POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell 'Informazione

Master of Science in Computer Engineering

Polo Territoriale di Como

Analysis and comparison of cross-platform mobile development approaches

Supervisor:
Prof. Marco Brambilla

Master Graduation Thesis by:
Ardian Isufi, student id: 739977

Academic Year 2015-2016

POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell 'Informazione

Corso di Laurea Specialistica in Ingegneria Informatica

Polo Territoriale di Como

Valutazione e confronto di approcci di sviluppo mobile cross-platform

Relatore:
Prof. Marco Brambilla

Tesi di Laurea Specialistica di:
Ardian Isufi, matricola: 739977

Anno Accademico 2015-2016

- To my wife, my children and my family for their endless support.
- To my late grandmother, the greatest influence in my life.

Abstract

The introduction of smart mobile devices (smart phones and tablets) has increased rapidly the number of mobile users and mobile applications. Only in 2016 around 224 billion mobile applications were downloaded worldwide, mainly for the following three main different mobile operating systems: Android, iOS and Windows Phone. Having more than one major mobile operating system, necessitates the development of mobile applications that will be ready to be deployed in mobile devices with different mobile operating system. A lot of efforts have been made lately to find tools that that can give the option to the mobile applications developers to develop an application that is compatible with more than one mobile OS, and these tools are known as cross-platforms mobile development frameworks. There are different available cross-platforms in the market, and as the topic of this master thesis is a detailed analysis and comparison of cross-platforms mobile development approaches.

Throughout this thesis, besides analyzing cross-platform mobile development approaches, our focus is oriented also on comparing two specific cross platforms: Xamarin and PhoneGap, each of them representing different mobile development approach, cross-platform native and hybrid mobile applications. The best way to do a proper comparison, besides relying on relevant work is by designing and implementing a mobile application in both platforms. During this research we design and implement a case study in both Xamarin and PhoneGap and then based on the gained experience during development experience we describe the comparison results according to the following factors: Graphical User Interface, architecture, service and sensors, local data storage and development efforts.

Keywords: mobile device, cross-platform, Xamarin, PhoneGap, plugin, mobile development approach, android, iOS, windows phone.

Sommario

L'introduzione di dispositivi mobili intelligenti (smartphone e tablet) è aumentato rapidamente il numero di utenti di telefonia mobile e applicazioni mobili. Solo nel 2016 intorno a 224 miliardi di applicazioni mobili sono stati scaricati in tutto il mondo, in genere per i tre sistemi operative principali (OS): Android, iOS e Windows Phone. Avere diversi sistemi operativi mobili, significa che c'è sempre una necessità per le aziende di eseguire le proprie applicazioni in sistemi operative diversi. Recentemente sono stati fatti molti sforzi per trovare strumenti che in un certo modo può offrire la possibilità agli sviluppatori di applicazioni mobili di sviluppare un'applicazione che è compatibile con più di un sistema operativo mobile, e questi strumenti sono noti come - cross-platform frameworks -. Ci sono diversi cross-piattaforme disponibili sul mercato, e il tema di questa tesi è una dettagliata valutazione e confronto di approcci di sviluppo mobile cross-platform.

In questo tesi, oltre alla valutazione degli approcci di sviluppo mobile cross-platform, la nostra attenzione è orientata sul confronto di due cross-platform: Xamarin e PhoneGap, ciascuna delle quali rappresentano un approccio diverso di sviluppo mobile e cio è cross-platform nativa (native-like) e le applicazioni mobili ibridi. Il modo migliore per fare un confronto, oltre alla fare un affidamento sul lavoro rilevante, è attraverso la progettazione e l'attuazione stessa della applicazione mobile in entrambe le piattaforme. In questa tesi abbiamo progettato e implementato un caso di studio sia di Xamarin e di PhoneGap, e poi sulla base dell'esperienza acquisita durante lo sviluppo abbiamo descritto i risultati del confronto basandosi nei seguenti fattori: Interfaccia grafica utente, l'architettura, il servizio e sensori, archiviazione dei dati locali e tutti i sforzi attuati per ottenere un sviluppo.

Parole chiave: dispositivi mobili, cross-platform, Xamarin, PhoneGap, plug-in, l'approccio di sviluppo mobile, Android, iOS, Windows Phone.

Contents

Abstract **i**

Sommario **ii**

List of Figures **v**

List of Tables **vi**

1 Introduction **1**
1.1 Context 1
1.2 Problem statement & proposed solution 2
1.3 Structure of the thesis 4

2 Background **5**
2.1 Introduction 5
2.2 Mobile platforms chronology 5
2.3 Cross platforms mobile development 7
Native applications development 10
Cross-platform applications development 10
Hybrid applications development 11
Comparison of applications development techniques . . 12
2.4 Xamarin 13
2.4.1 How does Xamarin work 14
2.5 PhoneGap 16
2.5.1 How does PhoneGap work 17

3 Related work **20**

4 Cross-platform approaches comparison method **29**
4.1 Introduction 29
4.2 Problem definition 29
4.3 Proposed solution 30

4.4 Comparison method 31

5 Case study and implementation experience 36
5.1 Introduction 36
5.2 Case study 36
5.2.1 General description 36
5.2.2 Requirements 37
5.2.3 Assumptions and dependencies 38
5.2.4 Entity model 39
5.2.5 Use cases 43
5.3 Implementation of WebAPI 53
5.4 Implementation with Xamarin 55
5.5 Implementation with PhoneGap 58

6 Comparison results 60
6.1 Introduction 60
6.2 Graphical User Interface 60
6.3 Architecture 61
6.4 Services and sensors 62
6.5 Local data storage 64
6.6 Development efforts 65
6.7 Comparison conclusion 66

7 Conclusion 67

Bibliography 69

Acknowledgments 71

List of Figures

2.1 Mobile platforms chronology 7
2.2 Global market share held by the leading smartphone operating systems in sales to end users from 3rd quarter 2009 to 3rd quarter 2015. Source:[1] 8
2.3 Mobile applications development types. 9
2.4 Mobile applications development types comparison. 12
2.5 Xamarin mobile applications architecture. 15
2.6 Hybrid mobile applications architecture. 19

5.1 Entity Relationship diagram 42
5.2 Page diagram 52
5.3 WebApi in our case study (PoiPolimi) 53

6.1 Same GUI implementation of our main page in Xamarin(A) and PhoneGap(B). 61
6.2 Same camera access implementation of our case-study in Xamarin(A) and PhoneGap(B). 63

List of Tables

3.1 Results of testing and evaluation of cross-platforms for mobile application development according to paper [2]. 22
3.2 Results of testing and evaluation of cross-platforms for mobile application development according to paper [3]. 23

5.1 Use case: Create new user account (UC01) 44
5.2 Use case: Login (UC02) 44
5.3 Use case: View and modify user profile (UC03) 45
5.4 Use case: Search for PoI (UC04) 46
5.5 Use case: Search for PoI by PoI type (UC05) 46
5.6 Use case: List PoI close to me (UC06) 47
5.7 Use case: List of PoI reviewed by me (UC07) 47
5.8 Use case: View PoI details (UC08) 48
5.9 Use case: View PoI reviews (UC09) 48
5.10 Use case: View PoI photos (UC10) 49
5.11 Use case: View PoI review photos (UC11) 49
5.12 Use case: New review for specific PoI (UC12) 50
5.13 Use case: List of PoI reviewed by me (UC13) 51
5.14 Use case: Submit an already pending review. (UC14) 51

Chapter 1

Introduction

1.1 Context

From the ancient times, mankind always has been pushed to create opportunities to solve different problems and requirements. If we go back in the history of computers, we see that multiple steps are made with different technologies to solve problems, precisely different algorithms and applications have been developed in different periods of time.

With the arrival of the personal computer and Graphical User Interface (GUI) implementation, a world until then unknown to engineers but also to the users opened in unpredictable way. Introduction of the Internet in the 90's, changed also the types of requirements and distribution of the applications. A number of application start being developed through the Web technology, through which there is no need for application to be installed physically, in order to access it. Later, usage of smart phones and the increase number of mobile users in a massive way, wrote a new page in the history of the mobile application. Now the application began to be more accessible to mobile portable devices such as:smart phone, tablet, etc.

Introduction of mobile devices, has increased rapidly the number of mobile users. According to latest statistics [4], only in the market of the United States in 2015, 51% of internet users were from mobile devices, 42% from personal computers and 7% from other devices. Interestingly, in 2008 this percentage was only 12%, while from other equipment occupied 80%.

Such a increase of mobile users is making them important user group in terms of users segmentation. This user category is not a small number, and what is more important, this is increasing in a fast way, thus marking the need for mobile applications, applications developed to be deployed in mobile device. Only in 2016 [5] world wide, were downloaded in total 224 billion

mobile applications.

Current mobile applications distribution market is divided by different mobile operating systems (OS). According to [6], we have three major mobile operating systems: Android by Google, iOS by Apple and Windows Phone by Microsoft.

In terms of development, coding mobile applications for different platforms and operating systems requires well trained staff for specific platform. The furious rate of technological change and growth in the mobile market has made it very challenging for developers to strategically plan a bespoke project, not only from a technical standpoint, but also because the market share for smartphones is changing rapidly between different systems [7].

Given the need for development in various markets and never forgetting the 90's inspiration from Java - "Build Once, Run Anywhere", market begins to feel the first steps of cross-platform mobile development frameworks. Cross-platform mobile development refers to the development of mobile application that can be run in different mobile operating systems. Using a cross-platform approach can decrease development time as a mobile application is written once and deployed to multiple platforms as opposed to developing an individual application for each environment [8].

During the recent years, many cross-platform mobile development framework were born. Some of them are still in the market and some of them have merged. Some of these are: Xamarin, initially created by Xamarin but now owned by Microsoft, PhoneGap (Cordova) initally created by Nitobi but then purchased by Adobe Systems, Appcelerator Titanium from Appcelerator, Corona by Corona Labs Inc., Sencha Touch by Sencha, RhoMobile by Zebra Technologies etc.

1.2 Problem statement & proposed solution

Taking into account this growing mobile market and current use of cross-platforms mobile development frameworks, we decided that the analysis and comparison of cross-platform approaches to be the main focus of our research for this master thesis.

The idea and the main purpose of this research is a thorough professional analysis and comparison of these cross-platforms. A comparison, based on arguments achieved after an evaluation but also a thorough study of the current literature. Because current market is filled-in with a range of cross-platforms mobile development frameworks, each with its specific features, for comparison purpose in this thesis we will select only two of them. To achieve more interesting and compact comparison, the choice of two platforms is

based on several key factors such as: the type of product they are offering, the financial cost to use them and their innovations in the market.

Xamarin and PhoneGap are our two approaches subject of comparison for this master thesis. They are chosen specifically because of different platform types, more precisely the type of product that is developed through them. The mobile applications developed through PhoneGap is hybrid mobile application, while Xamarin one is cross-platform cross-platform native application.

Given the analysis of cross-platform mobile development approaches and our implementation experience from case-study in two specific cross-platforms, then the problem we are trying to solve in this thesis is as follow:

Taking into account the following factors:

- *Graphical User Interface (GUI);*
- *Architecture;*
- *Services and sensors;*
- *Local data storage;*
- *Development efforts;*

"Which of the cross-platforms mobile development frameworks, compared in this thesis - each of them representing specific mobile application approach, is the most convenient one for the development of new mobile application?"

