UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

FÁBIO JÚNIOR LAMPUGNANI

SISTEMA WEB PARA CONTROLE DE PROCESSOS JURÍDICOS

MONOGRAFIA DE ESPECIALIZAÇÃO

PATO BRANCO
2017

FÁBIO JÚNIOR LAMPUGNANI

SISTEMA WEB PARA CONTROLE DE PROCESSOS JURÍDICOS

Monografia de Conclusão de Curso, apresentada ao Curso de Especialização em Tecnologia Java, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Especialista.

Orientadora: Profª. Beatriz Terezinha Borsoi

PATO BRANCO
2017

MINISTÉRIO DA EDUCAÇÃO
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco
Departamento Acadêmico de Informática
Curso de Especialização em Tecnologia Java

UTFPR

TERMO DE APROVAÇÃO

SISTEMA WEB PARA CONTROLE DE PROCESSOS JURÍDICOS

por

FÁBIO JÚNIOR LAMPUGNANI

Este trabalho de conclusão de curso foi apresentado em 17 de novembro de 2017, como requisito parcial para a obtenção do título de Especialista em Tecnologia Java. Após a apresentação o candidato foi arguido pela banca examinadora composta pelos professores Andreia Scariot Beulke e Vinicius Pegorini. Em seguida foi realizada a deliberação pela banca examinadora que considerou o trabalho aprovado.

Beatriz Terezinha Borsoi
Prof. Orientador (UTFPR)

Andreia Scariot Beulke
Banca (UTFPR)

Vinicius Pegorini
Banca (UTFPR)

Robison Cris Brito
Coordenador da IV Especialização em Tecnologia Java

A Folha de Aprovação assinada encontra-se na Coordenação do Curso.

RESUMO

LAMPUGNANI, Fábio Júnior. Sistema web para controle de processos jurídicos. 2017. 40f. Monografia (Trabalho de Conclusão de Curso) - Curso de Especialização em Tecnologia Java, Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

Os profissionais da área de direito trabalham com documentos denominados processos judiciais, sejam eles civis, criminais ou de outra natureza. Os processos definem as ações e as defesas impetradas por esses profissionais em favor dos seus clientes, que são as pessoas físicas ou jurídicas que eles representam perante um juizado. Aos processos são vinculados documentos como comprovantes ou partes em resposta às ações pelas quais esses processos passam no seu trâmite judicial. Os processos seguem um ciclo que inclui, entre outros, reuniões com os clientes a quem o advogado representa ou a quem o processo está vinculado, e reuniões com pessoas do judiciário, as chamadas audiências, e a juntada de documentos. O controle desse ciclo pelo qual os processos passam pode ser auxiliado por um aplicativo computacional. Assim, o advogado pode mais efetivamente gerenciar sua agenda. Por meio deste trabalho é apresentado um sistema que foi desenvolvido visando auxiliar os profissionais da área de direito no gerenciamento da sua agenda e das atividades relacionadas aos processos. Como principais tecnologias envolvidas no desenvolvimento do sistema que é *web* estão PrimeFaces, JavaServer Faces, Spring Security, a linguagem Java e o banco de dados MySQL.

Palavras-chave: PrimeFaces. JavaServer Faces. Spring Security.

ABSTRACT

LAMPUGNANI, Fábio Júnior. Web system to manage legal processes. 2017. 40f. Monografia (Trabalho de Conclusão de Curso) - Curso de Especialização em Tecnologia Java, Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Pato Branco, 2017.

Law professionals work with documents such as civil, criminal, and other legal proceedings. The juridical processes define actions filed by these professionals in favor of their clients who are individuals or legal entities that they represent before a court. These processes are accompanied by documents. And they follow a cycle that includes, among others, meetings with the clients whom the lawyer represents to whom the process is linked and meetings with people in the judiciary, the called hearings and the gathering of documents. The control of this cycle can be managed by a software aiming to help in the control of the professional's agenda. In this work, it is proposed a web system to assist these professionals in the management of their agenda and the activities related to the processes. The main technologies involved in the development of the system are PrimeFaces, JavaServer Faces, Spring Security, the Java language and the MySQL database.

Keywords: PrimeFaces. JavaServer Faces. Spring Security.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGOS

LISTA DE SIGLAS

CDI	*Context and Dependency Injection*
HTML	*Hypertext Markup Language*
IDE	*Integrated Development Environment*
JPA	*Java Persistence API*
JSF	*JavaServer Faces*
JVM	*Java Virtual Machine*
LDAP	*Lightweight Directory Access Protocol*
OAB	Ordem dos Advogados do Brasil
POM	*Project Object Model*
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	*Structured Query Language*
URL	*Uniform Resource Locator*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Entre as diversas possibilidades de atuação em uma carreira advocatícia estão a de concursos, abertura de um escritório próprio, o compartilhamento de escritório quando advogados de uma mesma área, de áreas complementares ou de áreas distintas compartilham em um mesmo ambiente, a atuação em empresa e a atuação como profissional autônomo.

Se o profissional atuar de maneira autônoma ou em escritório próprio ou com terceiros, buscando, assim, seu espaço no mercado de trabalho, é de suma importância que ele tenha organização e diligência na realização das suas atividades, visando, assim, destaque no campo do trabalho advocatício e realizar um bom trabalho. Cumprir prazos e evitar o retardamento de processos auxilia na excelência do serviço prestado, tornando-o mais eficaz e produtivo.

Os profissionais da área de direito podem utilizar diversas ferramentas ou tecnologias para auxiliar no gerenciamento de processos, de prazos e para facilitar a comunicação com clientes, tribunais e demais instituições. Recursos como *e-mail*, telefone, sistemas de troca de mensagens eletrônicas e correspondência são bastante utilizados. Esses meios, mesmo que oferecendo um suporte considerado adequado, dificultam, muitas vezes, que o trabalho seja realizado com agilidade.

Considerando esse contexto, este trabalho tem como objetivo apresentar uma solução de software que foi desenvolvida para integrar as funções de várias tecnologias, concentrando-as em uma única ferramenta. Sendo assim, o foco deste projeto é facilitar a prática profissional pela otimização do tempo, aumento de produtividade e legibilidade nos meios de gerenciamento de processos jurídicos.

Os advogados autônomos são, em princípio, no contexto deste projeto, os usuários do sistema. Por serem autônomos, eles não precisam de aplicativos e sistemas complexos para gerenciar o trabalho nos seus escritórios. Embora o sistema forneça os recursos necessários para atendimento às funcionalidades essenciais desse tipo de negócio.

Desta forma, a intenção de um sistema *web* surge com o intuito de auxiliar no gerenciamento dos processos, contratos, prazos e agenda dos profissionais. Assim, o objetivo deste trabalho é desenvolver um sistema *web* utilizando a linguagem Java para auxiliar no gerenciamento dos processos jurídicos, contratos, prazos e agenda.

2 FERRAMENTAS, TECNOLOGIAS E PROCEDIMENTOS

A seguir são apresentadas as ferramentas e as tecnologias utilizadas na modelagem e na implementação do aplicativo desenvolvido como resultado deste trabalho.

2.1 FERRAMENTAS E TECNOLOGIAS

O Quadro 1 apresenta as ferramentas e tecnologias utilizadas para modelar e implementar o aplicativo.

Ferramenta / Tecnologia	Versão	Disponível em	Aplicação
StarUml	2.0	http://www.baixaki.com.br/download/staruml.htm	Modelagem do sistema.
Linguagem Java	JDK 1.8	http://www.oracle.com	Linguagem para desenvolvimento da aplicação.
Eclipse Neon		https://www.eclipse.org/neon/	Ambiente para desenvolvimento da aplicação.
JavaServer Faces (JSF)	2.2.13	http://javaserverfaces.java.net/	Especificação Java para a construção de interfaces de usuário baseadas em componentes para aplicações web.
Spring Security	4.1.3.	https://spring.io	Framework que possui recursos avançados e de simples configuração para aspectos como a segurança da aplicação.
PrimeFaces	6.0	https://primefaces.org/	*Framework* de para projetos JavaServer Faces.
MySQL	5.6	https://www.mysql.com/	Banco de dados da aplicação.

