Desenvolvimento de aplicativos com o framework Flutter

marco.mangan@pucrs.br

Visão geral

Tipos de Interfaces de Usuário

- Command Line Interface
- Graphical User Interface

- Desktop (nativo)
- **Web** (JavaScript)
- IoS (serviços: REST)
- **Mobile** (nativo: Android)
- IoT (nativo: Arduino)

- Natural
  - assistente virtual, reconhecimento e síntese de voz
  - Siri, Alexa, Google Assistant)

Dispositivos

- Smartphone
- Tablet
- *Wearable*

- Televisão
- *Desktop*

- Unidade Veicular
- Unidade de Resposta Auditiva

Cross-platform

- Qt
- Swift UI (Apple)
- Java FX (Oracle)

- Flutter (Google)
- Cordova, Xamarim, PhoneGAP, Genexus...

- Angular (Google)
- React (Facebook)
- JQuery

Linguagens de Programação

- C
- C++
- Swift (Apple)
- Java (Oracle)
- JavaScript
- C# (Microsoft)
- Dart (Google)

- HTML
- CSS

- Smalltalk, Ruby, Python, PHP...

CheatSheets

- **Dart**
- https://koenig-media.raywenderlich.com/uploads/2019/08/RW-Dart-Cheatsheet-1.0.2.pdf

- **Flutter**
- https://gist.github.com/matteocrippa/3a8b84c7b49c10bc070e58a66860e83f
- https://medium.com/flutter-community/flutter-layout-cheat-sheet-5363348d037e

Flutter

- Framework da Google para Interfaces de Usuário
    - portável
    - nativo
    - reativo
- iOS, **Android**, Desktop, Web, IoT...
    - **Material** e Cupertino
- Código aberto, Dart

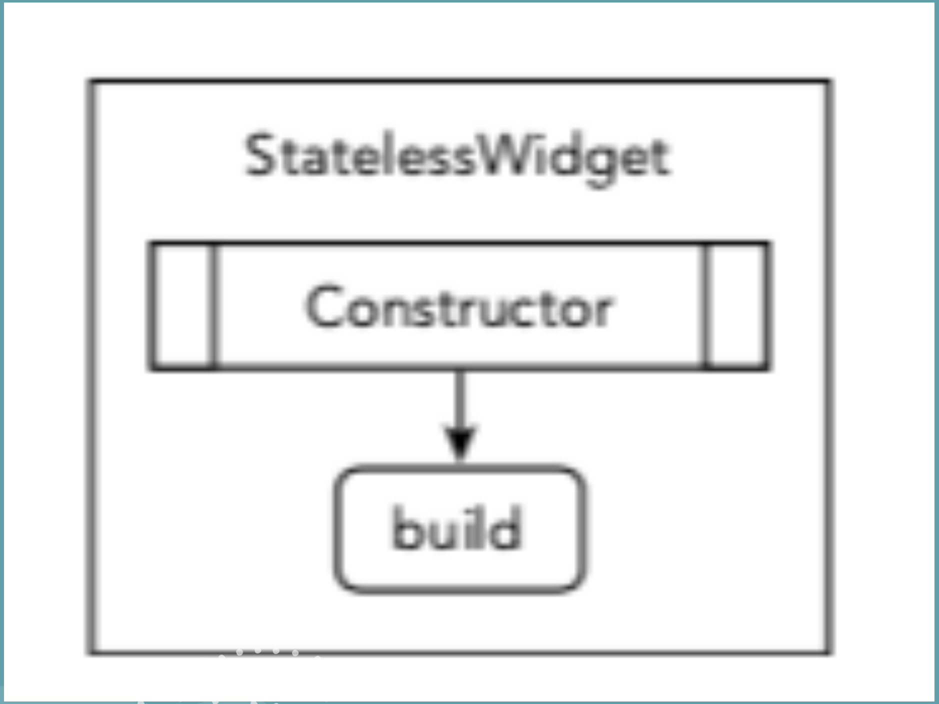
Widget vs Element

- Widget: instruções declarativas sobre a Interface de Usuário
- Element: desenho do Widget no dispositivo
- Exemplos de widgets:
  - list, grid, text, buttons
  - form, form fields, listeners
  - fonts, colors, borders, shadows
  - row, column, stack, centering, padding
  - touch, gestures, dragging, dismiss
  - animations, transitions, scale
  - assets, images, icons

