Flutter course 2024

Coding with Flutter

Everything is a widget

POLITECNICO OPEN unix LABS

Manuel Zani
zani.manuel@poul.org

POLITECNICO OPEN
unix LABS

Who are we?

We are a non-profit association and hacking community formed by students from 🔗Politecnico di Milano

POLITECNICO OPEN unix LABS

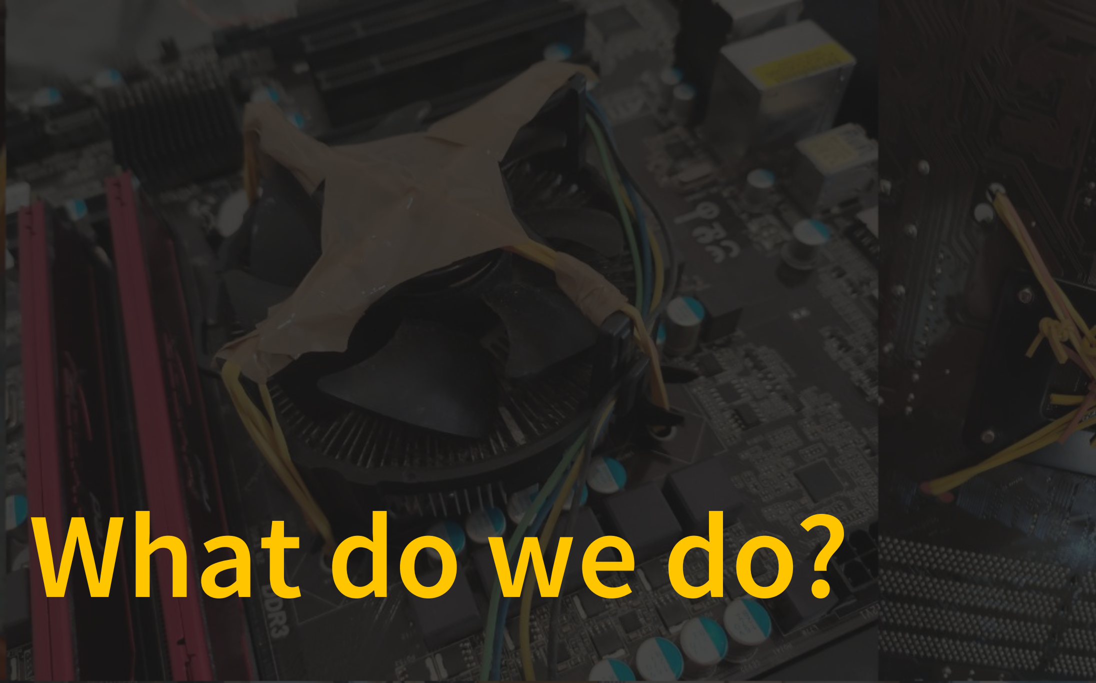
What do we do?

We promote the use of 🔗 **Free and Open Source Software** through courses, talks and workshops

POLITECNICO OPEN unix LABS

What do we do?

We tinker with **software and hardware** and share knowledge with anyone who is curious about technology



POLITECNICO OPEN unix LABS

What do we do?

