



**Instituto Federal de Educação, Ciência e Tecnologia**

**Triângulo Mineiro – Campus Uberlândia Centro**

**Tecnologia em Sistemas para Internet**

---

# **Projeto e Desenvolvimento de Software II**

**Prof. Carlos Eduardo de Carvalho Dantas**

[carloseduardodantas@iftm.edu.br](mailto:carloseduardodantas@iftm.edu.br)

---

# **Parte I – Técnicas e Ferramentas para Desenvolvimento de Sistemas**

# AGENDA

---

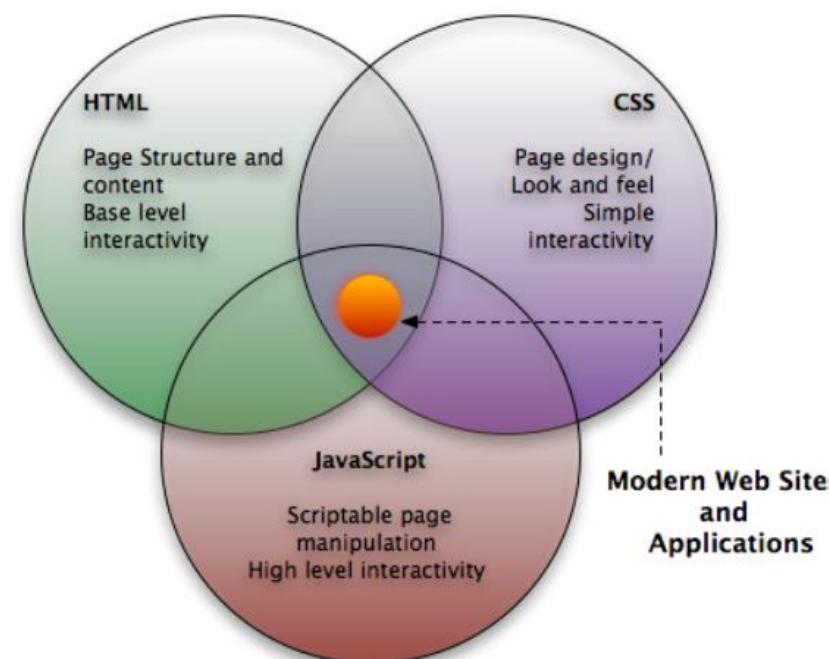
## 1. Desenvolvimento Front-End

- Histórico
- Aplicações RIA
- Frameworks front-end
- Modelo MVW
- Web Responsiva
- Escalabilidade e websockets

# A WEB

## - Definição

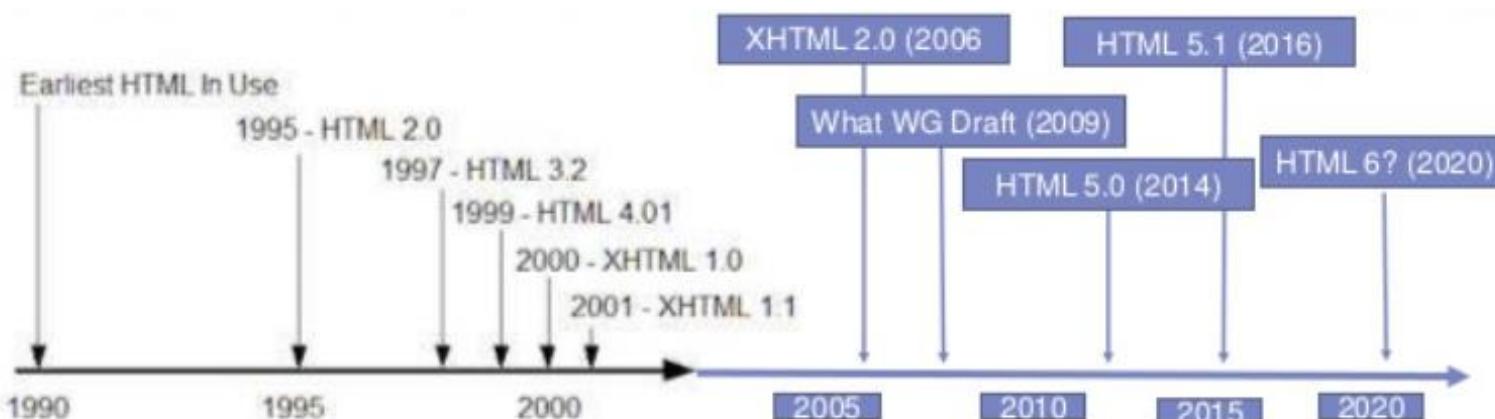
- Sistemas de documentos *Hipermídia* que são interligados e executados na *internet*.
- Documentos *Web* são uma combinação de *HTML*, *JavaScript* e *CSS*.



# A WEB

## - HTML (*HyperText Markup Language*)

- É controlado desde 1995 pelo *W3C*, e desde 2009 trabalha em conjunto com *WHATWG* na confecção do *HTML 5*. Desde a primeira versão, cada atualização adicionou diversas tags, desde a adição de *hyperlinks*, formulários, dentre outros.
- Com o *HTML 5*, ferramentas como *Silverlight* e *Flash* perderam parte de sua utilidade, pois esta versão suporta a maior parte dos recursos que antes só eram possíveis com *plug-ins*. Exemplos: exibição de vídeos, gráficos 2d ou 3d, etc..



## - CSS(Cascading Style Sheets)

- Foi criado para controlar a aparência dos documentos HTML, formatando a informação.
- O CSS 3 **multiplicou as possibilidades** de se utilizar estilos na criação de páginas. Exemplos: propriedades como *box-shadow* e funções como *linear-gradient* permitiram a criação de formas, cores e efeitos das páginas diretamente no código, sem precisar de criar imagens para tal.

```
<html>
  <head>
    <style>
      .tnt {
        border-radius: 50%;
        border: 5px solid #000;
        height: 50px;
        line-height: 50px;
        text-align: center;
        width: 50px;
      }
    </style>
  </head>
  <body>
    <h3 class='tnt'>TNT</h3>
  </body>
</html>
```



## - JavaScript

- Começou como uma “*simples*” linguagem *client-side* para operações simples como validação de formulários.
- O uso primário de *JavaScript* é escrever funções que são embarcadas ou incluídas em páginas *HTML* e que interagem com o *Modelo de objeto nos documentos* (DOM) da página.
- Recursos como *Ajax*, tipagem dinâmica, avaliação em tempo de execução, protótipos, objetos *Json*, dentre outros transformaram *JavaScript* na linguagem mais popular da *web*.

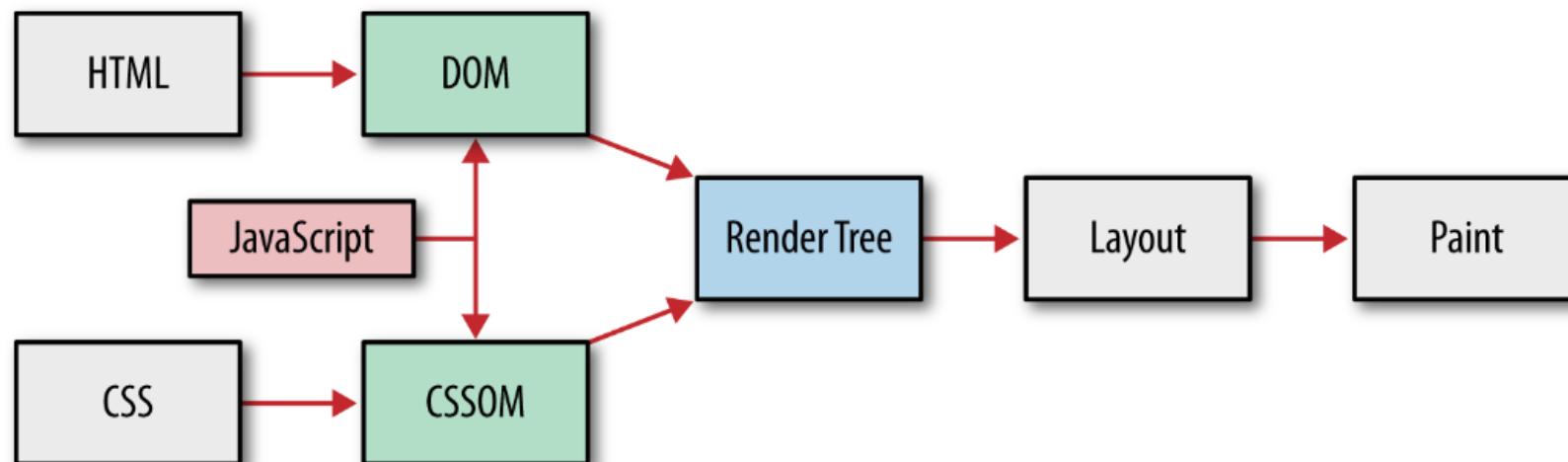
## - DOM (Domain Object Model)

- É uma árvore de elementos da página *HTML* criada, ou um espelho em memória de toda a página criada no navegador quando esta é carregada. Possui funções e propriedades que, ao serem modificadas, disparam alterações instantâneas no que é exibido ao usuário.
- Sendo assim, é possível manipular o *DOM* utilizando código *JavaScript*, ou seja, estes scripts possuem capacidade de modificar dinamicamente o *DOM*.

```
<!-- página html -->
<button class="botao-grava">Novo</button>
<p class="contatos">Contatos cadastrados: 0 </p>
var contatos = document.querySelector('.contatos');
var total = 0;
var botao = document.querySelector('.botao-grava');
botao.addEventListener('click', function(event) {
    total++;
    contatos.textContent = 'Contatos cadastrados: ' + total;
});
```

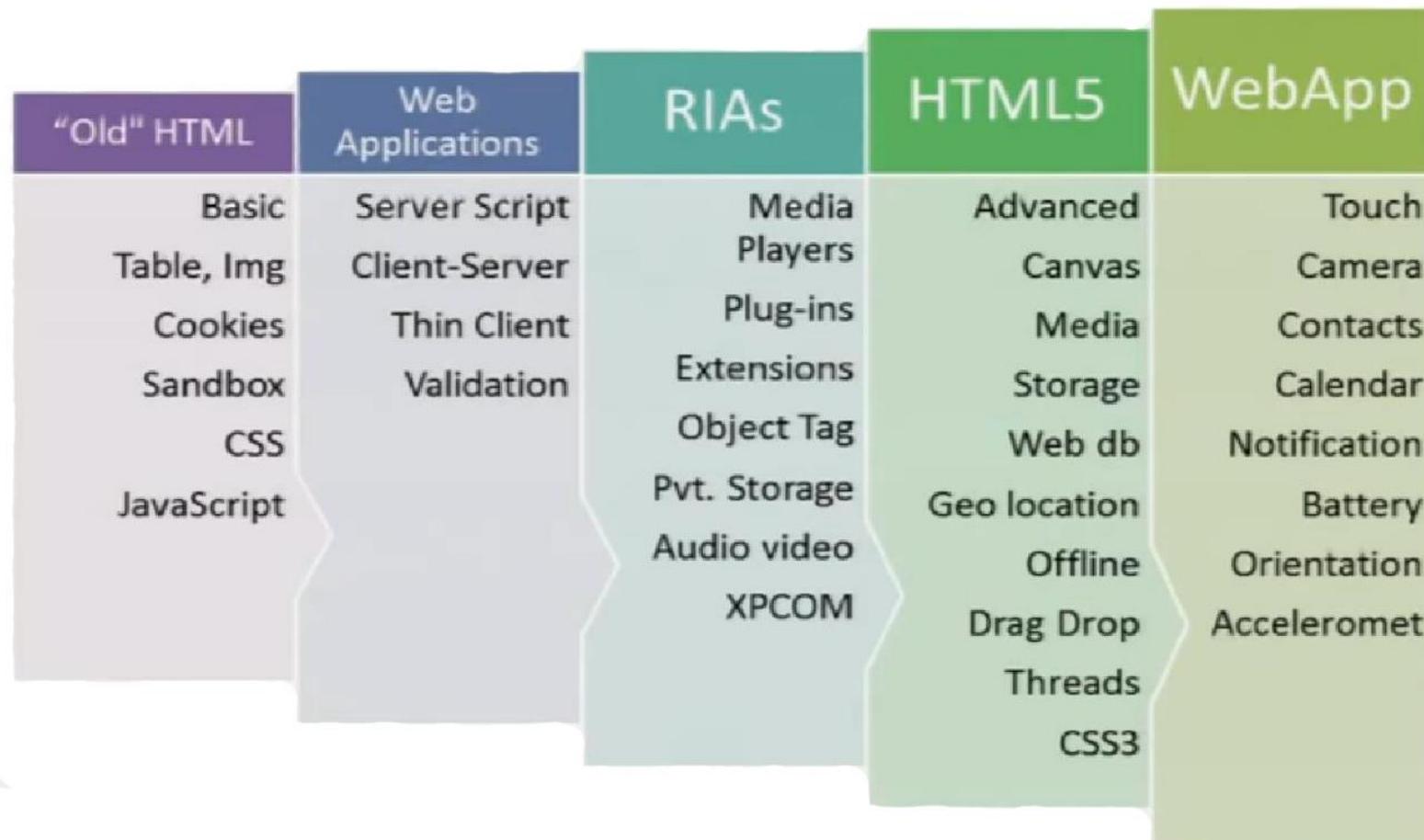
## - Pipeline de Processamento do Browser

- O *parsing* do documento *HTML* constrói o *DOM*. As regras e recursos das folhas de estilos constroem o *CSSOM*. A combinação de ambos formam a árvore de renderização, onde o navegador possui informação suficiente para construir o layout e mostrar algo na tela.
- A execução de *Scripts* podem editar alguma informação do *DOM*. *Scripts* também podem perguntar por estilos de quaisquer objetos, acessando a *CSSOM*.



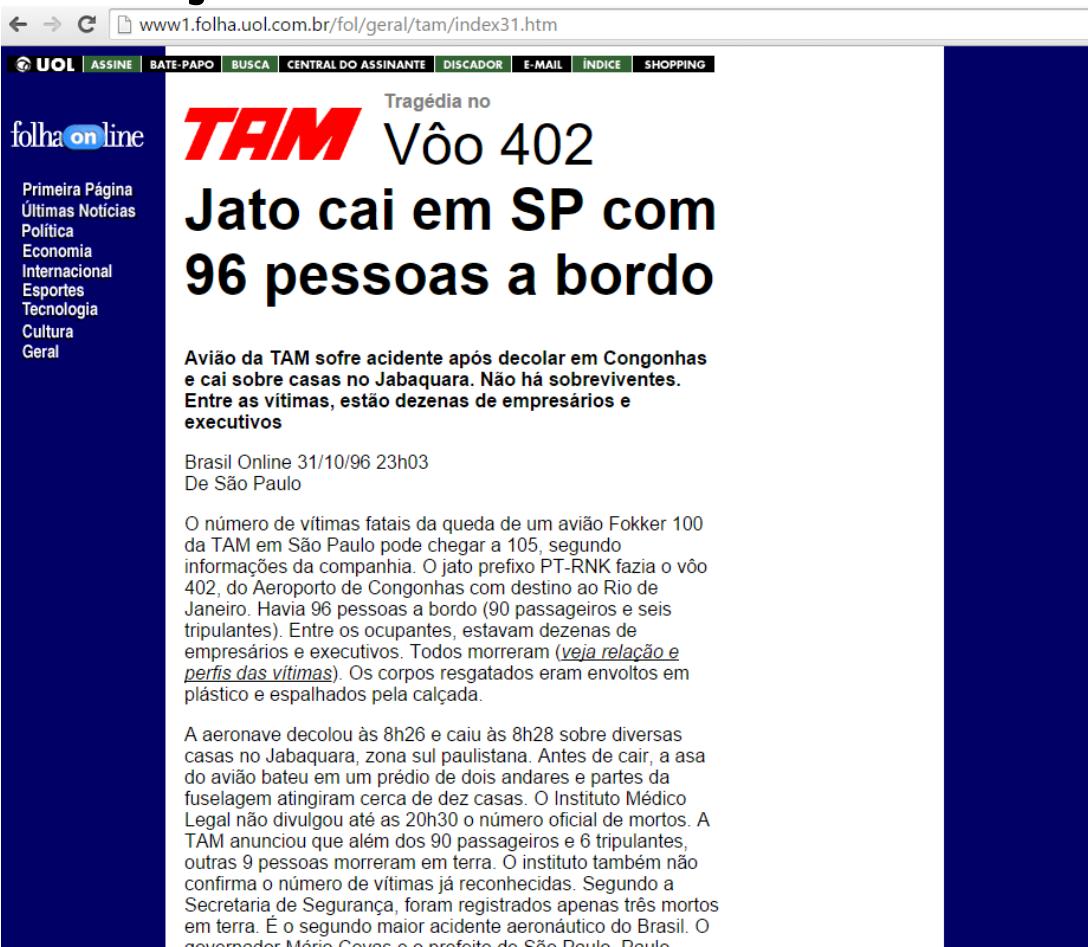
# Evolução de tecnologias Web

## - Evolução



# Evolução de tecnologias Web

## - Evolução – Old HTML



**TAM Vôo 402**  
**Jato cai em SP com 96 pessoas a bordo**

Avião da TAM sofre acidente após decolar em Congonhas e cai sobre casas no Jabaquara. Não há sobreviventes. Entre as vítimas, estão dezenas de empresários e executivos

Brasil Online 31/10/96 23h03  
De São Paulo

O número de vítimas fatais da queda de um avião Fokker 100 da TAM em São Paulo pode chegar a 105, segundo informações da companhia. O jato prefixo PT-RNK fazia o voo 402, do Aeroporto de Congonhas com destino ao Rio de Janeiro. Havia 96 pessoas a bordo (90 passageiros e seis tripulantes). Entre os ocupantes, estavam dezenas de empresários e executivos. Todos morreram (*veja relação e perfis das vítimas*). Os corpos resgatados eram envoltos em plástico e espalhados pela calçada.

