



APLICAÇÕES INTERATIVAS

ANGULAR

Sobre Angular

- Angular (também conhecido como Angular 2) é uma plataforma de código aberto para construção de aplicações web front end em formato SPA (Single-Page Applications)

Sobre Angular

- Angular é feito em TypeScript e mantido pela equipe Angular, do Google, e pela comunidade
- Como é feito em TypeScript, Angular promove o uso desta linguagem para composição das aplicações

Angular e TypeScript

- Angular 2 é uma reescrita completa do AngularJS
- Angular 2 (e suas versões superiores) é bastante diferente do AngularJS, tanto em composição quanto em linguagem (AngularJS utiliza JavaScript)

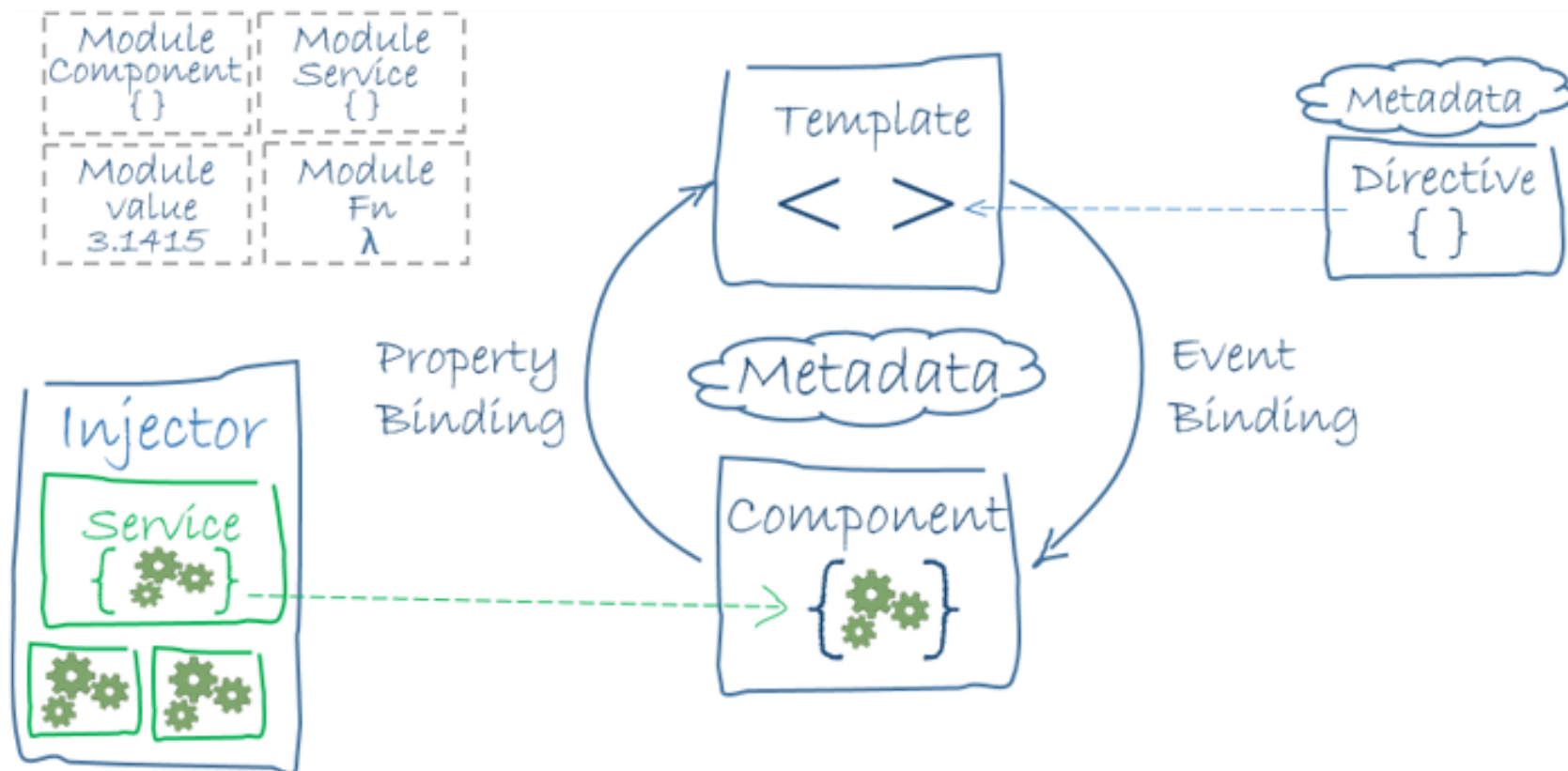
Arquitetura do Angular

- O Angular utiliza uma arquitetura baseada em componentes visuais, templates, diretivas e serviços

Arquitetura do Angular

- Os elementos de arquitetura do angular visam, principalmente, estimular o reuso e o encapsulamento de elementos visuais, comportamentos e funções

Arquitetura do Angular



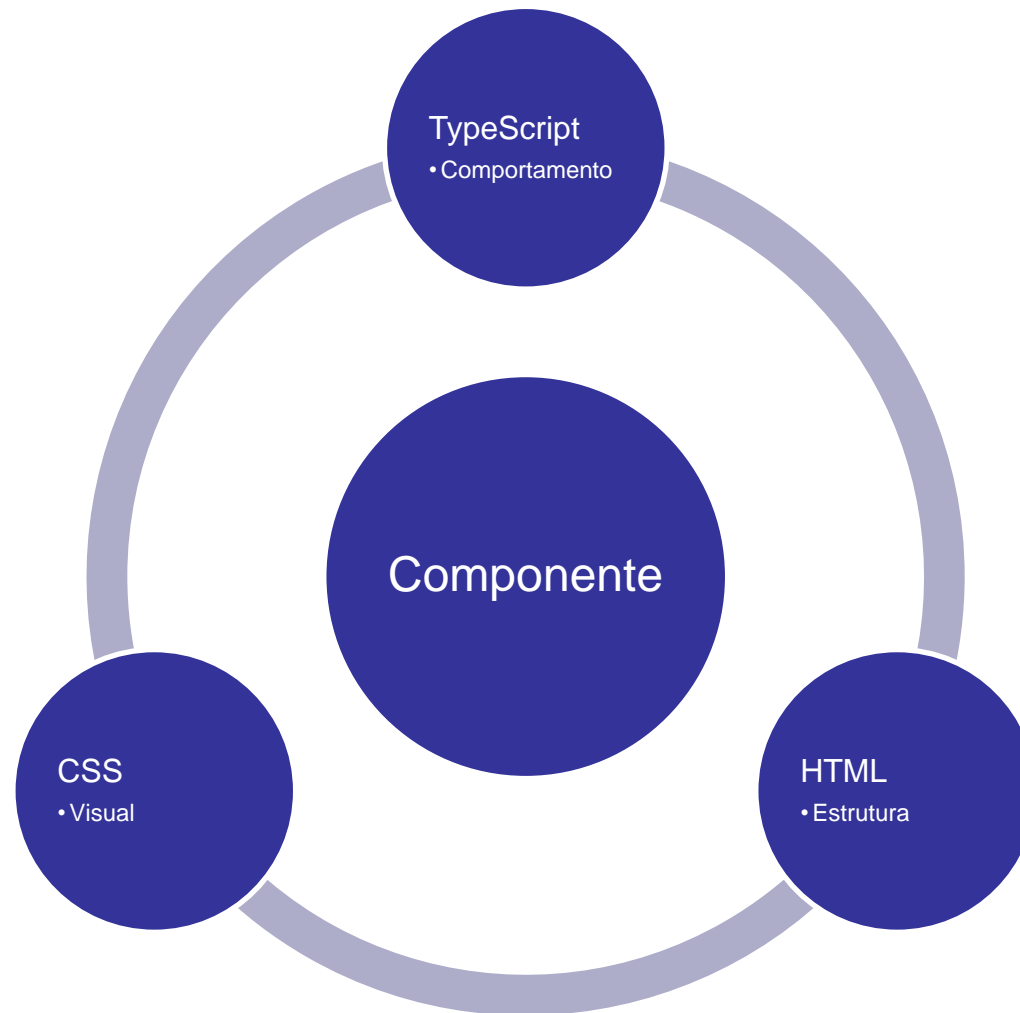
Componentes

- Em Angular, elementos visuais da interfaces são construídos através de componentes
- Cada componente representa um “pedaço” da página que será exibido e terá interações com o usuário

Componentes

- Os componentes são formados pelo código de comportamento (TypeScript), estrutura (HTML) e estilo visual (CSS)

Componentes



Templates

- Templates compõem a estrutura (HTML) de componentes que formam a página

Templates

- Cada componente possui um template específico, mas podem ser utilizados componentes no interior de outros componentes para composição de toda a aplicação

Diretivas

- Diretivas permitem estender e customizar HTML através do uso de elementos que indicam como a estrutura deve ser composta

Diretivas

- Por exemplo, é possível especificar se um trecho de HTML deve ou não ser exibido, de acordo com uma condição (ngIf) ou repetir um trecho para cada posição de uma lista (ngFor)

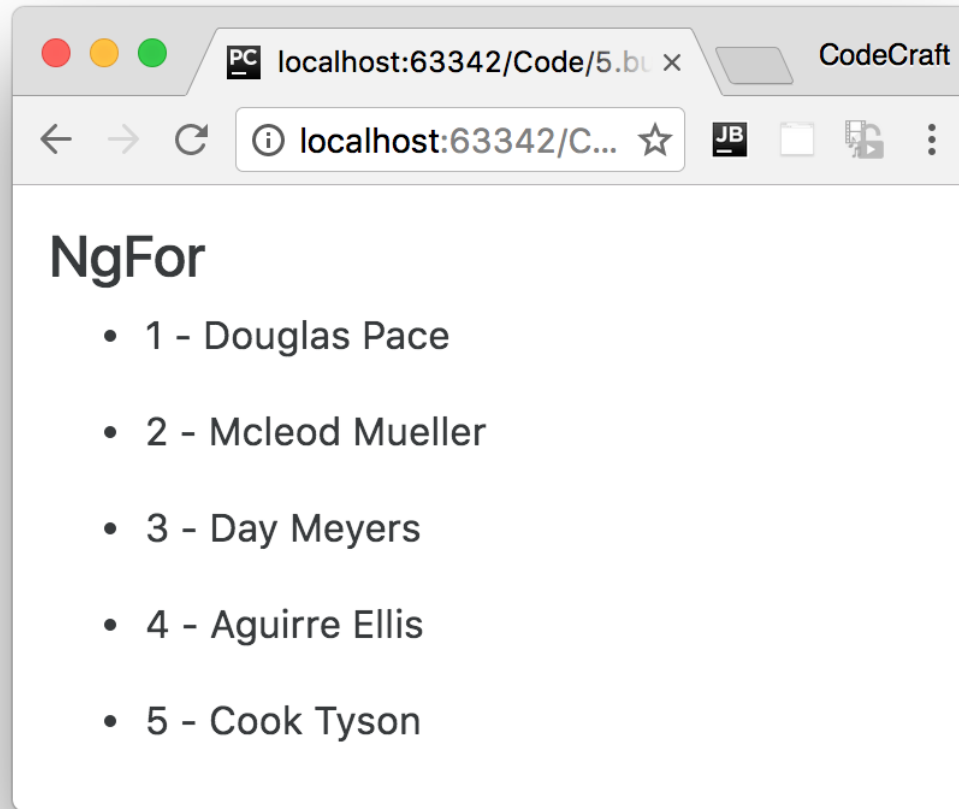
Diretivas e Binding

- Através das diretivas e da ligação de elementos HTML a código TypeScript (procedimento conhecido como binding), é possível customizar como a estrutura do componente será montada e exibida

Diretivas e Binding

- Por exemplo, podemos exibir uma timeline onde cada item (post) possui um template que é repetido para cada item de um array (array de posts), tendo seus valores de exibição alterados de acordo com os valores do item de post de cada posição do array

Diretivas e Binding



Serviços

- Serviços são elementos genéricos que representam quaisquer valores, funções ou recursos utilizados comumente num aplicativo
- O Angular distingue os componentes dos serviços para aumentar modularidade e reutilização

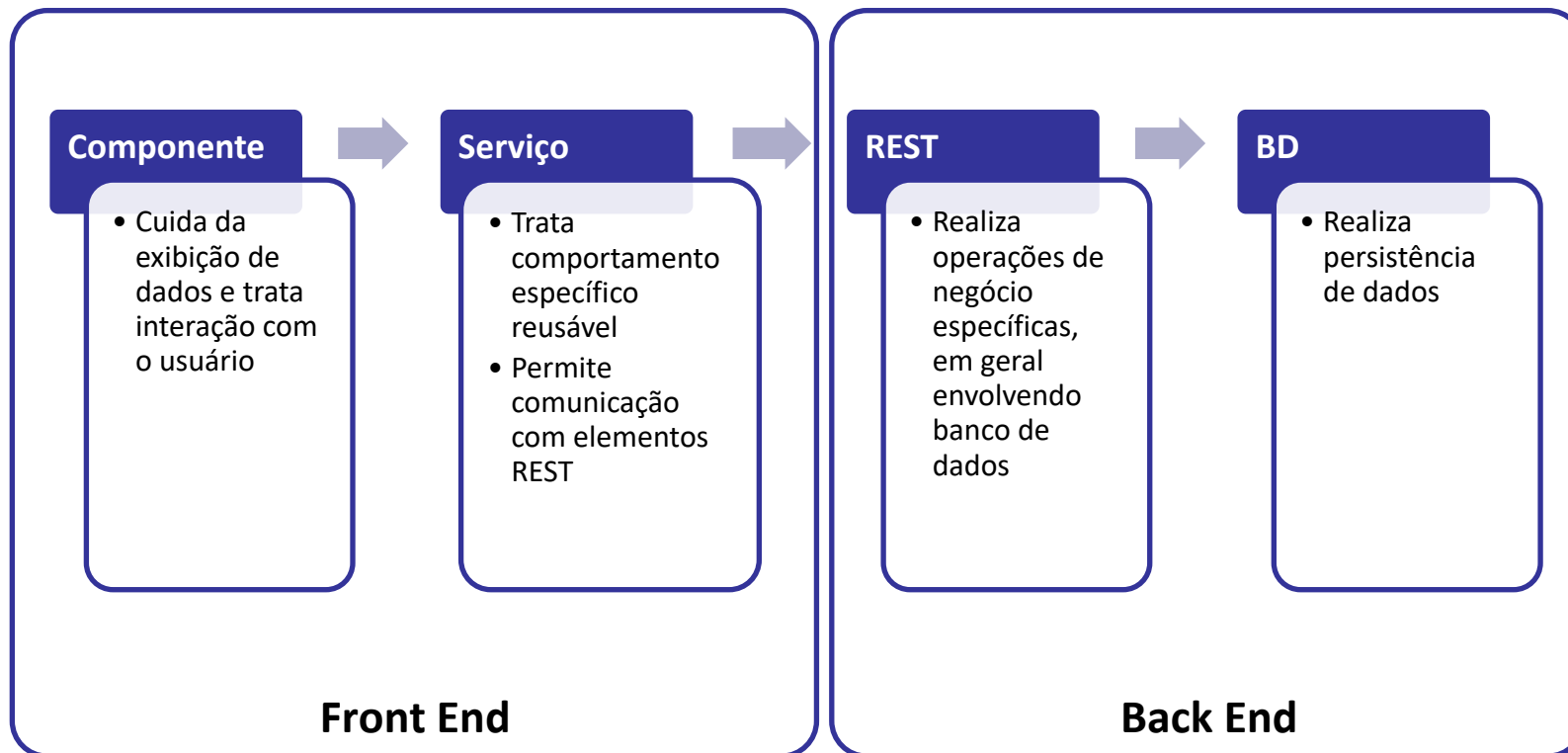
Serviços

- Ao separar funcionalidade mais específicas (serviços) de visualização e comportamento de interface (componentes), o angular promove o reuso de código, a separação de camadas e a facilidade de manutenção de elementos

