

DEVELOPMENT OF RESTAPOINTS MOBILE APPLICATION

Author: Kristian Tuusjärvi

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala			
Koulutusohjelma/Tutkinto-ohjelma Tietotekniikan koulutusohjelma			
Työn Tekijä(t) Kristian Tuusjärvi			
Työn nimi RestaPoints Mobiilisovelluksen Kehittäminen			
Päiväys	25.04.2019	Sivumäärä/Liitteet	32
Ohjaaja(t) Lehtori Keijo Kuosmanen, Lehtori Jussi Koistinen			
Toimeksiantaja/Yhteistyökumppani(t) Tuusplan Oy			
<p>Tiivistelmä</p> <p>Tämä opinnäytetyö käsittelee RestaPoints mobiilisovelluksen kehittämistä. RestaPoints on pienille ravintoloille suunnattu bonus piste mobiilisovellus. Työ toteutettiin Tuusplan Oy:n toimeksiantona. Työn tarkoituksena oli arvioida olemassa olevia järjestelmäriippumattomia ohjelmistokehyksiä ja kehittää niistä parhaalla RestaPoints mobiilisovellus. Järjestelmäriippumattoman ohjelmointikehyksen käyttö oli tärkeää ajan säästämiseksi ja kulujen laskemiseksi.</p> <p>Tämä opinnäytetyö koostuu testausosasta ja RestaPoints mobiilisovelluksen kehittämisosasta. Testausosa sisältää tietoa kolmesta testatusta järjestelmäriippumattomasta ohjelmistokehyksestä, niiden vertailusta ja niihin kohdistuvista vaatimuksista. Vertailussa päädyttiin käyttämään Flutter ohjelmointikehystä RestaPoints mobiilisovelluksen kehittämiseen. Kehittämisosassa käydään läpi sovelluksen tärkeimmät ominaisuudet ja arkkitehtuuri.</p> <p>Lopputuloksena on toimiva mobiilisovellus, jonka avulla ravintoloiden asiakkaat pystyvät etsimään ravintoloita, keräämään pisteitä ja paljon muuta. Sovellusta tullaan jatkokehittämään, jos sovellukselle löytyy tarpeeksi asiakkaita.</p>			
Avainsanat React Native, Flutter, Xamarin Forms, Järjestelmäriippumaton mobiilisovellus, RestaPoints			
Kieli Englanti			

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Kristian Tuusjärvi			
Title of Thesis Development of RestaPoints Mobile Application			
Date	25 April 2019	Pages/Appendices	31
Supervisor(s) Mr Keijo Kuosmanen, Senior Lecturer and Mr Jussi Koistinen, Senior Lecturer			
Client Organisation /Partners Tuusplan Oy			
<p>Abstract</p> <p>The topic of this thesis was to develop a mobile application called RestaPoints, commissioned by Tuusplan Oy. RestaPoints is a bonus point application that is targeted at small restaurants. Three cross platform mobile development frameworks were evaluated for the development. The best framework was Flutter and it was chosen for the development. Only cross platform frameworks were evaluated to save time and keep costs low.</p> <p>The first part of this thesis covered evaluating three cross platform frameworks. The best framework was Flutter and it was chosen for the development of the RestaPoints mobile application. The second part focused on the development of RestaPoints mobile application, its most important features and architecture.</p> <p>As a result of this thesis a functional RestaPoints mobile application was developed. RestaPoints allows restaurant's customers to find the restaurant, collect bonus points and much more. RestaPoints will be developed in the future if enough customers are interested in it.</p>			
Keywords React Native, Flutter, Xamarin Forms, Cross platform mobile application, RestaPoints			
Language English			

1	INTRODUCTION	5
2	DEVELOPMENT FRAMEWORKS	6
2.1	React Native	7
2.1.1	Advantages	8
2.1.2	Disadvantages.....	9
2.2	Xamarin Forms	10
2.2.1	Advantages.....	11
2.2.2	Disadvantages.....	12
2.3	Flutter	13
2.3.1	Advantages.....	15
2.3.2	Disadvantages.....	16
2.4	Evaluating Frameworks	16
3	SUPPORT TOOLS	18
3.1	Git & GitHub	18
3.2	Visual Studio Code	19
3.3	OneNote.....	20
4	DEVELOPING RESTAPOINTS.....	20
4.1	Views	20
4.2	Routing	24
4.3	Animations	25
4.4	Plugins	25
4.5	State management.....	26
4.6	Https calls	26
4.7	Studying.....	28
5	CONCLUSION	29
6	REFERENCES.....	30

Terms

AI: Artificial Intelligence

AOT: Ahead of time compilation

API: Application Programming Interface

BLOC: Business Logic Component

FPS: Frames Per Second

IL: Intermediate Language

JIT: Just in time compilation

LINQ: Language Integrated Query

LLVM: Low Level Virtual Machine

MVVM: Model View View Model

MVX: Model View Controller

NDK: Native Development Kit

NPM: Node Package Manager

OEM: Original Equipment Manufacturer

RN: React Native

RUNTIME: The environment where Program runs

SDK: Software Development Kit

Thread: Processor instruction. Programs can split themselves up to multiple threads to allow multiple tasks to be ran at same the time.

UI: User Interface

XAML: Extensible Application Markup Language

1 INTRODUCTION

This thesis studies three different cross platform frameworks and goes through building an application with one of them. The frameworks are Xamarin Forms, React Native and Flutter. The application is for gathering bonus points from restaurant visits, it is called RestaPoints.

RestaPoints is a bonus point platform for restaurants. The basic idea is for restaurants to have the ability to make their own bonus point program. Restaurants can reward their customers with points that can be exchanged for drinks and meals. Bonus programs are widely used in store chains such as S-group and Kesko. They are also in use with some big restaurant chains like Subway and Hesburger. However, these bonus point programs are expensive to develop and upkeep. Smaller restaurants cannot afford to have them. RestaPoints attempts to solve this problem, giving any restaurant the platform, on which build their own bonus point program.

The idea for RestaPoints was born on a Product development course at Savonia University of Applied Sciences. The members of RestaPoints the team are Kristian Tuusjärvi, Moona Partanen, Tapio Riihimäki and Tuija Simonen. The team received good feedback from teachers and decided to test their idea in Savonia Draft Program. The Savonia Draft program is an innovation competition where teams present their business ideas to a jury that will then pick four teams to get rewarded. The jury liked RestaPoints and rewarded RestaPoints team with 1 000€ of investment money.

The RestaPoints platform consists of three parts: 1) Mobile application to enable customers to gather points, 2) a web application so restaurants can control their bonus program, and finally 3) a backend that serves data to the previously mentioned two. Mobile application is developed with Flutter, the web application with C# .NET Core 2.0. The backend uses MySQL, NGINX and Ubuntu. It will be done by Tapio Riihimäki. He is writing his own bachelors' thesis on it. This thesis focuses on the development of the mobile application.

2 DEVELOPMENT FRAMEWORKS

Xamarin Forms, React Native and Flutter will be tested in this thesis. All the frameworks are open source and free to use. There are no plans for a web-based version of the RestaPoints application, so no web-view based cross platform framework was tested. The goal is to find the best-suited framework to develop RestaPoints. Small sample applications will be developed with each framework, to determine the most suitable framework. However, due to the limitations in time these sample applications are not identical and so cannot be compared directly with benchmarking. When choosing a framework, the focus is on few key areas: the developer, performance, technology, documentation and ease of use.

2.1 React Native

React Native is an open source cross platform framework developed by Facebook and other contributors. React Native was born at Facebook's Hackathon event in the summer of 2013 – the first official version, v0.5.0, was introduced to the public in 2015 at React.js Con. (Sekulić, 2016)

With React Native it is possible to build mobile applications using JavaScript and JSX instead of the traditional Swift/Objective C for iOS and Kotlin/Java for Android. JSX is an extension for JavaScript syntax. It is like HTML. When running a React Native application it converts its own elements to platform specific counterparts. For example: The Text element of React Native changes on Android to TextView and on iOS to UIView containing text. So, it is possible to write close to native applications with JavaScript. (Evkoski, 2017)

React Native has two very important threads, one for React Native's JavaScript engine and one for native code. (FIGURE 1). Native codes thread is called Main thread. The Main thread draws the user interface and processes user gestures. The JavaScript thread handles the business logic of the application, things like functions and the interface structure. These threads do not talk to each other directly, so as not to block each other's work. Between the threads there is a bridge that relays messages asynchronously from one thread to another. (Evkoski, 2017)

