Cleisson Santos Guterres

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o *framework* Play

Vitória, ES

2019

Cleisson Santos Guterres

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o *framework* Play

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

2019

Cleisson Santos Guterres

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o *framework* Play/ Cleisson Santos Guterres. – Vitória, ES, 2019-

61 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Vítor E. Silva Souza

Monografia (PG) – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Departamento de Informática, 2019.

1. Palavra-chave1. 2. Palavra-chave2. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o *framework* Play

CDU 02:141:005.7

Cleisson Santos Guterres

Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o *framework* Play

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, 25 de setembro de 2014:

Prof. Dr. Vítor E. Silva Souza
Orientador

Profa. Dra. Monalessa Perini Barcellos
Departamento de Informática, CT/UFES

Félix Luiz Zanetti
Programa de Pós-Graduação em Informática, CT/UFES

Vitória, ES
2019

Resumo

Devido à ascensão da Internet, também cresceu a demanda por software de qualidade funcionando nesta plataforma. Esses programas se tornaram grandes e complexos demais para serem desenvolvidos de maneira *ad hoc*. Portanto, foi necessário aplicar conceitos de Engenharia de Software no desenvolvimento de sistemas e aplicações Web, nascendo assim a Engenharia Web.

Neste contexto foi proposto o método FrameWeb (*Framework-based Design Method for Web Engineering*), que sugere a utilização de uma série de *frameworks*, juntamente com varias recomendações de modelagem, que agilizam as principais fases de desenvolvimento.

O FrameWeb vem sendo aplicado em diferentes projetos e arquiteturas/plataformas, para verificar a generalidade de suas propostas. Este trabalho se propõe a testar o método FrameWeb em uma nova linguagem — Scala — e um novo *framework* — *Play* — através da reimplementação de uma WebApp — o Sistema de Controle de Afastamentos de Professores, ou SCAP.

Palavras-chaves: Engenharia Web. WebApp. FrameWeb. Framework Play. Scala.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
CASE	Computer-Aided Software Engineering
CRUD	Create Read Update Delete
CT	Centro Tecnológico
DAO	Data Access Object
DI	Departamento de Informática
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
I/O	Input/Output
MVC	Model-View-Controller
PRPPG	Pró-Reitoria de Pesquisa e Pós-Graduação
REST	Representational State Transfer
SCAP	Sistema de Controle de Afastamento e Professores
UML	Unified Modeling Language
WIS	Web Information System

Sumário

1 Introdução

É impossível deixar de notar um grande crescimento no número de aplicações desenvolvidas para a Web. Essa plataforma de desenvolvimento permite que aplicações sejam acessadas de qualquer lugar e a qualquer momento, características fundamentais para sistemas modernos que atendem um grande número de usuários.

Inicialmente, a construção de WebApps (aplicações para a Web) acontecia de forma *ad hoc*. Entretanto, com o seu crescimento em número e complexidade, viu-se a necessidade de adaptar métodos da Engenharia de Software ao desenvolvimento dessas aplicações, e isso se deu com o surgimento da Engenharia Web.

No contexto da Engenharia Web, com o objetivo de propor uma abordagem de construção de sistemas para Web baseadas em *frameworks*, surgiu o método FrameWeb (SOUZA, 2007).

Para aplicar o FrameWeb e sua capacidade de atender, a partir de modelos genéricos, vários *frameworks* diferentes, em particular na categoria Controlador Frontal, propõe-se o desenvolvimento de um *WebApp*, o SCAP (Sistema de Controle de Afastamentos de Professores). Esse sistema já foi desenvolvido anteriormente em Java nos *frameworks*: JSF por Duarte (2014), VRaptor 4 por Prado (2015), Spring MVC e Vaadin por Matos (2017), Ninja por Avelar (2017); e combinando as linguagens PHP e JavaScript por Pinheiro (2017). Aqui se propõe seu desenvolvimento na linguagem Scala e no *framework* Play.

Uma das responsabilidade do Departamento de Informática (DI) da UFES é o gerenciamento e o registro de solicitações de afastamento de seus professores para eventos no Brasil ou no exterior. O SCAP é um aplicativo Web que auxilia o DI nessa tarefa, proporcionando um ambiente de fácil usabilidade, que possa ser empregado por professores e funcionários no decorrer do processo do afastamento.

1.1 Motivação e Justificativa

O processo de requisição de afastamento envolve muitos detalhes, portanto a possibilidade de um sistema Web para auxiliar o DI é bastante atraente. Tal proposta de um ambiente de fácil usabilidade e com um mecanismo de notificações automáticas beneficia não apenas os secretários do DI, como também os professores que requisitam ou avaliam afastamentos, facilitando o trabalho para todos.

Além de ser um tema relevante dentro do contexto do DI, o desenvolvimento do sistema, com o método FrameWeb, apresenta uma vasta gama de possibilidades de estudo dentro da área de computação e desenvolvimento Web, proporcionando um grande

aprendizado durante o processo de sua implementação.

Por fim, esse projeto também representa um passo a mais no desenvolvimento do método FrameWeb, oferecendo uma oportunidade de analisar como sua aplicação se encaixa no contexto de uma linguagem diferente — Scala — e de um *framework* diferente, o *Play*.

1.2 Objetivos

O objetivo geral deste trabalho é aplicar o método FrameWeb (SOUZA, 2007) em uma nova implementação do SCAP, a partir dos requisitos já levantados por Duarte (2014) e Prado (2015).

Os objetivos específicos deste trabalho são:

- Aplicar o conhecimento adquirido em diversas disciplinas ao longo do curso de graduação em Engenharia de Computação, tais como Programação III, Engenharia de Software, Banco de Dados, Desenvolvimento Web e Web Semântica, entre outras;
- Analisar como o framework *Play* se encaixa ao método FrameWeb no contexto do SCAP, assim auxiliando na avaliação do método e propondo modificações nos modelos, caso necessário.

1.3 Método de Desenvolvimento do Trabalho

As seguintes atividades foram realizadas no contexto deste trabalho:

- Aquisição de Conhecimento: de forma geral, estudo nas áreas de Engenharia de Software, Engenharia de Requisitos, Desenvolvimento de Softwares e Desenvolvimento Web. De forma específica, estudo de Engenharia Web, do método FrameWeb (SOUZA, 2007) e do *framework Play*;
- Análise do Problema Proposto: estudo do sistema SCAP, de acordo com propostas levantadas anteriormente por Duarte (2014) e Prado (2015);
- Projeto e Codificação: fazer a modelagem e a codificação do projeto, tendo em vista os recursos do *framework Play* e da linguagem Scala;
- Testes e ajustes finais: testar o sistema e averiguar se suas funcionalidades funcionam como previsto, fazendo correções e ajustes quando necessário.
- Apresentação do projeto e dos resultados: feito através da elaboração do texto da monografia.

1.4 Conteúdo da Monografia

A monografia entregue como produto final deste trabalho é dividida da seguinte maneira:

- **Capítulo 1: Introdução**

  Esse capítulo introdutório explica o contexto do projeto, explicitando seus objetivos e a metodologia empregada.

- **Capítulo 2: Fundamentação Teórica e Tecnologias Utilizadas**

  O capítulo que aborda o referencial teórico apresentará três principais tópicos:

  - Explicar o que é **Engenharia Web**, destacando seus objetivos, sua metodologia e suas fases (Análise de Requisitos, Modelo de Análise, Projeto da WebApp, Implementação e Testes).
  - Explicar o que é o método **FrameWeb**, destacando seus objetivos, sua arquitetura de software com divisão em camadas e pacotes e sua linguagem de modelagem.
  - Apresentar brevemente o *framework* **Play** e a linguagem Scala.

- **Capítulo 3: O Sistema SCAP**

  Esse capítulo apresenta a descrição do escopo do sistema, seu modelo de casos de uso, e faz uma análise de sua modelagem de classes e restrições de integridade.

- **Capítulo 4: Projeto e Implementação**

  Esse capítulo apresenta o projeto do sistema, isto é, a arquitetura utilizada, bem como a forma com que esta é tratada pelo *framework* Play; e a implementação do sistema, isto é, sua interface, funcionalidades e seu desenvolvimento, por meio de imagens da aplicação, diagramas e códigos relevantes.

- **Capítulo 5: Considerações Finais**

  Conclusão do trabalho explicitando seus pontos fortes e fracos, destacando o papel do *framework* Play para o projeto, além de possíveis direcionamentos para trabalhos futuros.

2 Fundamentação Teórica e Tecnologias Utilizadas

Neste capítulo serão apresentados: a Engenharia Web e suas fases; o método *FrameWeb*, sua arquitetura de software e sua linguagem de modelagem; e os *frameworks* MVC, categoria a qual o *Play* pertence. Também será feita uma breve introdução a linguagem Scala e ao *framework Play*.

2.1 Engenharia Web

A Internet passou a ser parte integral das nossas vidas: ela está presente nas indústrias, no governo, nos bancos, nas lojas (muitas das vezes, lojas completamente virtuais), entre outros lugares. À medida que as pessoas dependem cada vez mais de sistemas e aplicativos baseados em Web, se torna imprescindível que esses apresentem um padrão alto de qualidade.

A Engenharia Web (*Web Engineering - WebE*) é a Engenharia de Software aplicada ao desenvolvimento Web (PRESSMAN, 2011). Ela estabelece abordagens disciplinadas e sistemáticas, através de um tom científico, técnicas de engenharia e princípios de gestão, para garantir o bom desenvolvimento, implementação e manutenção de sistemas e aplicações baseadas na Web (MURUGESAN et al., 2001).

De acordo com Olsina, Lafuente e Rossi (2001), os atributos mais relevantes para a garantia de qualidade nas aplicações Web são:

- Usabilidade: refere-se à facilidade de compreensão e uso do site, incluindo usuários com pouco conhecimento técnico e também os que possuem alguma deficiência. Para isso, deve-se empregar recursos de ajuda, *feedback online* e planejamento da interface nos aspectos estéticos e com características especiais voltadas a determinados usuários;
- Funcionalidade: refere-se ao funcionamento correto do sistema, isso é, sua capacidade de busca e recuperação de informações, a adaptação aos diferentes *browsers*, e ser capaz de executar as funções relacionadas ao domínio da aplicação do sistema;
- Confiabilidade: refere-se à garantia da validação dos dados de entrada dos usuários, recuperação de erros e funcionamento correto dos *links* no site;
- Eficiência: refere-se a garantir uma velocidade satisfatória na geração de páginas, fazendo com que o tempo de resposta seja o menor possível;

- Manutenibilidade: refere-se à facilidade do sistema em ter seus erros corrigidos, se adaptar a modificações e estender seu uso a novas situações. Esse aspecto é fundamental devido à rápida evolução tecnológica e à necessidade de atualização constante de conteúdo na Web.

A metodologia da Engenharia Web trabalha para que tais atributos sejam plenamente satisfeitos. O modelo proposto pela Engenharia Web é divido em várias fases e segue uma abordagem iterativa. As fases estão descritas a seguir.

2.1.1 Análise de Requisitos

A primeira fase da Engenharia Web é a Análise de Requisitos, etapa na qual tudo o que é considerado necessário para a entrega da aplicação é levantado por meio de consulta aos *stakeholders*, i.e., todas as pessoas interessadas no sistema, incluindo donos e possíveis usuários. São coletadas as informações sobre os requisitos funcionais — as funções que o sistema deve executar — e sobre os requisitos não-funcionais — as restrições sob as quais o sistema deve operar.

Nessa fase, os seguintes questionamentos tem que ser respondidos:

- Quais são os objetivos visados pela *WebApp* (aplicação Web)?
- Quem serão os usuários da *WebApp*?
- Quais necessidades de negócio devem ser atendidas pela *WebApp*?

2.1.2 Modelo de Análise

A próxima fase é o Modelo de Análise e foca-se principalmente nas seguintes categorias: **Conteúdo**, que identifica as classes de conteúdo que devem ser fornecidas para a *WebApp*; **Interação**, que descreve a forma pelo qual o usuário interage com a *WebApp*; **Funcional**, que define as operações aplicadas ao conteúdo da Webapp e a sequência de processamentos consequentes; **Navegação**, que define a estratégia geral de navegação para a *WebApp*; e **Configuração**, que descreve o ambiente operacional e a infraestrutura na qual a *WebApp* reside.

2.1.3 Projeto da *WebApp*

Com base nos resultados do modelo de análise, a próxima fase da Engenharia Web é o Projeto da *WebApp*, que é focado em seis tópicos fundamentais: **interface**, **estética**, **conteúdo**, **navegação**, **arquitetura** e **componentes**.

2.1.4 Implementação

Na fase de implementação, o projeto é propriamente desenvolvido utilizando uma linguagem de programação adequada às suas características e requisitos.

2.1.5 Testes

Por fim, a fase de testes visa garantir que a *WebApp* esteja correta, principalmente em termos de conteúdo (isto é, atende aos requisitos levantados), navegabilidade, segurança, carga, eficiência e interoperabilidade entre diferentes navegadores Web.

2.2 FrameWeb

O FrameWeb (SOUZA, 2007) (*Framework-based Design Method for Web Engineering*) é um método baseado em *frameworks* para o desenvolvimento de sistemas de informação Web (*Web Information System* – WIS). Dentre as diversas propostas para Engenharia Web, até então não havia nenhuma que considerasse diretamente os aspectos característicos dos *frameworks* na construção de WISs, porém o FrameWeb apresenta essa abordagem.

O método não prescreve nenhum processo de desenvolvimento de software específico. A fase de projeto arquitetural concentra as propostas principais do método, a saber:

- definição de uma arquitetura padrão que divide o sistema em camadas, para promover sua melhor integração com os *frameworks* utilizados;
- proposição de um conjunto de modelos de projeto que empregam conceitos utilizados pelos *frameworks* por meio de um perfil UML que aproxima os diagramas da implementação

Na fase de implementação a produção do código é agilizada pela utilização dos *frameworks* e pela fidelidade que existe entre os modelos da fase de projeto e sua implementação.

2.2.1 Arquitetura de software do FrameWeb

A arquitetura lógica padrão para WISs utilizada no FrameWeb é apresentada na Figura 1. O sistema é dividido em três camadas, que apresentam subdivisões internas em pacotes:

- Camada de **Lógica de Apresentação**:

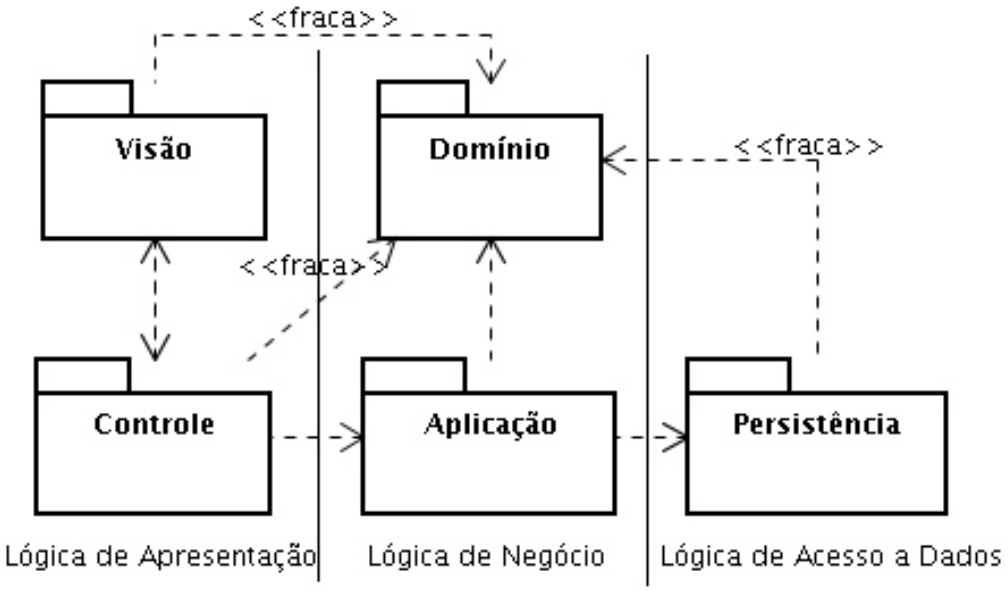

Figura 1 – Arquitetura padrão para WIS baseada no padrão arquitetônico Service-Layer (FOWLER, 2002).

  - Pacote de **Visão**: é onde estão páginas Web, imagens e scripts que são executados do lado do cliente, e também os arquivos exclusivamente relacionados com apresentação de informações aos usuários.
  - Pacote de **Controle**: é responsável pelo monitoramento dos estímulos enviados pelo cliente através dos elementos da Visão, e também abrange as classes de controle e outros arquivos relacionados ao *framework* Controlador Frontal.

- Camada de **Lógica de Negócio**:
  - Pacote de **Domínio**: nele encontram-se as classes que representam o domínio do problema, já modeladas nos diagramas de classe na fase de análise.
  - Pacote de **Aplicação**: nele estão implementados os casos de uso, provendo uma camada de serviços independente da interface com o usuário.

- Camada de **Lógica de Acesso a Dados**:
  - Pacote de **Persistência**: contém as classes responsáveis por armazenar os objetos persistentes em um banco de dados. Tais classes seguem o padrão de projeto *Data Access Object*, ou DAO.

Acerca das relações de dependência entre pacotes, tem-se que:

- O pacote de **Aplicação** manipula objetos do pacote de **Domínio**;

- O pacote de **Aplicação**, por meio do pacote **Persistência**, recupera, grava, altera e exclui objetos de domínio de acordo com a execução dos casos de uso;
- Elementos do pacote de **Visão** enviam os estímulos do usuário (clique de um botão, preenchimento de um campo de texto, acionamento de uma caixa de seleção, etc) para o pacote de **Controle**, que por sua vez responde aos estímulos;
- O pacote de **Controle**, que provê ao usuário acesso às principais funcionalidades, possui dependência com o pacote de **Aplicação**;
- Os pacotes de **Visão, Controle e Persistência** possuem uma dependência fraca com o pacote de **Domínio**: não fazem alterações nos objetos de domínio, apenas os exibem ou os usam para enviar informações nas chamadas de métodos.

2.2.2 Linguagem de modelagem de FrameWeb

O método FrameWeb apresenta uma linguagem de modelagem baseada em UML cujo propósito é guiar a implementação das camadas explicadas na seção anterior, a fim de representar componentes utilizados no desenvolvimento Web e componentes relacionados com o uso de *frameworks*. São propostos quatro tipos de diagrama:

- O **Modelo de Entidades** representa os objetos que fazem parte do domínio do problema e seu mapeamento para a persistência. Ele é desenvolvido a partir do diagrama de classes criado na fase de análise.
- O **Modelo de Persistência** representa os objetos que são responsáveis pela persistência dos dados gerados pelo sistema; isto é, guia a implementação das classes DAO do sistema, as interfaces e suas implementações, juntamente com os métodos específicos de cada uma.
- O **Modelo de Aplicação** representa as classes de serviço, que são responsáveis por implementar a lógica de negócio do sistema, e suas dependências.
- O **Modelo de Navegação** representa as páginas Web do sistema e seus atributos, que interagem com a camada de controle do sistema para administrar os estímulos mandados e recebidos do usuário.

2.2.3 Avanços no FrameWeb: Metamodelos, Editor e Gerador de Código

Foram propostos metamodelos para definir formalmente a linguagem de modelagem do FrameWeb (MARTINS, 2016), 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 originalmente por Souza (2007).

Baseado nesses metamodelos, foi desenvolvida uma ferramenta CASE chamada Editor FrameWeb, que provê um ambiente gráfico para a criação de modelos FrameWeb (CAMPOS, 2017). O que o diferencia de editores UML de propósito geral é que essa ferramenta é totalmente voltada para as características e propriedades do FrameWeb. O editor faz verificações no modelo e impede ações inválidas do ponto de vista do método FrameWeb como, por exemplo, criação de classes fora de pacotes, ou associações indevidas entre componentes, além de não serem disponibilizandos conceitos da UML que não fazem parte do método FrameWeb.

