

Desenvolvimento Web II

Aula 04 - Desenvolvendo um Sistema de Informação Web com JSF

Apresentação

Olá, seja bem-vindo(a) à nossa quarta aula sobre o framework JSF. Agora que já conhecemos os principais recursos oferecidos por esse framework, tais como tags e formas de validações, desenvolveremos uma aplicação JSF, a qual chamaremos de Sistema Integrado de Transportes Urbanos (SITURB), tendo como objetivo cadastrar as empresas, linhas de ônibus, horários e funcionários (motorista e cobrador).

Empolgado para desenvolver essa aplicação que coloca à prova os seus conhecimentos? Então, mãos à obra!

Objetivos

- Desenvolver uma aplicação JSF que utiliza todos os conhecimentos obtidos nas aulas anteriores;
- Fixar os passos necessários para criação de aplicações web com framework JSF.

Desenvolvimento do Modelo

Como citado anteriormente, nossa aplicação tem o objetivo de cadastrar empresas de transportes urbanos, suas linhas, horários e funcionários (motorista e cobrador). Então, nessa aplicação JSF existem os seguintes conceitos:

- Empresa: empresas de ônibus com razão social e CNPJ;
- Ônibus: veículo com marca, modelo e ano;
- Linha: rota com número, origem, destino, descrição, horário de saída e chegada;
- Cobrador: com nome, matrícula, CPF e endereço;
- Motorista: com mesmos dados do cobrador e informações da carteira de habilitação.

Vamos implementar essas classes? Primeiro, crie um projeto dinâmico web com o nome SITURB. Após a criação desse projeto, realize os passos para que possamos fazer com que esse projeto web seja JSF: adicione a biblioteca do JSF na pasta lib do projeto e defina a Servlet do Faces no web.xml. Lembre-se que esses são os mesmos descritos na primeira aula sobre JSF. A Listagem 1 apresenta a definição dessa servlet.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3 xmlns="http://java.sun.com/xml/ns/javaee"
4 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5 http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
6
7   <display-name>AulaValidacoesJSF</display-name>
8   <welcome-file-list>
9     <welcome-file>index.jsf</welcome-file>
10  </welcome-file-list>
11
12   <servlet>
13     <servlet-name>Faces Servlet</servlet-name>
14     <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
15     <load-on-startup>1</load-on-startup>
16   </servlet>
17
18   <servlet-mapping>
19     <servlet-name>Faces Servlet</servlet-name>
20     <url-pattern>*.jsf</url-pattern>
21   </servlet-mapping>
22
23 </web-app>
```

Listagem 1 - Definição da servlet do JSF no web.xml.

Com esses passos, nosso projeto dinâmico web já é JSF. Agora vamos criar um pacote denominado **br.ufrn.imd.model** que conterá as classes de domínio da nossa aplicação. A começar pela classe Empresa, cujo código é apresentado na Listagem 2.

```

1 package br.ufrn.imd.model;
2
3 /**
4  * Classe que representa uma empresa de ônibus.
5  * @author itamir.filho
6  *
7  */
8 public class Empresa {
9
10     private String razaoSocial;
11
12     private String cnpj;
13
14     public String getRazaoSocial() {
15         return razaoSocial;
16     }
17
18     public void setRazaoSocial(String razaoSocial) {
19         this.razaoSocial = razaoSocial;
20     }
21
22     public String getCnpj() {
23         return cnpj;
24     }
25
26     public void setCnpj(String cnpj) {
27         this.cnpj = cnpj;
28     }
29
30     @Override
31     public String toString() {
32         return razaoSocial;
33     }
34
35 }

```

Listagem 2 - Código da classe Empresa.

Sabemos que uma empresa de transportes urbanos possui vários ônibus, que são dirigidos por motoristas e têm suas passagens coletadas por cobradores. Com isso, vejamos as próximas entidades: Cobrador e Motorista. A classe Cobrador é apresentada na Listagem 3.

```
1 package br.ufrn.imd.model;
2
3 /**
4  * Classe que representa a entidade cobrador.
5  * @author itamir.filho
6  *
7  */
8
9 public class Cobrador {
10
11     private String nome;
12
13     private String cpf;
14
15     private String matricula;
16
17     private String endereco;
18
19     public String getNome() {
20         return nome;
21     }
22
23     public void setNome(String nome) {
24         this.nome = nome;
25     }
26
27     public String getCpf() {
28         return cpf;
29     }
30
31     public void setCpf(String cpf) {
32         this.cpf = cpf;
33     }
34
35     public String getMatricula() {
36         return matricula;
37     }
38
39     public void setMatricula(String matricula) {
40         this.matricula = matricula;
41     }
42
43     public String getEndereco() {
44         return endereco;
45     }
46
47     public void setEndereco(String endereco) {
48         this.endereco = endereco;
49     }
50
51     @Override
```

```
52 public String toString() {  
53     return nome;  
54 }  
55  
56 }
```

Listagem 3 - Código da classe Cobrador.

Agora vamos refletir sobre a classe Motorista: todo motorista também é um funcionário da empresa e o que diferencia um motorista dos cobradores são as informações sobre sua carteira e categoria de habilitação. Portanto, podemos usar um recurso muito importante da programação orientada a objetos para representar a classe Motorista, lembra qual é? Se você pensou em herança, acertou! A classe Motorista, apresentada na Listagem 4, utiliza o conceito de herança. Observe o uso da palavra **extends** na definição dessa classe.

```
1 package br.ufrn.imd.model;  
2  
3 /**  
4  * Classe que representa um motorista.  
5  * @author itamir.filho  
6  *  
7  */  
8  
9 public class Motorista extends Cobrador {  
10  
11     private String registroCnh;  
12  
13     private String categoriaCnh;  
14  
15     public String getRegistroCnh() {  
16         return registroCnh;  
17     }  
18  
19     public void setRegistroCnh(String registroCnh) {  
20         this.registroCnh = registroCnh;  
21     }  
22  
23     public String getCategoriaCnh() {  
24         return categoriaCnh;  
25     }  
26  
27     public void setCategoriaCnh(String categoriaCnh) {  
28         this.categoriaCnh = categoriaCnh;  
29     }  
30  
31 }
```

Listagem 4 - Código da classe Motorista.

Sabemos também que um ônibus faz um percurso de origem e destino em determinados horários. A classe que representa o conceito de uma linha de ônibus é apresentada na Listagem 5.