A aeronave decolou às 8h26 e caiu às 8h28 sobre diversas casas no Jabaquara, zona sul paulistana. Antes de cair, a asa do avião bateu em um prédio de dois andares e partes da fuselagem atingiram cerca de dez casas. O Instituto Médico Legal não divulgou até as 20h30 o número oficial de mortos. A TAM anunciou que além dos 90 passageiros e 6 tripulantes, outras 9 pessoas morreram em terra. O instituto também não confirma o número de vítimas já reconhecidas. Segundo a Secretaria de Segurança, foram registrados apenas três mortos em terra. É o segundo maior acidente aeronáutico do Brasil. O governador Mário Covas e o prefeito de São Paulo, Paulo



**novos inclusões** **café?** **eventos** **veja** ?

Consulta  Busca [Informações](#)

**Feliz Natal!**

**Ciência e Tecnologia**  
[Institutos, Centros de Pesquisa](#)

**Cultura**  
[Museus, Música, Personalidades](#)

**Esportes**  
[Automobilismo, Futebol](#)

**Governo**  
[Estados, Prefeituras](#)

**Compras**  
[CDs, Flores, Livros](#)

**Educação**  
[Cursos, Escolas, Universidades](#)

**Finanças**  
[Bancos, Bolsas, Seguros](#)

**Indústria e Comércio**  
[Automóveis, Telecomunicações](#)

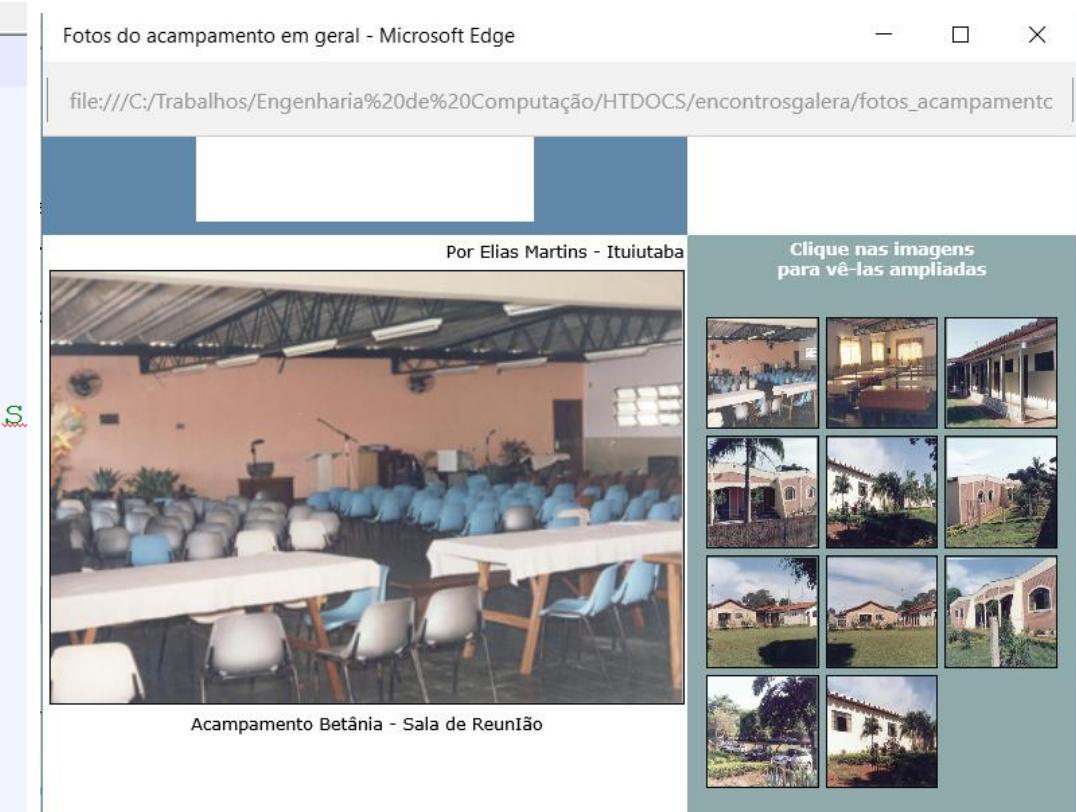
# Evolução de tecnologias Web

## - Evolução – Old HTML

```
fotos_acampamento.HTM
<SCRIPT>
  id_album = "fotos_acampamento";
  thumbs = 11; //numero de fotos no total
  fotoabre =6; // foto padrao que abre
  imgs_roothpath = "fotospeq"
  imgs_rootpath2 = "fotos_acampamento"
  reverso = 1;
  dropdown_todosalbuns = "selecao.htm"; //caixa de seleção
  banner = "logomarca";
  abreurl = 0;
  naodaravolta = 0; // coloca Anterior e Próximo

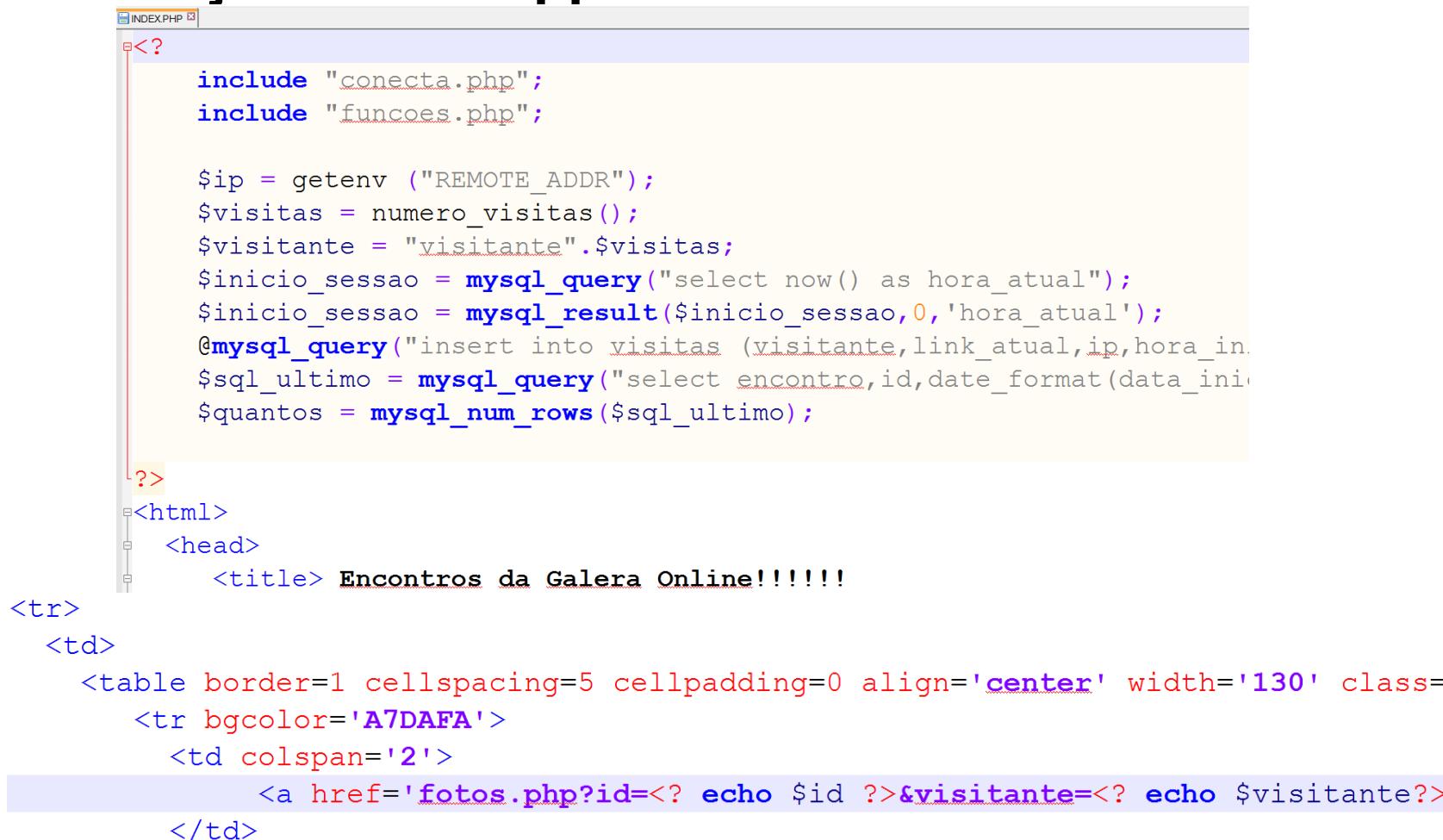
  y = x = 1 ;
  legenda=new Array();
  credito=new Array();

  credito[x++] = "Por Elias Martins - Ituiutaba";
```



# Evolução de tecnologias Web

## - Evolução – Web Application PHP



```
INDEXPHP
<?
include "conecta.php";
include "funcoes.php";

$ip = getenv ("REMOTE_ADDR");
$visitas = numero_visitas();
$visitante = "visitante".$visitas;
$inicio_sessao = mysql_query("select now() as hora_atual");
$inicio_sessao = mysql_result($inicio_sessao,0,'hora_atual');
@mysql_query("insert into visitas (visitante,link_atual,ip,hora_in.
$sql_ultimo = mysql_query("select encontro,id,date_format(data_ini.
$quantos = mysql_num_rows($sql_ultimo);

?>
<html>
  <head>
    <title> Encontros da Galera Online!!!!!
<tr>
  <td>
    <table border=1 cellspacing=5 cellpadding=0 align='center' width='130' class=
      <tr bgcolor='A7DAFA'>
        <td colspan='2'>
          <a href='fotos.php?id=<? echo $id ?>&visitante=<? echo $visitante?>
    </td>
```

# Evolução de tecnologias Web

## - Evolução – Web Application Java SERVLET + JSP

```
<!-- Código adicionado para solucionar controle de grupos -->
<%@page import="java.security.*" %>
<%@page import="javax.security.auth.*" %>
<%
    if (!request.isUserInRole("SGP_Administradores") && !"carlosec".equals(re
        <logic:redirect forward="error403"/> <%
    }
%>

<!-- Código adicionado para solucionar controle de grupos -->

<html><!-- InstanceBegin template="/Templates/pgTemplateForm.dwt.jsp" codeOut
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<c:if test="false">
    <link href="/css/estilos.css" rel="stylesheet" type="text/css">
</c:if>
<link href="<% request.getContextPath() %>/css/estilos.css" rel="stylesheet"
<style type="text/css">
```

# Evolução de tecnologias Web

## - Evolução – Web Application ASP

```
<td width="70%">
    <input type="text" class='text' name="strPesquisa" value="<%="Request("strPesquisa")%>" :>
</td> </table>

<BR><BR>

<%
strPesquisa = Request("strPesquisa")

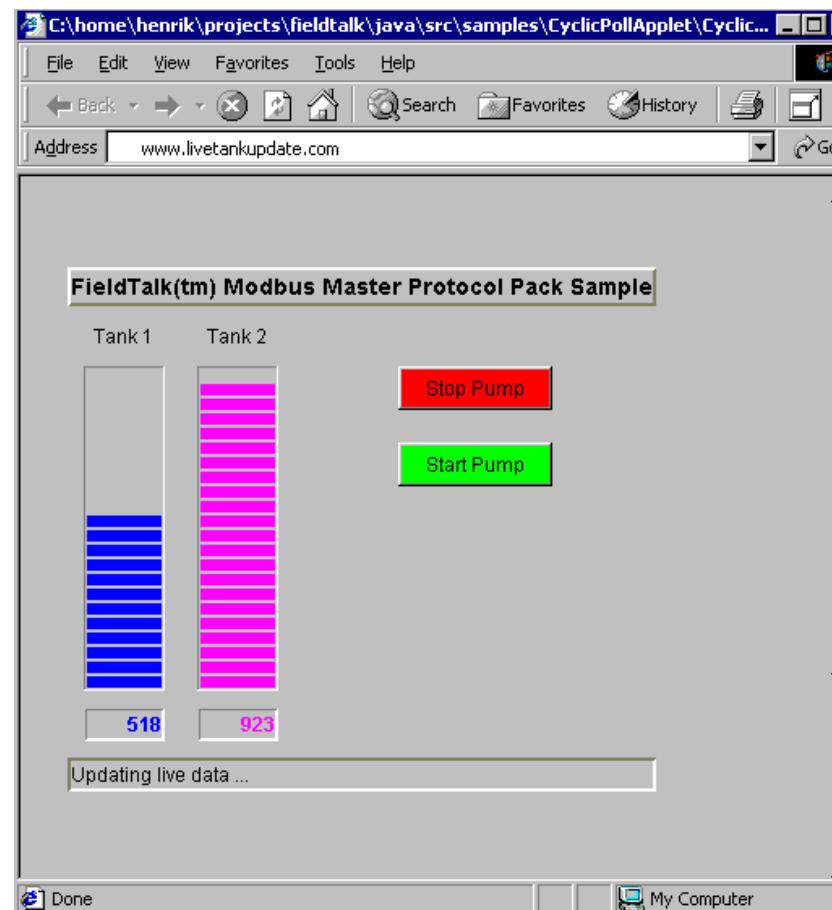
set objConn = Server.CreateObject("ADODB.Connection")
set objRecset = Server.CreateObject("ADODB.Recordset")

objConn.Open strConn

strQuery = "S_sp_Busca Grupo Descricao '%'" & strPesquisa & "%'"
objRecset.Open strQuery, objConn
```

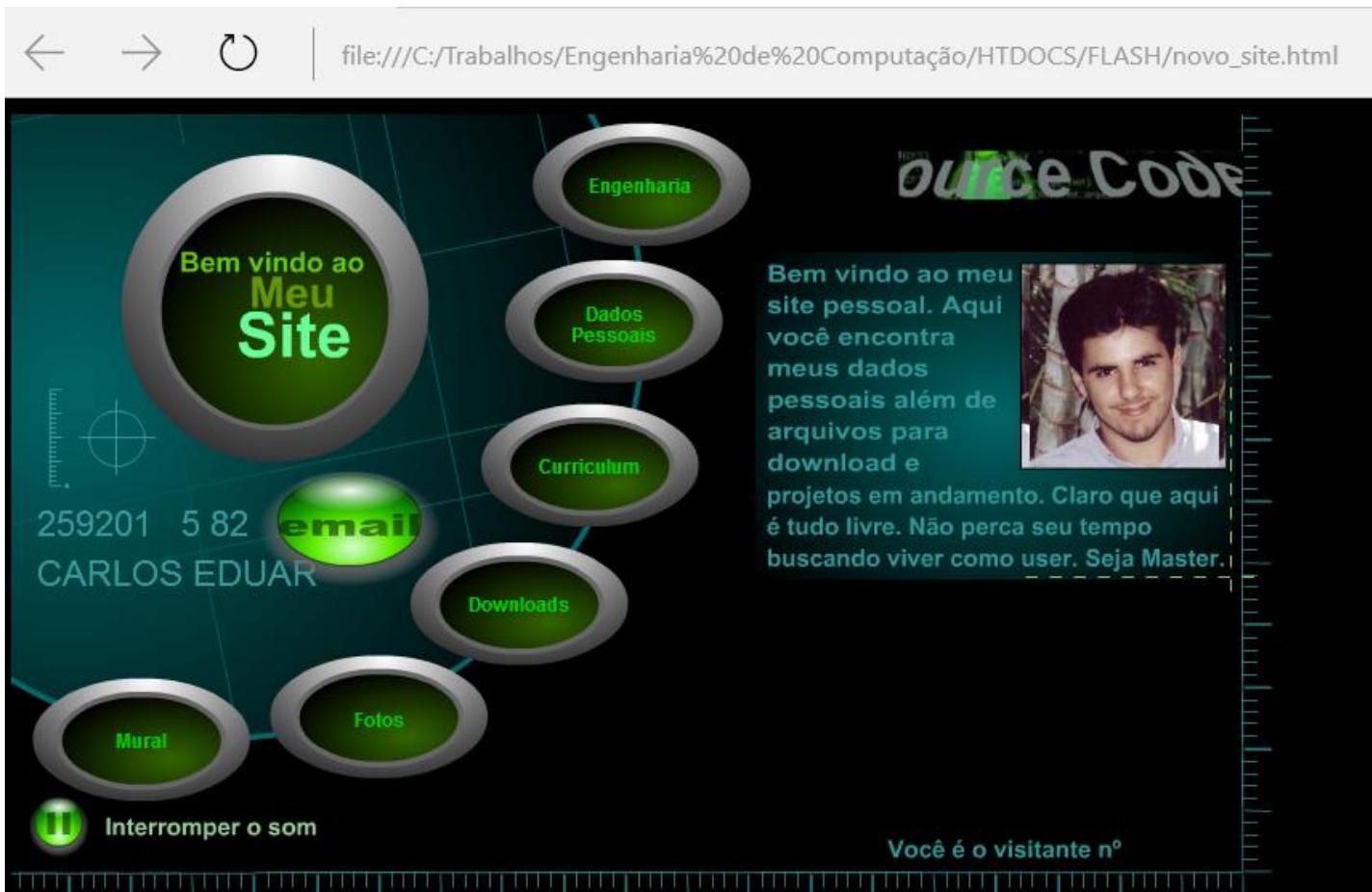
# Evolução de tecnologias Web

## - Evolução – RIA Applet Java



# Evolução de tecnologias Web

## - Evolução – RIA FLASH



# Evolução de tecnologias Web

## - Evolução – RIA JSF



```
</h:selectOneMenu>

<h:outputText value="Validade:" id="outval" styleClass="outputText" title=
<pxt:panelGrid columnsWidth="90px;20px;*" cellspacing="0"
    cellpadding="0" id="pn1val">
    <rich:calendar inputClass="calendarTam" id="calval"
        datePattern="dd/MM/yyyy"
        disabled="#{acaoComercialBean.disabledOnDefault}"
        enableManualInput="true" inputSize="8"
        value="#{acaoComercialBean.domain.validadeNaoNula.dataInicio}">
        <a4j:support event="onchanged" id="evtcalval" />
    </rich:calendar>
    <h:outputText value="à" styleClass="outputText" id="outdat"
        style="margin: 5px;" />
    <pxt:cell align="left">
```

# Evolução de tecnologias Web

## - Evolução – RIA ASP.NET

```
<TR>
    <TD style="WIDTH: 451px; HEIGHT: 38px"><asp:label id="Label1" ControlToValidate="txtMensagem" ErrorMessage="*" Value="Responda a pergunta acima."></asp:label>
    <TD style="WIDTH: 337px; HEIGHT: 38px"><asp:radiobuttonlist AutoPostBack="True">
        <asp:ListItem Value="1">Sim</asp:ListItem>
        <asp:ListItem Value="0">Não</asp:ListItem>
    </asp:radiobuttonlist></TD>
<TR>
    <TD style="WIDTH: 451px; HEIGHT: 14px"><asp:label id="Label2" ControlToValidate="txtMensagem" ErrorMessage="*" Value="Responda a pergunta acima."></asp:label>
    <TD style="WIDTH: 337px; HEIGHT: 14px"><asp:textbox id="txtMensagem" MaxLength="255" TextMode="MultiLine" Height="50px">
```

# Evolução de tecnologias Web

## - Evolução – HTML 5



# Evolução de tecnologias Web

## - Evolução – Web App



# Evolução de tecnologias Web

- **Old HTML** – páginas estáticas retornadas pelo servidor. Rápido, simples, porém sem conteúdo dinâmico;
- **Web Application** – servidor processa toda a informação, devolvendo o *HTML* pronto para o navegador. Uso de *JavaScript* para funções específicas, especialmente componentes mais complexos.
- **RIA** – abstrai a construção de componentes estilosos e complexos, gerando processamento mais intenso do lado cliente. Porém, geralmente depende do servidor para controlar o estado da interface gráfica.
- **HTML 5** – basicamente eliminou ferramentas *RIA* como *Flash*, *Silverlight*, *Java FX* e *Applets*, com um padrão aberto, e rodando nativamente no navegador.
- **Web App** – o conteúdo é escrito uma vez, sendo adaptável a qualquer dispositivo.

