INF1300: Tópicos em Computação I

Desenvolvimento Ágil de Aplicativos Móveis Multi-Plataforma

(*Usando Flutter e Dart)*

Prof. Markus Endler
Felipe Carvalho

http://www.inf.puc-rio.br/~endler/courses/Flutter

LAC
Laboratory for Advanced Collaboration

Pontifícia Universidade Católica
do Rio de Janeiro

Objetivos e Ementa

Objetivos:

- Domínio prático dos conceitos fundamentais do framework **Flutter** e da linguagem de programacão **Dart**;
- Compreensão e prática dos conceitos e técnicas de programação (básicas e avançadas);
- Utilização de plug-ins e módulos
- Obtenção de experiência com o projeto e teste de aplicativos para Android e iOS.

Objetivos e Ementa

Objetivos:

- Domínio prático dos conceitos fundamentais do framework **Flutter** e da linguagem de programacão **Dart**;
- Compreensão e prática dod conceitos e técnicas de programação (básicas e avançadas);
- Utilização de plug-ins e módulos
- Obtenção de experiência com o projeto e teste de aplicativos para Android e iOS.

Ementa:

- Visão geral do framework Flutter
- Introdução à Programação Reativa
- Conceitos de Widgets, Widget tree, coleções, listas, mapas, Futures, Stream Objects
- Estrutura de um aplicativo Flutter
- Exemplos de Aplicativos
- Acesso (sincrono e assíncrono) a servidores remotos
- Inferface a funções e recursos, e sensores nativos
- Plug-ins e Packages

Avaliacão e Datas

- Tudo pode ser feito em dupla.
- Não haverá prova, mas dois trabalhos práticos (T1 e T2) , e uma apresentação oral (A1) para a turma (um “pitch”). No pitch os dois deverão falar!
- Datas:
  - T1 em 18/maio,
  - T2 em 29/junho.
  - A1 em 29/junho.

- Média = (3*T1 + A1 e T2)/5
- As Duas melhores duplas receberão uma camiseta StarsOfScience

- **PS: Será feita chamada. Exige-se pelo menos 75% de presença.**

Camisetas StarsOfScience

Camiseta temática "Stars of Science"

- Homenagem a 19 pioneiros da Computação: A.Turing, J. Backus, D. Knuth, K. Thompson, D. Ritche, L. Lamport, N. Wirth, etc.
- vestir uma camisa SoS não é ser nerd… mas
- homenagear pessoas que "fizeram a diferença" para a computação e contribuiram para o progresso da humanidade

Avaliacão: Entregáveis e Critérios

Entregáveis:

- Para T1 e T2: Código do aplicativo móvel (código Dart) e um relatório (PDF) sobre o apicativo
- Para a A1: o ppt

Obs: T2 necessariamente deverá ser ter o Cliente como ProSumer (Producer+Consumer) de serviço na cloud ou servidor.

Critérios de avaliação (com os seguintes pesos):

- Relevância(*) da aplicação escolhida (**peso 2**)
- Qualidade/ robustez dos apps T1 e T2 desenvolvidos (**peso 3**)
- Utilização de todos os recursos solcitados (**peso 2**);
- Iniciativa para aprender e usar outros recursos não ensinados (**peso 1**)
- Qualidade da infercface gráfica do app (**peso 2**)

Aulas

Serão em formato de laboratório:

1. Curta exposição sobre os conceiros/ técnicas a serem usadas naquela aula;
2. alunos farão pequenos exercicios de programacao Dart/Flutter, modificando ou estendendo modelos de programas, obtidos no git (https://bitbucket.org/account/user/endler/projects/FLUT)

Obs:

- A aula começará sempre as 9:10.
- Só serão tolerados atrasos até as 9:30.
- A presença em sala durante algum período de tempo qualquer (exemplo: chegada nos ultimos 20 minutos da aula) não dá direito a marcação da presença.

Orientação e Monitoria

- Será dada ajuda durante as aulas-laboratório;
- Dúvidas não derão tiradas por Email, só presencialmente;
- Uma frase que vcs irão ouvir bastante de mim:

  "Ih, isso também não sei, vamos perguntar ao Felipe"

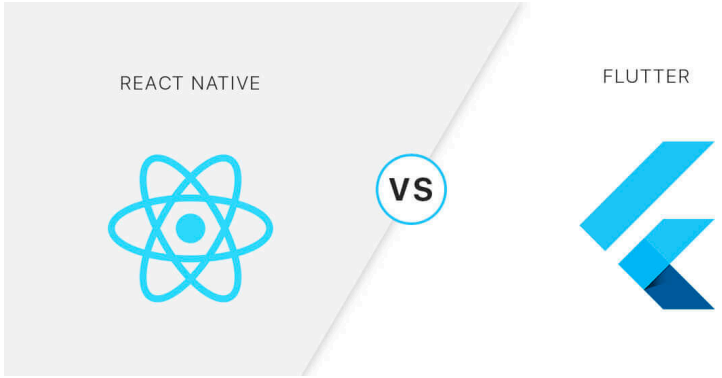
- Antes das aulas, será disponibilizado um novo modelo de programa no Git.

- Recomenda-se trazer o próprio notebook (de preferência um macbook).

Flutter

É um framework multi-plataforma desenvolvido pela Google com muitos recursos e ferramentas

Está dando o que falar, e muitas oportunidades de emprego estão surgindo. Muitas empresas (start-ups) estão abandonando **React Native**