```
1 package br.ufrn.imd.model;
2
3 /**
4  * Classe que representa uma linha de ônibus.
5  * @author itamir.filho
6  */
7 public class Linha {
8
9     private String ident;
10
11     private String origem;
12
13     private String destino;
14
15     private String horaSaida;
16
17     private String horaChegada;
18
19     public String getOrigem() {
20         return origem;
21     }
22
23     public void setOrigem(String origem) {
24         this.origem = origem;
25     }
26
27     public String getDestino() {
28         return destino;
29     }
30
31     public void setDestino(String destino) {
32         this.destino = destino;
33     }
34
35     public String getHoraSaida() {
36         return horaSaida;
37     }
38
39     public void setHoraSaida(String horaSaida) {
40         this.horaSaida = horaSaida;
41     }
42
43     public String getHoraChegada() {
44         return horaChegada;
45     }
46
47     public void setHoraChegada(String horaChegada) {
48         this.horaChegada = horaChegada;
49     }
50
51     public String getIdent() {
```

```
52     return ident;
53 }
54
55 public void setIdent(String ident) {
56     this.ident = ident;
57 }
58
59 @Override
60 public String toString() {
61     return ident;
62 }
63
64 }
```

Listagem 5 - Código da classe Linha.

Finalmente vamos implementar a classe Onibus, apresentada na Listagem 6. Um ônibus tem um motorista, um cobrador, uma linha e pertence a uma empresa.

```
1 package br.ufrn.imd.model;
2
3 /**
4  * Classe que representa a entidade ônibus.
5  * @author itamir.filho
6  *
7  */
8
9 public class Onibus {
10
11     private String marca;
12
13     private String modelo;
14
15     private int ano;
16
17     private Empresa empresa;
18
19     private Linha linha;
20
21     private Cobrador cobrador;
22
23     private Motorista motorista;
24
25     public String getMarca() {
26         return marca;
27     }
28
29     public void setMarca(String marca) {
30         this.marca = marca;
31     }
32
33     public String getModelo() {
34         return modelo;
35     }
36
37     public void setModelo(String modelo) {
38         this.modelo = modelo;
39     }
40
41     public int getAno() {
42         return ano;
43     }
44
45     public void setAno(int ano) {
46         this.ano = ano;
47     }
48
49     public Empresa getEmpresa() {
50         return empresa;
51     }
52 }
```

```

52
53 public void setEmpresa(Empresa empresa) {
54     this.empresa = empresa;
55 }
56
57 public Linha getLinha() {
58     return linha;
59 }
60
61 public void setLinha(Linha linha) {
62     this.linha = linha;
63 }
64
65 public Cobrador getCobrador() {
66     return cobrador;
67 }
68
69 public void setCobrador(Cobrador cobrador) {
70     this.cobrador = cobrador;
71 }
72
73 public Motorista getMotorista() {
74     return motorista;
75 }
76
77 public void setMotorista(Motorista motorista) {
78     this.motorista = motorista;
79 }
80
81 }

```

Listagem 6 - Código da classe Onibus.

Ótimo, nosso modelo está pronto e agora vamos aos controllers!

Desenvolvimento dos Controllers e Views

Para iniciar o desenvolvimento dos controllers, crie um pacote denominado **br.ufrn.imd.controllers**. Dentro desse pacote, implemente o controller **CadastrarEmpresaMBean**, apresentado na Listagem 7, para realizar o cadastro de empresas de transportes urbanos. Esse controller possui os atributos: **empresa** e **empresasCadastradas**. Observe nessa listagem a presença da anotação do JSF **@SessionScope**, a qual determina que os objetos desse Managed Beans estarão no escopo de sessão.

```
1 package br.ufrn.imd.controllers;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import javax.faces.application.FacesMessage;
7 import javax.faces.bean.ManagedBean;
8 import javax.faces.bean.SessionScoped;
9 import javax.faces.context.FacesContext;
10
11 import br.ufrn.imd.model.Empresa;
12
13 /**
14  * Controller para cadastro das empresas.
15  * @author itamir.filho
16  *
17  */
18
19 @ManagedBean
20 @SessionScoped
21 public class CadastrarEmpresaMBean {
22
23     private Empresa empresa;
24
25     private List<Empresa> empresas;
26
27     public CadastrarEmpresaMBean() {
28         empresa = new Empresa();
29         empresas = new ArrayList<Empresa>();
30     }
31
32     public String entrarCadastro(){
33         return "/form_empresa.jsf";
34     }
35
36     public String voltar(){
37         return "/index.jsf";
38     }
39
40     public String cadastrar() {
41         empresas.add(empresa);
42         empresa = new Empresa();
43         FacesMessage msg = new FacesMessage("Empresa cadastrada com sucesso!");
44         msg.setSeverity(FacesMessage.SEVERITY_INFO);
45         FacesContext.getCurrentInstance().addMessage("", msg);
46         return "/form_empresa.jsf";
47     }
48
49     public Empresa getEmpresa() {
50         return empresa;
51     }
52 }
```

```

52
53 public void setEmpresa(Empresa empresa) {
54     this.empresa = empresa;
55 }
56
57 public List<Empresa> getEmpresas() {
58     return empresas;
59 }
60
61 public void setEmpresas(List<Empresa> empresas) {
62     this.empresas = empresas;
63 }
64
65 }

```

Listagem 7 - Código da classe CadastrarEmpresaMBean.

Como não temos um banco de dados para nossa aplicação, veja que estamos colocando os objetos cadastros em listas para que possamos acessá-los nos outros cadastros. Verifique a view para cadastrar empresas que é apresentada na Listagem 8. Essa página, denominada form_empresa.xhtml, contém os atributos para inserção das informações de uma empresa e deve ser incluída na pasta **WebContent** do seu projeto.

```


1 <html xmlns="http://www.w3.org/1999/xhtml"
2   xmlns:h="http://java.sun.com/jsf/html"
3   xmlns:f="http://java.sun.com/jsf/core">
4   <h:head>
5     <title>Empresa</title>
6   </h:head>
7
8   <h:body>
9     <h:messages/>
10    <h1>Nova Empresa</h1>
11    <h:form>
12      <fieldset>
13        <p> CNPJ: <h:inputText value="#{cadastrarEmpresaMBean.empresa.cnpj}" size="50"
14          required="true" requiredMessage="CNPJ: Campo obrigatório." /> </p>
15        <p> Razão Social: <h:inputText value="#{cadastrarEmpresaMBean.empresa.razaoSocial}"
16          required="true" requiredMessage="Razão social: Campo obrigatório." /> </p>
17        <p><h:commandButton value="Cadastrar" action="#{cadastrarEmpresaMBean.cadastrar}"
18          <h:commandButton value="Voltar" action="#{cadastrarEmpresaMBean.voltar}" immediate
19      </fieldset>
20    </h:form>
21  </h:body>
22 </html>

```

Listagem 8 - Código da página form_empresa.xhtml.

Ao implementar as Listagens 7 e 8, podemos acessar esse formulário pelo endereço `http://localhost:8080/SITURB/form_empresa.jsf`, onde obtemos a página apresentada na Figura 1.

Figura 01 - Página para cadastrar empresas de transportes urbanos.



A imagem mostra uma captura de tela de um navegador web. No topo, há uma barra de endereços com o texto "Empresa" e um ícone de lupa. Abaixo, a barra de endereços exibe o URL "http://localhost:8080/SITURB/form_empresa.jsf". Abaixo da barra de endereços, há uma barra de navegação com ícones para "Mais visitados", "Favoritos" e "webi".

Nova Empresa

CNPJ:

Razão Social:

Na mesma ordem de criação das classes de domínio, vamos implementar o cadastro de cobradores. Começaremos com o controller `CadastrarCobradorMBean`, apresentado na Listagem 9, que deve ser incluído no pacote `br.ufrn.imd.controllers` e possuir os atributos: `cobrador` e `cobradoresCadastrados`.

