

Internet of Things

Lab1

Introduction to Flutter

Grading scheme

- ▶ Final examination: 40%
- ▶ Lab - 30% (average of lab activity, hw) - min. grade 5 required
- ▶ Project - 30% - min. grade 5 required

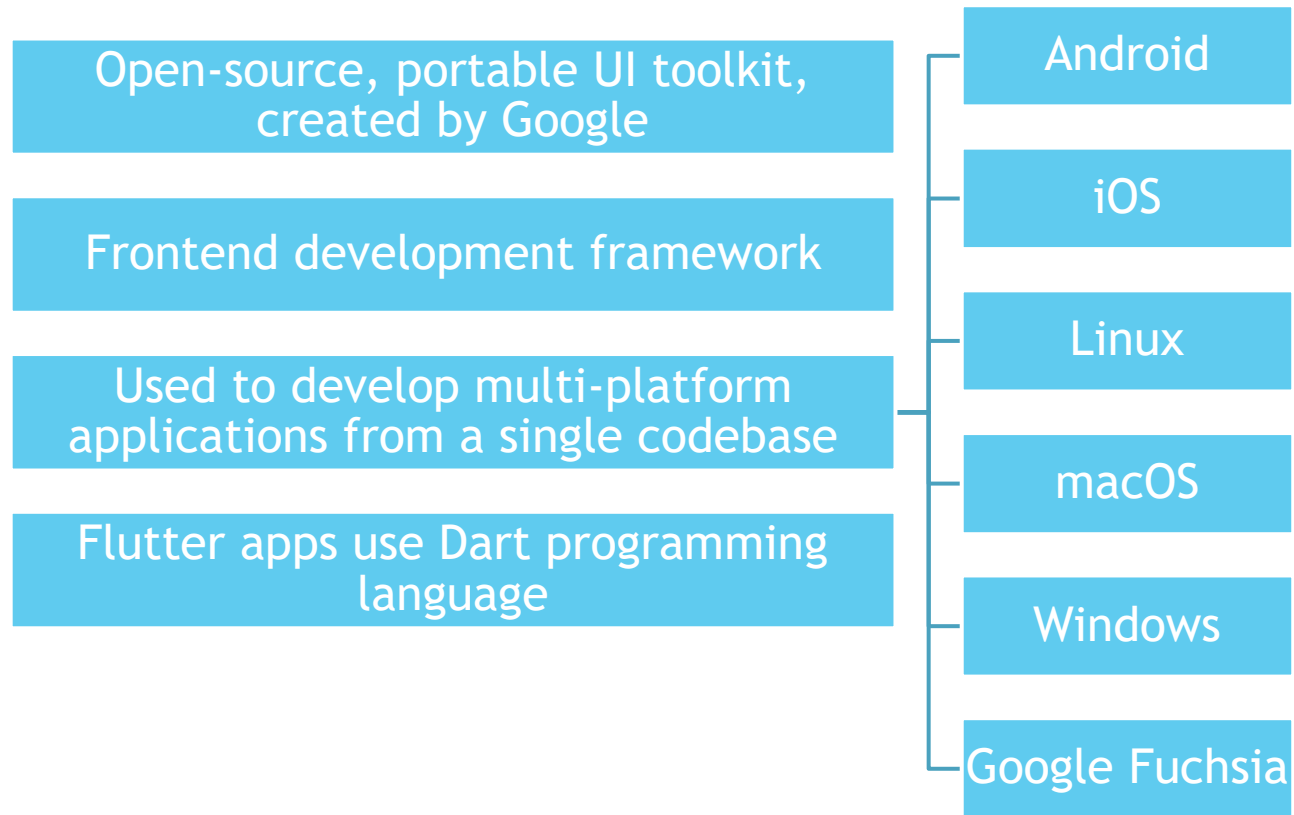
Grades: <https://docs.google.com/spreadsheets/d/1h-MISNrNQCNIUIkFUzWtNb9QOZrV-VnwZpx6nLksw8o/edit#gid=0>

If you have any issues:

ciungan_alexandra@yahoo.ro

vlad.garmacea@gmail.com

Flutter



Installation

- ▶ For the installation, please, follow the steps inside this tutorial:
<https://docs.flutter.dev/get-started/install>
- ▶ You will need to install it for at least Web
- ▶ Please, be aware, that, for Windows, additional paths may need to be set up (for example, in system variables path
C:\Windows\System32\WindowsPowerShell\v1.0 and in user variables path
C:\Program Files\Git\bin\git.exe ; C:\Program Files\Git\cmd ;
C:\Windows\System32)

Widgets

Central class hierarchy in the Flutter framework

All the instances of a widget are immutable

Describe what their view should look like given their current configuration and state