Quadro 1 – Ferramentas e tecnologias utilizadas

A seguir são apresentadas sucintamente as tecnologias constantes no Quadro 1.

a) StarUml

A ferramenta StarUML 2 é compatível com os padrões da UML2 fornecendo suporte aos onze tipos de diagramas dessa versão da UML que são: classes, objetos, casos de uso, componente, implantação, estrutura composta, sequência, comunicação, estados, atividade e perfil (STARUML, 2017).

b) Linguagem Java

Java é uma linguagem de programação interpretada e orientada a objetos. O código escrito na linguagem Java é compilado para *bytecode* que é interpretado por uma máquina virtual, a *Java Virtual Machine* (JVM) (ORACLE, 2017) .

c) Eclipse Néon

Ferramenta ou *Integrated Development Environment* (IDE) para o desenvolvimento de aplicações. No caso do desenvolvimento do aplicativo vinculado a este trabalho foi utilizado para desenvolvimento na linguagem Java (NEAON, 2017).

d) JavaServer Faces

JavaServer Faces (JSF) é uma tecnologia para criar aplicações Java para *web* utilizando componentes visuais predefinidos, de forma que o desenvolvedor não precise preocupar-se com JavaScript e *Hypertext Markup Language* (HTML). Para compor a interface do sistema basta adicionar os componentes (calendários, tabelas, formulários) que eles serão renderizados e exibidos em formato HTML (JAVASERVER FACES, 2017).

e) Spring Secutiry

Spring Security é um *framework* que provê autenticação e autorização para aplicações Java (SPRING, 2017). O Spring Security possui recursos avançados e que são de configuração simples que visam auxiliar com na segurança da aplicação (AFONSO, 2017). A autenticação pode ser realizada por banco de dados *Lightweight Directory Access Protocol* (LDAP) ou em memória. E a autorização é realizada por meio de permissões aos usuários autenticados. O LDAP é um protocolo de aplicação aberto, que não está vinculado a fornecedor específico ou padrão de indústria. Esse protocolo permite acessar e manter serviços de informação de diretório distribuído sobre uma rede de protocolo da Internet.

f) Primefaces

O PrimeFaces é uma biblioteca de componentes de interface gráfica para as aplicações *web* baseadas em JSF **(**SCHIECK, 2017)**.**

g) MySQL

O MySQL é um Sistema de Gerenciamento de Banco de Dados (SGBD), que utiliza a linguagem *Structured Query Language* (SQL) como interface (MYSQL, 2017).

3 RESULTADOS

Este capítulo apresenta o resultado da realização do trabalho que é o desenvolvimento de um aplicativo *web* para o controle de processos jurídicos.

3.1 ESCOPO DO SISTEMA

O *software* visa fornecer suporte à estrutura de trabalho de advogados em relação à logística organizacional que a rotina da sua profissão envolve.

Como primeira etapa na condução de cada processo é necessário haver organização em relação à carteira de clientes e de advogados. Manter um cadastro de contatos atualizado evita transtornos posteriores. Após os contatos serem cadastrados é possível criar o processo com as seguintes informações: a qual cliente o processo pertence, qual o advogado responsável e se o processo é judicial ou administrativo.

Durante o andamento do processo é possível anexar informações na forma de anotação ao mesmo. O sistema disponibiliza, ainda, a opção de anexar documentos que foram utilizados durante o andamento do processo. Esse armazenamento é feito em nuvem evitando perda de informações, além de ser um ambiente seguro de arquivamento devido à iniciação por meio de *login* e senha presentes no sistema.

A relação com o cliente também é facilitada por meio do aplicativo, no que diz respeito ao agendamento de horários. No ciclo de andamento do processo também é possível visualizar se serão necessárias novas reuniões, pois o sistema fornece o *status* do processo para acompanhamento do seu ciclo de vida.

3.2 MODELAGEM DO SISTEMA

O Quadro 2 apresenta os requisitos funcionais identificados para o sistema.

	Nome	**Descrição**
1	Manter pessoas	Adiciona os clientes (pessoas físicas e jurídicas) e advogados caso ainda não possuam cadastro. Também é possível alterar, excluir cadastros desde que não tenham vínculos com outros dados e consultar cadastros.
2	Manter processos	Cadastrar, alterar, excluir e consultar processos.
3	Manter documentos	Vincular documento selecionado a um processo.
4	Adicionar ações ao	Ações vinculadas ao processo.

	processo	
5	Manter agendamentos	Adicionar compromissos referentes ao processo.
6	Finalizar processo	O advogado seleciona um *status* para o processo que pode ser o que indica a sua finalização. Quando finalizado, o processo sai da fila de processos pendentes.

Quadro 2 – Requisitos funcionais

No requisito manter pessoas é realizada a manutenção dos clientes (nome, telefone, *email* e outros) e de advogados (número da Ordem dos Advogados do Brasil (OAB), nome, telefone, endereço, *email*.). Esses cadastros são utilizados para a realização dos demais requisitos. Nessa mesma etapa o advogado adiciona os processos novos ou em andamento, possibilitando assim, uma consulta posterior. Para realizar a inclusão de um processo o advogado precisa informar as partes (cliente, advogado e outros) e o número do processo.

No requisito “manter documentos” é possível vincular anexos de documentos ao processo e futuramente realizar o *download* deles.

No requisito “manter agendamentos”, o advogado pode incluir agendamentos vinculados a processos e agendamentos sem vínculo.

O requisito “adicionar ações” permite que sejam incluídos eventos de acontecimento planejado ou que já aconteceram no processo.

No requisito “finalizar o processo”, o advogado informa um *status* para o processo e pode finalizá-lo. Assim, o referido processo sai da fila de processos pendentes.

A Figura 1 apresenta o diagrama de casos de uso. Esse diagrama foi composto a partir dos requisitos levantados para o sistema. O sistema terá apenas um ator que é o profissional da área de direito, denominado no diagrama de advogado.

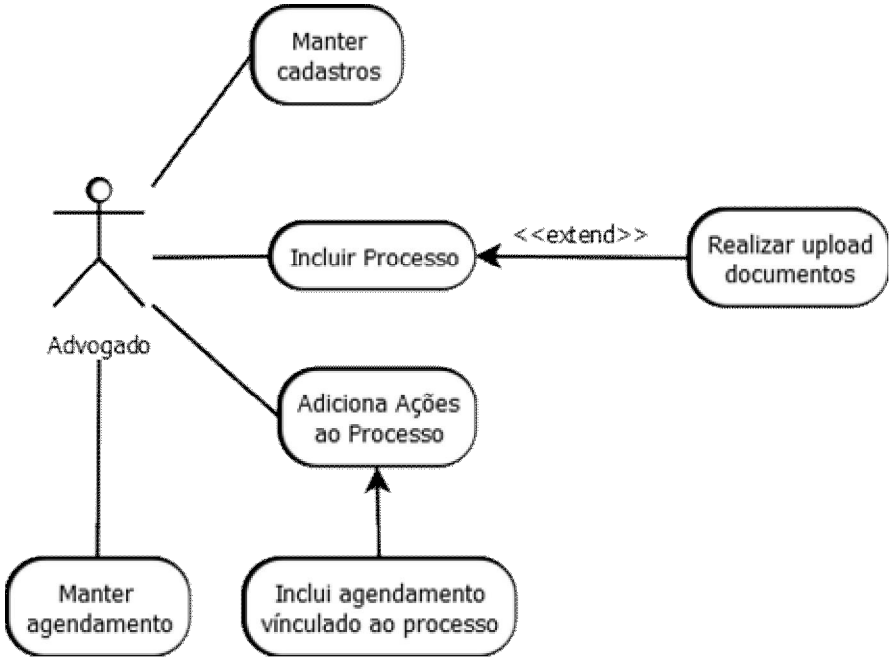

Figura 1 – Diagrama de casos de uso

A seguir são descritos os casos de uso apresentados na Figura 1.

O caso de uso manter cadastros é apresentado no Quadro 3.

Caso de uso: Manter cadastros. **Descrição**: O advogado pode cadastrar, alterar, consultar ou excluir uma nova pessoa de acordo com seu tipo: cliente ou advogado. O cliente pode ser pessoa física ou jurídica. O cliente é parte no processo. **Atores**: Advogado. **Pré-condição**: Advogado autenticado no sistema. **Sequência de Eventos**: 1. Advogado seleciona o formulário no qual deseja realizar a operação. 2. Sistema apresenta o formulário. 3. Advogado realiza a operação desejada: incluir, excluir, consultar, alterar. 4. Sistema verifica se os dados para a operação estão consistentes e realiza a operação. **Pós-Condição**: Operação de inclusão, exclusão, consulta ou alteração realizada.

Quadro 3 – Caso de uso manter cadastros

O caso de uso incluir processos é apresentado no Quadro 4.

Caso de uso: Incluir processos **Descrição**: O advogado pode cadastrar, consultar, alterar ou excluir um processo um processo cadastrado no sistema além de consultar esses processos. **Atores**: Advogado.

Pré-condição:
Advogado autenticado no sistema.
Sequência de Eventos:
1. Advogado seleciona o formulário no qual deseja realizar a operação.
2. Sistema apresenta o formulário.
3. Ação no documento.
    3.1 Inclusão
        3.1.1 Ator realiza o preenchimento dos campos obrigatórios.
    3.2 Alteração
    3.3 Exclusão
4. Sistema insere o processo no banco de dados se for inclusão, apresenta os dados para edição ou exclui o processo.
Pós-Condição:
Operação de inclusão, exclusão, consulta ou alteração realizada.