Um passo além foi dado com o Gerador de Código FrameWeb, criado para executar a geração automática de arquivos de código para um WIS modelado no Editor FrameWeb (ALMEIDA, 2017). A entrada do Gerador de Código é um arquivo *.frameweb* produzido no Editor FrameWeb e a saída são os arquivos fonte de código para o sistema. Visou-se criar uma ferramenta flexível, capaz de gerar código para diversas linguagens e *frameworks*. O objetivo principal é reduzir o trabalho manual do programador com boa parte do código.

2.3 Frameworks MVC

Para garantir a manutenibilidade e permitir maior escalabilidade, é fundamental que *WebApps* complexas façam uma separação entre os dados (*Model*) e o layout (*View*). Desse modo, mudanças realizadas no layout das páginas não afetam os dados, e mudanças nos dados não afetam o layout.

O MVC (*Model-View-Controller*) resolve este problema através da separação das tarefas de acesso aos dados e lógica de negócio, lógica de apresentação e de interação com o utilizador, introduzindo um componente entre os dois: o controlador. A Figura 2 ilustra a arquitetura MVC.

Os elementos da *View* (Visão) representam as informações do *Model* (Modelo) para os usuários que, a partir disso, interagem com a aplicação. Essa interação é tratada pelo *Controller* (Controlador), capaz de modificar elementos do Modelo, e também notifica a Visão dessas alterações, que então atualiza a informação apresentada ao usuário.

Porém, na plataforma Web a arquitetura MVC precisa ser levemente adaptada, a esse padrão aplicado a Web dá-se o nome de *Front Controller* (Controlador Frontal), e sua arquitetura é ilustrada na Figura 3

O navegador apresenta as páginas para o cliente, que faz uma requisição HTTP — um pedido de leitura de uma página ou envio de dados para processamento — ao servidor. O servidor Web delega a requisição ao controlador frontal de tratar a requisição e este passa a gerenciar todo o processo (lê a configuração, instancia e executa uma ação). Então

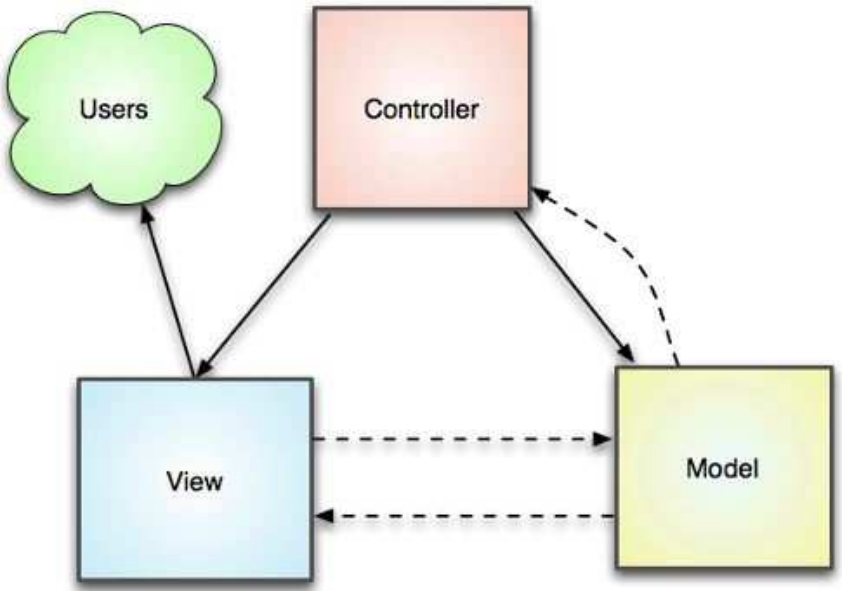

Figura 2 – Diagrama representando a arquitetura MVC.

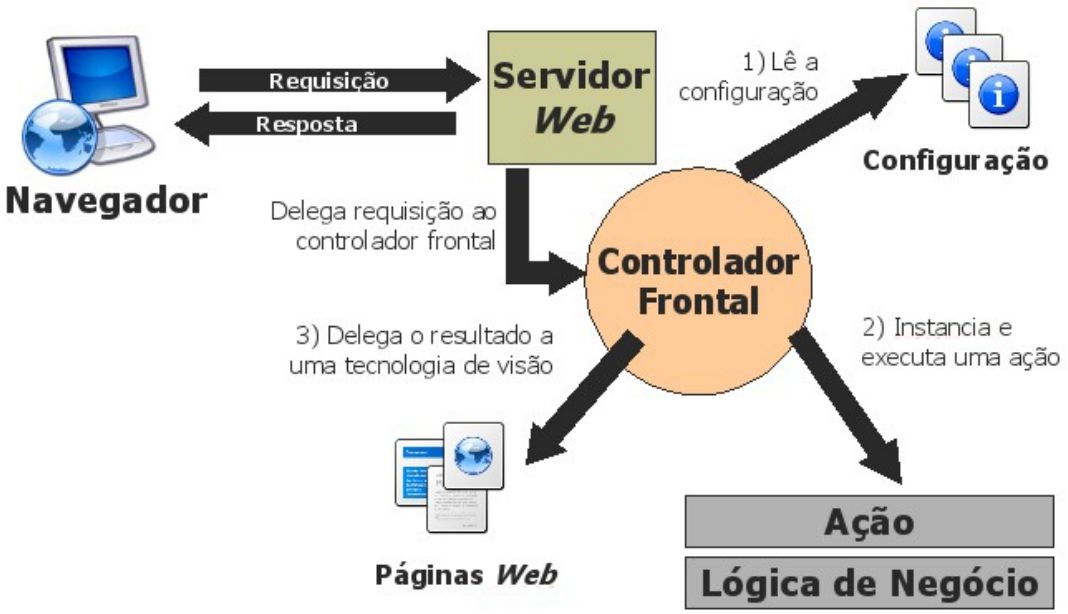

Figura 3 – Representação de um Controlador Frontal na Web (SOUZA, 2007).

é emitida uma resposta apropriada à requisição. O resultado é delegado a uma tecnologia de visão para ser exibido para o cliente pelo navegador. Os *frameworks* MVC fornecem um controlador frontal a ser configurado pelo desenvolvedor para melhor adaptação ao seu projeto.

2.3.1 Framework Play

Play é um *framework* MVC *open-source* para aplicações Web, escrito em Scala mas também usável em Java, de alta produtividade, que integra componentes e APIs para o desenvolvimento moderno de aplicativos da Web (PLAY-TEAM, 2017).

Entre suas características, podemos citar que o *Play* é totalmente RESTful; isto é, ele segue os princípios da Transferência de Estado Representacional, usando o protocolo HTTP de forma explícita e representativa para se comunicar em todas as requisições. Além disso, não possui estado entre comunicações, ou seja, cada comunicação é independente e uniforme (padronizada) e passa toda a informação necessária; isto torna o *Play* mais escalável que outros *frameworks*. Também apresenta I/O assíncrono, conseguindo disponibilizar e tratar uma enorme quantidade de requisições assíncronas.

Outros detalhes mais particulares do *framework Play* serão discutidos no Capítulo 4 dentro do contexto da implementação do projeto.

2.4 Linguagem Scala

Scala é uma linguagem que combina orientação a objetos e programação funcional em uma linguagem concisa de alto nível. Sua tipagem estática ajuda a evitar bugs em aplicações complexas. Sua interoperabilidade com Java também permite ao usuário construir sistemas de alta performance com fácil acesso a imensos ecossistemas de bibliotecas (SCALA, 2019).

Detalhes mais particulares da linguagem Scala serão discutidos no Capítulo 4 dentro do contexto da implementação do projeto.

3 O Sistema SCAP

O SCAP é um aplicativo Web cujo objetivo é apoiar o Departamento de Informática (DI) da UFES no gerenciamento e registro de solicitações de afastamento de seus professores para eventos no Brasil ou no exterior.

Para eventos realizados no Brasil, o processo corre internamente no DI. Os funcionários do departamento são informados por e-mail, e caso ninguém apresente um parecer contrário num prazo de dez dias, o pedido é aprovado.

No caso de uma solicitação de afastamento para um evento no exterior, é escolhido um professor que não possua relação de parentesco com o solicitante para ser o relator do pedido. Após a emissão do parecer do relator, o processo é avaliado pelo DI no mesmo molde do caso acima. Caso o pedido seja aprovado no DI, ele é então encaminhado para o Centro Tecnológico (CT) e para a Pró-reitoria de Pesquisa e Pós-Graduação (PRPPG), e, caso aprovado, o afastamento é publicado no Diário Oficial da União. O SCAP é uma ferramenta a ser utilizada dentro do DI, portanto, decisões competentes a outras instâncias seriam apenas transmitidas no sistema por um secretário do DI.

Baseado nos requisitos já levantados por Duarte (2014) e Prado (2015), este capítulo apresenta os modelos de casos de uso e diagramas de classe referentes ao sistema SCAP.

3.1 Modelos de Casos de Uso

A Tabela 1 lista e descreve os atores identificados.

Tabela 1 – Atores do SCAP.

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

Os **secretários** são responsáveis pela parte administrativa do sistema. Fazem o cadastro de professores, registram as relações de parentesco ente eles e fazem o cadastro dos mandato de chefe do departamento, podendo excluir relações de parentescos ou mandatos cadastrados erroneamente. O secretário também é responsável por integrar o sistema com setores externos ao DI, cadastrando pareceres provenientes do CT e da PRPPG. Além disso, eles arquivam solicitações de afastamento já finalizadas.

Os **professores** cadastram suas próprias solicitações de afastamento no sistema, e podem se manifestar contrariamente a solicitações de afastamento de outros professores dentro do prazo. Se um professor é designado para ser relator de uma solicitação de afastamento internacional, ele deve emitir um parecer favorável ou contrário à tal solicitação.

Chefe de departamento é uma função exercida por um professor durante um mandato temporário. É incumbência do chefe de departamento designar relatores para solicitações de afastamento internacional.

Usuário é qualquer usuário do sistema, seja ele professor — chefe de departamento ou não — ou secretário.

Na Figura 4 é apresentado o diagrama de casos de uso do SCAP.

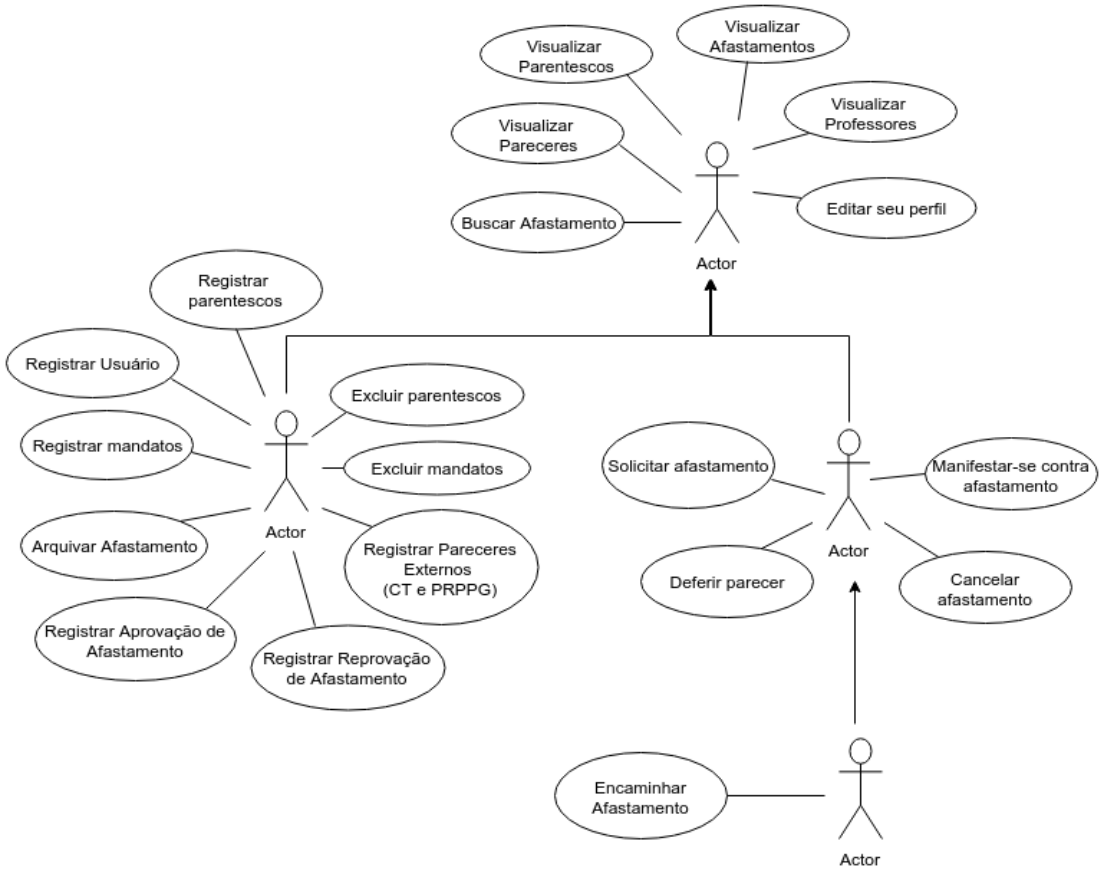

Figura 4 – Diagrama de Casos de Uso do SCAP.

Qualquer **usuário** do sistema pode **Visualizar Afastamentos**, **Visualizar Pareceres**, **Visualizar Parentescos**, **Visualizar Professores**, **Buscar Afastamentos** por professor, status ou relator, e **Editar seu Perfil**.

No caso de uso **Solicitar Afastamento**, o professor (autor do pedido) deve preencher um formulário, informando o nome do evento, o período de afastamento, o período do evento, o motivo do afastamento, o ônus para a instituição, informar se o

evento é no Brasil ou no exterior e a cidade onde será realizado. Também pode fazer o *upload* de documentos pertinentes. A solicitação é cadastrada no sistema e os professores do DI são notificados a respeito dela pelo recebimento de um e-mail automático.

Nesse ponto, outro professor pode **Manifestar-se Contra Afastamento** e, nesse caso, após uma reunião do DI, o Secretário **Registra Aprovação do Afastamento** ou **Registra Reprovação do Afastamento**. Caso nenhum professor se manifeste durante o prazo de dez dias, a solicitação é automanticamente aprovada pelo DI.

No caso de solicitações para eventos no Brasil, o processo segue como descrito acima. Porém, caso a solicitação seja para um evento fora do Brasil, existem outras etapas adicionais além dessas. Assim que a solicitação é cadastrada no sistema, um e-mail automaticamente é enviado ao chefe do departamento, que deve **Encaminhar o Afastamento** para outro professor do DI de sua escolha que atuará como relator (também notificado automaticamente por e-mail).

O relator vai preencher um formulário para **Deferir Parecer**. O parecer é cadastrado, o sistema gera um documento contendo esse parecer e o professor que fez a solicitação é notificado por e-mail. Caso o parecer seja negativo, uma reunião do DI é marcada para decidir se a solicitação será realmente reprovada.

A qualquer momento, o professor que solicitou o afastamento pode **Cancelar Afastamento**. Um e-mail é enviado notificando o(a) chefe do departamento sobre o cancelamento.

Um secretário pode **Registrar Usuários**, informando nome, matrícula, e-mail, senha e tipo de usuário (professor ou secretário). Pode também **Registrar Mandatos**, registrando um professor já previamente cadastrado como chefe ou subchefe de departamento, e informando a data de início e fim previsto do seu mandato.

Para casos de solicitações de afastamento relativas a eventos no exterior, fica a cargo dos secretários **Registrar Pareceres Externos**, do CT e da PRPPG. Em ambos os casos, o secretário cadastra o parecer e faz *upload* dos documentos enviados contendo o parecer. Baseado no resultado do parecer, o status do afastamento é alterado para reprovado ou aprovado.

Quando o afastamento termina e o professor retorna, o Secretário tem a incumbência de **Arquivar Afastamento**, alterando o status da solicitação.

3.2 Modelo de Classes

A primeiro etapa na análise de requisitos é modelar as entidades do mundo real dentro do paradigma de Orientação a Objetos. As entidades são descritas por classes, suas características são manifestas nos atributos das classes, e as interações entre elas são

representadas pelas relações entre as classes. É fundamental destacar a multiplicidade de relações entre as classes, o que deve ser retratado na persistência de dados.

O Modelo de Classes do SCAP é apresentado na Figura 5.

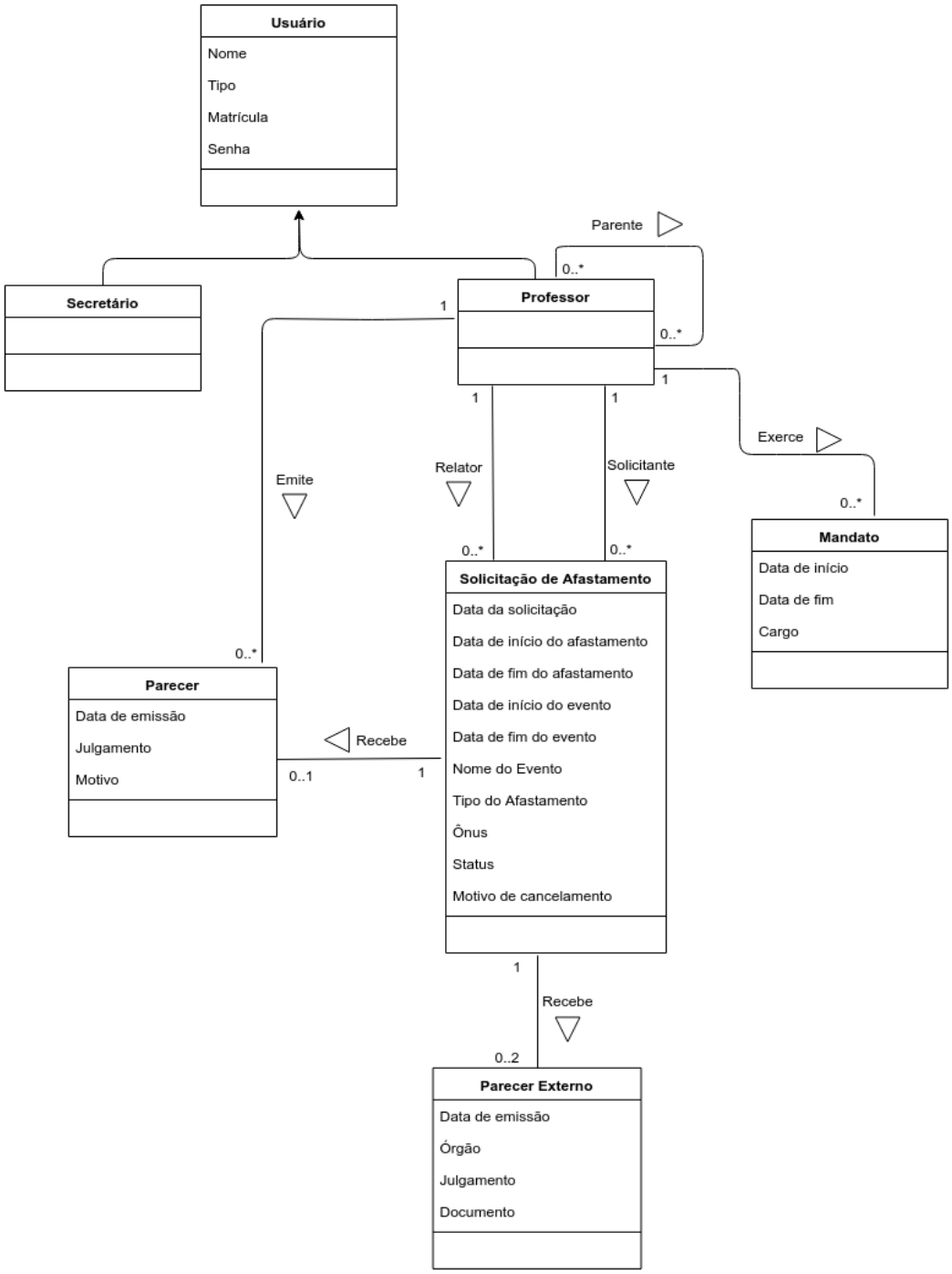

Figura 5 – Diagrama de Classes do SCAP.

