



**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

# Estudo de Caso: *Angular*

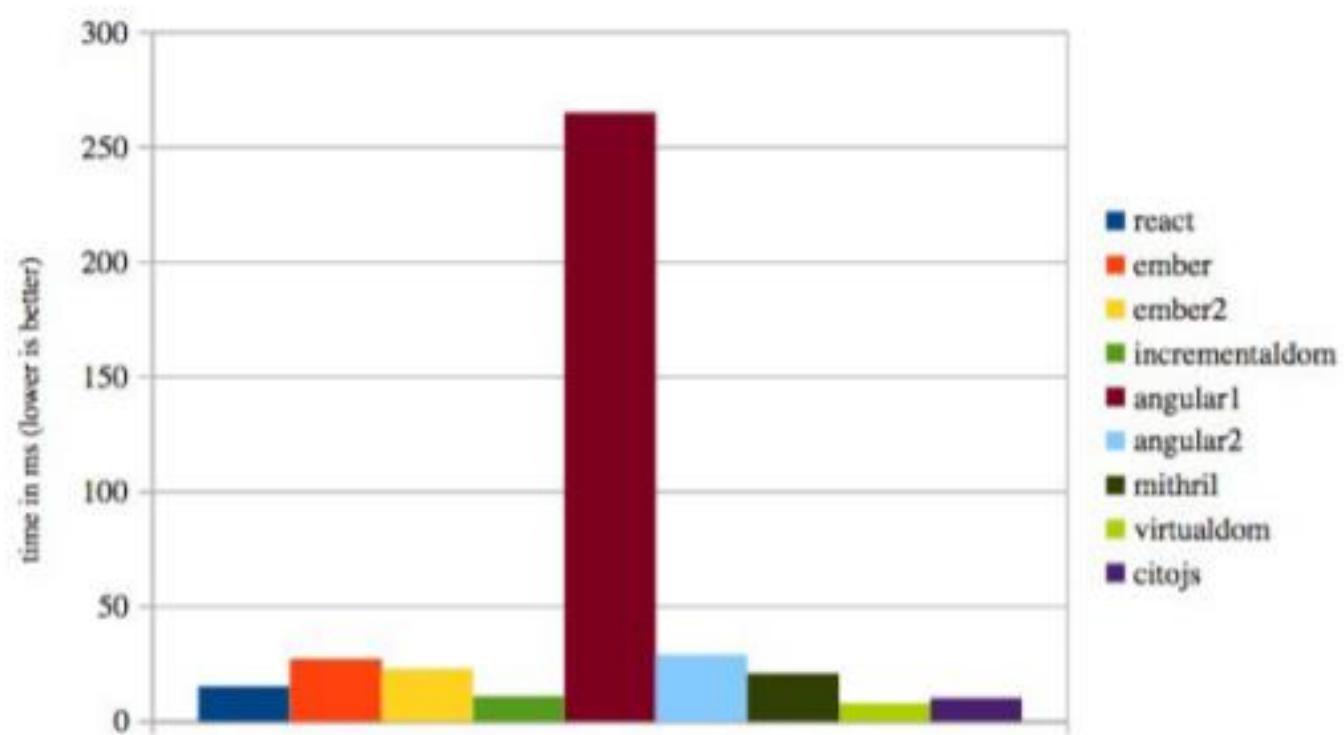
## - Definições

- 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**
  - AngularJS faz mais alterações no DOM



# Estudo de Caso: *Angular*

## - Definições

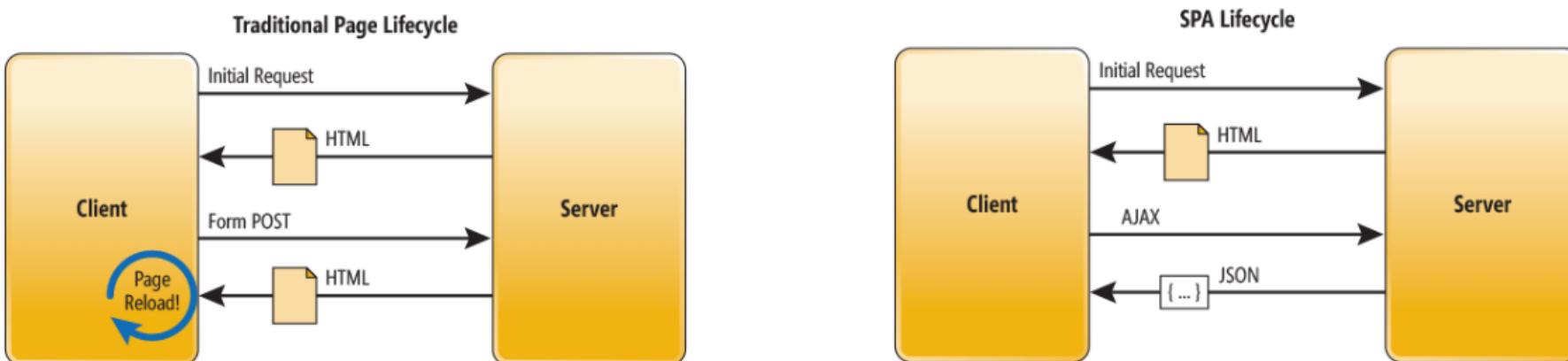
- O desenvolvimento em Angular é feito por meio de codificação TypeScript.
- Implementa funcionalidades do ES6
  - Tipagem de variáveis
  - Sintaxe clara e fácil de entender, parecendo-se com C# e Java.
- Javascript é somente um dialeto/apelido para a linguagem ECMAScript (ES)
- Em 2016 o ES chegou na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

# Estudo de Caso: *Angular*

## - Definições

- SPA – Single Page Applications

- Aplicações Javascript que rodam no lado cliente (browser) –
- Aplicação armazenada no lado cliente em forma de templates (pedaços de HTML)
- Só fazem requisições no servidor para buscar dados enviados via JSON
- A página não tem refresh, todo o HTML é trabalhado em mostrar/esconder o conteúdo.
- PERFORMANCE – tráfego entre server e client mais leve
- EXPERIÊNCIA DO USUÁRIO – soa como uma aplicação desktop



# Estudo de Caso: Angular

## - Características

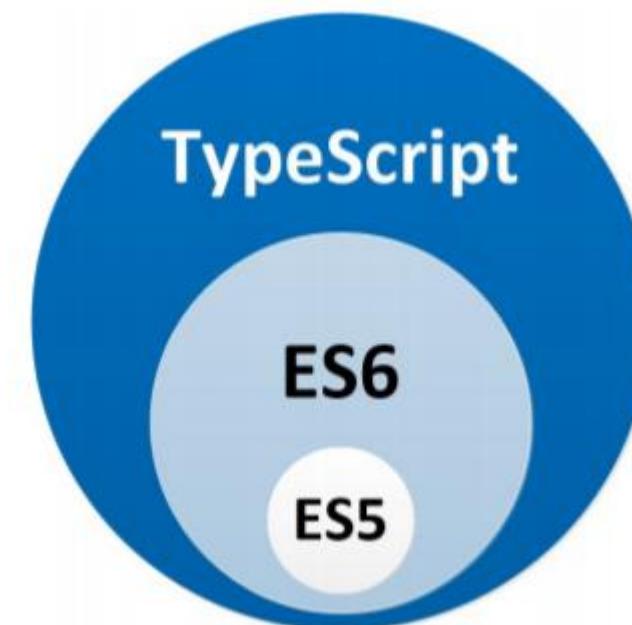
- Angular **força uma separação** entre o *front-end* e o *back-end*. Com isso, empresas podem ter equipes especializadas em cada parte da aplicação. Isso ajuda a eliminar situações onde a mesma equipe cria as duas camadas, especialmente quando se usa *frameworks* como *ASP.NET* e *JSF*.
- Em um projeto, 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*

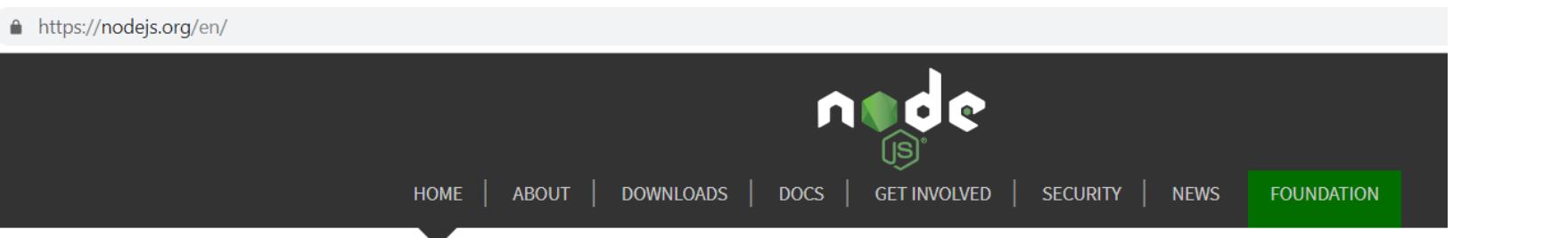
## - Definições

- TypeScript permite escrever o código na forma como se codifica no paradigma de Orientação a Objetos.
- TypeScript é um superset, um escudo para desenvolver em JavaScript colocando tipos.



# 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

```
PS C:\xampp\htdocs\SistemaEcommerceCLI> node -v
v10.15.1
PS C:\xampp\htdocs\SistemaEcommerceCLI>
```

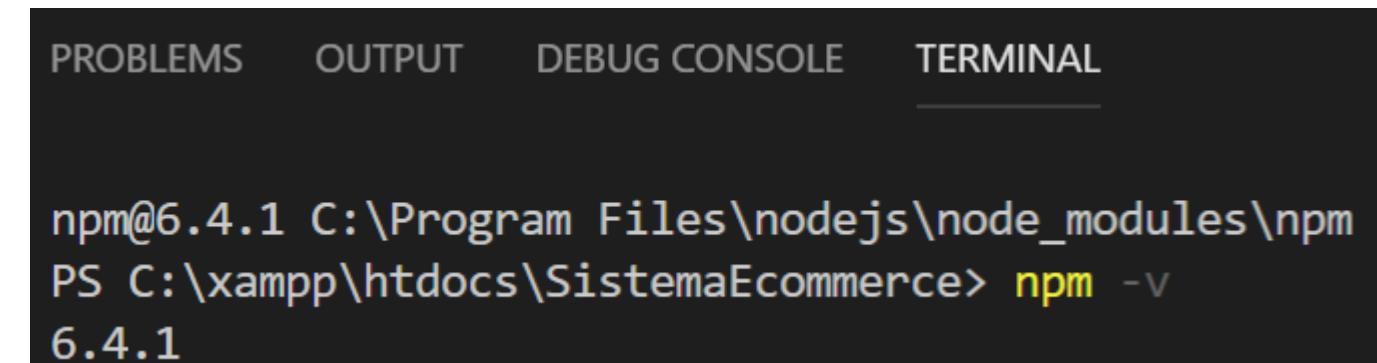
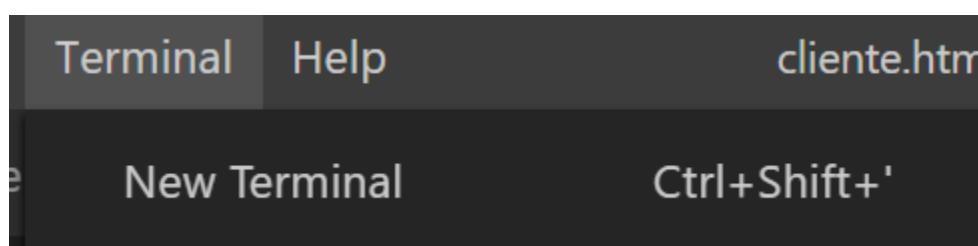
# Estudo de Caso: *Angular*

- **Instalação do ambiente de desenvolvimento**
- Visual Studio Code
  - Suporte a TypeScript
  - Autocomplete informativo
  - Integrado ao prompt de comando
- Alternativas: Sublime, Brackets, WebStorm, Atom.
- Download: <https://code.visualstudio.com/>



# Estudo de Caso: Angular

- **NPM**
- Node Package Manager (gerenciador de pacotes do node)
  - Usado para interagir por linha de comando no console do computador, para fazer a instalação de pacotes e gerenciamento de dependências do projeto.
  - Já vem instalado junto com o Node.