# Frameworks Web

## - Introdução

- *HTML 5, CSS 3 e JavaScript* possuem diversos recursos interessantes, permitindo criar sites e aplicações com visuais elegantes. Contudo, **criar** estilos, **manipular** o *DOM*, e se **preocupar** em ter compatibilidade com *browsers* mais antigos são tarefas relativamente demoradas para se fazer sozinho, sem o auxílio de ferramentas.
- Os frameworks surgem como ferramentas **para auxiliar** neste trabalho.
- Serão mostradas nas próximas seções frameworks para diversas finalidades distintas em sistemas Web

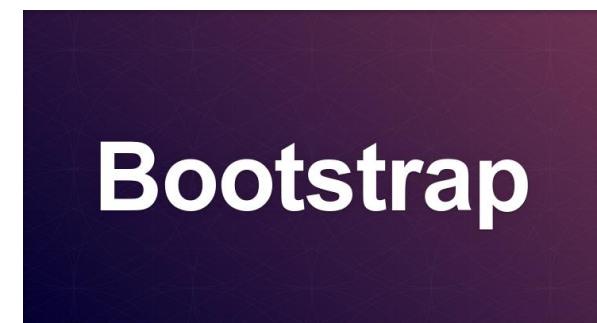
# Frameworks Web

## - Componentes Responsivos

- Frameworks que incluem kits com diversos componentes web prontos para desenvolver aplicações web/mobile responsivas. Está envolvido **unicamente com o desenho da tela HTML**, possuindo diversos *templates* prontos para serem utilizados.
- É possível até escrever aplicações completas **sem digitar nenhuma linha de CSS**. Por isso são tão aclamados em equipes que precisam de sistemas com telas não muito complexas, sem necessidade de tanto conhecimento em design e CSS.



Foundation  
Start here, build everywhere.



# Frameworks Web

## - JavaScript server-side

- Com a evolução da linguagem *JavaScript* e da sua *engine V8* (interpretador *JavaScript* criado pela *google*, que pode ser executado fora de um *browser*), servidores tradicionais como Apache e IIS ganharam novos concorrentes, que são servidores *JavaScript*.
- O *Node.js* é o servidor *JavaScript* mais conhecido. É extremamente escalável, *single threaded (non-blocking-thread)*, sem *deadlocks*, orientado a eventos, assíncrono



# Frameworks Web

## - Componentes JavaScript server-side

- Assim como o *Ruby* tem o *Rails*, o *Python* tem o *Django*, o *Groovy* tem o *Grails*, o *Node.js* tem o *Meteor* ou *Express*.
- Estes frameworks ajudam na organização de uma aplicação Web MVC do lado servidor, fornecendo templates para diversas funções, como disponibilizar REST endpoints, conectar no banco de dados, etc...

express



# Frameworks Web

## - Manipulação do DOM

- Uma das dificuldades de manipular o *DOM* diretamente com *JavaScript* se deve ao fato que cada navegador costuma implementar funções e propriedades de uma forma diferente, gerando incompatibilidade.
- Bibliotecas de manipulação do *DOM* foram criadas para blindar o programador das idiossincrasias do *DOM* entre os navegadores, além de adicionar recursos facilitadores. Uma das mais famosas é o *JQuery*.



# Frameworks Web

## - Manipulação do DOM

- Comparativo *JavaScript puro vs JQuery*.

### JavaScript puro

```
var contatos = document.querySelector('.contatos');
var total = 0;
var botao = document.querySelector('.botao-grava');
botao.addEventListener('click', function(event) {
  total++;
  contatos.textContent = 'Contatos cadastrados: ' + total;
});
```

### JQuery

```
var contatos = $('.contatos');
contatos.data('total', 0);
$('.botao-grava').click(function() {
  var total = contatos.data('total') + 1;
  contatos.data('total', total);
  contatos.text('Contatos cadastrados: ' + total);
});
```

# Frameworks Web

## - Manipulação do DOM

- Limitações do *JQuery*

1) Ausência de Padrões – as mesmas funcionalidades com *JQuery* podem ser escritas de diversas maneiras.

### Forma 1

```
var numerosEl = $('.numero');
$(numerosEl).on('keyup', function() {
    var resultado = 1;
    $.each(numerosEl, function() {
        resultado*=$(this).val();
    });
    $('.resultado').text(resultado);
});
```

### Forma 2

```
var numerosEl = $('input[type=text]');
$(numerosEl).keyup(function() {
    var resultado = 1;
    $.each(numerosEl, function() {
        resultado*=$(this).val();
    });
    $('.resultado').text(resultado);
});
```

# Frameworks Web

## - Manipulação do DOM

- Limitações do *Jquery* (cont.)

**2) Preso à estrutura do DOM** – É necessário saber qual função chamar para obter o valor do documento.

```
// se fosse um input, seria val()  
$('.resultado').text(resultado);
```

Também é necessário saber se cada elemento possui classe no documento HTML

```
var numerosEl = $('.numero');
```

```
<p>  
    Quando é <input class="numero"> vezes <input class="numero">  
</p>  
<p>Resultado: <span class="resultado"></span></p>
```

# Frameworks Web

## - Manipulação do DOM

- Limitações do *Jquery* (*cont.*)

3) **Testabilidade** – Como não existe separação entre a lógica e os elementos do *DOM*, testes automatizados se torna uma tarefa complexa, como por exemplo, realizar testes de unidade simulando a página.

```
// obtém um elemento do DOM
var numerosEl = $('.numero');
```

4) **Consistência entre Model e View** – toda vez que o *model* for atualizado, a apresentação na camada *view* também precisa ser atualizada.

```
// alterando o equivalente ao model
$.each(numerosEl, function() {
    resultado*=$(this).val();
});
// sincronizando o model com a view
$('.resultado').text(resultado);
```

# Frameworks Web

## - Manipulação do DOM

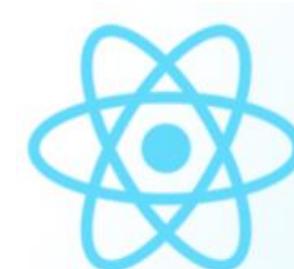
- Outras opções – *Frameworks MVC client-side*



ANGULARJS  
by Google



ANGULAR



ReactJS



BACKBONE.JS

# Estudo de Caso: *AngularJS*

## - Definições

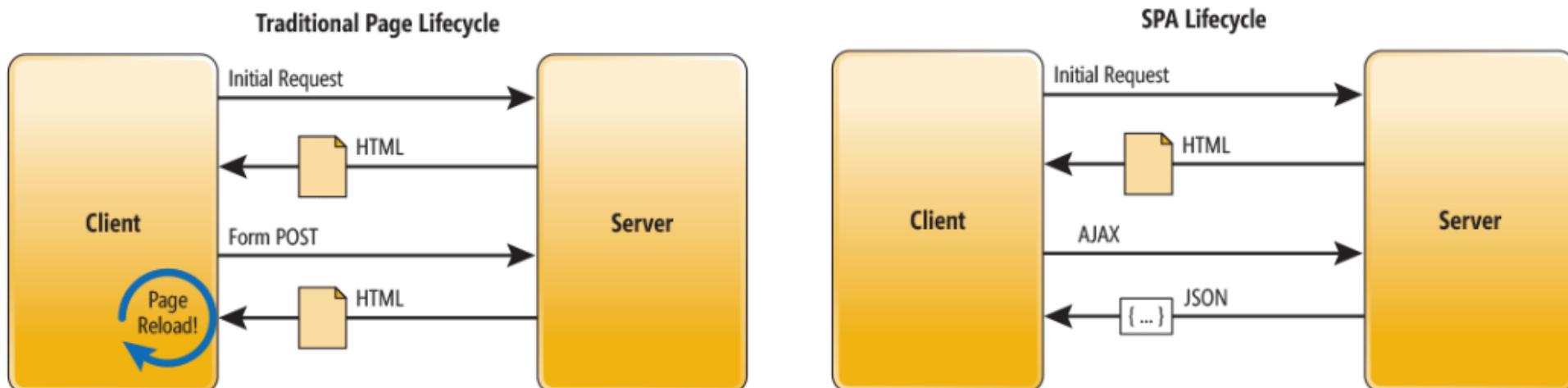
- *AngularJS* é um *framework JavaScript client-side*, criado pela *Google* e disponibilizado em 2009. Abstrai a manipulação do *DOM* de forma elegante, utilizando recursos como *two-way data binding*.



# Estudo de Caso: AngularJS

## - Características

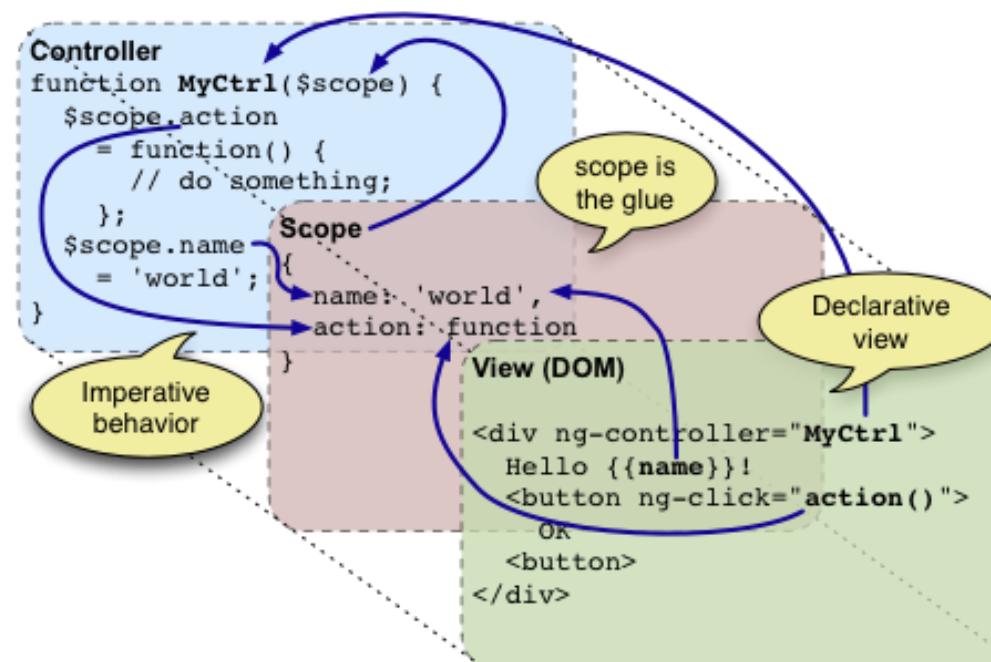
- AngularJS é **voltado para SPA** (*Single Page Web Applications*), que consiste basicamente em não recarregar a página durante o seu uso, ou seja, todo o *JavaScript* e *CSS* necessários já são carregados inicialmente. Com isso, escreve-se **menos código server-side**, **transferindo parte da lógica e dados para o lado client-side**.
- Neste contexto, o papel do AngularJS é **facilitar** o recebimento dos dados e **execução** da lógica de negócios diretamente no cliente.



# Estudo de Caso: *AngularJS*

## - Características

- *AngularJS* é um **framework MVC**. Com isso, toda a lógica é desacoplada dos elementos DOM, além de facilitar itens como manutenção, reusabilidade através de componentes e testabilidade.



# Estudo de Caso: AngularJS

## - Características

- AngularJS **força uma separação** entre o *front-end* e o *back-end*. Com isso, empresas podem ter equipes especializadas em cada parte da aplicação. Isso ajuda a eliminar situações onde a mesma equipe cria as duas camadas, especialmente quando se usa *frameworks* como ASP.NET e JSF.
- Em um projeto, as camadas **poderão evoluir em paralelo**.
- Esta separação pode inclusive envolver servidores, pois o *front-end* pode estar em um servidor razoavelmente escalável para tratar requisições *HTTP*, como o *Apache*.



# Estudo de Caso: *Angular*

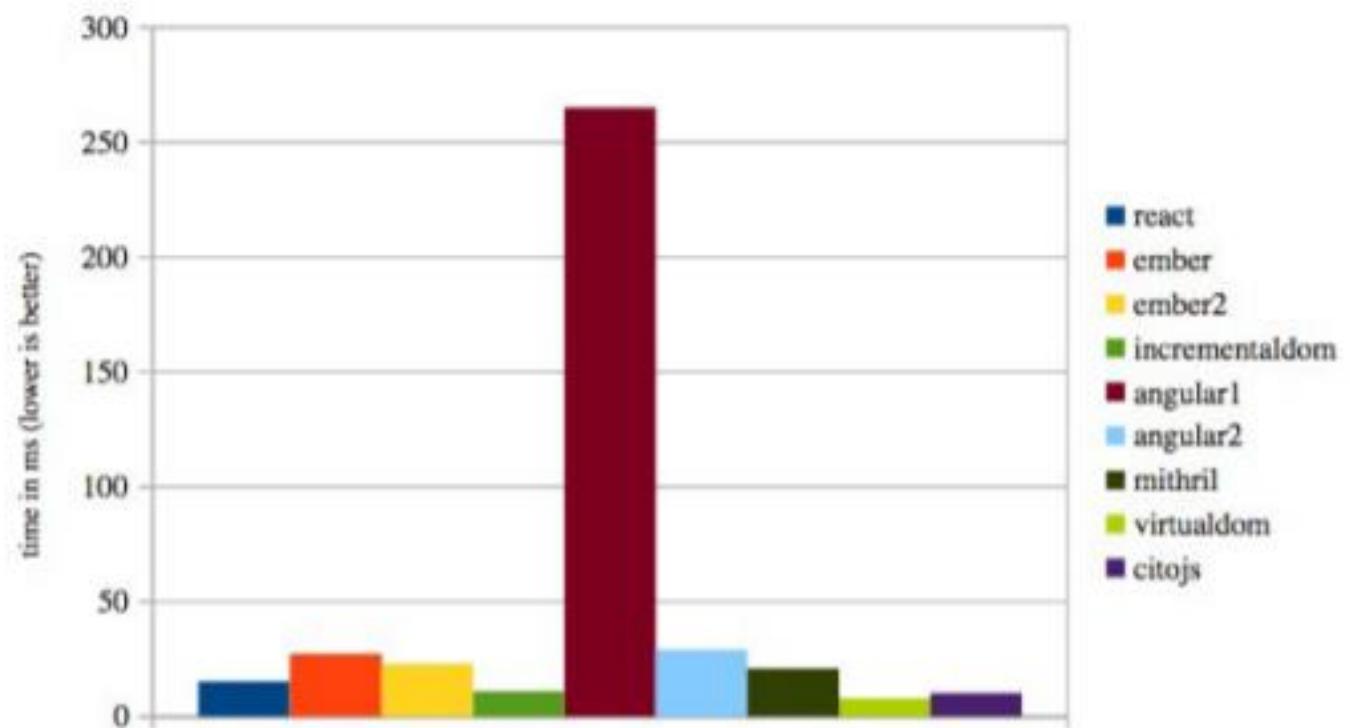
## - Definições

- Angular é um framework para desenvolvimento front-end, com HTML, CSS e Typescript, que no final é compilado para Javascript.
- Não é continuação do AngularJS. É um framework novo e remodelado, codificado do zero, com lições aprendidas do AngularJS.
- Aplicativos Mobile como Ionic possuem como base o Angular.



# Estudo de Caso: Angular

## - Tempo de resposta



# Estudo de Caso: *Angular*

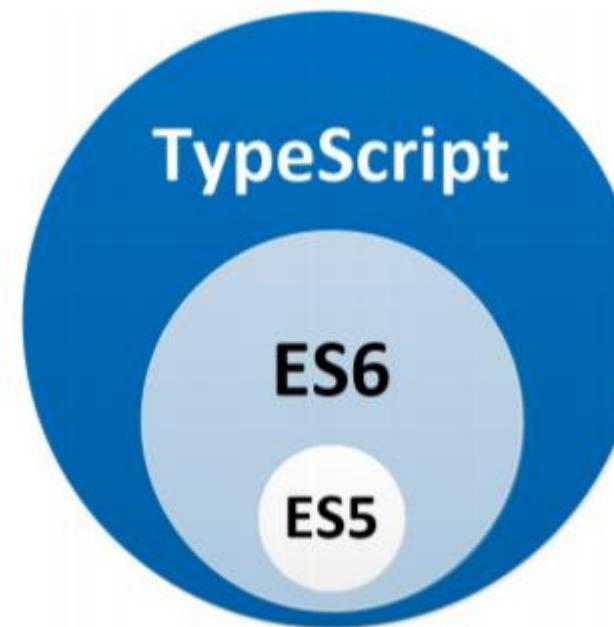
## - Definições

- O desenvolvimento em Angular é feito por meio de codificação TypeScript.
- Implementa funcionalidades do ES6
  - Tipagem de variáveis
  - Sintaxe clara e fácil de entender, parecendo-se com C# e Java.
- Javascript é somente um dialeto/apelido para a linguagem ECMAScript (ES)
- Em 2016 o ES chegou na versão 6
- Typescript permite escrever códigos utilizando estruturas fortemente tipadas e ter o código compilado para Javascript.

# Estudo de Caso: *Angular*

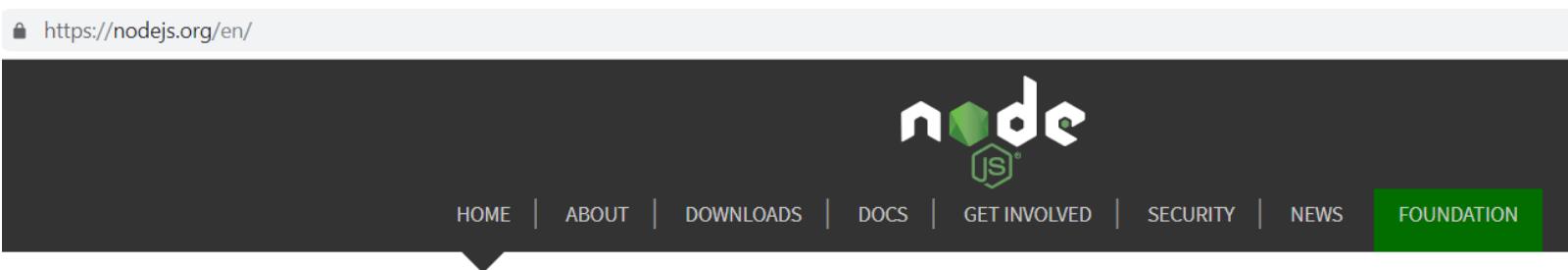
## - Definições

- TypeScript permite escrever o código na forma como se codifica no paradigma de Orientação a Objetos.