Types:

- Stateful widgets
- Stateless widgets

Basic widgets:

- Text
- Row, Column
- Stack
- Container

Stateless vs Stateful Widgets

State = information that can be read synchronously when the widget is built and might change during the lifetime of the widget

Stateless

- ▶ Does not require a mutable state
- ▶ They are used when the part of the UI does not depend on other parts

```
1 import 'package:flutter/material.dart';
2
3 Run | Debug | Profile
4 void main() {
5   runApp(const MyApp());
6 }
7
8 class MyApp extends StatelessWidget {
9   const MyApp({super.key});
10
11   // This widget is the root of your application.
12   @override
13   Widget build(BuildContext context) {
14     return MaterialApp(
15       title: 'Flutter Demo',
16       theme: ThemeData(
17         primarySwatch: Colors.blue,
18       ), // ThemeData
19       home: const MyHomePage(title: 'Flutter Demo Home Page'),
20     ); // MaterialApp
21   }
```

Stateful

- ▶ Has a mutable state
- ▶ They are used when the UI might change multiple times over time

```
23 class MyHomePage extends StatefulWidget {
24   const MyHomePage({super.key, required this.title});
25   final String title;
26
27   @override
28   State<MyHomePage> createState() => _MyHomePageState();
29 }
30
31 class _MyHomePageState extends State<MyHomePage> {
32   int _counter = 0;
33
34   void _incrementCounter() {
35     setState(() {
36       _counter++;
37     });
38   }
```

```
40 @override
41 Widget build(BuildContext context) {
42   return Scaffold(
43     appBar: AppBar(
44       title: Text(widget.title),
45     ), // AppBar
46     body: Center(
47       child: Column(
48         mainAxisAlignment: MainAxisAlignment.center,
49         children: <Widget>[
50           const Text(
51             'You have pushed the button this many times:',
52           ), // Text
53           Text(
54             '$_counter',
55             style: Theme.of(context).textTheme.headline4,
56           ), // Text
57         ], // <Widget>[]
58       ), // Column
59     ), // Center
60     floatingActionButton: FloatingActionButton(
61       onPressed: _incrementCounter,
62       tooltip: 'Increment',
63       child: const Icon(Icons.add),
64     ), // FloatingActionButton
65   ); // Scaffold
66 }
67 }
```

Basic widgets

Text: Lets you create a run of styled text within your application

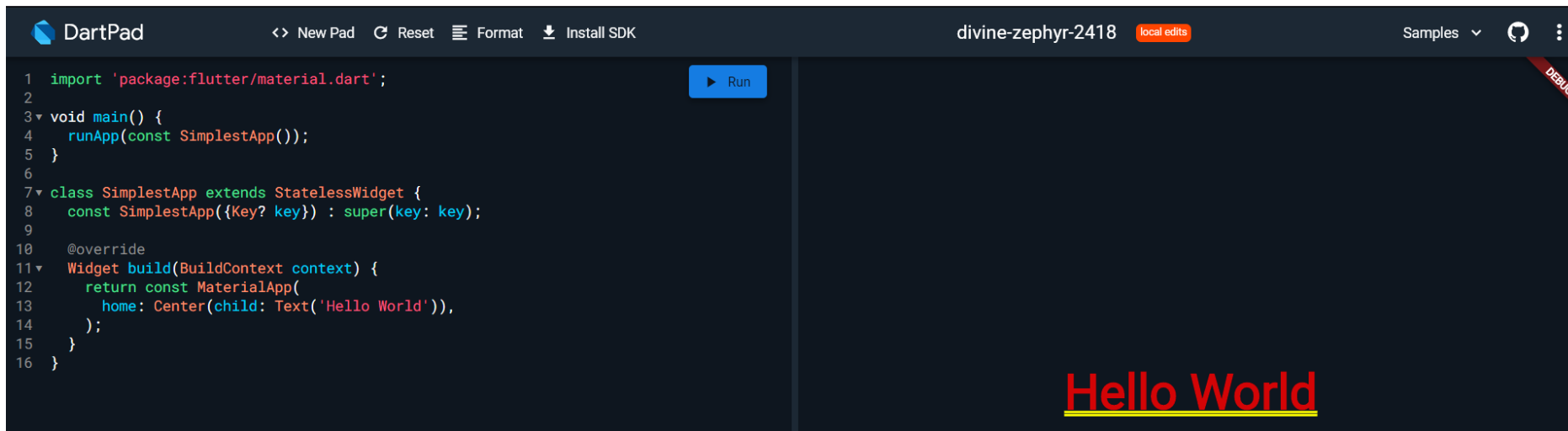
Row, Column: Flex widgets that let you create flexible layouts in both the horizontal (Row) and vertical (Column) directions

Stack: Instead of being linearly oriented (either horizontally or vertically), a Stack widget lets you place widgets on top of each other in paint order. You can then use the Positioned widget on children of a Stack to position them relative to the top, right, bottom, or left edge of the stack.