As classes **Professor** e **Secretário** são subclasses de **Usuario** e representam os professores e secretários do DI.

A classe **Parentesco** é usada para representar relação de parentesco — sanguínea

ou matrimonial — entre professores. Essa classe não possui atributos pois, como o foco do trabalho são os afastamentos e não os professores, não foi considerado importante guardar a informação de qual exatamente é o parentesco entre dois professores. Basta saber que, havendo um parentesco de qualquer tipo, um professor fica inelegível para ser relator da solicitação de afastamento de outro professor.

A classe **Mandato** representa o mandato de um professor designado para o cargo de chefe ou subchefe de departamento.

A classe **Afastamento** representa um pedido de afastamento solicitado por um professor. Se a solicitação for para um evento no exterior, é necessário que outro professor seja designado **Relator** e seu parecer é descrito pela classe **Parecer**. Cada parecer é emitido por um único professor em relação a um único afastamento. Embora um professor possa emitir vários pareceres para vários afastamentos diferentes. Uma manifestação contrária a um evento nacional é representada também por um parecer.

Os pareceres advindos do CT e da PRPPG são representados pela classe **Parecer Externo**. Esses pareceres possuem um documento associado (representado por um atributo da classe).

Feito o diagrama de classes, a próxima etapa é listar as restrições de integridade, que consistem em relacionamentos entre as entidades que não podem ser representados no diagrama de classes, mas são fundamentais para a compreensão e implementação do sistema. As restrições de integridade pertencentes ao sistema SCAP são listadas a seguir:

- Um professor não pode ser relator de um afastamento solicitado por um parente ou por ele mesmo;
- Não pode haver mais de dois professores (chefe e subchefe de departamento) exercendo um mandato simultaneamente;
- 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;
- Secretários do departamento não podem criar solicitação de afastamento;
- Um mesmo usuário do SCAP não pode ser professor e secretário.

4 Projeto e Implementação

A fase de projeto envolve a modelagem de como o sistema será implementado de fato com a adição dos requisitos tecnológicos e de caráter não funcional. Esses recursos são intimamente correlacionados às plataformas de implementação adotadas, tais como linguagem de programação, *frameworks*, mecanismo de persistência, entre outros.

Na Seção 4.1 serão apresentadas as tecnologias utilizadas; na Seção 4.2 serão mostrados os modelos FrameWeb produzidos para essa implementação; na Seção 4.3 será mostrada a arquitetura do sistema com suas camadas; e finalmente na Seção 4.4 será apresentado o sistema por meio de capturas de tela e explicações de como funciona a navegação.

4.1 Tecnologias Utilizadas

Como já previamente mencionado, essa versão do SCAP foi implementada na linguagem **Scala**, utilizando o *framework* **Play**. Também usou-se a IDE **Eclipse** como ambiente de desenvolvimento para facilitar a implementação.

Para persistência de dados, isto é, o mapeamento entre as tabelas no banco de dados e os objetos, foi utilizada a ferramenta **Slick**. *Slick* é uma moderna biblioteca de consulta e acesso a banco de dados para a linguagem Scala que permite que o usuário trabalhe com os dados no banco praticamente como se estivesse usando coleções Scala, mas ao mesmo tempo com total controle de quando o acesso ao banco acontece e quais dados são transferidos. O usuário pode escrever suas consultas ao banco em Scala ao invés de SQL, com o benefício da verificação estática em tempo de compilação, e da composicionalidade da linguagem Scala.

Slick provê códigos para diferentes servidores, mas a API utilizada primariamente no projeto foi a de integração com o **MySQL**, o sistema de gerenciamento de banco de dados utilizado no projeto.

4.2 Modelos FrameWeb

Nesta seção, serão apresentados os modelos de entidades, persistência, aplicação e navegação do FrameWeb.

4.2.1 Modelo de Entidades

O modelo de entidades do SCAP é apresentado na Figura 6. Para cada entidade, existe um tipo 'simples' que interage com o Banco de Dados, e um tipo 'completo' que é o tipo que de fato será utilizado na interface com o usuário em camadas superiores.

Por exemplo, no tipo 'simples' da solicitação de afastamento, professor e relator são representados simplesmente pelas suas IDs em um valor **Long**; já no tipo 'completo', um professor e um relator são objetos completos do tipo **Usuário**. Numa solicitação 'simples', as datas de início e término de afastamentos e eventos são representadas pelo tipo **Timestamp**; na solicitação 'completa' elas são representadas pelo tipo **LocalDate**.

Os tipos enumerados do SCAP também estão presentes no pacote de modelo, e são listados conforme mostra a Figura 7.

No tipo **Onus**, TOTAL indica que a universidade arcará com os custos da viagem relacionada ao afastamento; PARCIAL significa que a universidade apenas manterá a remuneração normal do professor, porém não custeará a viagem; e INEXISTENTE representa o caso em que mesmo a remuneração do professor é suspensa durante o afastamento. **TipoAfastamento** apresenta valor NACIONAL para eventos realizados no Brasil e INTERNACIONAL para eventos no exterior. Em **TipoParecer** os valores FAVORÁVEL e DESFAVORÁVEL indicam que o parecer é a favor ou contra o afastamento, respectivamente (PRADO, 2015).

O **TipoCargo**, com os valores de CHEFE e SUBCHEFE, é necessário para melhor garantir a restrição de integridade de que no máximo dois professores podem executar mandatos simultaneamente, dado que estejam em cargos diferentes. Entretanto, o sistema não faz distinção na prática entre chefe e subchefe, ambos podem executar as mesmas atribuições.

O **TipoSetor**, com os valores de CT e PRPPG serve para designar qual é o setor que emitiu determinado parecer externo.

O **TipoUsuario** foi criado para separar PROFESSOR e SECRETÁRIO. Como não há nenhum atributo específico de modelo que separa professor de secretário resolveu-se fazer a distinção entre os dois por tipo enumerado ao invés de herança de classe apenas por conveniência, facilitando a implementação da classe no pacote de persistência. Embora o diagrama de classes da seção anterior apresente a relação por herança, o modelo de domínio foi modificado para refletir a mudança na implementação.

O tipo **StatusSolicitacao** indica em qual estágio o afastamento se encontra. Foram modelados dois diagramas de estado para explicar quando e como ocorrem as mudanças de um status para outro.

A Figura 8 representa o diagrama de estados de uma solicitação de afastamento

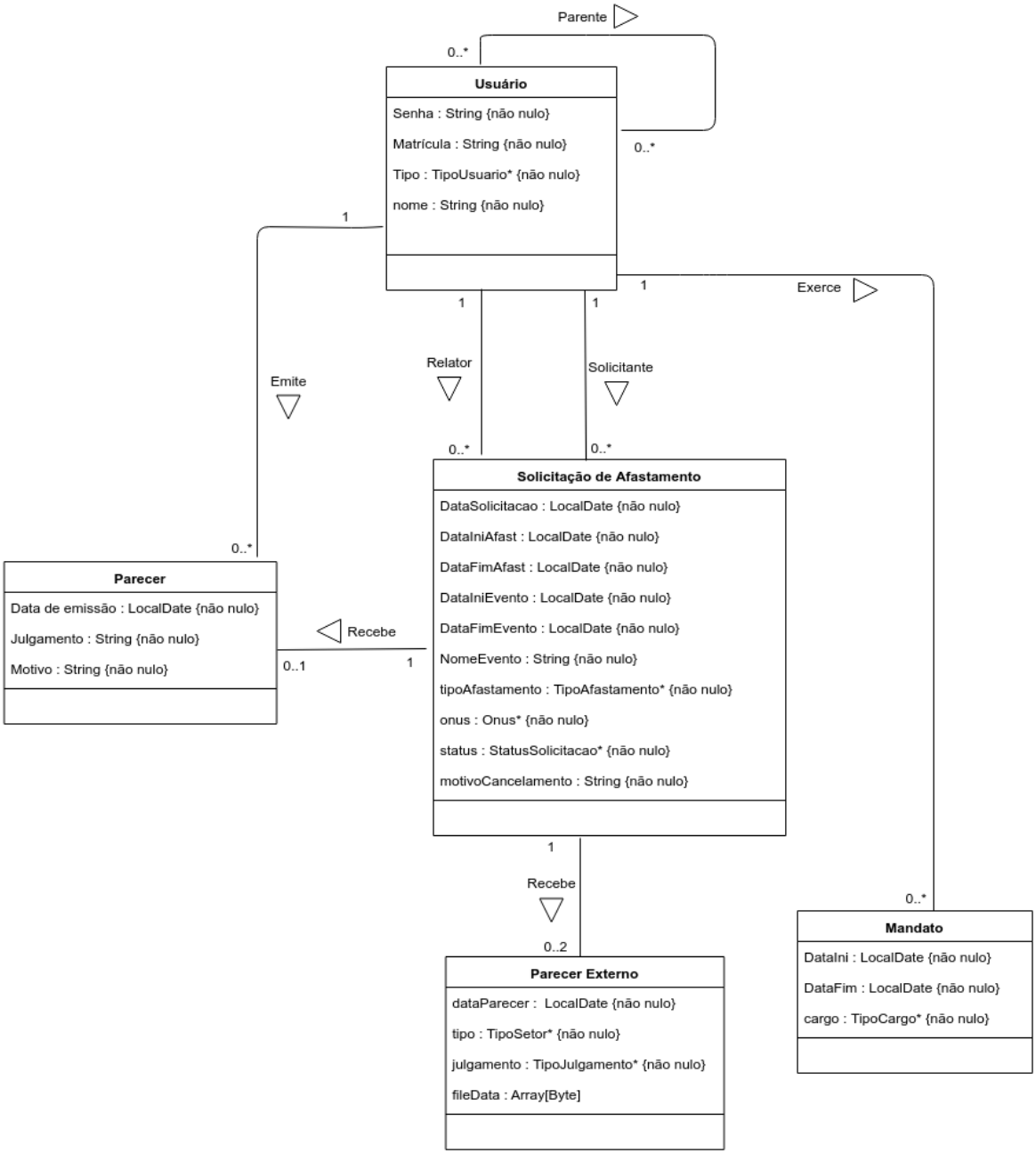

Figura 6 – Modelo de Domínio do SCAP

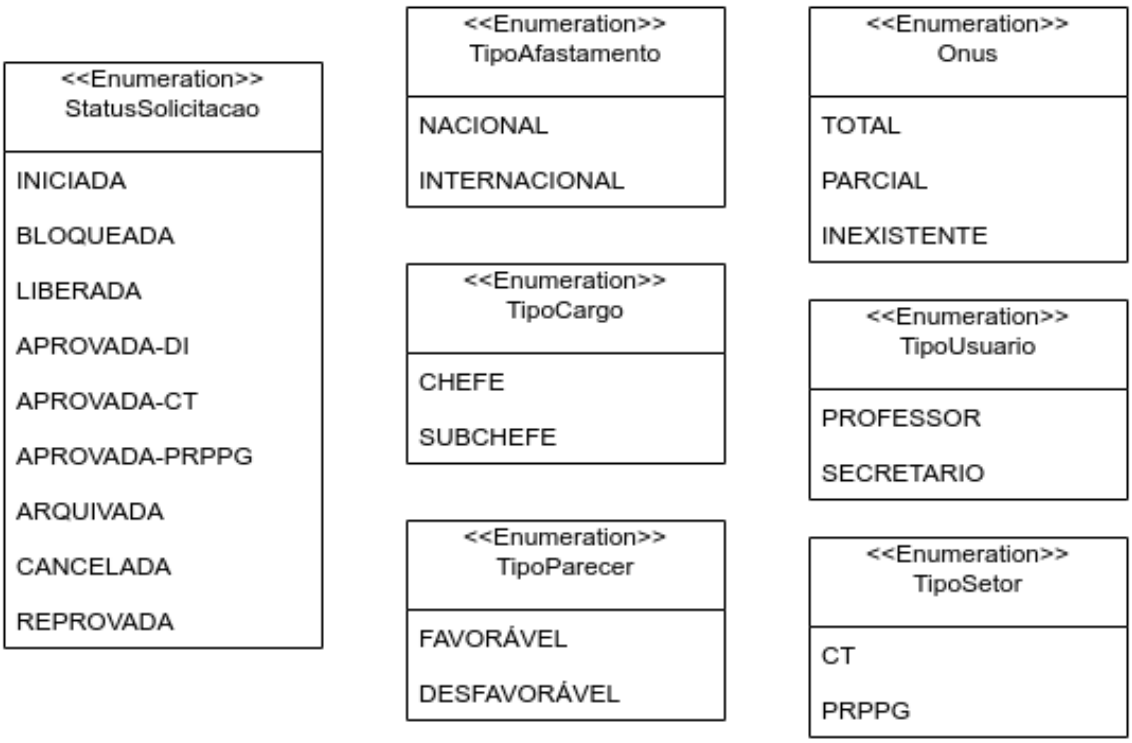

Figura 7 – Tipos Enumerados do SCAP

nacional enquanto a Figura 9 representa o diagrama de estados de uma solicitação de afastamento internacional.

Um detalhe importante é que cada tipo enumerado é um *object* que estende a classe abstrata **Enumeration**. Isso será explicado com maior detalhamento em seções seguintes, mas cada *object* é unico, não sendo criado um elemento separado para cada uso. Instâncias de classes que fazem uso desses tipos enumerados na verdade o fazem como *strings*, porém com uma conversão forçada do valor enumerado para *string*. Portanto, no modelo de entidades, as instâncias de tipo enumerado foram representadas com asterisco para relembrar esse detalhe.

Para quem gostaria de usar tipos enumerados como instâncias particulares, bastaria importar a classe Enum de Java e usá-la, essa opção está sempre disponível. Optou-se por usar os tipos enumerados próprios do Scala para apresentar suas diferenças em relação a versão tradicional.

4.2.2 Modelo de Persistência

O modelo de persistência do sistema é apresentado na Figura 10. Os métodos **save** e **update** têm como parâmetro qualquer elemento possível (**Any**) porque essa é a assinatura desses métodos em **BaseDAO**. Porém dentro do método **save** e **update** em

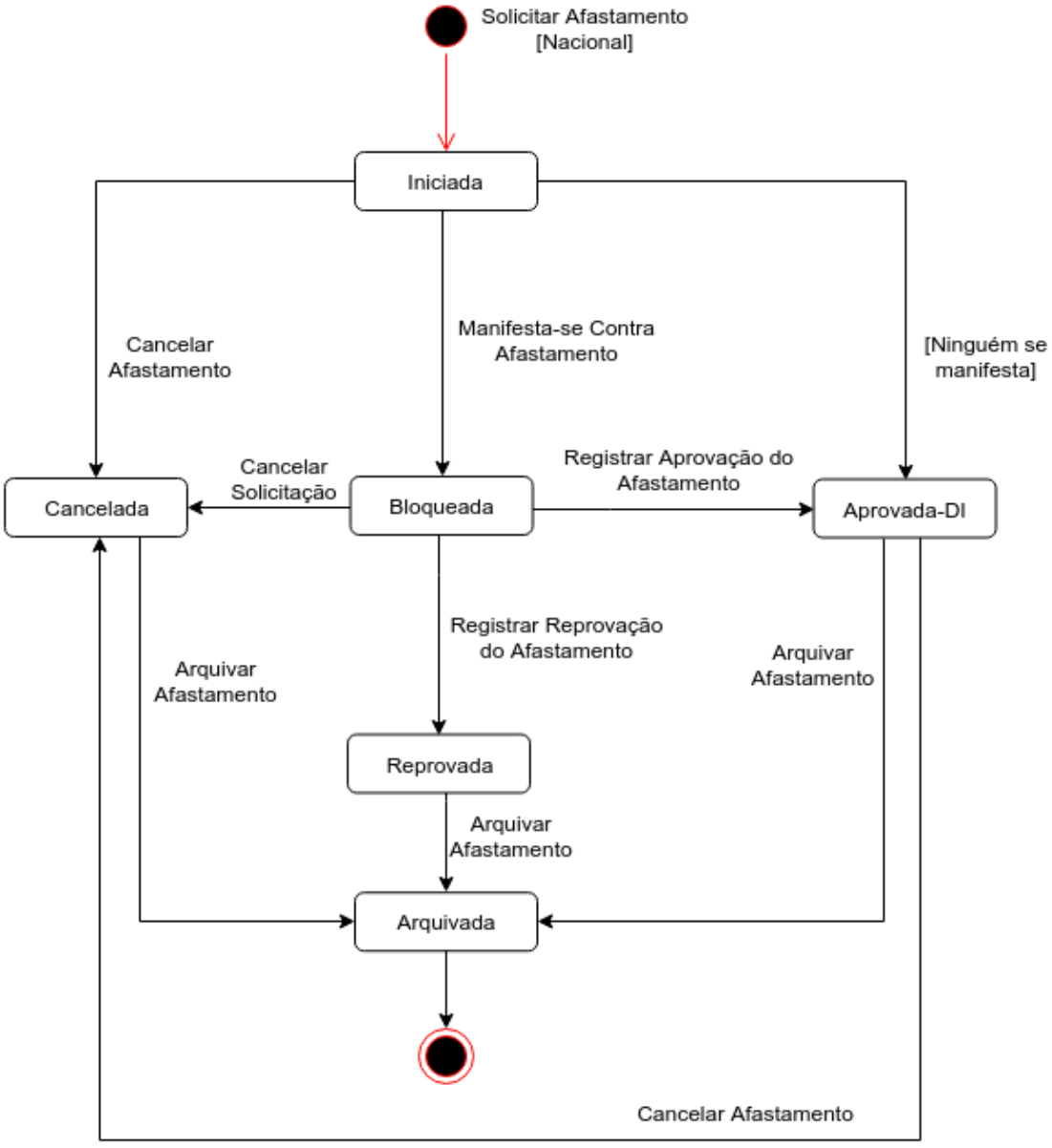

Figura 8 – Diagrama de estados para solicitação de afastamento nacional no SCAP

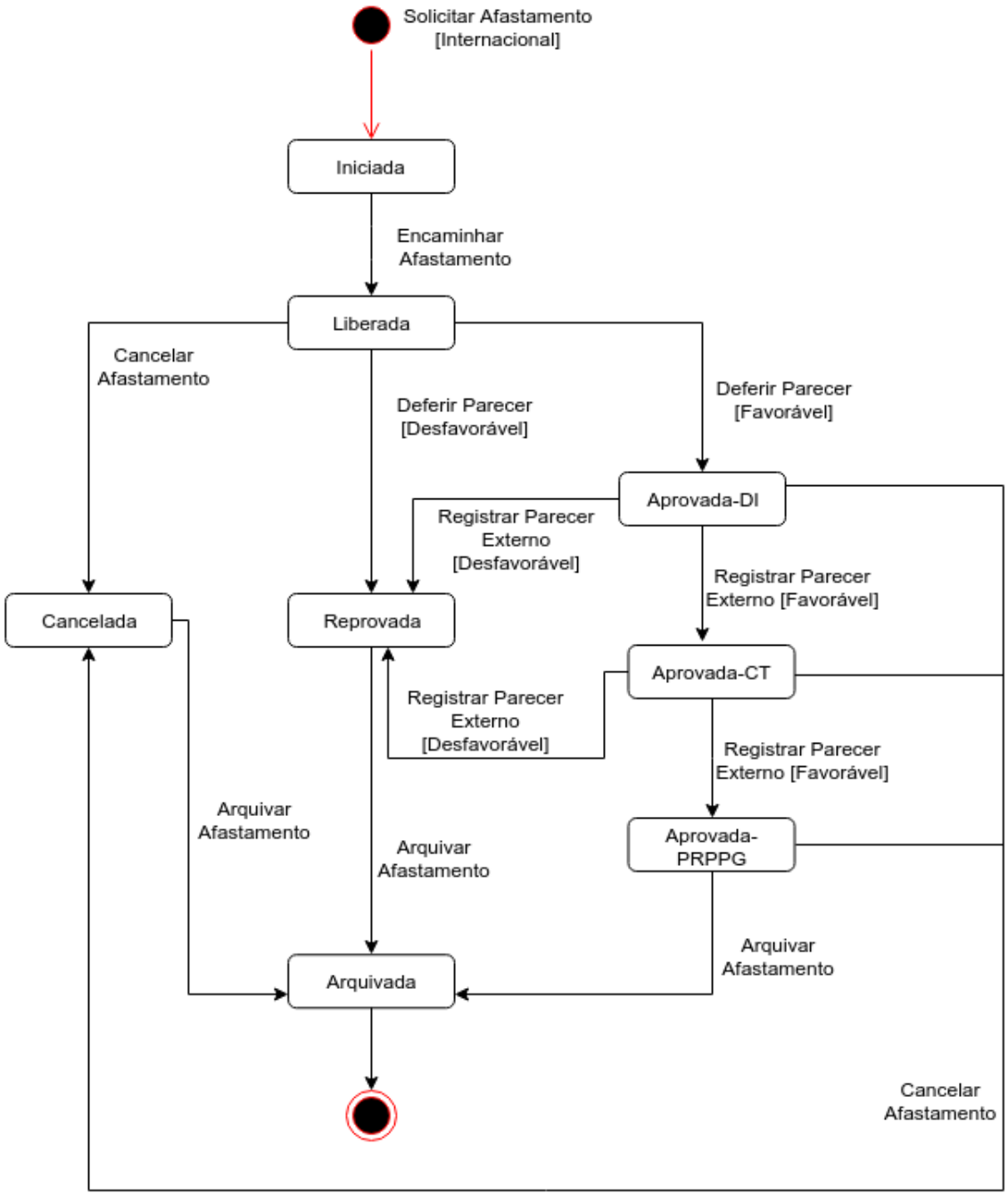

Figura 9 – Diagrama de estados para solicitação de afastamento internacional no SCAP

