**Aula XX - DevAppII**

**Material complementar**

**Revisão de estrutura de Projeto e Código de Flutter**

**Retomada de conceitos**

**Estrutura e anatomia do Flutter**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

***Fontes:***

<https://flutter.dev/>

<https://flutter.dev/docs/get-started/flutter-for/android-devs>

<https://pub.dev/>

<https://www.flutterparainiciantes.com.br/>

<https://github.com/rubensdemelo/flutter-para-iniciantes>

<https://flutter.dev/showcase>

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Estrutura e anatomia do Flutter**

**Por que Flutter?**

De acordo com a documentação do Flutter, o Flutter é baseado em 3 pilares:

* **Desenvolvimento rápido**
* **Interfaces super bonitas**
* **Performance nativa**

**Por que escolher Flutter para o seu app?**

* Criado e mantido pelo Google;
* Mais de 170 widgets (componentes) prontos para serem utilizados;
* Principais plugins para acesso à recursos nativos do celular (bateria, câmera, conectividade, webview, etc ) também são mantidos pelo Google;
* Seu código Dart é executado diretamente pelo aparelho (em um botão, Flutter não utiliza o componente já existente de cada plataforma);
* Flutter desenha todos os pixels na tela, tornando o aplicativo altamente customizável;
* Alta performance: Aplicativos rodam em 60 frames por segundo (ou em até 120, caso o aparelho suporte);
* Alta produtividade. Alterações no código refletidas no celular ou emulador em até 0,5s. Caso precise reiniciar por completo o app, isto é feito em menos de 2 segundos;

(<https://www.flutterparainiciantes.com.br/por-que-o-flutter> )

**Overview**

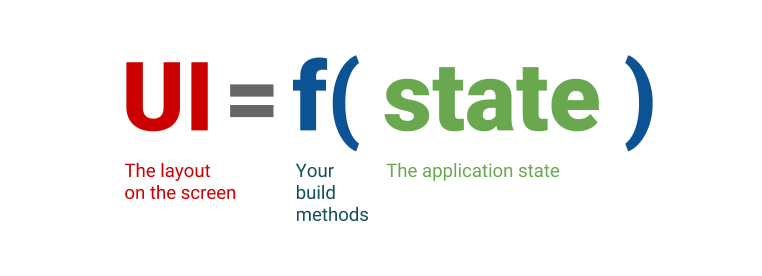
| **No Flutter, tudo é um widget!** |
| --- |

**Widget:** É um componente do aplicativo que criamos, seja um botão, um ícone, um texto, uma imagem, etc. Ao final, vários widgets irão compor o nosso aplicativo.

**Widget tree:** É a estrutura que representa como nossos widgets estão organizados. Conforme os widgets vão sendo criados e/ou destruídos, a árvore de widgets também vai sofrendo alterações. **Assim como em uma página web temos o DOM, no Flutter, temos a widget tree.**

**Declarative UI**

Este conceito merece um destaque especial, pois muda a forma como estamos acostumados a construir interfaces. Apesar de não ser um conceito novo, vem sendo cada vez mais adotado e ganhando popularidade na comunidade, devido à facilidade que ele proporciona, não só de construir, como também em entender nossa interface.



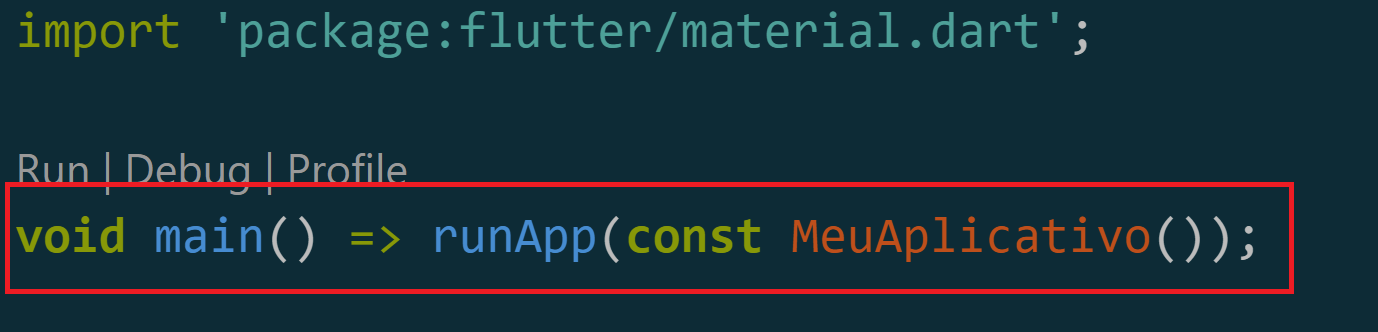
*Url da Imagem:* [*https://flutter.dev/docs*](https://flutter.dev/docs)

