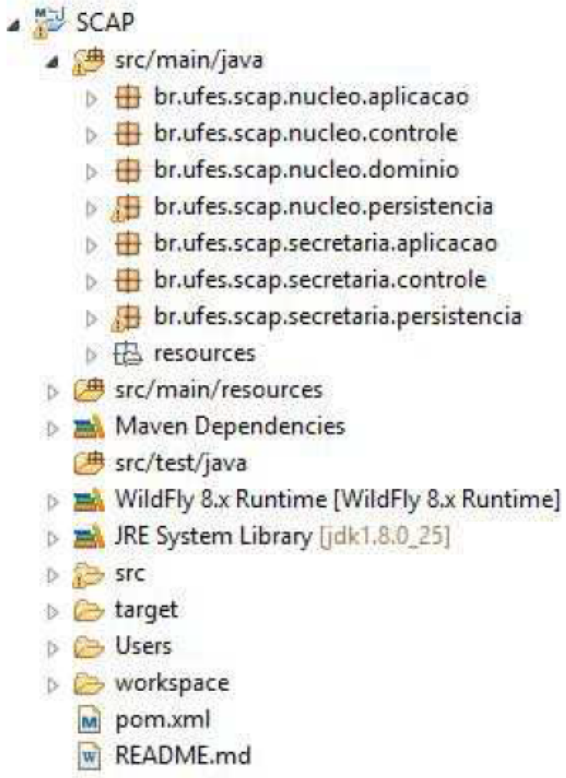

Figura 19 – Estrutura da pasta do sistema implementado por (PRADO, 2015)

aos casos de uso, respectivamente, dos professores e dos secretários. Assim, cada subsistema possui seus pacotes de controle, aplicação e persistência. Contudo, havia muita interdependência entre eles, algo que não colaborava com a divisão.

Pensando no intuito de Prado (2015) e na imposição feita pelo Ninja, de padronizar a localização das camadas do sistema, é apresentada alternativa cabível dentro da configuração do Ninja: cada camada do Padrão MVC possui dois pacotes, um sub-pacote chamado **nucleo** e outro chamado **secretaria**.

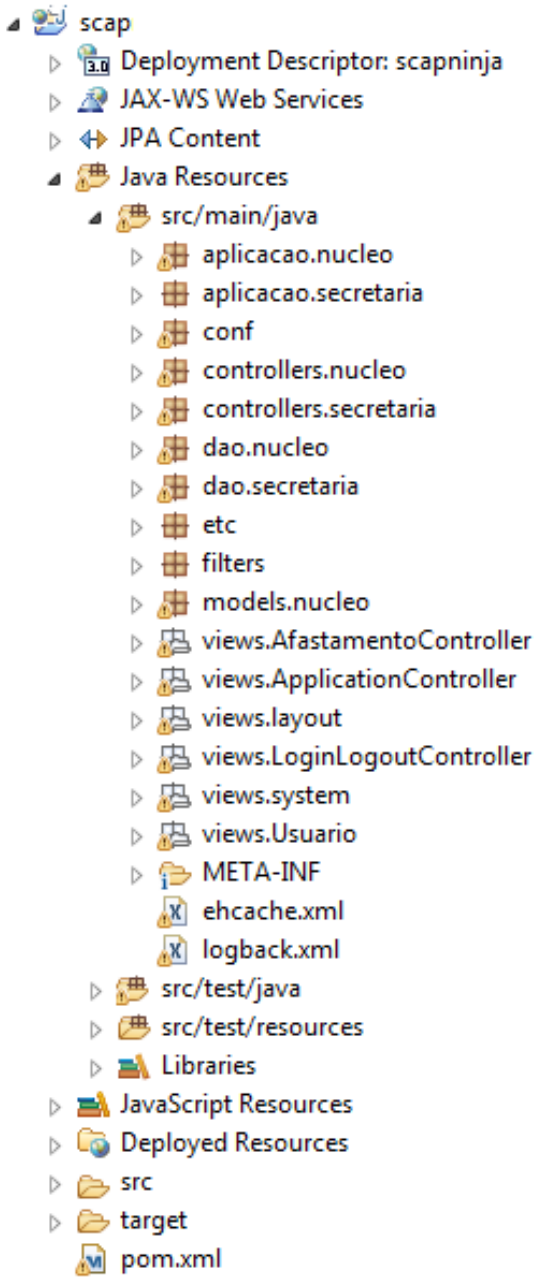

Figura 20 – Estrutura de pastas da nova implementação do SCAP com o framework Ninja.

5 Considerações Finais

Atualmente, para a criação de *WIS* de alta qualidade são utilizadas técnicas de reutilização de componentes já existentes (PRESSMAN, 2011). Os frameworks são alguns desses componentes(assim como as plataformas na qual eles se baseiam), por isso que propostas como a de Souza (2007) são tão importantes para o desenvolvimento tecnológico. Seu trabalho ao categorizar os frameworks ajuda desenvolvedores, conferindo-lhe uma visão melhor das ferramentas que escolheu para o sistema que pretende desenvolver. Os modelos de projeto propostos guiam a implementação destas novas aplicações, ajudando no desacoplamento das camadas do sistema e conferindo eficiência à fase de desenvolvimento.

Tal fato pode ser evidenciado quando houveram frameworks novos inéditos em implementações anteriores do SCAP em conjunto com frameworks já utilizados por Prado (2015). Os trabalhos anteriores de Duarte (2014), e posteriormente, Prado (2015) foram valiosos devido ao amadurecimento da documentação do SCAP, assim como mais exemplos de uso do método FrameWeb.

