FACULDADE DE TECNOLOGIA DA PARAÍBA – FATEC/PB
SISTEMAS PARA INTERNET

VICTOR AMARAL FREITAS

CONTROLE DE COMPROVANTES: SISTEMA WEB DESENVOLVIDO COM O FRAMEWORK JSF

JOÃO PESSOA
2017

VICTOR AMARAL FREITAS

CONTROLE DE COMPROVANTES: SISTEMA WEB DESENVOLVIDO COM O FRAMEWORK JSF

Trabalho de conclusão do Curso de graduação, apresentado como base e parte dos requisitos para obtenção do Grau de Tecnólogo do Curso de Sistema para Internet da faculdade de Tecnologia da Paraíba-FATEC/PB.

Orientadora Profª Sheyla Natália de Medeiros

JOÃO PESSOA

2017

LISTA DE FIGURAS

LISTA DE SIGLAS

AE	Autorização Especial
AFE	Autorização de Funcionamento de Empresas
ANVISA	Agência Nacional de Vigilância Sanitária
API	*Application Programming Interface*
CDI	*Context and Dependency injection*
CRUD	*Create, Read, Update e Delete*
CSS	*Cascating Style Sheets*
DAO	*Data Access Object*
FTP	*File Transfer Protocol*
HTML	*Hypertext Markup Language*
HTTP	*Hypertext Transfer Protocol*
JCP	*Java Community Process*
JS	*JavaScript*
JSF	*JavaServer Faces*
MVC	*Model-View-Controller*
ORM	Mapeamento Objeto-Relacional
RDBMS	*Relational Database Management System*
RG	Registro Geral
SEBRAE	Serviço Brasileiro de Apoio às Micro e Pequenas Empresas
SQL	*Structured Query Language*
XHTML	*Xtensible Hypertext Markup Language*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Existem várias necessidades organizacionais, utilizar um sistema de controle interno é uma das maneiras de manter a empresa em constante crescimento. Apesar da grande necessidade de um controle nos processos, verifica-se que muitas empresas ainda não utilizam-se destes recursos de forma adequada.

No Brasil, temos como modelo controles fiscais que segundo o SEBRAE (Serviço Brasileiro de Apoio às Micro e Pequenas Empresas) "teve sua implementação gradual, iniciada em 2010, tornou-se obrigatória para a maioria das empresas [...]. O objetivo da Receita Federal é a modernização do procedimento, a diminuição de custos e o controle nos processos fiscais" (SEBRAE, 2017).

A emissão de Nota Fiscal eletrônica foi um marco para a padronização eletrônica e o uso de tecnologias entre empresas, além das próprias soluções internas que foram resolvidas com softwares completos, várias funções, diversas melhorias nos setores internos como financeiro, contabilidade, comercial, estoque e na aquisição de serviços de transporte.

Segundo a Agência Nacional de Vigilância Sanitária (ANVISA, 2017) "para se realizar o transporte de medicamentos, a empresa deve obter Autorização de Funcionamento de Empresa (AFE) junto à ANVISA. Caso entre os medicamentos transportados existam medicamentos que contenham substâncias sujeitas a controle especial, a transportadora, além de obter AFE, deverá também obter Autorização Especial (AE)." Deste modo, torna o transporte mais seguro e consequentemente mais burocrático. Contudo, o grande problema encontrado é em relação à comprovação da entrega, não existe um sistema para gerir o armazenamento nem a consulta desses comprovantes.

A logística do transporte muitas vezes é realizada por terceiros, são recolhidas as mercadorias junto com a nota fiscal, válido para o transporte e constatação de compra ou venda em postos fiscais. Além disso, nesse tipo de serviço é cobrado o canhoto para comprovação do recebimento total da mercadoria.

Entretanto, algumas transportadoras não realizam esse processo da maneira correta, o canhoto deve ser assinado com o nome e o RG (Registro Geral) de quem está recebendo a mercadoria, porém em alguns casos não são protocolados nem arquivados da maneira correta e consequentemente dificulta o controle da inadimplência, utilizam o comprovante de entrega para realizar cobranças de dívidas antigas.

Após análise da problemática, este processo pode ser melhorado através de um simples sistema que utiliza as seguintes ferramentas, Mysql que será utilizado para criação do

banco de dados que auxiliará no armazenamento das informações, JSF (*JavaServer Faces*) para criação das ações do sistema e o Bootstrap para criação do layout.

"Um sistema de fácil manutenção é um sistema em que as várias responsabilidades estão separadas e implementadas em locais bem-definidos." (LUCKOW e MELO, 2010, p. 185). Portanto, a divisão do sistema atenderá a perspectiva da arquitetura MVC (*Model, View, Controller*).

Com a utilização dos *frameworks* citados, é proposto o desenvolvimento de um sistema para gerenciar os comprovantes de entrega, à medida que forem entregues as mercadorias, serão anexadas às fotos dos canhotos junto com as informações de quem está recebendo as mercadorias em um sistema web. Para isso será necessário o empenho dos motoristas que realizam as entregas e a administração interna da transportadora em desenvolver métodos para aplicar a solução resolvendo o problema e disponibilizando as informações no sistema onde todos os clientes da transportadora terão acesso ao canhoto.

1.1 OBJETIVO GERAL

Desenvolver um sistema web, para controlar comprovantes de entrega de mercadoria, utilizando o *framework* JSF.

1.2 OBJETIVOS ESPECÍFICOS

- Levantar requisitos funcionais para criação de diagramas.
- Aplicar arquitetura MVC (*Model-View-Controller*) isolando as regras de negócios da lógica de apresentação.
- Integrar os padrões de projeto DAO (*Data Access Object*) e MVC diminuindo a dependência entre partes do sistema.
- Desenvolver cadastros com *framework* JSF integrado com Bootstrap.

1.3 METODOLOGIA

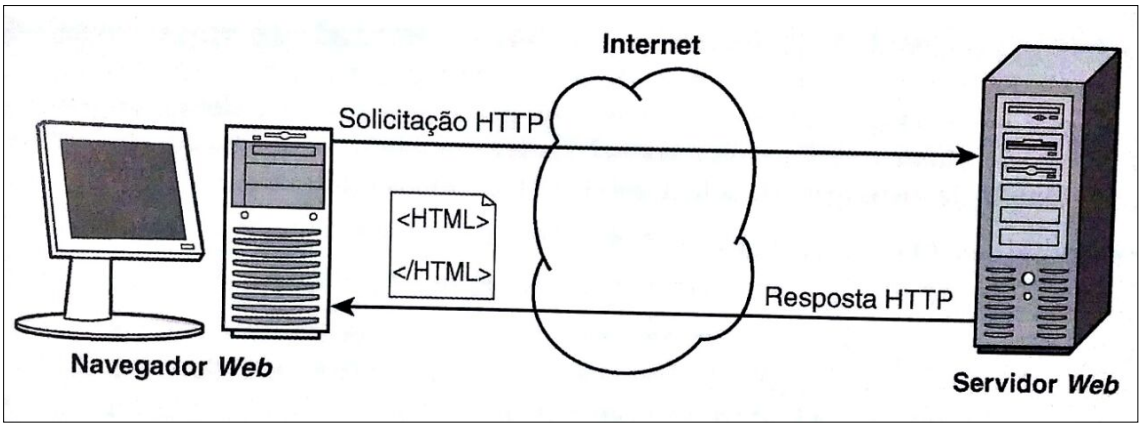
O trabalho será de natureza prática, descritiva e bibliográfica. Prática por ter a finalidade de produzir um sistema web como produto, descritiva por elaborar, registrar e produzir, bibliográfica por ter como base livros, artigos e revistas. Consiste em uma produção detalhada de um sistema baseado uma arquitetura MVC e os relatos ao decorrer do desenvolvimento com ênfase em sistemas existentes usando *JavaServer Faces*.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 APLICAÇÕES WEB

A Web utiliza o modelo cliente/servidor em grande escala, onde o cliente (um navegador Web ou um programa FTP (*File Transfer Protocol*)) se conecta ao servidor usando um protocolo. O mais comum desses protocolos é o HTTP (*Hypertext Transfer Protocol*), onde em uma requisição do *browser* é devolvido pelo servidor textos e imagens (GONÇALVES, 2007). O processo de solicitações e respostas pode ser visualizado pela Figura 1.

Figura 1 - Solicitação e resposta HTTP



Fonte: QIAN et al, 2010, p.3.

HTTP (Protocolo de Transferência de Hipertexto) é um protocolo sem estado que dá suporte a solicitações seguidas de repostas (padrão de troca de mensagens solicitação/resposta)[...] mensagens HTTP também são comumente trocadas entre servidores web ou outros aplicativos que não requeiram interação humana, como um navegador web (QIAN et al, 2010).

As aplicações webs funcionam através das requisições HTTP, conforme apresentado na figura 1, o *browser* envia solicitações HTTP para o servidor responsável pela verificação e responde o que foi requisitado pelo cliente.

2.2 JAVA

A linguagem Java surgiu em 1991 na Sun Microsystems. Inicialmente era parte de outro projeto, chamado Green Project, que tinha como objetivo possibilitar a convergência entre computador, equipamentos eletrônicos e eletrodomésticos. (LUCKOW e MELO, 2010).

Segundo Gonçalves (2007), aconteceram evoluções com as mudanças ocorridas no mundo, diversas implementações foram criadas, com intuito de abranger essas mudanças. Hoje é possível utilizar aplicativos *desktop*, páginas para a internet ou até mesmo aplicativos pequenos para celulares, todos desenvolvidos com a linguagem Java.

As principais características do Java são:

- Utilização do paradigma orientado a objetos
- Proximidade com a linguagem humana
- Fortemente tipada
- Estruturada, segura, dinâmica e robusta.
- Variedades de bibliotecas para facilitar o desenvolvimento de sistemas

A existência de *frameworks* Java garante o desenvolvimento de maneira produtiva e possibilitam a junção das principais características da linguagem Java, como o *JavaServer Faces.*

2.3 *JAVASERVER FACES*

O *JavaServer Faces* (JSF) é a especificação para um *framework* de componentes para desenvolvimento web em Java. Essa especificação foi definida por meio da *Java Community Process* (JCP), que é a entidade que tem como objetivo especificar a evolução da linguagem Java de acordo com o interesse do mercado e não apenas da Sun, criadora da linguagem Java. (LUCKOW e MELO, 2010).

