

FIAP GRADUAÇÃO

DIGITAL BUSINESS ENABLEMENT

Prof. THIAGO T. I. YAMAMOTO

#11 - AJAX E PRIMEFACES



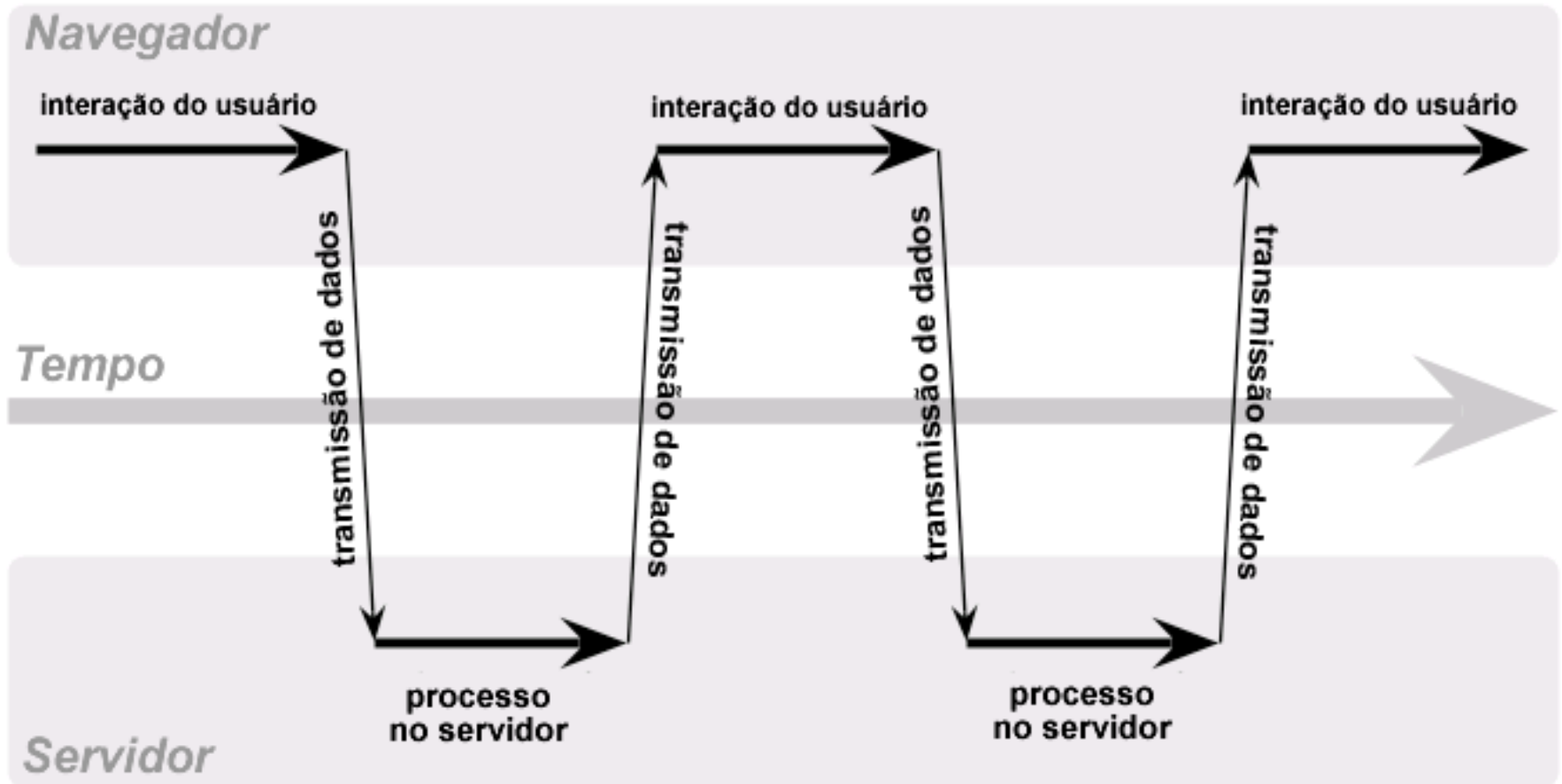
- ♦ Ajax com JSF
- ♦ Primefaces
- ♦ Componentes de formulários
- ♦ Tabelas
- ♦ Painéis, abas e menus
- ♦ Dialogs
- ♦ Gráficos
- ♦ Upload de arquivo

AJAX

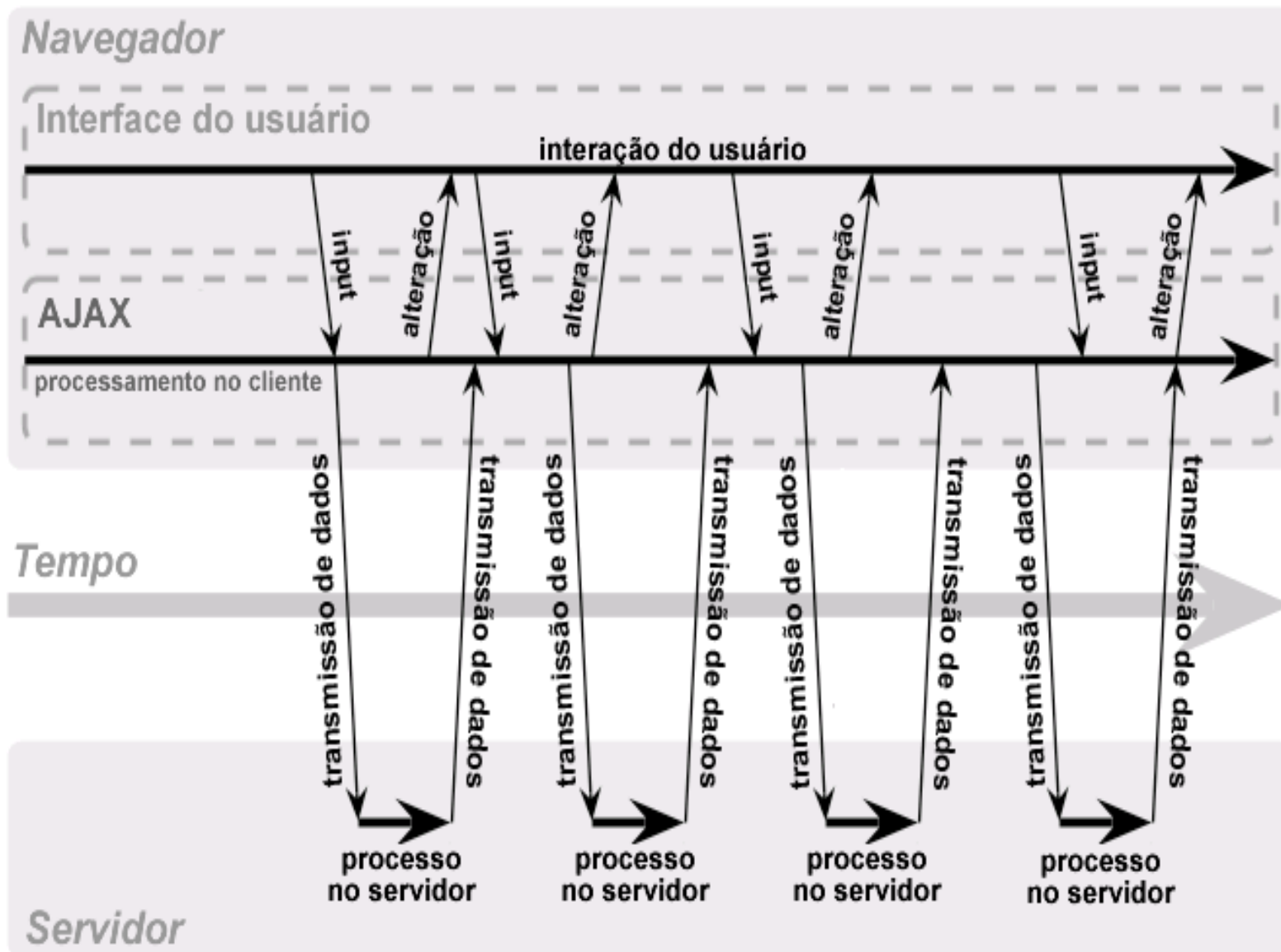
AJAX (*Asynchronous Javascript and XML*) é uma técnica que permite o desenvolvimento de aplicações web mais responsíveis e com interface mais sofisticada