# Estudo de Caso: *Angular*

- Instalação
- NodeJs – o Angular2 roda em cima desta plataforma. Versão LTS



Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

Download for Windows (x64)

**10.15.1 LTS**

Recommended For Most Users

**11.10.0 Current**

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)    [Other Downloads](#) | [Changelog](#) | [API Docs](#)

# Estudo de Caso: AngularJS

## - Características

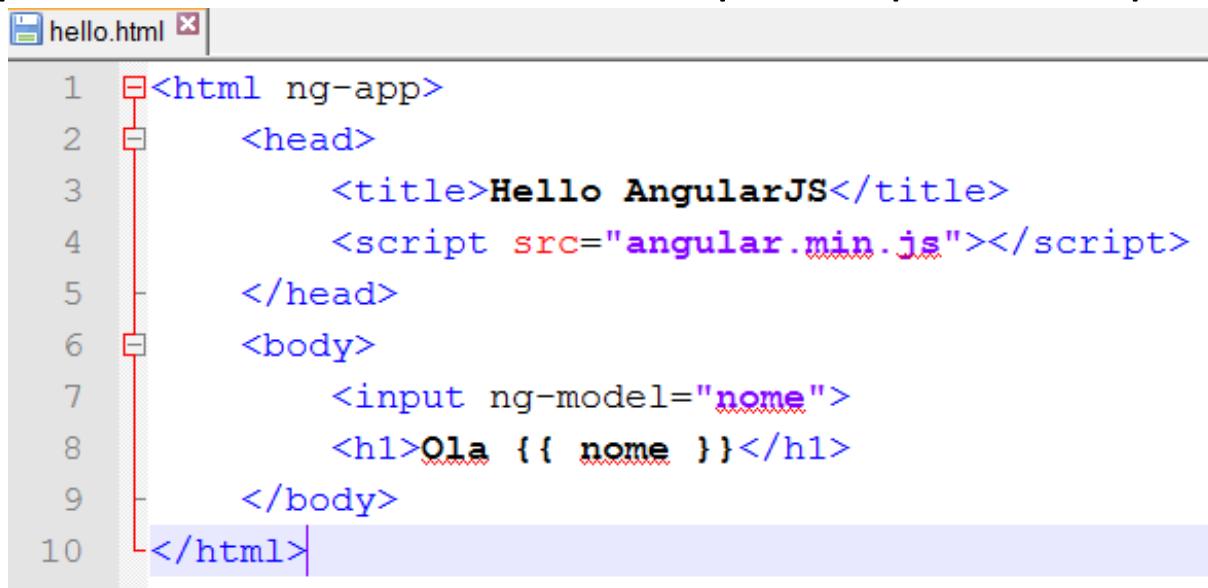
- AngularJS estende a sintaxe do *HTML*, **ampliando seu vocabulário** para expressar componentes usando diretivas. As diretivas atualizam partes da tela **sem qualquer intervenção manual**.
- Depois que o navegador transforma o texto da marcação na árvore DOM, o AngularJS percorre a estrutura DOM analisada. A cada diretiva encontrada, o AngularJS **executa sua lógica** para transformar diretivas em partes dinâmicas da tela.

```
<html ng-app="crudAtendimento">
<div ng-controller="AtendimentoController">
<button ng-click="salvar()">
<input type="text" ng-model="atendimento.protocolo">
<tr ng-repeat="atendimento in atendimentos | filter:criterio" ng-click="seleciona(atendimento)">
```

# Estudo de Caso: *AngularJS*

## - Hello World

- Deve-se incluir a biblioteca do *AngularJS* no documento
- Deve-se incluir a propriedade *ng-app* para “ativar” o *AngularJS*
- *Tags* personalizados com atributos para adicionar comportamento dinâmico no documento *HTML*
- Chaves duplas usadas como delimitador para expressões que exibem valores do modelo.

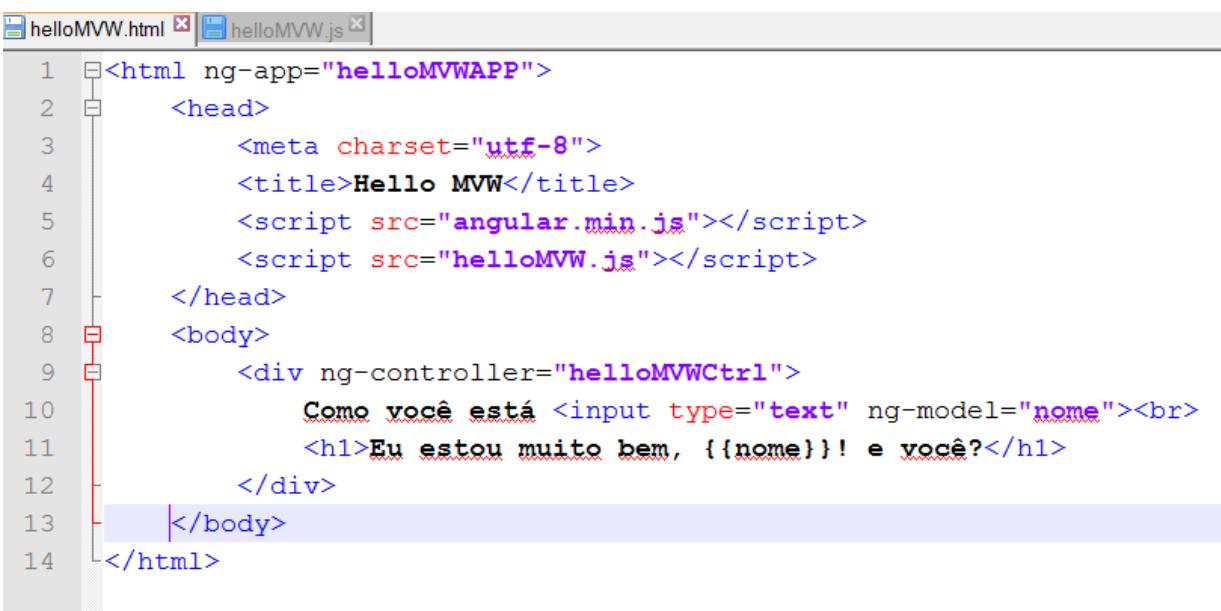


```
hello.html
1 <html ng-app>
2   <head>
3     <title>Hello AngularJS</title>
4     <script src="angular.min.js"></script>
5   </head>
6   <body>
7     <input ng-model="nome">
8     <h1>Ola {{ nome }}</h1>
9   </body>
10  </html>
```

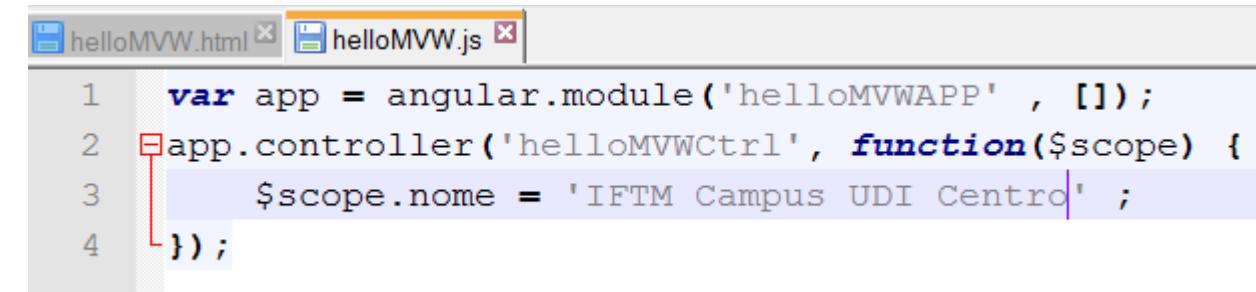
# Estudo de Caso: AngularJS

## - Utilizando AngularJS como MVW

- Possui estratégias envolvendo inicialização de dados e lógica, separando responsabilidades em camadas view e control.



```
helloMVW.html x helloMVW.js x
1 <html ng-app="helloMVWAPP">
2   <head>
3     <meta charset="utf-8">
4     <title>Hello MVW</title>
5     <script src="angular.min.js"></script>
6     <script src="helloMVW.js"></script>
7   </head>
8   <body>
9     <div ng-controller="helloMVWCtrl">
10       Como você está <input type="text" ng-model="nome"><br>
11       <h1>Eu estou muito bem, {{nome}}! e você?</h1>
12     </div>
13   </body>
14 </html>
```



```
helloMVW.html x helloMVW.js x
1 var app = angular.module('helloMVWAPP' , []);
2 app.controller('helloMVWCtrl' , function($scope) {
3   $scope.nome = 'IFTM Campus UDI Centro';
4 }) ;
```

# Estudo de Caso: *AngularJS*

## - **\$Scope**

- Um *\$Scope* é um objeto do *AngularJS* que expõe o modelo de domínio para a camada de visão. Tudo o que for atribuído para uma instância de escopo, sejam dados ou funcionalidades, poderão estar acessíveis para a camada de visão.
- Controla em qual parte do modelo as operações estão disponíveis para a camada de visão.
- Exemplo:

```
var OláCtrl = function ($scope) {
    $scope.lerNome = function() {
        return $scope.nome;
    };
};
```

`<h1>Olá, {{ lerNome() }}!</h1>`

# Estudo de Caso: AngularJS

## - Diretiva ng-repeat

- Permite iterar sobre uma coleção de objetos e criar novos elementos DOM para cada item em uma coleção.
- Exemplo:

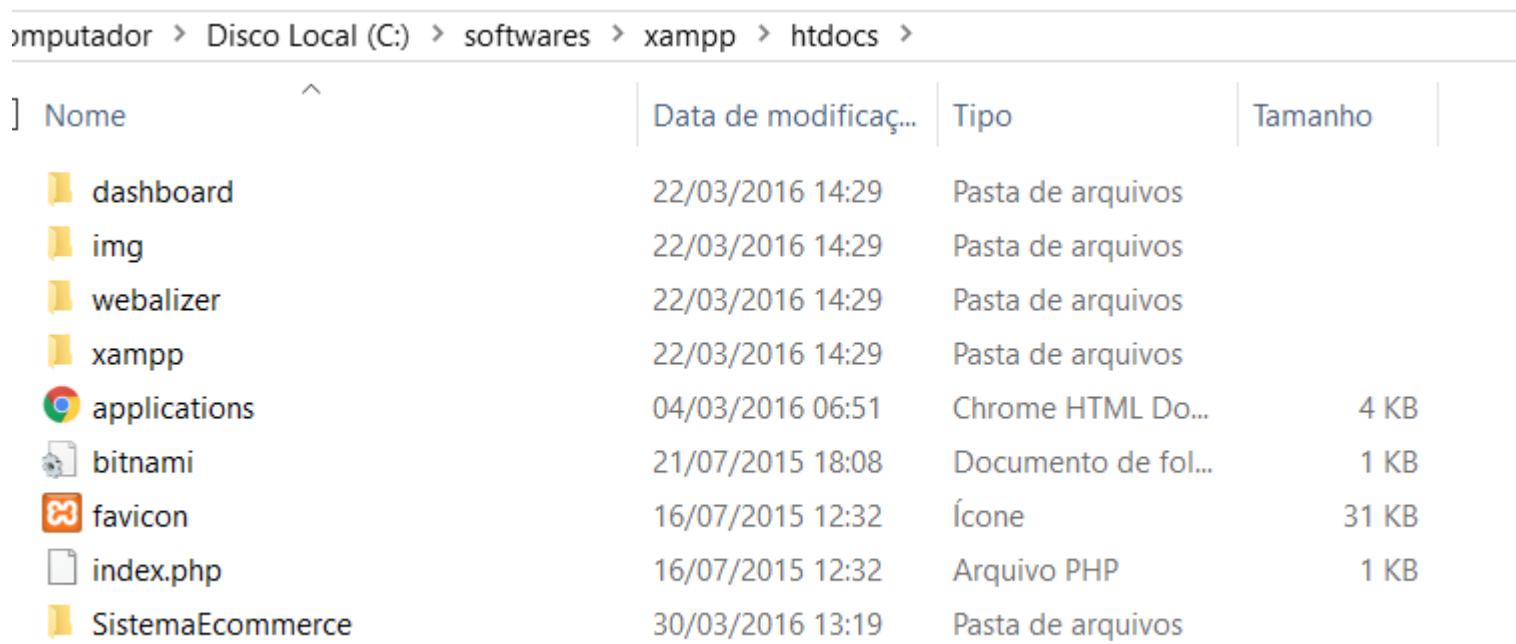
```
var MundoCtrl = function ($scope) {  
    $scope.populacao = 70000000;  
    $scope.paises = [  
        {nome: 'Brasil', populacao: 63.1},  
        {nome: 'Alemanhã', populacao: 60.1}  
    ];  
};
```

```
<ul ng-controller="MundoCtrl">  
    <li ng-repeat="pais in paises">  
        {{pais.nome}} tem uma população de {{pais.populacao}}  
    </li>  
    <hr>  
    População do mundo: {{populacao}} Milhões de pessoas  
</ul>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap

- Dentro do diretório *htdocs* do *Xampp*, criar um diretório *SistemaEcommerce*



Computador > Disco Local (C:) > softwares > xampp > htdocs >			
Nome	Data de modificaç...	Tipo	Tamanho
dashboard	22/03/2016 14:29	Pasta de arquivos	
img	22/03/2016 14:29	Pasta de arquivos	
webalizer	22/03/2016 14:29	Pasta de arquivos	
xampp	22/03/2016 14:29	Pasta de arquivos	
applications	04/03/2016 06:51	Chrome HTML Do...	4 KB
bitnami	21/07/2015 18:08	Documento de fol...	1 KB
favicon	16/07/2015 12:32	Ícone	31 KB
index.php	16/07/2015 12:32	Arquivo PHP	1 KB
SistemaEcommerce	30/03/2016 13:19	Pasta de arquivos	

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Criar a estrutura de diretórios, adicionando os arquivos abaixo (buscar em links – penúltimo slide)

Este Computador > Disco Local (C:) > xampp > htdocs > SistemaEcommerce				
	Nome	Data de modificaç...	Tipo	Tamanho
:	css	14/04/2018 10:23	Pasta de arquivos	
:	img	14/03/2019 16:46	Pasta de arquivos	
:	js	14/03/2019 17:13	Pasta de arquivos	
:	index.html	14/03/2019 17:13	Chrome HTML Do...	1 KB
:	menu.html	14/03/2019 17:13	Chrome HTML Do...	3 KB

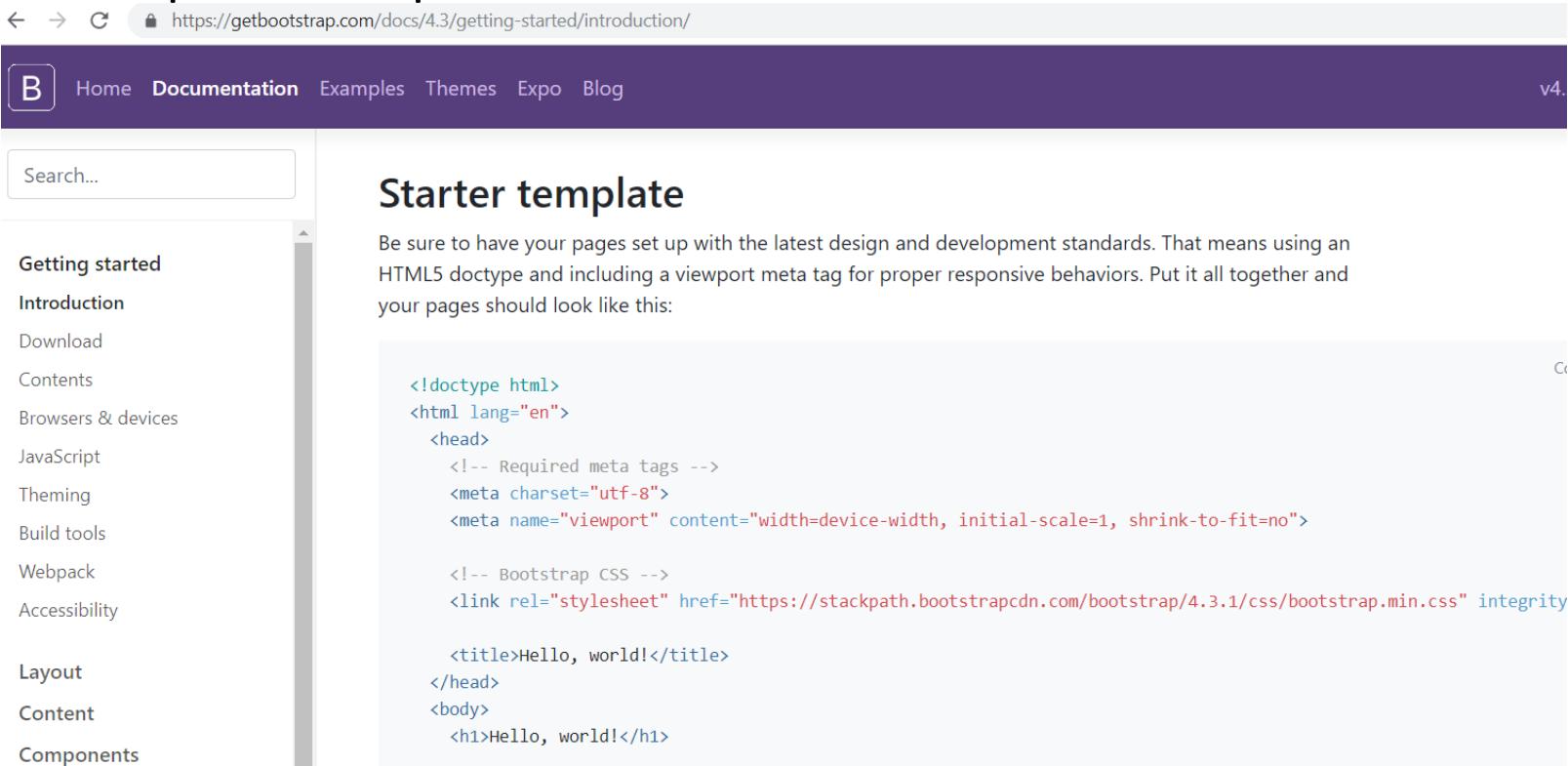
Este Computador > Disco Local (C:) > xampp > htdocs > SistemaEcommerce > css				
	Nome	Data de modificaç...	Tipo	Tamanho
	bootstrap.css	13/02/2019 14:01	Documento de fol...	188 KB

Computador > Disco Local (C:) > xampp > htdocs > SistemaEcommerce > js				
	Nome	Data de modificaç...	Tipo	Tamanho
	angular.js	13/03/2019 17:25	Arquivo JavaScript	1.340 KB
	angular-resource.js	13/03/2019 17:25	Arquivo JavaScript	38 KB
	bootstrap.js	09/04/2018 12:58	Arquivo JavaScript	120 KB
	jquery-3.3.1.slim.min.js	14/03/2019 17:13	Arquivo JavaScript	69 KB
	popper.min.js	14/03/2019 17:08	Arquivo JavaScript	21 KB

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- No arquivo *index.html*, adicionar a estrutura que está no site do bootstrap, clicando em Get Started -> e pegando o código-fonte em Starter Template, modificando os links para os arquivos baixados



The screenshot shows a web browser displaying the Bootstrap documentation at <https://getbootstrap.com/docs/4.3/getting-started/introduction/>. The page has a purple header with navigation links for Home, Documentation (which is selected), Examples, Themes, Expo, and Blog. A sidebar on the left lists various documentation sections like Getting started, Introduction, Download, Contents, Browsers & devices, JavaScript, Theming, Build tools, Webpack, Accessibility, Layout, Content, and Components. The main content area features a heading 'Starter template' and a paragraph about ensuring pages are set up with the latest standards. Below this is a code editor showing the HTML code for the starter template:

```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="...
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Efetuar o download de alguma imagem para adicionar na página inicial e colocar dentro do diretório *img*;

me Computador > Disco Local (C) > softwares > xampp > apache > SistemaEcommerce > img

<input type="checkbox"/> Nome	Data de modificaç...	Tipo	Tamanho
 logo	30/10/2015 01:07	Arquivo JPG	330 KB

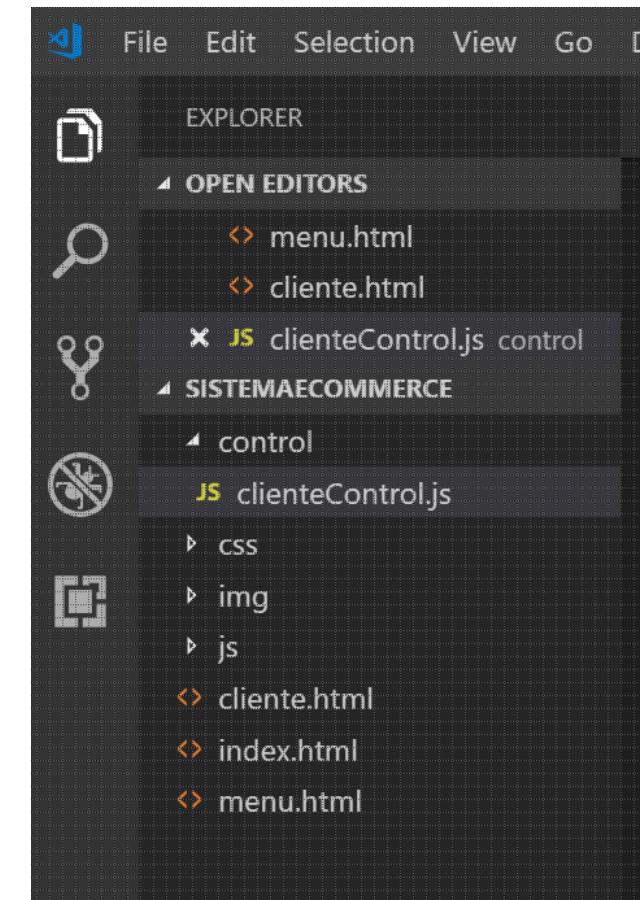
- Adicionar na página index.html o suporte para AngularJS;
- Adicionar importação para o arquivo menu.html que será criado (próximo slide).
- Ao criar o arquivo menu.html, copie todo o conteúdo do index.html, retirando apenas a imagem e o include para o menu.

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap (cont.)

- Conteúdo do arquivo index.html (também pode ser usado na ferramenta Visual Studio Code – File – Open Folder)

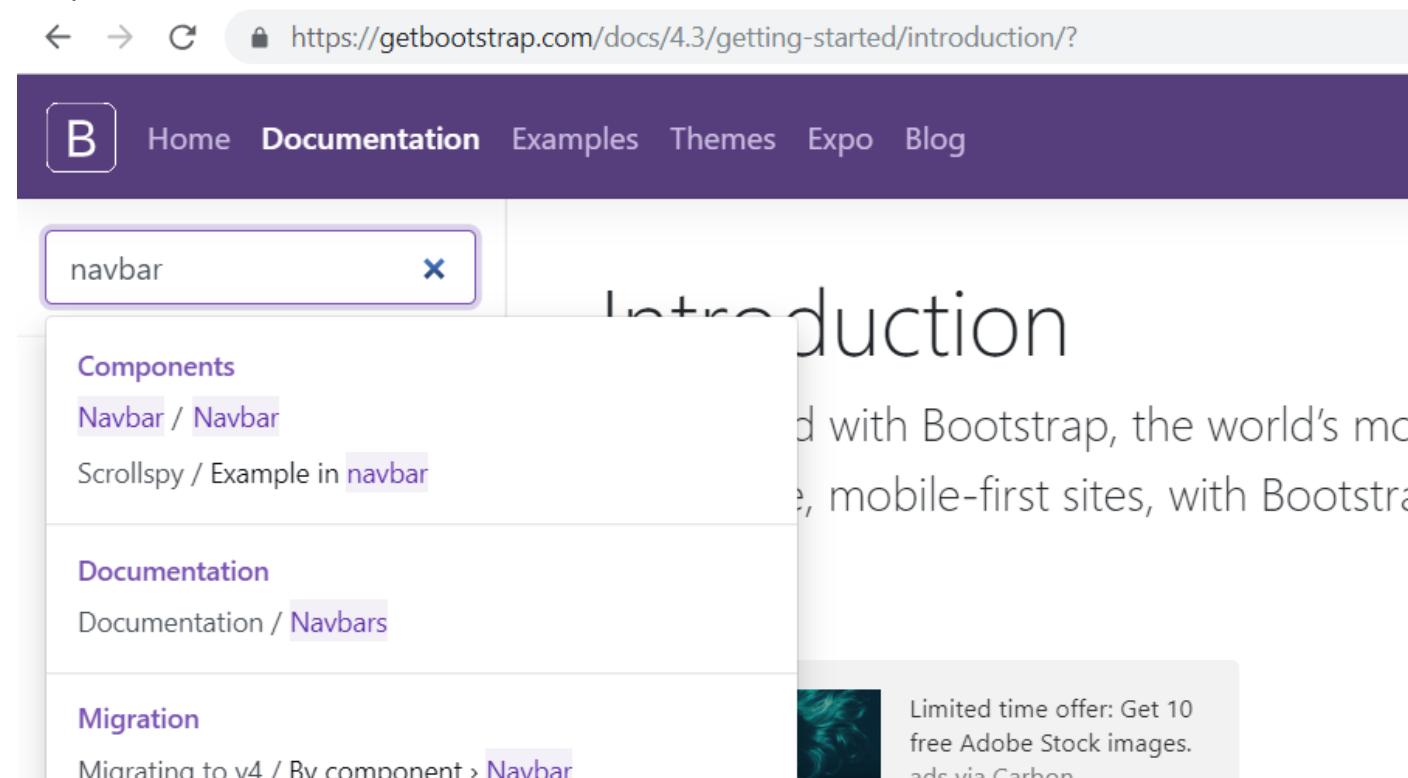
```
1 <html lang="en" ng-app>
2   <head>
3     <link rel="stylesheet" href="css/bootstrap.css">
4     <title>Sistema Ecommerce</title>
5   </head>
6   <body>
7     <script src="js/jquery-3.3.1.slim.min.js"></script>
8     <script src="js/popper.min.js"></script>
9     <script src="js/bootstrap.js"></script>
10    <script src="js/angular.js"></script>
11    <div ng-include src="'menu.html'"></div>
12    
13  </body>
14 </html>
```



# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap (cont.)

- No site do Bootstrap (<http://getbootstrap.com/>), acessar a aba *documentation*
- Nos links à direita, clicar em *navbar*



# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

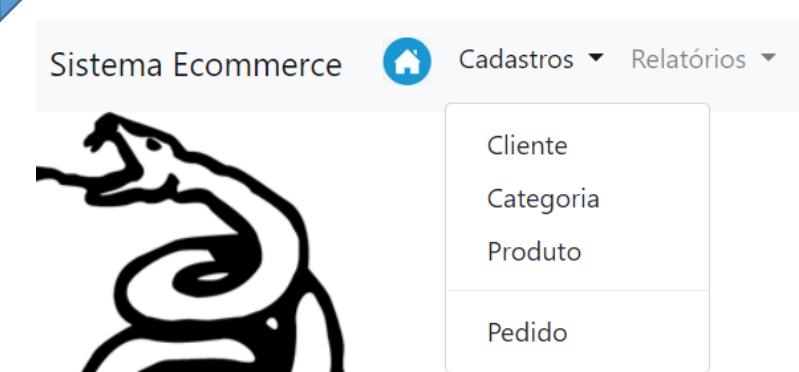
- Copie o código-fonte mostrado e cole na página *menu.html*.



```
Navbar Home Link Dropdown ▾ Disabled
Search Search

<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Navbar</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
```

- Edite para ser mostrado conforme figura



# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap (cont.)

- Em *menu.html*, adicione um *link* para imagem

```
<li class="nav-item active">
  <a class="nav-link" href="index.html">
    
    <span class="sr-only">(current)</span></a>
</li>
```

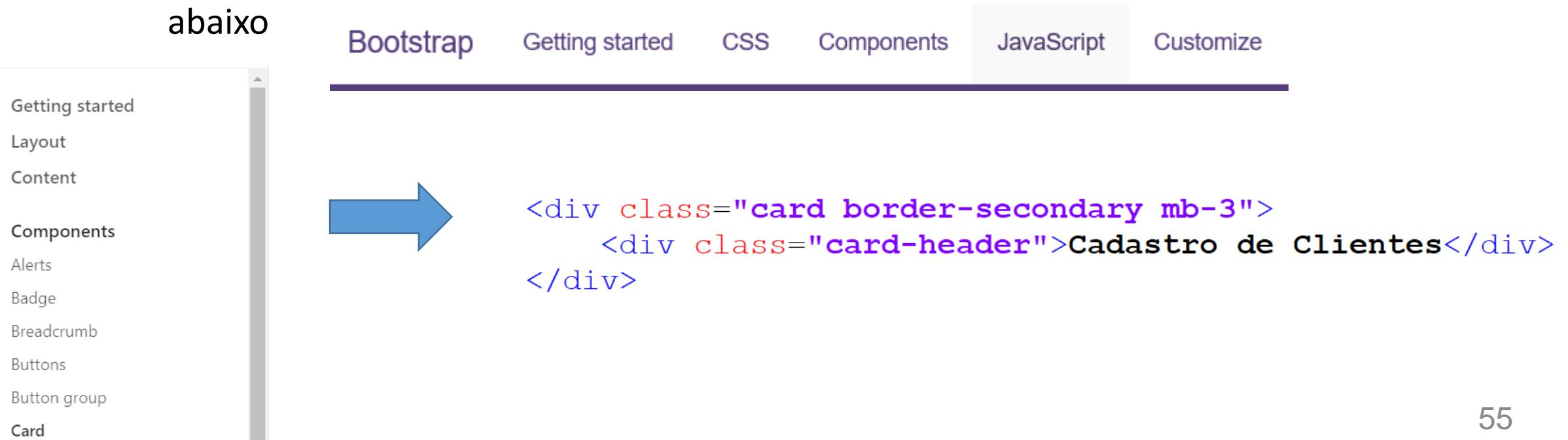
- Nos links, adicione os caminhos abaixo

```
<li class="nav-item dropdown">
  <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"
    data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Cadastros
  </a>
  <div class="dropdown-menu" aria-labelledby="navbarDropdown">
    <a class="dropdown-item" href="cliente.html">Cliente</a>
    <a class="dropdown-item" href="categoria.html">Categoria</a>
    <a class="dropdown-item" href="produto.html">Produto</a>
    <div class="dropdown-divider"></div>
    <a class="dropdown-item" href="pedido.html">Pedido</a>
  </div>
</li>
<li class="nav-item dropdown">
  <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button"
    data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
    Relatórios
  </a>
  <div class="dropdown-menu" aria-labelledby="navbarDropdown">
    <a class="dropdown-item" href="relatorioPedido.html">Pedido</a>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap (cont.)

- Crie o arquivo *cliente.html*, com conteúdo quase idêntico ao *index.html*, retirando apenas a tag de imagem.
- No site do *Bootstrap-Documetation*, pesquise por Cards e cole o código-fonte abaixo



The screenshot shows the Bootstrap Documentation website with the 'Cards' component selected. A large blue arrow points from the left margin towards the code example below.

Bootstrap Documentation navigation bar:

- Getting started
- Layout
- Content
- Components
- Alerts
- Badge
- Breadcrumb
- Buttons
- Button group
- Card

Component tabs:

- Bootstrap
- Getting started
- CSS
- Components
- JavaScript
- Customize

Code example:

```
<div class="card border-secondary mb-3">
  <div class="card-header">Cadastro de Clientes</div>
</div>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Pesquise por Tabs    Tabs

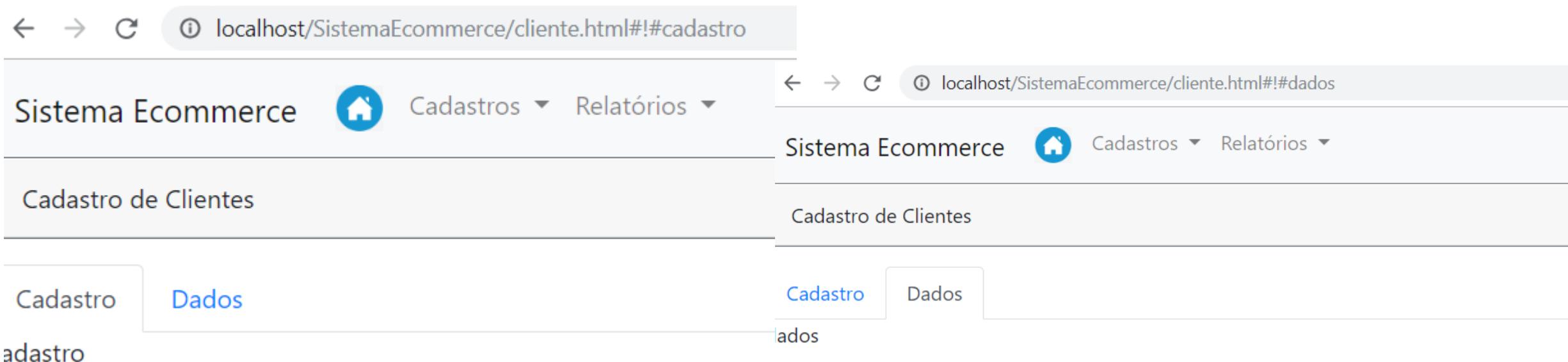
Takes the basic nav from above and adds the `.nav-tabs` class to generate a tabbed interface. Use them to

```
<ul class="nav nav-tabs" id="clienteTab" role="tablist">
  <li class="nav-item">
    <a class="nav-link active" id="cadastro-tab" data-toggle="tab" href="#cadastro" role="tab" aria-controls="tabCadastro" aria-selected="true">
      Cadastro</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" id="dados-tab" data-toggle="tab" href="#dados" role="tab" aria-controls="tabDados" aria-selected="false">
        Dados</a>
    </li>
  </ul>
  <div class="tab-content" id="clienteTabContent">
    <div class="tab-pane fade show active" id="cadastro" role="tabpanel" aria-labelledby="cadastro-tab">
      cadastro
    </div>
    <div class="tab-pane fade" id="dados" role="tabpanel" aria-labelledby="dados-tab">
      dados
    </div>
  </div>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Resultado:



The image displays two side-by-side screenshots of a web application interface, likely built using Bootstrap, for managing client data in a commerce system.

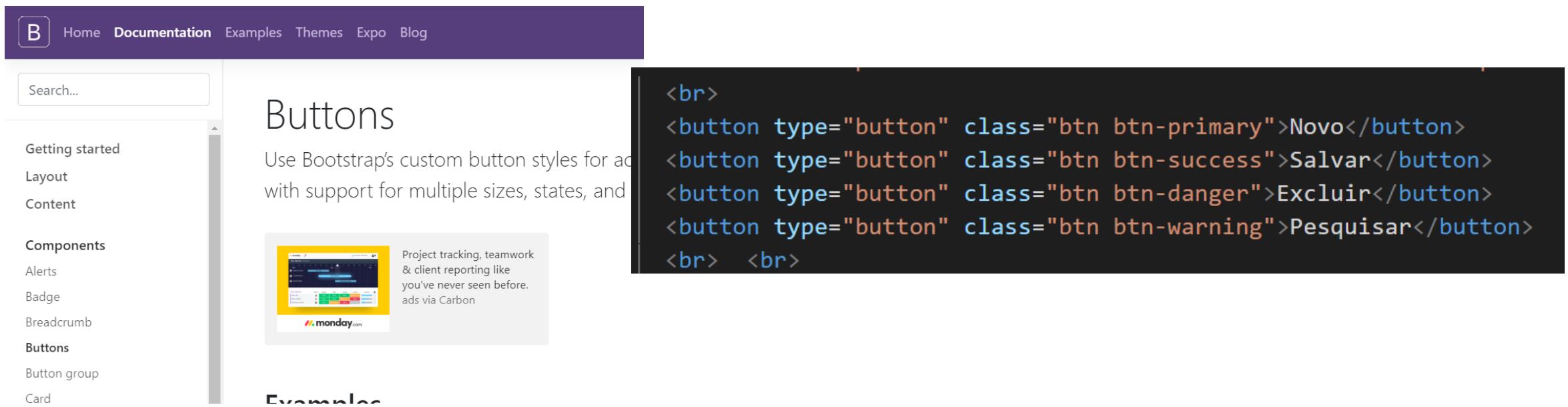
**Screenshot 1 (Left):** The URL in the browser is `localhost/SistemaEcommerce/cliente.html#!#cadastro`. The page title is "Sistema Ecommerce". The main content area is titled "Cadastro de Clientes". Below this, there are two tabs: "Cadastro" (highlighted in blue) and "Dados".

**Screenshot 2 (Right):** The URL in the browser is `localhost/SistemaEcommerce/cliente.html#!#dados`. The page title is "Sistema Ecommerce". The main content area is titled "Cadastro de Clientes". Below this, there are two tabs: "Cadastro" and "Dados" (highlighted in blue).

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap (cont.)

- *Pesquisa por button*



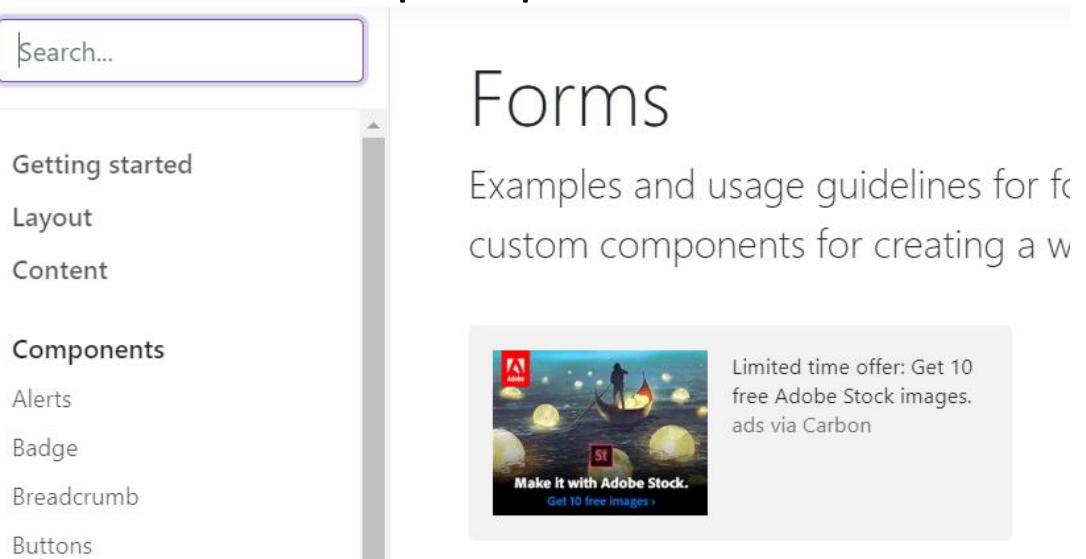
The screenshot shows the Bootstrap documentation website. The top navigation bar includes a logo with a letter 'B', 'Home', 'Documentation' (which is highlighted in purple), 'Examples', 'Themes', 'Expo', and 'Blog'. A search bar is also present. On the left, a sidebar lists 'Getting started', 'Layout', 'Content', 'Components' (with 'Buttons' selected), 'Alerts', 'Badge', 'Breadcrumb', 'Buttons', 'Button group', and 'Card'. The main content area has a title 'Buttons' and a sub-section 'Use Bootstrap's custom button styles for action items...'. It features a screenshot of a project management tool and a snippet of code for creating buttons:

```
<br>
<button type="button" class="btn btn-primary">Novo</button>
<button type="button" class="btn btn-success">Salvar</button>
<button type="button" class="btn btn-danger">Excluir</button>
<button type="button" class="btn btn-warning">Pesquisar</button>
<br> <br>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Pesquise por forms



The screenshot shows the Bootstrap documentation page for 'Forms'. The left sidebar has a search bar and navigation links for 'Getting started', 'Layout', 'Content', 'Components', 'Alerts', 'Badge', 'Breadcrumb', and 'Buttons'. The main content area has a title 'Forms' and a sub-section 'Examples and usage guidelines for forms'. It includes a 'Form Validation' section with code snippets and a 'Custom Components' section with a screenshot of an Adobe Stock advertisement.