Serviços

- Normalmente, utilizamos serviços para chamadas diretas a serviços REST remotos (por exemplo, PHP, Node ou Java)

Visão Geral de SPAs

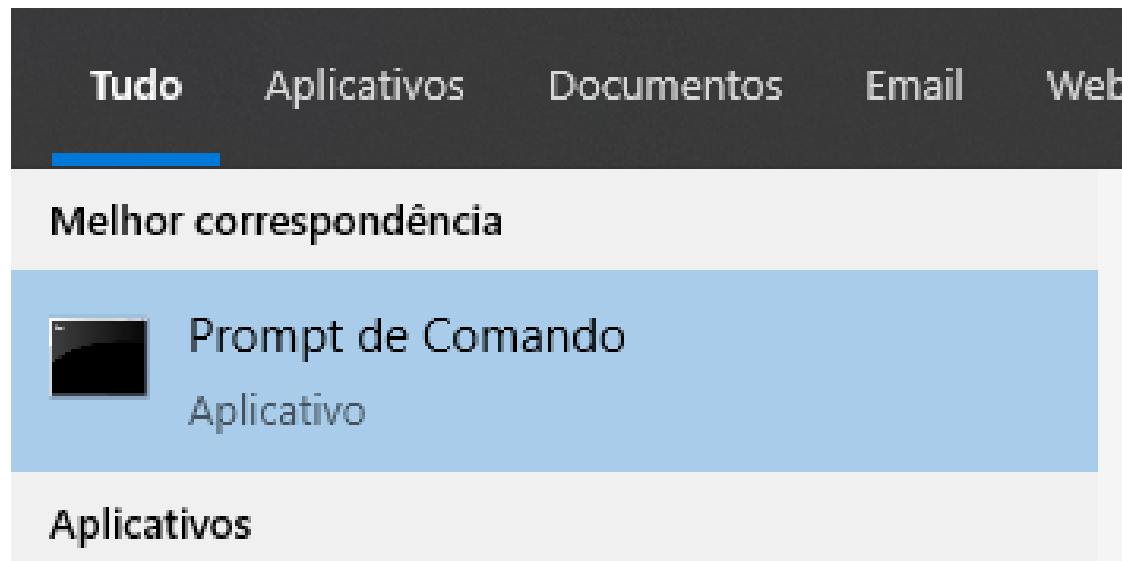


Aplicação SPA (Single-Page Application ou Aplicação de Página Única)

CRIANDO UM PROJETO ANGULAR

Criando um projeto Angular

- Vamos criar um projeto Angular
- Para isso, abra o **Prompt de Comando**



Criando um projeto Angular

- No prompt de comando, digite

```
npm install -g @angular/cli
```

Criando um projeto Angular

- Este comando utiliza o gerenciador de dependências do Node.js (o NPM) para instalar ferramentas de linha de comando do Angular
- As ferramentas de linha de comando do Angular são acionadas pelo comando **ng**

Criando um projeto Angular

```
C:\> Prompt de Comando

Microsoft Windows [versão 10.0.18362.657]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

C:\Users\abenza>npm install -g @angular/cli
[sill resolveWithNewModule JSONStream@1.3.5 checking
installabl[sill resolveWithNewModule JSONStream@1.3
.5 checking in[sill resolveWithNewModule through@2.3
.8 checking [sill resolveWithNewModule genfun@5.0.
0 ch[sill resolveWithNewModule fs-minipass@1.2.[sill
.....] \ loadDep:through: sill resolveWithNewModule semver@5.7.1 chenpm WARN dep
recated request@2.88.2: request has been deprecated, see https://github.com/request/reque
st/issues/3142
C:\Users\abenza\AppData\Roaming\npm\ng -> C:\Users\abenza\AppData\Roaming\npm\node_module
s\@angular\cli\bin\ng

> @angular/cli@9.0.4 postinstall C:\Users\abenza\AppData\Roaming\npm\node_modules\@angula
r\cli
> node ./bin/postinstall/script.js

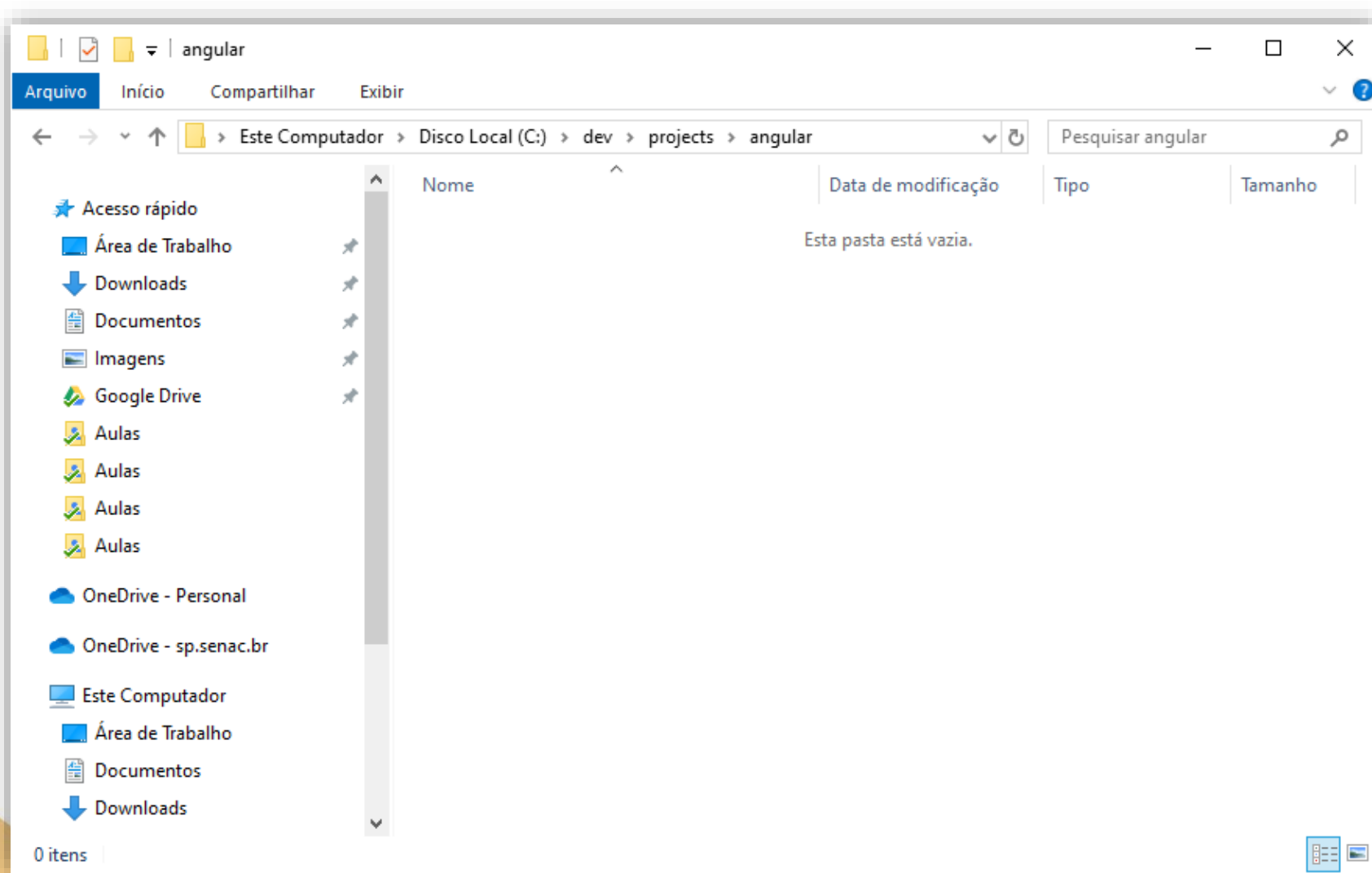
+ @angular/cli@9.0.4
added 12 packages from 23 contributors, removed 3 packages and updated 26 packages in 15.
189s

C:\Users\abenza>
```

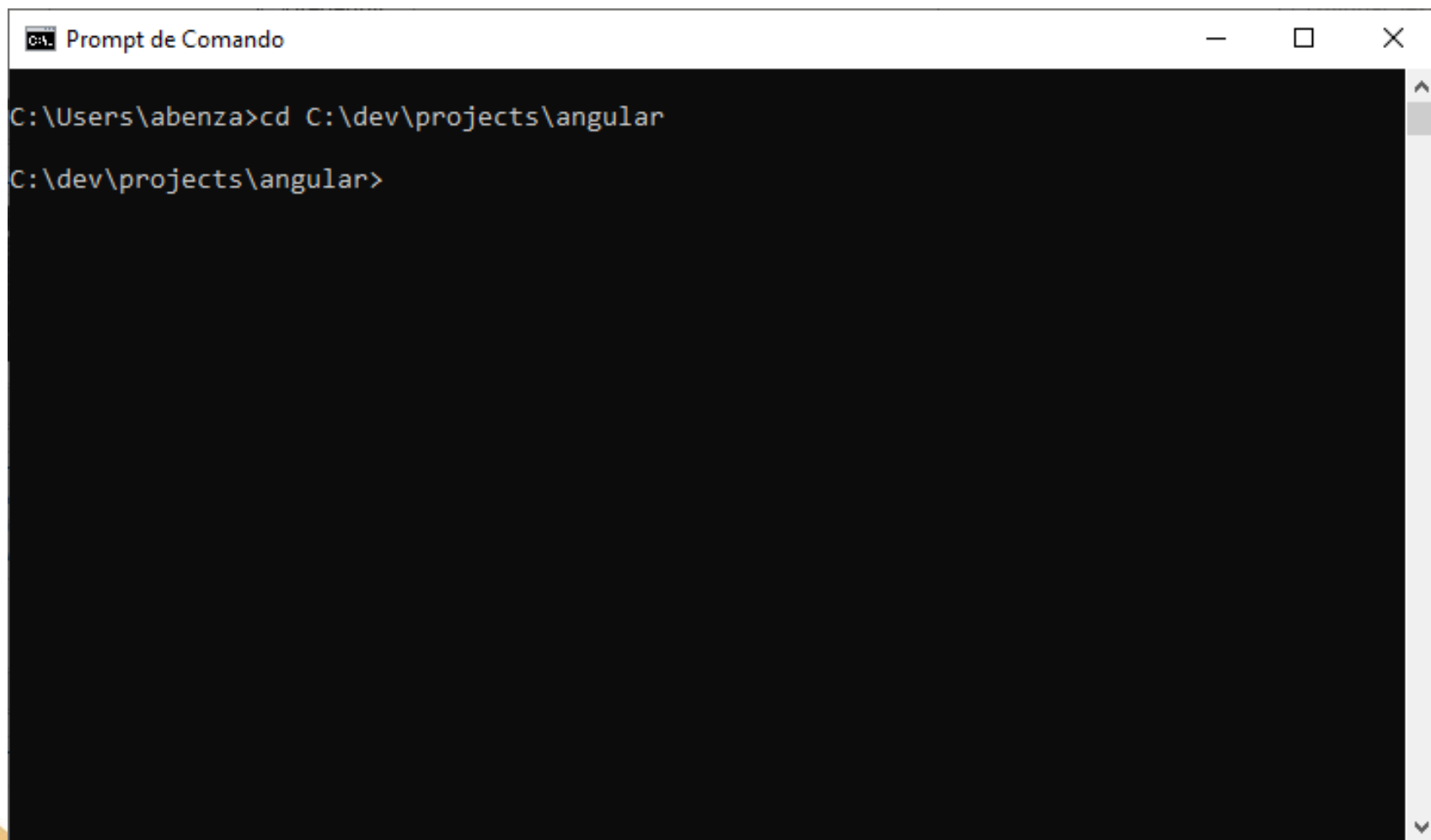
Criando um projeto Angular

- Com o **angular-cli** instalado, podemos solicitar a criação de um projeto Angular
- Antes disso, crie uma pasta para centralizar os projetos no local desejado e utilize o comando **cd** no prompt para chegar até ele

Criando um projeto Angular



Criando um projeto Angular



A screenshot of a Windows Command Prompt window titled "Prompt de Comando". The window has a black background and white text. The command prompt shows the user navigating to a directory. The first line shows the user at the C:\Users\abenza prompt, typing "cd C:\dev\projects\angular". The second line shows the user at the C:\dev\projects\angular prompt, ready for the next command. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

```
C:\Users\abenza>cd C:\dev\projects\angular  
C:\dev\projects\angular>
```

Criando um projeto Angular

- Para criar um projeto Angular na pasta indicada usando o angular-cli, digite o comando abaixo no prompt:

```
ng new aplicativo1
```

- Responda sim (y) e aceite os demais padrões pressionando **Enter**

Criando um projeto Angular

```
C:\Users\abenza>cd C:\dev\projects\angular

C:\dev\projects\angular>ng new aplicativo1
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE aplicativo1/angular.json (3607 bytes)
CREATE aplicativo1/package.json (1288 bytes)
CREATE aplicativo1/README.md (1028 bytes)
CREATE aplicativo1/tsconfig.json (543 bytes)
CREATE aplicativo1/tslint.json (1953 bytes)
CREATE aplicativo1/.editorconfig (246 bytes)
CREATE aplicativo1/.gitignore (631 bytes)
CREATE aplicativo1/browserslist (429 bytes)
CREATE aplicativo1/karma.conf.js (1023 bytes)
CREATE aplicativo1/tsconfig.app.json (210 bytes)
CREATE aplicativo1/tsconfig.spec.json (270 bytes)
CREATE aplicativo1/src/favicon.ico (948 bytes)
CREATE aplicativo1/src/index.html (297 bytes)
CREATE aplicativo1/src/main.ts (372 bytes)
CREATE aplicativo1/src/polyfills.ts (2835 bytes)
CREATE aplicativo1/src/styles.css (80 bytes)
CREATE aplicativo1/src/test.ts (753 bytes)
CREATE aplicativo1/src/assets/.gitkeep (0 bytes)
```