JSF é o *framework* Java mais utilizado para o desenvolvimento de aplicações web. Alguns dos principais motivos para isso são: oferece um excelente conjunto de funcionalidades, possui bons materiais de estudo e exige pouco conhecimento inicial para construção de interfaces de usuários tradicionais (FRANCO, 2011).

Para desenvolver usando JSF existem quatro passos básicos:

- Criar classe *Backing Bean*.
- Mapear a classe *Bean*.
- Criar página JSF.
- Realizar o mapeamento entre páginas.

Alguns aspectos do projeto melhoram com a utilização do *framework JavaServer Faces*, maximiza a produtividade, minimiza a complexidade de manutenção e proporciona melhor integração com outras tecnologias web.

2.3.1 Classe *Backing Bean*

A classe Bean (conhecida também como *Becking Bean*) é uma classe Java normal, que suportará todo o funcionamento das páginas JSF. Isso permite que você desenvolva seu projeto separando as regras de funcionamento da organização visual da página e do layout (LUCKOW e MELO, 2010).

A classe Bean fornecerá as informações que serão exibidas na página e as operações que serão executadas, mas para que isso ocorra é necessário realizar o mapeamento no arquivo *face-config.xml* ou usando *Annotations*. O exemplo do mapeamento no *faces-config.xml* pode ser visualizado pela Figura 2.

Figura 2 - Elementos *managed-bean*

```
<managed-bean>
    <managed-bean-name>
        MeuBean
    </managed-bean-name>
    <managed-bean-class>
        meupacote.MeuBean
    </managed-bean-class>
    <managed-bean-scope>
        session
    </managed-bean-scope>
</managed-bean>
```

Fonte: GONÇALVES, 2007, p.451.

O mapeamento visualizado na Figura 2 é realizando usando o elemento *managed-bean-name* onde é atribuído um nome, *managed-bean-class* faz referência ao nome da classe instânciada e *managed-bean-scope* define o tempo de vida da instância da classe criada.

O mapeamento via *annotations* dispensa o uso do arquivo *faces-config.xml*, sendo feito totalmente com declarações na própria classe Java. No lugar do elemento *managed-*

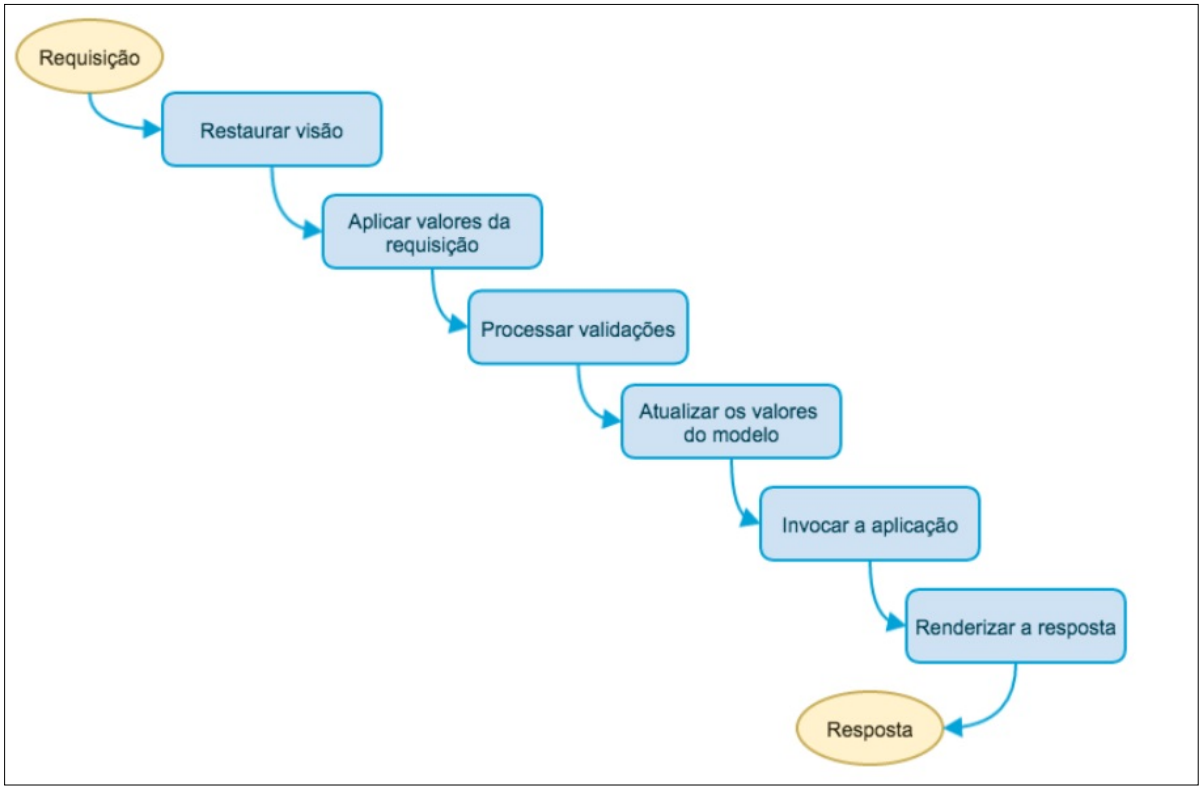
bean-name do arquivo xml, utilizará a *annotation @managedBean* e no lugar do elemento *managed-bean-scope*, utilizará a *annotation @ResquestScoped*.

2.3.2 Páginas *JavaServer Faces*

A construção de páginas com o JSF apresenta uma diferença notável com outros *frameworks* para aplicações em web. O JSF é baseado em componentes, ou seja, ele mantém uma estrutura por trás da tela, e apresenta apenas o código HTML, porém o desenvolvedor não precisa se preocupar a conversão de uma página JSF, o XHTML, para HTML fica a cargo do *framework* (FREIRE, 2014).

Segundo Gonçalves (2007), cada componente pode ser associado com os métodos e atributos de um *bean*, cada componente também podem ser associados com uma função de validação ou classe, sendo assim um ciclo de vida do JSF é composto por várias fases visualizadas na Figura 3.

Figura 3 - 6 fases do ciclo de vida JSF



Fonte: FARIA, 2015, p.68.

Conforme Gonçalves (2008) o ciclo de vida é composto de 6 fases:

1 Restaurar a apresentação' — Essa fase inicia o processamento da requisição do ciclo de vida por meio da construção da árvore de componentes do JSF. Cada árvore de componentes possui um identificador único durante todo o aplicativo. O JSF constrói a

apresentação da página e salva na instância *Faces-Context* para processamento das fases seguintes.

2 Aplicar os valores de requisição — Nessa fase, quaisquer novos valores inseridos são extraídos e armazenados por seus apropriados componentes. Se o valor do componente não for uma *string*, então ele é convertido para o seu determinado tipo. Se a conversão falhar, ocorrem diversas situações:

a) Uma mensagem de erro é gerada e associada com o componente;

b) Uma mensagem de erro é armazenada no *FacesContext* e depois mostrada posteriormente na fase de Renderizar a Resposta;

c) O ciclo de vida pula a fase de Renderizar a Resposta quando esta se completou.

3 Processar validações — Depois do valor de cada componente ser atualizado, na fase de processo de validações, os componentes serão validados, se necessário. Um componente que necessita de validação deve fornecer implementação da lógica de validação. Por exemplo, em um carrinho de compras, você pode determinar a quantidade mínima a ser digitada e máxima. O valor requisitado é um inteiro, e como passou pela fase 2, nessa fase pode ser barrado por estar além do determinado.

4 Atualizar valores do Modelo — Alcança-se essa fase após todos os componentes serem validados. Nessa fase são atualizados os dados do modelo do aplicativo. Por exemplo, na página que criou para enviar o nome, o que você digitou foi armazenado no bean MeuBean durante esta fase. Por ter passado pelo processo de validação, o valor armazenado será garantido nessa fase. Entretanto, os dados podem violar a lógica de negócios, ao qual a validação ocorre na fase seguinte.

5 Invocar a aplicação — Durante esta fase, a implementação JSF manipula quaisquer eventos do aplicativo, tal como enviar um formulário ou ir a outra página através de um link. Esses eventos são ações que retornam geralmente urna string que está associada a navegação, criada pelo arquivo *faces-config*.xml, no qual se encarrega de chamar a página determinada. Foi o que ocorre quando você digita o nome correto e a página seguinte foi exibida.

6 Renderizar a resposta — Essa é a fase final, na qual é renderizada a página. Se esse é um pedido inicial para esta página, os componentes são acrescentados à apresentação neste momento. Se esse é um *postback*, os componentes já foram acrescidos à apresentação. Se há mensagens de conversão ou erros de validação e a página contém um ou mais componentes <*mensage*/> ou um componente <*messages*/>, esses serão exibidos. Reciprocamente, se a página não contém um componente de mensagem, nenhuma informação aparecerá.

2.4 PADRÕES DE PROJETO

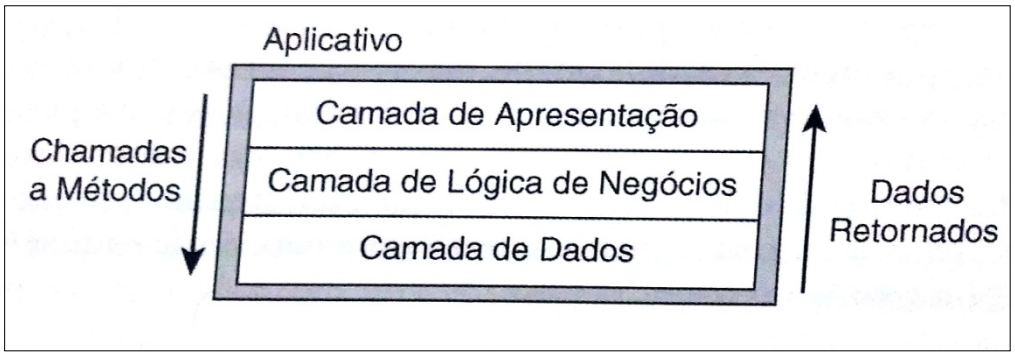
Padrões de projeto são soluções para problemas recorrentes no desenvolvimento de software e que podem ser reusadas por outros desenvolvedores (FRANCO, 2011). Desta forma, os padrões melhoram o acoplamento e a coesão do projeto de software.

2.4.1 Arquitetura MVC

O MVC (*Model – View – Controller*) é um padrão de arquitetura muito utilizado no desenvolvimento de projeto web, mas pode ser aplicado também a qualquer outro tipo de projeto de software que exige integração com usuário (LUCKOW e MELO, 2010).

