

Widgets Stateless

Um Widget Stateless, como o próprio nome já sugere, é um Widget que não possui um estado. Isso quer dizer que esse tipo de Widget não pode sofrer alterações em seu estado em tela, ou seja, uma vez que ele é “construído” não poderá ser modificado. Dessa forma, não podemos modificá-los através de interações com o usuário. Caso seja necessário alterá-lo, necessitamos que o Flutter reconstrua ele novamente do zero, agora contando com as alterações necessárias.

De forma mais técnica, o que acontece é que um Widget Stateless possui um método Build, o qual é chamado no momento da inicialização da aplicação e seu estado não é monitorado pelo Flutter. Dessa forma, esse Widget é imutável, e não poderá ser modificado durante a execução da aplicação.

Um bom exemplo de um Widget Stateless é o Widget Text, que como o próprio nome sugere, é usado para exibir um texto na tela da aplicação. Sendo esse texto não modificável pelo usuário.

Widgets Stateful

Se um Widget Stateless é imutável, um Widget Stateful é completo contrário, pois possui um gerenciador de estado. Ou seja, seu conteúdo pode ser alterado por interações do usuário com o próprio Widget ou mesmo com outro Widget que irá alterar seu estado.

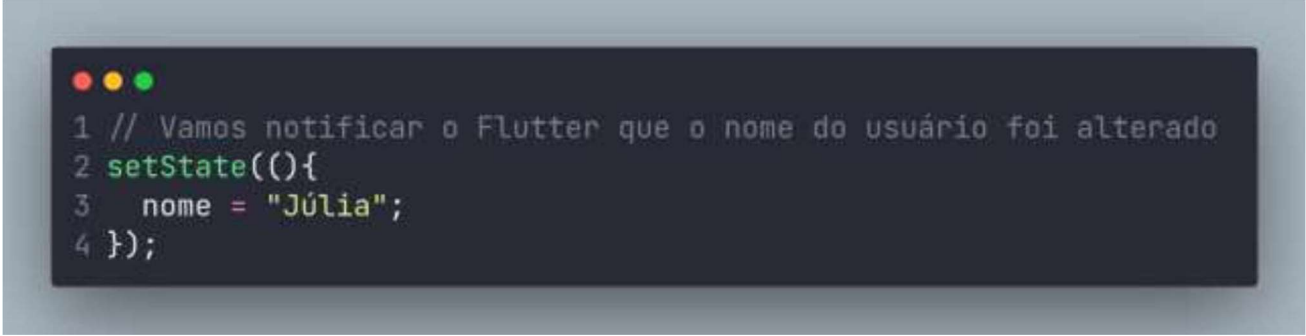
Isso é possível graças ao sistema de controle de estados do Flutter, o qual define um método setState para objetos Stateful, método esse herdado da classe State do Flutter. Esse método especial tem como função notificar que o Flutter deve atualizar a tela, fazendo com que o método Build do Widget seja novamente executado e a mesma seja reconstruída, mas agora contendo as alterações feitas no mesmo.

Esse processo é muito rápido, devido ao fato do Flutter conhecer a árvore de Widgets e saber qual parte da árvore deve atualizar, evitando assim uma reconstrução completa da tela da aplicação e deixando a aplicação mais fluída e rápida.

SetState

O método setState é um método provido da classe State, nativa do Flutter. Por esse motivo é necessário que a classe que for usar esse método herde a classe **StatefulWidget**. Pois será ela que proverá tal método

Ao executar o método, estaremos notificando o Flutter da necessidade de atualizar a tela para que as alterações possam ser vistas pelo usuário. Na prática, o Flutter irá executar novamente o método **Build**, o que fará com que a tela seja totalmente reconstruída e as alterações feitas fiquem visíveis para o usuário.



```
1 // Vamos notificar o Flutter que o nome do usuário foi alterado
2 setState(){
3   nome = "Júlia";
4 };
```

Método runApp

O método **runApp** é responsável por definir qual será o Widget raiz da aplicação. Normalmente esse Widget é um Widget **MaterialApp** ou **CupertinoApp**.

Esse método recebe como parâmetro um Widget Raiz. Para aplicações Android utilizamos o Widget **MaterialApp**, já para aplicativos IOS usamos o **CupertinoApp**.



```
1 void main() {
2   runApp(
3     MaterialApp(
4       title: "Travel",
5       home: Home(),
6       theme: ThemeData(),
7       // Demais propriedades
8     ),
9   );
10 }
```

Scaffold

O Scaffold é um dos mais importantes e significativos Widgets do Flutter, pois é ele que cria o esqueleto da aplicação Material Design.

Esse Widget possui as principais partes de uma aplicação Material, pois é ele o responsável por incluir a **barra de títulos da aplicação**, o **botão de ação** e o **menu lateral da aplicação**. Assim como outros componentes de layout de aplicação.

Agora que já entendemos a importância do Scaffold, vamos aprender um pouco mais sobre os componentes de layout que podemos incluir em nossa aplicação fazendo uso deste Widget.