Quadro 4 – Caso de uso incluir processos

A descrição do caso de uso adicionar ações é apresentada no Quadro 5.

Caso de uso:
Adicionar ações ao processo.
Descrição:
O advogado pode inserir os eventos do processo por meio das ações bem como consultá-las posteriormente.
Atores:
Advogado.
Pré-condição:
Advogado autenticado no sistema.
Sequência de Eventos:
1. Advogado seleciona o processo e em seguida realiza a abertura do formulário de ações.
2. Sistema apresenta o formulário.
3. Advogado realiza o preenchimento dos campos.
4. Sistema insere a nova ação no banco de dados.
Pós-Condição:
Dados do pedido inseridos no banco de dados e vinculados ao processo.

Quadro 5 – Caso de uso adicionar ações

A descrição do caso de uso adicionar documentos é apresentada no Quadro 6.

Caso de uso:
Adicionar documentos.
Descrição:
O advogado pode inserir documentos ao processo bem como baixá-los posteriormente.
Atores:
Advogado.
Pré-condição:
O advogado deve possuir documentos para adicionar.
O advogado deve abrir a tela e selecionar os documentos desejados.
Sequência de Eventos:
1. Advogado abre o formulário para selecionar documentos.
2. Advogado seleciona o processo.
3. Advogado seleciona o documento.
4. Sistema salva os documentos no banco de dados.
Pós-Condição:
Documentos vinculados ao processo e inclusos no sistema.

Quadro 6 – Caso de uso adicionar documentos

A descrição do caso de uso adicionar compromissos é apresentada no Quadro 7.

Caso de uso: Manter agendamentos. **Descrição**: São todos os compromissos que pertencem ao processo, como as audiências. **Atores**: Advogado. **Pré-condição**: Processo cadastrado. **Sequência de Eventos**: 1. Advogado abre o formulário para adicionar agendamento a processo. 2. Sistema apresenta formulário. 3. Dados do compromisso são adicionados. 4. Sistema salva informações. **Pós-Condição**: Compromisso vinculado ao processo e dados adicionados no banco de dados.

Quadro 7 – Caso de uso adicionar compromissos

Na Figura 2 está o diagrama de classes. Este diagrama apresenta as classes utilizadas no sistema e os seus relacionamentos. A classe pessoa, da qual herdam a classe física e jurídica e advogado, é uma pessoa física à qual é incluído um número de OAB. Portanto, a classe advogado herda de pessoa física que herda de pessoa.

A classe processo é a principal do sistema. É de um processo que são realizados agendamentos. Os processos possuem uma natureza, ou seja, um tipo. Ao processo estão relacionados eventos e documentos são vinculados aos processos.

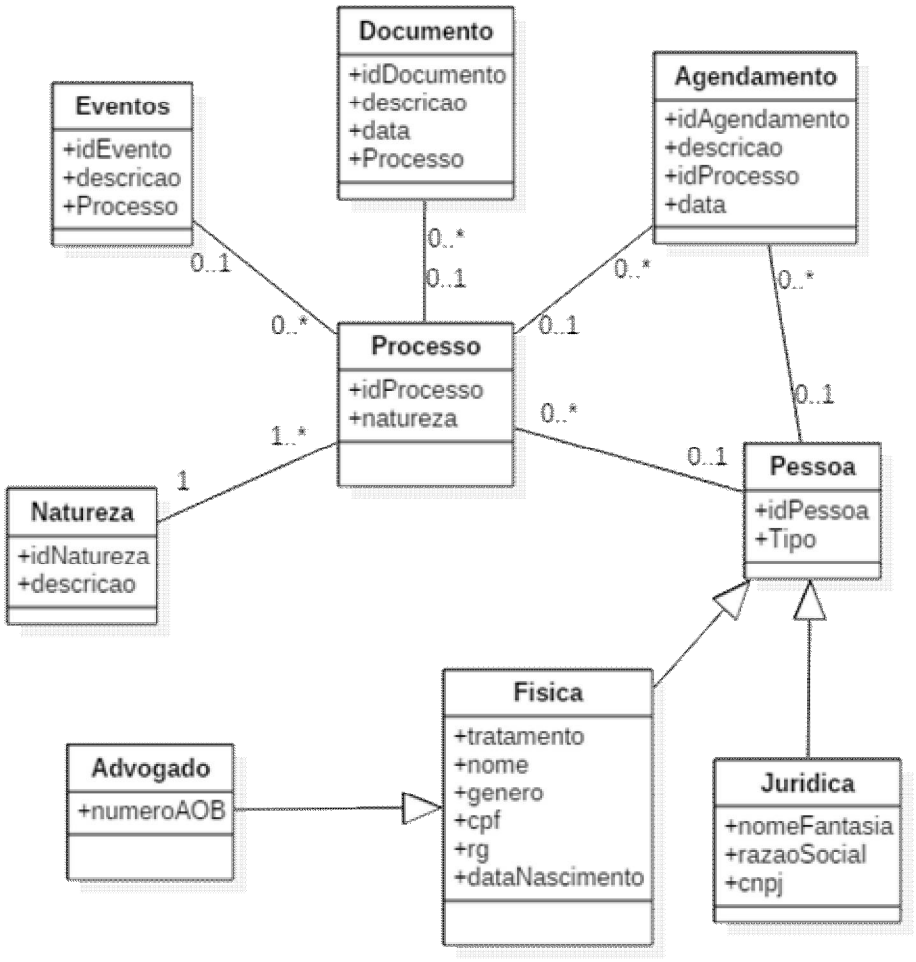

Figura 2 – Diagrama de classes

O diagrama de entidades e relacionamentos do banco de dados é apresentado na Figura 3. Processo é a tabela principal. A um processo estão vinculados clientes, advogados e natureza (que categoria ou classifica o processo de acordo com um tipo). Os processos possuem agendamentos para as ações realizadas como audiências e necessidade de juntada de documentos. Esses documentos podem ser arquivados no sistema e para isso são vinculados ao processo. Os processos possuem eventos que determinam o seu ciclo de vida, até que seja finalizado.

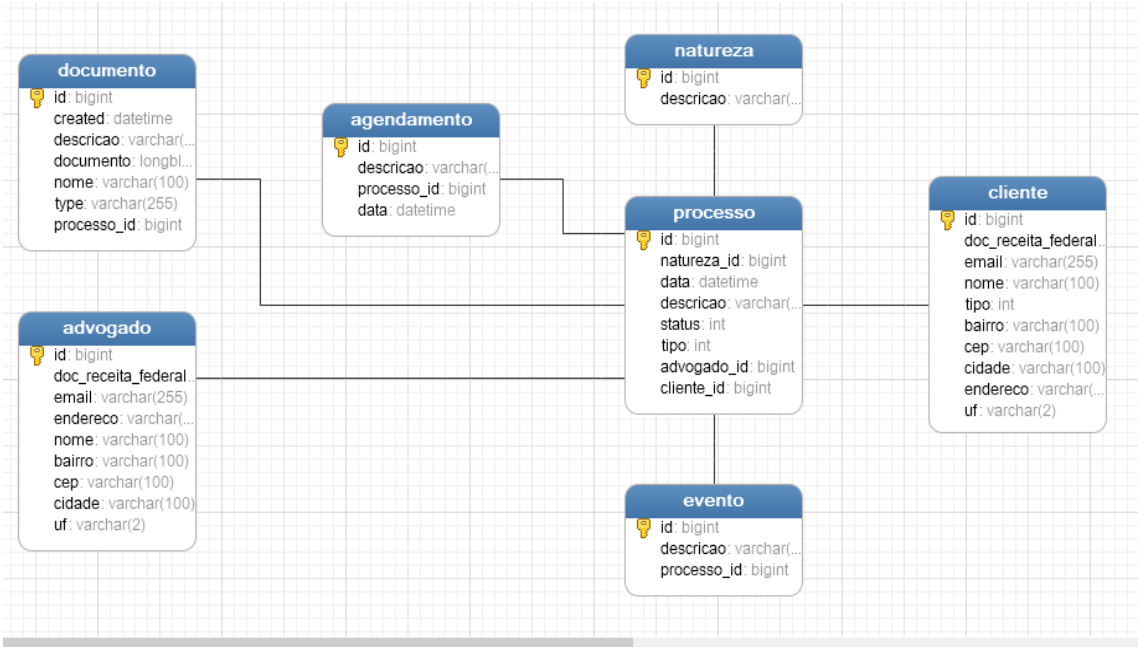

Figura 3 – Diagrama de entidades e relacionamentos do banco de dados

3.3 APRESENTAÇÃO DO SISTEMA

O leiaute do sistema é dividido em três partes: o cabeçalho, que ocupa o topo da página, o menu, que ocupa a lateral esquerda e o corpo que ocupa o restante da página. A Figura 4 apresenta o modelo de construção do leiaute.

