



Angular 9

Aula #4

Vídeo Aula e Exercícios

Esta apostila é referente a vídeo aula #4 e a sequência de exercícios #4 do curso Angular 9. Não deixe de assistir a vídeo aula e fazer os exercícios propostos para esta aula em nossa Plataforma Online.

Sobre os materiais da Grande Porte

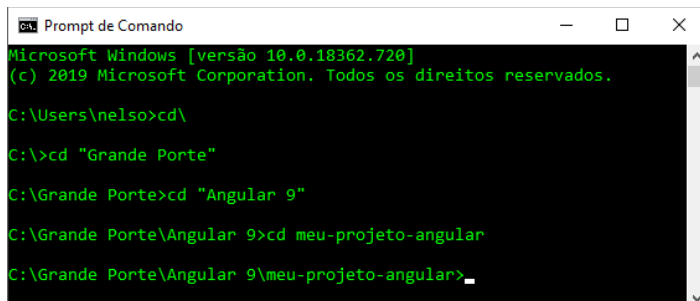
Todos os materiais desenvolvidos pela Grande Porte objetivam ensinar de maneira eficiente, explorando potenciais e sanando dificuldades. As atualizações são feitas constantemente para manter o conteúdo atualizado com as demandas de mercado e tendências para o futuro. Esperamos que você aproveite este material. Comentários, críticas e sugestões serão muito bem-vindos.

ATENÇÃO

Os materiais desenvolvidos pela Grande Porte são distribuídos através da Plataforma Online apenas para assinantes. Todos os direitos são reservados à Grande Porte. A distribuição, cópia, revenda e utilização para ministrar treinamentos, sem autorização, são absolutamente proibidas. Se você deseja obter autorização para usar nossos materiais comercialmente, por favor, entre em contato.

Colocando a aplicação para executar

É preciso subir o servidor NodeJS para executar nossa aplicação Angular 9. Para isso, abra um terminal no Windows e mude o diretório (pasta) para onde você criou o primeiro projeto, conforme figura 1.



```
Prompt de Comando
Microsoft Windows [versão 10.0.18362.720]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

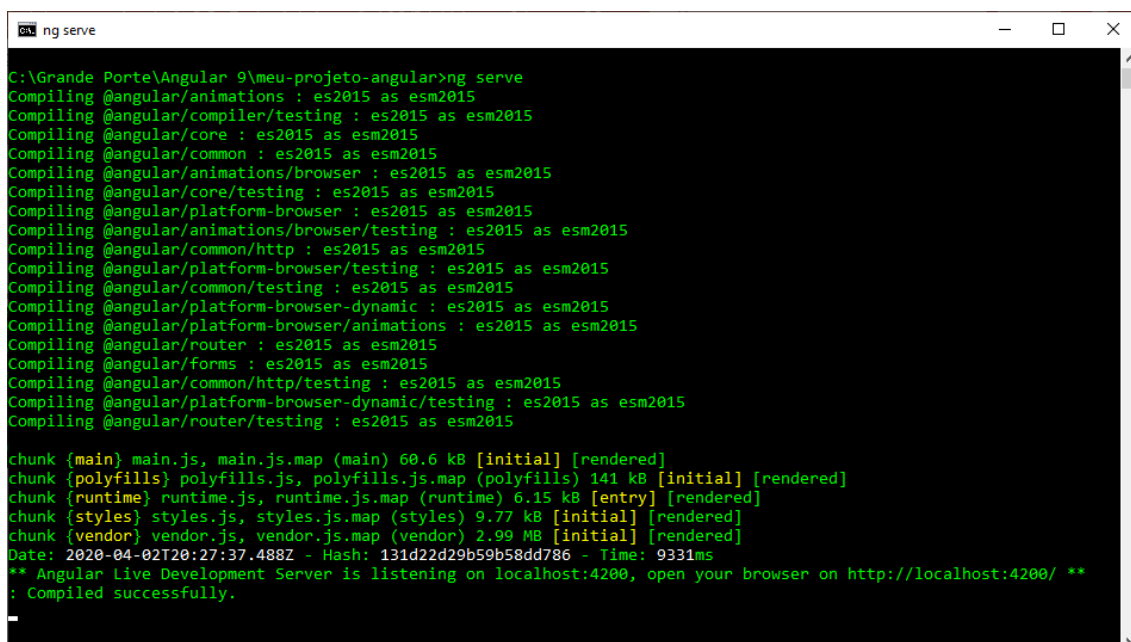
C:\Users\nelso>cd\
C:\>cd "Grande Porte"
C:\Grande Porte>cd "Angular 9"
C:\Grande Porte\Angular 9>cd meu-projeto-angular
C:\Grande Porte\Angular 9\meu-projeto-angular>
```

Figura 1 – Indo para o diretório (pasta) onde o projeto Angular foi criado.