**Estrutura**

|  | **Android:**  Contém arquivos específicos da plataforma Android.  **iOS:**  Contém arquivos específicos da plataforma iOS.  **lib:**  Contém o código Dart do aplicativo. É possível criar pastas e subpastas aqui dentro, porém o arquivo **main.dart** deve estar na raiz.  **main.dart:**  O ponto de partida de qualquer aplicativo em Flutter. Dentro dele, deve conter a chamada ao método **main()**.  **test:**  Pasta que contém os arquivos de testes do aplicativo.  **pubspec.yaml:**  Este arquivo contém várias informações do aplicativo, como nome, versão, descrição, pacotes/dependências. Também contém o caminho para arquivos estáticos, como imagens e fontes.  **pubspec.lock:**  Contém metadados dos pacotes/dependências do aplicativo.  **Obs: A pasta .idea e o arquivo \*.iml são utilizados apenas pela IDE IntelliJ.** |
| --- | --- |

**main.dart**

O método main() é o ponto de partida para todo aplicativo Flutter. O seu retorno é sempre void.



A função **runApp()** recebe um widget como parâmetro, **ou seja, todo nosso aplicativo também é um widget.**

**Widgets**

**Em Flutter, (quase) tudo é um Widget.**

Todo aplicativo Flutter é um widget. Composto de outras centenas de widgets. Portanto, entender é mais do que a nossa obrigação. Uma coisa bacana, é que na própria **documentação do Flutter, eles deixam claro que a inspiração para os widgets vieram do React.** Assim como no framework do Facebook, todo aplicativo é baseado em componentes, no Flutter é baseado em widgets.

**"A ideia central é que você construa sua interface com widgets"**

Com esta afirmação, retirada da documentação, acredito que dá para entender, efetivamente, o que é um widget. No Flutter, vários widgets são combinados para compor um aplicativo.

| **Existem apenas 2 tipos de widgets:**  **Stateless e Stateful.**  **Podemos classificar os widgets em 2 grupos principais:**  **Layout e UI (user interface).** |
| --- |

**Widgets de layout,** são aqueles que se preocupam apenas em posicionar outros widgets.

Alguns deles - os mais usados:

* **Column:** [**Column class - widgets library - Dart API**](https://api.flutter.dev/flutter/widgets/Column-class.html)
* **Row:** [**Row class - widgets library - Dart API (flutter.dev)**](https://api.flutter.dev/flutter/widgets/Row-class.html)
* **Scaffold:** [**Scaffold class - material library - Dart API**](https://api.flutter.dev/flutter/material/Scaffold-class.html)
* **Center:** [**Center class - widgets library - Dart API**](https://api.flutter.dev/flutter/widgets/Center-class.html)
* **Stack:** [**Stack class - widgets library - Dart API**](https://api.flutter.dev/flutter/widgets/Stack-class.html)

**Widgets de interface**, são aqueles que efetivamente estão visíveis ao usuário, como:

* **Text:** [**Text class - widgets library - Dart API**](https://api.flutter.dev/flutter/widgets/Text-class.html)
* **RaisedButton:** [**RaisedButton class - material library - Dart API**](https://api.flutter.dev/flutter/material/RaisedButton-class.html)
* **Switch:** [**Switch class - material library - Dart API - Flutter**](https://api.flutter.dev/flutter/material/Switch-class.html)

Aqui ressalto as duas **principais coleções de Widgets** que já falamos em aula:

* **Material Design**
* **Cupertino**

Temos também o catálogo de Widget que podemos usar para inserir nas nossas aplicações, são mais de 170 modelos. [Widget catalog](https://flutter.dev/docs/development/ui/widgets)

**Stateless e Stateful**

No Flutter temos apenas 2 tipos de widgets:

**​Stateless:** Um widget que não tem seu estado alterado. Um texto, é um widget que não tem seu estado alterado.

**​Stateful:** Um widget que tem seu estado alterado. Um switch, é um widget que tem seu estado alterado.

Por definição, na própria documentação da API, temos!