O *JavaServer Faces* (JSF) é um *framework* para desenvolvimento de aplicações web. É baseado no padrão MVC (*Model-View-Controller*), dividindo em três camadas, separando a lógica de negócio, a apresentação e a persistência dos dados (SILVA et al, 2012). Na Figura 4 mostra a representação do funcionamento do MVC.

Figura 4 - Camadas MVC (Model – View – Controller)



Fonte: QIAN el al, 2010, p.4.

A idéia central do MVC é dividir o programa em três conjuntos, cada um com sua respectiva responsabilidade. Assim, a camada de Visão seria a responsável pela apresentação dos dados. O Controle seria incumbido de receber os dados inseridos pelo usuário, manipulá-los utilizando-se da camada de Modelo, e retornar alguma informação para a camada de Visão. Já o Modelo teria a função de definir o modo como os dados são organizados no meio persistente (MARAFON, 2006).

O padrão MVC possibilita a existência de várias interfaces com o usuário que permitem ser modificadas sem que haja a necessidade da alteração das regras de negócios, possibilitando uma maior flexibilidade.

2.4.2 *Data Access Object*

Segundo Gonçalves (2007), o padrão de projeto DAO (*Data Access Object*) é o padrão mais utilizado para acesso a dados contidos em um banco de dados. O padrão DAO fornece uma interface independente, no qual pode ser usado para persistir objetos de dados.

O componente de negócio que depende do DAO usa a interface mais simples exposta pelo DAO para seus clientes. O DAO oculta completamente os detalhes de implementação da persistência dos dados de seus clientes. Porque a interface exposta pelo DAO para clientes não se altera quando a implementação do código é modificada, este padrão permite que o DAO se adapte a diferentes regimes de armazenamento, sem afetar seus clientes ou os componentes de negócio (OLIVEIRA, 2009). Essencialmente, o DAO funciona como um adaptador entre o componente e a fonte de dados.

2.5 HIBERNATE

O *Hibernate* é um projeto que tem como finalidade ter uma solução completa para o problema de gerenciamento de dados persistidos em Java. Além de ser um *framework* que serve como mediador no relacionamento com banco de dados, relacionamento conhecido como mapeamento objeto/relacional (ORM) para a linguagem Java (GONÇALVES, 2008). A Figura 5 demonstra duas técnicas de mapeamento através do XML ou fazendo uso de *Annotations*.

Figura 5 - Arquitetura simplificada do *Hibernate*

Fonte: LUCKOW e MELO, 2010, p.123.

Hibernate oferece várias ferramentas que ajudam a persistir de forma automatizada e transparente entre elas o ORM (Mapeamento Objeto-Relacional).

2.5.1 Mapeamento Objeto- Relacional

ORM (Mapeamento Objeto-Relacional) é responsável por fazer a transformação de dados de maneira reversível. Contudo, utilizando essas decisões sobre a arquitetura de um sistema, técnica de mapeamento tem como consequência uma possível queda de desempenho. Porem existe mais benefícios que prejuízos (LUCKOW e MELO, 2010, p.121).

Luckow e Melo (2010, p.121) explicam que uma solução ORM contém os seguintes pontos:

- Uma API para realizar as operações CRUD básicas em objetos de classes persistentes.
- Uma Linguagem ou API para especificar consultas que se referem às classes persistentes.
- Facilidade de especificar o metadado de mapeamento.
- Uma técnica para que a implementação ORM interaja com objetos transacionais permitindo executar verificações do tipo leitura suja ou carregamento sobre demanda (LUCKOW e MELO, 2010, p.121).

Portanto, o *Hibernate* apresenta todas essas características que o torna uma aplicação ORM.

2.6 MYSQL

MySQL é um banco de dados relacional, desenvolvido para plataformas Linux *like*, OS/2, Windows. Sendo um software de livre distribuição para plataformas não-Windows que o utilizam em um servidor Web. MySQL é um servidor multiusuário, multitarefa, compatível com o padrão SQL (*Structured Query language* Linguagem de Consulta estruturada), linguagem essa amplamente utilizada para manipulação de dados em RDBMS (*Relational Database Management System*), sendo considerada um ferramenta de manipulação de base de dados de tamanho moderado(ORACLE, 2017).

As principais características que destacam MySQL são: seu suporte as instruções SQL; sua natureza de distribuição gratuita; facilidade de integração com servidor Web e linguagens de programação de desenvolvimento de sites dinâmicos.

2.7 BOOTSTRAP

O Bootstrap é um conjunto de ferramentas de código aberto para desenvolvimento com HTML (*Hypertext Markup Language*), CSS (*Cascating Style Sheets*) e JS (*JavaScript*) que possibilita projetar ideias de vários estilos ou cria um aplicativo completo com *templates* prontos, sistema de grade responsivo, extensos componentes pré-construídos e *plugins* poderosos criados no jQuery (BOOTSTRAP, 2017).

A utilização de classes disponível no Bootstrap facilita na organização dos aspectos visuais de *tags* de demarcação (HTML), possibilitando uma qualidade visual no *front-end* das aplicações web e consequentemente o tempo de criação dos layouts será reduzido.

3 SISTEMA WEB DE COMPROVANTES DE ENTREGA

A preparação para desenvolvimento do sistema feita a partir de diagramas de casos de uso baseados no levantamento de requisitos, estrutura formada pela arquitetura MVC, construção do sistema utilizando o *framework* JSF integrado com Bootstrap e o Hibernate para persistência de dados no MySql.

3.1 LEVANTAMENTO DE REQUISITOS FUNCIONAIS

O sistema web é composto por um administrador que realiza consultas e armazenamentos de canhotos no sistema de acordo com os requisitos funcionais apresentados na Figura 6.

Figura 6 - Requisitos funcionais

Identificação	Nome	Descrição
RF01	Efetuar login	Efetua login, acesso ao sistema.
RF02	Cadastrar usuário	Cadastrar usuários que administraram o sistema
RF03	Cadastrar empresa	Cadastrar as empresas que terceirizam as entregas de suas mercadorias
RF04	Alterar dados da empresa	Alterar qualquer dado da empresa que já foi cadastrada.
RF05	Excluir empresa	Apagar a empresa por completo.
RF06	Cadastrar canhoto	Cadastrar canhoto de cada nota fiscal que foi entregue com o nome do destinatário que recebeu a mercadoria e o número da nota fiscal.
RF07	Alterar canhoto	Alterar qualquer dado do canhoto que já foi cadastrado.

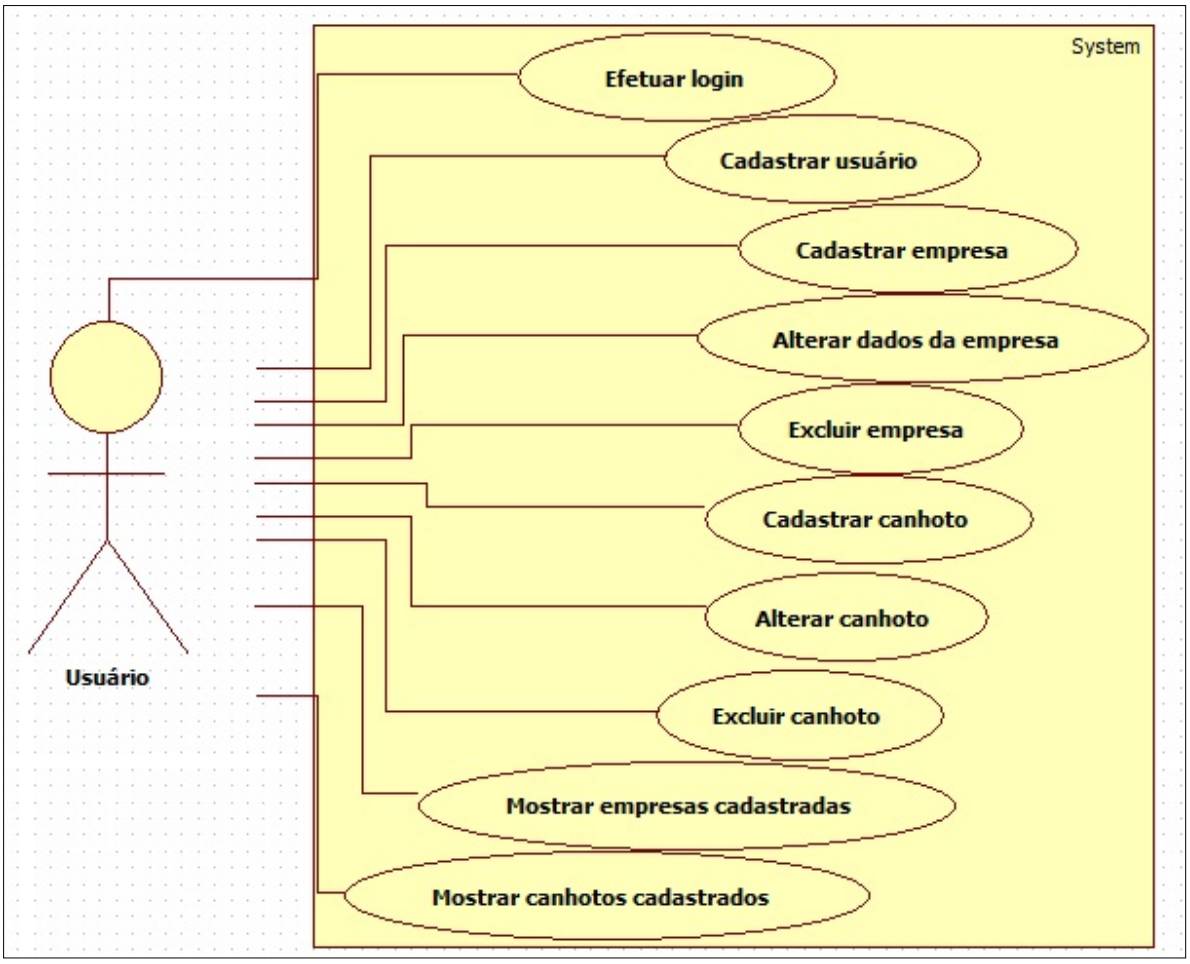
RF08	Excluir canhoto	Apagar canhoto cadastrado
RF09	Mostrar empresas cadastradas	Mostrar uma lista de todas as empresas cadastradas.
RF10	Mostrar canhotos cadastrados	Mostrar uma lista de todos os canhotos cadastrados.