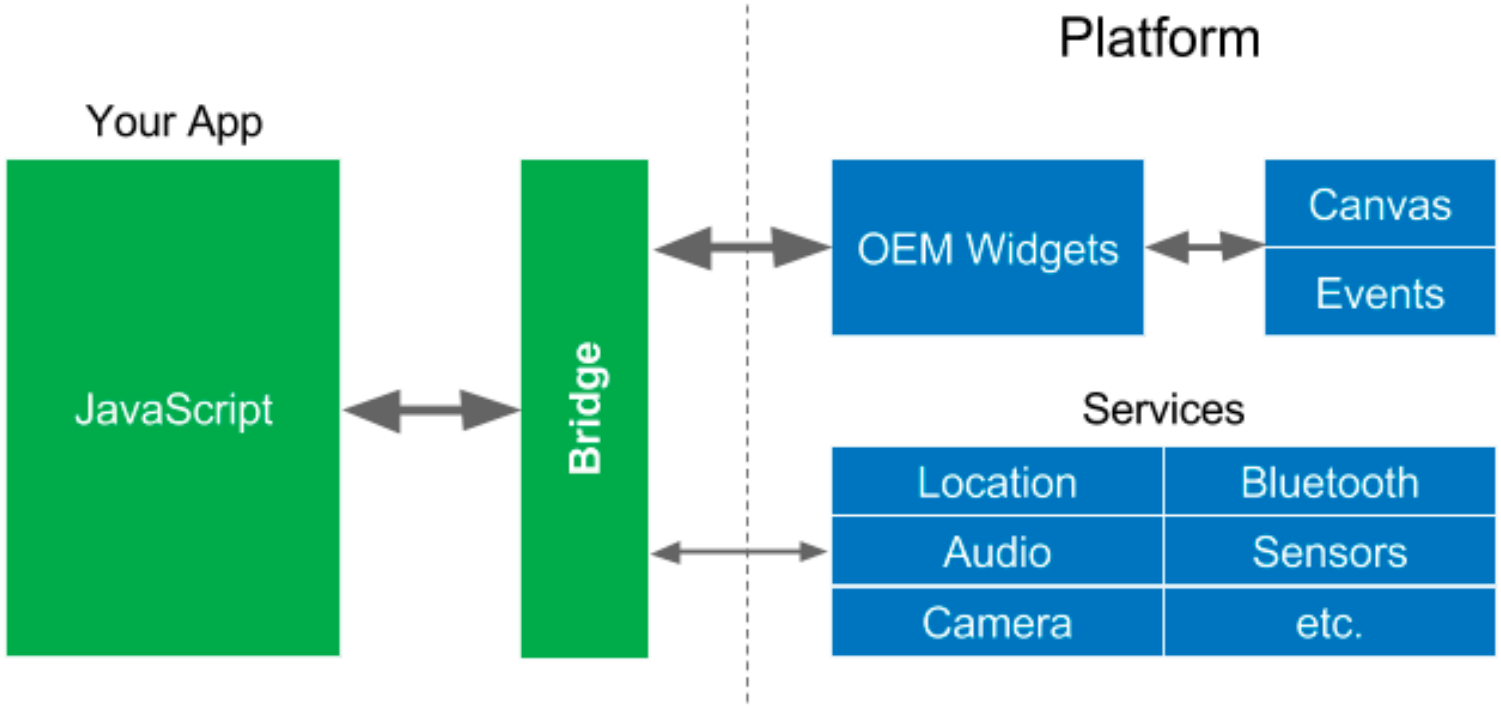


Além de gerar código nativo para as plataformas nativas (ganho enorme de desempenho), Flutter ainda tem o *hot reload*.

Exemplo: Mude o visual ou o comportamento dos widgets, faça um ctrl+s e pronto, quase que instantaneamente o resultado está renderizado!

React Native

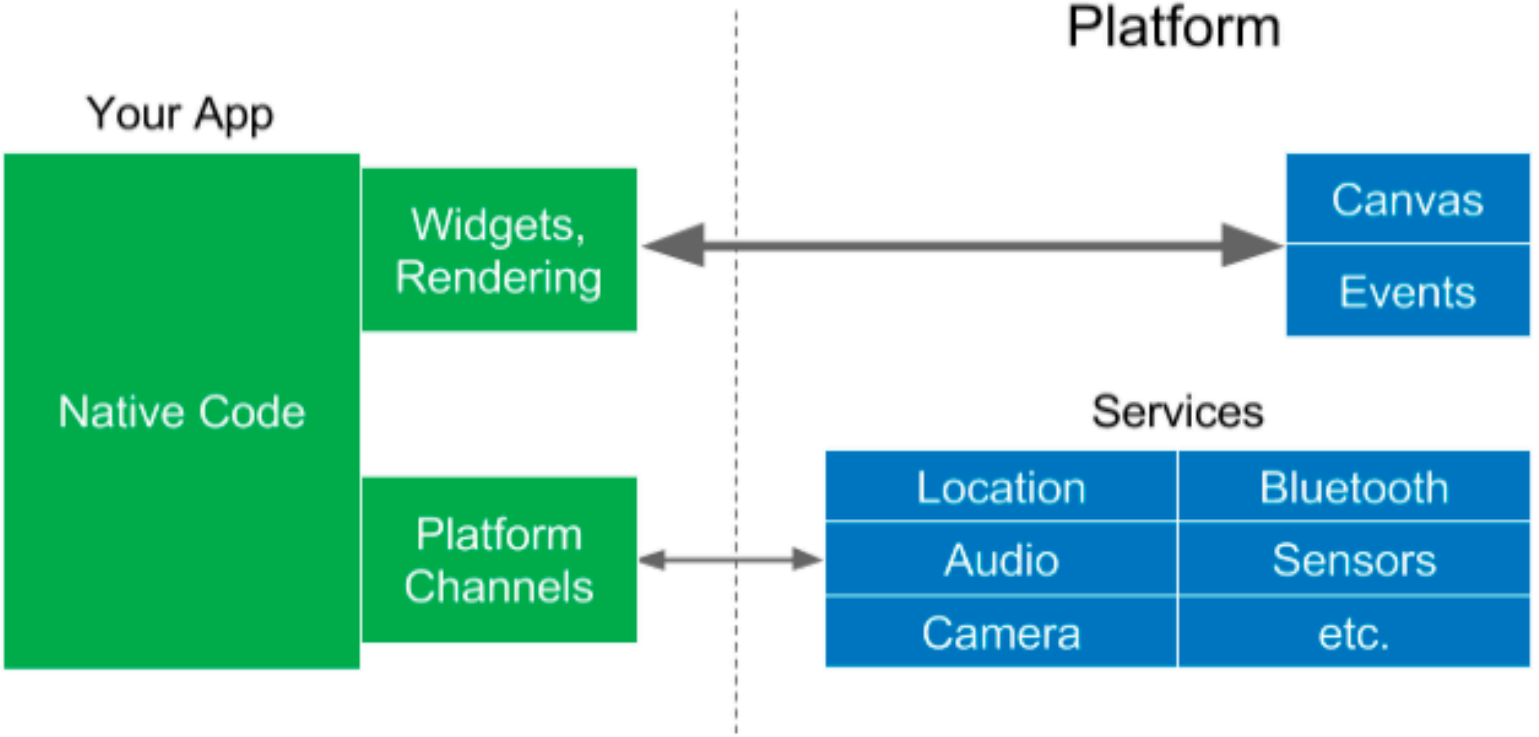
- **React Native** utiliza sim recursos nativos das plataformas, mas o código JS não é convertido para código nativo, e para ter essa comunicação é necessário utilizar uma ponte (bridge).
- A ponte passa a ser um gargalo na comunicação JS e plataforma native.



