

Programmazione Mobile

Nicola Noviello

nicola.noviello@unimol.it

Corso di Laurea in Informatica
Dipartimento di Bioscienze e Territorio
Università degli Studi del Molise
Anno 2023/2024

Lezione: Flutter e Dart Foundation (quarta ed ultima parte)

- Basi di Flutter
- Sintassi di Dart
- Comprendere il funzionamento del framework
- Primi approcci alla creazione di un progetto
- Widgets





Dove eravamo
rimasti?

Parametri nominali in una classe

```
const startAlign = Alignment.topLeft;  
const endAlign = Alignment.bottomRight;
```

```
class MyWidget extends StatelessWidget {
```

```
  const MyWidget(  
    {super.key,  
    required this.color1,  
    required this.color2});
```

```
  final Color color1;  
  final Color color2;  
  @override  
  Widget build(BuildContext  
    ) => return Container(  
    );
```

```
MaterialApp(  
  home: Scaffold(  
    appBar: AppBar(  
      title: const MyText('Programmazione Mobile'),  
      backgroundColor: const Color.fromARGB(255, 36, 52, 73),  
    ), // titolo dell'app // AppBar  
    body: const MyWidget(  
      color1: Color.fromARGB(255, 58, 40, 89),  
      color2: Color.fromARGB(255, 220, 20, 17)), // MyWidget  
    ), // Scaffold  
  ), // MaterialApp
```

Costruttori multipli

```
class MyWidget extends StatelessWidget {  
  const MyWidget({super.key, required this.color1, required this.color2});  
  const MyWidget.appdefault({super.key})  
    : color1 = const Color.fromARGB(255, 58, 40, 89),  
      color2 = const Color.fromARGB(255, 220, 20, 17);  
  final Color color1;  
  final Color color2;  
  @override  
  Widget build(BuildContext context) {  
    return Container(  
      // definiamo lo stile con Container
```

Costruttori multipli

```
appBar: AppBar(  
  title: const MyText('Programmazione Mobile'),  
  backgroundColor: const Color.fromARGB(255, 36, 52,  
) , // titolo dell'app // AppBar  
  // body: const MyWidget(  
  //   color1: Color.fromARGB(255, 58, 40, 89),  
  //   color2: Color.fromARGB(255, 220, 20, 17),  
  // ),  
  body: const MyWidget.appdefault(),  
)
```

Button

```
child: Column(  
  mainAxisAlignment: MainAxisAlignment.center,  
  children: <Widget>[  
    Image(  
      image: AssetImage('images/registro.jpeg'),  
      width: 400,  
    ), // Image  
    MyText('Come andrà l\'esame?'),  
    // ElevatedButton(onPressed: onPressed, child: child) // questo pulsante ha background e ombra  
    TextButton(onPressed: onPressed, child: child) // questo pulsante ha solo un testo "pressable"  
    // OutlinedButton(onPressed: onPressed, child: child) // questo pulsante ha solo il contorno  
  ], // <Widget>[]  
) , // Column  
) , // Center  
); // Container  
}
```

Button

```
children: <Widget>[
  Image(
    image: AssetImage('images/registrazione.png'),
    width: 400,
  ), // Image
  MyText('Come andrà l\'esame?'),
  // ElevatedButton(onPressed: onPressed, child: child) // questo pu
  TextButton(onPressed: onPressed, child: child) // questo pulsante
  // OutlinedButton(onPressed: onPressed, child: child) // questo pu
], // <Widget>[]
// Column
```

{required Widget child}

Type: Widget

Create a [TextButton].

Button

```
// ElevatedButton(onPressed: onPressed,  
  TextButton(  
    onPressed: onPressed,  
    child: Text(  
      'Scoprilo')) // questo pulsante  
  // OutlinedButton(onPressed: onPressed,  
], // <Widget>[]  
) // Column
```

Button

```
children: <Widget>[
  Image(
    image: AssetImage('images/registro.jpeg'),
    width: 100,
  ), //
  MyText(
    // E
    Text(
      onPressed: onPressed,
      child: Text(
        'Scoprilo')) // questo pulsante ha solo un testo "pres
    // OutlinedButton(onPressed: onPressed, child: child) // quest
  ], // <Widget>[]
), // Column
```

Type: void Function()? onPressed

Create a [TextButton].

Vuole una “Function as Values” oppure null

Funzione come parametro



Funzione anonima

```
Image.asset(images, AssetImageType) //  
width: 400,  
, // Image  
MyText('Come andrà l\'esame?'),  
// ElevatedButton(onPressed: onPressed, child: child) // questo pulsante ha background e ombra  
TextButton(  
  onPressed: () {}, // funzione anonima  
  child: Text(  
    'Scoprilo')) // questo pulsante ha solo un testo "pressable" // Text // TextButton  
// OutlinedButton(onPressed: onPressed, child: child) // questo pulsante ha solo il contorno  
], // <Widget>[]  
, // Column
```

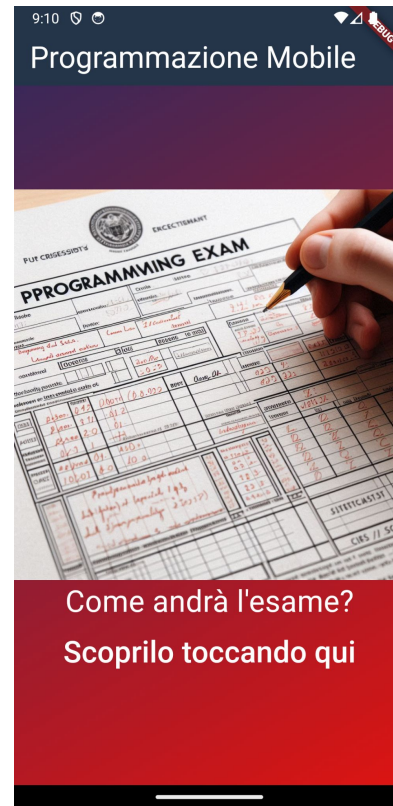
Non è riusabile, non la useremo

Definiamo un metodo nella classe

```
15 void generaVoto() {
16     // ... codice per generare il voto
17 }
18
19 @override
20 Widget build(BuildContext context) {
21     return Container(
22         // definiamo lo stile con Container
23         decoration: BoxDecoration(
24             // definiamo il gradiente
25             gradient: LinearGradient(
26                 // specifica il gradiente
27                 begin: startAlign,
28                 end: endAlign,
29                 colors: [color1, color2],
30             ), // LinearGradient
31         ), // BoxDecoration
32         child: const Center(
33             child: Column(
34                 mainAxisAlignment: MainAxisAlignment.center,
35                 children: <Widget>[
36                     Image(
37                         image: AssetImage('images/registro.jpeg'),
38                         width: 400,
39                     ), // Image
40                     MyText('Come andra l\'esame?'),
41                     // ElevatedButton(onPressed: onPressed, child: child) // questo pulsante ha background e ombra
42                     TextButton(
43                         onPressed: generaVoto,
44                         child: Text(
45                             'Scoprilo') // questo pulsante ha solo un testo "pressable" // Text // TextButton
```

Padding

Potrei definire un padding tra gli elementi oppure...



SizedBox

```
        colors: [color1, color2],
      ), // LinearGradient
    ), // BoxDecoration
    child: Center(
      child: Column(
        mainAxisAlignment: MainAxisAlignment.center,
        children: <Widget>[
          const Image(
            image: AssetImage('images/registro.jpeg'),
            width: 400,
          ), // Image
          const SizedBox(height: 20),
          MyText('Come andrà l\'esame?'),
          const SizedBox(height: 50),
          // ElevatedButton(onPressed: onPressed, child: child) // questo pulsante ha background
          TextButton(
            onPressed: generaVoto,
            child: const MyText(
              'Scopri lo tocando qui') // questo pulsante ha solo un testo "pressabile" // My
          // OutlinedButton(onPressed: onPressed, child: child) // questo pulsante ha solo il con
        ], // <Widget>[]
      ), // Column
    ), // Center
```



Come NON costruire Widget interattivi

```
const startAlign = Alignment.topLeft;
const endAlign = Alignment.bottomRight;

class MyWidget extends StatelessWidget {
  const MyWidget({super.key, required this.color1, required this.color2});
  const MyWidget.appdefault({super.key})
    : color1 = const Color.fromARGB(255, 58, 40, 89),
      color2 = const Color.fromARGB(255, 220, 20, 17);
  final Color color1;
  final Color color2;
  var immagineVoto = 'images/registro.jpeg';
  void generaVoto() {
    immagineVoto = 'images/alto.jpeg';
    print('funziona?')
  }

  @override
  Widget build(BuildContext context) {
    return Container(
      // definiamo lo stile con Container
      decoration: BoxDecoration(
        // definiamo il gradiente
        gradient: LinearGradient(
          // specifica il gradiente
          begin: startAlign,
          end: endAlign,
          colors: [color1, color2],
        ), // LinearGradient
      ), // BoxDecoration
      child: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Image(
              image: AssetImage(immagineVoto),
              width: 400
            ), // Image
          ],
        ),
      ),
    );
  }
}
```


Come NON costruire Widget interattivi

```
34  child: Column(  
35    mainAxisAlignment: MainAxisAlignment.center,  
36    children: <Widget>[  
37      Image(  
38        image: AssetImage(immagineVoto),  
39        width: 400,  
40      ), // Image  
41      const SizedBox(height: 20),  
42      const MyText('Come andrà l\'esame?'),  
43      const SizedBox(height: 50),  
44      // ElevatedButton(onPressed: onPressed, child: child) // questo pulsante ha background e ombra
```

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS Filter (e.g. text, !exclude)

Restarted application in 2.712ms.
D/EGL_emulation(4338): app_time_stats: avg=899892.81ms min=714.53ms max=1799071.25ms count=2
I/flutter (4338): funziona?
D/EGL_emulation(4338): app_time_stats: avg=2027.29ms min=89.24ms max=3965.35ms count=2
I/flutter (4338): funziona?
D/EGL_emulation(4338): app_time_stats: avg=34.59ms min=4.84ms max=484.23ms count=26
D/EGL_emulation(4338): app_time_stats: avg=70.41ms min=6.11ms max=1516.38ms count=28
I/flutter (4338): funziona?



Come NON costruire Widget interattivi

```
import 'package:lezione6/my_text.dart';
```

```
cons Don't invoke 'print' in production code.  
cons Try using a logging framework. dart(avoid_print)
```

```
cons void print(Object? object)
```

```
clas Type: void Function(Object?)
```

```
My  
My dart:core
```

Prints an object to the console.

On the web, `object` is converted to a string and that string is output to the web console using `console.log`.

On native (non-Web) platforms, `object` is converted to a string and that string is terminated by a line feed (`'\n'`, U+000A) and written to `stdout`. On Windows, the terminating line feed, and any line feeds in the string representation of `object`, are output using the Windows line terminator sequence of (`'\r\n'`, U+000D + U+000A).

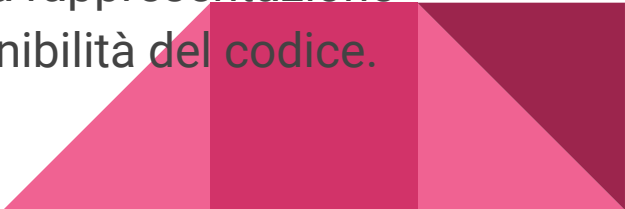
```
print('funziona?');
```



StatefulWidget

Un **StatefulWidget** è un tipo di widget che può mantenere uno stato interno, il quale può cambiare durante il ciclo di vita dell'applicazione. Questo significa che un **StatefulWidget** può essere ricostruito più volte nel corso del tempo, ma mantiene comunque lo stato tra le ricostruzioni.

Quando si lavora con **StatefulWidget**, è importante distinguere tra il **widget StatefulWidget** (che non cambia durante il ciclo di vita dell'applicazione) e l'**oggetto di stato** associato ad esso (che può cambiare). Questa separazione consente di mantenere la logica dello stato separata dalla rappresentazione visuale del widget, migliorando la modularità e la manutenibilità del codice.



StatefulWidget

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

```
class Voto extends StatefulWidget {  
  const Voto({super.key});
```

```
  @override  
  State<Voto> createState() => _VotoState();  
}
```

```
class _VotoState extends State<Voto> {  
  @override  
  Widget build(BuildContext context) {  
    return const Placeholder();  
  }  
} ✨
```

Da StatelessWidget a StatefulWidget

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

class Voto extends StatefulWidget {
  const Voto({super.key});

  @override
  State<Voto> createState() => _VotoState();
}

class _VotoState extends State<Voto> {
  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Image(
          image: AssetImage(immagineVoto),
          width: 400,
        ), // Image
        const SizedBox(height: 20),
        const MyText('Come andrà l'esame?'),
        const SizedBox(height: 50),
        // ElevatedButton(onPressed: onPressed, child: child) // questo pulsante ha background e ombra
        TextButton(
          onPressed: generaVoto,
          child: const MyText(
            'Scoprilo tutto',
          ), // questo pulsante ha solo un testo "pressabile" // MyText // TextButton
        ), // OutlinedButton(onPressed: onPressed, child: child) // questo pulsante ha solo il contorno
      ], // <Widget>[]
    ); // Column
  }
}
```

Da StatelessWidget a StatefulWidget

```
class _VotoState extends State<Voto> {  
    var immagineVoto = 'images/registro.jpeg';  
    void generaVoto() {  
        immagineVoto = 'images/alto.jpeg';  
    }  
  
    @override  
    Widget build(BuildContext context) {  
        return Column(  
            mainAxisAlignment: MainAxisAlignment.center,  
            children: <Widget>[  
                Image(  

```

Da StatelessWidget a StatefulWidget

Quindi ora funziona? No!

```
import 'package:flutter/material.dart';
import 'package:lezione6/my_text.dart';

class Voto extends StatefulWidget {
  const Voto({super.key});

  @override
  State<Voto> createState() => _VotoState();
}

class _VotoState extends State<Voto> {
  var immagineVoto = 'images/registro.jpeg';
  void generaVoto() {
    immagineVoto = 'images/alto.jpeg';
  }

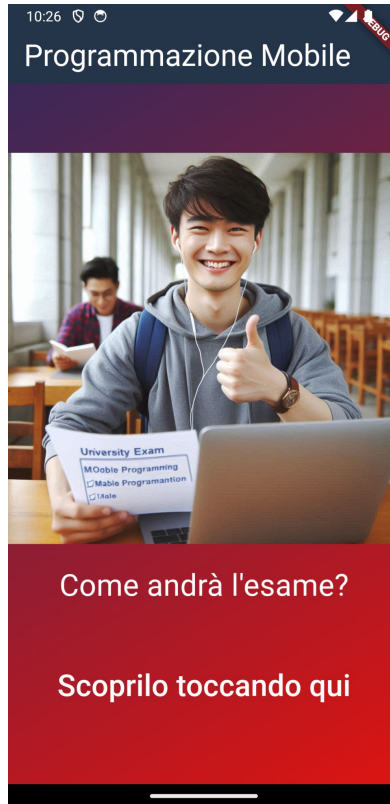
  @override
  Widget build(BuildContext context) {
    return Column(
      mainAxisAlignment: MainAxisAlignment.center,
      children: <Widget>[
        Image(
          image: AssetImage(immagineVoto),
          width: 400,
        ), // Image
        const SizedBox(height: 20),
        const MyText('Come andrà l\'esame?'),
        const SizedBox(height: 50),
        // ElevatedButton(onPressed: onPressed, child: child) // questo pulsante ha background e ombra
        TextButton(
          onPressed: generaVoto,
          child: const MyText(
            'Scoprilo toccando qui') // questo pulsante ha solo un testo "pressable" // MyText // TextButton
        ), // OutlinedButton(onPressed: onPressed, child: child) // questo pulsante ha solo il contorno
      ], // <Widget>[]
    ); // Column
  }
}
```

setState

setState serve a rilanciare il metodo build

```
10
11 class _VotoState extends State<Voto> {
12     var immagineVoto = 'images/registro.jpeg';
13     void generaVoto() {
14         setState(() {
15             immagineVoto = 'images/alto.jpeg';
16         });
17     }
18
19     @override
20     Widget build(BuildContext context) {
21         return Column(
```


setState



Una funzione Random e un po' di codice per migliorare il tutto

```
import 'package:flutter/material.dart';
import 'package:lezione6/my_text.dart';
import 'dart:math';

final randomizer = Random();

// se vuoi scrivere decentemente il codice, crea una classe Range
// e usa la funzione generaVoto() commentata
// class Range {
//   final int start;
//   final int end;

//   Range(this.start, this.end);

//   bool includes(int value) => start <= value && value <= end;
// }

class Voto extends StatefulWidget {
  const Voto({super.key});

  @override
  State<Voto> createState() => _VotoState();
}

class _VotoState extends State<Voto> {
  var immagineVoto = 'images/registro.jpeg';
  var currentGrade = 0;
  void generaVoto() {
    setState(() {
      currentGrade =
        randomizer.nextInt(21) + 11; // genera un numero tra 11 e 31
      if (currentGrade >= 11 && currentGrade <= 14) {
        immagineVoto = 'images/bocciato.jpeg';
      } else if (currentGrade >= 15 && currentGrade <= 17) {
        immagineVoto = 'images/perpoco.jpeg';
      } else if (currentGrade >= 18 && currentGrade <= 26) {
        immagineVoto = 'images/medio.jpeg';
      } else if (currentGrade >= 27 && currentGrade <= 31) {
        immagineVoto = 'images/alto.jpeg';
      }
    });
  }
  // void generaVoto() {
```

Suggerimento per il progetto

Strapi

The image shows a screenshot of a web browser displaying the Strapi website. The website has a dark blue background with a grid pattern. The main heading is "Manage any content. Anywhere." followed by the text "The leading open-source headless CMS. 100% JavaScript / TypeScript and fully customizable." Below this is a code snippet: `npx create-strapi-app@latest my-project`. There are two buttons: "Get Started" and "Try the live demo".

Below the website screenshot, there is a preview of the Strapi Content-type Builder interface. It shows a "Restaurant" content type with a "Media" field. The field is configured with "Multiple Media" selected. To the right, there is a JSON snippet representing the content type schema:

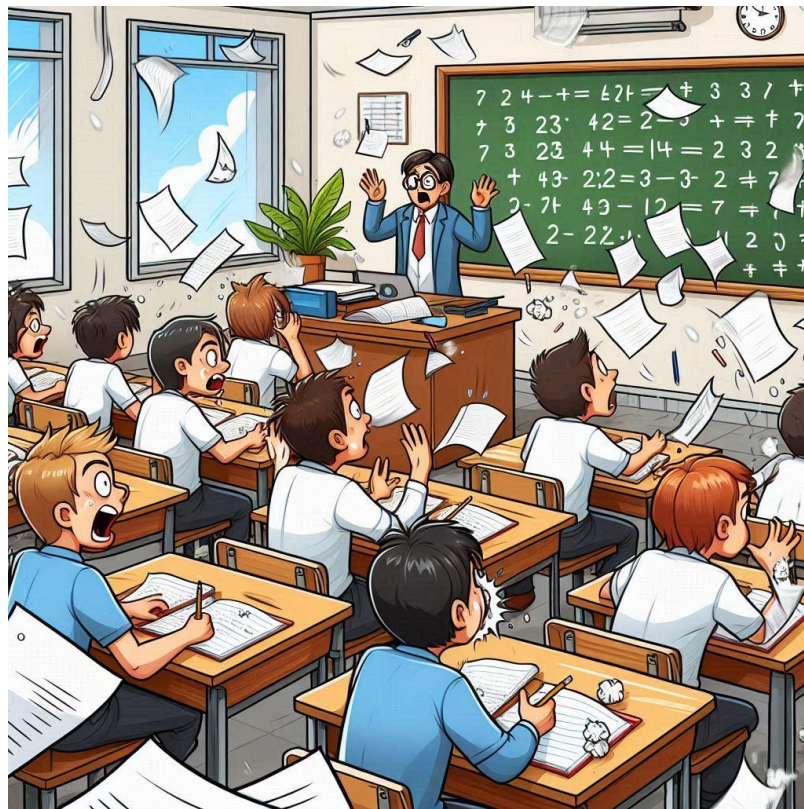
```
1 {
2   "kind": "collectionType",
3   "collectionName": "restaurants",
4   "info": {
5     "singularName": "restaurant",
6     "pluralName": "restaurants",
7     "displayName": "Restaurant",
8     "name": "Restaurant"
9   },
10  "options": {
11    "draftAndPublish": true
12  },
13  "pluginOptions": {},
14  "attributes": {
15    "name": {
16      "type": "string"
17    }
18  }