**Um widget é uma descrição imutável de parte de uma interface de usuário**

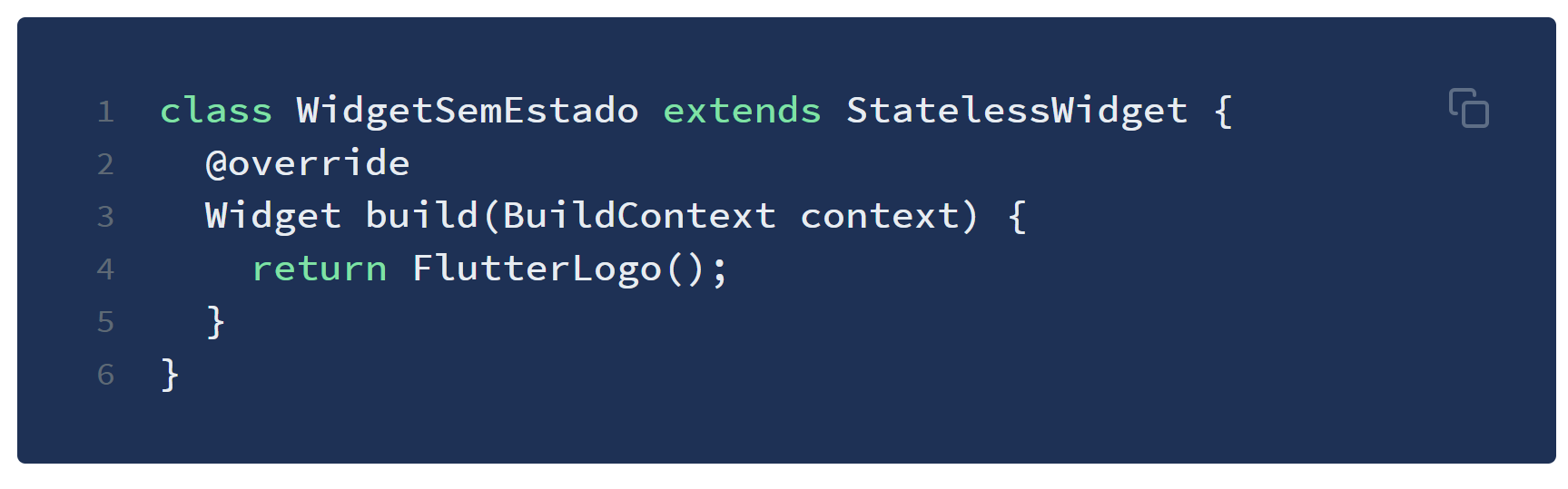
De acordo com a documentação do Flutter === **Widgets por si só, não possuem estado mutável (todos os campos devem ser finalizados (StateLess))**. Se precisarmos associar um estado mutável a um widget, considere utilizar o **StatefulWidget,** que cria um objeto do tipo State…

| **Logo, podemos assumir que com o objeto** [**State**](https://api.flutter.dev/flutter/widgets/State-class.html)**, o widget do tipo** [**StatefulWidget**](https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html) **torna-se mutável.** |
| --- |

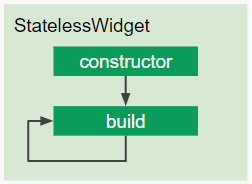
**StatelessWidget**

Criamos um widget com nome de **WidgetSemEstado** com extends **StatelessWidget**.

