CREA jr-BA
Um novo olhar para o futuro

recode

# Introdução ao Flutter com MobX e Modular

Orlando Eduardo Pereira
BCET-Computação-UFRB
Recode Jr. - DAF
Mobile Developer

# Quem sou?

- Estudo BCET/Eng. De Computação
- Diretor Administrativo Financeiro - Recode Jr
- Mobile Developer
- 1,5 ano com java
- 10 meses com Flutter
- Instagram: @smaaalll
- GithHub: OrlandoEduardo101
- Youtube: Orlando Eduardo Pereira

# O que é Flutter??

Flutter é um SDK de código aberto criado pelo Google para o desenvolvimento de aplicativos para Android, iOS, Desktop ou Web, além de ser o principal método de criação de aplicativos para o Google Fuchsia.

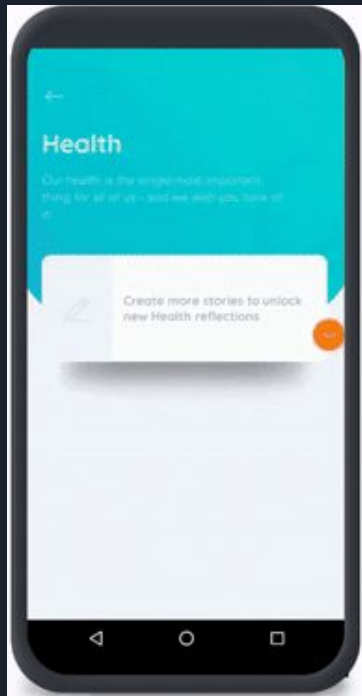
O Flutter é:

- Um framework reativo moderno,

- Um mecanismo de renderização em 2D rápido,

- Ferramentas para desenvolvimento

- Widgets prontos, que constituem os componentes da IU do aplicativo.



# Qual a linguagem?

- Originalmente se chamava Dash
- Linguagem de script para web
- Lançada em 2011
- Substituir o JavaScript
- Pode ser executada em VM ou compilada para JS
- Dart 1.0 lançado em Nov. de 2013, primeira versão estável
- Dart 2.0 lançado em Ago. 2018, Reboot
- Otimizado para Web e Móveis

# Porque usar Flutter?


- Desenvolvimento rápido
- Interfaces super bonitas
- Performance nativa
- Criado e mantido pelo Google e pela comunidade
- Comunidade ativa e crescendo
- 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
- 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

# Algumas considerações sobre Dart

- Tipada, porém isto é opcional.
- Morderna (generics<T>, funções, interfaces e mixins).
- High Order Functions ( forEach(), map(), etc ).
- Utiliza sintáxe C-style (como C#, Java, Javascript).
- Pode ser compilada e interpretada.
- Também pode ser aplicada ao Back-End (Aqueduct)
- testes : https://dartpad.dev

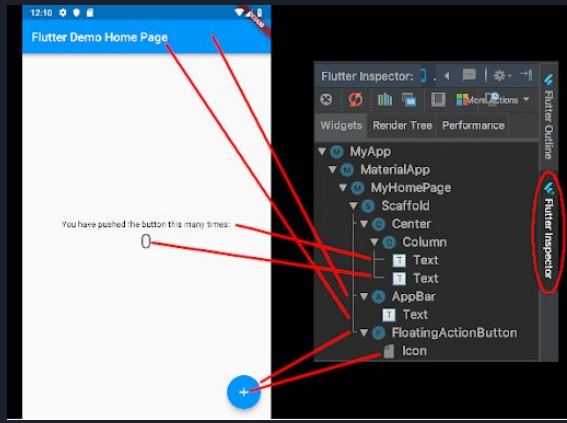
```dart
var A = "Fala Zezé,";
String B = "Bom dia cara.";

void main() {
    print(A + B);
}
```

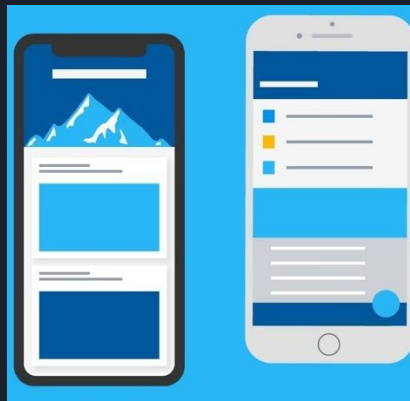
```
Console

  Fala Zezé,Bom dia cara.
```

