FLUTTER

Approfondimento del framework cross-platform

Storia

La prima versione fu “Sky” presentata nel 2015

Flutter 1.0 fu rilasciato il 4 dicembre 2018

INTRODUZIONE

Flutter è un SDK per dispositivi mobili, creato da Google, per lo sviluppo di applicazioni native per iOS e Android da una singola codebase

Utilizza l'approccio **CROSS-COMPILED**

Le applicazioni sono scritte in **Dart**

PIATTAFORME SUPPORTATE

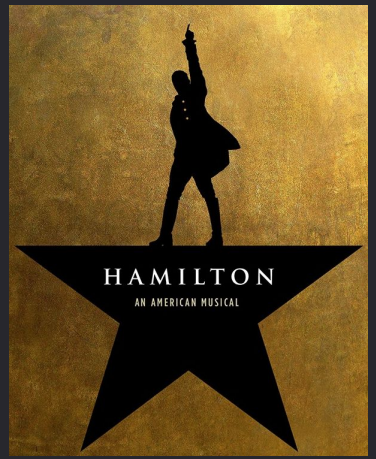
SHOWCASE

Google Ads

Alibaba.com™

APPTREE

Greentea



reflectly
MINDFULNESS JOURNAL

Abbey Road Studios

CARATTERISTICHE

- Rapido sviluppo
- UI espressiva e flessibile
- Performance native

RAPIDO SVILUPPO

- **Hot reload:** permette di ricaricare il codice mentre l'app è in esecuzione
  - Stateful
- Widgets pre-confezionati

UI ESPRESSIVA E FLESSIBILE

- Esperienze utente personalizzate grazie alla vastità di widget in stile **material design** e **Cupertino**

PERFORMANCE NATIVE

- App native
- I widget incorporano tutte le criticità tra le diverse piattaforme (e.g. scrolling, icone, fonts)

PRO e CONTRO

- Free e open-source
- Singola codebase
- Facile setup
- Hot reload
- Widgets
- Performance native
- Plugins per IDE
- Documentazione

- Disponibile solo per mobile
- Scarsità di librerie
- Difficile creare animazioni
- Conoscenza di Dart

PRINCIPI GUIDA DI FLUTTER

- Controllo
- Performance
- Fedeltà

ACCESSIBILITÀ

Componenti di supporto all'accessibilità:

- Font grandi
- Screen reader
- Contrasto sufficiente

COMMUNITY

- Github
- Stack Overflow
- Google groups
- Youtube
- Slack
- Twitter
- Medium
- Meetup

Sul sito ufficiale sono presenti:

- Cookbook
- Codelabs
- Tutorial

DART

IL LINGUAGGIO DART

È un linguaggio di programmazione, orientato agli oggetti, usato per costruire applicazioni web, server, desktop e mobile, sviluppato da Google (inizialmente il suo nome era Dash)

DART - TIPI SUPPORTATI

- Numeri (int o double, sottotipi di num)
- Stringhe (String)
- Booleani (bool)
- enum
- List
- Sets
- Maps
- Runes (per esprimere i caratteri Unicode in una stringa)
- Symbols
- dynamic
- Generics (es: List<tipo> o List<dynamic>)

VARIABILI

Ogni variabile in Dart si riferisce ad un oggetto e memorizza un riferimento

```
var name = 'Bob';
```

```
String name = 'Bob';
```

Le variabili non inizializzate hanno null come valore di default

```
int lineCount;
```

Gli identificatori possono iniziare con lettere o _ e il nome può contenere entrambi e le cifre

COSTANTI

E’ possibile usare final o const per dichiarare costanti

```
final name = 'Bob'; // tipo inferito dal compilatore
final String nickname = 'Bobby';
```

Le variabili di istanza possono essere solo final

La keyword const si può utilizzare anche per i valori

```
final bar = const [];
const baz = []; // equivalente a `const []`
```

LIBRERIE E VISIBILITÀ - 1

Ogni Dart app è una libreria

E' possibile utilizzare librerie per dare modularità al codice

```
import 'dart:html';
```