Ciclos de vida

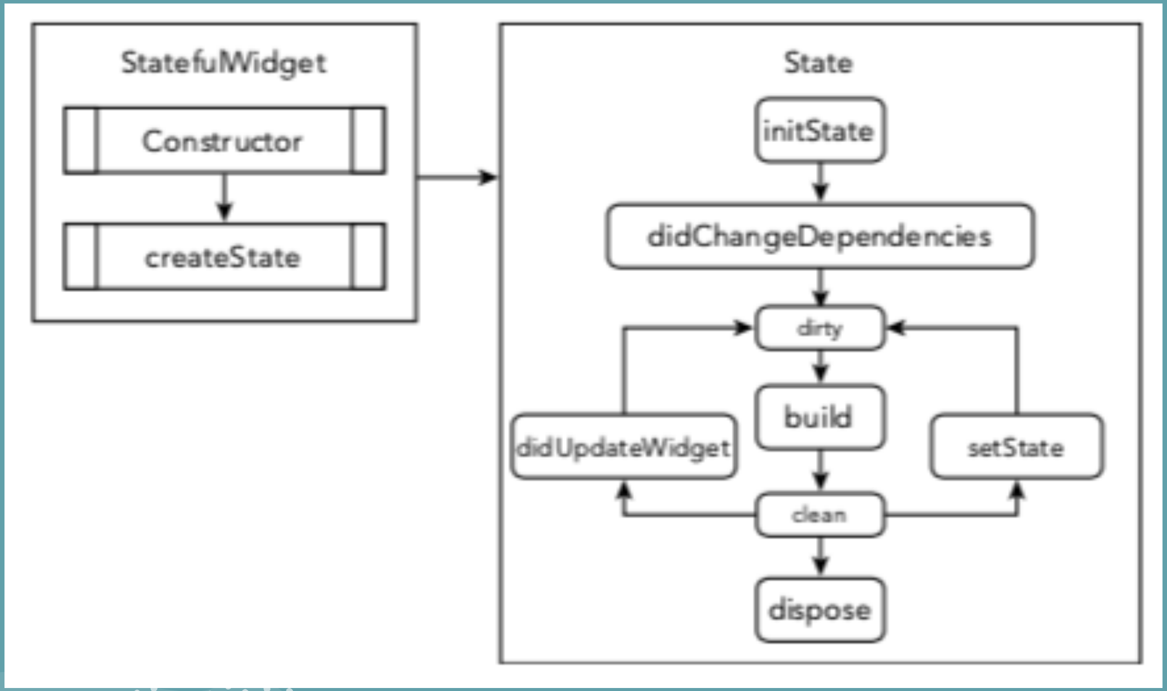
StatelessWidget vs StatefulWidget

Stateless Widget



```
class JournalList extends  StatelessWidget {
@override
Widget build(BuildContext context) {
  return Container(); }
}
```