# “No Flutter, tudo é um widget”

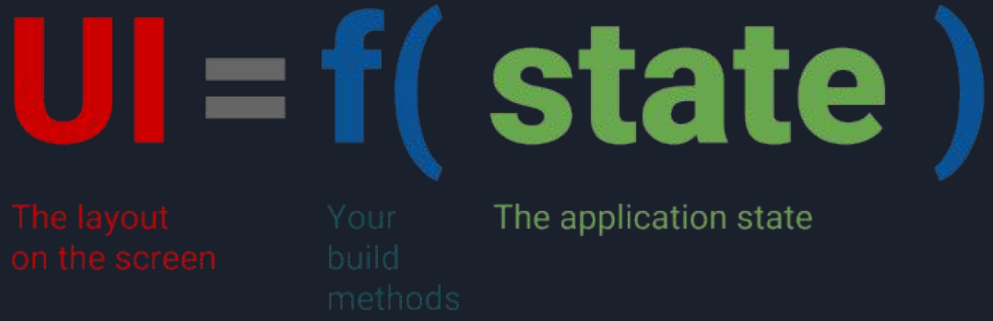



- Widget: É um componente do seu aplicativo, seja um botão, um ícone, um texto, uma imagem, etc. Ao final, vários widgets irão compor o nosso aplicativo.
- Widget tree: É a estrutura que representa como nossos widgets estão organizados.
- pub.dev: Gerenciador de pacotes da linguagem Dart.
- Package: Um módulo/pacote, puramente com código Dart.
- Plugin: Contém código específico da plataforma (java/kotlin, swift/obj-c), geralmente faz acesso à algum recurso nativo, como bateria, câmera.
- Material package: Conjuntos de widgets que seguem as definições do Material Design.
- Cupertino package: Conjunto de widgets que seguem as definições de interface do iOS.

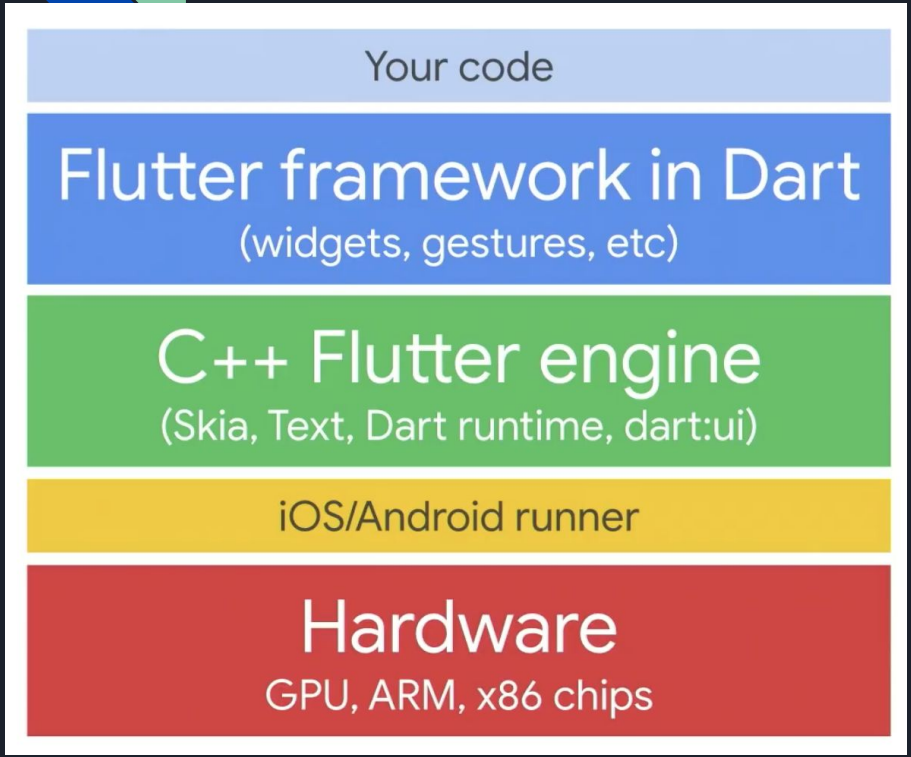


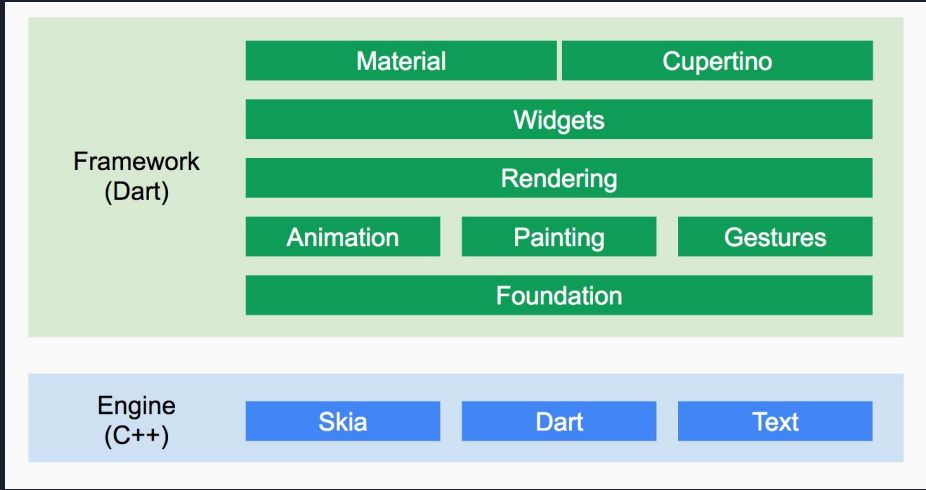
# Declarative UI

Nossa interface, reflete o estado da nossa aplicação. Sempre que o estado da aplicação muda, os widgets são reconstruídos para atender aquele novo estado. Seja um botão ou ícone que muda de cor após ser pressionado ou até mesmo toda uma tela após o usuário logar no aplicativo.