...but also

Creative ~~Cazzeggio~~ Development

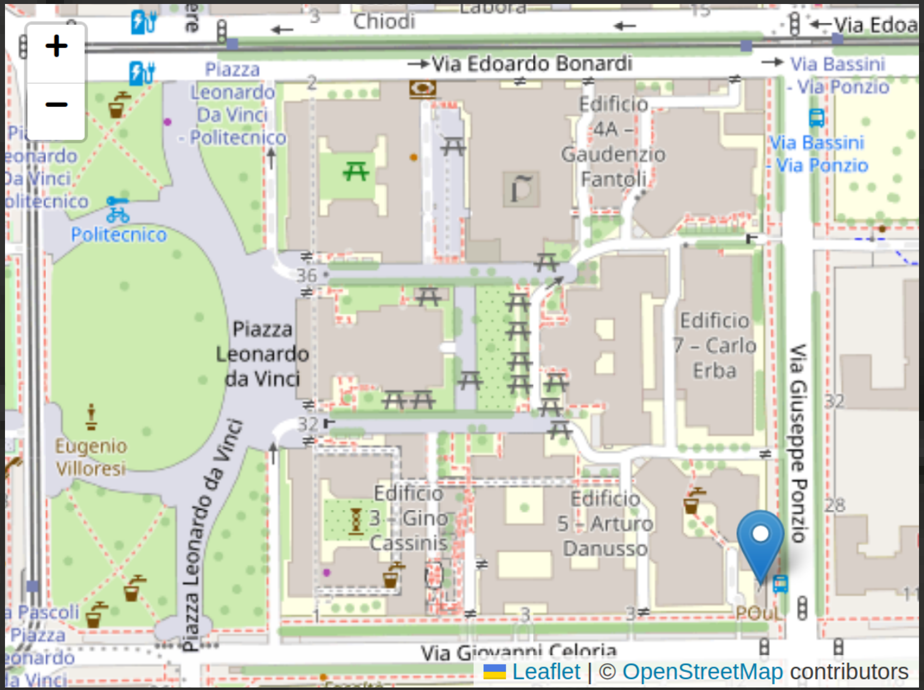
POLITECNICO OPEN unix LABS

Where are we?

Politecnico di Milano (Leonardo campus), building 9A

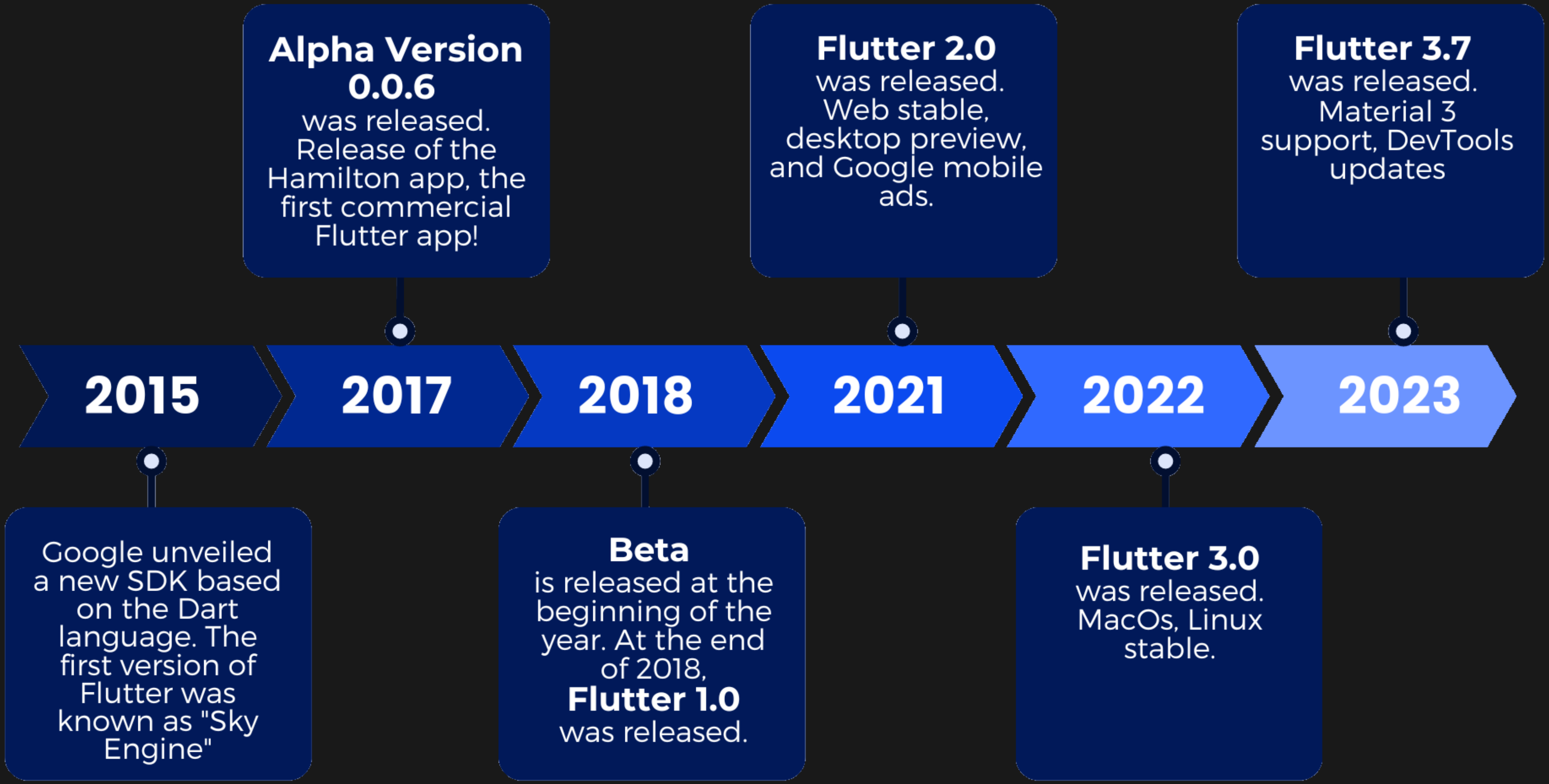
Come visit us!