Fonte: Próprio Autor.

3.2 DIAGRAMA DE CASO DE USO

O sistema possui um usuário administrador que é responsável pelos cadastros de empresa, canhoto e usuários e pela alteração e exclusão de todos os cadastros.

Figura 7 - Diagrama de caso de uso

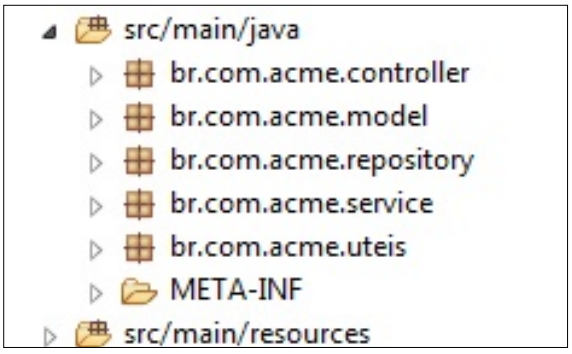


Fonte: Próprio Autor.

3.3 IMPLEMENTAÇÃO DO SISTEMA

No início da implementação foram criados pacotes para dividir e implementar os padrões de projeto abordados nos tópicos 2.4.1, 2.4.2, 2.4.3. A divisão dos pacotes caracteriza partes do padrão MVC, conforme a Figura 8.

Figura 8 - Pacotes do sistema



Fonte: Próprio Autor.

A camada modelo do padrão MVC foi subdividida em dois pacotes: *br.com.acme.repository* e *br.com.acme.service* responsável pelo diálogo entre a camada de controle e a própria camada. O pacote *br.com.acme.model* contém as entidades que possui atributos, conforme apresentado na Figura 9 uma das entidades implementadas.

Figura 9 - Entidade usuário

```
@Table(name="tb_usuario")
@Entity
public class UsuarioEntity implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue
    @Column(name="id_usuario")
    private String codigo;

    @Column(name="ds_login")
    private String usuario;

    @Column(name="ds_senha")
    private String senha;
```

```
    public String getCodigo() {
        return codigo;
    }
    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }
    public String getUsuario() {
        return usuario;
    }
    public void setUsuario(String usuario) {
        this.usuario = usuario;
    }
    public String getSenha() {
        return senha;
    }
    public void setSenha(String senha) {
        this.senha = senha;
    }

}
```

Fonte: Próprio Autor.

Na Figura 9 estão presentes as anotações do *Hibernate* que servem para mapear as classes no banco de dados Mysql e auxiliar a construção, organização e manutenção dos dados.

Além do mapeamento cada entidade possui um repositório ou classe DAO que se comunica diretamente com o banco de dados Mysql, conforme Figura 10.

Figura 10 - Classe empresaDAO

```
public class EmpresaDAO implements Serializable {

    private static final long serialVersionUID = 1344324234243234L;
    @Inject
    Empresa empresa;
    EntityManager entityManager;
    public boolean gravar(Empresa empresa){
        boolean sucesso = false;
        try {
        entityManager =  Uteis.JpaEntityManager();
        entityManager.merge(empresa);
            sucesso = true;
        } catch (Exception e) {
            e.printStackTrace();
        }
        return sucesso;
    }
```

```
    public Empresa selecionar(Long codigo){
        Empresa empresa = null;
        try {
            entityManager =  Uteis.JpaEntityManager();
            empresa = entityManager.find(Empresa.class, codigo);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return empresa;
    }
    public boolean remover(Empresa empresa){
        boolean sucesso = false;
        try {
            entityManager =  Uteis.JpaEntityManager();
            empresa = entityManager.find(Empresa.class, empresa.getCodigo());

            entityManager.remove(empresa);
```

Fonte: Próprio Autor.

As classes DAO foram implementadas de acordo com o padrão DAO fundamentado no tópico 2.4.2. Todas as classes DAO possuem uma instância da fábrica que gerencia as conexões JDBC e suas transações, contém métodos para realização de CRUDs (*Create, Read, Update e Delete*) criação, consulta, atualização e destruição de dados.

A camada de controle está presente no pacote *br.com.acme.controller* na figura 8, dentro possui as classes que controlam toda as ações solicitadas pela camada de visão e informa para a camada modelo quais os dados devem ser verificados ou persistidos no banco de dados, conforme apresentado na arquitetura MVC. A implementação da classe de controle está contida na Figura 11.

Figura 11 - Classe empresaBean

```
@Named(value = "empresaBean")
@SessionScoped
public class EmpresaBean implements Serializable {

    private static final long serialVersionUID = 14324242L;

    @Inject
    private EmpresaDAO empresaDAO;
    private transient Empresa empresa;
    private CanhotoDAO canhotoDAO;
    private transient Canhoto canhoto;

    private transient List<Empresa> empresas;

    private transient List<Canhoto> canhotos;

    private Part file;

    public String novo(){
        empresa = new Empresa();
        return "empresa.xhtml";
    }
    @PostConstruct
```

```
    protected void init() {
        empresaDAO = new EmpresaDAO();
        empresa = new Empresa();
        canhoto = new Canhoto();
        canhotoDAO = new CanhotoDAO();
        this.canhotos= new ArrayList<>();

    }
    public String gravar() {
        boolean resultado = empresaDAO.gravar(empresa);
        if (resultado) {
            empresa = new Empresa();
            FacesContext.getCurrentInstance().addMessage("", new FacesMessage("Sucesso ,Empresa Cadastrado!"));
        } else {
            FacesContext.getCurrentInstance().addMessage("", new FacesMessage("ERRO!"));
        }
```

Fonte: Próprio Autor.

Nas classes de controle existem anotações CDI(*Context and Dependency injection*) como *@Named* caracterizado como uma classe *Becking Bean e @Inject* que é responsável pela injeção de dependências. As requisições são disparadas pela camada de visão representada pela Figura 12.

Figura 12 - Formulário de cadastro

```
            <h3 class="sub-header">Cadastro de Empresa</h3>
<div class="container-fluid" id="dadosEmpresa">
    <section class="container">
        <div class="container-page">
            <div class="col-md-10">
                <h:form  role="form">
                   <h:messages class="alert alert-success" role="alert" />

                        <div class="form-group col-lg-8">
                            <h:outputLabel value="Razão Social:"/>
                            <h:inputText value="#{empresaBean.empresa.razao}"  class="form-control" placeholder="Email address"/>
                            <br></br>
                        </div>

                        <div class="form-group col-lg-5">
                            <h:outputLabel value="CNPJ:"/>
                            <h:inputText value="#{empresaBean.empresa.cnpj}"  class="form-control" placeholder="Idade"/>
                            <br></br>
                        </div>

                        <h:commandLink  action="#{empresaBean.gravar}" class="btn btn-primary pull-right" value="Salvar" update="dadosEmpresa"/>

                </h:form>
```

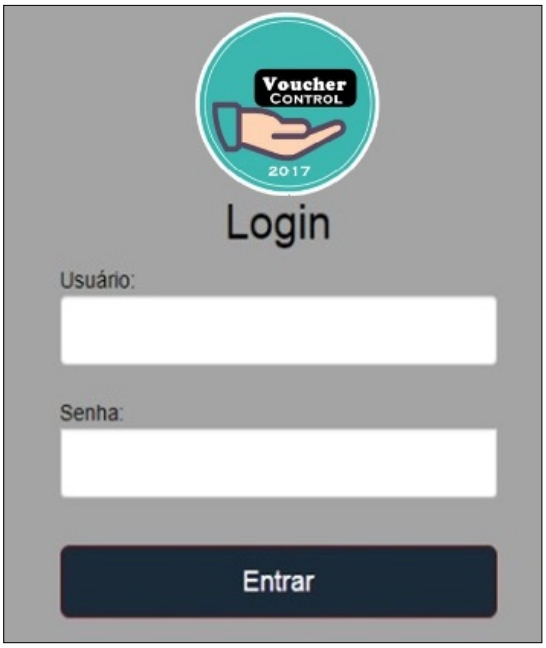
Fonte: Próprio Autor.

Na camada de visão, conforme a Figura 12 estão presentes as classes Bootstrap e *tags* JSF que serão apresentadas no *front-end* da aplicação web.

3.4 APRESENTAÇÃO DO SISTEMA

A tela inicial da aplicação é a tela de login, conforme a Figura 13, a partir dessa tela é realizado o acesso ao sistema.

Figura 13 - Tela de login



Fonte: Próprio Autor.

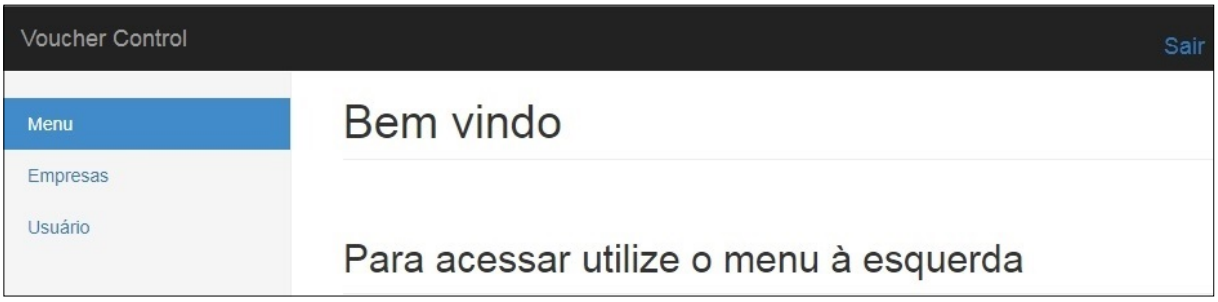
Para construção das páginas foi utilizado JSF, HTML e *Bootstrap.* Ao tentar efetuar o acesso ao sistema, é realizada uma verificação dos dados informados, caso sejam inválidos é apresentada uma mensagem de erro, conforme a Figura 14.

Figura 14 - Login inválido

Fonte: Próprio Autor.

Ao efetuar o acesso com sucesso (login e senha válidos), o usuário é encaminhado para a tela de apresentação do sistema, que é exibida pela Figura 15. O menu na lateral esquerda apresenta as opções (funcionalidades) para o usuário que está logado.

Figura 15 - Tela inicial



Fonte: Próprio Autor.

O sistema é composto por um menu lateral esquerdo onde estão localizados as opções de cadastros e o setor central que contém o conteúdo da página. Para realização de um cadastro de usuário é necessário escolher a opção no menu esquerdo, conforme a Figura 16.