Possibilità di lazy loading delle librerie

```
import 'package:greetings/hello.dart' deferred as hello;
```

LIBRERIE E VISIBILITÀ - 2

Keyword show e hide:

```
import 'package:lib1/lib1.dart' show foo;
import 'package:lib2/lib2.dart' hide foo;
```

Gli identificatori che iniziano con _ sono visibili solo all'interno della libreria

STATEMENT PER IL CONTROLLO DI FLUSSO

```
if (isRaining()) {
    ...
} else if
(isSnowing()) {
    ...
} else {
    ...
}
```

```
for (var i=0; i<5; i++) {
    print(i)
}

while (!isDone()) {
    doSomething();
}

do {
    printLine();
} while(!atEndOfPage());
```

```
switch(expression) {
  case 'A':
    ...
    break;
  case 'B':
    ...
    break;
  default:
    ...
}
```

ECCEZIONI

Le eccezioni sono non controllate

```
try {
    breedMoreLlamas();
} on OutOfLlamasException { // un'eccezione specifica
    buyMoreLlamas();
} on Exception catch (e) { // tutto ciò che è un'eccezione
    print('Unknown exception: $e');
}
```

EREDITARIETÀ

Le classi possono estendere altre classi ma una sola alla volta (single-inheritance)

Keywords abstract, extends, implements, @override

```
class TV {
  void turnOn() {
    _illuminateDisplay();
    _activateIrSensor();
  }
}
```

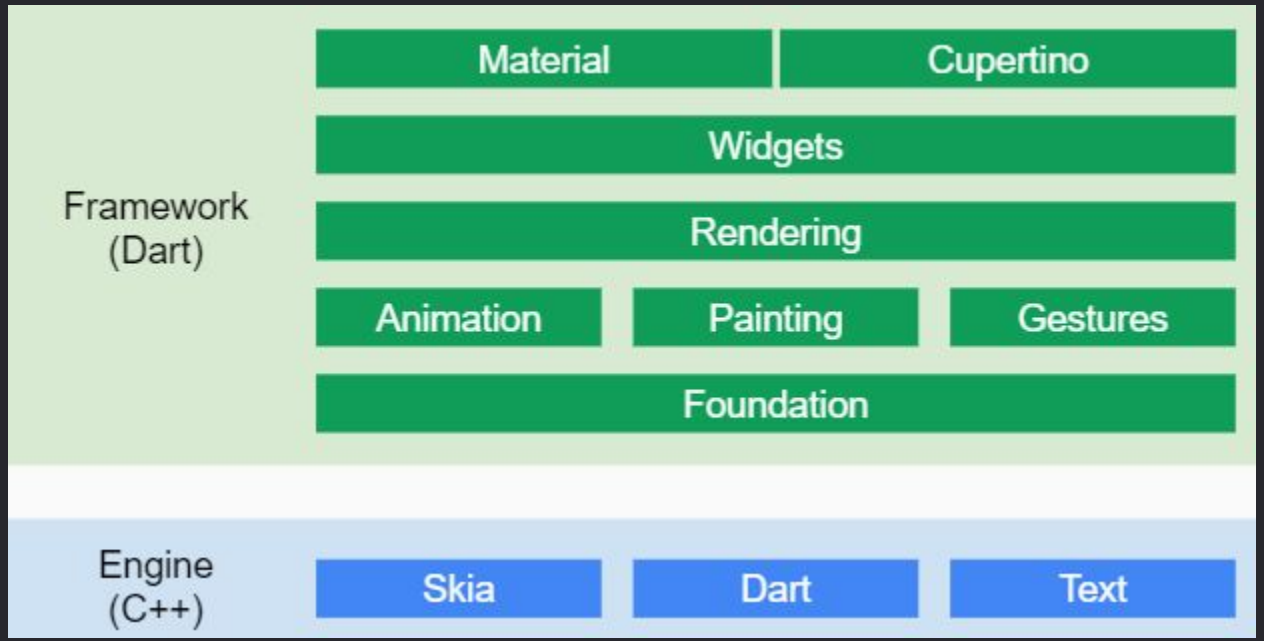
```
class SmartTV extends TV {
  void turnOn() {
    super.turnOn();
    _bootNetworkInterface();
  }
}
```

