**BABEŞ-BOLYAI UNIVERSITY**

**FACULTY OF MATHEMATICS AND COMPUTER SCIENCE**

**SPECIALIZATION COMPUTER SCIENCE (IN ENGLISH)**

**DIPLOMA THESIS**

**Solving parking in Cluj-Napoca with Flutter**

**Author**

**Tudor KALISCH**

**Supervisor**

**Dr. COROIU Adriana Mihaela, *Lecturer Professor***

**CLUJ-NAPOCA**

**2020**

Table of Contents

[Introduction 4](#_Toc49716361)

[Theoretical 6](#_Toc49716362)

[Mobile operating systems 7](#_Toc49716363)

[Android 7](#_Toc49716364)

[iOS 8](#_Toc49716365)

[Comparisons 8](#_Toc49716366)

[Front end approach 10](#_Toc49716367)

[Native 10](#_Toc49716368)

[Android Native 10](#_Toc49716369)

[iOS Native 13](#_Toc49716370)

[Hybrid 15](#_Toc49716371)

[React Native 15](#_Toc49716372)

[Flutter 17](#_Toc49716373)

[Front end decision 19](#_Toc49716374)

[Back end approach 20](#_Toc49716375)

[.NET Core 20](#_Toc49716376)

[Spring Framework 21](#_Toc49716377)

[21](#_Toc49716378)

[Conclusion 23](#_Toc49716379)

[Application 24](#_Toc49716380)

[26](#_Toc49716381)

[28](#_Toc49716382)

[Architecture 31](#_Toc49716383)

[Front-end 31](#_Toc49716384)

[Back-end 35](#_Toc49716385)

[Experiences working with Flutter and Spring 37](#_Toc49716386)

[Conclusion 41](#_Toc49716387)

[Bibliography 43](#_Toc49716388)

[Electronic resources 45](#_Toc49716389)

# 

# Introduction

At the time of writing this paper, parking is one of the few issues that the city of Cluj-Napoca is struggling with right now. We all know the feeling when we want to go get a coffee or go out to dinner, and our first thought is, "Where I am going to park?". With this in mind, many people miss out on experiences because they do not want to go through the struggle of finding a parking space, and with the stress that they might not find one and they have to go home disappointed. Not going out is also one reason why many residents of Cluj-Napoca do not use their cars at all, resulting in many parking spots being forever taken by cars that are only used weekly or monthly.

Everyone wants to have a parking space assigned to the at the destination before getting in the car. This parking space would save a lot of time in which the user is not looking for a parking space and releases the stress of looking for a parking spot.

This is the issue that I am trying to solve. I wanted to create a solution with which the users can get their parking spot booked at a particular time and location and pay for it directly, without remembering to pay a ticket when they leave the parking lot or worrying about how much is going to cost. The goal is to make the application easy to use and all the flows in the application to work seamlessly. Because not all the users are familiar with technology, the design should be direct and straightforward to remove the confusion.

The application is also a great business opportunity for parking services. The parking locations can join for a subscription. All they have to do is provide a price and a server endpoint to send and receive the information. We will send the data to the parking location that a booking is created for a user, and the application will generate a code that will be scanned in at the parking lot.

I think the best solution for this would be a mobile application in which you could book the parking spots while you are in the car. The application should be available on both Android and iOS and we also need a server in order to keep all the bookings made for each user and to have a list of all the locations where the users can park. In my opinion, having a web application is not really necessary because each person has a mobile phone and it is way easier to make a booking on a phone than to get a laptop and open a website.

In the next chapters I am going to explain what technologies we have available for developing this application, both front-end and back-end. For front-end I am going to give more details on native and hybrid with their own features, programming language and IDEs and the same for back-end side. With that information we can make an educated choice on which technology is the best solution. After that, there will be an in-depth description for the features of the application and the experience of developing the application with the technologies used. After that, we will summarize all these details into a conclusion and see where we can go next with the application.

# Theoretical

One application that solves a part of this problem is Cluj Parking. It is a software application that gives the user the ability to check the parking space's availability in parking lots with a barrier feature of Cluj-Napoca with the help of a smartphone. This application exists for Android and iOS with a partnership with Cluj-Napoca City Hall. It is an intersecting idea, but this solves only half of the problem. This application gives a solution for looking for a parking space, but it does not guarantee that the client will have a parking spot exactly when he gets to the spot. Currently, the reviews are mixed. Some users are complaining that the applications are not synchronized correctly, and they would like to have a feature with which they would find the parking spot that is closest available to them or the destination of their choosing.

Figure   
Cluj Parking logo

Another application that is in this field in Cluj-Napoca is YeParking. YeParking is trying to solve this issue with a different approach. They are using a parking sharing feature. Using this application, users can share a parking spot. When the user is using the parking spot, he has to let the other people know it is taken, and when he leaves the spot, he has to announce to the other users that the parking spot is available. The decision to share a parking space is exciting from an environmental perspective, and it might also be implemented with the idea of car sharing. The issue here is the community. With a low number of users that do this, the application is not applicable. Nevertheless, if all citizens of Cluj-Napoca would use this solution, driving in Cluj-Napoca would be much more comfortable.

Figure   
YeParking logo

One more company that gives a solution for this is TPark. The way they are trying to solve this problem is by having a different application for each city they work with. Being a bigger company gives them the means to invest and give the users a top-notch UI and maybe some more features that the other applications are lacking. The only issue I see with TPark is also one of the benefits. Being a huge app with more cities to manage makes the application a bit impersonal. The impersonality can make the application a little inconsistent in different cities. When they join a new city, they might not be aware of all the different parking facilities that exits in the city and they may not be able to update the application data consistently because of the different data that each parking facility needs to use their display system and payment functionality.

Figure 3  
TPark logo

We have many solutions when it comes to implementing a mobile application. One of them is going native and one of them is going hybrid. There are also some progressive apps that can do that and some hybrid framework that's actually rendered web page as a mobile application. I'm going to go through all the software that is available that ends that are viable for our problem. [21,22,23]

# Mobile operating systems

## Android

Android is an operating system (OS) released in 2007 and its first launch on a mobile device was in September 2008 which was the HTC Dream

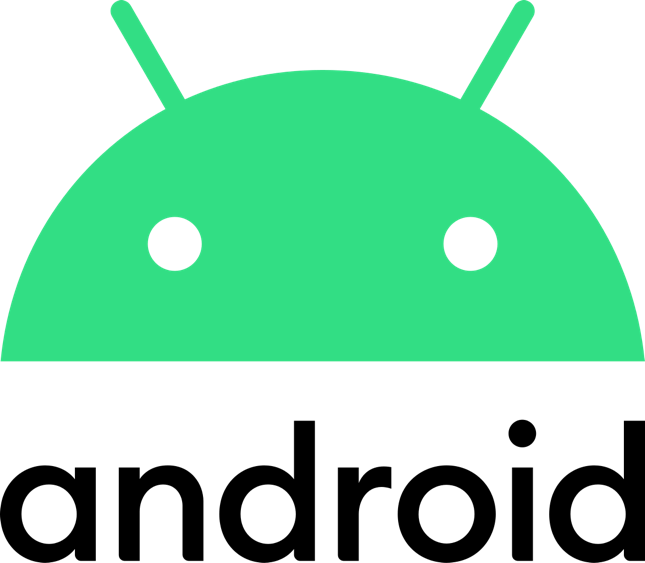
It is a free open-source software and its source code is known as Android Open Source Project or AOSP. Its kernel is based on the Linux kernel with added software from the open-source area. The OS is designed mainly for mobile devices with a touchscreen feature, but in the recent years, many devices use Android as their Operating System because of its open-source decision and the applications that it can generate. The devices for which Android was firstly designed for are smartphones and tablets.

Figure   
Android logo

The main developers for this platform are Open Handset Alliance and it is sponsored by Google. Because of the Google sponsorship, most devices (more than 70%) come with applications developed by Google already installed on their phones, like Google Chrome, Google Maps, Google Play Music, YouTube, and many more. With this decision, Google is keeping users in their ecosystem which is raising their market share on Android.[7]

## iOS

A close up of a logo

Description automatically generatediOS or formerly known as iPhoneOS is a mobile operating system released on June 29, 2007, created and developed by Apple Inc. They are also the developers and designers for their hardware. Their software powers some of their company’s devices, including the iPhone, iPod touch, and their iPad until 2019 when iPadOS was released.

Figure   
iOS logo

Releases are made annually for the operating system. The current stable version being iOS 13, which was released on September 19, 2019.

