Transcrição

Você pode ter sentido como se estivesse em uma aula de *back-end*, com todos os comandos no terminal, mas vamos voltar para o nosso universo de *front-end*, que é tratar de HTML e CSS. Neste vídeo, vamos entender a estrutura de pastas de um projeto *Angular* e como começamos a interagir com essas coisas!

Entendendo o *hot reload*

Primeiro, vamos abrir o *VS Code* e depois abrir a pasta do projeto "indexa", que está na área de trabalho. Para isso, clicamos em "*Open Folder*" ("Abrir Pasta") e vamos em "Desktop > indexa".

Feito isso, serão abertas todas as pastas existentes no projeto base. Vamos interagir com esses arquivos um pouco de cada vez, em vez de olhar tudo primeiro para depois começar a mexer. A ideia é intercalar um pouco de conceito e prática.

A primeira coisa que queremos mostrar é o arquivo `package.json`.

Conhecendo o arquivo `package.json`

O `package.json` vai configurar, por exemplo, o comando `start`, que é o `ng serve`; ele tem o nome do projeto (`name`); as dependências do projeto (`dependencies`); e as dependências em tempo de desenvolvimento (`devDependencies`), ou seja, quando fazemos o *build* e publicamos a aplicação em algum lugar, as dependências de dev não vão juntas.

Perceba que, para darmos o primeiro passo com uma aplicação de Angular, temos uma grande quantidade de dependências. Por isso o Angular CLI nos ajuda.

Conhecendo o arquivo `app.component.html`

O próximo arquivo que vamos conhecer fica em "src > app > `app.component.html`". Nesse arquivo, temos uma tag `<style>` na linha 10, que contém todos os estilos da página inicial.

Logo abaixo, temos a tag `<main>` com vários componentes, como um `<svg>`, um `<h1>Hello, {{ title }}</h1>`, e uma mensagem `<p>Congratulations! Your app is running.</p>`. Ao comparar, o que visualizamos na página da aplicação no navegador é exatamente o que está no código.

Mais uma vantagem que o Angular entrega para nós: no terminal, ainda é executado o `ng serve`. Esse comando precisa permanecer rodando, então enquanto trabalhamos, o terminal fica preso com esse comando.

Alterando o arquivo `app.component.html`

De volta ao arquivo `app.component.html`, vamos selecionar tudo e, sem medo, vamos deletar e criar um `<h1>`, que é o título principal, e vamos chamar de "Indexa".

app.component.html :

```
<h1>Indexa</h1>
```

COPIAR CÓDIGO

Alguns segundos após salvar, o navegador é atualizado para nós, limpando tudo que tinha e mantendo apenas o `<h1>`. O Angular entrega para nós o que chamamos de ***hot reload***, ou *hot module replacement*. Basicamente, o que isso faz é o seguinte: ao salvarmos um arquivo, o Angular rapidamente recompila o projeto, dispara um sinal para o navegador de que há algo novo e ele atualiza.

Assim, ganhamos velocidade de desenvolvimento. Basta salvar e será atualizado por padrão, sem necessidade de instalar nenhuma extensão no VS Code. O Angular CLI cuida disso para nós.

Aplicando o fundo gradiente azul

Agora vamos acessar o *Figma* e pegar o gradiente azul do fundo da aplicação, que começa com azul-claro e desce para o branco. Esse gradiente é uma propriedade CSS, então vamos clicar nesse fundo azul e procurá-lo no lado direito do Figma.

> Estamos trabalhando com o ***Dev Mode*** (Modo de Desenvolvimento) ligado, que fica no canto superior direito da ferramenta. Você pode alternar se quer visualizar no modo de desenvolvimento ou não.

Abaixo na aba "*Inspect*", temos a seção "*Style*" com o `background` que é um `linear-gradient` (gradiente linear). Vamos copiar esse estilo e voltar ao VS Code.

Como estamos falando de estilo, poderíamos fazer a mesma estratégia que o Angular fez de criar uma tag `<style>` e colocar o estilo dentro dela. Porém, quando falamos de **estilos globais**, o Angular entrega um arquivo CSS específico para fazermos isso.

Vamos abrir o "*Explorer*" do projeto e, na pasta "src", encontrar um arquivo chamado `styles.css`. Ao abri-lo, temos um comentário do Angular dizendo que podemos adicionar estilos globais e fazer `import` do que precisar. Vamos remover esse comentário e selecionar a tag `body`. Feito isso, vamos colar o `linear-gradient` que o Figma apresentou.

Porém, há um detalhe: no código de gradiente que copiamos, é sugerido usar uma variável CSS. Isso é uma coisa do Figma que não temos disponível, então vamos remover essa função `var(--Linear)` do CSS.

styles.css :

```
body {
    background: linear-gradient(180deg, #3D8BFD 0%, #E7F1FF 100%);
}
```

COPIAR CÓDIGO

Com isso, temos o `background` com `linear-gradient()` de 180 graus, a cor de início e a cor do fim. Podemos salvar o arquivo `styles.css`. Como resultado, temos várias réplicas do `background`, porque não definimos o tamanho do `body`, então o CSS interpretou o comando dessa forma.

Além disso, a primeira coisa que vamos fazer será remover a margem de `body`, pois ele sempre vem com uma margem por padrão. Depois, queremos dizer para o navegador que o **tamanho mínimo** do `body` é o máximo que houver disponível na tela, então definimos `min-height`, ou seja, altura mínima de 100vh.

```
body {
    background: linear-gradient(180deg, #3D8BFD 0%, #E7F1FF 100%);
    margin: 0;
    min-height: 100vh;
}
```

COPIAR CÓDIGO

Dessa forma, o que estiver disponível de tela será preenchido pelo gradiente e, assim, paramos de ter o padrão de fundo repetido e temos um gradiente igual ao do Figma.

Conclusão

De volta ao navegador, temos o título "Indexa" sobre o fundo gradiente. Missão cumprida!

Porém, há um detalhe: quando falamos em utilizar um *framework* de front-end, não é exatamente ficar apenas nos arquivos HTML e CSS; precisamos entender uma parte da aplicação que se chama **componente**.

O que isso quer dizer no final das contas? Em vez de escrevermos muito HTML e CSS, vamos escrever pequenas partes reaproveitáveis. Angular é sobre componentes, e é isso que vamos fazer no próximo vídeo. **Te esperamos lá!**