Mateus Tassinari Ferreira

Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Wicket e Tapestry

Vitória, ES

2018

Mateus Tassinari Ferreira

Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Wicket e Tapestry

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

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Departamento de Informática

Orientador: Prof. Dr. Vítor E. Silva Souza

Vitória, ES

2018

Mateus Tassinari Ferreira

Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Wicket e Tapestry

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

Trabalho aprovado. Vitória, ES, 29 de Novembro de 2018:

Prof. Dr. Vítor E. Silva Souza
Orientador

Prof. Dr. Davidson Cury
Universidade Federal do Espírito Santo

Félix L. Zanetti
Instituto Federal do Espírito Santo

Vitória, ES
2018

Agradecimentos

Agradeço primeiramente a Deus por ter me guiado com sabedoria e feito com que eu chegasse até aqui.

Também agradeço a minha família, em especial meus pais e meu irmão, por terem me amado com carinho a minha vida toda e que me ajudaram a enfrentar todas as dificuldades e barreiras.

Aos meus amigos que me acompanharam e sem eles nada disso seria possível. E por último, também os professores que tive durante a graduação pelos ensinamentos e incentivo, em especial meu orientador Vítor.

"A imaginação é mais importante que a ciência, porque a ciência é limitada, ao passo que a imaginação abrange o mundo inteiro."

(Albert Einstein)

Resumo

As primeiras aplicações *WebApps* eram, geralmente, desenvolvidas sem considerar princípios de Engenharia de Software. Entretanto, com a demanda por maior rapidez e qualidade dessas aplicações, surgiu uma necessidade de abordagens e técnicas que levassem em conta as características do ambiente, os cenários operacionais e a vasta variedade de perfis de usuários. Então, foi criada a Engenharia Web.

Foi proposto então o método FrameWeb (*Framework-based Design Method for Web Engineering*), em que é sugerida a utilização de uma série de *frameworks* para agilizar o desenvolvimento de uma *WebApp*, juntamente com várias recomendações para aumentar a agilidade nas principais fases de desenvolvimento. Na proposta inicial foi utilizado apenas um *framework* de cada categoria, de modo que não é possível saber se o método é aplicável quando um conjunto diferente de *frameworks* são usados.

Para testar a força do método foi desenvolvido uma *WebApp*, o SCAP, com um conjunto diferente de *frameworks*, baseados na plataforma Java EE 7. O objetivo deste trabalho é reimplementar o SCAP testando o método FrameWeb com os *frameworks* controladores Tapestry e Wicket ao invés dos *frameworks* utilizados em experimentos anteriores com o método.

Palavras-chaves: WebApp. FrameWeb. Engenharia Web. Tapestry. Wicket.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UFES	Universidade Federal do Espírito Santo
SCAP	Sistema de Controle de Afastamento de Professores
UML	*Unified Modeling Language*
HTTP	*HyperText Transfer Protocol*
HTML	*HyperText Markup Language*
CGI	*Common Gateway Interface*
WIS	*Web-based Information Systems*
MVC	*Model-View-Controller*
AOP	*Aspect Oriented Programming*

Sumário

1 Introdução

No início da *World Wide Web*, a infraestrutura de software por trás dos servidores dava apoio somente a páginas estáticas, ou seja, documentos entregues ao cliente como retorno a requisições HTTP. A partir de 1993, com o surgimento da tecnologia CGI e de linguagens de programação para a *Web*, os servidores ficaram mais poderosos, admitindo o desenvolvimento de aplicações de negócio, em que grandes sistemas começaram a ser desenvolvidos para a *Web*.

Nesse momento surge o conceito de *WebApp*, que consiste em um conjunto de páginas *Web* que interagem com o usuário, fornecendo e processando informações. Exemplos são websites informativos, interativos, aplicações transacionais ou baseadas em fluxos que executam na *Web*, ambientes colaborativos de trabalho, comunidades online e portais (GINIGE; MURUGESAN, 2001). Nesta monografia, no entanto, iremos focar em uma categoria de aplicações *Web*, a qual chamamos Sistemas de Informação Baseados na Web (*Web-based Information Systems* - WISs), que são como sistemas de informação tradicionais, porém disponíveis na *Internet*, como ambientes cooperativos, sistemas de gerencia empresarial, dentre muitos outros.

As primeiras aplicações *WebApps* eram, geralmente, desenvolvidas de maneira *ad hoc*, sem considerar princípios de Engenharia de Software. Entretanto, com a demanda por maior rapidez e qualidade dessas aplicações, surge uma necessidade de abordagens disciplinadas de Engenharia de Software. Tais abordagens e técnicas devem levar em conta as características do ambiente e os cenários operacionais e a vasta variedade de perfis de usuários, que adicionam desafios ao desenvolvimento de aplicações *Web*. Nasceria, assim, a Engenharia Web (PRESSMAN, 2011).

A Engenharia Web pode ser definida como o estabelecimento e uso de princípios científicos, de engenharia e de gerência, além de abordagens sistemáticas, para o bem-sucedido desenvolvimento, implantação e manutenção de aplicações e sistemas *Web* de alta qualidade (MURUGESAN et al., 1999).

Neste novo ramo, muitos métodos, *frameworks* e linguagens de programação para desenvolvimento de aplicações *Web* têm sido propostos. Tecnologias para codificação de *WebApps* também se desenvolveram bastante. O uso de *frameworks* ou arquiteturas fundamentadas em *containers* para fornecer uma sólida base para o desenvolvimento e implantação de aplicações para a *Web* é quase que obrigatório atualmente. Essa infraestrutura geralmente inclui uma arquitetura MVC (*Model-View-Controller*) (GAMMA et al., 1994), interceptadores AOP (*Aspect Oriented Programming*, Programação Orientada a Aspectos) (RESENDE; SILVA, 2005), um mecanismo de injeção de dependências (FOWLER,

2004), mapeamento objeto/relacional automático para persistência de dados (BAUER; KING, 2005) e outras facilidades. O uso desses *frameworks* reduz consideravelmente o tempo de desenvolvimento de um projeto por reutilizar código já desenvolvido, testado e documentado.

Recentemente, foi proposto o método FrameWeb (*Framework-based Design Method for Web Engineering*) (SOUZA, 2007), em que é sugerida a utilização de uma série de *frameworks* para agilizar o desenvolvimento de uma *WebApp*, juntamente com várias recomendações para aumentar a agilidade nas principais fases de desenvolvimento (requisitos, análise, projeto e outras).

O método oferece 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, dessa forma, a comunicação entre os projetistas e programadores durante o desenvolvimento da aplicação. Na proposta original de FrameWeb, no entanto, não foram feitos experimentos com diferentes instâncias de *frameworks*.

Para analisar a eficácia do método FrameWeb, Duarte (2014) desenvolveu uma *WebApp* — um Sistema de Controle de Afastamentos de Professores, ou SCAP, para auxiliar um departamento de universidade a controlar solicitações de afastamento de seus professores efetivos. A partir dos requisitos levantados por Duarte (2014) e revisados por Prado (2015), o objetivo deste trabalho é desenvolver o SCAP em duas novas versões, utilizando diferentes *frameworks Web*, com o intuito de analisar a eficácia do método FrameWeb em diferentes contextos, sugerindo melhorias nos modelos, quando necessário.

1.1 Objetivos

O objetivo deste trabalho é a aplicação do método FrameWeb (SOUZA, 2007) em novas implementações do SCAP, utilizando os *frameworks Web* Tapestry e Wicket, a partir dos requisitos levantados por Duarte (2014) e revisados por Prado (2015), avaliando a aplicabilidade e eficácia do método em diferentes contextos. Podemos subdividir este objetivo nos seguintes sub-objetivos:

- O desenvolvimento do SCAP a partir de seus requisitos utilizando o *frameworks* Tapestry e Wicket;
- A documentação do projeto arquitetural referente às implementações acima, utilizando FrameWeb;
- Análise da aplicabilidade de FrameWeb neste contexto e sugestões de melhoria, quando cabível, baseadas na experiência adquirida com o uso desses *frameworks*.

1.2 Metodologia

O trabalho vai ser dividido nas seguintes atividades:

1. Pesquisa: estudo do método FrameWeb e dos diversos *frameworks* nos quais ele foi inicialmente testado, bem como dos *frameworks* propostos em projetos exemplos;
2. Análise do Problema Proposto: análise dos requisitos do SCAP (DUARTE, 2014);
3. Projeto: a partir dos resultados obtidos no item acima, construção do projeto arquitetural de duas novas versões do SCAP, utilizando FrameWeb (SOUZA, 2007);
4. Codificação: seguindo a arquitetura projetada, implementação do SCAP utilizando os *frameworks* propostos;
5. Apresentação do Projeto: apresentação final, na qual o projeto e a ferramenta SCAP com diferentes implementações serão entregues e demonstradas.

1.3 Organização da Monografia

Essa monografia é construída em 5 capítulos, incluindo a introdução:

- **Capítulo 2** – Fundamentação Teórica: realiza-se um levantamento dos principais temas abordados ao longo deste trabalho que são: Engenharia Web, *Frameworks* e o método FrameWeb;
- **Capítulo 3** – SCAP: realiza-se uma apresentação do sistema SCAP, ou seja, quais seus objetivos e funcionalidades;
- **Capítulo 4** – Projeto e Implementação: apresenta o resultado da aplicação do FrameWeb no projeto, podendo ser vistos os modelos propostos e a implementação dos mesmos com os *frameworks* utilizados;
- **Capítulo 5** – Considerações Finais: apresenta as conclusões desse trabalho, analisa o desenvolvimento do SCAP em diferentes *frameworks* e as expectativas para trabalhos futuros.

2 Fundamentação Teórica

Este capítulo apresenta os principais conceitos teóricos que fundamentaram o desenvolvimento do sistema SCAP e está organizado em três seções. A Seção 2.1 aborda a Engenharia Web, destacando os principais conceitos e processos utilizados. A Seção 2.2 apresenta o método FrameWeb. A Seção 2.3 fala sobre os tipos de *frameworks* existentes, classificando os que serão utilizados no trabalho.

2.1 Engenharia Web

No início do surgimento da *Web* até hoje, ela passou de um conjunto de páginas estáticas desenvolvidas de maneira *ad hoc* para ser uma das principais plataformas de comunicação.

O que vemos atualmente é que a *Internet* é parte presente em nossas vidas, usada, por exemplo, como entretenimento e trabalho. Para que todo esse grande número de sistemas fosse transferido para a *Web*, foi imprescindível a mudança de um conjunto de páginas estáticas para verdadeiros sistemas complexos desenvolvidos especialmente para esta plataforma.

Tais aplicações executam em um servidor conectado à *World Wide Web* e podem ser acessadas de qualquer computador ou dispositivo conectado à *Internet* através de um *browser* (navegador da *Internet*), que se utiliza de protocolos como o HTTP e da linguagem HTML para exibir em páginas *Web* uma interface gráfica para o usuário. Esse desenho vem removendo os obstáculos que existiam, como o usuário que só podia acessar um sistema ou aplicação caso essa estivesse instalada em seu computador ou caso ele estivesse conectado fisicamente à rede onde a aplicação está instalada.

Tornou-se capital a aplicação de conceitos existentes na Engenharia de Software, adaptados para essa nova plataforma, no desenvolvimento das aplicações *Web*. Recentemente então, observamos acontecer uma nova "Crise de Software" (GIBBIS, 1994), porém dessa vez ocorrida no desenvolvimento de aplicações para *Web* e por esse motivo chamada de "Crise Web" (MURUGESAN et al., 1999). Esse fato e as características do ambiente *Web*, como concorrência, carga imprevisível, disponibilidade, sensibilidade ao conteúdo, evolução dinâmica, imediatismo, segurança e estética motivou a criação da Engenharia Web (*Web Engineering*), que é a Engenharia de Software voltada para o desenvolvimento de aplicações *Web* (PRESSMAN, 2011).

