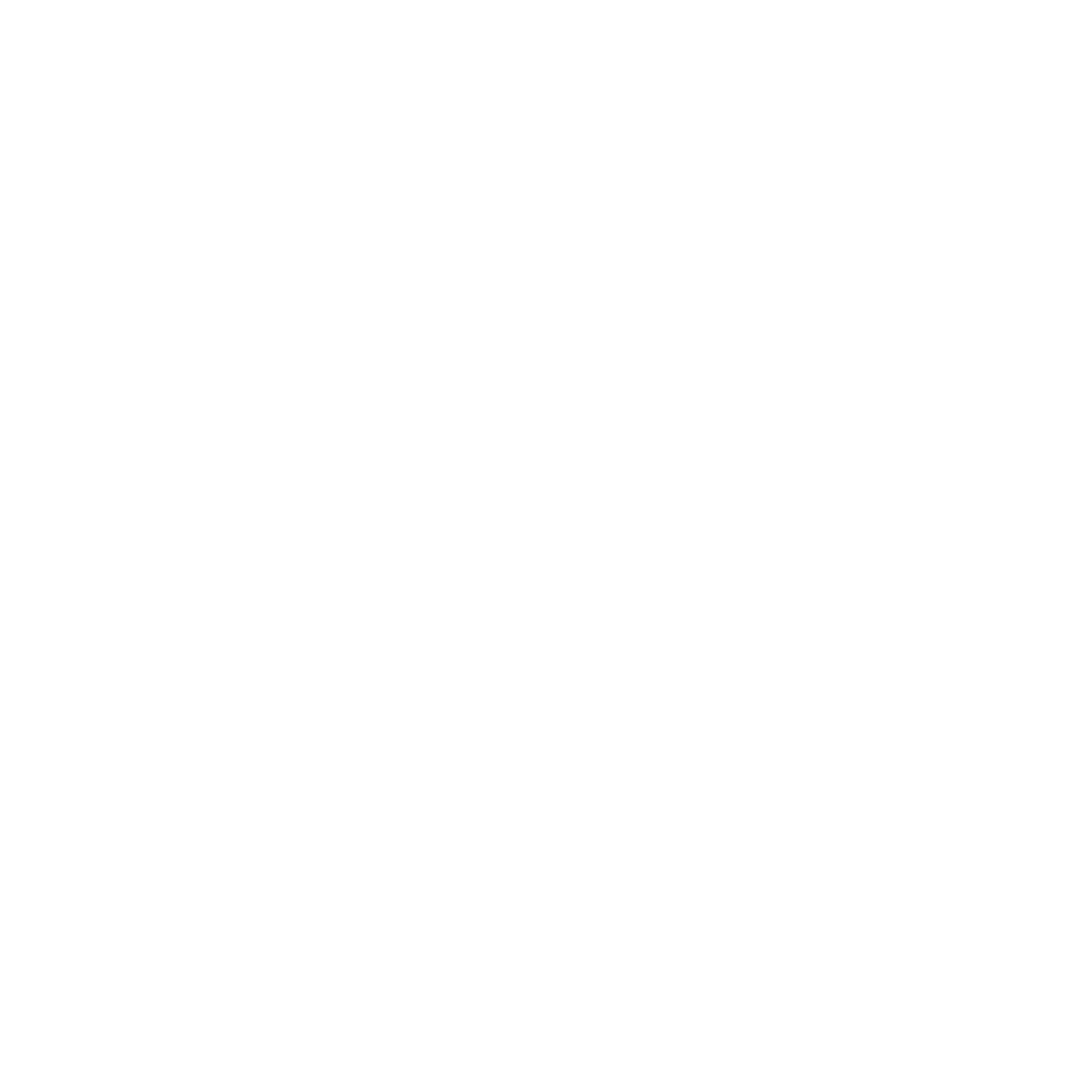
**DIPLOMA THESIS**

**YourTable**

******

Author: Meris Bihorac

Vincenz Wimmer

Sebastian Stecher

Tobias Bressler

Supervisor: Mag. Dr. Monika Putzinger

Computer Science Department - Software Engineering, 5EHIF

Table of Contents

[Table of Contents 2](#_Toc97020679)

[1 Mobile development with Flutter 3](#_Toc97020680)

[1.1 Differences between Native, Web and Hybrid-Apps 3](#_Toc97020681)

[1.1.1 Native apps 3](#_Toc97020682)

[1.1.2 Web apps 4](#_Toc97020683)

[1.1.3 Hybrid apps 4](#_Toc97020684)

[1.2 What is Flutter 5](#_Toc97020685)

[1.3 Main features of Flutter 6](#_Toc97020686)

[1.3.1 Single language for development 6](#_Toc97020687)

[1.3.2 Full support of Google 6](#_Toc97020688)