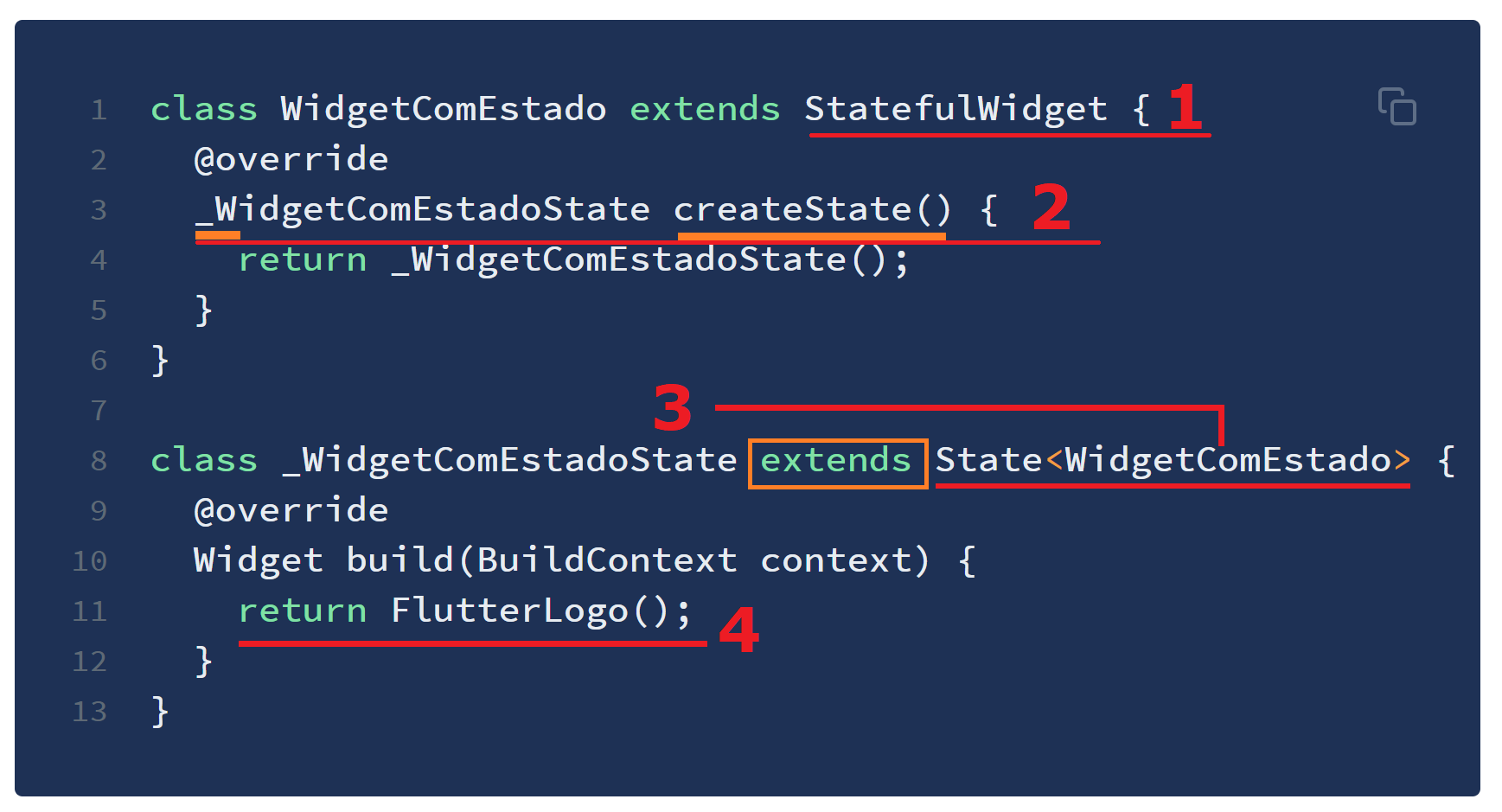
**Nosso widget abaixo retorna um FlutterLogo widget.**



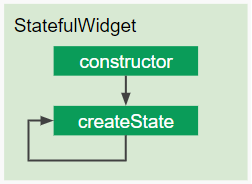
**Fluxo:**



**StatefulWidget**



1. Criamos um widget com extends StatefulWidget.
2. Nosso widget, através da função **createState()**, retorna o widget **WidgetComEstadoState.** (lembrando que o **"\_"** faz com que o objeto seja privado).
3. Como o widget **WidgetComEstadoState** extends de um objeto do tipo State, conseguimos armazenar informações, que podem ser lidas e atualizadas. E com este objeto, temos acesso ao método **setState(), que quando é executado, faz com que o nosso widget seja reconstruído. Ou seja, ao atualizarmos uma informação e executarmos o método setState, o nosso widget é reconstruído com a informação atualizada.**
4. O widget WidgetComEstadoState retorna um FlutterLogo widget.