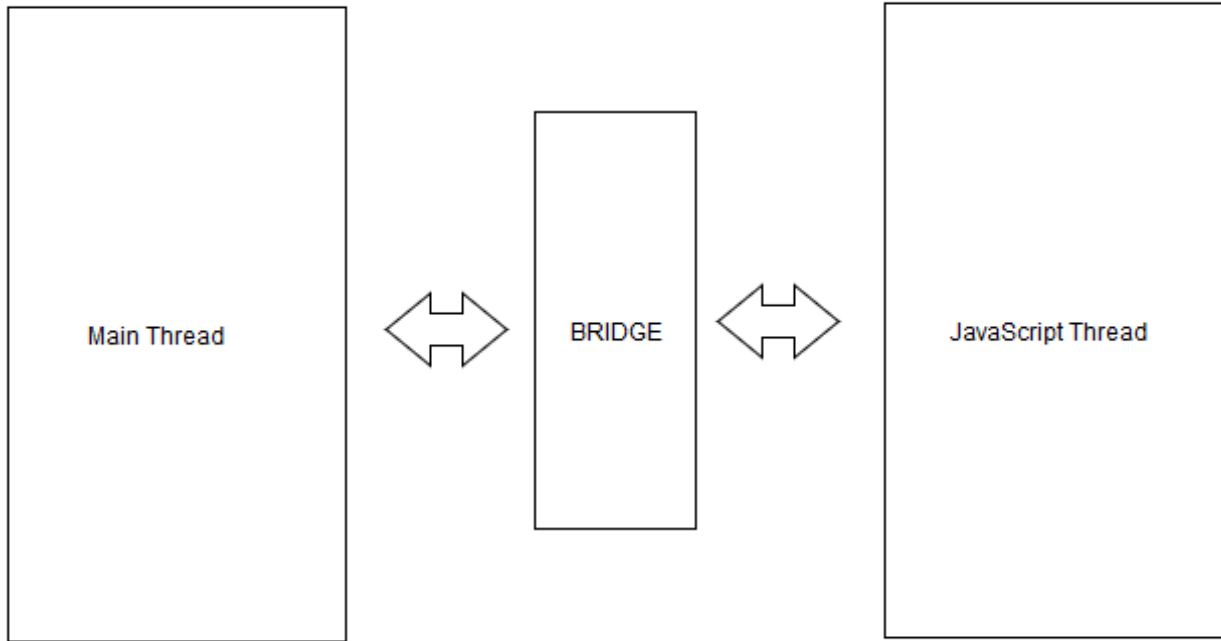


FIGURE 1. Bridge between threads

A popular development model for React Native is Redux. Redux works in three parts: actions, a store and reducers. (FIGURE 2). Actions are events dispatched from the application to reducers. Reducers hold the current state of the application, like the model in MVC. Store is a container for the state, it connects all the reducers. When pushing a button, action is dispatched to reducer and reducer then changes the state of the application updating the store. (Levkovsky, 2017)

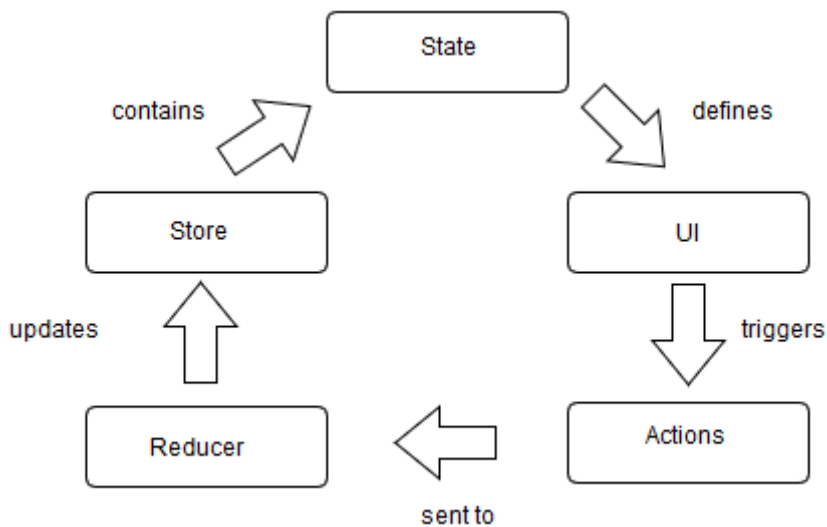


FIGURE 2. Redux pattern workflow

2.1.1 Advantages

React Native has similarities to already popular web technologies, which makes it easy for web developers to pick up and start making mobile applications with. This has allowed React Native-project to grow to one of the biggest open source projects in Git. The large community also creates a lot of third-party plugins which is all the better for the end user. React Native syntax has many similarities to HTML tags and CSS styles. (FIGURE 3). (Sekulić, 2016)

```

01. import React, { Component } from 'react';
02. import { Text, View } from 'react-native';
03.
04. export default class HelloWorldApp extends Component {
05.   render() {
06.     return (
07.       <View style={{ flex: 1, justifyContent: "center", alignItems: "center" }}>
08.         <Text>Hello world!</Text>
09.       </View>
10.     );
11.   }
12. }

```

FIGURE 3. React Native example code

Development with native code takes a lot of time, companies need to hire two teams for the two major platforms: Android and iOS. Having two teams is very expensive. React Native is based on JavaScript. It only needs one code base. There is no need for two teams anymore, which save time and money. Because React Native uses JavaScript it has access to the world's largest package ecosystem: NPM. NPM has thousands of ready to use packages that developers can add to their applications and further speed up the development. React Native is used in many big companies such as Skype, Facebook, Bloomberg and many more. (Chrzanowska, 2019)

"We built the very same app with both React Native and Swift. The latter took as much as 33% more time to build and still was working only on iOS – with no Android version. Significant savings can come with developing for more than one platform. As RN lets you share a big part of codebase between operating systems within a few hours, you can potentially save time and money." (Chrzanowska, 2019)

Hot reloading enables React Native developers to update the application while it is running. This feature does not require rebuilding the application and saves time. Hot reload cuts the normal 30 – 60 seconds development cycle down to just couple of seconds. (Chrzanowska, 2019)

2.1.2 Disadvantages

React Native is supported by many big companies, but developers still run into problems with new versions not being compatible with older versions. Debugging has weird issues and hot reload might stop working out of the blue. It's frustrating trying to figure out the reasons behind these bugs and it takes a lot of time away from development. Time lost equals money lost. Bug and other weird issues are partly caused by the third-

party packages that have not been fully developed or lack support for bug fixing. There still is need for Android and Swift developers since it is not possible to use native functionalities with JavaScript. If there is not a package already for something, then it needs to be built from the ground up using native languages. (Chrzanowska, 2019)

Facebook is the company behind React Native and while Facebook is not a small company that would disappear overnight, it also is not a software development company. This is a concern for the developer, because nobody knows what Facebook is going to do with React Native. There is not a clear road map and React Native still hasn't been released out of beta version after years of development. (Williams, 2017)

JavaScript is the main language behind React Native. JavaScript is one of the most popular languages out there, yet it has some big flaws. For example: integer type is missing, typing is loose, automatic semicolon "feature" in which JavaScript will insert a semicolon if you have forgotten, global variables cause confusion, scope rules are bad, execution fails without error messages, JavaScript does not scale well to large applications, async programming is hard, there is no lambdas, and finally, JavaScript is slow to evolve. Because of these inconsistencies JavaScript can be a hard language to learn. It takes a lot of time to track down the weird bugs JavaScript forces developers into making. That is why JavaScript is considered by some to not to be a proper programming language like C#, C++ and Java. (Eng, 2016)

2.2 Xamarin Forms

Xamarin Forms was introduced in 2014, allowing Xamarin developers to develop from one code base to Android, iOS and Windows phone (Friedman, 2014). In 2016 Microsoft acquired Xamarin and open sourced its SDK (Guthrie, 2016).

Xamarin Forms is a shared UI abstraction layer on top of Xamarin Traditional, Xamarin Traditional gives developers a full access to underlying Android, IOS and UWP APIs. (FIGURE 4). Xamarin has a cross-platform compiler which takes C# .NET Xamarin code and compiles it depending on the platform: on Android with JIT compiling at application launch, on IOS AOT compiling to an ARM assembly code and on Windows using IL compiling. Xamarin Forms uses C# .NET framework for business logic and XAML for UI. XAML is close to xml so developers with Android background might feel at home with it, nevertheless it has a very mild learning curve. (Burns; Britch; Umbaugh; & Dunn, 2017) Xamarin Traditional allows around 60 to 70% code reusability while Xamarin Forms enables around 90-95% code reusability. However, this comes with a performance cost as Xamarin Forms is slower than native Android or IOS. (Williams, 2017)