Arquitetura básica React Native

Flutter

- Como Flutter interage com as componentes nativas (do Android e IoS)
- A renderização é feita no widget, que usa somente o canvas da plataforma
- O que voce desenvolver com Flutter/Dart vai rodar igualzinho em todos os smartphones e versões de plataformas – as 28 do Android e as 12 do IoS!



Arquitetura básica Flutter

Cross-Platform Mobile Dev

- Xamarin (mais antigo)
- React Native (escrito em Java Script)
- Flutter (Google 2017
- Progressive Web Apps (PWA)
- Kotlin Native
- J2ObjC/Doppl
- Ionic2
- Cordova/PhoneGap/Titanium
- Unity

Flutter

Porém, aviso aos navegantes (“word of advice”)

- Flutter está em construção: sua base já está estável, mas sempre aparecem novidades
- Já existem muitos pacotes de terceiros

➔ Podem haver *bugs*.

Por tudo isso, fique sempre atento às novidades.

Flutter

Evolução acelerada

- Google começou a trabalhar no Flutter em 2015,
- Em 2016 foi apresentado pela primeira vez no Dart Developer Summit,
- Em 2017, ganhou maior notoriedade e adesão da comunidade a partir do release
- A partir de 2018 o ocorreu um boom de lançamento de versões Beta e Release preview
- Em 4 de dezembro de 2018 foi lancada a versão 1.0, com novidades:
  - Codemagic (automatizacãod e builds), Flare (animações em vetor, que podem reodar como widget, Flutter multiplataforma para Windows, Mac & Raspberry Pi, Hummingbird (para web apps)

Project Hummingbird: run an app on any system we wish

Etapas gerais da Instalação

Flutter:

- Visite flutter.dev, faça o download o sdk (em arquivo.zip), abra arquivo
- Abra o assistente de instalação, escolha a pasta, por exemplo: <MyUser>/development/flutter
- Adicione a pasta ao `PATH` do .bash_profile

```
export PATH=/Users/<MyUser>/development/flutter/bin:$PATH
```

- (Obs: flutter é uma aplicação de linha de comando. Por exemplo: `flutter create my_first_app`)

Native Platforms:

- Faça o download e instale Xcode para IoS e/ou Android Studio

IDE (Vistual Studio Code)

- Instale VS Code para a sua plataforma (Mac/Linux/Windows)
- Selecione o flutter plugin para VS code
- Reinicie o VS Code

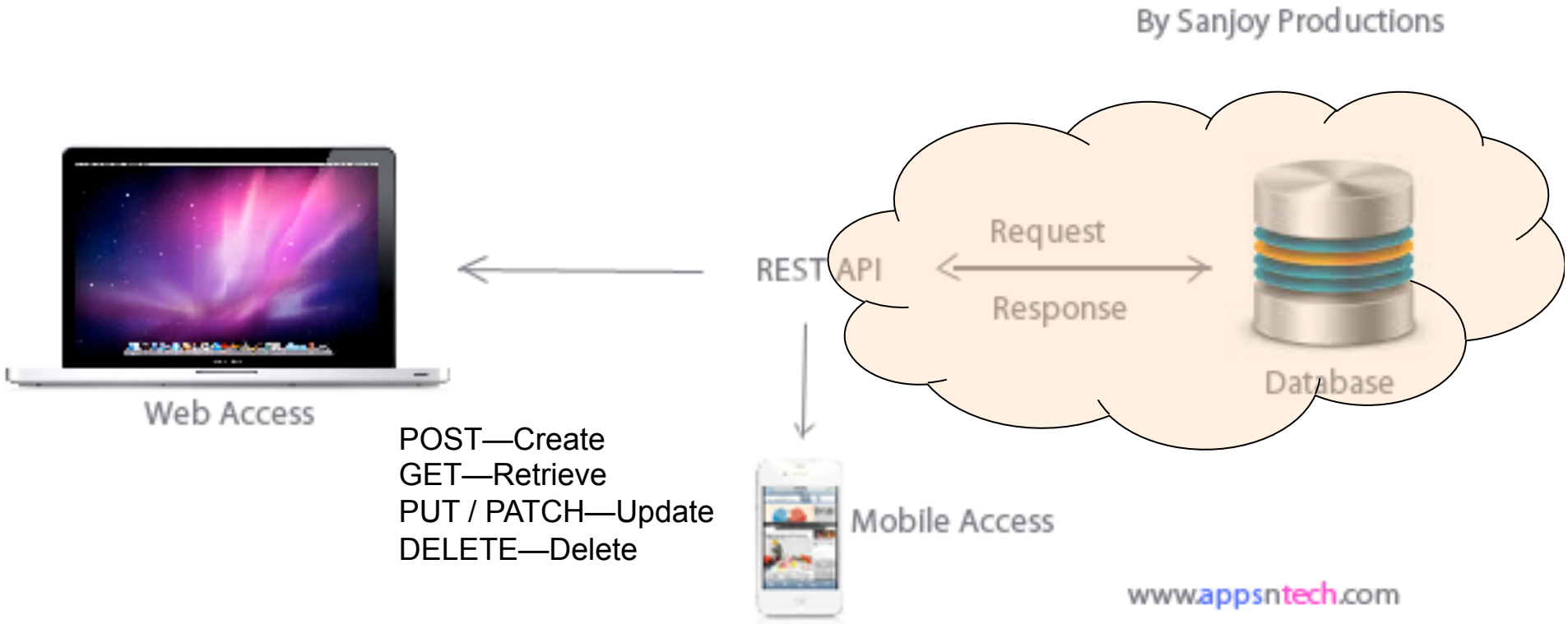
Passo a passo da instalação

Acessar o site:

https://flutter.dev/docs/get-started/install

App móvel *versus* Aplicação mobile-cloud

- O App móvel é somente o terminal de acesso a uma aplicacão distribuida
- A base de dados e o processamento estão em um servidor de aplicação ou em um serviço em cloud.