```
1 package br.ufrn.imd.controllers;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import javax.faces.application.FacesMessage;
7 import javax.faces.bean.ManagedBean;
8 import javax.faces.bean.SessionScoped;
9 import javax.faces.context.FacesContext;
10
11 import br.ufrn.imd.model.Cobrador;
12
13 /**
14  * Controller para cadastrar cobradores.
15  * @author itamir.filho
16  *
17  */
18
19 @ManagedBean
20 @SessionScoped
21
22 public class CadastrarCobradorMBean {
23
24     private Cobrador cobrador;
25
26     private List<Cobrador> cobradores;
27
28     public CadastrarCobradorMBean() {
29         cobrador = new Cobrador();
30         cobradores = new ArrayList<Cobrador>();
31     }
32
33     public String entrarCadastro(){
34         return "/form_cobrador.jsf";
35     }
36
37     public String voltar(){
38         return "/index.jsf";
39     }
40
41     public String cadastrar() {
42         cobradores.add(cobrador);
43         cobrador = new Cobrador();
44         FacesMessage msg = new FacesMessage("Cobrador cadastrado com sucesso!");
45         msg.setSeverity(FacesMessage.SEVERITY_INFO);
46         FacesContext.getCurrentInstance().addMessage("", msg);
47         return "/form_cobrador.jsf";
48     }
49
50     public Cobrador getCobrador() {
51         return cobrador;
52     }
53 }
```



```

52 }
53
54 public void setCobrador(Cobrador cobrador) {
55     this.cobrador = cobrador;
56 }
57
58 public List<Cobrador> getCobradores() {
59     return cobradores;
60 }
61
62 public void setCobradores(List<Cobrador> cobradores) {
63     this.cobradores = cobradores;
64 }
65 }

```

Listagem 9 - Código da classe CadastrarCobradorMBean.

Vamos à view para cadastrar cobradores que é apresentada na Listagem 10. Essa página, denominada form_cobrador.xhtml, contém os atributos para inserção das informações de um cobrador e também deve ser incluída na pasta **WebContent** do seu projeto.

```

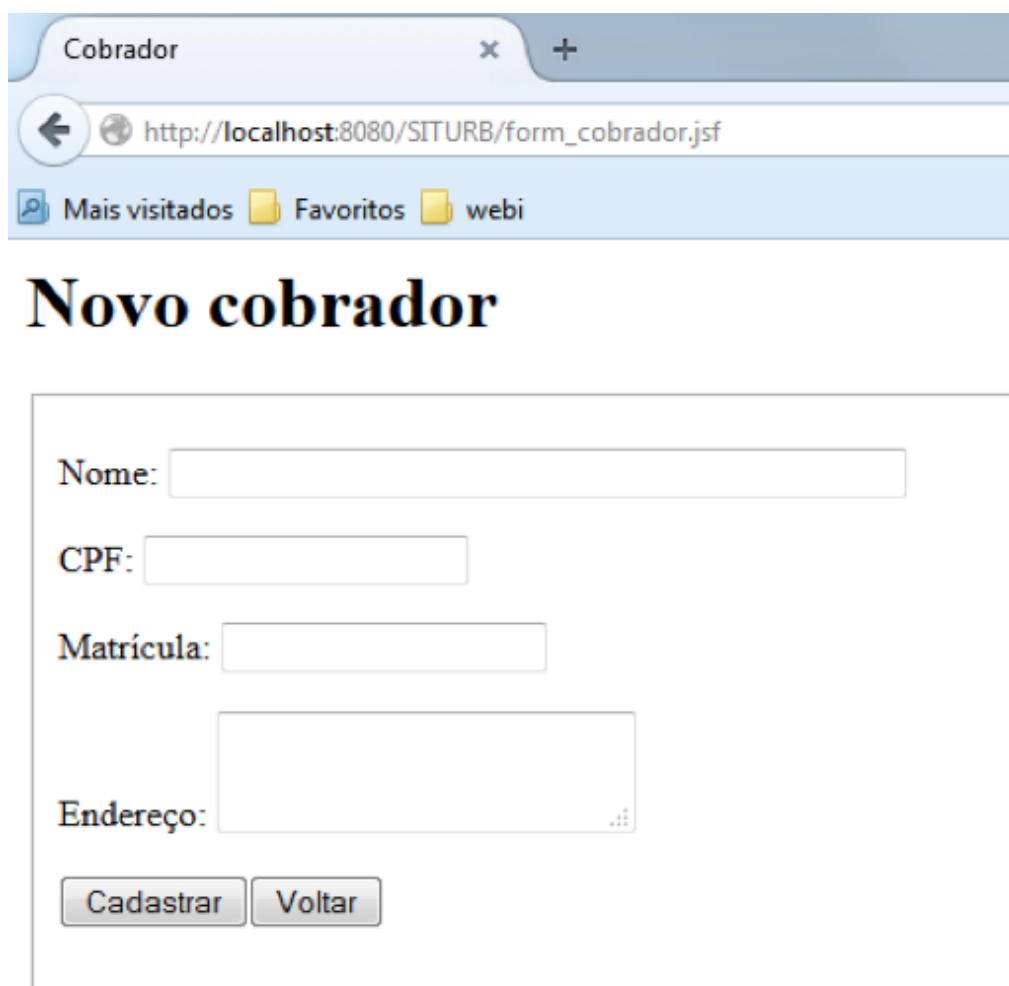
1 <html xmlns="http://www.w3.org/1999/xhtml"
2 xmlns:h="http://java.sun.com/jsf/html"
3 xmlns:f="http://java.sun.com/jsf/core">
4     <h:head>
5         <title>Cobrador</title>
6     </h:head>
7
8     <h:body>
9         <h:messages/>
10        <h1>Novo cobrador</h1>
11        <h:form>
12            <fieldset>
13                <p> Nome: <h:inputText value="#{cadastrarCobradorMBean.cobrador.nome}" size="50" r
14                <p> CPF: <h:inputText value="#{cadastrarCobradorMBean.cobrador.cpf}" size="20" requir
15                <p> Matrícula: <h:inputText value="#{cadastrarCobradorMBean.cobrador.matricula}" size
16                <p> Endereço: <h:inputTextarea value="#{cadastrarCobradorMBean.cobrador.endereco}"
17                <p> <h:commandButton value="Cadastrar" action="#{cadastrarCobradorMBean.cadastra
18                    <h:commandButton value="Voltar" action="#{cadastrarCobradorMBean.voltar}"
19                    immediate="true"/> </p>
20            </fieldset>
21        </h:form>
22    </h:body>
23 </html>

```

Listagem 10 - Código da página form_cobrador.xhtml.

Ao implementar as Listagens 9 e 10, podemos acessar esse formulário pelo endereço `http://localhost:8080/SITURB/form_cobrador.jsf`, onde obtemos a página apresentada na Figura 2.

Figura 02 - Página para cadastrar cobradores que são funcionários de empresas de transportes urbanos.



The image shows a web browser window with a single tab titled 'Cobrador'. The address bar displays the URL `http://localhost:8080/SITURB/form_cobrador.jsf`. Below the address bar, there are icons for 'Mais visitados', 'Favoritos', and 'webi'. The main content of the page is a form titled 'Novo cobrador' in a large, bold, black serif font. The form is enclosed in a thin black border and contains the following elements:

- A label 'Nome:' followed by a single-line text input field.
- A label 'CPF:' followed by a single-line text input field.
- A label 'Matrícula:' followed by a single-line text input field.
- A label 'Endereço:' followed by a multi-line text input field.
- At the bottom of the form, there are two buttons: 'Cadastrar' and 'Voltar', both with a light gray gradient and rounded corners.

Continuando com a criação dos controllers, verifique o cadastro de motoristas realizado pelo controller `CadastrarMotoristaMBean`, apresentado na Listagem 11. Esse controller também deve ser incluído no pacote **`br.ufrn.imd.controllers`**.