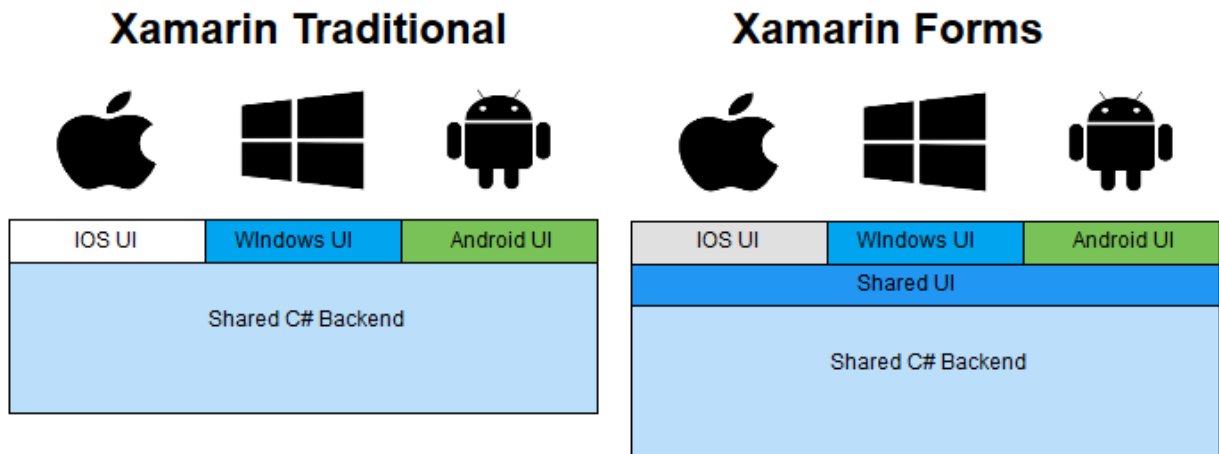


FIGURE 4. The difference between Xamarin Traditional and Xamarin Forms

Xamarin Forms is easy to pick up for people that have experience with C# frameworks, but harder for people who are used to web languages such as JavaScript. Development is done with Visual Studio which supports Android, IOS and Windows emulators. Like React Native with its hot Reload, Xamarin Forms has a previewer, which does what it says. It previews the UI to the developer, so there is no need to run the app. This feature is very helpful, as it can take up to a minute to run the application, depending on the size.

The recommended development model for Xamarin Forms is MVVM. (FIGURE 5), it is easy to learn and very powerful. MVVM is composed of model, view and view model. The view is the UI structure made with XAML markup language. (FIGURE 6), in XAML there are bindings that connect the view to the view model. The View model has properties and commands that the view can bind to. The view model implements functionalities and view displays them in the UI. The model has the data of the application and the corresponding validation. (Britch, Dunn, & Poulad, 2017)

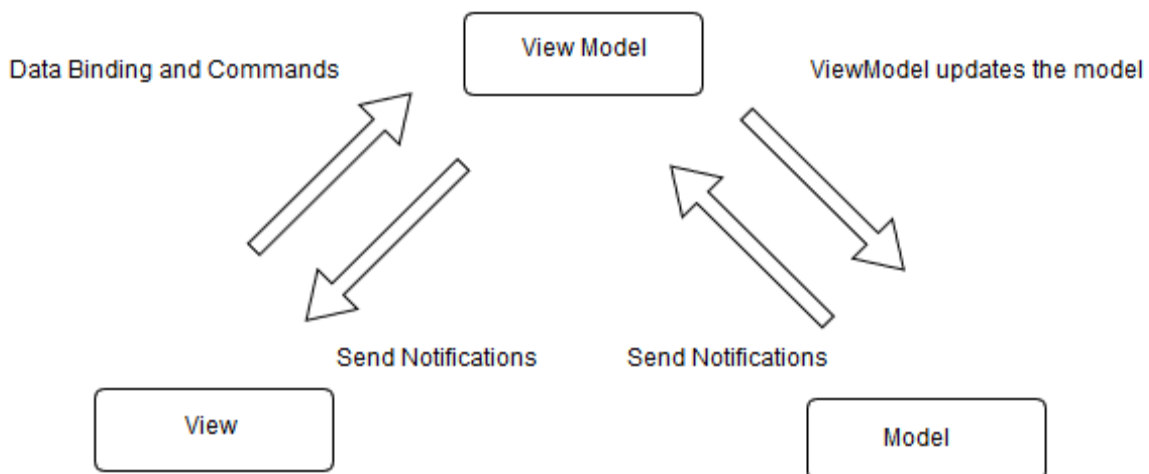


FIGURE 5. MVVM pattern

```

01. <?xml version="1.0" encoding="UTF-8"?>
02. <CarouselPage xmlns="http://xamarin.com/schemas/2014/forms"
03.             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
04.             x:Class="CarouselPageNavigation.MainPage">
05.     <ContentPage>
06.         <ContentPage.Padding>
07.             <OnPlatform x:TypeArguments="Thickness">
08.                 <On Platform="iOS, Android" Value="0,40,0,0" />
09.             </OnPlatform>
10.         </ContentPage.Padding>
11.         <StackLayout>
12.             <Label Text="Red" FontSize="Medium" HorizontalOptions="Center" />
13.             <BoxView Color="Red" WidthRequest="200" HeightRequest="200" HorizontalOptions="Center" VerticalOptions="CenterAndExpand" />
14.         </StackLayout>
15.     </ContentPage>
16.     <ContentPage>
17.         <ContentPage.Padding>
18.             <OnPlatform x:TypeArguments="Thickness">
19.                 <On Platform="iOS, Android" Value="0,40,0,0" />
20.             </OnPlatform>
21.         </ContentPage.Padding>

```

FIGURE 6. XAML language example

2.2.1 Advantages

Xamarin uses C# as its main language, which is a popular language. It's possible to develop with C# almost everything; here is a list of examples: client applications, libraries and components, web applications, web APIs, mobile apps, backend, Azure cloud applications, database using ML/Data tools, Office, SharePoint, SQL, AI, block chains, games, and much more (Chand, 2016). Almost every developer has encountered C# in their studies or in work - it is literally everywhere. Even if a developer has not encountered C# before, it is very similar to Java and other popular languages, so it is easy to learn.

With Xamarin Forms it is possible to write Android, iOS and Universal Windows applications. These three platforms account for 91% of all desktop and mobile devices (Statcounter, 2018). This means that there is only need for one programming language and one development team that can develop for all the major platforms from single code base. The savings are huge for companies that have many applications and programs running on different platforms. (Leuschenko, 2018)

"Over a 3-year period, an organization with multiple mobile applications can expect: \$1,365,003- reduction in mobile application development costs using shared C# code base across platforms, \$829,475- mobile app maintenance and upgrade efficiency gains using Xamarin for Visual Studio's shared code base, \$6,558,360- savings on platform-specific mobile application developer expenses." (Leuschenko, 2018)

Xamarin has been around for a long time. Visual Studio is the development environment for Xamarin. Visual Studio is one of the best if not the best development environments that there is. It has emulators, UI designer, nugget packages, integrated testing and much more, making it unrivalled compared to any other IDE. (Williams, 2017) Because Xamarin has been around for over ten years, there is well designed working plugins and packages for it. There is even online Xamarin University, where developers can start as beginners and study their way to a professional.

As previously mentioned, Xamarin Forms uses C# .NET framework. (FIGURE 7). C# has many benefits to it, for example: type-safety, automatic garbage collection, namespaces, scalability, static typing, lambda support, generic support, LINQ support, consistency, visual studio development environment, and finally, great performance. (Green, 2017)

```

03. namespace HelloWorld
04. {
05.     class Hello
06.     {
07.         static void Main()
08.         {
09.             Console.WriteLine("Hello World!");
10.
11.             // Keep the console window open in debug mode.
12.             Console.WriteLine("Press any key to exit.");
13.             Console.ReadKey();
14.         }
15.     }
16. }

```

FIGURE 7. Example of C# code

2.2.2 Disadvantages

Xamarin Forms is open source framework, so it is free to use, but Visual Studio is not. There are three versions of Visual Studio. Students, academics, researchers and small non-enterprise teams can use Visual Studio Community for free (Microsoft, 2018). Small companies need to buy the professional version which is \$1,199 per year. Enterprise companies; "250 PCs or >\$1 Million US Dollars in annual revenue" (Microsoft, 2019) need to use Enterprise version of Visual Studio, which is the most expensive version, costs 5999\$ per year (Microsoft, 2019). Other than Visual Studio there really is not a good development platform for Xamarin.

Because Xamarin is relying on native SDKs. Changes that happen to the native SDKs won't be immediate in Xamarin's SDK. Xamarin's development team first needs to implement the changes to their own SDK. This means that it will take some days before changes are available to Xamarin users. (Azilen, 2017)

It is easy to build Xamarin Forms applications badly, which results in slowly performing applications. Reading the documentation and following the good practices it is the best thing that a beginner developer can do. (Williams, 2017)

2.3 Flutter

Flutter is an open source cross platform framework developed by Google. Flutter uses Dart as its programming language. The first public version of Flutter was presented to the world in 2015 Dart developer summit (Amadeo, 2018). In year 2018, Flutter team released 1.0 version, the first "stable" version (Google,

2018). Flutter can be used on Android and iOS, with up to 95% code reusability (Sarvaiya, 2018). Flutter can also be used in the web with around 60-70% code reusability (Leler, 2018).