Aplicações Relevantes (algumas sugestões)

Saúde

- Cuidados pessoais (ingestão de líquidos, medicamentos, exercícios, dieta, etc)
- Desempenho em exerciciso físicos
- Busca por emergênci amais proxima, etc.

Mobilidade

- Segurança no transporte
- Otimizacão no transporte (multimodal)

Meio Ambiente

- Educação ambiental
- Aviso sobre lixo acumulado (lixeira, bueiro, etc.)

Lazer Outdoor

- Informações sobre atrações turisticas/eventos, trilhas,

Aplicações Relevantes (algumas sugestões)

Segurança Pública

- Notificação de ocorrência e chamada de policia (190) Bombeiros (193) Defesa Civil (199)

Agricultura de precisão

- ??

Industria e Logística

- Falta de um insumo/peça gera uma ordem de compra em fornecedor.

Rede Social

- Denúncia/compartilhamento dos problemas de infra-estrutura (luz, vazamento gás, problema de sanemaento básico

Atenção: Não será permitido desenvolver um **game por si só**, mas sua aplicação pode ser usar elementos de gamificacão (para incentivar o seu uso)

Começando a usar Flutter & Dart

Flutter

Inclui:

- Reactive Framework
- 2D rendering Engine
- Várias ferramentas de desenvolvimento
- Ready-made widgets

Principais Conceitos

- Widgets
- States
- Material Design
  - Coleção de elementos para a UI
- User Interaction and gestures
  - Definição e formas de captar os gestos do usuário
- Packages

Widget

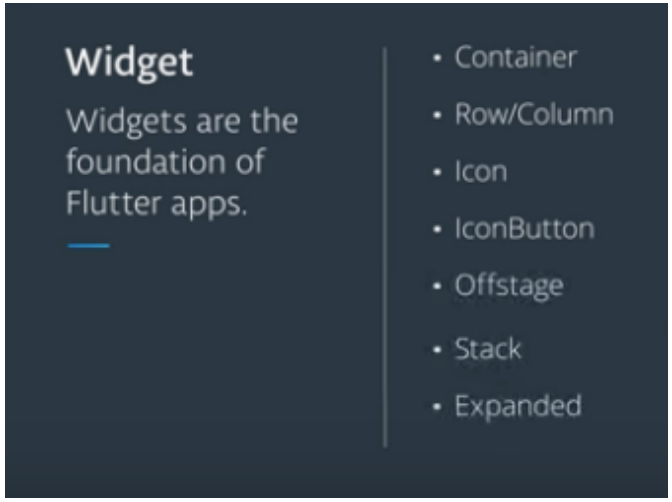
- Is a description of part of a UI
- Flutter has no separate files for layout or customization, all code relating to the UI element is defined in the corresponding widget
- Every widget has a series of attributes and event-based functions
  - textTheme
  - Color
  - onPressed
  - Shape
  - ..
- Widgets have state, which changes as the user interacts with it

Flutter Dev Tools

- Hot reload
  - Changes in the code are reflected instantaneously on the UI (on device or emulator)
- Flutter Inspector
  - Relates and lets you navigate back and forth: pixel-level UI position – widget tree – source code
- Code auto-formatter (`dartfmt`)
  - Formats your code so that it becomes clearer to maintain

Widgets

- A interface do usuário é composta de widgets aninhados ➔ widget tree
- Flutter oferece uma grande gama de widgets



- widgets Stateless ➔ uma vez que uma propriedade é instanciada, ela não pode mais ser modificada. (cor de fundo, altura, largura)
  - Exemplo: Container widget – para particionamento da tela e enmolduramento de outras widgets etc.
- widgets Stateful ➔ são elementos da interface que encorporam uma interação
  - Exemplo: switch

Widgets – 3 Princípios

Todos os widgets devem obedecer a três princípios:

1. Aparência
   - Devem ser visualmente agradáveos, bonitos, preservem o "Look & Feel" de cada plataforma (se desejado). L&F do Android difere daquele do Cuperino / ioS
   - Widgets com estilo *Cupertino* (Apple) e *Material Design* (Google)

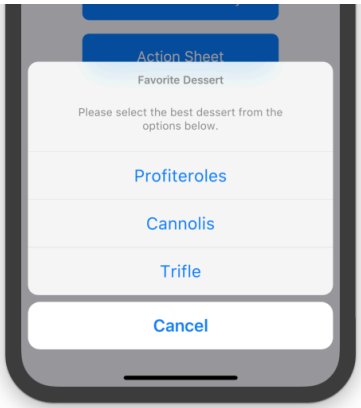
2. Alto desempenho
   - Quando são acionados devem produzir uma reacão rápida e suave, incluindo transições e animações

3. Extensibilidade e Customização
   - Possibilidade de customizar/modificar quase tudo no wirdget para manter o estilo & branding do aplicativo
   - Isso é possivel porque não existe mais tradução/ponte: **os widgets são renderizados pelo próprio aplicativo** e não na plataforma

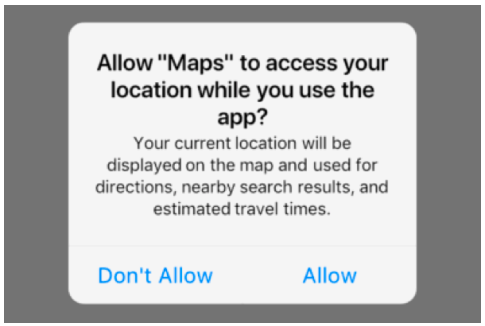
Widgets - Catálogo

Widget Catalog: https://flutter.dev/docs/development/ui/widgets

- Existem widegts para TUDO, desde estrutura/ layout de uma tela, icones, buttons, animações, input, Semântica de widgets, Modelos de Interação, scrolling, display de texto, theme style, etc.
- Cada widget em seu app é uma classe **Dart** que estende uma classe que vem em um package (exemplo: package:flutter/material.dart)
- Exemplos (Cupertino iOS)