Figura 16 - Página de cadastro de usuário

Fonte: Próprio Autor.

O cadastro de novos usuários é de responsabilidade do usuário administrador do sistema. Cada novo cadastro realizado terá um usuário e senha com acesso total ao sistema. A página de empresas contém a opção de cadastrar novas empresas e cada empresa cadastrada será exibida na mesma página, conforme visto nas Figuras 17,18 e 19.

Figura 17 - Página de empresas

Fonte: Próprio Autor.

Figura 18 - Página de cadastro de empresa

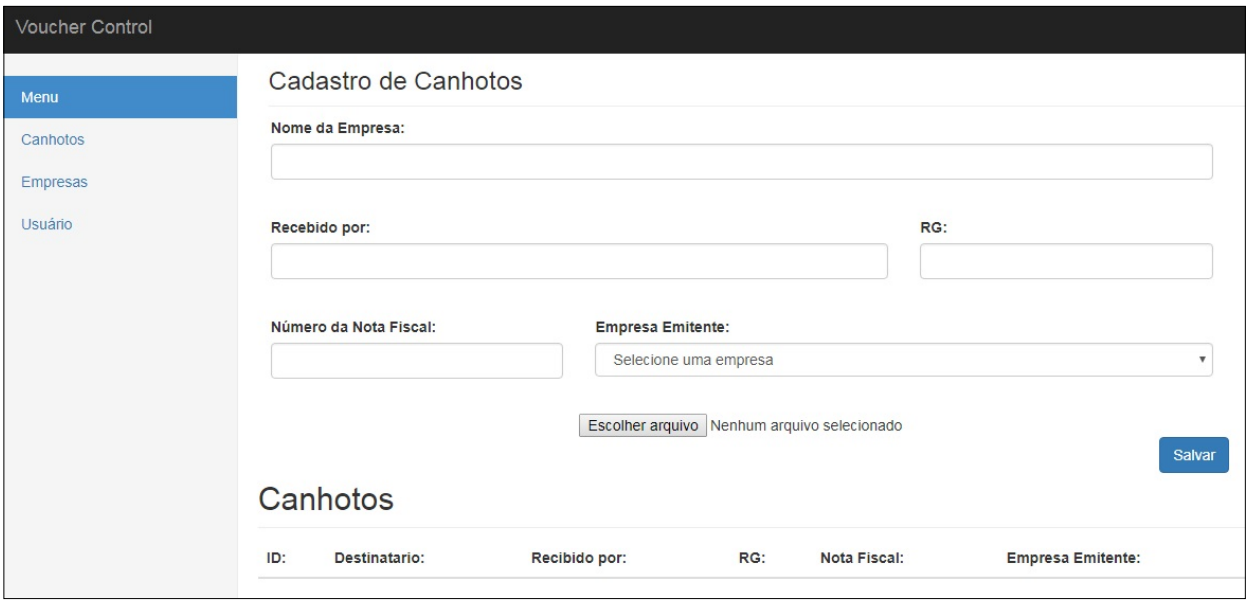
Fonte: Próprio Autor.

Figura 19 - Página de empresas com uma empresa cadastrada

Fonte: Próprio Autor.

As empresas que serão cadastradas são aquelas que iram emitir nota fiscal e possuem entregas terceirizadas. Já os usuários podem ser funcionários da empresa ou transportadora. Na Figura 19 as empresas que estão cadastradas contém três botões, são eles: editar, excluir e canhotos, que serão cadastrados, conforme apresentado pela Figura 20.

Figura 20 - Página de canhotos



Fonte: Próprio Autor.

Na Figura 20, o canhoto será cadastrado e anexada uma foto do comprovante de entrega, além de preencher os campos com o nome de quem recebeu a mercadoria e o número da nota fiscal que o canhoto se refere. A foto anexada será salva em uma pasta dentro do sistema para fins de consulta e comprovação que a mercadoria foi entregue. Dessa forma os canhotos podem ser armazenados e visualizados, conforme a Figura 21.

Figura 21 - Canhotos cadastrados

Fonte: Próprio Autor.

4 CONCLUSÃO

O processo de desenvolvimento do *software*, conta com o auxílio de ferramentas como *framework* JSF, *hibernate, Mysql e bootstrap* que facilitaram a criação do sistema web para controle interno dos comprovantes de entrega. A aplicação foi desenvolvida para plataforma web e de grade responsiva, que possibilita armazenar os comprovantes de entrega a partir de qualquer dispositivo com acesso a internet.

O *software* possibilita a melhora no processo de busca pelos comprovantes de entrega, descarta a procura e o armazenamento dos canhotos impressos, reduz o tempo de separação e entrega dos canhotos de posse das transportadoras para as empresas que contratam o serviço de transporte.

Durante o desenvolvimento, dificuldades surgiram na integração do framework JSF com o *bootstrap*. São poucos os projetos que trabalham essa integração pelo fato de existir uma biblioteca de componentes, *PrimeFaces* específica para trabalhar com JSF. Contudo, alguns dos problemas, como o *upload* de arquivo, foram solucionados na versão 2.2 do JSF.

Como trabalhos futuros, pretende-se aprimorar o armazenamento de arquivos, implementar campo de busca avançado que possibilitara a filtragem dos canhotos armazenados, visualização das fotos armazenadas direto no sistema e envio por email da relação das mercadorias entregues.

5 REFERÊNCIAS

ANVISA – **Agência Nacional de Vigilância Sanitária**. Disponível em: < http://portal.anvisa.gov.br/registros-e-autorizacoes/empresas/autorizacao-de-funcionamento/distribuidora-importadora-transportadora> Acessado em 01/09/2017.

BOOTSTRAP, **bootstrap**. 2017. Disponível em: <getbootstrap.com> Acessado em 10/09/2017.

FARIA, T. **Java EE 7 com JSF, PrimeFaces e CDI.** 2015 Disponível em < http://s3.amazonaws.com/algaworks-assets/ebooks/algaworks-ebook-java-ee-7-com-jsf-primefaces-e-cdi-2a-edicao-20150228.pdf> Acessado em 20/09/2017.

FRANCO, T.S.R. **Estudo comparativo entre frameworks java para desenvolvimento de aplicações web: jsf 2.0, grails e spring web mvc.** Curitiba, 2011. Disponível em <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/492/1/CT_JAVA_VI_2010_16.PDF >Acessado em 02/09/2017.

FREIRE, S.M. **Ferramenta de apoio a auditoria de programa de segurança de software.** Brasília, 2014. Disponível em <https://fga.unb.br/articles/0000/5474/TCC_Matheus_Freire.pdf> Acessado em 05/09/2017.

GONÇALVES, E. **Desenvolvendo aplicações Web com JSP, Servlets, JavaServer Faces, Hibernate, EJB 3 Persistence e Ajax.** Rio de Janeiro: Ciência Moderna, 2007.

GONÇALVES, E. **Dominando Java Server Faces e Facelets utilizando Spring 2.5, hibernate e JPA.** Rio de Janeiro: Ciência Moderna, 2008.

LUCKOW, D. H; MELO, A. A. **Programação Java para a web:** aprenda a desenvolver uma aplicação financeira pessoal com as ferramentas mais modernas da plataforma Java. São Paulo:Novatec, 2010.

MARAFON, L. D. **Integração javaserver faces e ajax estudo da integração entre as tecnologias jsf e Ajax.** Florianópolis, 2006. Disponível em < https://projetos.inf.ufsc.br/arquivos_projetos/projeto_491/TCC%20-%20Diego%20Luiz%20Marafon.pdf> Acessado em 05/09/2017.

OLIVEIRA, R. R. **Desenvolvimento de um sistema de informação web utilizando padrões de projeto e jsf para um laboratório de análises físico-químicas.** Minas Gerais, 2009. Disponível em < http://repositorio.ufla.br/bitstream/1/5232/1/MONOGRAFIA_Desenvolvimento_de_um_sistema_de_informacao_WEB_utilizando_padroes_deProjeto_e_JSF_para_um_laboratorio_de_analises_fisico-quimicas.pdf> Acessado em 06/09/2017

ORACLE, **ORACLE CORPORATION**, Why MySQL. [s.l. : s.n.], 2011. Disponível em: http://www.mysql.com/why-mysql. Acessado em: 10/09/2017.

QIAN, K, ET AL. **Desenvolvimento Web Java.** Rio de Janeiro:Grupogen LTC, 2010.

SEBRAE NACIONAL. **Nota Fiscal eletrônica:** tudo o que você precisa saber, 2017. Disponível em: <https://www.sebrae.com.br/sites/PortalSebrae/artigos/nota-fiscal-eletronica-tudo-o-que-voce-precisa saber,b3310d49ac0f3510VgnVCM1000004c00210aRCRD> Acessado em: 27/08/2017.

SILVA, R. A,GEOFFROY, R, SILVA, N. N. **Sistema web para biblioteca da empresa Marluvas Calçados de Segurança.** MINAS GERAIS, 2012. Disponível em < http://www.unipac.br/site/bb/tcc/tcc-5e4e08c0a81fbfc69b57b187475a81a2.pdf> Acessado em 05/09/2017

ANEXO A - EXEMPLOS JAVASERVER FACES + BOOTSTRAP

Classe usuário

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

@Table(name="tb_usuario")
@Entity
@NamedQuery(name = "UsuarioEntity.findUser",
query= "SELECT u FROM UsuarioEntity u WHERE u.usuario = :usuario AND u.senha =
:senha")
public class UsuarioEntity implements Serializable {

	private static final long serialVersionUID = 1L;

	@Id
	@GeneratedValue
	@Column(name="id_usuario")
	private String codigo;

	@Column(name="ds_login")
	private String usuario;

	@Column(name="ds_senha")
	private String senha;

	public String getCodigo() {
		return codigo;
	}
	public void setCodigo(String codigo) {
		this.codigo = codigo;
	}
	public String getUsuario() {
		return usuario;
	}
	public void setUsuario(String usuario) {
		this.usuario = usuario;
	}
	public String getSenha() {
		return senha;
	}
	public void setSenha(String senha) {
		this.senha = senha;
	}

	}
```

Classe usuário DAO

```
package br.com.acme.repository;

import java.io.Serializable;
```

```
import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import br.com.acme.model.UsuarioEntity;
import br.com.acme.uteis.Uteis;

public class UsuarioRepository implements Serializable {

	private static final long serialVersionUID = 1L;

