02

SPAs e consumo de APIs

Transcrição

Chegou a hora de realizarmos a integração da nossa aplicação Angular com uma API, um back end. Neste momento, nossa aplicação está rodando em um servidor disponibilizado pelo Angular CLI. Ainda não entraremos na questão de gerar o projeto para que seja colocado em produção, mas é importante entendermos que o projeto que criamos será transformado em alguns arquivos que eventualmente poderão ser inclusos em um servidor PHP, Apache, Java, ou qualquer outro.

No caso, o Angular CLI faz este papel para nós, e cria um servidor de desenvolvimento com todas as vantagens que vimos até aqui. Continuando, uma aplicação em Angular é **estática**, ou seja, roda no navegador, sendo assim alguém precisa fornecer os dados para ela.

Assim, do outro lado da aplicação haverá outro servidor responsável por fornecer os dados para a aplicação Angular. Em uma *Single Page Application*, este servidor normalmente é chamado de **Web API**, e funciona fornecendo-se dados, após o qual a aplicação Angular conhecerá o endereço de onde obtê-los, os acessará, trazendo e renderizando-os para a exibição dos mesmos na tela.

Então, em um fluxo natural de uma *Single Page Application*, a aplicação em Angular fica em um servidor A, e a API com os dados e regras de negócio ficam em um servidor B, totalmente independentes. A vantagem disso é que a equipe que trabalha na API, no caso do back end, pode trabalhar como preferir sem interferir no que a equipe de front end estiver fazendo, e vice versa.

Claro, deve haver uma comunicação entre as aplicações front e back/Web API. E é isso que veremos nesta aula, além das maneiras como o Angular consegue acessar outro servidor e consumir seus dados para exibi-los ao usuário final. No entanto, não entraremos no mérito de criação deste servidor, dessa Web API, porque este é um curso de Angular, e não de design e criação de Web APIs.

Em um dos próximos exercícios há um link com os endereços da API que iremos utilizar no curso, disponível para download, dentre os quais está `api.zip`, que precisará ser descompactada e então recortada.

> **Cuidado:** caso apareça uma pasta "api" dentro de outra com mesmo nome, a que deverá ser recortada é a que contém arquivos como `data.db`, `package`, `package-lock`, `data` e `server`.

Colaremos a pasta na área de trabalho, onde também se encontra "alurapic". Pensando em produção, "alurapic" se relaciona ao servidor que hospeda a aplicação Angular, enquanto "api" é o servidor que hospeda a Web API.

Temos um terminal rodando "alurapic", e abriremos outro Prompt de Comando para rodar nossa API, pois **ambos precisam estar rodando simultaneamente**. Digitaremos os comandos `cd Desktop`, `cd api` e `npm install`, sendo que este último só deverá ser usado uma única vez, para que a API baixe todas as dependências necessárias para o funcionamento da aplicação.

> Em alguns casos, devido a atualizações de um componente da API, pode ser nescessário utilizar `npm install --unsafe-perm -g sqlite3`!

Em seguida usaremos `npm start`, comando que executará a Web API. São exibidas algumas informações de acesso, usuário, e-mail, senhas, dados que por ora não nos serão relevantes, já que dizem respeito à parte de autenticação. O importante é que, ao abrirmos o navegador e digitarmos a URL "localhost:3000/flavio/photos", teremos acesso a dados de diversas imagens cadastradas para uso em nossa aplicação.

Neste curso, faremos então a integração da aplicação Angular com a Web API. O *array* `photos` virá com os dados retornados pela Web API. Lembrem-se de que os dois terminais precisam estar rodando; caso o estudo seja pausado em um dia para continuidade no próximo, é preciso abrir o terminal na pasta do Angular CLI e usar o comando `ng serve --open`, e ir à pasta da Web API e usar `npm start`.