The best way to do a proper comparison, besides relying on relevant work is by designing and implementing a case study - mobile application in both cross-platforms development frameworks.

In our thesis we design and develop a "review" mobile application (our case study) in both cross-platforms mobile development frameworks, and based on our implementation (development) experience we group the comparison results according to the chosen factors.

1.3 Structure of the thesis

This master thesis is organized in the following chapters:

- *Chapter 1 – Introduction*: This chapter gives the reader a general introduction on the content of the master thesis.
- *Chapter 2 – Background*: We write general information about mobile development and cross-platforms, and in last two sections, more specific information for two selected cross-platforms, Xamarin and PhoneGap, as part of our comparison for this master thesis.
- *Chapter 3 - Related work*: Briefly we mention some other works that address the same problem.
- *Chapter 4 - Cross-platform approaches comparison method*: We write about the problem statement and proposed solution together with description of comparison method.
- *Chapter 5 - Case study and implementation experience*: We explain in details our case-study design and its technical implementation.
- *Chapter 6 - Comparison results*: A cross-platform mobile development comparison for each factor individually based on our implementation experience in Xamarin and PhoneGap.
- *Chapter 7 - Conclusion*: In short, it is summarized all the research, conducted several months, which was the main aim of this master thesis.

Part of the thesis organization are also: a list of tables, list of figures and bibliography that is used in this research master thesis.

Chapter 2

Background

2.1 Introduction

In this chapter we give to a reader, general information about mobile development and cross-platforms, and in last two sections, more specific information for two selected cross-platforms, Xamarin and PhoneGap, as part of our comparison for this master thesis.

2.2 Mobile platforms chronology

A mobile application is a computer program running on a mobile device and presenting value to the mobile user [9]. The history of mobile application (mobile app) begins with the history of invention of the first mobile devices, obviously with the first mobile phones. In the beginning the aim of mobile application was only the basic software for the mobile phone with the main duties of sending and receiving calls.

In 1973, Motorola launched first portable phone. Developed by Dr Martin Cooper, who set up a base station in New York, it was the first working prototype of a cellular phone [10]. It took ten years for this prototype to be the first commercially available mobile phone. First marketed in 1983, it was 13 x 1.75 x 3.5 inches in dimension, weighed about 2.5 pounds, and allowed you to talk for a little more than half an hour [11]. It was this period, when big companies begin to develop the simplest mobile device games, first interactive mobile applications, mobile applications other than those for sending and receiving calls. It all started with the famous game "Snakes", appearing in earliest phones of Nokia. Other games also followed later such as: Tetris and Pong.

With the introduction of handled computers, notably PDAs we see the

first popular handled computer applications. As more and more people began carrying handy devices with them, and mainly because by this time the prices of this mobile devices (mobile phones and PDAs) dropped and battery improved, they began asking more features and games for their devices. Keeping clients, but by not having the resources to develop every application that is wanted by the user, was a struggle for the companies producing mobile devices. They desperately needed a way to provide entertainment to the user but not programming everything they wanted in the device. So what they did is they turned to the internet. But this turn was not the best option at that time because phones had low resolution screens, and websites were heavy with colors, pictures and other type of files, not excluding also the financial cost of the user for bandwidth requirement.

Solution to this problem was attempted to be given on June 26, 1997. On that date, three industry heavyweights - Ericsson, Motorola, and Nokia - an a relative unknown - Unwired Planet, now Phone.com - announced the creation of a new technology for delivering Internet content to all types of mobile and wireless devices [12]. This is the beginning of WAP or Wireless Application Protocol. They start sharing information and create an open standard through creation of WAP forum. Simply put, WAP was a stripped-down version of Hypertext Transfer Protocol (HTTP), which is the backbone protocol of the World Wide Web [13]. WAP sites were simpler that WWW pages and it was a great opportunity for mobile companies, but the main problem was that those who should deliver through WAP, they didn't or more precisely delivery was in a limited way.

But this was not enough. Users always wanted more graphic involved so they can interact easily, but this was impossible with WAP technology. To take advantage of these opportunities, Psion and the leaders in the mobile phone industry - for example, Nokia, Ericsson and Matsushita (Panasonic) - formed a joint venture, called Symbian, which was to take ownership of and further develop the EPOC operating system core, now called Symbian OS [14]. And this is the time when proprietary mobile platforms were born. They were also other platforms in the market such as: Palm OS, RIM BlackBerry OS, Java Micro Edition etc.

With the growing number of smartphones and other mobile devices (tablets) other manufactures start moving to mobile market and thus creating their proprietary mobile platforms as well. From the current biggest mobile platforms the first one to be released was iOS (originally as iPhone OS) by Apple in 2007, followed by Android by Google released in 2008 and Windows Phone by Microsoft in 2010 as replacement of Windows Mobile.

Most platforms offers to software developers their programming environment and programming tools to be able to code specific applications for

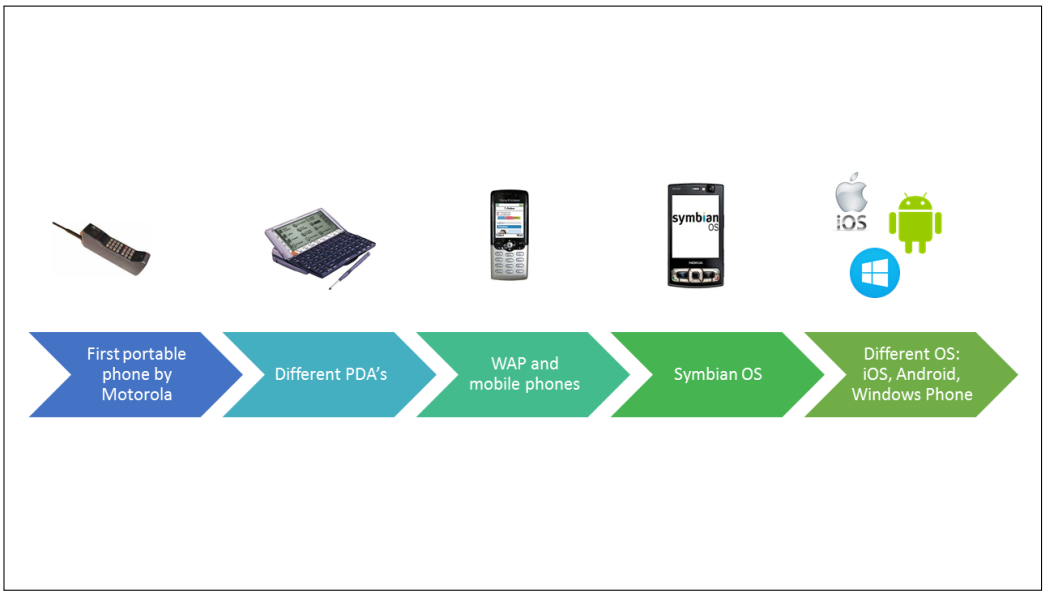

Figure 2.1: Mobile platforms chronology

specific platform. Software developers should have proper skills to be able to code in specific platforms, ie to know Apple's Objective C for iOS, Java for Android or C# for Windows Phone.

2.3 Cross platforms mobile development

For software developers, designing and implementing good application architecture is paramount to success. Enterprise software architecture activities must take into consideration myriad concerns when choosing an approach: everything from technology standards to deployment options, potential user profiles, expected user loads and don't forget scalability, extendibility, and maintainability [15]. Most of the above steps are closely linked with the choice of platform for software development. Lately, besides the software development platform, an important role is playing also the type of device on which the application is deployed.

In the world of software development, including mobile applications, up to recent times we had two main types of applications development: native application and web-based application.

A native application is a program that is in the form of an executable file in the machine language of the computer's CPU. Native applications contrast both with programs in an interpreted language such as Java and programs in a machine language that is not understood by the computer's CPU and

thus require some form of emulation in order to run [16].

A web application is just an application that is deployed on the web. It is a Web page, or series of Web pages, allowing users to accomplish a task like obtaining information and forms, shopping, applying for a job, listening to Internet radio, or any of the many activities possible on the web [17].

The same definition for native and web-based applications applies also for mobile applications (mobile apps), with the only difference, the device where the native application is deployed or web-based application is accessed is now mobile device (tablet, smartphone etc.).

The fast-growing market for mobile devices, drives the need for faster response by software development companies to develop mobile applications.

The history of the mobile applications development is closely linked with mobile devices. With the born of mobile devices, emerged also mobile operating systems. But it is very interesting that mobile operating systems have a weird history of their expansion in the market in different time periods, namely the years. While some operating systems have increased their share into the mobile market year after year, some have remained almost in the same share of global markets with a narrowly positive or negative difference over the years.

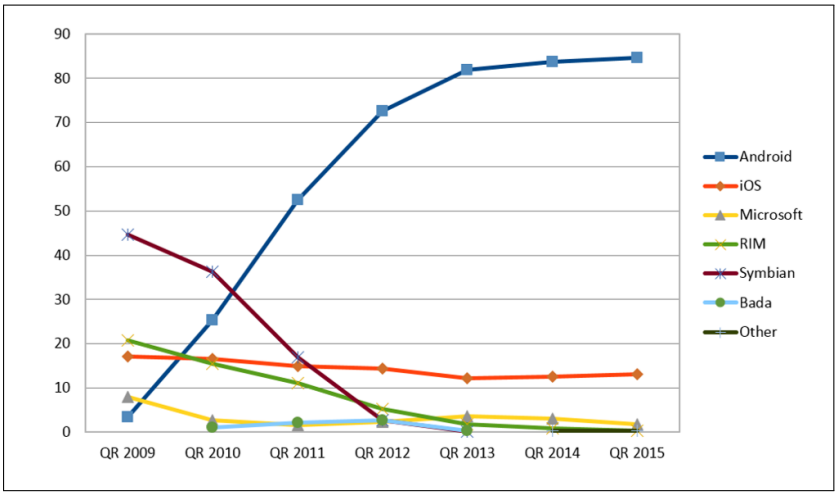

Figure 2.2: Global market share held by the leading smartphone operating systems in sales to end users from 3rd quarter 2009 to 3rd quarter 2015. Source:[1]

Given the current state of mobile share market expansion, there are currently in the mobile operating systems market two major mobile OS: Android by Google and iOS by Apple. For a developer to be able to develop a mobile

application in one of these platforms it is a must to have knowledge of a programming language, the tool and the platform framework. Such a technique for the development of mobile applications through vendor programming language and tools is known as the development of native applications.

Given the increasing number of mobile users and high costs to develop native applications, developers were really in front of a great dilemma, how to develop mobile applications in one programming language, but the deployed version to be compatible with more than one mobile operating system, especially to be able to be installed in devices with Android or iOS. The solution to this problem is the raise of cross-platform development frameworks.

Cross-platform development consists in developing mobile applications in one of the programming languages offered by cross-platform framework, and ready to be deployed in more than one mobile OS.