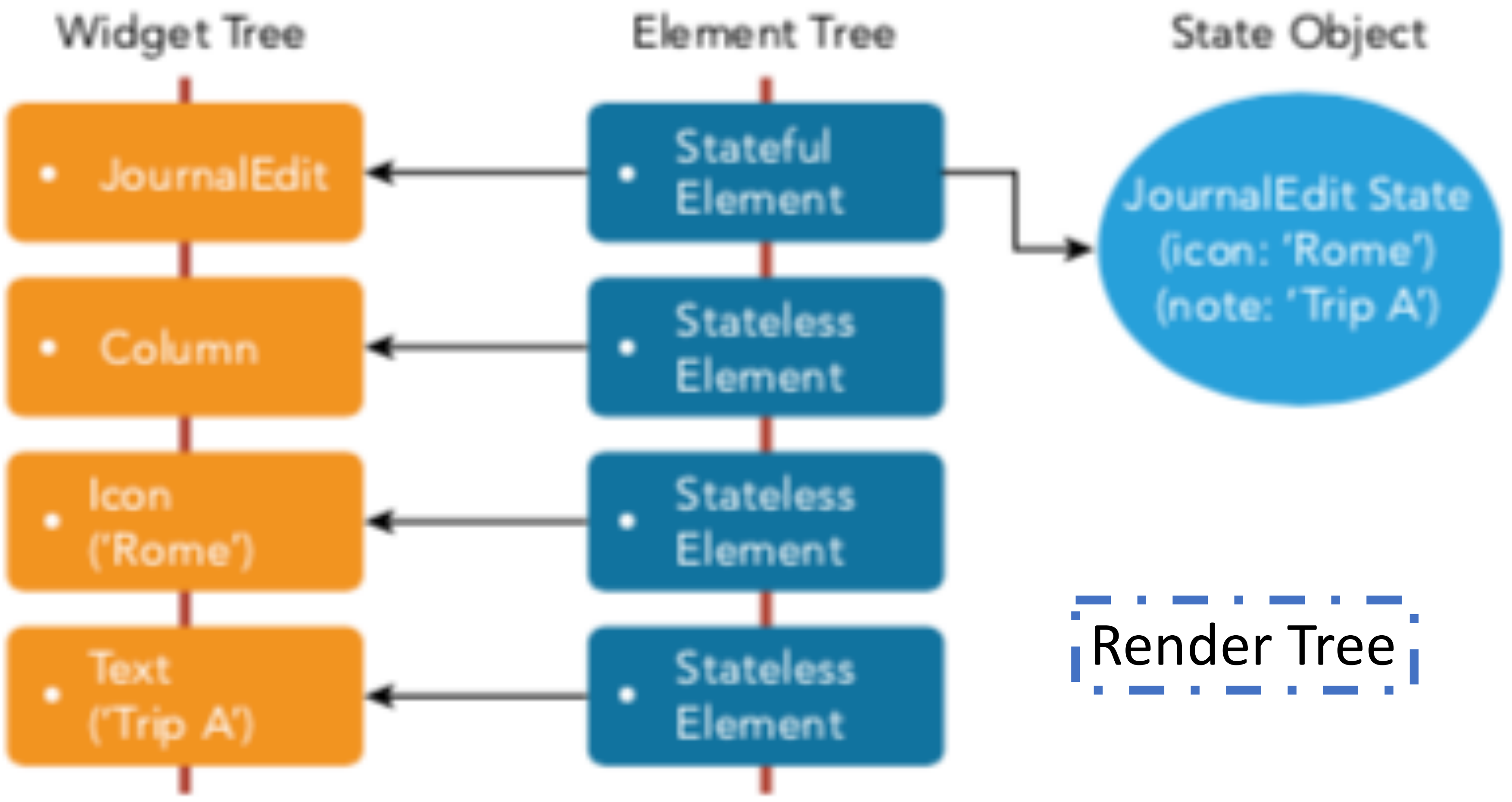
StatefulWidget



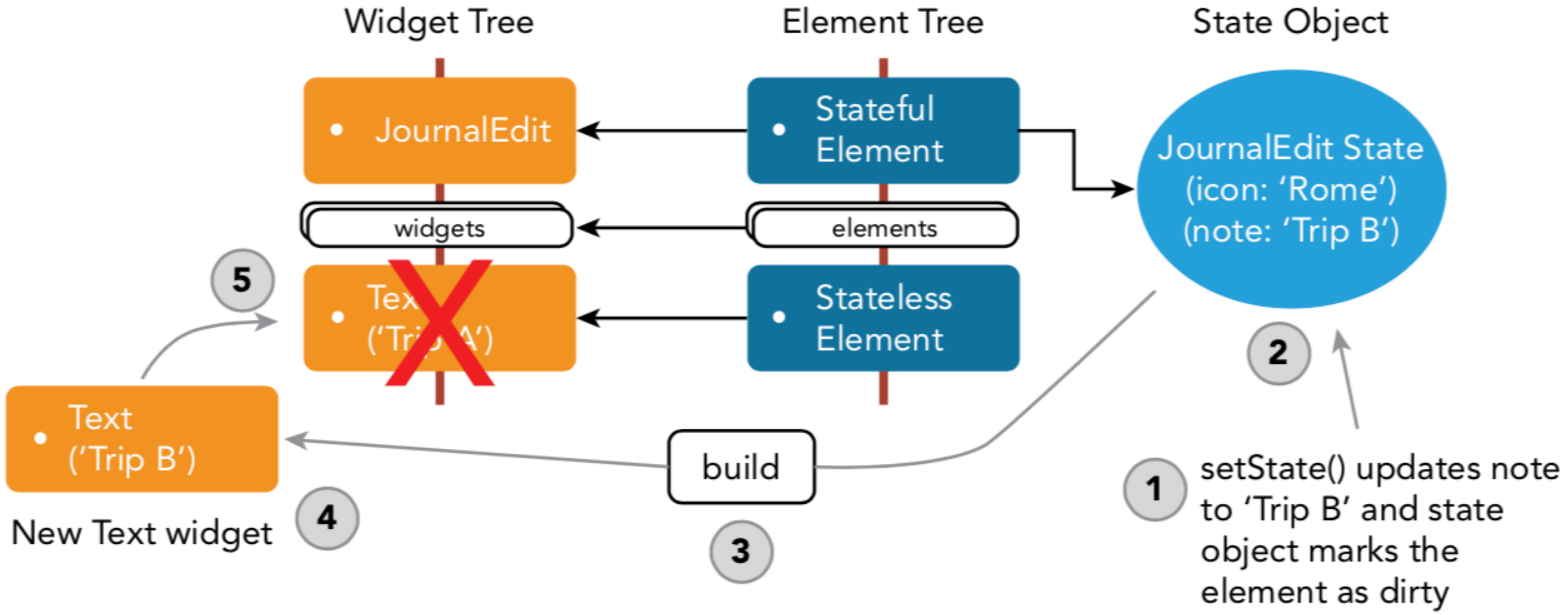
```
class JournalEdit extends StatefulWidget {
@override
_ JournalEditState createState() => _
JournalEditState();
}
class _JournalEditState extends State<JournalEdit> {
@override
Widget build(BuildContext context) {
return Container(); }
}
```