Container: Lets you create a rectangular visual element. A container can be decorated with a BoxDecoration, such as a background, a border, or a shadow. A Container can also have margins, padding, and constraints applied to its size.

Text Widget

- ▶ Import the package necessary for building flutter apps
- ▶ Main function->runApp function is called by passing a widget as a parameter (making it the root of the app)
- ▶ SimplestApp - user defined class that inherits StatelessWidget (also has a constructor with 1 optional parameter - key)
- ▶ @override - marks an instance member as overriding a superclass member with the same name.
- ▶ build - method responsible for creating the components
- ▶ MaterialApp - predefined class used to access all the other components of Flutter SDK
- ▶ Center - center the content both vertically and horizontally

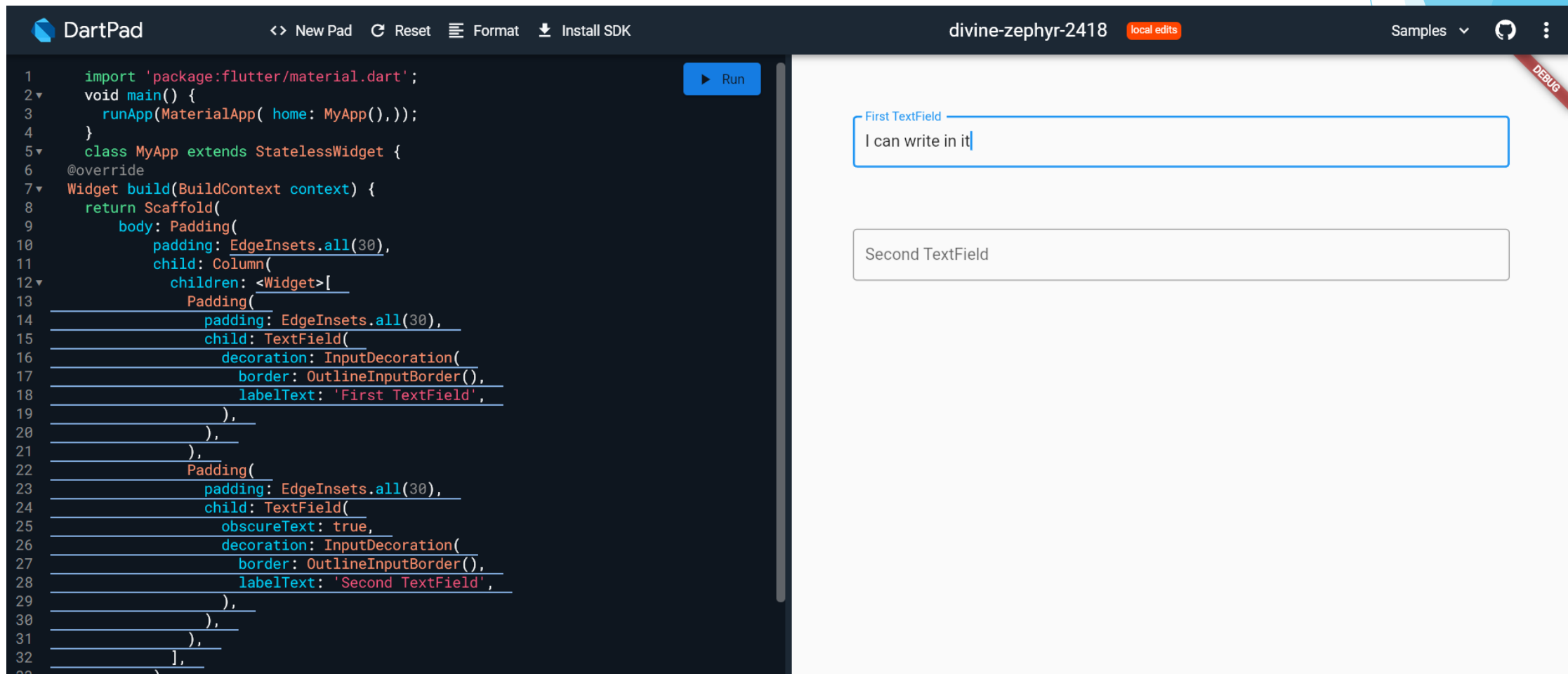


```
1 import 'package:flutter/material.dart';
2
3 void main() {
4   runApp(const SimplestApp());
5 }
6
7 class SimplestApp extends StatelessWidget {
8   const SimplestApp({Key? key}) : super(key: key);
9
10  @override
11  Widget build(BuildContext context) {
12    return const MaterialApp(
13      home: Center(child: Text('Hello World')),
14    );
15  }
16 }
```