A Engenharia Web trata de abordagens disciplinadas e sistemáticas para o desenvolvimento, a implantação e a manutenção de sistemas e aplicações baseados na

Web (MURUGESAN et al., 1999). Sua essência é gerenciar a diversidade e a complexidade das aplicações e, assim, evitar falhas de projeto que possam causar grandes problemas (GINIGE; MURUGESAN, 2001).

Como os *WebApps* foram ficando mais robustos e complexos, cresceu a dificuldade de desenvolvimento. Problemas que eram vistos no desenvolvimento de aplicações *Desktop* passaram a aparecer na criação dos *WebApps*. Dificuldades como: inexperiência e formação inadequada dos desenvolvedores, falta de modelos de processo, métodos inadequados, planejamento incorreto, prazos e custos excedidos, falta de documentação, dificuldades de implementação e manutenção, *layout* inadequado, mudança contínua dos requisitos, da tecnologia e da arquitetura, falta de rigor no processo de desenvolvimento, dentre outros (PERUCH, 2007).

Existe então um conjunto de atributos de qualidade a serem considerados neste caso (OLSINA; LAFUENTE; ROSSI, 2001):

- **Usabilidade:** trata-se de um requisito de qualidade como também um objetivo de aplicações *Web*: permitir a acessibilidade do sistema. Logo, o site deve ter uma inteligibilidade global, permitir o *feedback* e ajuda *online*, planejamento da interface/aspectos estéticos e aspectos especiais (acessibilidade por deficientes);
- **Funcionalidade:** o software *Web* deve ter capacidade de busca e recuperação de informações/*links*/funções, aspectos navegacionais e relacionados ao domínio da aplicação;
- **Eficiência:** o tempo de resposta deve ser inferior a 10s (velocidade na geração de páginas/gráficos);
- **Confiabilidade:** relativa à correção no processamento de *links*, recuperação de erros, validação e recuperação de entradas do usuário;
- **Manutenibilidade:** existe uma rápida evolução tecnológica aliada à necessidade de atualização constante do conteúdo e das informações disponibilizadas na *Web*, logo o software *Web* deve ser fácil de corrigir, adaptar e estender.

A primeira fase da Engenharia Web é a Análise de Requisitos. Requisitos de um sistema são descrições dos serviços que devem ser fornecidos por esse sistema e as suas restrições operacionais (SOMMERVILLE, 2007).

A análise de requisitos contém as atividades responsáveis pela esquematização do problema que a *Webapp* deve solucionar e coletar os requisitos com os *stakeholders* e os principais objetivos dessa fase são: definir as diversas categorias de usuário, definir os objetivos da aplicação *Web* e estabelecer um conhecimento básico respondendo as questões:

- Quais são os objetivos que a *Webapp* deve atingir?
- Quem vai utilizar a *Webapp*?
- Qual a principal necessidade de negócio que a *Webapp* deve suprir?

A próxima fase sugere a criação de um modelo de análise. O modelo de análise pode ser divido em quarto tópicos: a análise de conteúdo que identifica as classes de conteúdo que devem ser fornecidas para a *Webapp*, a análise de interação que descreve a forma pelo qual o usuário interage com a *Webapp*, a análise funcional define as operações que são aplicadas ao conteúdo da *Webapp* e a sequência de processamentos que ocorrem como consequência e por último, a análise de configuração que responsável por descrever o ambiente operacional e a infraestrutura na qual a *Webapp* reside (PRESSMAN, 2011).

Após o modelo de Análise da *Webapp*, a Engenharia Web propõe que seja elaborado o modelo de projeto que foca em seis tópicos principais: projeto de conteúdo, projeto arquitetural, projeto de estética, projeto de navegação, projeto de interface e projeto de componentes (PRESSMAN, 2011).

Na sequência do processo da Engenharia Web, encontra-se a fase de implementação, na qual o projeto é devidamente construído utilizando uma linguagem de programação escolhida de acordo com as características e requisitos do projeto. Por último, temos a fase de testes que é responsável por garantir a qualidade e garantir que a *WebApp* foi construída de acordo com os requisitos obtidos na fase de levantamento de requisitos.

Dentro desse contexto da Engenharia Web, foi proposto o método FrameWeb, um método que propõe a otimização dos processos da Engenharia Web.

2.2 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 com o uso em larga escala dos *frameworks* no desenvolvimento dos aplicativos, não havia nada que abordasse diretamente aspectos característicos dos *frameworks* na Engenharia Web. O método 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*.

O FrameWeb concentra-se na fase de projeto. No entanto, espera-se que um processo de desenvolvimento completo seja conduzido de modo a produzir um produto final de qualidade. Na fase de projeto, FrameWeb propõe que a arquitetura lógica do sistema siga o padrão *Service Layer* (FOWLER, 2002), representado na Figura 1. O sistema é dividido em três camadas: lógica de apresentação, lógica de negócio e lógica de acesso a

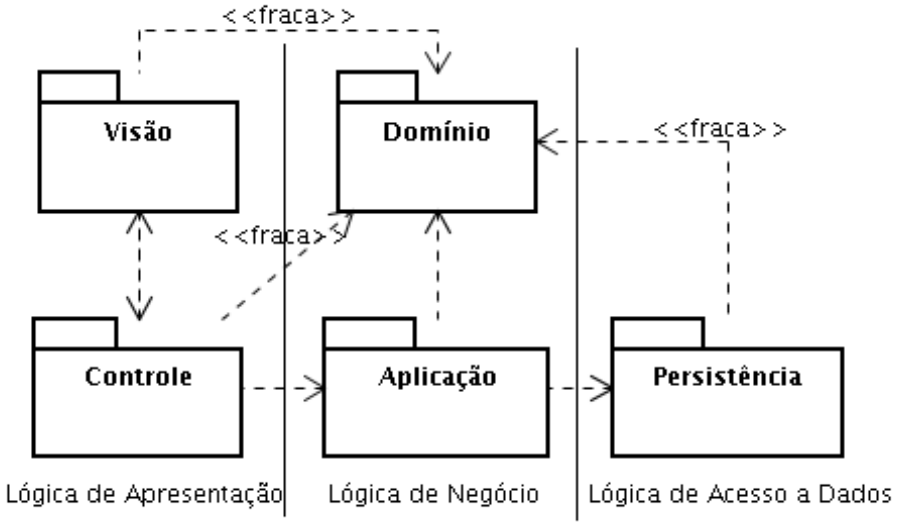

Figura 1 – Arquitetura padrão para WIS prescrita por FrameWeb (SOUZA, 2007).

dados. Essas que serão subdivididas nos pacotes: Visão, Controle, Aplicação, Domínio e Persistência.

Para representar componentes típicos da plataforma *Web* e dos *frameworks* utilizados, o FrameWeb estende o meta-modelo da UML, especificando, assim, uma sintaxe própria. Com isso, é possível utilizá-lo para a construção de diagramas de quatro tipos (MAI, 2017):

- **Modelo de Entidades:** é um diagrama de classes da UML que representa os objetos de domínio do problema e seu mapeamento para a persistência em banco de dados relacional;
- **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.
- **Modelo de Navegação:** é um diagrama de classe da UML que representa os diferentes componentes que formam a camada de Lógica de Apresentação, como páginas *Web*, formulários HTML e *controllers*.
- **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.

Atualmente, (MARTINS, 2016) propôs um meta-modelo para definir formalmente a linguagem de modelagem do FrameWeb, o que permite validar os modelos e construir outras ferramentas baseadas nesse meta-modelo, além de permitir que o método seja estendido para outros *frameworks* além dos originalmente propostos por (SOUZA, 2007).

2.3 Frameworks

Os WIS, e a maioria dos *WebApps*, possuem uma infraestrutura semelhante. Por isso a partir do momento em que os primeiros projetos na área foram implementados, *frameworks* foram criados que generalizavam essa base e poderiam ser aproveitados para o desenvolvimento de novas aplicações.

Um *framework* é uma aplicação reutilizável e semicompleta que pode ser especializada para produzir aplicações personalizadas (HUSTED et al., 2004). Assim, esses *frameworks* permitem que sistemas *Web* de grande porte sejam construídos com arquiteturas de múltiplas camadas sem muito esforço de codificação.

Além de permitir a reutilização, um *framework* minimiza o esforço de desenvolvimento de aplicações, por portar a definição da arquitetura das aplicações geradas a partir dele, como também por ter predefinido o fluxo de controle dessas aplicações (TALIGENT, 1993).

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.

Os dois *frameworks* utilizados neste trabalho, a saber, Wicket e Tapestry, pertencem à categoria de *frameworks* MVC.

2.3.1 Frameworks MVC

Model View Controller ou Modelo Visão Controle é um padrão de arquitetura de aplicações que visa separar a lógica da aplicação (*Model*) da interface do usuário (*View*) e do fluxo da aplicação (*Controller*), permitindo que a mesma lógica de negócios possa ser acessada e visualizada por várias interfaces. A Figura 2 ilustra a relação entre *Model*, *View*, *Controller* e Usuários, onde as linhas sólidas indicam associação direta e as tracejadas associações indiretas.

Os elementos da *View* (Visão) fazem uma representação das informações do *Model* (Modelo) para os usuários, a partir das quais os mesmos podem interagir com a aplicação.

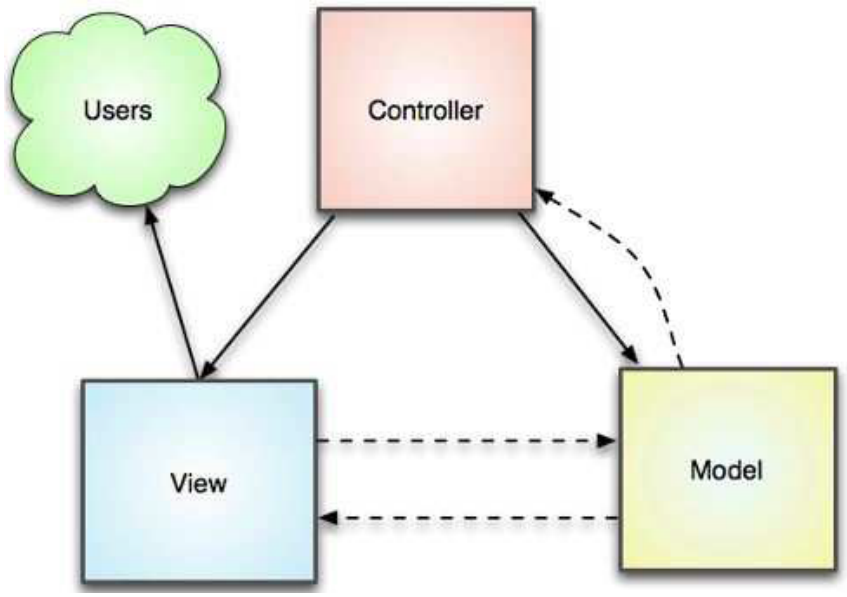

Figura 2 – Diagrama representando a arquitetura MVC (PRADO, 2015).

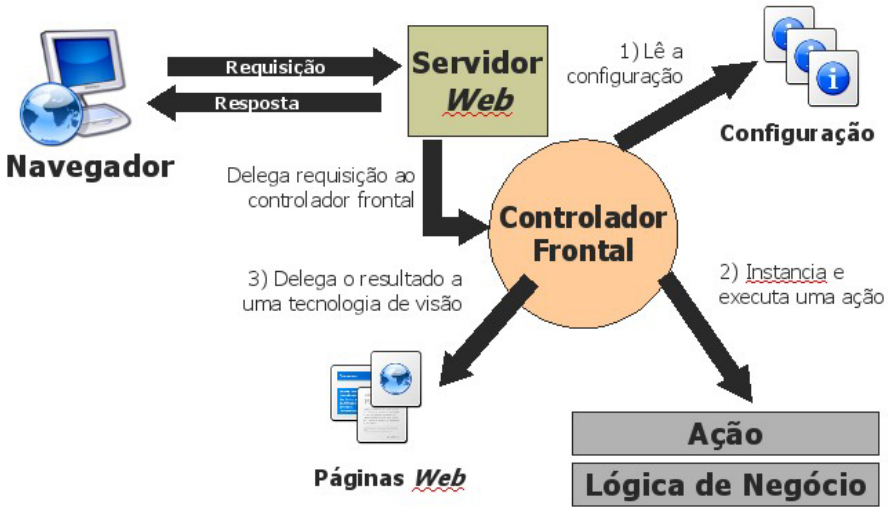

Figura 3 – Representação de um Controlador Frontal na *Web* (SOUZA, 2007).