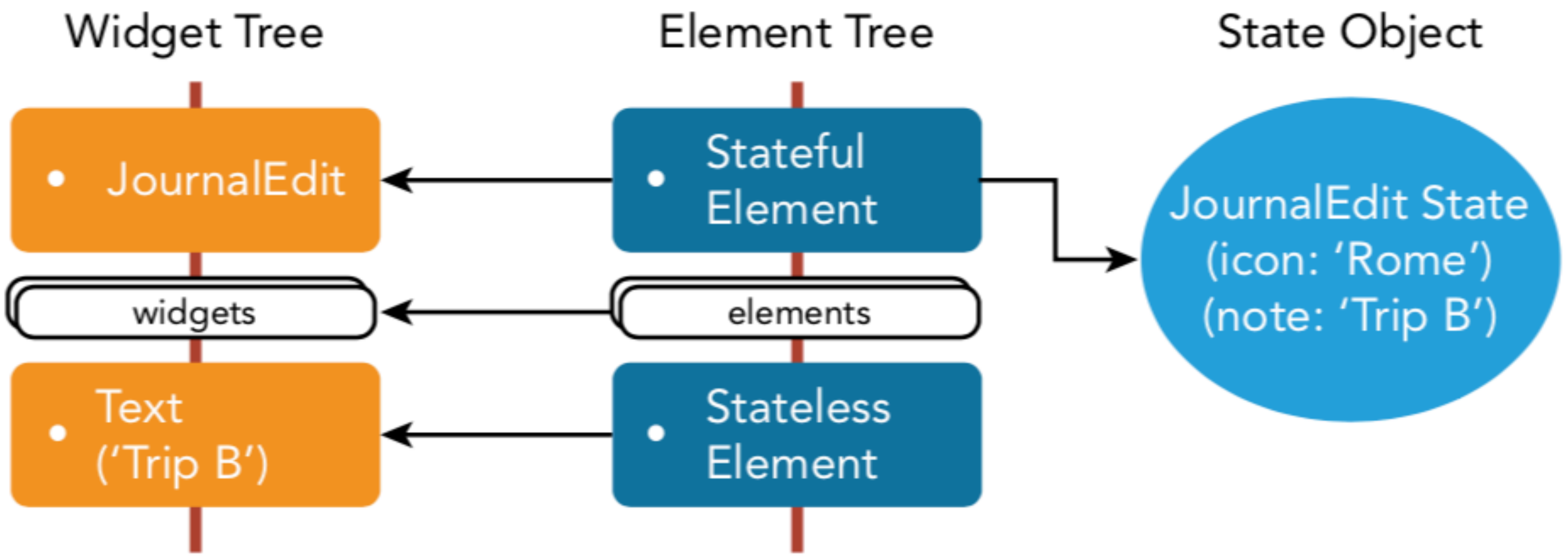
State Object and Three Trees: Widget, Element, and Render Trees

StatelessWidget vs StatefulWidget


Widget Tree
Element Tree
State Object
JournalEdit
Column
Icon ('Rome')
Text ('Trip A')
Stateful Element
Stateless Element
Stateless Element
Stateless Element
JournalEdit State (icon: 'Rome') (note: 'Trip A')

Render Tree

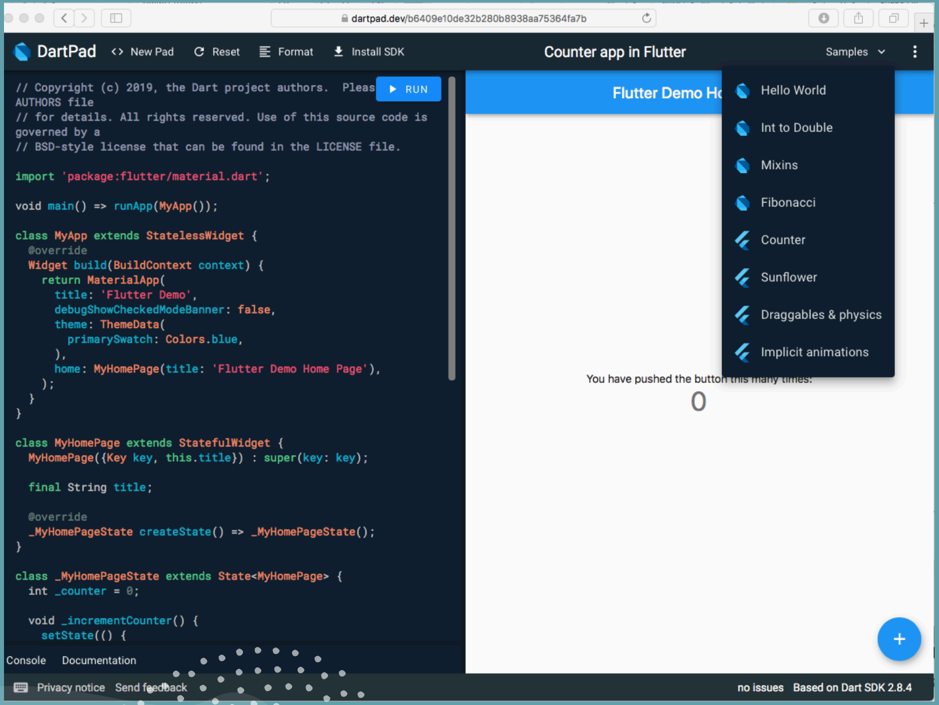

Widget Tree
Element Tree
State Object
JournalEdit
Stateful Element
JournalEdit State (icon: 'Rome') (note: 'Trip B')
widgets
elements
Text ('Trip A')
Stateless Element
5
2
Text ('Trip B')
build
New Text widget
4
3
1
setState() updates note to 'Trip B' and state object marks the element as dirty


