UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO

CENTRO TECNOLÓGICO

COLEGIADO DO CURSO DE CIÊNCIA DA COMPUTAÇÃO

Eduardo Gorayeb Dalapicola

Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Quasar e AdonisJS

Vitória, ES

2021

Eduardo Gorayeb Dalapicola

Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Quasar e AdonisJS

Monografia apresentada ao Curso de Ciência da Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Colegiado do Curso de Ciência da Computação

Orientador: Vítor E. Silva Souza

Vitória, ES

2021

Eduardo Gorayeb Dalapicola

Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Quasar e AdonisJS/ Eduardo Gorayeb Dalapicola. – Vitória, ES, 2021-

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

Orientador: Vítor E. Silva Souza

Monografia (PG) – Universidade Federal do Espírito Santo – UFES
Centro Tecnológico
Colegiado do Curso de Ciência da Computação, 2021.

1. Palavra-chave1. 2. Palavra-chave2. I. Souza, Vítor Estêvão Silva. II. Universidade Federal do Espírito Santo. IV. Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Quasar e AdonisJS

CDU 02:141:005.7

Eduardo Gorayeb Dalapicola

Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Quasar e AdonisJS

Monografia apresentada ao Curso de Ciência da Computação do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Bacharel em Ciência da Computação.

Trabalho aprovado. Vitória, ES, (dia) de (mês) de (ano):

Vítor E. Silva Souza
Orientador

Prof[a]. Dr[a]. Monalessa Perini Barcellos
Universidade Federal do Espírito Santo

Camila Zacche de Aguiar
Universidade Federal do Espírito Santo

Vitória, ES
2021

A todos os familiares que sempre me deram todo apoio.

Agradecimentos

Agradeço primeiramente a Deus por me conceder a inteligência para poder poder iniciar e concluir o curso de Ciências da Computação. Agradecer pela inspiração e perseverança que Ele me deu para a conclusão da monografia.

Agradeço também aos meus familiares que sempre me deram suporte no tempo que passei na Universidade, sempre acreditaram em meu potencial e acreditaram em mim. Agradeço à minha namorada e aos seus familiares que sempre me apoiaram e me acolheram como parte da família por não me deixarem desanimar nos momentos mais difíceis.

Agradeço aos colegas de curso que me ajudaram compartilhando experiências de suas próprias monografias e artigos publicados para que pudesse me sentir mais seguro ao dissertar a minha própria monografia

E por fim agradeço à Universidade Federal do Espírito Santo e todos os professores, principalmente ao professor orientador, Vítor Estêvão Silva Souza por contribuírem para a minha formação e possibilitaram que pudesse chegar ao fim com entusiasmo e dedicação.

"Nós só podemos ver um pouco do futuro,
mas o suficiente para perceber que há muito a fazer".
(Alan Turing)

Resumo

O histórico do desenvolvimento de Sistemas Web é comumente marcado pela falta de preocupação com padronização de métodos ou boas práticas. Dessa forma cada desenvolvedor emprega os seus próprios métodos no sistema. Com escopos pequenos e fechados, não se vê muita necessidade em ter um processo bem definido, porém com a crescente demanda do mercado para aplicações cada vez maiores, com número de usuários alto, expansão do escopo com incremento de funcionalidades de acordo com o amadurecimento da aplicação, entre outros desafios que surgem, rapidamente foi surgindo a necessidade de ter um processo bem definido de desenvolvimento. Assim surge a Engenharia Web (*Web Engeneering* ou WebE) que propõe vários métodos para análise, projeto e desenvolvimento de Sistemas de Informação.

Nesse cenário é que surge o método FrameWeb (*Framework-based Design Method for Web Engineering*), onde são feitas tentativas de organizar as diferentes partes do sistema para que seja de fácil compreensão para qualquer um que deseje integrar ao desenvolvimento de um projeto Web, tornando o processo de desenvolvimento mais rápido e eficiente. O FrameWeb recomenda categorias de *frameworks* onde cada *framework* pode ser utilizado para auxiliar no projeto de desenvolvimento.

O objetivo desse trabalho é verificar se os *frameworks* existentes se aplicam às categorias que o FrameWeb implementa. Para isso, nesse trabalho, assim como feito em trabalhos anteriores, foi implementado parte do sistema já existente SCAP, utilizando novos *frameworks* Javascript: AdonisJS (Persistência de Dados) e Quasar (Visualização). O SCAP é um sistema que permite gerenciamento no processo de afastamento de professores da UFES. Neste sistema professores podem pedir afastamento do cargo por um tempo para, por exemplo, participação em eventos.

Para a implementação da aplicação, adequações ao *framework* AdonisJS e Quasar foram feitas para darem suporte à estrutura predefinida do método *FrameWeb* que apresenta camadas bem definidas: Apresentação, Lógica de Negócio e Acesso a Dados. Comandos foram criados que auxiliam na construção de cada componente de camada, estes comandos criam classes que obedecem a dependência entre camadas por meio de injeção de dependências e agilizam a produção do *software*.

Os modelos *FrameWeb* foram construídos com base na implementação. A visualização apresentada pelos modelos foram importantes como uma ferramenta de consulta e validação para a garantia que o método está sendo implementado com coerência respeitando a comunicação entre as camadas e que cada camada desempenhe a sua função sem interferir

na outra. Desta forma, toda vez que era detectada uma incoerência em algum modelo *FrameWeb*, este era alterado assim como o código de programação equivalente.

Apesar de uma grande adequação no início ser necessária, a organização trouxe vantagens como: agilidade para a construção de módulos novos, manutenção dos módulos e realização de testes. Os *frameworks* deram suporte para a criação de um sistema com interface simples, intuitiva, que comunica-se com o usuário de forma efetiva atendendo aos requisitos funcionais.

Palavras-chaves: Engenharia Web. FrameWeb. Javascript. NodeJS. AdonisJS. Quasar.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

UML	Unified Modeling Language
NPM	Node Package Manager
MVC	Model View Controller
MVVM	Model, View, View-Model
PM	Presentation Model
HTML	Hypertext Markup Language
CSS	Cascading Style Sheet
SPA	Single Page Application
SSR	Server Side Rendering
PWA	Progressive Web Application
SGBDR	Sistemas Gerenciadores de Banco de Dados Relacional
SQL	Structured Query Language
ORM	Object–relational Mapping
WIS	Web Information Systems
DAO	Data Access Object
SCAP	Sistema de Controle de Afastamento de Professores
XML	Extensible Markup Language
AJAX	Asynchronous JavaScript and XML

Sumário

1 Introdução

O crescimento da Internet modifica a forma com que interagimos com o mundo e é responsável por um grande impacto nos negócios, comércio, indústria, setor de bancos e finanças, educação, etc. Empresas estão migrando as informações e sistemas de banco de dados cada vez mais para o ambiente de rede online. Essas aplicações por muitas vezes são complexas, com grande quantidades de informações. A ausência de processos bem definidos para o desenvolvimento foram gerando problemas para a implementação correta de funcionalidades, bem como implantação, operação e manutenção do *software* (MURUGESAN; DESHPANDE, 2001).

No passado, com aplicações *ad-hoc*, desenvolvidas sem metodologia ou processos de apoio, foi percebendo-se que cumprir requisitos não-funcionais é um trabalho árduo. Para apoiar no desenvolvimento foram aplicados os conceitos de Engenharia de Software nos sistemas Web, que futuramente seriam chamados de *WebApps*, ou aplicações na Web, que seriam sistemas altamente interativos, armazenando e provendo informações pela Internet (SOUZA, 2007). Posteriormente foram criadas novas ferramentas para apoiarem processos já existentes, criando assim uma nova sub-área da Engenharia de Software chamada Engenharia Web, que define novas abordagens para o desenvolvimento, manutenção e implantação de aplicações Web (PRESSMAN, 2011).

Nesse contexto, surge o FrameWeb *(Framework-based Design Method for Web Engineering)*, um método que utiliza uma abordagem com classes de *frameworks* que darão suporte ao desenvolvimento de aplicativos Web. Esse método é direcionado para WISs (*Web-based information Systems*), Sistemas deInformação Web que estão no ambiente da Internet, assim como aplicações de *e-commerce*, sistemas de gerenciamento de empresas, etc. (SOUZA, 2007).

O FrameWeb dispõe quatro tipos de modelo para a visualização da fase de projeto. Cada modelo é a representação gráfica de um conjunto de *frameworks* e como eles interagem entre si para a formação de uma camada do sistema. Dessa forma, facilita a comunicação dos projetistas com a equipe de desenvolvimento, agilizando assim o processo de desenvolvimento da aplicação.

O FrameWeb suporta algumas categorias de *frameworks*. Dentro de cada categoria são propostos alguns *frameworks* que podem ser utilizado no desenvolvimento. Na proposição do método foi utilizado um *framework* para cada categoria, dessa forma não foi possível comprovar se o método é adequado com a utilizações de outros *frameworks* para a mesma categoria.

Em 2014, no trabalho de conclusão de curso, Duarte (2014), utilizando o método

FrameWeb, desenvolveu o SCAP (Sistema de Controle de Afastamentos de Professores), sistema Web que possibilita o afastamento de professores pela Internet. Para a implementação foi utilizado a plataforma Java EE para desenvolvimento, juntamente um conjunto maior de *frameworks* do que os utilizados anteriormente na proposta original. Posteriormente outros trabalhos expandiram a utilização de novos *frameworks* e contribuíram para a análise do método. Por exemplo, Ferreira (2018) desenvolveu o SCAP utilizando os *frameworks* Tapestry e Wicket, Avelar (2018) com o *framework* fullstack Ninja, Meirelles (2019) com os *frameworks* Codeigniter e NodeJS, dentre outros.

1.1 Objetivo

O objetivo desse trabalho de conclusão de curso é aplicar o método FrameWeb para a implementação do SCAP (Sistema de Controle de Afastamento de Professores) a partir dos requisitos levantados por Duarte (2014) e posteriormente refinados por Prado (2015), utilizando novos *frameworks*, Quasar e AdonisJS, ambos utilizando a linguagem de programação Javascript. Dessa forma, ampliamos o número de *frameworks* para análise do método e contribuímos com a evolução do método, sugerindo melhorias.

Os objetivos específicos são:

- Projetar o sistema SCAP utilizando o método *FrameWeb*;
- Verificar a adequação do método *FrameWeb* ao conjunto de *frameworks* escolhidos;
- Dar sugestões de melhorias ou propostas de modificações para o método *FrameWeb*;
- Implementar o Sistema SCAP utilizando os *frameworks* AdonisJS e Quasar.

1.2 Método

Para o desenvolvimento do trabalho foram feitas as seguintes atividades

- Levantamento bibliográfico: estudo sobre Engenharia de Software, Engenharia de Requisitos, desenvolvimento de softwares, Engenharia Web, método FrameWeb, requisitos do SCAP e também dos *frameworks* que foram utilizados;
- Análise de Requisitos: análise dos requisitos de software do SCAP levantadas por Duarte (2014) e Prado (2015);
- Projeto: utilização do método FrameWeb para projetar o SCAP utilizando os *frameworks* propostos;
- Implementação: codificação do SCAP utilizando os *frameworks* propostos;

- Testes e ajustes: realização de testes das funcionalidades e ajustes de interface para melhorar a experiência do usuário com o Sistema;
- Finalização: elaboração da monografia e apresentação dos resultados.

1.3 Organização

Este trabalho foi dividido em cinco capítulos, incluindo esta introdução.

O Capítulo 2 apresenta o levantamento do arcabouço teórico do tema abordado. Neste capítulo são discutidas as técnicas e *frameworks* utilizados na implementação, o método *FrameWeb* e a Engenharia Web.

O Capítulo 3 mostra a especificação de requisitos do sistema SCAP (Sistema de Controle de Afastamento de Professores). Este capítulo é responsável por descrever as funcionalidades e os objetivos do sistema.

O Capítulo 4 é referente ao Projeto Arquitetural do sistema e como foi implementado utilizando o método *FrameWeb*. Neste capítulo encontram-se os modelos propostos, implementação e tecnologias que foram utilizadas.

O Capítulo 5 descrevem as conclusões desse trabalho. Como o método *FrameWeb* ajudou na implementação quais são as melhorias propostas para melhorar a utilização do mesmo.

2 Referencial Teórico

Neste capítulo estão descritas das bases teóricas que foram utilizados na aplicação do SCAP. Inicialmente são introduzidos os conceitos de Engenharia Web, uma área da Engenharia de Software dedicada a Sistemas Web. Posteriormente, são apresentadas a descrição de cada *framework* utilizado e uma explicação breve de seu funcionamento. Por último, há uma descrição breve de como funciona o método *FrameWeb*.

2.1 Engenharia de Software

A Engenharia de Software é a aplicação de uma abordagem sistemática que possa contribuir para a manutenção de *softwares* e apoiar na implementação de uma aplicação confiável que seja economicamente viável. Uma abordagem de engenharia deve se comprometer com a qualidade e a gestão desta qualidade promover uma cultura de melhoramento contínuo dos processos de uma organização. É com a implementação de processos bem definidos que leva um software ser entregue dentro do prazo (PRESSMAN, 2011).

2.1.1 Engenharia de Requisitos

A Engenharia de Requisitos é um método importante da Engenharia de Software, pois tem como objetivo colher a maior parte de informação relevante para a produção de um sistema. É nesta fase que o engenheiro de requisitos fica em contato mais próximo do cliente para absorver, entender e dar sugestões nas funcionalidades do aplicativo. Esta fase precisa ser feita com muito cuidado para que, no futuro, você possa entregar um produto que atenda ou até exceda as expectativas dos *stakeholders*.

Uma atividade importante na engenharia de requisitos é a especificação dos requisitos. Esta atividade não tem uma definição formal de como deve ser feita, o nível de detalhamento e a formatação pode mudar de acordo com o tamanho e a complexidade do software. Podem ser gerados artefatos como: documentos escritos em linguagem natural, conjunto de modelos gráficos, modelos matemáticos, conjunto de casos de uso, protótipo ou qualquer combinação dos itens citados.

As atividades realizadas na Engenharia de Requisitos do SCAP foram:

- Descrição do Escopo: tem a finalidade de descrever em linguagem natural o fluxo principal do aplicativo e todas as funcionalidades presentes;
- Caso de Uso: apresentam todos atores e seus papéis no sistema, ou seja, quais são as funcionalidades que eles exercem (SILVA, 2007). Após a descrição do escopo, é

necessário que descreva com detalhe cada funcionalidade do sistema e como cada ator realiza tal interage com o sistema;

- Análise: foi realizada com base no paradigma de Orientação a Objetos. Esse paradigma modela os conceitos do problema como um conjunto de classes, onde cada classe tem suas propriedades, métodos e relacionamentos com outras classes. Essas classes devem ser apropriadas para resolverem o problema de forma que seja possível implementar um projeto orientado a objetos (PRESSMAN, 2005);

- Restrições de Integridade: como o modelo de classe não é suficiente para descrever toda lógica de negócio, são criadas regras adicionais que complementam o modelo para melhorar a descrição. Tais regras são chamadas de Restrição de Integridade. Essas restrições existem para representar aquelas informações que não podem ser obtidas a partir de elementos gráficos permitidos no modelo. Essas regras devem ser documentadas e anexadas juntamente com o modelo conceitual estrutural do sistema (FALBO, 2011).

2.1.2 Projeto de Arquitetura

Na fase do projeto arquitetural do sistema partimos de uma solução inicial que foi levantada nas fases de levantamento de requisitos e adequamos para incorporar as tecnologias utilizadas na implementação. Dessa forma, é necessário conhecer quais tecnologias estão disponíveis no mercado e quais destas são adequadas para a implantação em um *hardware* escolhido. Durante o projeto começamos a resolver o problema com níveis mais alto de abstração e devemos aos poucos progredir tomando decisões de projeto que nos levem a níveis cada vez mais concretos, que se aproximam da linguagem de programação (FALBO, 2011).

Na fase de análise não se pensa muito no sistema que essa aplicação será implantada, então por vezes, não se atenta à capacidade de processamento da máquina, quantidade de memória disponível, armazenamento de disco, possibilidade de falha e requisitos não-funcionais de uma forma geral. Então, quando chega-se à fase de projeto, ela envolve a modelagem do problema levando em consideração todos os requisitos tecnológicos e aspectos não funcionais de modo geral (PRESSMAN, 2011).

2.2 Engenharia Web

A Engenharia Web é uma sub-área da Engenharia de Software que é específica para o desenvolvimento de aplicações na Web e como lidar com o seus desafios (PRESSMAN, 2005). A Engenharia Web aborda assuntos relacionados com ao desenvolvimento, implantação e manutenção dos Sistemas Web (MURUGESAN et al., 2001).

Atualmente os aplicativos na Web se tornam vitais para uma organização, uma vez que são parte importante do lucro da empresa. Assim, aplicativos vão crescendo em complexidade e necessitam de um certo padrão de qualidade. Por isso, é necessário que aplique-se uma abordagem sistemática para maior organização. Na medida que as aplicações se tornam grandes, problemas que antes eram presentes em aplicações locais, começam a aparecer nas Aplicações Web, por exemplo: inexperiência da equipe de desenvolvimento, não aplicação de métricas para estimativas de custo, falta de planejamento de processos, métodos inadequados ou obsoletos, entrega de módulos fora do prazo e sem planejamento do custo, péssima experiência do usuário (interface visual), mudança frequente no escopo do projeto, tecnologia, dentre outros (PERUCH, 2007). Sendo assim, a Engenharia Web se torna muito importante e presente em todas as fases do projeto.