Essa interação é tratada pelo *Controller* (Controle) que pode alterar o status dos elementos do *Model*. Este que pode notificar a *View* tais alterações, fazendo com que esta última possa atualizar a informação apresentada ao usuário (PRADO, 2015).

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) (ALUR; CRUPI; MALKS, 2003; SOUZA, 2007). A Figura 3 representa o funcionamento de um Controlador Frontal na *Web*.

Vemos que o navegador apresenta as páginas para o cliente que faz uma requisição

ao servidor, este que a delega para o Controlador Frontal que, ao interagir com as classes da aplicação, executa a ação para, em seguida, delegar o resultado para a tecnologia de visão. Os *frameworks* MVC fornecem um controlador frontal a ser configurado pelo desenvolvedor para que se adapte ao seu projeto.

2.3.2 Wicket

O Apache Wicket[1] é um *framework* orientado a componentes para o desenvolvimento de aplicações *web* com Java. Essa característica o torna diferente da maioria dos outros *frameworks* como, por exemplo, Struts ou Spring MVC, que baseiam-se na interceptação de requisições HTTP e a execução de ações associadas a essa requisição. No Wicket, uma ação é iniciada através de um evento enviado para determinado componente. Existem outros *frameworks* web orientados a componentes, por exemplo, como o próprio Tapestry e JSF. Ou seja, o processamento se inicia pela Visão que, quando necessário, obtém os dados por meio do *Managed Bean* (controlador), invoca as classes necessárias do Modelo e, por fim, disponibiliza os dados necessários para a Visão poder continuar o processamento.

Podemos também destacar as seguintes características: a configuração do Wicket não exige a criação/manipulação de XMLs e as páginas *web* e componentes do Wicket são objetos Java, que suportam encapsulamento, herança e eventos, isto é, o Wicket não mistura HTML com código Java nem faz uso de *taglibs* (bibliotecas de *tags* utilizadas por frameworks como JSF, por exemplo, para montagem das páginas HTML). Os arquivos HTML possuem apenas código HTML e são associados com classes Java por meio de atributos chamados *Wicket ids*, existentes nos elementos HTML.

2.3.3 Tapestry

O Tapestry,[2] como dito acima, também é um *framework* orientado a componentes para o desenvolvimento de aplicações *web* com Java.

É POJO[3] puro, ou seja, você não precisa implementar interfaces. Classes Java simples (e portanto fáceis de serem testadas) fazem todo o trabalho. O framework possui ainda uma característica interessante e diferente do Wicket, que é a existência de um *container* de IoC (*Inversion of Control*) embutido que cuida de toda a injeção de dependências de serviços e o ciclo de vida destes objetos.

[1] <https://wicket.apache.org/>
[2] <http://tapestry.apache.org/>
[3] *Plain Old Java Objects.*

3 SCAP

Neste capítulo apresentam-se o os requisitos do SCAP levantados anteriormente por Duarte (2014) e Prado (2015). A Seção 3.1 descreve o escopo do sistema; a Seção 3.2 apresenta os requisitos levantados na forma de casos de uso; e a Seção 3.3 mostra o diagrama de classes pela análise de tais requisitos.

Os requisitos apresentados aqui foram usados como base para construção de novos modelos FrameWeb e implementação de novas versões do SCAP, considerando os *frameworks* Wicket e Tapestry, a serem apresentados no capítulo seguinte.

3.1 Descrição do Escopo

O SCAP é um sistema para controle de afastamento de professores do Departamento de Informática (DI) da UFES, onde devem estar disponíveis informações sobre as solicitações de afastamento. As solicitações de afastamento podem ocorrer tanto para eventos no Brasil quanto no exterior. Essas solicitações precisam ser avaliadas pelos professores do DI e, em alguns casos, pela Câmara Departamental do Centro Tecnológico (CT) e pela Pró-reitoria de Pesquisa e Pós-Graduação (PRPPG) para que sejam aprovadas e o professor possa realizar sua viagem.

Caso o pedido de afastamento realizado para o professor seja para um evento que será realizado no Brasil o professor precisa da aprovação de seu chefe direto, que é o professor chefe do Departamento ao qual pertence, e também nenhum funcionário e representante discente do DI se manifeste contra em 10 dias. Dessa forma, para realizar um afastamento para um evento nacional a avaliação da solicitação não sai de dentro do DI.

No caso de um pedido de afastamento para um evento que acontecerá no exterior, é escolhido um professor que não tenha relação de parentesco com o solicitante para ser o relator do pedido. Após parecer do relator, o pedido passa por aprovação no departamento como no caso acima. Porém, também será necessário que o professor receba a aprovação do CT e da PRPPG para que a solicitação de afastamento seja aprovada e publicada em Diário Oficial da União.

No entanto, o SCAP trata de informações de solicitação de afastamentos apenas dentro do Departamento de Informática (DI) da UFES, de forma que os processos utilizados pelo CT e pela PRPPG evadem ao escopo do sistema e não há nenhuma integração com o sistema de acompanhamento de Processo da UFES. Ele deverá, no entanto, lidar com os pedidos de afastamento no exterior enquanto se encontrarem dentro do departamento.

Podemos dizer, então, que o SCAP foi projetado para que possa auxiliar o professor nessas solicitações e para tornar o processo de solicitação de afastamento mais dinâmico, através do envio de e-mails automáticos para os interessados e do uso de formulários para ajudar na criação dos documentos necessários para se realizar uma solicitação. O sistema também permite que os demais professores e funcionários do DI possam consultar as solicitações de afastamento em andamento.

3.2 Modelo de Casos de Uso

Duarte (2014) descreveu e identificou os atores do sistema SCAP, de acordo com a Tabela 1.

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

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.

Os secretários suportam a parte administrativa, é responsabilidade deles cadastrar os professores, bem como mandatos dos chefes do departamento e relações de parentesco. Também cadastram os pareceres de fora do DI e arquivam os pedidos quando estiverem devidamente regularizados.

Os professores cadastram seus pedidos de afastamento no sistema e ainda podem se manifestar contra o afastamento de outro professor, caso ainda seja tempo hábil para tal. Se for adicionado como relator de um afastamento no exterior, é responsabilidade do professor decidir se o DI aprova ou não tal afastamento.

O chefe do departamento é um professor que foi selecionado para exercer a função administrativa de chefe do DI por um mandato. Assim ele pode executar todos os casos de uso associados a um professor, e é de responsabilidade dele encaminhar solicitações a relatores que deferirão pareceres sobre afastamentos no exterior.

O SCAP foi dividido em dois subsistemas: o Núcleo que contém os casos de uso referente aos atores Professor e Chefe de Departamento, enquanto o subsistema Secretaria os casos de usos referentes ao ator Secretário. Nas figuras 4 e 5 são apresentados os diagramas de caso de uso destes subsistemas. Nos parágrafos seguintes apresentamos resumidamente a descrição dos casos de uso levantados.

No caso de uso **Solicitar Afastamento** o professor cadastra uma solicitação de afastamento no sistema para que seja analisado, podendo somente ele ou o chefe do departamento cancelar o pedido por meio do caso de uso **Cancelar Afastamento**.

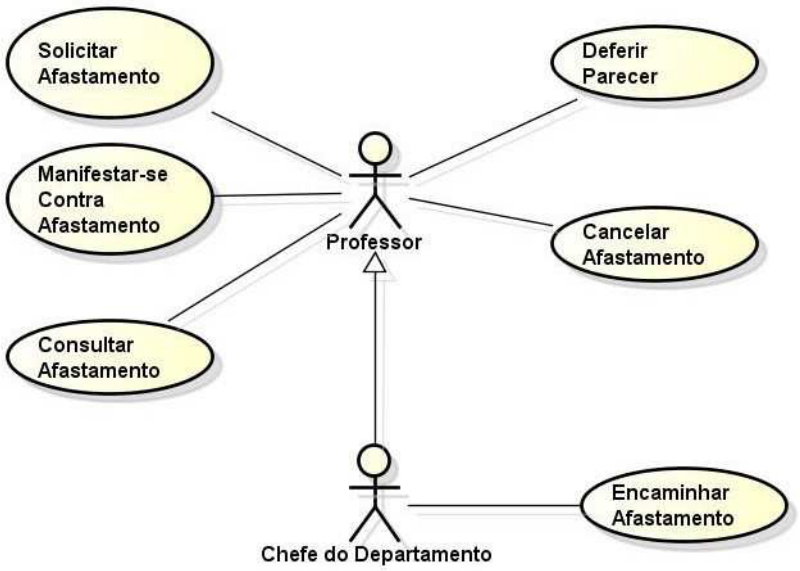

Figura 4 – Diagrama de Casos de Uso do subsistema Núcleo (PRADO, 2015).

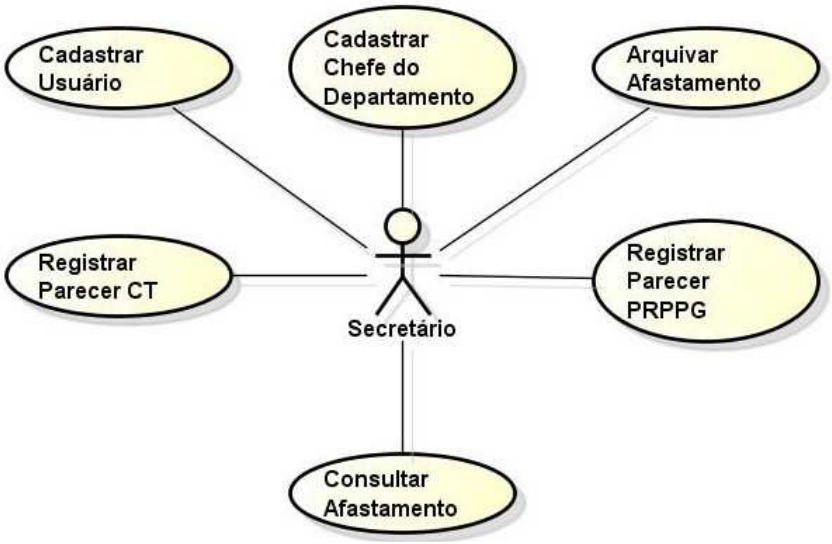

Figura 5 – Diagrama de Casos de Uso do subsistema Secretaria (PRADO, 2015).

O caso de uso **Encaminhar Afastamento** é executado quando ocorre uma solicitação de afastamento internacional e é realizado pelo Chefe do Departamento, que determina um professor para ser relator deste afastamento. O caso de uso **Definir Parecer** é realizado quando o professor escolhido como relator de um afastamento internacional cadastra no sistema o parecer sobre a referida solicitação.

O caso de uso **Consultar Afastamento** é utilizado quando o professor, chefe de departamento ou secretário busca informação sobre um pedido de afastamento. Quando um professor decide se manifestar contra um afastamento nacional ele faz uma busca por esse afastamento e logo depois utiliza o caso de uso **Manifestar-se Contra Afastamento** para realizar a sua manifestação.

Os secretários são responsáveis pela parte administrativa, logo utilizam-se dos casos de uso **Cadastrar Usuário** e **Cadastrar Chefe do Departamento** para adicionar um novo professor ou secretário ao sistema e registrar um professor previamente cadastrado como chefe do departamento, informando a data de início e fim do seu mandato.

Para um pedido de afastamento internacional é necessário inserir no sistema os pareceres de outras instâncias de aprovação e, para essas tarefas, utiliza-se os casos de uso **Registrar Parecer CT** e **Registrar Parecer PRPPG** para cadastro destes.

O caso de uso **Arquiva Afastamento** acontece no final da tramitação onde ocorre a mudança de status para "Arquivado".

