09

# Entendendo a estrutura

## Transcrição

Vamos entender a estrutura que foi gerada? Inicialmente, abriremos a pasta "alurapic" no Visual Studio Code, em "File > Open Folder...", ou pelo atalho "Ctrl + K Ctrl + O". Selecionaremos a pasta, cujo conteúdo será exibido em um painel lateral esquerdo no programa.

O Visual Studio se integra muito bem à linguagem utilizada pelo Angular, que é o **TypeScript**. Mas onde é que se localiza a página gerada anteriormente, com "Welcome to app!", em nosso projeto? Neste painel lateral, não há nenhum `index.html`, e o importante, neste momento, é que todo código que diz respeito à aplicação está dentro de "src" (de *source*), inclusive `index.html`.

Pode ser tentador abrirmos diretamente a pasta "src" no programa, porém isto não é recomendado. Deve-se sempre abrir a "alurapic" como pasta raiz do Visual Studio Code, pois ele precisa encontrar alguns arquivos de configuração dentro dela. Caso contrário, teremos muitos problemas.

Vamos, então, abrir `index.html`, que é a página padrão criada pelo servidor local configurado pelo Angular CLI. Entretanto, a página do navegador (`localhost:4200`) possui uma imagem, lista de links, entre outros elementos que não aparecem no código de `index.html`. Além disso, o conteúdo dentro das tags `<body>` são tags que nem existem no mundo HTML, `<app-root>`.

Em Angular, **tudo é um componente**, capaz de guardar um comportamento, o CSS, e a marcação HTML, a estrutura, em um único local. Assim, a página `localhost:4200` não é diferente no Angular, já que uma página também é un

componente, e possui HTML, CSS e, caso haja, JavaScript, tudo vinculado em um único objeto denominado componente. Assim, `<app-root>` indica a existência de componentes.

Mas onde se encontram estes componentes, em nosso projeto?

Tudo que formos criar ao longo do curso ficará dentro de "app", em que há `app.component.ts`. Abrindo-o, veremos que, basicamente, temos uma classe do ***ECMAScript 6***, com um *Decorator* anotado com `@Component`, o qual torna a classe um componente. O *Decorator* é uma sintaxe especial do Angular, do TypeScript, em que é possível incluir uma **metainformação** sobre uma determinada classe, no caso.

O que é uma metainformação?

Ao criarmos instâncias desta classe, criamos objetos. Estamos incluindo mais uma informação desta classe, que diz respeito ao framework. Então, a classe `AppComponent` só é um componente porque está anotada com `@Component`. Nele, existe um `selector: 'app-root'`, mesmo nome encontrado em `index.html`.

Este *selector* nos permite utilizar o componente em templates em sua forma declarativa, então, todo o conteúdo de `app.component.ts`, sua apresentação, o que ele faz, seu CSS, são acessados por meio dele. Também neste arquivo, há `templateUrl: './app.component.html'`, que informa a apresentação deste componente.

Se abrirmos `app.component.html`, encontraremos o código referente à apresentação que vemos na página do navegador. Então, o Angular carregará, exibindo o primeiro componente, e seu template. Voltando a `app.component.ts`, o `styleUrls: ['./app.component.css']` informa onde se localiza o CSS utilizado por este componente.

Quando uma aplicação Angular é carregada pela primeira vez, sabemos que é a `<app-root>` que será carregada, pois ela, na sua forma declarativa no *selector*, é que está no `index.html`.

De que forma o Angular consegue acessar tudo isso que vimos e transformá-los em algo que o navegador consegue carregar e entender?

Internamente, o Angular utiliza um *webpack*, um *module bundle*, isto é, um empacotador de módulos famoso, utilizado por *Create React App, Vue CLI*, e outros frameworks *Single Page Application*. O interessante é que em nenhum momento estas configurações do *webpack* são exibidas, pois elas ficam encapsuladas pelo Angular CLI.

Com isso, fizemos o primeiro tour em relação à aplicação. Antes de começarmos a criar nossos próprios componentes, veremos mais um detalhe, a seguir.