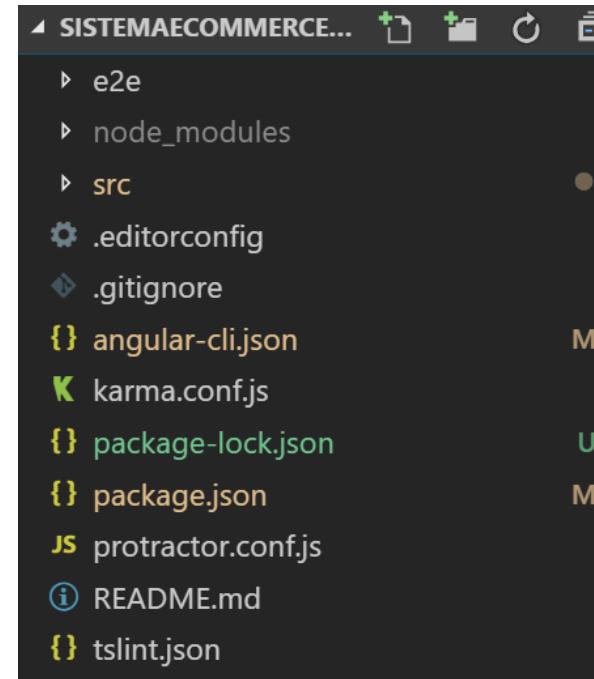


A screenshot of a terminal window showing the output of the 'npm -v' command. The output is as follows:

```
npm@6.4.1 C:\Program Files\nodejs\node_modules\npm
PS C:\xampp\htdocs\SistemaEcommerce> npm -v
6.4.1
```

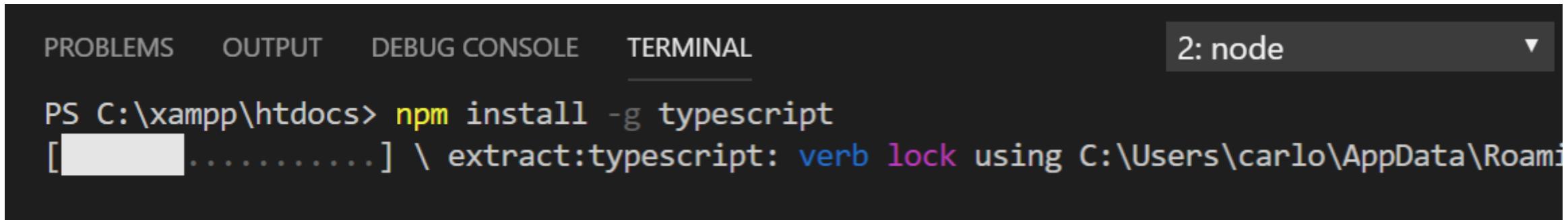
# Estudo de Caso: Angular

- **NPM**
- O NPM irá gerenciar as dependências dos projetos Angular através de um arquivo chamado **package.json**, que declara as dependências do projeto, e uma pasta chamada **node\_module** que conterá os pacotes instalados.



# Estudo de Caso: *Angular*

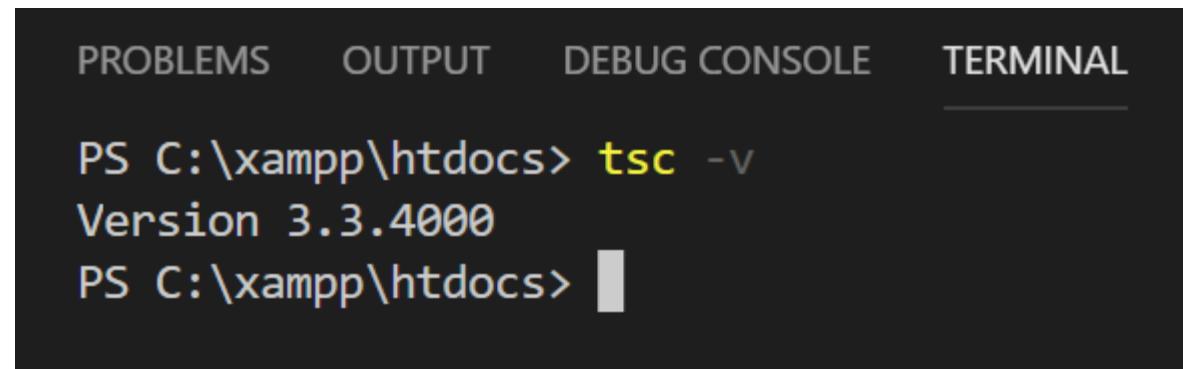
- **TypeScript**
- Para instalar, deve-se digitar o comando abaixo.
  - O parâmetro `-g` significa instalar o typescript de maneira global no PC



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

2: node ▾

```
PS C:\xampp\htdocs> npm install -g typescript
[.....] \ extract:typescript: verb lock using C:\Users\carlo\AppData\Roaming\npm\ts...
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

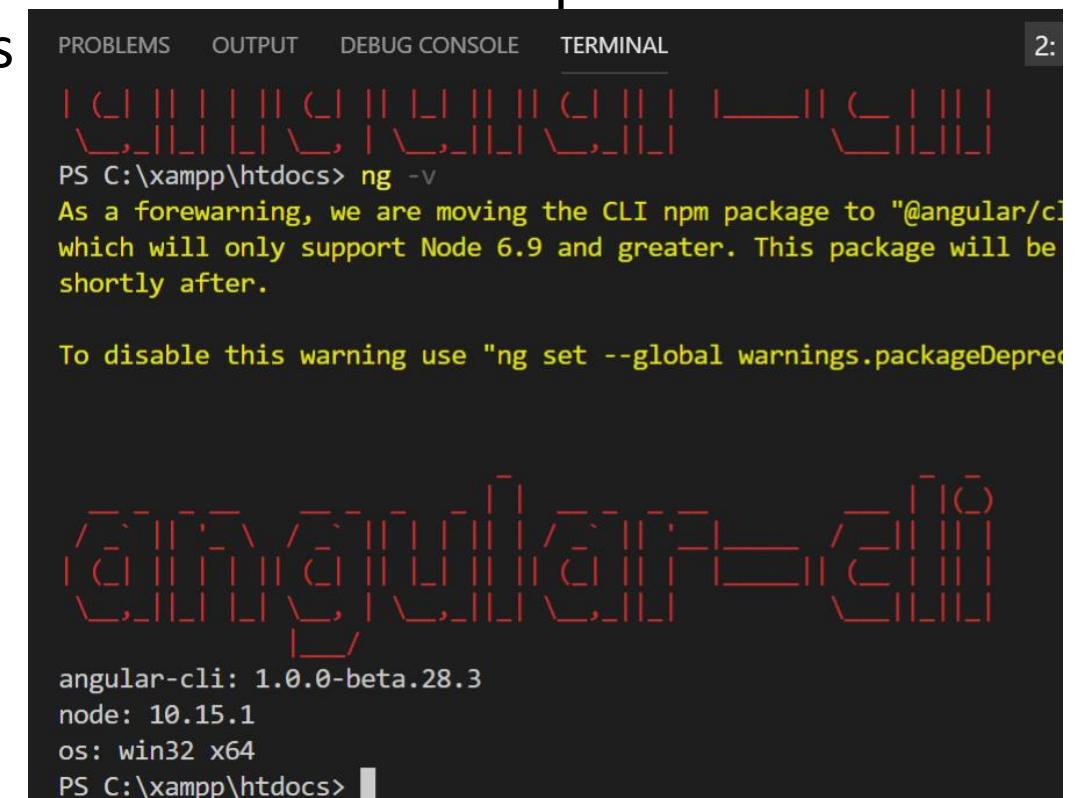
```
PS C:\xampp\htdocs> tsc -v
Version 3.3.4000
PS C:\xampp\htdocs>
```

# Estudo de Caso: Angular

## - Angular CLI

- O Angular Cli automatiza o processo de declarar e criar as dependências do projeto, efetuar os downloads dos pacotes

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\xampp\htdocs> npm install -g angular-cli
npm WARN deprecated angular-cli@1.0.0-beta.28.3: angular-cli
te your dependencies.
[.....] - fetchMetadata: sill removeObsoleteDep
```



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2:
PS C:\xampp\htdocs> ng -v
As a forewarning, we are moving the CLI npm package to "@angular/cli"
which will only support Node 6.9 and greater. This package will be
shortly after.

To disable this warning use "ng set --global warnings.packageDeprecationWarning=false"

angular-cli: 1.0.0-beta.28.3
node: 10.15.1
os: win32 x64
PS C:\xampp\htdocs>

# Estudo de Caso: Angular

## - Angular CLI

- Bootstrap

```
PS C:\projetos\SistemaEcommerceCLI> npm install -g bootstrap
npm WARN bootstrap@4.3.1 requires a peer of jquery@1.9.1 - 3 but no
```

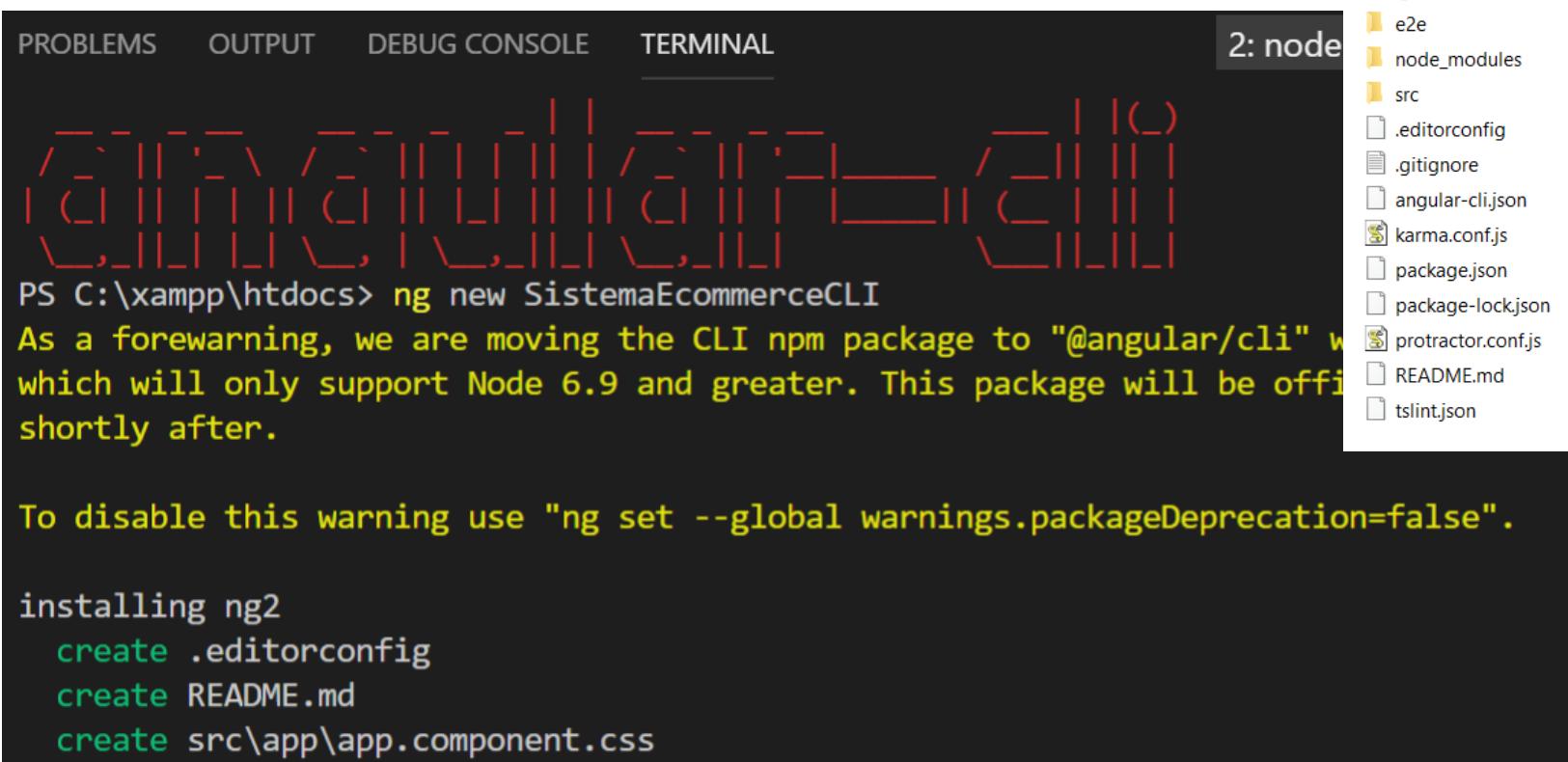
```
PS C:\projetos\SistemaEcommerceCLI> npm install -g jquery
+ jquery@3.3.1
added 1 package from 1 contributor in 0.845s
```

```
PS C:\projetos\SistemaEcommerceCLI> npm install -g popper.js
+ popper.js@1.14.7
added 1 package from 2 contributors in 0.783s
```

# Estudo de Caso: Angular

## - Angular CLI

- Novo projeto.



```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
                                                2: node
PS C:\xampp\htdocs> ng new SistemaEcommerceCLI
As a forewarning, we are moving the CLI npm package to "@angular/cli" w
which will only support Node 6.9 and greater. This package will be offi
shortly after.

To disable this warning use "ng set --global warnings.packageDeprecation=false".

installing ng2
create .editorconfig
create README.md
create src\app\app.component.css
  
```

Computador > Disco Local (C:) > xampp > htdocs > SistemaEcommerceCLI >			
Nome	Data de modificaç...	Tipo	Tamanho
.git	21/03/2019 10:42	Pasta de arquivos	
e2e	21/03/2019 10:42	Pasta de arquivos	
node_modules	21/03/2019 10:43	Pasta de arquivos	
src	21/03/2019 10:42	Pasta de arquivos	
.editorconfig	21/03/2019 10:42	Arquivo EDITORC...	1 KB
.gitignore	21/03/2019 10:42	Documento de Te...	1 KB
angular-cli.json	21/03/2019 10:42	Arquivo JSON	2 KB
karma.conf.js	21/03/2019 10:42	Arquivo JavaScript	2 KB
package.json	21/03/2019 10:42	Arquivo JSON	2 KB
package-lock.json	21/03/2019 10:43	Arquivo JSON	372 KB
protractor.conf.js	21/03/2019 10:42	Arquivo JavaScript	1 KB
README.md	21/03/2019 10:42	Arquivo MD	2 KB
tslint.json	21/03/2019 10:42	Arquivo JSON	3 KB

# Estudo de Caso: Angular

## - Angular CLI

Tipo	Comando
Componente	<i>ng g c meu-nome</i>
Serviços	<i>ng g s meu-nome</i>
Classe	<i>ng g cl meu-nome</i>
Interface	<i>ng g i meu-nome</i>

# Estudo de Caso: *Angular*

## - Angular CLI

- **Pasta e2e** – arquivos para teste de integração do projeto com Protractor. Configurações para os testes do projeto – verificar a integração e comunicação dos componentes da aplicação
- **Pasta node\_modules** – programas de dependência para o projeto funcionar. Tudo o que estiver declarado no **arquivo package.json** será baixado e armazenado neste diretório
- **Pasta src** – Código-fonte html, javascript e typescript
- **Arquivo angular-cli.json** – representação do projeto – caminhos, arquivos e configurações
- **Arquivo karma.conf.js** – configuração do Karma, biblioteca para testes unitários.

