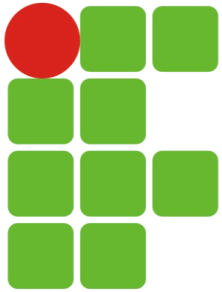


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE
Campus Natal - Central

PrimeFaces

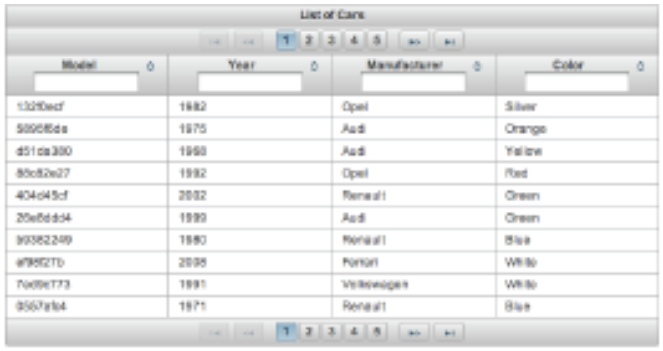
Prof. Fellipe Aleixo (*fellipe.aleixo@ifrn.edu.br*)

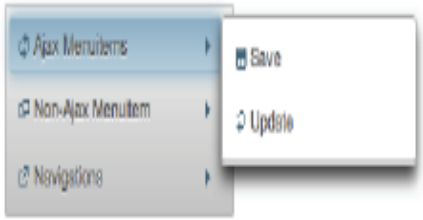
PrimeFaces

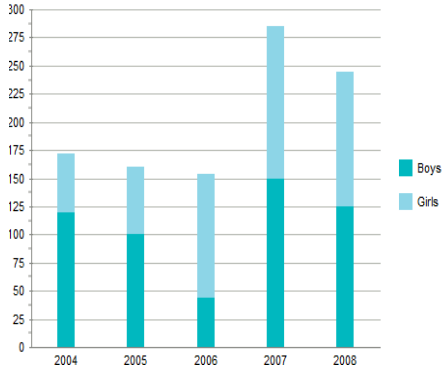
- Biblioteca *open source* de componentes JSF
- Encapsulamento da tecnologia AJAX
- Conjunto de componentes voltados para aplicações Web para dispositivos móveis

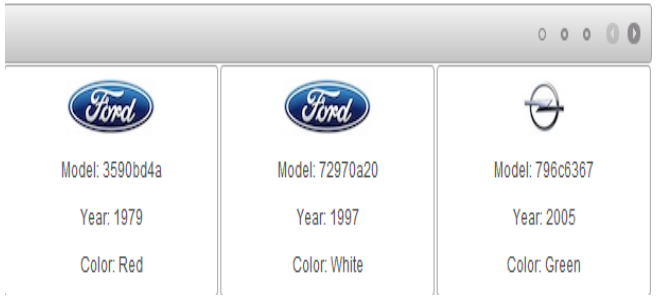
PrimeFaces

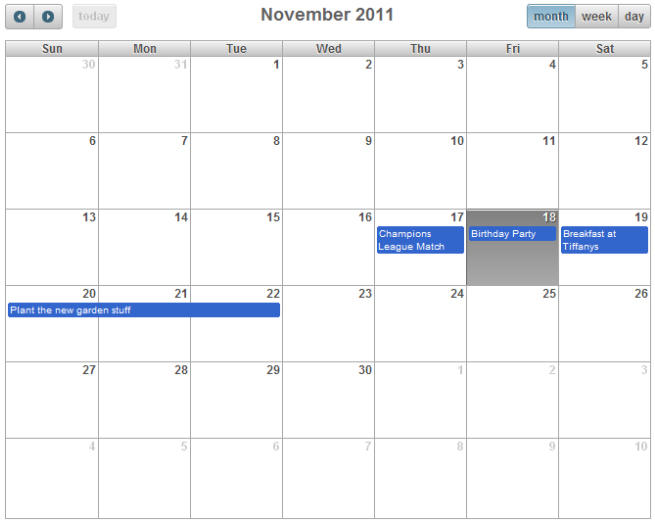
(+100 componentes)