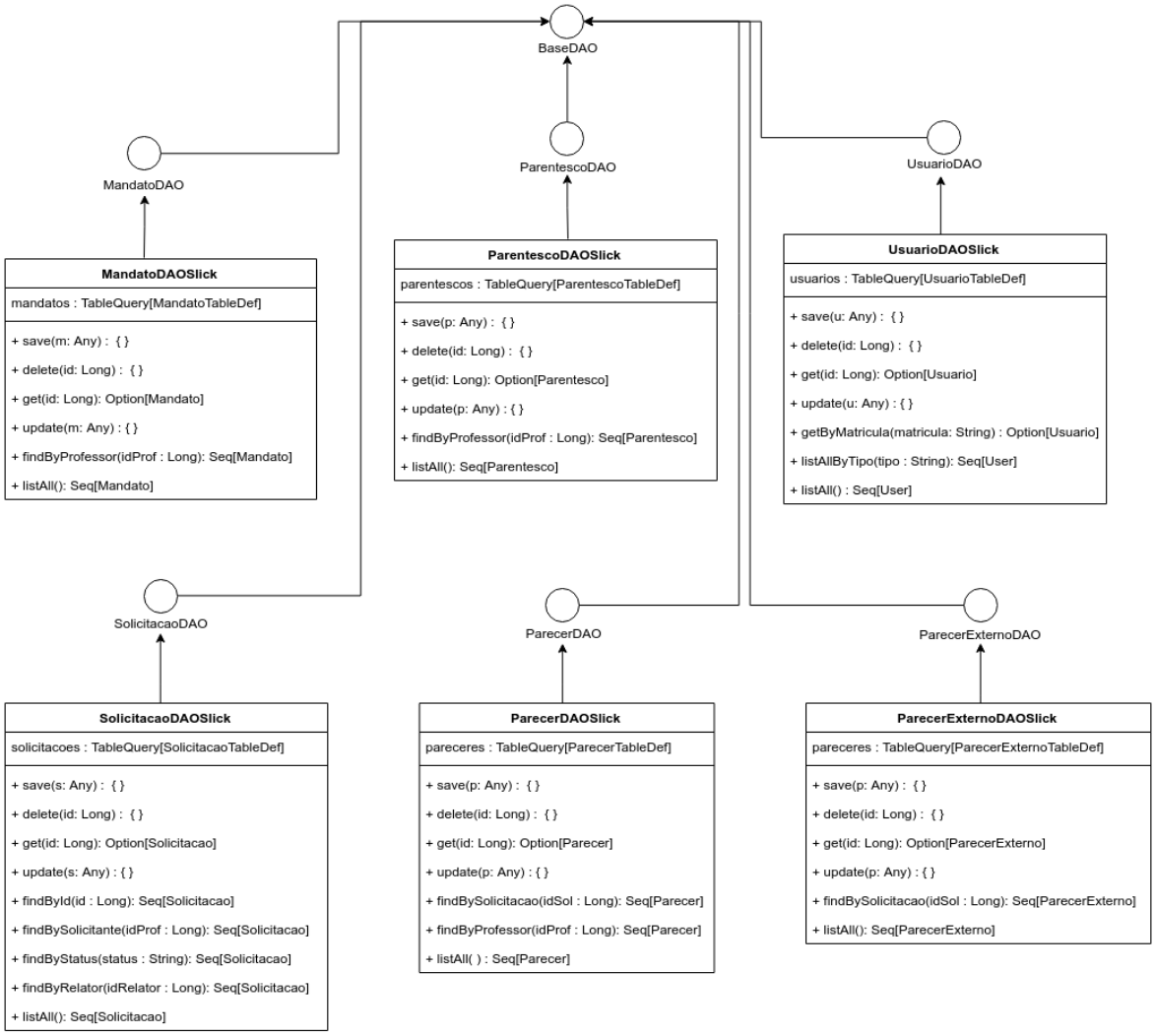

Figura 10 – Modelo de Persistência do SCAP

cada classe, o elemento deve ser obrigatoriamente convertido em um elemento pertencente à própria classe antes de usado.

Além dos métodos básicos do CRUD, outros métodos foram adicionados às classes de persistência por demanda, especialmente métodos de busca.

4.2.3 Modelo de Aplicação

O modelo de aplicação é o responsável por mostrar como as classes da camada de aplicação — que implementam a lógica de negócio do sistema, descrita nos casos de uso — se relacionam com as classes de controle e de persistência.

As classes de controle (*controller*) dependem das classes de aplicação (*service*) e estas, por sua vez, dependem das classes de persistência (sufixo DAO, representadas no modelo pelos *traits* que as definem) para salvar, atualizar, recuperar ou remover (CRUD) os objetos afetados pelos estímulos gerados pelo usuário do SCAP.

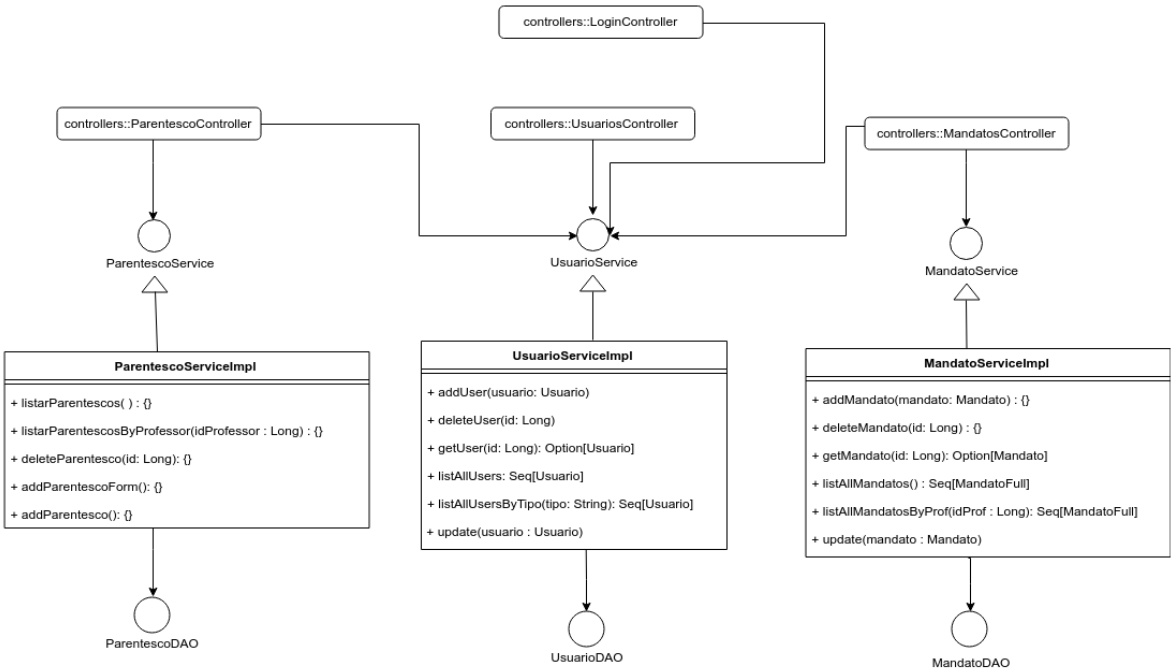

Figura 11 – Modelo de Aplicação do SCAP - Aplicações relevantes a usuários, mandatos e parentescos

Para uma melhor apresentação visual, são apresentados dois modelos. A Figura 11 mostra as aplicações relevantes a usuários, mandatos e parentescos. A Figura 12 mostra as aplicações relevantes a solicitações, pareceres internos e externos, e-mails e usuários (as aplicações referentes a usuários são representadas duas vezes pois são relevantes nos dois contextos).

Nessa versão do SCAP, optou-se por permitir que uma classe de controle pudesse ser dependente de múltiplas classes de aplicação, enquanto uma classe de aplicação é ligada a no máximo uma única classe de persistência, para facilitar a implementação.

4.2.4 Modelos de Navegação

Modelos de Navegação são diagramas de classes da UML que representam os diferentes componentes da camada de Lógica de Apresentação, como páginas Web, formulários HTML e classes de ação do *framework Front Controller*. Esses modelos guiam a codificação das classes e componentes dos pacotes de Visão e Controle (SOUZA, 2007).

Como exemplos, apresentamos aqui modelos de navegação para três casos de uso: Registrar parentesco, Registrar Parecer Externo e Editar seu Perfil.

Para registrar parentesco, inicialmente a partir do menu principal, o secretário vai para a página Web onde preenche um formulário com um novo parentesco ao selecionar dois professores distintos que já não tenham parentesco cadastrado previamente. Terminado o processo, o usuário é redirecionado de volta para o menu principal. Este modelo de

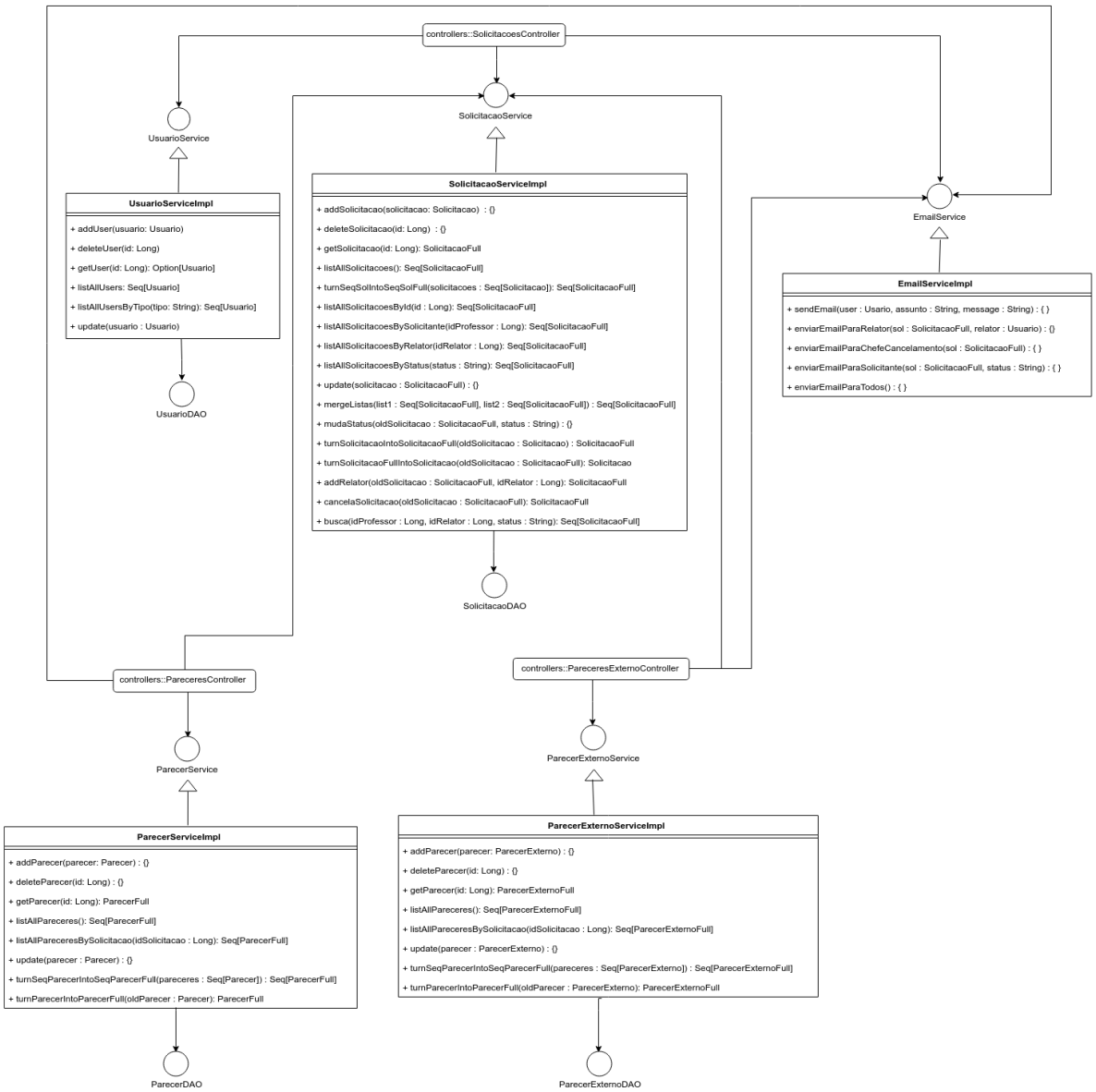

Figura 12 – Modelo de Aplicação do SCAP - Aplicações relevantes a solicitações, pareceres internos e externos, e-mails e usuários

navegação é mostrado na Figura 13.

Para registrar parecer externo, inicialmente a partir do menu principal, o secretário vai para a página Web na qual são listadas as solicitações e escolhe visualizar uma solicitação específica, sendo direcionado para uma página Web que exibe informações mais detalhadas sobre a solicitação escolhida. Dado que a solicitação seja para um evento internacional e seu status atual seja **aprovada-DI** ou **aprovada-CT**, há uma opção para o usuário adicionar o parecer, preenchendo o formulário ao selecionar o julgamento (Favorável ou Desfavorável), fazer o *upload* do documento associado, enquanto o órgão emissor já é pré-determinado (CT caso o status atual da solicitação seja **aprovada-DI** ou PRPPG caso seja **aprovada-CT**). Concluído o processo, o secretário é redirecionado de volta para