# Estudo de Caso: *Angular*

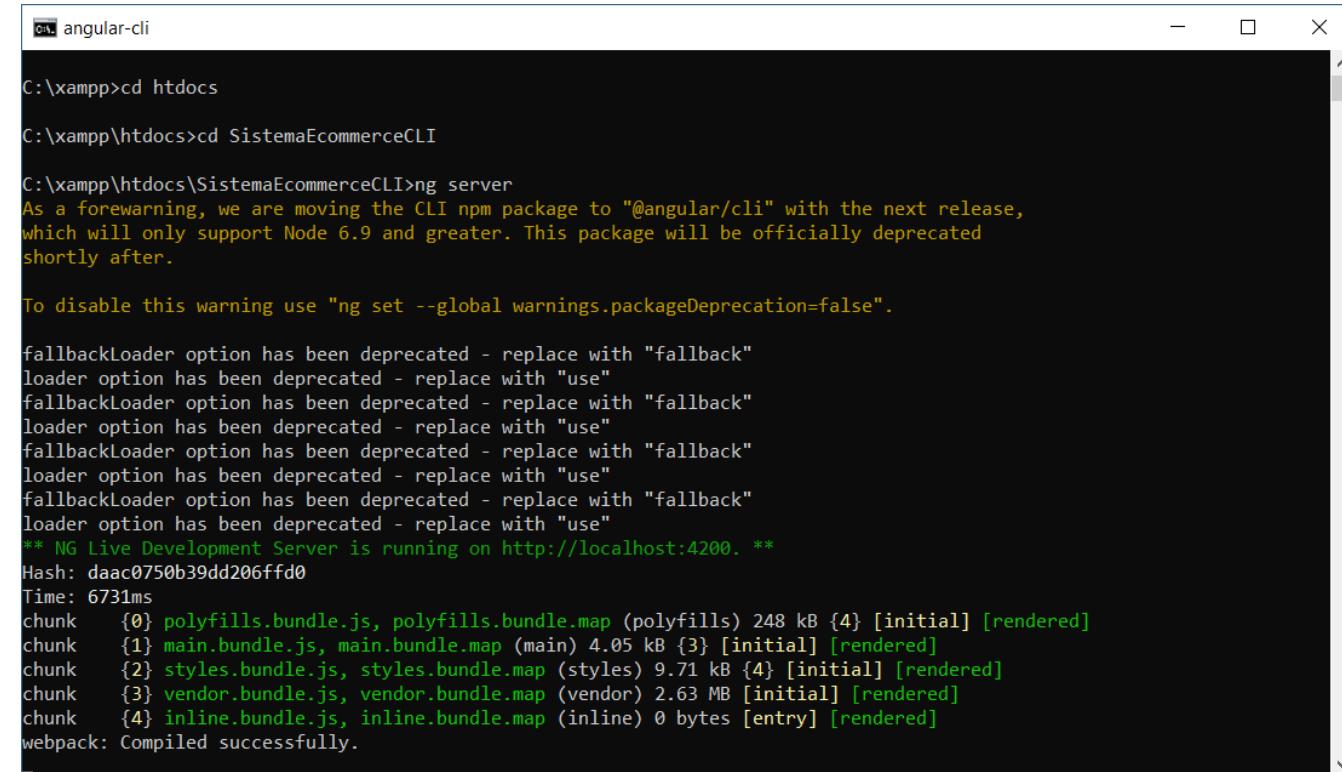
## - Angular CLI

- **Arquivo package.json** – dependências do projeto
  - Separado em 3 partes:
    - Primeira parte: dados do projeto, como versão, nome, scripts e teste
    - Dependencies: dependências do projeto para produção
    - devDependencies: dependências do projeto para desenvolvimento. Arquivos de testes estarão neste diretório.
  - **Arquivo protractor.conf.js** – arquivo de configuração da biblioteca Protractor, feito para testes de integração do projeto.
  - **Arquivo tslint.json** – configurações para usar o comando ng lint, que são boas práticas do desenvolvimento em Angular.

# Estudo de Caso: Angular

## - Angular CLI

- Iniciar o servidor, diretório dentro do projeto. O prompt é fora do Visual Studio Code, para evitar de travar o prompt. Pode-se criar um novo terminal no visual studio code



```
C:\xampp>cd htdocs
C:\xampp\htdocs>cd SistemaEcommerceCLI
C:\xampp\htdocs\SistemaEcommerceCLI>ng server
As a forewarning, we are moving the CLI npm package to "@angular/cli" with the next release,
which will only support Node 6.9 and greater. This package will be officially deprecated
shortly after.

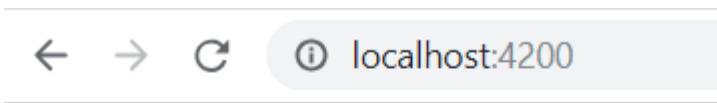
To disable this warning use "ng set --global warnings.packageDeprecation=false".

fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
fallbackLoader option has been deprecated - replace with "fallback"
loader option has been deprecated - replace with "use"
** NG Live Development Server is running on http://localhost:4200. **
Hash: daac0750b39dd206ffd0
Time: 6731ms
chunk {0} polyfills.bundle.js, polyfills.bundle.map (polyfills) 248 kB {4} [initial] [rendered]
chunk {1} main.bundle.js, main.bundle.map (main) 4.05 kB {3} [initial] [rendered]
chunk {2} styles.bundle.js, styles.bundle.map (styles) 9.71 kB {4} [initial] [rendered]
chunk {3} vendor.bundle.js, vendor.bundle.map (vendor) 2.63 MB [initial] [rendered]
chunk {4} inline.bundle.js, inline.bundle.map (inline) 0 bytes [entry] [rendered]
webpack: Compiled successfully.
```

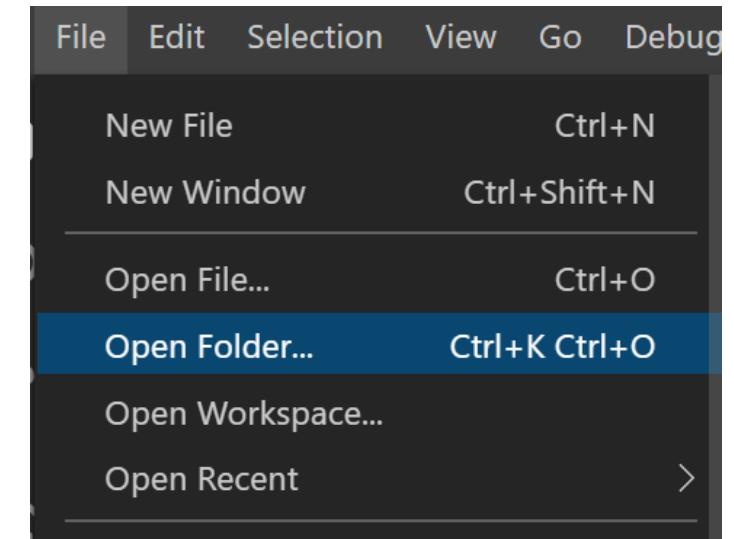
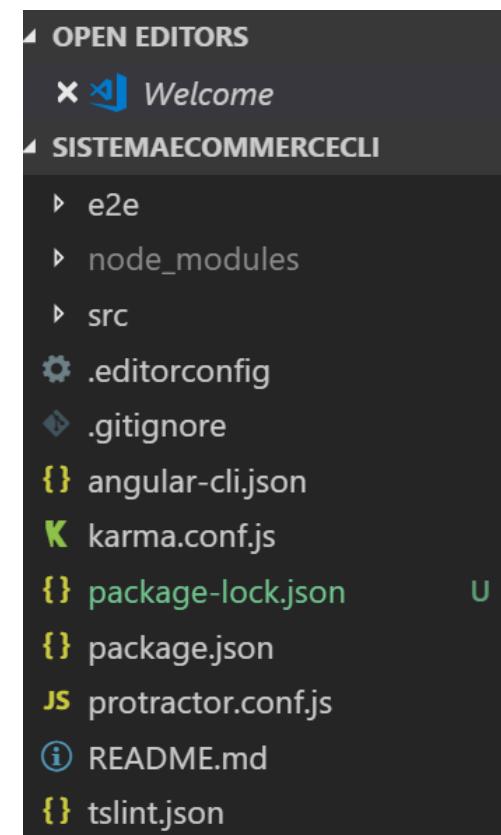
# Estudo de Caso: Angular

## - Angular CLI

- Abrir o projeto no Visual Studio Code



app works!



# Estudo de Caso: *Angular*

## - Angular CLI

- Pasta src
  - Pasta app – código-fonte da aplicação
  - Pasta environment – configuração do build
  - Favicon.ico – imagem que ficará na barra de navegação do browser
  - Index.html – página inicial, tags para criar as páginas web (não será editado)
  - main.ts – direção para inicialização do projeto no navegador
  - polyfills.ts – bibliotecas para fazer um de / para ES6 em ES5
  - styles.css – css global da aplicação

# Estudo de Caso: Angular

## - Angular CLI

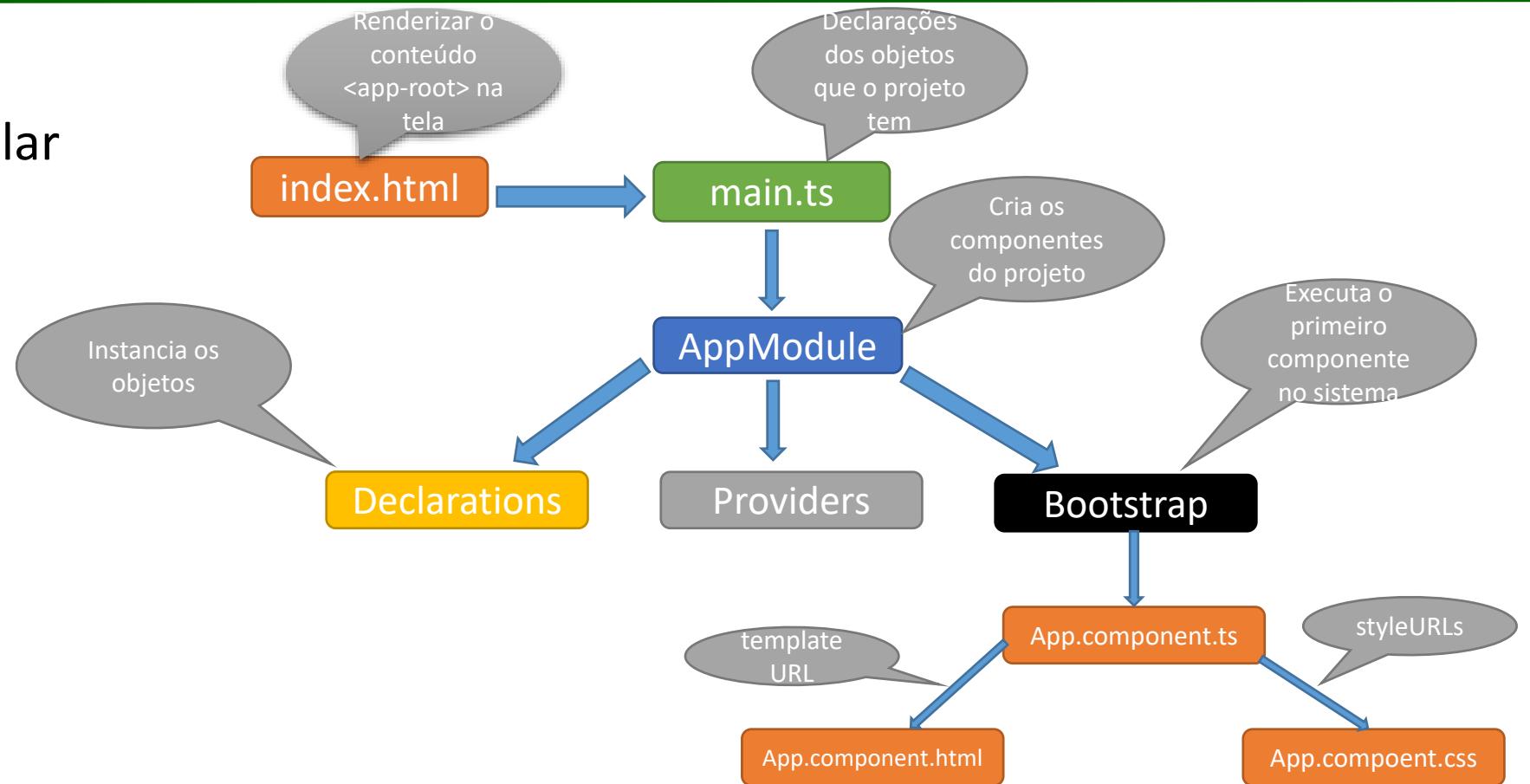
- Pasta src/app
  - Conterá todos os novos componentes, serviços, templates, configuração de CSS e arquivos de teste dos componentes.
  - Contém por padrão 4 arquivos de mesmo nome: app.component {css, html, .spec.ts, ts}
  - app.module.ts – arquivo de declaração dos objetos dentro do projeto.
    - Dentro de declarations é colocado o nome de cada novo objeto criado dentro do projeto.
    - Dentro de bootstrap, é colocado o nome do arquivo que será carregado primeiramente na aplicação.