PARADIGMA REQUEST/WAIT/RESPONSE FIAP

Padrão clássico de uma aplicação Web (*Request / Wait / Response*)



Aplicação web utilizando AJAX



- As novas versões de JSF estão provendo forte integração com AJAX
- O uso de AJAX com JSF pode ser provido diretamente com bibliotecas javascript (Jquery ou DOJO)
- A interação com AJAX em JSF pode ser encapsulada utilizando o modelo de componente de JSF já existentes:
 - JSF (Bibliotecas de funcionamento básico)
 - **Primefaces** (<http://www.primefaces.org>)
 - Richfaces (<http://www.jboss.org/richfaces>)
 - Icefaces (<http://www.icefaces.org/main/home/>)

PRIMEFACES

Standard UI Components: são os componentes padrão do JSF, como por exemplo:

- **UIForm:** Representa um formulário de entrada de informações pelo usuário. Ele é utilizado como container de outros componentes, pois é possível coloca-los dentro do formulário e enviá-los ao ManagedBean através de uma única ação;
- **UICommand:** Representa componentes como: Botões, Hyperlinks e itens de Menu;
- **UIInput:** Representa componentes como: Campos de entrada de texto, Campos de entrada de números, campos de senhas, etc..

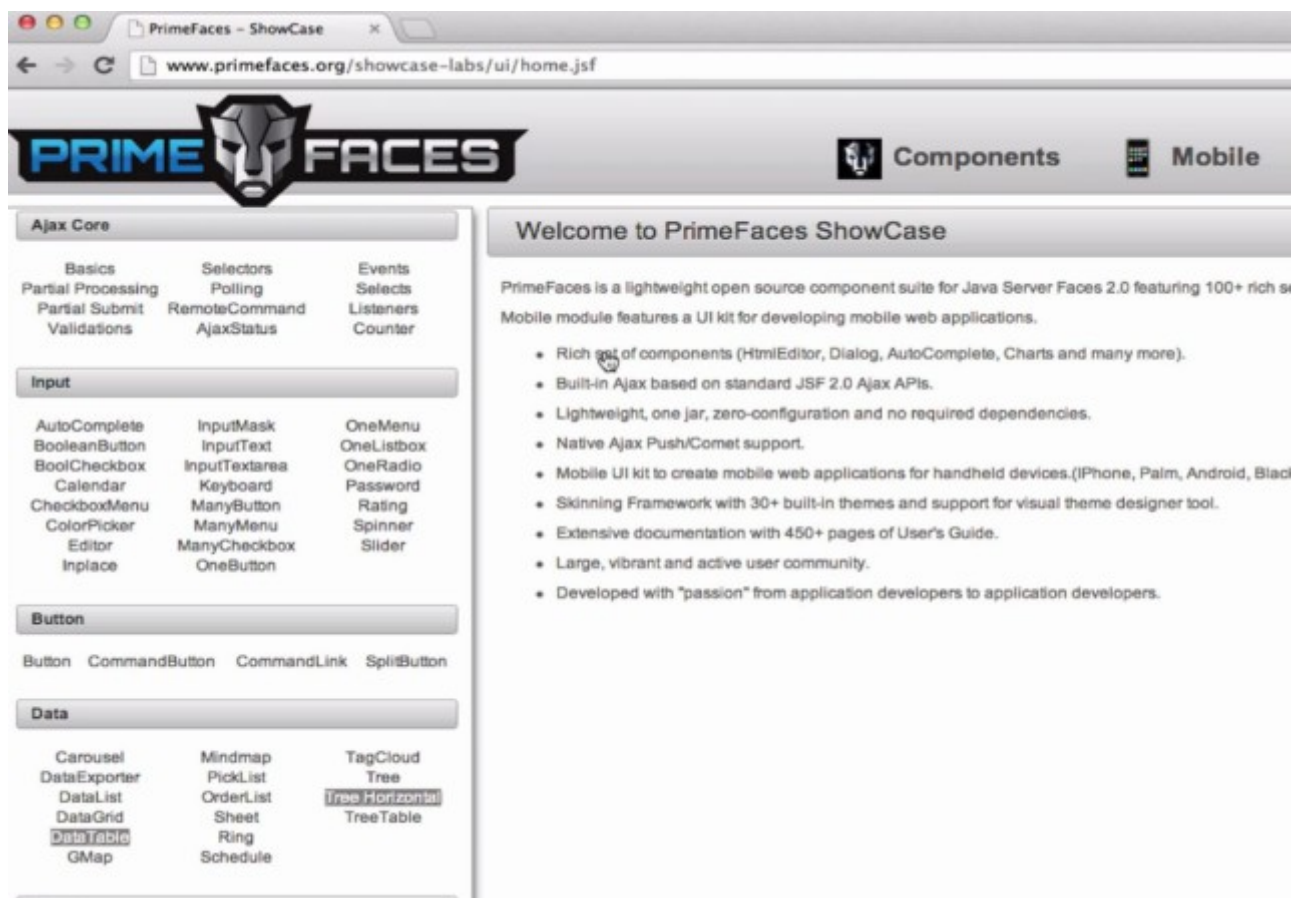
Custom UI Components: São componentes customizados que normalmente são desenvolvidos por diversos fornecedores. Podem ser utilizados quando necessitamos de componentes com mais estilo ou robustez, economizando tempo com desenvolvimento. Alguns de seus fornecedores são: Richfaces (Jboss), Primefaces, Apache Tomahawk , GMaps4JSF(Google) entre outros.

CONFIGURAÇÃO DO PRIMEFACES

Para configurar o PrimeFaces é necessário importar a biblioteca (WEB-INF\lib\primefaces-5.3.jar)

Uma outra opção é copiar o arquivo no tomcat para deployment diretamente no servidor

<http://primefaces.org/index.html>



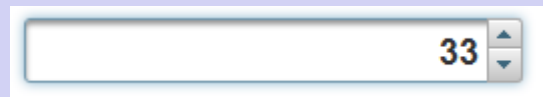
- Referência da biblioteca em faces é *http://primefaces.org/ui*
- Obrigatório o uso da estrutura `<h:head>` e `<h:body>` para permitir que o framework controle as requisições Ajax
- Componente **spinner** permite a entrada de valores numéricos com setas

```
<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:p="http://primefaces.org/ui">
```

```
<h:head></h:head>
```

```
<h:body>
```

```
    <p:spinner />
```



```
</h:body>
```

```
</html>
```

XHTML

VALIDAÇÃO COM AJAX

- Podemos utilizar ajax para fazer validações de campos do formulário.

Cadastro de Cliente

☒ RG: Erro de validação: o valor é necessário.

Cliente

Código 0 CPF * 0

RG * Nome *

Email * Telefone * 0

Data Nascimento * 27/04/2013 Endereço *

CEP * 0 Cidade *

Estado *

Enviar Cadastrar Novo

```
<p:messages id="erros"/>
```

XHTML

```
<p:inputText required="true" value="#{clienteBean.cliente.rg}" id="rg" >
```

```
  <p:ajax update="erros"/>
```

```
</p:inputText>
```

Define a área que será atualizada

Valida o campo no onBlur (perda de foco) Podemos alterar o evento com o atributo event. Ex. `event="keyup"`

- Podemos enviar para um processamento somente partes específicas da tela utilizando Ajax.

```
<p:commandButton value="Enviar" id="btnEnviar" process="id" update="id"
  actionListener="#{bean.metodo}" >
```

XHTML

Id dos elementos que serão enviados

Id dos elementos que serão atualizados

Podemos utilizar palavras chaves para identificar grupos de componentes:

@all refere-se a todos os componentes da tela.