Search...

Getting started

Layout

Content

Components

Alerts

Badge

Breadcrumb

Buttons

## Forms

Examples and usage guidelines for forms

Form Validation

```
<form class="was-validated form-row">
  <div class="col-md-1 mb-3">
    <label for="txtCodigo">Código</label>
    <input type="text" class="form-control" disabled id="txtCodigo">
  </div>

  <div class="col-md-4 mb-3">
    <label for="txtNome">Nome</label>
    <input type="text" class="form-control is-valid" id="txtNome" required>
    <div class="invalid-feedback">
      Digite o nome.
    </div>
  </div>
  <div class="col-md-4 mb-3">
    <label for="txtCargo">Cargo</label>
    <input type="text" class="form-control is-valid" id="txtCargo" required>
    <div class="invalid-feedback">
      Digite o cargo.
    </div>
  </div>
</form>
```

Limited time offer: Get 10 free Adobe Stock images. ads via Carbon

Make it with Adobe Stock. Get 10 free images.

```
<form class="was-validated form-row">
  <div class="col-md-1 mb-3">
    <label for="txtCodigo">Código</label>
    <input type="text" class="form-control" disabled id="txtCodigo">
  </div>

  <div class="col-md-4 mb-3">
    <label for="txtNome">Nome</label>
    <input type="text" class="form-control is-valid" id="txtNome" required>
    <div class="invalid-feedback">
      Digite o nome.
    </div>
  </div>
  <div class="col-md-4 mb-3">
    <label for="txtCargo">Cargo</label>
    <input type="text" class="form-control is-valid" id="txtCargo" required>
    <div class="invalid-feedback">
      Digite o cargo.
    </div>
  </div>
</form>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

```
<div class="col-md-5 mb-3">
  <label for="txtEndereco">Endere&ccedil;o</label>
  <input type="text" class="form-control is-valid" id="txtEndereco" required>
  <div class="invalid-feedback">
    | Dite o endere&ccedil;o.
  </div>
</div>

<div class="col-md-3 mb-3">
  <label for="txtCidade">Cidade</label>
  <input type="text" class="form-control is-valid" id="txtCidade" required>
  <div class="invalid-feedback">
    | Dite o cidade.
  </div>
</div>

<div class="col-md-2 mb-2">
  <label for="txtCep">CEP</label>
  <input type="text" class="form-control is-valid" id="txtCep" required>
  <div class="invalid-feedback">
    | Dite o CEP.
  </div>
</div>
```

```
<div class="col-md-3 mb-3">
  <label for="txtPais">Pa&iacute;s</label>
  <input type="text" class="form-control is-valid" id="txtPais" required>
  <div class="invalid-feedback">
    | Dite o pa&iacute;s.
  </div>
</div>

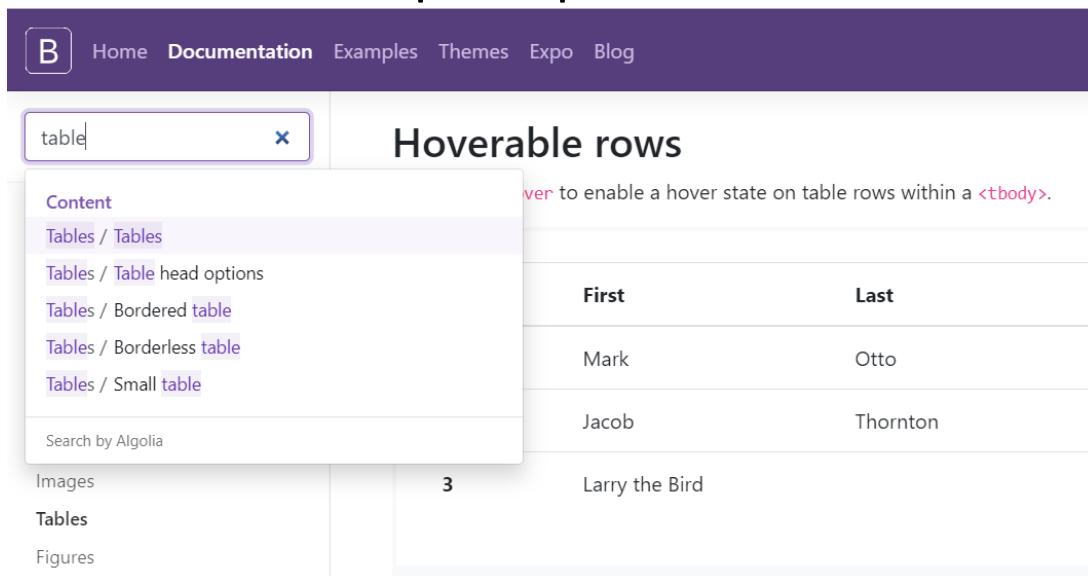
<div class="col-md-2 mb-2">
  <label for="txtTelefone">Telefone</label>
  <input type="text" class="form-control is-valid" id="txtTelefone" required>
  <div class="invalid-feedback">
    | Dite o telefone.
  </div>
</div>

<div class="col-md-2 mb-2">
  <label for="txtFax">Fax</label>
  <input type="text" class="form-control is-valid" id="txtFax" required>
  <div class="invalid-feedback">
    | Dite o fax.
  </div>
</div>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)

- Pesquisar por Table

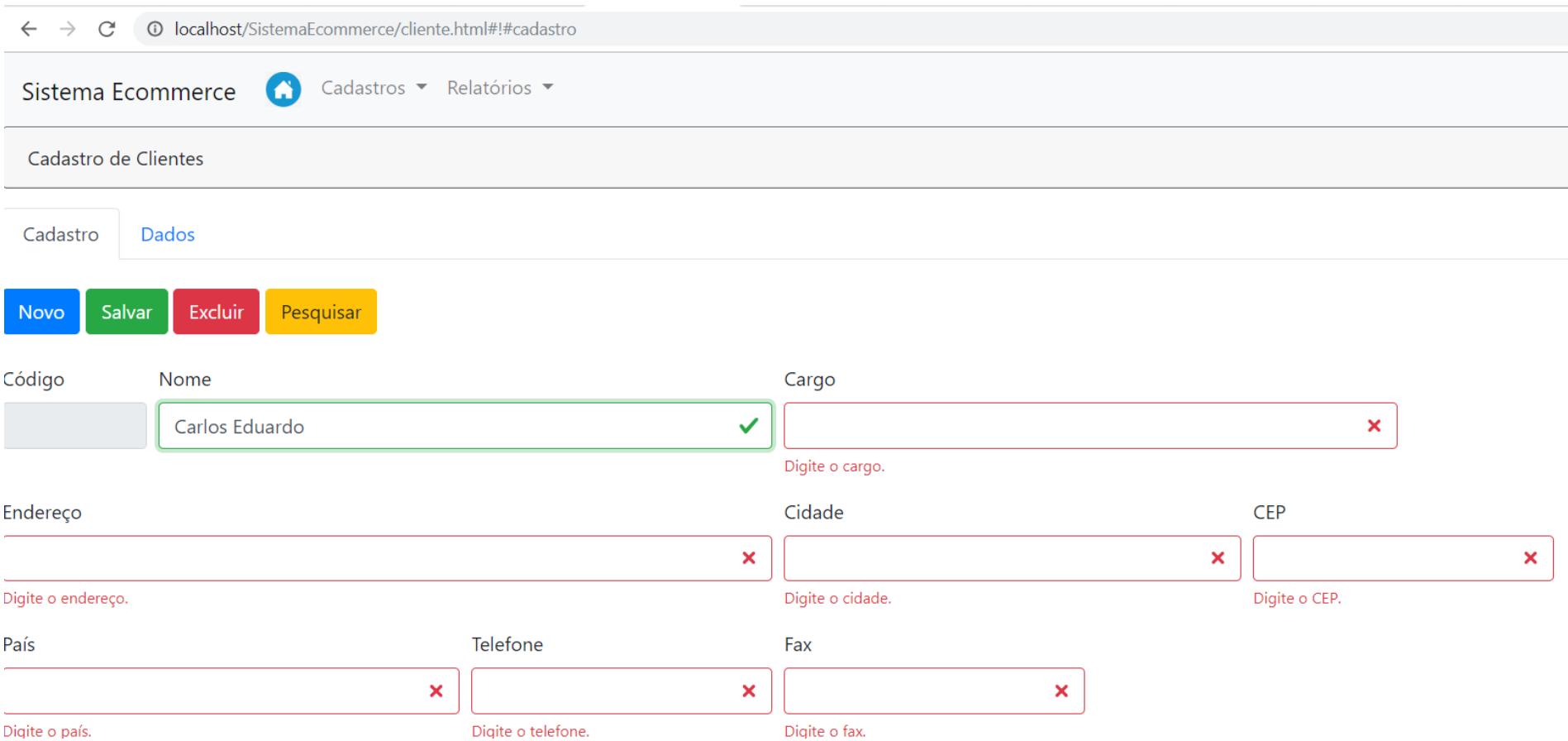


The screenshot shows the Bootstrap documentation page for tables. A search bar at the top has 'table' typed into it. Below the search bar, the main content area has a title 'Hoverable rows'. To the left, there's a sidebar with a navigation tree under 'Content' and a search bar labeled 'Search by Algolia'. The main content area displays a table with two columns: 'First' and 'Last'. It contains three rows with data: 'Mark' and 'Otto', 'Jacob' and 'Thornton', and 'Larry the Bird'. At the bottom of the table, there's a note: 'ver to enable a hover state on table rows within a <tbody>'.

```
<div class="tab-pane fade" id="dados" role="tabpanel" aria-labelledby="dados-tab">
<div role="tabpanel" class="tab-pane" id="dados">
  <input type="text" class="form-control" placeholder="O que voc&ecirc; est&aacute; procurando?" /><br />
  <table class="table table-hover">
    <thead>
      <tr>
        <th>C&oacute;digo</th>
        <th>Nome</th>
        <th>Cargo</th>
        <th>Endere&ccedil;o</th>
        <th>Cidade</th>
        <th>CEP</th>
        <th>Pa&iacute;s</th>
        <th>Telefone</th>
        <th>Fax</th>
      </tr>
    </thead>
    <tbody>
    </tbody>
  </table>
</div>
</div>
```

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)



The screenshot shows a web application interface for managing client data. At the top, there's a header with a back button, forward button, refresh button, and a URL bar showing "localhost/SistemaEcommerce/cliente.html#!#cadastro". Below the header, the title "Sistema Ecommerce" is followed by navigation links "Cadastros" and "Relatórios". The main content area is titled "Cadastro de Clientes". It has two tabs: "Cadastro" (selected) and "Dados". Below the tabs are four buttons: "Novo" (blue), "Salvar" (green), "Excluir" (red), and "Pesquisar" (yellow). The "Nome" field contains "Carlos Eduardo" and has a green checkmark icon to its right. The "Cargo" field is empty and has a red "X" icon to its right, with the placeholder "Digite o cargo.". The "Endereço" field is empty and has a red "X" icon to its right, with the placeholder "Digite o endereço.". The "Cidade" field is empty and has a red "X" icon to its right, with the placeholder "Digite a cidade.". The "CEP" field is empty and has a red "X" icon to its right, with the placeholder "Digite o CEP.". The "País" field is empty and has a red "X" icon to its right, with the placeholder "Digite o país.". The "Telefone" field is empty and has a red "X" icon to its right, with the placeholder "Digite o telefone.". The "Fax" field is empty and has a red "X" icon to its right, with the placeholder "Digite o fax.". All fields with validation errors have a red border.

# Estudo de Caso: *Bootstrap*

## - Criação de uma tela CRUD usando bootstrap(cont.)



The screenshot shows a web browser displaying a client-side application for managing customer data. The URL in the address bar is `localhost/SistemaEcommerce/cliente.html#!#dados`. The page has a header with the title "Sistema Ecommerce" and navigation links for "Cadastros" and "Relatórios". Below the header, the section "Cadastro de Clientes" is visible. A tab menu at the top allows switching between "Cadastro" (selected) and "Dados". A search bar asks "O que você está procurando?". A table below lists customer data with columns: Código, Nome, Cargo, Endereço, Cidade, CEP, País, Telefone, and Fax.

Código	Nome	Cargo	Endereço	Cidade	CEP	País	Telefone	Fax

# Estudo de Caso: AngularJS

- Para que o DOM seja modificado facilmente com esta tela CRUD, será necessário adicionar diretivas AngularJS no html

- 1) Diretiva ng-app

```
1 <html lang="en" ng-app="clienteModule">
```

- 2) Diretiva ng-controller

```
8 <body ng-controller="clienteControl">
```

- 3) Diretiva ng-click

```
<button type="button" ng-click="novo()" class="btn btn-primary">Novo</button>
<button type="button" ng-click="salvar()" class="btn btn-success">Salvar</button>
<button type="button" ng-click="excluir()" class="btn btn-danger">Excluir</button>
<button type="button" ng-click="pesquisar()" class="btn btn-warning">Pesquisar</button>
```

# Estudo de Caso: AngularJS

- Para que o DOM seja modificado facilmente com esta tela CRUD, será necessário adicionar diretivas AngularJS no html
  - 4) Diretiva ng-model

```
<label for="txtCodigo">C&acute;digo</label> <label for="txtTelefone">Telefone</label>
<input type="text" ng-model="cliente.codigo" class="form-control" <input type="text" ng-model="cliente.telefone"
<label for="txtNome">Nome</label> <label for="txtFax">Fax</label>
<input type="text" ng-model="cliente.nome" class="form-control" <input type="text" ng-model="cliente.fax"
<label for="txtCargo">Cargo</label> <input type="text" ng-model="criterio" class="form-control"
<input type="text" ng-model="cliente.cargo" class="form-control" placeholder="O que v&oc;&ecirc; est&aacute; procurando?" /><br />
<label for="txtEndereco">Endere&ccedil;o</label>
<input type="text" ng-model="cliente.endereco" class="form-control"
<label for="txtCidade">Cidade</label>
<input type="text" ng-model="cliente.cidade" class="form-control"
<label for="txtPais">Pa&iacute;s</label>
<input type="text" ng-model="cliente.pais" class="form-control"
```

# Estudo de Caso: AngularJS

- Para que o DOM seja modificado facilmente com esta tela CRUD, será necessário adicionar diretivas AngularJS no html
  - 5) Diretiva ng-repeat

```
<tbody>
  <tr ng-repeat="clienteTabela in clientes | filter:criterio"
      | ng-click="seleciona(clienteTabela)">
    <td>{{clienteTabela.codigo}}</td>
    <td>{{clienteTabela.nome}}</td>
    <td>{{clienteTabela.cargo}}</td>
    <td>{{clienteTabela.endereco}}</td>
    <td>{{clienteTabela.cidade}}</td>
    <td>{{clienteTabela.cep}}</td>
    <td>{{clienteTabela.pais}}</td>
    <td>{{clienteTabela.telefone}}</td>
    <td>{{clienteTabela.fax}}</td>
  </tr>
</tbody>
```

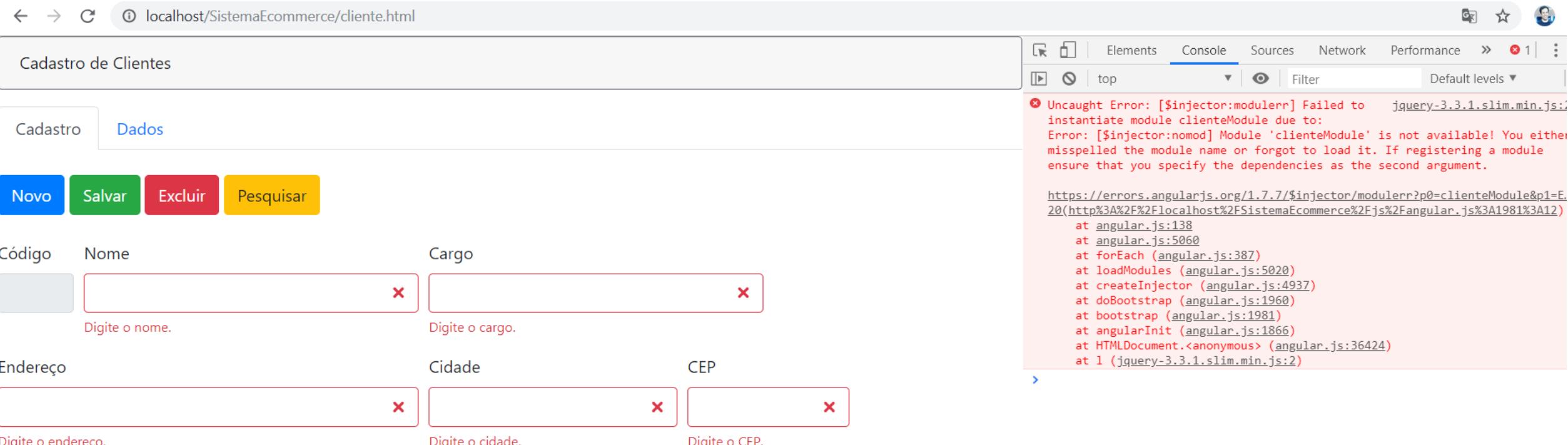
# Estudo de Caso: AngularJS

- Para que o DOM seja modificado facilmente com esta tela CRUD, será necessário adicionar diretivas AngularJS no html
  - 6) Diretiva ng-show

```
    </table>
    <span ng-show="(clientes | filter: criterio).length == 0">Infelizmente
    |   n&atilde;p temos o que voc&ecirc; est&aacute; procurando</span>
</div>
```