### BOOTSTRAP DO ANGULAR 2

Não confunda o *bootstrap* do Angular 2 com o framework Bootstrap para CSS. O bootstrap do Angular 2 tem o sentido de **inicialização** de alguma coisa. Ele serve para dar o *starter*, o início para algo acontecer. Sempre que for falado *bootstrap* fora do ambiente de HTML, considere que o Angular 2 vai iniciar algum arquivo.

# Estudo de Caso: Angular

- Angular CLI
  - Build Angular

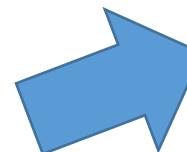


# Estudo de Caso: Angular

## - Angular CLI

- Template - visão html
- Componente – controlador view
- Metadata – configuração entre a classe e o template

```
TS app.component.ts ✘  ↗ app.component.html
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   title = 'app works!';
10 }
```



```
TS app.component.ts
1 <h1>
2   | {{title}}
3 </h1>
```

↗ app.component.html ✘



app works!

# Estudo de Caso: Angular

## - Angular CLI

- Componentes do Angular 2
  - Modularização – Angular 2 faz um sistema modularizado, onde a junção de pequenas partes formam uma página completa.
    - Cada componente é uma pequena parte do sistema, podendo ser utilizada em vários pontos
    - Facilidade de testes e manutenção
  - Data Binding – enviar e sincronizar dados entre o componente e o template
    - **Interpolação:** sentido componente para o template

```
<h1>
| {{title}}
</h1>
```

- **Property Binding** – sentido componente para o template, passando informações para alguma propriedade da tag do template `foto: string = 'favicon.ico';`  
`<img [src]="foto">`

# Estudo de Caso: Angular

## - Angular CLI

### - Data Binding

- **Event Binding** – passar dados do template para a classe do componente

```
<button (click)="msgAlerta()">Enviar Alerta</button>

msgAlerta(): void {
    alert('Livro Angular 2');
}
```

- **Two-way data binding** – junta o binding de propriedade com o de evento. Atualiza o template e a classe do componente a qualquer momento que a variável declarada for modificada. Se for atualizado pelo template, o valor vai para a classe do componente; se for atualizado pela classe do componente, o valor é enviado para o template.

```
nome: string = "Thiago";

<input type="text" [(ngModel)]="nome">
```

# Estudo de Caso: *Angular*

## - Angular CLI

### - Diretivas

- Forma de interação com o template – modificam ou interagem dinamicamente com as tags html
- **Diretivas estruturais** – modificam a estrutura da página

```
<ul>
    <li *ngFor="let pessoa of pessoas">
        {{pessoa}}
    </li>
</ul>
pessoas: string [] = ['João', 'Maria', 'Angular 2'];
```

# Estudo de Caso: *Angular*

- Angular CLI
  - Diretivas
    - **Diretivas de atributos** – altera a aparência, comportamento ou conteúdo do elemento que está na tela

```
<input type="text" [(ngModel)]="nome">
```

# Estudo de Caso: *Angular*

## - Angular CLI

- A exibição dos dados é feito pelos templates, construído em HTML com algumas marcações Angular
- Praticamente todas as tags do HTML são válidas, exceto <script> <html> <body>
  - A tag <script> é proibida por segurança
  - <html> e <body> não são usados porque todos os componentes já são renderizados dentro da tag <body> do index.html da aplicação

# Estudo de Caso: Angular

## - Angular CLI

- Criação de novo componente
  - Componente é uma classe responsável por controlar a view
  - Nesta classe, se coloca apenas a lógica de controle do componente

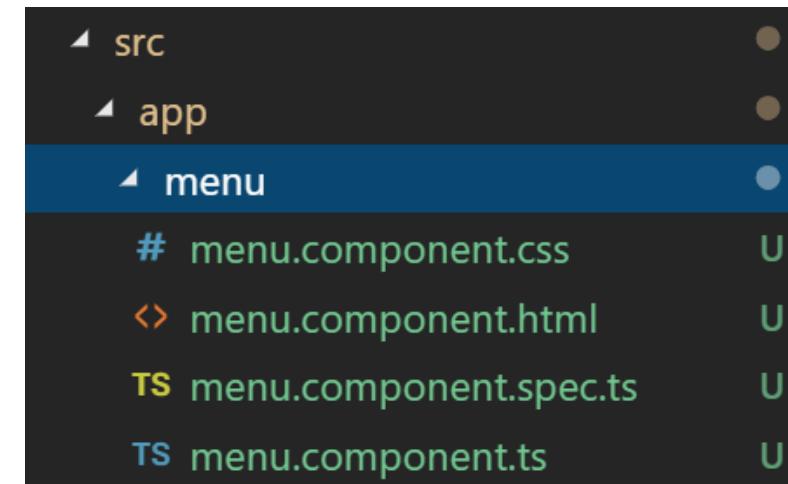
Comando	Descrição
ng	Comando do Angular cli
g	Abreviação do <i>generate</i>
c	Abreviação do <i>component</i>

# Estudo de Caso: Angular

## - Angular CLI

- Criação de novo componente
  - Classe demarcada com o decorator @component

```
PS C:\projetos\SistemaEcommerceCLI> ng g c menu
As a forewarning, we are moving the CLI npm package
which will only support Node 6.9 and greater. This will
shortly after.
```



# Estudo de Caso: Angular

## - Angular CLI

- Copia o código do menu.html, a partir da tag <nav>

```
↳ menu-component.component.html ×  
1   <nav class="navbar navbar-expand-lg navbar-light bg-light">  
2     <a class="navbar-brand" href="#">index.html>Sistema Ecommerce</a>  
3     <button class="navbar-toggler" type="button" data-toggle="collapse" data-  
4       <span class="navbar-toggler-icon"></span>  
5     </button>  
6  
7     <div class="collapse navbar-collapse" id="navbarSupportedContent">  
8       <ul class="navbar-nav mr-auto">  
9         <li class="nav-item active">  
10           <a class="nav-link" href="#">index.html>  
11             img/home.png" width="100%" height="100%"/>  
12             <span class="sr-only">(current)</span></a>  
13           </li>  
14         </ul>  
15       </div>  
16     </div>  
17   </nav>
```

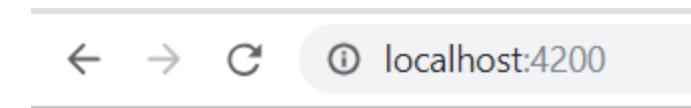
# Estudo de Caso: Angular

## - Angular CLI

- Ao adicionar a referência para o componente

```
menu.component.html      app.module.ts
```

```
1 <h1>
2   {{title}}
3 </h1>
4 <app-menu></app-menu>
5
```



app works!

[Sistema Ecommerce](#)

- [Responsive image \(current\)](#)
- [Cadastros](#)
- [Cliente](#) [Categoria](#) [Produto](#)
- [Pedido](#)
- [Relatórios](#)
- [Pedido](#)

# Estudo de Caso: Angular

## - Bootstrap

- 1) Instalar com npm
- 2) Adicionar scripts em angular-cli.json
- 3) Adicionar estilo em styles.css

```
PS C:\projetos\SistemaEcommerceCLI> npm install bootstrap
```

```
PS C:\projetos\SistemaEcommerceCLI> npm install popper.js
```

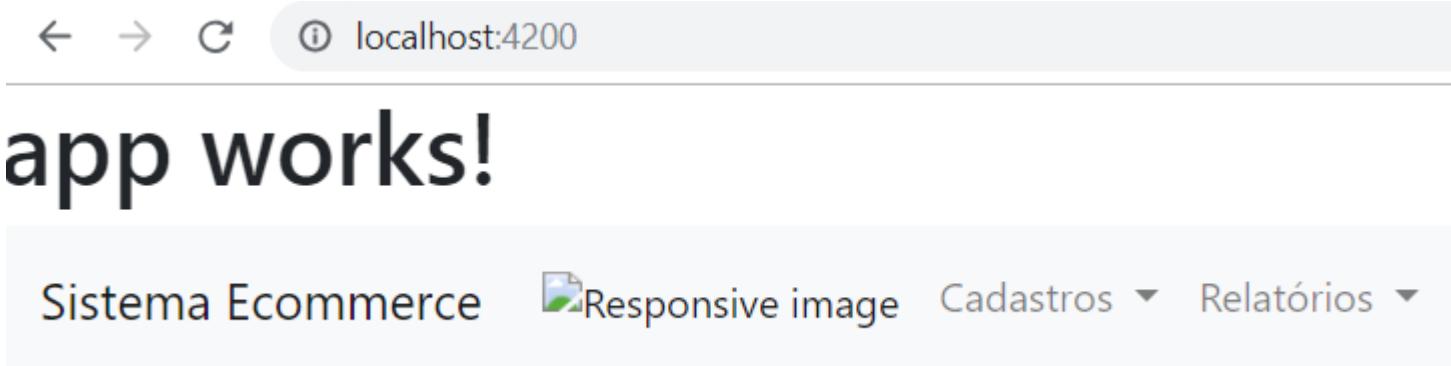
```
PS C:\projetos\SistemaEcommerceCLI> npm install jquery
```

```
{ angular-cli.json x
17   "test": "test.ts",
18   "tsconfig": "tsconfig.json",
19   "prefix": "app",
20   "styles": [
21     "styles.css"
22   ],
23   "scripts": [
24     "../node_modules/jquery/dist/jquery.js",
25     "../node_modules/popper.js/dist/umd/popper.js",
26     "../node_modules/bootstrap/dist/js/bootstrap.js"
27   ],
}
```

```
# styles.css  x
1  /* You can add global styles to this file, and also import other style files */
2  @import "../node_modules/bootstrap/dist/css/bootstrap.min.css"
```

# Estudo de Caso: *Angular*

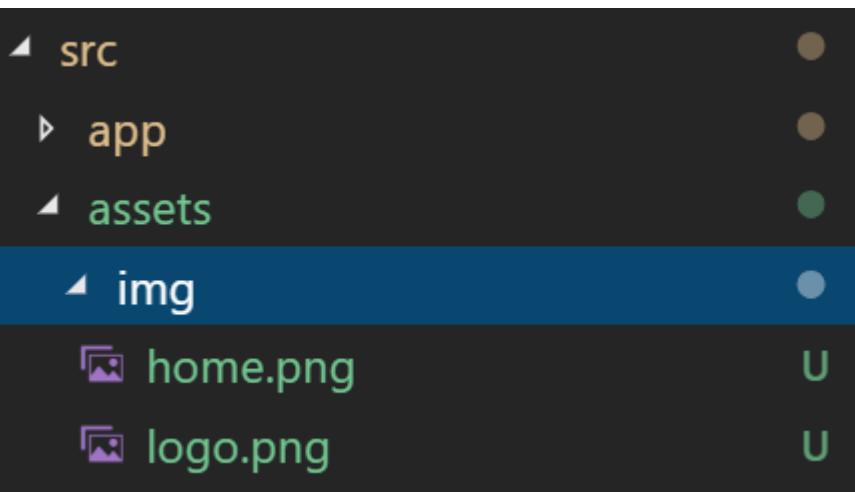
- Angular CLI
  - Bootstrap



# Estudo de Caso: Angular

## - Menu

- 1) Criar diretório de imagens
- 2) Criar componente home
- 3) Adicionar link para a imagem em home.component.html



```
PS C:\projetos\SistemaEcommerceCLI> ng g c home
As a forewarning, we are moving the CLI npm packa
          ↵ home.component.html ✘
          1   
```

# Estudo de Caso: Angular

## - Menu

- 4) Adicionar a referência para menu e home no app.component.html
- 5) Alterar o arquivo menu.html apontando para o novo caminho das imagens

```
↳ app.component.html x
1   <app-menu></app-menu>
2   <app-home></app-home>
```

```
↳ menu.component.html x
8     <ul class="navbar-nav mr-auto">
9       <li class="nav-item active">
10         <a class="nav-link" href="index.html">
11           
  <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>
```

- Uma alternativa seria fazer o Angular trabalhar com rotas

# Estudo de Caso: Angular

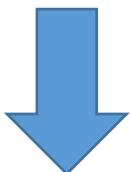
## - Rotas

- As rotas e navegação proporcionam a interação entre componentes no Angular 2.
- As rotas são fundamentais para que o SPA (Single Page Applications) funcione efetivamente.
- O angular irá ler os parâmetros da URL, e carregar/redirecionar o componente responsável pela rota.