3.3 Análise

Na criação do diagrama de classes deve se destacar a relevância da multiplicidade das relações que afeta diretamente a persistência de dados. O SCAP foi divido em dois subsistemas, entretanto as classes de domínio do problema são representadas em um único diagrama, que pode ser observado na Figura 6. Todas as classes, portanto, pertencem ao subsistema Núcleo. Segundo Prado (2015), tal divisão foi realizada para facilitar a implementação e não devido à natureza das classes criadas.

As classes **Professor** e **Secretário** representam os professores e secretários do departamento e herdam os atributos da classe **Pessoa**. Os professores que possuem relação de parentesco são representadas pela classe **Parentesco**. A classe **Mandato** representa um professor que é escolhido como chefe de departamento.

A classe **Afastamento** possui os dados referentes a uma solicitação de afastamento realizada por um professor. Esta pode conter documentos que são representados pela classe **Documento**. Se o afastamento for para fora do Brasil ele precisa de um relator que está relacionado na classe **Relator**.

A classe **Parecer** representa os pareceres dos professores e do relator (para o caso de afastamento internacional), bem como os pareceres do CT e PRPPG, e cada parecer está relacionado a um afastamento.

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, 2017).

Listamos abaixo as restrições de integridade que foram obtidas por Duarte (2014) e também as acrescentadas por Prado (2015), juntamente com novas restrições referentes a funcionalidades que não foram contempladas em projetos passados do SCAP.

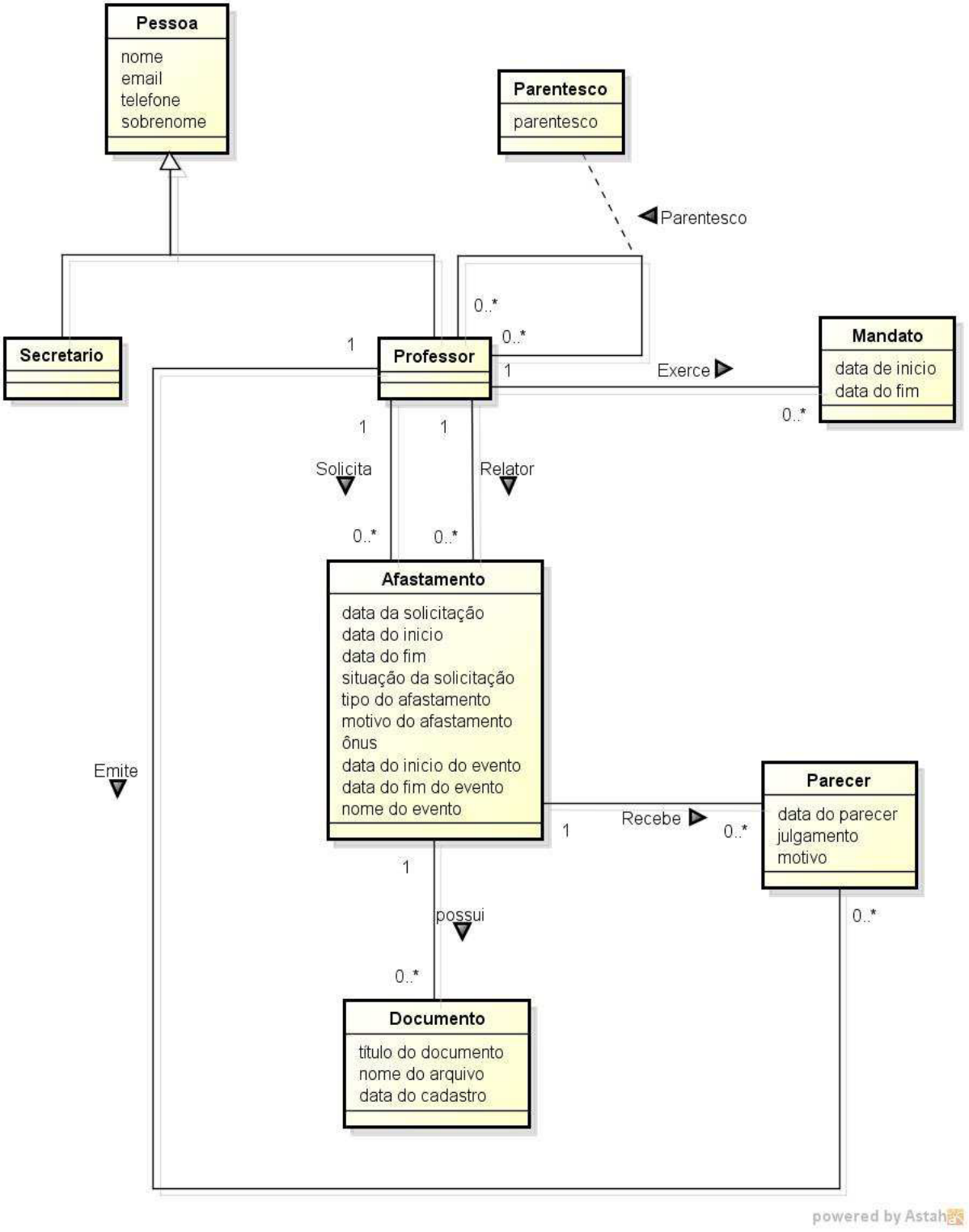

Figura 6 – Diagrama de Classes do SCAP (PRADO, 2015).

- 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 à data de fim do mesmo afastamento;

- A data de início de um mandato de professor não pode ser posterior à 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 e Implementação

Este capítulo descreve as duas novas implementações do SCAP, cada uma utilizando um *framework* MVC diferente, a saber: Wicket e Tapestry. Ambas foram baseadas nos resultados produzidos anteriormente por Duarte (2014) e Prado (2015) e seguiram o método FrameWeb. Como já concluído por Duarte (2014), a mudança do controlador frontal impacta diretamente no modelo de navegação do FrameWeb, pois existe uma correspondência direta entre esse modelo e esta categoria de *framework*.

Deve-se ressaltar que foram aproveitados para as duas novas implementações os códigos-fonte da implementação do SCAP realizada por Duarte (2014) e Prado (2015), em particular aqueles referentes ao modelo de domínio e persistência, porém a interface com usuário e a aplicação desenvolvidas neste trabalho. Nas próximas seções apresentam-se os novos modelos de navegação e aplicação propostos para cada uma das novas implementações do SCAP.

Como o objetivo principal deste trabalho era a aplicação do método FrameWeb com diferentes *frameworks*, e não a entrega da ferramenta SCAP em si, alguns requisitos não foram contemplados. A ferramenta não envia e-mails automáticos e também não é capaz de gerar as atas das reuniões que acontecem quando algum professor se manifesta contra um afastamento.

Outra funcionalidade que não foi implementada é a capacidade da ferramenta gerar os documentos atrelados aos afastamentos, ela pode apenas cadastrar um documento gerado em algum outro *software* e realizar a persistência do mesmo no banco de dados.

Este capitulo possui a seguinte estrutura: as Seções 4.1 e 4.2 apresentam a implementação completa do SCAP com o Wicket e Tapestry e, em seguida, a Seção 4.3 apresenta a comparação entre as versões do sistema SCAP já implementados.

4.1 Desenvolvimento do SCAP com o Wicket

O SCAP foi implementado utilizando a linguagem de programação Java, na plataforma Java EE 7 (Java Enterprise Edition 7). A plataforma foi escolhida por oferecer os frameworks CDI, utilizado para injeção de dependências entre as classes do sistema com objetivo de reduzir a quantidade de código utilizado, e JPA, para mapeamento objeto/relacional.

Utilizou-se Java Persistence API (JPA) para persistência de dados, padronizando o mapeamento objeto/relacional por meio de uma interface de dados comum aos principais *frameworks* desta categoria. Utilizamos o *framework* Hibernate, que implementa o padrão

JPA. Assim a Lógica de Acesso a Dados ficou mais simplificada, pois as consultas ao banco são escritas de forma mais próxima do padrão orientado ao objeto. Ainda na parte relacionada à persistência, utilizamos o banco de dados MySQL que armazena dados em tabelas, seguindo o modelo relacional.

Ademais, foram utilizados o Apache Maven, que é uma ferramenta de gerência de projeto capaz de simplificar o download e instalação de bibliotecas; o servidor de aplicação Wildfly 8; o Wicket como controlador frontal; e o ambiente de desenvolvimento Eclipse IDE.

Por fim, não foi utilizado nenhum framework de decoração, apenas o template HTML Bootstrap, pois o projeto não exige muitos recursos visuais.

4.1.1 Pacotes SCAP

A Figura 7 apresenta a distribuição das classes do SCAP na implementação do projeto utilizando o *framework* Wicket. O sistema possui dois subsistemas, Núcleo e Secretaria, que foram observados na fase de análise e especificação de requisitos. Em seguida, temos a subdivisões que respeitam a arquitetura em camadas proposta pelo FrameWeb, assim as classes foram separadas em 5 pacotes: Visão, Controle, Aplicação, Domínio e Persistência.

Podemos notar que no Wicket as páginas são codificadas em Java facilitando para o desenvolvedor, pois é mesma linguagem escolhida para implementar o sistema. Por isso, o pacote Visão se encontra no mesmo diretório que os outros pacotes.

O pacote **br.ufes.scap.nucleo.aplicacao** inclui todas as classes de aplicação do subsistema Núcleo, elas são responsáveis pele execução dos casos de uso referentes a este subsistema e implementam sua lógica de negócio.

A responsabilidade da comunicação entre o pacote visão e o pacote de aplicação é do pacote **br.ufes.scap.nucleo.controle** que contém as classes de controle que executam a tarefa anteriormente citada, tratando os estímulos enviados e recebidos pelo usuário ou pelo sistema.

O pacote **br.ufes.scap.nucleo.dominio** contém as classes de domínio de todo o sistema, que representam as entidades do mundo real e seus atributos. Por último, temos nesse subsistema o pacote **br.ufes.scap.nucleo.persistencia**, no qual encontram-se as classes responsáveis pela lógica de acesso e armazenamento de dados. Elas são responsáveis por fazer a persistência das classes de Domínio.

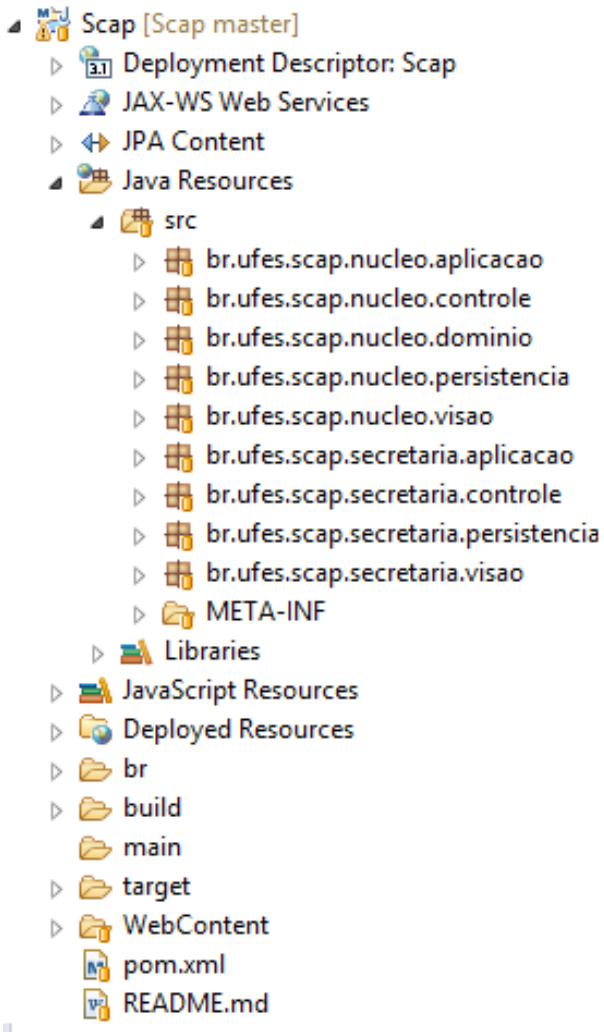

Figura 7 – Projeto SCAP com Wicket visto no Package Explorer do Eclipse IDE.