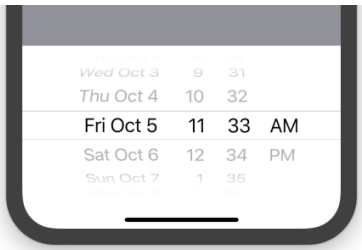


Bottom action sheet



Alert Dialog

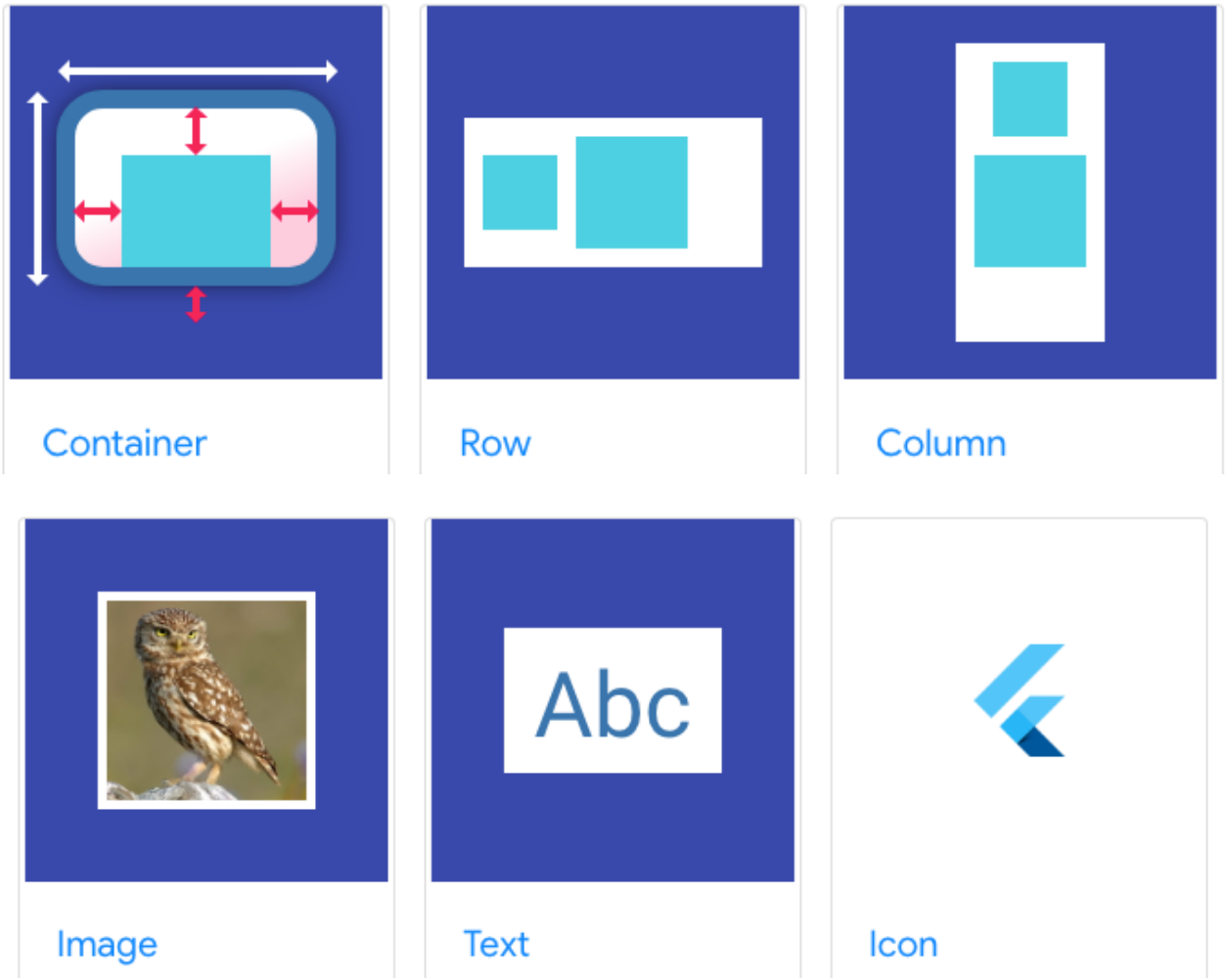
Activity Indicator/s



Date Pickers

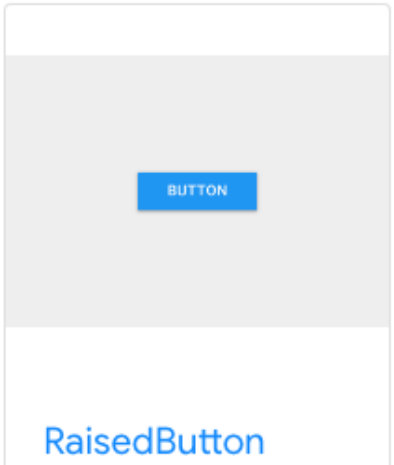
Widgets Essenciais

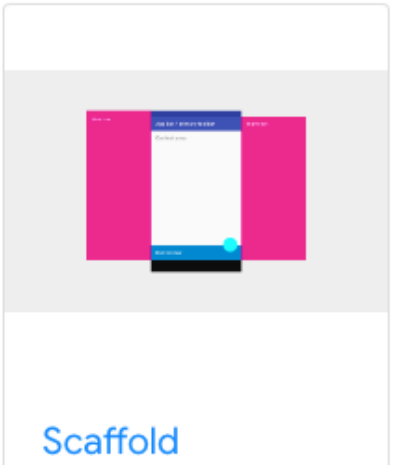
- Necessários para quase toda app (Basic Widgets)