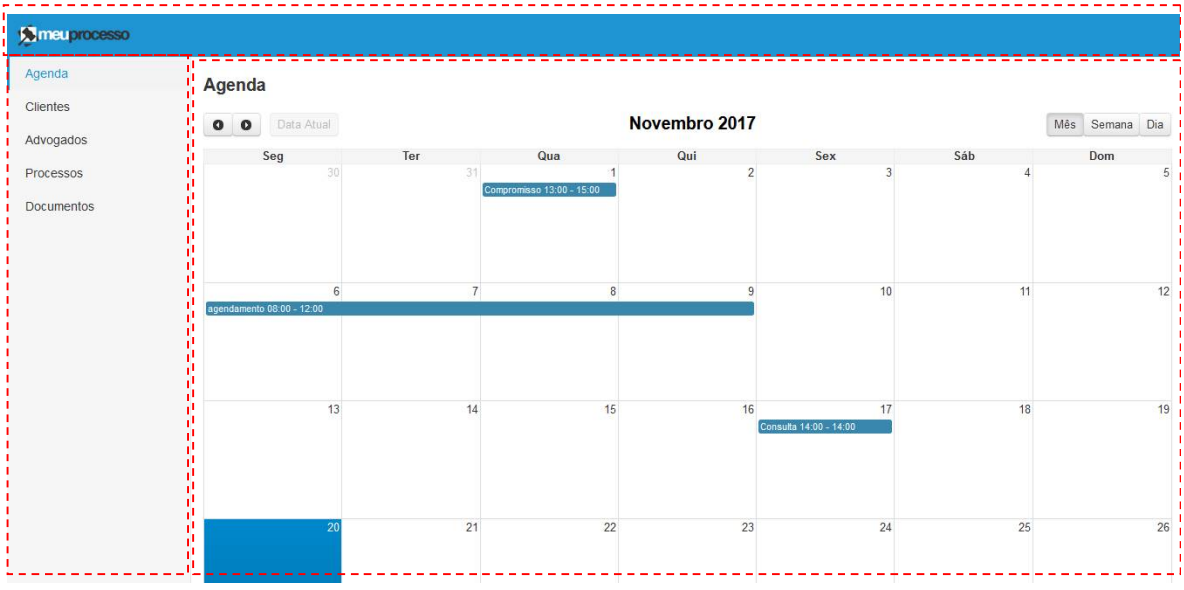

Figura 4 – Leiaute do sistema

O cabeçalho é composto pela marca de identificação do sistema, que fica no lado esquerdo.

As Figuras 5, 6 e 7 apresentam o corpo da página preenchido com o conteúdo da tela de manutenção de cadastros. Nessa página, o advogado pode visualizar todas as entidades cadastradas e também realizar as operações básicas de cadastro como inclusão (botão novo), alteração (ícone na coluna editar) e exclusão (botão na coluna excluir).

Figura 5 – Tela de manutenção de clientes

Telas semelhantes são apresentadas para os cadastros de advogados e processos, apresentadas nas Figuras 6 e 7.

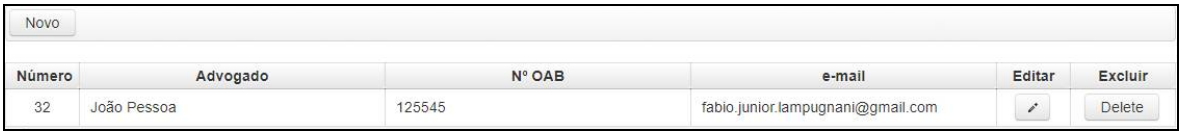

Figura 6 – Tela de manutenção de advogados

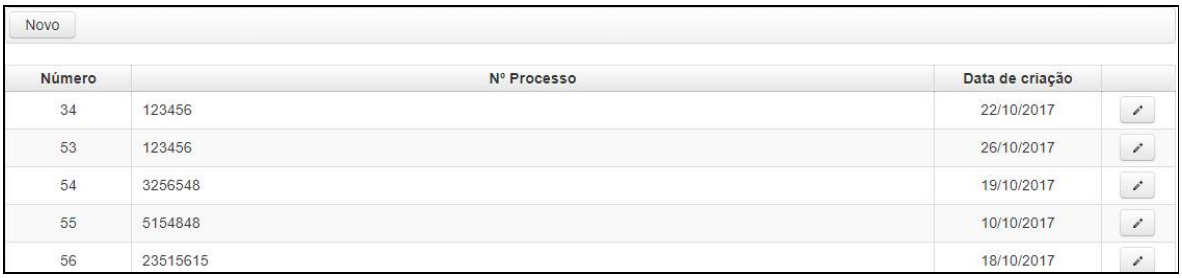

Figura 7 – Tela de manutenção de processos

Na tela de cadastro de documentos apresentada na Figura 8, o advogado conta com um botão para fazer o *upload* do documento no momento do cadastro. Após o documento estar cadastrado, ele pode ser aberto (visualizado) por meio do botão “Download”.

Figura 8 – Tela de manutenção de documentos

Todas as telas de manutenção possuem o botão “Novo” que quando pressionado o sistema apresenta uma janela contendo um formulário com os campos necessários para a

inclusão de um novo cadastro, como mostra as Figuras 9, 10 e 11. A Figura 12 apresenta uma tela de diálogo na qual o advogado deve selecionar o documento e o processo.