4.1.2 Modelo de Aplicação

O Modelo de Aplicação é um diagrama de classes da UML que representa as classes de serviço, que são responsáveis pela codificação dos casos de uso, e suas dependências (SOUZA, 2007). Ele é utilizado para demostrar as dependências entre classes do pacotes de Controle, Aplicação e Persistência. Podemos ver quais classes de ação dependem de quais classes de serviço e quais classes DAO são necessárias.

Assim como para as classes de controle do Modelo de Navegação, o projetista deve escolher o nível de granularidade das classes de serviço. Há também uma semelhança com o modelo de Persistência, pois no Modelo de Aplicação também não há definição de extensões UML e também valem as regras de programação por interfaces: cada classe de serviço deve ter uma interface e uma implementação (SOUZA, 2007).

As Figuras 8 e 9 são os Modelos de Aplicação criados para o subsistemas Núcleo e Secretaria respectivamente. No SCAP, optou-se por manter uma classe de serviço para cada classe controladora do pacote de controle, assim fica mais simples entender a relação

entre os dois pacotes.

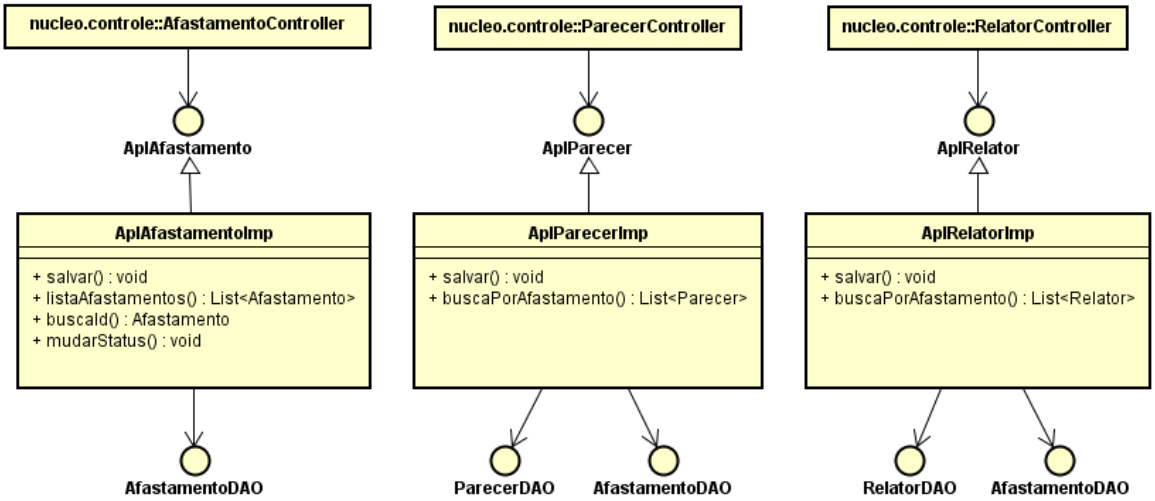

Figura 8 – Modelo de aplicação do subsistema Núcleo.

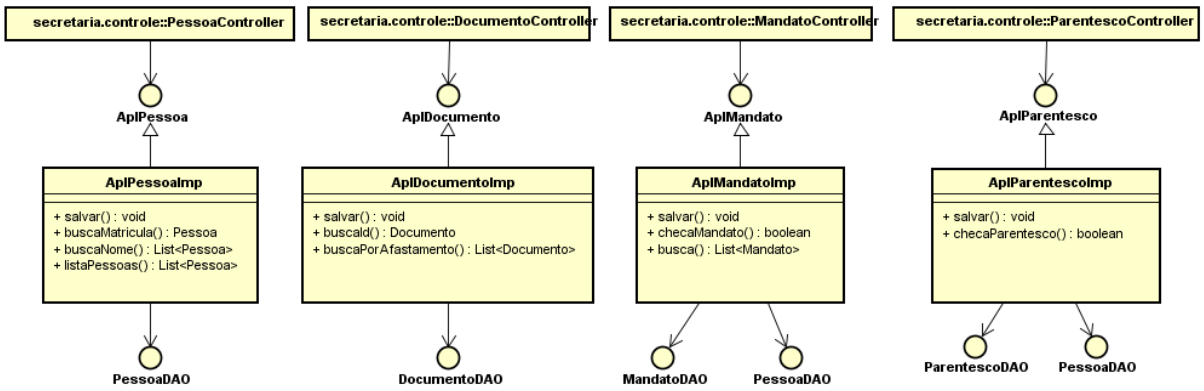

Figura 9 – Modelo de aplicação do subsistema Secretaria.

Para exemplificar a lógica do diagrama podemos tomar com exemplo o caso de uso **Cadastrar Chefe do Departamento**, onde o secretário cadastra um professor como chefe do departamento. Vemos que o **MandatoController** interage com a classe de serviço **AplMandatoImp** através da interface **AplMandato**. Para realizar o cadastro a classe de serviço precisa associar um professor, assim se faz necessário o **PessoaDAO**, e um mandato, logo precisaremos do **MandatoDAO**.

4.1.3 Modelo de Navegação

O Modelo de Navegação do FrameWeb é um diagrama de classe da UML que representa os diferentes componentes que formam a camada de Lógica de Apresentação, como páginas Web, formulários HTML e classes de ação do framework Front Controller (SOUZA, 2007).

Para ilustração, nessa seção são descritos os casos de uso **Solicitar Afastamento**, na Figura 10, e aos casos de uso **Manifestar-se contra Afastamento** e **Deferir Parecer**, na Figura 11.

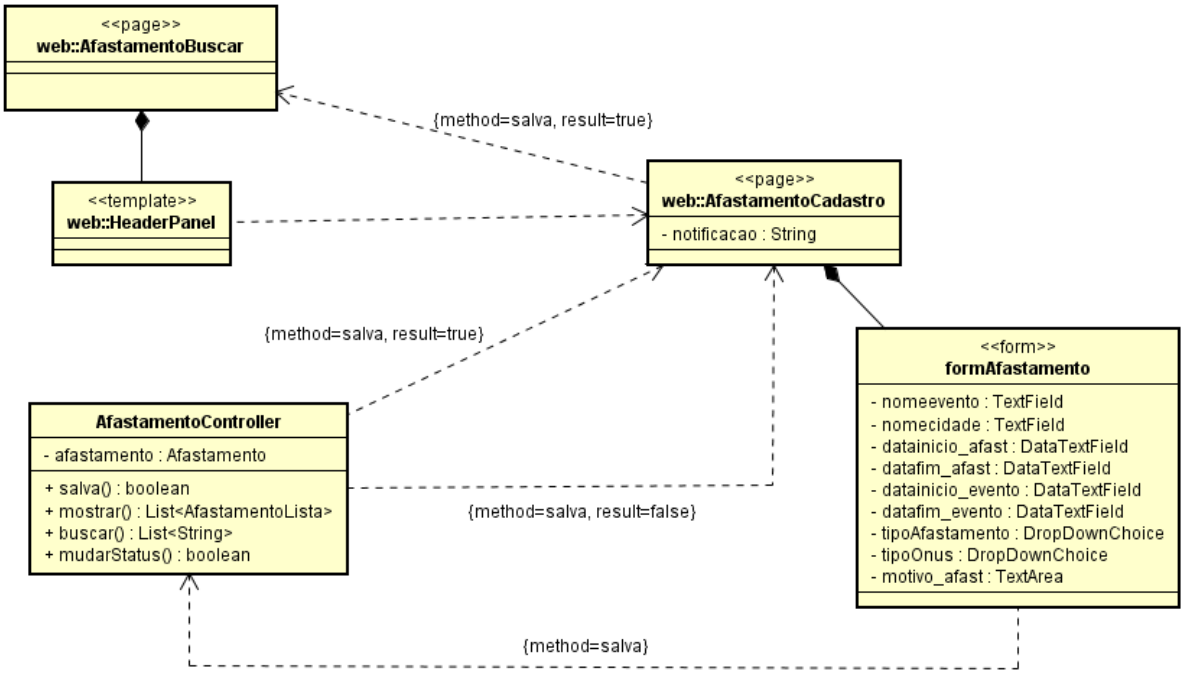

Figura 10 – Modelo de Navegação do caso de uso Solicitar Afastamento.

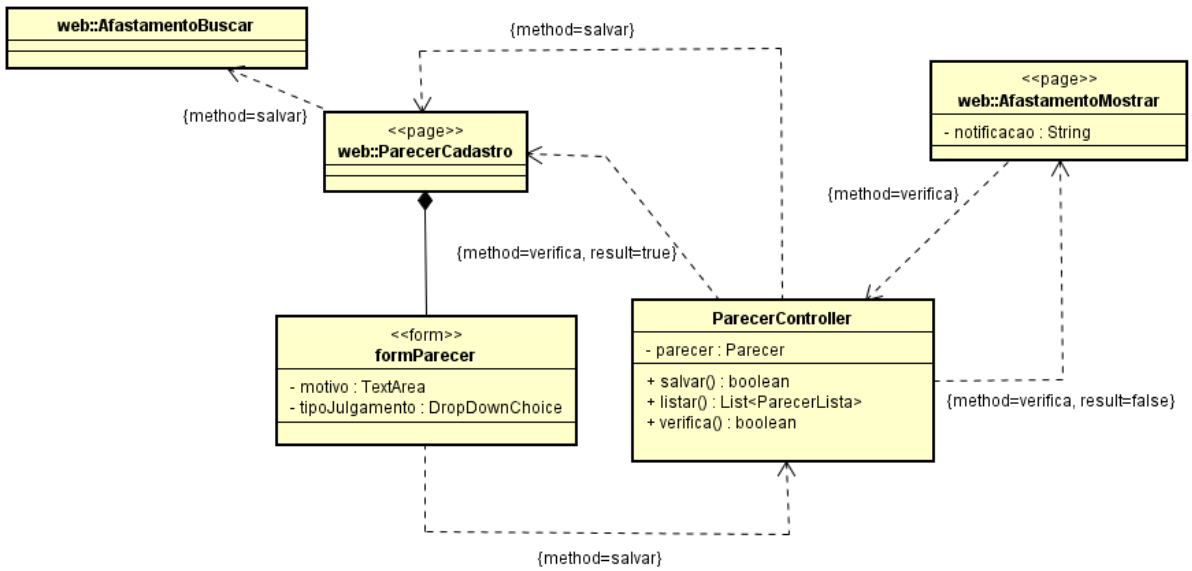

Figura 11 – Modelo de Navegação que cobre o caso de uso Manifestar-se Contra Afastamento e Deferir Parecer.

Logo após realizar o login, o professor cadastrado no sistema é encaminhado para a página principal do sistema, representada no diagrama de navegação pela classe **AfastamentoBuscar**, podendo realizar uma solicitação de afastamento. A classe **Afasta-**

mentoController é a classe controladora que lida com a lógica de apresentação referente a este caso de uso.

Quando ele realiza essa solicitação, o usuário é redirecionado para a pagina **AfastamentoCadastro**, contendo o formulário para preenchimento. Após preencher o formulário, o professor clica no botão salvar e a classe **AfastamentoController** invoca o método **salva** que, se comunica com a classe de aplicação responsável por esse caso de uso. Em seguida, em caso de sucesso, o usuário retorna à página inicial. Do contrário, é redirecionado novamente para a página **AfastamentoCadastro** com a notificação do erro ocorrido.

O modelo de navegação da Figura 11, referente aos casos de uso **Manifestar-se contra Afastamento** e **Deferir Parecer**, representa a situação na qual algum professor quer se manifestar em uma determinada solicitação de afastamento, podendo fazê-lo na página **AfastamentoMostrar**. Quando o botão *Deferir um Parecer* é acionado, o método **verifica** é chamado para ver se o respectivo professor pode deferir um parecer para o afastamento. Em caso de sucesso, é redirecionado para a página em que se preenche as informações relacionadas à manifestação. Devidamente preenchido, o controlador **ParecerController** chama o método **salvar**, que recebe esses dados do formulário e os envia para a aplicação, voltando para a página inicial do sistema **AfastamentoBuscar**. Já em caso de insucesso, uma notificação dizendo o motivo é mostrada na página **AfastamentoMostrar**.