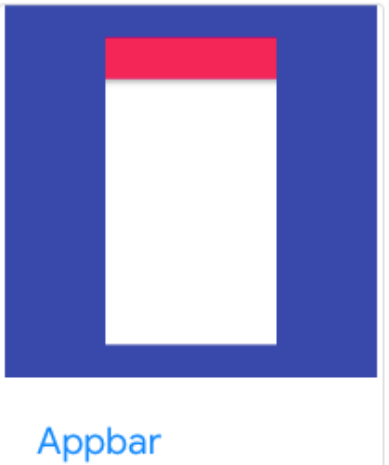


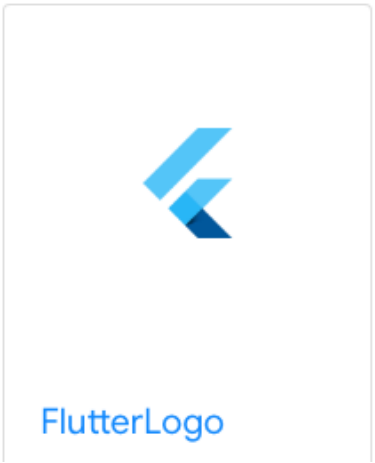
Widgets Essenciais

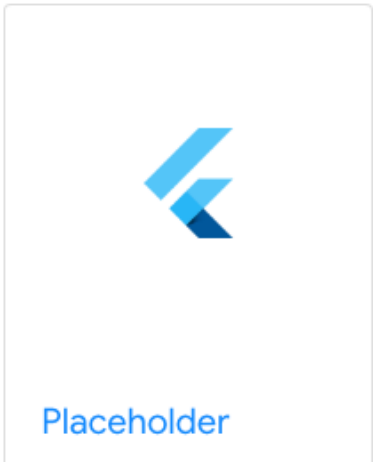
- Necessários para quase toda app (Basic Widgets)







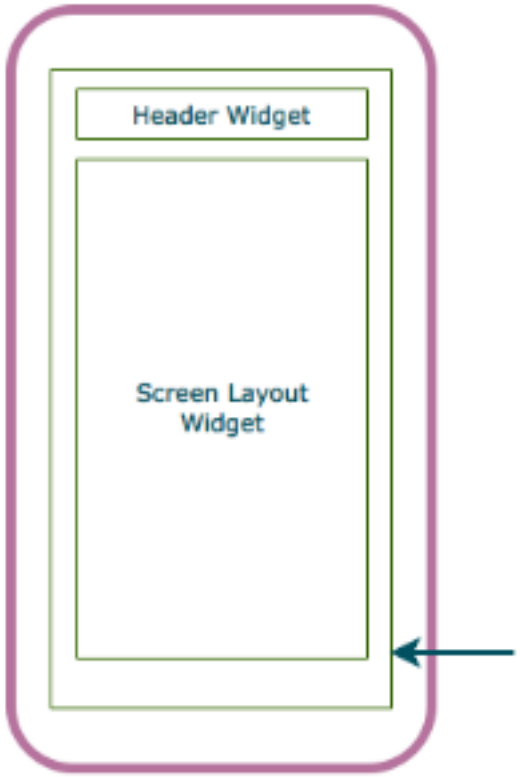




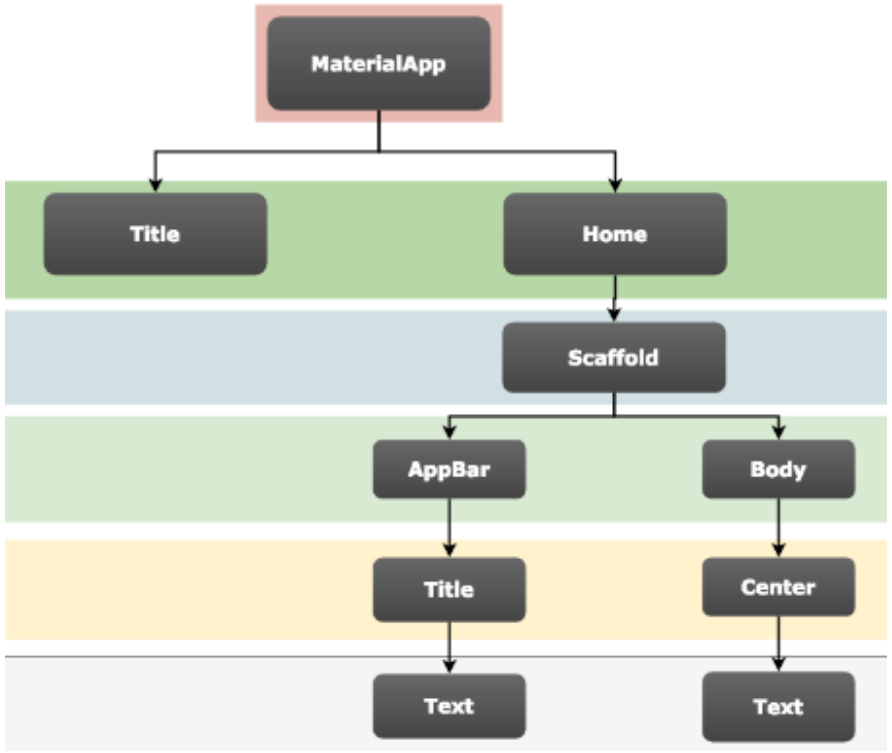
Scaffold: Implementa a estrutura basica de layout do Material Design.
Com APIs para posicionar e mostrar drawers, bottom sheets, snackbars, etc.