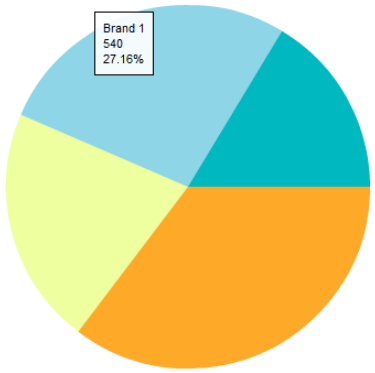












PrimeFaces

(+30 temas predefinidos)

Utilizando a PrimeFaces

- Definida em um único JAR
  - primefaces-{version}.jar
  - http://www.primefaces.org/downloads
- PrimeFaces não obriga nenhuma configuração
  - Opções de configuração são definidas como parâmetros de contexto

```
<context-param>
        <param-name>primefaces.THEME</param-name>
        <param-value>bootstrap</param-value>
</context-param>
```

Olá Mundo!

- Página simples, chamada **test.xhtml**:

```
<!DOCTYPE html>
<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">

 <h:head></h:head>

 <h:body>
      <p:editor />
 </h:body>

</html>
```

Componente: **AccordionPanel**

- Um “contêiner” de informações, que exibe as mesmas de forma empilhada

Godfather Part I

The story begins as Don Vito Corleone, the head of a New York Mafia family, oversees his daughter's wedding. His beloved son Michael has just come home from the war, but does not intend to become part of his father's business. T hrough Michael's life the nature of the family business becomes clear. The business of the family is just like the head of the family, kind and benevolent to those who give respect, but given to ruthless violence whenever anything stands against the good of the family.

Godfather Part II

Godfather Part III

Componente: **AccordionPanel**

- Alguns atributos:
  - rendered [boolean - true] – define se o componente será ou não renderizado
  - value [List – null] – recebe a coleção de elementos que serão exibidos nas “abas”
  - var [String – null] – variável que referencia o elemento corrente sendo exibido

Componente: **AccordionPanel**

- Exemplo:

```
<p:accordionPanel>
        <p:tab title="First Tab Title">
                <h:outputText value= "Lorem"/>
                ...More content for first tab
        </p:tab>
        <p:tab title="Second Tab Title">
                <h:outputText value="Ipsum" />
        </p:tab>
        //any number of tabs
</p:accordionPanel>
```

Componente: **AccordionPanel**

- Habilitando eventos AJAX:

```
<p:accordionPanel>
        <p:ajax event="tabChange" listener="#{bean.onChange}" />
</p:accordionPanel>
```

```
public void onChange(TabChangeEvent event) {
        //Tab activeTab = event.getTab();
        //...
 }
```

Componente: **AutoComplete**

- Oferece sugestões enquanto uma entrada está sendo realizada

Componente: **AutoComplete**

- Alguns atributos:
  - value [Object – null] – valor associado (EL/literal)
  - completeMethod [MethodExpr – null] – método que provê um conjunto de sugestões
  - var [String – null] – nome da sugestão corrente
  - itemLabel [String – null] – rótulo do item
  - itemValue [String - null] – valor do item
  - maxResults [Integer - unlimited] – número máximo de sugestões

Componente: **AutoComplete**

- Exemplo:

```
<p:autoComplete value="#{bean.text}" completeMethod="#{bean.complete}" />
```

```
public class Bean {
	private String text;
	public List<String> complete(String query) {
		List<String> results = new ArrayList<String>();
		for (int i = 0; i < 10; i++)
			results.add(query + i);

		return results;
	}

	//getter setter
}
```

Componente: **AutoComplete**

- Suporte a POJOs:

```
<p:autoComplete value="#{playerBean.selectedPlayer}"
        completeMethod="#{playerBean.completePlayer}"
        var="player"
        itemLabel="#{player.name}"
        itemValue="#{player}"
        converter="playerConverter"/>
```

```
public class Player {
        private String name;
        //getter setter
}
```

Componente: **AutoComplete**

```
public class PlayerBean {

	private Player selectedPlayer;

	public Player getSelectedPlayer() {
		return selectedPlayer;
	}
	public void setSelectedPlayer(Player selectedPlayer) {
		this.selectedPlayer = selectedPlayer;
	}

	public List<Player> complete(String query) {
		List<Player> players = readPlayersFromDatasource(query);

		return players;
	}
}
```