Hello World

TextField Widget

- ▶ [Scaffold](#) - predefined class used under MaterialApp, giving basic functionalities
- ▶ [Padding](#) - widget used to inset its child by the given padding
- ▶ Column - aligns the children by column



The screenshot displays the DartPad web IDE interface. The left pane shows the Dart source code for a Flutter application. The code imports the necessary Flutter packages, defines a `main` function to run the app, and creates a `MyApp` class that extends `StatelessWidget`. The `build` method of `MyApp` returns a `Scaffold` widget. Inside the `Scaffold`, a `Padding` widget is used to wrap a `Column` widget. The `Column` contains two `TextField` widgets. The first `TextField` has a label 'First TextField' and the text 'I can write in it'. The second `TextField` has a label 'Second TextField' and its text is obscured (password field). The right pane shows the rendered UI of the application, which matches the code on the left. The top bar of the IDE shows the username 'divine-zephyr-2418', a 'local edits' indicator, and a 'DEBUG' button. The bottom of the code editor has a 'Run' button.

```
1 import 'package:flutter/material.dart';
2 void main() {
3   runApp(MaterialApp( home: MyApp(),));
4 }
5 class MyApp extends StatelessWidget {
6   @override
7   Widget build(BuildContext context) {
8     return Scaffold(
9       body: Padding(
10        padding: EdgeInsets.all(30),
11        child: Column(
12          children: <Widget>[
13            Padding(
14              padding: EdgeInsets.all(30),
15              child: TextField(
16                decoration: InputDecoration(
17                  border: OutlineInputBorder(),
18                  labelText: 'First TextField',
19                ),
20              ),
21            ),
22            Padding(
23              padding: EdgeInsets.all(30),
24              child: TextField(
25                obscureText: true,
26                decoration: InputDecoration(
27                  border: OutlineInputBorder(),
28                  labelText: 'Second TextField',
29                ),
30              ),
31            ),
32          ],
33        ),
34      ),
35    );
36  }
```

Row, Column Widget Container Widget

- ▶ [AppBar](#) - used to create a fixed widget at the top of the screen
- ▶ [Container](#) - used to surround a child with padding, then add additional constraints to it
- ▶ [BoxDecoration](#) - adds decoration to a container