Subindo um projeto Angular

- Para executar o projeto criado, podemos utilizar um servidor de testes disponibilizado pelas ferramentas de linha de comando do Angular

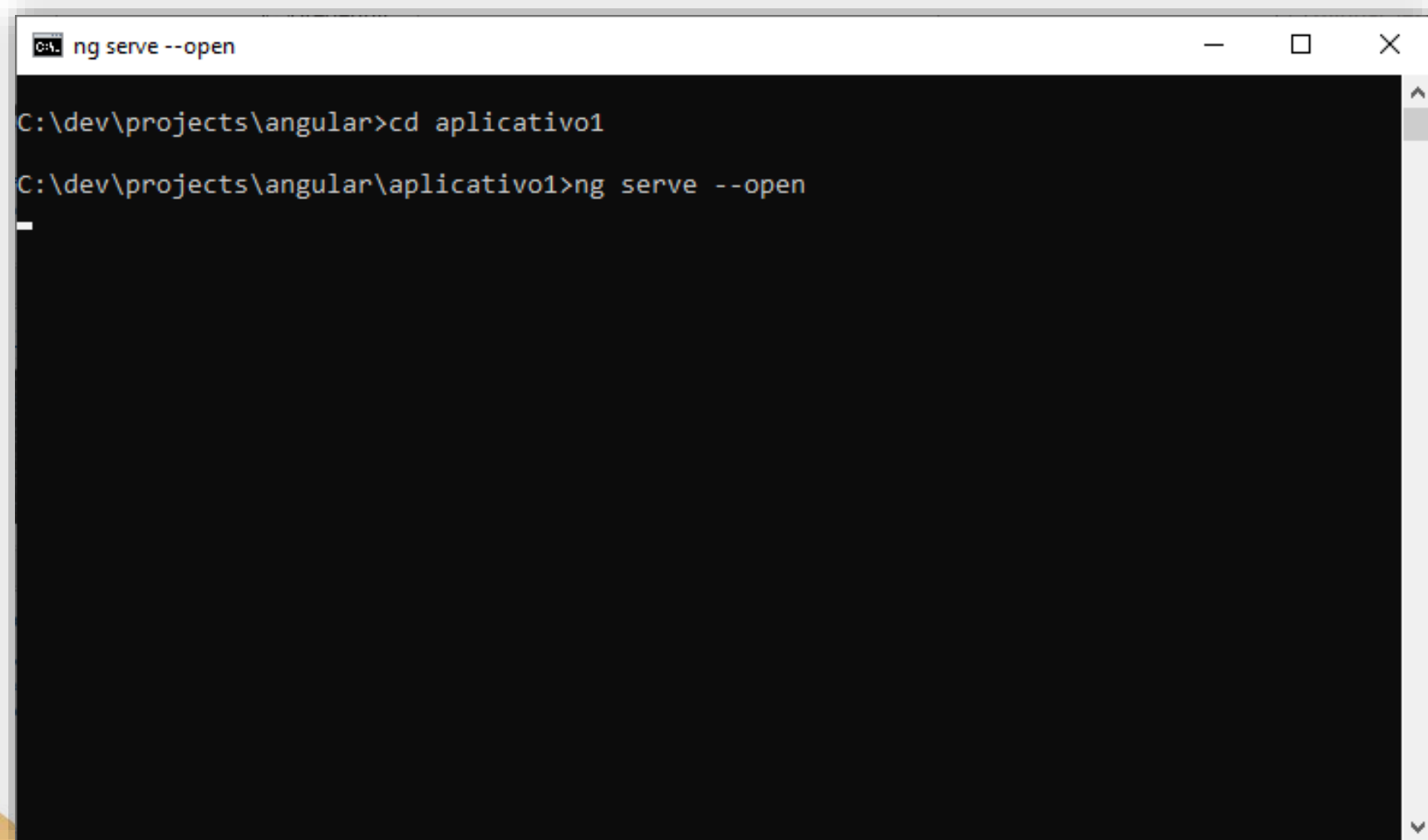
Subindo um projeto Angular

- Para isso, entre no diretório do projeto (**cd**) e execute o comando *ng serve --open*:

```
cd aplicativo1
```

```
ng serve --open
```

Subindo um projeto Angular

A screenshot of a Windows Command Prompt window. The title bar at the top reads "C:\> ng serve --open". The window has standard Windows window controls (minimize, maximize, close) on the right. The main area is black with white text. The first line shows the command "C:\dev\projects\angular>cd aplicativo1". The second line shows "C:\dev\projects\angular\aplicativo1>ng serve --open". A small white cursor is visible on the line following the command.

```
C:\dev\projects\angular>cd aplicativo1
C:\dev\projects\angular\aplicativo1>ng serve --open
-
```

Subindo um projeto Angular

```
C:\> ng serve --open

Compiling @angular/common : es2015 as esm2015

Compiling @angular/platform-browser : es2015 as esm2015

Compiling @angular/platform-browser-dynamic : es2015 as esm2015

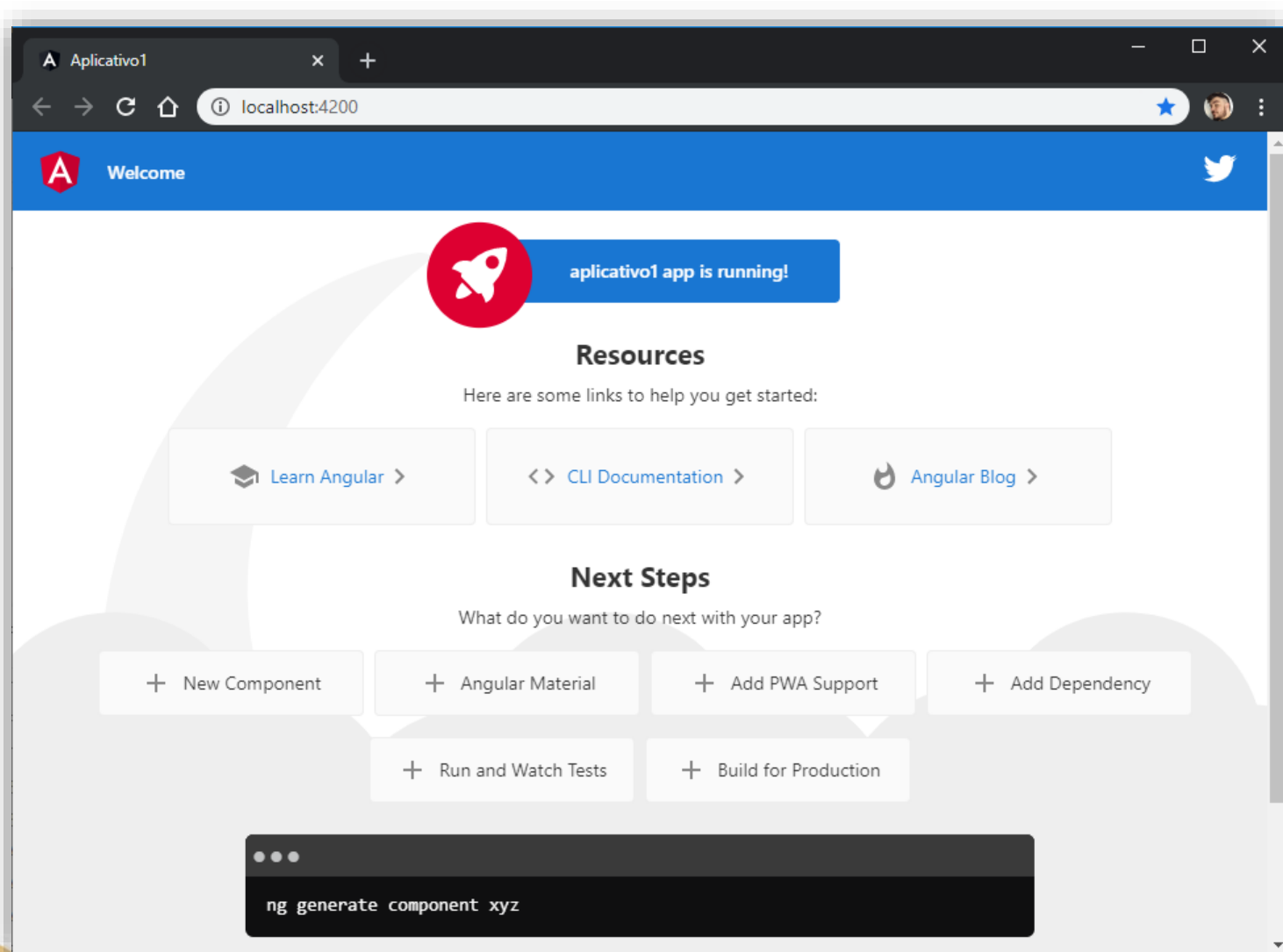
Compiling @angular/router : es2015 as esm2015

chunk {main} main.js, main.js.map (main) 60.6 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 140 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 9.71 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 2.99 MB [initial] [rendered]
Date: 2020-03-03T01:27:13.049Z - Hash: 0198dce9124d69ea6a29 - Time: 22220ms
** Angular Live Development Server is listening on localhost:4200, open your browser on h
ttp://localhost:4200/ **
: Compiled successfully.

Date: 2020-03-03T01:27:14.664Z - Hash: 0198dce9124d69ea6a29
5 unchanged chunks

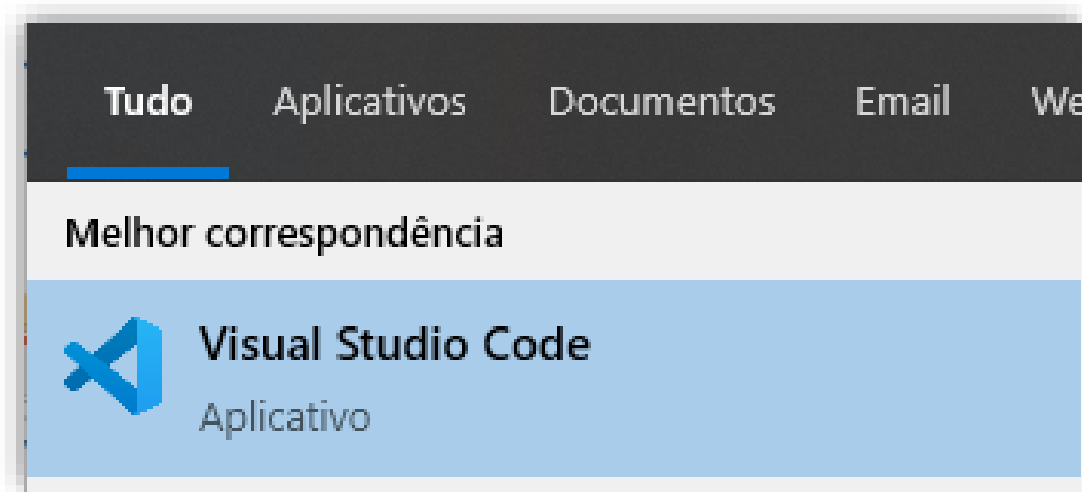
Time: 1181ms
: Compiled successfully.
```

Aplicativo Angular iniciado



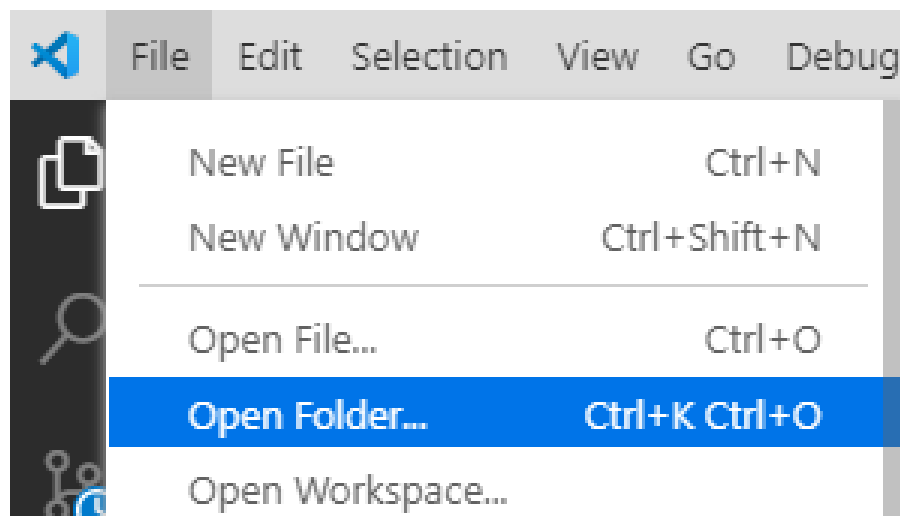
Abrindo o Projeto no VS Code

- Com o projeto criado, vamos abrir sua pasta no Visual Studio Code
- Para tanto, é necessário abrir o editor