# Arquitetura



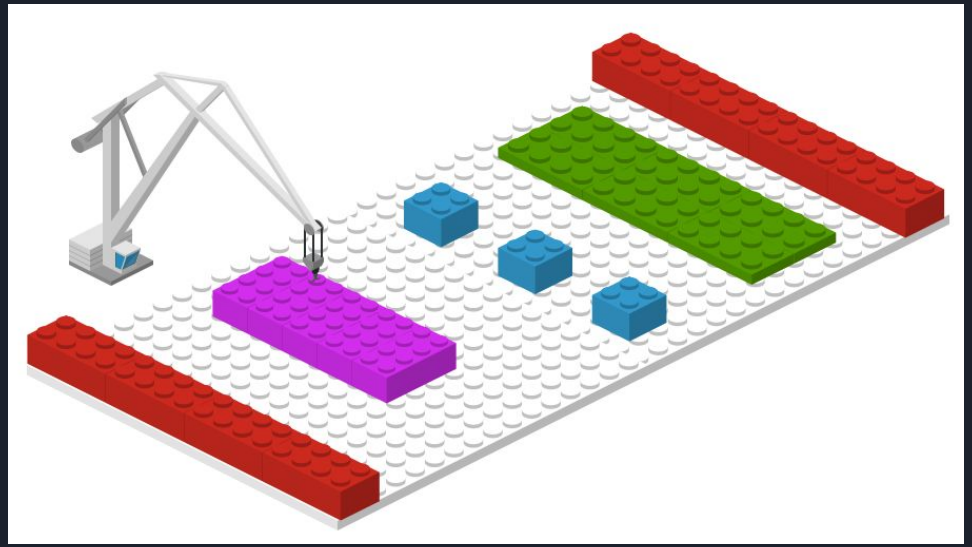




- Your code: Dart, Kotlin, Swift, Packages e Plugins
- Flutter: o framework em si.
- Engine: une o seu codigo e o Framework e prepara pro hardware.
  - Usa a biblioteca Skia e o processador de texto do Chrome.
- Runners: delega e controla os binários ao hardware.
- Hardware: Processador, RAM, Chips...

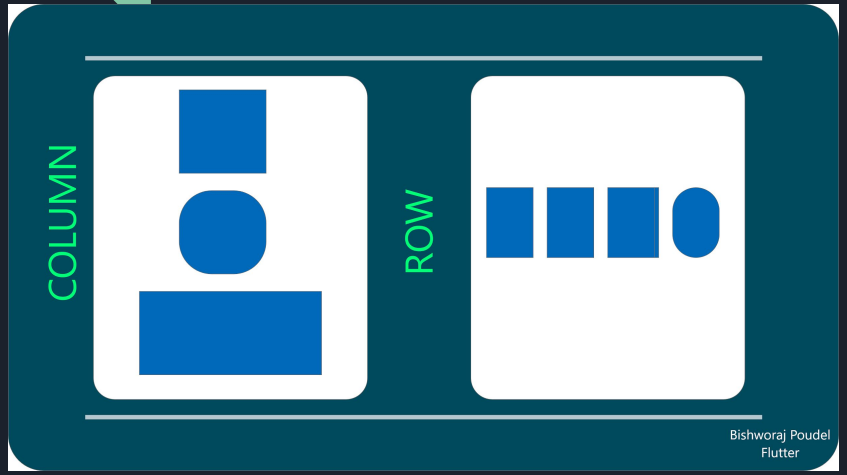
# Widgets

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

- Cada Widget é como uma peça de lego, existem tamanhos diferentes que podem ser acopladas para formar um objeto maior.