`http://sistemas.com.br/Clientes`

`http://sistemas.com.br/Clientes/:id`

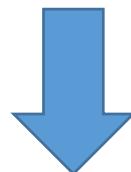
`http://sistemas.com.br/Clientes/:id/edit`



`ListaClientesComponent`



`ClienteDetalhesComponent`



`ClienteFormComponent`

# Estudo de Caso: Angular

## - Rotas

- 1) Criar o arquivo app.router.ts no mesmo diretório do app.module.ts

```
TS app.router.ts ✘

1  import { Routes, RouterModule } from '@angular/router'
2  import { HomeComponent } from './home/home.component';
3  const routes: Routes = [
4      {
5          path: '',
6          component: HomeComponent
7      },
8      {
9          path: 'home',
10         component: HomeComponent
11     }
12 ];
13 export const RoutingModule = RouterModule.forRoot(routes);
```

# Estudo de Caso: Angular

## - Rotas

- 2) Alterar o arquivo app.component.html, chamando o menu e as rotas
- 3) Alterar o arquivo app.module.ts para contemplar o módulo de rotas

```
app.component.html x

1 <app-menu></app-menu>
2 <div class="container">
3   <router-outlet></router-outlet>
4 </div>
```

```
TS app.module.ts x
1 import { NgModule } from '@angular/module';
2 import { HomeComponent } from './home/home.component';
3 import { RouterModule } from './app.router';
4
5 @NgModule({
6   declarations: [
7     AppComponent,
8     MenuComponent,
9     HomeComponent
10   ],
11   imports: [
12     BrowserModule,
13     FormsModule,
14     HttpModule,
15     RouterModule
16   ],
17   providers: []
18 })
19
20 
```

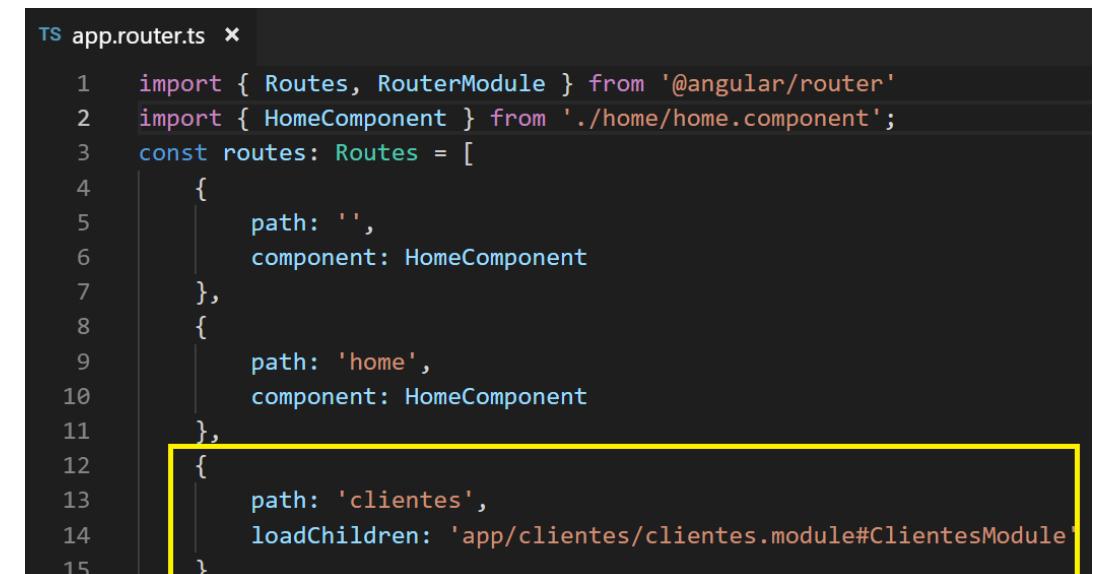
# Estudo de Caso: Angular

## - Rotas

- 4) Alterar o arquivo menu.component.html para contemplar o routerLink

```
TS app.router.ts      <> menu.component.html x

1  <nav class="navbar navbar-expand-lg navbar-light bg-light">
2    <a class="navbar-brand" routerLink="home">Sistema Ecommerce</a>
3    <button class="navbar-toggler" type="button" data-toggle="collapse"
4      <span class="navbar-toggler-icon"></span>
5    </button>
6
7    <div class="collapse navbar-collapse" id="navbarSupportedContent">
8      <ul class="navbar-nav mr-auto">
9        <li class="nav-item active">
10          <a class="nav-link" routerLink="home">
11             menu.component.html ×
17
18   </a>
19   <div class="dropdown-menu" aria-labelledby="navbarDropdown">
    <a class="dropdown-item" routerLink="clientes">Cliente</a>
```

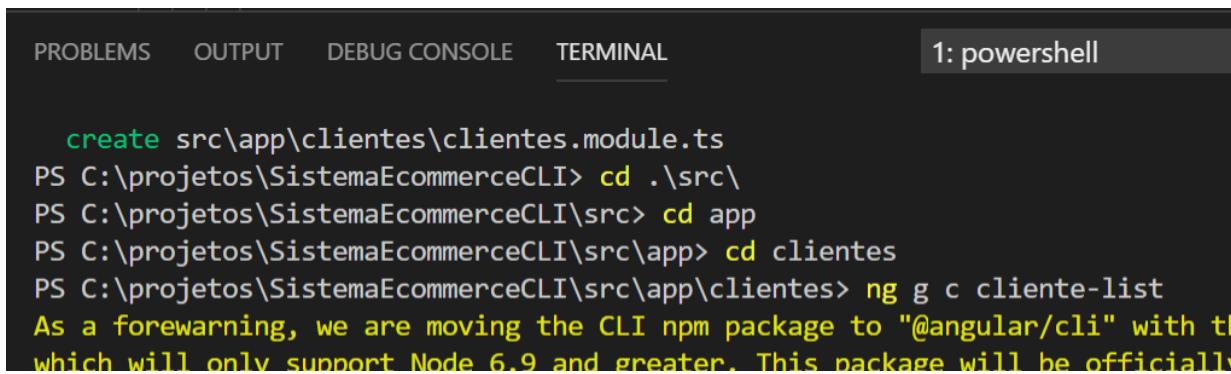


```
TS app.router.ts ×
1  import { Routes, RouterModule } from '@angular/router'
2  import { HomeComponent } from './home/home.component';
3  const routes: Routes = [
4    {
5      path: '',
6      component: HomeComponent
7    },
8    {
9      path: 'home',
10     component: HomeComponent
11   },
12   {
13     path: 'clientes',
14     loadChildren: 'app/clientes/clientes.module#ClientesModule'
15 }
```

# Estudo de Caso: Angular

## - Componente de listar os clientes

- 1) Criar um novo componente para Listar Clientes dentro do módulo de clientes



The screenshot shows a terminal window with the following command history:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
1: powershell

create src\app\clientes\clientes.module.ts
PS C:\projetos\SistemaEcommerceCLI> cd .\src\
PS C:\projetos\SistemaEcommerceCLI\src> cd app
PS C:\projetos\SistemaEcommerceCLI\src\app> cd clientes
PS C:\projetos\SistemaEcommerceCLI\src\app\clientes> ng g c cliente-list
As a forewarning, we are moving the CLI npm package to "@angular/cli" with th
which will only support Node 6.9 and greater. This package will be officially
```

# Estudo de Caso: *Angular*

## - Componente de listar os clientes

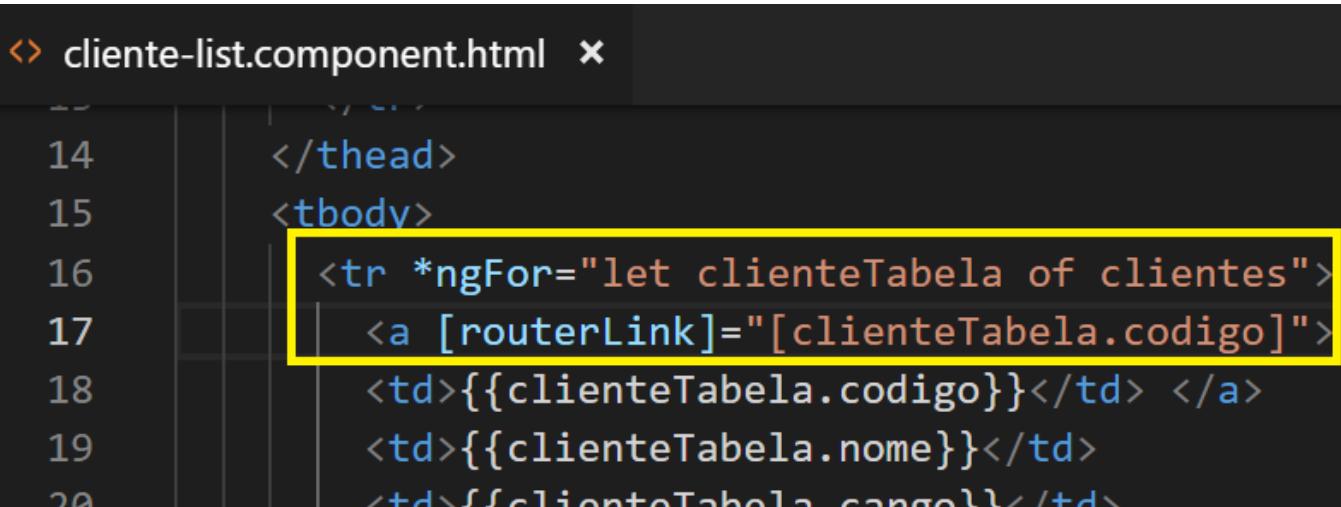
- 2) Colocar o código-fonte da tabela no arquivo lista-clientes.component.html

```
<> cliente-list.component.html ×  
  
1   <table class="table table-hover">  
2     <thead>  
3       <tr>  
4         <th>C&acute;digo</th>  
5         <th>Nome</th>  
6         <th>Cargo</th>  
7         <th>Endere&ccedil;o</th>  
8         <th>Cidade</th>  
9         <th>CEP</th>  
10        <th>Pa&iacute;s</th>  
11        <th>Telefone</th>  
12        <th>Fax</th>  
13      </tr>  
14    </thead>
```

# Estudo de Caso: Angular

## - Componente de listar os clientes

- 3) Alterar as diretivas para a sintaxe do Angular 2\*



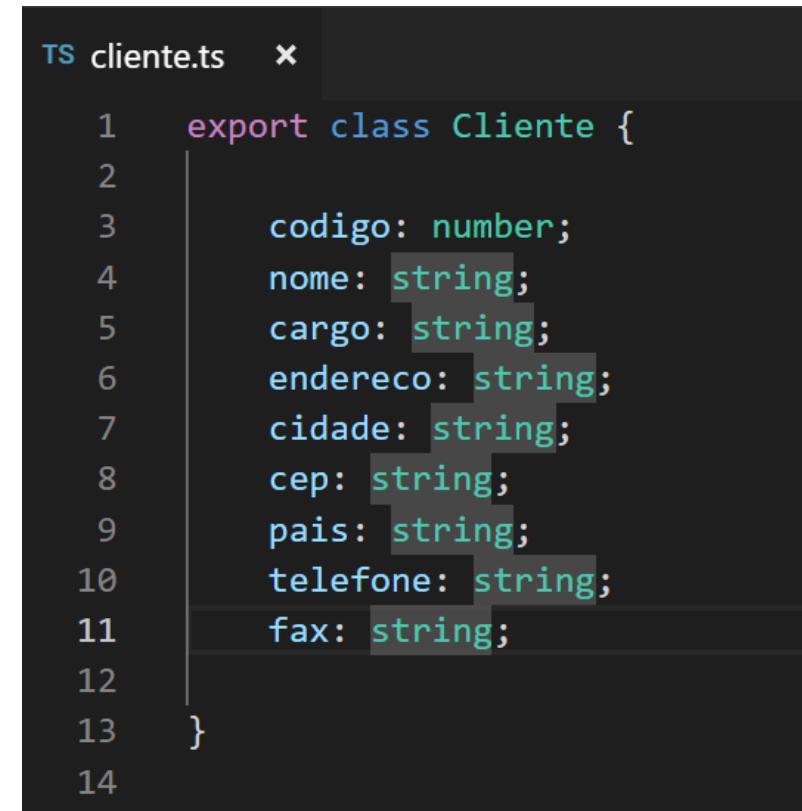
```
<!-- cliente-list.component.html -->
14   </thead>
15   <tbody>
16     <tr *ngFor="let clienteTabela of clientes">
17       <a [routerLink]=[clienteTabela.codigo]">
18         <td>{{clienteTabela.codigo}}</td> </a>
19         <td>{{clienteTabela.nome}}</td>
20         <td>{{clienteTabela.cargo}}</td>
```

- 4) Criar a classe Cliente