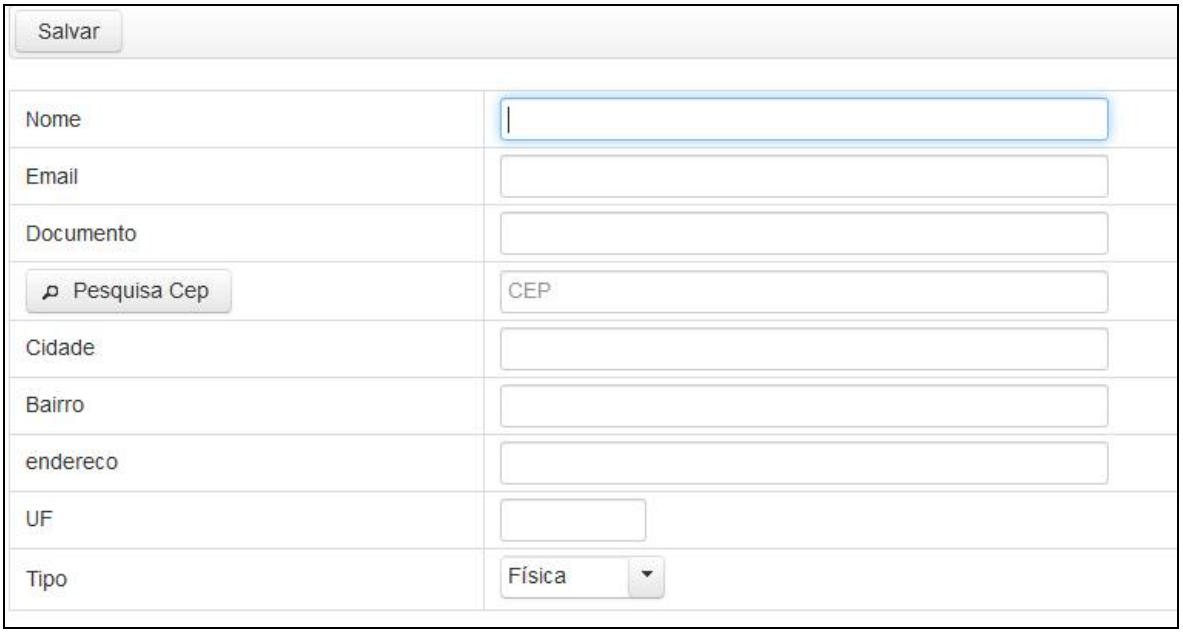

Figura 9 – Formulário para inclusão de um novo cliente

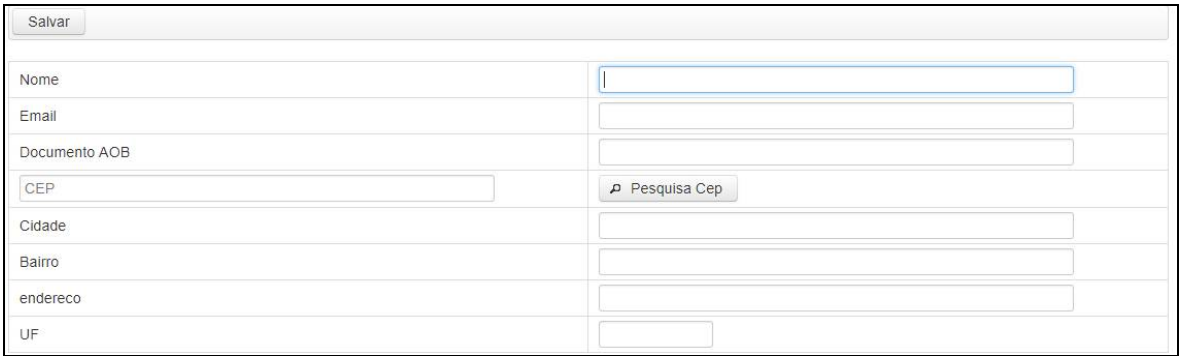

Figura 10 – Formulário para inclusão de um novo advogado

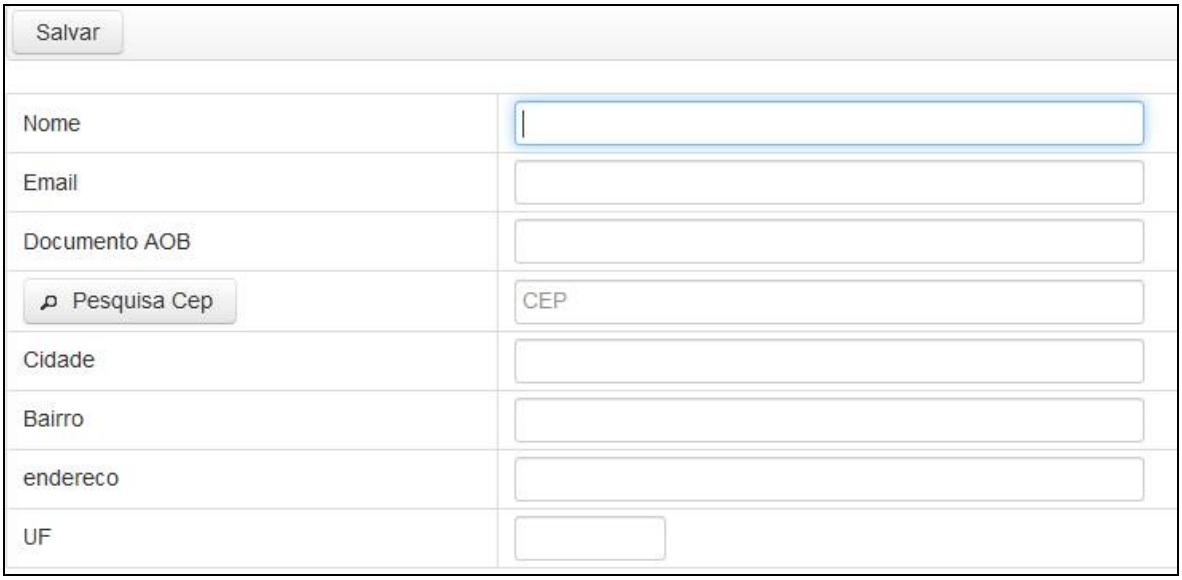

Figura 11 – Formulário para inclusão de um novo processo

Para incluir um cadastro, o advogado deve preencher os campos obrigatórios e clicar no botão “Salvar”. A operação de alteração é semelhante à de inclusão, sendo que, para alterar algum cadastro o advogado deve clicar no botão “Alterar” da linha referente ao curso desejado. Dessa forma, o sistema fará o mesmo procedimento utilizado na inclusão, mas já trará o formulário preenchido com os dados do cadastro selecionado. Para realizar a operação de exclusão, o advogado deve clicar no botão “Remover” da linha referente ao cadastro desejado.

Na Figura 12 é apresenta a tela para importar documentos vinculados ao processo. Os documentos vinculados podem ser visualizados por meio do botão “Download” da Figura 10.

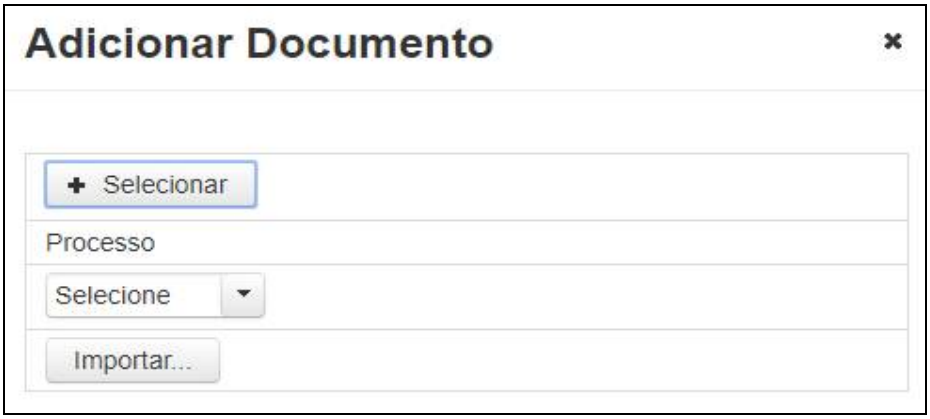

Figura 12 – Formulário para inclusão de um novo documento

Outro processo do sistema é a agenda conforme mostra a Figura 13. Nela é possível realizar inclusão e alteração de eventos.

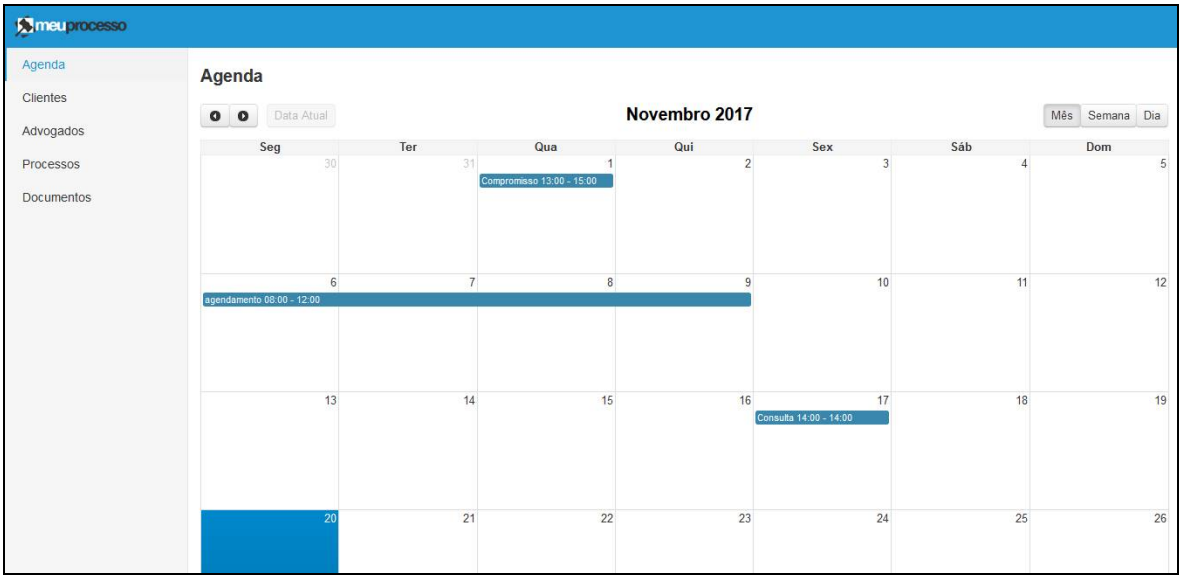

Figura 13 – Formulário de manutenção de agendamentos

Ao clicar em uma data em branco é apresentada a tela para inserir um novo agendamento. Para editar um agendamento realizado basta clicar sobre o agendamento cadastrado na agenda que abrirá a tela de edição conforme a Figura 14.