Flutter uses Dart programming. (FIGURE 8), which is perhaps the only mainstream programming language that supports JIT and AOT compiling. Because of this, Flutter can be compiled with JIT in development mode and with AOT in production. JIT enables Flutter's hot reload feature. Hot reload allows developers to change code and reload without rebuilding the application. Hot reload decreases the development cycle below one second. AOT compiling in production adds better performance and removes the need for JavaScript-style Bridge. (Leler, 2018)

```
01. import 'package:flutter/material.dart';
02.
03. void main() => runApp(MyApp());
04.
05. class MyApp extends StatelessWidget {
06.   @override
07.   Widget build(BuildContext context) {
08.     return MaterialApp(
09.       title: 'Welcome to Flutter',
10.       home: Scaffold(
11.         appBar: AppBar(
12.           title: Text('Welcome to Flutter'),
13.         ),
14.         body: Center(
15.           child: Text('Hello World'),
16.         ),
17.       ),
18.     );
19.   }
20. }
```

FIGURE 8. Flutter example code

Flutter does not use OEM widget or native controls. Flutter draws its UI itself with its high-performance rendering engine Skia. Flutter does not depend on native UI systems to talk to the graphics hardware. (FIGURE 9). This means that it performs better than other cross platform frameworks, and has a lot more flexibility, giving developers and designers more freedom. Flutter has its own widgets that are pixel perfect copies of Android and iOS. These widgets also have the same physics, so scrolling and tapping feels native. (Stoll, 2018)

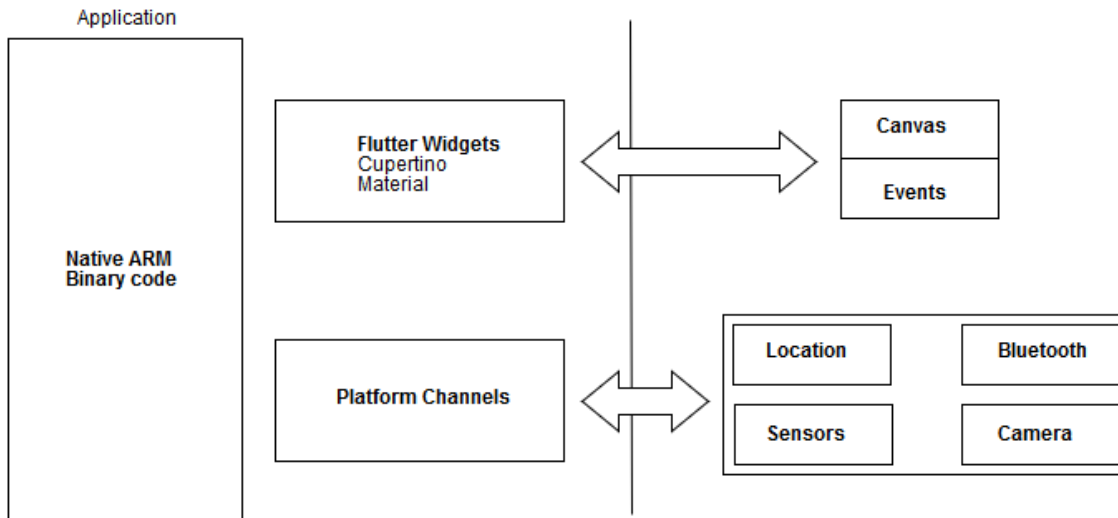


FIGURE 9. Flutter rendering

A Popular state management pattern for Flutter is Scoped Model. (FIGURE 10). Scoped Model consists of a model, a scoped model and a scoped model descendant. The model class contains the state of the application which notifies the scoped model and the descendant of its changes. Widgets can be wrapped in the scoped model to pass the state down the widget tree. The scoped model descendant can be used to listen for the changes in the model's state, descendant will rebuild when the model notifies it. The scoped model pattern was extracted from Google's Fuchsia project. (Wilson & Brian, 2018)

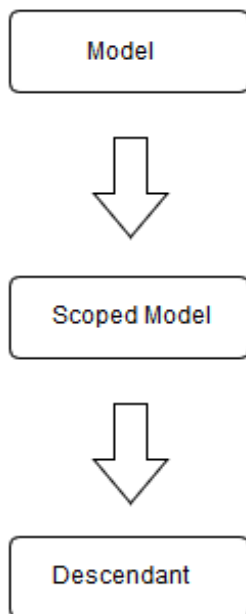


FIGURE 10. The scoped model pattern

2.3.1 Advantages

Everything in Flutter is a widget, even the application. If there is a need to adjust the opacity of image, it can be wrapped with Opacity widget. There are plenty of widgets that can be used to compose new widgets depending on the needs of the developers or designers. It is easy and fast to combine widgets to new ones. (Flutter, 2018)

Flutter does not need a separate layout language like JSX or XML, because Dart's programmatic layout is so easy to read and see. This makes creating advanced tooling for Flutter easier. Dart has similarities with static and dynamic languages, such as JavaScript and C#. That is why it is easy to pick up and learn and feels familiar to developers. (Leler, 2018)

Hot reload is a fast way to develop with Flutter. Because of Darts JIT compiling, hot reload can load the new code to a running application in under a second. Hot reload changes the way developers write applications as changes in the code can be seen in almost real time on the running application. Hot reload can even regain the state of the application after running errors. (Leler, 2018)

Though Flutter is a new framework, it is very mature for its age. It already has many guides and videos online. There is a thriving community that creates packages and plugins. Flutter's package managing site has hundreds of plugins/packages, which is a lot for a framework that hit version 1.0 only in 2018. Flutter has already been adopted by many big projects like, Google Ads, Alibaba and Google Greentea. (Flutter, 2018)

2.3.2 Disadvantages

Flutter does not have a separated UI language, which might complicate things for some people. Many developers are used to creating the UI separately. In theory, a designer could make the UI with an editor while a programmer can focus on coding the business logic. In Flutter developers also need to decide if they are using CupertinoWidgets (iOS) or MaterialWidgets (Android). Making the UI twice for each widget set is a lot of work. There is not a clear standard in development patterns like in Android or iOS. Flutter has multiple, for example, Bloc, Redux and Rxdart. Knowing which one to pick can be hard when starting to use Flutter. (Bellinaso, 2018)

2.4 Evaluating Frameworks

The key elements that are evaluated from these frameworks are: developer, performance, technology, documentation and ease of use. React Native succeeded in some of these key areas. React Native performs well if using the bridge between the Main and JavaScript threads is avoided. It is fast to develop because of hot reload. There is plenty of documentation and guides for it. Because there is a large developer community around it. Problems come with its technology, especially JavaScript which has many flaws. These

flaws are transferred to React Native. For example, debugging is hard, and every time dependency is added the whole system breaks. These two make up an unpleasant development experience. React Native is mainly supported by Facebook, which is not really a software company, but rather a social networking company. All this together makes for the somewhat inconsistent framework. A small RestaPoints demo application was built with React Native. (Figure 11).

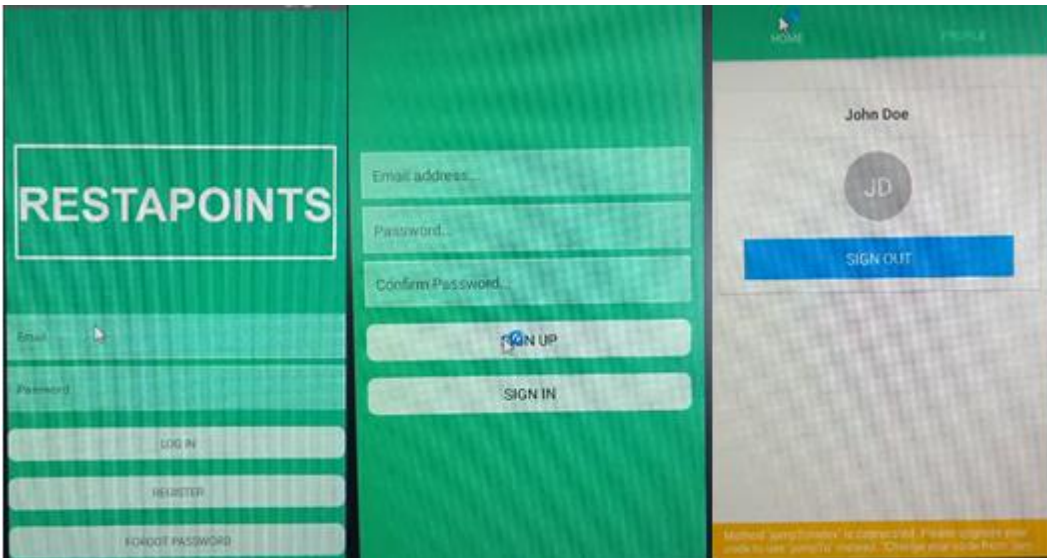


FIGURE 11. RestaPoints test application with React Native

Xamarin Forms has great support from Microsoft. For example, Visual Studio allows for a lot of integration to other developer tools and is by far the best development environment out there. Learning Xamarin Forms is easy because it has its own Xamarin University. C# .NET language that Xamarin Forms uses is consistent and well supported. Xamarin Forms is fast to develop with if the previewer works, which it rarely does. The Most popular development pattern in Xamarin Forms is MVVM, it is very clear and easy to understand. The company behind Xamarin is Microsoft which is not going anywhere and has clear roadmap for Xamarin Forms. In the testing of Xamarin Forms problems came from its weak performance. Because Xamarin Forms is another layer on top of Tradition Xamarin, it is not very fast, also it is easy for developers to write badly performing code. This issue is fine with small applications but with large applications it is amplified. So even though the integrated development environment and the language is great in Xamarin Forms, its performance is lacking, and its problematic previewer cannot be compared to the smoothness of hot reload in React Native or Flutter. RestaPoints sign in and registration developed with Xamarin Forms. (FIGURE 12. The RestaPoints test application with Xamarin Forms).