Widget Tree
Element Tree
State Object
JournalEdit
Stateful Element
widgets
elements
Text ('Trip B')
Stateless Element
JournalEdit State (icon: 'Rome') (note: 'Trip B')

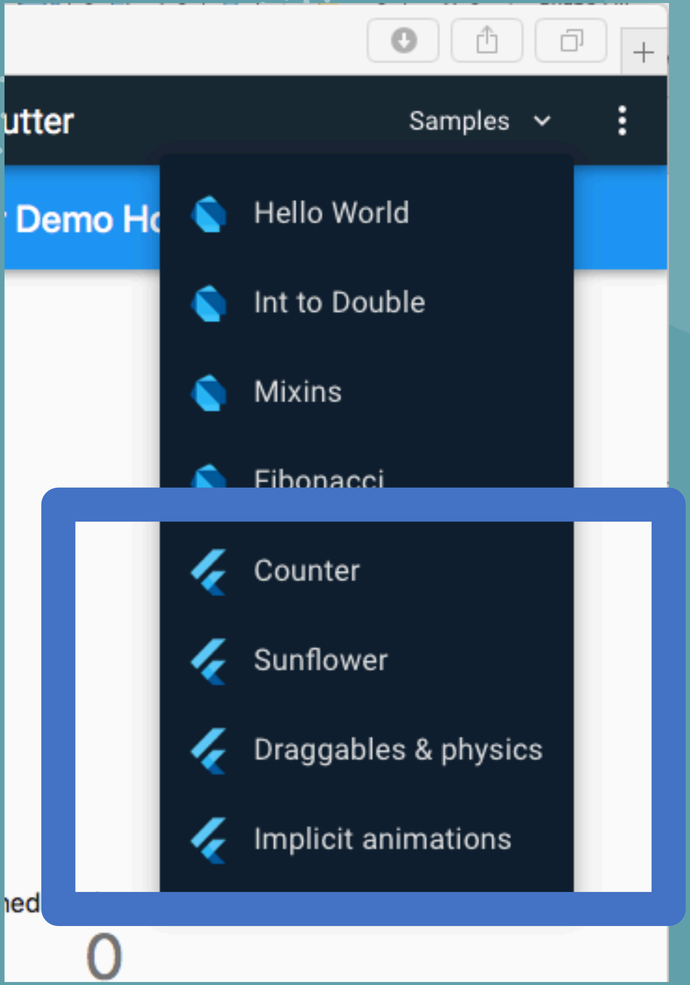
Ferramentas

Nuvem vs local

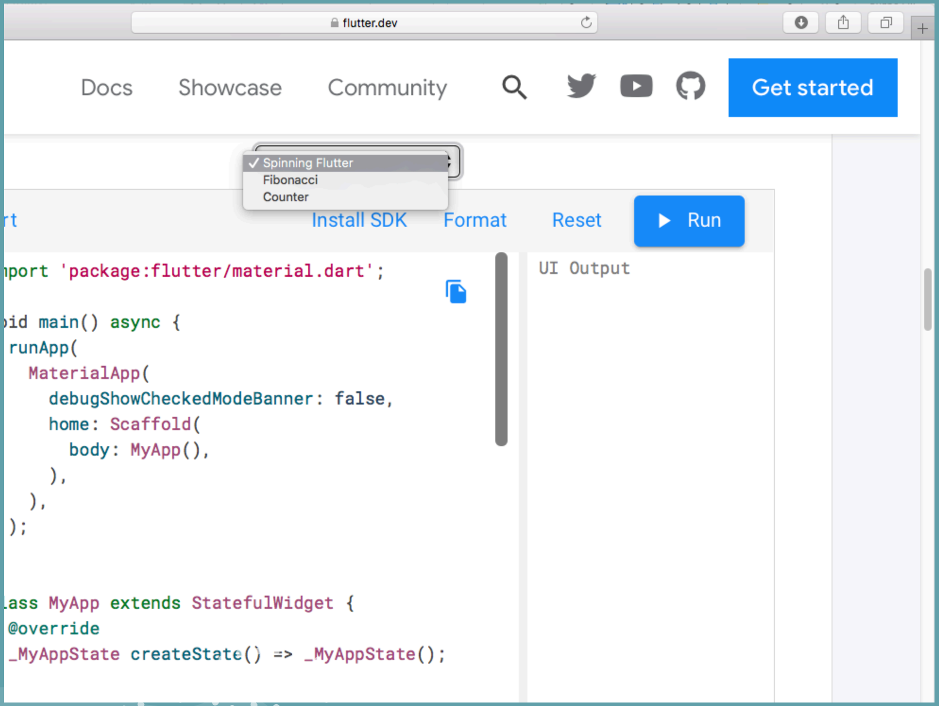
Dartpad



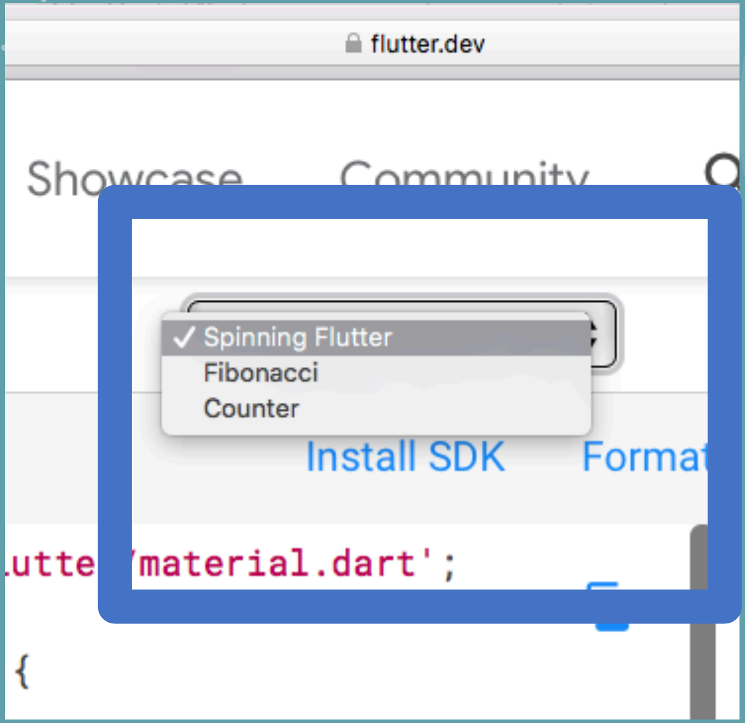
```dart
// Copyright (c) 2019, the Dart project authors.  Pleas
AUTHORS file
// for details. All rights reserved. Use of this source code is
governed by a
// BSD-style license that can be found in the LICENSE file.

import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      debugShowCheckedModeBanner: false,
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: MyHomePage(title: 'Flutter Demo Home Page'),