COMPILAZIONE CODICE DART

- Il codice Dart può essere compilato in diversi modi
  - just-in-time (JIT)
  - **ahead-of-time (AOT)**
    - Rende il framework **cross-compiled**

ARCHITETTURA

COMPONENTI DEL FLUTTER SDK



ARCHITETTURA FRAMEWORK

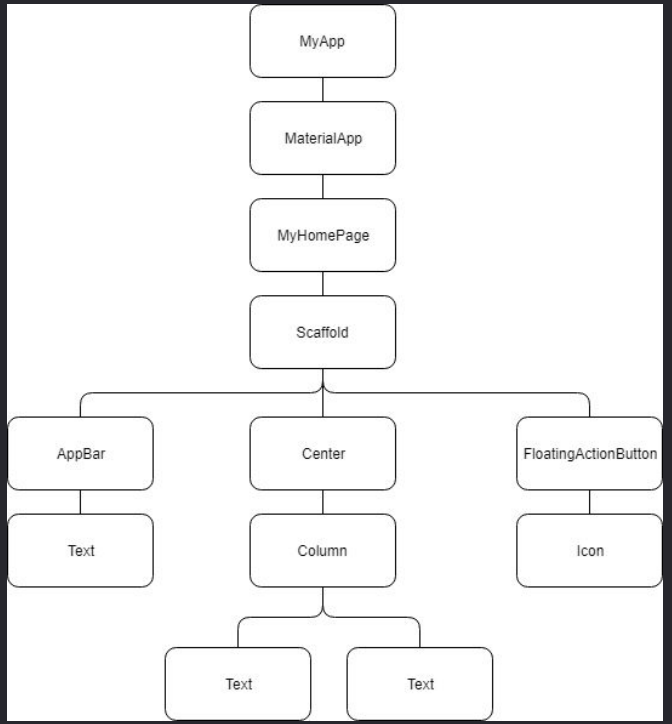
- Flutter presenta un'architettura composta dai seguenti strati:
  - **Material e Cupertino** : implementano widget in stile Material (Android) e Cupertino (iOS)
  - **Widgets** : implementa widget generici
  - **Rendering** : semplifica il processo di layout
  - **Animation** : tween e physics-based
  - **Painting, Gestures**
  - **Foundation**
  - **Dart:ui** : gestisce le comunicazioni con il Flutter engine