Agora vamos subir o servidor NodeJS digitando o seguinte comando Angular CLI:

```
ng serve
```

isso inicializa o servidor para a nossa aplicação. Pode demorar um pouco, dependendo das configurações de seu computador, mas após o servidor ser inicializado você verá a seguinte tela, conforme figura 2.



```
ng serve
C:\Grande Porte\Angular 9\meu-projeto-angular>ng serve
Compiling @angular/animations : es2015 as esm2015
Compiling @angular/compiler/testing : es2015 as esm2015
Compiling @angular/core : es2015 as esm2015
Compiling @angular/common : es2015 as esm2015
Compiling @angular/animations/browser : es2015 as esm2015
Compiling @angular/core/testing : es2015 as esm2015
Compiling @angular/platform-browser : es2015 as esm2015
Compiling @angular/animations/browser/testing : es2015 as esm2015
Compiling @angular/common/http : es2015 as esm2015
Compiling @angular/platform-browser/testing : es2015 as esm2015
Compiling @angular/common/testing : es2015 as esm2015
Compiling @angular/platform-browser-dynamic : es2015 as esm2015
Compiling @angular/platform-browser/animations : es2015 as esm2015
Compiling @angular/router : es2015 as esm2015
Compiling @angular/forms : es2015 as esm2015
Compiling @angular/common/http/testing : es2015 as esm2015
Compiling @angular/platform-browser-dynamic/testing : es2015 as esm2015
Compiling @angular/router/testing : es2015 as esm2015

chunk {main} main.js, main.js.map (main) 60.6 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 141 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 9.77 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 2.99 MB [initial] [rendered]
Date: 2020-04-02T20:27:37.488Z - Hash: 131d22d29b59b58dd786 - Time: 9331ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
: Compiled successfully.
```

Figura 2 – O servidor NodeJS inicializado com nossa primeira aplicação Angular 9.

O servidor irá fazer a compilação do código Typescript (convertendo para Javascript) entre outras tarefas, deixando a aplicação pronta para ser usada pelo browser.

Observe que a última mensagem indica que o servidor está ouvindo a porta 4200 no localhost, que é seu computador local. Abra o browser de sua preferência e digite o endereço indicado pela mensagem:

`http://localhost:4200`

Isso irá subir nossa aplicação no browser, conforme figura 3.

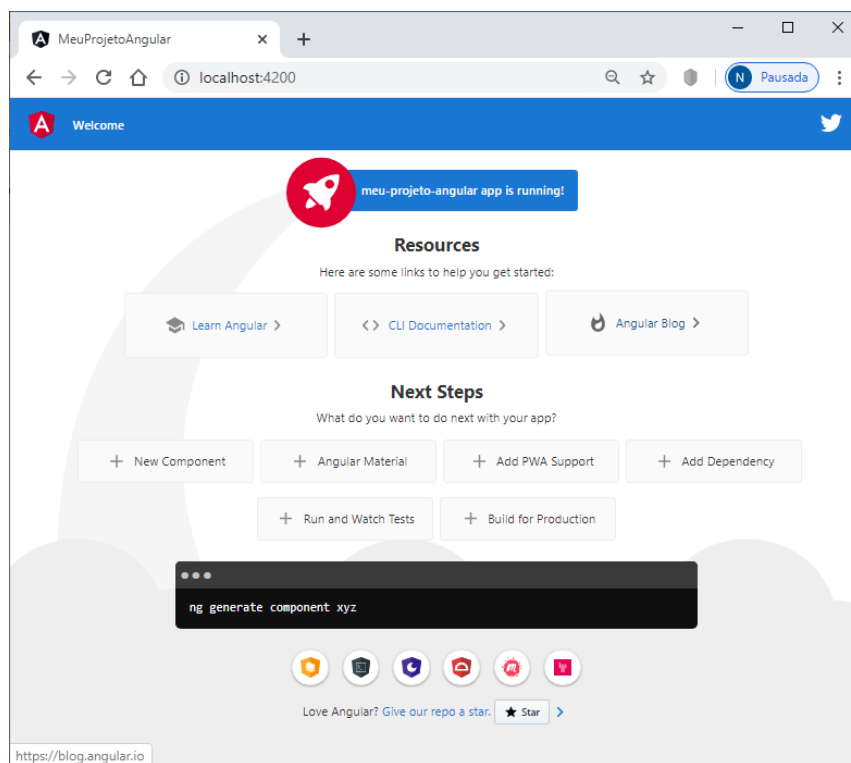


Figura 3 – O browser Google Chrome exibindo nossa aplicação

Todos esses textos estão no arquivo **app.component.html** onde faremos uma pequena alteração para ver o resultado.

Arquivos de um componente

Um componente é formado por diversos arquivos que contêm a palavra **component** no nome, conforme figura 4.

