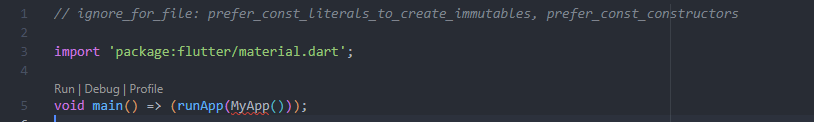
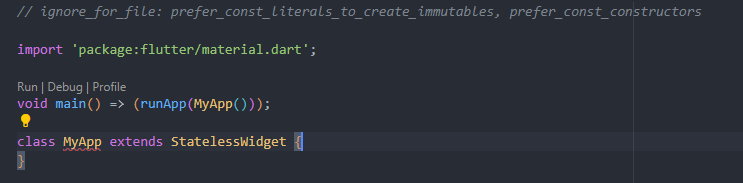
Aprendizado FLUTTER

Fontes: <https://www.dtidigital.com.br/blog/flutter/>

Criando projeto Flutter do 0

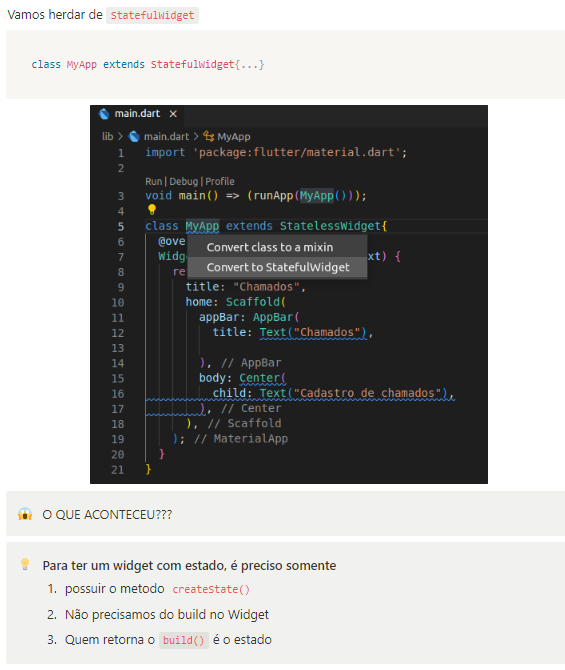


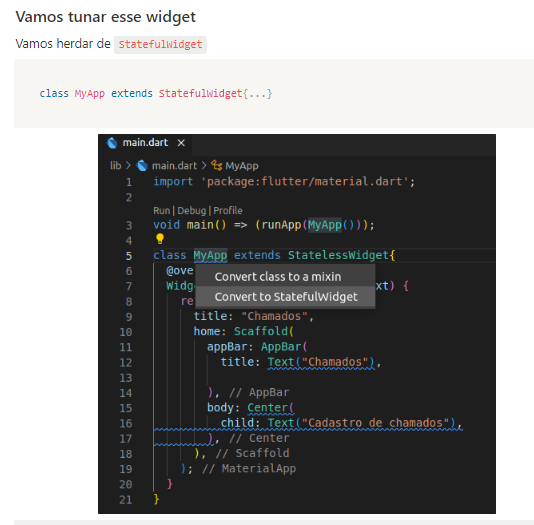


\*Erro por que precisa ciar o Build já que extente de State less



STATE FULL WIDGET







**Estrutura de arquivos**

**Android / iOS / Windows / Linux:** Código específico para cada plataforma

**.idea:** Conterá configurações relacionadas ao editor de código.

**.gitignore:** Arquivos ignorados no versionador de código

**.metadata:** Contém metadados exigidos pela ferramenta flutter para rastrear o projeto flutter.

**.packages:** Arquivos para todas as bibliotecas usadas no projeto

**pubspecam.lock:** Versão das dependências do projeto

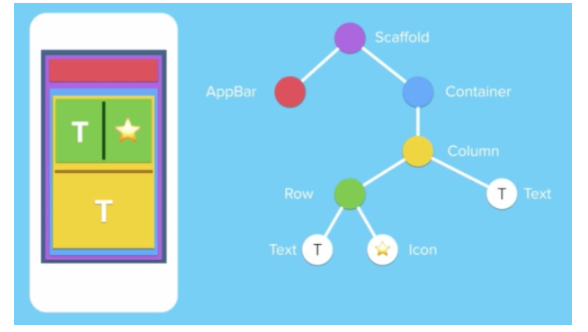
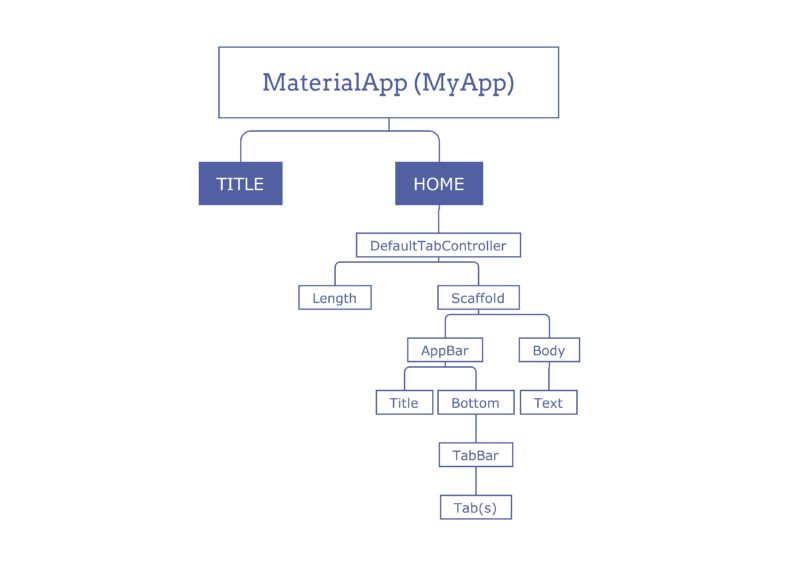
**pubspec.yaml:** Usado para adicionar metadados e configurações específicas ao nosso aplicativo. Com a ajuda deste arquivo, podemos configurar dependências como ativos de imagem, fontes e versões de aplicativos.

**README:** Este arquivo markdown é usado para descrever seu aplicativo no repositório GitHub.

l**ib:** Pasta mais importante do projeto, usada para escrever a maior parte do código. Contém o arquivo main.dart, ponto de partida do aplicativo.

De acordo com que os Widgets são utilizados para construção da interface é construída uma árvore de Widgets. Por exemplo, para uma tela utilizamos um Widget base chamado Scaffold, e dentro dele podemos posicionar Widgets de texto, ícones e imagens utilizando outros Widgets como Column, Row, e Container.

Na imagem abaixo podemos ver a raiz da árvore com o Widget Scaffold (em roxo), e como filhos dele temos uma AppBar e um Container. Como filho do Container temos um Widget para posicionamento de texto e outro para posicionamento de outros Widgets em linha (Row). Como filhos da Row temos um Widget para texto e outro para ícone.