É importante ressaltar dois pontos referentes aos modelos de navegação:

1. A Relação entre as páginas Web: devido às páginas em Wicket serem implementadas com a linguagem Java e possuir um método que executa facilmente a navegação de uma página a outra, chamado **setResponsePage**, assim que o controlador executa seus métodos, ele retorna o controle para a página Java que faz o redirecionamento devido. Por isso a associação entre duas páginas, que não é prevista originalmente em FrameWeb.

2. Relação entre os dados do formulário e o controlador: primeiramente, como já dito na Seção 2.3.2, um dos principais objetivos do Wicket é usar JavaBeans e POJO como modelo de dados. Logo, a ligação entre os dados do formulário e os atributos do domínio se dá através da classe **PropertyModel**. Essa classe tem apenas um construtor com dois parâmetros: o objeto de modelo (**Afastamento** em nosso exemplo) e o nome da propriedade que queremos ler/escrever (Como “nomeevento” em nosso exemplo). Isto também diverge de FrameWeb, que representa a troca de dados entre páginas/formulários e controladores por meio de atributos UML com mesmo nome.

4.1.4 Implementação do SCAP

Nesta seção apresentamos algumas capturas de tela de diferentes partes do sistema SCAP implementado com Wicket.

A Figura 12 exibe a tela de acesso ao sistema. Por meio dela todos o indivíduos cadastrados tem acesso às funcionalidades do sistema. Caso o usuário entre com um login invalido, inexistente ou incorreto, a classe controladora recarrega a página passando uma String com a mensagem de erro.

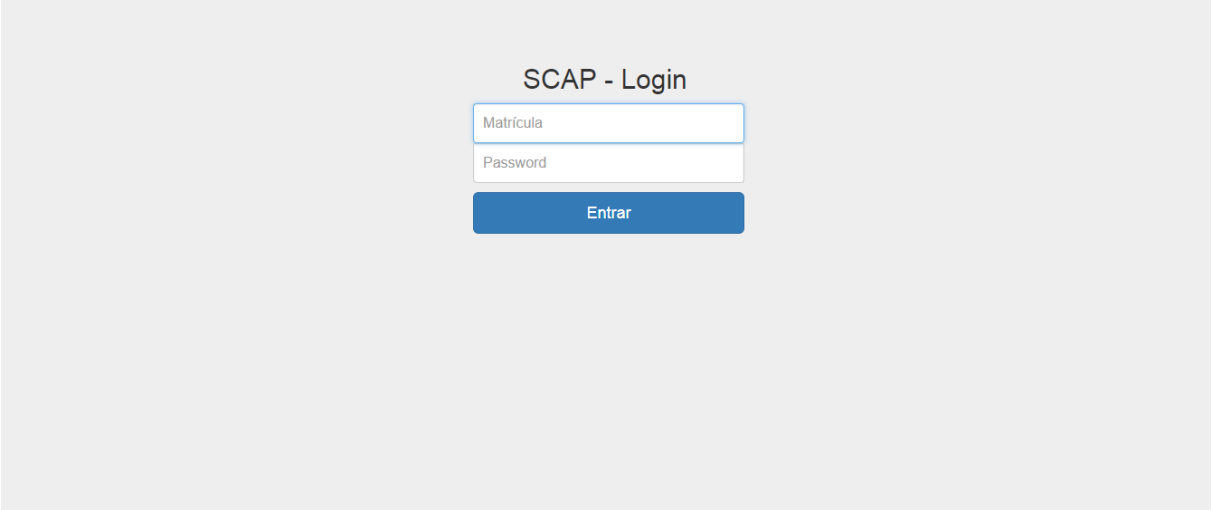

Figura 12 – Tela de login do SCAP.

Quando o login é realizado o usuário é direcionado para a página contendo a lista de todos os pedidos de afastamento ativos no sistema, vista na Figura 13. Nela, o usuário pode digitar o número identificador da solicitação e ela será encaminhada para a página que mostra as informações relativas àquela solicitação. Se o número identificador estiver incorreto, uma notificação é exibida nessa mesma tela. Para facilitar, a página contém uma lista de solicitações de afastamento com algumas informações sobre as mesmas, inclusive um botão que redireciona para a página onde as informações são exibidas por completo.

Figura 13 – Tela com o formulário de busca e lista de solicitações de afastamento.

Quando um professor solicita um afastamento ele é encaminhado para página

representada na Figura 14, na qual preenche o formulário com as informações referentes à solicitação de afastamento como, por exemplo, o nome do evento e da cidade, a data de início e data de fim do afastamento, dentre outras informações. Se um secretário tentar acessar essa funcionalidade, uma notificação será exibida na tela que somente professores tem acesso a essa tarefa.

Figura 14 – Tela com o formulário de cadastro de solicitação de afastamento.

A Figura 15 mostra as informações de uma determinada solicitação de afastamento. Nesta pagina ocorre a maioria das interações com a solicitação de afastamento: nela são realizadas as ações de mudança de status do afastamento, definição do parecer, cadastro do relator, cadastro de documentos, etc. Ela também contém a lista de documentos cadastrados para a solicitação.

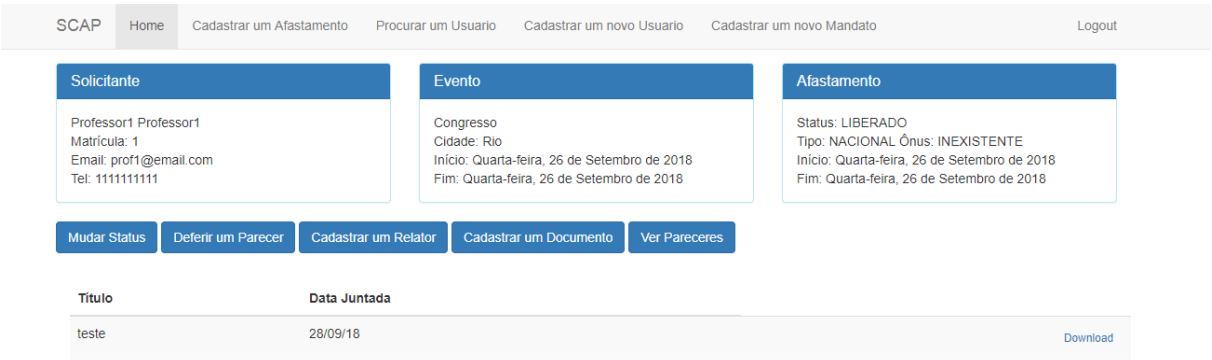

Figura 15 – Tela que apresenta os dados de um Pedido de Afastamento no SCAP.

É importante salientar que o caso de uso **Manifestar-se contra Afastamento** foi implementado junto com o caso de uso **Deferir Parecer**. Assim quando um professor quiser se manifestar contra um afastamento basta ele cadastrar um parecer desfavorável a um afastamento. Essa junção se deve a uma questão de praticidade e pela similaridade

entre os dois casos de uso.

4.2 Desenvolvimento do SCAP com o Tapestry

Apresenta-se a seguir a implementação do SCAP com o *framework* Tapestry. Como o Modelo de Aplicação é o mesmo que a implementação com Wicket discutida na seção anterior, mostra-se somente o Modelo de Navegação.

4.2.1 Tecnologias Utilizadas

Manteve-se praticamente as mesmas tecnologias utilizadas na implementação do SCAP com o Wicket, sendo também implementado utilizando a plataforma Java juntamente com o ambiente de desenvolvimento Eclipse IDE, com a única diferença sendo o Apache Tomcat como servidor de aplicação utilizado.

As duas implementações utilizam novamente o banco de dados MySQL e o framework Hibernate em conjunto com a JPA para facilitar a transição dos objetos gerados no sistema para a tabelas relacionais utilizadas no banco de dados.

O *framework* controlador frontal utilizado foi o Tapestry, que possui um container de IoC (*Inversion of Control*) embutido que cuida de toda a injeção de dependências de serviços e o ciclo de vida destes objetos, não necessitando assim da utilização do CDI como na implementação com o Wicket.

Por fim, também não foi utilizado nenhum *framework* de decoração, apenas o template HTML Bootstrap, pois o projeto não exige muitos recursos visuais.

4.2.2 Pacotes SCAP

A Figura 16, apresenta a distribuição das classes do SCAP na implementação do projeto utilizando o framework Tapestry. O sistema possui dois subsistemas, que foram observados na fase de análise e especificação de requisitos, seguindo a arquiteta proposta pelo método FrameWeb as classes foram divididas em 7 pacotes: Controle, Aplicação, Persistência, Entities, Components, Services e Pages.

Podemos notar diferenças na divisão dos pacotes da implementação com Wicket e Tapestry:

- **Pacote Services:** necessário em um projeto Tapestry, contém arquivos de configuração;
- **Pacote Components:** contém arquivos de *layout* das páginas (os *templates* no formato **.tml** ficam no pasta **src/main/resources**);

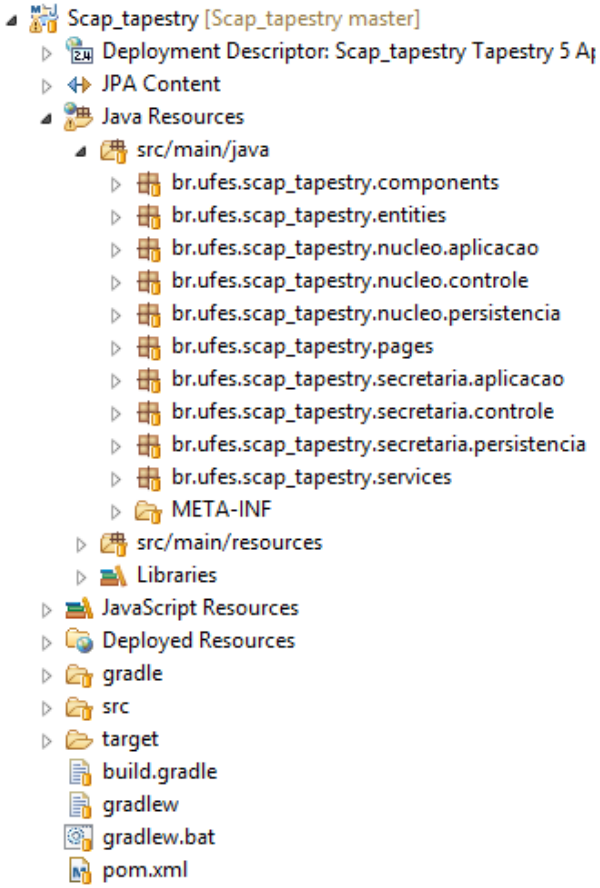

Figura 16 – Projeto SCAP com Tapestry visto no Package Explorer do Eclipse IDE.

- **Pacote Pages:** contém as páginas *Web* do projeto em Java (os *templates* no formato **.tml** ficam no pasta **src/main/resources**);
- **Pacote Entities:** onde devem ficar as classes de Domínio em um projeto Tapestry.

Os pacotes controle, aplicação e persistência do subsistema Núcleo possuem o mesmo objetivo anteriormente citados na seção 4.1.1. Visto que não abordamos anteriormente os pacotes do subsistema Secretaria, apresentamo-los nessa seção.

O pacote **br.ufes.scap_tapestry.secretaria.aplicacao** contém as classes de aplicação do subsistema Secretaria, responsáveis pela execução dos casos de uso referentes a este subsistema e implementam sua lógica de negócio.

Logo depois encontra-se o pacote **br.ufes.scap_tapestry.secretaria.controle** que contém as classes de controle que são responsáveis pela comunicação entre os pacotes de visão e aplicação, recebendo os estímulos do usuário pela visão e encaminhando para aplicação para execução dos casos de uso.

E por último o pacote **br.ufes.scap_tapestry.secretaria.persistencia**, no qual encontra-se as classes responsáveis pela lógica de acesso e armazenamento de dados. Elas são responsáveis em fazer a persistência das classes de Domínio.