	@Inject
	UsuarioEntity usuario;

	EntityManager entityManager;

	public boolean gravar(UsuarioEntity usuario){
	  boolean sucesso = false;
	  try {
	  entityManager =  Uteis.JpaEntityManager();
	  entityManager.merge(usuario);
	      sucesso = true;
	  } catch (Exception e) {
	      e.printStackTrace();
	  }

	  return sucesso;
    }

	public UsuarioEntity ValidaUsuario(String usuario, String senha){

		try {
			//QUERY QUE VAI SER EXECUTADA (UsuarioEntity.findUser)
			Query query =
Uteis.JpaEntityManager().createNamedQuery("UsuarioEntity.findUser");
			//PARAMETROS DA QUERY
			query.setParameter("usuario", usuario);
			query.setParameter("senha", senha);
			//RETORNA O USUARIO SE FOR LOCALIZADO
			return (UsuarioEntity)query.getSingleResult();
		} catch (Exception e) {
			return null;
		}



	}
}
```

Classe sessão Controller

```
import java.io.Serializable;
import javax.annotation.PostConstruct;
import javax.enterprise.context.SessionScoped;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.inject.Inject;
import javax.inject.Named;
import javax.servlet.http.HttpSession;
import br.com.acme.model.UsuarioEntity;
import br.com.acme.repository.UsuarioRepository;
```

```
@Named
@SessionScoped
public class SessaoMB implements Serializable {

	private static final long serialVersionUID = -1683346421712151629L;

	@Inject
	private UsuarioEntity usuarioEntity;
	private UsuarioEntity usuario;

	@Inject
	private UsuarioRepository usuarioRepository;

    @PostConstruct
    protected void init() {
       usuarioEntity = new UsuarioEntity();
       usuario = new UsuarioEntity();
    }
    public UsuarioEntity getUsuarioEntity() {
		return usuarioEntity;
	}
	public void setUsuarioEntity(UsuarioEntity usuarioEntity) {
		this.usuarioEntity= usuarioEntity;
	}
	public UsuarioEntity getUsuario() {
		return usuario;
	}
	public void setUsuario(UsuarioEntity usuario) {
		this.usuario = usuario;
	}
	public UsuarioEntity GetUsuarioSession(){

		FacesContext facesContext = FacesContext.getCurrentInstance();

		return
(UsuarioEntity)facesContext.getExternalContext().getSessionMap().get("usuarioAuten
ticado");
	}
	public void gravar() {

	  boolean resultado = usuarioRepository.gravar(usuario);

	  if (resultado) {
	      usuario = new UsuarioEntity();
	      FacesContext.getCurrentInstance().addMessage("", new
FacesMessage("Usuário cadastrado com sucesso!"));
	  } else {
	      FacesContext.getCurrentInstance().addMessage("", new
FacesMessage("ERRO"));
	  }
    }
	public String login(){

		usuarioEntity =
usuarioRepository.ValidaUsuario(usuarioEntity.getUsuario(),usuarioEntity.getSenha(
));
```

```
            if(usuarioEntity == null){
                   usuarioEntity = new UsuarioEntity();
                  FacesContext.getCurrentInstance().addMessage(
                   null,
                   new FacesMessage(FacesMessage.SEVERITY_ERROR, "Usuário não
encontrado!",
                               "Erro no Login!"));
                  return null;
            }
            else{
                  usuarioEntity.setCodigo(usuarioEntity.getCodigo());
                  FacesContext facesContext = FacesContext.getCurrentInstance();

      facesContext.getExternalContext().getSessionMap().put("usuarioAutenticado",
usuarioEntity);
                  return "paginas/index.xhtml?faces-redirect=true";

            }

      }
      public String logout(){
            HttpSession sessao = (HttpSession)
FacesContext.getCurrentInstance().getExternalContext().getSession(true);
            sessao.invalidate();
            return "/login.xhtml";
      }
}
```

Classe Empresa

```
import java.util.List;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
import javax.persistence.Table;

@Entity
@Table(name = "Empresa")

public class empresa {

      @Id
      @GeneratedValue(strategy = GenerationType.AUTO)
      private Long codigo;

      @Column
      private String razao;
      @Column
      private String cnpj;
      @Column
      private String telefone;
      @Column
      private String inscricao;
```

```
	@OneToMany(mappedBy = "empresa", targetEntity = Canhoto.class, fetch =
FetchType.LAZY, cascade = CascadeType.ALL)
	private List<Canhoto> canhotos;

	public List<Canhoto> getCanhotos() {
		return canhotos;
	}

	public void setCanhotos(List<Canhoto> canhotos) {
		this.canhotos = canhotos;
	}

	public Long getCodigo() {
		return codigo;
	}

	public String getTelefone() {
		return telefone;
	}

	public void setTelefone(String telefone) {
		this.telefone = telefone;
	}

	public String getInscricao() {
		return inscricao;
	}

	public void setInscricao(String inscricao) {
		this.inscricao = inscricao;
	}

	public String getRazao() {
		return razao;
	}

	public void setRazao(String razao) {
		this.razao = razao;
	}

	public String getCnpj() {
		return cnpj;
	}

	public void setCnpj(String cnpj) {
		this.cnpj = cnpj;
	}

	public void setCodigo(Long codigo) {
		this.codigo = codigo;
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((codigo == null) ? 0 : codigo.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;
		Empresa other = (Empresa) obj;
		if (codigo == null) {
			if (other.codigo != null)
				return false;
		} else if (!codigo.equals(other.codigo))
			return false;
		return true;
	}

	}
```

Classe empresa DAO

```
import java.io.Serializable;
import java.util.List;
import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.Query;
import br.com.acme.model.Empresa;
import br.com.acme.uteis.Uteis;

public class EmpresaDAO implements Serializable {

    private static final long serialVersionUID = 1344324234243234L;

	@Inject
	Empresa empresa;

	EntityManager entityManager;

	public boolean gravar(Empresa empresa){
	  boolean sucesso = false;
	  try {
	  entityManager =  Uteis.JpaEntityManager();
	  entityManager.merge(empresa);
	      sucesso = true;
	  } catch (Exception e) {
	      e.printStackTrace();
	  }

	  return sucesso;
    }

    public Empresa selecionar(Long codigo){
        Empresa empresa = null;
        try {
             entityManager =  Uteis.JpaEntityManager();
            empresa = entityManager.find(Empresa.class, codigo);
        } catch (Exception e) {
            e.printStackTrace();
```

```
        }

        return empresa;
    }

    public boolean remover(Empresa empresa){
        boolean sucesso = false;
        try {
             entityManager =  Uteis.JpaEntityManager();
            empresa = entityManager.find(Empresa.class, empresa.getCodigo());

            entityManager.remove(empresa);
            sucesso = true;
        } catch (Exception e) {
            e.printStackTrace();
        }

        return sucesso;
    }

    public List<Empresa> listar() {
        List<Empresa> empresas = null;
        try {
             entityManager =  Uteis.JpaEntityManager();
            Query query = entityManager.createQuery("Select c from Empresa c");
           empresas = query.getResultList();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return empresas;
    }
      }
```

Classe empresa controller

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.Serializable;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import javax.annotation.PostConstruct;
import javax.enterprise.context.SessionScoped;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.faces.event.ActionEvent;
import javax.inject.Inject;
import javax.inject.Named;
import javax.servlet.http.Part;
import br.com.acme.model.Canhoto;
import br.com.acme.model.Empresa;
import br.com.acme.repository.CanhotoDAO;
import br.com.acme.repository.EmpresaDAO;
```

```
@Named(value = "empresaBean")
@SessionScoped
public class EmpresaBean implements Serializable {

       private static final long serialVersionUID = 14324242L;

       @Inject
    private EmpresaDAO empresaDAO;
    private transient Empresa empresa;
    private CanhotoDAO canhotoDAO;
    private transient Canhoto canhoto;

    private transient List<Empresa> empresas;

    private transient List<Canhoto> canhotos;


    private Part file;


    private String imagemDoCanhotoSelecionado;


       public String novo(){
         empresa = new Empresa();
         return "empresa.xhtml";
    }


    @PostConstruct
    protected void init() {
       empresaDAO = new EmpresaDAO();
       empresa = new Empresa();
       canhoto = new Canhoto();
       canhotoDAO = new CanhotoDAO();
       this.canhotos= new ArrayList<>();

    }

    public String gravar() {
       boolean resultado = empresaDAO.gravar(empresa);

         if (resultado) {
             empresa = new Empresa();
             FacesContext.getCurrentInstance().addMessage("", new
FacesMessage("Sucesso ,Empresa Cadastrado!"));
         } else {
              FacesContext.getCurrentInstance().addMessage("", new
FacesMessage("ERRO"));
         }
         return "empresaLista.xhtml";
    }

    public void selecionar(ActionEvent event) {
        Long codigo = (Long) event.getComponent().getAttributes().get("codigo");
        empresa = empresaDAO.selecionar(codigo);

    }
```

```
    public void selecionarCanhoto(ActionEvent evento) {
        Long id_dependente = (Long)
evento.getComponent().getAttributes().get("id_dependente");
         canhoto = canhotoDAO.selecionar(id_dependente);

    }

    public void removerCanhoto(ActionEvent evento) {
       Long codigo = (Long)
evento.getComponent().getAttributes().get("id_dependente");
        canhoto = canhotoDAO.selecionar(codigo);

        FacesContext context = FacesContext.getCurrentInstance();
        boolean resultado = canhotoDAO.remover(canhoto);

        if (resultado) {
            canhoto = new Canhoto();
            context.addMessage(null, new FacesMessage("Canhoto removido com
sucesso"));
        } else {
            context.addMessage(null, new FacesMessage("Falha ao remover
canhoto!"));
        }
    }

    public void remover(ActionEvent evento) {
       Long codigo = (Long) evento.getComponent().getAttributes().get("codigo");
        empresa = empresaDAO.selecionar(codigo);

        FacesContext context = FacesContext.getCurrentInstance();
        boolean resultado = empresaDAO.remover(empresa);

        if (resultado) {
            empresa = new Empresa();
            context.addMessage(null, new FacesMessage("Empresa removido com
sucesso"));
        } else {
            context.addMessage(null, new FacesMessage("Falha ao remover
empresa!"));
        }
    }

    public String addCanhoto(){

             if(canhoto.getEmpresa().getCodigo() == null || canhoto.getNota() ==
null || canhoto.getNome() == null || canhoto.getRg() == null || file == null) {
                    FacesContext.getCurrentInstance().addMessage("", new
FacesMessage("Erro, Existem campos sem preenchimento!"));
                    return null;
                    }
             else{
                    String nomeArquivoSaida = "/Users/VICTOR/Desktop/canhotos/o" +
file.getSubmittedFileName();

                    try (InputStream is = file.getInputStream();
                                    OutputStream out = new
FileOutputStream(nomeArquivoSaida)) {

                  int read = 0;
```

```
                    byte[] bytes = new byte[1024];

                    while ((read = is.read(bytes)) != -1) {
                        out.write(bytes, 0, read);
                    }
                    canhoto.setImagem(nomeArquivoSaida);

                     } catch (IOException e) {

                     }

                     boolean resultado = this.canhotoDAO.gravar(canhoto);

                           if(resultado) {
                                 canhoto = new Canhoto();
                                 this.file = null;

                           }
                     }
                return "canhoto.xhtml";


       }

    public void imagemAPartirDoArrayDeByte(){

        try{
              if(canhoto.getImagem() != null){

                    Path path =
Paths.get(FacesContext.getCurrentInstance().getExternalContext().getRealPath("/").
toString()+"/templates/canhoto");

                    if(!Files.exists(path)){
                          Files.createDirectories(path);
                    }

                    path = Paths.get(path.toRealPath() + "/" +
canhoto.getId_dependente() + ".jpg");
                    if(!Files.exists(path)){
                          FileOutputStream fos = new
FileOutputStream(path.toString());
                         fos.close();
                    }

                  imagemDoCanhotoSelecionado =
"../templates/canhoto/"+canhoto.getId_dependente() + ".jpg";

               }else {
                     imagemDoCanhotoSelecionado = null;
               }
        }catch(IOException e){
             imagemDoCanhotoSelecionado = null;
             }
    }

    public Empresa getEmpresa() {
            return empresa;
      }
```

```
    public String getImagemDoCarroSelecionado() {
            return imagemDoCanhotoSelecionado;
      }
    public void setImagemDoCarroSelecionado(String imagemDoCarroSelecionado) {
            this.imagemDoCanhotoSelecionado = imagemDoCarroSelecionado;
      }
    public Part getFile() {
            return file;
      }
    public void setFile(Part file) {
            this.file = file;
      }
    public Canhoto getCanhoto() {
            return canhoto;
      }
    public void setCanhoto(Canhoto canhoto) {
            this.canhoto = canhoto;
      }
    public List<Canhoto> getCanhotos() {
            this.canhotos = canhotoDAO.list();
            return canhotos;
      }
    public void setCanhotos(List<Canhoto> canhotos) {
            this.canhotos = canhotos;
      }
    public void setEmpresa(Empresa empresa) {
            this.empresa = empresa;
      }
    public List<Empresa> getEmpresas() {
            this.empresas = empresaDAO.listar();
            return empresas;
      }
    public void setEmpresas(List<Empresa> empresas) {
            this.empresas = empresas;
      }
      }
```