WIDGET

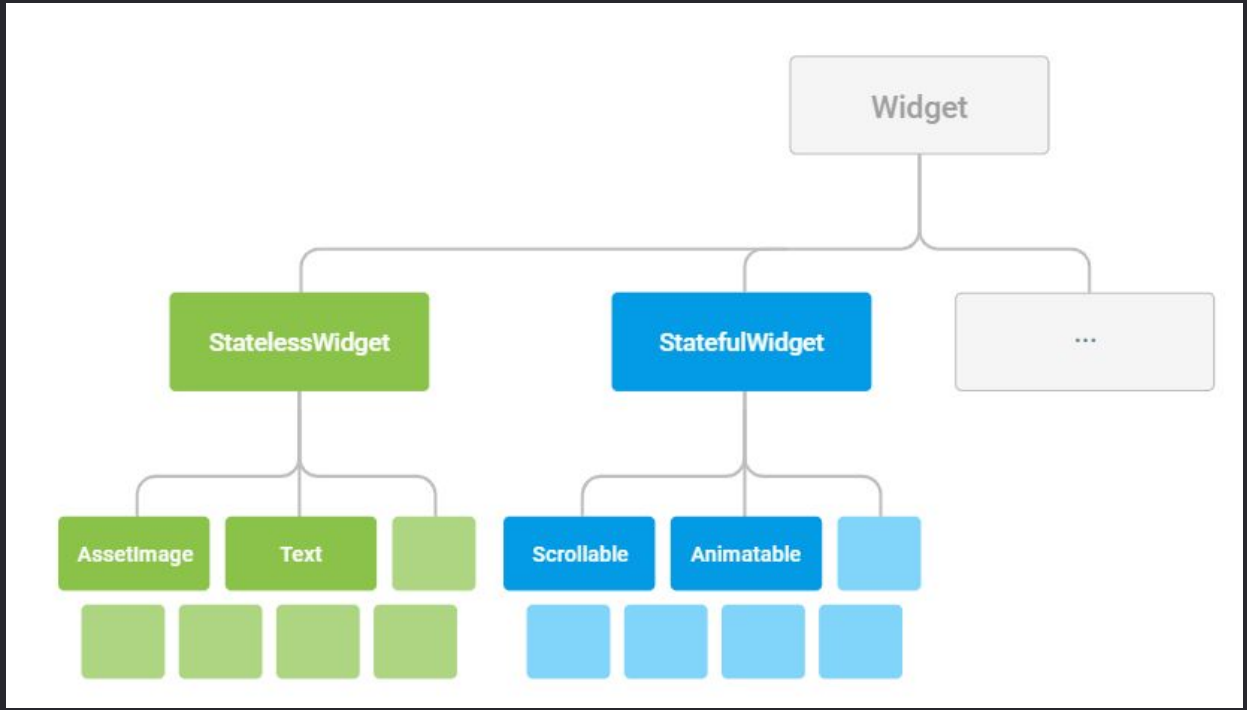
- Sono i componenti di base dell'interfaccia utente
- Ogni widget è una dichiarazione immutabile dell'interfaccia utente
- Un widget può definire:
  - Un elemento strutturale (bottone, menu, …)
  - Un elemento stilistico (font, …)
  - Un aspetto del layout (padding, …)
- Formano una gerarchia basata sulla composizione
- Permettono di rispondere agli eventi

COSTRUZIONE DI UN WIDGET

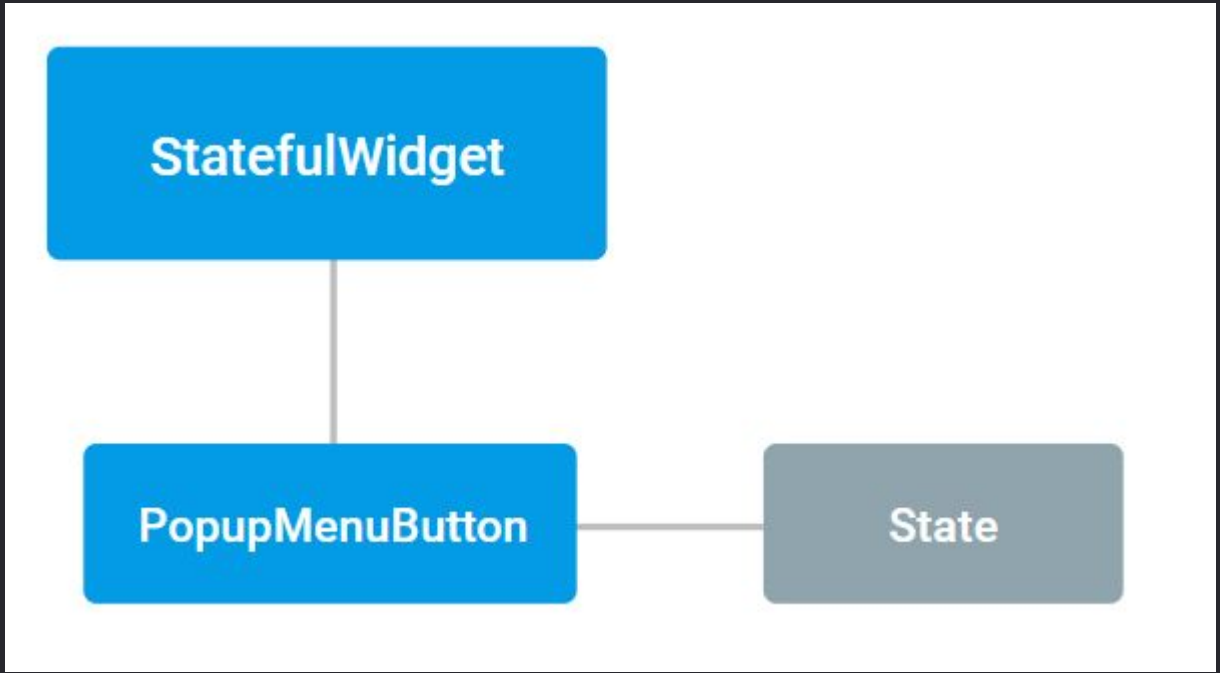
- Metodo build()
- Concetto di albero di widget