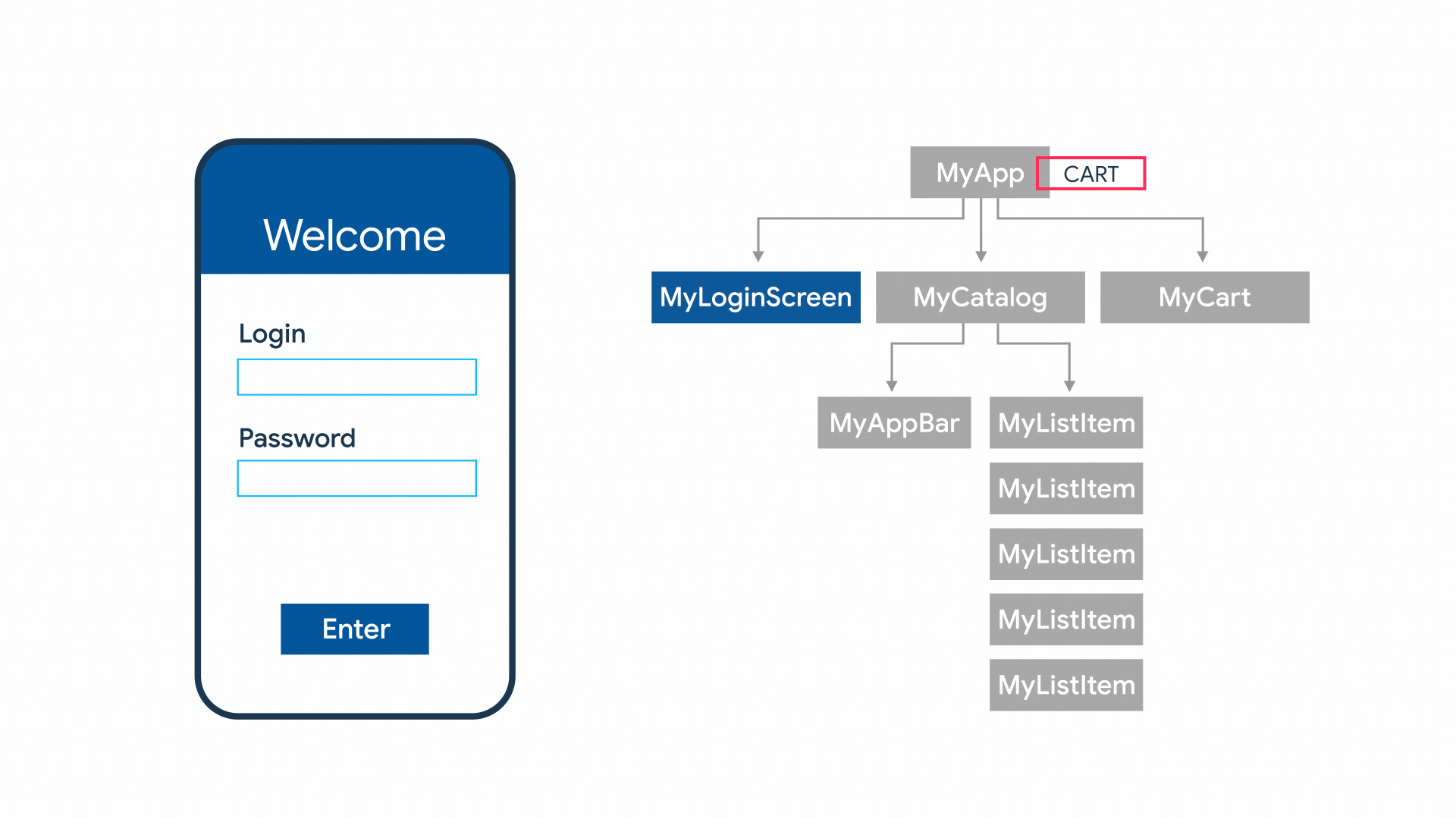
**Fluxo:**

**Widget Tree**

**A raiz de todo o "problema"**

Como já conversamos em aula, widget tree (árvore de widgets) é a estrutura que representa como nossos widgets estão organizados de forma hierárquica. Desta forma, conforme vamos construindo nosso aplicativo, compondo widgets uns aos outros, esta estrutura pode (e com certeza vai) crescendo. Estando ela cada vez maior, fica mais complicado de que a nossa interface represente o estado da nossa aplicação (declarative ui - User Interface).

A imagem e bem como o texto em itens abaixo foi retirado da documentação do Flutter e é excelente para exibir como se dá o fluxo da aplicação feita em Flutter ….. widgets.



* **MyApp - Nosso widget raiz. É por aqui que o aplicativo começa.**
* **MyLoginScreen - Tela inicial exibida ao usuário.**
* **MyCatalog - Após o usuário logar, a tela possui uma barra com acesso ao carrinho de compras e também tem acesso à lista de produtos.**
* **MyCart - Quando o usuário escolhe um produto, este vai para o seu carrinho de compras.**

Catálogo