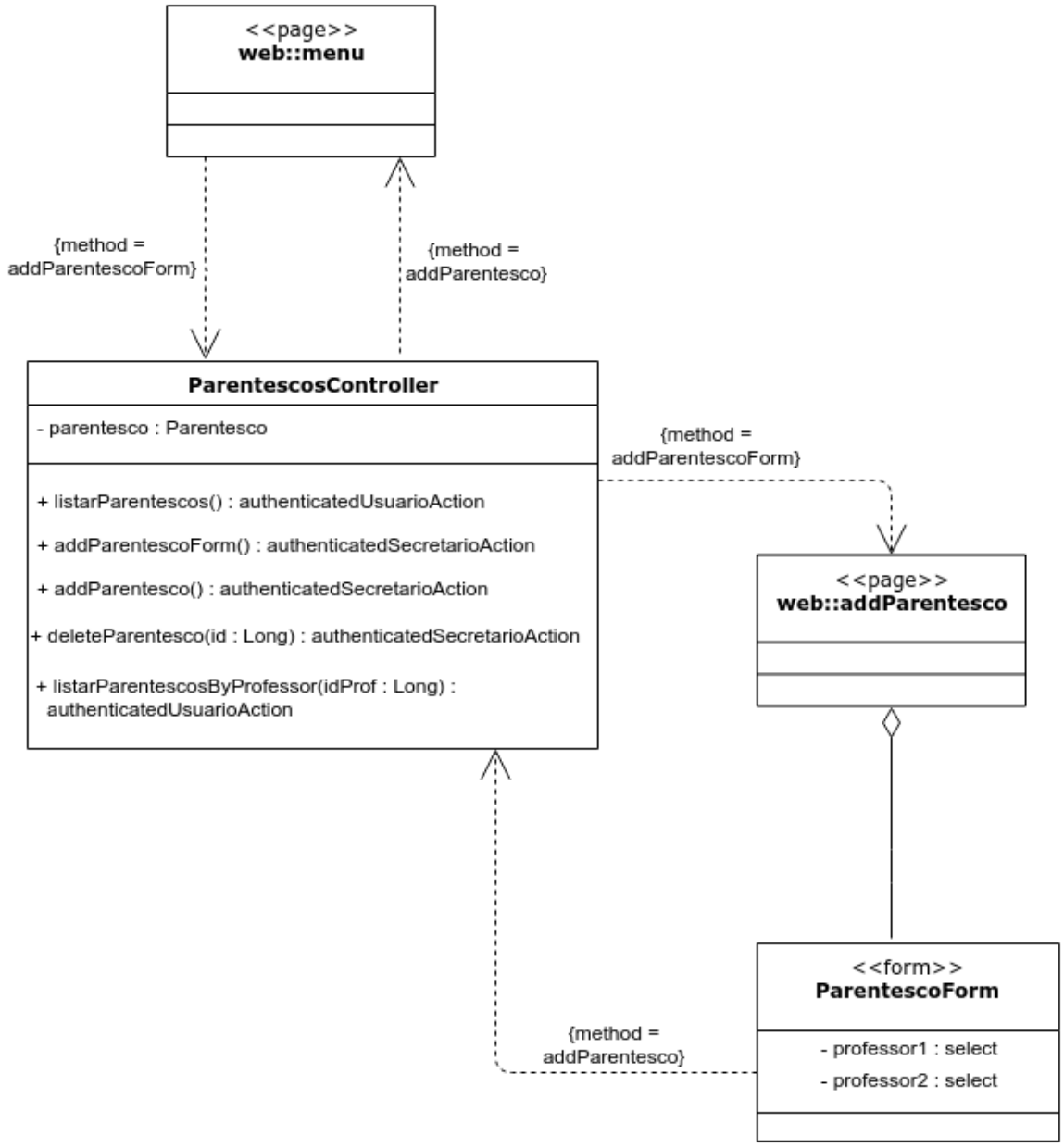

Figura 13 – Modelo de Navegação do SCAP - Registrar Parentesco

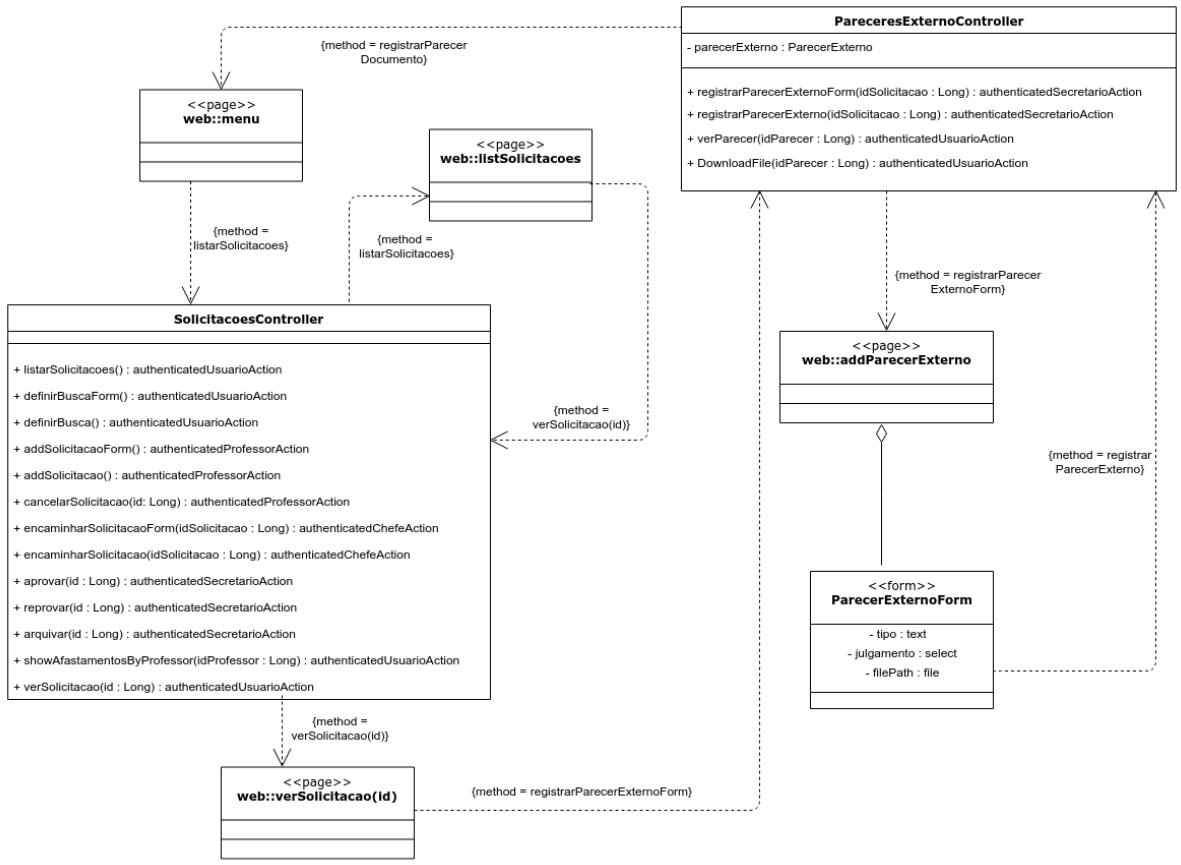

Figura 14 – Modelo de Navegação do SCAP - Registrar Parecer Externo

o menu principal. Este modelo de navegação é mostrado na Figura 14.

Para editar seu perfil (um usuário, seja ele professor ou secretário, pode editar apenas seu próprio perfil), inicialmente a partir do menu principal, o usuário vai para a página Web na qual é apresentado um fórmulário com seu nome, e-mail e senha, por meio do qual ele modifica os campos que quiser e confirma a edição, sendo direcionado de volta para o menu principal. Este modelo de navegação é mostrado na Figura 15

Cada formulário tem um objeto *form* e uma classe *case* específicos para receber seus dados, embora isso não seja representado no modelo de navegação por questões didáticas.

4.3 Arquitetura do Sistema

O SCAP foi subdividido em três camadas — Lógica de Apresentação, que abrange os pacotes de Controle e Visão; Lógica de Negócio, que abrange os pacotes de Aplicação e Domínio; e Lógica de Acesso a Dados, que representa o pacote de persistência — seguindo a arquitetura do FrameWeb previamente explicada no Capítulo 2.

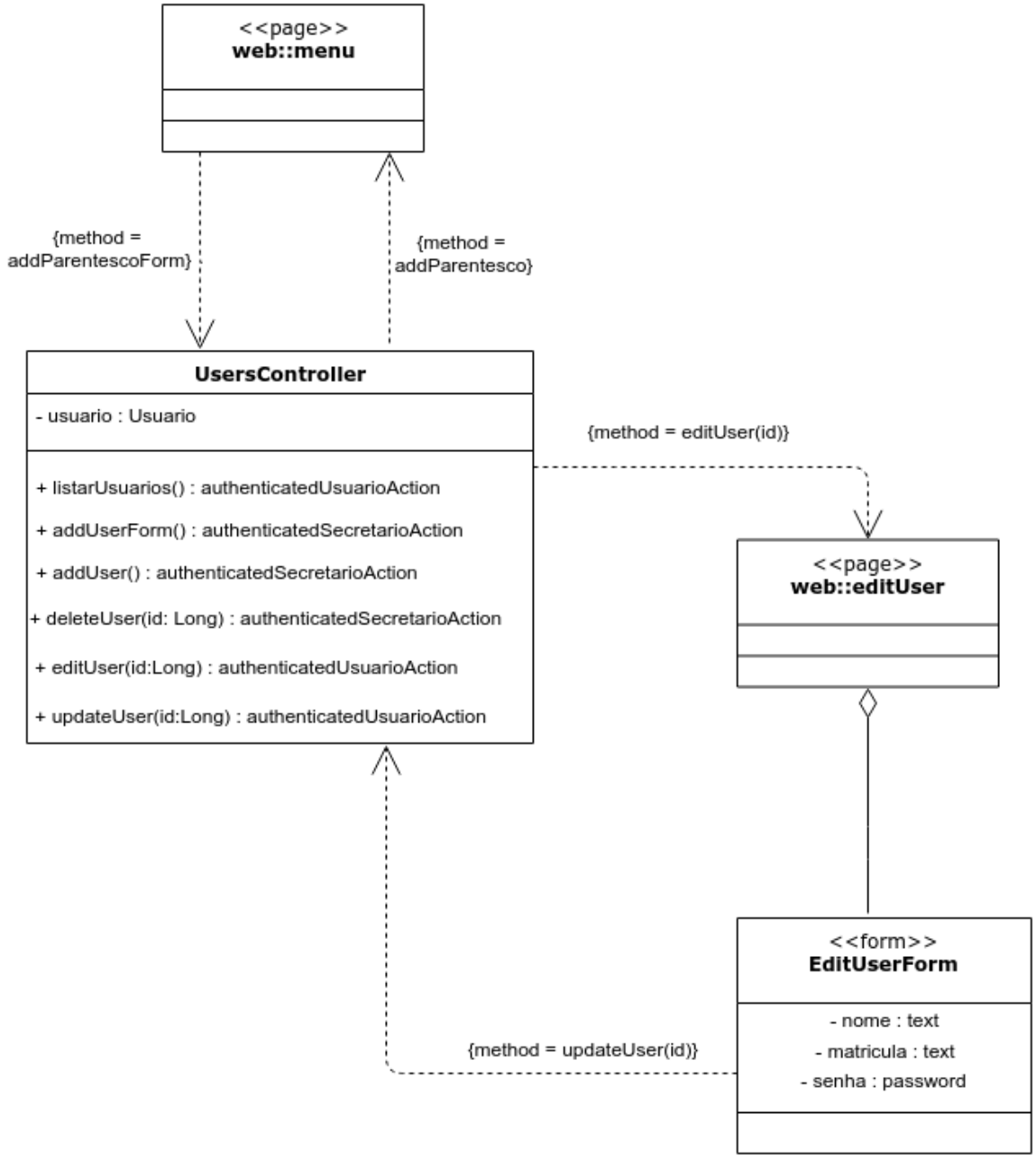

Figura 15 – Modelo de Navegação do SCAP - Editar seu perfil

4.3.1 Camada de Lógica de Apresentação

Nesta subseção, será explicado o funcionamento da camada de apresentação (com seus pacotes de Controle e Visão), destacando como as particularidades do *framework Play* e da linguagem Scala influenciaram a sua implementação.

4.3.1.1 Ações simples, Ações Customizadas e Autenticação

A maioria das requisições recebidas por uma aplicação *Play* são respondidas por meio de Ações (*Actions*). Uma Ação é basicamente uma função que recebe uma requisição

e gera um resultado a ser enviado para um cliente.

Um exemplo de uma Ação simples no contexto do SCAP se encontra na função de login, conforme mostra a Listagem 4.1.

Listagem 4.1 – Exemplo do uso de uma Ação no SCAP

```
1  import play.api.mvc._
2
3  def loginForm() = Action {
4      Ok(br.ufes.scap.views.html.login(UserLoginForm.form))
5  }
```

Quando essa função é chamada — o que acontece por padrão sempre que o sistema é acessado sem nenhuma URL específica — o usuário é direcionado para a tela de login, exibida na Figura 16.

Figura 16 – Tela de login do SCAP.

Porém, usar ações simples não é adequado para a maioria dos métodos, pois muitas vezes certo usuário não tem permissão para executar certo método. Nesses casos, mesmo que não haja uma navegação direta no sistema que permita ao usuário usar o método, ainda seria possível usá-lo diretamente via URL. A melhor maneira de evitar isso utilizando os recursos do *framework* Play é usar **ActionBuilders** customizados. Foram criados quatro ações customizadas para o SCAP:

- **AuthenticatedUsuárioAction**: permite que qualquer usuário — professor ou secretário — autenticado no sistema possa chamar a função. Este tipo de ação é usado em funções como: listar usuários, listar mandatos, listar parentescos, etc.;
- **AuthenticatedProfessorAction**: permite que apenas professores autenticados no sistema possam chamar a função. É usada em funções como: adicionar solicitação de afastamento, manifestar-se contra uma solicitação de afastamento, etc.;

- **AuthenticatedChefeAction**: permite que apenas professores autenticados no sistema e que estão no exercendo mandato de chefe ou subchefe do DI possam chamar a função. É usada, por exemplo, na função de designar um relator para uma solicitação de afastamento;

- **AuthenticatedSecretarioAction**: permite que apenas secretários autenticados no sistema possam chamar a função. É usada em funções como: adicionar um novo usuário no sistema, adicionar o parecer do CT ou da PRPPG relativo a uma solicitação, adicionar um mandato de um professor no sistema, etc.

A estrutura dessas ações customizadas segue o padrão demonstrado na Listagem 4.2.

Listagem 4.2 – Modelo de criação de uma *Authenticated Action* para o SCAP

```
1  import javax.inject.Inject
2  import play.api.mvc.Results._
3  import play.api.mvc._
4
5  class Authenticated____Action @Inject() (parser: BodyParsers.Default)(implicit
       ec: ExecutionContext)
6  extends ActionBuilderImpl(parser) {
7
8      override def invokeBlock[A] (request: Request[A], block: (Request[A]) =>
9      Future[Result]) = {
10         /* Verifica se o usuário corresponde ao desejado. Caso positivo, a
               função é executada normalmente. */
11
12             val res: Future[Result] = block(request)
13             res
14
15         /* Caso negativo, mostra uma tela de erro */
16
17             Future.successful(Ok(br.ufes.scap.views.html.erro()))
18     }
19 }
```

Um elemento que pode facilmente trazer confusão é a nomenclatura das funções **invokeBlock** e **block**. Ao contrário do que parece a primeira vista, a função **block** não bloqueia a resposta da requisição, pelo contrário, ela libera a função para seguir o fluxo normal de eventos. Nesse caso, não deve-se associar a palavra **block** com bloqueio, mas sim com o bloco de código a ser executado. Como tais funções pertencem à biblioteca do *Play*, o autor não é responsável por essa nomenclatura.

Tendo construído essas novas ações, elas estão disponíveis para uso, bastando injetá-las nos controladores, conforme mostra a Listagem 4.3.

Listagem 4.3 – Injectando e usando Ações customizadas em um Controlador no SCAP

```
1  class SolicitacoesController @Inject() (authenticatedUsuarioAction:
       AuthenticatedUsuarioAction, authenticatedSecretarioAction :
```

```
        AuthenticatedSecretarioAction, authenticatedProfessorAction :
        AuthenticatedProfessorAction, authenticatedChefeAction :
        AuthenticatedChefeAction) extends Controller {
2
3       def funcaoExemplo = authenticated____Action {
```

Ainda existem algumas condições de autenticação que não são cobertas pelos métodos descritos anteriormente; certas funções que só podem ser chamadas pelo autor da solicitação de afastamento ou pelo relator da solicitacação. Para esses casos específicos, optou-se por usar a estrutura simples de *if-then-else* para fazer as verificações necessárias.

Quando um usuário tenta acessar uma página a qual ele não pode ter acesso, ele é automaticamente redirecionado a uma tela de erro, exibida na Figura 17.

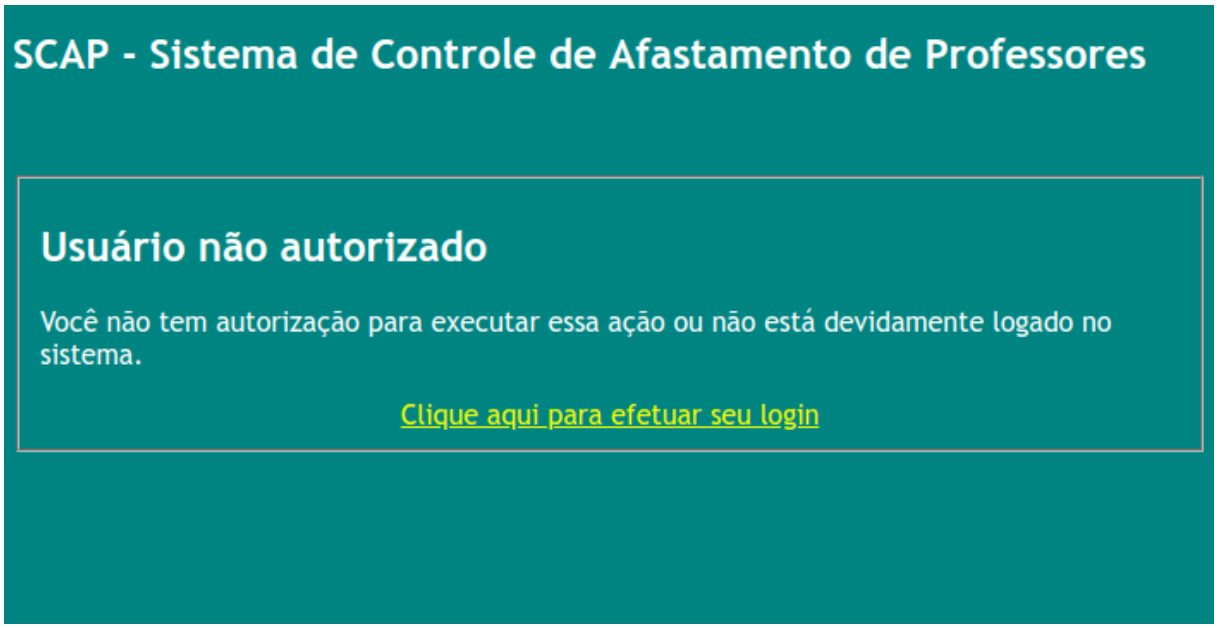

Figura 17 – Tela de acesso não autorizado no SCAP.

4.3.1.2 Submissão de Formulários, Classes *Case* e *Forms*

Scala suporta o conceito de classes *case*. Classes *case* são classes regulares que são imutáveis por padrão; decompostas por meio de correspondência de padrões; comparadas por igualdade estrutural ao invés de referência; e sucintas para instanciar e operar (SCALA, 2019).

A maneira que o *Play* lida com formulários é baseada no conceito de ligação (*binding*) de dados. Quando os dados vêm de uma requisição POST, o *Play* vai procurar por dados formatados e ligá-los a um objeto *Form*. Então, o Play pode usar esses valores para preencher uma classe *case*, chamar uma validação customizada, entre outras opções.

Para entender melhor como esse processo funciona, será ilustrado aqui em dois exemplos: criação e submissão de solicitações de afastamento e busca por afastamento. Na Figura 18 é mostrada a tela do formulário de cadastro de solicitação de afastamento.

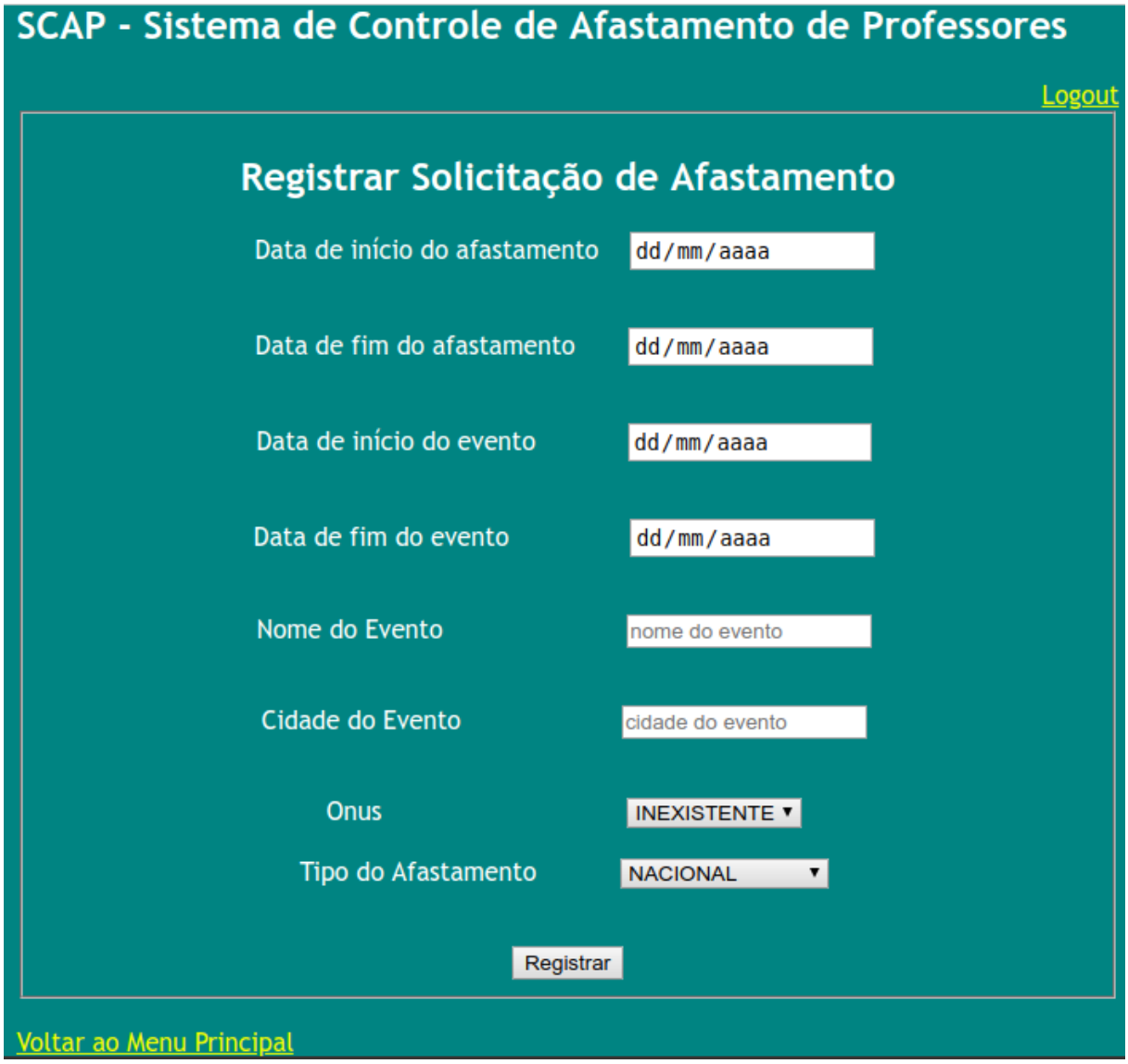

Figura 18 – Formulário de cadastro de solicitação de afastamento no SCAP.

