01

# Adicionando Bootstrap ao projeto

## Transcrição

Tivemos um *overview* da estrutura do projeto, do Angular, e do *Data binding*, e podemos criar nosso primeiro componente. Antes disso, precisamos resolver uma questão: como iremos estilizar estes componentes?

Em um primeiro momento, utilizaremos o ***Bootstrap***, então precisaremos carregá-lo em nossa *Single Page Application* usando o Angular. Vamos voltar ao Visual Code, abrir o `index.html`, e incluir a tag `<link>` do Bootstrap, certo?

Não! No Angular, quando precisamos importar um CSS global como o Bootstrap, Normalize (https://necolas.github.io/normalize.css/), CSS Reset ou outro similar, isso não funciona desta forma. Isto porque esses arquivos CSS precisam estar no processo de *build*, de construção da nossa aplicação, tanto no ambiente de desenvolvimento quanto no ambiente de produção.

E se incluirmos a tag diretamente no código, o Angular CLI não saberá que isto é uma dependência da aplicação, e que ele precisa entrar neste processo de construção. Isso é um tanto estranho para quem está familiarizado com uma aplicação tradicional: o Angular CLI pode acessar o CSS e transformá-lo em um único código JavaScript, ou então separar o arquivo `.css`. Então, ele precisa saber onde se encontra o arquivo CSS, e isto não se dá por meio da tag `<link>`.

## Em que lugar carregamos CSSs ou *scripts* globais?

No caso, como estamos falando sobre CSS, há um arquivo em "ALURAPIC" chamado `angular.json`, com uma série de configurações que dizem respeito

ao *build* do projeto, tanto que no meio dela há chaves referentes a `styles` e `scripts`. Estas propriedades servem inicialmente para carregarmos em `styles` todos os CSSs globais da aplicação, ou seja, os que serão aplicados em todos os componentes, bem como os *scripts* globais, os quais não pertencem a nenhum componente específico.

No caso, já existe `src/styles.css` em `styles`, o que significa que o programa busca dentro da pasta "src" o arquivo `styles.css`. No entanto, agora precisamos carregar o Bootstrap, e para tal precisamos baixá-lo, e indicar o caminho de onde se localiza este arquivo. Mas não precisaremos acessar o site e baixá-lo dali. Já que estamos utilizando o gerenciador de pacotes do Node para o Angular CLI funcionar, poderemos utilizá-lo para baixar todas as dependências de front end de que a aplicação necessita.

Se quisermos usar jQuery (https://jquery.com/), Bootstrap (https://getbootstrap.com/), Foundation (https://foundation.zurb.com/apps/docs/#!/), e afins, conseguimos baixá-los pelo terminal.

Vamos, então, pausar o Angular CLI, usar o comando `npm install bootstrap@4.1.1` dentro da pasta do projeto, isto é, iremos utilizar a versão `4.1.1` do Bootstrap. Não se preocupe caso não o conheça, pois aprenderemos seus aspectos fundamentais ao longo do curso.

Feito isso, o repositório será acessado, o Bootstrap será baixado, e a dependência do nosso projeto será incluída em `package.json`. Todo projeto em Node possui este arquivo, que lista todas as dependências e módulos que foram baixados pelo npm.

No passado era necessário acrescentarmos `--save` ou `-S` ao fim do comando que acabamos de utilizar, para que o arquivo fosse devidamente listado em `package.json`. Entretanto, já que estamos utilizando a versão Node superior à

8, no caso, `10`, isto não é necessário, pois a dependência é adicionada à listagem automaticamente.

Sabemos que o Bootstrap se encontra em "node_modules > bootstrap > dist > css", mas como o carregaremos em nossa aplicação?

Ele precisa fazer parte de uma lista de CSSs globais que o Angular CLI levará em consideração em tempo de construção da aplicação em si, seja ela para rodar em desenvolvimento ou em produção. Esta indicação é feita em `angular.json`, que possui uma chave denominada `build`, dentro do qual está `styles`. Alteraremos e salvaremos seu conteúdo para incluir o caminho completo do Bootstrap na pasta "node_modules":

```
"styles": [
    "src/styles.css",
    "./node_modules/bootstrap/dist/css/bootstrap.min.css"
],
"scripts": []
```

COPIAR CÓDIGO

Podemos notar que em "src" já existia um arquivo `styles.css`, que se encontra vazio. O importante é entendermos que, ao alterarmos `angular.json`, somos obrigados a fechar e reabrir o Angular CLI para que a modificação seja efetiva. Se o Bootstrap for carregado corretamente, no navegador, a fonte de "alurapic" será alterada.

No terminal, tendo o Angular CLI pausado, usaremos o comando `ng serve --open` para que se abra uma nova aba que carrega o Bootstrap, e veremos a modificação no navegador. Com isso, partiremos para a melhoria na responsividade, já que a imagem não diminui conforme diminuímos a página.

Vamos voltar ao Visual Code, abrir o template `app.component.html`, que exibe uma tag `<img>` apontando para uma `url` e um `title`, que vêm do próprio

`app.component.ts`, e então acrescentaremos a classe do Bootstrap, `img-thumbnail`:

```
<h1>{{ title }}</h1>
<img class="img-thumbnail" [src]="url" [alt]="title">
```

COPIAR CÓDIGO

Assim, ao retornarmos ao navegador e manipularmos as dimensões da página, veremos que a imagem se adequará. Isso é outra prova de que o Bootstrap está funcionando. Contudo, podemos fazer muito mais por esta simples estilização, e é isso que veremos a seguir.