```
PS C:\projetos\SistemaEcommerceCLI\src\app\clientes> ng g class Cliente
As a forewarning, we are moving the CLI npm package to "@angular/cli" with
```

# Estudo de Caso: Angular

- Componente de listar os clientes
  - 5) Codificar a classe Cliente



```
TS cliente.ts  ×

1  export class Cliente {
2
3      codigo: number;
4      nome: string;
5      cargo: string;
6      endereco: string;
7      cidade: string;
8      cep: string;
9      pais: string;
10     telefone: string;
11     fax: string;
12
13 }
14
```

# Estudo de Caso: Angular

- Componente de listar os clientes
  - 6) Alterar cliente-list.component.ts

```
TS cliente-list.component.ts ×

1  import { Component, OnInit } from '@angular/core';
2  import { Cliente } from '../cliente';
3
4  @Component({
5    selector: 'app-cliente-list',
6    templateUrl: './cliente-list.component.html',
7    styleUrls: ['./cliente-list.component.css']
8  })
9  export class ClienteListComponent implements OnInit {
10
11    constructor() { }
12
13    clientes: Cliente[] = [];
14
15    ngOnInit() {
16      this.clientes = [
17        { 'codigo': 1,
18          'nome': 'Carlos',
```

```
TS cliente-list.component.ts ×

14
15    ngOnInit() {
16      this.clientes = [
17        { 'codigo': 1,
18          'nome': 'Carlos',
19          'cargo': 'Professor',
20          'endereco': 'Rua teste, 65, Jardim das Palmeiras',
21          'cidade': 'Uberlandia',
22          'cep': '38400-000',
23          'pais': 'Brasil',
24          'telefone': '34944423402',
25          'fax': '343434344'
26      },
27      {
28        'codigo': 2,
29        'nome': 'Martin Fowler',
30        'cargo': 'CEO',
```

# Estudo de Caso: Angular

## - Componente de listar os clientes

- 7) Gerar uma rota para o módulo de clientes, criando o arquivo de rota

```
▲ clientes
  ▾ cliente-list
  TS cliente.ts
  TS clientes.module.ts
  TS clientes.routing.ts
```

```
TS clientes.routing.ts ✘

1  import { Routes, RouterModule } from '@angular/router';
2  import { ClienteListComponent } from './cliente-list/cliente-list.component';
3
4  const CLIENTES_ROUTES: Routes = [
5    { path: '',
6      component: ClienteListComponent
7    }
8  ];
9
10 export const clientesRouting = RouterModule.forChild(CLIENTES_ROUTES);
11
```

# Estudo de Caso: Angular

## - Componente de listar os clientes

- 8) Alterar o arquivo de módulo de clientes, para contemplar a rota.

```
s clientes.module.ts ✘  ↗ cliente-list.component.html  ↗ cliente-form.component.html  TS cliente
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { ClienteListComponent } from './cliente-list/cliente-list.component';
4  import { clientesRoutingModule } from './clientes.routing';
5  import { FormsModule } from '@angular/forms';

6
7  @NgModule({
8    imports: [
9      CommonModule, clientesRoutingModule, FormsModule
10   ],
11   declarations: [ClienteListComponent]
12 })
13 export class ClientesModule { }
```

# Estudo de Caso: Angular

## - Componente de listar os clientes

### Resultado



A screenshot of a web browser displaying a client list in a System Ecommerce application. The URL in the address bar is localhost:4200/clients.

The page header includes the system name "Sistema Ecommerce" and navigation links for "Cadastros" and "Relatórios".

The main content is a table with the following data:

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: Angular

## - Serviço de Clientes

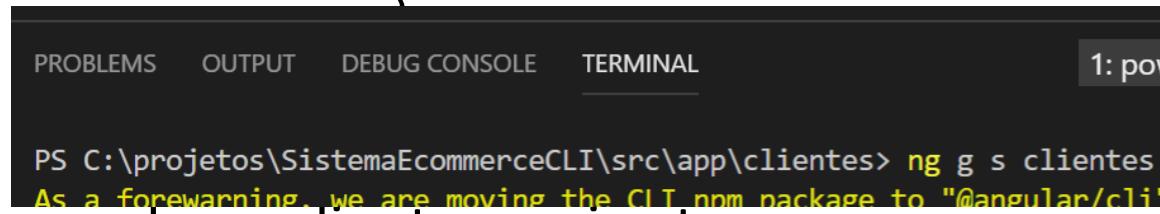
- E se for necessário consumir serviços REST? Usa-se o conceito de Serviços
- Serviços
  - Classe com um decoration @Injectable.
  - Classes usadas para validação de dados, serviços de log, tratamento de erros, tratamento de dados, conexão com o banco, ou seja, **tudo o que estiver fora do escopo de um componente.**

	Comando	Descrição
	ng	Comando do Angular CLI
	g	Abreviação do <i>generate</i>
	s	Abreviação do <i>service</i>

# Estudo de Caso: Angular

## - Serviço de Clientes

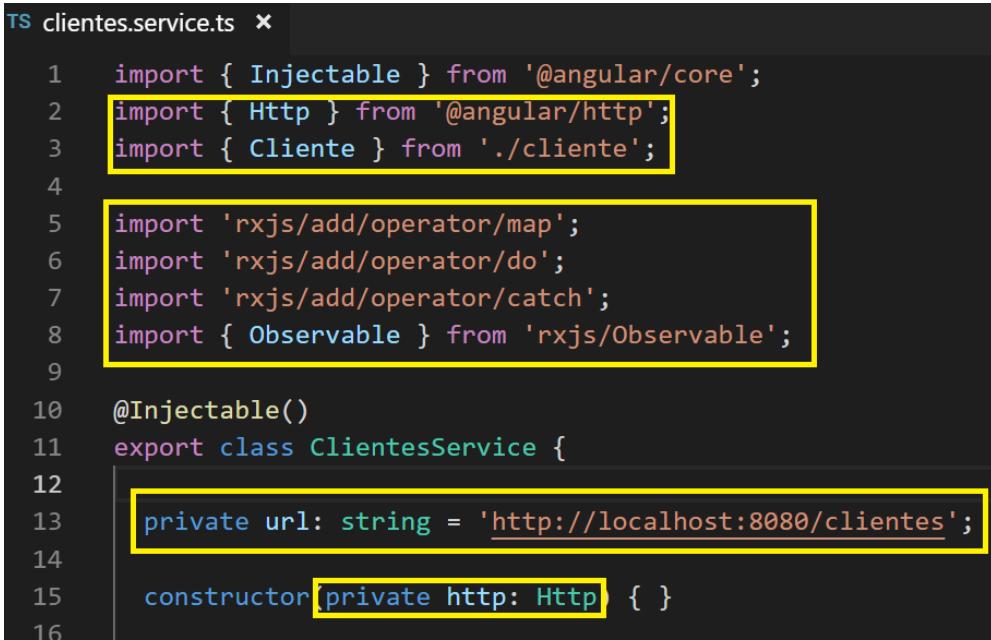
- 1) Criar um novo serviço



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1: pov

PS C:\projetos\SistemaEcommerceCLI\src\app\clientes> ng g s clientes
As a forewarning, we are moving the CLI npm package to "@angular/cli"
```

- 2) Alterar a classe cliente.service.ts



```
ts clientes.service.ts ×

1  import { Injectable } from '@angular/core';
2  import { Http } from '@angular/http';
3  import { Cliente } from './cliente';
4
5  import 'rxjs/add/operator/map';
6  import 'rxjs/add/operator/do';
7  import 'rxjs/add/operator/catch';
8  import { Observable } from 'rxjs/Observable';
9
10 @Injectable()
11 export class ClientesService {
12
13   private url: string = 'http://localhost:8080/clientes';
14
15   constructor(private http: Http) { }
```



```
ts clientes.service.ts ×

19   getAll(): Observable<Cliente[]> {
20     return this.http.get(this.url)
21       .map(res => res.json())
22       .catch(this.handleError);
23   }
24
25   private handleError(error: any) {
26     let erro = error.message || 'Server error';
27     console.error('Ocorreu um erro',error);
28     return Observable.throw(erro);
29 }
```

# Estudo de Caso: Angular

## - Serviço de Clientes

- 3) Alterar a classe lista-clientes.component.ts
- 4) Declarar o serviço em clientes.modules.ts

```
TS cliente-list.component.ts ✘

11
12  constructor(private clienteService: ClientesService) { }
13
14  clientes: Cliente[] = [];
15
16  criterio: String;
17
18  ngOnInit() {
19    this.clienteService.getAll()
20      .subscribe(data => this.clientes = data, err => {
21        alert('Aconteceu um erro!');
22      });
23  }
24
25 }
```

```
import { ClientesService } from './clientes.service';

@NgModule({
  imports: [
    CommonModule, clientesRouting, FormsModule
  ],
  declarations: [ClientelistComponent],
  providers: [ClientesService]
})
export class ClientesModule { }
```

# Estudo de Caso: Angular

## - Serviço de Clientes

- 5) Alterar o cross origin

```
J ClienteResource.java ✘
  1 import org.springframework.web.servlet.support.ServiceOut
  2
  3 @RestController
  4 @RequestMapping(value = "/clientes")
  5 @CrossOrigin(origins = "http://localhost:4200")
  6 public class ClienteResource {
  7
  8     @Autowired
  9     private ClienteRepository service;
 10
 11     @GetMapping
 12     public ResponseEntity<List<Cliente>> findAll() {
 13         List<Cliente> alunos = service.findAll();
 14         return ResponseEntity.ok().body(alunos);
 15     }
 16 }
```

# Estudo de Caso: Angular

## - Injeção de Dependência

- Declarar e injetar dentro da classe do componente uma classe de serviço
- Colocar a dependência dentro do app.module.ts cria uma injeção global na aplicação

```
import { Component } from '@angular/core';

import { AlertaService } from './alerta.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  constructor(private service: AlertaService) { }

  enviarMsg(): void {
    this.service.msgAlerta();
  }
}
```

# Estudo de Caso: Angular

## - Componente de formulário dos clientes

- 1) Criação do componente

```
Windows PowerShell
PS C:\Users\carlo\projetonovo\src\app\clientes> ng g c cliente-form
As a forewarning, we are moving the CLI npm package to "@angular/cli"
```

- 2) Copiar o código html do formulário

```
<> cliente-form.component.html ×

1   <button type="button" ng-click="novo()" class="btn btn-primary">Novo</button>
2   <button type="button" ng-click="salvar()" class="btn btn-success">Salvar</button>
3   <button type="button" ng-click="excluir()" class="btn btn-danger">Excluir</button>
4   <button type="button" ng-click="pesquisar()" class="btn btn-warning">Pesquisar</button>
5
6   <form class="was-validated form-row">
7     <div class="col-md-1 mb-3">
8       <label for="txtCodigo">Código</label>
9       <input type="text" ng-model="cliente.codigo" class="form-control" disabled>
10      </div>
11
```

# Estudo de Caso: Angular

- Componente de formulário dos clientes
  - 3) Editar para a sintaxe do Angular (editar todos os campos)

```
<button type="button" (click)="novo()" class="botao">
<button type="button" (click)="salvar()" class="botao">
<button type="button" (click)="excluir()" class="botao">
<button type="button" (click)="pesquisar()" class="botao">

<div class="col-md-4 mb-3">
  <label for="txtNome">Nome</label>
  <input type="text" [(ngModel)]="cliente.nome"
    class="form-control is-valid" name="txtNome" required>
  <div class="invalid-feedback">
    Digite o nome.
  </div>
</div>
```

# Estudo de Caso: Angular

## - Componente de formulário dos clientes

- 4) Alterar o cliente.service.ts para contemplar as funções de salvar, atualizar e excluir

TS clientes.service.ts x

```

1 import { Injectable, EventEmitter } from '@angular/core';
2 import { Http, Headers } from '@angular/http';
3 import { Observable } from 'rxjs/Rx';
4 import { Cliente } from './cliente';

5 import 'rxjs/add/operator/map';
6 import 'rxjs/add/operator/do';
7 import 'rxjs/add/operator/catch';

8
9

11 export class ClientesService {
12   private url: string = 'http://localhost:8080/clientes';
13
14   clientesChanged = new EventEmitter<Observable<Cliente[]>>();
15
16   constructor(private http: Http) { }

17
18   getAll(): Observable<Cliente[]> {
19     return this.http.get(this.url)
20       .map(res => res.json().data)
21       .catch(this.handleError);
22   }
23 }
```