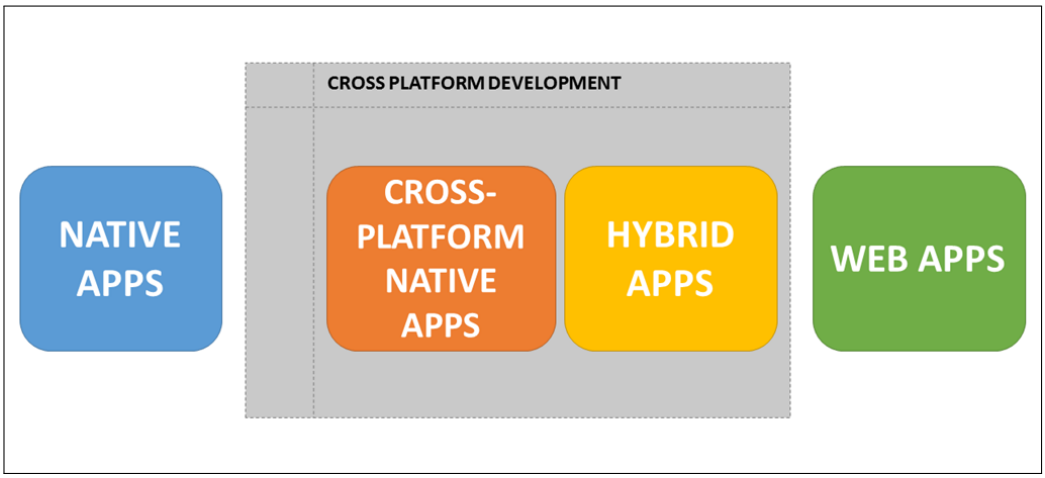

Figure 2.3: Mobile applications development types.

Currently we can group cross-platform development applications in two types: hybrid mobile apps and cross-platform apps. We call them cross-platform, because they are behaving like native but are not developed through native development environment with platform native programming language. In some literatures they also are known as Native like Apps. Hybrid mobile apps in other hand, are a special category of web applications that extend the web-based application environment through their use of native platform APIs available on a given device [18].

To summarize, the main types of mobile applications in terms of development, we can group in the following main four categories:

- Native mobile applications (native apps);

- Native cross-platform mobile applications (native-like apps);
- Hybrid mobile applications (hybrid apps);
- Web mobile applications (web apps);

Native applications development

One of the oldest ways of developing mobile applications, is the development of native mobile applications. Native mobile applications are applications developed based on tools and programming language recommended by the platform, tool and language, running only on that specific platform. They are installed on the device and can be run through the device operating system. This type of application is packaged in a solution consisting essentially of the code developed in specific programming language. Since they are developed only for specific platform, they can take full advantage of all device capabilities, such as: geolocation, list of contacts, camera etc.

Because each platform has its runtime and its programming language, then development of native mobile application does not allows code sharing and reusing between platforms, meaning every developer or developer community needs prior knowledge of specific platform libraries and classes. This implies higher development time and cost for the development community. But beside this disadvantages they have also many advantages compared to other types of mobile applications development. They are far better in user experience, and have the highest performance possible, due to no extra layer involved. Development of mobile applications in iOS, is based on either Objective-C or Swift. Android on the other hand uses Java as its programming language.

Cross-platform applications development

Cross-platform apps are native mobile applications developed by cross-platform frameworks and not platform-specific development environment. They are compatible with more than a mobile operating system, being available to download from different mobile app market such as: Apple Store or Google Play Store. Developing and application through these cross-platform frameworks, makes it possible code sharing and reusing. In some literature these mobile apps may be recognized as a native like applications.

The product developed by cross-platform framework is truly native application, and thus making it very different from hybrid mobile applications. Through cross-platform framework, applications development community does not need to have prior technical and development (programming)

knowledge for several platforms, but it will be enough to know the tools and programming language offered by cross-platform.

All products use a sort of runtime or interpreter. This kind of runtime or interpreter is used depending on the product and the target platform for product development. To provide the possibility of code sharing and reusing between platforms, for development of all products (for all platforms) it is used a single programming language.

In general, we can divide cross-platform framework architecture in two parts: code sharing and specific platform parts. Depending on the cross-platform framework, code sharing may slightly changes, but in general the following layers are included in the code sharing part: business logic layer, domain layer, data access layer, service access layer and shared UI forms. In other hand, specific platform application part, has the following: UI layer; and platform specific code.

Because of this architecture, these applications have 100% access in device native core APIs for each platform. This includes but is not limited to native device functionalities such as camera, connection, contacts, file system, geolocation, media, storage etc. Compared to mobile native applications developed on platform-specific development environment, the development of native cross-platform applications has a lower cost and development time, and this entirely thanks to development in a single programming language.

Hybrid applications development

Hybrid mobile apps, are a special category of web applications that extend the web-based application environment through their use of native platform APIs available on a given device [18].

This type of mobile applications, itself contains coding in plain HTML, CSS and JavaScript. The running logic behind of this kind of applications, unlike web-applications where access to them is through the smartphone mobile browser, is that accessing this type of application (hybrid mobile apps) is done through downloading it as other native mobile applications. Basically, the whole code (HTML, CSS and JavaScript) is packaged within the application, and the wrapper is added to start a chromeless browser. Hybrid apps use a web-to-native abstraction layer (also known as bdrige layer) that allows JavaScript to access many device specific capabilities and native APIs that re not generally accessible from the mobile web browser alone [18], such as camera, contact, geolocation etc. For the community of developers, this is a big relief, because all the previous knowledge used in the development of web applications can also be used during the development of hybrid mobile applications.

So, a hybrid mobile app it is called hybrid, because it is a mobile web-application, but it comes together with a new but very important part, native part, called: the wrapper which is provided by chosen hybrid framework, enabling this type of mobile application to be used in mobile application stores.

Comparison of applications development techniques

Comparing different mobile development solutions/techniques it is never an easy thing to do. To have a correct comparison, usually we have to rely on some specific parameters, and then based on these parameters we get a comparison result. The following criteria/parameters are used for this comparison: development cost and time versus better user experience.

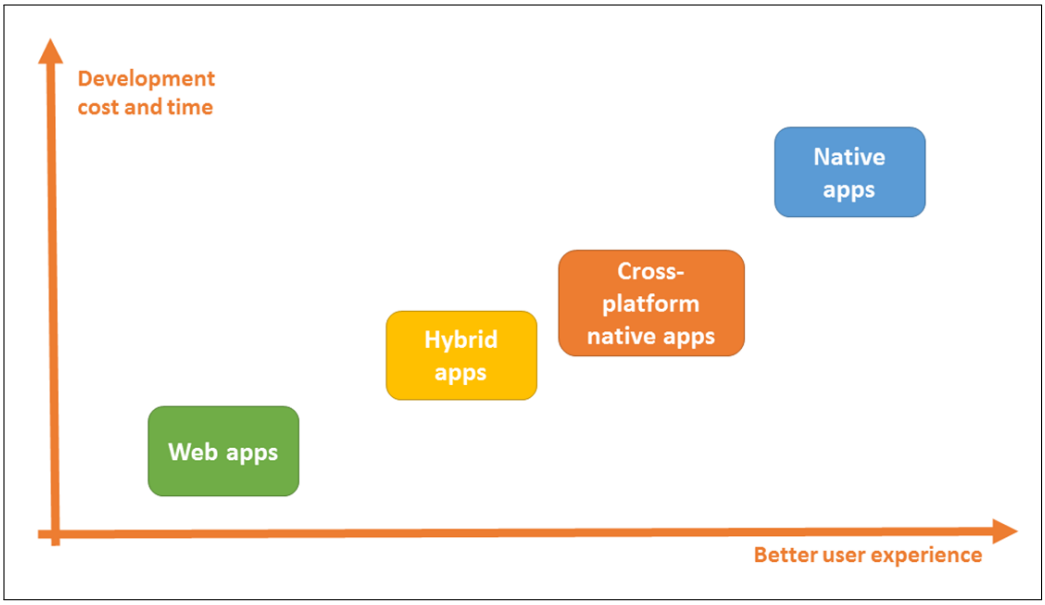

Figure 2.4: Mobile applications development types comparison.

According to our simple comparison, the cheapest solution in terms of development cost and time (first criteria) is web-based application. This is because developers code in one programming language and they deploy the application on the server, from where users access it through their mobile browsers. But user experience (second criteria) for this solution leaves much to be desired. This is because user at the beginning of running the application faces a big obstacle, to have always a connection to be able to run the application. Another issue is compatibility with different browsers and the lack of access to mobile device capabilities. Web-based applications cannot

use mobile device capabilities such as: location (GPS), camera and other important capabilities for the user.

The ability to use mobile device capabilities but keeping pretty much the same development process as for web-based application but with a bit more efforts, is positioning hybrid mobile application in better position comparing to web-based position in terms of better user experience (better design, interaction and ability to have access in mobile device capabilities such as: camera, geolocation, contact list etc.). But since it needs extra efforts for developers, that means the development cost and time is higher compared to web-based apps. Also another factor is rendering speed, which in this case is slower comparing to native and native-cross-platform mobile applications.

And almost same pattern goes also with last to types of mobile application development types. Better use experience always means high mobile application development costs and more time for development process. The highest possible performance for mobile applications is guaranteed if the mobile app is developed as native application, for example in the case of programming games, then native application development is the choice. But the code sharing is not possible for native application, and that means, developers have to code the same code in different languages for different platforms.

So to summarize a general comparison, when it comes to decision for which category of mobile application development to go, the real answer is: there is never an easy decision and this is because there is no perfect solution in the market. The choice is always depending on a lot factors and different situations, that developers might face during the process of developing a mobile application, but one thing is for sure, better user experience always goes together with high costs.

2.4 Xamarin

Cross-platform apps are native mobile applications developed by cross-platform frameworks. They are compatible with more than a mobile operating system, being available to download from different mobile app market such as: Apple Store or Google Play Store and install in more than one mobile operatin system such as: Android and iOS. Developing an application through these cross-platform frameworks, makes it possible code sharing and reusing. Currently in the market we have different cross-platform frameworks making possible the development of native-cross-platform mobile applications. Among the most popular one is Xamarin. Xamarin allows development of native-cross-platform mobile applications using the programming language C#. Xamarin is unique in this space by offering a single language – C#,

class library, and runtime that works across all three mobile platforms of iOS, Android, and Windows Phone (Windows Phone's native language is already C#), while still compiling native (non-interpreted) applications that are performant enough even for demanding games. Each of these platforms has a different feature set and each varies in its ability to write native applications – that is, applications that compile down to native code and that interop fluently with the underlying Java subsystem [19].

Integrated Development Environment (IDE) is an important part of every development framework. Developers of Xamarin cross-platform mobile applications can use currently two available IDE, Xamarin Studio and Visual Studio with Xamarin integration (Xamarin component within Visual Studio). Depending on the usage of Integrated Development Environment, users can develop mobile applications for one ore multiple platforms. Not all commercial Xamarin products can run in all computer operating systems and not all of them can produce an application that can run in multiple mobile operating systems[20].

While Xamarin products such as: Xamarin.iOS, Xamarin.Android, Xamarin.Forms may run on Windows and Mac OS X, the same scenario is not also possible for Xamarin Intergrated Development Environments, Xamarin Studio and Visual Studio with Xamarin Integration. Indeed an IDE, Xamarin Studio is designed for both Mac OS X and Windows OS but Visual Studio with Xamarin integration can be run only on Windows OS computers. To develop for iOS on Windows computers there must be a Mac computer accessible on the network, for remote compilation and debugging. This also works if you have Visual Studio running inside a Windows VM on a Mac computer [20].

2.4.1 How does Xamarin work

Writing mobile applications code through just a single programming language, by making possible code reusing and code sharing through different platforms, and at the same time implementing this code natively depending on targeting platform, is making Xamarin as one of the most interesting and attractive cross-platform framework in the market.