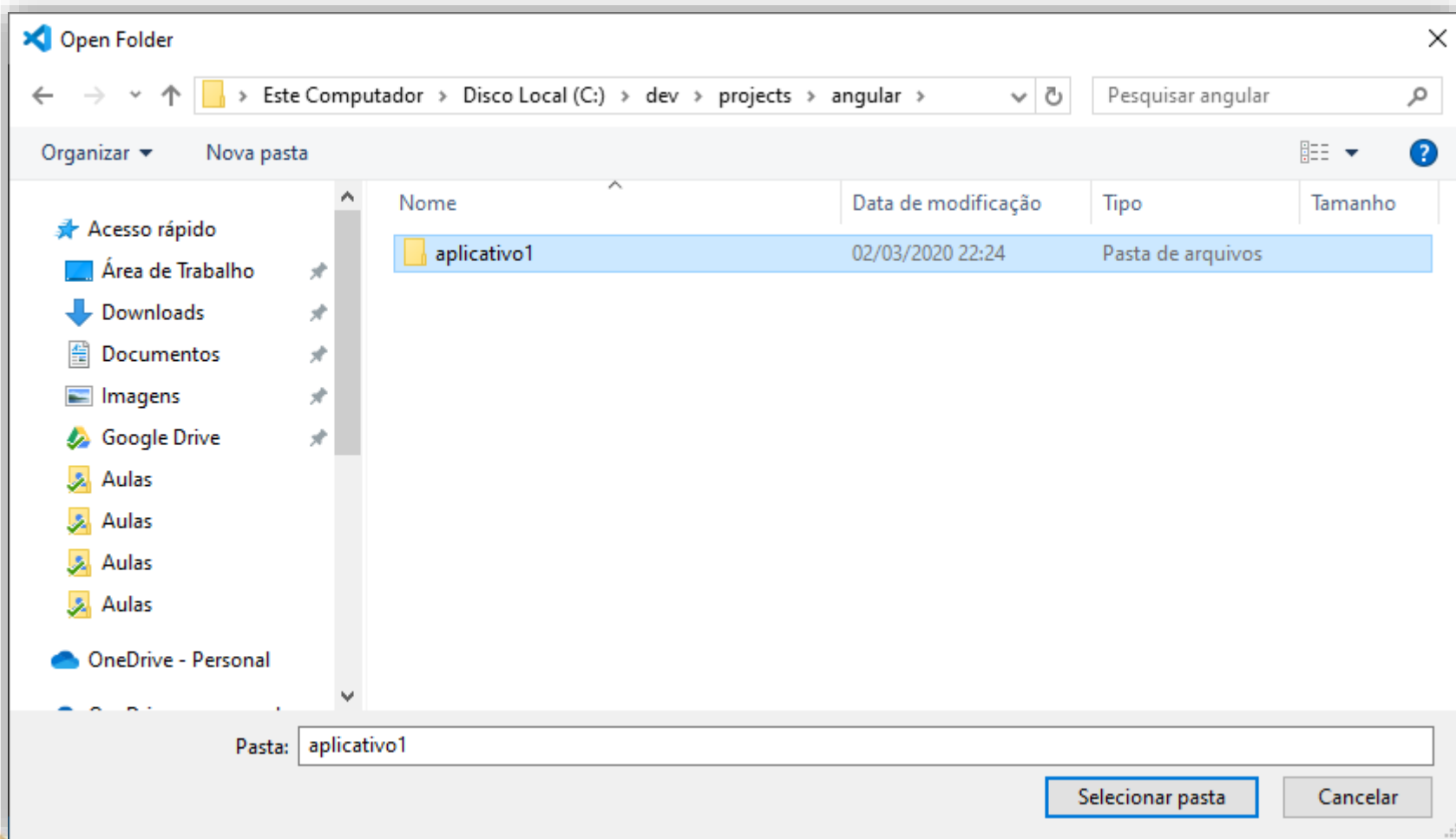


Abrindo o Projeto no VS Code

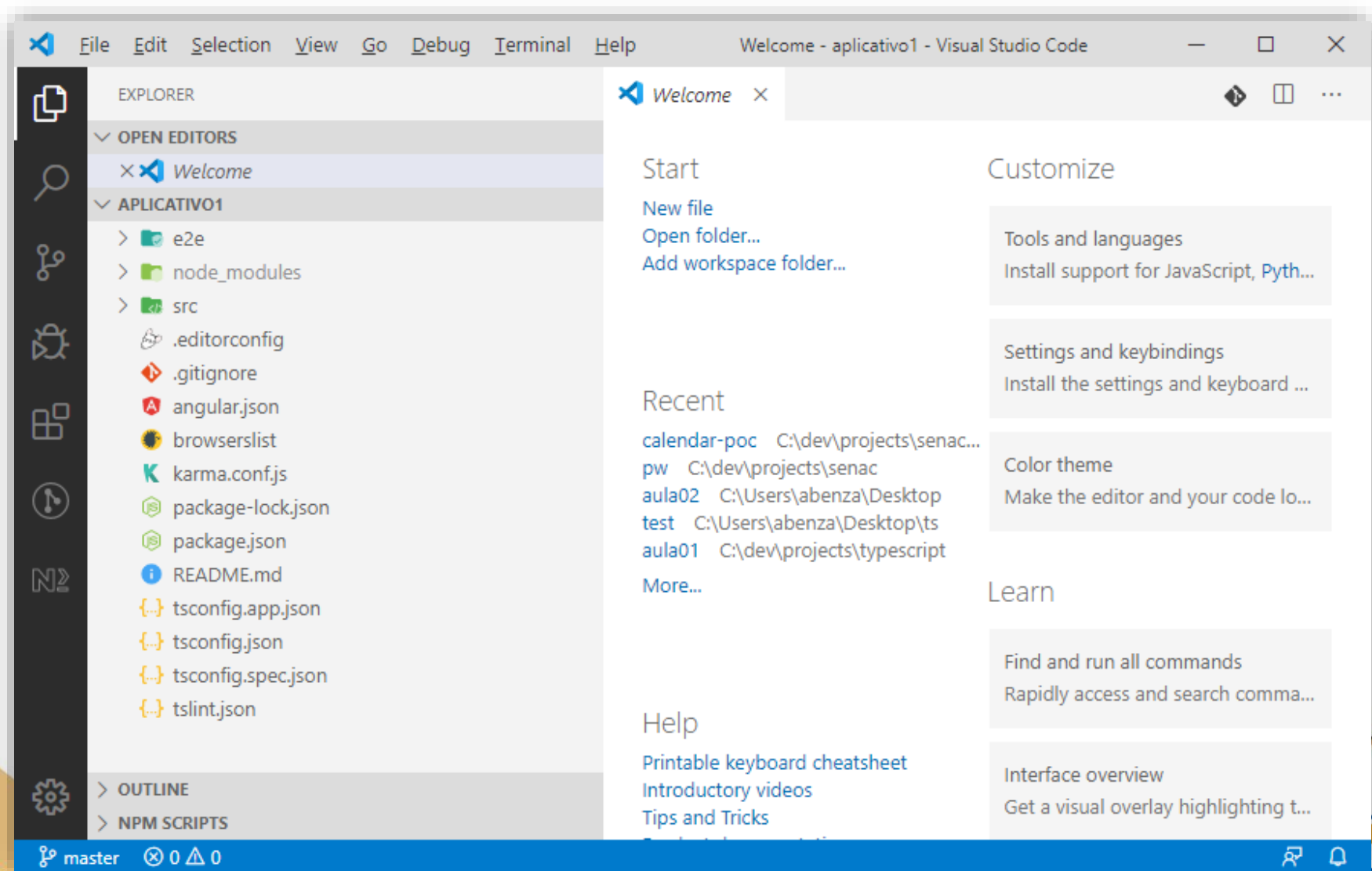
- No Visual Studio Code, vá até **File > Open Folder**
- Aponte para a pasta do projeto criado anteriormente e clique em **Selecionar pasta**



Abrindo o Projeto no VS Code



Projeto no VS Code



Entregando uma aplicação

- Salvamos e enviamos para o git o projeto sem a pasta de módulos de bibliotecas NPM (node_modules) utilizadas pelo Angular, pois a mesma pode ocupar gigabytes de espaço
- Ao entregar ou subir um projeto Angular, zip o projeto **excluindo a pasta node_modules**

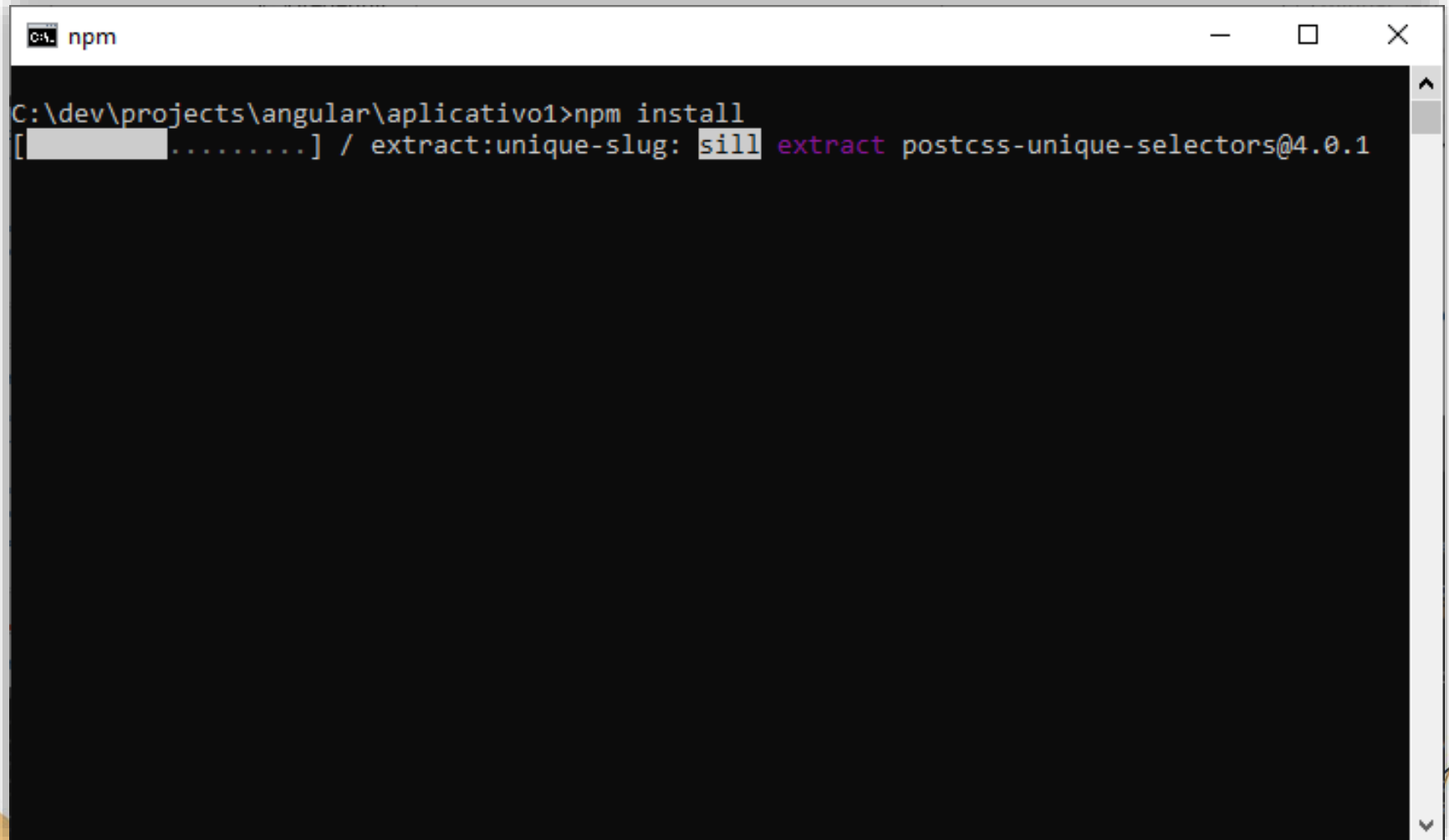
Restaurando uma aplicação

- Caso a pasta node_modules seja excluída, é necessário restaurá-la para que o mesmo torne-se executável novamente
- Para isso, basta executar novamente o comando npm install na pasta do projeto num prompt de comando

```
cd C:\pasta\onde\está\o\projeto
```

```
npm install
```

Restaurando uma aplicação



```
C:\dev\projects\angular\aplicativo1>npm install
[.....] / extract:unique-slug: sill extract postcss-unique-selectors@4.0.1
```

The image shows a Windows command prompt window titled 'C:\dev\ npm'. The prompt is at 'C:\dev\projects\angular\aplicativo1>' and the command 'npm install' has been executed. The output shows a progress bar '[.....]' followed by the log message '/ extract:unique-slug: sill extract postcss-unique-selectors@4.0.1'. The window has standard Windows window controls (minimize, maximize, close) in the top right corner.

Restaurando uma aplicação

C:\ Prompt de Comando

r\cli

> node ./bin/postinstall/script.js

npm **WARN** optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.11 (node_modules\webpack-dev-server\node_modules\fsevents):

npm **WARN** notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.11: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

npm **WARN** optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.11 (node_modules\webpack\node_modules\fsevents):

npm **WARN** notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.11: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

npm **WARN** optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.1.2 (node_modules\fsevents):

npm **WARN** notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.1.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

added 1319 packages from 1113 contributors and audited 15574 packages in 55.519s

35 packages are looking for funding

run `npm fund` for details

found 0 vulnerabilities

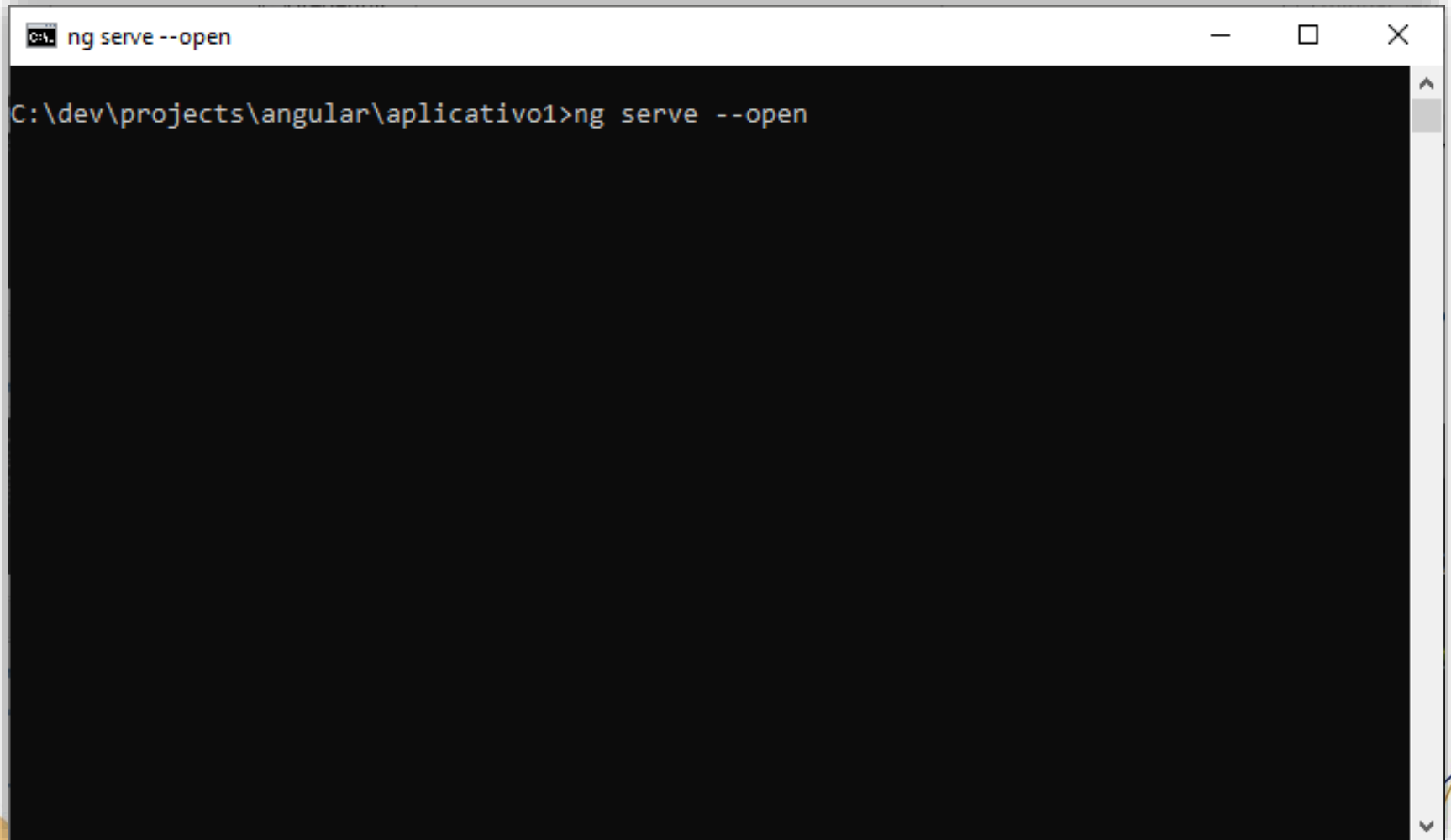
C:\dev\projects\angular\aplicativo1>

Restaurando uma aplicação

- Para subir novamente a aplicação no servidor de testes do Angular, execute o comando abaixo na pasta do projeto

```
ng serve --open
```