Classe canhoto

```
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.Table;

@Entity
@Table(name = "canhoto")
public class Canhoto {

      @Id
      @GeneratedValue(strategy=GenerationType.AUTO)
      private Long id_dependente;

      @Column
      private String nome;
```

```
@Column
private String rg;

@Column
private String nota;

@Column
private String destina;

@Column
private String imagem;

@ManyToOne
@JoinColumn(name="codigo")
private Empresa empresa = new Empresa();

public Long getId_dependente() {
	return id_dependente;
}
public void setId_dependente(Long id_dependente) {
	this.id_dependente = id_dependente;
}
public String getNome() {
	return nome;
}

public void setNome(String nome) {
	this.nome = nome;
}
public String getRg() {
	return rg;
}
public void setRg(String rg) {
	this.rg = rg;
}
public String getDestina() {
	return destina;
}
public void setDestina(String destina) {
	this.destina = destina;
}
public String getNota() {
	return nota;
}
public void setNota(String nota) {
	this.nota = nota;
}
public Empresa getEmpresa() {
	return empresa;
}
public void setEmpresa(Empresa empresa) {
	this.empresa = empresa;
}

public String getImagem() {
	return imagem;
}
public void setImagem(String imagem) {
	this.imagem = imagem;
```

```
	}
	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((empresa == null) ? 0 :
empresa.hashCode());
		result = prime * result + ((id_dependente == null) ? 0 :
id_dependente.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;
		Canhoto other = (Canhoto) obj;
		if (empresa == null) {
			if (other.empresa != null)
				return false;
		} else if (!empresa.equals(other.empresa))
			return false;
		if (id_dependente == null) {
			if (other.id_dependente != null)
				return false;
		} else if (!id_dependente.equals(other.id_dependente))
			return false;
		return true;
	}
	}
```

Classe canhoto DAO

```
import java.io.Serializable;
import java.util.List;
import javax.inject.Inject;
import javax.persistence.EntityManager;
import javax.persistence.TypedQuery;
import br.com.acme.model.Canhoto;
import br.com.acme.model.Empresa;
import br.com.acme.uteis.Uteis;


public class CanhotoDAO implements Serializable{

	private static final long serialVersionUID = 1L;

	@Inject
	Canhoto canhoto;

	@Inject
	Empresa empresa;
```

```
	private EntityManager entityManager;

	  public boolean gravar(Canhoto canhoto){
		boolean sucesso = false;
		try {
		entityManager =  Uteis.JpaEntityManager();
		entityManager.merge(canhoto);

		    sucesso = true;
		} catch (Exception e) {
		    e.printStackTrace();
		}

		return sucesso;
	    }
	    public Canhoto selecionar(Long id_dependente){
		Canhoto canhoto = null;

		try {
		     entityManager =  Uteis.JpaEntityManager();
		    canhoto = entityManager.find(Canhoto.class, id_dependente);

		} catch (Exception e) {
		    e.printStackTrace();
		}

		return canhoto;
	    }


	    public boolean remover(Canhoto canhoto){
		boolean sucesso = false;
		try {
		     entityManager =  Uteis.JpaEntityManager();
		    canhoto = entityManager.find(Canhoto.class,
canhoto.getId_dependente());
		    entityManager.remove(canhoto);
		    sucesso = true;
		} catch (Exception e) {
		    e.printStackTrace();
		}

		return sucesso;
	    }

	public List<Canhoto> list() {
	      entityManager = Uteis.JpaEntityManager();
	      TypedQuery<Canhoto> query = entityManager.createQuery("Select d From
Canhoto d", Canhoto.class);
	      return query.getResultList();
	}
	}
```

Página de login

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
```

```
      xmlns:ui="http://java.sun.com/jsf/facelets">

  <h:head>
    <meta charset="utf-8"/>
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-
fit=no"/>
    <meta name="description" content=""/>
    <meta name="author" content=""/>
    <link rel="icon" href="../../../../favicon.ico"/>

    <title>Login Acme</title>

    <!-- Bootstrap core CSS -->
     <h:outputStylesheet library="css" name="bootstrap.min.css" />
     <h:outputStylesheet library="css" name="signin.css" />

  </h:head>

  <body>

    <div class="container">

       <h:form class="form-signin" style="margin-top:100px;">
       <h2 class="form-signin-heading" align="center"><h:graphicImage
library="imagens" name="Logo.png" class="img-fluid" alt="Responsive image" />

       Login</h2>

        <h:messages class="alert alert-danger" role="alert" />

        <h:outputLabel  class="sr-only" value="E-mail"/>
        Usuário:<h:inputText value="#{sessaoMB.usuarioEntity.usuario}"
class="form-control" placeholder="Email address"/>
        <br></br>

        <h:outputLabel  class="sr-only" value="Senha"/>
        Senha:<h:inputSecret  value="#{sessaoMB.usuarioEntity.senha}" class="form-
control" placeholder="Password"/>
        <br></br>
        <h:commandLink  action="#{sessaoMB.login()}" class="btn btn-lg btn-primary
btn-block" value="Entrar"/>
      <br></br>

      </h:form>
      </div>
     <h:outputScript library="js" name="ie10-viewport-bug-workaround.js"/>
  </body>
       </html>
```

Página inicial

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:ui="http://java.sun.com/jsf/facelets">


  <ui:include src="/templates/head.xhtml" />
```

```
  <h:body>

      <ui:include src="/templates/menu_top.xhtml" />
    <ui:include src="/templates/menu_lateral.xhtml" />


    <div class="container-fluid">

       <ui:include src="/templates/conteudo.xhtml" />
                  <ui:insert name="corpo">Corpo original do template </ui:insert>


    </div>

       <ui:include src="/templates/js_rodape.xhtml" />
  </h:body>
      </html>
```

Página de cadastrar usuário

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets">

<ui:composition template="/templates/template.xhtml">

      <ui:define name="corpo">

<div class="row">
                  <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2
main">
                        <h1 class="page-header">ACME</h1>

                        <br></br>
                        <h4 class="sub-header">Cadastro de Usuário</h4>

            <div class="container-fluid">
                  <section class="container">
                        <div class="container-page">
                              <div class="col-md-10">


                  <h:form id="dadosUsuario">
                  <h:messages class="alert alert-success" role="alert" />
                  <div class="form-group col-lg-8">
                        <h:outputText value="Nome do usuario:" />
                        <h:inputText value="#{sessaoMB.usuario.usuario}"
class="form-control" />
                        <br></br>
                   </div>
                   <div class="form-group col-lg-6">
                        <h:outputText value="Senha:" />
                        <h:inputSecret value="#{sessaoMB.usuario.senha}"
class="form-control" />
```

```
                        <br></br>
                    </div>

                    <div class="form-group col-lg-8">
                     <h:commandButton value="Salvar" action="#{sessaoMB.gravar}"
class="btn btn-primary pull-right" update="dadosUsuario" />
                    <br></br>
                    </div>

                    </h:form>

                          </div>
                          </div>
                      </section>


                    </div>

                    </div>

              </div>

       </ui:define>

</ui:composition>
       </html>
```

Página para cadastrar empresa

```
<html xmlns="http://www.w3.org/1999/xhtml"
       xmlns:ui="http://java.sun.com/jsf/facelets"
       xmlns:h="http://java.sun.com/jsf/html"
       xmlns:f="http://java.sun.com/jsf/core">

<ui:composition template="/templates/template.xhtml">

       <ui:define name="corpo">
              <div class="row">
                    <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2
main">


                          <br></br>
                          <h3 class="sub-header">Cadastro de Empresa</h3>