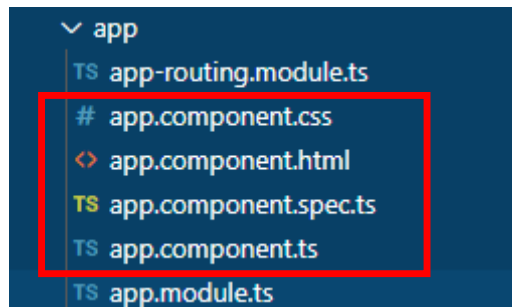


Figura 4 – Os arquivos do componente app.

Todos esses arquivos têm sua finalidade, mas os principais são:

- **app.component.html** – é o html deste componente, também chamado de template.
- **app.component.css** – é o css específico para este componente.
- **app.component.ts** – é o typescript deste componente, que será convertido para Javascript pelo servidor.

Os arquivos que contêm a palavra **module** fazem parte do módulo app.

Sempre que um componente for alterado, ele será recompilado pelo servidor e automaticamente carregado no browser.

Vamos fazer uma pequena alteração no arquivo **app.component.html** acrescentando “**Grande Porte**” na linha 333 (ou próximo dela). Abaixo temos o texto original da tag .

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

Acrescente Grande Porte ao final da frase “app is running!”, conforme texto abaixo:

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

Na figura 5 temos o Visual Studio Code editando o arquivo **app.component.html**, mostrando a linha 333 onde será feita a alteração indicada acima.

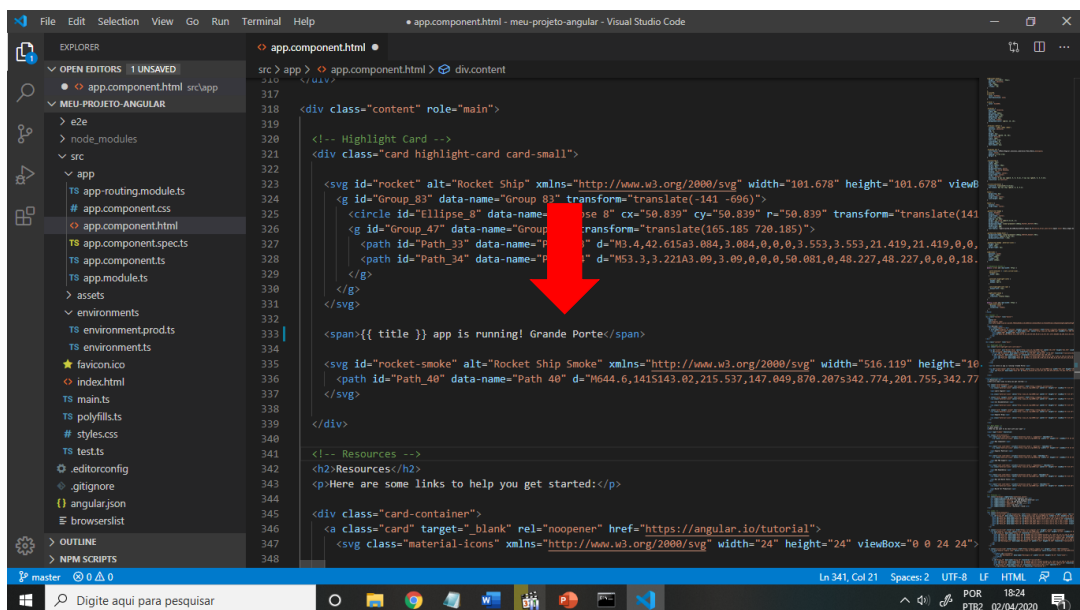


Figura 5 – O arquivo **app.component.html** aberto na linha da alteração.

Como o autosave do Visual Studio Code está ativado por default, essa alteração será salva, o servidor irá compilar esse componente e o browser será automaticamente atualizado e já exibirá essa alteração, conforme figura 6.