Restaurando uma aplicação



A screenshot of a Windows command prompt window. The title bar at the top reads "C:\> ng serve --open". The window has standard Windows window controls (minimize, maximize, close) on the right. The main area is black with white text. The prompt "C:\dev\projects\angular\aplicativo1>" is followed by the command "ng serve --open". The rest of the terminal area is empty.

```
C:\dev\projects\angular\aplicativo1>ng serve --open
```

Restaurando uma aplicação

```
C:\> ng serve --open

Compiling @angular/common : es2015 as esm2015

Compiling @angular/platform-browser : es2015 as esm2015

Compiling @angular/platform-browser-dynamic : es2015 as esm2015

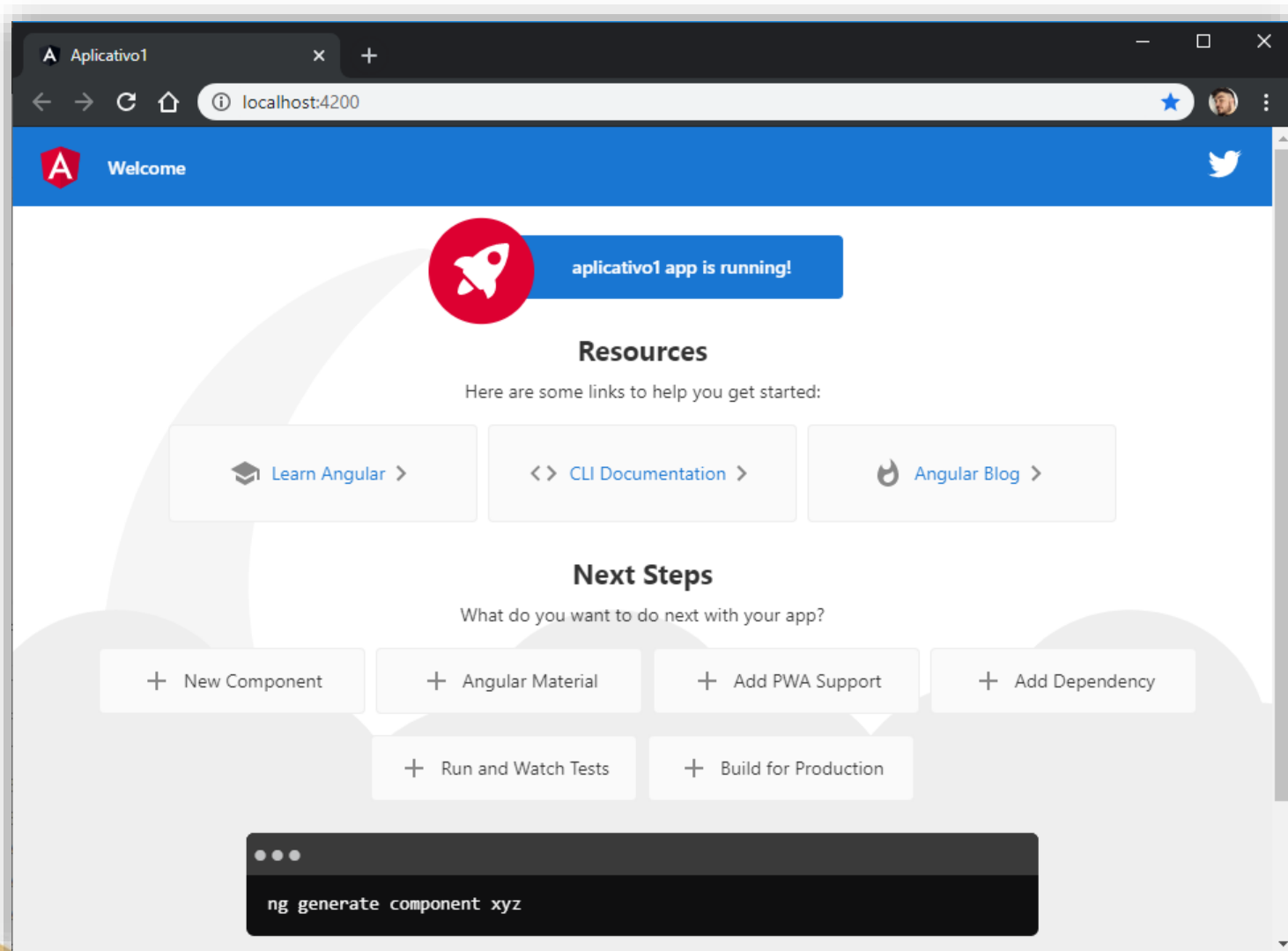
Compiling @angular/router : es2015 as esm2015

chunk {main} main.js, main.js.map (main) 60.6 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 140 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 9.71 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 2.99 MB [initial] [rendered]
Date: 2020-03-03T01:35:38.715Z - Hash: 0198dce9124d69ea6a29 - Time: 21739ms
** Angular Live Development Server is listening on localhost:4200, open your browser on h
ttp://localhost:4200/ **
: Compiled successfully.

Date: 2020-03-03T01:35:40.076Z - Hash: 0198dce9124d69ea6a29
5 unchanged chunks

Time: 941ms
: Compiled successfully.
```


Restaurando uma aplicação

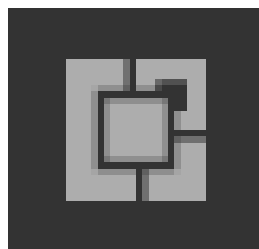


Instalando ferramentas auxiliares

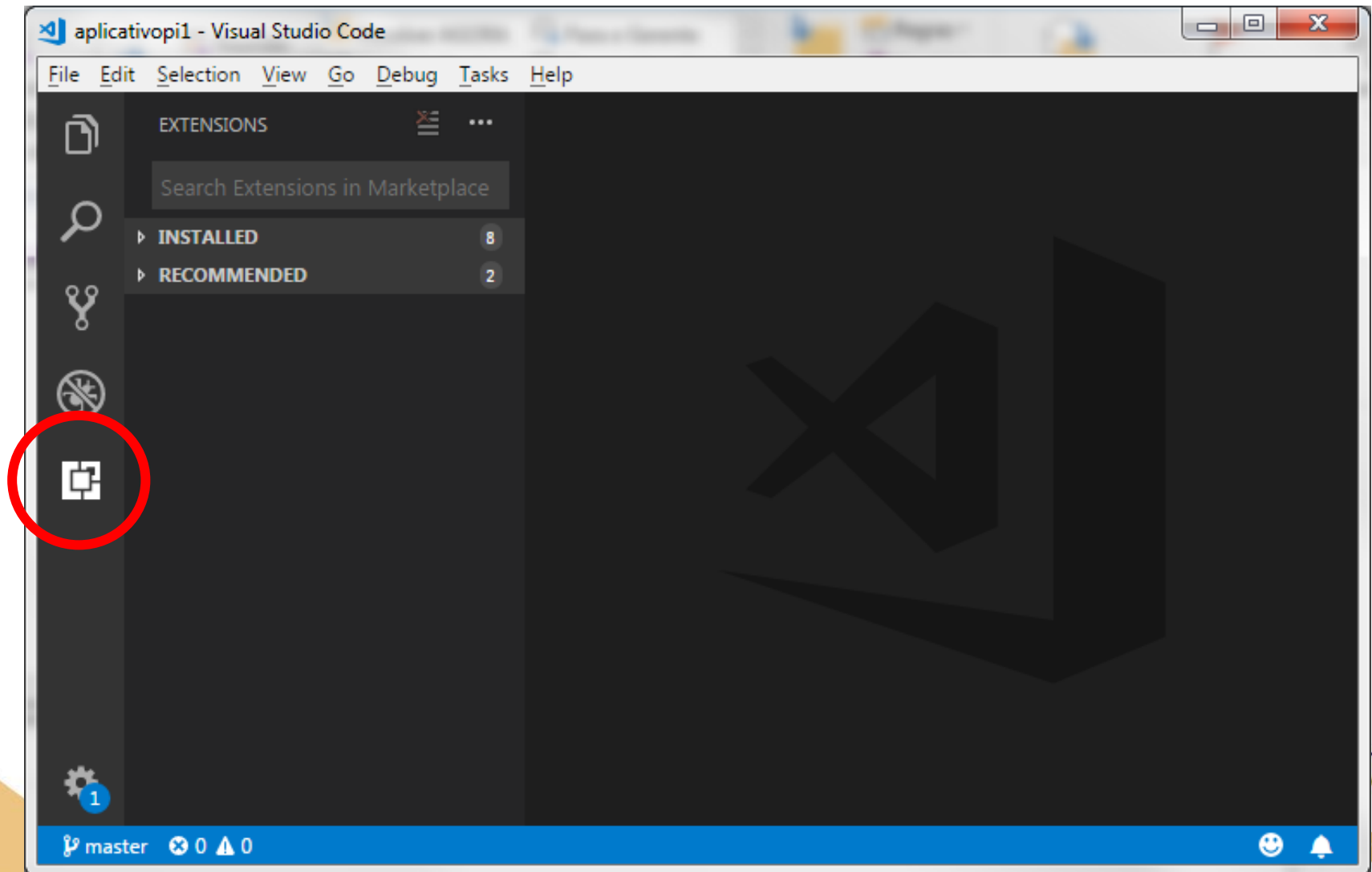
- O Visual Studio Code possui diversas extensões que podemos utilizar para melhorar a experiência de desenvolvimento com tecnologias específicas

Instalando ferramentas auxiliares

- Para gerenciar extensões, clique no ícone respectivo para exibir extensões instaladas e disponíveis







Instalando ferramentas auxiliares







Instalando ferramentas auxiliares


- Instale as extensões a seguir (utilizando a caixa de pesquisa e clicando em **Install**)






Angular Snippets (Version 11) 11.0.0  2.3M  5
Angular version 11 snippets by John Papa
John Papa 



TSLint 1.2.3  1.7M  3
TSLint support for Visual Studio Code
Microsoft 



Material Icon Theme 4.0.1  3.5M  5
Material Design Icons for Visual Studio Code
Philipp Kief 

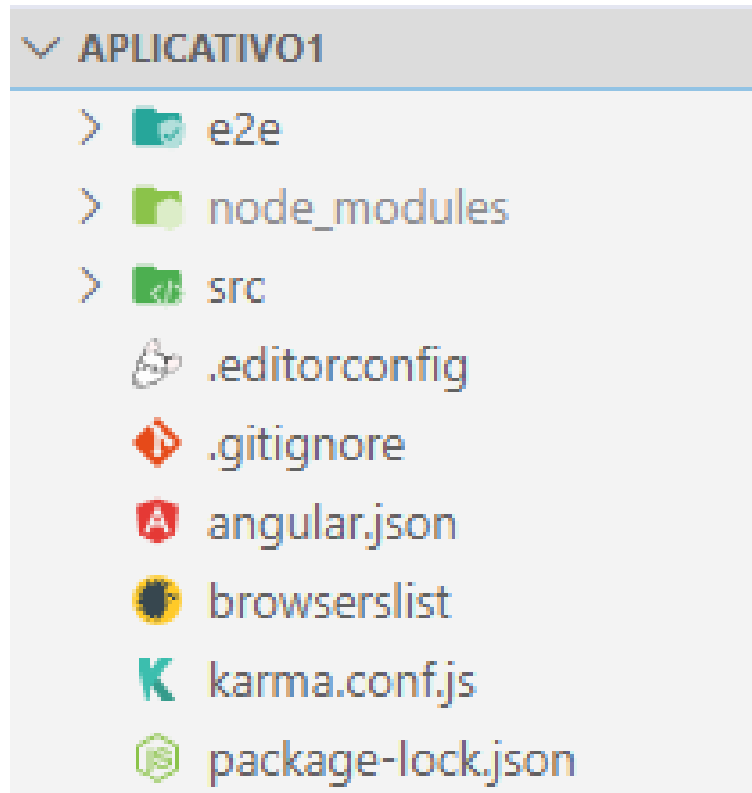
Estrutura do Projeto

Testes “end to end” (testes de integração)

Módulos do NPM

Pasta de códigos-fonte do Angular

Arquivos de configuração do projeto



Estrutura de códigos-fonte Angular

- Dentro da pasta “src”, podemos verificar a estrutura de elementos de código, template e estilo do Angular

Estrutura de códigos-fonte Angular

- A ferramenta “ng” cria automaticamente um componente padrão (“app”)
- Este componente é o principal elemento da aplicação. Nele, os demais componentes poderão ser inseridos

Estrutura da pasta “src”

Pasta de códigos-fonte do projeto Angular

Componente “app”

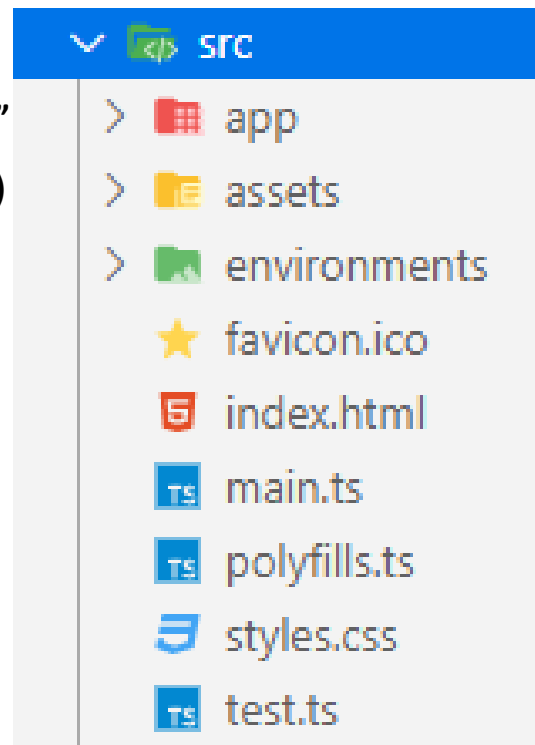
Pasta de recursos da aplicação (como imagens)

Pasta de configuração de ambientes

Ícone de favoritos e barra de abas

Página inicial, que carrega o componente principal

Arquivos de configuração específicos do Angular



Estrutura de um Componente

Pasta de códigos-fonte do projeto Angular

Componente “app”