Hierarquia de Widgets

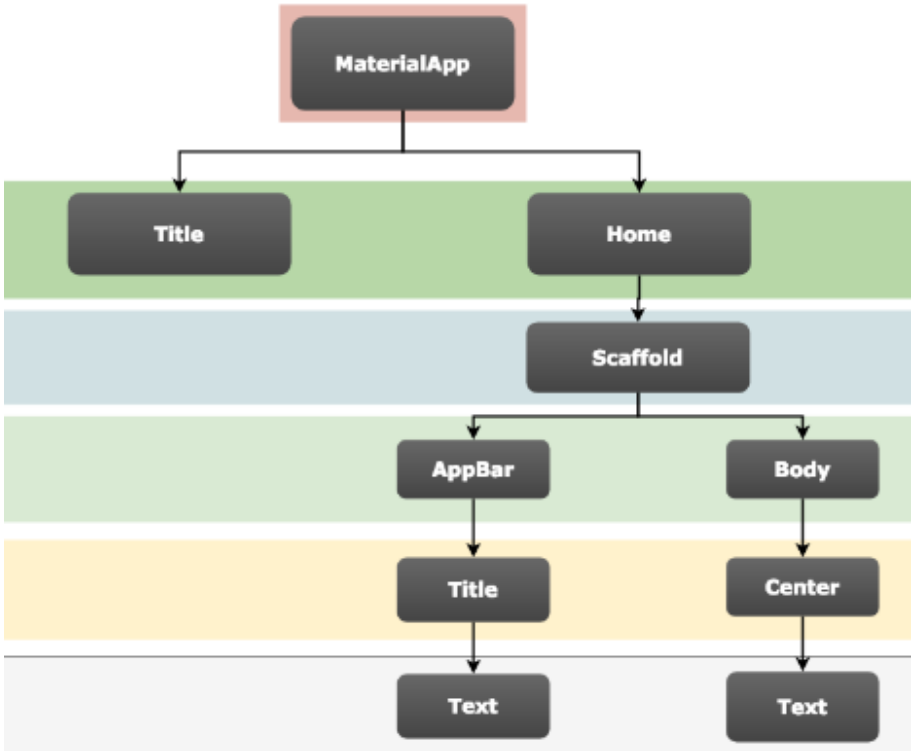
A very basic app



The widget hierarchy



Widgets - Hierarchy




```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext ctxt) {
    return new MaterialApp(
      title: "MySampleApplication",
      home: new Scaffold(
        appBar: new AppBar(
          title: new Text("Hello Flutter App"),
        ),
        body: new Center(
          child: new Text("Hello Flutter"),
        ),
      ),
    );
  }
}
```

Hello Flutter

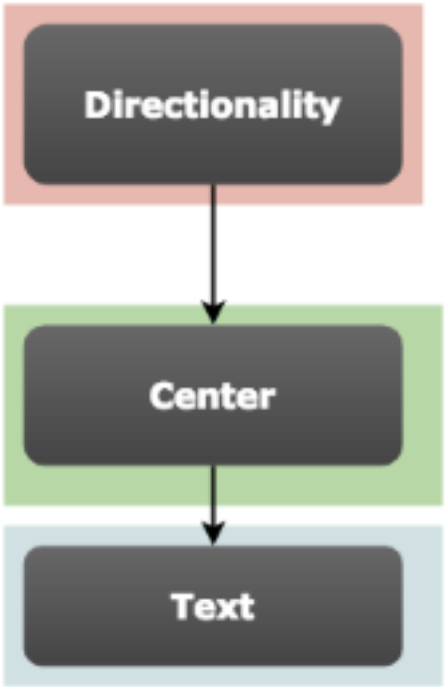
- Usaremos um Container Widget chamado Directionality e um Text Widget
- Precisamos importar o pacote material design (pois é ele que contém os widgets)

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

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext ctxt) {
    return new Directionality(
        textDirection: TextDirection.ltr,
        child: new Text("Hello Flutter")
    );
  }
}
```

- MyApp é uma widget que vai criar o layout da tela.
- Criamos um container Directionality que terá um sub-widget chamado 'Text'
- Cada widget them um método "build" e retorna um widget.

Hello Flutter

- Se quisermos centralizar o texto, criamos um sub-widget filho de Directionality, e pai do widgetText.