Make sure to check 🔗BITS to know if the headquarters are open!



Welcome to Flutter

Some history



2015

Google unveiled a new SDK based on the Dart language. The first version of Flutter was known as "Sky Engine"

2017

Alpha Version 0.0.6 was released. Release of the Hamilton app, the first commercial Flutter app!

2018

Beta is released at the beginning of the year. At the end of 2018, **Flutter 1.0** was released.

2021

Flutter 2.0 was released. Web stable, desktop preview, and Google mobile ads.

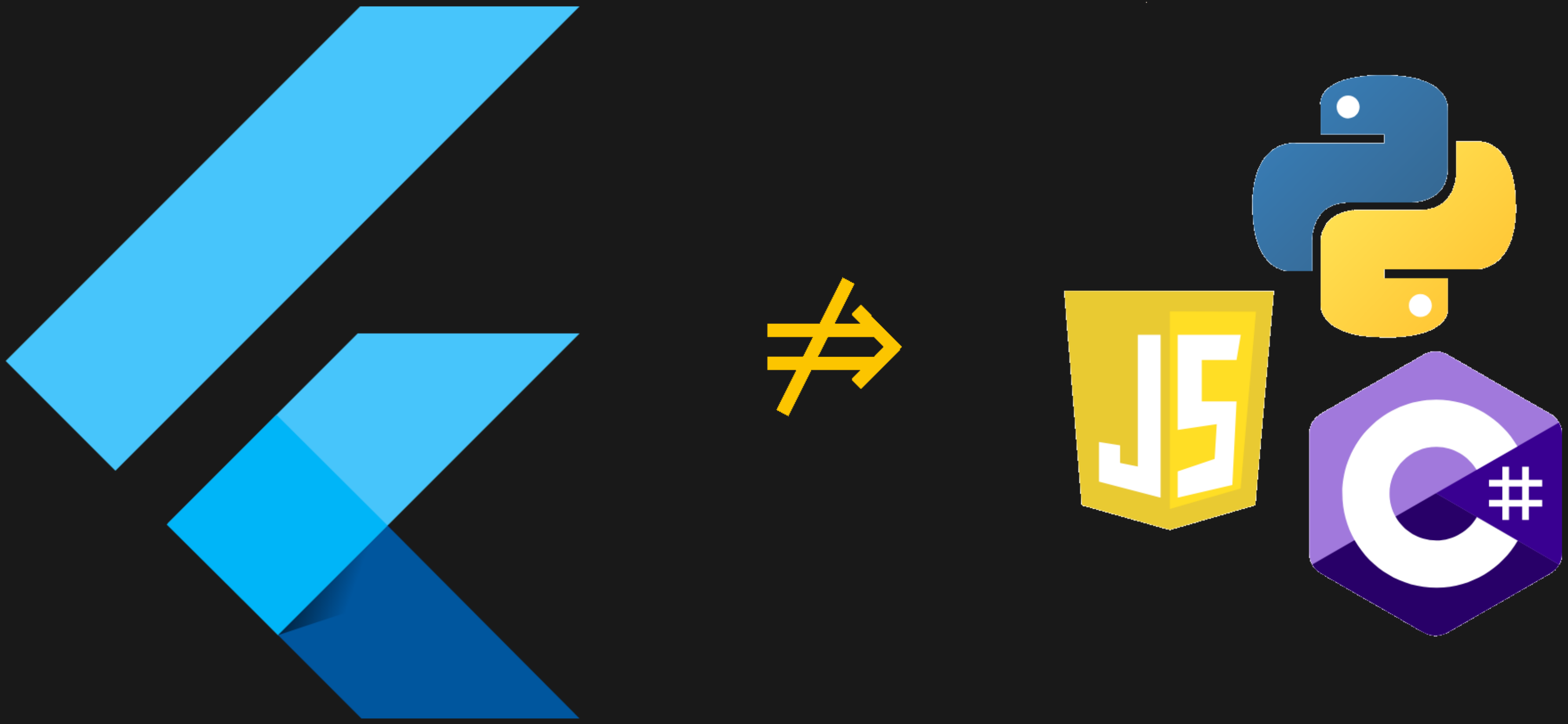
2022

Flutter 3.0 was released. MacOs, Linux stable.

2023

Flutter 3.7 was released. Material 3 support, DevTools updates