Componente: **AutoComplete**

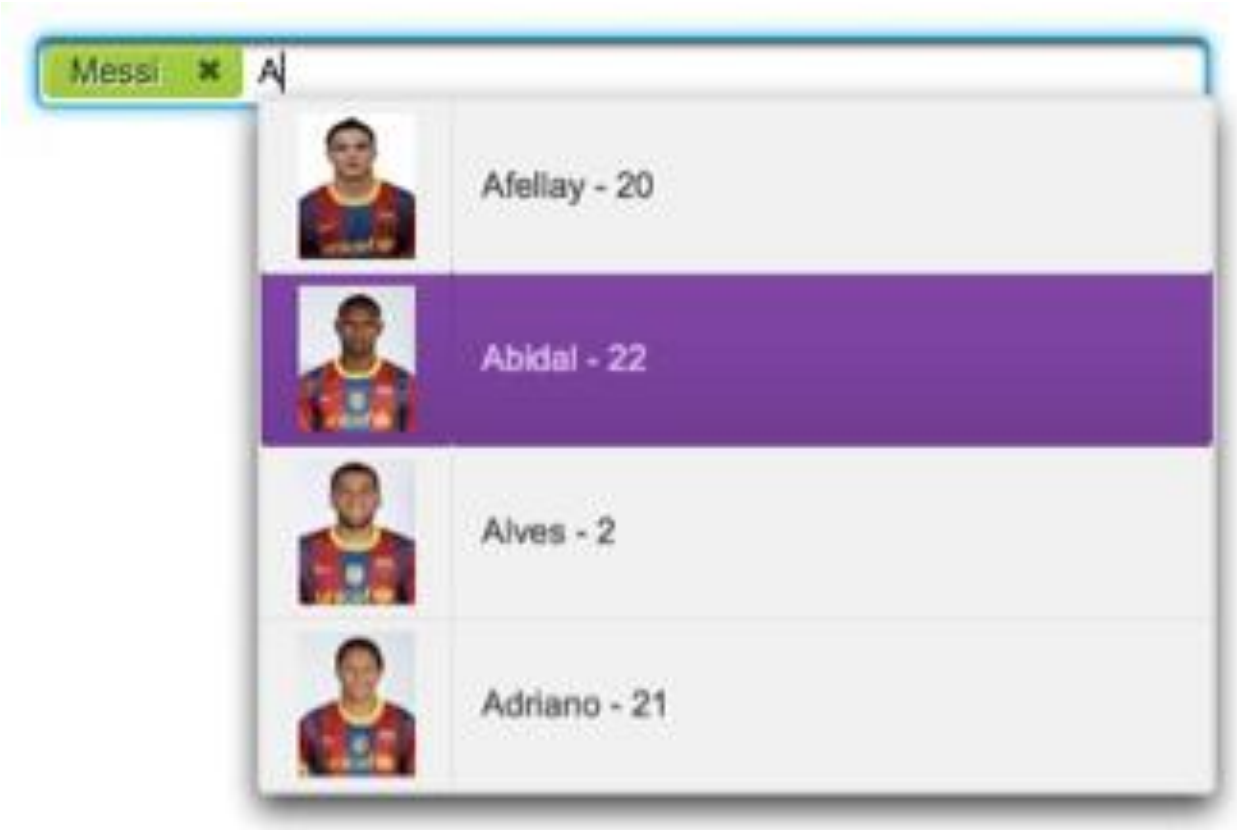
- Seleção múltipla:

```
<p:autoComplete id="advanced" value="#{autoCompleteBean.selectedPlayers}"
      completeMethod="#{autoCompleteBean.completePlayer}"
                  var="p" itemLabel="#{p.name}" itemValue="#{p}" converter="player"
                  multiple="true">

      <p:column style="width:20%;text-align:center">
            <p:graphicImage value="/images/barca/#{p.photo}"/>
      </p:column>

      <p:column style="width:80%">
            #{p.name} - #{p.number}
      </p:column>
</p:autoComplete>
```