AppBar

A AppBar, como o próprio nome já sugere, é a barra de navegação principal da aplicação, ou seja, aquela barra na parte superior da aplicação.

É nela que colocamos o título da tela, ou da aplicação. Também é nela que o botão de voltar é exibido ao navegarmos para dentro de uma página da aplicação.

Outro componente visual que encontramos nessa barra é o menu principal da aplicação, o qual é chamado de **actions**.

Na figura 27 podemos ver um exemplo de uma AppBar, contando com o título da página e os botões de ação da aplicação.



Figura 27 – AppBar

Fonte: Própria do autor

Propriedade title

A propriedade title, como o próprio nome já sugere, é responsável por definir o nome da tela atual. Ela deverá receber um Widget, normalmente utilizamos o Widget Text.



Propriedade leading

A propriedade `leading`, nos permite adicionar um Widget que irá representar o ícone da aplicação em nossa AppBar. Normalmente esse Widget é uma imagem ou um ícone. Para nosso exemplo prático, colocamos um ícone nativo do Flutter. Para tal utilizamos a constante `Icons` e escolhemos o ícone `adb`.

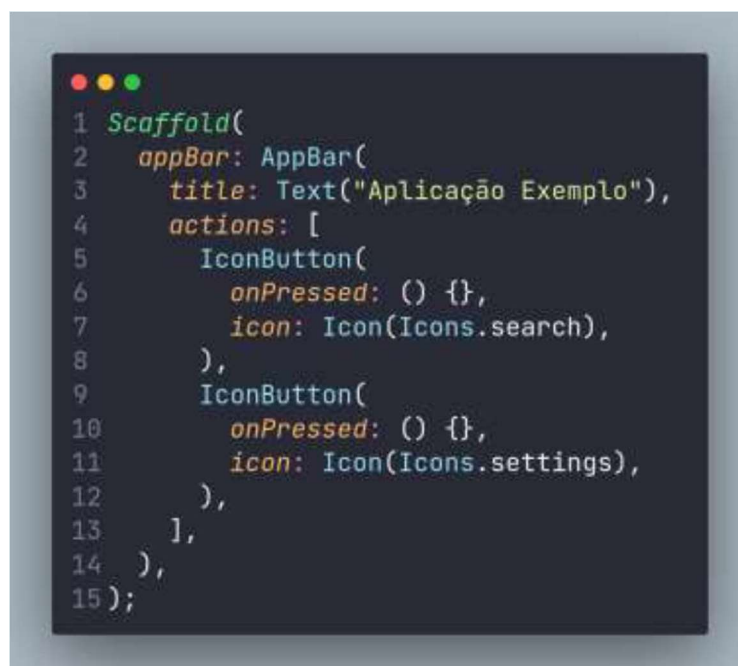


```
1 Scaffold(  
2   appBar: AppBar(  
3     title: Text("Aplicação Exemplo"),  
4     leading: Icon(Icons.adb),  
5   ),  
6 ),
```

Propriedade actions

A propriedade `actions`, nos permite adicionar uma série de botões de ação. Esses botões representam as principais ações possíveis para a tela atual.

Essa propriedade recebe uma lista de Widgets, os quais serão apresentados ao lado direito da AppBar. Normalmente usamos o Widget `IconButton` como Widget padrão. Mas podemos definir qualquer outro Widget que desejarmos, apesar do `IconButton` ser o mais recomendado para tal.



```
1 Scaffold(  
2   appBar: AppBar(  
3     title: Text("Aplicação Exemplo"),  
4     actions: [  
5       IconButton(  
6         onPressed: () {},  
7         icon: Icon(Icons.search),  
8       ),  
9       IconButton(  
10        onPressed: () {},  
11        icon: Icon(Icons.settings),  
12      ),  
13    ],  
14  ),  
15 );
```

body

A propriedade `body`, como o próprio nome já sugere, é a propriedade que recebe o corpo do Scaffold, ou seja, os Widgets que serão utilizados para compor o layout da página.

Essa propriedade, normalmente, recebe um Widget multi-child, pois esses Widgets permitem que exista uma árvore de outros Widgets dentro de si. Os Widgets mais utilizados no `body` são o `Column` e o `Row`.

Para nosso exemplo prático, vamos colocar três Widgets `Text` dentro de um Widget `Column`. O resultado final poderá ser observado nas figuras abaixo:

A screenshot of a code editor with a dark background and light-colored text. The code is written in Dart and defines a Scaffold widget. The Scaffold's body is a Column widget, which contains a list of three Text widgets. The Text widgets contain the strings "Primeiro texto", "Segundo texto", and "Terceiro texto". The code is as follows:

```
1 Scaffold(  
2   body: Column(  
3     children: [  
4       Text("Primeiro texto"),  
5       Text("Segundo texto"),  
6       Text("Terceiro texto"),  
7     ],  
8   ),  
9 ),
```