```
1 package br.ufrn.imd.controllers;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import javax.faces.application.FacesMessage;
7 import javax.faces.bean.ManagedBean;
8 import javax.faces.bean.SessionScoped;
9 import javax.faces.context.FacesContext;
10
11 import br.ufrn.imd.model.Motorista;
12
13 /**
14  * Controller para cadastrar motoristas.
15  * @author itamir.filho
16  *
17  */
18
19 @ManagedBean
20 @SessionScoped
21
22 public class CadastrarMotoristaMBean {
23
24     private Motorista motorista;
25
26     private List<Motorista> motoristas;
27
28     public CadastrarMotoristaMBean() {
29         motorista = new Motorista();
30         motoristas = new ArrayList<Motorista>();
31     }
32
33     public String entrarCadastro(){
34         return "/form_motorista.jsf";
35     }
36
37     public String voltar(){
38         return "/index.jsf";
39     }
40
41     public String cadastrar() {
42         motoristas.add(motorista);
43         motorista = new Motorista();
44         FacesMessage msg = new FacesMessage("Motorista cadastrado com sucesso!");
45         msg.setSeverity(FacesMessage.SEVERITY_INFO);
46         FacesContext.getCurrentInstance().addMessage("", msg);
47         return "/form_motorista.jsf";
48     }
49
50     public Motorista getMotorista() {
51         return motorista;
```

```
52 }  
53  
54 public void setMotorista(Motorista motorista) {  
55     this.motorista = motorista;  
56 }  
57  
58 public List<Motorista> getMotoristas() {  
59     return motoristas;  
60 }  
61  
62 public void setMotoristas(List<Motorista> motoristas) {  
63     this.motoristas = motoristas;  
64 }  
65 }  
66
```

Listagem 11 - Código da classe CadastrarMotoristaMBean.

Vejamos a view para cadastrar motoristas, apresentada na Listagem 12. Essa página, denominada form_motorista.xhtml, também deve ser incluída na pasta **WebContent** do seu projeto.