Os dados obtidos através do preenchimento desse formulário pelo usuário na página Web devem ser ligados a um objeto correspondente no controlador, cujo modelo é descrito na Listagem 4.4.

Listagem 4.4 – Modelo do objeto correspondente ao formulário da solicitação de afastamento no SCAP.

```
1 import play.api.data.Form
2 import play.api.data.Forms._
3 /* ... */
4
5 case class SolicitacaoFormData(dataIniAfast : Date, dataFimAfast :
6     Date, dataIniEvento: Date, dataFimEvento : Date,
7     nomeEvento : String, cidade : String, onus : String,
8     tipoAfastamento : String
9     )
10
```

```
11 object SolicitacaoForm {
12
13   val form = Form(
14     mapping(
15       "dataIniAfast" -> date,
16       "dataIniAfast" -> date,
17       "dataFimAfast" -> date,
18       "dataIniEvento" -> date,
19       "dataFimEvento" -> date,
20       "nomeEvento" -> nonEmptyText,
21       "cidade" -> nonEmptyText,
22       "onus" -> nonEmptyText,
23       "tipoAfastamento" -> nonEmptyText
24     )(SolicitacaoFormData.apply)(SolicitacaoFormData.unapply)
25         .verifying("Erro: Data de inicio do afastamento posterior a data de fim
               do afastamento",
26             s => SharedServices.checaData(s.dataIniAfast, s.dataFimAfast))
27             /* ... */
28   )}
```

A chamada de verificação (*veryfing*) serve para garantir que os dados inseridos possam ser validados, respeitando as restrições de integridade do sistema.

Deve-se enfatizar que *SolicitacaoForm* e *SolicitacaoFormData* são usados apenas para receber os dados de uma requisição POST, conforme mostra a Listagem 4.5 e não correspondem a um objeto solicitação conforme mostrado no modelo de entidades nem a um objeto solicitação mostrado no modelo de persistência.

Listagem 4.5 – Ligando os dados provenientes da interação com o usuário ao objeto form no SCAP — criação de solicitação.

```
1 def addSolicitacao() = authenticatedProfessorAction.async { implicit request =>
2     SolicitacaoForm.form.bindFromRequest.fold(
3         errorForm =>
4             // Se houve erro, isso é, se o usuário inseriu dados que violam as
                  restrições de integridade do sistema, há um redicionamento de
                   volta para a página de cadastrar solicitação.
5         solicitacaoForm => {
6             // Se não houve erro, Aqui nesse bloco de código, a partir da
                  solicitacaoForm constrói-se o que realmente é uma solicitação de
                  afastamento no contexto do SCAP, e segue-se o fluxo normal.
7
8       })}
```

Um exemplo mais claro disso pode ser visto na função de busca. A tela de busca é apresenta na Figura 28 (apresentada mais adiante, na Seção 4.4.4). A Listagem 4.6 mostra o modelo do objeto correspondente ao formulário de busca de afastamento no SCAP, e a Listagem 4.7 mostra a ligação dos dados recebidos do formulário ao objeto correspondente no controlador.

Listagem 4.6 – Modelo do objeto correspondente ao formulário de busca de afastamento no SCAP.

```
1
2 case class BuscaFormData(idProfessor : Long, idRelator : Long, status : String)
3
4 object BuscaForm {
5   val form = Form(
6   mapping(
7       "idProfessor" -> longNumber,
8       "idRelator" -> longNumber,
9       "status" -> text
10      )(BuscaFormData.apply)(BuscaFormData.unapply)
11      )
12 }
```

Listagem 4.7 – Ligando os dados provenientes da interação com o usuário ao objeto form no SCAP — busca por solicitação.

```
1   def definirBusca = authenticatedUsuarioAction.async { implicit request =>
2         BuscaForm.form.bindFromRequest.fold(
3         // if any error in submitted data
4         errorForm =>
5           Future.successful
6           (BadRequest(br.ufes.scap.views.html.buscarSolicitacoes(errorForm,
                UserService.listAllUsersByTipo(TipoUsuario.Prof)))),
7         buscaForm => {
8           val solicitacoes = SolicitacaoService.busca(buscaForm.idProfessor,
                buscaForm.idRelator, buscaForm.status)
9           Future.successful(Ok(br.ufes.scap.views.html.listSolicitacoes(
                solicitacoes)))
10        }
11        )
12  }
```

Note que não existem classes de busca no modelo de entidades do sistema nem no modelo de persistência. A *form BuscaForm* e a classe *case BuscaFormData* existem apenas para atender a requisições de busca.

4.3.2 Camada de Lógica de Negócio

Nesta subseção, será explicado o funcionamento da camada de lógica de negócio, destacando como as particularidades do *framework Play* e da linguagem Scala influenciaram a sua implementação.

As classes de domínio, apresentadas no modelo de entidades na Seção 4.2.1, representam os elementos do mundo real no escopo do problema tratado pelo SCAP. Os objetos destas classes são manipulados pelos serviços da aplicação e armazenados no banco de dados pelos DAOs da persistência.

O pacote de serviço serve como uma ponte entre os controladores e a camada de acesso ao banco de dados. Também faz a conversão entre os elementos 'crus' que vêm do banco para os elementos elaborados que são usados nas camadas superiores, conforme

citado anteriormente.

Muitas das funções de verificação necessárias para garantir que as restrições de integridade do sistema sejam mantidas foram implementadas na camada de serviço, assim como muitas verificações acerca da natureza do usuário atual no sistema (se é autor ou relator de uma solicitação de afastamento em questão, por exemplo).

Outra importante tarefa feita na camada de serviços é o envio de e-mails. Para isso, utilizou-se o JavaMail e o Apache Commons Email. JavaMail é uma API Java usada para enviar e receber e-mails via SMTP (que é o protocolo utilizado neste projeto), POP3 e IMAP.

A Apache Commons é um projeto da Apache Software Foundation, cujo principal é fornecer software Java de código aberto e reutilizável. O objetivo da Apache Commons Email é prover uma API para enviar e-mails. Ela foi projetado tendo por base a API JavaMail, e simplifica bastante o seu uso.

Embora essas API sejam projetadas para Java, graças a interoperabilidade entre Scala e Java, foi plenamente possível usar tais APIs no projeto para implementar as funcionalidades que envolvem o envio de emails. Observa-se, no entanto, que foi implementado o envio de e-mails apenas via Gmail, dado que o projeto se trata de um protótipo.

4.3.3 Camada de Lógica de Acesso a Dados

Nessa subseção, será explicado o funcionamento da camada de lógica de acesso a dados, destacando como as particularidades do *framework Play* e da linguagem Scala influenciaram a sua implementação.

Inicialmente, há duas definições importantes próprias da linguagem Scala que precisam ser explicadas:

- *Trait*: equivalente a interfaces de Java. São utilizadas para definir tipos de objetos apenas especificando as assinaturas dos métodos suportados (SCALA, 2019);
- *Object*: métodos e valores que não são associados com instâncias individuais de uma classe são considerados objetos *singleton*, indicados pela palavra-chave *object* ao invés de *class*. Pode-se dizer que um *object* se trata de uma classe de instância única, diferente de uma classe comum que pode ter várias instâncias. *Objects* podem herdar de classes e *traits* (SCALA, 2019).

Outro detalhe relevante da linguagem Scala a ser destacado é o tipo *Future*. Esse tipo é usado para executar operações não-bloqueantes assíncronas em paralelo. Um *Future* representa um valor que pode ou não estar disponível no momento atual, mas estará disponível em algum momento (ou uma exceção caso esse valor não fique disponível).

Seguindo o modelo do FrameWeb (SOUZA, 2007), mas adaptando-o ao *framework* Play e à linguagem Scala, inicialmente criou-se um *trait* chamado **BaseDAO** que define os métodos que serão usados para todas as classes de persistência: **save**, **delete**, **update**, **get** e **listAll**.

Então foram criados *traits* para todas as demais classes, denotados por **NomeDaClasseDAO**. Todos herdam de **BaseDAO** e também especificam quais serão os métodos de persistência específicos para cada classe.

Finalmente foram criados *objects* que herdam dos *traits*, chamados **NomeDaClasseDAOSlick**. Esses objects implementam todos os métodos definidos nos *traits*, e utilizam as bibliotecas *Slick* para realizar o acesso ao Banco de Dados.

A Listagem 4.8 mostra um exemplo de implementação de uma classe da camada de persistência no SCAP.

Listagem 4.8 – Consulta ao banco de dados no SCAP

```
1  import scala.concurrent.ExecutionContext.Implicits.global
2  import slick.driver.JdbcProfile
3  import slick.driver.MySQLDriver.api._
4  import play.api.db.slick.DatabaseConfigProvider
5  import scala.concurrent._
6  import scala.concurrent.duration._
7  /* ... */
8
9  object UserDAOSlick extends UserDAO {
10
11     val dbConfig = DatabaseConfigProvider.get[JdbcProfile](Play.current)
12
13     val users = TableQuery[UserTableDef]  //representa a tabela de usuários no
           banco, sendo essencial para a sintaxe.
14
15     def get(id: Long): Option[User] = {
16         val Usuario : Future[Option[User]] = dbConfig.db.run(users.filter(_.id ==
               = id).result.headOption) // isso equivale a query SELECT TOP 1 * FROM
               USERS WHERE ID = {id}
17
18         return Await.result(Usuario, Duration.Inf)
19
20         /* ... Outras funções ... */
21   }
```

Os *imports* são necessários para acessar os recursos da biblioteca *Slick* e do tipo *Future*. Os métodos do *Slick* retornam valores *Future*, e para isso deve-se usar a função **Await.result(-Future[T]-, -Duração de espera-)** para obter o resultado real, que será retornado para as outras camadas.

4.4 Apresentação do Sistema

Nesta seção, são apresentadas as telas do sistema e um pouco de como funciona a navegação pelo sistema.

4.4.1 Login e erro de autenticação

Inicialmente, deve-se fazer o login, cuja tela já foi apresentada na Figura 16. Caso o usuário tente acessar alguma página via URL sem estar autenticado, vai receber uma mensagem de erro, também já mostrado anteriormente na Figura 17.

4.4.2 Menu principal

Após um login bem sucedido, o usuário é redirecionado para a tela do menu principal. Dependendo do tipo do usuário — professor ou secretário — o menu mostrará opções diferentes, conforme pode ser visto nas figuras 19 e 20.

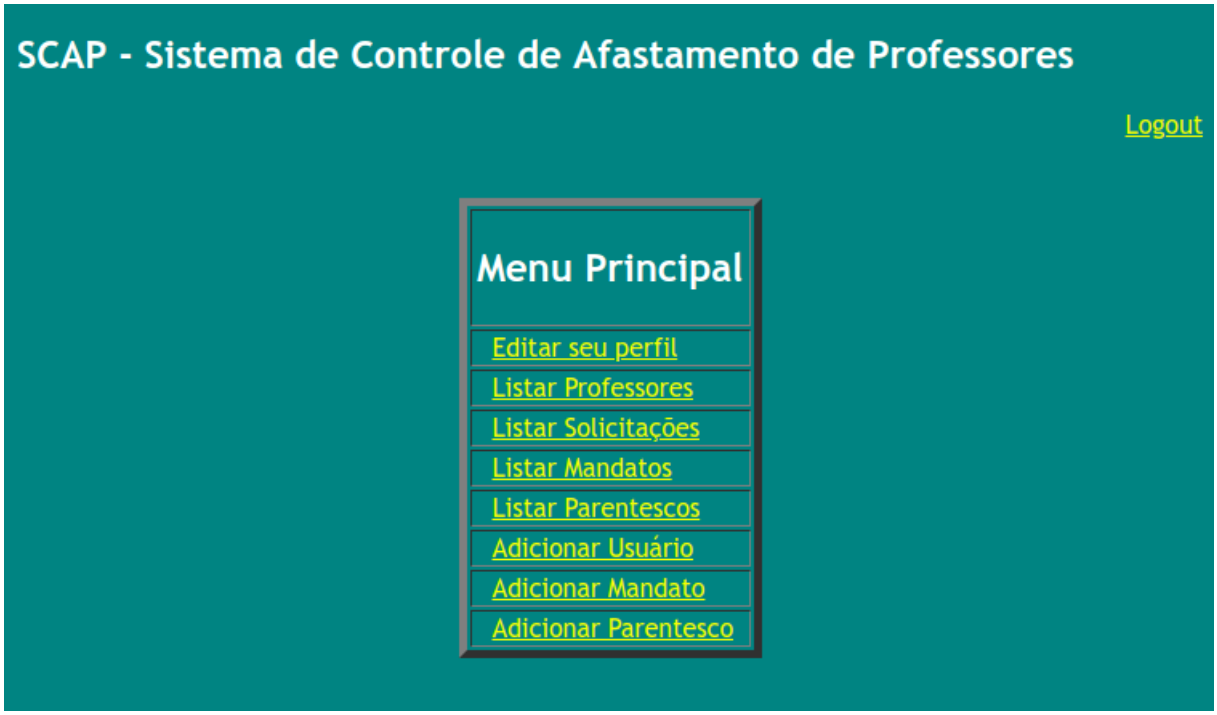

Figura 19 – Menu visto por um Secretário do SCAP.

As diferentes telas refletem algumas das restrições de integridade do sistema: apenas um professor pode adicionar uma solicitação de afastamento, enquanto apenas um secretário pode adicionar novos usuários, parentescos ou mandatos de chefia.

4.4.3 Usuários, Parentescos e Mandatos

Para registrar um novo usuário no sistema (opção “Adicionar Usuário”), o secretário preenche um formulário informando nome, matricula, e-mail, senha e tipo — Professor(a)

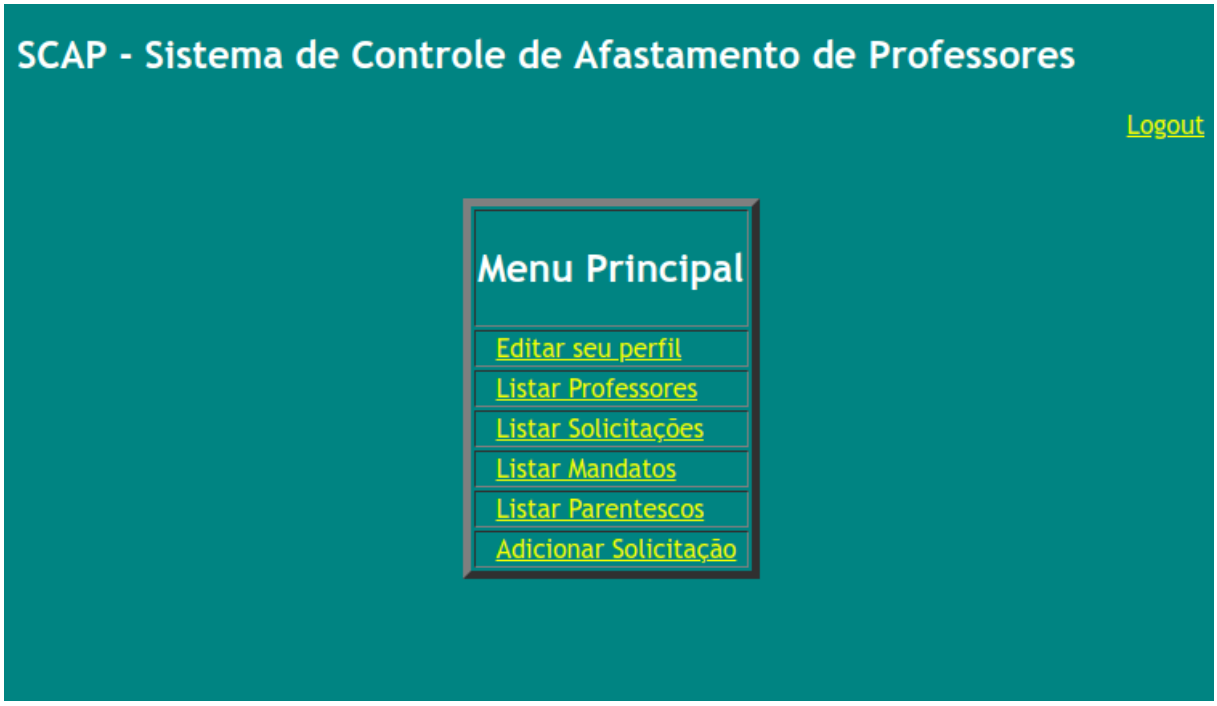

Figura 20 – Menu visto por um Professor do SCAP.

ou Secretário(a) —, conforme mostra a Figura 22. Posteriormente, qualquer usuário pode modificar seu próprio perfil (opção "Editar seu perfil") e alterar nome, e-mail e senha, conforme mostra a Figura 21, porém matrícula e tipo são inalteráveis após o cadastro.

Quando a lista de professores é exibida, como mostra a Figura 23, é possível visualizar as informações públicas de cada usuário (ID, nome, matrícula, e-mail) e também é possível ver mandatos, parentescos e solicitações de afastamentos específicas do(a) professor(a) desejado(a).

Para adicionar um parentesco (opção "Adicionar Parentesco"), o secretário seleciona dois professores da lista de professores do sistema, conforme mostra a Figura 24. Parentescos podem ser sanguíneos ou matrimoniais, porém tal diferenciação é irrelevante no contexto do sistema, por isso foi omitida. Qualquer usuário pode visualizar a lista de parentescos no sistema (opção "Listar Parentescos"), mas para um secretário existe a opção de excluir o parentesco, conforme mostra a Figura 25, enquanto um professor pode apenas visualizá-o, conforme mostra a Figura 26.

A situação é análoga para mandatos: um secretário os cadastra preenchendo um fórmulário (informando professor, cargo, data de início e data de fim); qualquer usuário pode visualizar o histórico de mandatos cadastrados no sistema, mas apenas secretários têm a opção de remover um mandato do sistema.

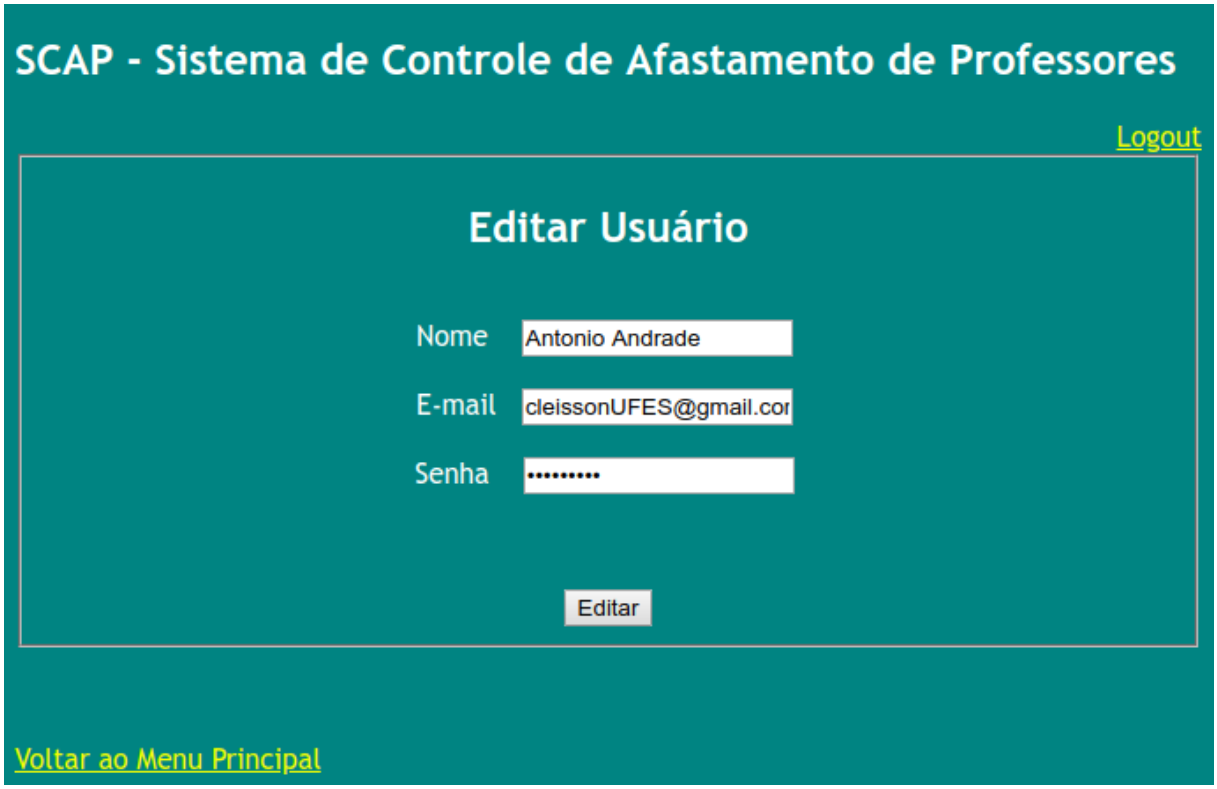

Figura 21 – Editando seu perfil no SCAP.