One of the best qualities of the iOS operating system is that it is built specifically for its devices. Having control of both the hardware and software of their devices makes iOS one of the more stable operating systems for mobile devices and it gives people a unique look at the software.

Apple phones are the most popular phones in the US. Even though Android has over 50% of the market, this percentage is shared among many manufactures. With the latest data, Apple covers 46% of the US phone market and it seems to be rising This data shows that the iPhone is the most popular phone in the US. [8]

## Comparisons

iOS and Android are the only viable options when choosing an operating system when developing an application. There are some different operating systems, but their application is so specific, and their usage is so low that we are not required to also build the application on those specific operating systems. Some of them are Samsung, Windows OS, Nokia, BlackBerry OS, etc. (see Fig.2). Their cumulated percent is lower than 1%.

iOS is a more closed system whereas Android is more open. In Android, you have the ability to change the interface, permission and you have the ability to install any application you want on the phone without being a developer. This can also reduce the quality of the applications available on Android. iOS being a more closed operating system is controlled by Apple and it is in their decision what application gets to the user and which do not. This increases the quality of the applications available on the App Store, but you lose some of the freedom. There is a question in the quality of the phones found with Android, with so many manufacturers, the quality of the phones will vary. This means that Android cannot guarantee a consistent quality in the hardware, but this gives more choices to the user on which phone he decides to buy.

In Romania, the percentage of the operating system is a little bit different than the rest of the world. As shown in the graph below, (see Fig 2.) Android has the overwhelming majority with 81.49% market share and iOS has 17.68%.

With this information, we can deduce that we still need a mobile application for iOS developed, but the main focus for our application should be Android. This information is important to know which version of the application needs more attention and to have a deciding factor on which application should get the most details. [15,16]

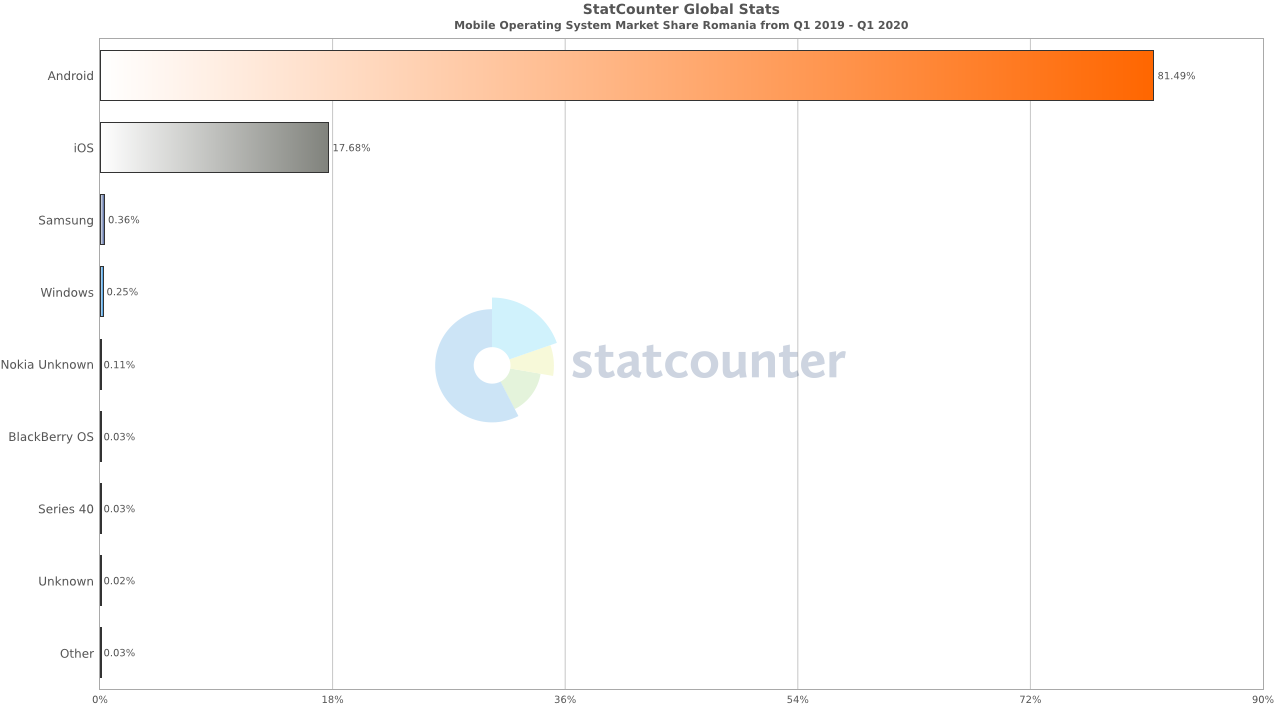


Figure   
Graph displaying the market share of operating systems in Romania

# Front end approach

## Native

Native implementation of the mobile application means using the Android SDK (Software Development Kit) for Android and iOS development kit. At first sight, native is what we think about when downloading an app from the Play Store or App Store. What distinguishes native applications from hybrid applications is that they are developed for specific devices. For Android, we use Java or Kotlin and Android Studio as the Integrated Development Environment (IDE) and for iOS we can use Objective-C or Swift as the programming language and XCode as the IDE.

The advantage of choosing native development is that it is the most reliable when it comes to user experience and some consider it the fastest way to develop an application. Native apps can also interact with all of the device’s features like Bluetooth, camera, location, microphone, etc. Some Hybrid SDKs struggle with the hardware part of the mobile device and this might be a hindrance in development.

However, building an application native requires more time for the implementation, because you need to implement all the features on both platforms. Also, native applications use difficult programming languages, which require experience to use correctly. These being said, native might not be the best solution when building a simpler application.

### Android Native

Building Android Native can have a lot of advantages, but also disadvantages. Having all the tools and the possibilities to use all of the mobile device’s features gives you the assurance that every requirement can be built, tested, and deployed. Every new feature that appears in the most recent version of Android will be available for them right at the start. This means that the software that you are building can always use the latest features and your application’s design will not fall behind. This cannot be said about hybrid apps. Because you rely on the developers of your framework to build the features that appear in Android, there is a possibility that the feature will not be available right from the start or even not be available at all.

Although, all this freedom is countered by the speed of the development. With all the freedom of this comes the hurdle of building a native application. The speed can be low, and you need to have a good basic knowledge of the Android SDK to properly build the application. Also, because Java and Kotlin are the main programming languages, some of the developers are drawn-back by the initial difficulty of the language as they are considered one of the more difficult languages to build software with. Google has been trying to counter the difficulty of the Java language with Kotlin, which gives you a little bit more freedom as it is not as strongly typed, but it still has a hard-null check features. This can get quite overwhelming for developers that are not familiar with the issue. [9]

Languages

Java

 Java is a set of software and specifications provided by Oracle Corporation. This includes an execution engine, Java Virtual Machine (JVM), a compiler and, a set of libraries. This provides a system for developing software for a cross-platform environment. It is designed to work on the JVM, and it borrows heavily from C and C++, but it is built to work with an object-oriented design. The Java code will be deployed on the JVM as bytecode and all memory management is handled automatically trough garbage collection, but it also gives you manual access.

Figure   
Java logo

Kotlin

Kotlin first appeared in 2011 and it was announced as the default Android

A close up of a logo

Description automatically generateddevelopment language at Google I/O 2017. It is developed and supported by JetBrains. It is a compiled, statically typed language. It is a free and open-source project under the Apache 2.0 license. Kotlin can be compiled for several different platforms. The target platform being JVM, is designed to interoperate with Java. This gives the user the ability interchange between Java and Kotlin in their Android project.

Figure   
Kotlin logo

On 7 May 2019, Google announced Kotlin to be the preferred programming language among Android developers. Now Kotlin is set as the default compiler for Android, pushing developers from Java to Kotlin more and more.

C/C++

###### C++ - Wikipedia

Android SDK cannot directly run C++ code, but it can be used with the Android Native Development Kit (NDK). This is a toolset that allows you to use C and C++ together with Java/Kotlin in Android. With the NDK you can manage activities and access physical components of the phone like camera, microphone, or sensors. C and C++ are usually used in Android for optimization of sensors or camera capabilities as it offers you more flexibility than Java and Kotlin, but because of its difficulty and development time, developers use Java or Kotlin as their main programming languages.

Figure   
C++ logo

IDE

 Android Studio is the official IDE for Google’s Android operating system. It is built by JetBrains IntelliJ IDEA software and it is specifically designed for Android. It comes available on all the desktop platforms: Windows, macOS, and Linux.