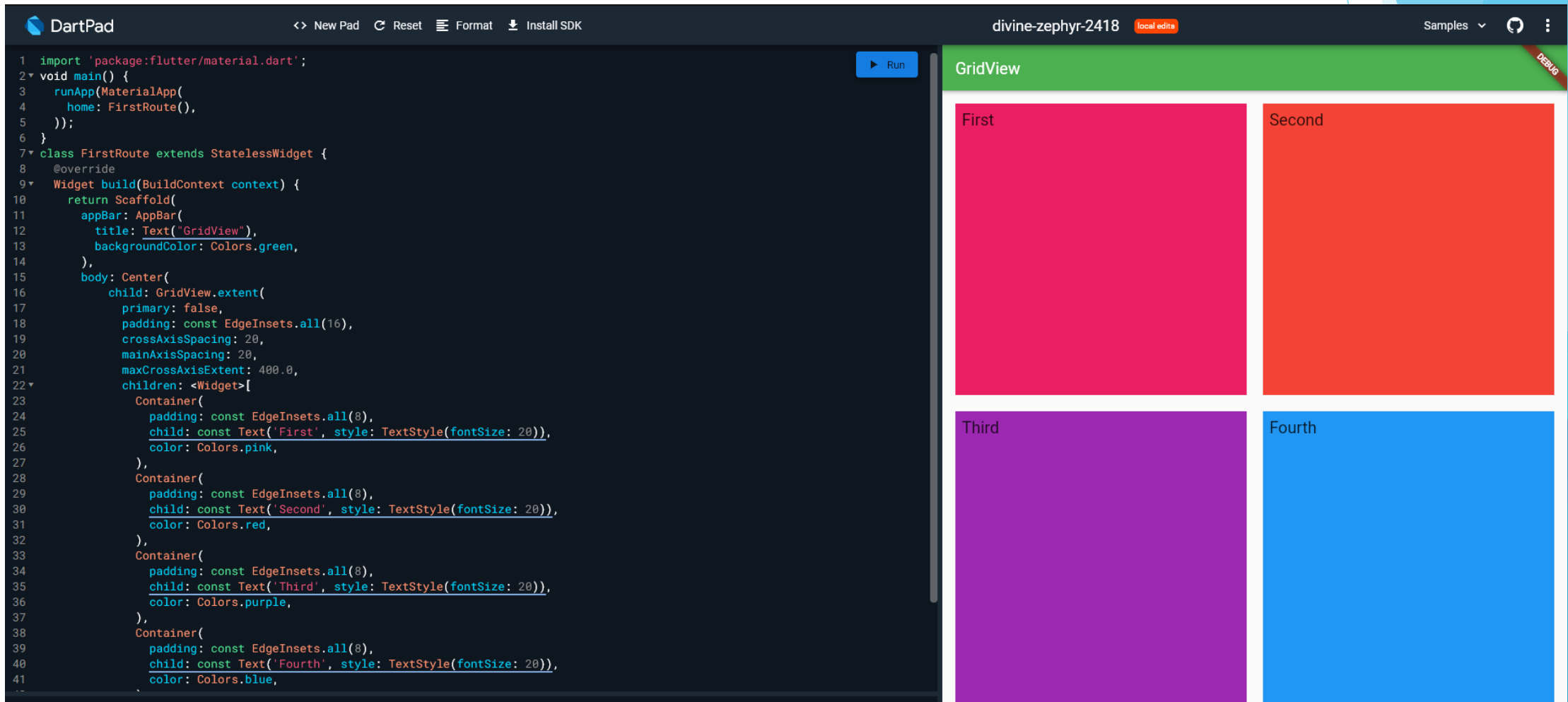
The screenshot displays the DartPad web editor interface. On the left, the code editor shows the following Dart code:

```
1 import 'package:flutter/material.dart';
2 void main() {
3   runApp(MaterialApp( home: MyApp(),));
4 }
5 class MyApp extends StatelessWidget{
6   @override
7   Widget build(BuildContext context) {
8     return Scaffold(
9       appBar: AppBar(
10        title: Text("Row Example"),
11      ),
12      body: Row(
13        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
14        children:<Widget>[
15          Container(
16            margin: EdgeInsets.all(10.0),
17            padding: EdgeInsets.all(10.0),
18            decoration:BoxDecoration(
19              borderRadius:BorderRadius.circular(10),
20              color:Colors.pink
21            ),
22            child: Text("Row1",style: TextStyle(color:Colors.blueAccent,fontSize:20,)),
23          ),
24          Container(
25            margin: EdgeInsets.all(10.0),
26            padding: EdgeInsets.all(10.0),
27            decoration:BoxDecoration(
28              borderRadius:BorderRadius.circular(10),
29              color:Colors.red
30            ),
31            child: Text("Row2",style: TextStyle(color:Colors.white,fontSize:20,)),
32          ),
33          Container(
34            margin: EdgeInsets.all(10.0),
35            padding: EdgeInsets.all(10.0),
36            decoration:BoxDecoration(
37              borderRadius:BorderRadius.circular(10),
38              color:Colors.purple
39            ),
40            child: Text("Row3",style: TextStyle(color:Colors.black,fontSize:20,)),
41          )
42        ],
43      ),
44    );
45  }
46 }
```

On the right, the visual output titled "Row Example" shows three rounded rectangular boxes arranged horizontally. The first box is pink with the text "Row1" in blue. The second box is red with the text "Row2" in white. The third box is purple with the text "Row3" in black. A "Run" button is visible above the code editor, and a "DEBUG" button is in the top right corner of the output area.

GridView Widget

- ▶ GridView.extent - creates a scrollable 2D array of widgets
- ▶ EdgeInsets - specify offsets in terms of visual edges, left, top, right and bottom



The screenshot displays the DartPad interface with a Flutter application. The code on the left defines a `GridView.extent` widget within a `Scaffold`. The grid contains four children, each a `Container` with a specific color and text: 'First' (pink), 'Second' (red), 'Third' (purple), and 'Fourth' (blue). The `GridView.extent` is configured with `primary: false`, `padding: EdgeInsets.all(16)`, `crossAxisSpacing: 20`, `mainAxisSpacing: 20`, and `maxCrossAxisExtent: 400.0`. The right side of the image shows the rendered application, titled 'GridView', which displays these four colored boxes in a 2x2 grid. The top bar of the DartPad shows the username 'divine-zephyr-2418' and a 'local edit' button. The bottom bar of the application shows a 'Run' button.

```
1 import 'package:flutter/material.dart';
2 void main() {
3   runApp(MaterialApp(
4     home: FirstRoute(),
5   ));
6 }
7 class FirstRoute extends StatelessWidget {
8   @override
9   Widget build(BuildContext context) {
10    return Scaffold(
11      appBar: AppBar(
12        title: Text("GridView"),
13        backgroundColor: Colors.green,
14      ),
15      body: Center(
16        child: GridView.extent(
17          primary: false,
18          padding: const EdgeInsets.all(16),
19          crossAxisSpacing: 20,
20          mainAxisSpacing: 20,
21          maxCrossAxisExtent: 400.0,
22          children: <Widget>[
23            Container(
24              padding: const EdgeInsets.all(8),
25              child: const Text('First', style: TextStyle(fontSize: 20)),
26              color: Colors.pink,
27            ),
28            Container(
29              padding: const EdgeInsets.all(8),
30              child: const Text('Second', style: TextStyle(fontSize: 20)),
31              color: Colors.red,
32            ),
33            Container(
34              padding: const EdgeInsets.all(8),
35              child: const Text('Third', style: TextStyle(fontSize: 20)),
36              color: Colors.purple,
37            ),
38            Container(
39              padding: const EdgeInsets.all(8),
40              child: const Text('Fourth', style: TextStyle(fontSize: 20)),
41              color: Colors.blue,
42            ),
43          ],
44        ),
45      ),
46    );
47  }
48 }
```

Stack Widget

- ▶ [Positioned](#) - controls where a child is positioned
- ▶ [Align](#) - used to set the alignment of the child (in this case, the text in the top widget)


DartPad

<> New Pad ↺ Reset ⚙ Format ⬇ Install SDK

▶ Run

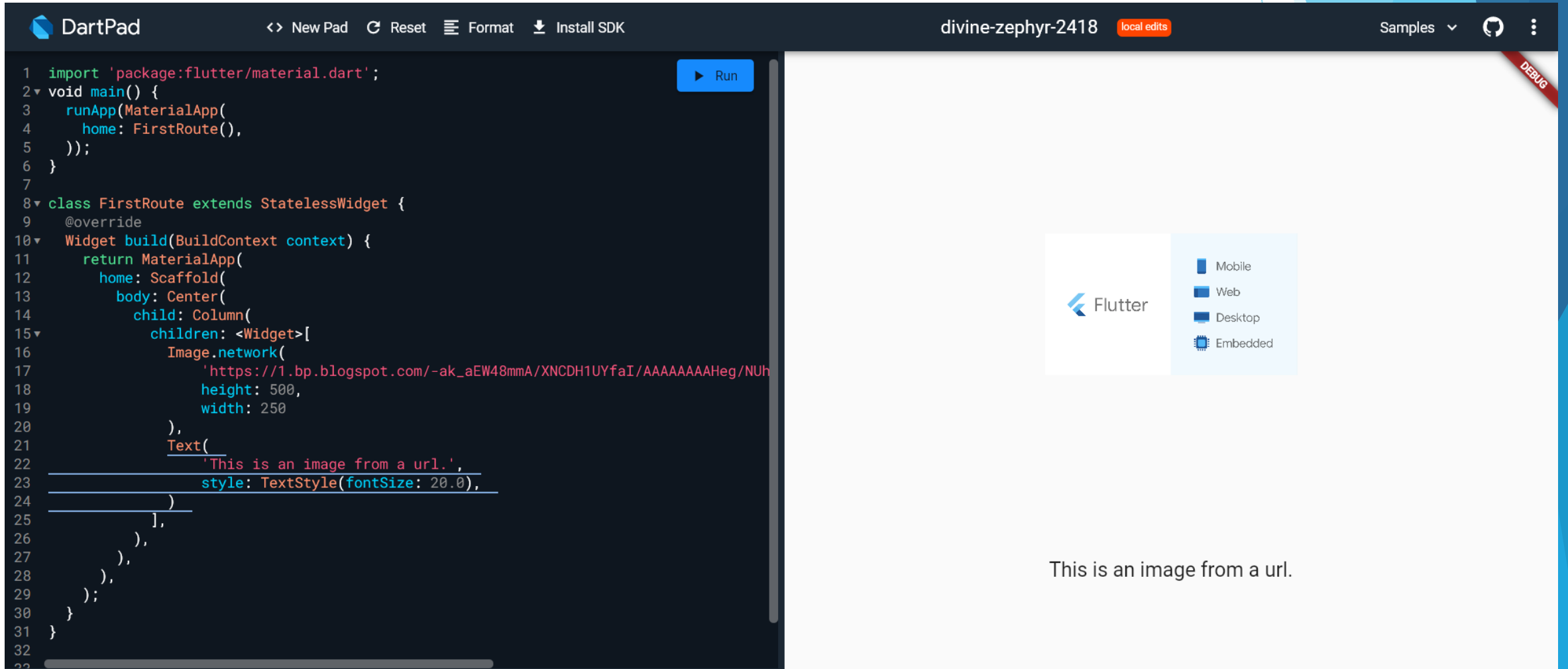
```
1 import 'package:flutter/material.dart';
2 void main() {
3   runApp(MaterialApp( home: MyApp()));
4 }
5 class MyApp extends StatelessWidget {
6   @override
7   Widget build(BuildContext context) {
8     return MaterialApp(
9       home: Scaffold(
10        appBar: AppBar(
11          title: Text("Stack Widget"),
12        ),
13        body: Center(
14          child: Stack(
15            fit: StackFit.passthrough,
16            children: <Widget>[
17              Container(
18                height: 300,
19                width: 400,
20                color: Colors.purple,
21                child: const Align(
22                  alignment: Alignment(0.2,0.6),
23                  child: Text('Top Widget',
24                    style: TextStyle(color: Colors.white, fontSize: 20),
25                  ),
26                ),
27              Positioned(
28                top: 30,
29                right: 20,
30                child: Container(
31                  height: 100,
32                  width: 150,
33                  color: Colors.blue,
34                  child: Center(
35                    child: Text('Middle Widget',
36                      style: TextStyle(color: Colors.white, fontSize: 20),
37                    ),
38                  ),
39                ),
40              Positioned(
41                top: 100,
42                left: 240,
43                child: Container(
44                  height: 100,
45                  width: 150,
46                  color: Colors.pink,
47                  child: Center(
48                    child: Text('Bottom Widget',
49                      style: TextStyle(color: Colors.white, fontSize: 20),
50                    ),
51                  ),
52                ),
53              ),
54            ],
55          ),
56        ),
57      ),
58    );
59  }
60 }
```

Stack Widget



Adding images

- ▶ [Image.network](#) - widget used to display an image from a link



The screenshot shows the DartPad web IDE interface. The top bar includes the DartPad logo, navigation links (New Pad, Reset, Format, Install SDK), the user name 'divine-zephyr-2418', a 'local edits' indicator, and a 'Samples' dropdown. The code editor on the left contains the following Dart code:

```
1 import 'package:flutter/material.dart';
2 void main() {
3   runApp(MaterialApp(
4     home: FirstRoute(),
5   ));
6 }
7
8 class FirstRoute extends StatelessWidget {
9   @override
10  Widget build(BuildContext context) {
11    return MaterialApp(
12      home: Scaffold(
13        body: Center(
14          child: Column(
15            children: <Widget>[
16              Image.network(
17                'https://1.bp.blogspot.com/-ak_aEW48mmA/XNCDH1UYfaI/AAAAAAAAHeg/NUh
18                height: 500,
19                width: 250
20            ),
21            Text(
22              'This is an image from a url.',
23              style: TextStyle(fontSize: 20.0),
24            ),
25          ],
26        ),
27      ),
28    );
29  };
30 }
31 }
```

A 'Run' button is located to the right of the code editor. The output area on the right shows the Flutter logo and a platform selection menu with options: Mobile, Web, Desktop, and Embedded. Below this, the text 'This is an image from a url.' is displayed. A red 'DEBUG' banner is visible in the top right corner of the output area.

Navigation

- ▶ ElevatedButton - used to create personalized buttons
- ▶ onPressed() - adds functionality if the button is pressed
- ▶ Navigator - widget used to navigate through pages
- ▶ push() - method used to push a route to the navigator
- ▶ pop() - method used to pop the top-most route

```
1 import 'package:flutter/material.dart';
2 void main() {
3   runApp(MaterialApp(
4     home: FirstRoute(),
5   ));
6 }
7
8 class FirstRoute extends StatelessWidget {
9   @override
10  Widget build(BuildContext context) {
11    return Scaffold(
12      appBar: AppBar(
13        title: Text('First Screen'),
14      ),
15      body: Center(
16        child: ElevatedButton(
17          child: Text('Click Here'),
18          style: ElevatedButton.styleFrom(
19            backgroundColor: Color.fromARGB(225,219,0,115)),
20          onPressed: () {
21            Navigator.push(
22              context,
23              MaterialPageRoute(builder: (context) => SecondRoute()),
24            );
25          },
26        ),
27      ),
28    );
29  }
30 }
```

First Screen

Click Here

```
33
34
35 class SecondRoute extends StatelessWidget {
36   @override
37   Widget build(BuildContext context) {
38     return Scaffold(
39       appBar: AppBar(
40         title: Text("Second Screen"),
41       ),
42       body: Center(
43         child: ElevatedButton(
44           style: ElevatedButton.styleFrom(
45             backgroundColor: Color.fromARGB(255,0,255,255)),
46           onPressed: () {
47             Navigator.pop(context);
48           },
49           child: Text('Go back'),
50         ),
51       ),
52     );
53   }
54 }
55
56
57
58
```