[1.3.3 Agile development 6](#_Toc97020689)

[1.3.4 Hot reload 6](#_Toc97020690)

[1.4 Widgets 8](#_Toc97020691)

[1.4.1 Common Widgets 8](#_Toc97020692)

[1.4.2 Text 8](#_Toc97020693)

[1.4.3 Row 9](#_Toc97020694)

[1.4.4 Column 10](#_Toc97020695)

[1.5 Stateless vs Stateful 10](#_Toc97020696)

[1.5.1 Stateless Widgets are 11](#_Toc97020697)

[1.5.2 Stateful Widgets are 11](#_Toc97020698)

[1.6 Architecture Overview 11](#_Toc97020699)

[1.7 Dart 12](#_Toc97020700)

[2 Literaturverzeichnis 13](#_Toc97020701)

# Mobile development with Flutter

***Sebastian Stecher***

An app, which is the abbreviation of “application”, is a software which can be installed and run on every electronic device for instance a smartphone or a computer. Most of the apps fulfil a specific purpose.

For example, an alarm clock app is only designed for users who need an alarm at a certain time. This app can’t be used for anything else, such as using it as an calendar or day planner. [1]

## Differences between Native, Web and Hybrid-Apps

There are two main types of applications. On the one side we have native apps and on the other side we have web apps. [2]

### Native apps

Native apps are installed on a specific device and are developed for this specific operating system of the device. These apps are available on app stores like Google Play, Apple App Store, etc.

As said before native apps are made for a specific operating system such as Android OS or Apple iOS. This means an app which is made for Android OS it will not run/work on a device which operating system is Apple iOS. Because of this it is necessary to build an app for every operating system which means we have to spend more money and more effort. [2]

#### Advantages of native apps

* Native apps are faster than web apps
* These apps can work without an internet connection
* These apps have more safety and security than web apps, as native apps must be approved by the App Store
* Native apps can access system/device resources such as a GPS or camera

#### Disadvantages of native apps

* These apps are more expensive to develop as compared to web apps
* Maintaining and consistently update of native apps cause more cost
* Developing the app for different operating systems is costly and time taking

### Web apps

Web apps are accessible through the mobile device’s web browser. They don’t need to be downloaded but you need an internet connection to be able to access the web app.

These applications can be used on every device which has a web browser and they are not operating system specific unlike the mobile app. The app is developed with HTML and CSS and the interactive parts can be done in JavaScript or similar languages. [2]

#### Advantages of web apps

* They function in-browser, so no download is needed
* Easy to maintain, as they have a common codebase regardless of the operating system
* These apps can be set to update themselves or automatically
* Easier and faster to build than native mobile apps

#### Disadvantages of web apps

* They do not work without an internet connection
* Web apps work slower than mobile apps
* It is difficult to discover web apps since they aren’t hosted in a specific database like an app store
* They have no security because web apps don’t need to be approved by an app store

### Hybrid apps

Mobile Hybrid-Apps are applications for smartphones and tablets that represent a mix of several application forms. A hybrid app combines the features of a native mobile app with a mobile web app. The first differences between a hybrid and a native app may be seen during the installation process. In a Hybrid-App, the entire program isn't installed; instead, only a minimal framework is set up for it. When a user opens a hybrid app, it opens in a browser-like interface, much like a web app.

**Advantages of hybrid apps**

* They have a strong platform independence
* They are cheap in development
* Hybrid apps do not need an internet connection

**Disadvantages of hybrid apps**

* Hybrid apps are not as good as native apps in performance
* Problems with native functions from device like the phone camera or GPS
* Even if you do not need to install the entire application, you must download a portion of it. This isn't the case with a web application.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Native apps | Web apps | Hybrid apps |
| Performance | + | - | - |
| Work without Internet connection | + | - | + |
| Approved by app store | + | - | + |
| Work without download | - | + | - |
| Fast to build and cheap | - | + | + |

## What is Flutter

Flutter was created by Google and is a free open-source mobile UI Framework. The first version was released in May 2017. Flutter allows you to create a native mobile application and a web application with only one codebase. A disadvantage of native apps is that you need to develop for every operating system a different application. With only one codebase means that Flutter combines Apple iOS and Android OS. Because of this it is possible use one programming language and one codebase to create two different apps.

Flutter consists of two important parts. First of all the Software Development Kit (SDK). This is a collection of tools which helps during the developing of an application. The SDK includes tools to compile the code into native machine code. The second part is the Framework which is an UI Library based on widgets. It contains a collection of reusable UI elements like buttons, text inputs and so on. These elements can be personalized by the developer.