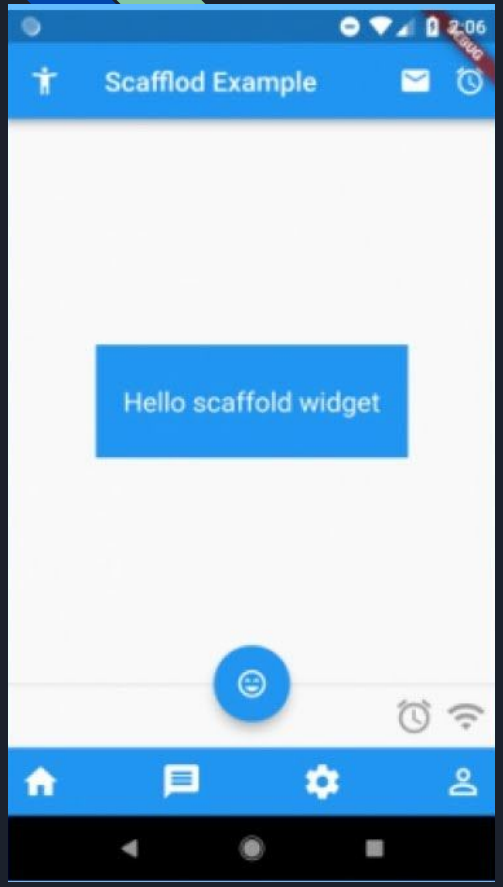
# Column & Row

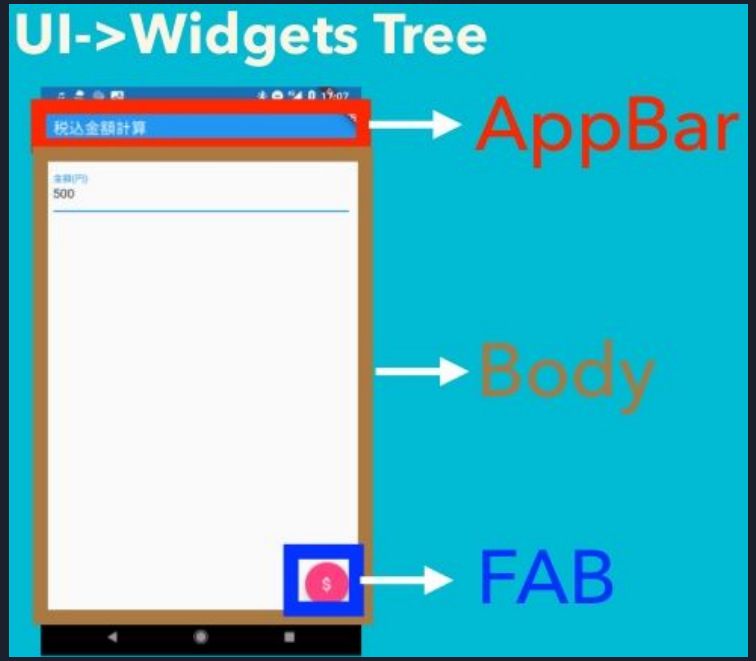




- Widgets de estrutura de layout, neles é possível colocar vários widgets alinhados.

# Scaffold (“Andaime”)



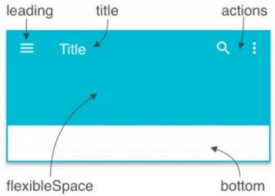




- Estrutura básica padrão para criar um layout rápido
- Material Design
- Possui APIs para construir Snack Bars, Bottom Navigator, Body, Float Action Button...

## Scaffold


UI / UX

## AppBar



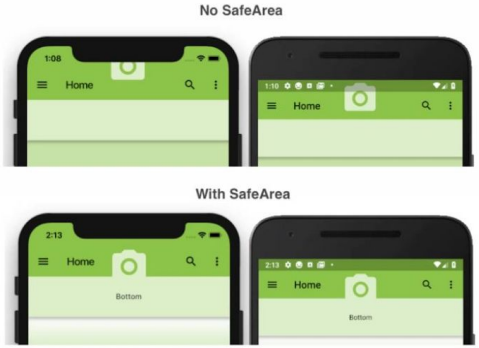

## Text

```
onPanUpdate:
DragUpdateDetails(Offset(0.3, 0.0))
```

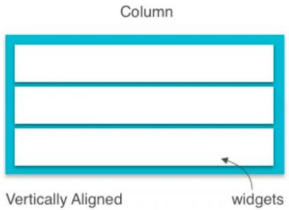
## RichText

*Flutter World for* **Mobile**

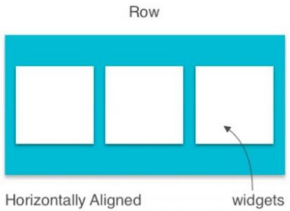
## SafeArea




## Column




## Row




## Container

## Button

# Stateless x Stateful

Stateless: Um widget que não tem seu estado alterado.