The main logic behind the working of Xamarin cross-platform framework is by dividing the process of mobile applications development in main two groups: Business Logic (including: business logic classes, database access, network and other common features) and User Interfaces. In the first category falls all of the mobile applications code that represents the way the application should work, mainly the code behind the application that takes care of the application events. Thanks to Xamarin, this code which it is

known also as the core of every application (including mobile applications) it is written in only one programming language – C#, and can be reused or shared across all targeting platforms. The second category is known as User Interfaces (UI). In this process, Xamarin implements and builds UI controls. The main reason that makes Xamarin-based mobile applications native-cross-platform or looking and behaving completely like other native mobile applications with the only difference being developed not through vendor IDEs, is exactly the way how Xamarin implements mobile application UI controls. They are implemented natively for each platform such as: iOS, Android or Windows Phone.

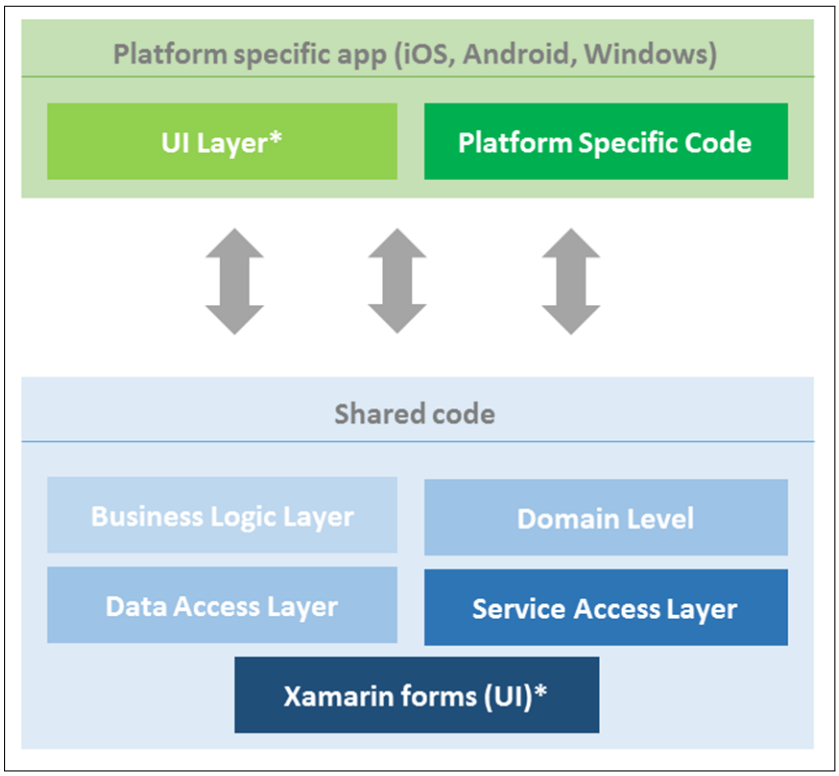

Figure 2.5: Xamarin mobile applications architecture.

Through Xamarin, developers community can use state of the art mobile applications development architectural design patterns such as: Model View Controller (MVC) or Model View ViewModel (MVVM). Layering the application architecture is another benefit of Xamarin-based mobile applications development. By using layering architecture for core functions, developer community can re-use and share the code across different supported platforms. Business Layer,Service Access Layer, Data Access Layer and Data

Layer represents the core library of the application development.

Xamarin offers to developers community two ways of implementing native User Interfaces controls, either by adding them programmatically to the application screen or adding these controls through visual designer by using already well known function in computer applications: drag-and-drop. In either way they are using native User Interface toolkits. These UI controls known also as screen layouts are stored in their respective files depending on targeting platform. In iOS they are stored as *.storyboard file, in Android they are stored as *.axml files and in Windows Phone they are stored as *.xaml files.

Compiling process of the Xamarin-based mobile application source code is very interesting process. The C# source makes its way into a native app in very different ways on each platform: iOS – C# is ahead-of-time (AOT) cmpiled to ARM assembly language; Android – C# is compiled to IL and packed with MonoVM + JIT'ing; and for Windows Phone – C# is compiled to IL and executed by the built-in-runtime, and does not require Xamarin tools [21].

Distribution of Xamarin-based mobile application development to the relevant mobile application stores is the final step before the application can be downloaded by the user and installed in his or her mobile device. Distribution process itself is not a complicated process but it must be noted that some distribution requirements may be different according to mobile application store and targeting platform. Android mobile applications can be distributed through more than one mobile application store. They can be distributed in Google Play, Amazon App Store for Android, Samsung Apps etc. iOS and Windows Phone mobile applications in other hand can only be distributed through platform operated mobile application stores: App Store for Apple and Marketplace for Windows. While some mobile application stores have very relaxed rules to check the mobile application before it is approved to be distributed for that specific application store, other application stores have more strict rules that application needs to pass. Before publishing mobile application, for every store it is neccessary to create an account.

2.5 PhoneGap

Hybrid mobile apps, are a special category of web applications that extend the web-based application environment through their use of native platform APIs available on a given device [18]. They are coded in: HTML, CSS and JavaScript, and later this code is packaged within the application, and the wrapper is added to start a chromeless browser. Hybrid apps use a web-to-

native abstraction layer (also known as bdrige layer) that allows JavaScript to access many device specific capabilities and native APIs that re not generally accessible from the mobile web browser alone [18], such as camera, contact, geolocation etc. Currently in the market we have different hybrid applicatons frameworks making possible the development of different cutting edge hybrid mobile applications. Among the most popular one is PhoneGap. PhoneGap allows development of hybrid mobile applications using JavaScript.

Taking into account that fact that PhoneGap is open source solution do develop and build hybrid application with HTML, JavaScript and CSS, means this is free for developers and companies. Actually with the current license developers can companies can use PhoneGap to develop applications that are for free or commercial use.

As an open-source framework, with possibility to use it for free, is making PhoneGap one of the most used cross-paltform mobile development frameworks. Especially this is true because users do not need to much more skills than skills they have to build a mobile website.

As with any other framework and platform, also developing in PhoneGap needs some necessary tools. Currently PhoneGap offers: Desktop App or PhoneGap CLI and PhoneGap Developer App. Desktop App is a drag and drop option to create PhoneGap mobile application. Desktop app is as alternative to PhoneGap CLI which is same function but with line interface approach. On this other hand, PhoneGap Developer App is a mobile app that runs on devices and allows you to preview and test the PhoneGap mobile apps you build across platforms without additional platform SDK setup. It automatically provides access to the PhoneGap core APIs providing instant access to the native device features without having to install any plugins or compile anything locally[22].

Designing and implementing a mobile application that is already build using web-technologies, but same time the application needs to be deployed to more than one mobile operating system (Android, iOS, Windows Phone etc.), needs to access some features of mobile device such as: camera, gps or calendar, and above all the development team has no time or it is costly to learn new native programming languages, then the perfect tool to implement this kind of mobile applications is PhoneGap.

2.5.1 How does PhoneGap work

PhoneGap is open source cross-platform mobile development framework to build mobile applications using HTML (HyperText Markup Language), Java Script and CSS (Cascading Sytel Sheet). In other words, PhoneGap is building hybrid mobile applications.

Since mobile applications developed in PhoneGap are hybrid applications, this means they are developed – designed and coded as any other mobile web application, and later they are packaged and wrapped to a web view to start a chrome less browser for any supported mobile operating system.

A web view is a native application component that is used to render web content (typically HTML pages) within a native application window screen. It's essentially a programmatically accessible wrapper around the built-in web browser included with the mobile device [22].

Taking this architecture into account, in principle we can divide they way how PhoneGap works in three parts:

- Deigning and developing web application;
- Using PhoneGap APIs to have access device features such as: camera, geolocation, file etc.; and
- Packing the already developed web application into mobile application ready to be used on mobile device.

Designing and developing web application would not be any problem for the developer or development team if necessary web skills are in place. So, if there are good skills is HTML, CSS and JavaScript, basically if developers can make a web application according to the user needs, they can be sure that whatever web-application does, it will do also as mobile application packed by PhoneGap.

No matter how good is designed and developed a mobile application, running only in mobile browser, makes it impossible for the user to interact with device components such as: files, contacts, camera or geolocation. So to make a mobile application closer to the user needs, we need to have access also in some of this device components. Through PhoneGap we can have access in this components by using PhoneGap APIs.

Implementing a feature that needs access to mobile device sensor such as: camera or geolocation is simple process as well. This can be done thanks to a lot of plugins available through PhoneGap project. This is particularly done using by calling API through JavaScript, and then application translates into native API. PhoneGap supports different types of device features meaning there are currently available APIs for the features such as: geolocation, compass, camera, contacts, media, files, storage etc.

But by finishing mobile web application with or without access to mobile device features, this mobile web application needs on more step to be ready to be distributed to different mobile apps market and later to be used by the

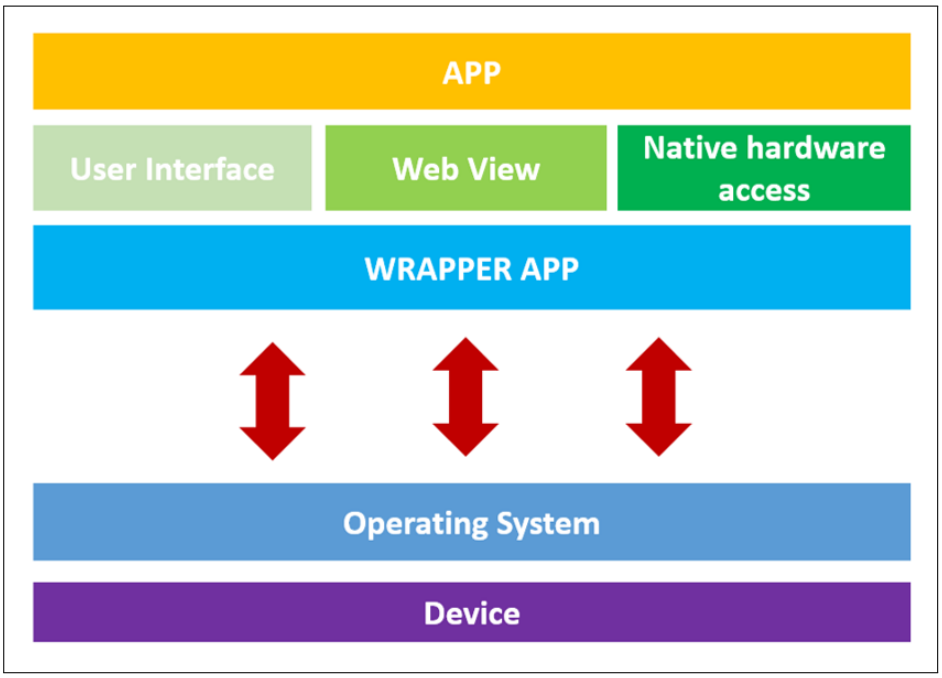

Figure 2.6: Hybrid mobile applications architecture.

user. This step is the one, where web application is packaged into application and becomes ready to run on mobile device.

Each of the mobile device platforms supported by the PhoneGap project has its own proprietary tools for packaging or building native applications for its platform. To build a PhoneGap application for each supported mobile platform, the application's web content (the HTML, CSS, JavaScript, and other files that comprise the application) must be added to an application project appropriate for each mobile platform and the be built using the platforms proprietary tools [22].