This came as a replacement for the initial Eclipse Android Development Tools (E-ADT). Google first announced Android Studio at Google I/O conference on May 16, 2013. It got a beta release in 2013 and the first official release was in June 2014.

Figure   
Android Studio logo

Android Studio is built to support all programming languages that IntelliJ Idea also supports, which are Java, C/C++. With the release of Android 3.0, Kotlin was also added to the list in October 2017.

Some of the features that Android Studio has are:

* A rich and useful design editor
* Gradle plug-in for build support
* App-signing capabilities
* Lint to improve the code quality and catch errors
* An Android Virtual Device which helps you build Emulators[3,4,7,19]

### iOS Native

You can create an iOS application using Objective-C or Swift. Unlike Android, iOS has a closed source code build specifically for Apple’s hardware. With this being said, you need a Mac device to develop Native apps for the iOS platform. All the iOS applications can be found on the App Store.

Native development can improve the speed of your application which can be important for user experience. This can also help when submitting an application to the App Store. Apple has a lengthy review process of each application and that can be daunting. Your application can fail the review and you will receive a reason or list of reasons why your application did not succeed. In general, native applications have an advantage to this. Reviewers are usually more lenient towards native development and hybrid development is checked thoroughly. Also, some hybrid applications that load a web-view cannot load in offline mode, which can be a deal-breaker for some apps.

Although native app development has multiple advantages, it’s not a simple process. Developers can lack certain skill or resources to build a native application. One more disadvantage for this can be that there exist a number of bugs and errors due to different platforms. It is not easy to control and fix them. This reduces the efficiency of the app. This can increase the cost of maintenance.

Language

Objective-C

A picture containing mirror

Description automatically generated Objective-C is a general-purpose, object-oriented programming language that adds Smalltalk-style messaging to the C language. It is considered a thin layer of the C programming language or a superset to C. It is possible to compile a C program with an Objective-C compiler and you can include C code in an Objective-C program.

Figure   
Objective-C logo

It was developed in early 1980 and it was the main language with which macOS and iOS was built. It was supported mainly by Apple until the release of Swift in 2014. Swift was quickly accepted by the majority of iOS developers as the better language for it and now you see less and less Objective-C code in projects as people usually migrate to Swift.

Swift

A picture containing drawing

Description automatically generated

Figure   
Swift logo

Swift is a compiled programming, general-purpose language created by Apple Inc. for macOS, iOS and other Apple software. It is designed to work together with Apple’s Cocoa framework and the code written in Objective-C for Apple products. It is included in XCode since version 6, released in 2014.

It was introduced to the public at the World Wide Developers Conference (WWDC) 2014

and it got a big update in 2015 with Swift 2.0. From then it was recommended as the primary programming language for iOS developers as it give more freedom and it speeds up the process.

IDE

A close up of a sign

Description automatically generated

Figure   
XCode logo

Xcode is an integrated development environment (IDE) for macOS containing the software development tools developed by Apple for developing software for macOS, iOS, and the other Apple software. It was first released in 2003, and the latest stable version is 11.6.

Unfortunately, Xcode only works on macOS and this restricts the developers that want to build iOS applications to do it on macOS. Usually, you have to have a macOS device from Apple in order to do that, but you can bypass this if you want to go through the struggle of building your own macOS virtual machine.[8,11,20]

## 

## Hybrid

A hybrid application is an application which shares code that runs on both iOS and Android. Developers usually chose this kind of software because it is more efficient, code does not need to be written multiple times for each platform and this reduces the development speed of the application.

Multiple frameworks are available for this and it can be done in different ways. One way frameworks solve this problem is by using the web stack to load the application. Using a web stack has advantages. It has plenty of frameworks available, you can share code between web and mobile platforms and the ability to take advantage of your web skills.

However, this comes with some challenges. Compatibility, performance, and memory usage usually are the first ones. Because you are rendering a webpage engine to load your application. In some frameworks, you can have an application not working offline because you are loading webpages and that can be a deal-breaker to some projects.

### React Native

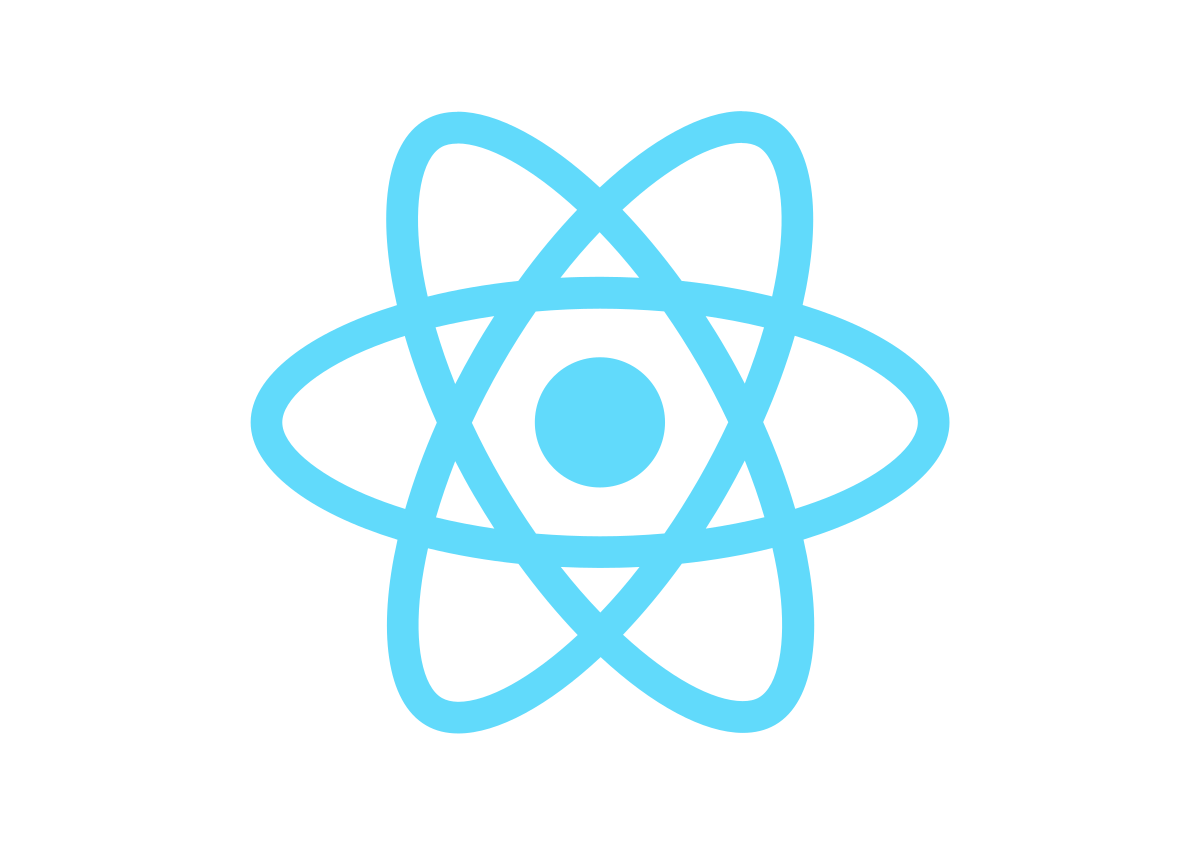
 React Native is one of the hybrid frameworks available for mobile development. It was created by Facebook, Inc. with its initial release in March 26, 2015. The idea for this framework came from a comment of Mark Zuckerberg, “The biggest mistake we made as a company was betting too much on HTML as opposed to native”.

Figure   
React Native logo

React primitives render to native platform UI, meaning your app uses the same native platform APIs other apps do. The principles of React Native are identical to React except that React manipulates the DOM via the Virtual DOM. React Native uses background processes that interpret JavaScript directly on the phone and communicate with the platform via serialization and Bridge.

Language

JavaScript

A picture containing drawing

Description automatically generated JavaScript (JS) is a lightweight, interpreted programming language. It is well-known as the scripting language for Web pages, even if it is used in many non-browser environments such as Node.js, Apache, or Adobe.

Figure   
JavaScript logo

The major advantage when it comes to JavaScript is its popularity. It’s very hard to find a programmer who hasn’t used it at some point. Difficulty is also a major point in its popularity. JavaScript is weakly typed, which means that certain types are going to be directly cast when used.

One of the disadvantages of are also its weakly-typed approach. While Java, Switch and, Objective-C are strongly-typed, JavaScript is interpreted and that means that your variables can be anything at any time. You need to be extra careful when writing your application, as you are not sure which type of object will be sent to which method or class. This might lead to more time searching for errors in your applications.