              <div class="container-fluid" id="dadosEmpresa">
                    <section class="container">
                          <div class="container-page">
                                <div class="col-md-10">

        <h:form  role="form">
              <h:messages class="alert alert-success" role="alert" />

         <div class="form-group col-lg-8">
              <h:outputLabel value="Razão Social:"/>
              <h:inputText value="#{empresaBean.empresa.razao}"  class="form-
control" placeholder="Email address"/>
```

```
            <br></br>
            </div>

        <div class="form-group col-lg-5">
            <h:outputLabel value="CNPJ:"/>
            <h:inputText value="#{empresaBean.empresa.cnpj}"  class="form-
control" placeholder="Idade"/>
            <br></br>
            </div>

               <div class="form-group col-lg-4">
            <h:outputLabel value="Inscrição Estadual:"/>
            <h:inputText value="#{empresaBean.empresa.inscricao}"  class="form-
control" placeholder="Idade"/>
            <br></br>
            </div>
            <div class="form-group col-lg-4">
            <h:outputLabel value="Telefone:"/>
            <h:inputText value="#{empresaBean.empresa.telefone}"  class="form-
control" placeholder="Idade"/>
            <br></br>
            </div>


        <h:commandLink  action="#{empresaBean.gravar}" class="btn btn-primary
pull-right" value="Salvar" update="dadosEmpresa"/>



      </h:form>
                                  </div>
                            </div>
                  </section>


                   </div>

                  </div>

            </div>

       </ui:define>

</ui:composition>

       </html>
```

Página para mostrar empresas cadastradas

```
<?xml version='1.0' encoding='UTF-8' ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:ui="http://java.sun.com/jsf/facelets">

<ui:composition template="/templates/template.xhtml">
```

```
      <ui:define name="corpo">
            <div class="row">
                  <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2
main">
                        <h1 class="page-header">ACME</h1>

                        <br></br>
                        <h2 class="sub-header">Empresas</h2>

        <h:form>
                <div>
                <h:commandButton value="Novo Cadastro"
action="#{empresaBean.novo}" class="btn btn-primary"/>
            </div>

           <fieldset style="width: 1100px">
             <h:dataTable value="#{empresaBean.empresas}" var="empresa"
class="table table-hover table-responsive">

                    <h:column >
                        <f:facet name="header"><h:outputText value="ID"
/></f:facet>
                        <h:outputText value="#{empresa.codigo}" />
                    </h:column>

                        <h:column>
                               <f:facet name="header"><h:outputText value="Razão
Social" /></f:facet>
                         <h:outputText value="#{empresa.razao}" />
                        </h:column>
                        <h:column>
                               <f:facet name="header"><h:outputText value="CNPJ"
/></f:facet>
                         <h:outputText value="#{empresa.cnpj}" />
                        </h:column>
                         <h:column>
                               <f:facet name="header"><h:outputText
value="Inscrição Estadual" /></f:facet>
                         <h:outputText value="#{empresa.inscricao}" />
                        </h:column>
                        <h:column>
                               <f:facet name="header"><h:outputText
value="Telefone:" /></f:facet>
                         <h:outputText value="#{empresa.telefone}" />
                        </h:column>

                    <h:column>
                        <h:commandButton value="Editar" class="update btn btn-
warning btn-sm" action="empresa.xhtml?faces-redirect=true"
actionListener="#{empresaBean.selecionar}">
                            <f:attribute name="codigo" value="#{empresa.codigo}"
/>
                            <f:ajax render=":dadosEmpresa" execute="@this" />
                        </h:commandButton>
                    </h:column>
                       <h:column>
                        <h:commandButton value="Excluir" class="delete btn btn-
danger btn-sm" action="empresaLista.xhtml?faces-redirect=true"
actionListener="#{empresaBean.remover}">
```

```
                            <f:attribute name="codigo" value="#{empresa.codigo}"
/>
                            <f:ajax render=":dadosEmpresa" execute="@this" />
                        </h:commandButton>
                    </h:column>
                    <h:column>
                        <h:commandButton value="Canhotos" class="btn btn-primary"
action="canhoto.xhtml?faces-redirect=true"
actionListener="#{empresaBean.selecionar}">
                            <f:attribute name="codigo" value="#{empresa.codigo}"
/>
                            <f:ajax render=":dadosEmpresa" execute="@this" />
                        </h:commandButton>
                    </h:column>
                </h:dataTable>
            </fieldset>
        </h:form>

                        </div>
```

Página para cadastrar canhotos e exibir canhotos cadastrados.

```
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:ui="http://java.sun.com/jsf/facelets"
      xmlns:h="http://java.sun.com/jsf/html"
      xmlns:f="http://java.sun.com/jsf/core"
      xmlns:p="http://primefaces.org/ui">


<ui:composition template="/templates/template.xhtml">

      <ui:define name="corpo">
            <div class="row">
                  <div class="col-sm-9 col-sm-offset-3 col-md-10 col-md-offset-2
main">


                        <br></br>
                        <h3 class="sub-header">Cadastro de Canhotos</h3>


            <div class="container-fluid" id="dadosCanhoto">
                  <section class="container">
                        <div class="container-page">
                              <div class="col-md-10">

       <h:form  role="form" enctype="multipart/form-data">
            <h:messages class="alert alert-success" role="alert" />
      <div class="form-group col-lg-12">
            <h:outputLabel value="Nome da Empresa:"/>
            <h:inputText value="#{empresaBean.canhoto.destina}"  class="form-
control" placeholder="Email address"/>
            <br></br>
            </div>
        <div class="form-group col-lg-8">
            <h:outputLabel value="Recebido por:"/>
            <h:inputText value="#{empresaBean.canhoto.nome}"  class="form-
control" placeholder="Email address"/>
```

```
            <br></br>
            </div>
        <div class="form-group col-lg-4">
            <h:outputLabel value="RG:"/>
            <h:inputText value="#{empresaBean.canhoto.rg}"  class="form-control"
placeholder="Idade"/>
            <br></br>
            </div>
        <div class="form-group col-lg-4">
            <h:outputLabel value="Número da Nota Fiscal:"/>
            <h:inputText value="#{empresaBean.canhoto.nota}"  class="form-
control" placeholder="Idade"/>
            <br></br>
            </div>

<div class="form-group col-lg-8">
            <h:outputLabel value="Empresa Emitente:"/>
            <h:selectOneMenu class="form-control"
value="#{empresaBean.canhoto.empresa.codigo}">
                    <f:selectItem itemValue = "" itemLabel="Selecione uma empresa"
/>
                    <f:selectItems value="#{empresaBean.empresas}" var="empresa"
                                            itemValue="#{empresa.codigo}"
itemLabel="#{empresa.razao}"/>
            </h:selectOneMenu>
            <br></br>
            </div>
            <h:inputFile value="#{empresaBean.file}" />

        <div class="form-group col-lg-13">
        <h:commandButton ajax="false"  action="#{empresaBean.addCanhoto}"
class="btn btn-primary pull-right" value="Salvar" update=":dadosEmpresa" >

      </h:commandButton>
      </div>



     </h:form>

       </div>

        <h:form>



           <fieldset style="width: 1050px">
             <h2 class="sub-header">Canhotos</h2>
             <h:dataTable id="depen" value="#{empresaBean.canhotos}"
var="canhoto" class="table table-hover table-responsive">

                    <h:column >
                        <f:facet name="header"><h:outputText value="ID:"
/></f:facet>
                        <h:outputText value="#{canhoto.id_dependente}" />
                    </h:column>
                    <h:column >
                        <f:facet name="header"><h:outputText value="Destinatario:"
/></f:facet>
```

```
                        <h:outputText value="#{canhoto.destina}" />
                    </h:column>

                        <h:column>
                                <f:facet name="header"><h:outputText
value="Recibido por:" /></f:facet>
                         <h:outputText value="#{canhoto.nome}" />
                        </h:column>
                        <h:column>
                                <f:facet name="header"><h:outputText value="RG:"
/></f:facet>
                         <h:outputText value="#{canhoto.rg}" />
                        </h:column>
                        <h:column>
                                <f:facet name="header"><h:outputText value="Nota
Fiscal:" /></f:facet>
                         <h:outputText value="#{canhoto.nota}" />
                        </h:column>
                        <h:column>
                                <f:facet name="header"><h:outputText
value="Empresa Emitente:" /></f:facet>
                         <h:outputText value="#{canhoto.empresa.razao}" />
                        </h:column>


                                <h:column>
                    <h:commandButton value="Editar" class="update btn btn-
warning btn-sm" action="canhoto.xhtml?faces-redirect=true"
actionListener="#{empresaBean.selecionarDependente}">
                            <f:attribute name="id_dependente"
value="#{canhoto.id_dependente}" />
                            <f:ajax render=":dadosCanhoto" execute="@this" />
                        </h:commandButton>
                    </h:column>
                     <h:column>
                        <h:commandButton value="Excluir" class="delete btn btn-
danger btn-sm" action="canhoto.xhtml?faces-redirect=true"
actionListener="#{empresaBean.removerDependente}">
                            <f:attribute name="id_dependente"
value="#{canhoto.id_dependente}" />
                            <f:ajax render=":dadosCanhoto" execute="@this" />
                        </h:commandButton>
                    </h:column>
                    <h:column>
                    <h:commandButton value="img" class="delete btn btn-danger btn-
sm" action="#{empresaBean.imagemAPartirDoArrayDeByte}"
oncomplete="PF('exibirCarro').show()" process="@this"
update=":formPesquisa:exibirCarroDialog">
                                        <f:setPropertyActionListener
value="#{canhoto}" target="#{empresaBean.canhoto}" />
                                </h:commandButton>
                                </h:column>

                </h:dataTable>

                            <p:dialog widgetVar="exibirCarro"
id="exibirCarroDialog" closable="false" header="#{empresaBean.canhoto.nome}" >
```

```
                                <p:graphicImage
value="#{empresaBean.imagemDoCarroSelecionado}"/>

                                <h:outputText
rendered="#{empresaBean.imagemDoCarroSelecionado == null}" value="Nenhuma imagem
cadastrada."/>

                                <br />

                                <p:commandButton process="@this" style="margin-
top:20px;" onclick="PF('exibirCarro').hide();" value="OK"/>

                        </p:dialog>
            </fieldset>
                    </h:form>
                        </div>
                    </section>


                     </div>

                    </div>

             </div>

      </ui:define>

</ui:composition>

      </html>
```