@none refere-se a nenhum componente.

@this refere-se ao componente que disparou a requisição AJAX.

@form refere-se aos componentes do formulário.

EXIBINDO MENSAGENS

<p: growl />

The screenshot displays a web application interface with a dark header and a light gray body. The header contains the FIAP logo and navigation links: Home, Cadastro, Busca, and Contato. A red box highlights a validation error message in the top right corner, which reads: "RG: Erro de validação: o valor é necessário." Below the header, the "Cadastro de Cliente" form is visible. It features a tab labeled "Cliente" and a validation error message at the top: "RG: Erro de validação: o valor é necessário." The form contains several input fields: Código (0), RG (empty), CPF (0), Nome (empty), Email (empty), Telefone (0), Data Nascimento (27/04/2013), Endereço (empty), CEP (0), Cidade (empty), and Estado (empty). At the bottom of the form, there are two buttons: "Enviar" and "Cadastrar Novo".

AUTOCOMPLETE

Podemos utilizar um componente para buscar informações cadastradas no banco de dados e disponibilizar para o usuário no auto-complete.

The screenshot shows a web interface for searching clients. At the top, there is a button labeled "Buscar Cliente". Below it, a search form contains a text input field with the text "Thia" and a "Buscar" button. A dropdown menu is open below the input field, showing two suggestions: "Thiago" and "Thiago T. I. Yamamoto". Below the search form, there is a table with the following data:

Código	Nome	Sexo	Data de Nascimento
1	Matador	MASCULINO	05/05/2013

At the bottom of the interface, there is a summary bar that reads "Total de Clientes: 1".


```
public List<String> buscarPorNomeCliente(String nome) {  
    TypedQuery<String> query = em.createQuery("select c.nome from Cliente c where  
    c.nome like :nome",String.class);  
    query.setParameter("nome","%" + nome + "%");  
    return query.getResultList();  
}
```

DAO

```
public List<String> completaNomeCliente(String nome){  
    return clienteDAO.buscarPorNomeCliente(nome);  
}
```

ManagedBean

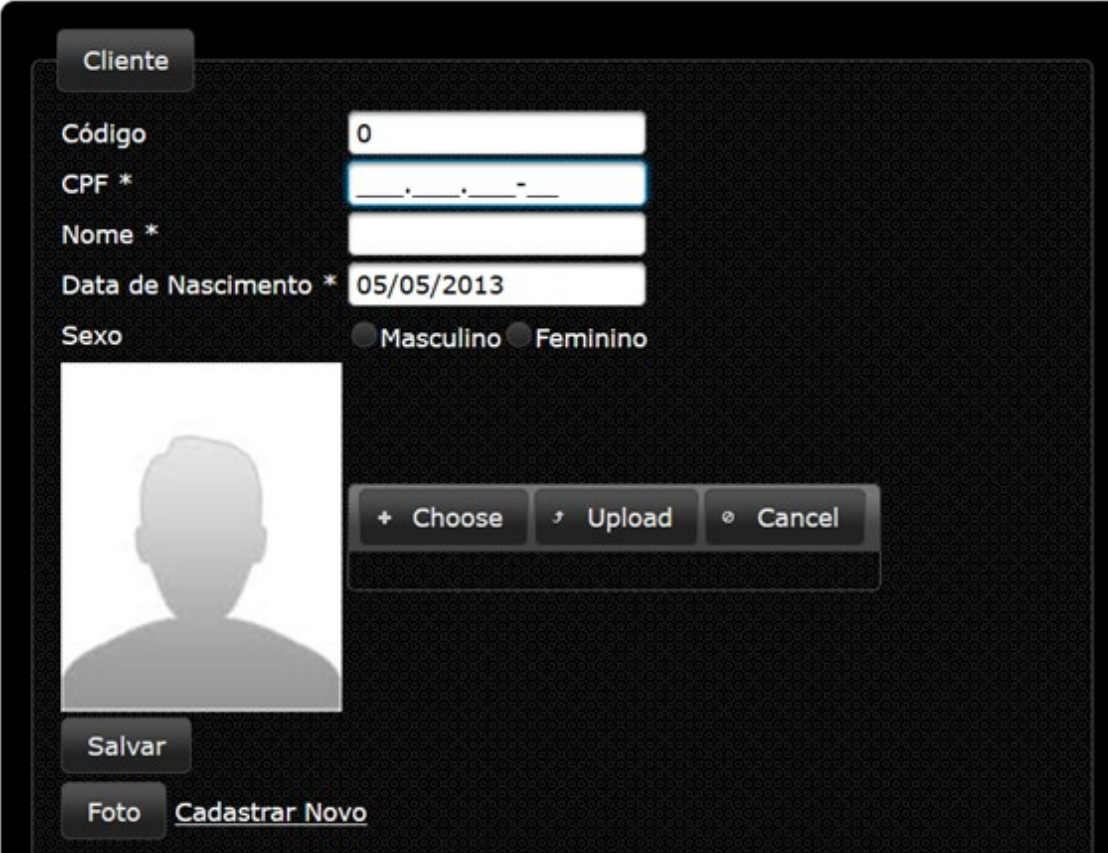
```
<h:form>  
<p:autoComplete id="nome" value="#{clienteBean.nomeBusca}"  
completeMethod="#{clienteBean.completaNomeCliente}"/>  
<p:commandButton value="Enviar" action="#{clienteBean.buscar}"/>  
</h:form>
```

XHTML

PRIMEFACES COMPONENTES DE FORMULÁRIO

COMPONENTES DE FORMULÁRIO

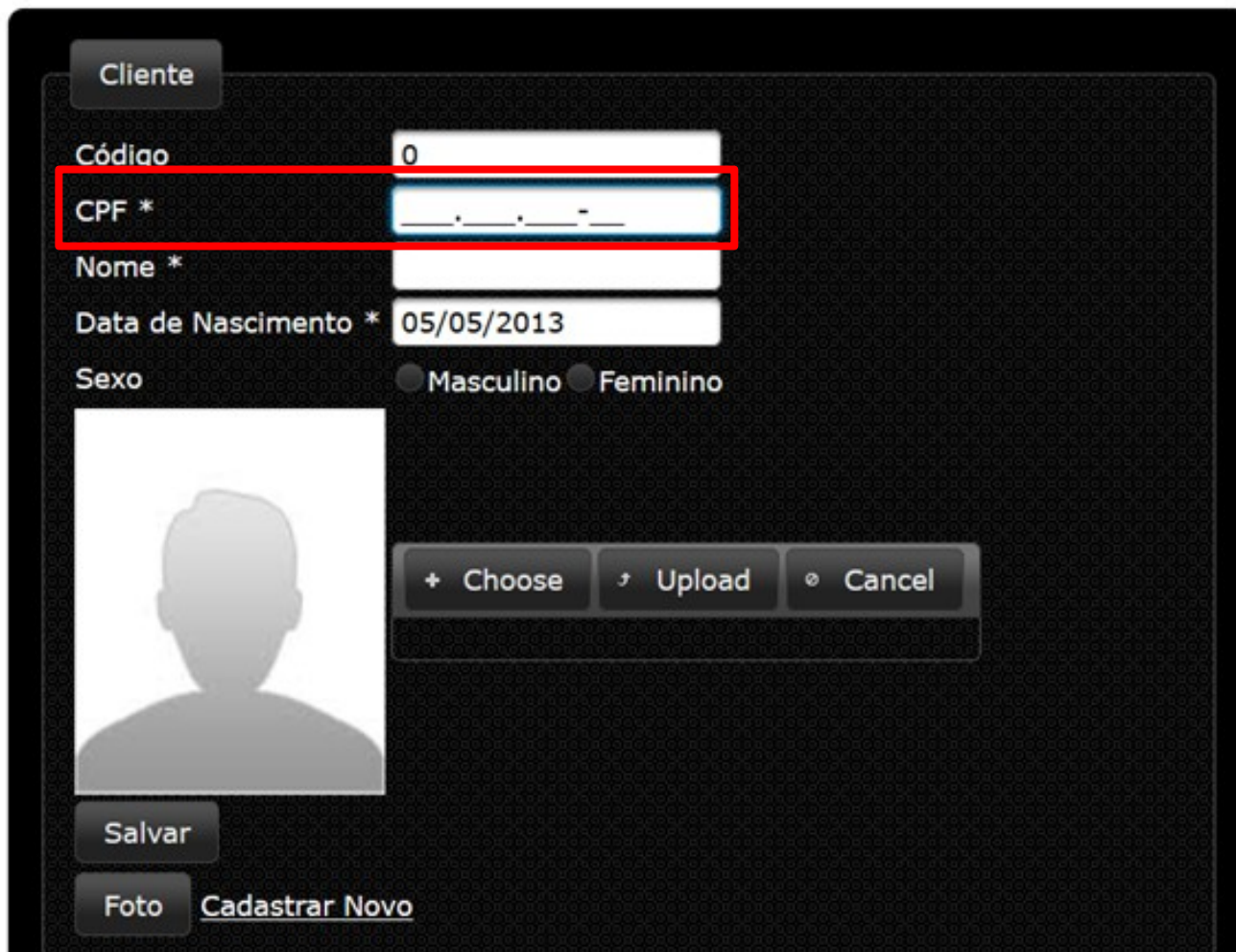
O Primefaces possui vários componentes customizados, além dos seus components correspondentes que já estudamos no JSF.