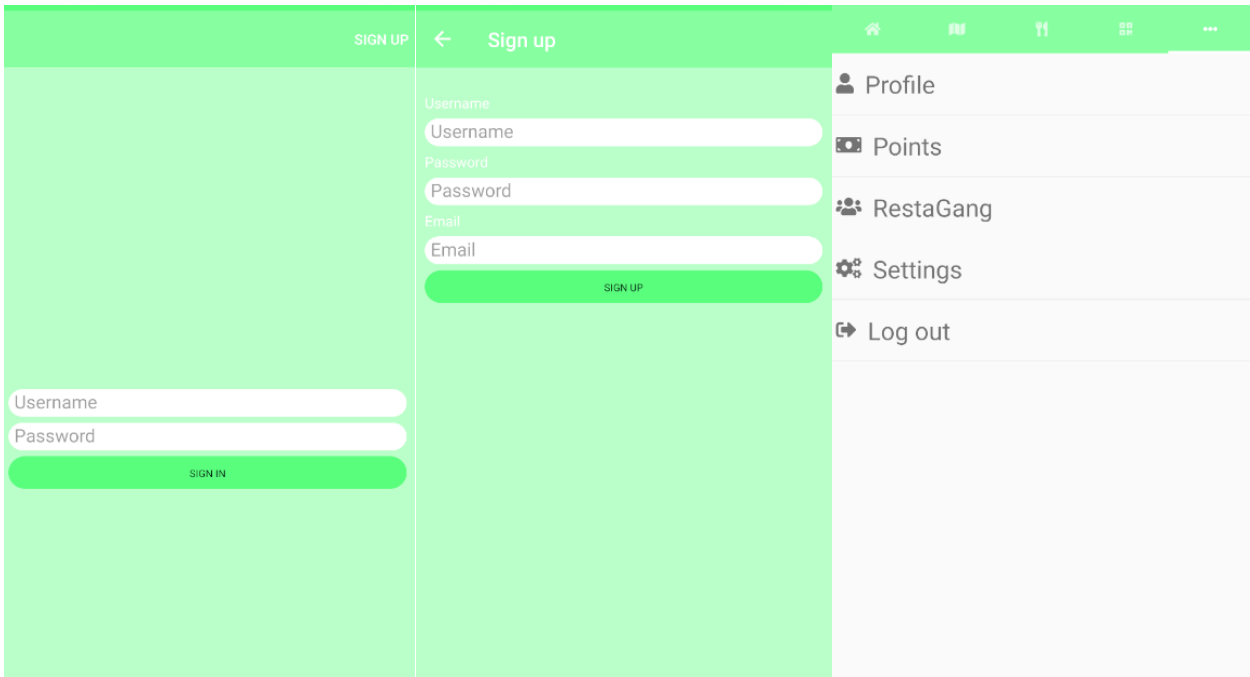


FIGURE 12. The RestaPoints test application with Xamarin Forms

The third cross platform framework tested is Google's Flutter. Flutter performs exceptionally well. It is consistent in all the key areas that were evaluated. It is easy to learn because of the many similarities with other programming languages, such as Java and JavaScript. There is many tutorials and videos online which also help. Development is fast because of hot reload; no need to rebuild the application when making code changes. Unlike React Native new packages do not break the application, so there is no endless bug hunting. Although Flutter is a new framework there is surprisingly many packages for it. Debugging is easy and there are many tools for profiling in Android Studio. Because Flutter can be compiled JIT and AOT, the performance is as close to native as possible. There is no telling apart Flutter and native application. Flutter's unique widget pattern makes for less code and faster, more natural way of coding. Couple that with hot reload and it will change the way coding is done. Google is the developer of Flutter, Google is designing a new mobile operating system, Fuchsia, which is done mainly with Dart code and supports Flutter applications. So, there should be a bright future for Flutter. Flutter is consistently good in all evaluated areas and in some cases a lot better compared to the other two frameworks. That is why Flutter was chosen as the development tool for RestaPoints mobile application.

3 SUPPORT TOOLS

3.1 Git & GitHub

Git is a version control tool for tracking changes in code. Git was created by Linus Torvalds. Linus Torvalds also created the very popular Linux operating system. The Linux development community used a version control tool called BitKeeper. In 2005, the Linux development community and BitKeeper had a falling out, which resulted in BitKeeper revoking the free-of-charge status of their tool. This led to Linus Torvalds and other Linux developers to create their own version control tool. Goals for the tool were these: *"speed, simple design, strong support for non-linear development (thousands of parallel branches), fully distributed, able to handle large projects like the Linux kernel efficiently (speed and data size)"*. (Git, 2018)

Git creates a code repository that has all the versions of the code that is being worked on, this is called content. Git also creates a working directory which is where the code is modified. Git workflow explained briefly: checkout code from the repository to the working directory, do changes and commit back new version of the content into the repository. When committing, changes are stored into content as snapshots. Snapshots are linked by parent-child relationship. The series of versions is called a branch. In a Git project there can be multiple branches, each one devoted to a different part of the project. This enables many developers to work on the project. Git is decentralized, which means that developers have always a repository that they can commit to, even in offline. (Fachat, 2015)

GitHub is a cloud-based service where developers can push their Git repositories, track changes, open issues, fork other people's code, contribute to projects and do code maintenance related work. In this thesis GitHub is used to get access to the code from anywhere. GitHub helps keep track of the changes that have been done to the code. Writing messages to commits is good practice and helps if something is not working later. The history of commits is shown in Github. (FIGURE 13). GitHub also allows reverting to earlier versions of the code.

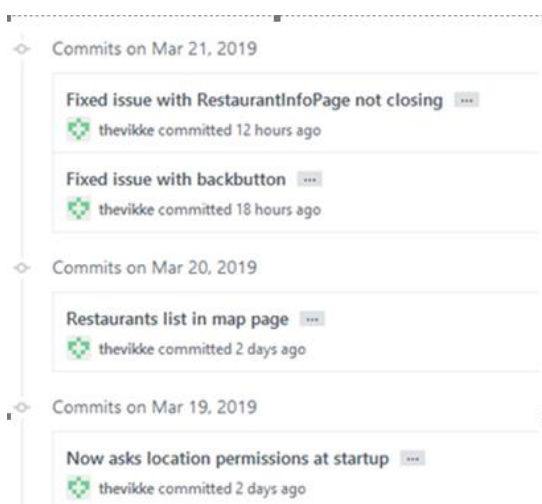


FIGURE 13. The history of commits in Github

3.2 Visual Studio Code

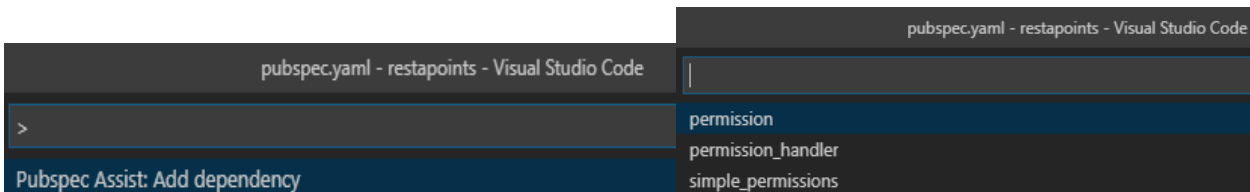
Visual Studio Code is a code editor developed by Microsoft. Visual Studio Code is minimalistic on its appearance and has lots of plugins to help development. In Stack OverFlow's survey 2018, Visual Studio Code was the most popular development environment, with 34.9% of developers using it (Stack OverFlow, 2018).

These Visual Studio Code plugins were used during the development: Flutter-plugin to get flutter syntax highlighting, Bracket Pair Colorizer 2-plugin to match bracket colors, Pubspec Assistant-plugin to easily add Flutter packages, and finally Better Comments-plugin that highlights comments depending on status (FIGURE 14 & Figure 15).



```
@override
Widget build(BuildContext context) {
  return BasicPage(
    title: "Change Password",
    middle: ChangePasswordForm(),
    bottom: Column(
      children: <Widget>[
        SizedBox(
          height: 10.0,
        ), // SizedBox
        ImportantButton(
          text: "CHANGE",
          onPressed: () {},
        ), // ImportantButton
      ], // <Widget>[]
    ), // Column
  ); // BasicPage
}
```

FIGURE 14. The Bracket Pair Colorizer package example



```
pubspec.yaml - restapoints - Visual Studio Code
pubspec.yaml - restapoints - Visual Studio Code
>
Pubspec Assist: Add dependency
permission
permission_handler
simple_permissions
```

FIGURE 15. The Pubsec Assistant example

3.3 OneNote

OneNote is a note keeping tool that works in the cloud. It is developed by Microsoft. OneNote is used to keep track of all the features for RestaPoints. When feature is implemented, it is checked as done in OneNote. Bigger features are splitted to smaller ones to make them more manageable. Working this way gives easy goals that can be reached daily.