And of course as the last step, after having a hybrid application ready to be used on targeted mobile operating systmes, is to distribute it. As for Xamarin based mobile application development, distribution of the PhoneGap based mobile development application to the relevant mobile application stores is the final step before the application can be downloaded by the user and installed in his or her mobile device. PhoneGap based mobile applications developed for Android mobile operating system can be distributed through more than one mobile application store such as: Google Play, Amazon App Store for Android, Samsung Apps etc. iOS and Windows Phone mobile applications in other hand can only be distributed through platform operated mobile application stores: App Storeand Marketplace.

Chapter 3

Related work

During the recent years, many cross-platform mobile development framework were born. Some of them are still in the market and some of them have merged. With the born of these cross-platforms, it has increased also the interest for using them, more precisely, a various studies and analyses are being made with the intention to see the difference and what this cross-platform offers to the client.

Though not in the big numbers, still the are some studies published so far where the main intention is the comparison of two different approaches of the mobile applications development - native like / cross-platform native and hybrid, approaches that are represented by specific development tools: Xamarin and PhoneGap. Most current published studies that aimed to compare the two cross-platforms that are also the subject of our study (this master thesis) mainly describe the comparisons made for more than two cross-platforms and unlike the research conducted on this master study, they lack in detailed analysis with a detailed description of the differences and what is more important in almost all literature studied for the purpose of this topic, they used different study method to reach to the final outcome.

Besides the vendor material or specific platform books for each of our targeted cross-platforms, we were not able to find in big numbers previous academic publications for the purposes of these cross-platforms comparison or either comparison of more cross-platforms. During our research, we were able to identify about single digit number of academic publications with the aim of finding a solution to the same problem or closely related, to what is the purpose of this master study. After finding them, these papers have been studied as part of this master thesis.

Lack of big number of academic research papers in this field shows that for such matter there is still room for future research, and especially taking into account the fact that now Xamarin can be used free of charge through

the Visual Studio Community Edition or Xamarin Studio for Mac computers, making both platforms (ie Xamarin and PhoneGap) easy accessible by every developer or development community.

Below we will mention each academic research (relevant work) used for the purposes of this master topic. For each study separately will give a brief description about the paper and what was the conclusion of the author / authors.

In [2], Pavel Sergeyevich Ptitsyn and Dmitrv Vladimirovich Radko from Research Institute of Semiconductor Engineering and Voronezh Innovation and Technology Center, Russian Federation, are doing a analysis of cross-platform technologies for mobile applications development.

Indeed they are not analyzing only Xamarin and PhoneGap but more platforms that are currently in the market. Their analyses are based on the following platforms: Appcelerator platform, Koneyone platform, PhoneGap, IBM work light, Telerik platform, Xamarin and Rho mobile.

Besides giving a brief description for each platform, paper continues with methodology, criteria and analysis results. According to the paper in the testing and evaluation phases were involved four people/experts. Evaluation of cross-platforms in this paper is done according to the main nine criterias, and each criteria has some sub-criterias. For each sub-criteria, every cross-platform could be evaluated with minimum 0 points and maximum 5 points. Description of each point (from 0 to 5) is given as below:

- 0 - Absence of related functionality;
- 1 - Does not satisfy the requirements;
- 2 - Partially satisfies the requirements;
- 3 - Satisfied, but there are serious drawbacks;
- 4 - Satisfied, but there are minor drawbacks;
- 5 - Fully satisfies the requirements;

According to the authors: the final rating evaluated cross-platform technologies are the sum of estimates for all functional criteria. The maximum possible overall rating of the technology equals the number of criteria multiplied by the maximum possible score is 235 (47 x 5). The averaged overall rating, exhibited by the group of experts, was determined by the following formula: Sum of final ratings of all experts divided by the number of experts.

	Xamarin	PhoneGap
Development environment	18	15
Project management tools	10	10
Testing environment	24	21
Publish and deploy applications	10	9
Analytic tools	20	3
Cross-platform development features	33	22
Visual design environment	23	18
Application store	8	0
The quality of the documentation and support	15	13
Total estimation	161	111

Table 3.1: Results of testing and evaluation of cross-platforms for mobile application development according to paper [2].

Paper is presenting the evaluation results for all cross-platforms, but we will present here only the results for Xamarin and PhoneGap. The evaluation results for these two cross-platforms based on their analysis is as follow:

In the end the authors conclude that: the main objective was to determine the effectiveness of existing cross-platform technologies for mobile applications development in terms of flexibility mobile application development process including such aspects as coding, debugging, testing, deployment [2].

And based in evaluation and testing criterias used in this paper, in general Xamarin is more effective than PhoneGap.

In [3], Assist. Prof. Dr. Volkan Tunali from Celal Bayar University, Turkey and Assoc. Prof. Dr. Senol Zafer Erdogan from Maltepe University, Turkey are doing a comparison of popular cross-platform mobile application development tools.

More precisely they are conducting a comparison of more cross-platform mobile development tools. They are comparing: PhoneGap/Cordova, Xamarin, Appcelerator Titanium and Smartface App Studio. In this paper authors claim to: provide a pragmatic comparison from different perspectives like ease and cost of development including programming language and tool support, end-product capability and performance, security and community support.

Authors begin the paper by offering a brief description for four approaches of mobile applications: mobile responsive web applications development, native applications development (platform-based native), hybrid applications development and cross-platform native applications development. Paper goes further by giving a description of compared platforms, describing criteria and

	Xamarin	PhoneGap
Output	Native	Hybrid
Supported platform	Cross-platform and OS specific	Cross-platform
Single code base	Only for Xamarin.Forms	Yes
Development environment	Visual Studio or Xamarin Studio	Jetbrain Webstorm, Sublime Text, Intel XDK, Eclipse
Development language	C#	HTML5, JavaScript, CSS
Design Editor	Partially with designers	Depends on tech
Code Based Design Editor	Yes	Depends on tech
iOS development on Windows	No (Workaround available with a Mac on network)	No
Mac required for iOS test and debug	Yes	Yes
Developer adaption	C# know-how	Web know-how
Look and feel, sense, UX	Native	Native-like
UI Responsiveness	Smooth	Not smooth
Performance	Faster	Fast enough
Device specific features	Yes	Depends on tech
Device sensors	Yes	Needs plug-in
Offline storage	Device storage	Limited
Stability	Higher	Higher
Security	Secure	Depends on tech
Community	Large	Large
Technical support	Community and inexpensive	Community and inexpensive
Risk	Dependency	Browser compatibility issues

Table 3.2: Results of testing and evaluation of cross-platforms for mobile application development according to paper [3].

comparison result. A list of criterias has been set and then all studied mobile development cross-platforms are compared against these criteria.

For each comparing criterion is given an evaluation for all compared platforms. Indeed, unlike in previous paper where the platforms for each criterion receive a numerical grade (evaluation), in this case the assessment is not numerical but is descriptive. Thus, making the selection a priori of the 'winner'

form the list of cross-platforms more difficult. Indeed by giving a description evaluation the selection of the best platform depends very well on the development situation.

Result section of the paper presents the comparison results for all mobile development cross-platforms according to the given criteria. Results are shown in tabular form. Because the subject of this thesis is only comparison between Xamarin and PhoneGap, here we are presenting only the compared results for PhoneGap and Xamarin (table 3.2 [3]).

Comparison criterias include both functional and non-functional criteria such as: look-and-feel, responsiveness, performance, stability and usage of device sensors such as: camera, GPS etc.

According to the authors hybrid mobile development approach which in this cases is represented by PhoneGap, it is very desirable taking into account the number of mobile platforms that are supported by it. But as downsize it is mentioned that application developed through this are not native and user might experience problems when navigating through interface. As for Xamarin, authors claim that is very good option for the developers with programming experience in C# and also compared to PhoneGap, mobile app developed by Xamarin is native. However according to authors if Xamarin.Forms are not used, developers should still know to code for each platform. They also claim that as disadvantage is that licenses should be purchased separately for every targeting mobile operating system, a true fact at the time when the paper was published, but since summer 2016, users can use Xamarin.Forms free of charge (community version).

In conclusion according to this paper, authors claim that: Despite their common objective, they all have distinctive features that make each one superior and preferable to the others. Therefore, software vendors need to understand the advantages and disadvantages of them, evaluate each one considering their own specific development requirements and constraints, and then make their choices wisely. Therefore, software vendors need to evaluate each one carefully, and we believe that it is best to try the ones that seem reasonable on small pilot projects to see if they are satisfactory in terms of the criteria we described [3].

In [23], V.V. Gerasimov and S.S. Bilovol and K.V. Ivanova are doing a comparative analysis between Xamarin and PhoneGap for .Net.

According to authors: Comparative analyses between technologies Xamarin and PhoneGap for .NET is discussed. The features of these technologies, their advantage and disadvantages, their sphere of application and prospects of their development are given[23].

Even though the authors claim that a comparative analyses is done, indeed the authors throughout the paper are not evaluating the cross platforms

against each other according to specific criteria, but they are describing each cross-platform independently - separately, Xamarin and PhoneGap, by highlighting advantages and disadvantages for each platform. The only section of the paper where a clear comparison between two platforms is done is conclusion. There, authors try to give an answer to the user for which platform to use in which circumstances. But, lack of evaluation through some specific criteria, makes the comparison result not clear enough.

Regarding the PhoneGap mobile application authors concludes that this mobile application indeed is just a website (mobile website), but implemented as mobile app through HTML Rendering Engine (WebView) with the inteface created with HTML, CSS and JavaScript, a common interface that can be shared across different mobile OS. But besides this, authors mention also the well-known fact of PhoneGap, that PhoneGap mobile application despite being a mobile website can use also mobile device hardware features such as: camera, storage, GPS etc. All these features can be used through PhoneGap plugins.

As for Xamarin authors mention the connection between the platform and native API for mobile OS. Besides describing in technical way the architecture of Xamarin, authors also mention the structure of the application developed through Xamarin which is divided in two parts: platform specific code and shared code. Xamarin.Forms is also described as the way for the developers to create common user interface for all targeted mobile OS. By using Xamarin.Forms according to this paper, the quantity of the shared code has increased to 90%.

Cheap in developing user interface, a wide variety of available plug-ins with various functionality with a quite good performance is how PhoneGap is described by the authors. On other part, despite small number of supported mobile operating systems, a powerful tool that provides .NET frameworks and native libraries is how Xamarin is described by the authors.

Taking into account the final opinion of the authors for the compared platforms in this paper, they conclude that for small cross-platforms mobile applications a good solution is PhoneGap, while for large mobile applications the better solution is Xamarin.

In [24], Mukesh Prajapati and Dhananjay Phadake from University of Mumbai, Indida and Archit Poddar from Manipal University Jaipur, Indida are conducting a study specifically on Xamarin cross-platform framework.

In fact, this paper does not compare the cross-platform frameworks for the development of mobile applications, but it is a paper that explains specific cross-platform – Xamarin in more details than other papers, which are part of this related work. Here, authors provide explanations for this specific framework, an explanation detailed in different section, where each section

represents one of the framework main components.

Besides the description of some techniques used for the development of mobile applications in use before Xamarin is born, rest of the paper attempts to describe the functioning of the cross-platform and that by dividing the entire description in several categories, in order to be meaningful.

They begin this paper by explaining the architecture behind Xamarin, by dividing it into several layers: User Interface, Application Layer, Business Layer, Data Access Layer, Service Layer and Data Access Layer. It is also worth mentioning that according to the authors, Xamarin supports both the development of software development patterns: MVC and MVVM.

Authors go further by describing Xamarin features, starting with Xamarin.Android, a feature of which, according to the authors Xamarin offers the possibility of developing mobile "native " applications in the Android platform by using the same controls that normally would be used in Java, but now in this case all of these by using the programming language C#.

A description in the same way is given also for Xamarin.iOS, a feature through which is given the possibility to the user to create native mobile applications for iOS, where also here, developers can develop most of the controls by using programming language C#, controls that normally would need programming skills in X-Code or Objective-C.

An architecture and flow of processes is described separately for Xamarin.iOS and Xamarin.Android.

They also provide more information for the other Xamarin components such as: Component Store and NuGet, places where developers can access and download necessary development libraries for each mobile operating system - platform, whether it is Android or iOS.

Authors also describe in more details one of the development patterns for Xamarin cross-platforms mobile applications, pattern: MVVM - Model View ViewModel. A pattern that separates the business logic, according to the authors of the paper in two separate objects: View and ViewModel. While User Interface View deals with the application, ViewModel main duty is to handle the classes dealing with business logic of mobile application developed by Xamarin.

Paper is concluded by the authors by providing a description for the another component of Xamarin, one of the best used tool, called Xamarin.Forms a tool that allows the creation of a UI layouts once and later can then be distributed to all targeted mobile operating systems.

Although paper provides an interesting description of the Xamarin cross-platform framework and its use in the development of mobile applications, this paper or more precisely its findings do not rely on specific caste study experience but is more a paper where its findings are based on researching

current literature and general experience by the authors.

In [25], Gatis Vitols, Ingus Smits and Aleksejs Zacepins from IT Competence Centre, Latvia are describing their experience on issues that they faced during the development of a hybrid mobile application case study with PhoneGap: Insurance Mobile Application.

Also in this paper, authors are not evaluating different mobile development cross-platforms frameworks, but they are letting us know their findings of evaluation of PhoneGap – hybrid cross-platform mobile development framework. Even though there is no comparison between different cross-platform frameworks, their finding are quite good and they are used on the practical experience during the development of one case study mobile application: Baltic Insurance House (BAN). An application developed by using the following technologies: HTML5, CSS3, JavaScript and jQuery. Primarily the mobile application is developed to be used on mobile operating systems: Windows Phone, to be followed also by other mobile operating systems such as: Android and iOS.

This mobile application consists of four subsystems: SOS, branches of organization, buy insurance, settings. Application functionality includes navigation options by using mobile device platform native features – Back button and Home button, as well as touch navigation. In addition it handles connection and automatic refresh of interactive map that shows locations of organization branches. Also it has integrated purchase subsystem for BAN services [25]. Two devices where used to test how the mobile application behaves in real environment: LG Nexus 4 for Android and Nokia Lumia 820 for Windows Phone Operating System (OS).

After developing and testing the behavior of their case study application in real environments - Android and Windows Phone, authors arrived in some comparison conclusions between running this application in Android and Windows Phone. In fact they are making nine comparison conclusions regarding different components such as: functionality, rendering, user interface and development efforts.

By summing them up, in terms of functionality authors find that it is equal for both mobile operating systems. As for user interface and user interaction they find that besides the application is working in both operating systems there are still some concerns or some points to take into account when developing with PhoneGap such as: call function is different, battery level and time is not visible in both platforms when using all-device screens. Also according to authors if the type of font is not preset then default fonts are different in different operating system. In terms of development efforts, authors claim that developers should be familiar with Android references to element dimensions and requirement if application is targeting Android OS.

In the end authors conclusion is: Based on research authors can conclude that up to 95% of developed code is not sensitive on device, but 5% is sensitive. This 5% consists of fonts, resolution and display size. If the application supposed to display extensive amount of data simultaneously and perform complicated calculations, native approach for development should be considered. [25]

Even though this is a good paper in terms of professional findings during development of hybrid mobile application, and indeed is doing a comparison how the same application behaves in two different mobile operating systems, it still lacks in terms of comparing PhoneGap with another cross-platform framework currently in the market. Also the comparison between different mobile operating systems for the mobile application developed in PhoneGap, lacks a set of predefined criteria, where the product would be evaluated against them.

Chapter 4

Cross-platform approaches comparison method

4.1 Introduction

In this chapter we give to a reader an idea about our problem definition and proposed solution to the problem already defined, including the comparison method description.

4.2 Problem definition

Taking into account current growing mobile market and current use of cross-platforms, we decided that the issue of cross-platform approaches to be the main focus of our research for this master thesis.

The idea and the main purpose of this research is a thorough professional analysis and comparison of these cross-platforms. A comparison, based on arguments achieved after an evaluation and thorough study of the current literature. Because current market is filled in with a range of cross-platforms, each with its specific features, to be more accurate in our thesis, we selected only two for comparison. Selection of two platforms is based on several key factors such as: the type of product offered, the financial cost of use of the platform and their innovations in the market.

PhoneGap and Xamarin are our cross-platform mobile development approaches of whom will be subject of research-comparison for this master thesis. They are chosen specifically because of different platform types, more precisely the type of product that is developed through them. The product - mobile application of PhoneGap is hybrid mobile application, while Xamarin product is cross-platform application - cross-platform native application.

Given the analysis of cross-platform mobile development approaches and our implementation experience from case-study in two specific cross-platforms, then the problem we are trying to solve in this thesis is as follow:

Taking into account the following factors:

- *Graphical User Interface (GUI);*
- *Architecture;*
- *Services and sensors;*
- *Local data storage;*
- *Development efforts;*

"Which of the cross-platforms mobile development frameworks, compared in this thesis - each of them representing specific mobile application approach, is the most convenient one for the development of new mobile application?"

4.3 Proposed solution

The best way to do a proper comparison, besides relying on relevant work is by designing and implementing a mobile application as case-study exclusively for this thesis research. Then this mobile application to implement in both our compared cross-platforms, Xamarin and PhoneGap, thus seeing the similarities and differences.

For the case-study purposes we design and develop a review mobile application - named: ReviewPOI. A mobile application with the main purpose to give the users the possibility of knowing what was the opinion of other users/visitors (their review) regarding a specific point of interest in a specific place. A point of interest can be anything worth visiting in a town or city, such as: hotel, restaurant, beach, museum etc. If a point of interest is not in the system, users can add them. If however a point of interest is in the system they can add a review about that specific point of interest. They can see all the time reviews for any point of interest in the system.

Later, based on our development experience for our case study with both compared cross-platforms, we group and evaluate the comparison results according to the already selected factors determined in our problem definition. For each comparison criteria we give an evaluation for both cross-platform development frameworks.

4.4 Comparison method

Comparison of cross-platform mobile development frameworks is never an easy process. To make a proper comparison we always should be based on several specific comparison factors. And this makes the comparison result closely related to the comparison factors we select.

Besides selecting comparison factors, another comparison issue is how we evaluate the comparison subject against these factors.

In our case, in problem definition we defined five comparison factors: Graphical User Interface (GUI), architecture, services and sensors, local data storage and development efforts. Because our comparison subjects are Xamarin and PhoneGap, we evaluate these cross-platforms mobile development approaches based on our implementation experience with these both cross-platforms. More precisely the result of our comparison is based on our gained experience with the five comparison factors during the implementation of our case-study in these both platforms.

Below we describe in details how we evaluate our subject for each comparison criterion individually.

Graphical User Interface (GUI) - is the first interaction between the user and mobile application. A good, well designed and looking native mobile app graphical user interface is one of the factors to increase the user satisfaction.

The main purpose of this criterion is to see whether the user interface of final product - mobile application, looks and behaves like native or not. To arrive in better evaluation conclusion we compare cross-platforms mobile development frameworks product according to the following features:

- *Page and navigation types*: main page types used in and by native applications are: *content* page, *master-detail* page, *navigation* page, *tabbed* page, and *carousel* page. In comparing this feature we will take into account what types of pages can we use when we design and implement a mobile application. Is it allowed for our project-mobile application to use more than one page type or combination of different page types or we should stick to one page type only?

- *Page layouts*: different page layouts are used and supported in native mobile applications. They include but are not limited to: *content view, grid, scroll view and linear layout*. In comparing this feature we will consider whether we can use these layouts or not and if yes can we combine more than layout type for the same project.

- *Controls*: GUI controls are features that through them, mobile user feels the application. Different controls are supported by native mobile application such as: entry, image, list, table, button etc. In comparing this feature we will take into account whether the controls offered by cross-platform mobile development framework are native looks like or not.
- *Themes*: Using default theme of mobile operating system makes the application to look and feel native. Theme is how the navigation, controls and layouts are grouped and presented to the user. In comparing this feature we will consider the default theme offered by cross-platform mobile development framework and how native it looks like.

For evaluation purpose of this criterion, the cross-platform mobile development framework with highest number of supported features is the better one in terms of Graphical User Interface (GUI).

Architecture - Good and well-organized software structure is not only important for the application functionalities but also for the maintenance phase.

The main purpose of this criterion is to see what kind of application architecture is possible by using each of the cross-platform mobile development frameworks. To arrive in better evaluation conclusion we compare cross-platforms mobile development frameworks architecture according to the following features:

- *Code sharing*: is very important feature when developing mobile applications for multiple platforms (operating systems). Through code-sharing we are able to code in one place and use the same code for every platform without the need to adopt/change it for other platforms. In comparing this feature we will consider how much share-code we can have when developing cross-mobile application.
- *Separation of responsibilities*: is a process where it is allowed that each component of application should perform well-defined purpose. In comparing this feature we will take into account if this feature is supported by our compared cross-platform mobile development frameworks.
- *Architecture patterns*:Different architecture pattern are currently used for application development. Common used patterns are MVC (Model, View, Controller) and MVVM (Model, View, ViewModel). In comparing this feature we will consider if any of this architecture pattern is supported.

For evaluation purpose of this criterion, the cross-platform mobile development framework with highest number of supported features is the better one in terms of architecture.

Service and sensors - Accessing device APIs and consuming web-services is another important feature when designing and implementing a mobile application. A lot of applications require access at least to GPS or camera and a lot of them nowadays consume data through web-service.

The main purpose of this criterion is to see how cross-platform mobile development framework is accessing services and mobile device sensors. To arrive in better evaluation conclusion we compare cross-platforms mobile development frameworks access to services and mobile device sensors according to the following features:

- *Consuming web-service*: There are different ways a platform or mobile application consumes data web-services. But, more important is how is the communication between application and web-service, i.e getting the request and returning the desired result. In comparing this feature we will consider whether it is possible to consume web-service and if yes how easy is to implement it.
- *Accessing GPS (geolocation)*: Accessing GPS (geolocation) of mobile device is how cross-platform mobile development framework accesses native feature of mobile device. In comparing this feature we will take into account whether it is possible to access GPS and if yes what is the way of getting the coordinates. Also the implementation easiness is part of evaluation.
- *Accessing camera*: Taking picture from mobile device camera is also a process that involves dealing with native feature of mobile device. In comparing this feature we will consider whether is it possible to take picture and if yes how do we get it and handle the picture. Also the implementation easiness is part of evaluation.
- *Accessing photos from photo album*: Getting a picture from mobile device photo album involves the process of accessing local files in mobile device. In comparing this feature we will take into account whether it is possible to select a picture from photo album and if yes how do we get it and handle the picture. Also the implementation easiness is part of evaluation.

For evaluation purpose of this criterion, the cross-platform mobile development framework with highest number of supported features and easier implementation is the better one in terms of sensors and services.

Local data storage - Is not always a case where mobile application reads or writes data to a database located somewhere on external server or data webservice. In many cases there are different scenarios where users want for a specific purpose either temporarily or permanently preserve some data locally in mobile device.

The main purpose of this criterion is to see how cross-platform mobile development framework handles local data and how we access them. To arrive in better evaluation conclusion we compare cross-platforms mobile development frameworks local data storage according to the following features:

- *Type of local data storage*: Currently there are different options to save data locally in mobile device and this is heavily depending on the cross-platform mobile development framework we choose. Key/Value pair and SQLite are some of the types. In comparing this feature we will consider what type of local data storage is offered and whether we are obliged to use only one of them or combination between them, if more than one local data storage type is offered.

- *Structure of local data storage*: Some type of local data storage are of *flat* structure and some of them are real SQL-based databases. While with flat data we cannot or it is very difficult to perform queries or transactions with SQL-based databases is only SQL language knowledge the skill that we need. In comparing this feature we will take into account the complexity of local data storage.

- *Local data storage size*: Size of local data storage is another key constraint. In comparing this feature we will consider the maximum size that it is allowed each type of supported local data storage.

- *Implementation process*: Accessing mobile device local data storage is really an important feature, but the complexity of implementation process is another factor that is influencing our evaluation. In some cross-platforms the process is very easy and straight forward while for some of them we need the help of some specific plugins. In comparing this feature we will take into account the complexity of implementation process.

For evaluation purpose of this criterion, the cross-platform mobile development framework with highest number of supported features and easier implementation is the better one in terms of local data storage.

Development efforts - Development efforts and cross-platform complexity are very important to developer or development team.

The main purpose of this criterion is to see how easy is to develop a mobile application through cross-platform mobile development framework. To arrive in better evaluation conclusion we compare development efforts according to the following features:

- *Development environment*: Different cross-platform mobile development frameworks are offering to the developer different development environments. Some of them are offering their own Integrated Development Environments (IDE) and some of them don't. Beside development process, another issue is how we can test the product, through emulator, real device or any other way. In comparing this feature we will consider the type of IDE is used and its complexity. Also running(testing) process of the mobile application is taken into account.

- *Available documentation*: Documentation is very important for every developer, especially if the developer is new with specific technology or when new features are implemented for specific technology. Each platform it is offering its documentation usually through their webpage. In comparing this feature we will consider the quantity and quality of documentation available for each cross-platform.

- *Development time*: Time is another crucial factor when we develop an application. In comparing this feature we will consider the time we spent to implement our case study in each platform.

For evaluation purpose of this criterion, the cross-platform mobile development framework with the best position in each feature is the better one in terms of development efforts.

Chapter 5

Case study and implementation experience

5.1 Introduction

In this chapter we explain in details our case-study design and its technical implementation. Because our case study development includes more than one technology, we describe our case study implementation separately for each involved technology. We start our implementation experience description by explaining design and implementation of WebApi for handling data part of the mobile application to proceed further with explanation of our experience on the development of our case study through Xamarin and PhoneGap.

5.2 Case study

For the thesis research purposes we implement a case study, namely a review mobile application - called: ReviewPOI. This is a mobile application developed in both cross-platform mobile development frameworks, whose main purpose is to give the users the possibility of knowing what was the opinion of other users/visitors (reviews) regarding a specific point of interest in a specific place.

5.2.1 General description

In most cases, during our visits abroad or not, besides socialization factor the main other purpose is sightseeing or shopping, mainly to visit other important tourist points or how they are called in other name: point of interests. These tourist points can be of different types, and visited by various groups or

individuals according to their interest. They might be but not limited only to the following categories: hotels, restaurants, pools, beach resorts, amusement parks, airports, shopping malls etc.

As human beings, always before want to visit any particular place or touristic object, we feel the need to have some advance information about that visit. These advanced information can be mainly logistics information such as: transport, surrounding area, safety. But besides logistics or safety information, a different category of information is also important, and this is the opinions of previous tourists about that place or object that we want to visit, mainly how they described their experience and did that touristic spot fulfilled their expectations. This type of information is called: *review*.

Given this type of information (reviews) in one side and the increasing number of mobile devices users and using frequency on daily basis on the other hand, we came to a decision, that precisely this issue, the issue of reviewing touristic places (touristic points of interest) will be as the core of our mobile application developed as a 'case -study' for this master thesis.

Thus during this case study we design and develop a mobile application that allows users to see what other users thought when they visited a specific touristic point of interest. More precisely our case-study will deal with reviews of any touristic point of interest. Besides viewing what other users said, they can also give a grade and textual opinion about the touristic points. Users can also take pictures and save them together with the review. We call this mobile application: ReviewPOI.

5.2.2 Requirements

ReviewPOI should fulfill the following requirements:

- Application-ready to be installed in mobile device with any of the following OS - Android, iOS or WindowsPhone;
- Database stored in web server. Read and Write data to be performed through web-service. In other words to be able to consume web-service.
- It should be possible to take photo directly from camera or read a photo from photo album aready in device and also know the current location of the user. So application should have access to device camera and photos. Also should have access to geolocation.
- User should be able to save data also in local data storage, before submitting them to the review database.

- Access to application is granted only to authorized users. Users to authenticate through their username and password. For non-existing users there should be a possibility to create a user account;
- User can search for their interested point of interest (PoI) through any of the following ways:
  - Search for PoI through free text search, by typing in full or just a part of: name, location or description;
  - Search for PoI through free text search, by typing in full or just a part of PoI type;
  - Get the list of all PoIs already reviewed by the user;
  - Get the list of all PoIs close to user geo-location;
- User can view details of specific PoI, including description and photos;
- User can view all reviews and photos from reviews for specific PoI;
- User can add a review for specific PoI, including a picture. Before submitting the review, user can save the review locally and after re-checking it, can submit to the review database;
- User can modify his/her profile data, except username and password;

5.2.3 Assumptions and dependencies

The following assumptions are made for this mobile application:

- There is an internet connection during the time this mobile application is used, always up and running;
- GPS (Geolocation) is available and enabled in device while this application runs;
- Camera is available and enabled in device while this application runs;

5.2.4 Entity model

In this subsection through Entity-Relationship model we present a core of our data model for our case study - ReviewPOI. Our data model for thesis project consists of six entities - Poi, Poitype, Poiphoto, Review, Reviewphoto and User. Each entity in our data model contains its own attributes. Entity with the minimum number of entities is with two attributes, while the entity with the maximum number of attributes is with thirteen.

Below we describe in details each entity and its attributes. Also the Entity-Relationship diagram is presented to show cardinalities (Figure 5.1).

- Entity: ***User*** - points out general information about the user of the system. Besides username and password it contains also contact details.

    1. Id - unique, type: integer. Autogenerated with autoincrement every time a new type user is inserted.
    2. Username - type: string. Points out the username of the user.
    3. Password - type: string. Points out the password of the user.
    4. Displayname - type: string. Points out the name of the user that he/she wants to be displayed.
    5. Telephone - type: string. Points out the telephone number of user.
    6. Website - type: string. Points out the website of the user.
    7. Email - type: string. Points out the e-mail of the user.

- Entity: ***Poitype*** - points out types of different POIs. Point of Interest (POI) can be: museum, lake, restaurant, hotel etc.

    1. Id - unique, type: integer. Autogenerated with autoincrement every time a new type of POI is inserted.
    2. Name - type: string. Points out the name of the type of Point of Interest such as: museum, lake, hotel etc.

- Entity: ***Poi*** - points out general data regarding Point of Interests. Has the following attributes:

    1. Id - unique, type: integer. Autogenerated with autoincrement every time a new POI is inserted.
    2. Name - type: string. Points out the name of Point of Interest.
    3. Description - type: string. Points out a brief description regarding the main characteristics of Point of Interest.

4. Location - type: string. Points out the name of the location where a POI is located. It can be a city or a town.
5. Locationlatitute - type: decimal. Points out the exact latitude coordinate of the POI as taken by GPS.
6. Locationlongitude - type: decimal. Points out the exact longitude coordinate of the POI as taken by GPS.
7. PoitypeId - type: integer. Points out the type of Poi. It is a Id of the poi type from entity: Poitype. Links Poi with Poi types.
8. UserId - type: int. It is id of the user who inserted POI in the system. Links Poi with the user.
9. InsertedOn - type: datetime. Points out the exact date and time when the POI has been inserted in the system.

- Entity: ***Poiphoto*** - points out photos for each Point of Interests. Has the following attributes:

1. Id - unique, type: integer. Autogenerated with autoincrement every time a new photo is inserted.
2. Photo - type: binary. Points out the photo of Point of Interest.
3. PoiId - type: integer. It is the Id of Poi to whom the photo belongs. Links Photo with Poi.
4. UserId - typet: int. It is id of the user who inserted POI photo in the system. Links Poiphoto with the user.
5. InsertedOn - type: datetime. Points out the exact date and time when the POI photo has been inserted in the system.

- Entity: ***Review*** - points out reviews in details for each Point of Interest. Has the following attributes:

1. Id - unique, type: integer. Autogenerated with autoincrement every time a new review is inserted.
2. Title - type: string. Title given to the review by the reviewer.
3. Positivecomment - type: string. Points out the positive characteristics regarding the POI given to the review by the reviewer.
4. Negativecomment - type: string. Points out the negative characteristics regarding the POI given to the review by the reviewer.
5. Scoreone - type: int. Given score for this review for category one.
6. Scoretwo - type: int. Given score for this review for category two.

7. Scorthree - type: int. Given score for this review for category three.
8. Scorefour - type: int. Given score for this review for category four.
9. Scorefive - type: int. Given score for this review for category five.
10. Scoreaverage -type: decimal. Average score of five scores.
11. PoiId - type: int. It is id of the Poi to whom the review belongs to. Links Review with the Poi.
12. UserId - type: int. It is id of the user who inserted review in the system. Links Review with the User.
13. InsertedOn - type: datetime. Points out the exact date and time when the review has been inserted in the system.

- Entity: ***Reviewphoto*** - points out photos for each review of Point of Interests. Has the following attributes:

1. Id - unique, type: integer. Autogenerated with autoincrement every time a new photo is inserted.
2. Photo - type: binary. Points out the photo of review for Point of Interest.
3. ReviewId -type: integer. Points out the Id of Review to whom the photo belongs. Links Photo with Review.
4. UserId - type: int. Points out the id of the user who inserted POI photo in the system. Links ReviewPhoto with the user.
5. InsertedOn - attribute of type: datetime. Points out the exact date and time when the review photo has been inserted in the system.