```
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext ctxt) {
    return new Directionality(
        textDirection: TextDirection.ltr,
        child: new Center(
                child: new Text("Hello Flutter"),
         )
    );
  }
}
```

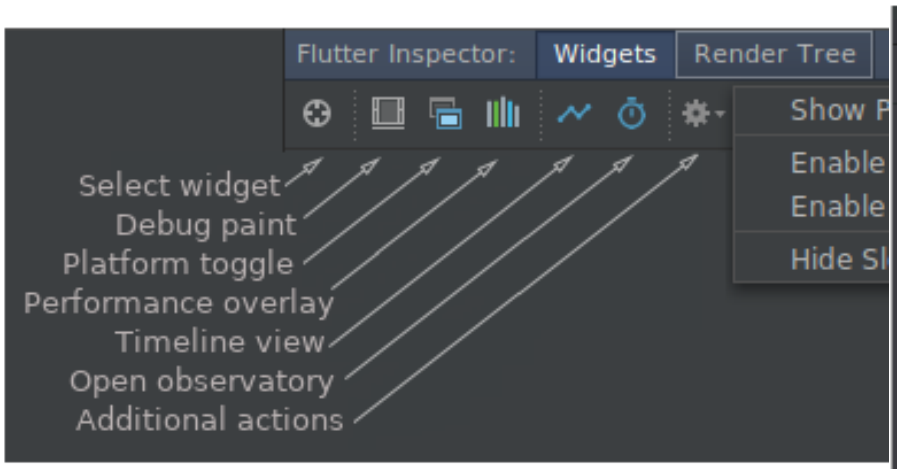
Widget Inspector

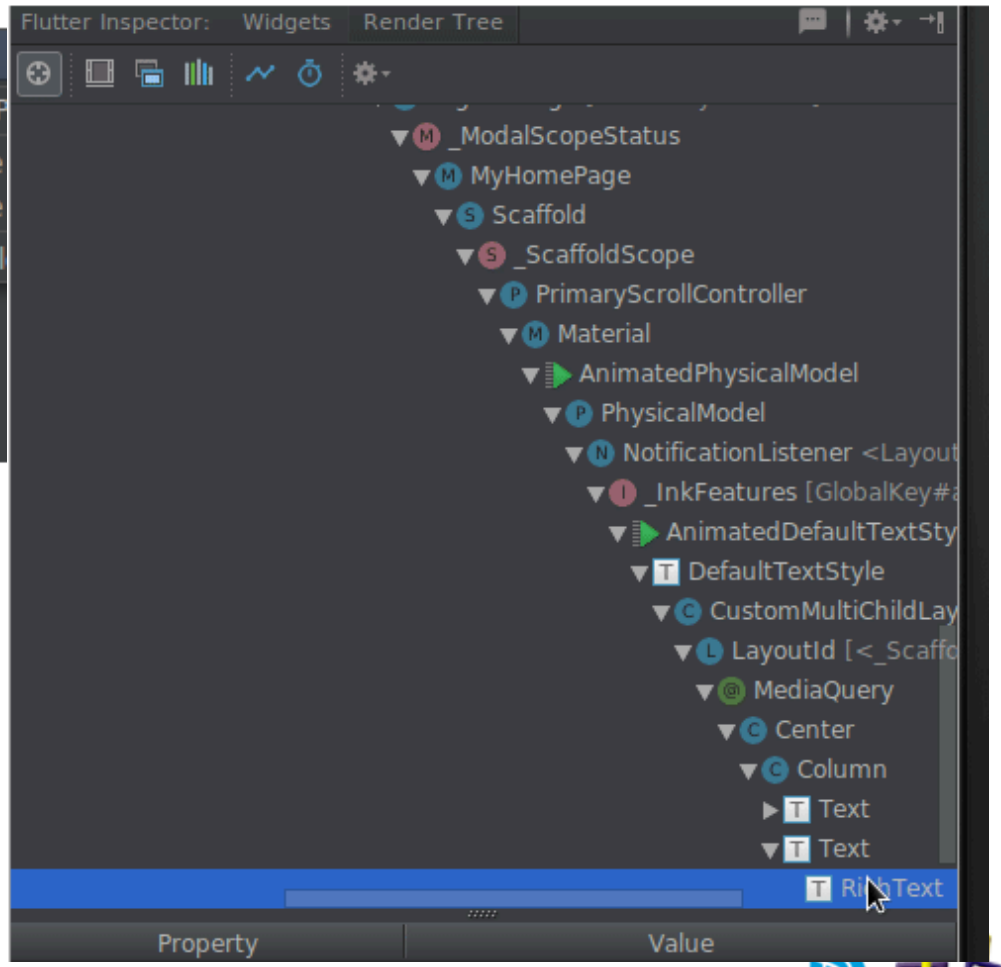
- O Flutter framework usa widgets como o conceito fundamental para tudo desde controles/eventos (text, buttons, toggles, etc.) a layout (centering, padding, rows, columns, etc.).

- **Widget inspector** é uma ferramenta poderosa para visualizar e inspeconar widget trees.
- Pode ser útil para :
  - Entender layouts existentes
  - Decubrindo problemas nos layouts

- O **Widget inspector** está disponivel no plugin Flutter do Android Studio ou a IDEA IntelliJ.

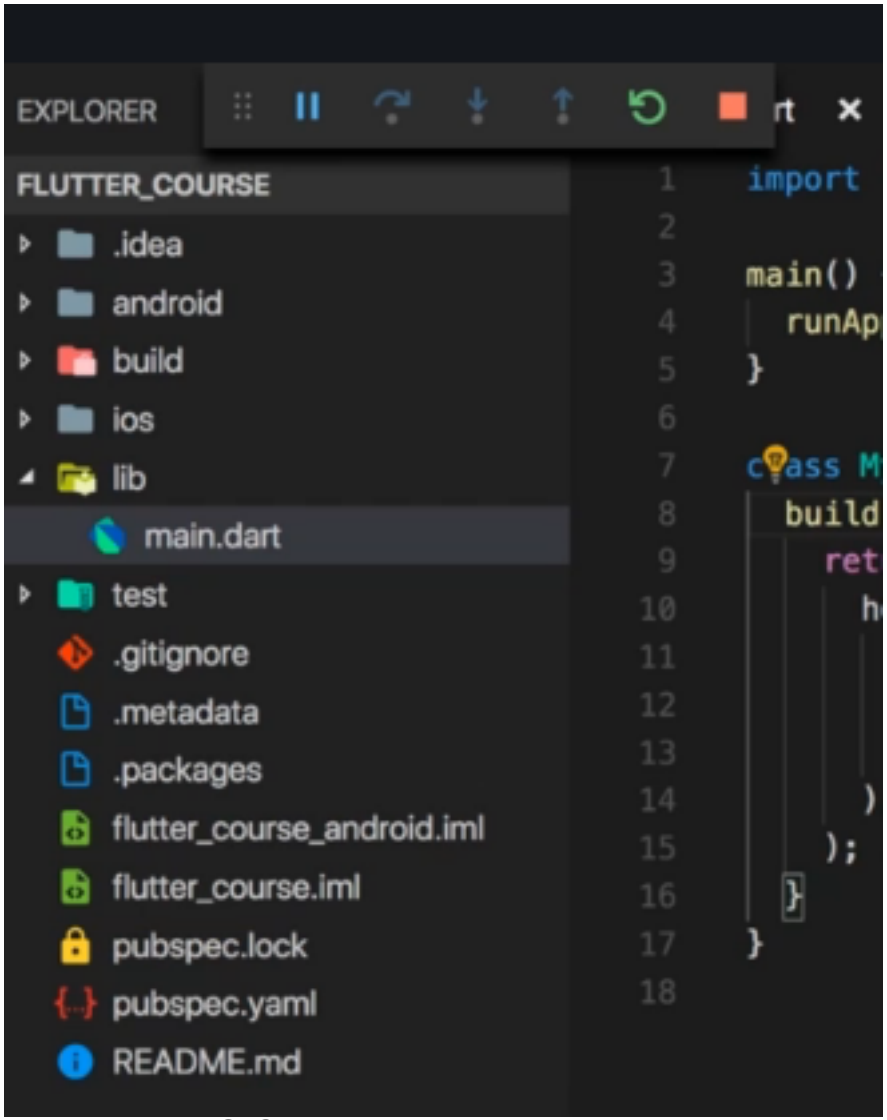
Widget Inspector





https://flutter.io/docs/development/tools/inspector

Um projeto de App Flutter



No VS Code com plugin Flutter

- **lib/main.dart** é onde é colocado o seu código dart
- **.idea**: usados pelo VS Code ou outras IDEs (não alterar)
- **android** e **ios**: código nativo gerado a partir do dart (não alterar)
- **build**: arquivos gerados no processo de build multi-plataforma (não alterar)
- **test**: pode ser usado para criar testes automatizados
  - .gitignore: para controle de versão no git
  - .metadata, .packages, etc.: arquivos de configuração
  - pubspec.yaml: descrição das dependências de módulos de código nativos

Thanks!

More Information
www.lac.inf.puc-rio.br

Partners: