Design And Implementation Of Mobile Applications

Laboratory

Christian Rossi

Academic Year 2024-2025

Abstract

The course is organized into five distinct parts.

The first part sets the stage by framing the problem and exploring the numerous opportunities that mobile devices present. It also provides a brief overview of various alternatives and competing solutions in the field.

In the second part, we focus on mobile application design, aiming to identify guidelines and recurring patterns that can facilitate the design process and contribute to producing high-quality solutions.

The third part introduces key innovations and features of important frameworks, specifically Flutter and React Native, which are essential for developing cross-platform applications.

The fourth part delves into Android development, covering how to create applications for a range of Android devices, including phones, tablets, watches, and TVs, using Kotlin and Jetpack Compose.

Finally, the fifth part addresses the development of applications for iOS-based devices, utilizing Swift and SwiftUI to create efficient and effective mobile applications.

Contents

CHAPTER 1

Flutter

1.1 Dart basics

The `main.dart` file contains the following code:

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

void main() {
    runApp(const MyApp());
}
```

In this case we have that when the application is started the main will call an immutable class `MyApp` (that becomes a Widget) that shows something on the screen.

An example of called class could be:

```
class MyApp extends StatelessWidget {
    const MyApp({super.key});

    @override
    Widget build(BuildContext context) {
        return MaterialApp();
    }
}
```

In this case we have a constructor with the same name of the class, and an override method taht returns a Widget that becomes the root of our application.

The `MaterialApp` needs a title and a home as follows:

```
MaterialApp (
    title: "Hello World!",
    home: Scaffold(
        appBar: AppBar(
            title: const Text("Hello World!"),
            style: TextStyle(color: Colors.red),
        )
    ),
)
```

In this case we use a `Scaffold`, that is the biggest element on the screen, and we set the top bar with the string `Hello World!` coloured in red. `Scaffold` also has an argument called `body` to inser directly a widget inside it.

To center an element we may use `Center`:

```
Center(
    child: HelloWorld(),
)
```

The constructor may have null elemebts, in this case we add a question mark after the variable type. An example of constructor:

```
class HelloWorlds extends StatelessWidget {
    const HelloWorld({Key? key}) : siper(key: key);
}

class HelloWorldPlus extends StatelessWidget {
    final int number;
    final Color color;

    const HelloWorldPlus(this.number, this.color);
}
```

To add a variable in a string we can simply use $ befor the variable name:

```
@override
Widget build(BuildContext context) {
    return Text("Hello World! $number",)
}
```