```

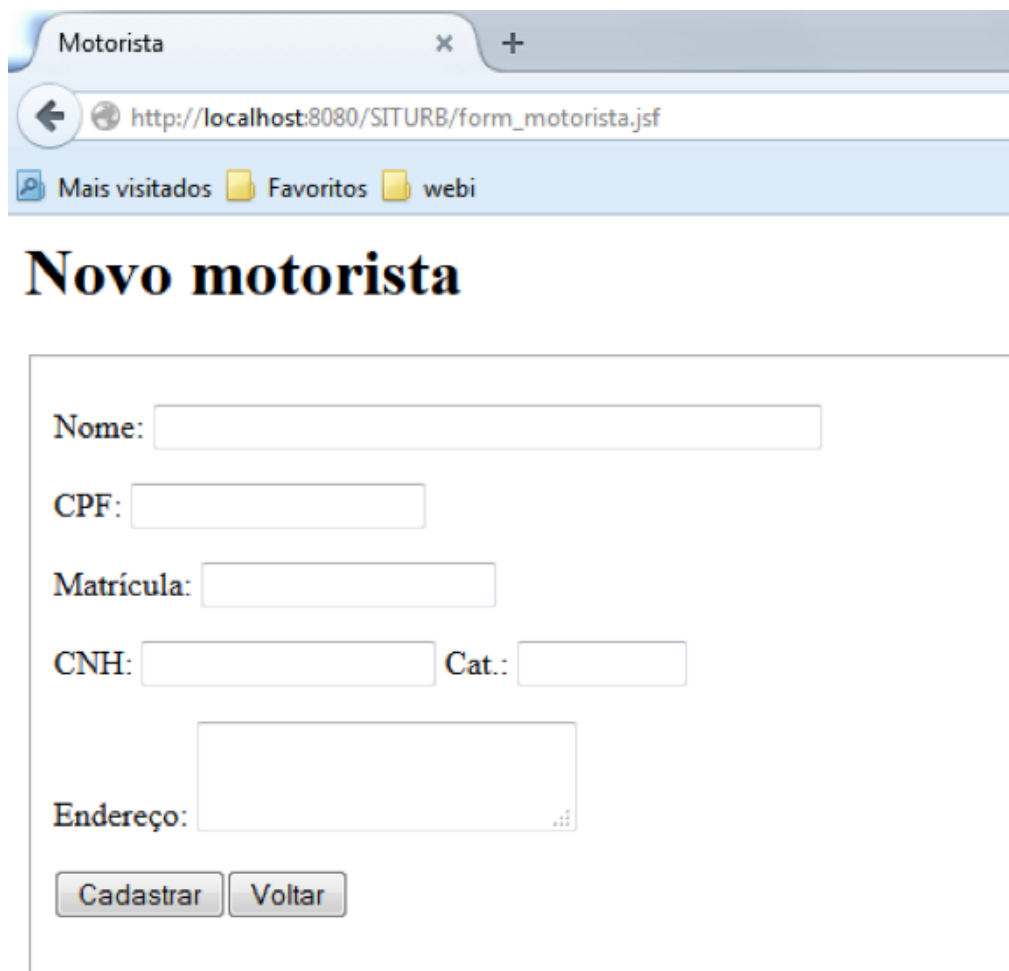
1 <html xmlns="http://www.w3.org/1999/xhtml"
2 xmlns:h="http://java.sun.com/jsf/html"
3 xmlns:f="http://java.sun.com/jsf/core">
4   <h:head>
5     <title>Motorista</title>
6   </h:head>
7   <h:body>
8     <h:messages/>
9     <h1>Novo motorista</h1>
10    <h:form>
11      <fieldset>
12        <p> Nome: <h:inputText value="#{cadastrarMotoristaMBean.motorista.nome}"
13          size="50" required="true" requiredMessage="Nome: Campo obrigatório." /> </p>
14        <p> CPF: <h:inputText value="#{cadastrarMotoristaMBean.motorista.cpf}"
15          size="20" required="true" requiredMessage="CPF: Campo obrigatório." /> </p>
16        <p> Matrícula: <h:inputText value="#{cadastrarMotoristaMBean.motorista.matricula}"
17          size="20" required="true" requiredMessage="Matrícula: Campo obrigatório." /> </p>
18        <p> CNH: <h:inputText value="#{cadastrarMotoristaMBean.motorista.registroCnh}"
19          size="20" required="true" requiredMessage="CNH: Campo obrigatório." />
20          Cat.: <h:inputText value="#{cadastrarMotoristaMBean.motorista.categoriaCnh}"
21            size="10" required="true" requiredMessage="Cat.: Campo obrigatório." /> </p>
22        <p> Endereço: <h:inputTextarea value="#{cadastrarMotoristaMBean.motorista.endereco}"
23          size="50" required="true" requiredMessage="Endereço: Campo obrigatório." /> </p>
24        <p><h:commandButton value="Cadastrar" action="#{cadastrarMotoristaMBean.cadastrar}"
25          <h:commandButton value="Voltar" action="#{cadastrarMotoristaMBean.voltar}"
26            immediate="true"/> </p>
27      </fieldset>
28    </h:form>
29  </h:body>
30 </html>

```

Listagem 12 - Código da página form_motorista.xhtml.

Ao implementar as Listagens 11 e 12, podemos acessar esse formulário pelo endereço http://localhost:8080/SITURB/form_motorista.jsf, onde obtemos a página apresentada na Figura 3.

Figura 03 - Página para cadastrar motoristas de ônibus.



The image shows a web browser window with a single tab titled "Motorista". The address bar displays the URL "http://localhost:8080/SITURB/form_motorista.jsf". Below the address bar, there are three icons: a blue folder icon labeled "Mais visitados", a yellow folder icon labeled "Favoritos", and a yellow folder icon labeled "webi". The main content area of the browser displays the title "Novo motorista" in a large, bold, black serif font. Below the title is a form with several input fields. The first field is labeled "Nome:" and is a single-line text box. The second field is labeled "CPF:" and is a single-line text box. The third field is labeled "Matrícula:" and is a single-line text box. The fourth field is labeled "CNH:" and is a single-line text box, followed by a label "Cat.:" and another single-line text box. The fifth field is labeled "Endereço:" and is a multi-line text box. At the bottom of the form, there are two buttons: "Cadastrar" and "Voltar".

Motorista

http://localhost:8080/SITURB/form_motorista.jsf

Mais visitados Favoritos webi

Novo motorista

Nome:

CPF:

Matrícula:

CNH: Cat.:

Endereço:

Vamos ao cadastro de linhas de ônibus realizada pelo controller `CadastrarLinhaMBean`, apresentado na Listagem 13 e também incluído no pacote **br.ufrn.imd.controllers**.

```
1 package br.ufrn.imd.controllers;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import javax.faces.application.FacesMessage;
7 import javax.faces.bean.ManagedBean;
8 import javax.faces.bean.SessionScoped;
9 import javax.faces.context.FacesContext;
10
11 import br.ufrn.imd.model.Linha;
12
13 /**
14  * Controller para cadastrar linhas de ônibus.
15  * @author itamir.filho
16  *
17  */
18
19 @ManagedBean
20 @SessionScoped
21
22 public class CadastrarLinhaMBean {
23
24     private Linha linha;
25
26     private List<Linha> linhas;
27
28     public CadastrarLinhaMBean() {
29         linha = new Linha();
30         linhas = new ArrayList<Linha>();
31     }
32
33     public String entrarCadastro(){
34         return "/form_linha.jsf";
35     }
36
37     public String voltar(){
38         return "/index.jsf";
39     }
40
41     public String cadastrar() {
42         linhas.add(linha);
43         linha = new Linha();
44         FacesMessage msg = new FacesMessage("Linha cadastrada com sucesso!");
45         msg.setSeverity(FacesMessage.SEVERITY_INFO);
46         FacesContext.getCurrentInstance().addMessage("", msg);
47         return "/form_linha.jsf";
48     }
49
50     public Linha getLinha() {
51         return linha;
```

```
52 }
53
54 public void setLinha(Linha linha) {
55     this.linha = linha;
56 }
57
58 public List<Linha> getLinhas() {
59     return linhas;
60 }
61
62 public void setLinhas(List<Linha> linhas) {
63     this.linhas = linhas;
64 }
65
66 }
```

Listagem 13 - Código da classe CadastrarLinhaMBean.

A view para cadastrar linhas de ônibus é apresentada na Listagem 12. Essa página, denominada `form_linha.xhtml`, também deve ser incluída na pasta **WebContent** do seu projeto.