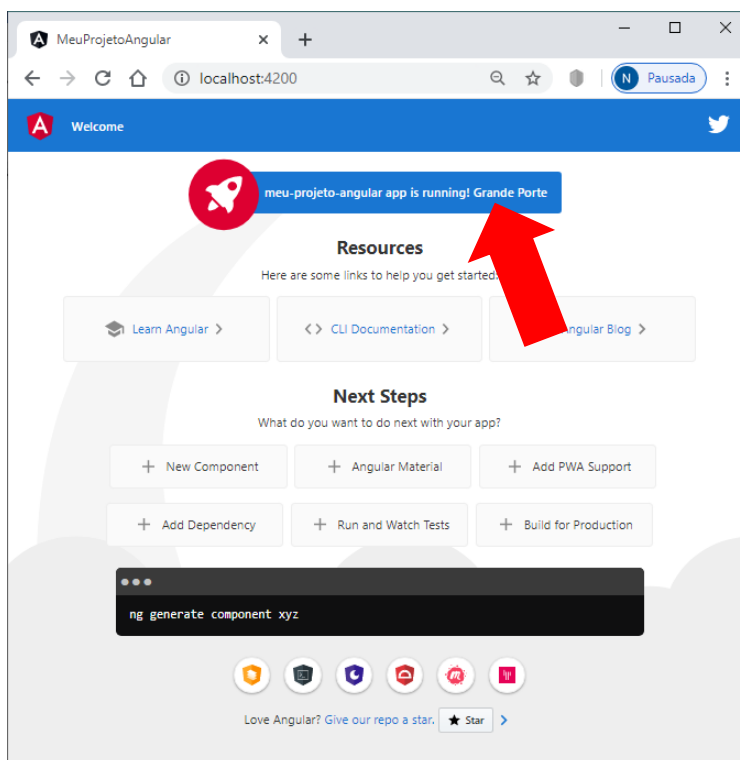


Figura 6 – Browser refletindo a nova alteração.

Isso ocorreu porque o Angular é autoloadeado, o que significa que as páginas são carregadas automaticamente.

Criando nosso primeiro componente

Grande parte do desenvolvimento com Angular 9 é feito nos componentes, que são basicamente classes que interagem com o arquivo .html do componente, que por sua vez é exibido no browser.

Como visto anteriormente na figura 4, o componente app possui os seguintes arquivos:

- app.component.css
- app.component.html
- app.component.spec.ts
- app.component.ts

Esses arquivos foram criados por default quando nós criamos um projeto com o comando Angular CLI. O arquivo **app.module.ts** é onde foi definido um módulo da aplicação. Ao abrir esse arquivo veremos inicialmente algumas bibliotecas que são importadas e uma declaração (*declaration*) que indica qual componente está neste módulo, conforme figura 7.

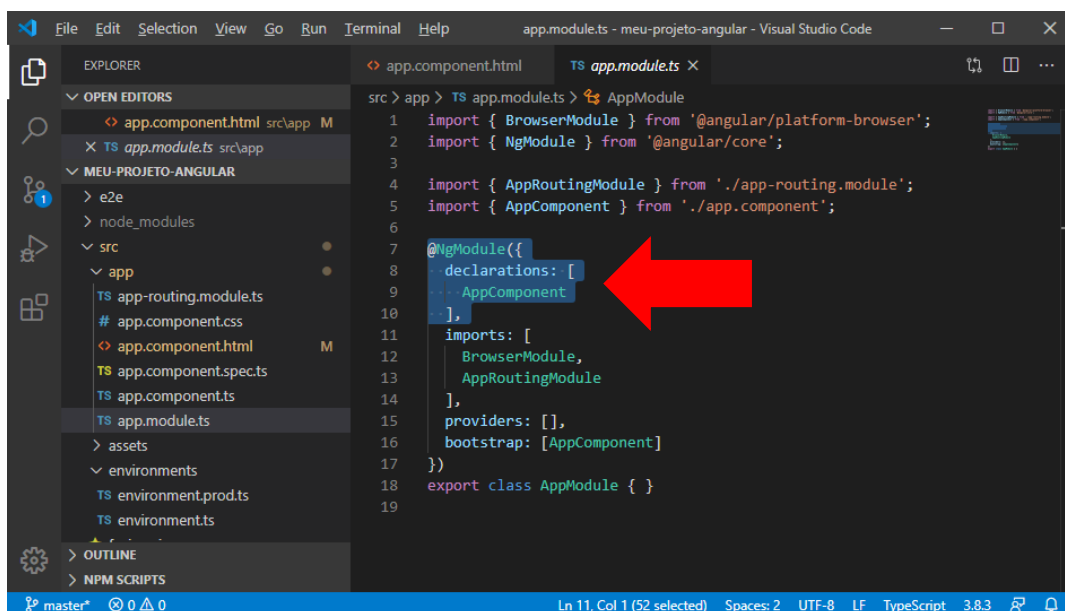


Figura 7 – O arquivo app.module.ts declarando o componente AppComponent.

Um componente tem que pertencer unicamente a um módulo, mas um módulo pode conter mais de um componente. O componente app, que foi criado por default é o componente **pai** ou **raiz** neste módulo, e os demais componentes serão **filhos** deste.

mais abaixo neste arquivo temos:

```
bootstrap: [AppComponent]
```

Em geral, o bootstrapping se refere a um processo de auto inicialização que deve prosseguir sem entrada externa. Na tecnologia de computadores, o termo geralmente se refere ao processo de carregamento do software básico na memória de um computador após a inicialização ou reinicialização geral, especialmente o sistema operacional que cuidará do carregamento de outro software, conforme necessário.

No Angular 9 o bootstrap indica qual é o componente que deve ser chamado primeiro (**raiz**) para inicializar a aplicação. A figura 8 relembra a estrutura de uma aplicação mostrando o componente raiz.