Configurador de rotas

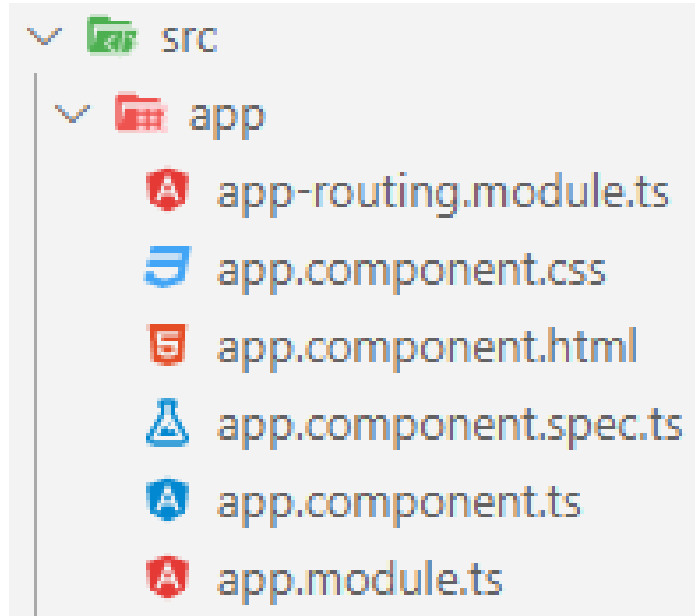
Arquivos de ESTILO do componente

Arquivo de TEMPLATE do componente

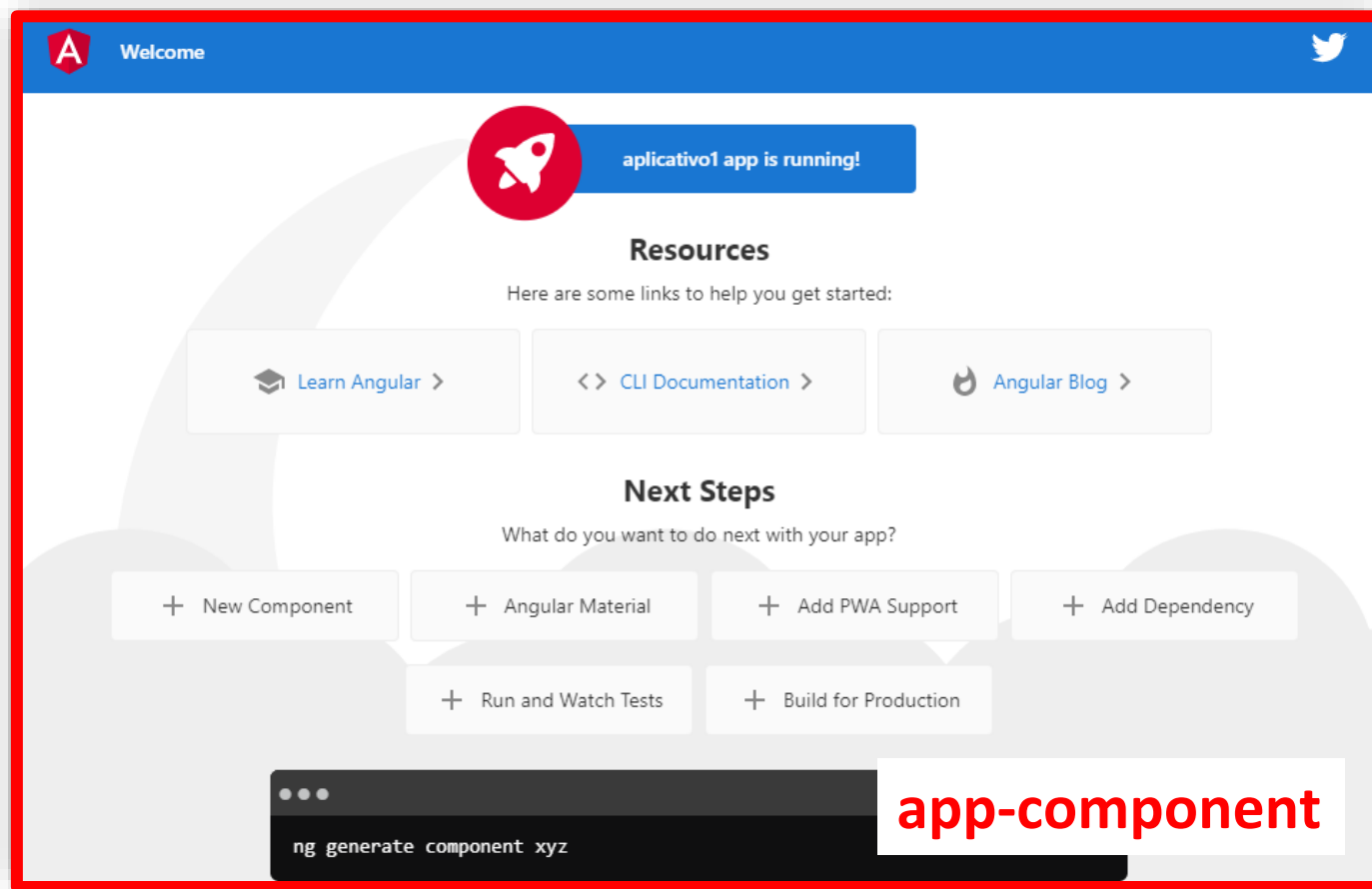
TESTES (gerenciado pelo Angular)

Principal arquivo de CÓDIGO-FONTE do componente

Módulo principal da aplicação



Estrutura de códigos-fonte Angular



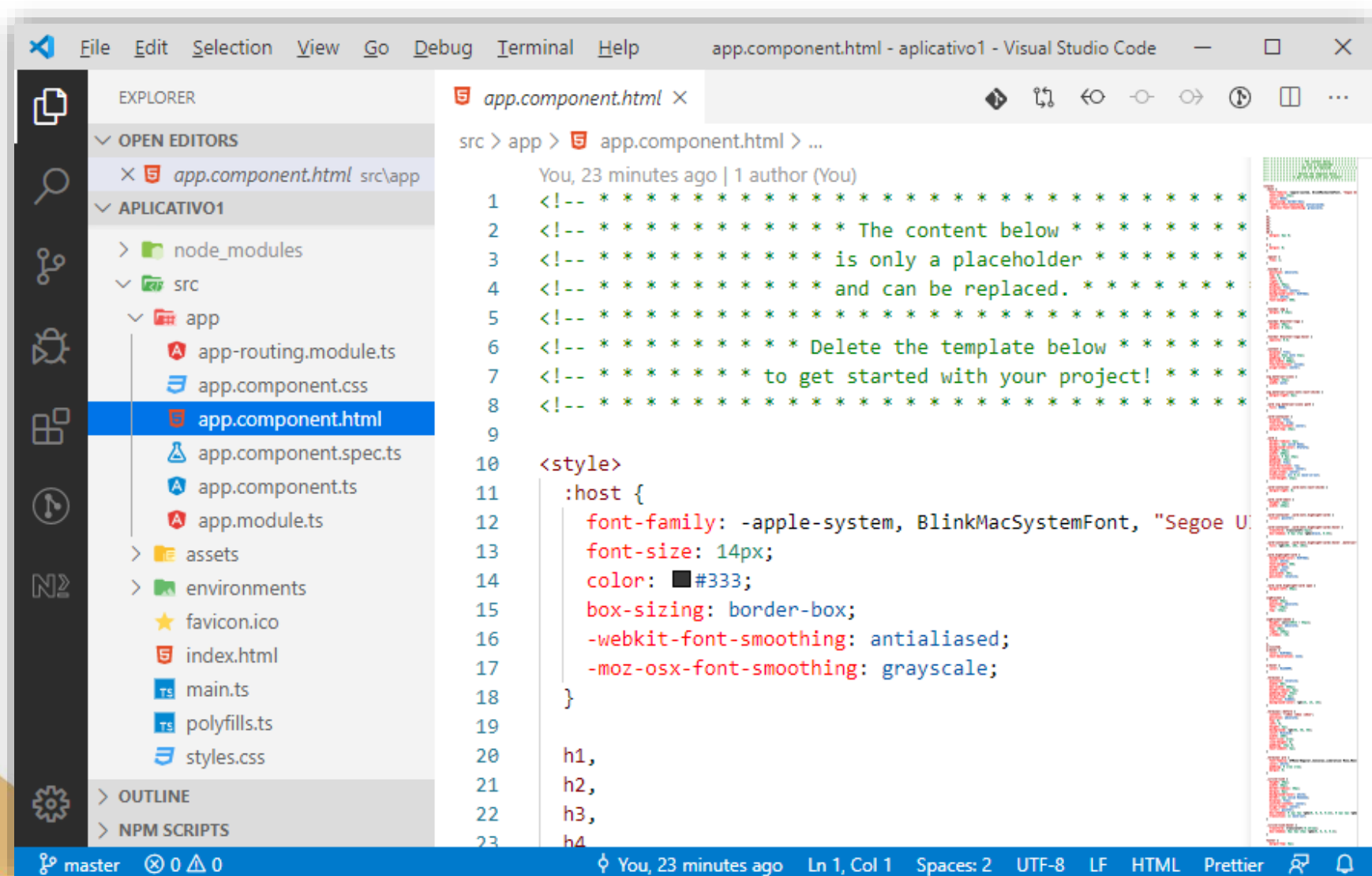
Componentes em Angular

- **Lembre-se!** Tudo em Angular é um componente
- O conceito de “link apontando para uma página HTML” é, em geral, obsoleto em Angular

Template do componente principal

- Abra o arquivo “**app.component.html**”, que representa o template do componente principal
- Este arquivo é um HTML simples e contém tags e elementos HTML padrão para estruturar o componente

Template do componente principal



Binding de elementos

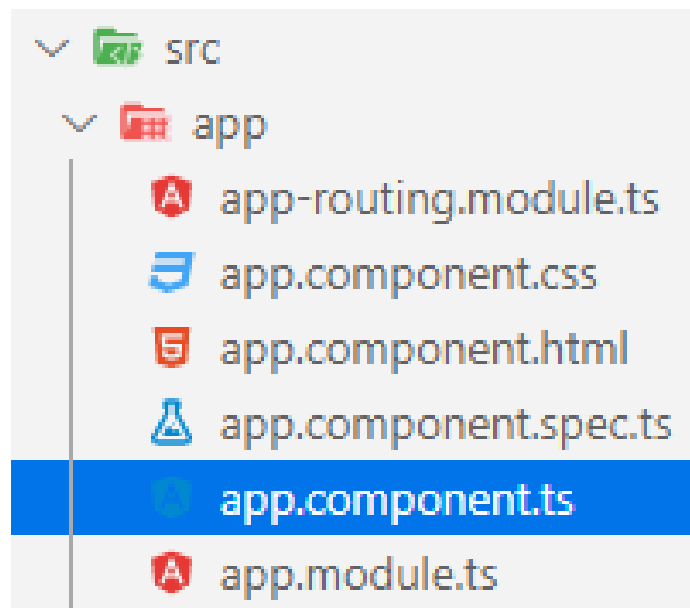
- Repare no elemento “{{ title }} app is running!”
(linha 345)
- Este elemento está utilizando “binding” para o código TypeScript (o elemento com duas chaves)

Binding de elementos

- Isto quer dizer que o elemento “`{{title}}`” será substituído pelo valor de uma propriedade da classe “`app.component.ts`” do componente

Classe do componente

- Abra o arquivo do componente
("app.component.ts")



Componente “app-component”

You, 25 minutes ago | 1 author (You)

```
1 import { Component } from '@angular/core';  
2
```

You, 25 minutes ago | 1 author (You)

```
3 ✓ @Component({  
4   selector: 'app-root',  
5   templateUrl: './app.component.html',  
6   styleUrls: ['./app.component.css']  
7 })  
8 ✓ export class AppComponent {  
9   title = 'aplicativo1';  
10 }
```

Estrutura de um componente

- O elemento da primeira linha representa um import
- imports em angular especificam quais arquivos externos devem ser referenciados (especificando a localização dos mesmos)

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

Anotação de componentes

- O elemento “@Component” declara um componente
- Ele permite especificar qual tag identifica o componente (utilizado para coloca-lo em outro template), o arquivo de template e o arquivo de CSS

Anotação de componentes

```
@Component({  
  selector: 'app-root',  
  templateUrl: './app.component.html',  
  styleUrls: ['./app.component.css']  
})
```

Classe do componente

- Em TypeScript, uma classe é declarada através do comando “export class”. Neste caso, está sendo declarada a classe “AppComponent”, a classe principal que representa o componente principal da aplicação

Classe do componente

```
export class AppComponent {  
  title = 'aplicativo1';  
}
```

Propriedade

- No interior da classe, há uma propriedade (ou atributo), “title”
- Esta propriedade é o elemento utilizado no HTML para exibir o texto atribuído na classe “aplicativo1”

Propriedade

```
export class AppComponent {  
  title = 'aplicativo1';  
}
```

Binding da Propriedade

Classe do componente (app.component.ts)

```
export class AppComponent {  
  title = 'aplicativo1';  
}
```

Template do componente (app.component.html)

```
<span>{{ title }} app is running!</span>
```

Alterando o valor do título

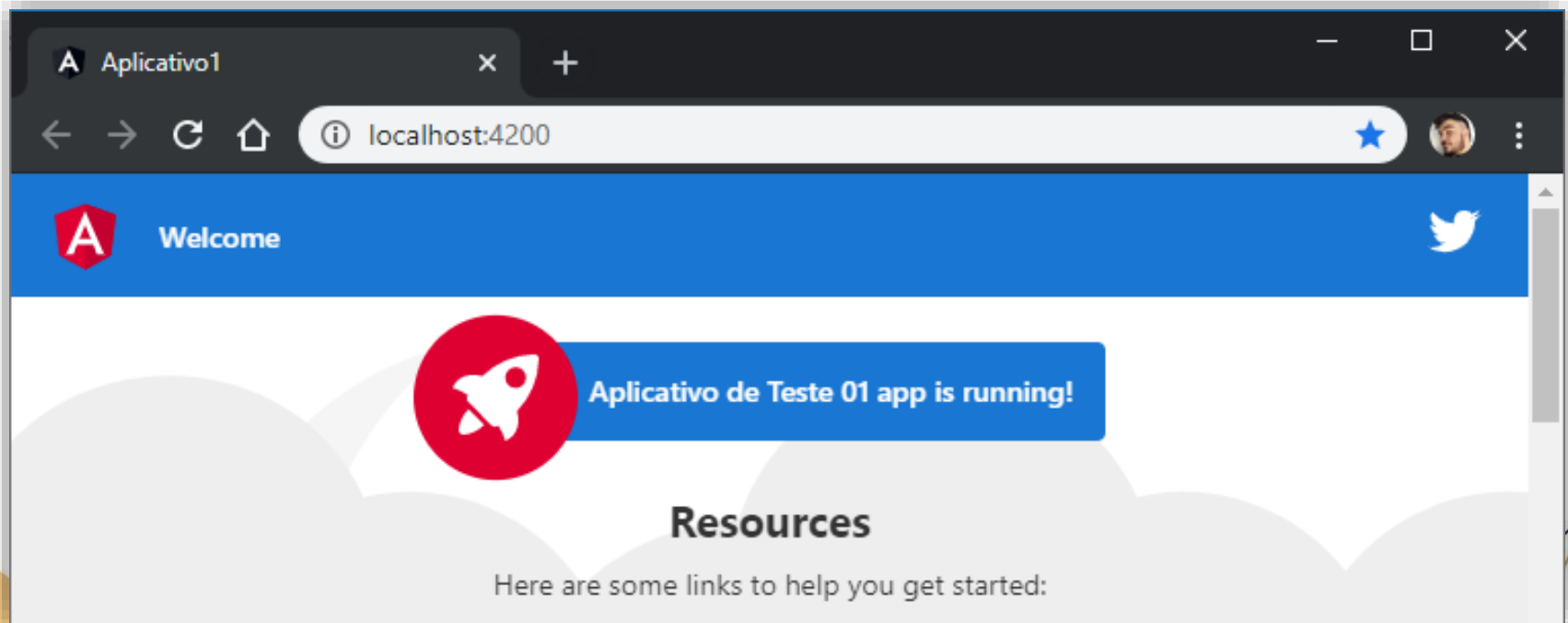
- Modifique o valor do “title” para “Aplicativo de Teste 01” (no arquivo “app.component.ts”) e salve o arquivo (“Ctrl + S”)