The programming language which is used by Flutter is Dart. Dart is a typed object programming language which can be compared to C# and Java. The language itself was created six years before the release of Flutter (October 2011). [3]

## Main features of Flutter

### Single language for development

As mentioned before Flutter uses only one programming language which is Dart. Because of this the developer only need to learn one language to develop apps for Android OS and Apple iOS. This simplifies the work of developer enormous. For developer who are familiar with C# or Java adapting to Dart will be not a difficult task.

### Full support of Google

The documentation of Flutter is quite complete, very detailed and well-structured. Apart from the classic docs, it is possible to watch video lessons from the Google team. Google also has a Youtube channel called “Flutter” which features a series called “Flutter Widget of the Week”. In this series you get every Widget of Flutter explained in around one minute. Furthermore, the community of Flutter became the last years bigger and bigger and is still growing, which brings a series of solutions for the best use of the tool.

### Agile development

Flutter provides much agility to the developers. It supplies a widget system for development, because of that it is much easier to create elements who are interactive when building the software. [4]

### Hot reload

With the agility of the development Flutter supply a feature called “hot reload”. This feature helps the developer to fast and easily experiment with the native application. Hot reload injects updated source code files into the running Dart Virtual Machine (VM). The Flutter framework then automatically rebuilds the widget tree after the VM updates classes with the new versions of fields and functions. This allows you to see the changes as fast as possible.

Beside the hot reload there are two other ways to run your application. One of them is called “hot restart”, this feature loads code changes into the VM and then restarts the Flutter app, losing the app state. The other way is the “full restart”. The full restart restarts the iOS, Android or web app. It recompiles the Java / Kotlin / ObjC / Swift code, because of this it takes longer than the other two methods. [5]



**Figure 8: Shows Android Studio and where the Hot Reload button is placed**

#### Special cases

There are also special cases where the hot reload not work as wanted. In this case a hot restart or a full restart is needed.

An application is killed:

* When an app is killed the hot reload can break. The app can break for example if it was in the background for too long.

Compilation errors:

* If a change in the code leads to a compilation error the hot reload generates an error message like:

Hot reload was rejected: '/Users/obiwan/Library/Developer/CoreSimulator/Devices/AC94F0FF-16F7-46C8-B4BF-218B73C547AC/data/Containers/Data/Application/4F72B076-42AD-44A4-A7CF-57D9F93E895E/tmp/ios\_testWIDYdS/ios\_test/lib/main.dart': warning: line 16 pos 38: unbalanced '{' opens here  
Widget build(BuildContext context) {  
^  
'/Users/obiwan/Library/Developer/CoreSimulator/Devices/AC94F0FF-16F7-46C8-B4BF-218B73C547AC/data/Containers/Data/Application/4F72B076-42AD-44A4-A7CF-57D9F93E895E/tmp/ios\_testWIDYdS/ios\_test/lib/main.dart': error: line 33 pos 5: unbalanced ')'  
);  
^

Changing fonts:

* Hot reload does not change fonts. In this case a hot restart is needed.

## Widgets

The inspiration of the Flutter widgets came from React (JavaScript library for building user interfaces). The idea behind the widgets is that the developer builds the user interface out of widgets. Widgets describe the appearance of their view based on their current setup and status. The widget rebuilds its description when the state changes, which the framework diffs against the previous description in order to determine the minimal changes needed in the underlying render tree to move from one state to the next. [6]

### Common Widgets

Flutter provides a set of strong basic widgets that are very commonly used. Some will be explained in more detail in the coming lines.

### Text

The Text widget displays a string as text with a single style. Depending on the layout requirements, the string can span numerous lines or be presented on the same. [7]

This example shows how the implementation of a Text widget works:

FittedBox(  
 fit: BoxFit.fitWidth,  
 child: Padding(  
 padding: EdgeInsets.only(top: 5.0, left: 10,bottom:10),  
 child: Text(  
 AuthService.*restaurants*[widget.restaurantIndex].restaurantName,  
 style: TextStyle(fontSize: 25),  
 ),  
 )  
),

Ein Bild, das Text, Essen, Sandwich, Zwischenmahlzeit enthält.

Automatisch generierte Beschreibung

**Figure x: Shows a Text widget**

### Row

This widget displays its children in a horizontal array. A Row widget is important if we want to have more widgets side by side. It is important to know that an error occurs when too much space is required. [8]

This example shows how to implement a Row widget with two children:

Row(  
 children: [  
 Text("Kinderstuhl"),  
 Checkbox(  
 activeColor: secondColor,  
 value: childChair,  
 onChanged: (value){  
 setState(() {  
 childChair = value!;  
 });  
 }  
 ),  
 ],  
),



**Figure x: Shows a Row widget**

Just a quick example of what it would look like if the line is longer than the space available:



**Figure 11: Shows an error because of too little space**

### Column

The Column widget displays its children in a vertical array. The column is like the Row widget but in a vertical way. It is used to have widgets among each other. As with the Ro widget a error will occur when too much space is required. [9]