```

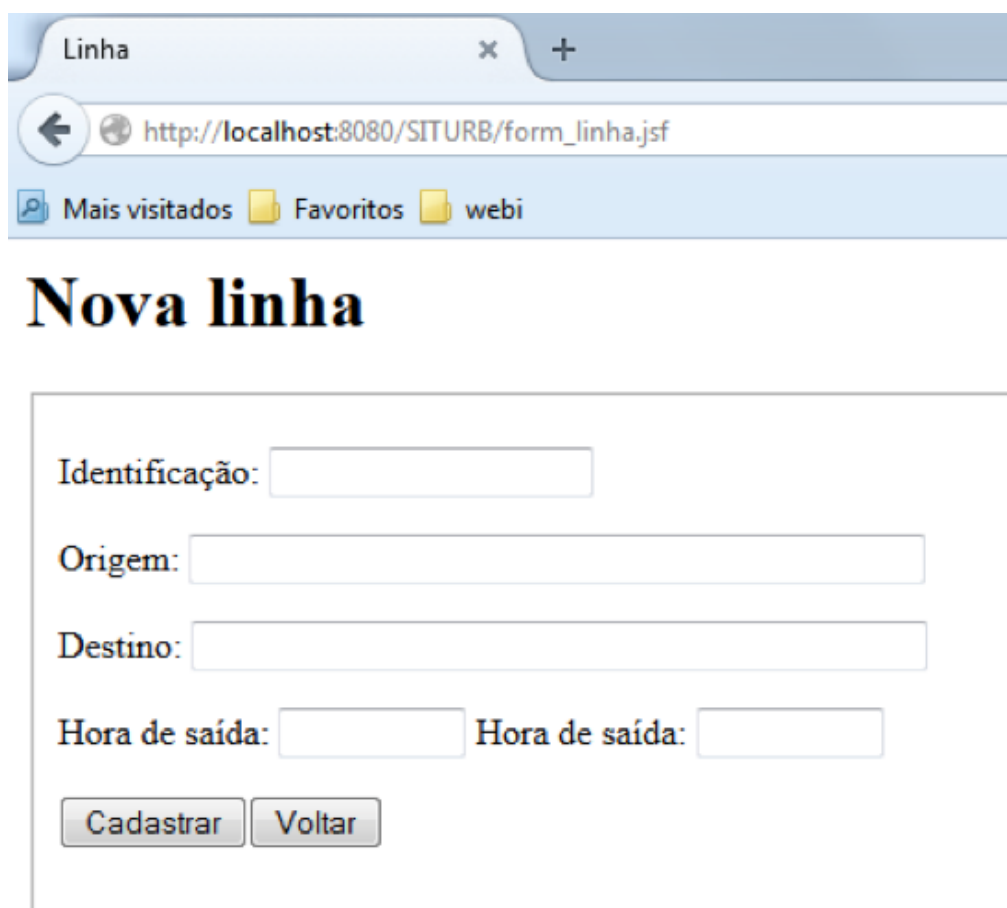
1 <html xmlns="http://www.w3.org/1999/xhtml"
2 xmlns:h="http://java.sun.com/jsf/html"
3 xmlns:f="http://java.sun.com/jsf/core">
4   <h:head>
5     <title>Linha</title>
6   </h:head>
7
8   <h:body>
9     <h:messages/>
10    <h1>Nova linha</h1>
11    <h:form>
12      <fieldset>
13        <p> Identificação: <h:inputText value="#{cadastrarLinhaMBean.linha.ident}"
14          size="20" required="true" requiredMessage="Identificação: Campo obrigatório." /> </p>
15        <p>Origem: <h:inputText value="#{cadastrarLinhaMBean.linha.origem}"
16          size="50" required="true" requiredMessage="Origem: Campo obrigatório." /> </p>
17        <p> Destino: <h:inputText value="#{cadastrarLinhaMBean.linha.destino}"
18          size="50" required="true" requiredMessage="Destino: Campo obrigatório." /></p>
19        <p>Hora de saída: <h:inputText value="#{cadastrarLinhaMBean.linha.horaSaida}"
20          size="10" required="true" requiredMessage="Hora de saída: Campo obrigatório." />
21          Hora de saída: <h:inputText value="#{cadastrarLinhaMBean.linha.horaChegada}"
22            size="10" required="true" requiredMessage="Hora de chegada: Campo obrigatório." />
23        </p>
24        <p><h:commandButton value="Cadastrar" action="#{cadastrarLinhaMBean.cadastrar}" />
25          <h:commandButton value="Voltar" action="#{cadastrarLinhaMBean.voltar}"
26            immediate="true"/> </p>
27      </fieldset>
28    </h:form>
29  </h:body>
30 </html>

```

Listagem 14 - Código da página form_linha.xhtml.

Ao implementar as Listagens 13 e 14, podemos acessar esse formulário pelo endereço `http://localhost:8080/SITURB/form_linha.jsf`, onde obtemos a página apresentada na Figura 4.

Figura 04 - Página para cadastrar linhas de ônibus.



The image shows a web browser window with a single tab titled 'Linha'. The address bar displays the URL 'http://localhost:8080/SITURB/form_linha.jsf'. Below the address bar, there are icons for 'Mais visitados', 'Favoritos', and 'webi'. The main content area features the heading 'Nova linha' in a large, bold, black serif font. Below the heading is a registration form enclosed in a thin black border. The form contains the following fields and buttons:

- 'Identificação:' followed by a text input field.
- 'Origem:' followed by a text input field.
- 'Destino:' followed by a text input field.
- 'Hora de saída:' followed by a text input field, and another 'Hora de saída:' followed by a text input field.
- At the bottom left of the form are two buttons: 'Cadastrar' and 'Voltar'.

Observe que já temos os cadastros de empresa, cobrador, motorista e linhas referentes ao nosso sistema de transportes urbanos. Então, vamos ao controller que realiza o cadastro de ônibus centralizando todas as informações cadastradas anteriormente: `CadastrarOnibusMBean`. Esse controller é apresentado na Listagem 15 e, assim como todos os outros controllers, deve ser incluído no pacote **br.ufrn.imd.controllers**.

```
1 package br.ufrn.imd.controllers;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 import javax.faces.application.FacesMessage;
7 import javax.faces.bean.ManagedBean;
8 import javax.faces.bean.SessionScoped;
9 import javax.faces.context.FacesContext;
10
11 import br.ufrn.imd.model.Cobrador;
12 import br.ufrn.imd.model.Empresa;
13 import br.ufrn.imd.model.Linha;
14 import br.ufrn.imd.model.Motorista;
15 import br.ufrn.imd.model.Onibus;
16
17 /**
18  * Controller para cadastrar os ônibus.
19  * @author itamir.filho
20  *
21  */
22 @ManagedBean
23 @SessionScoped
24
25 public class CadastrarOnibusMBean {
26
27     private Onibus onibus;
28
29     private List<Onibus> listagem;
30
31     public CadastrarOnibusMBean() {
32         iniciarValores();
33         listagem = new ArrayList<Onibus>();
34     }
35
36     private void iniciarValores() {
37         onibus = new Onibus();
38         onibus.setMotorista(new Motorista());
39         onibus.setLinha(new Linha());
40         onibus.setEmpresa(new Empresa());
41         onibus.setCobrador(new Cobrador());
42     }
43
44     public String entrarCadastro(){
45         return "/form_onibus.jsf";
46     }
47
48     public String listar(){
49         return "/list_onibus.jsf";
50     }
51 }
```