Figura 14 – Formulário para inclusão de um novo evento

3.4 IMPLEMENTAÇÃO DO SISTEMA

O sistema foi implementado utilizando o IDE Eclipse Mars e o Maven, que vem instalado por padrão nessa versão de IDE. Maven é uma ferramenta de gerenciamento, construção e implantação de projetos, criada pela Fundação Apache. Sua unidade básica de configuração é um arquivo *eXtensible Markup Language* (XML) chamado *Project Object Model* (POM) que deve ficar na raiz do projeto. Nesse arquivo, declara-se a estrutura, as dependências e as características do projeto. A Listagem 1 mostra o código do arquivo pom.xml.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>com.algaworks</groupId>
	<artifactId>MeuProcesso</artifactId>
	<version>1.0.0-SNAPSHOT</version>
	<packaging>war</packaging>

	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
	</properties>

	<dependencies>
		<!-- Tema do Bootstrap -->
		<dependency>
			<groupId>org.primefaces.themes</groupId>
			<artifactId>bootstrap</artifactId>
			<version>1.0.10</version>
			<scope>compile</scope>
		</dependency>

		<!-- Usado para geração de arquivos Excel -->
		<dependency>
			<groupId>org.apache.poi</groupId>
			<artifactId>poi</artifactId>
			<version>3.11</version>
			<scope>compile</scope>
		</dependency>

		<!-- Implementacao do Bean Validation -->
		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-validator</artifactId>
			<version>5.3.1.Final</version>
			<scope>compile</scope>
		</dependency>

		<!-- Núcleo do Hibernate -->
		<dependency>
			<groupId>org.hibernate</groupId>
			<artifactId>hibernate-core</artifactId>
```

```
                        <version>5.2.4.Final</version>
                        <scope>compile</scope>
                </dependency>

                <!-- Implementação de EntityManager da JPA -->
                <dependency>
                        <groupId>org.hibernate</groupId>
                        <artifactId>hibernate-entitymanager</artifactId>
                        <version>5.2.4.Final</version>
                        <scope>compile</scope>
                </dependency>
                <!-- Pool de conexoes com C3P0 -->
                <dependency>
                        <groupId>org.hibernate</groupId>
                        <artifactId>hibernate-c3p0</artifactId>
                        <version>5.2.4.Final</version>
                        <scope>compile</scope>
                </dependency>

                <!-- Driver JDBC do MySQL -->
                <dependency>
                        <groupId>mysql</groupId>
                        <artifactId>mysql-connector-java</artifactId>
                        <version>5.1.34</version>
                        <scope>compile</scope>
                </dependency>

                <!-- https://mvnrepository.com/artifact/org.postgresql/postgresql --
>
                <dependency>
                        <groupId>org.postgresql</groupId>
                        <artifactId>postgresql</artifactId>
                        <version>42.1.4</version>
                </dependency>


                <!-- Weld (implementação do CDI) -->
                <dependency>
                        <groupId>org.jboss.weld.servlet</groupId>
                        <artifactId>weld-servlet</artifactId>
                        <version>2.2.9.Final</version>
                        <scope>compile</scope>
                </dependency>

                <!-- Weld depende do Jandex -->
                <dependency>
                        <groupId>org.jboss</groupId>
                        <artifactId>jandex</artifactId>
                        <version>1.2.2.Final</version>
                        <scope>compile</scope>
                </dependency>

                <!-- PrimeFaces (biblioteca de componentes) -->
                <dependency>
                        <groupId>org.primefaces</groupId>
                        <artifactId>primefaces</artifactId>
                        <version>6.0</version>
                        <scope>compile</scope>
                </dependency>
```

```
<!-- Mojarra (implementacao do JSF) -->
<dependency>
	<groupId>org.glassfish</groupId>
	<artifactId>javax.faces</artifactId>
	<version>2.2.13</version>
	<scope>compile</scope>
</dependency>

<!-- OmniFaces (utilitarios para JSF) -->
<dependency>
	<groupId>org.omnifaces</groupId>
	<artifactId>omnifaces</artifactId>
	<version>2.5.1</version>
	<scope>compile</scope>
</dependency>

<!-- Log4J -->
<dependency>
	<groupId>org.apache.logging.log4j</groupId>
	<artifactId>log4j-api</artifactId>
	<version>2.7</version>
</dependency>
<dependency>
	<groupId>org.apache.logging.log4j</groupId>
	<artifactId>log4j-core</artifactId>
	<version>2.7</version>
</dependency>
<dependency>
	<groupId>org.apache.logging.log4j</groupId>
	<artifactId>log4j-jcl</artifactId>
	<version>2.7</version>
</dependency>

<!-- Commons Logging (abstrai a implementação de logging) -->
<dependency>
	<groupId>commons-logging</groupId>
	<artifactId>commons-logging</artifactId>
	<version>1.2</version>
	<scope>compile</scope>
</dependency>

<!-- Commons Lang (utilidades) -->
<dependency>
	<groupId>org.apache.commons</groupId>
	<artifactId>commons-lang3</artifactId>
	<version>3.1</version>
	<scope>compile</scope>
</dependency>

<!-- Abstração para envio de e-mails -->
<dependency>
	<groupId>com.outjected</groupId>
	<artifactId>simple-email</artifactId>
	<version>0.2.1</version>
	<scope>compile</scope>
</dependency>

<!-- Requerido para envio de e-mails -->
```

```
		<dependency>
			<groupId>javax.mail</groupId>
			<artifactId>mail</artifactId>
			<version>1.4.7</version>
			<scope>compile</scope>
		</dependency>

		<!-- Usamos para templates de emails -->
		<dependency>
			<groupId>org.apache.velocity</groupId>
			<artifactId>velocity</artifactId>
			<version>1.7</version>
			<scope>compile</scope>
		</dependency>

		<!-- Usamos para formatar números em templates -->
		<dependency>
			<groupId>velocity-tools</groupId>
			<artifactId>velocity-tools-generic</artifactId>
			<version>1.1</version>
			<scope>compile</scope>
		</dependency>

		<!-- Spring Security (autenticação e autorização) -->
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-core</artifactId>
			<version>4.1.3.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-web</artifactId>
			<version>4.1.3.RELEASE</version>
		</dependency>
		<dependency>
			<groupId>org.springframework.security</groupId>
			<artifactId>spring-security-config</artifactId>
			<version>4.1.3.RELEASE</version>
		</dependency>

		<dependency>
			<groupId>net.sf.jasperreports</groupId>
			<artifactId>jasperreports</artifactId>
			<version>6.0.3</version>
			<scope>compile</scope>
		</dependency>

		<dependency>
			<groupId>net.sf.jasperreports</groupId>
			<artifactId>jasperreports-fonts</artifactId>
			<version>6.0.0</version>
			<scope>compile</scope>
		</dependency>

		<!-- API de Servlet -->
		<dependency>
			<groupId>javax.servlet</groupId>
			<artifactId>javax.servlet-api</artifactId>
			<version>3.1.0</version>
```

```
                <scope>provided</scope>
            </dependency>
        </dependencies>

        <build>
            <finalName>MeuProcesso</finalName>
            <plugins>
                <plugin>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.0</version>
                    <configuration>
                        <source>1.8</source>
                        <target>1.8</target>
                    </configuration>
                </plugin>
            </plugins>
        </build>

        <repositories>
            <repository>
                <id>prime-repo</id>
                <name>PrimeFaces Maven Repository</name>
                <url>http://repository.primefaces.org</url>
                <layout>default</layout>
            </repository>
        </repositories>

</project>
```

Listagem 1 – Pom.xml

As dependências contidas no arquivo POM são todas as bibliotecas externas das quais o projeto depende para ser construído. Essas dependências são gerenciadas pelo próprio Maven. Primeiramente, ele cria um diretório local e, então, por meio da configuração contida no arquivo POM, baixa as dependências de um repositório *online* e as armazena no diretório criado. Ao ser feita a construção do projeto, o Maven busca as dependências necessárias, contidas no diretório local e as adiciona ao projeto.

Além do gerenciamento das dependências, o Maven permite a criação do leiaute do projeto, ou seja, a estrutura de diretórios. Ele possui uma convenção para a criação, mas, também é possível criar uma estrutura customizada que é realizada por meio da configuração no arquivo POM. A Figura 15 mostra a estrutura do projeto, criada sem customização.

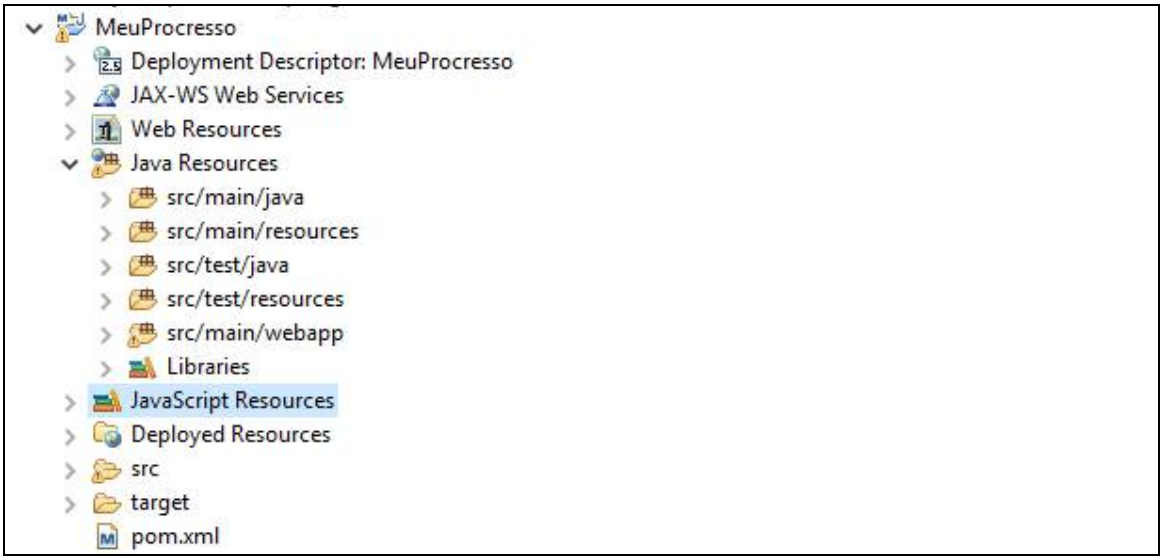

Figura 15 – Estrutura do projeto

A Listagem 2 apresenta a dependência do Weld que é a implementação da especificação da injeção de contextos e dependência (*Context and Dependency Injection* – CDI) que está sendo usada para injeção de dependência.