This example shows how the implementation of a Column widget with three children works:

Column(  
 mainAxisAlignment: MainAxisAlignment.start,  
 crossAxisAlignment: CrossAxisAlignment.start,  
 children: [  
 Row(…),  
 Padding(…),  
 Padding(…),  
 ],  
),



**Figure x: Shows a Column widget**

## Stateless vs Stateful

A widget in Flutter can either be stateless or stateful. Widgets who are reacting to an user activity - it is stateful. This means a stateless widget does not change. These widgets are useful when only the configuration information in the object itself and the BuildContext in which the widget is inflated are important. For example, the Text widget is stateless because it does not change when a user interacts with it. Stateful widgets can change their properties during run-time. [10], [11], [12]

### Stateless Widgets are

* static widgets
* which have no state, after they rendered once they will not update themselves
* and do not depend on data change
* Example: Text and Icon

### Stateful Widgets are

* dynamic widgets
* whose user action or data change can cause an update during runtime of the widget
* in an internal state
* Example: Checkbox and Radio Button

[13]

## Architecture Overview

Flutter reuses as much code as possible across different operating systems and provides applications with a direct interface to the underlying platform services. While the application is in the development phase, Flutter apps run on a virtual machine that provides hot-reload capability. Once the application is released, Flutter apps are compiled to the machine code.

“Flutter is designed as an extensible, layered system. It exists as a series of independent libraries that each depend on the underlying layer. No layer has privileged access to the layer below, and every part of the framework level is designed to be optional and replaceable.“[14]

Generally, developers work with Flutter through the Flutter framework which is written in the programming language Dart. It includes the layout, a diverse selection of platforms, and numerous levels of foundational libraries. These layers are organized in the following order from bottom to top:

* Basic core classes, as well as building block services like animation, drawing, and gestures, provide regularly used abstractions over the underlying base.
* The rendering layer provides an abstraction for working with layout and constructing a renderable object tree.
* The widgets layer is a composition abstraction. In the widgets layer, each render object in the rendering layer has a matching class. Furthermore, the widgets layer lets developers create reusable class combinations. This is where the reactive programming model is first introduced.
* The Material and Cupertino libraries provide extensive sets of controls that implement the Material and iOS design languages using the widget layer's composition primitives.

[14]



**Figure 6: Flutter architecture overview**

## Dart

Dart is a client-optimized programming language for developing fast apps on any platform. Its purpose is to provide the most productive cross-platform programming language and a versatile runtime environment for app frameworks.

# Literaturverzeichnis

|  |  |
| --- | --- |
| [1] | "Indeed," [Online]. Available: https://www.indeed.com/career-advice/career-development/what-is-an-app. [Accessed 29 December 2021]. |
| [2] | „Geeksforgeeks,“ [Online]. Available: https://www.geeksforgeeks.org/difference-between-native-apps-and-web-apps/ . [Zugriff am 29 December 2021]. |
| [3] | "Freecodecamp," [Online]. Available: https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/ . [Accessed 30 December 2021]. |
| [4] | "Logap," [Online]. Available: https://logap.com.br/en/blog/what-is-flutter/ . [Accessed 30 December 2021]. |
| [5] | "Docs Flutter," [Online]. Available: https://docs.flutter.dev/development/tools/hot-reload. [Accessed 30 December 2021]. |
| [6] | "Docs-Flutter," [Online]. Available: https://docs.flutter.dev/development/ui/widgets-intro . [Accessed 2 January 2022]. |
| [7] | "API Flutter," [Online]. Available: https://api.flutter.dev/flutter/widgets/Text-class.html . [Accessed 4 January 2022]. |
| [8] | "API-Flutter," [Online]. Available: https://api.flutter.dev/flutter/widgets/Row-class.html. [Accessed 4 January 2022]. |
| [9] | "API-Flutter," [Online]. Available: https://api.flutter.dev/flutter/widgets/Column-class.html. [Accessed 4 January 2022]. |
| [10] | "Docs Flutter," [Online]. Available: https://docs.flutter.dev/development/ui/interactive. [Accessed 6 January 2022]. |
| [11] | "API-Flutter," [Online]. Available: https://api.flutter.dev/flutter/widgets/StatelessWidget-class.html. [Accessed 6 January 2022]. |
| [12] | "API-Flutter," [Online]. Available: https://api.flutter.dev/flutter/widgets/StatefulWidget-class.html. [Accessed 6 January 2022]. |
| [13] | "Geeksforgeeks," [Online]. Available: https://www.geeksforgeeks.org/difference-between-stateless-and-stateful-widget-in-flutter/ . [Accessed 6 January 2022]. |
| [14] | “Flutter architectural overview,” [Online]. Available: https://docs.flutter.dev/resources/architectural-overview. [Accessed 21 February 2022]. |