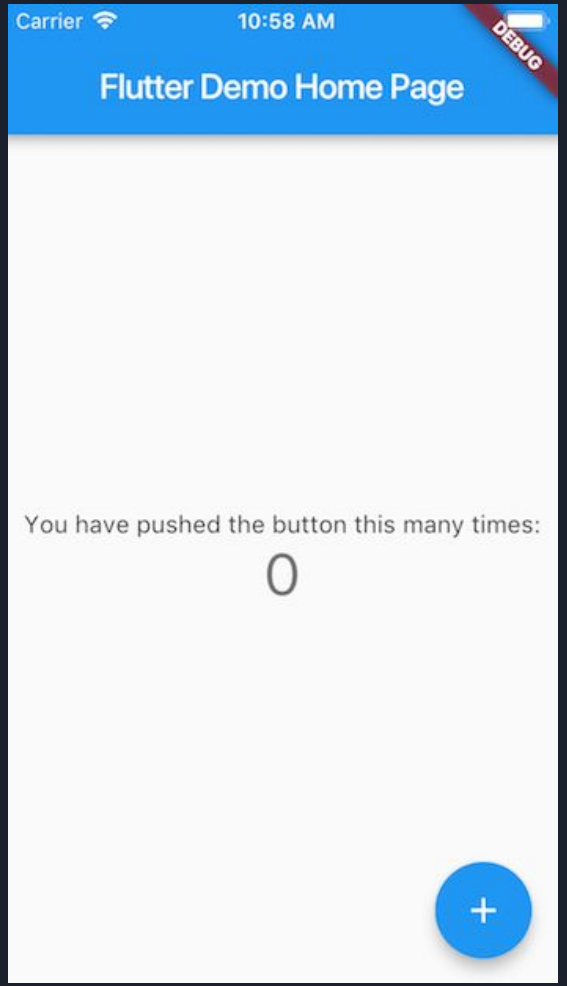
- Um texto, é um widget que não tem seu estado alterado.

Stateful: Um widget que tem seu estado alterado.

- Um switch, é um widget que tem seu estado alterado.

# Comandos básicos

- flutter doctor (analisa se as ferramentas estão ok)
- flutter --version (mostra a versão)
- flutter devices (exibe dispositivos disponiveis)
- flutter create meu_app (cria um projeto)
- flutter create --org <com.recodejr> meu_app (cria um projeto com domínio)
- flutter run -d <id-device> (executa o app)
- flutter run -d all (executa em todos devices)
- hot  reload e hot restart



# Hello World

```dart
import 'package:flutter/material.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Welcome to Flutter',
      home: Scaffold(
        appBar: AppBar(
          title: Text('Welcome to Flutter'),
        ),
        body: Center(
          child: Text('Hello World'),
        ),
      ),
    );
  }
}
```

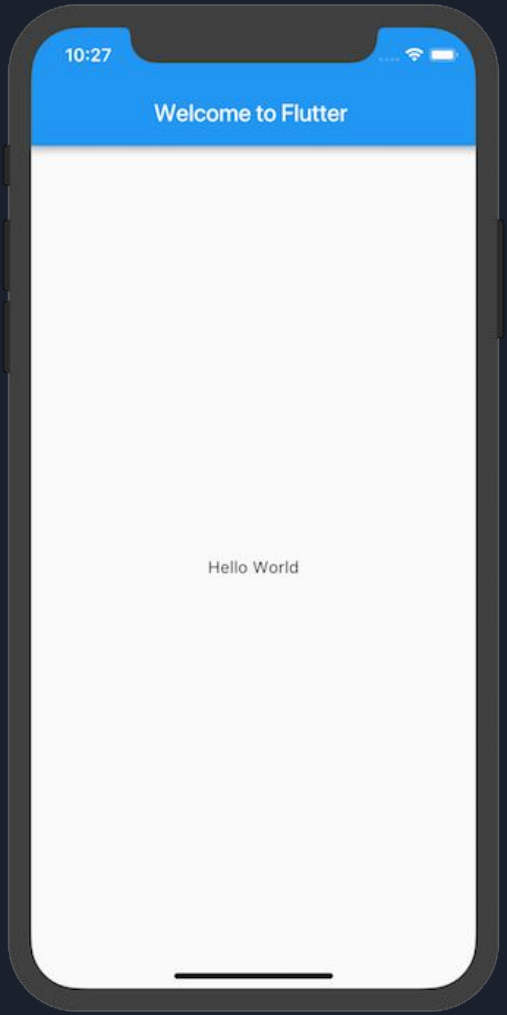
- Documentação

```dart
import 'package:flutter/material.dart';

void main() {
 runApp(
   MaterialApp(
   home: Home(),
 )); // MaterialApp
}

class Home extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Bem vindo ao Flutter"),
      ), // AppBar
      body: Center(
        child: Text("Olá Mundo!"),
      ), // Center
    ); // Scaffold
  }
}
```

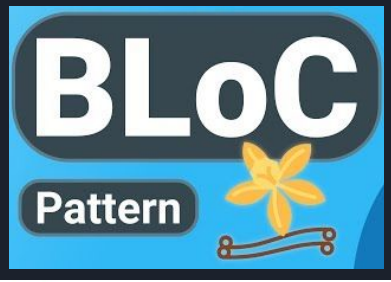
Meu Hello World



# O que é Gestão de estados?

Geralmente o aplicativo é dividido em duas partes, a UI e o Estado, as UI correspondem ao estado da aplicação, ou do componente, naquele momento, logo, gerenciar esses estados em uma aplicação grande é imprescindível e não é uma tarefa simples.
Precisamos fazer com que a view seja notificada quando o estado alterar para que a view seja atualizada.