WIDGET : STATEFUL E STATELESS



STATEFUL WIDGET



Metodi importanti:

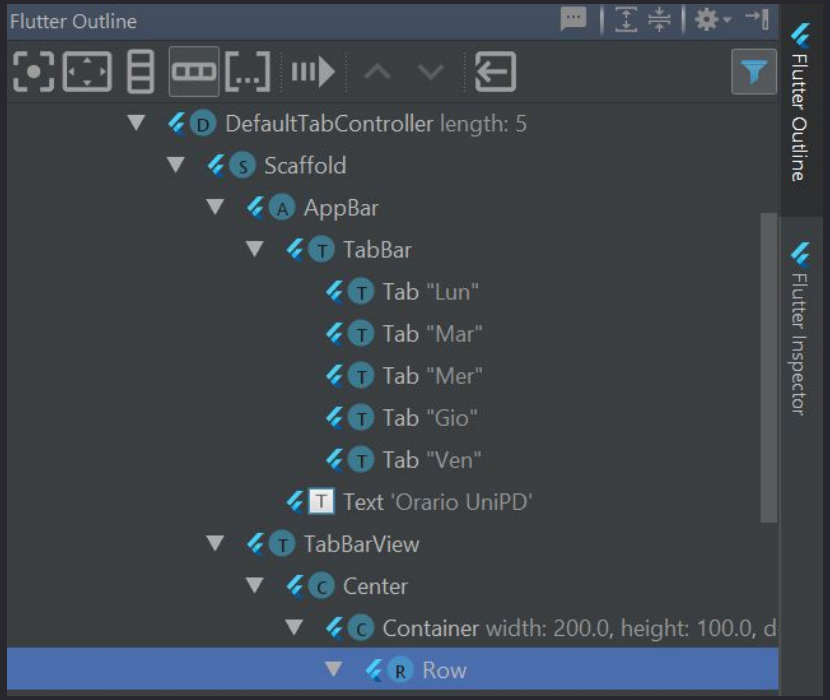
- createState()
- setState()

ESEMPI DI WIDGET

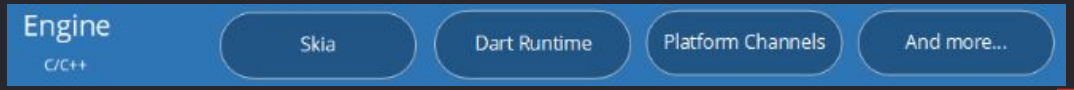
Flutter contiene una serie di widget di base, i più comunemente usati sono:

- Text
- Row
- Column
- Image
- RaisedButton
- AppBar

FLUTTER INSPECTOR



FLUTTER ENGINE

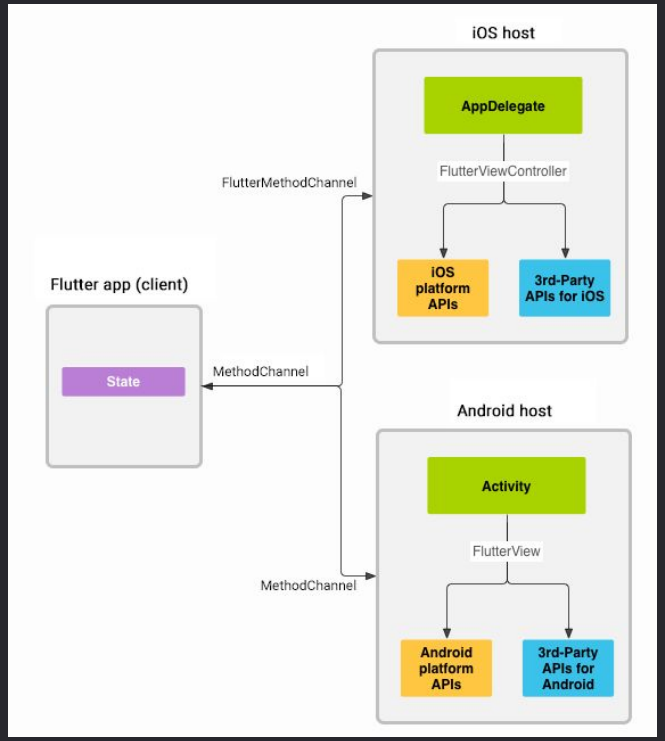


- Runtime enviroment scritto in C++
- Implementa le librerie chiave di Flutter
- Mette a disposizione:
  - Dart runtime
  - Skia
  - Platform channels

PLATFORM CHANNELS

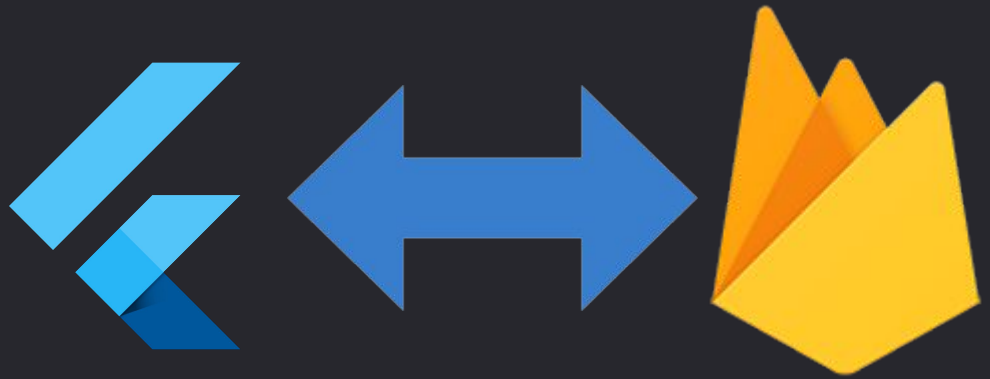
- Permettono la comunicazione tra codice Dart e codice specifico per la piattaforma
- Tipologie di canali:
  - BinaryMessages
  - MessageChannel
  - MethodChannel

CODE FORKING



ESTENSIONI

- Package
- Firebase



AMBIENTE DI SVILUPPO
ESEMPIO DI CODICE

AMBIENTE DI SVILUPPO

Per costruire applicazioni in Flutter sono necessari:

- Flutter SDK
- Un editor o un IDE, sono consigliati:
  - Android Studio
  - IntelliJ IDEA
  - Visual Studio Code
- Per gli IDE proposti sono disponibili i **flutter plugin**

INSTALLAZIONE FRAMEWORK

- È possibile installare Flutter su Windows, macOS o Linux
- Processo installazione:
  - installazione SDK
  - consigliata la modifica della variabile PATH
  - comando **flutter doctor** :
    - controllo dipendenze mancanti

FLUTTER DOCTOR

```
C:\Users\tomma>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, v1.2.1, on Microsoft Windows [Versione 10.0.17134.590], locale it-IT)
[√] Android toolchain - develop for Android devices (Android SDK version 28.0.3)
[√] Android Studio (version 3.1)
[!] IntelliJ IDEA Ultimate Edition (version 2018.1)
    X Flutter plugin not installed; this adds Flutter specific functionality.
    X Dart plugin not installed; this adds Dart specific functionality.
[!] Connected device
    ! No devices available

! Doctor found issues in 2 categories.

C:\Users\tomma>
```

UN SEMPLICE ESEMPIO DI CODICE

In questo semplice esempio impareremo ad utilizzare le seguenti componenti del framework:

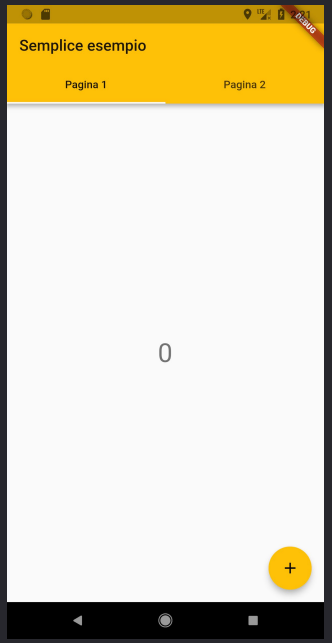
- Stateful widget
- Stateless widget
- Layout a schede

L'applicazione consiste in un layout a schede con le seguenti pagine:

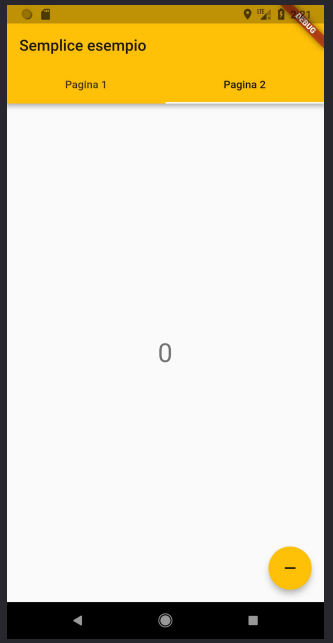
- Pagina 1: permette di incrementare un contatore tramite la pressione di un pulsante
- Pagina 2: permette di decrementare un puntatore tramite la pressione di un pulsante

IL NOSTRO OBIETTIVO

Pagina 1



Pagina 2



CLASSI NECESSARIE

```
class MyApp extends StatelessWidget {...}
class FirstPage extends StatefulWidget {...}
class SecondPage extends StatefulWidget {...}
class _FirstPageState extends State<FirstPage> {...}
class _SecondPageState extends State<SecondPage> {...}
```

PRIMA PAGINA

```
class FirstPage extends StatefulWidget {
    FirstPage({Key key, this.title}) : super(key: key);
    final String title;
    @override
    _FirstPageState createState() => _FirstPageState();
}
```

STATO DELLA PRIMA PAGINA - 1

```
class _FirstPageState extends State<FirstPage> {
 int _counter1 = 0;
 void _incrementCounter() {
   setState(() {
     _counter1++;
   });
 }
```