4.4.4 Solicitações de Afastamento e Pareceres

Professores podem adicionar solicitações de afastamento preenchendo o formulário (já apresentado anteriormente na Figura 18). Qualquer usuário pode ver as solicitações de afastamento listadas no sistemas, conforme mostra a Figura 27.

É possível procurar uma solicitação específica por relator, professor e/ou status através da opção "Fazer Busca". A tela de busca é exibida na Figura 28. Pode-se abrir mão de usar qualquer um dos campos de busca bastando deixar selecionada a opção "TODOS".

Também é possível ver mais informações sobre uma solicitação através da opção "ver mais detalhes". Essa opção exibirá a tela, como nos exemplos mostrados nas figuras 29 e 30.

É possível notar que uma figura mostra as opções "Designar Relator" e "Ver Pareceres", enquanto a outra mostra as opções "Registrar parecer externo" e "Ver Pareceres". Isso acontece porque as opções exibidas abaixo dos dados da solicitação dependem do usuário autenticado no sistema e do status da solicitação, de acordo com as seguintes regras:

- A opção "Ver Pareceres" aparece em todas as situações;
- A opção "Cancelar Solicitação" aparece se o usuário for o autor da solicitação;

Figura 22 – Secretário registrando um novo usuário no SCAP.

Figura 23 – Lista de professores cadastrados no SCAP.

- A opção “Designar Relator” aparece se o usuário está exercendo um mandato de chefe ou vice-chefe e a solicitação é internacional e ainda não há relator designado para ela;
- A opção “Registrar Parecer” aparece se o usuário é relator da solicitação e ainda não emitiu um parecer sobre ela;
- A opção “Manifestar-se contra” aparece se a solicitação é nacional e o usuário é

Figura 24 – Secretário registrando um novo parentesco no SCAP.

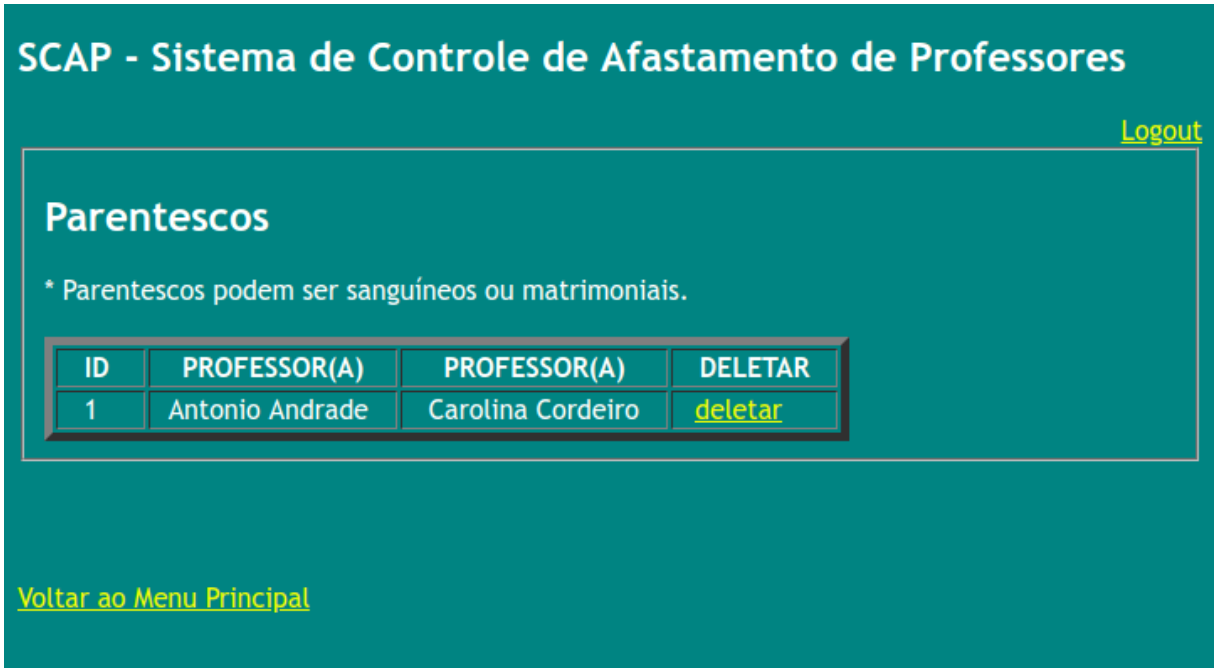

Figura 25 – Secretário visualizando lista de parentescos no SCAP.

professor;

- A opção "Registrar Parecer Externo" aparece se o usuário é secretário e a solicitação já foi aprovada pelo relator do DI;
- A opção "Aprovar" ou "Reprovar" aparece se o usuário é secretário e houve uma manifestação contra uma solicitação para evento nacional. Nesse caso, é marcada uma reunião do DI, que aprova ou reprova a solicitação, cabendo ao secretário registrar

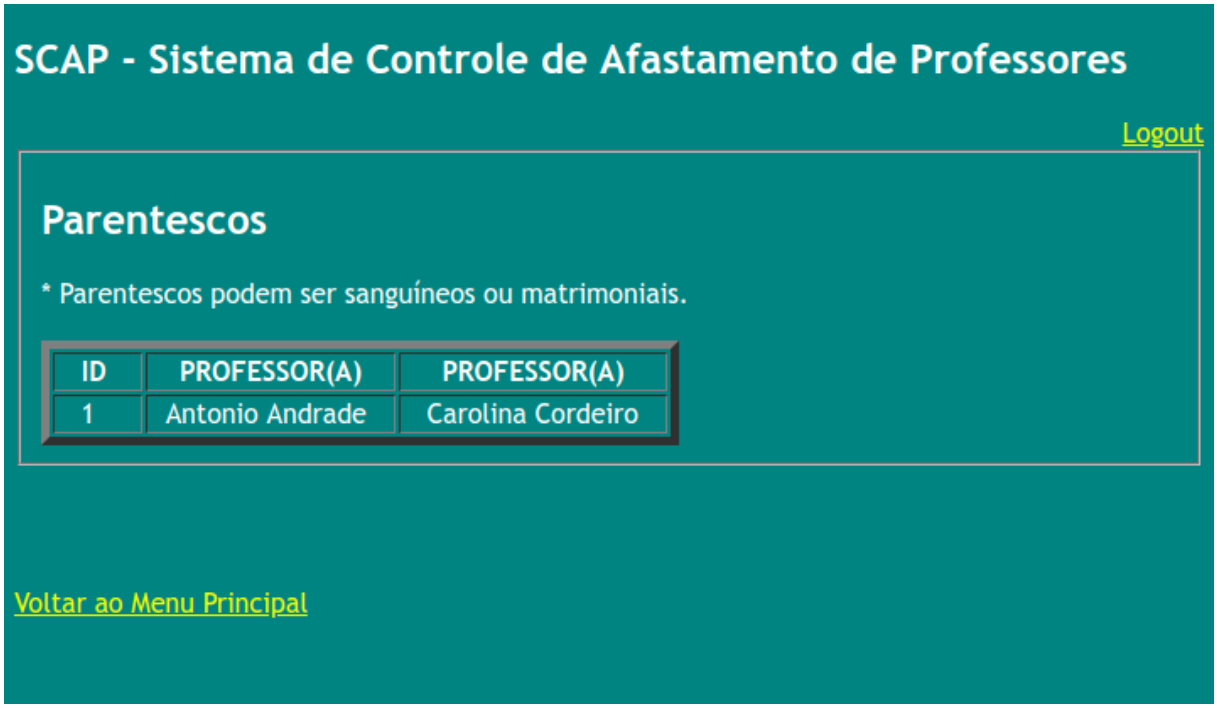

Figura 26 – Professor visualizando lista de parentescos no SCAP.

Figura 27 – Professor visualizando lista de solicitações de afastamento no SCAP.

esse resultado;

- A opção "Arquivar" aparece se o usuário é secretário e a solicitação já está terminada, ou porque é nacional e já foi aprovada pelo DI, ou porque é internacional e já foi aprovada pelo PRPPG, ou se a solicitação foi reprovada.

Quando um professor é designado como relator de uma solicitação de afastamento para um evento internacional ele deve registrar um parecer a favor ou contra o afastamento, conforme mostra a Figura 31.

Se uma solicitação de afastamento para um evento internacional for aprovada pelo relator do DI, ela será avaliada também pelo CT e depois pela PRPPG, e esses pareceres devem ser registrados por um secretário, conforme mostra a Figura 32.

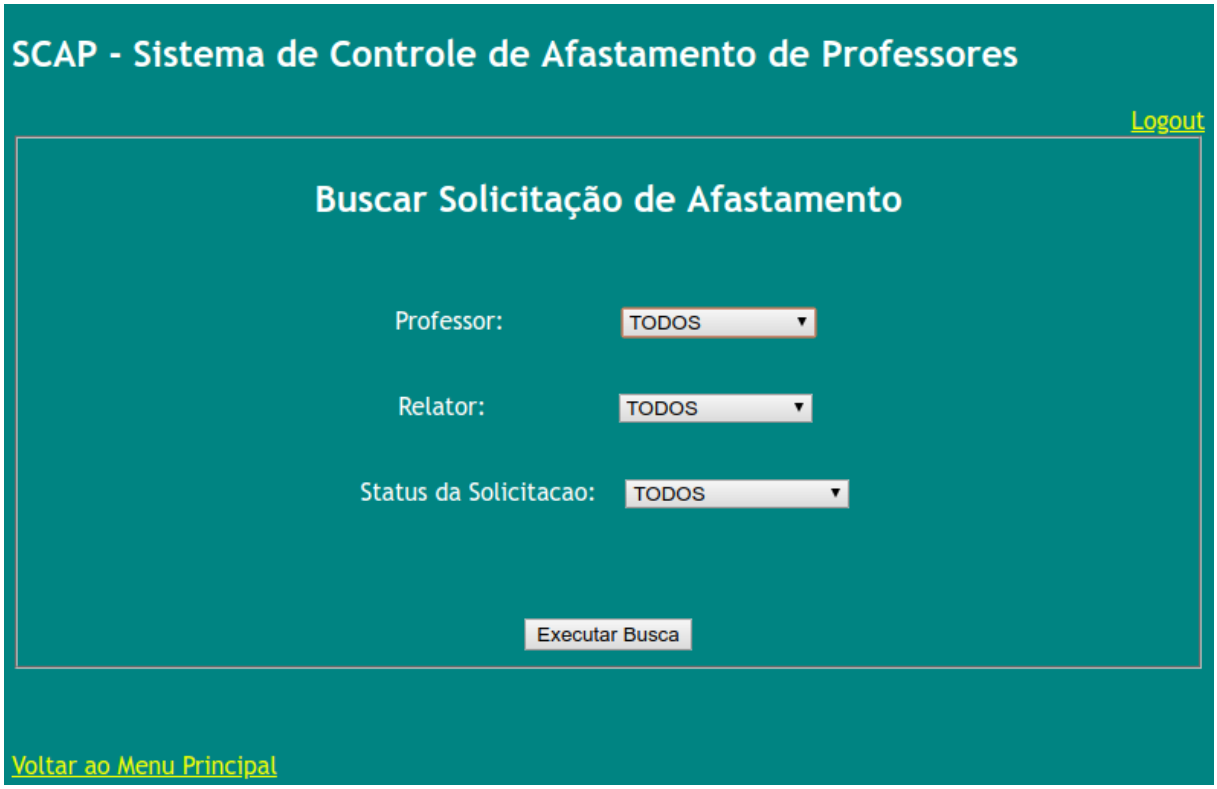

Figura 28 – Tela de busca de solicitações de afastamento no SCAP.

Qualquer usuário pode consultar a lista de pareceres sobre uma determinada solicitação, como foi mencionado anteriormente basta ir para a página que exibe os dados da solicitação desejada e clicar em "Ver Pareceres". A tela exibida é exemplificada na Figura 33.

O usuário pode visualizar os dados de cada parecer com mais detalhes, conforme mostra a Figura 34. Para pareceres externos ao DI (isto é, pareceres do CT e da PRPPG) é possível também fazer download do documento associado a eles.

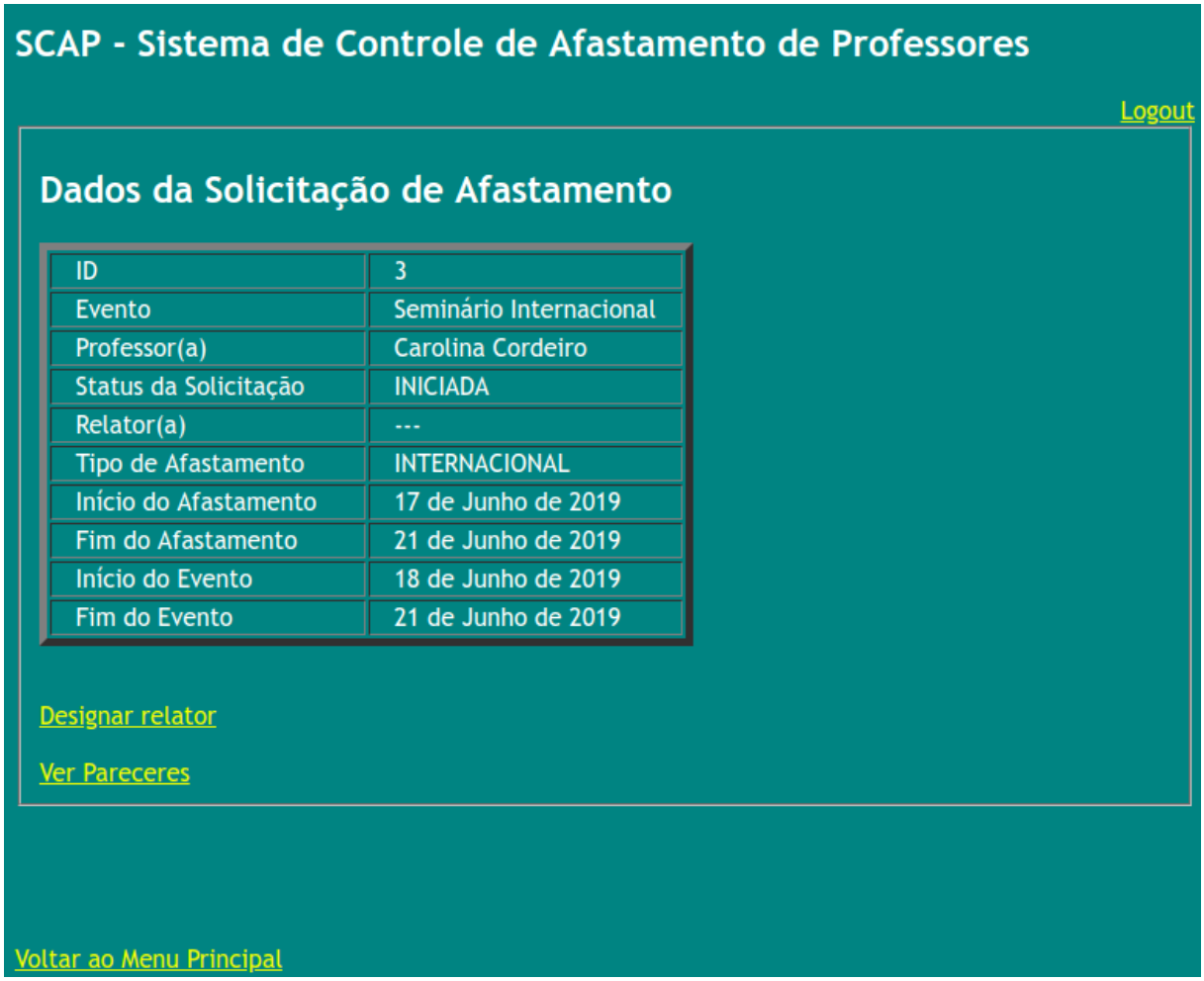

Figura 29 – Tela de visualização em detalhes de uma solicitação de afastamento no SCAP.

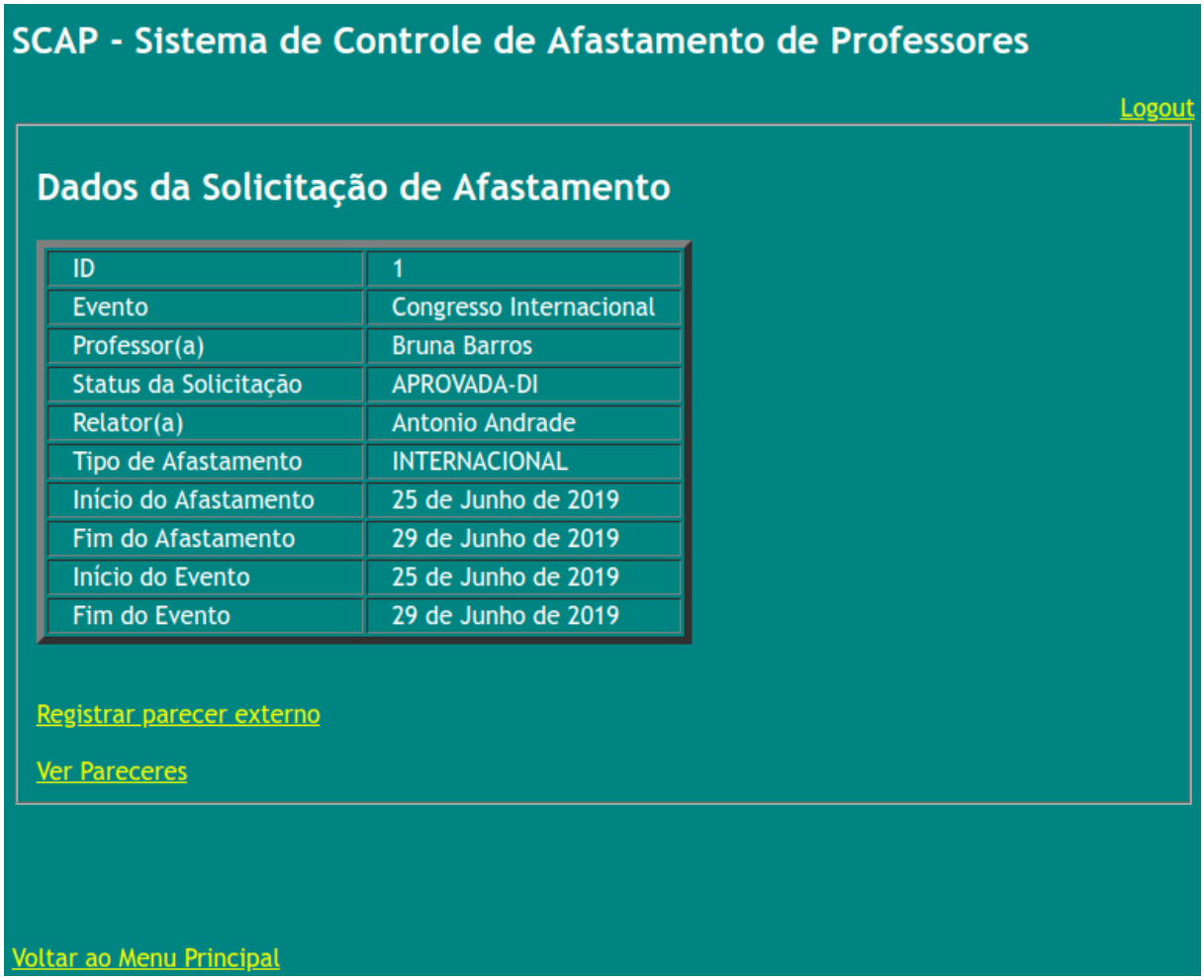

Figura 30 – Tela de visualização em detalhes de uma solicitação de afastamento no SCAP.