Conceitos de atributos de qualidade de um software também são empregados para Engenharia Web e precisam de ser levados em consideração no desenvolvimento (OLSINA; LAFUENTE; ROSSI, 2001), tais como:

- Usabilidade: o sistema deve ser utilizado por todos os tipo de usuários. Desde o técnico ao leigo, e deve ser acessível a portadores de necessidades especiais;
- Funcionalidade: o sistema deve cumprir com os requisitos funcionais para qual foi contratado. Exibindo informações coerentes e realizando os procedimentos corretamente;
- Eficiência: o tempo de resposta esperado para cada funcionalidade deve ser respeitado, sendo sempre o menor possível. Hoje também significa acessibilidade priorizada aos procedimentos mais utilizados;
- Confiabilidade: o sistema deve ser confiável, deve se recuperar após a ocorrência de erros, redirecionamentos devem ser levados a páginas corretas, validação de informações sensíveis devem ser checadas;
- Manutenibilidade: o sistema deve ser de fácil manutenção. Facilidade de adicionar, alterar ou remover módulos. Criação fácil de novas atualizações, mudanças na regra de negócio e aplicação de novas tecnologias. Isso é de extrema importância para um sistema não cair em desuso permitindo a evolução e adaptação para o futuro.

A Engenharia Web utiliza-se de uma abordagem incremental, onde em cada passo é feita uma análise das atividades, projeto, implementação e teste. O resultado final é submetido à aprovação dos *stakeholders*. Na primeira fase o projeto passa pela análise de requisitos, onde são feitas atividades que procuram modelar o sistema como todo e criar o escopo inicial. Nessa fase são levantadas as funcionalidades do sistema (requisitos funcionais) e as restrições (requisitos não funcionais) (SOMMERVILLE, 2007).

São quatro tipos de análises propostas: **conteúdo**, na qual é discutido quais serão as informações e como elas serão mostradas para o usuário; **interação**, que discute como o usuário interage com o sistema; **funcional**, que verifica como o conteúdo é formado a partir das ações do usuário no sistema; e de **configuração**, que discute como será implantada a aplicação e guarda as informações necessárias para a infraestrutura geral.

Assim que são feitas as análises, é proposto elaborar um modelo de projeto que, acrescido dos modelos da fase de análise, é dividido em seis partes: projeto de conteúdo, projeto arquitetural, projeto de estética, projeto de navegação, projeto de interface e projeto de componentes (PRESSMAN, 2005). Alguns métodos podem ser utilizados no desenvolvimento, como: WAE (CONALLEN, 2002), OOWS (PASTOR; FONS; PELECHANO, 2003), e o próprio FrameWeb como descrito na Seção 2.4.

Logo depois entra na fase da implementação, na qual é escolhida uma linguagem de programação que atenda aos requisitos do sistema e outras características operacionais como latência e escalabilidade. Na finalização, é feita na fase de testes, todos os testes que garantem que a aplicação foi implementada com todos os requisitos desejados, interoperabilidade (teste em vários dispositivos e navegadores diferentes), segurança da informação, eficiência, etc.

2.3 Frameworks

Com o passar do tempo foi percebendo-se que aplicações Web possuem características similares, pensando na reutilização de código e otimização de tempo, foram desenvolvidos modelos que proviam partes já implementadas dessa infraestrutura. Estes modelos eram aplicações Web generalizadas que podiam ser reutilizadas para diversos tipos de Aplicações Web, futuramente receberiam o nome de *frameworks*.

Frameworks, portanto, são aplicações incompletas que podem ser personalizadas para cada tipo de sistema (HUSTED, 2004). A utilização dessa ferramenta possibilita que códigos já prontos sejam mesclados ao código do programador para que, de forma muito mais dinâmica e otimizada, sejam desenvolvidas aplicações desde o início.

Um exemplo de utilização de código de *frameworks* é prover uma camada de abstração de Banco de Dados, que usualmente é feita por Banco de Dados Relacional. Alguns *frameworks* modularizam partes dos códigos de forma independentes, dando para o projetista a possibilidade de combinar diferentes módulos. Souza (2007) organiza os tipos de *frameworks* em seis categorias específicas:

- Frameworks MVC (Controladores Frontais);
- Frameworks Decoradores;

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

Nas próximas seções serão descritas em mais detalhes os *frameworks* de MVC, Mapeamento Objeto/Relacional, Injeção de Dependência e Autenticação e Autorização utilizados neste trabalho. Discutimos também como o Quasar e o AdonisJS implementam e integram cada módulo desse.

2.3.1 NodeJS

NodeJS é um interpretador de Javascript[1] assíncrono baseado no motor Javascript V8 do navegador Chrome. É conhecido pelo seu desempenho e se destacou por levar códigos de Javascript antes criados no cliente (*client-side*) para o Servidor (*server-side*) criando, assim, servidores Web altamente eficientes e escaláveis. Podem gerenciar milhares de requisições simultâneas em uma única máquina utilizando a mesma linguagem de programação utilizada pelo cliente.

O NodeJS vem acompanhado do NPM (*Node Package Manager*) ou gerenciador de pacotes Node, que provê plugins e bibliotecas que podem ser integrados ao seu código.

2.3.2 Frameworks MVC

De acordo com Souza (2007), o MVC (*Model-View-Controller*) ou Modelo-Visão-Controlador foi baseado no modelo proposto pelo Centro de Pesquisas Xerox para a linguagem SmallTalk em 1979 (REENSKAUG, 1979).

Com o intuito de melhorar a manutenção de sistemas de grande porte, o *framework* MVC, separa partes do sistema de forma a manter os dados (*Model*) bem separados do *layout* (*View*). Dessa forma o *framework* garante que alguma mudança na camada de apresentação não tenha consequências diretas na camada de persistência da aplicação. Para a comunicação dessas camadas, existe uma camada intermediária chamada Controle (*Controller*).

Para exemplificar, numa aplicação de simulação de financiamento imobiliário o cliente preenche um formulário com os dados pessoais, qual a entrada do financiamento, e em quantas parcelas ele deseja financiar. Ao clicar em "enviar formulário", a ação do botão dispara um evento que é enviado para um intermediário (*Controller*) que envia os dados

[1] <https://www.javascript.com/>

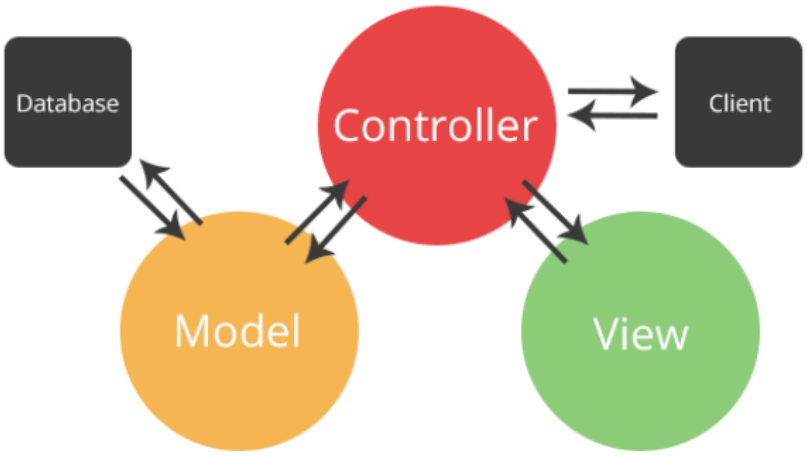


Figura 1 – Padrão MVC (PROGRAMMINGHELP, 2013).

para o Modelo. O modelo é responsável por realizar os cálculos necessários da regra de negócio do financiamento imobiliário e retornar o valor de cada parcela do financiamento para a Visão.

O controlador é o único que conhece a camada de lógica. Portanto, essa arquitetura de MVC, quando utilizada em Sistemas Web, precisa ser adaptada, pois o controlador não pode notificar diretamente a *View* sobre alterações, já que esta se encontra no navegador e a comunicação sempre é iniciado pelo lado do cliente. Por isso é necessário criar o conceito de controlador frontal, que é acionado pelo Servidor Web com uma requisição da *View*, o controlador que executa uma ação. A Figura 1 demonstra o comportamento.

2.3.3 MVVM

O padrão MVVM (*Model, View, View-Model*) foi utilizado a primeira vez por John Gossman, arquiteto de software da *Microsoft*,[2] originalmente criado para aplicativos *Windows Presentation Foundation* (WPF),[3] é uma variação do padrão PM (*Presentation Model*) (FOWLER, 2004). A diferença desse padrão é que uma abstração da *view* é criada e comportamentos e estados são separados da camada de apresentação (SMITH, 2009). O padrão MVVM e PM são todos derivados do padrão MVC.

Nesse padrão, semelhante de como propõe o padrão MVC, o *Model* representa um objeto referente ao nível do domínio, representando uma camada de acesso aos dados, enquanto a *View* representa a camada de apresentação, ou seja, tudo aquilo que o usuário consegue ver e interagir. Porém, diferentemente do *Controller* no MVC, temos um elemento chamado *Model* que não é separado da visualização. Ele pode invocar métodos provenientes da camada de Modelo e provê um mecanismo para sincronizar a *View* que suporta o fluxo

[2] <https://www.microsoft.com>
[3] <https://msdn.microsoft.com/pt-br/library/cc564903.aspx>

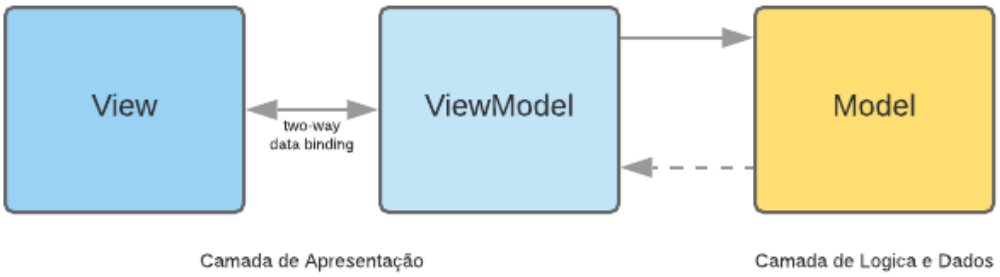


Figura 2 – Padrão MVVM.

de dados nos dois sentidos (*two-way data binding*), facilitando, assim, a propagação de mudanças entre *View-Model* e *View*. Na Figura 2 é possivel ver esse comportamento.

2.3.4 VueJS e Quasar

VueJS[4] é um poderoso *framework* que utiliza o padrão MVVM. Ele é voltado para a camada de visualização, apenas para a implementação de interfaces com o usuário. Utiliza-se de componentes com propriedades reativas para construir o HTML de forma dinâmica. Ele pode ser facilmente integrado em aplicações com HTML e Javascript puro, por meio de importação de bibliotecas, ou utilizando um projeto *Node*, construindo aplicações utilizando-se de várias bibliotecas de suporte e ferramentas modernas opcionais.

Quasar[5] é um *framework* de código aberto licenciado pelo MIT, baseado em VueJS, para a criação rápida de *websites/apps* responsivos. Utiliza uma biblioteca gráfica que implementa o padrão de interface criado pela Google (Material Design[6]). Dá suporte a várias plataformas, tipos de navegadores diferentes e as principais características existentes do Quasar são:

- SPA (*Single Page Application*): aplicações implementadas em apenas uma página HTML. O Conteúdo é alterado dinamicamente por meio de funções de *Javascript*. Aplicações implementadas com esse conceito permitem que a renderização das páginas possa ser feita exclusivamente pelo navegador do cliente, sendo o papel do servidor apenas prover dados.
- SSR (*Server Side Rendering)*: aplicações renderizadas pelo servidor. Dessa forma todo o processamento da página é feita pelo servidor, sendo de responsabilidade do navegador apenas a exibição do cliente.

[4] <https://vuejs.org/>
[5] <https://quasar.dev/>
[6] <https://material.io/design>

- PWA (*Progressive Web Application*): aplicações Web que utilizam de ferramentas provenientes de navegadores para quando utilizadas por um dispositivo *mobile* tenha a maior proximidade com aplicativos nativos da plataforma. São tentativas de fazer que o usuário tenha uma melhor experiência ao acessar o sistema.

- Electron:[7] plataforma que utiliza de HTML, *Javascript* e CSS (Cascading Style Sheet), formatação de estilos utilizados no HTML, para gerar aplicativos de multi-plataforma para *Desktop*.

- Cordova:[8] plataforma que utiliza de HTML, *Javascript* e CSS para gerar aplicativos multi-plataforma para dispositivos móveis.

Além disso da suporte na criação de extensões para navegadores e é compatível com *TypeScript*,[9] *superset* de *Javascript* que introduz tipagem estática. *TypeScript* é traduzido no final para código de *Javascript* puro.

2.3.5 Frameworks de Mapeamento Objeto/Relacional

Hoje em dia, grande parte dos dados das aplicações são persistidas em banco de dados relacionais. Porém a maior parte das linguagens utilizadas para estruturação da aplicação no servidor são orientadas a objetos. Isso causa um problema de incompatibilidade de paradigmas (*paradigm mismatch*) (SOUZA, 2007).

Isso se torna um desafio na programação, pois em Banco de Dados relacionais é utilizada SQL (linguagens estruturadas de consulta para SGBDR's), que busca dados em tabelas associadas por chaves. Isso é bem diferente de grafos de objetos relacionados que é utilizado na maioria das linguagens de programação (BAUER; KING, 2005).

Então, surgem na decada de 80, os primeiros ORM's (Mapeamento Objeto/Relacional), que mapeia Tabelas em Classes e registros (linhas) de tabelas em instancias (objeto) dessas mesmas classes. Assim, o programador não precisa se preocupar como será executado o SQL que persistirá os dados da sua aplicação, apenas precisa sincronizar os objetos (modelos) que estão instanciados na sua aplicação com a tabela/registro correspondente. Isso faz com que erros sejam menos frequentes ao recuperar ou salvar dados.

2.3.6 Frameworks de Injeção de Dependência

O objetivo desses *frameworks* é prover baixo acoplamento entre as classes, fazendo, assim, que alguns objetos não sejam responsáveis em instanciar seus objetos de dependências. Essa obrigação fica a cargo do *framework* de injeção de dependência, que injetará uma

[7] <https://www.electronjs.org/>
[8] <https://cordova.apache.org/>
[9] <https://www.typescriptlang.org/>

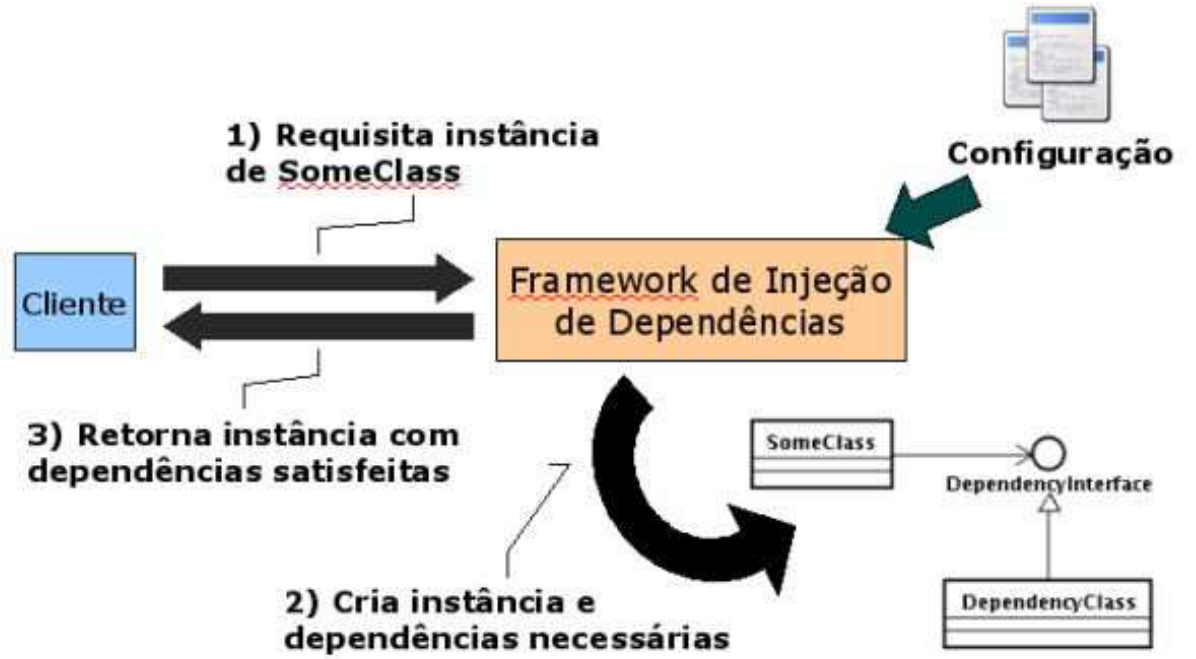


Figura 3 – Injeção de Dependências (SOUZA, 2007).

instancia para esses objetos automaticamente. Esse comportamento pode ser mostrado na Figura 3

2.3.7 AdonisJS

AdonisJS é um *framework* MVC para NodeJS. Oferece um grande suporte para aplicações no servidor (*server-side*). Pode ser utilizado tanto para aplicações *full-stack*, que criam a visualização juntamente com a lógica de negócio, ou aplicações apenas *server-side* ou até API de micro serviços.

O AdonisJS utiliza como *Frameworks* de Mapeamento Objeto/Relacional o LucidJS. Possui injeção de dependência no controle nativa.

2.4 FrameWeb

O FrameWeb (SOUZA, 2020) é um método para desenvolvimento de Sistemas de Informação Web (WISs). Utiliza-se de categorias de *frameworks* para guiar o desenvolvimento. Mesmo que haja muitos *frameworks*, não tinha nenhuma proposta de reuni-los em categorias e discutir a característica que cada categoria contém e como pode auxiliar no desenvolvimento de aplicações de Engenharia Web.