```
<!-- Weld (implementação do CDI) -->
<dependency>
    <groupId>org.jboss.weld.servlet</groupId>
    <artifactId>weld-servlet</artifactId>
    <version>2.2.9.Final</version>
    <scope>compile</scope>
</dependency>
```

Listagem 2 – Dependência weld

Para realizar a configuração do Weld para o Tomcat é necessário adicionar dentro da pasta scr/main/webapp/META-INF/contexto.xml conforme Listagem 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
    <!-- disables storage of sessions across restarts -->
    <Manager pathname=""/>
    <Resource name="BeanManager" auth="Container"
        type="javax.enterprise.inject.spi.BeanManager"
        factory="org.jboss.weld.resources.ManagerObjectFactory"/>
</Context>
```

Listagem 3 – Adição do weld na pasta scr/main/webapp/META-INF/contexto.xml

Outra configuração a ser realizada é no arquivo src/main/webapp/WEB-INF/web.xml conforme Listagem 4.

```
<listener>
    <listener-class>org.jboss.weld.environment.servlet.Listener</listener-class>
</listener>
<resource-env-ref>
    <resource-env-ref-name>BeanManager</resource-env-ref-name>
    <resource-env-ref-type>javax.enterprise.inject.spi.BeanManager</resource-env-ref-type
</resource-env-ref>
```

Listagem 4 – Configuração do arquivo src/main/webapp/WEB-INF/web.xml

Por fim, nas classes de controle é substituído o @managedbean pelo @Named. O código para essa substituição é apresentado na Listagem 5.

```
@Named
@ViewScoped
public class CadastroClienteBean implements Serializable {
```

Listagem 5 – Substituição do @managedbean pelo @Named

Uma das configurações realizadas pelo método "webSecurityConfigurerAdapter", refere-se ao módulo de segurança. Esse método retorna um novo objeto da classe "WebSecurityConfig", apresentada na Listagem 6.

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurit
y;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfi
gurerAdapter;
import org.springframework.security.web.util.matcher.AntPathRequestMatcher;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

	@Bean
	public AppUserDetailsService userDetailsService() {
		return new AppUserDetailsService();
	}

	@Override
	protected void configure(HttpSecurity http) throws Exception {
		JsfLoginUrlAuthenticationEntryPoint jsfLoginEntry = new
JsfLoginUrlAuthenticationEntryPoint();
		jsfLoginEntry.setLoginFormUrl("/Login.xhtml");
		jsfLoginEntry.setRedirectStrategy(new JsfRedirectStrategy());

		JsfAccessDeniedHandler jsfDeniedEntry = new
JsfAccessDeniedHandler();
		jsfDeniedEntry.setLoginPath("/AcessoNegado.xhtml");
		jsfDeniedEntry.setContextRelative(true);
```

```
            http
                  .csrf().disable()
                  .headers().frameOptions().sameOrigin()
                  .and()

            .authorizeRequests()
                  .antMatchers("/Login.xhtml", "/Erro.xhtml",
"/javax.faces.resource/**").permitAll()
                  .antMatchers("/Home.xhtml", "/AcessoNegado.xhtml",
"/dialogos/**").authenticated()
                  .antMatchers("/agenda/**",
"/cadastros/**").hasRole("ADMINISTRADORES")
                  .and()

            .formLogin()
                  .loginPage("/Login.xhtml")
                  .failureUrl("/Login.xhtml?invalid=true")
                  .and()

            .logout()
                  .logoutRequestMatcher(new AntPathRequestMatcher("/logout"))
                  .and()

            .exceptionHandling()
                  .accessDeniedPage("/AcessoNegado.xhtml")
                  .authenticationEntryPoint(jsfLoginEntry)
                  .accessDeniedHandler(jsfDeniedEntry);
      }

}
```

Listagem 6 – Código da SecurityConfig.java

Como apresentado na Listagem 6, a classe “WebSecurityConfig” estende uma classe do Spring chamada “WebSecurityConfigurerAdapter”. Essa classe está contida no módulo Spring Security. Ainda na Listagem 6, há um método chamado “configure” que recebe um objeto da classe “HttpSecurity” por parâmetro. É nesse método que é realizada a configuração do controle de acesso das *Uniform Resource Locator* (URL), permitindo ou negando acesso de acordo com a URL digitada e do perfil do usuário autenticado.

Outra dependência é o Hibernate que é um *framework* criado para facilitar a persistência de dados em Java, sendo que é uma especificação *Java Persistence API* (JPA). JPA é uma coleção de classes e métodos voltados para armazenar persistentemente as vastas quantidades de dados em um banco de dados. Para realizar a configuração do Hibernate é necessário adicionar as dependências apresentadas na Listagem 7.

```
<!-- Núcleo do Hibernate -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.2.4.Final</version>
    <scope>compile</scope>
</dependency>

<!-- Implementação de EntityManager da JPA -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-entitymanager</artifactId>
    <version>5.2.4.Final</version>
    <scope>compile</scope>
</dependency>
```

Listagem 7 – Configuração do Hibernate

Outra dependência importante é o *driver* do banco de dados. Como está sendo usado o banco de dados MySQL deve ser informado o seu *driver*, como apresentado na Listagem 8.

```
<!-- Driver JDBC do MySQL -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>5.1.34</version>
    <scope>compile</scope>
</dependency>
```

Listagem 8 – Adição do driver JDBC do MySQL

Outra configuração a ser feita é no arquivo persistence.xml na pasta src/main/webapp. Neste XML é informado *login* e senha do banco de dados e nome do banco de dados, entre outras configurações necessárias, como apresentado na Listagem 9.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1"
      xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">

      <persistence-unit name="PedidoPU">
            <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
            <properties>
                  <property name="javax.persistence.jdbc.url"
value="jdbc:mysql://localhost/MeuProcesso" />
                  <property name="javax.persistence.jdbc.user" value="root" />
                  <property name="javax.persistence.jdbc.password" value="mysql"/>
                  <property name="javax.persistence.jdbc.driver"
value="com.mysql.jdbc.Driver" />

                  <property name="hibernate.hbm2ddl.auto" value="update" />
                  <property name="hibernate.show_sql" value="true" />
```

```
                    <property name="hibernate.dialect"
value="org.hibernate.dialect.MySQL5Dialect" />

                    <property name="hibernate.connection.provider_class"
                        value="org.hibernate.connection.C3P0ConnectionProvider" />
                    <property name="hibernate.c3p0.max_size" value="20" />
                    <property name="hibernate.c3p0.min_size" value="5" />
                    <property name="hibernate.c3p0.acquire_increment" value="1" />
                    <property name="hibernate.c3p0.idle_test_period" value="300"/>
                    <property name="hibernate.c3p0.max_statements" value="50" />
                    <property name="hibernate.c3p0.timeout" value="300" />
            </properties>
      </persistence-unit>
</persistence>
```

Listagem 9 – Persistence.xml

A Listagem 10 apresenta o código da classe “Cliente” que possui as anotações do JPA e o atributo “id” do tipo “Long”.

```
import java.io.Serializable;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;
import javax.persistence.Transient;

@Entity
@Table(name = "cliente")
public class Cliente implements Serializable {

      private static final long serialVersionUID = 1L;

      private Long id;
      private String nome;
      private String email;
      private String documentoReceitaFederal;
      private TipoPessoa tipo;
      private String endereco;

      @Id
      @GeneratedValue
      public Long getId() {
            return id;
      }

      public void setId(Long id) {
            this.id = id;
      }

      @Column(nullable = false, length = 100)
      public String getNome() {
            return nome;
      }

      public void setNome(String nome) {
            this.nome = nome;
```

```
	}

	@Column(nullable = false, length = 255)
	public String getEmail() {
		return email;
	}

	public void setEmail(String email) {
		this.email = email;
	}

	@Column(name = "doc_receita_federal", nullable = false, length = 14)
	public String getDocumentoReceitaFederal() {
		return documentoReceitaFederal;
	}

	public void setDocumentoReceitaFederal(String documentoReceitaFederal) {
		this.documentoReceitaFederal = documentoReceitaFederal;
	}

	@Column(name = "endereco", nullable = false, length = 60)
	public String getEndereco() {
		return endereco;
	}

	public void setEndereco(String endereco) {
		this.endereco = endereco;
	}

	@Column(nullable = false)
	public TipoPessoa getTipo() {
		return tipo;
	}

	public void setTipo(TipoPessoa tipo) {
		this.tipo = tipo;
	}

	@Transient
	public boolean isNovo() {
		return getId() == null;
	}

	@Transient
	public boolean isExistente() {
		return !isNovo();
	}
	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		return result;
	}
	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
```

```
                return false;
            if (getClass() != obj.getClass())
                return false;
            Cliente other = (Cliente) obj;
            if (id == null) {
                if (other.getId() != null)
                    return false;
            } else if (!id.equals(other.getId()))
                return false;
            return true;
        }
}
```