4 DEVELOPING RESTAPOINTS

4.1 Views

The RestaPoints mobile application can be split in two parts: unregistered and registered (FIGURE 16 & FIGURE 17). Unregistered users can only access to log in, register and forgot password view, and registered users can access everything. Authentication is done with an email and password. A Https call is sent to API which then checks the database for the user info. If the user is valid, a security token is generated and returned to the device. The token is stored into a secure storage in the device and a new token is generated every time a https-call is made to ensure its integrity. If the user has forgotten the password there is a view where the user can get a link to their email address to reset. When the user logs in for the first time, he/she is greeted with an intro view that explains how to use the application.

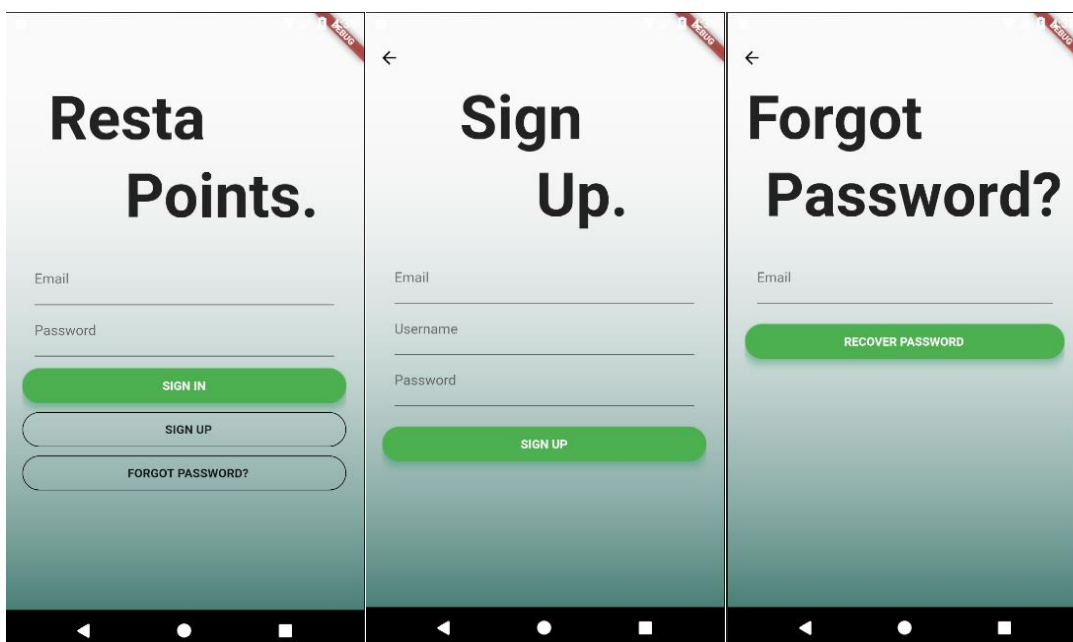


FIGURE 16. RestaPoints for unregistered users with Flutter

The main application has a TabBarNavigation view with three tabs. The First view has QR-code that is read in the restaurant to identify the user and reward them with points. The second view has a map with a horizontal list of restaurants on top of it. Scrolling the list will move the map's camera to the location of the desired restaurant. The third view has a vertical list of restaurants with sorting options on top. In this view the user can sort restaurants depending on, how close the restaurants are, how many points the user has with restaurants and how well the restaurants are rated by other users.

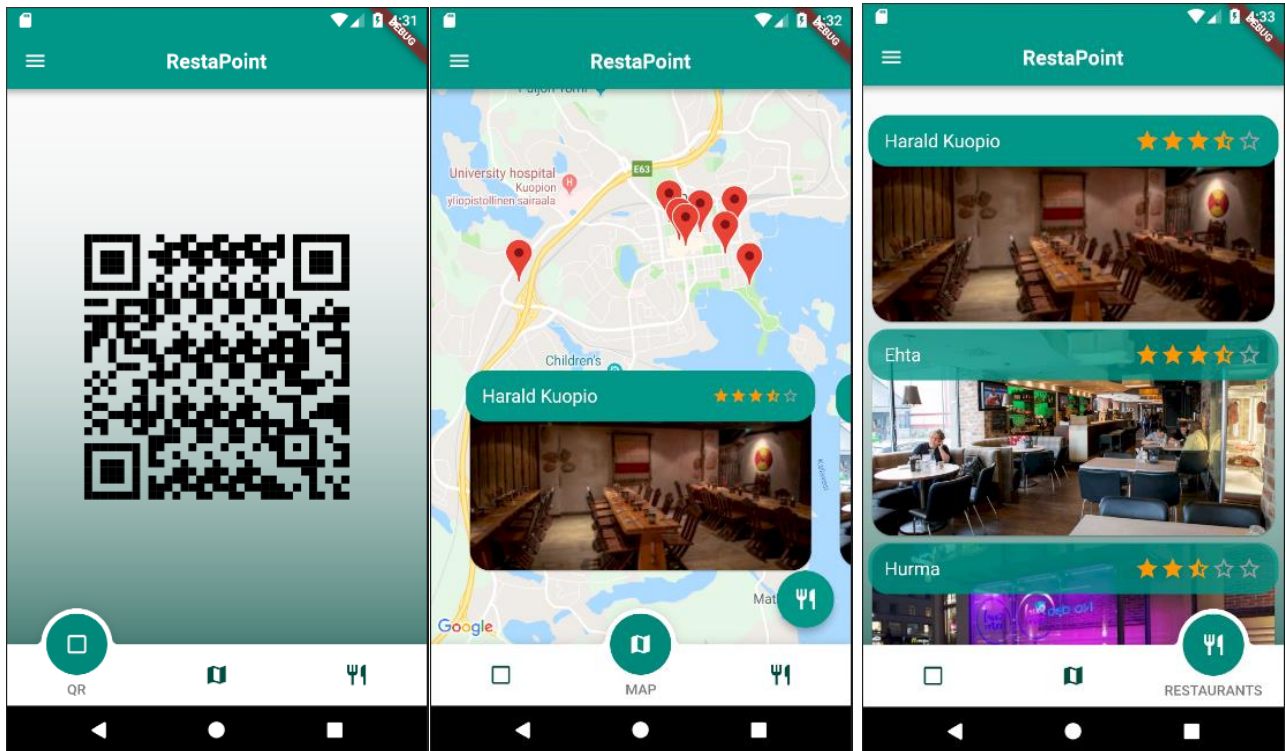


FIGURE 17. RestaPoints for registered users with Flutter

Swiping from the left or pushing the hamburger button in the left corner opens the menu view with buttons to other views. (FIGURE 18). These views are: profile, settings, points and restagang view. In the profile view, the user can change his/her profile image, username and password. In the settings view the user can change language and the theme of the application. In the points view the user can track the accumulation of points per a restaurant and sort them accordingly. In the restagang view the user can create a restagang and add friends to it. A restagang is a group of maximum four members that get notified when other members of the same restagang go to eat in a RestaPoints restaurant. The members also get more points if they go the same restaurant within a certain time period. The last item in the menu view is a logout button: when the user clicks it, he/she will be taken to the login view of the application and the security token is destroyed.

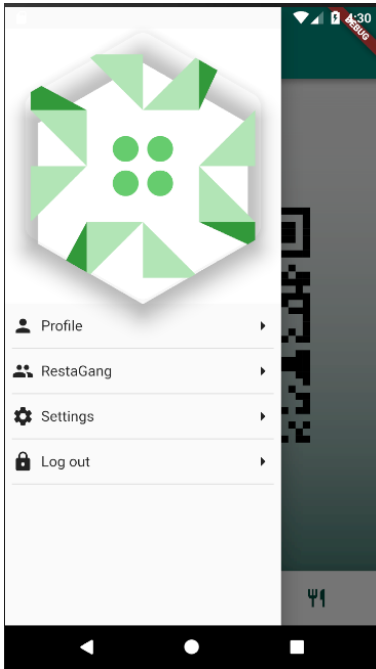


FIGURE 18. The menu view with buttons

All but one view in the application is implemented with a widget called `BasicPage`. (FIGURE 19). This widget wraps the content of a view and animates it during the view's initialization. The `BasicPage` is a customized widget that acts as a blueprint that all the views can implement. `BasicPage` has top, middle and bottom sections that animate from left to right with small difference in timing. Animation is done with `Transform` widget, which will be discussed in detail in the animation section. `BasicPage` widget changes the background, title and few other details of the view. To use `BasicPage`, UI widgets need to be added to the top, middle and bottom -parameters. The `BasicPage` does the animating of the UI widgets when view is initialized. Splitting up the application to smaller pieces has many benefits: code is more readable, custom widgets can be used in many places and bugs are easier to find.

```
typedef void FabCallBack();

class BasicPage extends StatefulWidget {
  BasicPage(
    {this.top,
     this.middle,
     this.bottom,
     this.title = "",
     this.titleColor = Colors.black,
     this.fabMenu = false,
     this.fabBackgroundColor,
     this.fabIcon,
     this.fabOnPressed});
  final Widget top;
  final Widget middle;
  final Widget bottom;
  final String title;
  final Color titleColor;
  final Color fabBackgroundColor;
  final IconData fabIcon;
  final bool fabMenu;
  final FabCallBack fabOnPressed;
  @override
  _BasicPageState createState() => _BasicPageState();
}
```

