Workshop
Introduction to App Development

A shallow dive into the world of App Development and Flutter

https://github.com/NEEECFEUP/WS-App-Development/

André Campanhã

November 2022

NEEEC - FEUP

Introduction

- Mobile framework for creating native(ish)[1] apps for Mobile (Android, iOS), Desktop (Win, Mac, Linux), Web, and Embedded
- Write once (dart), run everywhere(ish)[1]

[1] Flutter can access native API calls via native platform code (i.e. Android SDK). This can be somewhat abstracted by using cross-platform dart packages.

NEEEC-FEUP

Prerequisites

- Programming basics (OOP familiarity recommended)
- Having followed Flutter's installation steps sent by e-mail
  - Install Flutter (Get the Flutter SDK and Android Setup)
  - Set up an editor (Recommended: VS Code)

NEEEC-FEUP

Outline

Basics

Widgets

Task 1 - Basic layout

Navigation

Task 2 - And another screen

Change the State

Basics

Everything is a widget

The core of Flutter's layout mechanism is widgets.

They describe what their view should look like given their current configuration and state.

When a widget's state changes, the widget rebuilds its description, which the framework diffs against the previous description in order to determine the minimal changes needed in the underlying render tree to transition from one state to the next.

NEEEC-FEUP

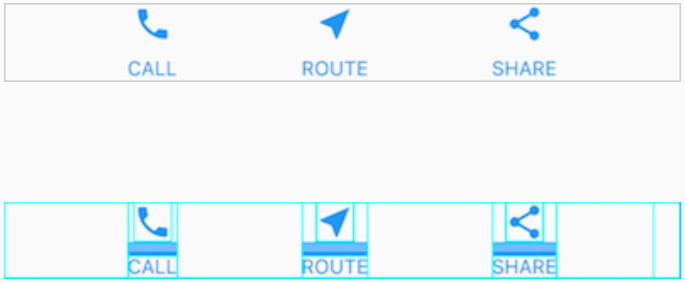
Everything is a widget

How many widgets here?

NEEEC-FEUP

Everything is a widget

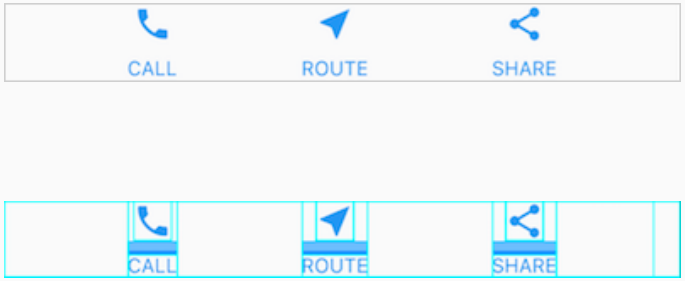
How many widgets here?



NEEEC-FEUP

Everything is a widget

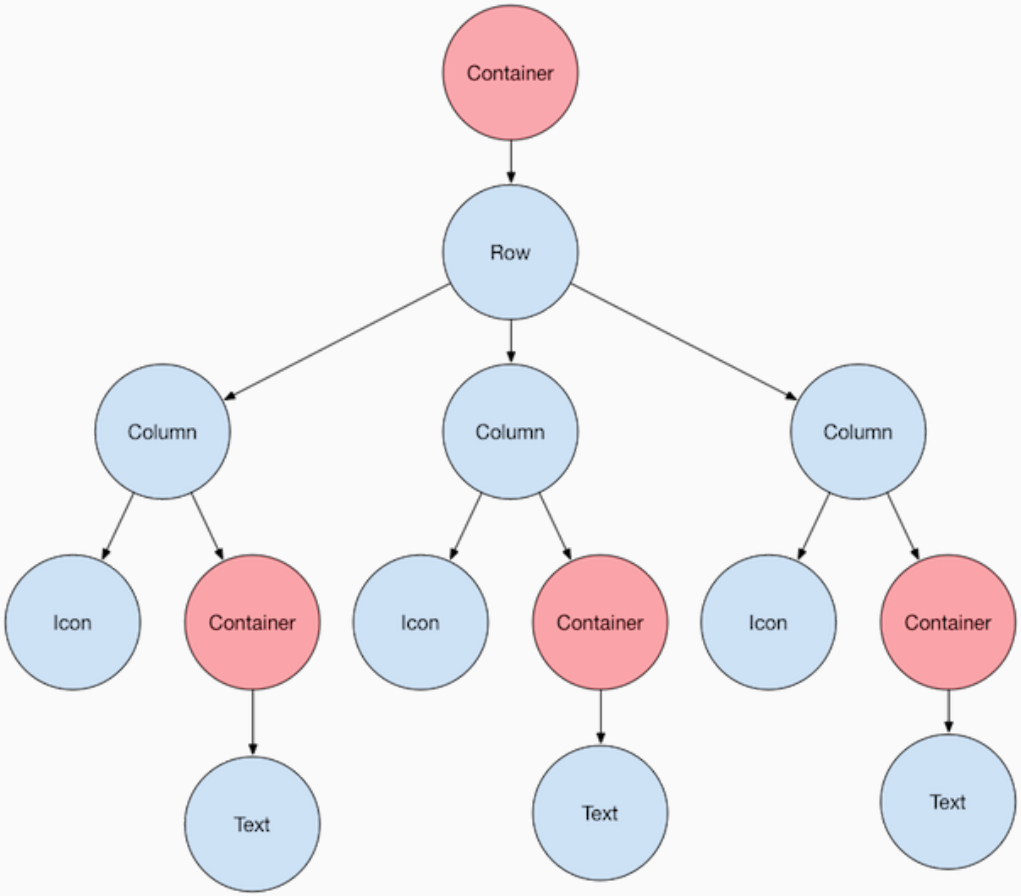
How many widgets here?



14 widgets!

NEEEC-FEUP

Everything is a widget



NEEEC-FEUP

What is State?

State

Information that can be read synchronously when the widget is built and might change during the lifetime of the widget.

NEEEC-FEUP

Types of widgets

There are 2 types of widgets:

Types of widgets

There are 2 types of widgets:

Stateless Widgets

They do not own a mutable state.

Useful when the part of the user interface you are describing does not depend on anything other than the configuration information in the object itself and the context in which the widget is inflated.

NEEEC-FEUP

Types of widgets

There are 2 types of widgets:

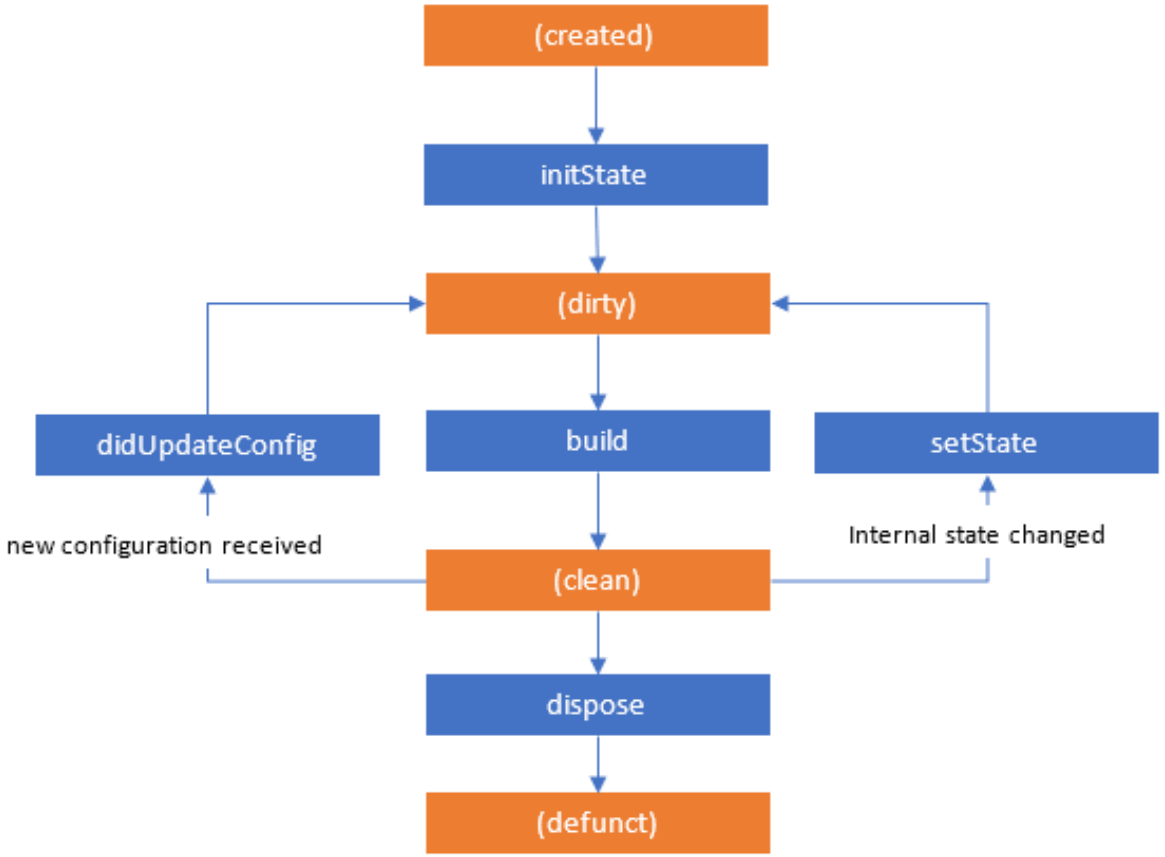
Stateful Widgets

They are themselves immutable. However, they own a State.

Useful when the part of the user interface you are describing can change dynamically, e.g. due to having an internal clock-driven state or depending on some system state

NEEEC-FEUP

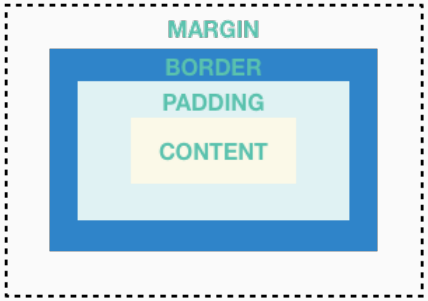
Widget Lifecycle



NEEEC-FEUP

Widgets

Container

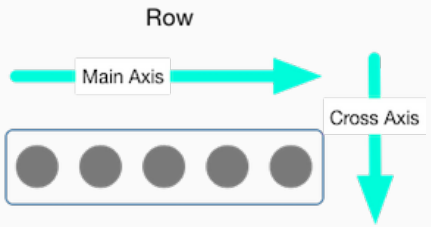



```
Container(
  margin: const EdgeInsets.all(10.0),
  color: Colors.amber[600],
  width: 48.0,
  height: 48.0,
);
```

https://api.flutter.dev/flutter/widgets/Container-class.html

NEEEC-FEUP

Row

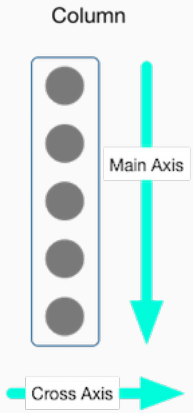



```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Image.asset('images/pic1.jpg'),
    Image.asset('images/pic2.jpg'),
    Image.asset('images/pic3.jpg'),
  ],
);
```

https://api.flutter.dev/flutter/widgets/Row-class.html

NEEEC-FEUP

Column




```
Column(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    Image.asset('images/pic1.jpg'),
    Image.asset('images/pic2.jpg'),
    Image.asset('images/pic3.jpg'),
  ],
);
```

https://api.flutter.dev/flutter/widgets/Column-class.html

NEEEC-FEUP

Text

```
Text(
  'Hello, $_name! How are you?',
  textAlign: TextAlign.center,
  overflow: TextOverflow.ellipsis,
  style: const TextStyle(fontWeight: FontWeight.bold),
);
```

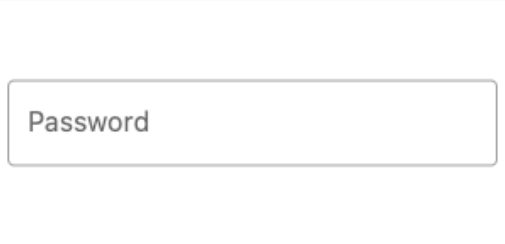
https://api.flutter.dev/flutter/widgets/Text-class.html

Elevated Button

```
ElevatedButton(
  style: style,
  onPressed: () {print("Hello")},
  child: const Text('Click me'),
),
```

https://api.flutter.dev/flutter/material/ElevatedButton-class.html

TextField




```
TextField(
  obscureText: true,
  decoration: InputDecoration(
    border: OutlineInputBorder(),
    labelText: 'Password',
  ),
)
```

https://api.flutter.dev/flutter/material/TextField-class.html

NEEEC-FEUP

Widget Catalog

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

Now that we got that covered...

Task 1 - Basic layout

Task 1 - Basic Layout

If you haven't, get the source code from the Github Repository.

Hint!

To print to the terminal, use:

```
import 'dart:developer' as developer;
```

and

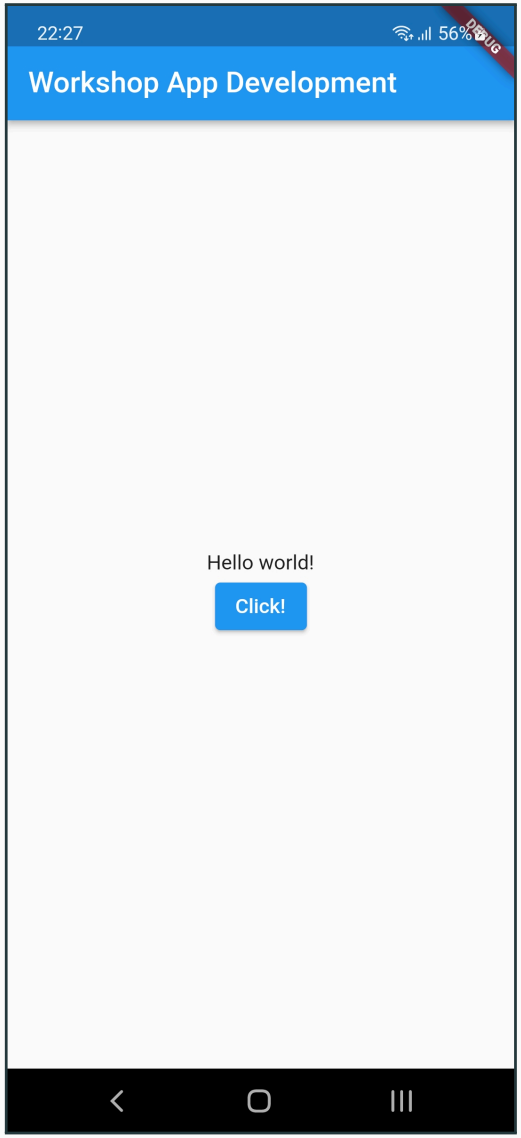
```
developer.log('Hello!', name: 'app.log');
```

Task 1 - Basic Layout

- Create a screen with a Text ”Hello World”
- A button ”Click” under it
- When you click the button, print something to the terminal

NEEEC-FEUP

Task 1 - Basic Layout



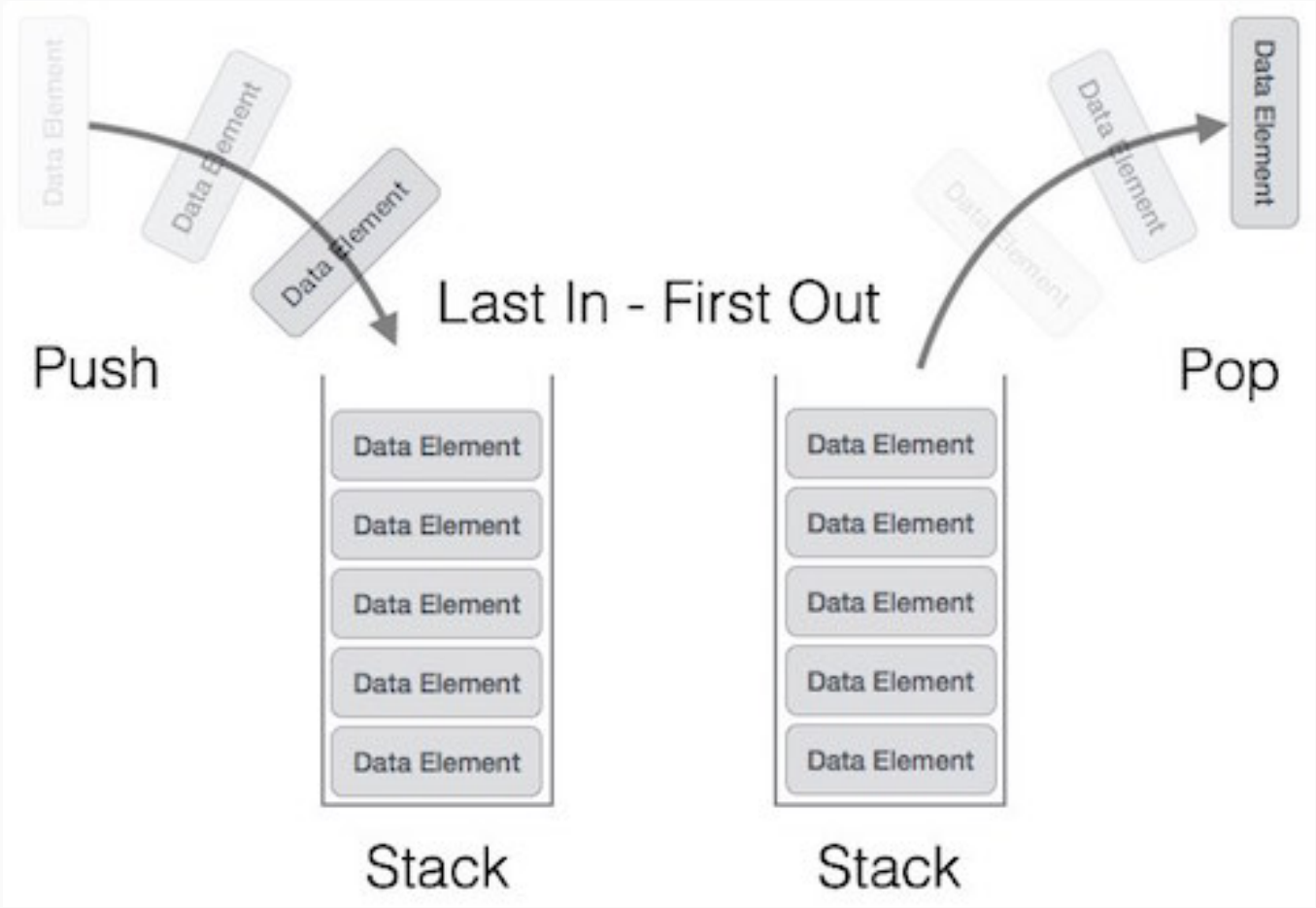
NEEEC-FEUP

Task 1 - Basic Layout

NEEEC-FEUP

Navigation

History stack



NEEEC-FEUP

Navigation

Push

```
Navigator.of(context).push(
    MaterialPageRoute(
        builder: (context) => const SongScreen(song: song),
    ),
);
```

Pop

```
Navigator.of(context).pop("Return value");
```

NEEEC-FEUP

Task 2 - And another screen

Task 2 - And another screen

Hint!

You can add properties to a widget and it's constructor

```
final String title;
const SecondScreen({super.key, required this.title});
```

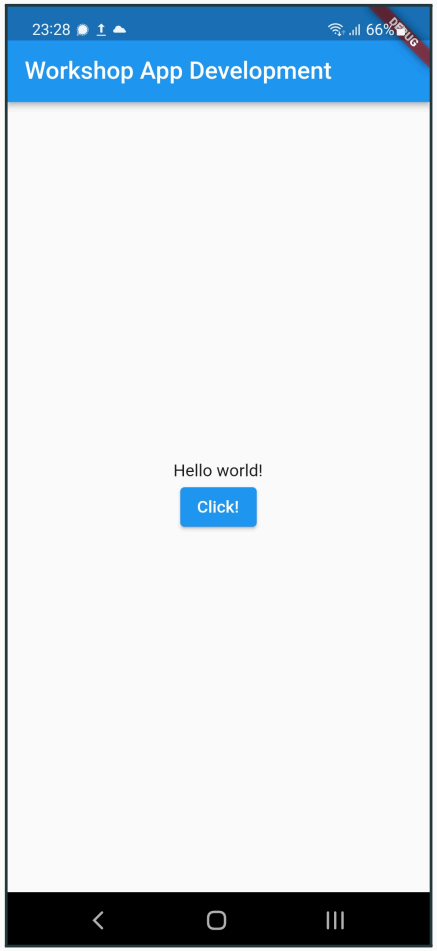
and access them in the State with, for example,

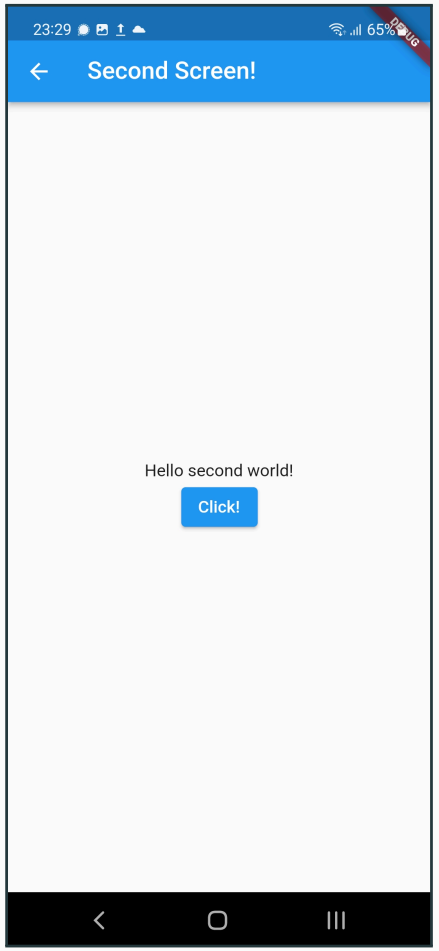
```
widget.title
```

Task 2 - And another screen

- Create a second screen with the same layout as the first one, but that takes in the title as an argument
- Make the button on the first open the second

Task 2 - And another screen





NEEEC-FEUP

Change the State

Detect Gestures

Sometimes we need to detect input in things other than buttons...

Gesture detector

```
GestureDetector(
    onTap: handleTap,
    child: Text("Hellooo"),
)
```

Change the state

Potential Problems

We can't simply mutate the state.

The framework needs to be notified, in order to rebuild the widget tree with the changes.

Change the state

setState

You must use the setState function to mutate the state!

```
setState(() {
    text = "banana";
});
```

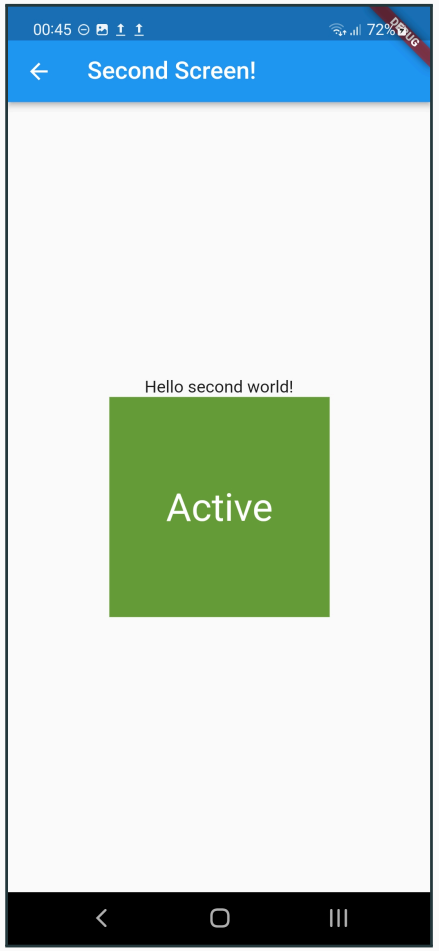
Task 3 - Just an ordinary box

Task 3 - Just an ordinary box

- Create a stateless widget called Box. It should:
  - have a width and a height of 200dp (default unit).
  - have a boolean (bool) property ”active”
  - if active, it should be green and have a text saying ”Active”
  - if inactive, it should be gray and a Text saying ”Inactive”
- Place a box on the second screen.
- Make it so that when you click on the box, it changes from active to inactive (remember GestureDetector?)
- It should be inactive by default

NEEEC-FEUP

Task 3 - Just an ordinary box



NEEEC-FEUP

References

[1] Google, “Flutter documentation.”
https://docs.flutter.dev/ (last accessed Nov. 04, 2022).

NEEEC-FEUP

Questions?