O método FrameWeb, hoje, está focado em auxiliar nas fases do projeto arquitetural, deixando a cargo dos projetistas e programadores livres para decidirem sobre as outras

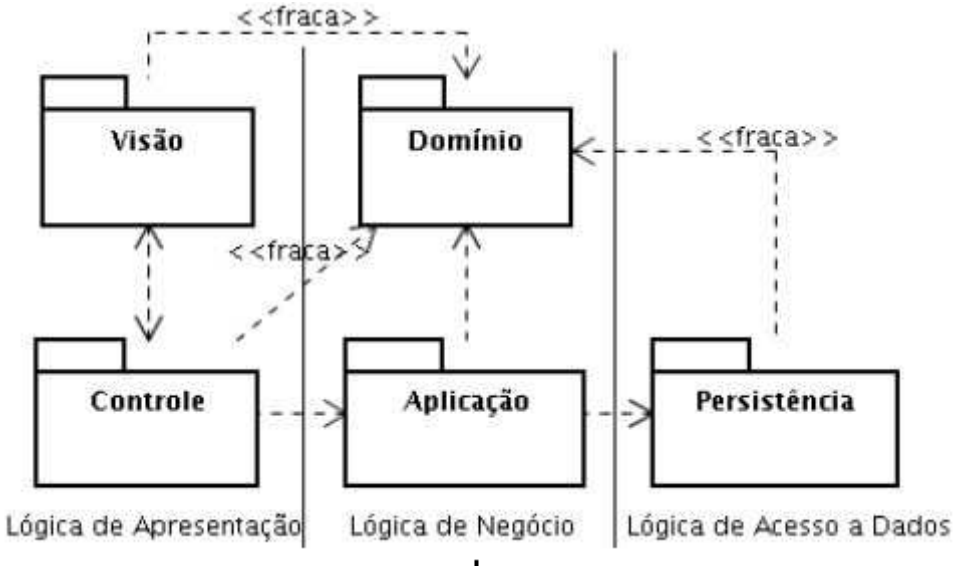


Figura 4 – Padrão de arquitetura WIS baseada no padrão de Camada de Serviço (FOWLER, 2002).

áreas do projeto. A proposição é de que a arquitetura WIS seja dividida em três partes, como mostra a Figura 4.

- Lógica de Apresentação: a lógica de apresentação é responsável por prover as interfaces gráficas para a aplicação. É a camada que interage diretamente com o usuário, dando opção de realizar alguma ação e esperar pela resposta. É subdividida em Visão e Controle:
  - Visão: contém os arquivos de página Web (HTML), estilização (CSS), arquivos de script (JS), imagens, ícones, fontes, etc. É responsável por captar todo o tipo de interação que o usuário tem com o sistema. Para cada estímulo será criado um evento que será posteriormente tratado e enviado para o controle.
  - Controle: contém classes de controle. Esta camada validará os dados e montará o formulário de requisição para o cliente. Essa parte também é relacionada a lógica de negócio que proverá as funcionalidades do sistema.
- Lógica de Negócio: provê classes que executam as funcionalidades do sistema. Contém os pacotes de domínio e aplicação:
  - Domínio: contém as classes provenientes dos conceitos extraídos na fase de análise. São conceitos proveniente do domínio do problema.
  - Aplicação: contém a implementação de conceitos extraídos na parte de análise de requisitos, provendo assim uma camada de serviço que é independente da interface. A aplicação tem dependência com as classes de domínio e também com o pacote de persistência.

- Lógica de Acesso de Dados: é a camada que provê o acesso as informações de persistência. Faz interface com o banco de dados. Contém um único pacote, pacote de persistência, que é responsável por criar, atualizar, recuperar e remover dados que precisam ser persistidos no sistema. O FrameWeb recomenda o padrão DAO (*Data Access Object*) (ALUR; CRUPI; MALKS, 2003), ou Objeto de Acesso a Dados, que é um objeto que representa um registro de uma tabela e torna a aplicação independente do *framework* ORM (PRADO, 2015).

O FrameWeb segue o mesmo padrão de linguagens de modelagens propostas na engenharia de software. Da mesma forma que WAE (CONALLEN, 2002) e UWE (KOCH et al., 2000), utilizando diferentes formas de representação baseados no metamodelo de UML (SOUZA, 2016), os modelos de FrameWeb representam os componentes mais utilizados na Web e como eles interagem com os *frameworks*. Com o objetivo de representar melhor as camadas citadas anteriormente foram criados quatro tipos de modelo (SOUZA, 2007; SOUZA, 2016):

- Modelo de Entidades: representa os conceitos e as relações entre os conceitos presentes do domínio do problema. Especifica os dados e o mapeamento para classes de domínio que serão persistidas.
- Modelo de Persistência: auxilia na criação das classes DAO do sistema, mostrando todos os dados, interfaces e métodos que serão implementados.
- Modelo de Navegação: um diagrama de classes que representa a dinâmica da camada de apresentação do sistema, como páginas Web interagem entre si para requisitarem uma certa funcionalidade.
- Modelo de aplicação: mostra as classes de aplicação do sistema e como interagem com suas próprias dependências.

3 Especificação de Requisitos

Esse capitulo tem como objetivo descrever o escopo do SCAP (Sistema de Controle de Afastamento de Professores) e apresentar os modelos de caso de usos que foram levantados por Duarte (2014) e Prado (2015).

3.1 Descrição do Escopo

O SCAP é um sistema para gerenciar os pedidos de afastamentos de professores do Departamento de Informática (DI). As solicitações de afastamentos são feitos para participação de eventos que ocorrem no Brasil ou no exterior. Estas solicitações passam pela avaliação dos professores do DI. Em alguns casos, oodem passar pela avaliação do Conselho do Centro Tecnológico (CT) e também pela Pró-Reitoria de Pesquisa e Pós Graduação (PRPPG), após passar por todos esses estágios o professor é afastado temporariamente e pode realizar a viagem.

Quando o pedido de afastamento é para um evento dentro do Brasil, o processo tramita apenas dentro do DI. Os pedidos são aprovados pela Câmara Departamental (representada pelos funcionários do departamento e os representantes discentes). É feita uma solicitação por e-mail para a lista de e-mails da Câmara Departamental, endereçada ao Chefe de Departamento, cargo temporário exercido por algum professor do DI. Caso ninguém se manifeste contra o afastamento num determinado prazo, o pedido é aceito.

Para afastamentos para um evento que acontece fora do Brasil, é escolhido um professor (que não tenha nenhum parentesco com o solicitante), para ser o relator do processo. Quando o relator emitir o seu parecer, o processo é encaminhado e aguarda pela aprovação do Departamento de Informática, assim como no caso anterior, porém, ainda é encaminhado para aprovação do Centro Tecnológico e da PRPPG. Após ter sido aprovado em todos os setores o afastamento é aprovado e publicado no Diário Oficial da União, conforme o art. 95 da lei 8.112 (<http://www.planalto.gov.br/ccivil_03/leis/l8112cons.htm>).

O SCAP não tem nenhuma comunicação com os sistemas da PRPPG ou do CT, dessa forma, afastamentos internacionais apenas são registrados no SCAP quando o processo é retornado ao DI.

O SCAP tem o objetivo de acelerar o processo de solicitação de afastamento, ele atinge seu objetivo enviando e-mails automáticos para os envolvidos e também a utilização de formulários para ajudar na criação dos documentos necessários.

3.2 Casos de Uso

Na Tabela 1 estão listados todos os atores do sistema.

Tabela 1 – Atores

Ator	**Descrição**
Professor	Professor efetivo do DI.
Chefe de Departamento	Professor eleito para um mandato temporário e realiza a função administrativa de chefe ou sub-chefe.
Secretário	Secretário do DI.

O Secretário lida com a parte cadastral do sistema. Ele é responsável pelo o cadastro de professores e seus parentes, é também sua responsabilidade manter o histórico e registrar quem ocupa os cargos de Chefe e Sub-Chefe do DI. Quando solicitado, é necessário que faça o registro do parecer do CT e do parecer da PRPPG. Quando finalizado, após a regularização, é responsável por arquivar o processo de afastamento.

Um Professor pode criar suas próprias solicitações de afastamento, ou emitir um parecer, que pode ser favorável ou não. O professor pode ser Relator de algum pedido de afastamento de evento internacional. O Chefe de Departamento além de poder realizar todas as funcionalidades do Professor, é responsável por indicar um relator para dar o parecer sobre uma solicitação.

O SCAP é dividido em dois módulos. O primeiro responsável pela parte cadastral ou de secretaria, o outro, responsável por gerenciar professores e chefes de departamento. Esses núcleos podem ser visualizados pelos modelos das Figuras 5 e 6. O restante dessa seção se concentrará em descrever de forma sucinta cada funcionalidade dos modelos de caso de uso. A descrição completa dos modelos podem ser obtidas em (DUARTE, 2014; PRADO, 2015).

Ao **Solicitar Afastamento** o professor preenche os dados de motivo de afastamento, data de início, data de fim e os dados do evento: data de início, data de fim, se o evento é internacional ou nacional e o nome do evento. Pode **Cancelar Solicitação**, quando essa solicitação foi feita por ele mesmo. Pode **Consultar Afastamentos** onde ele visualiza todas as solicitações feitas nos sistema por outros professores e as próprias solicitações. Um professor pode ainda **Manifestar-se Contra Afastamento** onde ele preenche o motivo da manifestação negativa. O Chefe de Departamento deve, no caso de evento internacional, **Encaminhar Afastamento** para um Relator. Este deve, quando solicitado pelo Chefe de Departamento, **Deferir Parecer**.

Em **Cadastrar Usuário**, o Secretário pode cadastrar um novo professor ou um novo secretário, sendo necessário preencher os dados pessoais do novo funcionário. Para registrar um novo Chefe de Departamento, ele deve informar a data final do mandato que se encerra (se algum) e a data inicial do novo mandato.

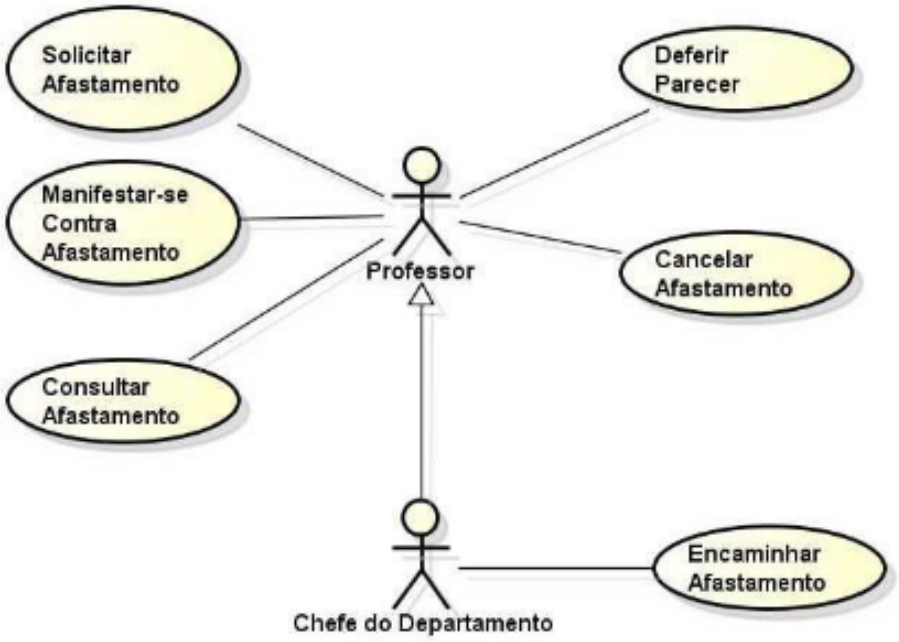


Figura 5 – Diagrama de Casos de Uso do subsistema Núcleo (PRADO, 2015).

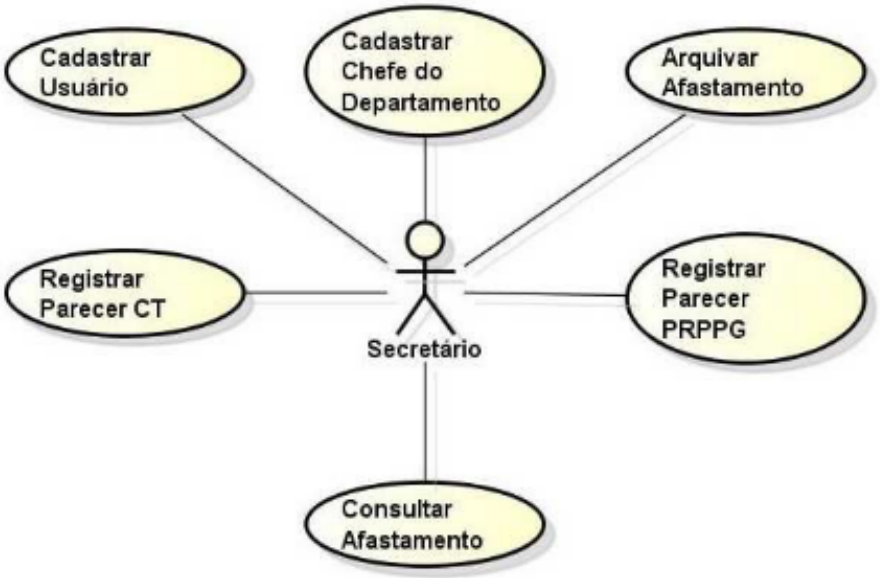


Figura 6 – Diagrama de Casos de Uso do subsistema Secretaria (PRADO, 2015).

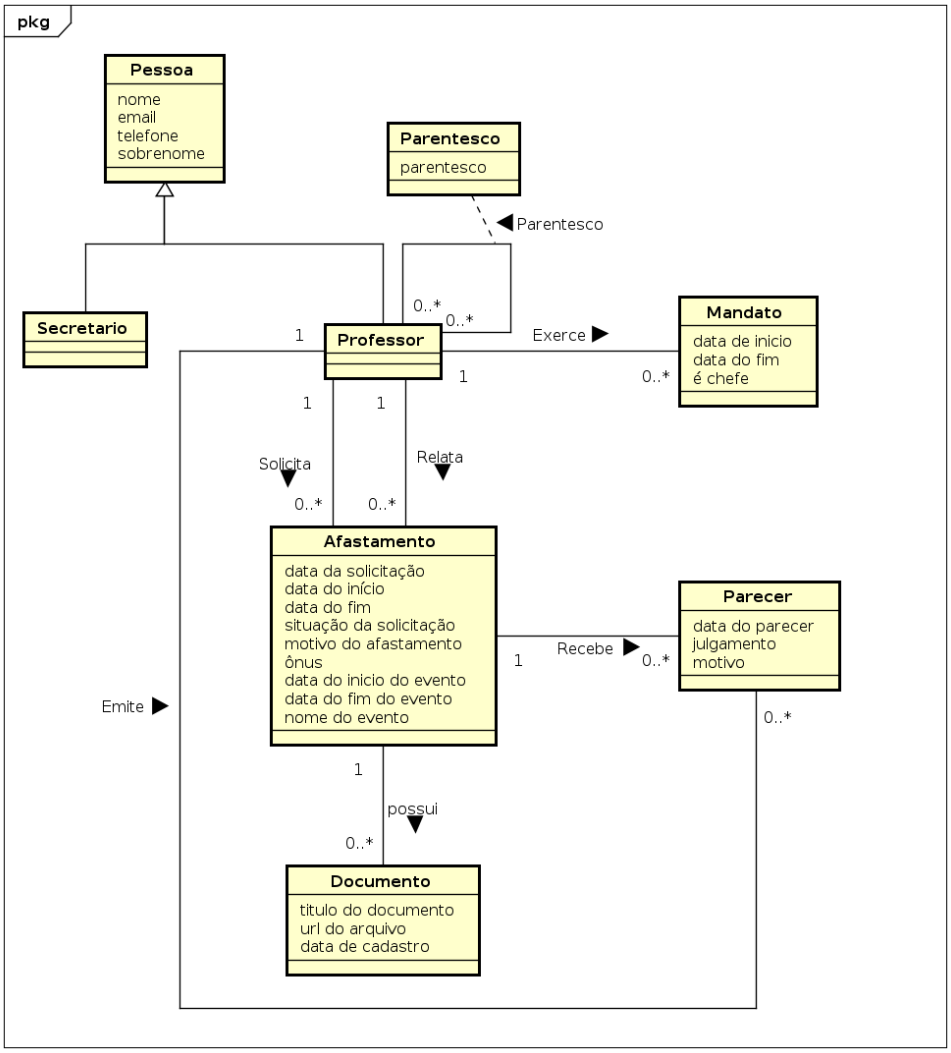


Figura 7 – Diagrama de Classes do SCAP

Os casos **Registrar Parecer CT** e **Registrar Parecer PRPPG** acontecem quando um secretário recebe um parecer do Centro Tecnológico ou da Pro-Reitoria de Pesquisa e Pós-Graduação sobre algum pedido de solicitação de afastamento iniciado pelo DI sobre um evento internacional. Ele utiliza estas funcionalidades para registrar a resposta destes órgãos no sistema.

Na finalização, o secretário deve **Arquivar Afastamento**, no qual faz o fechamento de um registro de afastamento concluído.

3.3 Análise do SCAP

O modelo de classes presente na Figura 7 é uma adaptação do levantamento feito por Prado (2015), que implementou o SCAP utilizando o *framework* VRaptor, incrementando o que havia antes sido levantado por Duarte (2014).

