03

Motivação para uso de uma SPA

A seguir, temos uma pequena história para motivar o uso do Angular em nosso projeto.

Rômulo: pessoal, tenho uma ideia de criar uma aplicação para gerenciamento de fotos.

Flávio: mas já existem dezenas delas no mercado, inclusive gratuitas!

Rômulo: você tem razão, porém todas são bem genéricas e não focam um nicho específico. A minha ideia é focar em estudantes de tecnologia. Dessa forma, poderemos atender melhor este nicho.

Nico: gostei da iniciativa. Vai ser uma aplicação web ou uma aplicação nativa? Ou quem sabe os dois?

Rômulo: primeiramente quero começar com uma aplicação web, e quem sabe no futuro criar uma versão nativa em Android o iOS. Precisamos estar preparados para esse cenário.

Flávio: bem, nesse caso não faz sentido nosso back end devolver HTML, pois teremos diferentes tipos de aplicações consumindo os dados.

Nico: concordo. Vamos criar uma API REST que retorne dados no formato JSON. O padrão REST utiliza o protocolo HTTP, e como HTTP é a web, tanto o navegador quanto a aplicação nativa em Android saberão consumir essa API para buscar, armazenar e enviar dados, inclusive executar regras de negócio.

Flávio: assim, deixaremos tanto a aplicação web quanto a nativa em Android apenas nos preocupando com a interface do usuário, que será construída com base nos dados recebidos.

Paulo: agora sim! Como não geraremos mais HTML dinâmico no back end, não estou mais amarrado a uma tecnologia específica de view de uma determinada linguagem. Posso criar o backend na linguagem que eu preferir. Pode ser Node.js?

Flávio: se você tem noção do padrão REST e sabe como implementá-lo com esta linguagem, não vejo problema nenhum. Todos concordam?

Todos: SIM!

Flávio: mas quem fará a aplicação em Android, quando chegar a hora?

Rômulo: eu farei, para vocês não reclamarem que eu não fiz nada.

Paulo: espere um pouco, pessoal. E como ficará nossa aplicação web? Se o navegador passa a receber dados no formato JSON do back end, e não mais HTML, como faremos para atualizar a view, isto é, a tela do usuário com base nos dados recebidos? Todos concordam que ela deve mudar de acordo com os dados que chegam, e com as ações do usuário, certo?

Flávio: nesse caso, o navegador só carregará a página `index.html` e todos os seus scripts e CSS de uma vez. Nenhuma outra página será carregada. Nesse sentido, podemos dizer que nossa aplicação será uma **Single Page Application**, aquela página que não recarrega durante o seu uso.

Nico: como assim, uma página que não recarrega?

Flávio: quando `index.html` for carregado, a mudança da página será feita via JavaScript com base nas ações do usuário e nos dados recebidos da API REST. `index.html` sempre será exibida, mas seu conteúdo será alterado dinamicamente através de JavaScript.

Nico: fazer tudo isso na mão? Serão zilhões de linhas de código. Como você vai organizar tudo isso? Como garantirá que funcionará nos navegadores do

mercado? Posso estar enganado, mas acho que será muito trabalhoso. Eu entendo que precisa ser assim, porque agora consumimos dados de uma API REST. E como o navegador não buscará outras páginas do back end, a única página carregada deve sofrer alterações dinâmicas no próprio navegador...

Rômulo: realmente. Como faremos para organizar o código de uma maneira padronizada, e que todos entendam? Além disso, todo o trabalho feito para criar os componentes (tabela, botões, listas) precisa ser reaproveitado. Não queremos código repetido para as mesmas coisas. E independente de tudo isso, gastaremos muitas linhas de código manipulando DOM!

Flávio: como podemos ver, precisamos solucionar os problemas de padronização, reuso de código e de manipulação de DOM, aliás, um dos pontos que mais consumirá nosso tempo.

Todos: [silêncio]

Paulo: bom, o negócio é escolher um *framework* SPA (*Single Page Application*). O Flávio tem experiência com Angular. O que vocês acham?

Flávio: estava esperando alguém se manifestar antes de eu vender o meu peixe. Se não temos um padrão para construir esse tipo de aplicação, um *framework* como Angular pode nos ajudar. Se adicionarmos outro desenvolvedor no projeto, basta que ele conheça esse *framework* para rapidamente começar a produzir, diferente do que se tivéssemos criado nosso próprio *framework*.

Paulo: e sobre a questão de reutilização de código e manipulação de DOM?

Flávio: Angular nos permite criar componentes reutilizáveis que encapsulam sua apresentação e comportamento. À medida em que formos criando nossos componentes, o tempo para desenvolver novos recursos ficará cada vez menor, pois poderemos combinar esses componentes entre si para criar novas funcionalidades. Sobre a questão de manipulação do DOM, raramente precisaremos fazer isso, pois basta definirmos um template que é a

apresentação do componente e ligá-lo a uma fonte de dados. A apresentação será atualizada toda vez que os dados associados em template mudarem. Chega de zilhões de linhas de código gastos com manipulação do DOM!

Nico: tudo muito bonito, mas sabemos que JavaScript não é uma linguagem estaticamente tipada, como Java ou C#, e por isso as IDEs ajudam muito pouco na detecção de erros enquanto estamos desenvolvendo. Nesse sentido, só saberemos que cometemos um erro de sintaxe em tempo de execução, isto é, quando nossa aplicação estiver rodando, e pode ser tarde demais. Tudo bem que um teste automatizado pode detectar problemas, mas poder detectá-los em tempo de desenvolvimento é um ganho e tanto.

Flávio: é verdade. Todavia, o desenvolvimento de uma aplicação Angular é feito na linguagem TypeScript, que nada mais é do que um superset do ES2015, ou seja, tem tudo o que o JavaScript tem, além de outros poderosos recursos. Aliás, um deles é o suporte à tipagem estática, a mesma presente em linguagens como Java, C, C#, entre outras.

Paulo: tipagem estática? Quer dizer que se empregamos a tipagem estática nossos editores e IDEs podem nos ajudar ainda mais no *autocomplete*, ou na verificação de erros enquanto estamos desenvolvendo? Como o que o Eclipse IDE faz com a linguagem Java ou o Visual Studio faz com o C#?

Flávio: exato!

Nico: como assim? Eu sei que o navegador só entende JavaScript e nenhuma outra linguagem de programação, mas como ele entenderá um código escrito em TypeScript?

Flávio: não se preocupe! A linguagem TypeScript existe apenas em tempo de desenvolvimento, pois antes do nosso código entrar em produção, ela é traduzida para JavaScript puro. Como o código que foi escrito em TypeScript não possui erros por causa da sua checagem estática, o código JavaScript gerado também não possuirá erros. É fantástico!

Rômulo: muito fantástico! Mas a infraestrutura deve ser complicada, não? Ter que configurar o projeto, compilador, *build* automático, minificar, concatenar, etc.

Flávio: que nada, há uma ferramenta chamada Angular CLI que faz tudo isso para nós. Nesse sentido, só precisamos focar na criação dos nosso componentes e mais nada.

Todos: vamos começar já!