# Estudo de Caso: AngularJS

**Ao executar o sistema, o menu desaparece. O comando F12 mostra que o módulo não foi instanciado.**

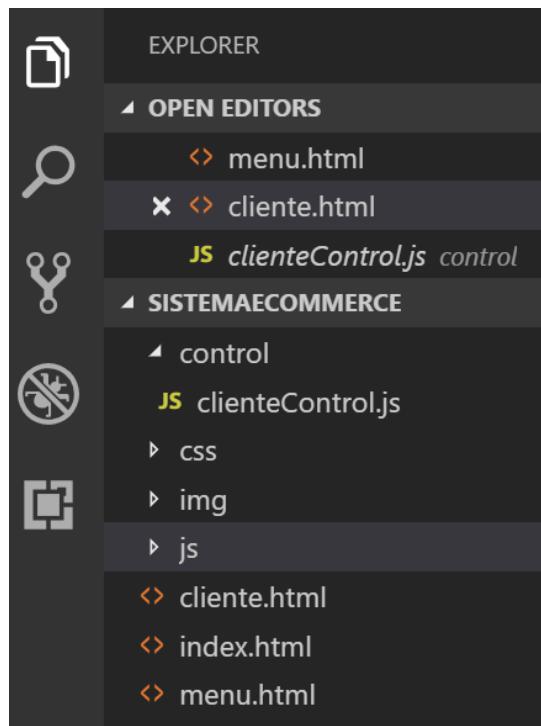


The screenshot shows a web browser window with the URL `localhost/SistemaEcommerce/cliente.html`. The page title is "Cadastro de Clientes". Below it, there are tabs for "Cadastro" and "Dados", with "Dados" being active. A row of buttons includes "Novo" (blue), "Salvar" (green), "Excluir" (red), and "Pesquisar" (yellow). The main area contains fields for "Código" (gray placeholder), "Nome" (red placeholder with error message "Digite o nome."), "Cargo" (red placeholder with error message "Digite o cargo."), "Endereço" (red placeholder with error message "Digite o endereço."), "Cidade" (red placeholder with error message "Digite a cidade."), and "CEP" (red placeholder with error message "Digite o CEP."). The developer tools' "Console" tab is selected, displaying the following error message:

```
Uncaught Error: [$injector:modulerr] Failed to instantiate module clienteModule due to:  
Error: [$injector:nomod] Module 'clienteModule' is not available! You either misspelled the module name or forgot to load it. If registering a module ensure that you specify the dependencies as the second argument.  
https://errors.angularjs.org/1.7.7/\$injector/modulerr?p0=clienteModule&p1=E20\(%20http%3A%2F%2Flocalhost%2FSistemaEcommerce%2Fjs%2Fangular.js%3A1981%3A12\)  
at angular.js:138  
at angular.js:5060  
at forEach (angular.js:387)  
at loadModules (angular.js:5020)  
at createInjector (angular.js:4937)  
at doBootstrap (angular.js:1960)  
at bootstrap (angular.js:1981)  
at angularInit (angular.js:1866)  
at HTMLDocument.<anonymous> (angular.js:36424)  
at 1 (jquery-3.3.1.slim.min.js:2)
```

# Estudo de Caso: AngularJS

- Para construir os métodos e objetos demandados pelo view, será criado o arquivo clienteControl.js
  - 1) Crie a pasta control e o arquivo clienteControl.js dentro da mesma



Computador > Disco Local (C:) > xampp > htdocs > SistemaEcommerce > control		
	Nome	Data de modificaç...   Tipo
	clienteControl	01/04/2016 01:50 Arquivo JavaScript

# Estudo de Caso: AngularJS

- Para construir os métodos e objetos demandados pelo view, será criado o arquivo clienteControl.js
- 2) Adicione uma referência no arquivo cliente.html para o control

```
<body ng-controller="clienteControl">
  <script src="js/jquery-3.3.1.slim.min.js"></script>
  <script src="js/popper.min.js"></script>
  <script src="js/bootstrap.js"></script>
  <script src="js/angular.js"></script>
  <script src="control/clienteControl.js"></script>
  <div ng-include src="'menu.html'"></div>
```

# Estudo de Caso: AngularJS

- Para construir os métodos e objetos demandados pelo view, será criado o arquivo clienteControl.js

## 3) Crie o conteúdo:

```
1  var app = angular.module('clienteModule',[]);
2  app.controller('clienteControl',function($scope) {
3
4      $scope.clientes = [
5          {'codigo':'1',
6              'nome':'Carlos',
7              'cargo':'Professor',
8              'endereco':'Rua teste, 65, Jardim das Palmeiras',
9              'cidade':'Uberlandia',
10             'cep':'38400-000',
11             'pais':'Brasil',
12             'telefone':'34944423402',
13             'fax':'343434344'},
14          {'codigo':'2',
15              'nome':'Martin Fowler',
16              'cargo':'CEO',
17              'endereco':'40, street view, google',
18              'cidade':'Miami',
19              'cep':'30111',
20              'pais':'USA',
21              'telefone':'55100912333',
22              'fax':'232323'},
23      ]]
```

# Estudo de Caso: AngularJS

- Para construir os métodos e objetos demandados pelo view, será criado o arquivo clienteControl.js

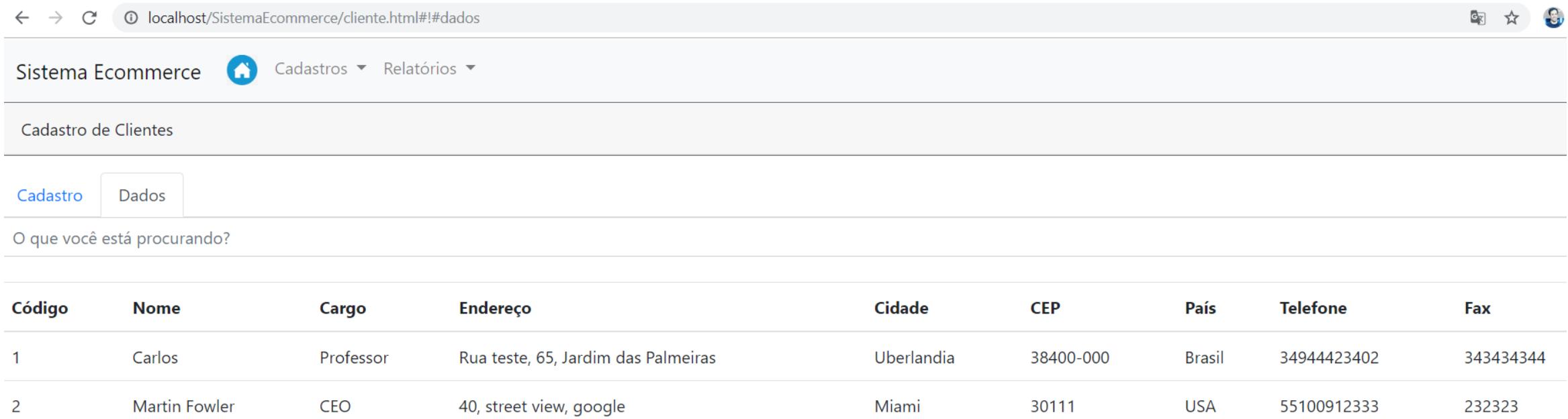
3) Crie o conteúdo:

```
1 var app = angular.module('clienteModule',[]);
2 app.controller('clienteControl',function($scope) {
3
4     $scope.clientes = [
5         {'codigo':'1',
6             'nome':'Carlos',
7             'cargo':'Professor',
8             'endereco':'Rua teste, 65, Jardim das Palmeiras',
9             'cidade':'Uberlandia',
10            'cep':'38400-000',
11            'pais':'Brasil',
12            'telefone':'34944423402',
13            'fax':'343434344'},
14         {'codigo':'2',
15             'nome':'Martin Fowler',
16             'cargo':'CEO',
17             'endereco':'40, street view, google',
18             'cidade':'Miami',
19             'cep':'30111',
20             'pais':'USA',
21             'telefone':'55100912333',
22             'fax':'232323'},
23     ]
```

# Estudo de Caso: AngularJS

- Para construir os métodos e objetos demandados pelo view, será criado o arquivo clienteControl.js

## Resultado:



The screenshot shows a web application interface for managing clients. At the top, there's a header with a back button, forward button, refresh button, a search bar containing 'localhost/SistemaEcommerce/cliente.html#!#dados', and user icons for profile, star, and settings. Below the header, the main navigation bar includes 'Sistema Ecommerce' with a home icon, 'Cadastros' (selected), and 'Relatórios'. The current page title is 'Cadastro de Clientes'. A tab bar at the bottom left shows 'Cadastro' (selected) and 'Dados'. A search bar asks 'O que você está procurando?'. A table lists client data with columns: Código, Nome, Cargo, Endereço, Cidade, CEP, País, Telefone, and Fax. Two rows of data are visible.

Código	Nome	Cargo	Endereço	Cidade	CEP	País	Telefone	Fax
1	Carlos	Professor	Rua teste, 65, Jardim das Palmeiras	Uberlandia	38400-000	Brasil	34944423402	343434344
2	Martin Fowler	CEO	40, street view, google	Miami	30111	USA	55100912333	232323

# Estudo de Caso: *AngularJS*

- Para construir os métodos e objetos demandados pelo view, será criado o arquivo clienteControl.js

3) Crie o conteúdo abaixo (cont):

```
$scope.novo = function() {
    $scope.cliente = {};
}

$scope.salvar = function() {
    $scope.clientes.push($scope.cliente);
    $scope.novo();
}

$scope.excluir = function() {
    $scope.clientes.splice($scope.clientes.indexOf($scope.cliente),1);
    $scope.novo();
}

$scope.seleciona = function(cliente) {
    $scope.cliente = cliente;
}
```

# Estudo de Caso: AngularJS

- Com isso, as operações de CRUD funcionam localmente, como um protótipo. É o ideal para que usuários façam os testes funcionais necessários.

Sistema Ecommerce

Cadastro de Clientes

Cadastro Dados

[Novo](#) [Salvar](#) [Excluir](#) [Pesquisar](#)

Código	Nome	Cargo
1	Carlos	Professor

Endereço	Cidade	CEP
Rua teste, 65, Jardim das Palmeiras	Uberlandia	38400-000

País	Telefone	Fax
Brasil	34944423402	343434344

# Estudo de Caso: AngularJS

- Observações:
  - Foi criada uma lista estática de clientes em formato JSON para testar a tabela e os campos do CRUD;
  - Todos os métodos do control estão simplesmente manipulando uma lista estática em memória. A grande diferença é que **esta abordagem é transparente ao HTML**, logo quando for modificado para requisições REST, não haverá mudança no código html;
  - Com a propriedade *two-way-data-binding*, **qualquer objeto que seja alterado simboliza mudança automática no DOM**. Com isso, ao selecionar um elemento de uma tabela, assim que o objeto `$scope.cliente` recebe o objeto da tabela, os campos do tipo *input type* automaticamente serão atualizados com o valor do objeto. Da mesma forma que, modificando o valor nos campos *input type*, o objeto irá ter seu valor alterado.
  - O método *pesquisar()* ainda não possui conteúdo porque o objeto `$scope.criterio` está sendo usado para filtrar na lista de clientes. Com as requisições REST, a intenção é que este método pesquise no servidor.

# Estudo de Caso: *AngularJS*

- Recurso `$http`
  - Serviço que permite a comunicação com servidores HTTP remotos via objetos `XMLHttpRequest` ou `JsonP`
  - Existem outras opções com um nível mais alto de abstração, como o serviço `$resource`
  - Métodos
    - `$http.get`
    - `$http.head`
    - `$http.post`
    - `$http.put`
    - `$http.delete`
    - `$http.jsonp`
    - `$http.patch`
  - Maiores informações sobre este recurso: [https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)

# Estudo de Caso: AngularJS

## - Recurso \$http

- Para utilizar o recurso no control, basta adicionar uma referência à variável \$http

```
1 var app = angular.module('clienteModule',[]);
2 app.controller('clienteControl',function($scope,$http) {
3   |
```

- Será implementado o conteúdo abaixo:

```
var app = angular.module('clienteModule',[]);
app.controller('clienteControl',function($scope,$http) {

  var url = 'http://localhost:8080/clientes';

  $scope.pesquisar = function() {
    $http.get(url).then(function (response) {
      $scope.clientes = response.data;
    }, function (error) {
      alert(error);
      console.log(error);
    });
  }

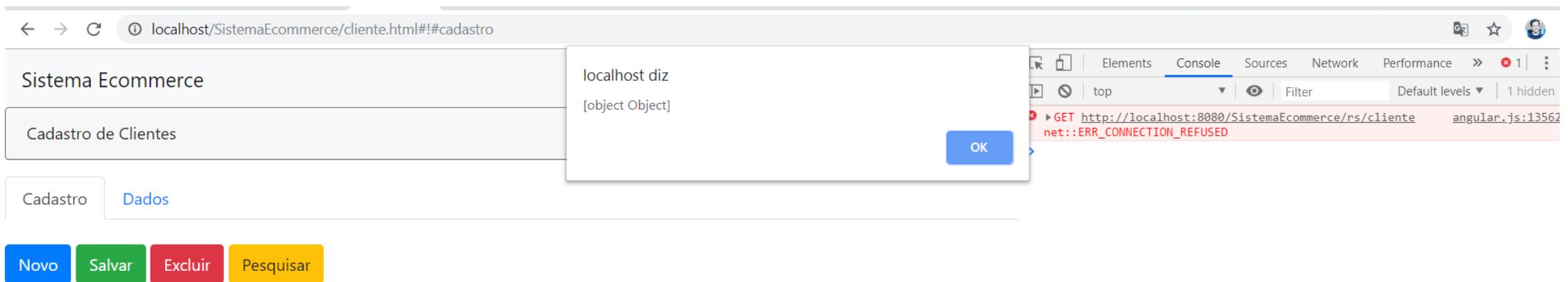
  $scope.pesquisar();
  /*$scope.clientes = [
    |           {'codigo': '1'}
```

# Estudo de Caso: *AngularJS*

- Recurso `$http`
  - A variável `url` foi criada porque os métodos `get`, `post` e `put` compartilharão da mesma url;
  - O método `$http.get` é executado, e em caso de sucesso, é retornada uma lista de clientes do servidor em formato JSON. A lista que antes era inicializada estaticamente, receberá essa lista. Os códigos de sucesso são de 200 a 299;
  - Em caso de erro, um alert é impresso na tela
  - Após a definição do método, este será executado ao carregar o script.

# Estudo de Caso: AngularJS

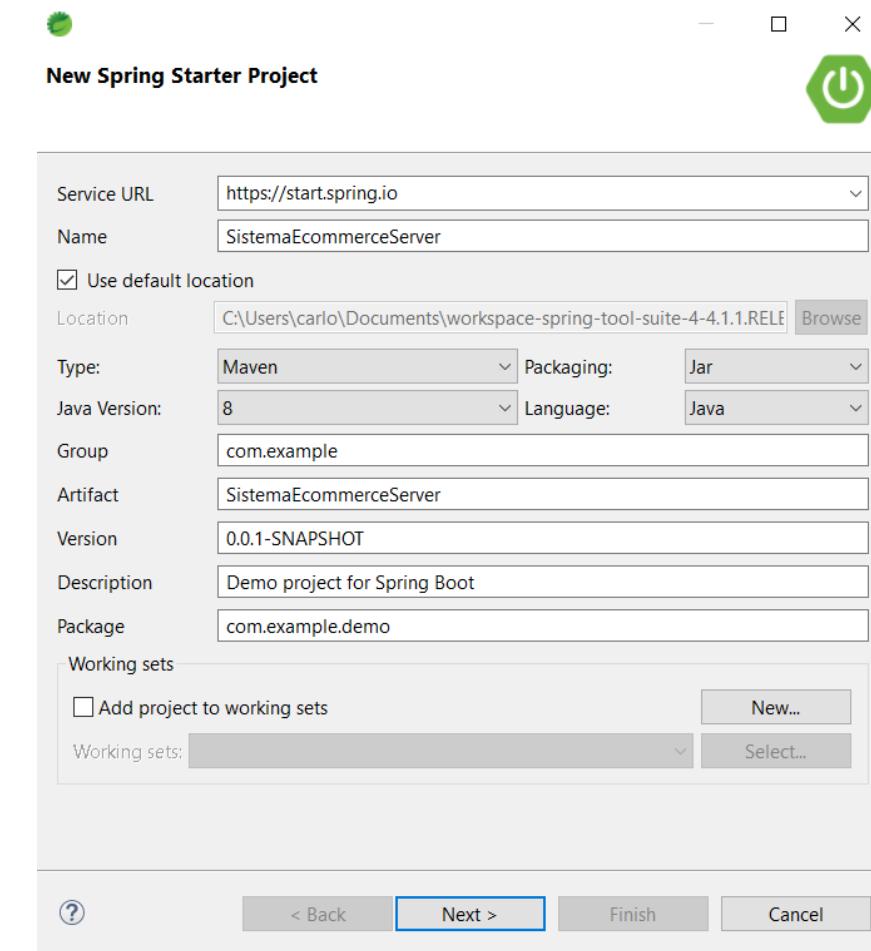
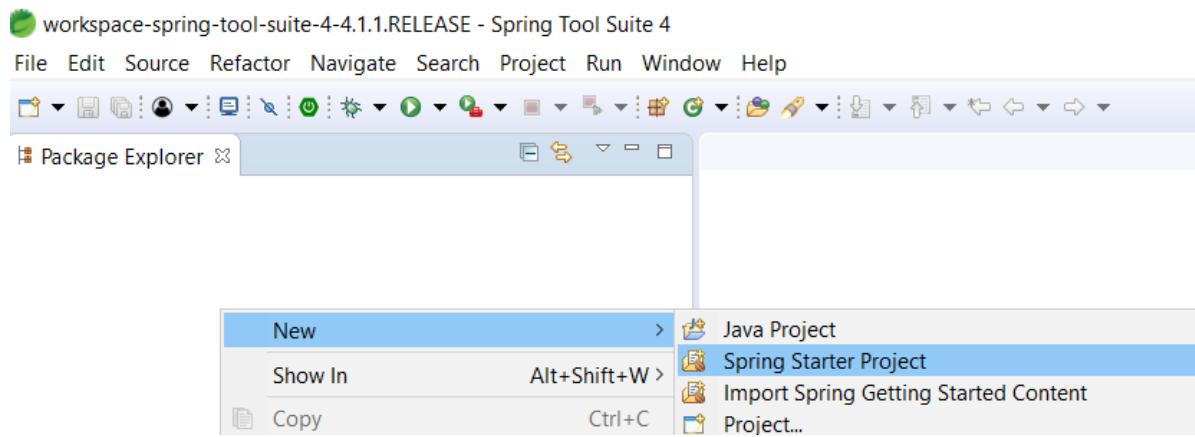
- Obviamente irá retornar erro, já que o serviço citado não existe



The screenshot shows a web browser window for 'Sistema Ecommerce'. The URL in the address bar is 'localhost/SistemaEcommerce/cliente.html#!#cadastro'. The page displays a form for 'Cadastro de Clientes' with tabs for 'Cadastro' and 'Dados'. Below the tabs are buttons for 'Novo' (blue), 'Salvar' (green), 'Excluir' (red), and 'Pesquisar' (yellow). A modal dialog box is open, displaying the message 'localhost diz [object Object]' with an 'OK' button. To the right of the browser window is the Chrome DevTools 'Console' tab, which shows a red error message: 'GET http://localhost:8080/SistemaEcommerce/rs/cliente angular.js:13562 net::ERR\_CONNECTION\_REFUSED'.