FIGURE 19. BasicPage constructor

4.2 Routing

In the Flutter framework “screens” and “pages” are called “routes”, defined as Route objects. A Navigator widget handles a stack of Route objects and allows navigating back and forth in the stack, for example, Navigator.pop and Navigator.push. (Flutter, 2019)

RestaPoints uses named routes, which allows changing the route later without changing all the references. Named routes use the MaterialPageRoute widget, which gives the default Android transition to the next screen. (FIGURE 20). Navigation in the tab view is done with the IndexedStack widget. IndexedStack is a stack of widgets on top of each other. The main difference to a normal Stack widget is that IndexedStack only shows one item at a time. Changing the index of the IndexedStack changes the tab view. IndexedStack is wrapped to the FadeTransition widget to give it transition animation.

```
void main() {
  AppStateModel state = AppStateModel();
  AuthenticationModel auth = AuthenticationModel();
  runApp(
    ScopedModel<AuthenticationModel>(
      model: auth,
      child: ScopedModel<AppStateModel>(
        model: state,
        child: MaterialApp(
          home: LandingPage(),
          onGenerateRoute: (RouteSettings settings) {
            // TabView page...
            switch (settings.name) {
              case '/App':
                return MaterialPageRoute(
                  builder: (BuildContext context) => App(),
                ); // MaterialPageRoute
                break;
              case '/IntroPage':
                return MaterialPageRoute(
                  builder: (BuildContext context) => IntroPage(),
                ); // MaterialPageRoute
                break;
              case '/RootPage':
                return MaterialPageRoute(
                  builder: (BuildContext context) => RootPage(),
                ); // MaterialPageRoute
                break;
              case '/ProfilePage':
                return MaterialPageRoute(
                  builder: (BuildContext context) => ProfilePage(),
                ); // MaterialPageRoute
            }
          },
        ),
      ),
    ),
  );
}
```

FIGURE 20. Routing in RestaPoints

4.3 Animations

Flutter does not need native UI widgets; it draws its own UI. This allows it to implement its own API on animations. This makes it easy to implement animations to any widget that there is. The BasicPage's sliding animations that runs when the view is initialized, is accomplished with the AnimationBuilder and the Transform widgets. Builder requires AnimatedController as parameter, the controller tells builder how long the animation is and if it repeats. (FIGURE 21). The transform controls the positions of the widgets that the BasicPage wraps. (FIGURE 22). The controller changes the values of the transform, moving it from left to right, giving it a sliding effect when view is initialized. The AnimationBuilder is used with complex animations. Simple fade in animations or scale animations can be accomplished with AnimatedOpacity and AnimatedContainer widgets. The AnimatedOpacity is used to fade in the back buttons to the AppBar when opening new routes.

```

AnimatedBuilder(
  animation: _controllerStartupAnimation,
  builder: (BuildContext context, Widget child) {
    return Container(
      height: height,

```

FIGURE 21. AnimatedBuilder example

```

Transform(
  transform: Matrix4.translationValues(
    _animationMiddle.value * width, 0.0, 0.0), // Matrix4.translationValues
  child: widget.top,
), // Transform

```

FIGURE 22. Transform example

4.4 Plugins

In Flutter dependencies are defined in the pubspec.yaml file, which is generated when creating a new Flutter application. (FIGURE 23). Some examples of the packages used in the RestaPoints: the intro_views_flutter package is used to make the intro view for when the user first logs in, the qr package is used to draw the qr code, the carousel_slider package is used to show the images of the restaurant in the restaurant info page, and finally the google_maps_flutter is used to integrate the Google Maps to the application. Even though there is plenty of dependencies, they all work together without a hitch. There are many packages available in Dart's package manager, <https://pub.dartlang.org/>.

```

01. environment:
02.   sdk: ">=2.0.0-dev.68.0 <3.0.0"
03. # pubspec assist
04. dependencies:
05.   flutter:
06.     sdk: flutter
07.   flip_box_bar: ^0.2.0
08.   intro_views_flutter: ^2.4.0
09.   polygon_clipper: ^1.0.1
10.   qr: ^1.0.1
11.   google_maps_flutter: ^0.0.3+3
12.   geolocator: ^3.0.0
13.   dio: ^1.0.9
14.   flutter_secure_storage: ^3.1.2
15.   fab_menu: ^0.0.2
16.   cached_network_image: ^0.5.1
17.   sticky_headers: ^0.1.7
18.   carousel_slider: ^1.0.1
19.   flutter_rating: ^0.0.2
20.   scoped_model: ^1.0.1
21.   fancy_bottom_navigation: ^0.3.1
22. # The following adds the Cupertino Icons font to your application.
23. # Use with the CupertinoIcons class for iOS style icons.
24.   cupertino_icons: ^0.1.2
25.   jdenticon_dart: ^1.1.1
26.   permission_handler: ^3.0.0
27.   #snaplist: ^0.1.8

```

FIGURE 23. The packages used in RestaPoints

4.5 State management

Many state management options were evaluated for RestaPoints: InheritedWidget pattern, Bloc and Redux. However, Scoped Model pattern was chosen in the end. The Scoped model is light weight, easy to implement and very powerful. (FIGURE 24). The Scoped model was extracted from the Googles Fuchsia project, which proves that it scales to bigger projects (Wilson & Brian, 2018). Since RestaPoints is a relatively simple application with very few moving parts, there was only need for a few scoped models.

```

01. class CounterModel extends Model {
02.   int _counter = 0;
03.
04.   int get counter => _counter;
05.
06.   void increment() {
07.     // First, increment the counter
08.     _counter++;
09.
10.     // Then notify all the listeners.
11.     notifyListeners();
12.   }
13. }

```

FIGURE 24. Scoped Model example

4.6 Https calls

Loading data into the application is done through the data repository. The data repository will be the only place that serves data to parts of the application. Data repository holds the functions for calling API, for

example, log in function and function for loading list of restaurants. All functions in the data repository are asynchronous so that they do not block UI tasks. The asynchronous functions in Flutter return a Future type. The UI uses the FutureBuilder widget to build the data returned from futures to UI components, before the data is returned FutureBuilder shows a loading text. (FIGURE 25). When the data is returned from the future, the builder builds all the widgets that use the data.

```

01. FutureBuilder<String>(  

02.   future: _callAPI, // async work  

03.   builder: (BuildContext context, AsyncSnapshot<String> snapshot) {  

04.     switch (snapshot.connectionState) {  

05.       case ConnectionState.waiting: return new Text('Loading...');  

06.     default:  

07.       if (snapshot.hasError)  

08.         return new Text('Error: ${snapshot.error}');  

09.       else  

10.         return new Text('Result: ${snapshot.data}');  

11.     }  

12.   },  

13. )

```

FIGURE 25. FutureBuilder example

Package named Dio is used for https calls to the API. The Dio enables all the CRUD operations with https. The data helper class was made to store the security token that is returned from the API. The data helper class inserts token to the header of the https call. When data is returned from the API helper class checks for the security token and saves it to secure storage. (FIGURE 26). RestaPoints mobile application does not generate the token. That is done in the server side.

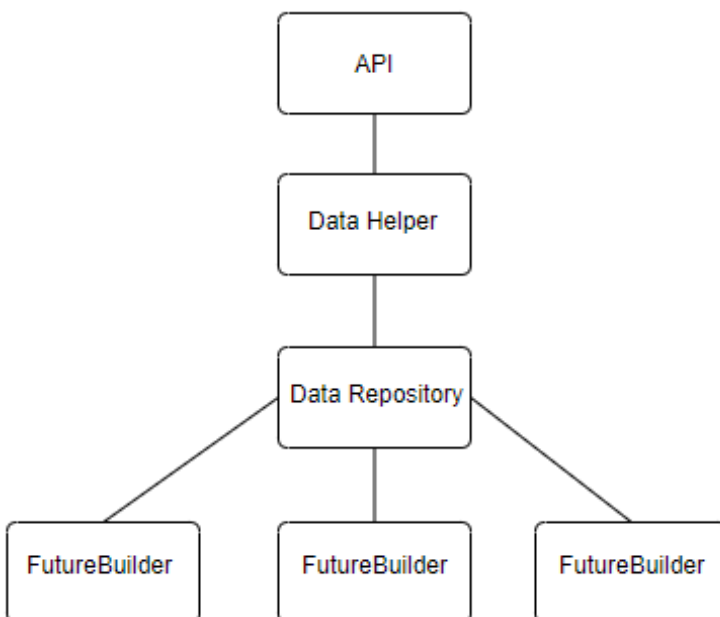


FIGURE 26. Data flow in RestaPoints

4.7 Studying

A lot studying went into learning all three frameworks, even at novice level. There was no prior experience of them when this thesis was started. All the frameworks had their own documentations; Xamarin even had its own university program. Real life examples were easy to find from articles, YouTube videos and Stack Overflow posts. One of the best websites for coding related articles is Medium. For Flutter there is Flutter weekly email that highlighted new articles, packages and videos.

It takes a lot of time to study and learn new frameworks. That is why it is even more important for the framework to be easy to learn. This was the case particularly with Flutter and one of the reasons why it stood out from the other two.

5 CONCLUSION

The purpose of this thesis was to find the best cross platform framework for today's mobile development and describe about the RestaPoints development with it. Frameworks were introduced in the development tools section; their advantages and disadvantages were evaluated. The best framework was chosen for development. Due to the lack of resources and time, there is not a straight-out comparison between the frameworks, rather an overview of the advantages and disadvantages, of each framework.

In the end, Flutter was the best for RestaPoints. Flutter has fast development, because of the hot reload and great performance because of the AOT compiling. Flutter is easy to learn because of many familiarities with other languages. It has a lot of tooling and support from Google.

To conclude, the development experience with the frameworks that were tested was faster than with their native counter parts. While they might lack performance that native languages offer, they still enable faster development with lower cost. There will be more cross platform developers in the future and less native developers. These frameworks have come far but will go even further in the future. They are getting faster and smarter every year. No doubt that they will take over the mobile development industry.

6 REFERENCES

- AMADEO, R. (2018). Google's Dart language on Android aims for Java-free, 120 FPS apps [online publication]. ARS Technica. [accessed: 2019-20-04]. Available: <https://arstechnica.com/gadgets/2015/05/googles-dart-language-on-android-aims-for-java-free-120-fps-apps/>
- AZILEN. (2017). Xamarin- The Good, The Bad and The Ugly [online publication]. Medium. [accessed: 2019-20-04]. Available: <https://medium.com/@AzilenTech/xamarin-the-good-the-bad-and-the-ugly-49e3fe72b7ae>
- BELLINASO, M. (2018). Flutter: the good, the bad and the ugly [online publication]. Medium. [accessed: 2019-20-04]. Available: <https://medium.com/asos-techblog/flutter-vs-react-native-for-ios-android-app-development-c41b4e038db9>
- Britch, D., Dunn, C., & Poulad. (2017). The Model-View-ViewModel Pattern [online publication]. Microsoft Docs. [accessed: 2019-20-04]. Available: <https://docs.microsoft.com/en-us/xamarin/xamarin-forms/enterprise-application-patterns/mvvm>
- Burns, A.; Britch, D.; Umbaugh, B.; & Dunn, C. (2017). Part 1 – Understanding the Xamarin Mobile Platform [online publication]. Microsoft Docs. [accessed: 2019-20-04]. Available: <https://docs.microsoft.com/en-us/xamarin/cross-platform/app-fundamentals/building-cross-platform-applications/understanding-the-xamarin-mobile-platform>
- CHAND, M. (2016). What Can C# Do For You [online publication]. C-sharpcorner. [accessed: 2019-20-04]. Available: <https://www.c-sharpcorner.com/UploadFile/maresh/what-can-C-Sharp-do-for-you/>
- CHRZANOWSKA, N. (2019). React Native Pros and Cons - Facebook's Framework in 2019 [online publication]. Netguru. [accessed: 2019-20-04]. Available: <https://www.netguru.com/blog/react-native-pros-and-cons>
- ENG, R. K. (2016). The Top 10 Things Wrong with JavaScript [online publication]. Medium. [accessed: 2019-20-04]. Available: <https://medium.com/javascript-non-grata/the-top-10-things-wrong-with-javascript-58f440d6b3d8>
- EVKOSKI, B. (2017). React Native: What it is and how it works [online publication]. Medium. [accessed: 2019-20-04]. Available: <https://medium.com/we-talk-it/react-native-what-it-is-and-how-it-works-e2182d008f5e>
- FACHAT, A. (2015). Learn the workings of Git, not just the commands [online publication]. IBM. [accessed: 2019-20-04]. Available: <https://developer.ibm.com/tutorials/d-learn-workings-git/>
- FLUTTER. (2018). Flutter [online publication]. Flutter. [accessed: 2019-20-04]. Available: <https://flutter.io/>
- FLUTTER. (2018). Flutter faq [online publication]. Flutter faq. [accessed: 2019-20-04]. Available: <https://flutter.io/docs/resources/faq>
- FLUTTER. (2019). *Navigator class* [online publication]. Flutter Docs. [accessed: 2019-20-04]. Available: <https://docs.flutter.io/flutter/widgets/Navigator-class.html>
- FRIEDMAN, N. (2014). Announcing Xamarin 3 [online publication]. Xamarin Blog. [accessed: 2019-20-04]. Available: <https://blog.xamarin.com/announcing-xamarin-3/>

- GIT. (2018). *Getting Started - A Short History of Git* [online publication]. *Git-scm*. [accessed: 2019-20-04]. Available: <https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git>
- GOOGLE. (2018). Announcing Flutter 1.0 [online publication]. Google Developers. [accessed: 2019-20-04]. Available: <https://www.youtube.com/watch?v=kpcjBD1XDwU&t=919s>
- GREEN, A. (2017). C# 6.0 draft specification [online publication]. Microsoft Docs. [accessed: 2019-20-04]. Available: <https://docs.microsoft.com/fi-fi/dotnet/csharp/language-reference/language-specification/introduction>
- GUTHRIE, S. (2016). Microsoft to acquire Xamarin and empower more developers to build apps on any device [online publication]. Microsoft Blog. [accessed: 2019-20-04]. Available: <https://blogs.microsoft.com/blog/2016/02/24/microsoft-to-acquire-xamarin-and-empower-more-developers-to-build-apps-on-any-device/>.
- LELER, W. (2018). Why Flutter Uses Dart [online publication]. Medium. [accessed: 2019-20-04]. Available: <https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf>
- LEUSCHENKO, O. (2018). The Pros and Cons of Xamarin for Cross-Platform Development [online publication]. Hackernoon. [accessed: 2019-20-04]. Available: <https://hackernoon.com/the-pros-and-cons-of-xamarin-for-cross-platform-development-2a31c6610792>
- LEVKOVSKY, M. (2017). Thinking in Redux (when all you've known is MVC) [online publication]. Hackernoon. [accessed: 2019-20-04]. Available: <https://hackernoon.com/thinking-in-redux-when-all-youve-known-is-mvc-c78a74d35133>
- MICROSOFT. (2018). *Visual Studio Community* [online publication]. Microsoft. [accessed: 2019-20-04]. Available: <https://visualstudio.microsoft.com/vs/community/>
- MICROSOFT. (2019). Buy Visual Studio [online publication]. Microsoft. [accessed: 2019-20-04]. Available: <https://visualstudio.microsoft.com/vs/pricing/#tab-b8953f16f0b68f60f18>
- SARVAIYA, D. (2018). Why I think Google's Flutter is here to stay as a Mobile App Development SDK [online publication]. Hackernoon. [accessed: 2019-20-04]. Available: <https://hackernoon.com/why-i-think-googles-flutter-is-here-to-stay-as-a-mobile-app-development-sdk-d3b300231028>
- SEKULIĆ, R. (2016). A brief history of React Native [online publication]. Medium [accessed: 2019-20-04]. Available: <https://medium.com/react-native-development/a-brief-history-of-react-native-aae11f4ca39>
- STACK OVERFLOW. (2018). *Most Popular Development Environments* [online publication]. Stack OverFlow. [accessed: 2019-20-04]. Available: <https://insights.stackoverflow.com/survey/2018/>
- STATCOUNTER. (2018). *Mobile Operating System Market Share Worldwide* [online publication]. StatCounter. [accessed: 2019-20-04]. Available: <http://gs.statcounter.com/os-market-share/mobile/worldwide>
- STOLL, S. (2018). In plain English: So what the heck is Flutter and why is it a big deal [online publication]? Medium. [accessed: 2019-20-04]. Available: <https://medium.com/flutter-community/in-plain-english-so-what-the-heck-is-flutter-and-why-is-it-a-big-deal-7a6dc926b34a>
- WILLIAMS, M. (2017). Xamarin vs React Native [online publication]. Flush Arcade. [accessed: 2019-20-04]. Available: <https://www.youtube.com/watch?v=3-FmJk9bFCM>

WILSON, A.;& Brian, E. (2018). Scoped Model [online publication]. Pub Dartlang. [accessed: 2019-20-04].
Available: https://pub.dartlang.org/packages/scoped_model