Componente: **AutoComplete**



Componente: **Calendar**

- Componente de entrada de datas:

Componente: **Calendar**

- Alguns atributos:
  - value [Date – null] – valor associado
  - mode [String – popup] – como será exibido
  - showOn [String – both] – evento que causa a exibição do calendário
  - pages [Integer – 1] – número de páginas exibidas (Ex.: quantos meses irão aparecer de uma vez)

Componente: **Calendar**

- Exemplo básico:

```
<p:calendar value="#{dateBean.date}"/>
```

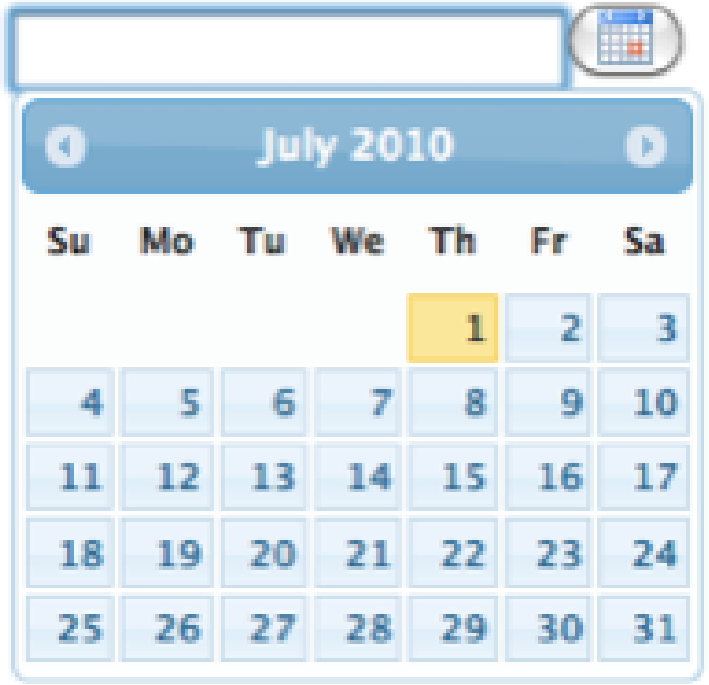
- Exemplo – modo “inline”:

```
<p:calendar value="#{dateBean.date}" mode="inline" />
```

Componente: **Calendar**

- Exemplo – botão + popup:

```
<p:calendar value="#{dateBean.date}" mode="popup" showOn="button" />
```



Componente: **Calendar**

- Exemplo – várias páginas:

```
<p:calendar value="#{dateController.date}" pages="3"/>
```

Componente: **Captcha**

- Componente de validação formulário baseado na API **Recaptcha**

Componente: **Captcha**

- Alguns atributos:
  - theme [String – red] – define o tema de exibição do formulário
  - languages [String – en] – linguagens suportadas

Componente: **Captcha**

- Passo anterior
  - Recuperar chaves públicas e privadas a partir do serviço **reCaptcha** – referenciar as mesmas no web.xml

```
<context-param>
    <param-name>primefaces.PRIVATE_CAPTCHA_KEY</param-name>
    <param-value>YOUR_PRIVATE_KEY</param-value>
 </context-param>

<context-param>
    <param-name>primefaces.PUBLIC_CAPTCHA_KEY</param-name>
    <param-value>YOUR_PUBLIC_KEY</param-value>
 </context-param>
```