Apesar do subsistema Secretaria não possuir classes de domínio, foi escolhido criar um pacote de persistência devido a relação estreita que existe entre os pacotes de aplicação e persistência.

Os pacotes **br.ufes.scap_tapestry.entities** e **br.ufes.scap_tapestry.pages** possuem todas as classes de domínio e visão respectivamente. Logo, porque são comuns a todo sistema, não houve divisão dessas classes entres os subsistemas.

4.2.3 Modelo de Navegação

Para ilustração, nessa seção são descritos os casos de uso **Consultar Afastamento**, na Figura 17, e **Cadastrar Usuário** na Figura 18.

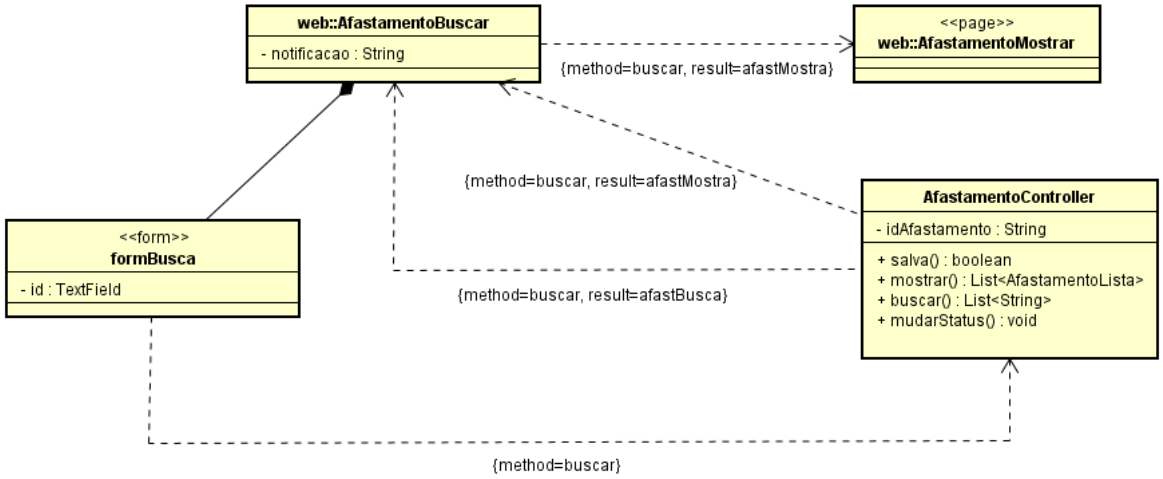

Figura 17 – Modelo de Navegação do caso de uso Consultar Afastamento.

O modelo de navegação da Figura 17 representa uma consulta a uma solicitação de afastamento. O usuário se encontra na página **AfastamentoBuscar** e o método **buscar** é chamado pelo controlador, na qual informa-se o identificador do afastamento. Em seguida o usuário é redirecionado para a página **AfastamentoMostrar** se o resultado da busca for bem sucedido e esta página contém todas as informações do afastamento, caso contrario é exibida um notificação ao usuário que indica que o afastamento não existe.

A Figura 18 mostra o modelo de navegação do caso de uso Cadastrar Usuário. A classe controladora **PessoaController** é responsável pelo cadastro de usuários no sistema. Quando ele realiza essa solicitação, o usuário é redirecionado para a pagina **PessoaCadastro**, contendo o formulário para preenchimento. Após preencher o formulário, o secretário clica no botão salvar e a classe **PessoaController** invoca o método **salvar** que, se comunica com a classe de aplicação responsável por esse caso de uso.

Assim como na implentação com Wicket, Tapestry também possui as características de ter páginas webs em Java e usar JavaBeans e POJO como modelo de dados. Logo as

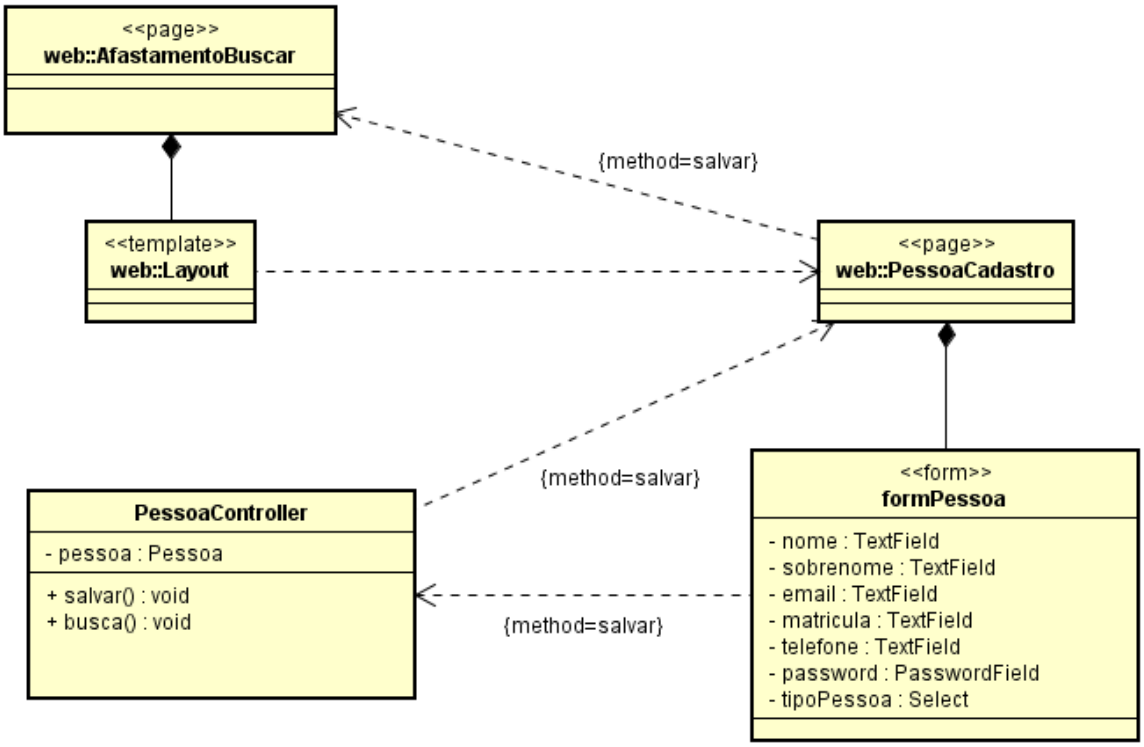

Figura 18 – Modelo de Navegação do caso de uso Cadastrar Usuário.

observações sobre a relação entre as páginas e a ligação dos dados do formulário com o controlador, explicadas na Seção 4.1.3 se aplicam aqui.

A única diferença é que no Wicket utiliza-se a classe **PropertyModel**, já em Tapestry faz-se uso da anotação **@Property**.

4.2.4 Implementação do SCAP

Nesta seção apresentamos algumas capturas de tela de diferentes partes do sistema SCAP implementado com Tapestry.

O funcionamento do login é idêntico a implementação com Wicket, então após o usuário realizar o login, ele é direcionado para a página contendo a lista de todos os pedidos de afastamento ativos no sistema, vista na Figura 19.

O formulário exibido na Figura 20 é referente ao cadastro de usuários no sistema e o secretário do departamento é o responsável pelo preenchimento dessas informações. Este usuários podem ser classificados em dois tipos: professor ou secretário. Se um professor tentar acessar essa funcionalidade, uma notificação será exibida na tela que somente secretários tem acesso a essa tarefa.

A Figura 21 apresenta a lista de todos os pareceres que foram gerados para um afastamento. Nela se pode ver a data, a pessoa responsável e o motivo.

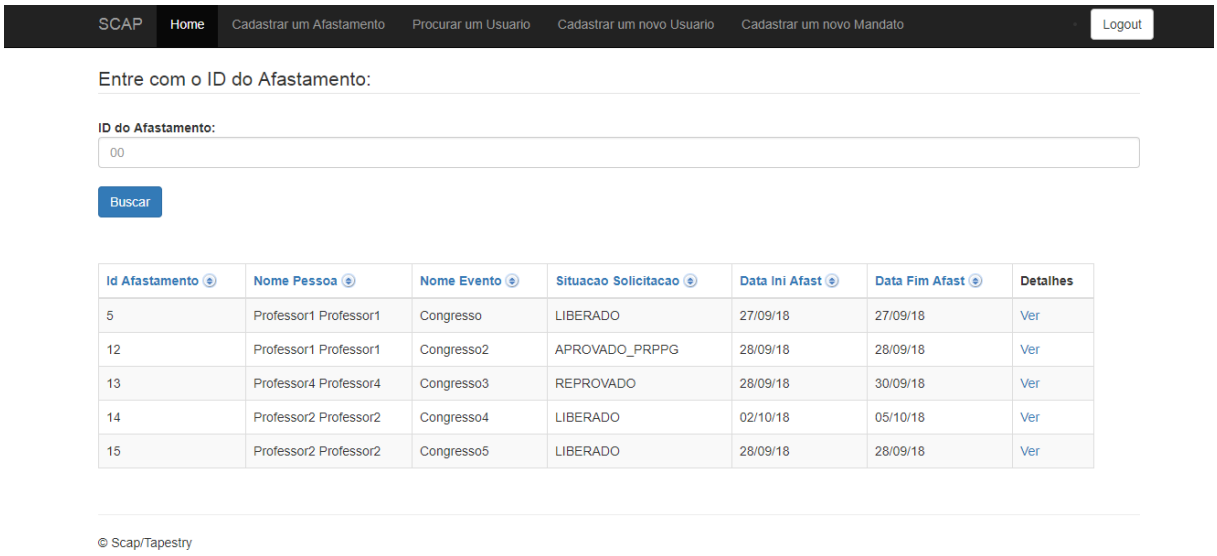

Figura 19 – Tela com o formulário de busca e lista de solicitações de afastamento.

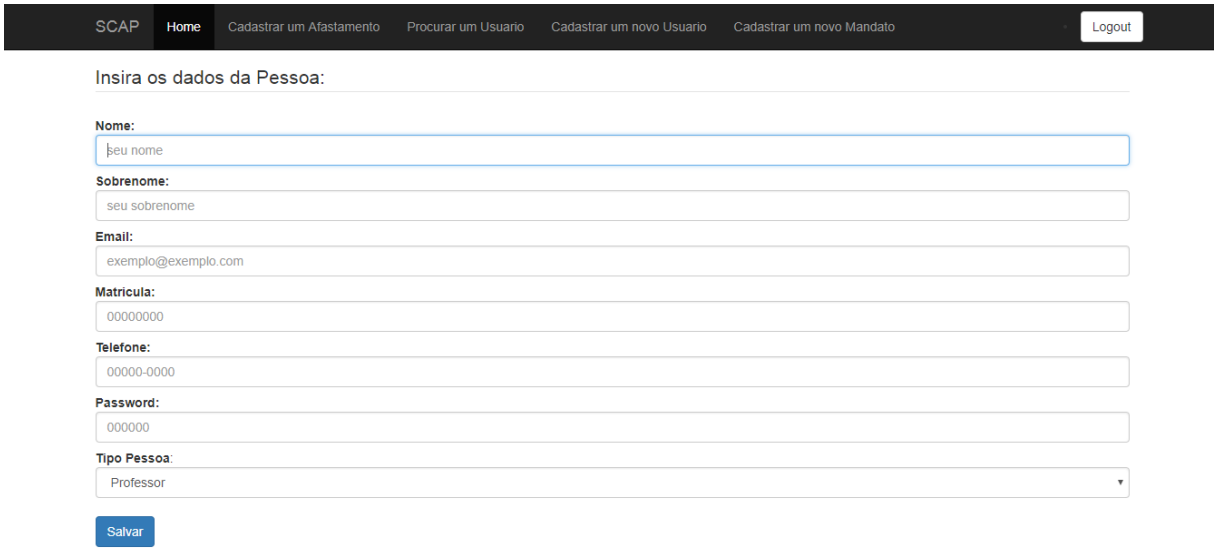

Figura 20 – Tela com o formulário de cadastro de usuário.