← Second Screen

Go back

Exercises

► Go to dartpad.dev and create 2 navigable pages:

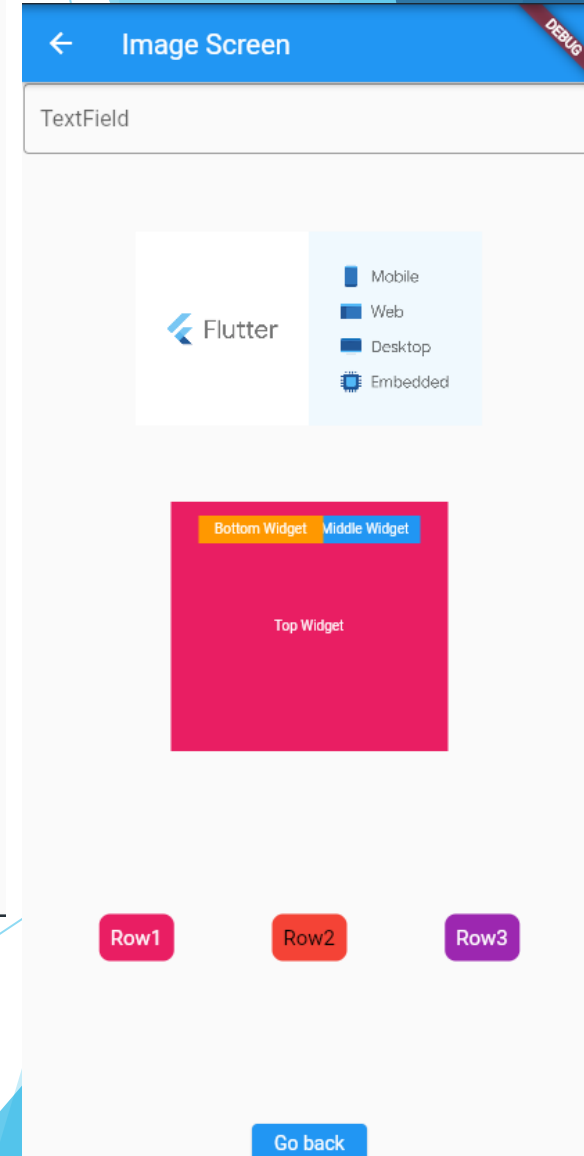
► Home page has:

- At least 1 Text Widget
- A grid with at least 6 different elements
- A button to go to the Image page

*(Hint: you may put these elements inside a Column; you can put GridView as a child inside a new [Expanded](#) class because you may need to constrain the height of GridView in order to fit inside Column)

► Image page has:

- At least 1 TextField Widget
- An image
- A stack of at least 3 elements
- A row with at least 3 elements
- Button to return to Home page



*The images are just an example; you can create any pages you wish as long as you follow the requirements

Exercises- example how to start

```
1 import 'package:flutter/material.dart';
2 void main() {
3   runApp(MaterialApp(
4     home: FirstRoute(),
5   ));
6 }
7
8 class FirstRoute extends StatelessWidget {
9   @override
10  Widget build(BuildContext context) {
11    return Scaffold(
12      appBar: AppBar(
13        title: Text('Home Screen'),
14      ),
15      body: Center(
16        child: Column(
17          children: <Widget>[
18            Element1-Text,
19            Element2-Expanded (child: GridView, children:Container),
20            Element3-ElevatedButton
21          ],
22        ),
23      ),
24    );
25  }
26 }
27
28 );
29 }
30 }
31 }
```

```
32 class SecondRoute extends StatelessWidget {
33   @override
34   Widget build(BuildContext context) {
35     return Scaffold(
36       appBar: AppBar(
37         title: Text("Image Screen"),
38       ),
39       body: Center(
40         child: Column(
41           children: <Widget>[
42             Element1 - Padding (TextField),
43             Element2 - Image,
44             Element3 - Stack,
45             Element4 - Expanded (child: Row, children: Container)
46             Element5 - Elevated Button
47           ],
48         ),
49       ),
50     );
51   }
52 }
53
54 }
55 }
```


Homework

- ▶ Complete the installation for at least Web (All the examples in the next labs are for web, Flutter SDK version **3.0.0**. If you use a different version, then you might encounter issues with dependencies provided in the lab)
- ▶ Create a new application, that we will use in the next labs:
 - Create a Log in page (email and password)
 - Create a Registration page (at least email and password)
 - Create a “Products” page that will display 2 products with image and name
 - Conditions:
 - ❑ The pages should be navigable
 - ❑ Log in and Registration pages should contain at least 2 TextField and 2 buttons
- ▶ All hw will be checked during the next lab