    );
  }
}

class MyHomePage extends StatefulWidget {
  MyHomePage({Key key, this.title}) : super(key: key);

  final String title;

  @override
  _MyHomePageState createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  int _counter = 0;

  void _incrementCounter() {
    setState(() {
```



Flutter

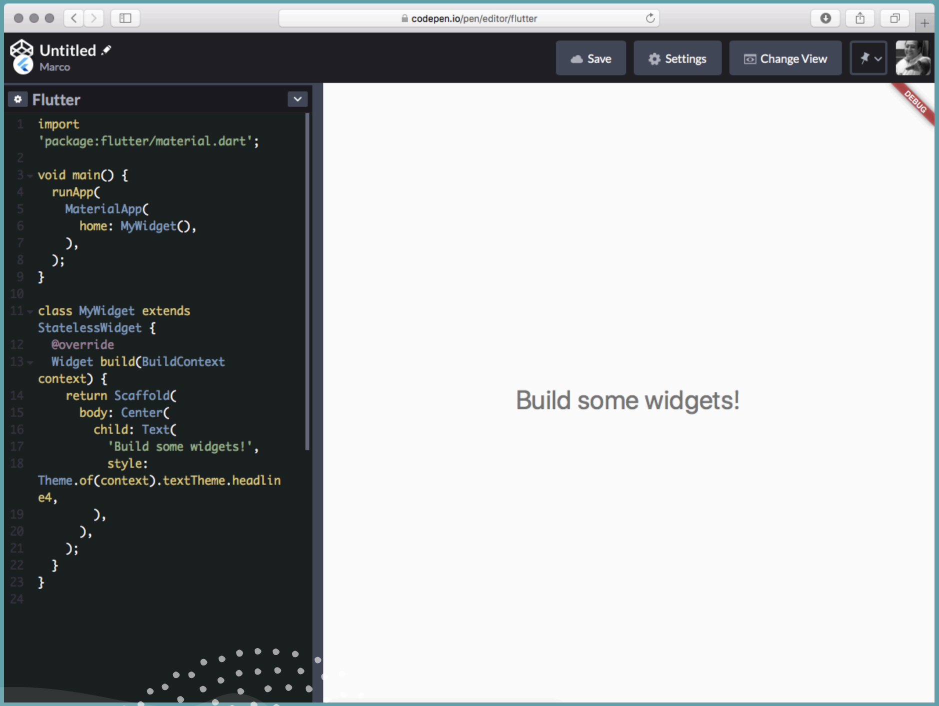


```
nport 'package:flutter/material.dart';

oid main() async {
 runApp(
   MaterialApp(
     debugShowCheckedModeBanner: false,
     home: Scaffold(
       body: MyApp(),
     ),
   ),
 );

lass MyApp extends StatefulWidget {
 @override
 _MyAppState createState() => _MyAppState();
```



```
utte material.dart';
{
```