IDE

Visual Studio Code

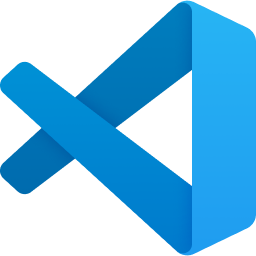


Figure   
Visual Studio Code logo

Visual Studio Code (VS Code) is a free code editor made by Microsoft. It is available on Windows, macOS, and Linux. It was first announced on April 29, 2015, by Microsoft at the 2015 Build conference. Features include are debugging, syntax highlighting, code completion, embedded GIT along with many others.

Along with developers, it is highly recommended because of if it’s compatibility with many programming languages. It has a store extension feature with which you can install free extensions to build applications. It has an intuitive yet simple user interface. The fact that it works on all platforms also increases its popularity, because you can change from platform to platform and still use the same software. VS Code goes for a language-agnostic approach which enables you to have different projects opened at the same time. [1,9,18,]

### A picture containing computer, computer Description automatically generatedFlutter

Figure   
Flutter logo

Flutter is an open-source, front-end framework built by Google. It can be used to create applications for Android, iOS, Web, Linux, macOS, and Windows with a single codebase.

The first version of Flutter was announced during the Google Developer days conference in May 2017. The first beta version was released on 13 March 2018 with its first stable version on 4December 2018.

Render engine

Flutters render engine is built primarily in C++ with the support of Google’s Skia graphics. With this engine, Flutter creates platform-specific widgets that will be rendered on the screen. This works both for Android and iOS. This is one of the advantages that Flutter has over the other hybrid solutions.

Design view

Material Design is one of the core features that Flutter has. All the specific Flutter widgets are created with Material Design in mind, this gives the user the ability to create applications that will always have a modern look on both platforms.

Compile ahead-of-time

The application is compiled ahead-of-time into bytecode, not at runtime compared to React Native. This gives a better performance because there is no bridge to slow down the execution.

Support

Because Google recommends Flutter for the hybrid building of applications and it is still recommended as the better solution for simpler applications, the support that it gets is quite heavy. In the first year since its release, all the new features that the Android SDK was supported by Flutter right from the start.

Truly cross-platform

With the announcement of the Hummingbird Project by the Flutter team, it makes Flutter a truly cross-platform framework. The goal of this project is to make Flutter make on all platforms available.

Performance

Flutter was built with performance in mind. It is still a little slower when compared to native, but it fares great against its competitors, like React Native or Ionic with both CPU usage and battery consumption.

Language

Dart

A close up of a logo

Description automatically generated Dart as a language is very robust. It is created by Google. It resembles C-like Object Oriented Programming languages like Java, C#, and Kotlin. Because it is a general-purpose programming language, it compiles fast. It is object-oriented, optionally typed, and class-based which is excellent for reactive programming which can usually be found on front-end frameworks.

Figure   
Dart logo

Because it is used by Flutter, developers try to compare it to Reacts Native’s JavaScript or TypeScript language. This is far from the truth. It is quite different from C# or Java, as it is not a verbose language. It stands shoulder to shoulder with Kotlin or Swift when it comes to difficulty and ease of use.

Dart is approximately two times faster than JavaScript. It is type-safe and is compiled with both ahead-of-time (AOT) and just-in-time (JIT) compliers.

Dart is backed by a tech giant. When you decide to invest time and energy into learning a new language, you want to be sure that the language that you are learning will still be here for years to come. After months of learning, the last thing you want to hear is that your language is no longer supported or that it goes deprecated. As far as Dart is concerned, there is nothing to worry about. The language is the fast-growing language inside Google, and it is used for internal products like AdWords, Flutter, AngularDart, etc.

IDE

Flutter uses the same IDE as React Native witch is Visual Studio Code. It was presented above.[2,5*,*6,12,14,15]

## Front end decision

As a conclusion, with all the information that was given above, I choose Flutter as the framework I want to use for the front-end of the application. With all the advantages to the time of development, solid programming language, the choice of the IDE (Visual Studio Code which I prefer over any other IDE) and overwhelming support from the Flutter team and Google makes it the better choice in my opinion.

Going Hybrid is a better solution in this case, as the application does not have a high complexity of features. The time of development is the high point here. Because we need to develop the same application for both Android and iOS, going Hybrid means that the time of development will be almost halved, with no significant drawback on performance, as stated in the performance testing in (Comparative Study with JavaScript) Thanh and Mohanity-Rekha Dey papers. The development of this application still tests the framework ability to work with basic mobile device features like Maps, Calendar, Payment, etc. This means that we still need to be careful when we implement platform-specific features.

Choosing Dart as the programming language is the better choice in my opinion. When compared to JavaScript which can raise issues which are hard to catch, Dart has a debugger, and its class-based design and optionally typed orient is better for reactive programming like mobile. When compared to Java and Kotlin of Android or Objective-C and Swift from iOS, it still fares better, as it has all the advantages of Kotlin and Swift without the hard entry-level. And when compared to Java and Objective-C which present a very slow development time and an entry level difficulty with no apparent advantage, Dart seems like the obvious choice.

With a combination of Flutter as a framework, Dart as the programming language, and Visual Studio Code as an IDE, Flutter seems like the better choice of all the frameworks available for mobile development. With its strong support from Google, it seems like it’s a framework that is here to stay for the long run. [1,2,5,6]

# Back end approach

## .NET Core

A close up of a sign

Description automatically generated .NET Core is an open-source computer software framework for Windows, macOS, and Linux. It is the cross-platform successor of the .NET framework which was only available on Windows. .NET Core is developed by Microsoft along with the .NET Foundation. The initial release was on June 27, 2016.

Figure   
.NET Core logo

.NET Core supports the previously mentioned operating systems as your development workstation. .NET Core supports the use of NuGet packages for its dependency installation, unlike .NET Framework which relies on Windows update. Two of the main components for .NET Core are CoreCLR and CoreFX.

Programming languages

.NET Core supports C#, Visual Basic, and F#. Because C# is the most popular of this and Visual Basic and F# being niche, I will cover C#.

A close up of a logo

Description automatically generatedC#

Figure   
C# logo

C# is an object-oriented, modern, type-safe language. C# started from the C family and it is most familiar with C++ and Java. It is a component-oriented language.

C# provides support for these concepts, making the development of software components as easy as possible. Since its first appearance 20 years ago, C# has added features to support new workloads emerging in software design.

IDE

Microsoft Visual Studio

A picture containing drawing, clock

Description automatically generated

Microsoft Visual Studio is an IDE created by Microsoft. It uses development platforms such as Windows API, Windows Forms, Windows Store etc. It can produce both native code and managed code. First released in 1997 as Visual Studio 97 it is one of the oldest IDEs that is still used today, showing that Microsoft can create software that lasts over the years.

It appears in three versions, Community, Professional, and Enterprise, each with its own set of features. Usually, Community edition is designed for personal use or academic purposes with its paid editions, Professional and Enterprise, being designed for companies. [3,10]

Figure   
Visual Studio logo

## Spring Framework

## A picture containing drawing, food, plate Description automatically generated

Figure   
Spring logo

Spring Framework is a framework and inversion control container for Java. The core features of this framework are used by a Java application. With extensions, it can be used for building web applications. It was first released on 1October 2002 is open source.

The Spring Framework makes use of modules in order to provide all the features that are built for the framework. This gives you the ability to configure the project in order to have everything you need for building your application.

It facilitates good programming practices such as programming using interfaces instead of classes. Spring enables developers to develop enterprise applications using POJO and POJI model programming.

Inversion of Control is another top feature of the Spring framework where application dependencies are satisfied by the framework itself. Framework creates the object in runtime and satisfies application dependencies.

Spring Boot is one of the features of the Spring Framework. With the design of ‘convention-over-configuration’ as its concept, Spring Boot is a feature that makes it easier to set up a new project, without the need for an ample configuration as it is usually needed on a server application. You have the ability to create stand-alone Spring applications that run directly from the start. This is one of the reasons for Spring’s popularity, the fact the you can get from setting up the project to writing code in a matter of minutes.

Programming Language

The programming language that is used in Spring is Java. Java was already presented in the Android Native part of the paper.

IDE

IntelliJ IDEA

A close up of a logo

Description automatically generated IntelliJ IDEA is an IDE written in Java and Kotlin for developing computer software. It part of the JetBrains (formerly known as IntelliJ) family. IntelliJ IDEA was first released in January 2001 and it was one of the first IDEs with advanced code navigation and code refactoring.

Figure   
IntelliJ IDEA logo

IntelliJ IDEA supports the install of additional plugins created by the community which can speed up the process of building applications with this IDE. It is also the base of Android Studio, which shows Google’s support to JetBrains.