A Classe **Pessoa** contém as informações pessoais dos usuários do sistema. **Profes-**

sor e **Secretário** herdam os atributos de pessoas e contém os dados e relações respectivamente dos usuários com papel de professor e secretário. A classe **Parentesco** guarda a relação de parentesco que existem entre os professores do Departamento. O Chefe de Departamento é um professor que possui um **Mandato** vigente.

A classe **Afastamento** representa todas as solicitações de afastamentos feitas por professores que serão analisadas. A classe **Afastamento** pode ou não possuir **Documentos** associados, possui um **Relator** caso o afastamento seja de algum evento internacional.

O **Parecer** é criado por Professores e é sempre relativo a apenas um único afastamento, porém um único afastamento pode conter vários pareceres relacionados. Professores podem ser relatores de vários afastamentos.

Listamos aqui todas as restrições de integridade que foram levantadas por Duarte (2014) e juntamente com as novas restrições que foram levantadas posteriormente por Prado (2015):

- Um professor não pode ser indicado para dar um parecer sobre a sua própria solicitação de afastamento;
- A data de início de um afastamento não pode ser posterior a data final do afastamento;
- A data de início do mandato de um professor não pode ser posterior a data final do mesmo mandato;
- Não pode haver mais que dois professores exercendo cargos administrativos (chefe e subchefe de departamento) em um mandato ao mesmo tempo;
- Um professor não pode ser relator de um processo de afastamento solicitado por algum parente.

4 Projeto Arquitetural e Implementação

Nesse capítulo está descrito, de forma técnica, como foi o processo de projeto e implementação do Sistema SCAP, utilizando os *frameworks* QuasarJS e AdonisJS e cada elemento arquitetural que foi utilizado.

Todos os casos de uso presentes nas Figuras 5 e 6 foram implementados neste trabalho. A único aspecto presente no escopo original que não foi implementado é o papel do professor sub-chefe do departamento de informática, este detalhe não interfere no objetivo principal que é a aplicação do método *FrameWeb* na aplicação.

4.1 Comparação com trabalhos Anteriores

Na Tabela 2 é possível visualizar as diferentes implementações do SCAP.

Tabela 2 – Implementações anteriores do SCAP.

Trabalho	***Framework* MVC**	**Tecnologia ORM**	**Tecnologia DI**
Duarte (2014)	JSF (Java)	Hibernate (Data Mapper)	CDI
Prado (2015)	VRaptor (Java)	Hibernate (Data Mapper)	CDI
Pinheiro (2017)	Laravel (PHP)	Eloquent (Active Record)	Não utilizou
Matos (2017)	Spring (Java)	Hibernate (Data Mapper)	Embutido no Spring
Matos (2017)	Vaadin (Java)	Hibernate (Data Mapper)	Não utilizou
Avelar (2018)	Ninja (Java)	Hibernate (Data Mapper)	Google Guice
Ferreira (2018)	Wicket (Java)	Hibernate (Data Mapper)	CDI
Ferreira (2018)	Tapestry (Java)	Hibernate (Data Mapper)	Embutido no Tapestry
Meirelles (2019)	CodeIgniter (PHP)	Embutido no CodeIgniter (Active Record)	Não utilizou
Meirelles (2019)	NodeJS (JavaScript)	Não utilizou	Não utilizou
Guterres (2019)	Play (Scala)	Slick (Padrão não identificado)	Não utilizou
Este trabalho	AdonisJS e Quasar (Javascript)	Lucid (Active Record)	Embutido no AdonisJS

Para esse trabalho foram desenvolvidas duas diferentes aplicações. Uma aplicação Quasar, que contém apenas a parte da visualização do sistema onde estão contidos os componentes gráficos que comunicam por requisições HTTP. A outra aplicação é uma API

RESTful que implementa a arquitetura REST (FIELDING, 2000) utilizando o *framework* AdonisJS, que recebe as requisições e as processa e enviando novamente à camada de aplicação as respostas em formato JSON.

As implementações de Duarte (2014), Prado (2015), Matos (2017), Avelar (2018), Ferreira (2018), Meirelles (2019) e Guterres (2019) utilizaram o padrão MVC com controlador frontal por padrão e implementaram de forma monolítica o sistema. O servidor é responsável por gerar a página inteira e disponibilizar para que o navegador possa executar. Existem vantagens ao se programar sistemas monolíticos, pois eles são mais rápidos na implementação, fáceis na implantação, requerem menos investimento com infra-estrutura, facilidade ao garantir a segurança e os desafios com autenticação e autorização são simplificados. É recomendado para aplicações com escopo pequeno e com baixo número de usuários ou quando a equipe de desenvolvimento é reduzida. Porém existem também desvantagens, encontram-se barreiras em sistemas monolíticos quando é necessário escalar a aplicação, principalmente quando deseja-se escalar apenas parte da aplicação, o alto acoplamento dos componentes se transforma-se em um dificultador.

Hoje em dia, com a popularização das API's RESTful, impulsionada pela demanda de aplicações *Mobile* (para Celulares) e aplicações Web SPA, percebe-se um grande avanço no desenvolvimento de sistemas com aplicações separadas com baixo acoplamento. Percebe-se uma grande facilidade ao desenvolver assim, pois é possível separar o desenvolvimento em duas frentes principais que podem ser desenvolvidas em paralelo. As aplicações se tornam mais fluídas, uma vez que não é necessário recarregar a página completamente para visualizar algum dado. Os trabalhos que se utilizaram desta técnica foram Pinheiro (2017) e este próprio trabalho, além disso implementaram utilizando o padrão MVVM com o mesmo *framework* no lado do cliente, Quasar.

A linguagem de programação mais utilizada foi Java, utilizando a tecnologia ORM *Hibernate* como Mapeamento objeto/relacional, que utilizam *Data Mapper*. Este mapeamento faz com que haja um desacoplamento do banco de dados mais eficiente, a entidade (objeto que representa os dados nesse tipo de mapeamento) não está ligada diretamente aos registros e delega para a camada de persistência o papel de comunicação com o banco de dados.

Diferentemente do *Data Mapper*, alguns trabalhos utilizaram *frameworks* de ORM de *Active Record*. A representação dos dados nesse tipo de dados é feita através de um modelo fortemente acoplado ao banco de dados. Estes modelos estendem a funcionalidade de um modelo base que contém todas as instruções para atualizar ou recuperar dados do banco de dados. Trabalhos que utilizaram este tipo de mapeamento foram mostrados na Tabela 2.

O único *framework* ORM que não seguiu um padrão conhecido foi o *Slick*[1] que segundo a própria documentação ele diferencia-se por utilizar coleções imutáveis presente na linguagem de programação *scala* ao invés de utilizar grafos de objetos mutáveis como presentes em outros *frameworks* como o *Hibernate*. Isso reduz os efeitos colaterais de assincronismo entre objeto e banco de dados presentes em ORM's de forma geral.

Em relação à Injeção de Dependências ou Inversão de Controle, alguns trabalhos não utilizaram nenhum tipo de injeção de dependência (Pinheiro (2017), Matos (2017) na implementação com Vaadin, Meirelles (2019) e Guterres (2019)). Outros trabalhos utilizaram a injeção de dependência ou *IoC Container* presentes no próprio *framework* MVC. Nas aplicações Java foram utilizados os *frameworks* CDI muito conhecido para a plataforma Java EE (*Java Enterprise Edition* ou Java edição para empresas), *Google Guice* (Plataforma Google para suporte à injeção de dependencia em Java).

4.1.1 Tecnologias Utilizadas

Na Tabela 3 são listadas as tecnologias utilizadas no desenvolvimento da ferramenta, bem como o propósito de sua utilização.

Tabela 3 – Plataforma de Desenvolvimento e Tecnologias Utilizadas

Tecnologia	Versão	Descrição	Propósito
JavaScript	ES6	Linguagem de programação orientada a eventos e independente de plataforma.	Criação de páginas Web, Processamento de requisições HTTP, implementação da regra de negócio.
PostgreSQL	13	Sistema Gerenciador de Banco de Dados Relacional gratuito.	Armazenamento de dados manipulados pelo sistema.
PgAdmin	4	Plataforma gratuita de administração e desenvolvimento para PostgreSQL.	Facilitar o gerenciamento dos dados da aplicação.
NodeJS	14	Plataforma de aplicação para desenvolvimento em JavaScript.	Pré-compilação e otimização de código JavaScript tanto para servidor quanto para o navegador.
NPM	6	Gerenciador de Pacotes NodeJS.	Gerencia e facilita o download de pacotes NodeJS para utilização em aplicações baseadas em JavaScript.
AdonisJS	4	*Framework* MVC para a criação de sistemas Web.	Gerenciar requisições e fornecer suporte para implementação das regras de negócio.
Lucid ORM	1	API para persistência de dados por meio de mapeamento objeto/-relacional.	Persistência dos objetos de domínio sem necessidade de escrita dos comandos SQL.
VueJS	2	*Framework* JavaScript para construção de interface de usuário.	Provê componentes de interface reativos para simplificar a forma de desenvolvimento de interfaces com o usuário.
QuasarJS	1	*Framework* baseado em VueJS para criação de websites/apps responsivos.	Criar a parte visual do sistema utilizando SPA (*Single Page Application*) com o padrão *Material Design*.

[1] <https://scala-slick.org/doc/3.0.0/orm-to-slick.html>

Na Tabela 4 vemos os softwares que apoiaram o desenvolvimento de documentos e também do código fonte.

Tabela 4 – Softwares de Apoio ao Desenvolvimento do Projeto

Tecnologia	Versão	Descrição	Propósito
Astah UML	1.0	Ferramenta de construção de modelos UML.	Criação dos modelos de Entidades, Aplicação, Persistência e Navegação.
TeX Live	2019	Implementação do LaTeX	Documentação do projeto arquitetural do sistema.
TexMaker	5.0	Editor de LaTeX.	Escrita da documentação do sistema, sendo usado o *template abnTeX*.[2]
Visual Studio Code	1.52	Ambiente de desenvolvimento (IDE) com suporte ao desenvolvimento Para várias Linguagens.	Implementação, implantação e testes da aplicação Web NodeJS.

4.2 Arquitetura do Sistema

Na fase arquitetural do sistema, os modelos da fase de análise são mapeados para elementos arquiteturais que utilizam as tecnologias de desenvolvimento para implementar os conceitos. Para se aproximar da linguagem de programação é necessário diminuir a abstração para elementos que possam ser representados agora na linguagem de programação alvo da arquitetura.

A estrutura principal utilizada pelo AdonisJS é o MVC, porém essa estrutura foi alterada, já que a parte da Visualização (*View*) do MVC é de responsabilidade de implementação do Quasar. Além dessa alteração foi utilizado o padrão de camadas de Serviço e Repositório para ajudarem na modularização e teste do código. Essas camadas podem ser vistas na Figura 8.

Na figura, os retângulos representam as camadas do sistema, as elipses são *middlewares*, ou seja, interceptadores de requisições para executarem certas rotinas. A camada de apresentação não é presente no AdonisJS, mas é desta camada que a comunicação com o servidor é iniciado. Utilizando-se de requisições assíncronas AJAX (*Asynchronous JavaScript and XML*), a camada de aplicação se comunica com o servidor enviando os dados. Nos *middlewares* de autenticação é verificado se o usuário foi identificado e na camada de autorização verifica se o usuário pode ter acesso àquele módulo.

Após a formatação dos dados no Controle, é chamada a camada de Serviço que lida com a parte da lógica do sistema. A camada de Serviço utiliza a camada de repositório, esta

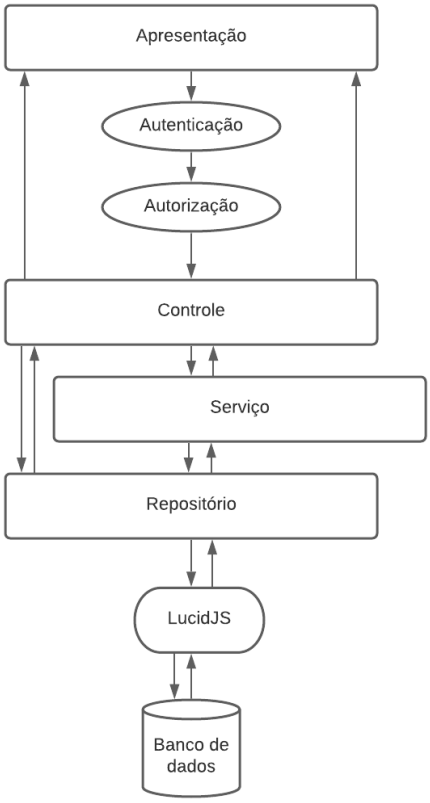


Figura 8 – Representação das camadas utilizadas no AdonisJS

camada é responsável por persistir as informações no banco de dados, atualizar recuperar e remover modelos obtidos pelo mapeamento objeto/relacional, que é feito através do *framework* LucidJS. O Sistema Gerenciador de Banco de Dados utilizado foi o PostgreSQL.

4.2.1 Lado do Servidor

No lado do servidor foi utilizado o *framework* AdonisJS. Este *framework* já tem integrado um servidor Web NodeJS, não sendo necessário utilizar um servidor externo. A estrutura de diretórios do AdonisJS é dividida em:

- app: Pasta principal da lógica de negócio, que contém os *Repositories*, *Controllers*, *Models* e *Services*:
  - Models: pasta que contém os arquivos dos modelos do padrão MVC. O modelo é o resultado do mapeamento Objeto/Relacional, ou seja, cada classe de modelo é referente a uma tabela do banco de dados e cada objeto instanciado dessa classe representa um registro desta mesma classe;
  - Repositories: pasta que contém os arquivos referentes à camada de persistência do banco de dados, se comunica diretamente com os *Models*. O objetivo destes arquivos é prover dados para a aplicação;

  - Services: pasta que contém os arquivos referentes à parte da camada de aplicação. Esses arquivos utilizam as camadas de repositórios para prover uma funcionalidade para o sistema, por exemplo a execução de um caso de uso;
  - Controllers: arquivos responsáveis por receberem a requisição HTTP do cliente e formatar os dados para serem utilizados posteriormente. São referentes aos *Controllers* do padrão MVC;

- config: pasta que contém arquivos de configurações dos *plugins* do AdonisJS;
- database: pasta que contém os arquivos referentes aos bancos de dados:
  - migrations: pasta que contém arquivos que são um histórico de comandos SQL que foram usados para montar toda a estrutura de banco de dados do sistema;
  - seeds e factories: arquivos que populam o banco de dados com dados falsos (*dummy data*), que são usados apenas para testes;
- public: pasta que contém os arquivos públicos e podem ser acessado via URL;
- node_modules: pasta que contém as dependências dos *plugins* instalados pelo NPM;
- start: pasta que contem o arquivo de rotas e as configurações de *middlewares* para inicialização do servidor;
- server.js: arquivo que inicialização do servidor.

4.2.2 Lado do Cliente

O *framework* no lado do cliente é responsável por apresentar a parte visual e interagir com o usuário. O *framework* utilizado foi o Quasar, este *framework* é baseado em VueJS e indicado para quem quer construir interfaces responsivas para SPA's com intuito de poder reutilizá-las em outras plataformas. O Quasar, com poucos ajustes, permite o programador a integrar no projeto diferentes tipos de visualização, dentre elas: Sistemas Web SPA, PWA, Aplicativos *Mobile*, etc.

A estrutura de arquivos de um projeto Quasar é definida da seguinte forma:

- dist: pasta alvo do comando *build*, responsável por minificar (remover espaços em branco e quebras de linha) os arquivos *Javascript*, otimizar o código e aplicar configurações necessárias para cada tipo de visualização;
- node_modules: pasta onde ficam as dependências do *plugins* provenientes do NPM;
- public: pasta onde são gerados os ícones do aplicativo ou PWA e também contém o *favicon.ico*, arquivo de ícone que é responsável por mostrar uma imagem customizada na aba do navegador;

- src: pasta principal do aplicativo, que contém a lógica da visualização. Esta é subdividida em *assets*, *boot*, *components*, *css*, *i18n*, *layouts*, *pages*, *router*, *utils* e contem o arquivo *index.tempĺate.html*, que é o arquivo base *html* utilizado como padrão para a construção dos componentes:
  - assets: contém arquivos estáticos do sistema, normalmente imagens;
  - boot: contém arquivos de inicialização de *plugins*;
  - router: arquivos *Javascript* que são utilizados pelo plugin *VueRouter* para realizar o roteamento. O roteamento é mecanismo que navegadores utilizam para poderem renderizar diferentes páginas do sistema apenas trocando a URL, porém diferente da navegação tradicional, no *Vue* não há a necessidade de trocar a página inteira por outra página *HTML*. A troca de páginas é feita de forma automática pelo *Javascript*, apenas alterando parte do DOM (*Document Object Model*) da página. O DOM é a representação estruturada dos documentos *XML* em memória para serem acessados e modificados utilizando o *Javascript*;
  - pages: contém os componentes *Vue* que representam as páginas do sistema. Esses arquivos são importados e utilizados dentro da pasta *router*;
  - layouts: contém componentes *Vue* que são utilizados como "esqueleto" de páginas. Trazem os componentes que estão presentes em todas as páginas, como por exemplo: menus, barra de ferramentas, etc. Estes elementos contém uma *tag* chamada *router-view*, utilizada pelo *VueRouter* para identificar onde devem ser montadas as *pages*;
  - components: contém os componentes *Vue* que são utilizados em várias partes do sistema, por exemplo: *inputs*, *selects*, tabelas, etc;
  - i18n: contém arquivos de tradução utilizados na internacionalização do site, quando deseja-se que o sistema tenha suporte para mais do que um idioma;
  - utils: pasta criada para guardar arquivos *Javascript* extras, que auxiliam no desenvolvimento do sistema. No Sistema do SCAP foram utilizados dois arquivos. Em **api.js**, se encontram os interceptadores de requisições AJAX. São necessários para antes de enviar a requisição para o servidor, ou logo após receber a resposta do servidor realizar alguma ação. Dentre as ações utilizadas no SCAP, estão: mostrar alertas de erros da API, mensagens personalizadas de sucesso de algum procedimento e autenticação automática. Já **formData.js** é o arquivo responsável por transformar um objeto *Javascript* em um formulário *FormData*. Tal padrão de formulário é indicado para quando se deseja, além de texto, enviar arquivos para o servidor;
- quasar.conf.js: arquivo de configuração do quasar.

O Quasar segue a arquitetura do padrão MVVM, descrita na Seção 2.3.3.

4.3 Modelos FrameWeb

Nessa seção apresentamos os modelos propostos pelo método *FrameWeb* para o sistema desenvolvido neste trabalho. Estes modelos foram construídos durante a fase de projeto arquitetural do sistema e ajudam a guiar o projeto centralizando a informação de forma organizada para que não haja dúvida na implementação.

Por decisão de projeto, a linguagem utilizada na implementação foi o inglês, portanto todos os conceitos, comentários no código, métodos e atributos de classes foram traduzidos para o inglês.

4.3.1 Modelo de Entidades

A Figura 9 apresenta as classes do domínio que foram implementadas. Complementando o modelo, alguns atributos são enumerações e foram representados na Figura 10.

Esses modelos ajudam na implementação das *migrations* que contém a criação de todas as tabelas de banco de dados, assim como os atributos e chaves estrangeiras precisam ser criadas. Qualquer alteração nesse modelo impacta diretamente na criação de mais um ou mais arquivos de *migration* descrevendo em SQL a alteração que precisa ser feita.

4.3.2 Modelo de Persistência

No servidor, as classes responsáveis por realizar a persistência de dados pertencem à camada de repositórios. Essa camada provê métodos que serão utilizados juntamente com os *Models* para criar, atualizar, recuperar ou remover dados.

Na Figura 11 estão ilustradas todas as classes de repositórios que foram utilizadas na implementação.

O método *FrameWeb* sugere o uso de interfaces para baixo acoplamento entre os módulos porém, a linguagem utilizada, *JavaScript*, não dá suporte para criação de interfaces. Portanto as interfaces não foram implementadas e por esse motivo não estão sendo representadas no modelo.

Os métodos chamados `all` são os métodos que recuperam uma listagem de elementos de uma certa tabela, os parâmetros deste método indicam filtros desta listagem. Por exemplo: em `FeedbackRepository` temos um método `all` com o parâmetro `professorId`, este método recupera todos os pareceres de um determinado professor.

Os métodos `create`, criam um novo registro na tabela com o conteúdo que é passado pelo parâmetro `fields`, que é um objeto com o valor das colunas da tabela que deseja criar

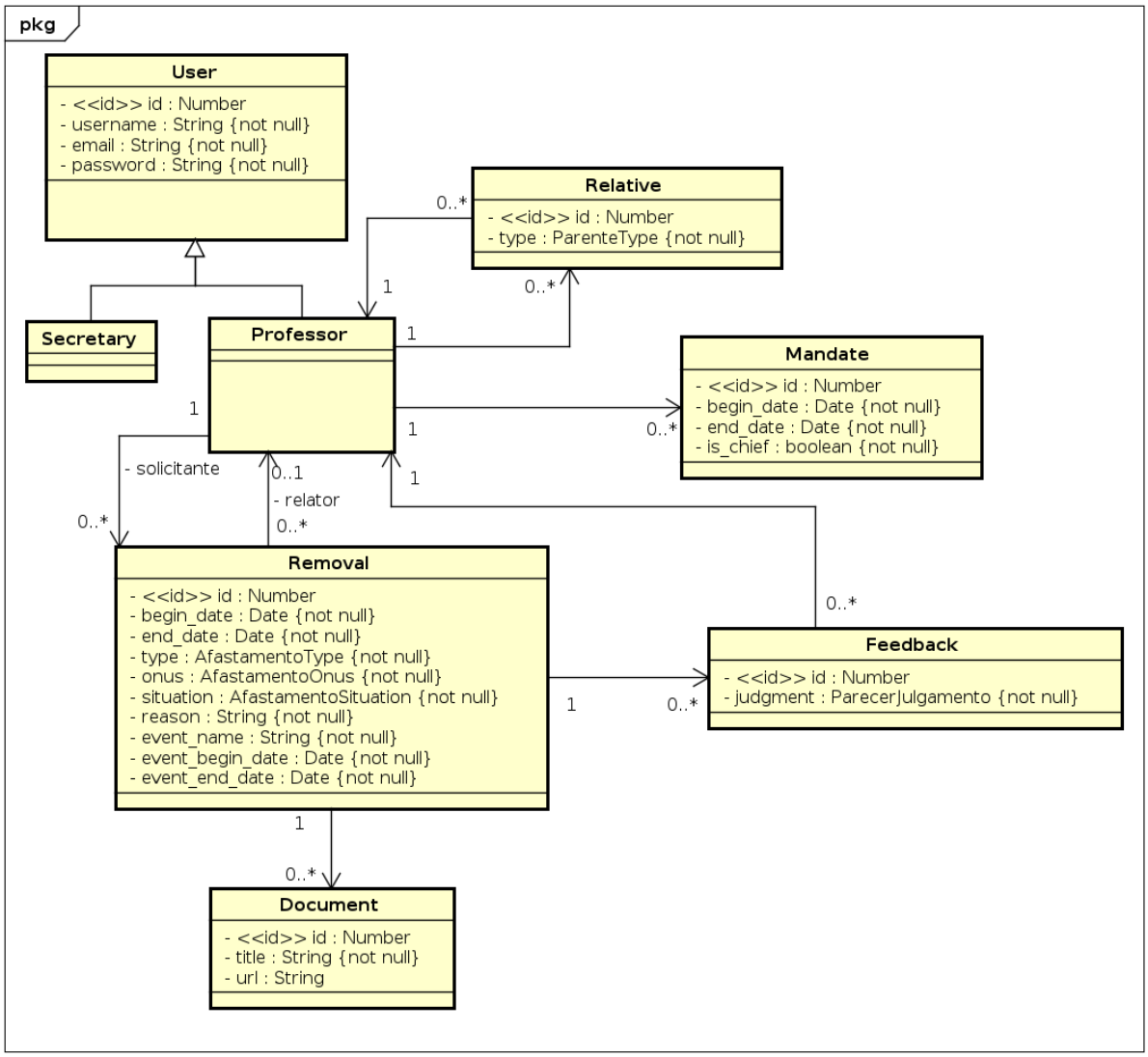


Figura 9 – Modelo de Entidades do SCAP.

um novo registro. Os métodos update atualizam um certo registro da tabela, assim como em create, necessitam de um objeto com o valor das colunas que se deseja alterar, e o segundo parâmetro é o identificador da linha da tabela que se deseja alterar.

Os métodos removeExceptIds, removem registros de uma tabela, porém ignorando algumas linhas que estão presentes no parâmetro ids. Por Exemplo: Em MandateRepository, o método RemoveExceptIds, remove todos os mandatos de algum professor mantendo apenas aqueles que estão na coleção ids.

Os métodos updateOrCreateMany são responsáveis por criar ou atualizar vários registros. É passada uma coleção de objetos, caso esse objeto contenha o identificador, significa que precisa ser atualizado. Caso não tenha identificador ele então é criado como um novo registro na tabela.

Os Métodos findById ou findByEmail são utilizados para encontrar um único registro

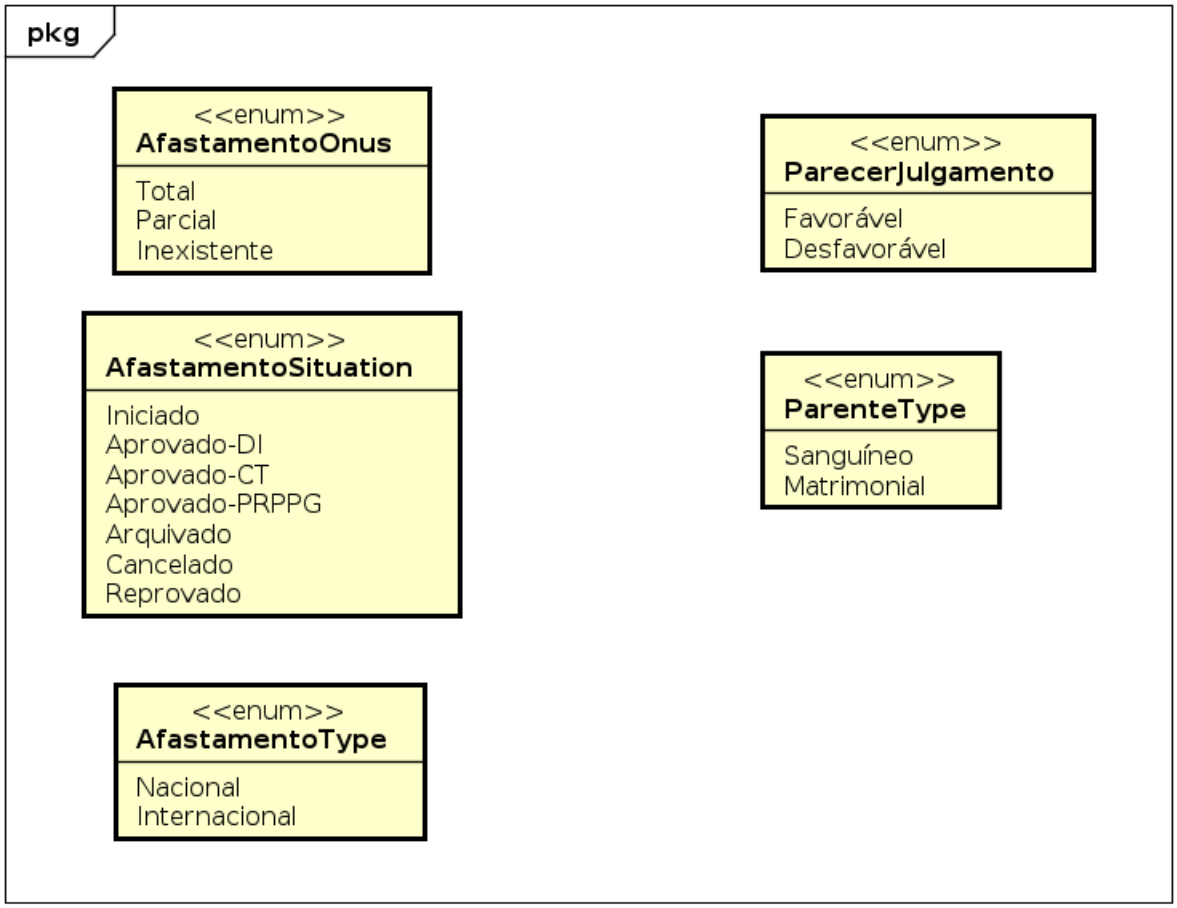


Figura 10 – Tipos enumerados do SCAP

na tabela de acordo com o parâmetro passado. Por exemplo: Em UserRepository, o método findById recupera um usuário pelo identificador.

4.3.3 Modelo de Navegação

Conforme explicado na Seção 2.4, o Modelo de Navegação de FrameWeb apresenta os componentes da camada de apresentação, como páginas Web, formulários HTML e controladores frontais do padrão *Front Controller* (vide Seção 2.3.2).

Tais elementos são muito utilizados em *frameworks* monolíticos, porém foi feita uma adaptação para representar a navegação utilizando um *framework* modular: no modelo, nas classes que contém o estereótipo "«page»" podem ser interpretados pelo *View-Model* que chamará uma requisição assíncrona para o servidor e as classes controladoras se referem às classes de Controle do servidor. Cada método público das classe do controle do servidor precisam ser chamados por uma rota, sendo que estas rotas não estão sendo representadas no modelo.

Utilizando o modelo de navegação proposto por Souza (2007), a representação dos dados proveniente dos formulários deveriam ser atributos na classe controladora, porém

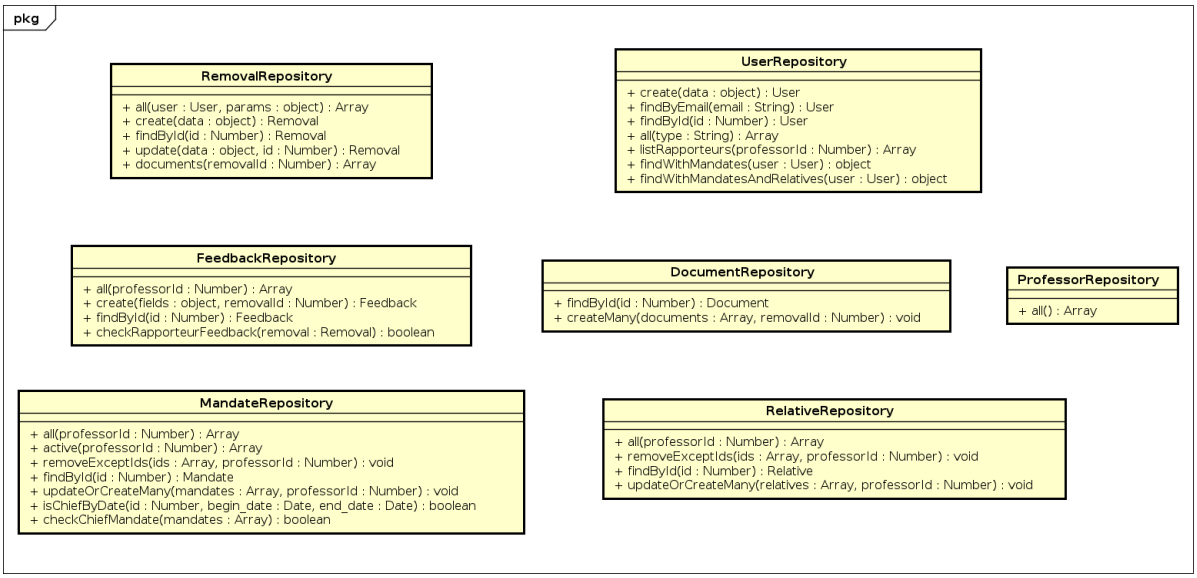


Figura 11 – Modelo de Persistência do SCAP

no *framework* AdonisJS os dados provenientes da requisição formam um objeto separado que contém os próprios métodos públicos para acesso de dados. Tal objeto é injetado no método de controle durante a execução da requisição. Este objeto está representado como parâmetro *data* dos métodos dos controladores.

Apesar de todo o escopo ter sido implementado, a apresentação e descrição de todos os possíveis modelos de navegação presentes no sistema seria impraticável, portanto foram escolhidos alguns casos de uso para ilustrar o funcionamento da navegabilidade da aplicação.

Como exemplos, apresentamos as figuras 12, 13, 14 e 15 referentes aos casos de uso **Solicitar Afastamento**, **Encaminhar Afastamento**, **Consultar Afastamento** e **Cadastrar Usuário** respectivamente.

No modelo de navegação do caso de Uso de solicitar afastamento, presente na Figura 12, o professor que estiver autenticado submeterá o formulário para o método *store* do controle. Os dados enviados do formulário serão enviados para o objeto *data* presente no método *store*. Caso não ocorra nenhum erro, serão registrados os dados do afastamento para futura análise e uma mensagem será enviada ao usuário confirmando a criação daquela solicitação. Porém, caso seja encontrado algum erro no servidor ou o usuário esteja sem autorização para executar a ação, é enviada uma mensagem de erro ao usuário e ele pode tentar submeter o formulário novamente.

Para encaminhar o afastamento para um relator, como descrito na Figura 13, o professor chefe do departamento entrará na página de atualizar um afastamento. Esta página exibirá uma lista de professores que podem ser utilizados como relator daquele afastamento. Esta listagem é obtida pelo método *listRapporteurs* e preenche os dados de

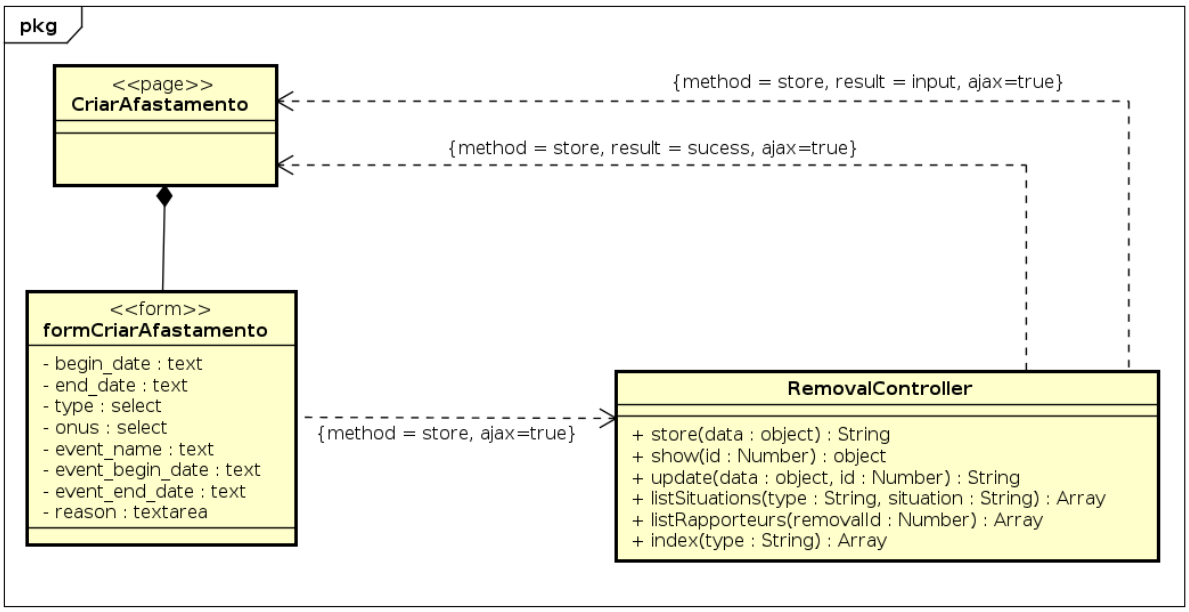


Figura 12 – Modelo de Navegação para criação de afastamentos.

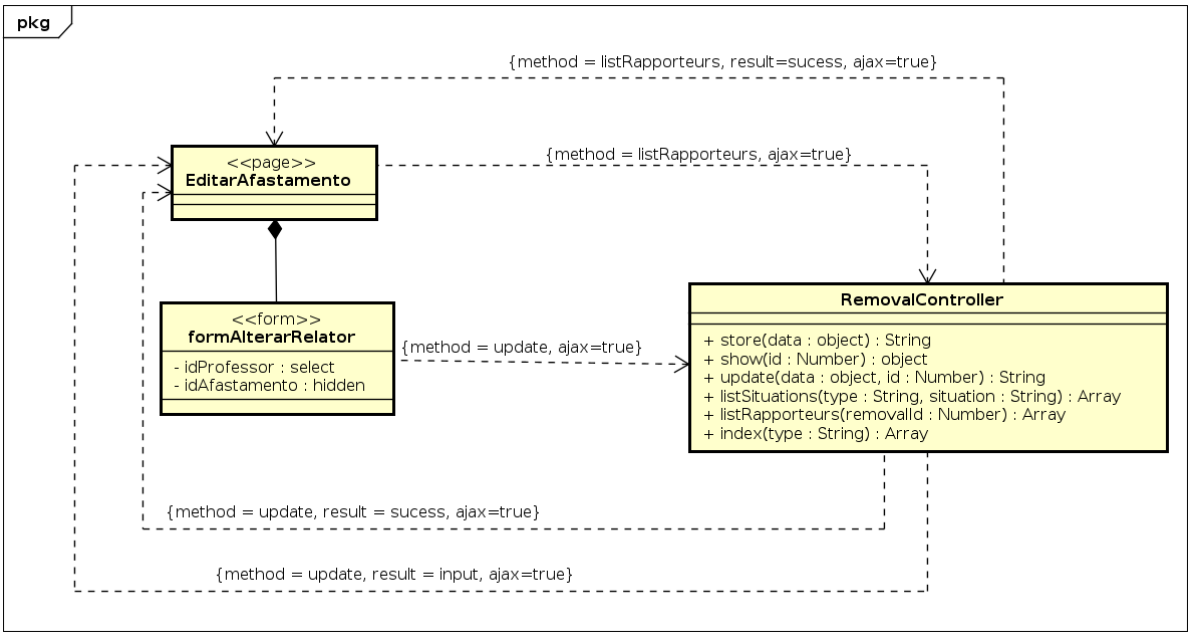


Figura 13 – Modelo de Navegação para Encaminhar Afastamento.

professores em um elemento *select* que será utilizado pelo usuário para fazer a escolha de apenas um dos professores para ser relator daquele afastamento.

Para poder atualizar um afastamento é necessário que ele seja encontrado em uma listagem de afastamentos. O modelo apresentado pela Figura 14 descreve como que o controlador envia os dados do afastamento para a página e eles são colocados em uma tabela de forma que ao clicar em alguma linha desta tabela é possível atualizar ou visualizar os dados do afastamento.

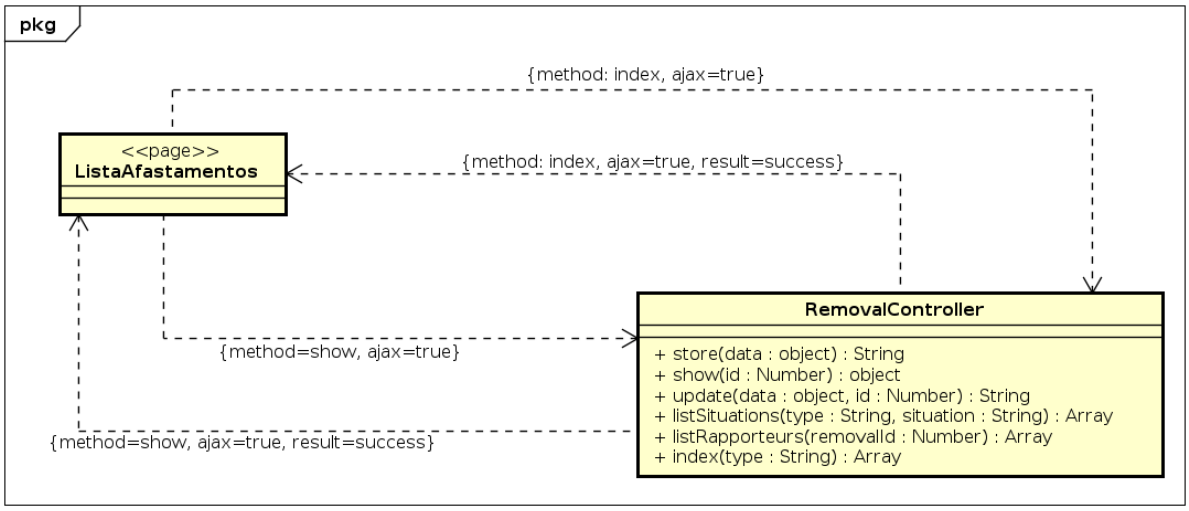


Figura 14 – Modelo de Navegação para Consultar Afastamento.

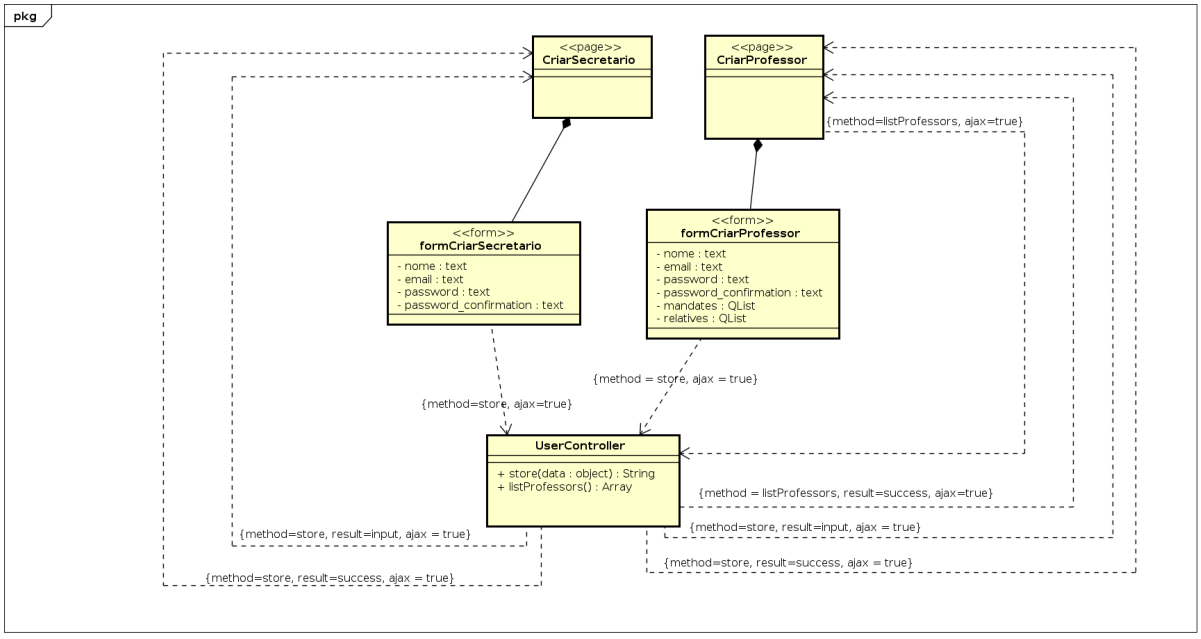


Figura 15 – Modelo de Navegação para Cadastrar Usuário.

O modelo para cadastro de usuários, como pode ser visualizado na Figura 15, mostra como secretários criam professores ou algum outro secretário. Para a criação de secretários é necessário apenas que submeta os dados de acesso para o servidor e o secretário já estará habilitado para entrar no sistema. Para a criação de professores é diferente: pode ser enviado os dados dos mandatos e também a das relações de parentesco no mesmo formulário. O componente *QList* é responsável por criar uma listagem na página com os dados dos mandatos, passando a data inicial e data final e ainda se aquele mandato é referente ao cargo de chefe de departamento. Pode ser utilizado também para listar os dados de parentes, no qual é feita a escolha de algum professor do sistema para ser parente desse novo professor criado.

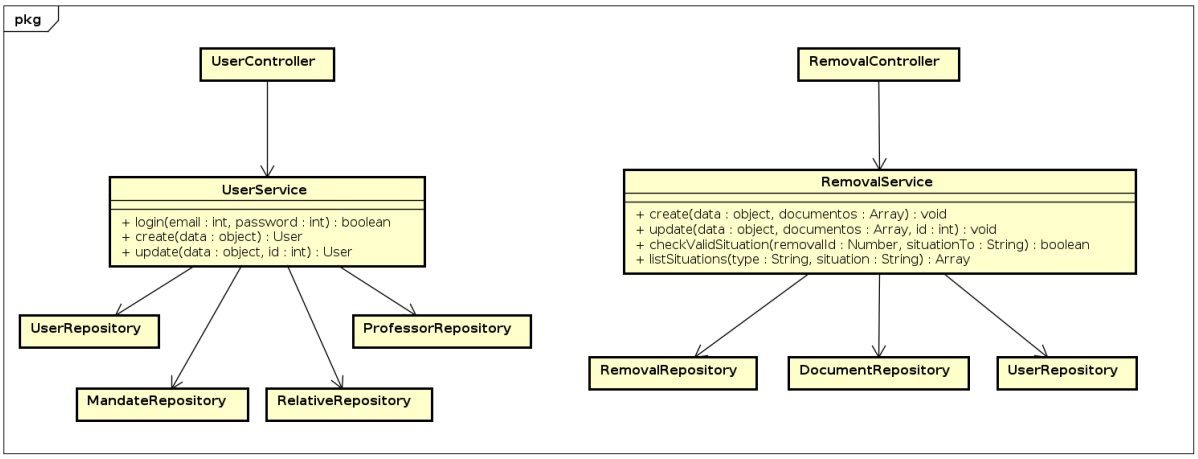
A listagem de professores é feita a partir do método *listProfessors*, esses dados serão utilizados para preencher o *select* para a escolha de parentes.

4.3.4 Modelo de Aplicação

A Figura 16 apresenta o modelo de aplicação do sistema. As classes de Controle (*UserController* e *RemovalController*) são responsáveis por receber as ações requisitadas pelo usuário e são enviadas para a camada de serviço (*UserService* e *RemovalService*) que de fato executa aquela ação utilizando as classes de repositórios necessárias para persistir os dados no banco de dados.

Pelo mesmo motivo descrito na Seção 4.3.2, não foram implementadas as classes de interface presentes na descrição original do modelo de aplicação, portanto os métodos são chamados diretamente das classes concretas e não utilizam-se de interfaces entre Controle e Serviço ou entre Serviço e Repositório.

Figura 16 – Modelo de Aplicação do SCAP



4.4 Apresentação de Resultados

Nesta seção será apresentada a aplicação web do SCAP que foi implementada, por meio de capturas de telas. Como esta aplicação também foi pensada para ser utilizada por um celular de forma parecida a uma aplicação nativa, além de capturas de telas do navegador, seguem visualizações de como a aplicação se comporta quando utilizado em telas menores, mostrando assim a responsividade das telas.

As imagens utilizadas no aplicativo foram baixadas por um site de imagens públicas chamado *undraw*[3] e a formatação das capturas de telas de celular foram feitas com ajuda

[3] <https://undraw.co/>

do site *appstorescreenshot*.[4]

A primeira página da aplicação pode ser vista na Figura 17. Nela é possível ver uma descrição para situar o usuário que ele está dentro do sistema SCAP e o botão "Entrar" na parte superior da tela. Esse botão faz a navegação para a tela de *Login*, que pode ser visualizada na Figura 18, e faz referência à parte de autenticação do usuário. O usuário seja ele professor ou secretário, informa o e-mail e a senha que foi previamente cadastrada por um outro secretário com acesso ao sistema. O usuário ainda pode clicar no ícone de olho para esconder ou mostrar a senha para ajudar no preenchimento do campo.

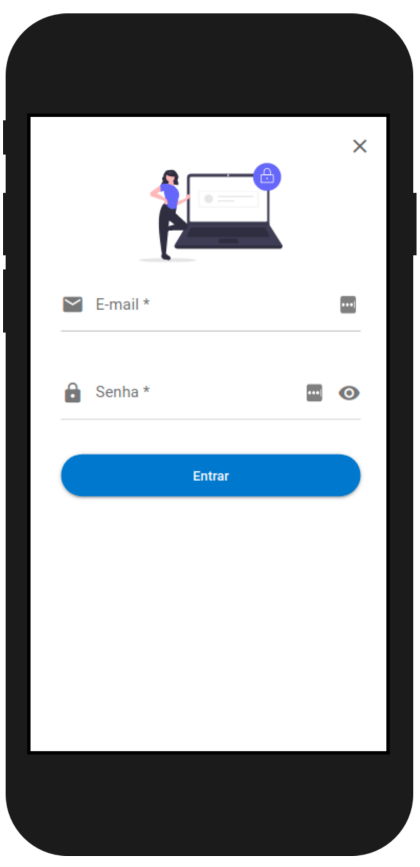
Figura 17 – Tela inicial pelo celular

Ao autenticar-se, o usuário é levado à pagina principal do sistema, que contém um menu lateral com uma listagem, onde cada item é referente a um módulo do sistema que o usuário tem acesso (Figura 19). Caso o usuário seja um professor, tem acesso aos módulos de Afastamento e Pareceres, se for secretário do DI tem acesso aos módulos de Professor, Secretário e Afastamento.

No canto superior direito é possível visualizar um botão com o ícone de uma seta apontando para baixo (Figura 20). Ao clicar neste botão é possível ver alguns dados pessoais e acessar a área de "Meus Dados", no qual torna possível a atualização do seus próprios dados, bem como a possibilidade de sair do sistema.

[4] <https://www.appstorescreenshot.com/>

Figura 18 – Tela de Login pelo celular



A página inicial de cada módulo é uma tabela que mostra uma listagem dos itens referentes àquele módulo. Estes itens podem ser filtrados por vários campos, como pode ser visto na Figura 21.

Quando o professor acessa o módulo de Afastamentos, como visto na Figura 22, ele tem a possibilidade de submeter os seus próprios pedidos de afastamentos, clicando no botão flutuante no canto inferior direito da tela, acompanhar os seus próprios afastamentos clicando na guia "Meus Afastamentos", acompanhar os afastamentos em que ele foi indicado como relator ("Relator"), assim como visualizar pedidos de afastamentos feitos por outros usuários ("Outros Afastamentos"), onde ele tem a possibilidade de emitir um parecer.

Para cada uma das opções que são mostradas na barra de guias, quando clicadas exibe uma tabela, como representado na Figura 21. Essas tabelas mostram alguns dados dos afastamentos e contém uma ultima coluna com as ações que podem ser tomadas para cada afastamento, estas ações podem ser:

- Ver documentos: representado na Figura 23, com a visualização de todos os documentos referente a um afastamento;
- Editar: exibição do fomulário de edição de um certo afastamento. Nesta tela (Figura 24) é possível adicionar documentos ou indicar um relator, caso o professor seja o chefe

Figura 19 – Tela com menu lateral pelo celular

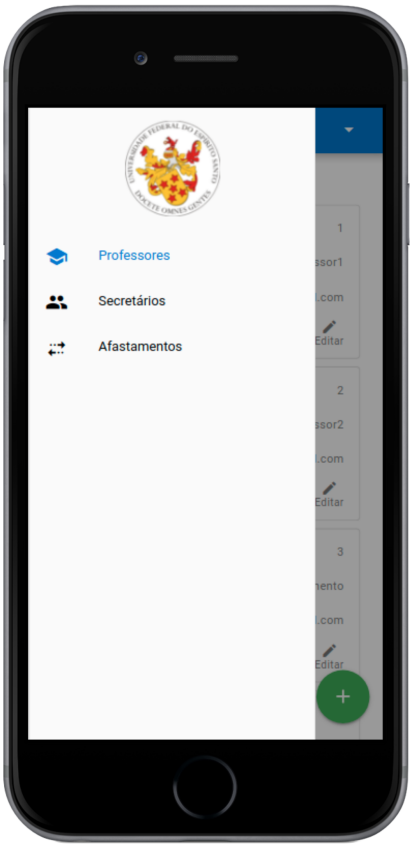


Figura 20 – Tela de Configuração

de departamento;

- Emitir Parecer: exibe um formulário para a criação de um parecer para um afastamento específico. Neste formulário é necessário que preencha o julgamento (Favorável ou Desfavorável) juntamente com o motivo. Este formulário pode ser visto na Figura 25;
- Cancelar: ao clicar nessa ação o professor desiste do prosseguimento do processo de afastamento, então o afastamento fica na situação de "Cancelado".