A screenshot of a PrimeFaces web form titled "Cliente" (Client). The form is set against a dark background. It contains several input fields: "Código" with the value "0", "CPF *" with a masked input (____.____.____-__), "Nome *" with an empty text field, and "Data de Nascimento *" with the value "05/05/2013". Below these is a "Sexo" section with two radio buttons: "Masculino" (selected) and "Feminino". To the left of the form is a placeholder for a profile picture, showing a grey silhouette. To the right of the placeholder are three buttons: "+ Choose", "Upload", and "Cancel". At the bottom left of the form area are two buttons: "Salvar" and "Foto". At the bottom right, there is a link "Cadastrar Novo".

MÁSCARA

Podemos forçar um formato adequado para o preenchimento de um campo.



Cliente


Código 0

CPF *

Nome *

Data de Nascimento * 05/05/2013

Sexo ☒ Masculino ☐ Feminino



+ Choose ↗ Upload ⌵ Cancel

Salvar

Foto [Cadastrar Novo](#)

XHTML

```
<p:outputLabel value="CPF" for="cpf" />
```

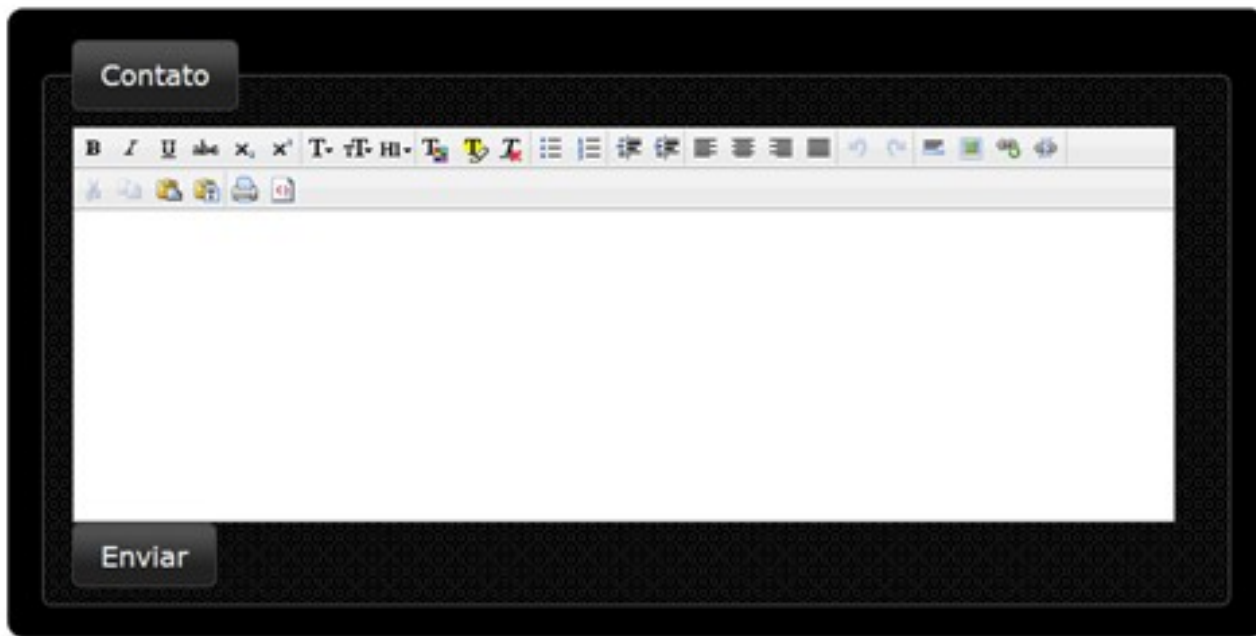
```
<p:inputMask required="true"  
value="#{clienteBean.cliente.cpf}"id="cpf"  
mask="999.999.999-99" />
```

```
<p:outputLabel value="#Telefone" for="telefone" />
```

```
<p:inputMask required="true"  
value="#{clienteBean.cliente.telefone}"id="telefone"  
mask="(99) 9999-9999" />
```

XHTML

```
<h:form>  
<p:editor width="700" value="#{contatoBean.conteudo}"/>  
<p:commandButton value="Enviar" action="#{contatoBean.enviar}"/>  
</h:form>
```



Package

Code

0

Description *

Departure Date *

05/05/2013

Amount of Days

0

Price

0.0

Rating

Save

[Register New](#)

Search by Package

Search by Description

Search by Departure Date

Description

Search

1

Code	Description	Departure Date	Amount of Days	Price	Rating
1	Teste	05/05/2013	2	0.0	

//Avaliação é uma propriedade int da Entidade Pacote

<p:outputLabel value="Avaliação" for="avaliacao" />

<p:rating value="#{pacoteBean.pacote.avaliacao}" id="avaliacao"/>

// Para a tabela, “pac” é a variavel de cada item da lista.

<p:column headerText="#{pacoteBundle.rating_field}">

<p:rating value="#{pac.avaliacao}" disabled="true"/>

</p:column>

XHTML

PRIMEFACES PAINÉIS, ABAS E MENUS

Painel aberto:



The screenshot shows a web application window titled "Cadastro de Cliente". Inside the window, there is a tab labeled "Cliente". Below the tab, there is a form with the following fields:

Código	0	CPF *	0
RG *	0	Nome *	
Email *		Telefone *	0
Data Nascimento *	27/04/2013	Endereço *	
CEP *	0	Cidade *	
Estado *			

At the bottom of the form, there are two buttons: "Enviar" and "Cadastrar Novo".

Painel fechado:



The screenshot shows the "Cadastro de Cliente" panel closed. Only the title bar is visible, which contains the text "Cadastro de Cliente" and a close button (X).

```
<p:panel header="Cadastro de Cliente"
  toggleable="true" toggleOrientation="vertical">
```

Código.....

```
</:panel>
```

XHTML

Busca por Período

Busca Por Status

Busca por Cliente

Busca Por Cliente

Nome Cliente Buscar

	Código	Descrição/Destino	Cliente	Status	Data
Alterar Excluir	5	Natal	Thiago	PENDENTE	27/04/2013



```
<p:tabView>
```

```
  <p:tab title="Busca por Descrição">
```

```
    <!-- Código da aba... -->
```

```
  </p:tab>
```

```
  <p:tab title="Busca por Data de Saída">
```

```
    <!-- Código da aba... -->
```

```
  </p:tab>
```

```
</p:tabView>
```

XHTML

Podemos habilitar ou desabilitar uma aba utilizando a propriedade **"disable"** do componente **p:tab**, esta propriedade recebe um valor **booleano**.