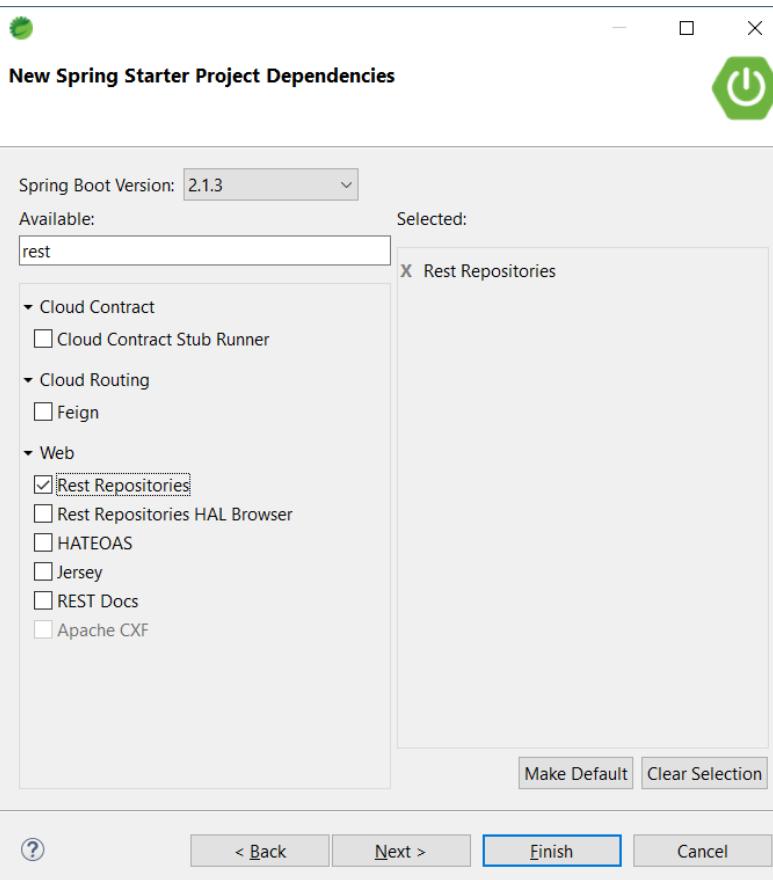
# Estudo de Caso: AngularJS

- Criar um novo Spring Starter Project



# Estudo de Caso: AngularJS

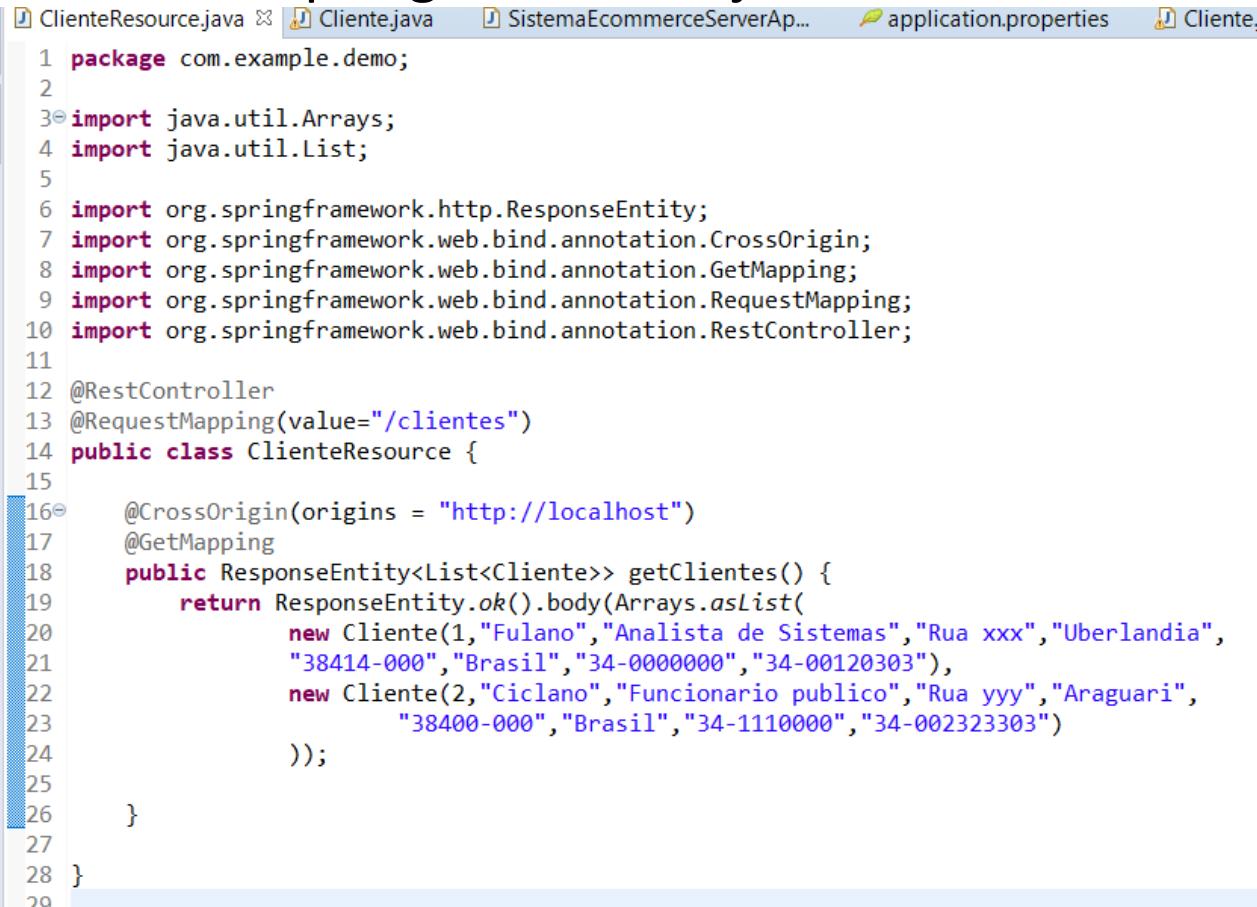
- Criar um novo Spring Starter Project



```
4 public class Cliente implements Serializable {
5
6     private int codigo;
7     private String nome;
8     private String cargo;
9     private String endereco;
10    private String cidade;
11    private String cep;
12    private String pais;
13    private String telefone;
14    private String fax;
15
16
17    public Cliente(int codigo, String nome, String cargo, String endereco, String cidade, String cep, String pais,
18                  String telefone, String fax) {
19        super();
20        this.codigo = codigo;
21        this.nome = nome;
22        this.cargo = cargo;
23        this.endereco = endereco;
24        this.cidade = cidade;
25        this.cep = cep;
26        this.pais = pais;
27        this.telefone = telefone;
28        this.fax = fax;
29    }
30
31    //getset
32}
```

# Estudo de Caso: AngularJS

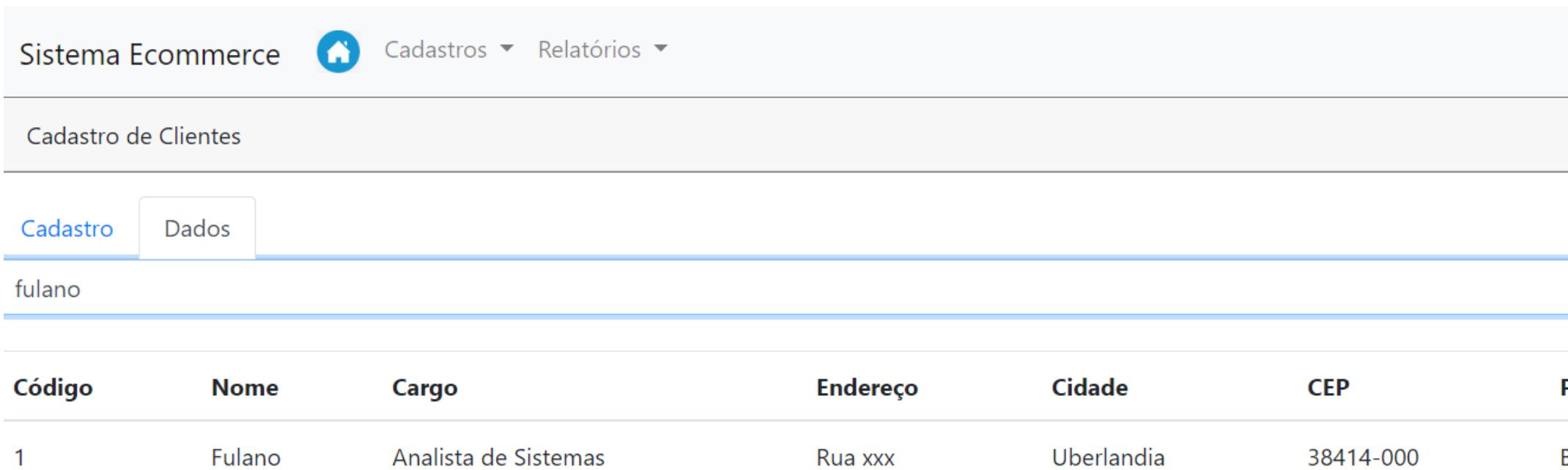
- Criar um novo Spring Starter Project



```
1 package com.example.demo;
2
3 import java.util.Arrays;
4 import java.util.List;
5
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.CrossOrigin;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RestController;
11
12 @RestController
13 @RequestMapping(value="/clientes")
14 public class ClienteResource {
15
16     @CrossOrigin(origins = "http://localhost")
17     @GetMapping
18     public ResponseEntity<List<Cliente>> getClientes() {
19         return ResponseEntity.ok().body(Arrays.asList(
20             new Cliente(1,"Fulano","Analista de Sistemas","Rua xxx","Uberlandia",
21             "38414-000","Brasil","34-00000000","34-00120303"),
22             new Cliente(2,"Ciclano","Funcionario publico","Rua yyy","Araguari",
23             "38400-000","Brasil","34-11100000","34-002323303")
24         ));
25
26     }
27
28 }
29 }
```

# Estudo de Caso: AngularJS

- Recurso \$http
  - O CORS foi criado porque a requisição está buscando um recurso de um domínio diferente. Por padrão, uma aplicação web só pode fazer requisições para seu próprio domínio.



The screenshot shows a web application interface for a "Sistema Ecommerce". At the top, there is a navigation bar with the title "Sistema Ecommerce" and icons for home, cadastros, and relatórios. Below the navigation, a header says "Cadastro de Clientes". A tab menu at the top left has "Cadastro" and "Dados"; "Cadastro" is active and highlighted with a blue border. The main content area displays a single client record with the name "fulano". Below this, a table lists multiple client records with columns for Código, Nome, Cargo, Endereço, Cidade, CEP, and P. The first row of the table is fully visible, showing values: Código 1, Nome Fulano, Cargo Analista de Sistemas, Endereço Rua xxx, Cidade Uberlandia, CEP 38414-000, and P B.

Código	Nome	Cargo	Endereço	Cidade	CEP	P
1	Fulano	Analista de Sistemas	Rua xxx	Uberlandia	38414-000	B

# Estudo de Caso: AngularJS

## - Recurso \$http

```
$scope.salvar = function() {
    if (typeof $scope.cliente.codigo == 'undefined') {
        $http.post(url,$scope.cliente).then(function (response) {
            $scope.clientes.push(response.data);
            $scope.novo();
        }, function (error) {
            alert(error);
            console.log(error);
        });
    } else [
        $http.put(url,$scope.cliente).then(function () {
            $scope.pesquisar();
            $scope.novo();
        }, function (error) {
            alert(error);
            console.log(error);
        });
    ]
}
```

# Estudo de Caso: AngularJS

- Recurso \$http

```
$scope.excluir = function() {
    if ($scope.cliente.codigo == '') {
        alert('Escolha um cliente');
    } else {
        urlExcluir = url+ "/" +$scope.cliente.codigo;
        $http.delete(urlExcluir).then(function () {
            $scope.pesquisar();
            $scope.novo();
        }, function (error) {
            alert(error);
            console.log(error);
        });
    }
}
```

# Estudo de Caso: AngularJS

- Restante do back-end

```
1 package br.edu.iftm.atividadeComplementar.domains;
2
3 import java.io.Serializable;
4
5 import javax.persistence.Entity;
6 import javax.persistence.GeneratedValue;
7 import javax.persistence.GenerationType;
8 import javax.persistence.Id;
9
10 @Entity
11 public class Cliente implements Serializable {
12
13     @Id
14     @GeneratedValue(strategy=GenerationType.IDENTITY)
15     private Integer codigo;
16     private String nome;
17     private String cargo;
18     private String endereco;
19     private String cidade;
20     private String cep;
21     private String pais;
22     private String telefone;
23     private String fax;
```

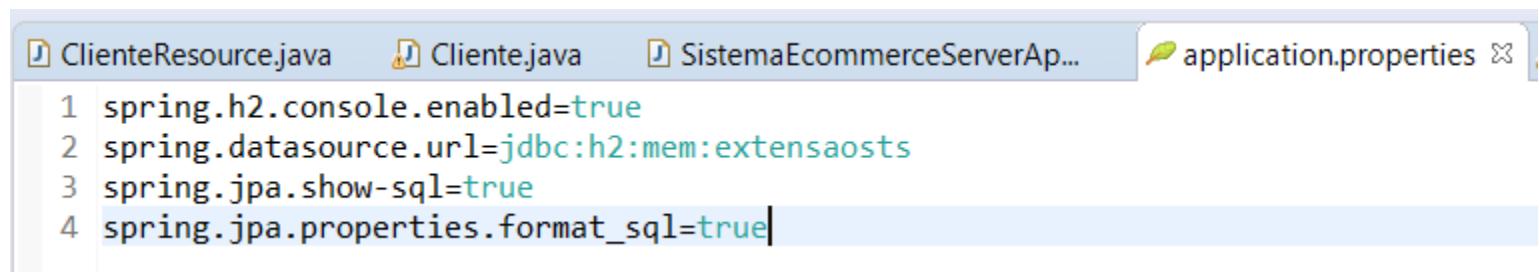
# Estudo de Caso: AngularJS

## - Restante do back-end

```
~.  
25 @RestController  
26 @RequestMapping(value="/clientes")  
27 @CrossOrigin(origins = "http://localhost")  
28 public class ClienteResource {  
29  
30     @Autowired  
31     private ClienteRepository service;  
32  
33     @GetMapping  
34     public ResponseEntity<List<Cliente>> findAll() {  
35         List<Cliente> alunos = service.findAll();  
36         return ResponseEntity.ok().body(alunos);  
37     }  
38  
39     @PostMapping  
40     public ResponseEntity<?> salvar(@Valid @RequestBody Cliente cliente) {  
41         service.save(cliente);  
42         URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}")  
43             .buildAndExpand(cliente.getCodigo()).toUri();  
44         return ResponseEntity.created(location).build();  
45     }  
46  
47     @PutMapping  
48     public ResponseEntity<?> atualizar(@Valid @RequestBody Cliente cliente) {  
49         service.save(cliente);  
50         return ResponseEntity.noContent().build();  
51     }  
52  
53     @DeleteMapping(value="{codigo}")  
54     public ResponseEntity<?> excluir(@PathVariable Integer codigo) {  
55         try {  
56             service.deleteById(codigo);  
57             return ResponseEntity.ok(codigo);  
58         } catch(EmptyResultDataAccessException e) {}  
59             return ResponseEntity.notFound().build();  
60         }  
61     }
```

# Estudo de Caso: AngularJS

- Restante do back-end



```
1 spring.h2.console.enabled=true
2 spring.datasource.url=jdbc:h2:mem:extensaosts
3 spring.jpa.show-sql=true
4 spring.jpa.properties.format_sql=true
```

# Estudo de Caso: AngularJS

- Restante do back-end

```
1 package br.edu.iftm.atividadeComplementar.repositories;  
2  
3 import org.springframework.data.jpa.repository.JpaRepository;  
4 import org.springframework.stereotype.Repository;  
5  
6 import br.edu.iftm.atividadeComplementar.domains.Cliente;  
7  
8 @Repository  
9 public interface ClienteRepository extends JpaRepository<Cliente, Integer>{  
10  
11  
12 }  
13
```

# Estudo de Caso: AngularJS

## - Restante do back-end

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-rest</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
```

# LINKS DOS SOFTWARES UTILIZADOS

---

- **Jquery** - <https://code.jquery.com/jquery-3.3.1.min.js>
- **Bootstrap** - <https://github.com/twbs/bootstrap/archive/v4.3.1.zip>
- **AngularJS** - <https://code.angularjs.org/1.7.7/angular-1.7.7.zip>
- PopperJS -  
<https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js>

# REFERÊNCIAS

- GRIGORIK, I. High Performance Browser Networking. O'Reilly, 2013. Disponível em <http://chimera.labs.oreilly.com/books/1230000000545>. Acesso em 05/08/2015.
- MICROSOFT. Single-Page Applications: Build Modern, Responsive Web Apps with ASP.NET. Disponível em <https://msdn.microsoft.com/en-us/magazine/dn463786.aspx>. Acesso em 05/08/2015.
- Front End Brasil. A história do HTML. Disponível em <http://www.frontendbrasil.com.br/artigos/a-historia-do-html/>. Acesso em 05/08/2015.
- Modern Web. 8 bootstrap alternatives. Disponível em <http://modernweb.com/2014/02/17/8-bootstrap-alternatives/>. Acesso em 05/08/2015.
- TutorialZine. 50 plugin for extending Twitter Bootstrap. Disponível em <http://tutorialzine.com/2013/07/50-must-have-plugins-for-extending-twitter-bootstrap/>. Acesso em 07/08/2015.