It is considered one of the best IDEs for Java and Kotlin, being also one of the most popular IDEs for these particular languages. Their code-completion aid software, IntelliSense, is considered one of the best in the business which includes a number of features that help you keep track of the code that you are writing. [3,10]

## Conclusion

I have chosen to continue with Spring Framework, Java and IntelliJ IDEA for the back end. Spring Framework has a lot of features that made me chose it over .NET Core, the main one being Spring Boot. Having a feature that configures a project for you where you just have to tell it what you want to build with it is a done deal for me.

As per the programming language, I prefer Java over C#. Here, the most important is my experience with Java. As listed above, we know that Java and C# have a quite high entry-level difficulty for each language, and here is where experience matters. I also appreciate Java’s platform availability, whereas for C# I would need to create a virtual machine in order to work with C# and .NET Core on macOS.

For the IDE, I also prefer the JetBrains approach to design and plugin installation over Microsoft Visual Studio. Because I still need to use some configuration for front end in Android Studio which is also developed by JetBrains, IntelliJ IDEA seems like the better choice. Familiarity plays a big role in this decision. Also, in my experience with Microsoft Visual Studio, I encountered problems where only a restart solved my problem which makes the IDE quite inconsistent in my opinion.

# Application

The goal for this application is to make it possible to book a parking space for a set amount with as few clicks as possible. We all know how annoying an application where you solicit a service can get if you have to set up too many details of if the user interface is not user friendly. With this in mind, I came up with the name Parker. A recognizable name, simple, and, intuitive. Just how I want the application to be.

I want the application to feel the same on both Android and iOS, but to also keep the platform-specific features, to not lose each unique feel of the platform.

The main features that I want the user to have are:

* Create a premium account
* Show all parking locations
* Book a parking space
* Add booking to phone calendar
* Show all the booked parking spots
* ‘Take me there’ feature
* Show a QR code feature

Creating a premium account

In order to create this feature first, we got to have a login and logout feature. Having a premium account without a default account defeats the purpose of this feature. Creating a premium account is a paid once feature when you create the application. You get this possibility in the register screen where you also get some information about the premium

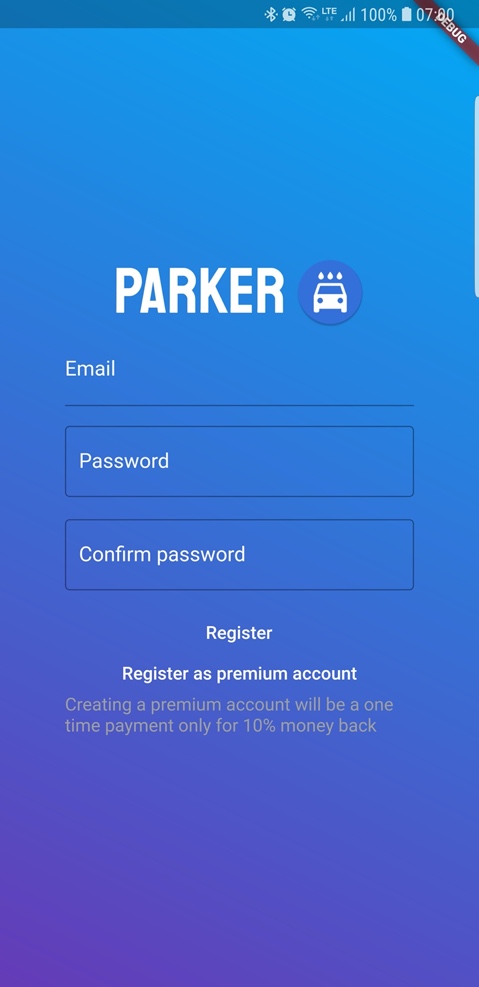
account feature.

A screenshot of a cell phone

Description automatically generated

Figure   
Screenshot of Register Screen

Figure   
Screenshot of Login Screen



Show all parking locations

When showing all the parking locations I wanted the user to be able to easily scroll over all the parking locations that are available in the application. Each location has a name, photo, rating, price, and a hidden location with coordinates. We will use this hidden location in the ‘Take me there’ feature

## 

Figure   
Screenshot of Available Screen

Book a parking space

Each location has a button on it to start a booking for the parking space. There are a few details that we need when booking a parking space. The location, date, time frame, and payment. We use a screen with a tab pager for this. The first tab is the Calendar. Here the user can select any date. When a date is selected the user is sent to the next screen. The second tab the user has to select a start time and an end time for this. The last tab contains the details of the booking and the pay button. When the user selects this button, he is taken to a screen where he inserts his card and pays for it.

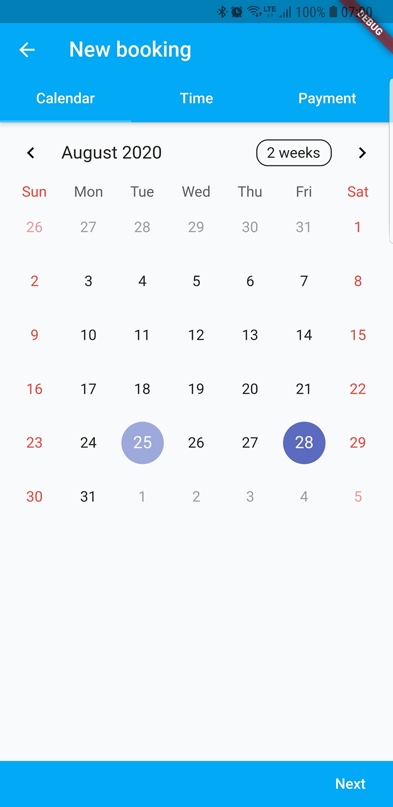
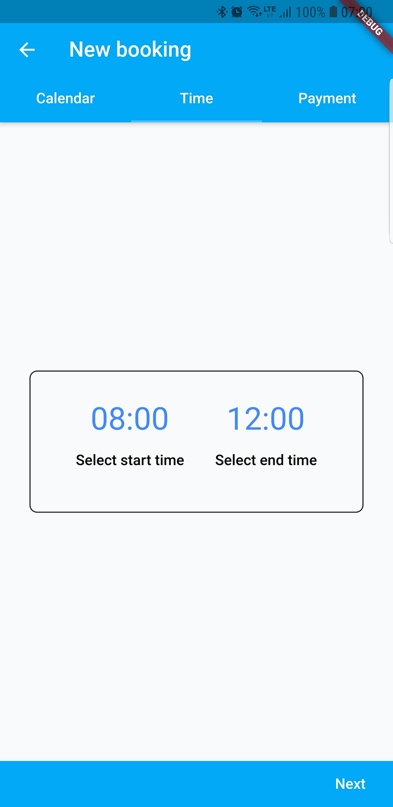
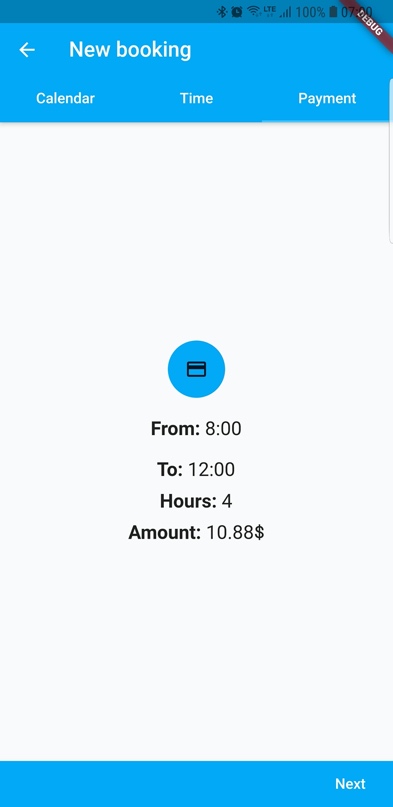
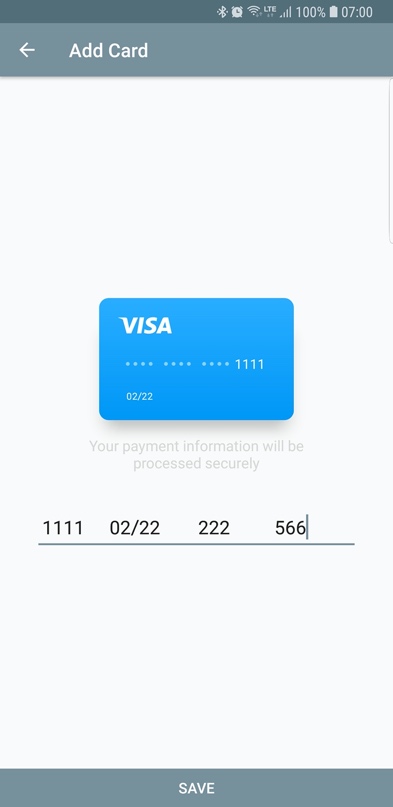


Figure   
Screenshot of Payment Screen

Figure   
Screenshot of Payment Tab

Figure   
Screenshot of Time Picker Tab

Figure   
Screenshot of Calendar Tab

Add the booking to the calendar

When the booking is finished, the user gets a yes-no pop-up which asks him if he wants to add this booking to his calendar. If the user chooses to press yes, the booking is automatically added to his calendar with no additional steps.

## 

Figure   
Screenshot of Successfult Booking

Show all the booked parking spots

After a booking is made, the booking tab will be updated with the user’s new booking. This booking contains all the information that the user needs for his bookings, like location, time, ‘Take me there’ button.

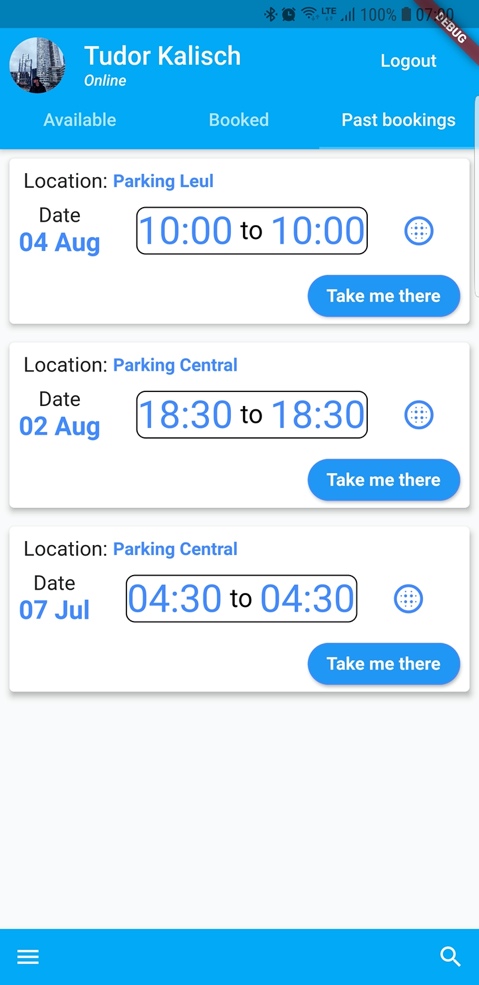
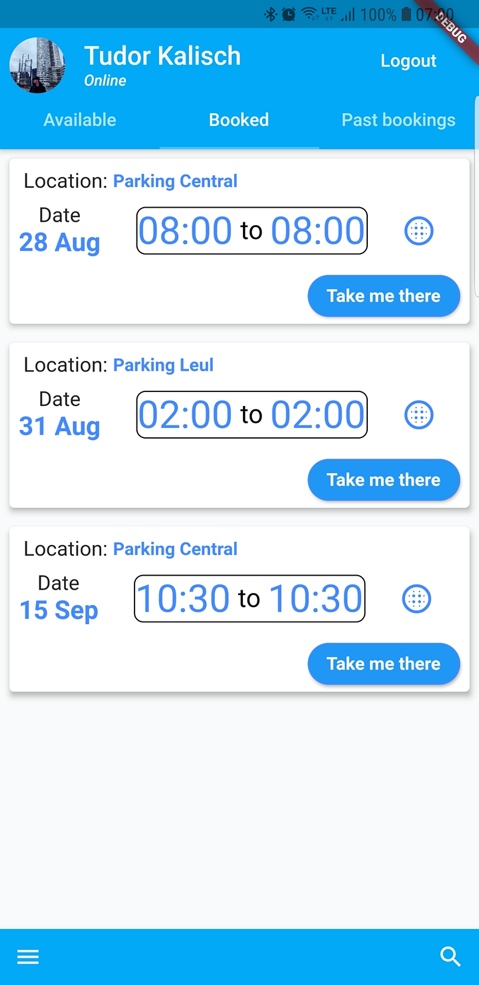


Figure   
Screenshot of Booked Tab

Figure   
Screenshot of Past Booking tab

‘Take me there’ feature

After a booking is made, each separate booking will have a button with the text ‘Take me there’. When the user presses this button, he will be able to select a navigation app on his from (Google Maps, Waze, etc.). This application will receive the coordinates from our application and start a route from the current location to those coordinates.

Show QR code feature

In order to make the user able to use the application to enter the parking space, he can generate a QR code from the booking. Right now, each booking has a specific QR code created from the information of that booking. This can be changed for each location specifications as it is sent from the server and not generated on the client.



Figure Screenshot of QR Code pop-up

# Architecture

## Front-end

For the front-end application, I want to take some time to explain the architecture that is suitable for Flutter, or better said, the lack of one. Because it is a rather new framework, there are no best-practices for the architecture or how the project should be structured. Most articles show small projects or cool tricks that you can do with Flutter, but I couldn’t find an article that actually showed the best architecture in a long-term project with comparisons.

Google has proposed in 2019 the Business – Logic Component (BLoC) as being one of the best architecture choices for Flutter, but it wasn’t yet adopted by the developers as being the best in class in all examples. Currently, on native the most popular architecture choice is Model – View – ViewModel (MVVM), and on Web-based frameworks is BLoC. This might indicate that Flutter is a little closer to Web programming than to native development.

My choice for this project was to get a little close to the BLoC pattern with my application being based on streams and events.

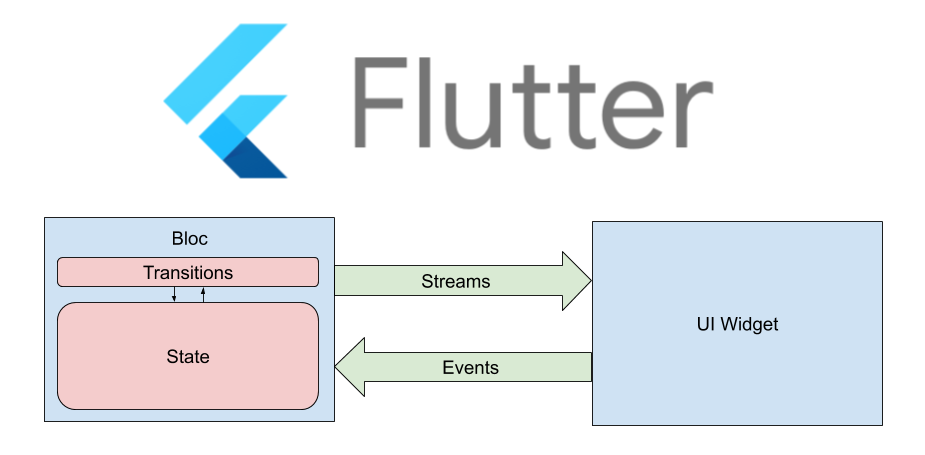


Figure 35  
BLoC pattern diagram

BLoC is essentially a global state manager which holds data and describes a particular state of the application. It also provides an easy way to subscribe to your widgets to the state and implement a logic to modify its state.

I changed the Bloc in the BLoC pattern in order to avoid getting stuck with too many events names which makes the application feel cluttered. The service streams the data to the state. When the state is updated, it sends an event to the widget to update itself. When we have a user interaction, an event is sent to the state which sends a request to the service with the new data. This removes all of the clutter that we can get from BLoC and cleans a little bit the code.

Figure   
Architecture Design diagram

Services



Figure   
`  
Service UML diagram

The services the I created for this application are Login Service, Booking Service and, Location Service, each one with its own separated purpose. Each one of them is using the Spring server to request data and uses Shared Preferences as a persistence feature

Shared Preferences is a feature that is available on Android. With a library, you can use the same feature on both Android and iOS. It creates a key-value storage which is saved on the phone as an XML file on the private folder of the application. You can use it in private or public mode. This means that it is either encrypted or not in order to keep the data safe. It is very simple to use, and it is quite useful to keep data available even after the application is closed.

static String userIdFiled = 'USER\_JWT';

static String isPremiumField = 'IS\_PREMIUM';

SharedPrefs();

static Future<void> setUserId(String id) async {

SharedPreferences sharedPreferences = await SharedPreferences.getInstance();

sharedPreferences.setString(userIdFiled, id);

}

static Future<String> getUserId() async {

SharedPreferences sharedPreferences = await SharedPreferences.getInstance();

return sharedPreferences.getString(userIdFiled);

}

static setIsPremium(bool isPremium) async {

SharedPreferences sharedPreferences = await SharedPreferences.getInstance();

sharedPreferences.setBool(isPremiumField, isPremium);

}

static Future<bool> getIsPremium() async {

SharedPreferences sharedPreferences = await SharedPreferences.getInstance();

return sharedPreferences.getBool(isPremiumField);

}

}

Service Locator is a class created for Dependency Injection (DI). This is used for inversion of control. Instead of each state that uses a service to instantiate each on service, each state uses Service Locator to get an already instantiated object of the type that is required. This feature needs a setup when the application starts.