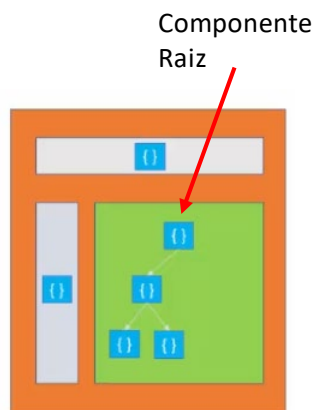


Figura 8 – Uma aplicação e o componente raiz

Agora vamos acrescentar um novo componente em nossa aplicação.

Criando outro componente

O Angular CLI possui um comando para criar o nosso próprio componente. No entanto, o **app.component** criado por padrão sempre permanecerá como o **pai** e os próximos componentes criados formarão os componentes **filhos**.

É preciso abrir um novo terminal do Windows para digitar o comando Angular CLI, entretanto é possível utilizar esse terminal de dentro do Visual Studio Code. Para isso utilize o menu Terminal → New Terminal, conforme figura 9.

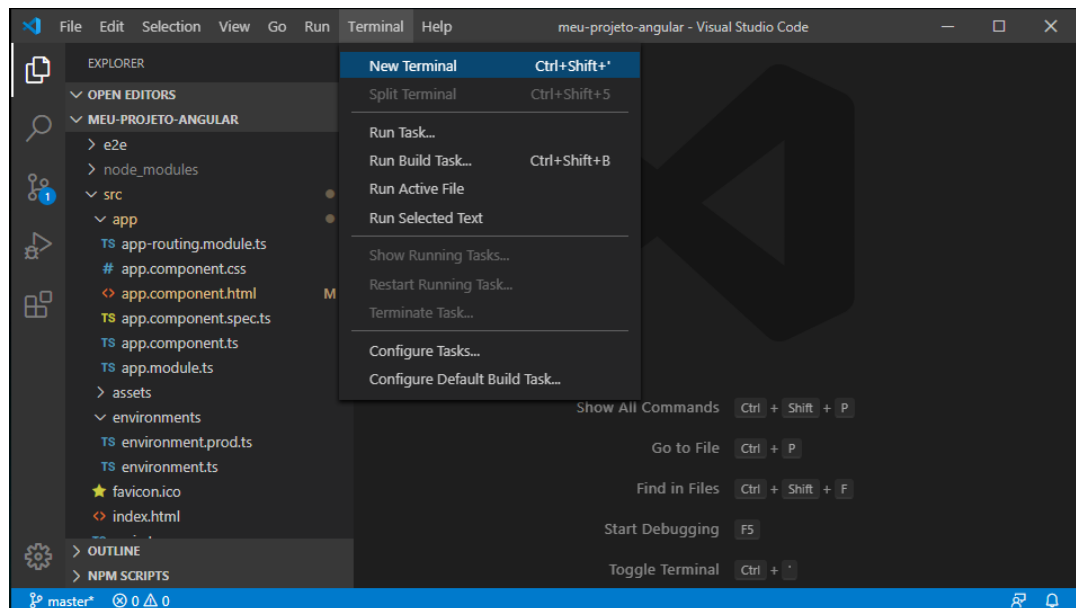


Figura 9 – Menu para abertura de um novo terminal.

O terminal já é aberto na pasta (diretório) de nosso projeto, conforme figura 10.

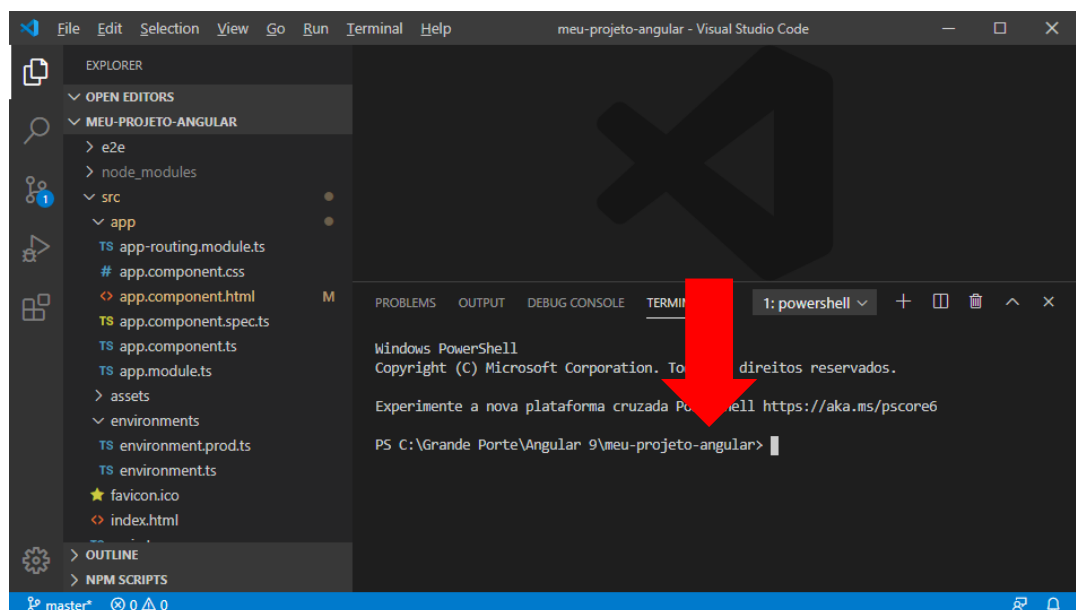


Figura 10 – O novo terminal aberto na pasta do projeto.