O Framework ninja se mostrou uma surpresa. Sua proposta de entregar um conjunto de frameworks já estabelecidos no mercado, pré-configurados e disponibilizá-los através do uso de arquétipos. Isso se mostrou interessante pois auxilia na etapa de configuração e pode diminuir o esforço gasto em possíveis configurações a serem realizadas. Essa iniciativa contribui com agilidade e confiabilidade na aplicação e desenvolvimento. E algo de novo que o Ninja pode trazer ao FrameWeb foi a interação com padrão REST(FIELDING; TAYLOR, 2000).

5.1 Propostas Para o FrameWeb

O FrameWeb tem o potencial para ser uma metodologia padrão no desenvolvimento de *WIS*, como contribuição para este fim, as seguintes propostas são feitas neste trabalho:

- A criação das notações **Request** e **path**: presente nos modelos de navegação, estes atributos considerados essenciais na dependência entre um elementos decoradores e o controlador frontal;
- a padronização dos nomes dos métodos dos controlares frontais e das páginas na qual são responsáveis;
- Criação do atributo **resultContents**: Esse elemento representa o resultado propriamente dito, por onde ele podem redirecionar do caminho padronizado pela

nomenclatura de dos templates e inseri valores, enviar mensagens de erro ou ativar e desativar conteúdos que as páginas irão exibir.

5.2 Trabalhos Futuros

Propostas foram sugeridas na seção anterior, porém existem muitos outros Frameworks da platadorma Java EE que ainda não foram testados em conjunto com o FrameWeb, e para que isso aconteça mais estudos sempre serão bem-vindos.

Referências

ALUR, D. et al. *Core J2EE Patterns (Core Design Series): Best Practices and Design Strategies*. 2. ed. Mountain View, CA, USA: Sun Microsystems, Inc., 2003. ISBN 0131422464. Citado na página 24.

BOURQUE, P.; FAIRLEY, R. E. et al. *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. [S.l.]: IEEE Computer Society Press, 2014. Citado 2 vezes nas páginas 17 e 18.

DUARTE, B. B. *Aplicação do Método FrameWeb no Desenvolvimento de um Sistema de Informação na Plataforma Java EE 7*. Monografia (Projeto de Graduação) — Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, ES, Brasil, 2014. Citado 10 vezes nas páginas 8, 9, 13, 14, 25, 26, 30, 31, 37 e 46.

FALBO, R. d. A. *Engenharia de Software*. 2014. 144 p. Notas de Aulas voltadas para a Disciplina Projeto de Sistemas de Software , ministrada entre agosto e dezembro de 2014, na Universidade Federal do Espírito Santo, Departamento de Informática, Curso de Cinência da Computação. Disponível em: <https://inf.ufes.br/~falbo/files/ES/Notas_Aula_Engenharia_Software.pdf>. Citado 2 vezes nas páginas 28 e 33.

FALBO, R. d. A. *Projeto de Sistemas de Software*. 2017. 135 p. Notas de Aulas voltadas para a Disciplina Projeto de Sistemas de Software , ministrada entre agosto e dezembro de 2017, na Universidade Federal do Espírito Santo, Departamento de Informática, Curso de Cinência da Computação. Disponível em: <https://inf.ufes.br/~falbo/files/PSS/Notas_Aula_Projeto_Sistemas_2017.pdf>. Citado na página 18.

FAYAD, M.; SCHMIDT, D. C. Object-oriented application frameworks. *Communications of the ACM*, ACM, v. 40, n. 10, p. 32–38, 1997. Citado na página 13.

FIELDING, R. T.; TAYLOR, R. N. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine Doctoral dissertation, 2000. v. 7. Citado 2 vezes nas páginas 22 e 46.

FOWLER, M. *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0321127420. Citado 5 vezes nas páginas 8, 19, 20, 21 e 23.

MACKINNON, T.; FREEMAN, S.; CRAIG, P. Endo-testing: unit testing with mock objects. *Extreme programming examined*, p. 287–301, 2000. Citado na página 22.

MENDES, A. Arquitetura de software: desenvolvimento orientado para arquitetura. *Rio de Janeiro: Campus*, 2002. Citado na página 18.

NASCIMENTO, B. M. do. *SAE - Sistema de Acompanhamento de Egressos*. [S.l.], 2016. Citado na página 13.

OLSINA, L.; LAFUENTE, G.; ROSSI, G. Specifying quality characteristics and attributes

for websites. In: *Web Engineering*. [S.l.]: Springer, 2001. p. 266–278. Citado na página 17.

PRADO, R. C. do. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework VRaptor 4*. Monografia (Projeto de Graduação) — Departamento de Informática, Universidade Federal do Espírito Santo, Vitória, ES, Brasil, 2015. Citado 15 vezes nas páginas 8, 9, 13, 14, 25, 26, 27, 28, 29, 30, 33, 34, 43, 44 e 46.

PRESSMAN, R. S. *Engenharia de software*. [S.l.]: Makron books Sao Paulo, 2011. v. 7. Citado 5 vezes nas páginas 13, 16, 18, 38 e 46.

SALVATORE, T. R. *AlocaWeb e BibLattes - Módulos do sistema Marvin*. [S.l.], 2016. Citado na página 13.

SOUZA, V. E. S. *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, 2007. Citado 11 vezes nas páginas 8, 13, 16, 19, 20, 21, 23, 24, 30, 37 e 46.

WOJCIECHOWSKI, J. et al. Mvc model, struts framework and file upload issues in web applications based on j2ee platform. In: *Proceedings of the International Conference Modern Problems of Radio Engineering, Telecommunications and Computer Science, 2004.* [S.l.: s.n.], 2004. p. 342–345. Citado na página 19.

Apêndices

Rafael: dependencias removidas, segundo a reunião