I BARRA DE MENU

- Primefaces possui vários componentes de menus;
- Vamos alterar o menu principal da nossa aplicação, assim precisamos alterar o arquivo _template.xhtml.

The screenshot displays a web application interface. At the top, there is a dark navigation bar with the following menu items: "Home", "Cliente" (highlighted in red), "Cotação", "Relatórios", and "Voucher". A dropdown menu is open under "Cliente", showing two options: "Cadastrar" and "Buscar". Below the navigation bar, there is a registration form titled "ite". The form contains several input fields arranged in two columns. The left column includes fields for "Código", "RG *", "Email *", "Data Nascimento *", "CEP *", and "Estado *". The right column includes fields for "CPF *", "Nome *", "Telefone *", "Endereço *", and "Cidade *". Some fields already contain data: "Data Nascimento *" has "27/04/2013", "CPF *" has "0", and "Telefone *" has "0". At the bottom left of the form, there is a button labeled "Enviar". At the bottom center, there is a link labeled "Cadastrar Novo".

XHTML

```
<p:menubar>
  <p:menuitem value="Home" outcome="login"/>
  <p:submenu label="Cliente">
    <p:menuitem value="Cadastrar" outcome="cliente"/>
    <p:menuitem value="Buscar" outcome="buscaCliente"/>
  </p:submenu>
  <p:submenu label="Cotação">
    <p:menuitem value="Cadastrar" outcome="cotacao"/>
    <p:menuitem value="Buscar" outcome="buscaCotacao"/>
  </p:submenu>
  <p:submenu label="Relatórios">
    <p:menuitem value="Grafico Pizza" outcome="graficoPizza"/>
    <p:menuitem value="Gráfico Linha" outcome="graficoLinha"/>
  </p:submenu>
  <p:menuitem value="Voucher" outcome="voucher"/>
</p:menubar>
```

O menuitem possui action e actionListener para executar ações no managedBean.

PRIMEFACES TABELAS

- Muitas aplicações necessitam manipular conjunto de dados
- O JPA pode solicitar este conjunto de dados utilizando a instrução `select`
- O JSF permite manipular coleções de dados
- Os componentes **dataTable** são responsáveis decompor os itens de uma coleção e apresentá-los como dados.
- Os dataTable são subdivididos em conteúdo, cabeçalho e rodapé.
- O JSF permite você incluir ações nas listas de dados por meio de `ActionListeners` que recebem as propriedades dos valores selecionados (`PropertyActionListener`)

Busca Cliente

Nome

Buscar

				1 2					
		CPF	RG	Nome	Email	Telefone	Data Nascimento	Endereço	Cidade
Alterar	Excluir	123	123	Thiago	thiago@gmail.com	123	22/04/2013	Rua Lins	Sao Paulo
Alterar	Excluir	123	123	Leandro	leandro@fiap.com	123	22/04/2013	Rua Marota	Rio de Janeiro
				1 2					
Total de Clientes: 4									

```
<p:dataTable value="#{clienteBean.lista}" var="cli" paginator="true" rows="2">
  <p:column headerText="CPF">
    #{cli.cpf}
  </p:column>
  <p:column headerText="RG">
    #{cli.rg}
  </p:column>
  <p:column headerText="Data Nascimento">
    <h:outputText value="#{cli.data.time}">
      <f:convertDateTime pattern="dd/MM/yyyy" />
    </h:outputText>
  </p:column>
  <p:column headerText="Endereço">
    #{cli.endereco}
  </p:column>
  <f:facet name="footer">
    <h:outputText value="Total de Clientes: #{clienteBean.totalCliente}" />
  </f:facet>
</p:dataTable>
```

```
@ManagedBean
@ViewScoped
public class ClienteBean implements Serializable {

    private int totalCliente;

    private List<Cliente> lista;

    private String nome;

    public void buscarTodos() {
        lista = clienteDao.buscarPorNome(nome);
        totalCliente = lista.size();
    }

    //mais códigos...
}
```

! DATATABLE - PROPRIEDADES

```
<p:dataTable value="#{clienteBean.lista}" var="cli"  
    paginator="true" rows="2"  
    emptyMessage="Não foram encontrados clientes!"  
    rowsPerPageTemplate="5,10,15"  
    sortOrder="acending">  
  
    <p:column sortBy="#{cli.cpf}" headerText="CPF">  
        #{cli.cpf}  
    </p:column>  
  
</dataTable>
```

CPF	RG	Nome	Email	Telefone	Data Nascimento	Endereço	Cidade
Não foram encontrados clientes!							
Total de Clientes: 0							

CPF

RG

Alterar

Excluir

123

123

Thiago

thiago@gmail.com

123

22/04/2013

Rua Lins

Sao Paulo

Alterar

Excluir

123

123

Leandro

leandro@fiap.com

123

22/04/2013

Rua Marota

Rio de Janeiro

5

10

15

1

2

Total de Clientes: 4

DATATABLE - AÇÕES EM LINHAS DA TABELA FIAP

```
<p:dataTable value="#{clienteBean.lista}" var="cli" id="tabelaCliente">
<p:column>
    <p:commandButton value="Alterar" action="#{clienteBean.alterar}" >
        <f:setPropertyActionListener target="#{clienteBean.cliente}" value="#{cli}"/>
    </p:commandButton>
    <p:commandButton value="Excluir" action="#{clienteBean.excluir}" update="tabelaCliente">
        <f:setPropertyActionListener target="#{clienteBean.cliente}" value="#{cli}" />
    </p:commandButton>
</p:column>
```

XHTML

		CPF	RG	Nome	Email	Telefone	Data Nascimento	Endereço	Cidade
Alterar	Excluir	123123	123	Ttt	sdfsdf	123	22/04/2013	sdfa	sdf
Alterar	Excluir	123	123	Thiago	fsa	234	22/04/2013	ds	123
Total de Clientes: 4									

| DATATABLE - AÇÕES EM LINHAS DA TABELA FIAP

Java

```
@ManagedBean
```

```
@ViewScoped
```

```
public class ClienteBean implements Serializable {
```

```
    private int totalCliente;
```

```
    private List<Cliente> lista;
```

```
    private Cliente cliente;
```

```
    private String nome;
```

```
    public void excluir() {
```

```
        clienteDao.remove(cliente);
```

```
        FacesContext.getCurrentInstance().addMessage(null,  
                                                    new FacesMessage("Usuário excluído com sucesso"));
```

```
    }
```

```
    //mais códigos...
```

```
}
```

ÍCONES - BUTTONS

- É possível associar ícones aos botões.



```
image="ui-icon ui-icon-disk"  
image="ui-icon-disk"  
image="ui-icon-search"  
image="ui-icon-pencil"  
image="ui-icon-trash"  
image="ui-icon-print"
```

```
<p:commandButton action="#{clienteBean.alterar}" icon="ui-icon-search">  
    <f:setPropertyActionListener target="#{clienteBean.cliente}" value="#{cli}" />  
</p:commandButton>  
<p:commandButton icon="ui-icon-close" action="#{clienteBean.excluir}" update="tabelaCliente">  
    <f:setPropertyActionListener target="#{clienteBean.cliente}" value="#{cli}" />  
</p:commandButton>
```

XHTML

■ DATATABLE - SELEÇÃO DE LINHA

- Podemos selecionar uma linha com o mouse com um clique;
- Para tirar a seleção, basta segurar a tecla ctr e dar um clique.