Principais Widgets:

* [Container](https://api.flutter.dev/flutter/widgets/Container-class.html): utilizado para agrupar um Widget. É possível definir propriedades como cor e tamanho;
* [Column](https://api.flutter.dev/flutter/widgets/Column-class.html): utilizado para agrupar Widgets em coluna. Ele recebe uma lista de filhos (Widgets), ao invés de um único filho;
* [Row](https://api.flutter.dev/flutter/widgets/Row-class.html): utilizado para agrupar Widgets em linha. Ele recebe uma lista de filhos(Widgets), ao invés de um único;
* [ElevatedButton](https://api.flutter.dev/flutter/material/ElevatedButton-class.html): utilizado como botão. Ele tem um atributo “*onPressed*“, que executa uma função quando pressionado, é um *child* como um widget filho (como *text* para fazer um botão com texto no centro);
* [ListView](https://api.flutter.dev/flutter/widgets/ListView-class.html): utilizado para criar listas e scroll na tela. Recebe uma lista de filhos(Widgets), ao invés de um único filho;
* [FutureBuilder](https://api.flutter.dev/flutter/widgets/FutureBuilder-class.html): utilizado para renderizar a tela de acordo com um certo estado. Usamos ele para gerenciar uma chama assíncrona, mostrando uma tela de loading enquanto não for completado, e quando a API retornar mostrar os dados;

**Usando MaterialApp e Scaffold**

MaterialApp: Widget que faz o container para criação de aplicativos Material Design.

Scaffold: Representa uma tela em um widget MaterialApp , pois pode conter vários widgets de layout do Material Design, incluindo AppBar, barra de navegação, botões e o corpo da tela em si.



Precisamos repassar o conceito para nossa tela.

De maneira geral, o Builder possui a capacidade de fazer um wrapper para que possamos acessar o contexto em que ele foi construido. Ele da o contexto em que foi construido de forma atualizada sempre que precisar