Vamos agora executar o comando para criar um componente. Digite a linha de código abaixo no terminal dentro do Visual Studio Code para criar um componente chamado **primeiro**.

```
ng g component primeiro
```

Após a execução deste comando, o componente com seus arquivos será criado dentro da pasta app, conforme figura 11.

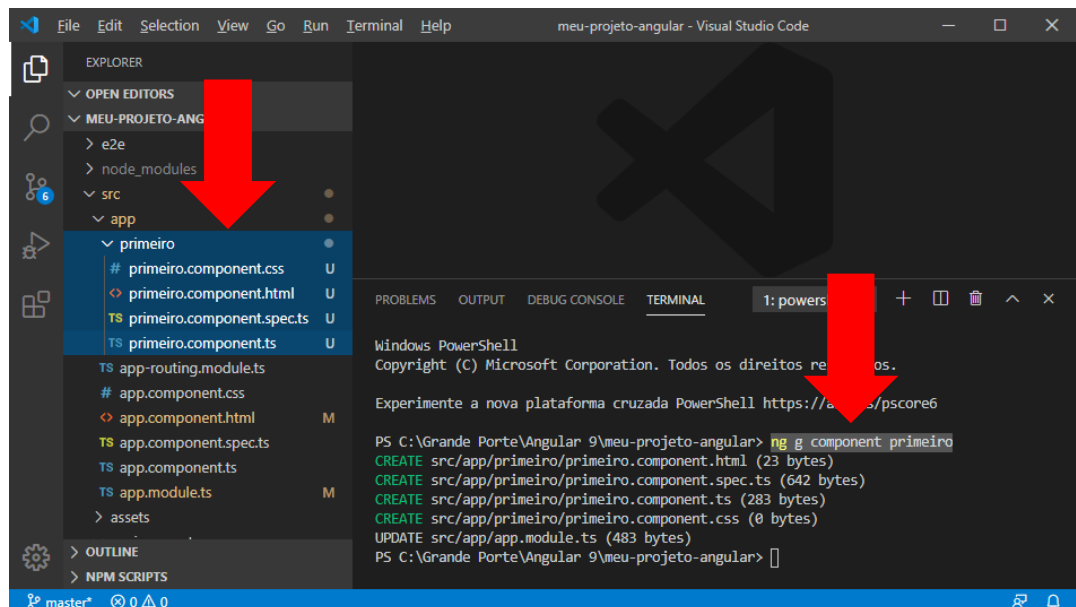


Figura 11 – A criação de um componente, cujo nome é: **primeiro**.

Todo novo componente é criado com esses quatro arquivos, mas como não iremos neste treinamento trabalhar com testes automatizados, podemos excluir o arquivo **primeiro.component.spec.ts**, sobrando então apenas três arquivos, que são os principais de um componente, conforme figura 12.



Figura 12 – Os três principais arquivos de um componente.

Vamos dar uma olhada em cada um desses três arquivos para ver o que foi automaticamente gerado.

O arquivo **primeiro.component.css** é gerado vazio, conforme figura 13.