Listagem 10 – Cliente.java

Na Listagem 11 está o código para realizar a injeção da classe EntityManager que é responsável por incluir, alterar e excluir registros no banco de dados.

```
public class Clientes implements Serializable {

        private static final long serialVersionUID = 1L;

        @Inject
        private EntityManager manager;

        public Cliente porId(Long id) {
                return this.manager.find(Cliente.class, id);
        }

        public Cliente guardar(Cliente cliente) {
                return this.manager.merge(cliente);
        }

        @Transactional
        public void remover(Cliente cliente) throws NegocioException0020{
                try {
                        cliente = porId(cliente.getId());
                        manager.remove(cliente);
                        manager.flush();
                } catch (PersistenceException e) {
                        throw new NegocioException("Cliente não pode ser excluído.");
                }
        }

        public List<Cliente> filtrados(ClienteFilter filtro) {
                CriteriaBuilder builder = manager.getCriteriaBuilder();
                CriteriaQuery<Cliente> criteriaQuery =
builder.createQuery(Cliente.class);
                List<Predicate> predicates = new ArrayList<>();
                Root<Cliente> clienteRoot = criteriaQuery.from(Cliente.class);
                if (StringUtils.isNotBlank(filtro.getNome())) {
        predicates.add(builder.like(builder.lower(clienteRoot.get("nome")),
                                        "%" + filtro.getNome().toLowerCase() + "%"));
                }

                criteriaQuery.select(clienteRoot);
                criteriaQuery.where(predicates.toArray(new Predicate[0]));
                criteriaQuery.orderBy(builder.asc(clienteRoot.get("nome")));
```

```
            TypedQuery<Cliente> query = manager.createQuery(criteriaQuery);
            return query.getResultList();
      }


      public List<Cliente> porNome(String nome) {
            return this.manager.createQuery("from Cliente where upper(nome) like
:nome", Cliente.class)
                        .setParameter("nome", nome.toUpperCase() +
"%").getResultList();
      }
}
```

Listagem 11 – Clientes.java

A aplicação utiliza o JSF que fornece um subprojeto chamado “Facelets”, esse projeto facilita a criação de *templates*. A Listagem 12 mostra o código do arquivo “Home.xhtml”, utilizado como base para a criação do leiaute.

```
<ui:composition template="/WEB-INF/template/Layout.xhtml"
    xmlns="http://www.w3.org/1999/xhtml"
    xmlns:h="http://xmlns.jcp.org/jsf/html"
    xmlns:f="http://xmlns.jcp.org/jsf/core"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    xmlns:p="http://primefaces.org/ui">

      <ui:define name="content">
            <h1 class="aw-page-title">Painel Dashboard</h1>

            <h:form id="frm">
                  Conteúdo do formulário
            </h:form>
      </ui:define>

</ui:composition>
```

Listagem 12 – Home.xhtml

Ainda há dois arquivos utilizados na criação do leiaute: um deles é o arquivo “layout.xhtml” (Listagem 13) e o outro é o arquivo “menu.xhtml” (Listagem 14). Os arquivos do leiaute ficam no diretório “resources/templates”.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:f="http://xmlns.jcp.org/jsf/core"
      xmlns:ui="http://xmlns.jcp.org/jsf/facelets">

<h:head>
      <f:facet name="first">
            <meta charset="UTF-8" />
            <meta name="viewport" content="width=device-width, initial-scale=1"
/>
      </f:facet>
```

```
        <title>MeuProcesso</title>

        <h:outputStylesheet library="meuprocesso" name="styles/custom.css" />
        <h:outputStylesheet library="meuprocesso" name="styles/layout.css" />
        <h:outputStylesheet library="meuprocesso" name="styles/components.css" />

        <h:outputScript target="body" library="primefaces" name="jquery/jquery.js"
/>
        <h:outputScript target="body" library="algaworks" name="javascripts/app.js"
/>
</h:head>

<h:body>

        <header class="aw-topbar">
                <h:graphicImage library="algaworks" name="images/logo.png" />

                <a href="#" class="aw-toggle  js-toggle"><i class="fa  fa-
bars"></i></a>
        </header>

        <aside class="aw-sidebar  js-sidebar">
                <ui:include src="Menu.xhtml" />
        </aside>

        <section class="aw-content  js-content">
                <ui:insert name="content" />
        </section>

</h:body>

</html>
```

Listagem 13 – Layout.xhtml

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
        xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
        xmlns:h="http://xmlns.jcp.org/jsf/html">

        <nav class="aw-menu">
                <ul>
                        <li class="is-selected">
                                <h:link outcome="/Home"><i class="fa  fa-fw  fa-
home"></i>Painel</h:link>
                        </li>
                        <li>
                                <h:link outcome="/cadastros/Clientes"><i class="fa  fa-
fw  fa-user"></i>Clientes</h:link>
                        </li>
                        <li>
                                <h:link outcome="/cadastros/Processos"><i class="fa  fa-
fw  fa-file-text"></i>Processos</h:link>
                        </li>
                        <li>
                                <h:link outcome="/cadastros/basic"><i class="fa  fa-fw
fa-cog"></i>Documentos</h:link>
                        </li>
                        <li>
                                <h:link outcome="/agenda/ScheduleView"><i class="fa  fa-
```

```
fw  fa-cog"></i>Agenda</h:link>
                    </li>
                </ul>
            </nav>

</ui:composition>
```

Listagem 14 – Menu.xhtml

4 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi implementar um aplicativo *web* para registrar os clientes, os processos e os documentos para auxílio no gerenciamento de atividades realizadas por advogados. O aplicativo foi desenvolvido utilizando diversas ferramentas e tecnologias. Foi necessário implementar diversas regras e restrições relacionadas à disponibilização da versão, tornando o processo de implementação mais demorado do que o inicialmente planejado.

Uma dessas tecnologias utilizada no desenvolvimento do sistema é o Spring Security. Essa ferramenta facilita o controle de autenticação de usuários, agilizando o desenvolvimento e possui ampla documentação e exemplos de utilização.

O Spring Security é facilmente integrado com o JSF, que é um *framework* que permite a elaboração de interfaces do usuário *web* colocando componentes em um formulário e ligando-os a objetos Java, permitindo a separação entre lógica e regra de negócio, navegação, conexões com serviços externos e gerenciamento de configurações.

Para a injeção de dependências foi utilizada a especificação CDI, que está incluída em todos os servidores de aplicação. CDI se aplica muito bem em projetos Java.

O PrimeFaces, assim como outras bibliotecas semelhantes, tem a finalidade de melhorar os componentes já existentes na especificação e, ao mesmo tempo, incluir novos. Ele possui uma aceitação muito boa pela comunidade e uma vasta documentação, que pode ser encontrada, inclusive, na sua página oficial (http://primefaces.org/), facilitando o desenvolvimento e a solução de possíveis problemas. Uma das suas desvantagens é a estilização visual da aplicação, pois, criar um estilo diferente do padrão é bastante complexo, limitando o programador, geralmente, a customizar o já existente.

REFERÊNCIAS

AFONSO, Alexandre. **O que é Spring Security**. Disponível em: <http://blog.algaworks.com/spring-security/>. Acesso em: 15 out. 2017.

ECLIPSE NEON. Disponível em: <https://www.eclipse.org/neon/>. Acesso em: 17 nov. 2017.

JAVASERVER FACES. **JavaServer Faces technology.** Disponível em: <http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html>. Acesso em: 17 nov. 2017.

MYSQL. Disponível em: <https://www.mysql.com/>. Acesso em: 17 nov. 2017.

ORACLE. **Essentials of the Java Programming Language: part 1**. Disponível em: <http://www.oracle.com/technetwork/java/index-138747.html>. Acesso em: 17 nov. 2017.

SCHIECK, Rodrigo. **Introdução ao PrimeFaces**. Disponível em: <http://www.devmedia.com.br/introducao-ao-primefaces/33139>. Acesso em: 15 out. 2017.

SPRING. **Spring Security**. Disponível em: <https://projects.spring.io/spring-security/>. Acesso em: 15 out. 2017.

STARUML. **StarUML**. Disponível em: <http://staruml.io/>. Acesso em: 15 out. 2017.