Quando selecionado no menu lateral a opção "Pareceres" (Figura 19), o usuário é levado para uma página que lista todos os pareceres que ele emitiu (Figura 26). Ao

Figura 21 – Tela de listagem de Afastamentos do Professor

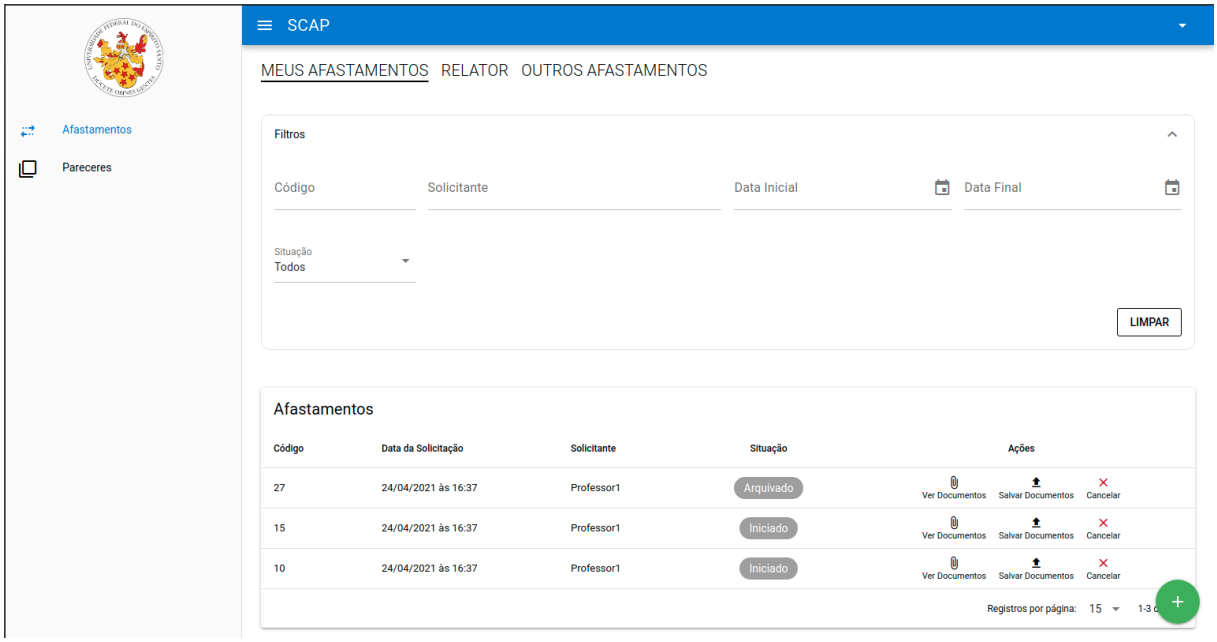
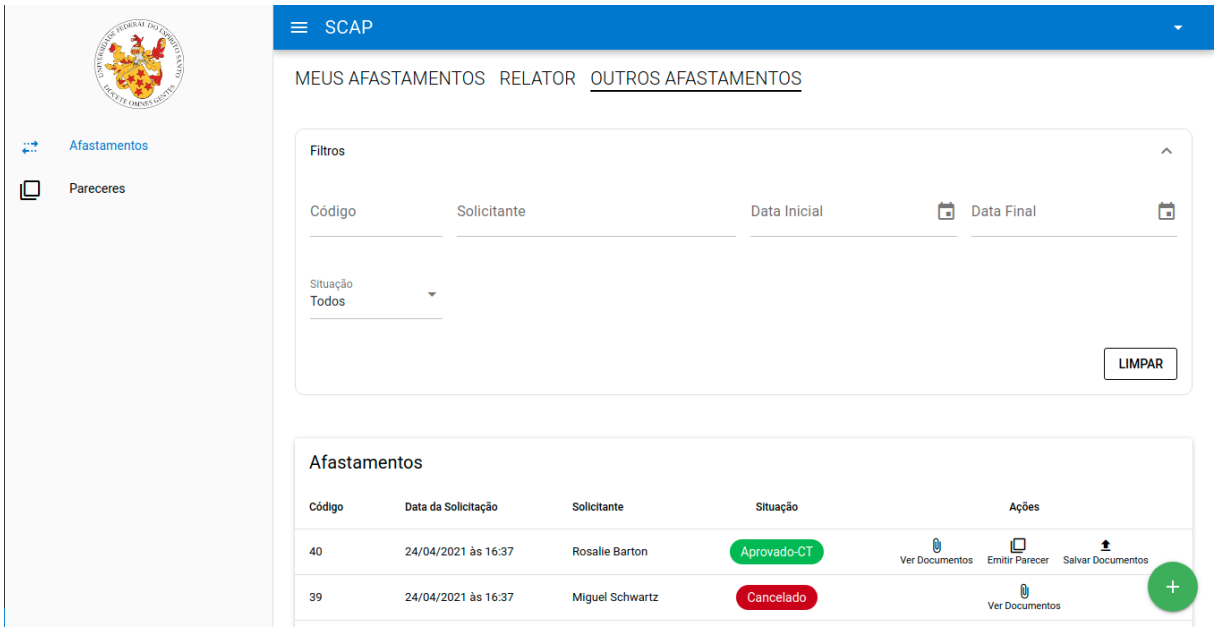


Figura 22 – Tela de listagem de Afastamento de outros Professores



clicar na ação “Ver Detalhes”, é levado a uma página que mostra o julgamento e o motivo daquele parecer.

Nos módulos referentes ao cadastro de novos usuários, assim como no caso de professores, o cadastro de mandatos e parentes são de responsabilidade do secretário do DI. Ao entrar no sistema e no módulo de professores, o secretário visualiza a tela representada pela Figura 32 ou, em tela de celular, pela Figura 33. Nesta tela o secretário tem a possibilidade de criar novos professores clicando no botão flutuante passando os

Figura 23 – Tela de listagem de documentos de um afastamento

dados pessoais (Figura 28), pelo celular a mesma tela anterior (Figura 29). A atualização dos mandatos e parentes de um professor é feita clicando na ação "Editar" da tabela.

A listagem de Secretários pode ser visualizada na Figura 31. A criação de secretários é feita de forma análoga à criação de professores, porém não é possível alterar nenhum dado após a criação. Os dados referentes à criação de secretários podem ser verificados na Figura 30.

O módulo de Afastamento para usuários secretários é composto pela parte de listagem, onde é possível ter acesso a todos os afastamentos criados no sistema, bem como às seguintes ações: ver documentos, envio de novos documentos e a alteração da situação do afastamento. Na Figura 27 é possível visualizar a listagem de afastamento para usuários da secretaria.

O código-fonte implementado neste trabalho está disponível em <https://bitbucket.org/vitorsouza-students/pg-2020-eduardo-dalapicola/src/master/code/>.