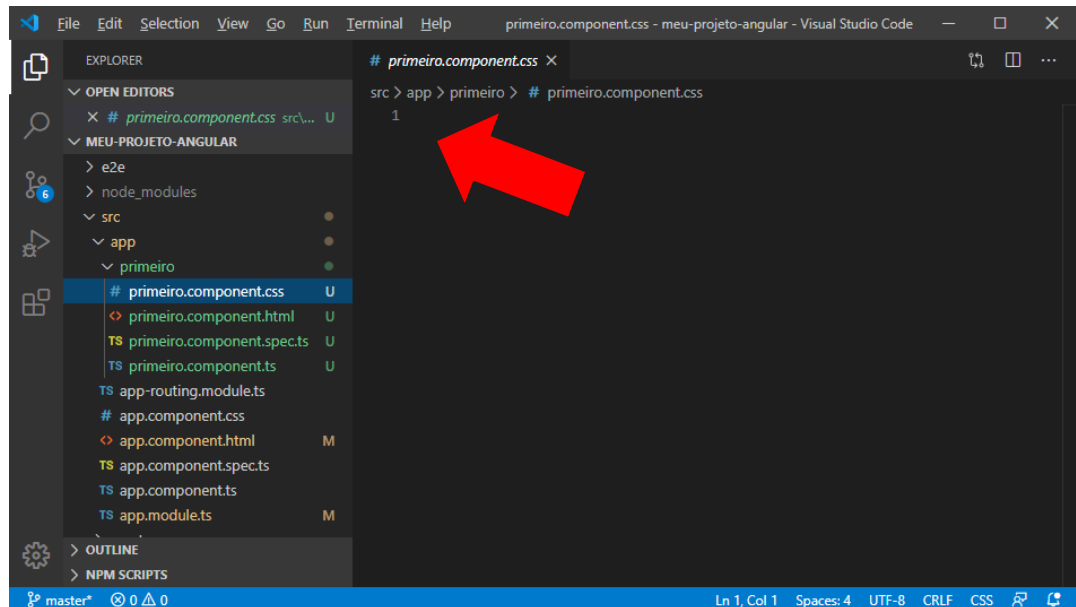


Figura 13 – O css do componente é criado vazio.

O arquivo **primeiro.component.html** é gerado praticamente vazio, possuindo apenas um parágrafo, conforme figura 14: tag `<p>primeiro works!</p>`

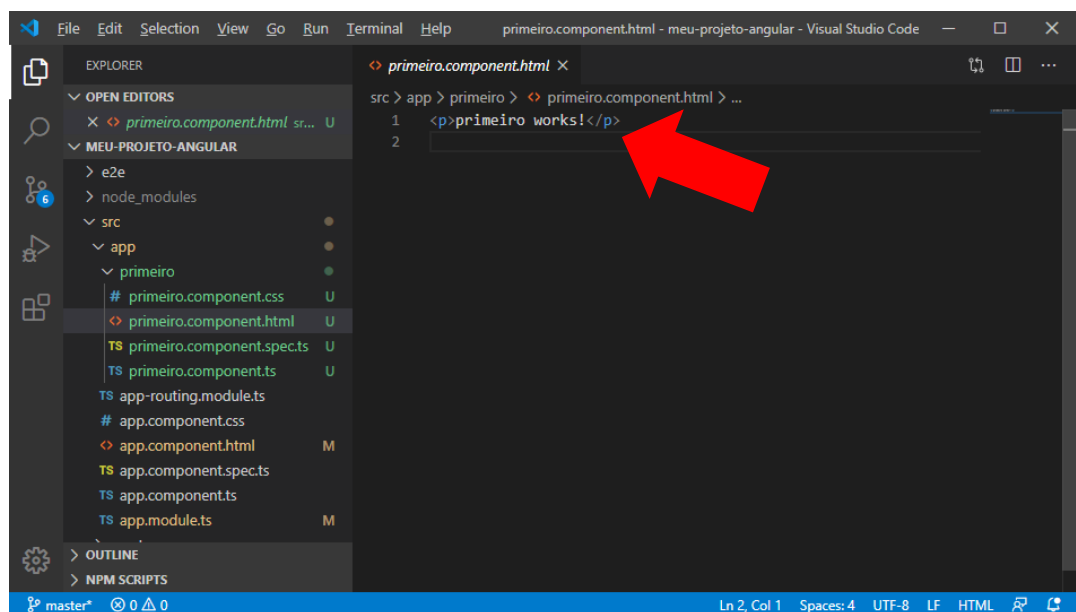


Figura 14 – O html do componente é criado apenas com um parágrafo.

Já o arquivo **primeiro.component.ts** possui um código typescript comum a todos os componentes, onde destacamos o JSON (*Javascript Object Notation*) para o objeto @Component e, em especial, o par nome:valor:

```
selector: 'app-primeiro'
```

cujo valor será usado para chamar este componente, conforme figura 15.