Componente: **Captcha**

- Exemplo básico:

```
<p:captcha />
```

- Exemplo – alterando o tema padrão:

```
<p:captcha theme="white"/>
```

- Exemplo – customizando a linguagem:

```
<p:captcha language="tr"/>
```

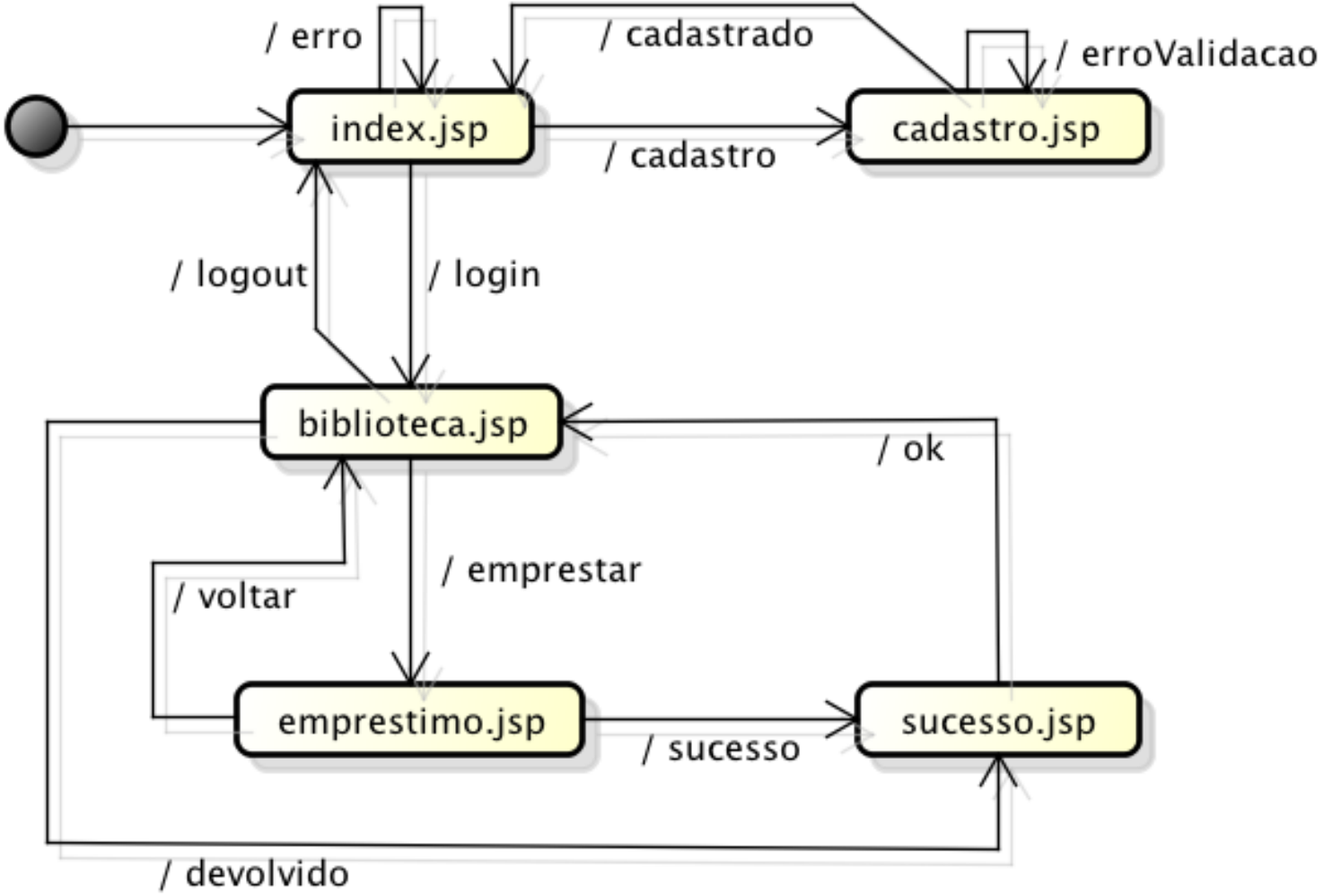
Utilize pelo menos QUATRO componentes PrimeFaces no desenvolvimento do seguinte