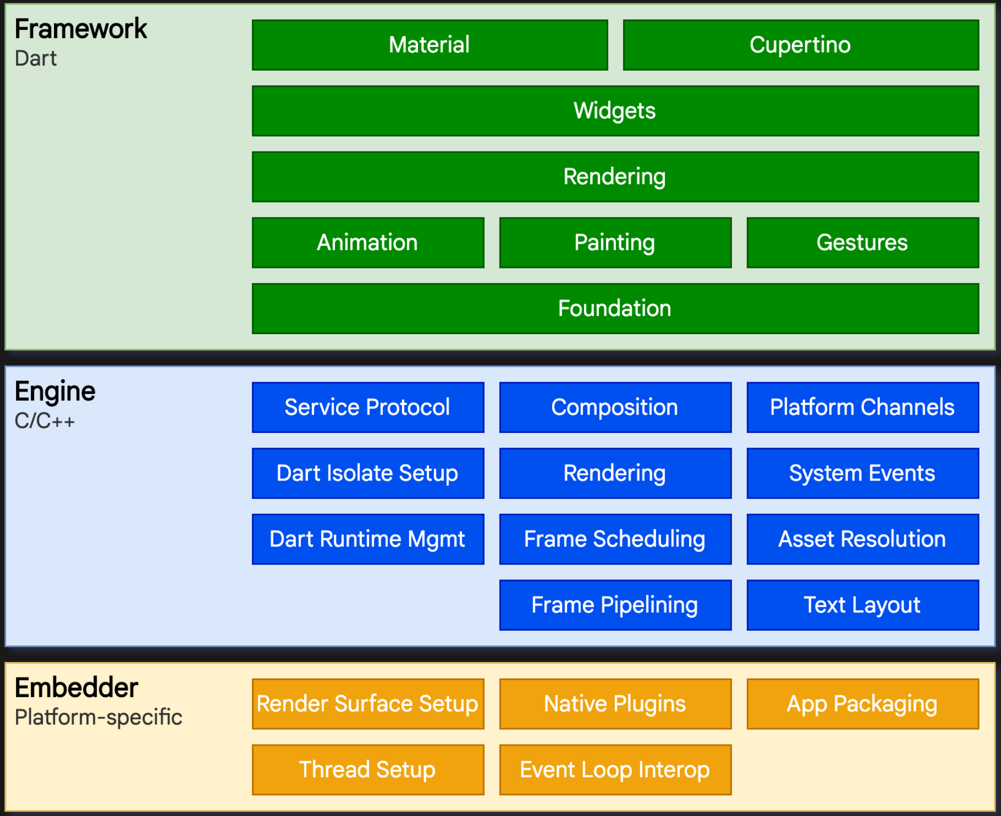
What is Flutter?



Speaker notes

Flutter is an SDK

Flutter is an SDK



Speaker notes

- Embedder: it's the lower layer and it's build
- Engine: it's built using C and C++
- Framework: it's the layer used by the programmer to use the tools and the API

The embedder

It’s the lowest layer and it’s **built** specifically **for every platform**

- Linux and Windows: C++
- Android: C++ and Java
- iOS and macOS: Objective-C and Objective C++

The engine

It’s the layer built on top of the embedder, built with C and C++

The framework

It’s the layer on top of everything:

- this is what allows to use the tools from the SDK
- it’s built with Dart

What’s Dart?