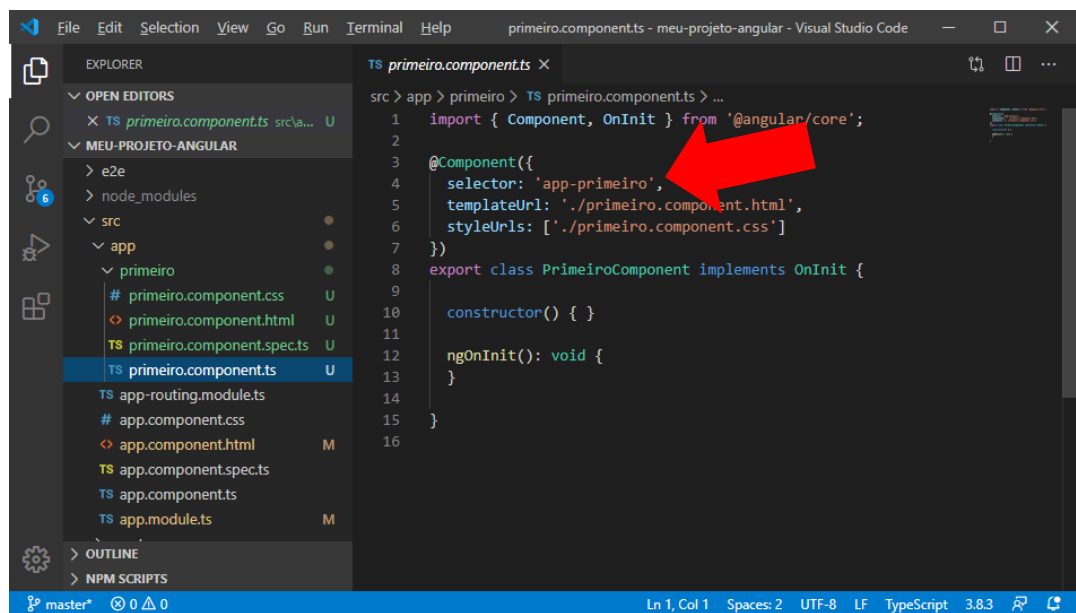


Figura 15 – O typescript do componente, destacando o par selector:"app-primeiro".

A chamada de componentes será vista já na próxima aula. Vamos agora aprender a criar um módulo em nossa aplicação.

Criando um módulo

Os módulos em Angular 9 se referem a lugares onde nós podemos agrupar os componentes, diretivas, serviços etc., que estão relacionadas à aplicação.

Se sua aplicação web possuir um cabeçalho, um rodapé, uma seção central, esquerda ou direita, todas essas “peças” serão parte de um módulo.

Vamos agora executar o comando para criar um modulo. Digite a linha de código abaixo no terminal dentro do Visual Studio Code para criar um modulo chamado **usuarios**.

```
ng g module usuarios
```

Após a execução deste comando, o modulo com seus arquivos será criado dentro da pasta app, conforme figura 16:

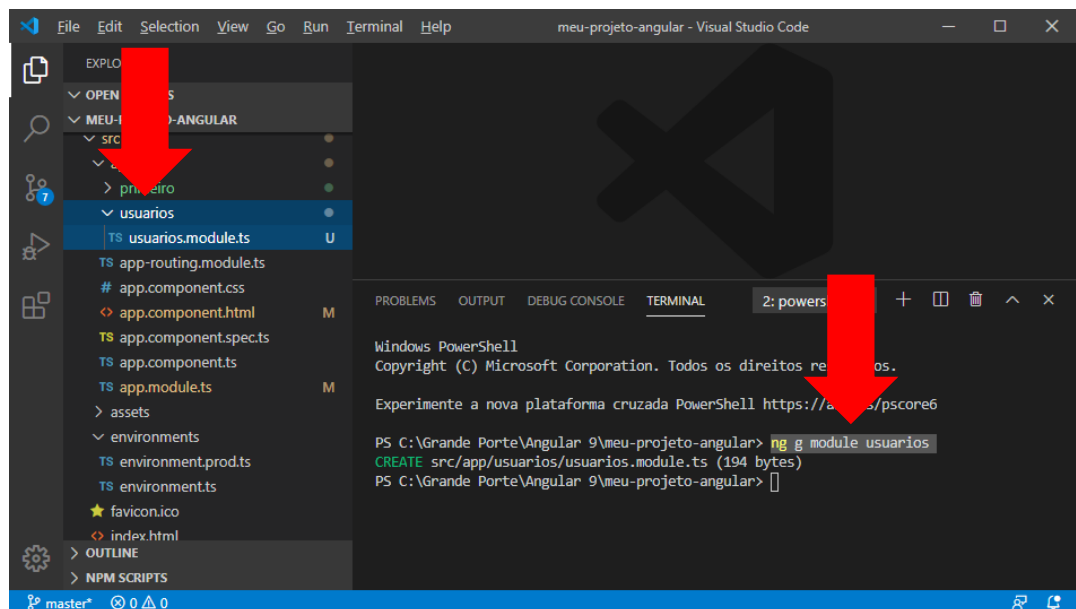


Figura 16 – A criação de um módulo cujo nome é: **usuarios**.

O módulo **usuarios** foi criado contendo um único arquivo chamado: **usuarios.module.ts**.

Na próxima aula aprenderemos a chamar um componente a partir de outro componente.

Vídeo Aula e Exercícios

Esta apostila é referente a vídeo aula #4 e a sequência de exercícios #4 do curso Angular 9. Não deixe de assistir a vídeo aula e fazer os exercícios propostos para esta aula em nossa Plataforma Online.

Sobre o autor

Fabrizio Borelli



Doutorando em Engenharia da Informação pela UFABC

Mestre em Ciência da Computação pela UFABC

Bacharel em Ciência da Computação pela UFABC

Professor Universitário

Empreendedor em TI