Alterando o valor do título

- Vá para o navegador e observe que as alterações são efetivadas instantaneamente (a aplicação deve exibir o título conforme especificado na propriedade)

Alterando o valor do título

```
export class AppComponent {  
  title = 'Aplicativo de Teste 01';  
}
```



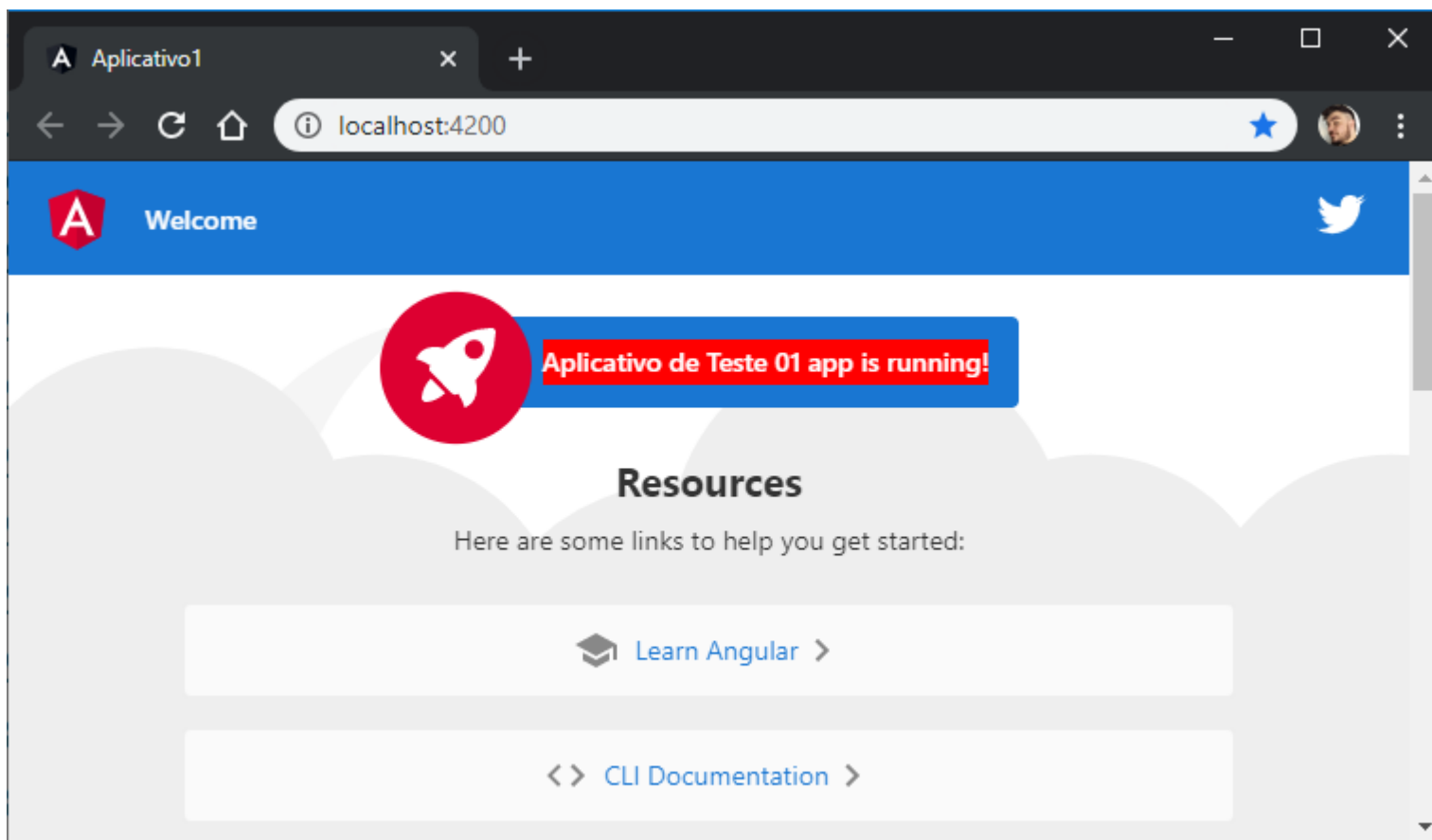
Atribuindo um estilo

- Vamos adicionar estilo aos elementos de título

```
<span class="spanvermelho">{{ title }} app is running!</span>
```

```
.spanvermelho {  
  background-color: ■ red;  
}
```

Atribuindo um estilo



Exercícios

1. Adicione uma nova propriedade a classe TypeScript (“frase”) contendo a frase “Meu primeiro app Angular!” e faça-a aparecer no template
2. Modifique a fonte e a cor do título para outro elemento de sua preferência
3. Adicione + dois links para páginas de sua escolha

CRIANDO UM COMPONENTE ANGULAR

Criando um componente

- Angular especifica que todos os itens visuais são componentes e, desta forma, possibilita criar novos destes elementos
- A CLI possui comandos que automatizam a criação de elementos do Angular, inclusive componentes

Criando um componente

- Para criar elementos em angular, é possível utilizar o comando **ng generate**

Criando um componente

- Para criar um componente, é necessário executar o comando **ng generate component pastas/nome**

Criando um componente

- Grande parte dos comandos do angular-cli pode ser simplificada utilizando apenas a primeira letra de cada palavra
- No exemplo anterior, teríamos **ng g c pastas/nome**

Criando um componente

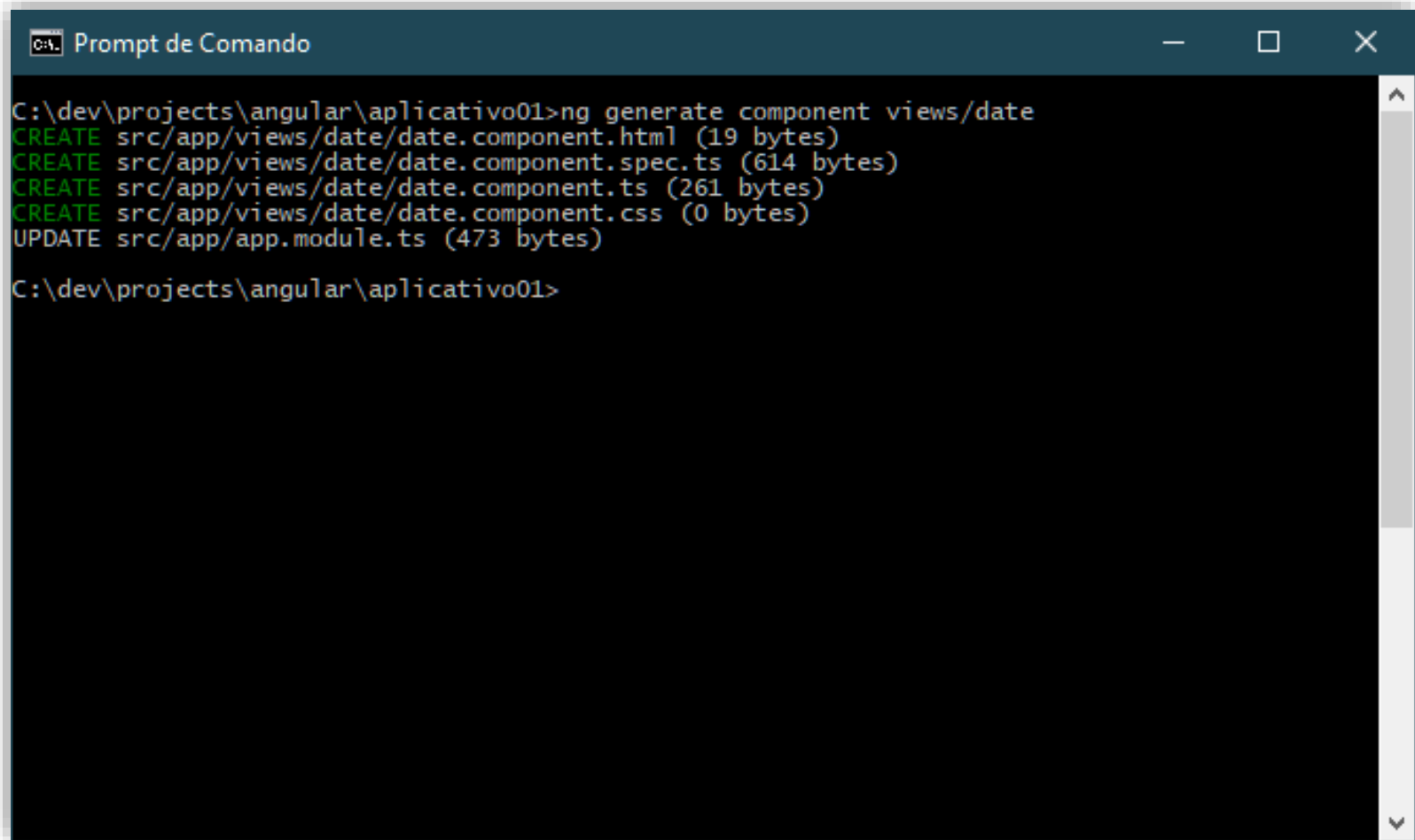
- Vamos criar um componente de exibição de data simples. Para tanto, execute o comando a seguir (estando na pasta do projeto, abra um novo prompt se preciso)

Criando um componente

`cd pasta_projeto` *(caso já não esteja)*

`ng generate component views/date`

Criando um componente



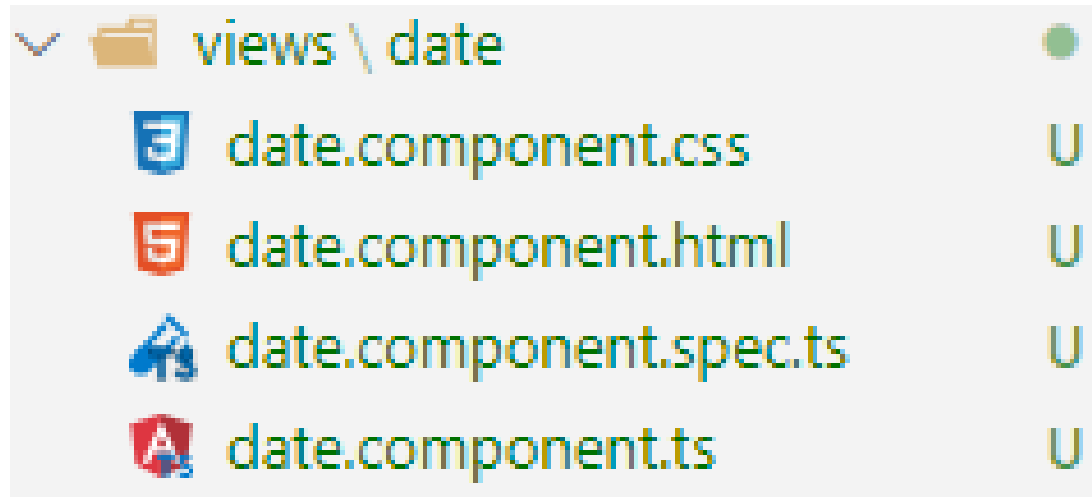
```
C:\dev\projects\angular\aplicativo01>ng generate component views/date
CREATE src/app/views/date/date.component.html (19 bytes)
CREATE src/app/views/date/date.component.spec.ts (614 bytes)
CREATE src/app/views/date/date.component.ts (261 bytes)
CREATE src/app/views/date/date.component.css (0 bytes)
UPDATE src/app/app.module.ts (473 bytes)

C:\dev\projects\angular\aplicativo01>
```


Componente criado

- Quando o comando for concluído, o componente será criado em uma pasta respectiva (e a configuração necessária para seu funcionamento será adicionada)

Componente criado



Elementos do componente

Template

 *date.component.html* ✕

src > app > views > date >  *date.component.html* >  p

1  **p**date works!</p>

2

CSS

 *date.component.css* ✕

src > app > views > date >  *date.component.css*

1

Elementos do componente - código

```
date.component.ts ×
src > app > views > date > date.component.ts > ...
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-date',
5    templateUrl: './date.component.html',
6    styleUrls: ['./date.component.css']
7  })
8  export class DateComponent implements OnInit {
9
10     constructor() { }
11
12     ngOnInit() {
13     }
14
15  }
16
```

Referência a classe de negócio

- Para ter uma variável que represente a data, faremos como a seguir:

Referência a classe de negócio

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

@Component({
  selector: 'app-date',
  templateUrl: './date.component.html',
  styleUrls: ['./date.component.css']
})
export class DateComponent implements OnInit {

  dataAtual: Date = new Date();

  constructor() { }

  ngOnInit(): void {
    this.dataAtual = new Date();
  }
}
```

ngOnInit

- Callback Angular executado quando o componente é inicializado para exibição
- **Executado todas as vezes que o componente é exibido**

constructor

- Construtor da classe do componente
- **Como o Angular instancia os componentes apenas uma vez, o construtor é chamado apenas uma vez**

Binding na página

- O atributo permitirá referenciar a data na página e tornará possível exibi-la no template (HTML)

Referência no template

 date.component.html ×

src > app > views > date >  date.component.html