```

52 public String voltar(){
53     return "/index.jsf";
54 }
55
56 public String cadastrar() {
57     listagem.add(onibus);
58     iniciarValores();
59     FacesMessage msg = new FacesMessage("Ônibus cadastrado com sucesso!");
60     msg.setSeverity(FacesMessage.SEVERITY_INFO);
61     FacesContext.getCurrentInstance().addMessage("", msg);
62     return "/form_onibus.jsf";
63 }
64
65 public Onibus getOnibus() {
66     return onibus;
67 }
68
69 public void setOnibus(Onibus onibus) {
70     this.onibus = onibus;
71 }
72
73 public List<Onibus> getListagem() {
74     return listagem;
75 }
76
77 public void setListagem(List<Onibus> listagem) {
78     this.listagem = listagem;
79 }
80
81 }

```

Listagem 15 - Código da classe CadastrarOnibusMBean.

A view para cadastrar os ônibus é apresentada na Listagem 16. Essa página, denominada form_onibus.xhtml, também deve ser incluída na pasta **WebContent** do seu projeto.

```

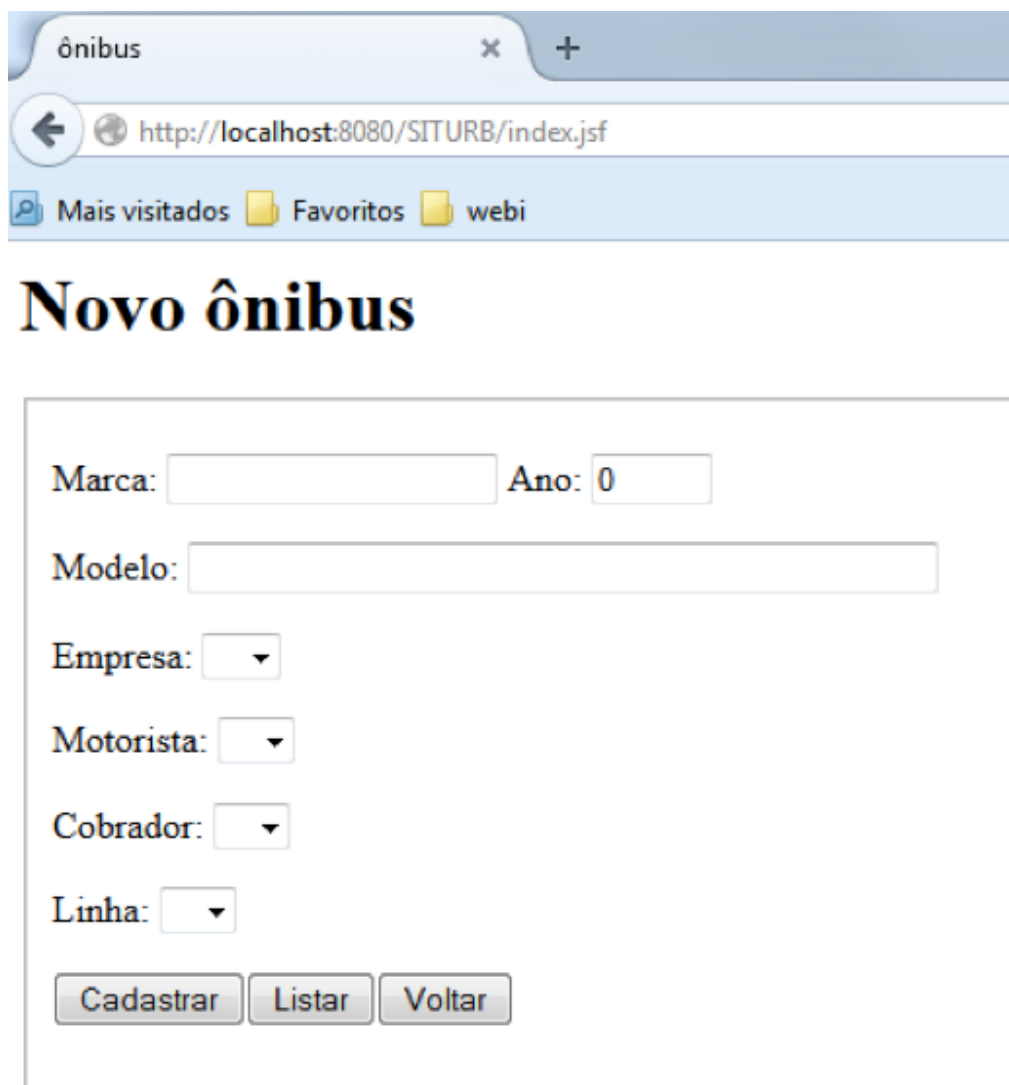
1 <html xmlns="http://www.w3.org/1999/xhtml"
2 xmlns:h="http://java.sun.com/jsf/html"
3 xmlns:f="http://java.sun.com/jsf/core">
4   <h:head>
5     <title>ônibus</title>
6   </h:head>
7
8   <h:body>
9     <h:messages/>
10    <h1>Novo ônibus</h1>
11    <h:form>
12      <fieldset>
13        <p>Marca: <h:inputText value="#{cadastrarOnibusMBean.onibus.marca}"
14          size="20" required="true" requiredMessage="Marca: Campo obrigatório." />
15        Ano: <h:inputText value="#{cadastrarOnibusMBean.onibus.ano}"
16          size="5" required="true" requiredMessage="Ano Campo obrigatório." />
17      </p>
18      <p> Modelo: <h:inputText value="#{cadastrarOnibusMBean.onibus.modelo}"
19        size="50" required="true" requiredMessage="Modelo: Campo obrigatório." /> </p>
20      <p> Empresa:
21        <h:selectOneMenu value="#{cadastrarOnibusMBean.onibus.empresa.razaoSocial}">
22          <f:selectItems value="#{cadastrarEmpresaMBean.empresas}" />
23        </h:selectOneMenu>
24      </p>
25      <p> Motorista:
26        <h:selectOneMenu value="#{cadastrarOnibusMBean.onibus.motorista.nome}">
27          <f:selectItems value="#{cadastrarMotoristaMBean.motoristas}" />
28        </h:selectOneMenu>
29      </p>
30      <p> Cobrador:
31        <h:selectOneMenu value="#{cadastrarOnibusMBean.onibus.cobrador.nome}">
32          <f:selectItems value="#{cadastrarCobradorMBean.cobradores}" />
33        </h:selectOneMenu>
34      </p>
35      <p> Linha:
36        <h:selectOneMenu value="#{cadastrarOnibusMBean.onibus.linha.ident}">
37          <f:selectItems value="#{cadastrarLinhaMBean.linhas}" />
38        </h:selectOneMenu>
39      </p>
40      <p><h:commandButton value="Cadastrar" action="#{cadastrarOnibusMBean.cadastrar}" />
41        <h:commandButton value="Listar" action="#{cadastrarOnibusMBean.listar}" immediate
42        <h:commandButton value="Voltar" action="#{cadastrarOnibusMBean.voltar}" immediate
43      </p>
44    </fieldset>
45  </h:form>
46 </h:body>
47 </html>

```

Listagem 16 - Código da página form_onibus.xhtml.

Ao implementar as Listagens 15 e 16, podemos acessar esse formulário pelo endereço `http://localhost:8080/SITURB/form_onibus.jsf`, onde obtemos a página apresentada na Figura 5.

Figura 05 - Página para cadastrar ônibus.



The screenshot shows a web browser window with a single tab titled 'ônibus'. The address bar displays the URL `http://localhost:8080/SITURB/index.jsf`. Below the address bar, there are navigation buttons: 'Mais visitados', 'Favoritos', and 'webi'. The main content area features a large heading 'Novo ônibus' in a bold, black, serif font. Below the heading is a registration form enclosed in a light gray border. The form contains the following fields and controls:

- Marca:** A text input field.
- Ano:** A text input field containing the value '0'.
- Modelo:** A wide text input field.
- Empresa:** A dropdown menu.
- Motorista:** A dropdown menu.
- Cobrador:** A dropdown menu.
- Linha:** A dropdown menu.
- At the bottom of the form are three buttons: 'Cadastrar', 'Listar', and 'Voltar'.

Observe na Figura 5 que utilizamos os dados informados nos demais cadastros: empresas, motoristas, cobradores e linhas de ônibus. Ainda com relação à Listagem 15, nós definimos um método denominado **listar** que apresentará as informações dos ônibus cadastrados, incluindo informações de marca, modelo, ano, empresa, motorista, cobrador e linha. Para exibição dessas informações, criaremos uma página dentro do **WebContent** denominada `list_onibus.xhtml`. Essa página tem seu código apresentado na Listagem 17.