TS clientes.service.ts x

```

30
31   add(cliente: Cliente) {
32     return this.http.post(this.url, JSON.stringify(cliente),
33       {headers: this.getHeaders()})
34       // .map(res => res.json())
35       .do(data => this.clientesChanged.emit(this.getAll()))
36       .catch(this.handleError);
37   }

38
39   remove(id: number) {
40     return this.http.delete(this.getUrl(id), {headers: this.getHeaders()})
41       .map(res => res.json())
42       .do(data => this.clientesChanged.emit(this.getAll()))
43       .catch(this.handleError);
44   }

```

# Estudo de Caso: Angular

- Componente de formulário dos clientes
  - 4) (continuação..)

```
TS clientes.service.ts ×

46  private getHeaders() {
47    let headers = new Headers();
48    headers.append('Content-Type', 'application/json');
49    return headers;
50  }
51
52  private getUrl(id: number) {
53    return `${this.url}/${id}`;
54  }
55
56  update(cliente: Cliente) {
57    return this.http.put(this.url, JSON.stringify(cliente),
58      {headers: this.getHeaders()})
59      //.map(res => res.json().data)
60      .do(data => this.clientesChanged.emit(this.getAll()))
61      .catch(this.handleError);
62  }
```

```
get(id: number){
  return this.getAll()
    .map((list: any) => list.find(cliente => cliente.codigo == id))
    .catch(this.handleError);
}
```

# Estudo de Caso: Angular

## - Componente de formulário dos clientes

- 5) Alterar o cliente.form.component.ts para contemplar as funções de salvar, atualizar e excluir

TS cliente-form.component.ts ×

```
1 import { Component, OnInit } from '@angular/core';
2 import { Subscription } from 'rxjs/Subscription';
3 import { ActivatedRoute } from '@angular/router';
4 import { ClientesService } from '../clientes.service';
5 import { Cliente } from '../cliente';
6
7 @Component({
8   selector: 'app-cliente-form',
9   templateUrl: './cliente-form.component.html',
10  styleUrls: ['./cliente-form.component.css']
11 })
12 export class ClienteFormComponent implements OnInit {
```

TS cliente-form.component.ts × ⇕ cliente-form.component.html TS clientes.ser

```
12   export class ClienteFormComponent implements OnInit {
13
14     private clienteIndex: number;
15     private isNew: boolean = true;
16     private cliente: Cliente;
17     private subscription: Subscription;
18
19     constructor(private route: ActivatedRoute,
20                 private clienteService: ClientesService) { }
21
22     ngOnInit() {
23       this.novo();
24       this.subscription = this.route.params.subscribe(
25         (params: any) => {
26           if (params.hasOwnProperty('id')) {
27             this.isNew = false;
28             this.clienteIndex = params['id'];
29           }
30         }
31       );
32     }
33
34     novo() {
35       this.cliente = new Cliente();
36     }
37
38     salvar() {
39       if (this.isNew) {
40         this.clienteService.inserir(this.cliente);
41       } else {
42         this.clienteService.alterar(this.cliente);
43       }
44     }
45
46     excluir() {
47       this.clienteService.excluir(this.clienteIndex);
48     }
49
50     cancelar() {
51       this.novo();
52     }
53   }
54 }
```

# Estudo de Caso: Angular

- Componente de formulário dos clientes
  - 5) (continuação)

TS cliente-form.component.ts ×

```

27         this.clienteIndex = +params['id'];
28         this.clienteService.get(this.clienteIndex)
29             .subscribe(data => this.cliente = data);
30     } else {
31         this.isNew = true;
32         -----
33     }
34 }
35 );
36 }
37
38 novo() {
39     this.cliente = new Cliente();
40 }
```

```

43     salvar() {
44         let result;
45         if (this.isNew) {
46             result = this.clienteService.add(this.cliente);
47         } else {
48             result = this.clienteService.update(this.cliente);
49         }
50         this.novo();
51         result.subscribe(data => alert('sucesso ' + data),
52                         err => {
53                             alert("An error occurred. " + err);
54                         });
55     }
56
57     excluir() {
58         if (this.cliente.codigo == null) {
59             alert("Selecione algum cliente");
60         } else {
61             if (confirm("Você realmente quer excluir o cliente " + this.cliente.nome + "?"))
62                 this.clienteService.remove(this.cliente.codigo)
63                     .subscribe(
64                         data => this.novo,
65                         err => {
66                             alert("Cliente não removido.");
67                         });
68     }
69 }
```

# Estudo de Caso: Angular

## - Componente de formulário dos clientes

- 6) Alterar a rota

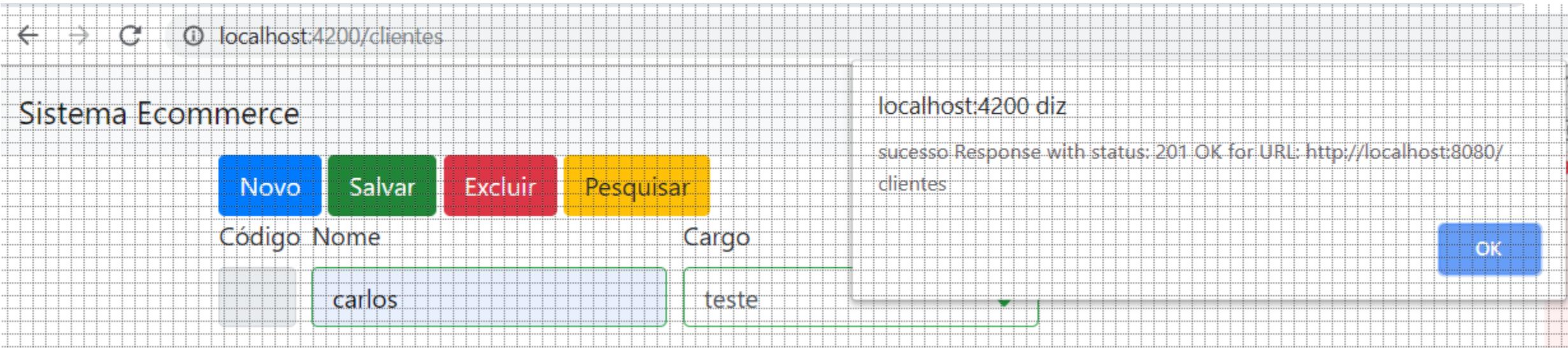
- 7) Incluir a dependência de FormsModule

```
TS clientes.module.ts ×
1 import { NgModule } from '@angular/core';
2 import { CommonModule } from '@angular/common';
3 import { ClientesRoutingModule } from './clientes-routing.module';
4 import { ClientesComponent } from './clientes.component';
5 import { ClienteListComponent } from './cliente-list/cliente-list.component';
6 import { ClienteFormComponent } from './cliente-form/cliente-form.component';
7
8 @NgModule({
9   declarations: [
10     ClientesComponent,
11     ClienteListComponent,
12     ClienteFormComponent
13   ],
14   imports: [
15     CommonModule,
16     ClientesRoutingModule,
17     FormsModule
18   ]
19 })
```

```
TS cliente-form.component.ts
1 import { Routes, RouterModule } from '@angular/router';
2 import { ClienteListComponent } from './cliente-list/cliente-list.component';
3 import { ClienteFormComponent } from './cliente-form/cliente-form.component';
4
5 const CLIENTES_ROUTES: Routes = [
6   {
7     path: '',
8     component: ClienteFormComponent
9   }
10];
11
12 export const ClientesRoutes = RouterModule.forChild(CLIENTES_ROUTES);
```

# Estudo de Caso: Angular

## - Componente de formulário dos clientes



The screenshot shows a browser window with the URL `localhost:4200/Clientes`. The page title is "Sistema Ecommerce". On the left, there is a navigation bar with back, forward, and refresh icons. Below the title, there are four buttons: "Novo" (blue), "Salvar" (green), "Excluir" (red), and "Pesquisar" (yellow). To the right of these buttons, there are two input fields: "Código" with value "carlos" and "Nome" with value "teste". Above the "Nome" field is a dropdown menu labeled "Cargo" with the value "teste". A success message from the server is displayed on the right side of the screen, reading: "localhost:4200 diz sucesso Response with status: 201 OK for URL: http://localhost:8080/Clientes". At the bottom right, there is a blue button labeled "OK".

# Estudo de Caso: Angular

## - Formulários com o Angular

- O Angular 2 possui um gerenciador de formulário
- O atributo `name` dentro da tag HTML vai servir para o Angular 2 ter o controle total do elemento no formulário. E em conjunto com a diretiva `[(ngModel)]` , poderá fazer a consulta e atualização do conteúdo dentro da tag.

```
<div class="col-md-4 mb-3">
  <label for="nome">Nome</label>
  <input type="text" [(ngModel)]="cliente.nome" class="form-control is-valid"
    name="nome" required>
  <div class="invalid-feedback">
    Digite o nome.
  </div>
</div>
```

# Estudo de Caso: Angular

## - Formulários com o Angular

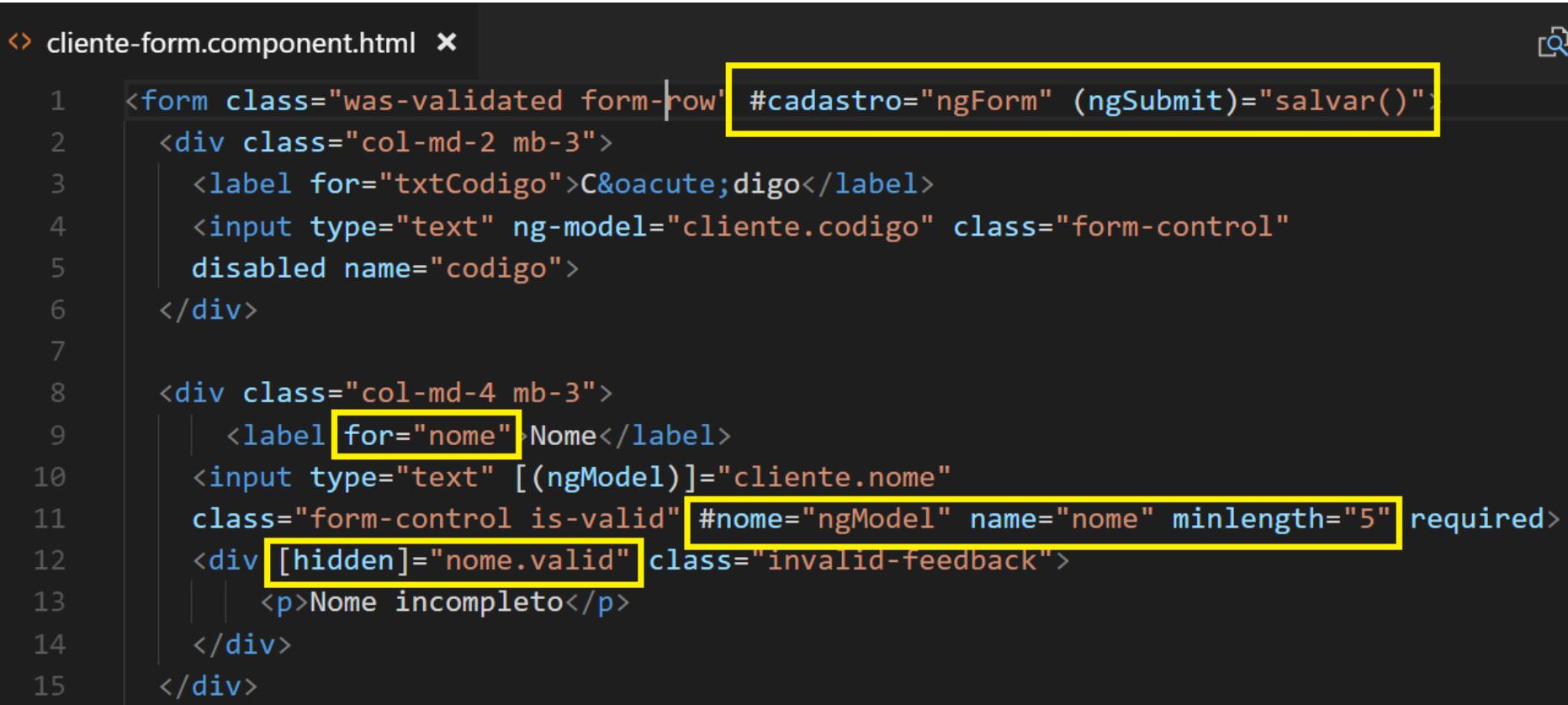
- Como os campos se transformaram em objetos de formulário no Angular 2, é possível validar os objetos

Estado do objeto	Verdadeiro	Falso
Objeto foi clicado	<i>ng-touched</i>	<i>ng-untouched</i>
Valor do objeto foi mudado	<i>ng-dirty</i>	<i>ng-pristine</i>
Objeto está válido	<i>ng-valid</i>	<i>ng-invalid</i>