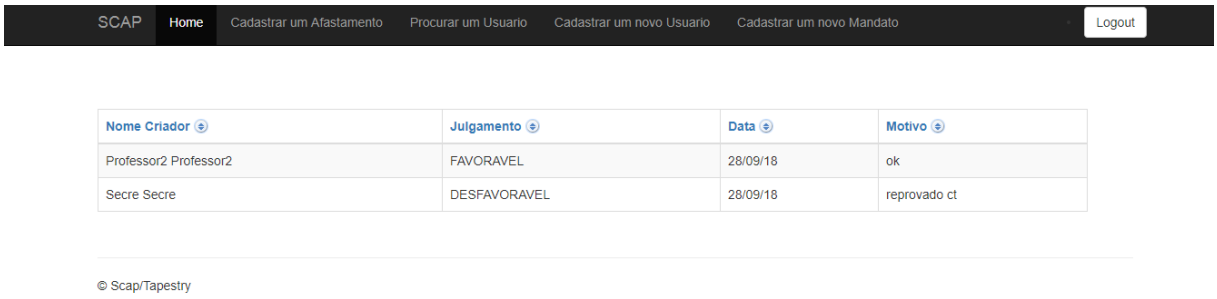

Figura 21 – Tela com a lista de pareceres de um afastamento.

Assim como na implementação com Wicket, o caso de uso **Manifestar-se contra Afastamento** foi implementado junto com o caso de uso **Deferir Parecer**.

4.3 Comparação entre versões do sistema SCAP

Como já visto, além da nova implementação do SCAP com os *frameworks* Wicket e Tapestry, existem os sistemas implementados por (DUARTE, 2014) e (PRADO, 2015). A tabela 2 mostra uma comparação contendo as principais mudanças encontradas entre as aplicações desenvolvidas.

Tabela 2 – Comparação entre versões do SCAP.

	SCAP (DUARTE, 2014)	SCAP (PRADO, 2015)	SCAP Atual
Relação de Parentesco	Não Possui	Possui	Possui
Campo para inserir motivo do afastamento	Não Possui	Não Possui	Possui
Alteração automática do status aprovado CT e PRPPG	Não Possui	Não Possui	Possui
Criação de página inicial de configuração	Não Possui	Não Possui	Possui

5 Considerações Finais

Este último capítulo descreve a experiência da utilização do FrameWeb com os *frameworks* Wicket e Tapestry, para a implementação do SCAP, e também discute a aplicabilidade do método com a mudança de *frameworks* proposta.

Primeiramente, tratou-se o desenvolvimento como uma tarefa de mudança de tecnologia. Ou seja, foi considerado que o software já tinha sido projetado e implementado, e era necessário realizar um estudo de todo o processo de desenvolvimento e executar alteração da tecnologia.

Depois de todo o estudo realizado, foram atingindo-se os objetivos, realizando a criação dos modelos de navegação para os *frameworks* adotados para a construção do sistema. A partir destes, juntamente com todos os modelos anteriormente levantados, partiu-se para a parte de aplicação prática na fase de implementação e, assim, atingiu-se o principal objetivo, que era a implementação de duas novas versões do SCAP com Wicket e Tapestry.

Ao final do processo, concluímos que o método FrameWeb agilizou a implementação, pois como a mudança proposta impacta principalmente no modelo de navegação, grande parte da estrutura do projeto não precisou ser alterada.

A alteração no modelo de navegação afeta diretamente as classes de controle e visão. Com isso, na implementação com Wicket e Tapestry, ambas tiveram que ser alteradas, pois caracterizam-se pela utilização da linguagem Java para produção dos artefatos de visão.

No entanto, essa característica facilitou, já que foi a mesma linguagem escolhida para implementação de todo o sistema, diferentemente da implementação com VRaptor 4 de Prado (2015). Assim, pôde-se perceber que alguns *frameworks* possuem características em comum e outros não.

Podemos concluir então, que os *frameworks* Wicket e Tapestry são ótimas escolhas para o desenvolvimento de aplicações *web*, pois possuem altíssima produtividade, aprendizado rápido e um modelo de desenvolvimento como já mencionado, de serem um mvc orientado a componentes e páginas codificadas em java não necessitando do uso de *taglibs*. Entretanto é mais sugerido o uso do Wicket, devido a sua fácil integração com CDI e também ser mais flexível.

Com a realização do trabalho, também pode-se notar a importância da utilização de um processo de Engenharia de Software para o desenvolvimento de aplicações de alta qualidade. Pois, com o uso dessas técnicas, temos vantagens como usabilidade e

manutenibilidade.

Outra consideração a ser feita é quanto as tabelas que foram implementadas no SCAP. Vimos anteriormente, que existe uma tabela dos afastamento ativos no sistema, onde o usuário pode clicar no botão "ver" para ver os dados de um afastamento sem precisar preencher o formulário de busca. Porém o método FrameWeb não fornece esse nível de granularidade em relação às paginas Web.

Duarte (2014) sugere a inclusão de um estereótipo «component», que representaria um elemento da página que sofre a ação de algum outro componente. Poderíamos aproveitar tal estereótipo para representar elementos dentro da página Web que direcionam a navegação do usuário, como as tabelas presentes no SCAP. No caso da implementação com Wicket, era um link através de ajax. A Figura 22 representa o modelo de navegação do caso de uso **Consultar Afastamento** usando o estereótipo *component* para representar a tabela.

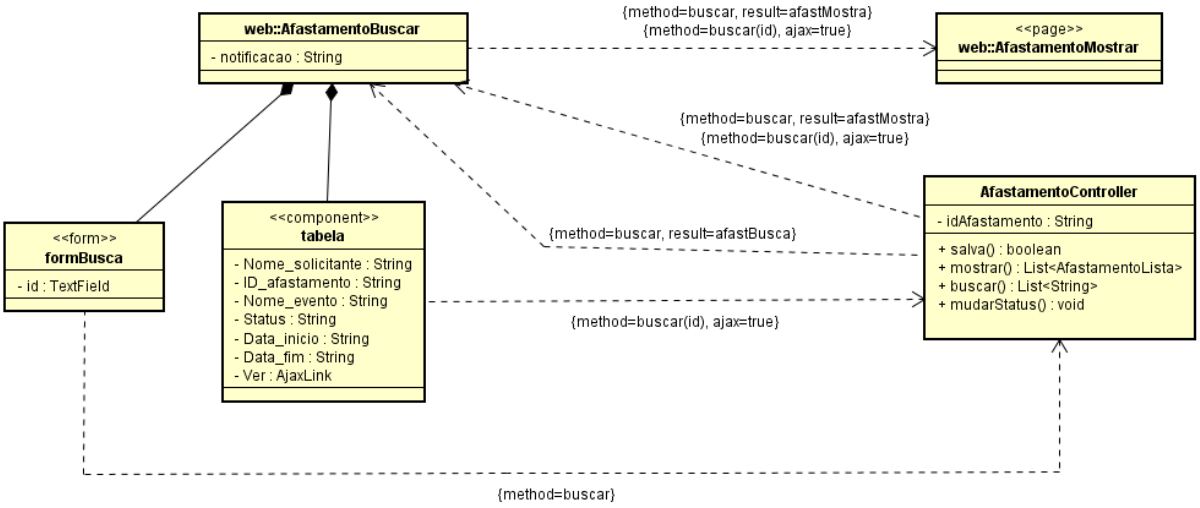

Figura 22 – Modelo de navegação proposto para o caso de uso Consular Afastamento, com adição de um novo estereótipo ao FrameWeb.

As observações sobre a relação entre as páginas e a ligação dos dados do formulário com o controlador, explicadas na seção 4.1.3, também se aplicam aqui.

Como trabalhos futuros, imagina-se uma nova implementação do SCAP usando uma coleção diferente de *frameworks* e tecnologias na plataforma Java EE 7 e uma aplicação de método FrameWeb usando uma plataforma e linguagem de programação diferente do Java, como por exemplo Python.

Referências

ALUR, D.; CRUPI, J.; MALKS, D. *Core J2EE Patterns: Best Practices and Design Strategies*. 2. ed. [S.l.]: Prentice Hall Core, 2003. Citado na página 18.

BAUER, C.; KING, G. *Hibernate em Ação*. 1. ed. [S.l.]: Ciencia Moderna, 2005. Citado na página 11.

DUARTE, B. B. *Aplicação do Método FrameWeb no Desenvolvimento de um Sistema de Informação na Plataforma Java EE 7*. Monografia (Projeto de Graduação) — Universidade Federal do Espírito Santo, 2014. Citado 9 vezes nas páginas 7, 11, 12, 20, 21, 23, 26, 39 e 41.

FALBO, R. A. *Projeto de Sistemas de Software*. [s.n.], 2017. 135 p. Disponível em: <https://inf.ufes.br/~falbo/files/PSS/Notas_Aula_Projeto_Sistemas_2017.pdf>. Citado na página 23.

FOWLER, M. *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 0321127420. Citado na página 15.

FOWLER, M. *Inversion of Control Containers and the Dependency Injection pattern*. 2004. Disponível em: <https://www.martinfowler.com/articles/injection.html>. Citado na página 11.

GAMMA, E. et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. [S.l.]: Addison-Wesley, 1994. Citado na página 10.

GIBBIS, W. Software's chronic crisis. Scientific American, September 1994. Citado na página 13.

GINIGE, A.; MURUGESAN, S. Web engineering: An introduction. *IEEE multimedia*, IEEE, v. 8, n. 1, p. 14–18, 2001. Citado 2 vezes nas páginas 10 e 14.

HUSTED, T. et al. *Struts em Ação*. 1. ed. [S.l.]: Ciência Moderna, 2004. 604 p. Citado na página 17.

MAI, L. F. F. *Khoeus: uma plataforma de aprendizado focada no ensino da programação*. Monografia (Projeto de Graduação) — Universidade Federal do Espírito Santo, 2017. Citado na página 16.

MARTINS, B. F. *Evolução do Método FrameWeb para o Projeto de Sistemas de Informação Web Utilizando uma Abordagem Dirigida a Modelos*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2016. Citado na página 16.

MURUGESAN, S. et al. Web engineering: A new discipline for development of web-based systems. *Proceedings of the Fist ICSE Workshop on Web Engineering (Australia)*, IEEE, 1999. Citado 3 vezes nas páginas 10, 13 e 14.

OLSINA, L.; LAFUENTE, G.; ROSSI, G. Specifying quality characteristics and attributes for websites. Springer-Verlag London, 2001. Citado na página 14.

PERUCH, L. A. *Aplicação e análise do Método FrameWeb com Diferentes Frameworks Web*. Monografia (Projeto de Graduação) — Universidade Federal do Espírito Santo, 2007. Citado na página 14.

PRADO, R. C. d. *Aplicação do Método FrameWeb no Desenvolvimento de um Sistema de Informação utilizando o framework VRaptor 4*. Monografia (Projeto de Graduação) — Universidade Federal do Espírito Santo, 2015. Citado 10 vezes nas páginas 6, 11, 18, 20, 22, 23, 24, 26, 39 e 40.

PRESSMAN, R. S. *Engenharia de software - Uma Abordagem Profissional*. 7. ed. [S.l.]: MacGraw-Hill, 2011. 780 p. Citado 3 vezes nas páginas 10, 13 e 15.

RESENDE, A.; SILVA, C. *Programação Orientada a Aspectos em Java*. 1. ed. [S.l.]: Brasport, 2005. Citado na página 10.

SOMMERVILLE, I. *Engenharia de software*. 8. ed. [S.l.]: Pearson, 2007. 568 p. Citado na página 14.

SOUZA, V. E. S. *FrameWeb – A Framework-based Design Method for Web Engineering*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2007. Citado 9 vezes nas páginas 6, 11, 12, 15, 16, 17, 18, 28 e 29.

TALIGENT. Leveraging object-oriented frameworks. A Taligent White Paper, 1993. Citado na página 17.