STATO DELLA PRIMA PAGINA - 2

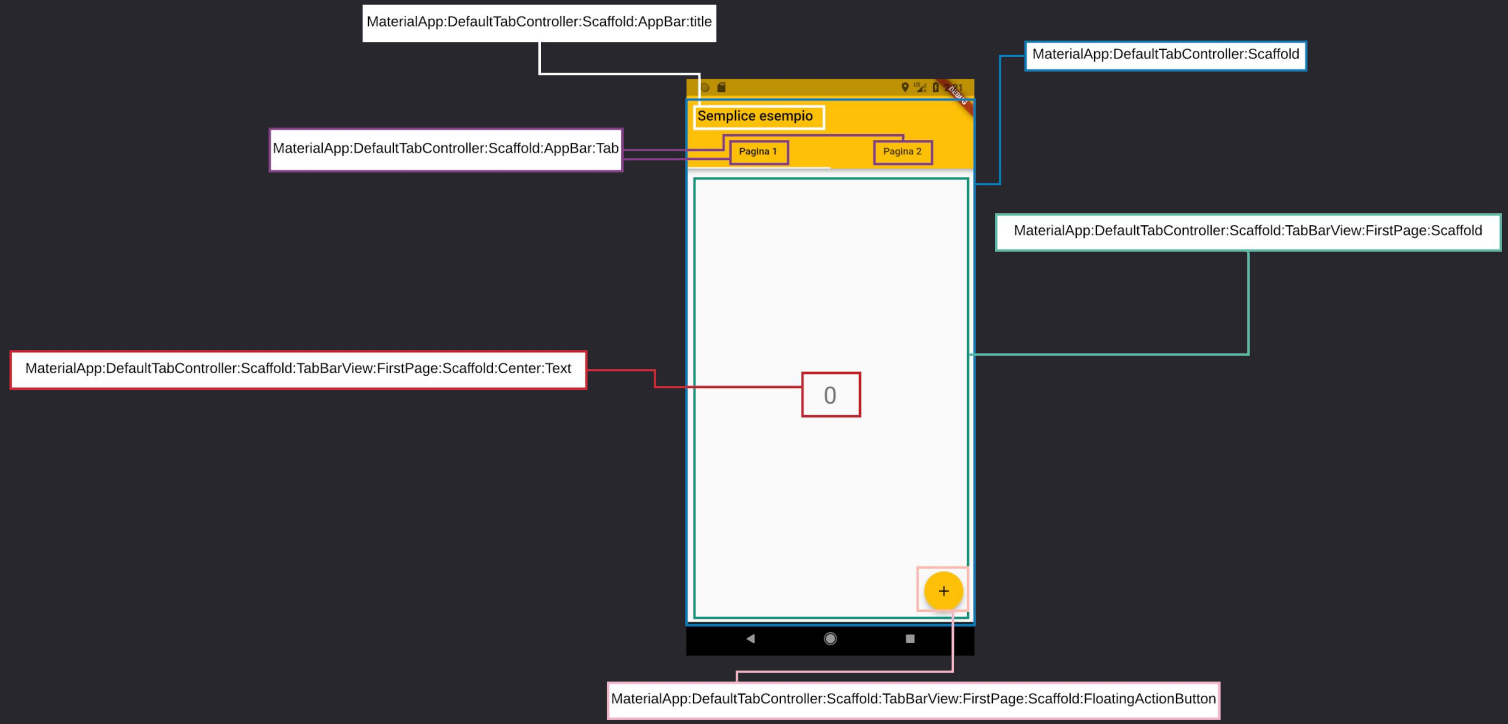
```
@override
 Widget build(BuildContext context) {
   return Scaffold(
     body: Center(
       child: Text(
             '$_counter1',
       ),
     ),
floatingActionButton: FloatingActionButton(
       onPressed: _incrementCounter,
       tooltip: 'Increment',
       child: Icon(Icons.add),
     ),
   );
 }
}
```

LA NOSTRA APPLICAZIONE

```
class MyApp extends StatelessWidget {
 // This widget is the root of your application.
 @override
 Widget build(BuildContext context) {
   return MaterialApp(
     title: 'Flutter Demo',
     theme: ThemeData(
       primarySwatch: Colors.amber;
     ),
     home: DefaultTabController(
         length: 2,
```

```
child: Scaffold(
    appBar: AppBar(
        bottom: TabBar(
            tabs: [ Tab(text: "Pagina 1"),
                    Tab(text: "Pagina 2")]
        ),
        title: Text("Semplice esempio"),
    ),
    body: TabBarView(
        children: [
            FirstPage(title: "Prima pagina"),
            SecondPage(title: "Seconda pagina")
        ]),));}}
```

VISUALIZZAZIONE GRAFICA



FONTI - 1

- Flutter - https://flutter.dev/
- Flutter Docs - https://docs.flutter.io/
- Dart - https://www.dartlang.org/
- Platform Channels - https://flutter.dev/docs/development/platform-integration/platform-channels
- Pro and cons of Flutter - https://hackernoon.com/flutter-pros-and-cons-for-seamless-cross-platform-development-c81bde5a4083
- Wikipedia - https://en.wikipedia.org/wiki/Flutter_(software)

FONTI - 2

- Flutter engine - https://github.com/flutter/engine
- Architettura Flutter - https://medium.com/flutter-community/the-layer-cake-widgets-elements-renderobjects-7644c3142401
- Flutter inspector - https://flutter.github.io/devtools/inspector
- Google SKIA - https://skia.org/