EXERCÍCIO

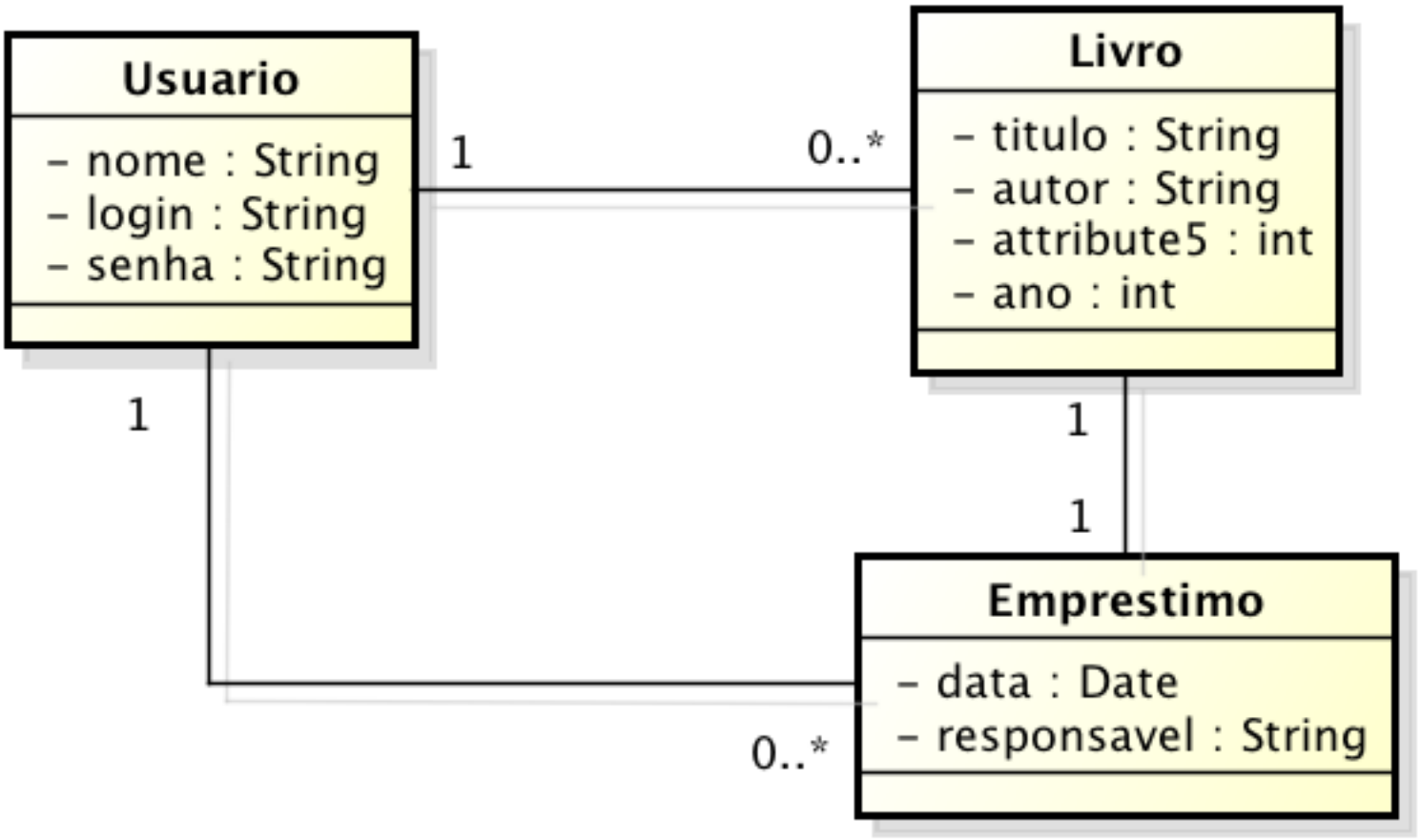
Exercício: Desenvolvimento de uma Aplicação Utilizando o PrimeFaces

- Sistema proposto: “Biblioteca Pessoal”
  - Sistema disponível da Web
  - Usuários podem realizar o seu cadastro
  - Usuários cadastrados podem “logar” no sistema
    - Manter a sua biblioteca pessoal (livros) – CRUD
    - Registrar o empréstimo de um de seus livros
    - Identificar os livros emprestados – informando o tempo em que estão emprestados
    - Registrar a devolução de um dos livros emprestados

Biblioteca Pessoal: Fluxo de Navegação



Biblioteca Pessoal: Classes de Domínio



Biblioteca Pessoal

- Vamos implementar todas as JSF e *managed beans* necessários para a implementação desse sistema
  - Podemos utilizar algum Servlet ou JSP em casos específicos – p.ex.: controle de autenticação
- Os dados serão armazenados em memória (versão 0.1)
- Mãos a massa!