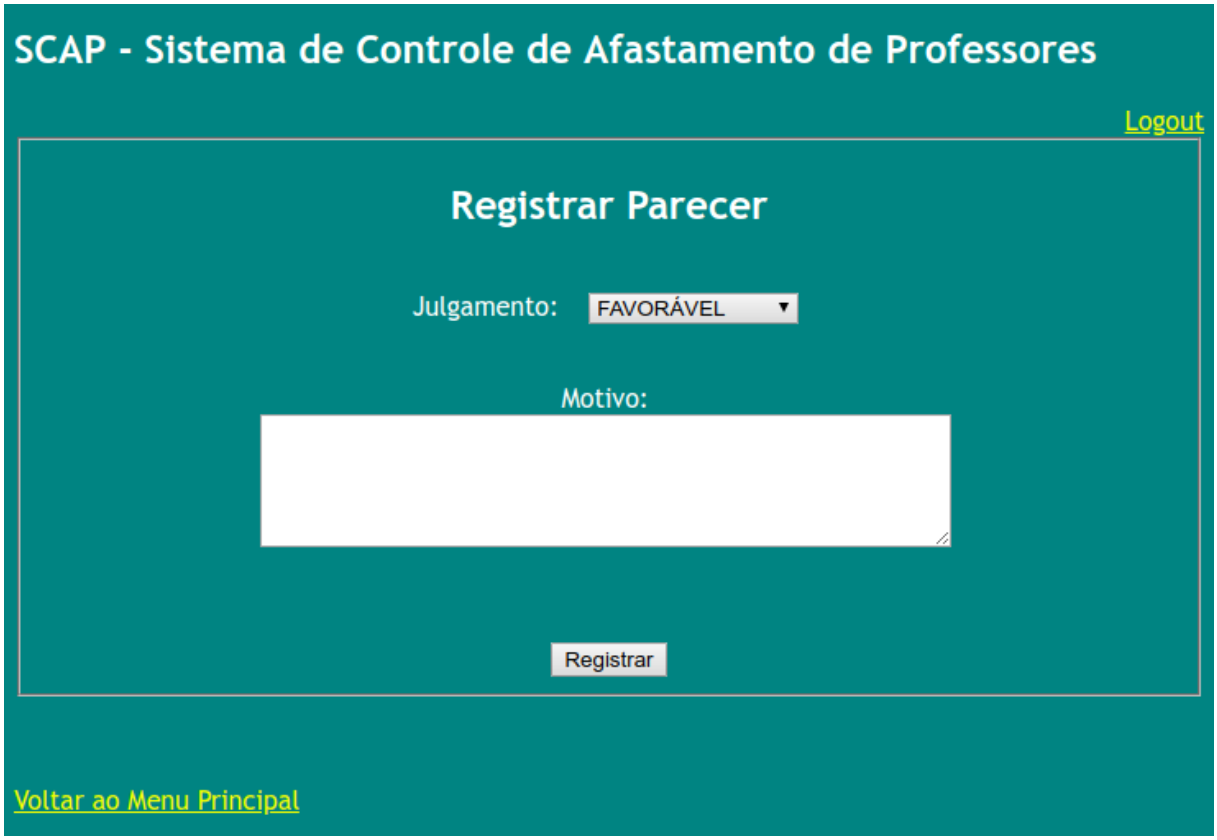

Figura 31 – Tela onde o relator registra seu parecer.

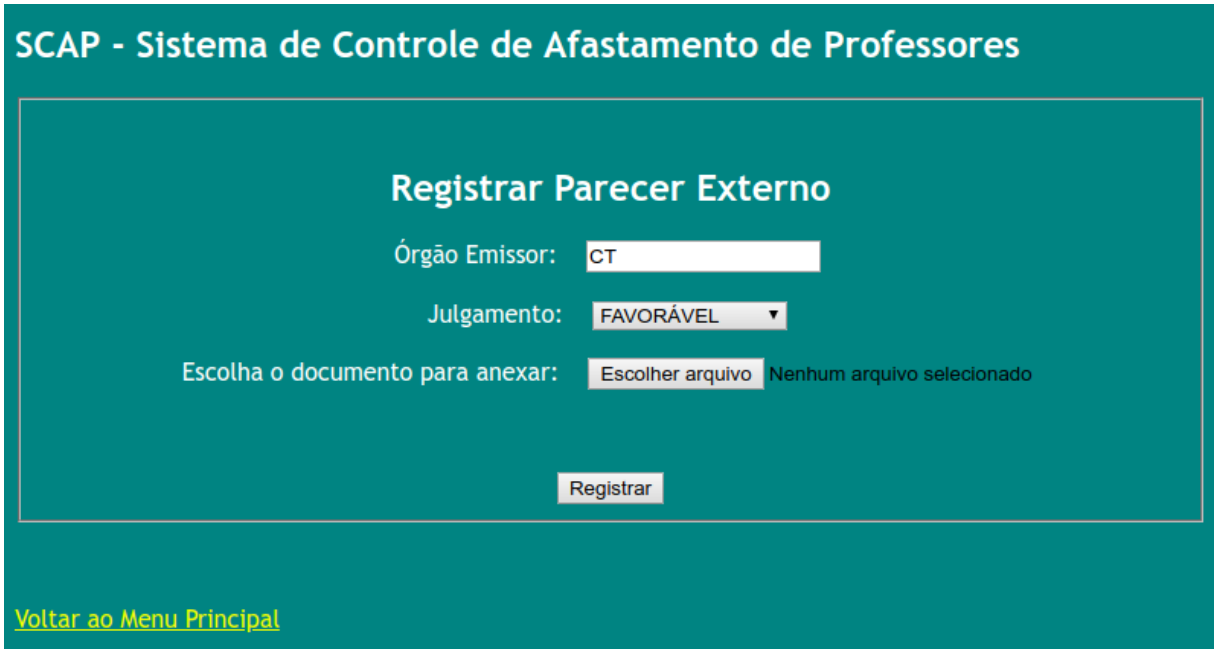

Figura 32 – Tela onde o secretário registra um parecer externo.

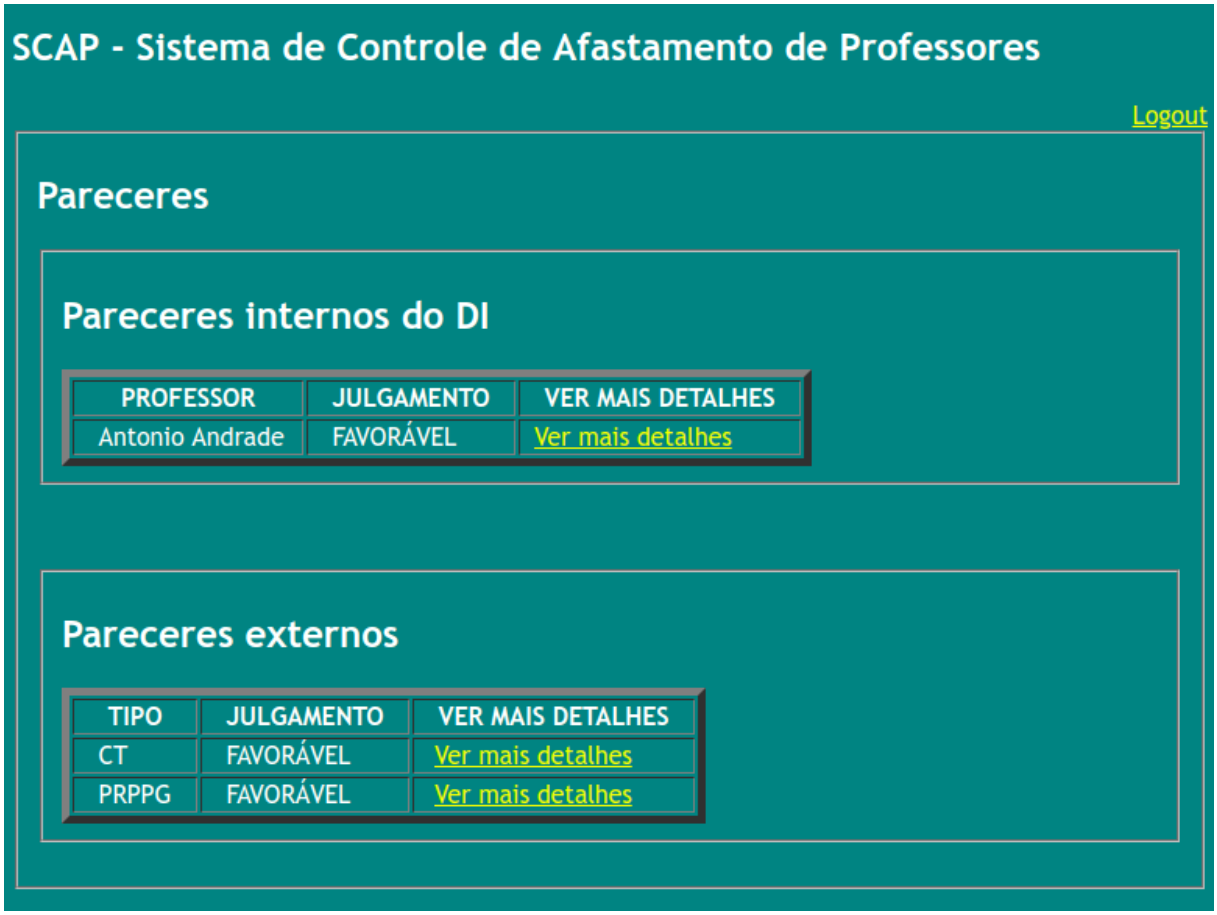

Figura 33 – Lista de pareceres para uma determinada solicitação de afastamento.

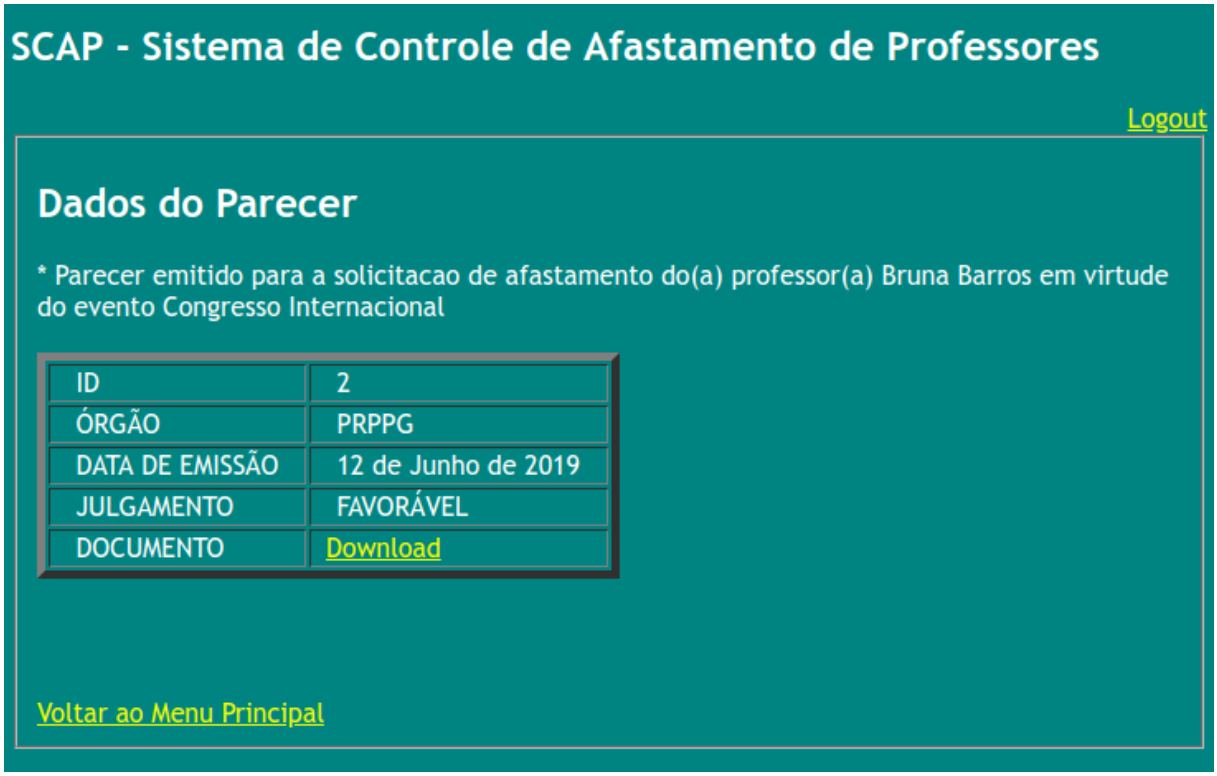

Figura 34 – Tela de exibição de dados de um determinado parecer.

5 Conclusões

Este capítulo apresenta as conclusões obtidas a partir do trabalho de implementação do SCAP em um novo *framework* e em uma nova linguagem; seus pontos positivos e negativos, dificuldades encontradas e possíveis melhorias.

Em relação aos objetivos do trabalho expostos anteriormente na Seção 1.2, pode-se destacar:

- O objetivo geral — aplicar o método FrameWeb em uma nova implementação do SCAP — foi alcançado com sucesso. Vale ressaltar que essa versão do SCAP foi implementada do zero, sem reaproveitamento de nenhum código de implementações anteriores, pois não havia ainda versão anterior feita na linguagem Scala, o que adicionou um grau de dificuldade maior ao projeto, mas ao mesmo tempo trouxe uma profunda experiência de aprendizado;
- O conhecimento adquirido nas diversas disciplinas ao longo do curso — entre elas, Programação III, Engenharia de Software, Banco de Dados, Desenvolvimento Web e Web Semântica — foi imprescindível. Certamente, sem essa base, não se teria chegado ao resultado final.
- Também foi feita uma análise de como as particularidades do *framework Play* e da linguagem Scala se encaixam no método FrameWeb no contexto do SCAP.

O uso do *framework Play*, e da biblioteca *Slick*, foram de grande valia na implementação do SCAP, pois além de fornecerem ferramentas que em muito simplificaram o processo, os componentes presentes nesses *frameworks* já foram extensivamente testados e foi possível encontrar vários exemplos, em sua documentação oficial e na Internet. Porém, em alguns momentos o autor considerou certas abordagens da linguagem e do *framework* contra-intuitivas e um pouco complexas, mas nada que travasse a implementação do sistema.

É fundamental destacar a versatilidade do método *FrameWeb*, pois, apesar de necessárias algumas adaptações, seguir os padrões já previamente estabelecidos por (SOUZA, 2007) foi muito mais do que uma mera obrigação e sim um facilitador na implementação do sistema, sendo um mapa para o projeto.

Como trabalho futuro sugerido, poderia ser implementada uma versão mais aprofundada do SCAP nessa linguagem, adicionando certas funcionalidades que ficaram de fora desse protótipo por questões de tempo e logística, tais como envio de e-mails automá-

ticos para outras plataformas além do Gmail, e criação de métodos de persistência para elementos genéricos.

Outro desenvolvimento futuro seria incorporar as ferramentas mais recentes do *FrameWeb*, o Editor e o Gerador de Código, e comparar com a implementação atual. Tais ferramentas não foram utilizadas pois o *framework Play* e a linguagem Scala não constavam nas biblioteca do FrameWeb Editor e o autor desse trabalho não tinha conhecimento suficiente da ferramenta para adicioná-los. Além disso, provou-se ser bastante proveitoso para maximizar o aprendizado da nova linguagem não utilizar nenhum gerador de código automático.

Referências

ALMEIDA, N. V. *A Model-Driven Approach for Code Generation forWeb-based Information Systems Built with Frameworks*. Dissertação (Artigo apresentado no 23º Simpósio Brasileiro de Sistemas Multimedia e Web) — Universidade Federal do Espírito Santo, Gramado, RS, Brasil, 2017. Citado na página 19.

AVELAR, R. d. A. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Ninja*. Monografia (Projeto de Graduação) — Departamento de Informática, Universidade Federal do Espírito Santo, Vitória (ES), Brasil, 2017. Citado na página 11.

CAMPOS, S. L. *FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb*. Dissertação (Artigo apresentado no 23º Simpósio Brasileiro de Sistemas Multimedia e Web) — Universidade Federal do Espírito Santo, Gramado, RS, Brasil, 2017. Citado na página 19.

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 3 vezes nas páginas 11, 12 e 22.

FOWLER, M. *Patterns of Enterprise Application Architecture*. [S.l.: s.n.], 2002. (Addison-Wesley). ISBN 0321127420. Citado 2 vezes nas páginas 5 e 17.

MARTINS, B. F. *Evolução do Método FrameWeb para o Projeto de Sistemas de Informação Web Utilizando uma Abordagem Dirigida a Modelos*. Dissertação (Mestrado em Informática) — Universidade Federal do Espírito Santo, Vitória, ES, Brasil, 2016. Citado na página 18.

MATOS, R. P. d. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando os frameworks Spring MVC e Vaadin*. Monografia (Projeto de Graduação) — Departamento de Informática, Universidade Federal do Espírito Santo, Vitória (ES), Brasil, 2017. Citado na página 11.

MURUGESAN, S. et al. Web engineering: A new discipline for development of web-based systems. In: *Web Engineering*. [S.l.]: Springer, 2001. p. 3–13. Citado na página 14.

OLSINA, L.; LAFUENTE, G.; ROSSI, G. Specifying quality characteristics and attributes for websites. *Web Engineering*, Springer, p. 266–278, 2001. Citado na página 14.

PINHEIRO, F. G. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando um framework PHP e um framework JavaScript*. Monografia (Projeto de Graduação) — Departamento de Informática, Universidade Federal do Espírito Santo, Vitória (ES), Brasil, 2017. Citado na página 11.

PLAY-TEAM. Play 2.6.x documentation. In: . [s.n.], 2017. Disponível em: <https://www.playframework.com/documentation/2.6.x/Home>. Citado na página 21.

PRADO, R. C. d. *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 4 vezes nas páginas 11, 12, 22 e 28.

PRESSMAN, R. S. *Engenharia de Software - Uma abordagem profissional*. 7ª edição. ed. [S.l.]: McGraw-Hill, 2011. ISBN 9788563308337. Citado na página 14.

SCALA. Scala documentation. In: . [s.n.], 2019. Disponível em: <https://docs.scala-lang.org/>. Citado 3 vezes nas páginas 21, 41 e 45.

SOUZA, V. E. S. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Dissertação (Mestrado em Informática) — Universidade Federal do Espírito Santo, Vitória, ES, Brasil, 2007. Citado 9 vezes nas páginas 5, 11, 12, 16, 18, 20, 34, 46 e 58.