	CPF	RG	Nome	Email	Telefone	Data Nascimento	Endereço	Cidade
 	123	123	Thiago	thiago@gmail.com	123	22/04/2013	Rua Lins	Sao Paulo
 	123	123	Leandro	leandro@fiap.com	123	22/04/2013	Rua Marota	Rio de Janeiro
 	123	0	Debora	teste@teste.com	0	26/04/2013		Sao Paulo

XHTML

```
<p:dataTable
id="tabelaCliente" value="#{clienteBean.lista}" var="cli"
rowsPerPageTemplate="5,10,15"
emptyMessage="Não foram encontrados clientes!"
selectionMode="single" rowKey="#{cli.codigo}" selection="#{clienteBean.cliente}"
sortOrder="acending" paginator="true" rows="2">
```

```
<f:facet name="header" >
    <div style="text-align: left;">
        <p:commandButton value="Alterar" icon="ui-icon-search"
            action="#{clienteBean.alterar}"/>
    </div>
</f:facet>
```

Colunas da tabela...

```
</p:dataTable>
```

rowKey: id do cliente;
selection: Atributo no ManagedBean
que receberá o cliente selecionado.

DATATABLE - SELEÇÃO DE MÚLTIPLAS LINHAS

- Podemos selecionar várias linhas utilizando a tecla ctr.



	CPF	RG	Nome	Email	Telefone	Data Nascimento	Endereço	Cidade
 	123	123	Thiago	thiago@gmail.com	123	22/04/2013	Rua Lins	Sao Paulo
 	123	123	Leandro	leandro@fiap.com	123	22/04/2013	Rua Marota	Rio de Janeiro
 	123	0	Debora	teste@teste.com	0	26/04/2013		Sao Paulo

```
<p:dataTable
id="tabelaCliente" value="#{clienteBean.lista}" var="cli"
rowsPerPageTemplate="5,10,15"
emptyMessage="Não foram encontrados clientes!"
selectionMode="multiple" rowKey="#{cli.codigo}" selection="#{clienteBean.lista}"
sortOrder="acending"paginator="true" rows="2">

<f:facet name="header" >
    <div style="text-align: left;">
        <p:commandButton value="Alterar" icon="ui-icon-search"
            action="#{clienteBean.alterar}"/>
    </div>
</f:facet>

Colunas da tabela...

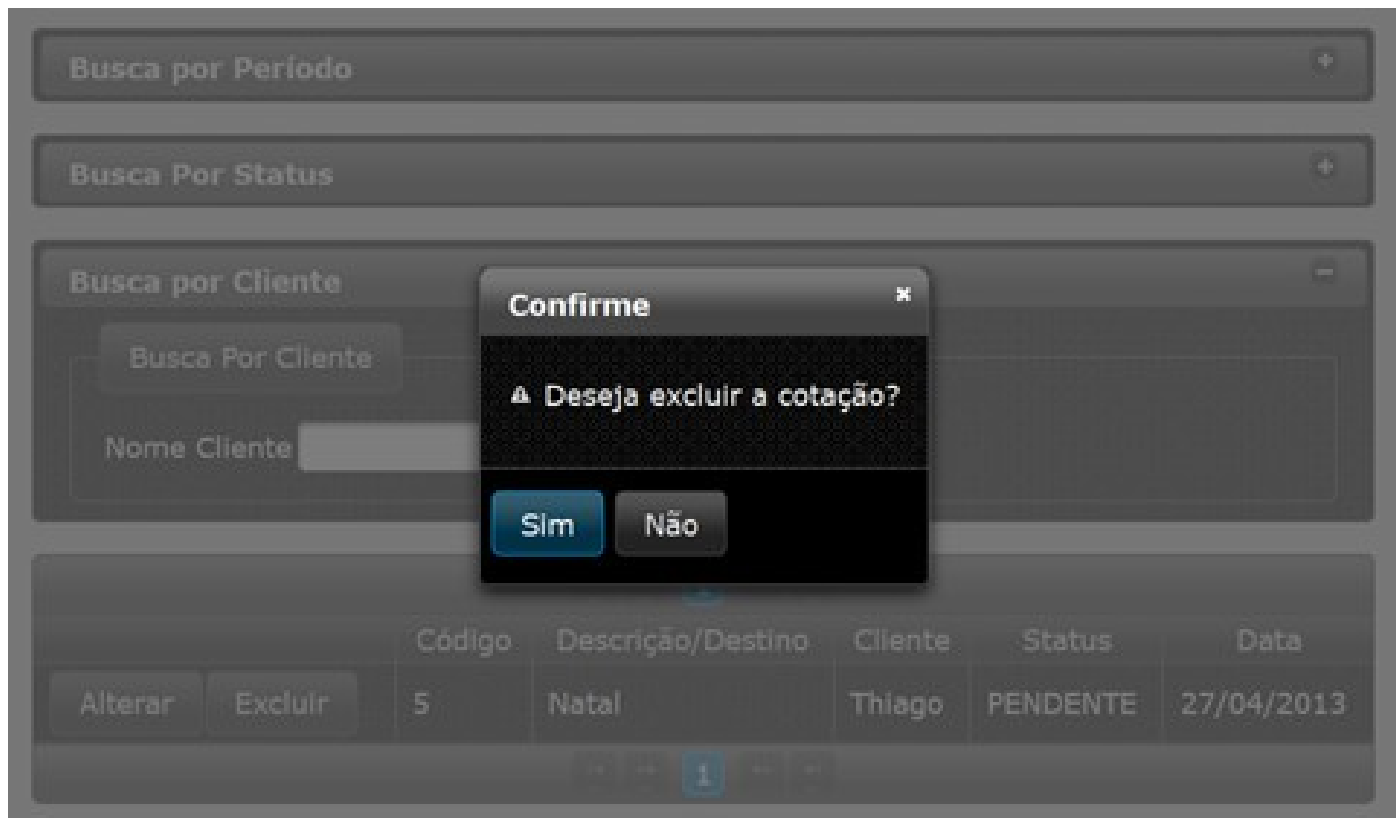
</p:dataTable>
```

ManagedBean deve possuir um atributo:
List<Cliente> lista;
Para receber os clientes selecionados.

PRIMEFACES DIALOGS

CONFIRM DIALOG

Caixa de diálogo para confirmação de alguma ação:

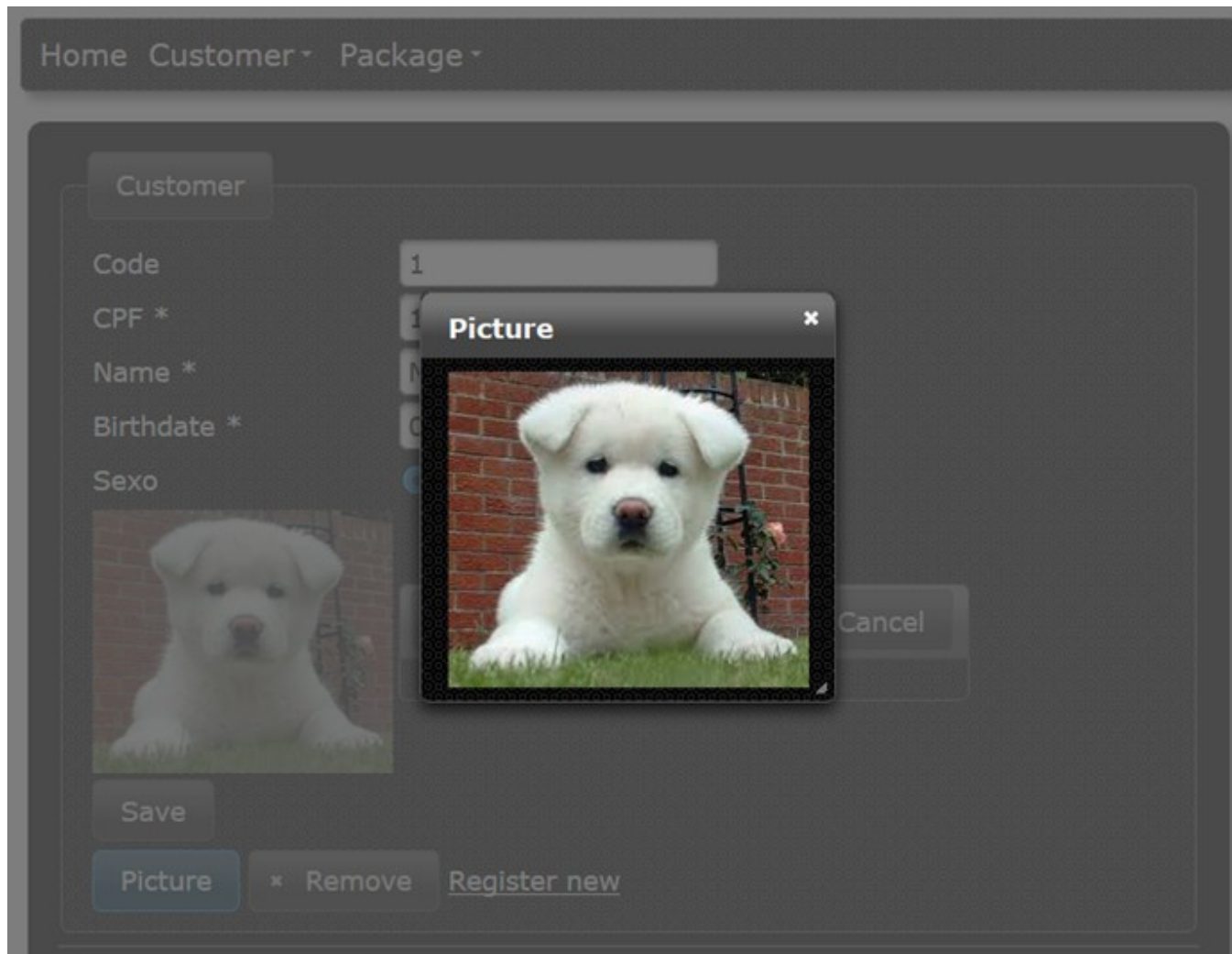


```
<p:dataTable value="#{cotacaoBean.lista}" var="cot">
<p:column>
<p:commandButton>
<p:commandButton value="Excluir" onclick="PF('confirmation').show()">
<f:setPropertyActionListener target="#{cotacaoBean.cotacao}" value="#{cot}" />
    </p:commandButton>
</p:column>

<p:confirmDialog id="confirmDialog" message="Deseja excluir a cotação?" header="Confirme"
severity="alert" widgetVar="confirmation">
    <p:commandButton id="confirm" value="Sim" oncomplete="PF('confirmation').hide()"
        actionListener="#{cotacaoBean.excluir()}" />
        Método de excluir
        no ManagedBean

    <p:commandButton id="decline" value="Não" onclick="PF('confirmation').hide()"
type="button" />
</p:confirmDialog>
```

É possível adicionar caixas de dialog nas páginas. Elas podem conter imagens, textos e até formulários.



XHTML

```
<p:commandButton value="Mostrar Foto"
                 onclick="PF('fotoDialog').show()"/>

<p:dialog header="Foto" widgetVar="fotoDialog" modal="true"
          showEffect="bounce" hideEffect="explode">

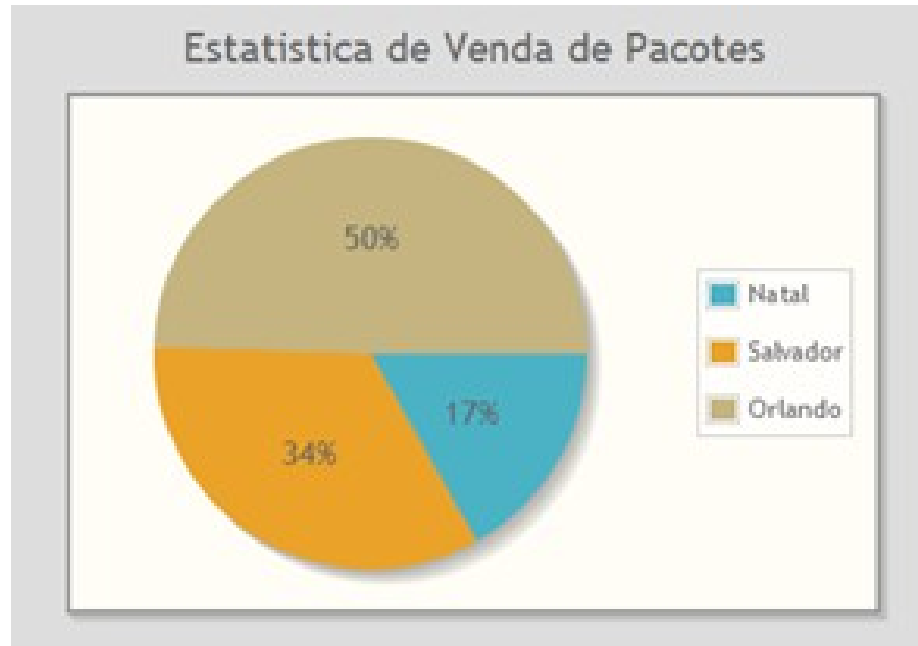
    <p:graphicImage value="#{clienteBean.foto}" />

</p:dialog>
```


PRIMEFACES TEMAS

A apresentação de dados consolidados em gráficos é um elemento importante no mundo corporativo;