19 }
20
21
22
23
24
25
26
27
```

At the bottom of the screenshot, there is a footer that says "Get early access. Try the Strapi 5 beta today!" and a close button (X).



Esercizio

Esercizio: Creazione di una UI



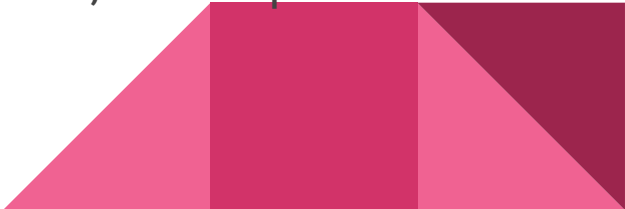
Esercizio: Creazione di una UI

L'obiettivo di questo esercizio è quello di verificare le conoscenze acquisite fino a questo punto del corso.

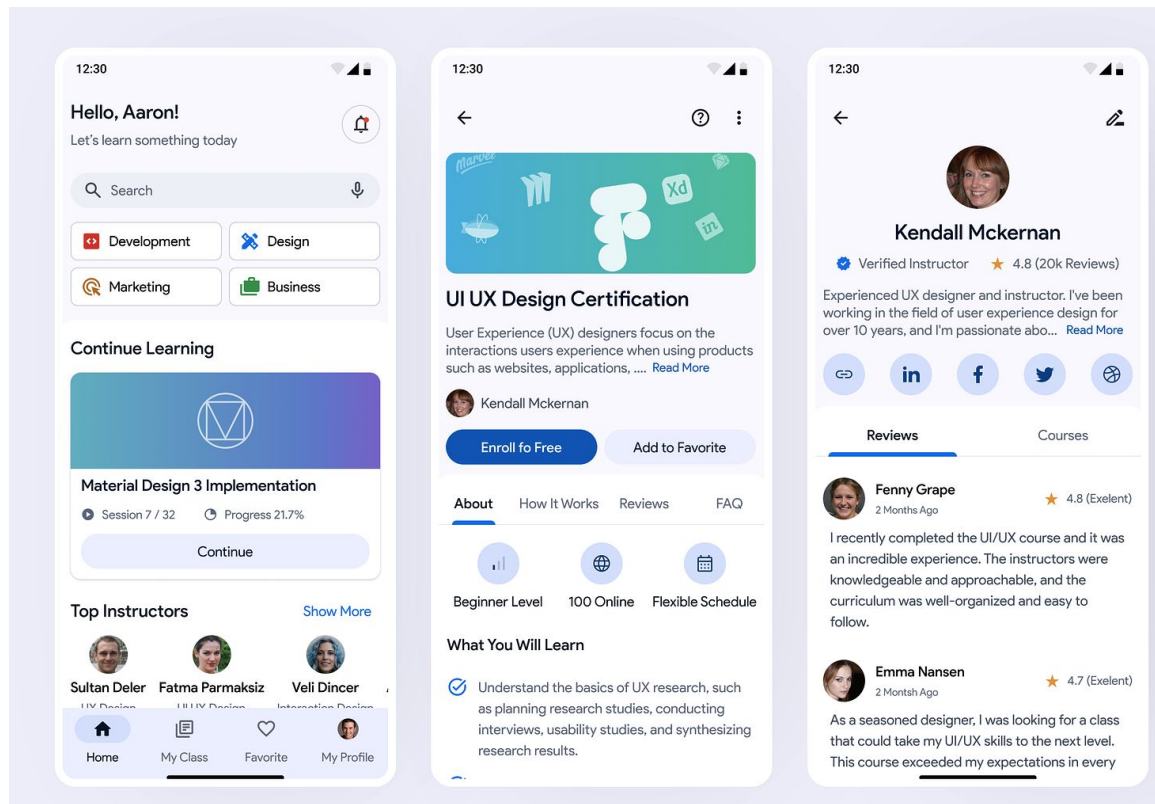
Allo studente è chiesto di provare a costruire le interfacce fornite nella slide successiva dalla prima (quella più a sinistra) all'ultima (quella di destra).

Non è richiesto di implementare nessun sistema di navigazione funzionante poiché non sono ancora stati trattati durante le lezioni.


È richiesto di utilizzare solo StatelessWidget, tutti gli elementi, anche quelli interattivi, non devono avere nessuna funzione collegata.



Esercizio: Creazione di una UI



Esercizio: Creazione di una UI

- Le schermate comprendono elementi visti a lezione ed elementi nuovi. Per questo è importante anche sfruttare la documentazione, come ad esempio <https://docs.flutter.dev/ui/widgets/material>
 - Non mi interessa che la schermata sia perfettamente uguale, mi interessa l'approccio, ad esempio fare riuso del codice quando possibile
 - Meglio una sola schermata, ma fatta con attenzione e non tutte e 3 le schermate fatte male
 - Per piacere lavorate in maniera individuale
 - Durante la prossima lezione frontale vi mostrerò una possibile soluzione
- 

Esercizio: Creazione di una UI

Consegna:

- Un gist GitHub per ogni singola schermata realizzata. Eventuali classi andranno tutte inserite in un unico file main.dart;
 - **Inviare tutto a nicola.noviello@unimol.it utilizzando come oggetto della e-mail “Lezione6: Nome Cognome Matricola”;**
 - Da completare nelle ore di lezione residue nella giornata dopo aver completato la lezione teorica;
 - Non obbligatorio, ma valutato positivamente ai fini dell’esame.
- 