Multiplatform object-oriented programming language

Object-oriented?

Means that we have to work with classes and objects

What are objects?

- An object is an instance of a class
- A class is a collection of “things”

For example:

1. Erik wants to subscribe to an app, so he has to become a user
2. He gives his data to the app and he becomes a user
3. Erik has become an instance of the app’s users

Speaker notes

- A class is a collection of things
- App subscription example
- this: means that it's considered the variable inside the class
- required: means that it's necessary to give a certain value when creating the object

Class methods

A class is not made just by variables but also methods

There are two type of methods:

- those accessed through the **object**
- those accessed through the **class**

Class inheritance

Inheritance gives the ability to reuse code:

- a child class inherits everything from the parent class

Keywords:

- **extends**: the child class inherits everything from the parent class
- **super**: it refers to the **parent** class
- **this**: it refers to the **child** class

The override

It allows the child to override “things” inherited from the parent

Dart’s Features

- AOT and JIT
- Automatic garbage management
- Static and dynamic types
- Sound null safety
- Async and await
- Shorthand syntax for functions

AOT (Ahead of time)

Can compile to native ARM or x64 machine code, assuring also short startup time

JIT (Just in time)

Uses incremental recompilation enabling:

- Hot reload
- Live metrics (DevTools)
- Rich debug tools

Automatic garbage management

Static and dynamic types

- **Static types**: more consistency and better performance
- **Dynamic types**: more flexibility

Advice

If you know the type of a certain variable you should use static types

Sound null safety

- **Problem**: access to variables set to null can cause errors
- **Solution**: force the programmer to write code where it’s impossible to access a null variable when it shouldn’t be null

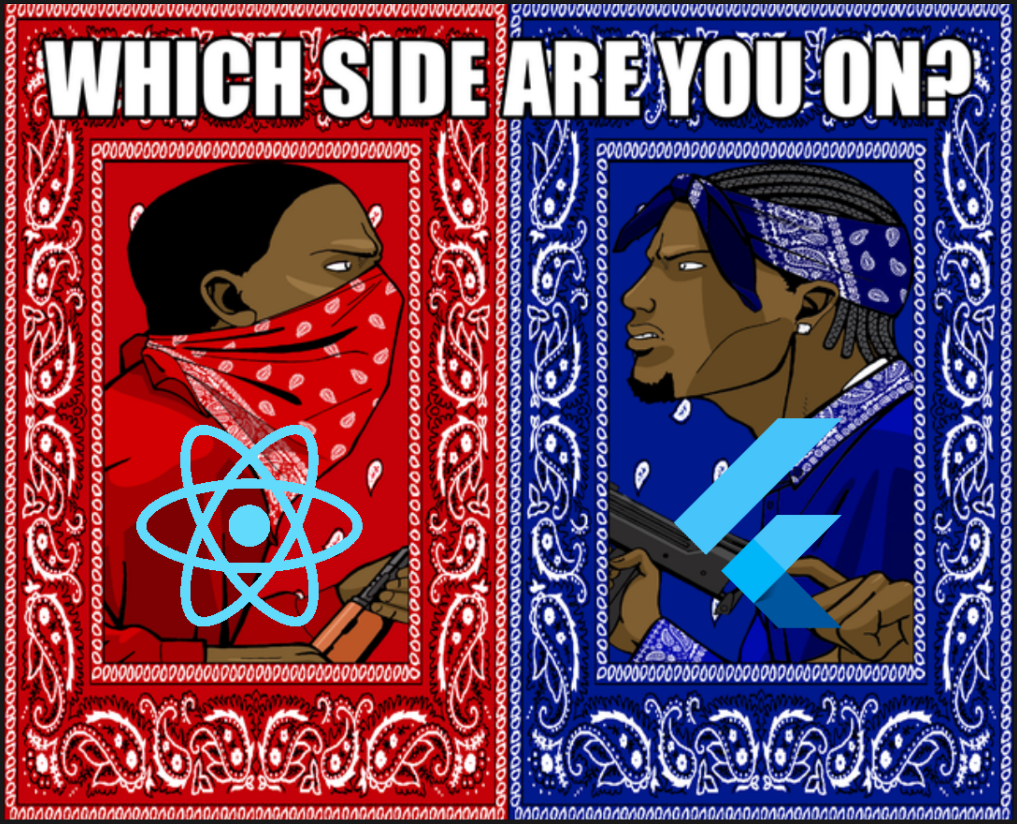
Async and await

- **Problem**: some operations take a long time to complete
- **Solution**: make sure that the execution waits for the end of a certain statement