Gestores de Estados:
- Set State
- ChangeNotifier
- BloC
- GetX
- MobX
- TemCoisaBagaraio


# O que é MobX?

O MobX é um pacote que implementa a gestão de estado no Flutter de uma forma simples. Na verdade, ele escreve o código para nós por baixo dos panos.

Observables :

Observáveis são porções de estado (variáveis, listas, objetos, campos de uma classe, etc) que notificam ouvintes (as reações) quando há mudanças. Observáveis são, em geral, definidos usando a função observable, exportada pelo módulo mobx.

Actions:

Funções que alteram o estado dos Observables

Observer:

Widget que vai observar os observáveis e reconstruir a parte da tela correspondente, ou a tela inteira.

Code_Gen:

Package para gerar codigos com notations

```dart
class Counter {
  Counter() {
    increment = Action(_increment);
  }

  final _value = Observable(0);
  int get value => _value.value;

  set value(int newValue) => _value.value = newValue;
  Action increment;

  void _increment() {
    _value.value++;
  }
}
```

Mobx sem code_gen

Mobx com code_gen

```dart
import 'package:mobx/mobx.dart';

part 'counter.g.dart';

class Counter = CounterBase with _$Counter;

abstract class CounterBase with Store {
  @observable
  int value = 0;

  @action
  void increment() {
    value++;
  }
}
```

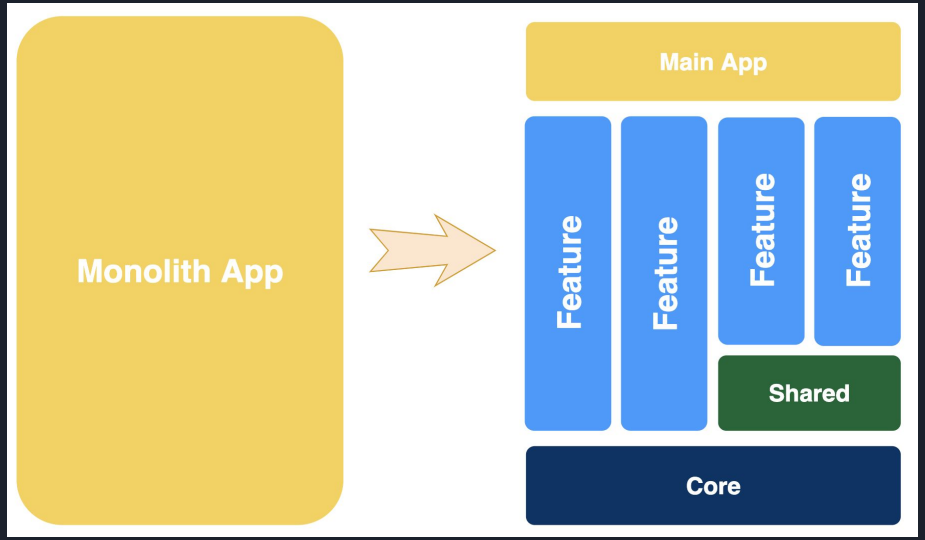
# Arquitetura Modular

Cada módulo é como se fossem mini apps dentro de um app,  com suas próprias camadas de códigos, que fazem parte de um módulo maior que é o próprio app, cada módulo também podem ter seus próprios sub-módulos e assim por diante.