Login Service is used for anything that is related to authentication. It contains two different methods, Login and Register. With the register, request it is also possible to create a premium or non-premium account.

Location Service is used to get from the server all the data that is needed for each location. Each location has an image, name, rating, price, and coordinates.

Booking Service is used to make a new booking, get upcoming bookings and previous bookings.

## Back-end



Figure   
Backend Architecture Design diagram

As an architecture for the back-end side of the application, I chose a simple MVC, without the view part. Spring recommends for the server MVC as a pattern and the back-end side not being so complex, it doesn’t need an overkill architecture. I used a Controller for exposing the endpoints for requests and a Service to make the connection to the Database.

The Controller is responsible to send the requests along to the Service; the Service queries the data that the Database sends back and manipulates it if it is needed and sends it back to the Controller. I tried to follow the Representational state transfer (REST) constraints as much as possible to make it open for more development.

public interface BookingDataAccessService extends CrudRepository<Booking, UUID>

I have used the CrudRepository interface to create each Service. This creates a repository with all the Create-Read-Update-Delete (CRUD) features that you need to interact with a database. This interface needs a type of object it expects to work with and the kind of key that the database uses. I have chosen to use Universal Unique Identifier (UUID) for each of the primary key because it is a little clearer than using a simple long for the database.

A picture containing drawing

Description automatically generatedDatabase

Figure   
PostgreSQL logo

For the database I have chosen to use PostgreSQL, which is also known as Postgres. It is a free, open-sourced relational database that uses SQL. It is a quite popular database system, and this means that the support online for it is extensive and this also made the decision a little bit easier for the database.

Spring provides a library that helps you connect to the database. You need to provide the address of the database, the username, and the password.

 For the database structure, I have used Hibernate. Hibernate is an Object Relational Mapping tool. Basically, it creates the database structure for you. Hibernate enabled you to develop classes in an Object-oriented manner, including inheritance, polymorphism, composition, etc.

Figure   
Hibernate logo

@Entity  
@Table(name = "booking")  
public class Booking {  
 @Id  
 @GeneratedValue(generator = "UUID")  
 @GenericGenerator(  
 name = "UUID",  
 strategy = "org.hibernate.id.UUIDGenerator"  
 )  
 UUID id;  
 @ManyToOne  
 @JoinColumn(name = "location\_id")  
 Location location;  
 UUID userId;  
 long startTime;  
 long endTime;  
 @Column(length = 1500000)  
 String qr;  
  
 public Booking() {  
 }

 After some tweaking, this is the database relationship model that I used for the database structure.

Figure   
Database design diagram

## Experiences working with Flutter and Spring

Flutter

Building an application with Flutter is certainly faster than building two native applications native, sometimes faster than one simple native solution. For instance, moving from one screen to another is quite simple using a Navigator class, which was made available on Android from 2019.

Function logout(BuildContext context) {

SharedPrefs.setUserId("");

return () => {

Navigator.of(context)

.push(MaterialPageRoute(builder: (context) => LoginView()))

};

}

Using features that are platform-specific work seamless, with no extra effort needed for different platforms. For instance, the feature to select time needs to work differently on each platform. iOS has a specific time picker which is familiar to the user and that is the one you should you, on the other hand, Android lets you decide on how your time picker should look like. When I used the time picker provided by Flutter it automatically used the specific time picker for each platform

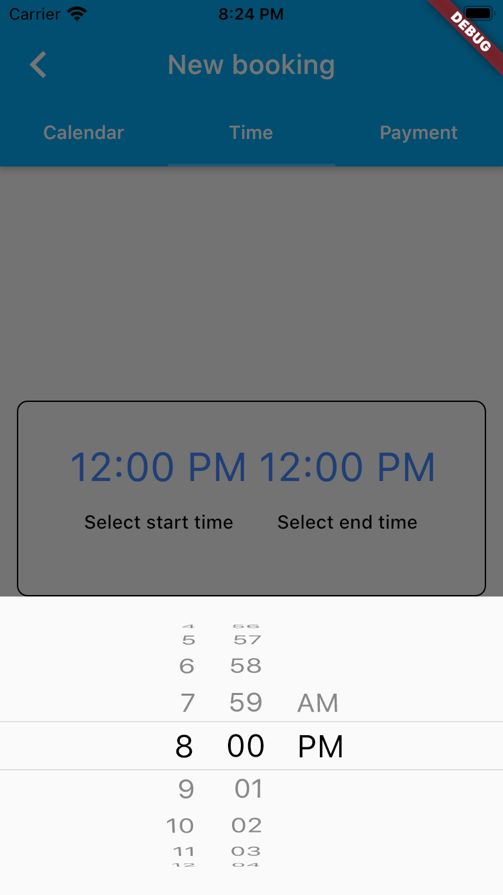
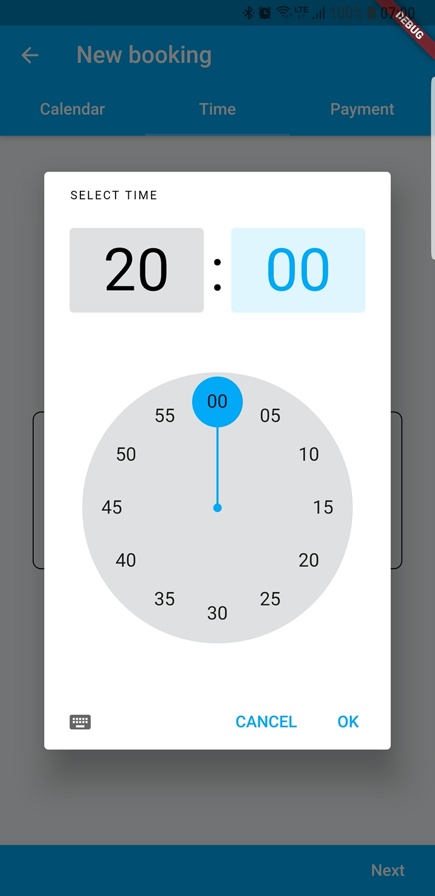


Figure   
Android screenshot of time picker

Figure   
iOS screenshot of time picker

The Hot-Reload feature from Flutter makes developing on mobile a lot faster. I am more familiar with Android development, where if you work on a big project, each build can get up to 5 minutes to work. This makes every test to take at least 5 minutes to try. With Hot-Reload, you just have to save the file and the changes are automatically rendered on UI. This speeds up the development of the application. This is quite a missed feature on Android.

There were a few things that didn’t work out of the box with Flutter. Every configuration needs to be changed for each platform, Flutter, Android, and iOS. I found that it is not quite easy to do that, and you need some general knowledge about each platform when you want to work with configurations. For instance, in the early phases of the project, the name of the application was simple ‘parking\_app’, as I was not exactly sure what the perfect name is for the application. When I changed the name of the application to Parker, I had to go into each project configuration and change the name, but to also change some of the scripts to work with the new name of the application. This caused the application to stop building until I found each config file that used that name. A script build for Flutter which automatically does that for you would be best. This was the same when changing the icon for the application. iOS needs some specific steps and Android needs other specific steps. It is easy to find out how to do each one of these tasks, but it would be much easier to just use a feature from Flutter.

After changing the name of the application, I have also changed the application id in Android to keep it streamlined. After I did that change, the ‘Clean project’ feature form Visual Studio Code stopped working which made building the application quite annoying, because every time I started a build, I had to use the Clean Project feature from Android Studio in order to get it working.

Backend

I enjoyed the annotation-based approach that Spring uses for its project. It is easily configurable, and it is not difficult to learn what each annotation does. This makes to project a lot cleaner with little code, and it works out of the box with the help of Spring Boot Initializr.

A screenshot of a cell phone

Description automatically generatedSpring Boot Initializr is a website where you can configure your project. You need to provide some information about your project. The name, type of project, type of packaging, language, and spring boot version. Then you can select all the dependencies that you want to use. The website creates a zip file where you have the project configured and ready to use.