```
1 {{dataAtual}}
```

Chamada ao componente

- Vá até o template do componente principal **app-component.html** e insira a chamada ao componente de cliente

Chamada ao componente

- Como vimos, este componente tem identificador app-date, então, é necessário adicionar esta tag no template principal

app-component.html

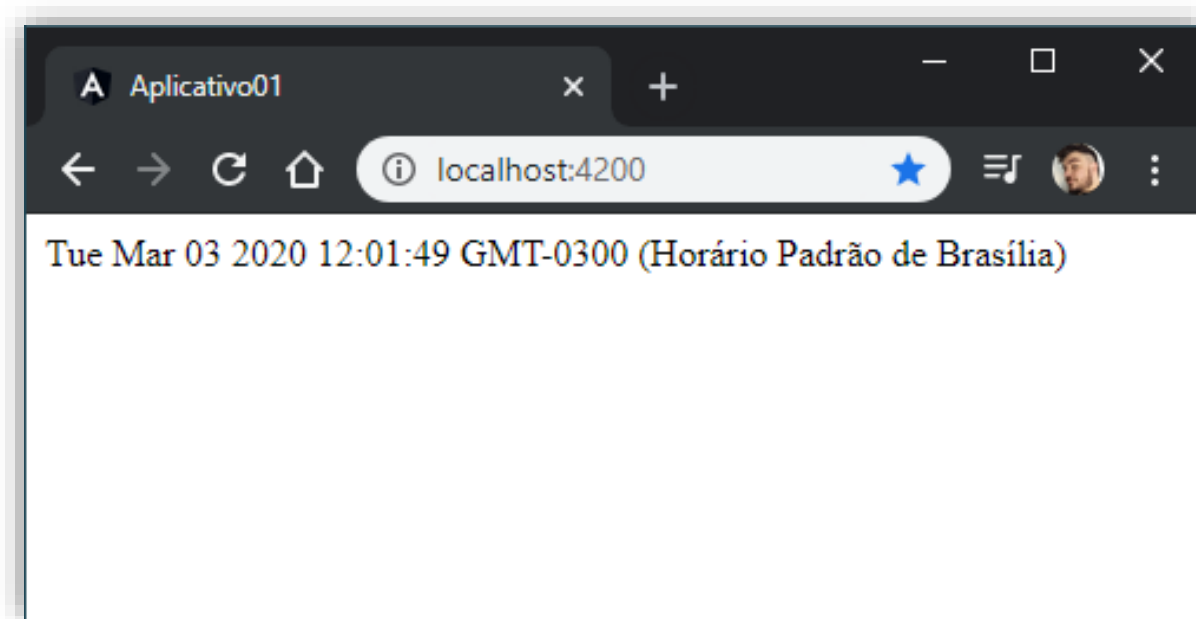
app.component.html ×

src > app > app.component.html > ...

```
1 <app-date></app-date>
```

Chamada ao componente

- Vá até a página com a aplicação em execução e verifique se as alterações estão funcionando



Pipes

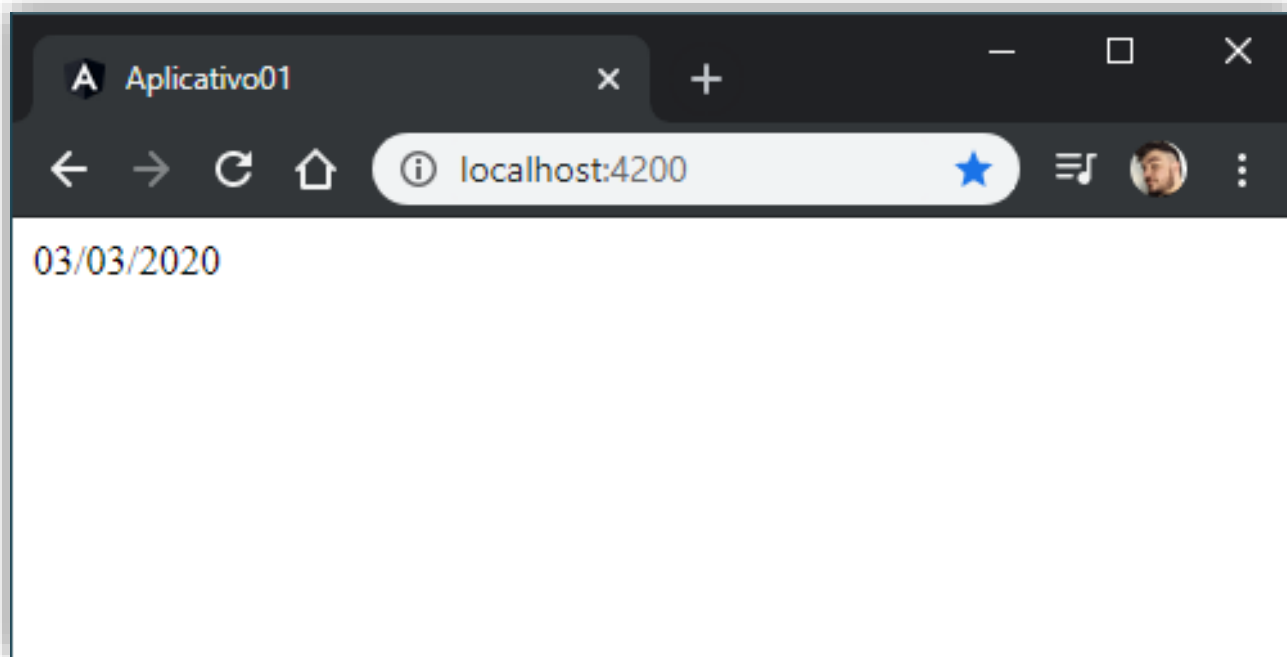
- Permitem formatar dados em Angular
- Muitos formatadores pré-definidos
- É possível criar o próprio formatador
- “date” é um dos formatadores mais comuns

Date Pipe

date.component.html ×

src > app > views > date > date.component.html

```
1 {{dataAtual | date: 'dd/MM/yyyy'}}
```



Exercícios

1. Crie um novo componente capaz de exibir uma mensagem fixa de “Sobre a aplicação”
2. Exiba o componente na página inicial
3. Customize o estilo de sua página de sobre

ANGULAR - BINDINGS

Bindings

- Bindings (“ligações” do inglês) são formas de conectar o template de um componente (página HTML) a algum código (propriedade ou método) presente no arquivo de comportamento do mesmo (arquivo TypeScript do componente)

Bindings

- Bindings podem ser feitos entre elementos de texto ou atributos de componentes
- Há diferentes tipos de bindings, utilizados em ocasiões específicas, que estudaremos a seguir

Interpolação

- Interpolação é o tipo de binding que utilizamos anteriormente
- Capaz de resolver expressões de propriedades e métodos do TypeScript do componente
- Usado nos templates dos componentes (HTML)

Interpolação

- Para realizar a interpolação de uma propriedade ou método, declaramos o mesmo entre {{ e }}
- O angular substituirá a ocorrência pelo valor da propriedade ou do retorno do método

Interpolação

- Podem ser utilizados tanto no texto da página quanto em atributos HTML e junto a texto (antes ou depois)
- É necessário que a propriedade ou o método exista no arquivo .ts do mesmo componente

Interpolação

{{ propriedade }}

Resolução de Interpolações

- As propriedades interpoladas serão resolvidas com base nos valores do template do componente
- Não é possível utilizar propriedades de outros componentes diretamente (apenas do próprio componente)

Resolução de Interpolações

test.component.ts

```
@Component({
  selector: 'app-test',
  templateUrl: './test.component.html',
  styleUrls: ['./test.component.css']
})
export class TestComponent implements OnInit {
  propriedade = 'Teste';

  constructor() { }

  ngOnInit() {
  }
}
```

test.component.html

<p>{{propriedade}}</p>

Quando o Angular processar e
exibir o HTML do componente,
o valor de {{propriedade}} será
resolvido para "Teste"

<p>Teste</p>

Formas de Uso de Interpolação

- Uso de interpolação no interior de tags ou junto a outro tipo de texto de conteúdo

{{ propriedade }}

Resultado: {{ propriedade }}

<p>{{ propriedade }}</p>

<p>Valor é {{ propriedade }}</p>

Formas de Uso de Interpolação

- Uso de interpolação em valores de atributos, integral ou parcialmente (junto a mais texto)

```

```

```

```

```
<p class="{{value}}">
```

```
<p class="p.{{highlightClass}}">
```

Chamada a Métodos

- É possível efetuar a chamada de métodos no interior de interpolações
- O método será substituído pelo valor de seu retorno e é possível fornecer-lhe parâmetros

`{{ obterValor() }}`

`<p>{{ somarValores(1, 1) }}</p>`

Interpolação e Expressões

- Interpolações são capazes de resolver expressões simples, como cálculos e concatenações

<p>A soma de 1 + 1 é {{1 + 1}}</p>

<p>Soma: {{ 1 + 1 + getValue() }}</p>

Tipos de Dados de Interpolações

- As interpolações podem ocorrer em quaisquer tipos de dados, basta que o Angular seja capaz de obter uma representação de string do mesmo
- string, number, boolean, Array, etc

Boas Práticas de Interpolações

- Simplicidade
 - As expressões de interpolação devem ser simples e de fácil entendimento
 - É possível escrever expressões bem complexas utilizando interpolações, **mas não é recomendado**
 - Focar em propriedades e métodos

Boas Práticas de Interpolações

- Execução Rápida
 - Não devem ser utilizadas expressões que realizem tarefas pesadas, já que o Angular precisará reprocessar a expressão sempre que algo atualizar
 - Evitar consultas remotas a bancos de dados em expressões de interpolação

Exercícios de Interpolação

- *Crie um componente novo para cada exercício e adicione-o em app-component*
1. Crie uma propriedade com seu nome e outra com seu sobrenome. Faça com que ambas apareçam juntas, separadas por um espaço, na página

Exercícios de Interpolação

2. Crie um método que aceite uma string e seja capaz de devolvê-la com todos os caracteres maiúsculos. Chame esse método na página através de interpolação, fornecendo-lhe como parâmetro a palavra “corona”

Exercícios de Interpolação

3. Defina uma propriedade com o nome de classe CSS “petroleo” e:

1. Crie esta classe no CSS com a fonte verde escuro, negrito, fundo amarelo
2. Faça com que a classe seja atribuída através de interpolação num h1 com a palavra “Petrobrás”
3. Utilize a função do exercício anterior para deixar também a palavra em maiúsculas

Binding Unidirecional

- A interpolação é uma forma de binding unidirecional (one-way binding)
- Representa que o valor é modificado em um lado (componente) e o efeito é propagado para o outro (template/HTML)

Binding Unidirecional

- O binding unidirecional, nesse caso, ocorre do componente em direção ao HTML
- Sempre que o componente tiver a propriedade atualizada, o HTML será atualizado pelo Angular

Binding Unidirecional

- Dizemos que o binding via interpolação é um tipo de binding que ocorre em source-to-view (código para a tela)

Property Binding

- Outra forma de realizar binding unidirecional (além da interpolação) é através da utilização de *property binding*
- Permite ligar uma propriedade de um elemento HTML diretamente a uma propriedade/método no *.ts* do componente

Property Binding

- Para realizar property binding, inserimos colchetes “[]” ao redor de um atributo HTML e, em seu valor, apontamos para o nome da propriedade ou do método que deve ser consultado para resolução do atributo
- Se for um método, deve retornar o valor

Property Binding

`<p [class]="styleClass">Teste</p>`

Property Binding

.html

```
<p [class]="styleClass">Teste</p>
```

.ts

```
...  
export class TestComponent implements OnInit {  
  
    styleClass = 'pHighlight';  
  
    constructor() { }  
  
    ...  
}
```

.css

```
.pHighlight {  
    color: yellow;  
    font-weight: bold;  
}
```

Property Binding

- É possível utilizar *property bindings* para controlar o estado de componentes
- Se o valor for alterado no .ts, as alterações serão refletidas automática e instantaneamente no HTML

Property Binding

.html

```
<input [disabled]="disableInput">
```

.ts

```
...  
export class TestComponent implements OnInit {  
  
    disableInput = false;  
  
    constructor() { }  
  
    ...  
}
```

Property Binding

- *Property binding* também é um tipo de binding que ocorre em source-to-view (do código para a tela)
- Ou seja, as propriedades alteram o HTML conforme são modificadas no .ts

Exercícios de Property Binding

4. Insira três imagens na pasta assets

- a) Crie três tags img capazes de resolver qual imagem mostrar através de property binding (3 propriedades diferentes)
- b) Crie também 3 classes CSS com configurações diferentes para imagens (tamanho, borda, etc) e atribua-as através de property binding às imagens

Binding de Eventos

- O binding de eventos ou “declarações de template” (*template statements*) são utilizados no Angular para permitir que eventos comuns de elementos HTML (clique, pressionamento de teclas, apontar com o mouse, etc) acionem métodos no .ts do componente

Binding de Eventos

- O binding unidirecional, nesse caso, ocorre do HTML em direção ao Componente
- Dizemos que o binding de eventos é um tipo de binding que ocorre em view-to-source (da tela para o código), o contrário dos anteriores

Binding de Eventos

- Sempre que um evento ocorrer, o HTML acionará o código .ts e este poderá alterar o estado das propriedades do componente como desejar (alterações nas propriedades serão refletidas automaticamente na página)

Binding de Eventos

- Para realizar binding de eventos, colocamos o atributo do evento do elemento HTML sem o “on” (click, mouseover, keypress, etc) entre parênteses “()” e, no valor, informamos o método que será executado quando o evento ocorrer

Binding de Eventos

- O método deve ser informado como uma chamada (com () no final) e parâmetros podem ser fornecidos
- O método deve estar no código .ts do próprio componente (não é possível chamar métodos de outros componentes diretamente)

Binding de Eventos

- O método poderá modificar quaisquer propriedades do componente, que, por sua vez, modificarão o HTML (via interpolação ou por property binding)

Binding de Eventos

```
<button (click)="tratarClick()">Clique aqui!</button>
```

Binding de Eventos

.html

```
<button (click)="tratarClick()">Clique aqui!</button>
```

.ts

```
...  
export class TestComponent implements OnInit {  
  
    constructor() { }  
  
    ...  
  
    tratarClick():void {  
        alert('Olá, mundo!');  
    }  
}
```

Binding de Eventos

.html

```
<input [disabled]="disableInput">  
<button (click)="tratarClick()">Clique aqui!</button>
```

.ts

```
...  
export class TestComponent implements OnInit {  
  
    disableInput = false;  
  
    ...  
  
    tratarClick():void {  
        this.disableInput = !this.disableInput;  
    }  
}
```


Exercícios de Event Binding

5. Crie um componente que contenha uma imagem e um botão. Ao clicar no botão, uma segunda imagem deve ser exibida. Se o botão for novamente clicado, alternar para a imagem original (toggle)

Exercícios de Event Binding

6. Crie 3 botões. Quando clicados, cada um deles deve atribuir um texto específico para uma propriedade. Use a propriedade num texto para exibir seu valor conforme os botões são clicados

Exercícios de Event Binding

7. Crie três botões e 3 divs. Ao serem clicados, os botões devem controlar quais classes cada div possui, alternando entre classes em que uma div está visível e as outras duas estão invisíveis, para cada uma das divs (alternando entre divs 1, 2 e 3)

"That's all Folks!"