# Estudo de Caso: Angular

## - Formulários com o Angular

- 1) Alterar o arquivo cliente-form.component.html



```
cliente-form.component.html ✘
1 <form class="was-validated form-row" #cadastro="ngForm" (ngSubmit)="salvar()">
2   <div class="col-md-2 mb-3">
3     <label for="txtCodigo">Código</label>
4     <input type="text" ng-model="cliente.codigo" class="form-control"
5       disabled name="codigo">
6   </div>
7
8   <div class="col-md-4 mb-3">
9     <label for="nome">Nome</label>
10    <input type="text" [(ngModel)]="cliente.nome"
11      class="form-control is-valid" #nome="ngModel" name="nome" minlength="5" required>
12    <div [hidden]="nome.valid" class="invalid-feedback">
13      <p>Nome incompleto</p>
14    </div>
15  </div>
```

# Estudo de Caso: Angular

## - Formulários com o Angular

```
cliente-form.component.html ×

17  <div class="col-md-4 mb-3">
18    <label for="cargo">Cargo</label>
19    <input type="text" [(ngModel)]="cliente.cargo"
20      class="form-control is-valid" #cargo="ngModel" name="cargo" required>
21    <div [hidden]="cargo.valid" class="invalid-feedback">
22      <p>Cargo é obrigatório</p>
23    </div>
24  </div>

25
26  <div class="col-md-5 mb-3">
27    <label for="endereco">Endereço</label>
28    <input type="text" [(ngModel)]="cliente.endereco"
29      class="form-control is-valid" #endereco="ngModel" name="endereco" required>
30    <div [hidden]="endereco.valid" class="invalid-feedback">
31      <p>Endereço é obrigatório</p>
32    </div>
```

# Estudo de Caso: Angular

## - Formulários com o Angular

```
↳ cliente-form.component.html •  
  
71   <div class="col-md-2">  
72     <label for="fax">Fax</label>  
73     <input type="text" [(ngModel)]="cliente.fax"  
74       class="form-control is-valid" #fax="ngModel" name="fax" required>  
75     <div class="invalid-feedback">  
76       <p>Fax é obrigatório</p>  
77     </div>  
78   </div>  
79  
80   <div class="container">  
81     <button class="btn btn-success" type="submit"  
82       [disabled]="cadastro.form.invalid">Enviar</button>  
83     <button class="btn btn-warning" type="button"  
84       (click)="cancelar()">Cancelar</button>  
85   </div>  
86 </form>
```

# Estudo de Caso: Angular

## - Formulários com o Angular

- 2) Alterar o arquivo cliente-form.component.ts

```
constructor(private route: ActivatedRoute,  
           private router: Router,  
           private clienteService: ClientesService) {  
}
```

```
48   cancelar() {  
49     this.router.navigate(['/clientes']);  
50   }  
51 }
```

# Estudo de Caso: Angular

## - Formulários com o Angular

← → ⌂ ⓘ localhost:4200/clientes

Sistema Ecommerce  Cadastros ▾ Relatórios ▾

Código	Nome	Cargo
<input type="text"/>	<input type="text"/> <span style="color: red;">×</span>	<input type="text"/> <span style="color: red;">×</span>
	Nome incompleto	Cargo é obrigatório
Endereço	Cidade	CEP
<input type="text"/> <span style="color: red;">×</span>	<input type="text"/> <span style="color: red;">×</span>	<input type="text"/> <span style="color: red;">×</span>
Endereço é obrigatório	Cidade é obrigatório	CEP é obrigatório
País	Telefone	Fax
<input type="text"/> <span style="color: red;">×</span>	<input type="text"/> <span style="color: red;">×</span>	<input type="text"/> <span style="color: red;">×</span>
País é obrigatório	Telefone é obrigatório	Fax é obrigatório

Enviar Cancelar

# Estudo de Caso: Angular

## - Componente de CRUD cliente

- 1) Criar o componente

```
PS C:\projetos\SistemaEcommerceCLI\src\app\clientes> ng g c cliente-crud
As a forewarning, we are moving the CLI npm package to "@angular/cli" wit
h the next major release. You can already use it by running "npm install @angular/cli@next".
```

- 2) Construir o cliente-crud.component.html

```
<> cliente-crud.component.html ×
```

```
1  <router-outlet></router-outlet>
2  <div class="card border-secondary mb-3">
3  |  <div class="card-header">Cadastro de Clientes</div>
4  </div>
5
6  <ul class="nav nav-tabs" id="clienteTab" role="tablist">
7  |  <li class="nav-item">
8  |  |  <a class="nav-link active" id="cadastro-tab" data-toggle="tab" href="#cadastro"
9  |  |  |  aria-controls="tabCadastro" aria-selected="true">Cadastro</a>
10 |  </li>
```

# Estudo de Caso: Angular

## - Componente de CRUD cliente

### - 2) Continuação

```
↳ cliente-crud.component.html ✘

 8   |   |   aria-controls="tabCadastro" aria-selected="true">Cadastro</a>
 9   |   </li>
10  |   <li class="nav-item">
11  |   |   <a class="nav-link" id="dados-tab" data-toggle="tab" href="#dados" role="button"
12  |   |   |   aria-selected="false">Dados</a>
13  |   </li>
14 </ul>
15 <div class="tab-content" id="clienteTabContent">
16   <div class="tab-pane fade show active" id="cadastro" role="tabpanel" aria-labelledby="cadastro-tab">
17   |   <app-cliente-form></app-cliente-form>
18   </div>
19   <div class="tab-pane fade" id="dados" role="tabpanel" aria-labelledby="dados-tab">
20   |   <div role="tabpanel" class="tab-pane" id="dados">
21   |   |   <app-cliente-list></app-cliente-list>
22   |   </div>
23   </div>
24 </div>
```

# Estudo de Caso: Angular

## - Componente de CRUD cliente

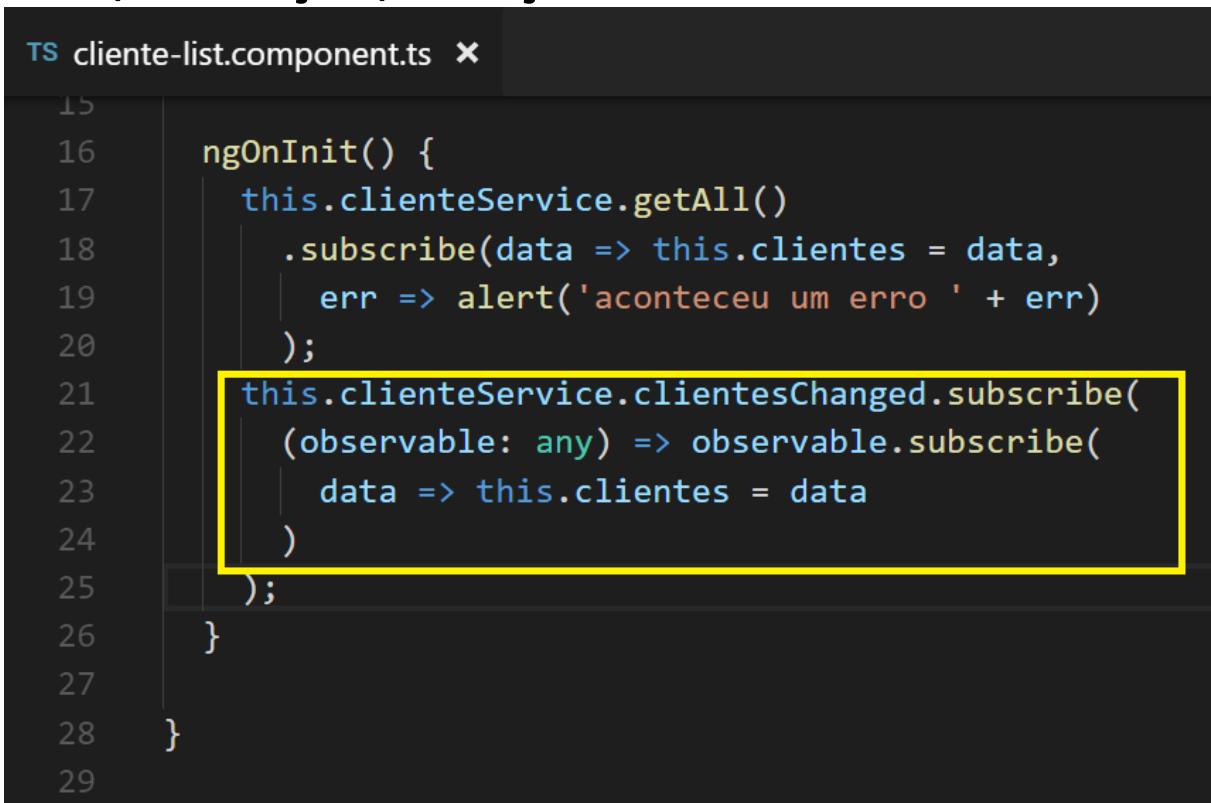
- 3) Alterar a rota

```
$ cliente-form.component.ts      <> cliente-crud.component.html      TS clientes.routing.ts ×  
  
1  import { Routes, RouterModule } from '@angular/router';  
2  import { ClienteListComponent } from './cliente-list/cliente-list.comp  
3  import { ClienteFormComponent } from './cliente-form/cliente-form.comp  
4  import { ClienteCrudComponent } from './cliente-crud/cliente-crud.comp  
5  
6  const CLIENTES_ROUTES: Routes = [  
7    { path: '', component: ClienteCrudComponent },  
8    { path: ':id', component: ClienteCrudComponent }  
9  ];  
10  
11 export const clientesRouting = RouterModule.forChild(CLIENTES_ROUTES);  
12
```

# Estudo de Caso: Angular

## - Componente de CRUD cliente

- 4) Alterar o arquivo cliente-list.component.ts para que a lista seja atualizada a cada inclusão/alteração/deleção de cliente



```
TS cliente-list.component.ts ×

15
16     ngOnInit() {
17         this.clienteService.getAll()
18             .subscribe(data => this.clientes = data,
19                         err => alert('aconteceu um erro ' + err)
20                     );
21         this.clienteService.clientesChanged.subscribe(
22             (observable: any) => observable.subscribe(
23                 data => this.clientes = data
24             )
25         );
26     }
27
28 }
29
```

# Estudo de Caso: Angular

- Modificando o CRUD para separação de telas
  - 1) Modificar cliente-crud.component.html

```
<> cliente-crud.component.html ✘  
  
1   <div class="row">  
2     <div class="col s5">  
3       <app-cliente-list></app-cliente-list>  
4     </div>  
5     <div class="col s7">  
6       <router-outlet></router-outlet>  
7     </div>  
8   </div>
```

# Estudo de Caso: Angular

- Modificando o CRUD para separação de telas
  - 2) Criar uma rota filha

```
TS clientes.routing.ts ✘

4  import { ClienteCrudComponent } from "./cliente-crud/cliente-crud.component";
5
6
7  const CLIENTES_ROUTES: Routes = [
8    {
9      path: '',
10     component: ClienteCrudComponent, children: [
11       { path: ':id', component: ClienteFormComponent }]
12   }
13 ];
14 export const clientesRouting = RouterModule.forChild(CLIENTES_ROUTES);
```

# Estudo de Caso: Angular

## - Modificando o CRUD para separação de telas

Sistema Ecommerce  Cadastros ▾ Relatórios ▾

Código	Nome	Cargo	Endereço	Cidade	CEP	País	Telefone	Fax
1	carlos	aa	a	a	a	a	a	a
3	a	a	a	a	a	a	a	a
4	a	a	a	a	a	a	a	a
5	a	a	a	a	a	a	a	a
6	a	a	a	a	a	a	a	a
7	a	a	a	a	a	a	a	qq
8	carlos	teste	a	a	a	a	a	a

**Novo** **Salvar** **Excluir** **Pesquisar**

Código Nome Cargo

Endereço Cidade CEP

País Telefone Fax

# Estudo de Caso: Angular

## - Modificando o CRUD para separação de telas

- 3) Alterar o layout dos campos

```
<div class="col-md-2 mb-3">
  <label for="txtCodigo">Código:</label>
  <input type="text" [(ngModel)]='cliente.codigo'>
</div>
```

- 4) Retirar o botão de novo do component de form e levar para o componente de lista

```
client-list.component.html ×

29   </table>
30
31   <a routerLink="new">
32     <button type="button" class="btn btn-primary">Novo Cliente</button>
33   </a>
34
```

# LINKS DOS SOFTWARES UTILIZADOS

---

- **Visual Studio Code** - <https://code.visualstudio.com/>
- **Node JS** - <https://nodejs.org/en/>
  
- Link repositório GITHUB: <https://github.com/carloseduardoxp/pds2-2019-1>

# REFERÊNCIAS

- GUEDES, Thiago. Crie aplicações com Angular. Casa do Código, 2018.
- GRONER, Loiane. Curso Angular. Disponível em:  
<https://www.youtube.com/watch?v=tPOMG0D57S0&list=PLGxZ4Rq3BOBoSRcKWEdQACbUCNWLczg2G>. Acesso em 02/04/2019.