MVC ? MVVM? ...
- Um pouco de cada

Facilitação do trabalho em equipe

Reaproveitamento de códigos



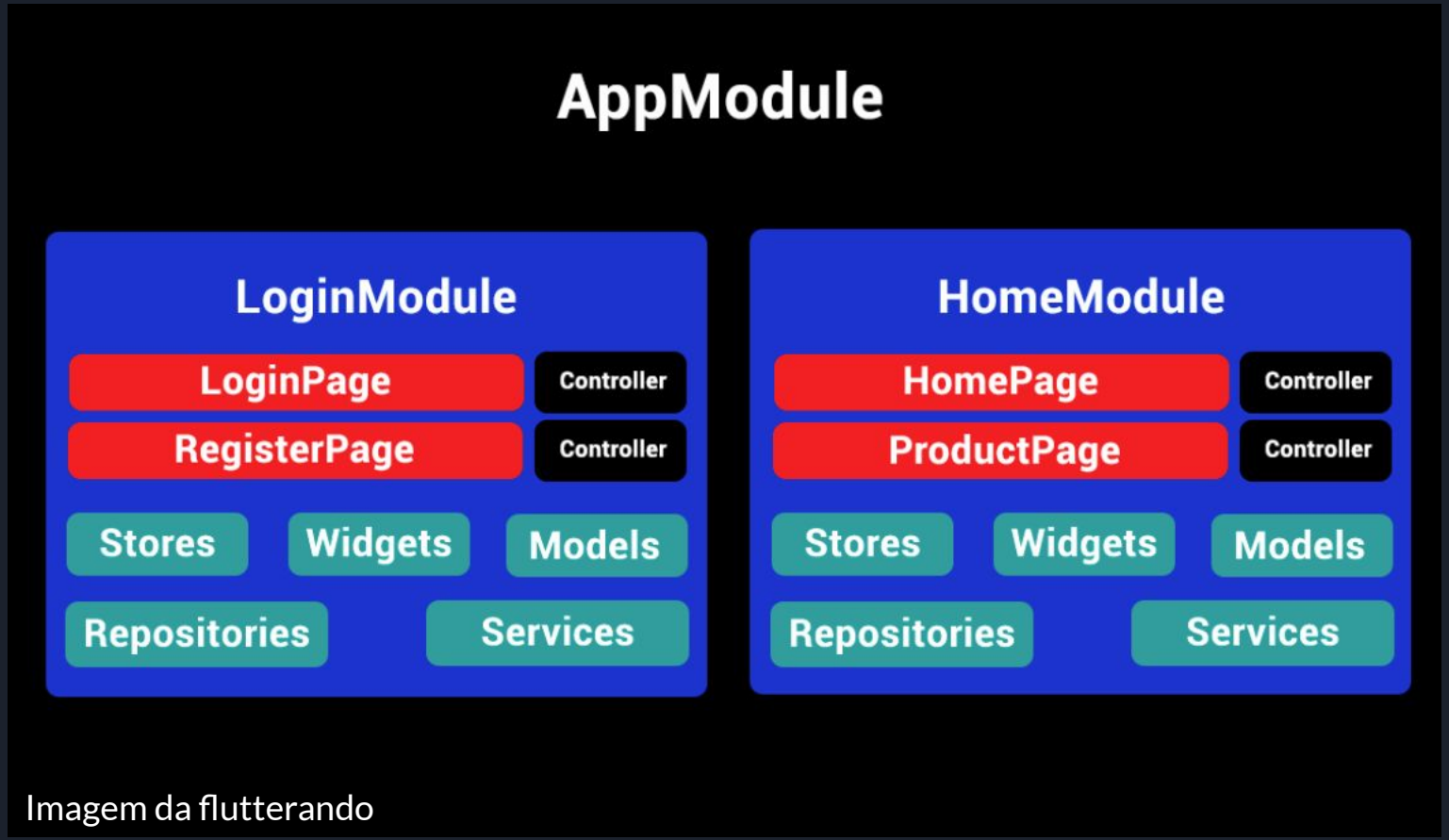



Imagem da flutterando

# Camadas

alguns princípios S.O.L.I.D:

- Cada camada, uma responsabilidade
- Princípio de inversão de dependências

View:

Interface de interação com o usuário

- Page: tela completa
- Widget: componentização da Page

Controller:

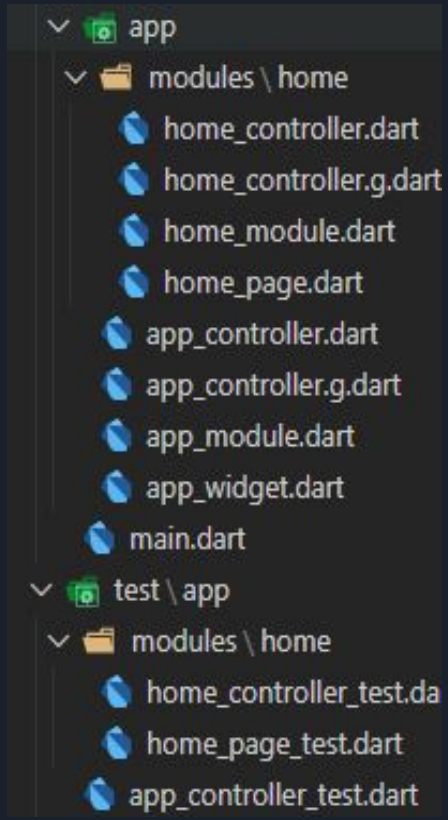
- Regra de negócios

Model:

- Modelos dos objetos
- Esquematização e Regra de negócio*
- métodos de conversão

ViewModel:

- Componentização do controller
- pode ser chamado de Store ou BloC



Services:

- Comunicação com serviços do sistema
- Câmera, DB local, BT...

Repository:

- Comunicação com API externas
- Cliente HTTP

Shared ou core:

- Itens globais

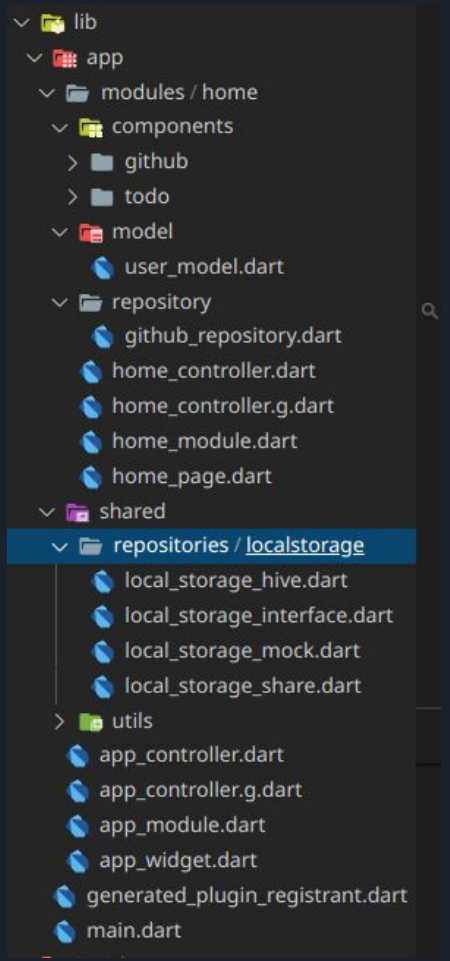
Module:

Injeção de Dependência:

- Injeção de instâncias por meio de construtores
- Flutter Modular
- Binds (CodeGen)

Rotas Nomeadas:

- Melhor controle do build de views
- Ganho de performance

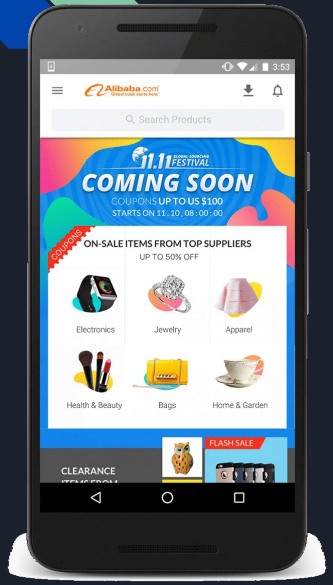


# Slidy CLI

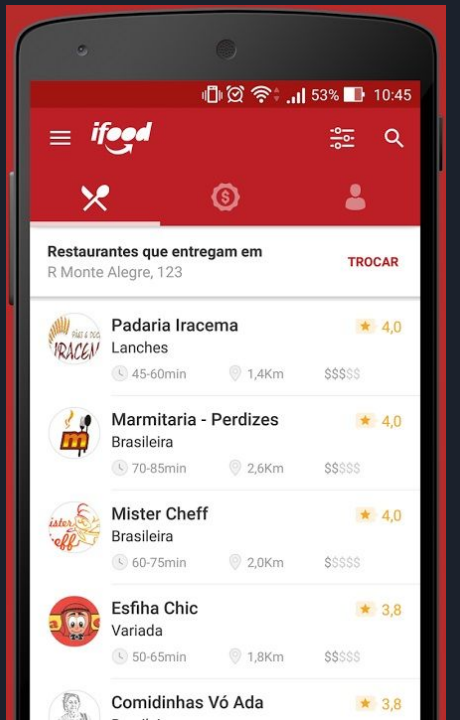

- Dart VM
- slidy upgrade
- slidy create **myproject**
- slidy create --org com.orlando-eduardo101 **myproject**
- slidy start
- Slidy run
- Slidy install <package>
- Slidy update <package>
- Slidy generate

```
vars:
  runner: flutter pub run build_runner
  clean: flutter clean
  get: flutter pub get
scripts:
    mobx_build: $clean & $get & $runner build --delete-conflicting-outputs
    mobx_watch: $clean & $get & $runner watch --delete-conflicting-outputs
```

# Apps famosos que usam Flutter





# Bibliografia


- Doc Flutter: https://flutter.dev
- Doc Dart: https://dart.dev
- DartPad: https://dartpad.dev
- Introdução: https://www.flutterparainiciantes.com.br/o-que-e-flutter
- Fluterando:
  - Slidy + Modular: https://www.youtube.com/playlist?list=PLlBnICoI-g-dCE_JiJd7bJnEYbigkX7pq
  - MobX: https://www.youtube.com/watch?v=kKNIaqZE8CY&list=PLlBnICoI-g-foW-Osr0PIpE1_AD3aItbZ
  - Arquitetura: https://www.youtube.com/watch?v=8Iqj7YQ71Io&list=PLlBnICoI-g-c_ZIHqzQjg5E4Re92-qYXn
  - ToDo List Firebase: https://youtu.be/D7X62KPDkMc (procure o video com Hasura tbm)

# Bibliografia


- Balta (artigos)
  - State Manager: https://balta.io/blog/flutter-setstate-bloc-mobx-state-management
  - MobX: https://balta.io/blog/flutter-mobx
- Docs:
  - Modular: https://pub.dev/packages/flutter_modular
  - MobX: https://pub.dev/packages/mobx
  - Slidy: https://pub.dev/packages/slidy
- Linkedin: https://www.linkedin.com/in/orlandoeduardopereira/
- GitHub: https://github.com/OrlandoEduardo101
- Instagram: https://www.instagram.com/smaaalll/
- Youtube: https://www.youtube.com/channel/UC2I-66HOudxk-hy3PrpdEdQ?view_as=subscriber