02

Hot Reload

Transcrição

Já criamos nosso projeto e fizemos a configuração das nossas ferramentas, então o que vamos fazer a seguir é começar a trabalhar nele.

Quando abrimos nosso projeto dentro do nosso simulador, vemos uma base de exemplo que foi gerada pelo próprio Android Studio. Mas não queremos que nosso aplicativo fique assim, queremos que ele seja diferente. Por isso, precisamos alterar o código do nosso projeto para começar a refletir o que realmente queremos ver.

Na parte esquerda do nosso Android Studio, temos uma estrutura de pastas, que representa tudo que está dentro de nosso aplicativo. Nesse painel, encontramos uma pasta chamada `lib`, é dentro dela que vamos trabalhar. Aqui, temos uma código `main.dart`. Dentro dele, vamos apagar tudo que vem depois da classe `MyApp`.

Com isso, o Android Studio vai indicar que a classe `MyHomePage` não existe - porque ela foi apagada por nós.

```
home: MyHomePage(tittle: 'Flutter Demo Home Page')
```

Então vamos deletar essa linha também. Em seguida, colocamos `Scafold()` e vamos salvar.

```
home: Scafold()
```

No emulador, nada vai aparecer, ele estará completamente em branco. Isso porque deletamos todo aquele código e colocamos `Scafold()`, que não é um elemento visual, mas de estrutura. Nossa página tem que ser inserida dentro de um elemento Scafold para poder ser exibida dentro do emulador.

Porém, quando salvamos não precisamos rodar de novo nossa aplicação para podermos fazer as alterações. Isso se chama *Hot Reload*, que é uma função dentro do Flutter, que só de salvarmos o código dentro do Android Studio permite que ele seja atualizado sem precisarmos ficar rodando a aplicação.

Para que apareça algo nesta página, vamos colocar um elemento filho que será um elemento de texto. Então, dentro do `Scafold()` vamos colocar o corpo dele, ou seja `body` que será um `Text()`. Esse elementos `Text()` receberá o nome da aplicação, ou seja, `Cozinhando em Casa`.

```
home: Scafold(
    body: Text('Cozinhando em Casa')
)
```

Ao voltar para o emulador, vemos que o texto não apareceu direito. Ele aparece no canto superior esquerdo, mas está muito pequeno e decentralizado, quase ilegível. Então precisaremos adicionar outro elemento para conseguirmos centralizar esse texto.

Vamos colocar fora do elemento `Text()` um elemento `Center()`, e o elemento `Text()` colocamos dentro deste novo. Porém, o Android não deixa isso acontecer. Isso porque dentro do Flutter, quando estivermos trabalhando com layout, vamos trabalhar com dois conceitos. O primeiro conceito é de composição. Seu layout geralmente vai ser composto de diversos

elementos, como aqui, tem o `Scafold()` que é estrutural e dentro colocamos o `Center()` que vai centralizar nosso texto e dentro o elemento `Text()` que é o texto que queremos centralizado. Dentro do Flutter vamos ter essa composição sempre que fizermos um layout.

Outro ponto, é que eles trabalham com hierarquias, então o elemento `Center()` precisa ter um filho. Na verdade, o elemento `Text()` é seu filho, por isso devemos colocar a propriedade `child` na frente.

```
home: Scafold(
    body: Center(
        child: Text('Cozinhando em Casa')
    )
)
```

Agora se formos no emulador, veremos "Cozinhando em Casa" centralizado. Esse é o começo do nosso layout, nos próximos vídeos vamos ver como faze-lo um pouco mais complexo.