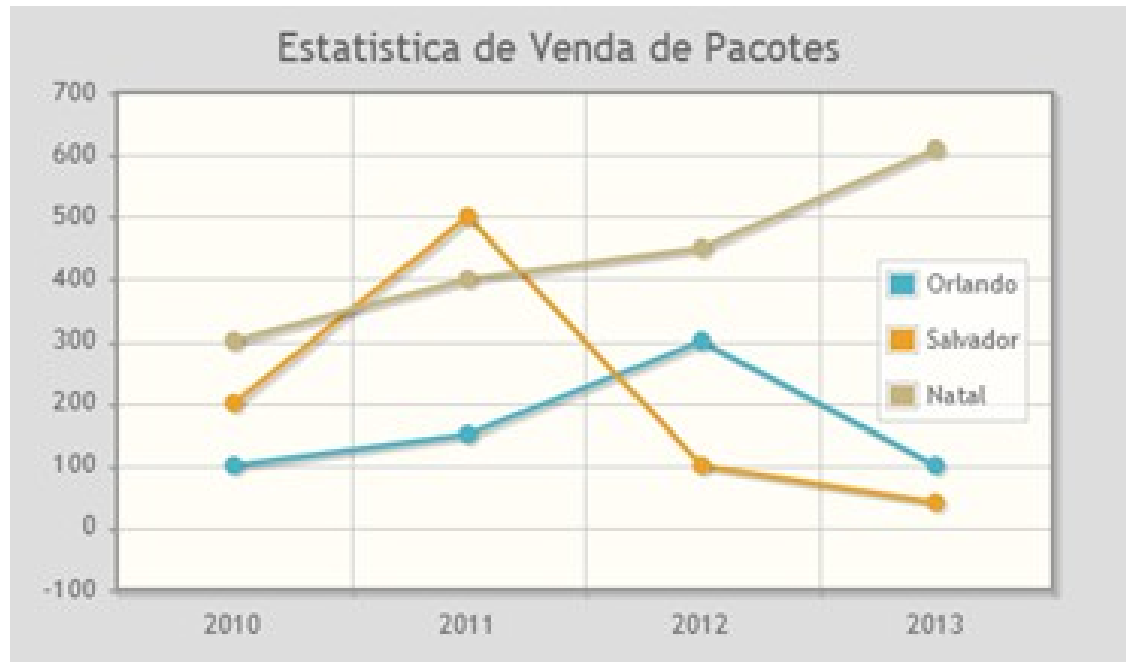
A biblioteca Primefaces propicia um conjunto muito rico de componentes focados na construção de gráficos.



```
<ui:define name="content">  
  <p:chart type="pie"  
    model="#{pacoteRelatorioBean.pacoteEstatistica}"  
    style="width:400px;height:300px;margin:auto"/>  
</ui:define>
```

```
@ManagedBean
@ViewScoped
public class PacoteRelatorioBean implements Serializable{
    private PieChartModel pacoteEstatistica;
    @PostConstruct
    public void init(){
        pacoteEstatistica = new PieChartModel();
        pacoteEstatistica.set("Natal", 150);
        pacoteEstatistica.set("Salvador", 300);
        pacoteEstatistica.set("Orlando", 442);
        pacoteEstatistica.setLegendPosition("e");
        pacoteEstatistica.setShowDataLabels(true);
    }

    public PieChartModel getPacoteEstatistica() {
        return pacoteEstatistica;
    }
}
```



```
<ui:define name="content">  
  <p:chart type="line"  
    model="#{relatorioLinhaBean.grafico}"  
    style="width:500px;height:300px;margin:auto"/>  
</ui:define>
```

```
@ManagedBean
public class RelatorioLinhaBean implements Serializable {

    private LineChartModel grafico;

    @PostConstruct
    private void init(){
        grafico = new LineChartModel();
        grafico.setLegendPosition("w");
        grafico.setTitle("Vendas");
        grafico.getAxes().put(AxisType.X, new CategoryAxis("Anos"));

        LineChartSeries serie1 = new LineChartSeries();
        serie1.setLabel("Salvador");
        serie1.set("2010", 200);
        serie1.set("2011", 500);
        serie1.set("2012", 100);
        serie1.set("2013", 40);

        LineChartSeries serie2 = new LineChartSeries();
```

```
serie2.setLabel("Natal");
serie2.set("2010", 300);
serie2.set("2012", 450);
serie2.set("2013", 609);

LineChartSeries serie3 = new LineChartSeries();
serie3.setLabel("Orlando");
serie3.set("2010", 100);
serie3.set("2011", 150);
serie3.set("2012", 300);
serie3.set("2013", 100);

grafico.addSeries(serie1);
grafico.addSeries(serie2);
grafico.addSeries(serie3);

}

public LineChartModel getGrafico() {
    return grafico;
}

}
```

PRIMEFACES UPLOAD DE ARQUIVOS

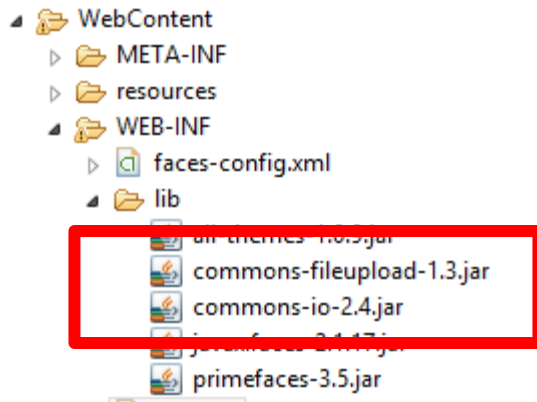
UPLOAD - CONFIGURAÇÃO

É necessário configurar o projeto web:

- Adicionar a configuração no arquivo web.xml

```
<filter>
  <filter-name>PrimeFaces FileUpload Filter</filter-name>
  <filter-class>org.primefaces.webapp.filter.FileUploadFilter</filter-class>
</filter>
<filter-mapping>
  <filter-name>PrimeFaces FileUpload Filter</filter-name>
  <servlet-name>Faces Servlet</servlet-name>
</filter-mapping>
```

Web.xml



As libs podem ser encontradas em:

<http://commons.apache.org/proper/commons-fileupload/>
<http://commons.apache.org/proper/commons-io/>

UPLOAD DE ARQUIVO

```
<p:form enctype="multipart/form-data">
```

```
<p:fileUpload
```

```
    fileUploadListener="#{clienteBean.uploadFile}"
```

```
    sizeLimit="100000"
```

```
    allowTypes="/(\\.|\\V)(gif|jpe?g|png)$/"
```

```
    auto="false"
```

```
    dragDropSupport="false"
```

```
    id="foto"/>
```

XHTML

The screenshot shows a web form titled 'Cliente' with the following fields: 'Código' (text input with '0'), 'CPF *' (text input), 'Nome *' (text input), 'Data de Nascimento *' (text input with '05/05/2013'), and 'Sexo' (radio buttons for 'Masculino' and 'Feminino'). Below these fields is a placeholder for a profile picture. A red rectangle highlights the file upload controls, which include a '+ Choose' button, an 'Upload' button, and a 'Cancel' button. At the bottom of the form are 'Salvar' and 'Foto' buttons, and a 'Cadastrar Novo' link.

```
public void uploadFile(FileUploadEvent event){
```

```
String arquivo = event.getFile().getFileName();
```

```
try{
```

```
    File file = new File("C:\\temp\\",arquivo);
```

```
    FileOutputStream fos = new FileOutputStream(file);
```

```
    fos.write(event.getFile().getContents());
```

```
    fos.close();
```

```
    cliente.setFoto(arquivo);
```

```
}catch(Exception e){}
```

```
}
```

Pasta onde o arquivo será salvo no servidor.

É armazenado no banco o nome do arquivo.

Java

EXIBINDO O ARQUIVO

Customer


Code

CPF *

Name *

Birthdate *

Sexo ☒ Masculino ☐ Feminino



+ Choose ↗ Upload ⌕ Cancel

Save

Picture × Remove [Register new](#)

EXIBINDO A IMAGEM SALVA

```
<p:graphicImage id="fotoImage" value="#{clienteBean.foto}"/>
```

O componente **graphicImage** pode exibir tanto imagens dentro do contexto da aplicação quanto **StreamedContent**, que pode ser preenchido com uma imagem proveniente de qualquer lugar.

XHTML

```
private StreamedContent foto;

public StreamedContent getFoto() {
    FacesContext context = FacesContext.getCurrentInstance();
    DefaultStreamedContent content = new DefaultStreamedContent();
    content.setContentType("image/jpg");
    try {
        if (context.getRenderResponse() || cliente.getFoto() == null) {
            content.setStream(new FileInputStream("C:\\temp\\semFoto.jpg"));
        } else {
            content.setStream(new FileInputStream("C:\\temp\\" + cliente.getFoto()));
        }
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }

    return content;
}
```

Nome da foto armazenada no banco.

O managedBean deve ter scope de sessão (sessionScoped)

Java

```
//Grava a imagem
public void uploadFile(FileUploadEvent event){
    try {
        cliente.setFoto(IOUtils.toByteArray(event.getFile().getInputStream()));
    } catch (IOException e) {
        e.printStackTrace();
    }
}

//Exibe a imagem
public StreamedContent getFoto(){
    FacesContext context = FacesContext.getCurrentInstance();
    DefaultStreamedContent content = new DefaultStreamedContent();
    content.setContentType("image/jpg");
    try{
        if (context.getRenderResponse() || cliente.getFoto() == null){
            content.setStream(new FileInputStream("C:\\temp\\semFoto.jpg"));
        }else{
            content.setStream(new ByteArrayInputStream(cliente.getFoto()));
        }
    }catch(Exception e){
        e.printStackTrace();
    }
    return content;
}
```

Java

A entidade cliente possui um atributo mapeado como @Lob, e no banco de dados a coluna relacionada com o atributo é do tipo BLOB.

- Não é necessário alterar o xhtml, somente o managed bean. As alterações estão marcadas em vermelho:

Copyright © 2013 - 2017 Prof. Me. Thiago T. I. Yamamoto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

"Somos nós que forjamos as correntes que usamos em nossas vidas"
Charles Dickens