```

1 <html xmlns="http://www.w3.org/1999/xhtml"
2 xmlns:h="http://java.sun.com/jsf/html"
3 xmlns:f="http://java.sun.com/jsf/core">
4   <h:head>
5     <title>ônibus</title>
6   </h:head>
7
8   <h:body>
9     <h:form>
10      <h:messages/>
11      <h1>Lista de ônibus</h1>
12      <h:dataTable value="#{cadastrarOnibusMBean.listagem}" var="onibus" border="1">
13        <h:column>
14          <f:facet name="header">Marca</f:facet>
15          #{onibus.marca}
16        </h:column>
17        <h:column>
18          <f:facet name="header">Modelo</f:facet>
19          #{onibus.modelo}
20        </h:column>
21        <h:column>
22          <f:facet name="header">Ano</f:facet>
23          #{onibus.ano}
24        </h:column>
25        <h:column>
26          <f:facet name="header">Empresa</f:facet>
27          #{onibus.empresa.razaoSocial}
28        </h:column>
29        <h:column>
30          <f:facet name="header">Motorista</f:facet>
31          #{onibus.motorista.nome}
32        </h:column>
33        <h:column>
34          <f:facet name="header">Cobrador</f:facet>
35          #{onibus.cobrador.nome}
36        </h:column>
37        <h:column>
38          <f:facet name="header">Linha</f:facet>
39          #{onibus.linha.ident}
40        </h:column>
41      </h:dataTable>
42      <h:commandLink action="#{cadastrarOnibusMBean.entrarCadastro}" value="Voltar"/> <br/>
43    </h:form>
44  </h:body>
45 </html>

```

Listagem 17 - Código da página list_onibus.xhtml.

Ao implementar a Listagem 17, podemos acessar essa lista de ônibus pelo endereço http://localhost:8080/SITURB/list_onibus.jsf, onde obtemos a página apresentada na Figura 6.

Figura 06 - Página que apresenta a lista de ônibus cadastrados.



Por fim, implementaremos nossa página principal denominada index.xhtml com links para as demais páginas dos cadastros. Essa página, que deve ser criada dentro da pasta **WebContent**, é apresentada na Figura 7 e tem seu código descrito na Listagem 18.

```
1 <html xmlns="http://www.w3.org/1999/xhtml"
2 xmlns:h="http://java.sun.com/jsf/html"
3 xmlns:f="http://java.sun.com/jsf/core">
4   <h:head>
5     <title>SITURB</title>
6   </h:head>
7
8   <h:body>
9     <h1>SITURB - Sistema de Transportes Urbanos</h1>
10    <hr/>
11    <h2>Operações:</h2>
12    <h:form>
13      <h:commandLink action="#{cadastrarEmpresaMBean.entrarCadastro}" value="Cadastrar Empresa" />
14      <h:commandLink action="#{cadastrarCobradorMBean.entrarCadastro}" value="Cadastrar Cobrador" />
15      <h:commandLink action="#{cadastrarMotoristaMBean.entrarCadastro}" value="Cadastrar Motorista" />
16      <h:commandLink action="#{cadastrarLinhaMBean.entrarCadastro}" value="Cadastrar Linha" />
17      <h:commandLink action="#{cadastrarOnibusMBean.entrarCadastro}" value="Cadastrar Ônibus" />
18    </h:form>
19    <hr/>
20  </h:body>
21</html>
```

Listagem 18 - Código da página index.xhtml.

Figura 07 - Página com links para as operações de cadastro.



E, assim, chegamos ao fim da nossa aula. Você está de parabéns, pois conseguiu desenvolver um sistema com várias funcionalidades acessíveis a partir de uma página principal (index.xhtml). Dessa forma, concluímos nosso curso e agora você está pronto para desenvolver suas aplicações web usando o framework JSF.

Resumo

Nessa aula, desenvolvemos uma aplicação web com o framework JSF utilizando os conhecimentos obtidos até agora. Essa aplicação foi organizada segundo o padrão MVC, sendo implementados o modelo (model), os managed beans (controllers) e as páginas XHTML (views).

Referências

<https://docs.oracle.com/javaee/6/javaxserverfaces/facelets/>

http://www.tutorialspoint.com/jsf/jsf_basic_tags.htm

<http://www.horstmann.com/corejsf/jsf-tags.html>

<http://www.jsftoolbox.com/documentation/help/12-TagReference/core/index.jsf>