Shorthand syntax for functions

These are some of the reasons why Flutter uses Dart

Let’s get back to Flutter



The difference is that:

- **React Native**: the code is run through javascript layers
- **Flutter**: the code is run natively

What’s Flutter adding?

- Widget
- State management
- No XML and similar formats for UI and UX

The Widgets

“A widget is an immutable description of part of a user interface”

🔗Widget documentation

- They are the building blocks for almost everything
- They have a hierarchical structure so they can be composed

How do they look?

```
Widget(
    propertyA: ,
    propertyB: ,
    ... ,
);
```

```
Widget(propertyA: , propertyB: , ... ,);
```

They’re nothing new

```
Widget(propertyA: , propertyB: , ... ,);
```

```
User(nickname: 'Amogus', age: 3, height: 3.47);
```

Widgets are just objects

So a widget is an instance of a class

```dart
class User{
    User({required this.nickname, required this.age, this.height=2});
    final String nickname;
    final int age;
    final double height;

    sayHi(){
      print("Hi");
    }
}
```

```dart
class Spacer extends StatelessWidget {
    const Spacer({super.key, this.flex = 1}): assert(flex > 0);
    final int flex;

    @override
    Widget build(BuildContext context) {
        return Expanded(
            flex: flex,
            child: const SizedBox.shrink(),
        );
    }
}
```

Widgets can be nested

Let’s say that I want some text in a yellow box

```
ColoredBox(
    color: Colors.yellow,
    child: Text("hi"),
)
```

Let’s say that I also need the Scaffold widget

```
Scaffold(
    body: ColoredBox(
        color: Colors.yellow,
        child: Text("hi"),
    )
)
```

Since I have a Scaffold widget I can add an AppBar with a title

```
Scaffold(
    appbar: AppBar(
        title: Text("Hello"),
    ),
    body: ColoredBox(
        color: Colors.yellow,
        child: Text("hi"),
    )
)
```

From this short example I just want to show how easy it is to create a nested structure:

- in Flutter it **is necessary** to be comfortable with it

Widgets state

A widget is immutable, so how can we use it in a world where we need *interactive* programs?

Here come the concepts of:

- Stateless
- Stateful

Stateless

A Stateless widget is immutable, so it doesn’t change, no matter what

Stateful

A Stateful widget has a state, so it can change

For example

1. There is some user input
2. Some data of the widget has been changed
3. So the state of the widget has changed
4. The widget is rebuilt with the new data

UI and UX

If you want to use Flutter you need to know multiple widgets

There are plenty of widgets and it’s almost impossible to know all of them, so now we are going to see just some of those that we will use in the next days

Scaffold

It’s the base of a Flutter app and you can put a lot of different widgets inside of it

Center

It centers the child widget inside

AppBar

It’s a highly customizable bar

FloatingActionButton

It’s a floating button

Row

It puts a list of widgets in a row

Column

It puts a list of widgets in a column

ListView.builder

It puts a list of widgets in a scrollable view

Card + ListTile

It’s a nice combination of two widgets:

- **Card**: it creates a colored elevated box
- **ListTile**: it has a lot of properties to show data in a nice way

Card + ListTile + ListView.builder

Combining **Card + ListTile** with **ListView.builder** creates a nice scrollable list

Expanded

It forces a widget to occupy the max amount of space

What will be next?

Thank you for your attention!

Flutter course 2024

Coding with Flutter

Speakers

Manuel Zani

zani.manuel@poul.org

Slides authors

Manuel Zani

zani.manuel@poul.org

Special thanks to

Dario Biščević

dbisc@poul.org

POLITECNICO OPEN
unix LABS