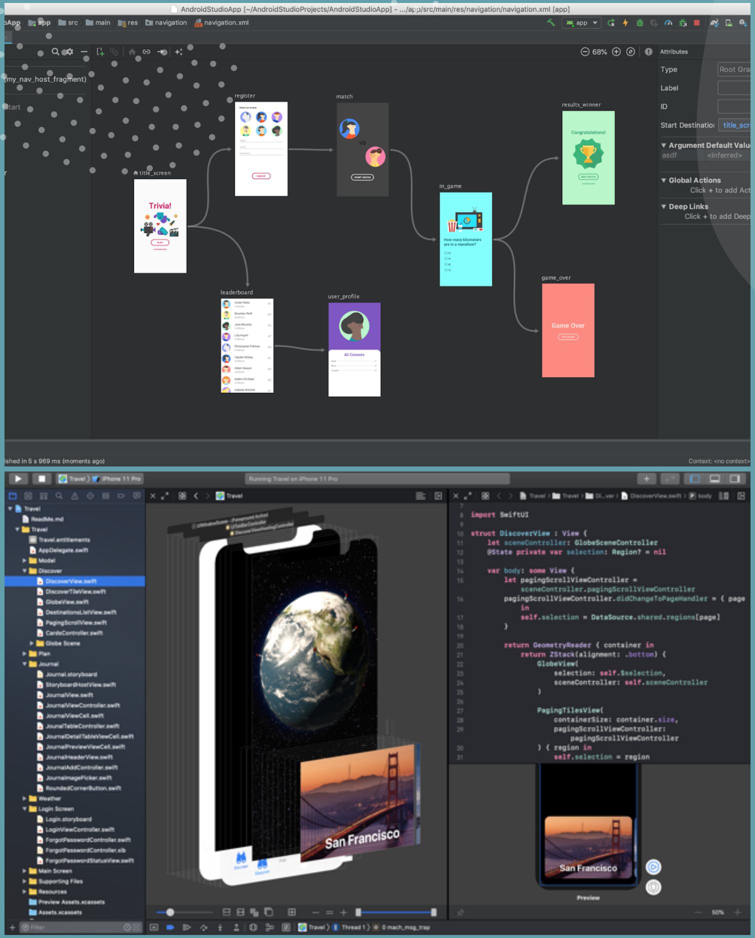
Codepen.io



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

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

class MyWidget extends
StatelessWidget {
  @override
  Widget build(BuildContext
context) {
    return Scaffold(
      body: Center(
        child: Text(
          'Build some widgets!',
          style:
Theme.of(context).textTheme.headlin
e4,
        ),
      ),
    );
  }
}
```

Build some widgets!

Android Studio, VSCode, XCode, Visual Studio



```swift
import SwiftUI

struct DiscoverView : View {
    let sceneController: GlobeSceneController
    @State private var selection: Region? = nil

    var body: some View {
        let pagingScrollViewController =
            sceneController.pagingScrollViewController
        pagingScrollViewController.didChangeToPageHandler = { page
            in
            self.selection = DataSource.shared.regions[page]
        }

        return GeometryReader { container in
            return ZStack(alignment: .bottom) {
                GlobeView(
                    selection: self.$selection,
                    sceneController: self.sceneController
                )