Figure   
Screenshot of Spring Initializr

The creation of the QR code was done quite easily and it can be reconfigured to work with different types of barcodes. I have decided to send directly the barcode as an Base64 image to remove the complexity the frontend.

ByteArrayOutputStream stream = QRCode  
 .*from*(barcodeText)  
 .withSize(250, 250)  
 .stream();  
ByteArrayInputStream bis = new ByteArrayInputStream(stream.toByteArray());  
String QR = Base64.*getEncoder*().encodeToString(bis.readAllBytes());  
return QR;

The only struggles that I encountered with the backend was working with Hibernate. Having a framework that automatically creates the relationships of the database is quite useful, but it is not so intuitive, and you might want to make some changes in the database structure to match your project and its quite hard to do that. Setting up one-to-many relationships between tables in the database is also not so straight-forward. But with some experience with the framework, you can remove some of the repetitive work that you need to do when configuring the database.

# 

# Conclusion

The goal of this application was to build a solution for the parking issue in Cluj-Napoca. Having a solution with which you can have the assurance of a parking space wherever you decide to go in Cluj-Napoca was met with all the features that this application provides.

Going hybrid on the front-end side of the application was the better solution in this case. The source code is the same for both iOS and Android and this made it possible to build the application at the same time for both platforms. Even with the difficulties encountered during the configuration for both platforms, Flutter provides many features that help you build an application for both iOS and Android that still has the feeling of a native application. The speed of the developing process needs to be also taken into consideration, as it is almost two times as fast to build a hybrid application rather than two separate ones.

Dart is a strong programing language that has all the freedom of JavaScript and TypeScript and the long-term scalability of Java and C#. It was easy to learn and use and it was never an hinderance during the development.

Spring was a better choice at the start. The Spring Boot Initializr speed up the process of configuring the project and Hibernate was a good choice for creating the relationships needed for the database. Spring is scalable, so changes that are needed for future features will be easily added.

There are some features that could get the project to the next level. A service where the parking locations can subscribe to and update their location details in real time. This will need an endpoint made on the server from which an application could receive data about the new bookings made and update the information showed at the location in real time.

The goal of this project was to find better solution to the parking issue in Cluj-Napoca. I think that with all the features developed for this application this goal was reached. There is still more work to be done until this problem is fixed, but I like to think this can bring us a little closer to that goal as a city.

# Bibliography

Articles and books

1. Wu W. React Native vs Flutter, Cross-platforms mobile application frameworks.
2. Bracha G. The Dart programming language. Addison-Wesley Professional; 2015.
3. Gosling J, Joy B, Steele G, Bracha G. The Java language specification. Addison-Wesley Professional; 2000.
4. Barton JJ, Nackman LR. Scientific and engineering C++. Massaschusetts; 1995.
5. Sabyasachi Mohanty, DART Evolved for Web - A Comparative Study with JavaScript; January 2014
6. Tran, Thanh Flutter Native Performance and Expressive UI/UX; 2020)
7. Gargenta M. Learning android. " O'Reilly Media, Inc."; 2011 Mar 10
8. Allan A. Learning iOS Programming: From Xcode to App Store. " O'Reilly Media, Inc."; 2013 Mar 12.
9. Eisenman B. Learning react native: Building native mobile apps with JavaScript. " O'Reilly Media, Inc."; 2015 Dec 3.
10. Walls C. Spring Boot in action. Manning Publications; 2016 Jan 3.
11. Kochan SG. Programming in objective-C. Addison-Wesley Professional; 2011 Jun 10.

Electronic

1. What’s Revolutionary about Flutter

URL: <https://hackernoon.com/whats-revolutionary-about-flutter-946915b09514>

Accessed Aug 17, 2020

1. Official documentation of Flutter Navigator

URL: <https://docs.flutter.io/flutter/widgets/Navigator-class.html>

Accessed Aug 20, 2020

1. Flutter FAQ

URL: <https://github.com/hemanthrajv/fludex>

Accessed Aug 20, 2020

1. HTML5 Biggest mistake

URL: <https://mashable.com/2012/09/11/html5-biggest-mistake/?europe=true>

Accessed Aug 20, 2020

1. Stats about mobile devices used in Romania

URL: <https://gs.statcounter.com/os-market-share/mobile/romania/#quarterly-201901-202001-bar>

Accessed Aug 13, 2020

1. Visual Studio Code

URL: <https://code.visualstudio.com/>

Accessed Aug 13, 2020

1. Visual Studio

URL: <https://visualstudio.microsoft.com/>

Accessed Aug 14, 2020

1. Android Studio

URL: <https://developer.android.com/studio/>  
Accessed Aug 14, 2020

1. Xcode

URL: <https://developer.apple.com/xcode/>

Accessed Aug 15, 2020

1. YeParking

URL: <https://yeparking.ro/>

Accessed Aug 08, 2020

1. Parking Cluj

URL: <https://play.google.com/store/apps/details?id=com.arobs.parking.cluj&hl=en>

Accessed Aug 08, 2020

1. TPark

URL: <https://www.tpark.ro/>

Accessed Aug 08, 2020

# Electronic resources

<https://upload.wikimedia.org/wikipedia/commons/thumb/c/ca/IOS_logo.svg/1024px-IOS_logo.svg.png>

<https://upload.wikimedia.org/wikipedia/commons/thumb/8/82/Android_logo_2019.svg/1173px-Android_logo_2019.svg.png>

<https://logos-download.com/wp-content/uploads/2016/10/Java_logo_icon.png>

<https://commons.wikimedia.org/wiki/File:Kotlin-logo.svg>

<https://seeklogo.com/vector-logo/340070/objective-c>

<https://en.wikipedia.org/wiki/Swift_(programming_language)#/media/File:Swift_logo_with_text.svg>

<https://upload.wikimedia.org/wikipedia/commons/thumb/1/1e/Xcode_Icon.png/300px-Xcode_Icon.png>

<https://commons.wikimedia.org/wiki/File:Unofficial_JavaScript_logo_2.svg>

<https://upload.wikimedia.org/wikipedia/commons/f/fe/Dart_programming_language_logo.svg>

[https://www.google.com/imgres?imgurl=https%3A%2F%2Fmiro.medium.com%2Fmax%2F700%2F1\*TkNd1PwwwdBi9Z3kdG5Hng.png&imgrefurl=https%3A%2F%2Fmedium.com%2Fflutter-community%2Fwhy-native-developers-should-pay-close-attention-to-flutter-e6aa9ce5c922&tbnid=3Fa2m4wjUEW3qM&vet=12ahUKEwjxpKmWh6jrAhWQLOwKHWBVDtEQMygDegUIARClAQ..i&docid=08co2qcBgQ3cEM&w=700&h=366&q=flutter%20logo%20png&ved=2ahUKEwjxpKmWh6jrAhWQLOwKHWBVDtEQMygDegUIARClAQ](https://www.google.com/imgres?imgurl=https%3A%2F%2Fmiro.medium.com%2Fmax%2F700%2F1*TkNd1PwwwdBi9Z3kdG5Hng.png&imgrefurl=https%3A%2F%2Fmedium.com%2Fflutter-community%2Fwhy-native-developers-should-pay-close-attention-to-flutter-e6aa9ce5c922&tbnid=3Fa2m4wjUEW3qM&vet=12ahUKEwjxpKmWh6jrAhWQLOwKHWBVDtEQMygDegUIARClAQ..i&docid=08co2qcBgQ3cEM&w=700&h=366&q=flutter%20logo%20png&ved=2ahUKEwjxpKmWh6jrAhWQLOwKHWBVDtEQMygDegUIARClAQ)

<https://estradaci.com/wp-content/uploads/2019/04/asp-net-core-logo.png>

<https://commons.wikimedia.org/wiki/File:Visual_Studio_2017_Logo.svg>

<https://en.wikipedia.org/wiki/Spring_Framework#/media/File:Spring_Framework_Logo_2018.svg>

<https://upload.wikimedia.org/wikipedia/commons/d/d5/IntelliJ_IDEA_Logo.svg>

<https://yeparking.ro/>

<https://play.google.com/store/apps/details?id=com.arobs.parking.cluj&hl=en>

<https://www.tpark.ro/>

[https://gs.statcounter.com/os-market-share/mobile/romania/#quarterly-201901-202001-bar](https://gs.statcounter.com/os-market-share/mobile/romania/" \l "quarterly-201901-202001-bar)

<https://hibernate.org/images/hibernate-logo.svg>

<https://start.spring.io/>