Figura 24 – Tela para indicar um relator

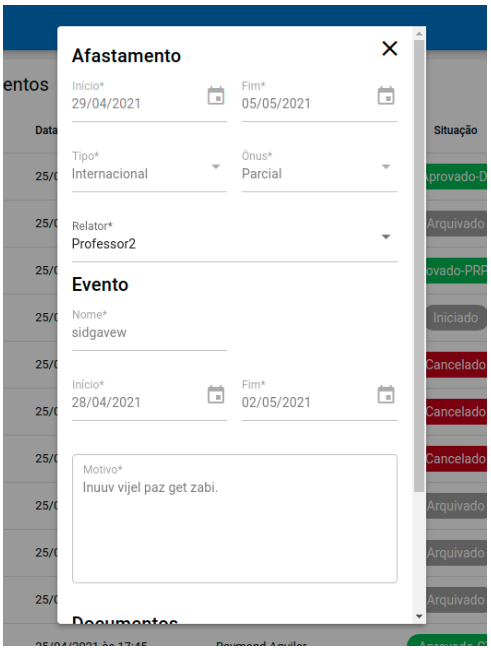


Figura 25 – Tela de emissão de Pareceres

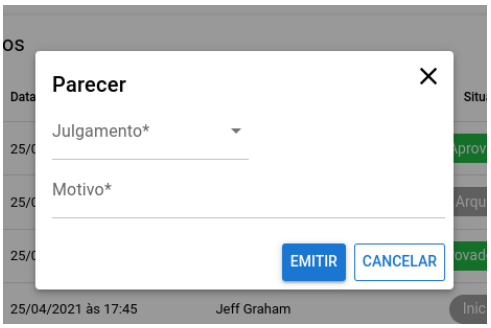


Figura 26 – Tela de listagem de Pareceres

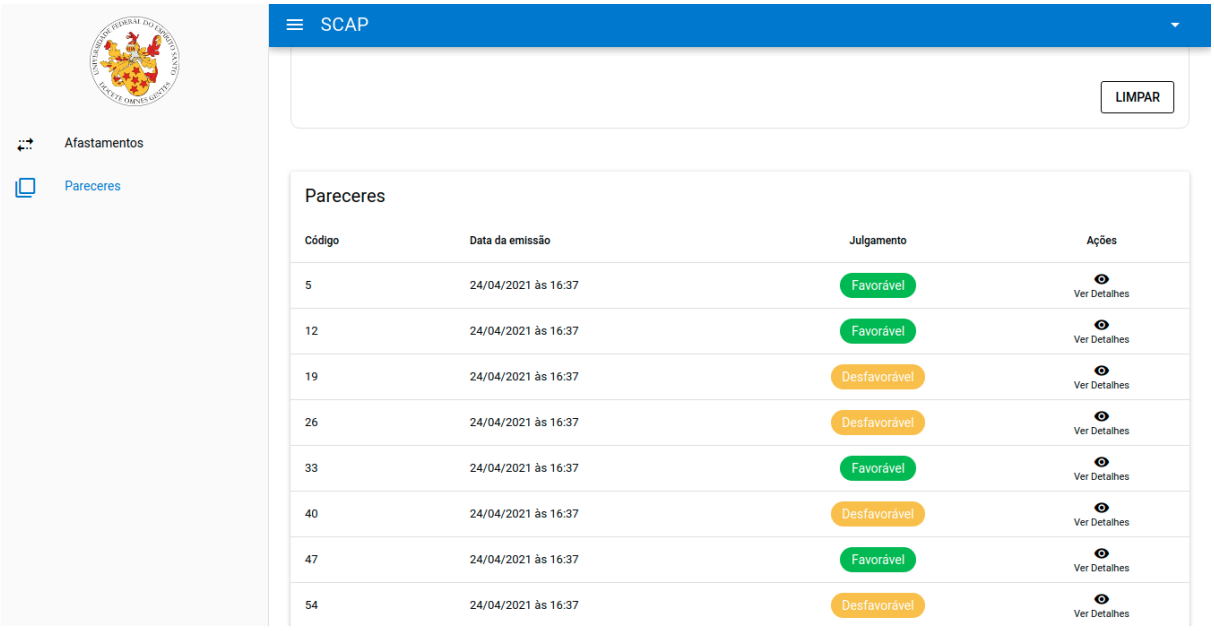


Figura 27 – Tela de listagem de Afastamentos

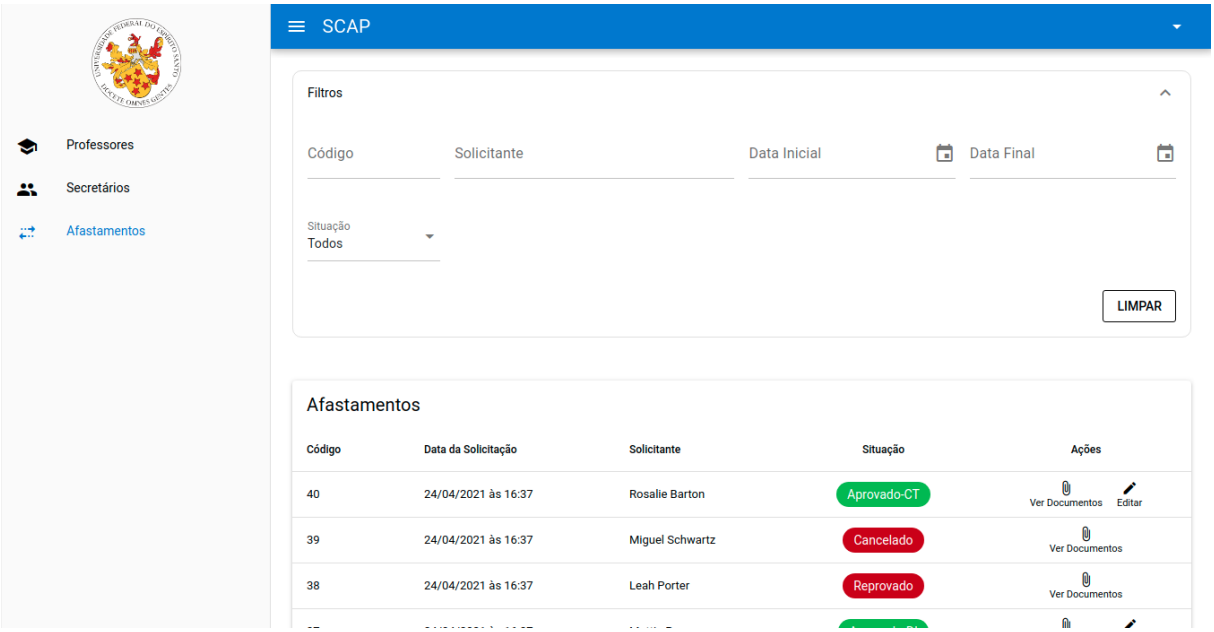


Figura 28 – Tela de criação de Professores

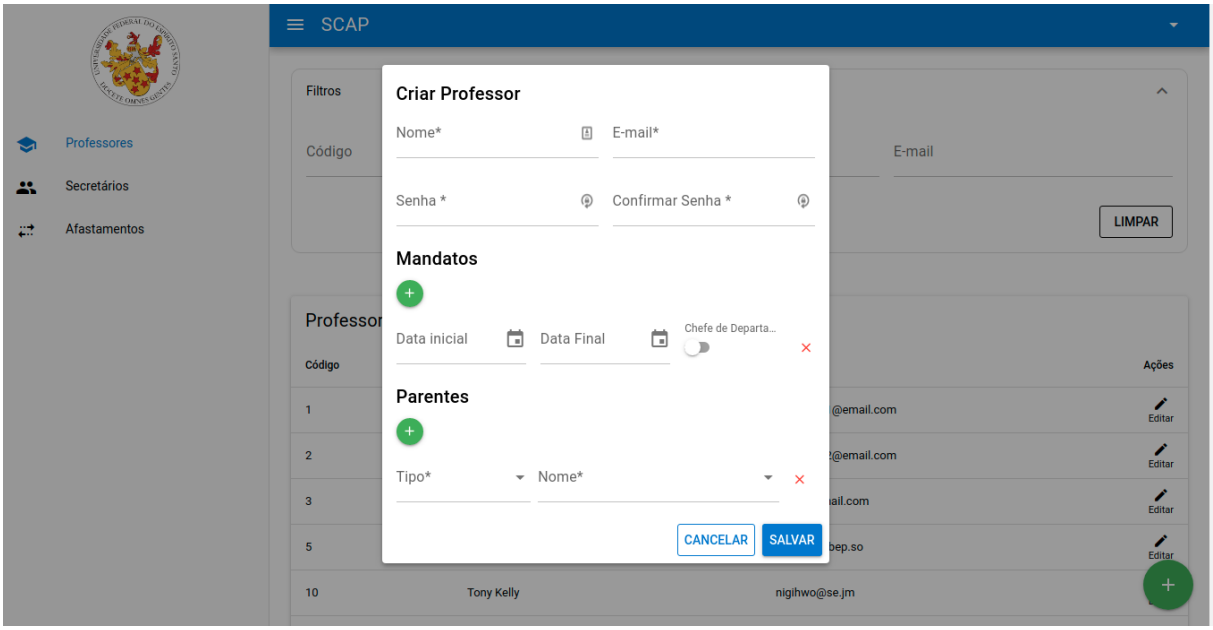


Figura 29 – Tela de criação de Professores pelo Celular

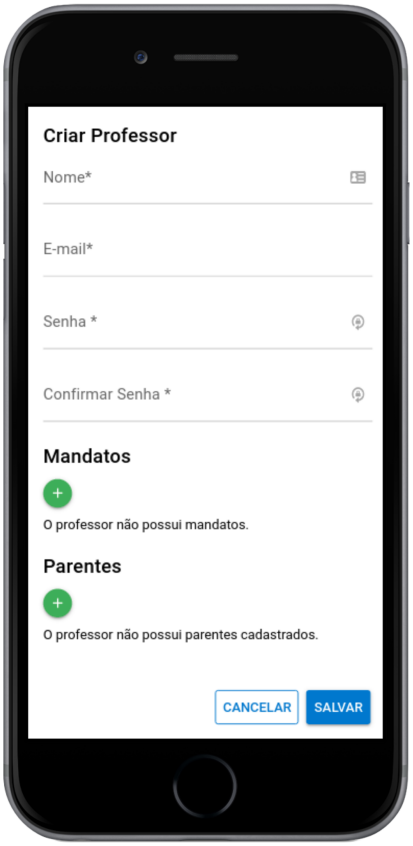


Figura 30 – Tela de criação de Secretários

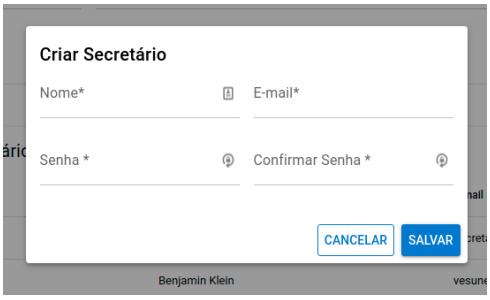


Figura 31 – Tela de listagem de Secretários

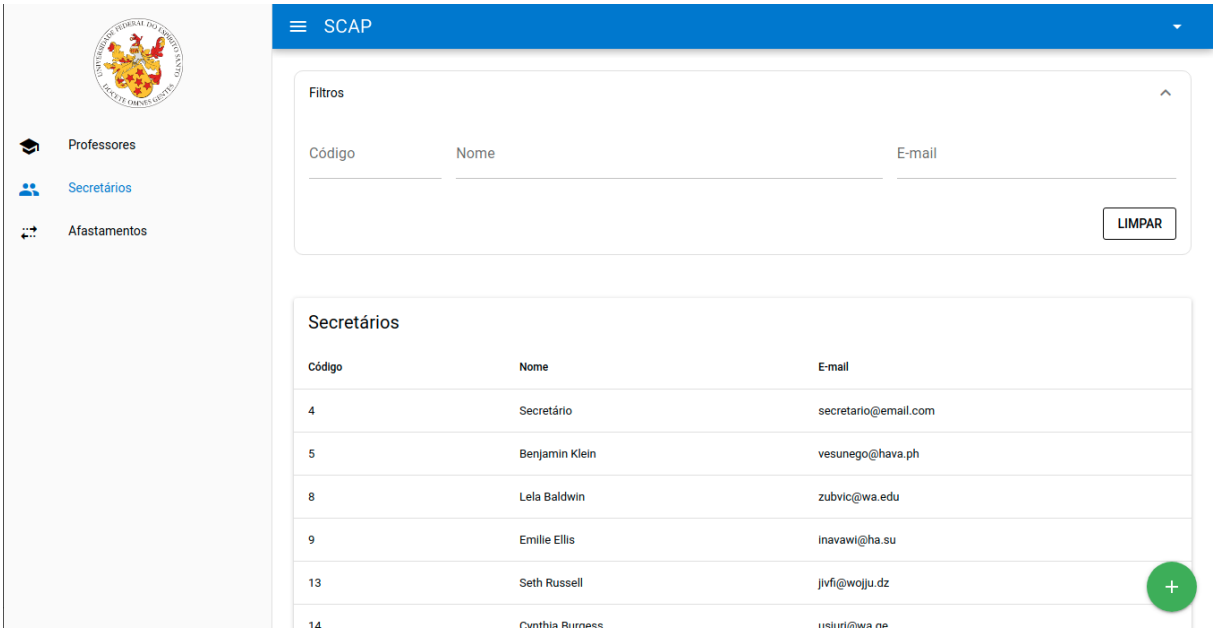


Figura 32 – Tela de listagem de Professores

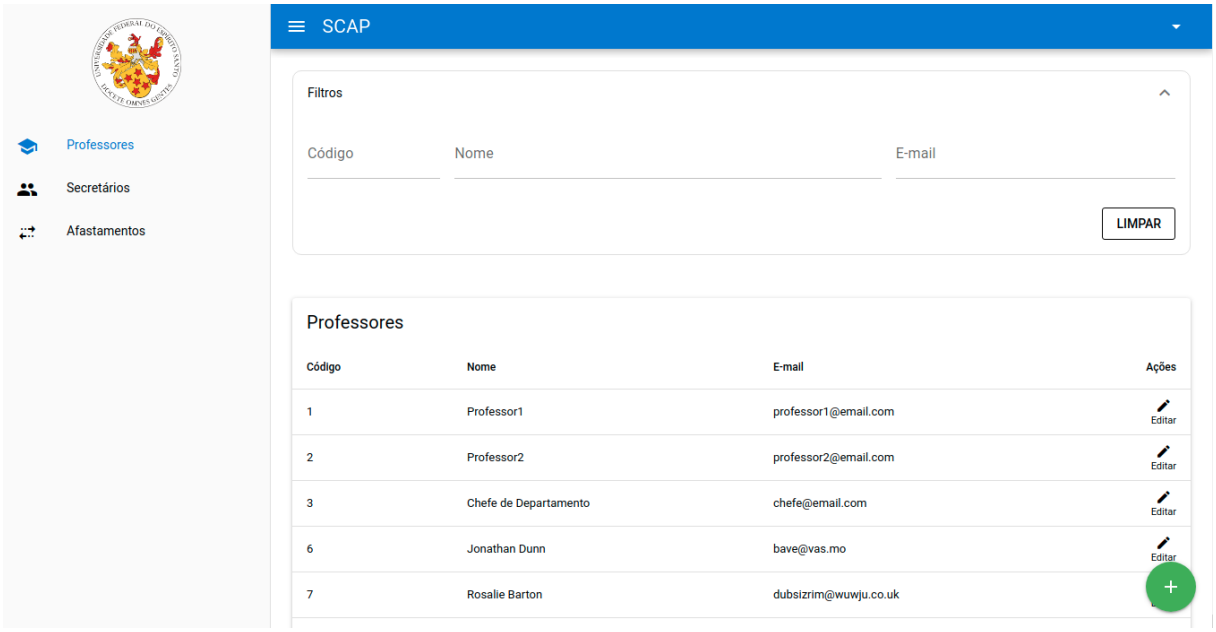
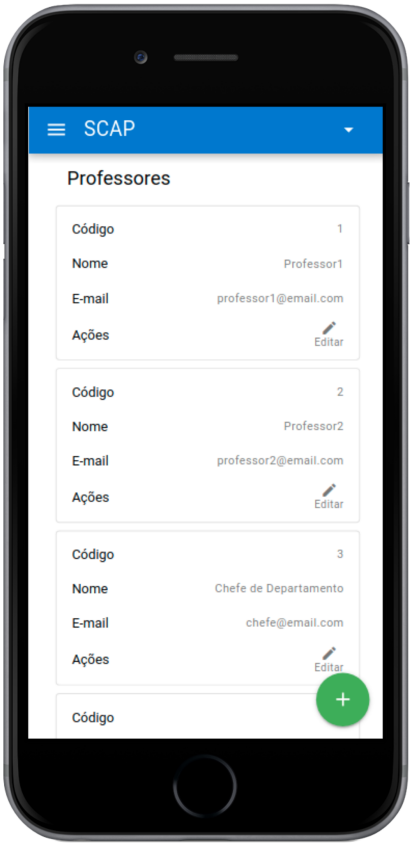


Figura 33 – Tela de listagem de Professores pelo Celular



5 Considerações Finais

As disciplinas de ligadas à Engenharia de Software do curso de Ciência da Computação ensinam várias técnicas relacionadas a planejamento, documentação, arquitetura de *software*, requisitos de *software*, análise e etc. Tais técnicas, muitas vezes são ignoradas quando o escopo do projeto é pequeno e quando a equipe de trabalho é pequena. Porém, por experiência própria, quando se trabalha com uma equipe de desenvolvimento maior e uma rotatividade alta de projetos é necessário que tenha uma forma fácil e rápida de documentar os conceitos do sistema para que toda a equipe possa visualizar.

Por vezes integrei projetos grandes em que não havia nenhuma documentação e os conceitos se perdiam por falhas de comunicação. O método *FrameWeb* propõe uma arquitetura e os modelos gerados no processo se apresentam como um bom modo de organizar as funcionalidades de um sistema, são simples e de fácil compreensão, podem ser utilizados como ferramentas para a instrução de novos membros de equipe e evitam centralizar o conhecimento em um grupo de pessoas e que as ideias sejam perdidas no tempo.

Por questões de organização do trabalho, a fase construção dos modelos do método *FrameWeb* foi conduzida após a implementação de todo o sistema. A construção dos modelos ajudaram na identificação de algumas incoerências: métodos que eram da camada de persistência estavam na camada de serviço, funções que não respeitavam o isolamento das camadas, métodos que continham funcionalidades parecidas e poderiam ser fundidos em apenas um método, métodos duplicados, etc. Ao implementar um projeto Web estas incoerências passam despercebidas, porém se tornam claras quando é feita a visualização dos modelos passaram a contribuir para a refatoração de código e reorganização das ideias.

Por conter camadas bem definidas e de baixo acoplamento sugeridas pelo método, o sistema comporta mudanças de forma muito mais adequada do que se fosse desenvolvido sem nenhum padrão. Um exemplo disso seria se por algum motivo a tecnologia de banco de dados fosse mudada em alguma parte do projeto, ou ainda, se fosse necessário criar um segundo tipo de visualização (Aplicação Mobile ou *Web Service*). Em todos os exemplos é possível trocar apenas uma camada da aplicação e, ainda assim, não afetariam as outras camadas.

Apesar do método se apresentar como uma boa forma de arquitetar o projeto, algumas dificuldades foram encontradas na hora de representar alguns elementos. Nos modelos de aplicação e persistência houve a dificuldade pois *JavaScript* não implementa interfaces de forma nativa sendo muito complexa a implementação deste elemento. Nos modelos de Navegação houve a necessidade de adaptação do modelo para poder representar

o padrão *REST*. Para trabalhos futuros que utilizassem o padrão *REST* seria interessante a introdução de novos elementos gráficos que representassem melhor a dinâmica de uma requisição HTTP e da resposta. Como exemplo de novos elementos podem ser citados:

- Tipo de requisição: *GET*, *POST*, *PUT*, *DELETE* que indicam se a ação da requisição é a recuperação de dados, criar novos dados, atualização ou exclusão;
- Classe de tipo da resposta: além de respostas do tipo HTML ou arquivos, poderiam ser introduzidas respostas de tipos estruturados como *JSON* que é amplamente utilizado;
- Campo adicional para informar quais requisições necessitam de autenticação e quais são de livre acesso.

Apesar deste sistema não possuir nenhuma comunicação com serviços externos, a maioria das aplicações hoje em dia faz uso de várias outras aplicações externas que estão presente na Internet e proveem serviços ao sistema. Um trabalho futuro poderia incrementar os modelos com uma forma de representar a ligação com serviços externos.

Um outro trabalho futuro seria a compilação de todos os resultados de trabalhos que utilizaram o método *FrameWeb* para uma melhor comparação da adequação do método em diferentes cenários de linguagem de programação, *frameworks* e plataformas de forma que seja possível visualizar o *FrameWeb* na prática.

O método *FrameWeb* possui um editor gráfico, Editor FrameWeb (CAMPOS, 2017), foram feitas tentativas para utilizá-lo como ferramenta na construção dos modelos, porém dificuldades na usabilidade do editor como: dificuldade na alteração dos nomes das classes, identificar associação entre classes, as mudanças das propriedades dos modelos não eram intuitivas e em algumas telas não tinha o salvamento automático do progresso. Todas estas dificuldades inviabilizaram que fosse utilizado na implementação do sistema deste trabalho. No seu lugar foi utilizado o *astah*,[1] que é um editor que suporta o padrão UML (*Unified Modeling Language*).

[1] <https://astah.net/>

Referências

ALUR, D.; CRUPI, J.; MALKS, D. *Core J2EE patterns: best practices and design strategies*. [S.l.]: Prentice Hall Professional, 2003. Citado na página 27.

AVELAR, R. d. A. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework Ninja*. [S.l.], 2018. Vitória, ES, Brasil. Citado 3 vezes nas páginas 15, 33 e 34.

BAUER, C.; KING, G. *Hibernate in action*. [S.l.]: Manning Greenwich CT, 2005. v. 1. Citado na página 24.

CAMPOS, S. L. *FrameWeb Editor: Uma Ferramenta CASE para suporte ao Método FrameWeb*. Vitória, ES, Brasil, 2017. Citado na página 58.

CONALLEN, J. *Engenharia de Software. 8 ed.* [S.l.]: Addison Wesley Professional, 2002. ISBN 9780201730388. Citado 2 vezes nas páginas 20 e 27.

DUARTE, B. B. *Aplicação do Método FrameWeb no Desenvolvimento de um Sistema de Informação na Plataforma Java EE 7*. [S.l.], 2014. Vitória, ES, Brasil. Citado 8 vezes nas páginas 14, 15, 28, 29, 31, 32, 33 e 34.

FALBO, R. de A. *Projeto de Sistema de Sofware*. 2011. Notas de Aula. Citado na página 18.

FERREIRA, M. T. *Aplicação do método FrameWeb no desenvolvimento do sistema SCAP utilizando os frameworks Wicket e Tapestry*. [S.l.], 2018. Vitória, ES, Brasil. Citado 3 vezes nas páginas 15, 33 e 34.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. [S.l.]: University of California, Irvine Irvine, 2000. v. 7. Citado na página 34.

FOWLER, M. *Patterns of Enterprise Application Architecture*. [S.l.]: Addison-Wesley Professional, 2002. ISBN 0321127420. Citado 2 vezes nas páginas 9 e 26.

FOWLER, M. *Presentation Models*. 2004. Disponível em: <https://martinfowler.com/eaaDev/PresentationModel.html>. Citado na página 22.

GUTERRES, C. S. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando o framework Play*. Vitória, ES, Brasil, 2019. Citado 3 vezes nas páginas 33, 34 e 35.

HUSTED, T. e. a. *Struts em Ação. 1 ed.* [S.l.]: Editora Moderna, 2004. ISBN 8573932996. Citado na página 20.

KOCH, N. et al. Extending uml to model navigation and presentation in web applications. In: CITESEER. *Proceedings of Modelling Web Applications in the UML Workshop*. [S.l.], 2000. Citado na página 27.

MATOS, R. P. de. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação usando os frameworks Spring MVC e Vaadin*. Vitória, ES, Brasil, 2017. Citado 3 vezes nas páginas 33, 34 e 35.

MEIRELLES, L. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando os frameworks Codeigniter e NodeJS*. [S.l.], 2019. Vitória, ES, Brasil. Citado 4 vezes nas páginas 15, 33, 34 e 35.

MURUGESAN, S.; DESHPANDE, Y. *Web engineering: managing diversity and complexity of web application development*. [S.l.]: Springer Science & Business Media, 2001. v. 2016. Citado na página 14.

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 18.

OLSINA, L.; LAFUENTE, G.; ROSSI, G. Specifying quality characteristics and attributes for websites. In: *Web Engineering*. [S.l.]: Springer, 2001. p. 266–278. Citado na página 19.

PASTOR, O.; FONS, J.; PELECHANO, V. Oows: A method to develop web applications from web-oriented conceptual models. In: *International Workshop on Web Oriented Software Technology (IWWOST)*. [S.l.: s.n.], 2003. v. 65, p. 70. Citado na página 20.

PERUCH, L. A. *Aplicação e Análise do Método FrameWeb com Diferentes Frameworks Web*. 2007. Vitória, ES, Brasil. Citado na página 19.

PINHEIRO, F. G. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando um framework PHP e um framework JavaScript*. Vitória, ES, Brasil, 2017. Citado 3 vezes nas páginas 33, 34 e 35.

PRADO, R. C. d. *Aplicação do método FrameWeb no desenvolvimento de um sistema de informação utilizando o framework VRaptor 4*. [S.l.], 2015. Vitória, ES, Brasil. Citado 10 vezes nas páginas 9, 15, 27, 28, 29, 30, 31, 32, 33 e 34.

PRESSMAN, R. S. *Software engineering: a practitioner's approach*. [S.l.]: Palgrave macmillan, 2005. Citado 2 vezes nas páginas 18 e 20.

PRESSMAN, R. S. *Engenharia de Software - Uma Abordagem Profissional 7.ed*. [S.l.]: McGraw-Hill, 2011. ISBN 9788563308337. Citado 3 vezes nas páginas 14, 17 e 18.

PROGRAMMINGHELP. *Fundamentals of an MVC Framework*. 2013. Disponível em: <http://www.programminghelp.com/mvc/fundamentals-mvc-framework>. Citado 2 vezes nas páginas 9 e 22.

REENSKAUG, T. M. H. The original mvc reports. 1979. Citado na página 21.

SILVA, R. *UML 2 - Modelagem Orientada a objetos*. Visual Books, 2007. ISBN 9788575022054. Disponível em: <https://books.google.com.br/books?id=WBxhvgAACAAJ>. Citado na página 17.

SMITH, J. Wpf apps with the model-view-viewmodel design pattern. *MSDN Magazine. Disponível em: http://msdn. microsoft. com/enus/magazine/dd419663.aspx.*, v. 9, 2009. Citado na página 22.

SOMMERVILLE, I. *Building Web Applications with UML. 2*. [S.l.]: Pearson–Addison Wesley, 2007. ISBN 9788588639287. Citado na página 19.

SOUZA, B. F. M. Evolução do método frameweb para o projeto de sistemas de informação web utilizando uma abordagem dirigida a modelos. 2016. Vitória, ES, Brasil. Citado na página 27.

SOUZA, V. E. S. *FrameWeb: um Método baseado em Frameworks para o Projeto de Sistemas de Informação Web*. Dissertação (Mestrado) — Universidade Federal do Espírito Santo, 2007. Citado 8 vezes nas páginas 9, 14, 20, 21, 24, 25, 27 e 42.

SOUZA, V. E. S. The FrameWeb Approach to Web Engineering: Past, Present and Future. In: ALMEIDA, J. P. A.; GUIZZARDI, G. (Ed.). *Engineering Ontologies and Ontologies for Engineering*. 1. ed. Vitória, ES, Brazil: NEMO, 2020. cap. 8, p. 100–124. ISBN 9781393963035. Disponível em: <http://purl.org/nemo/celebratingfalbo>. Citado na página 25.