                PagingTilesView(
                    containerSize: container.size,
                    pagingScrollViewController:
                        pagingScrollViewController
                ) { region in
                    self.selection = region
```

Command-line tools

- mkdir, cd
- git
- curl
- clear
- vi
- ls
- chmod
- export
- diff
- history
- cat
- ps, kill
- pwd
- ping, traceroute
- unzip, tar

Em resumo

Flutter

Algumas classes do Flutter até o momento

1. AppBar
2. BuildContext
3. Center
4. Colors
5. Column
6. Container
7. Drawer
8. FloatingActionButton
9. Icon
10. Icons
11. Key
12. ListTile
13. ListView
14. MainAxisAlignment
15. MaterialApp
16. Scaffold
17. State
18. StatefulWidget
19. StatelessWidget
20. Text
21. TextStyle
22. Theme
23. ThemeData
24. Widget

Algumas classes do *Flutter* até o momento

1. Aplicação
   - AppBar, Drawer, FloatingActionButton, MaterialApp
2. Componentes
   - BuildContext, Key, State, StatefulWidget, StatelessWidget, Widget
3. Grupos de componentes
   - Column, Container, Row, Scaffold
4. Informação
   - Text, ListView, ListTile
6. Alinhamento
   - Center, MainAxisAlignment
7. Estéticos
   - Colors, TextStyle, Icon, Icons, Theme, ThemeData

Atividades

Exemplo A e desafios

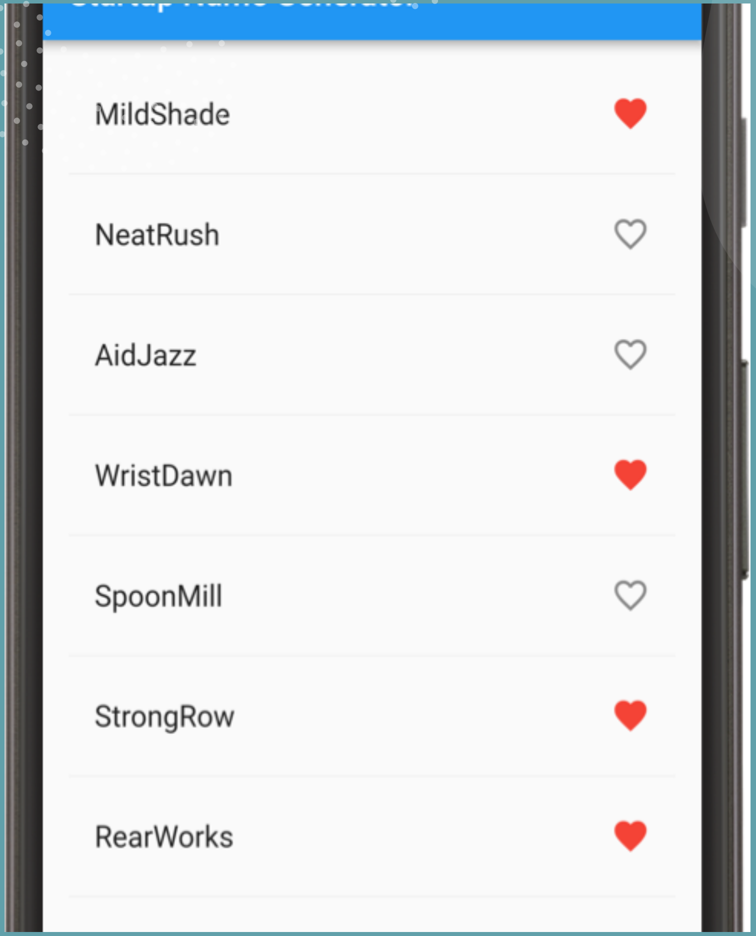
- Crie um Flutter project no Android Studio
- Execute o projeto em um emulador Android ou iOS

- Altere o projeto para compor as telas do desafio.
    - Veja enunciado no Slack
    - Veja respostas APÓS tentar resolver

Write your first Flutter app

https://codelabs.developers.google.com/codelabs/first-flutter-app-pt1/#2

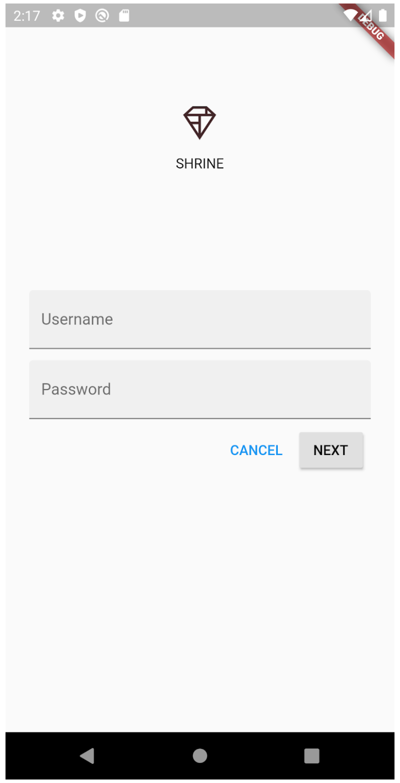
https://codelabs.developers.google.com/codelabs/first-flutter-app-pt2/#0



MildShade ♥
NeatRush ♡
AidJazz ♡
WristDawn ♥
SpoonMill ♡
StrongRow ♥
RearWorks ♥

Shrine

- https://codelabs.developers.google.com/codelabs/mdc-101-flutter/#0



Cookbooks

- Lists
  - Create a grid list
  - Create a horizontal list
  - Create lists with different types of items
  - Place a floating app bar above a list
  - Use lists
  - Work with long lists
- Navigation
  - Navigate to a new screen and back
  - Navigate with named routes
  - Pass arguments to a named route
  - Return data from a screen
  - Send data to a new screen
- Networking
  - Delete data on the internet
  - Fetch data from the internet
  - Make authenticated requests
  - Parse JSON in the background
  - Send data to the internet
  - Update data over the internet
  - Work with WebSockets
- Unit Testing
  - An introduction to unit testing
  - Mock dependencies using Mockito
- Forms
  - Build a form with validation
  - Create and style a text field
  - Focus and text fields
  - Handle changes to a text field
  - Retrieve the value of a text field

Referências

- Napoli (2020) Beginnig Flutter
- flutter.dev
- dart.dev

- http://repl.it
- https://dartpad.dev