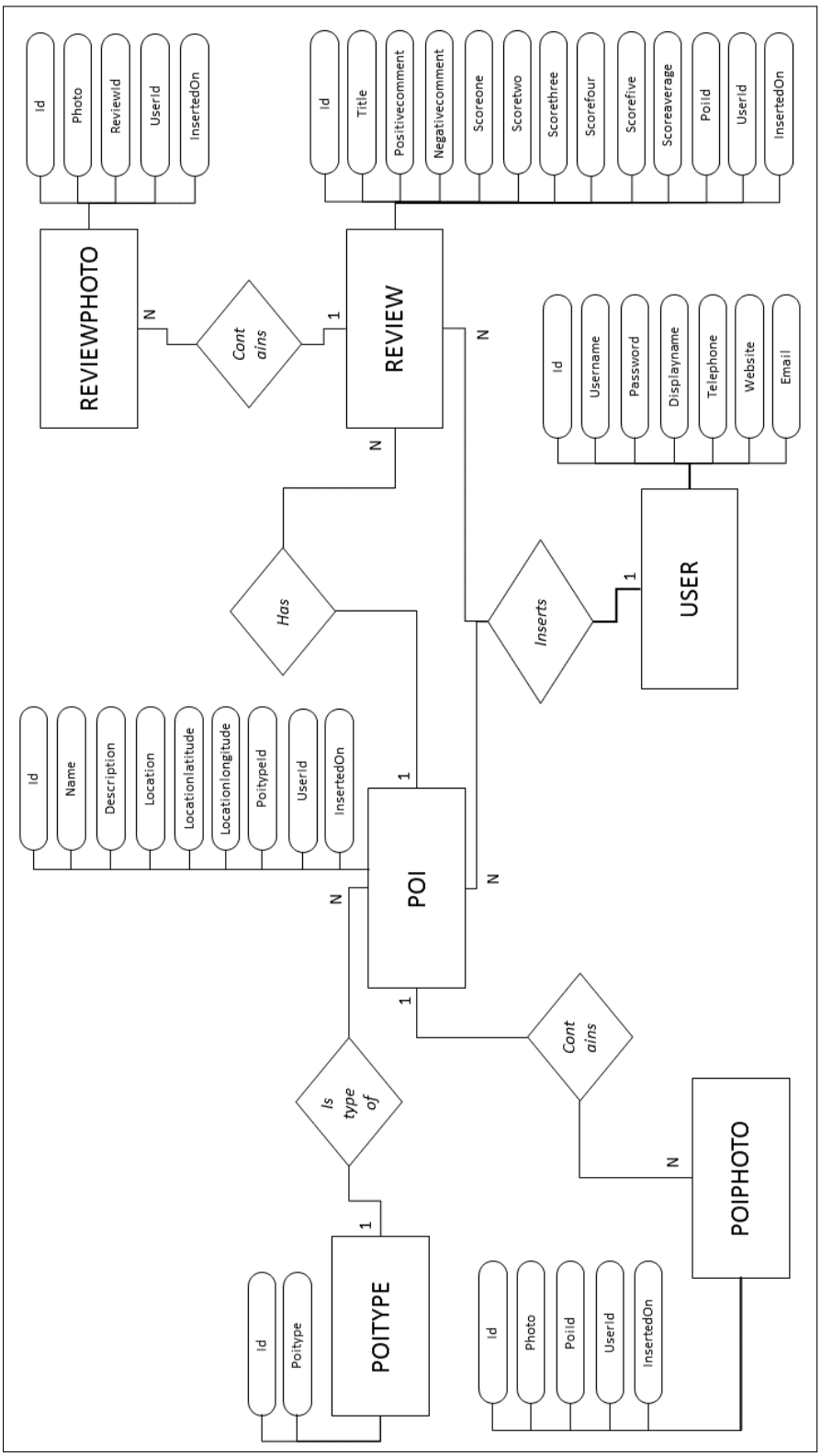

Figure 5.1: Entity Relationship diagram

5.2.5 Use cases

Besides describing our case study data model through Entity-Relationship diagram, next stage is to describe our case study in proper way and this is through Use case diagram. A use case generally describes the scenarios and interactions that an actor (or user) has with your application, and a use case diagram visually represents the actor and the application and the interactions that occur between them [13].

Further in this section, we describe in details each use case.

For our case study in this master thesis, we have identified 14 use cases.

Each case study is described by the following features:

- *Use case ID* - Unique identification for each use case.
- *Use case name* - Unique name for each use case.
- *Description* - Description of what the user can do through this use case.
- *Preconditions* - Description of conditions that have to be met, to be able to start successfully the use case.
- *Basic flow* - Description of all the necessary steps that have to be taken by the user, for the use case to be completed successfully.
- *Postconditions* - Description of the system state at the end of the use case.
- *Open issues* - Any possible open issue.

Use case ID:	UC01
Use case name:	Create new user account
Description:	Allows user to create a new account to log-in later with it in the application.
Actor:	User.
Precondition:	Internet connection.
Standard flow:	1. This use case starts when user opens application and tries to create new user account. 2. User enters his/her desired username and password and also other optional data. System save the data in the system and creates new user account for the user.
Postcondition:	User can log-in with his/her newly created account if account has been registered successfully.
Open issues:	None.

Table 5.1: Use case: Create new user account (UC01)

Use case ID:	UC02
Use case name:	Login
Description:	Allows user to log-in with his/her credentials.
Precondition:	Internet connection. User has already an account.
Standard flow:	1. This use case starts when user opens application. A login page is displayed with the following required fields: username and password. 2. User enters correct username and password. System checks user credentials and displays main page of application if they are correct.
Postcondition:	User either has logged-in successfully and got access in application, or log-in was unsuccessful.
Open issues:	None.

Table 5.2: Use case: Login (UC02)

Use case ID:	UC03
Use case name:	My profile
Description:	Allows user to modify optional data from profile.
Precondition:	Internet connection. User has already logged in successfully.
Standard flow:	1. This use case starts after user is logged in. User has to go to the page of his/her profile. 2. User sees his/her profile information. User also can modify optional data such as: display name, e-mail and website. 3. System updates data in the system if they are modified by the user.
Postcondition:	User either has updated data and goes to main page or an error is displayed.
Open issues:	None.

Table 5.3: Use case: View and modify user profile (UC03)

Use case ID:	UC04
Use case name:	Search for POI
Description:	Allows user to search for POI list according to search criteria (name, location and description).
Precondition:	Internet connection. User has already logged in successfully.
Standard flow:	1. This use case starts after user is logged in. User has to go to the page to search for POI. 2. User begins to enter the search phrase. Search phrase can be: name, location or description of the POI. 3. System gives to the user the list of PoIs.
Postcondition:	User either can get the list of PoIs or if there is no POI gets the message that there is no PoI.
Open issues:	Ordering list of PoI.

Table 5.4: Use case: Search for PoI (UC04)

Use case ID:	UC05
Use case name:	Search for PoI by PoI type
Description:	Allows user to search for POI list according to their type.
Precondition:	Internet connection. User has already logged in successfully.
Standard flow:	1. This use case starts after user is logged in. User has to go to the page to search for POI by the type. 2. User begins to enter the type in search phrase. 3. System gives to the user the list of PoIs.
Postcondition:	User either can get the list of PoIs or if there is no POI.
Open issues:	Ordering list of PoI.

Table 5.5: Use case: Search for PoI by PoI type (UC05)

Use case ID:	UC06
Use case name:	List PoI close to me
Description:	Allows user to see list of PoI close to his/her location.
Precondition:	Internet connection. User has already logged in successfully. Geolocation (GPS) enabled.
Standard flow:	1. This use case starts after user is logged in. User has to go to the page to see list of PoI close to his/her current location. 2. System gives to the user the list of PoIs.
Postcondition:	User either can get the list of PoIs or if there is no POI gets the message that there is no PoI.
Open issues:	Ordering list of PoI.

Table 5.6: Use case: List PoI close to me (UC06)

Use case ID:	UC07
Use case name:	List of PoI reviewed by me
Description:	Allows user to see list of PoI already reviewed by the user.
Precondition:	Internet connection. User has already logged in successfully.
Standard flow:	1. This use case starts after user is logged in. User has to go to the page to see list of PoI already reviewed. 2. System gives to the user the list of PoIs.
Postcondition:	User either can get the list of PoIs or if there is no POI gets the message that there is no PoI.
Open issues:	Ordering list of PoI.

Table 5.7: Use case: List of PoI reviewed by me (UC07)

Use case ID:	UC08
Use case name:	View PoI details
Description:	Allows user to see details of specific PoI.
Precondition:	Internet connection. User already search for PoIs
Standard flow:	1. This use case starts after user search for list of PoIs. User clicks the desired PoI and the system redirects him to the page with details. 2. User sees details, including textual description, reviews and photos.
Postcondition:	PoI details are shown.
Open issues:	None.

Table 5.8: Use case: View PoI details (UC08)

Use case ID:	UC09
Use case name:	Reviews for specific PoI
Description:	Allows user to see details reviews of specific PoI.
Precondition:	Internet connection. User already is in page of PoI details.
Standard flow:	1. This use case starts after user is in PoI details. User clicks reviews. 2. System shows the list of reviews for that specific PoI.
Postcondition:	List of reviews is shown if there are reviews, else 'no review' message is shown.
Open issues:	Review ordering.

Table 5.9: Use case: View PoI reviews (UC09)

Use case ID:	UC10
Use case name:	Photos specific PoI
Description:	Allows user to see photos of specific PoI.
Precondition:	Internet connection. User already is in page of PoI details.
Standard flow:	1. This use case starts after user is in PoI details. User clicks photos. 2. System shows the list of photos for that specific PoI. 3. User selects specific photo to see it.
Postcondition:	List of photos is shown if there are photos, else 'no photo' message is shown.
Open issues:	Review ordering.

Table 5.10: Use case: View PoI photos (UC10)

Use case ID:	UC11
Use case name:	Review photos specific PoI
Description:	Allows user to see review photos of specific PoI.
Precondition:	Internet connection. User already is in page of PoI details.
Standard flow:	1. This use case starts after user is in PoI details. User clicks review photos. 2. System shows the list of photos from reviews for that specific PoI. 3. User selects specific photo to see it.
Postcondition:	List of photos is shown if there are photos, else 'no photo' message is shown.
Open issues:	Review ordering.

Table 5.11: Use case: View PoI review photos (UC11)

Use case ID:	UC12
Use case name:	New review for specific PoI
Description:	Allows user to write a review for specific PoI.
Precondition:	Internet connection. Camera available. User already is in page of PoI details.
Standard flow:	1. This use case starts after user is in PoI details. User clicks new review. 2. System shows fields that user fills in for this specific review. Fields can be textual or numeric. User can also take a picture. 3. User fills in textual fields: tile of review, positive things, negative things. 4. User fills in numerical fields giving the grade for the following parameters: position, transportation, shops near by, clean and recommendation. 5. If a picture is to be taken, user click camera and takes the photo. If user wants to browse for photo from files, he/she clicks pick a photo and gets photo from files. 6. After user finishes with review, system saves the review.
Postcondition:	Review is saved in local database and appears in pending reviews.
Open issues:	None.

Table 5.12: Use case: New review for specific PoI (UC12)

Use case ID:	UC13
Use case name:	List of my pending reviews
Description:	Allows user to see list of his/her pending reviews. Pending reviews are reviews entered by the user for specific PoI, but not yet submitted to the system.
Precondition:	Internet connection. User has already logged in successfully.
Standard flow:	1. This use case starts after user is logged in. User has to go to the page to see list of pending reviews. 2. System gives to the user the list of his/her pending reviews.
Postcondition:	User either can get the list of pending reviews or if there is no pending gets the message that there is no pending reviews.
Open issues:	Ordering list of PoI.

Table 5.13: Use case: List of PoI reviewed by me (UC13)

Use case ID:	UC14
Use case name:	Submit an already pending review
Description:	Allows user to submit an already pending review.
Precondition:	Internet connection. User has already selected specific review.
Standard flow:	1. This use case starts after user selected pending review. 2. System shows the user pending review with all fields. User can modify anything if he/she wants to and gives the command to submit it. 3. System saves the review in the system.
Postcondition:	Review is submitted if no error occurred.
Open issues:	None.

Table 5.14: Use case: Submit an already pending review. (UC14)

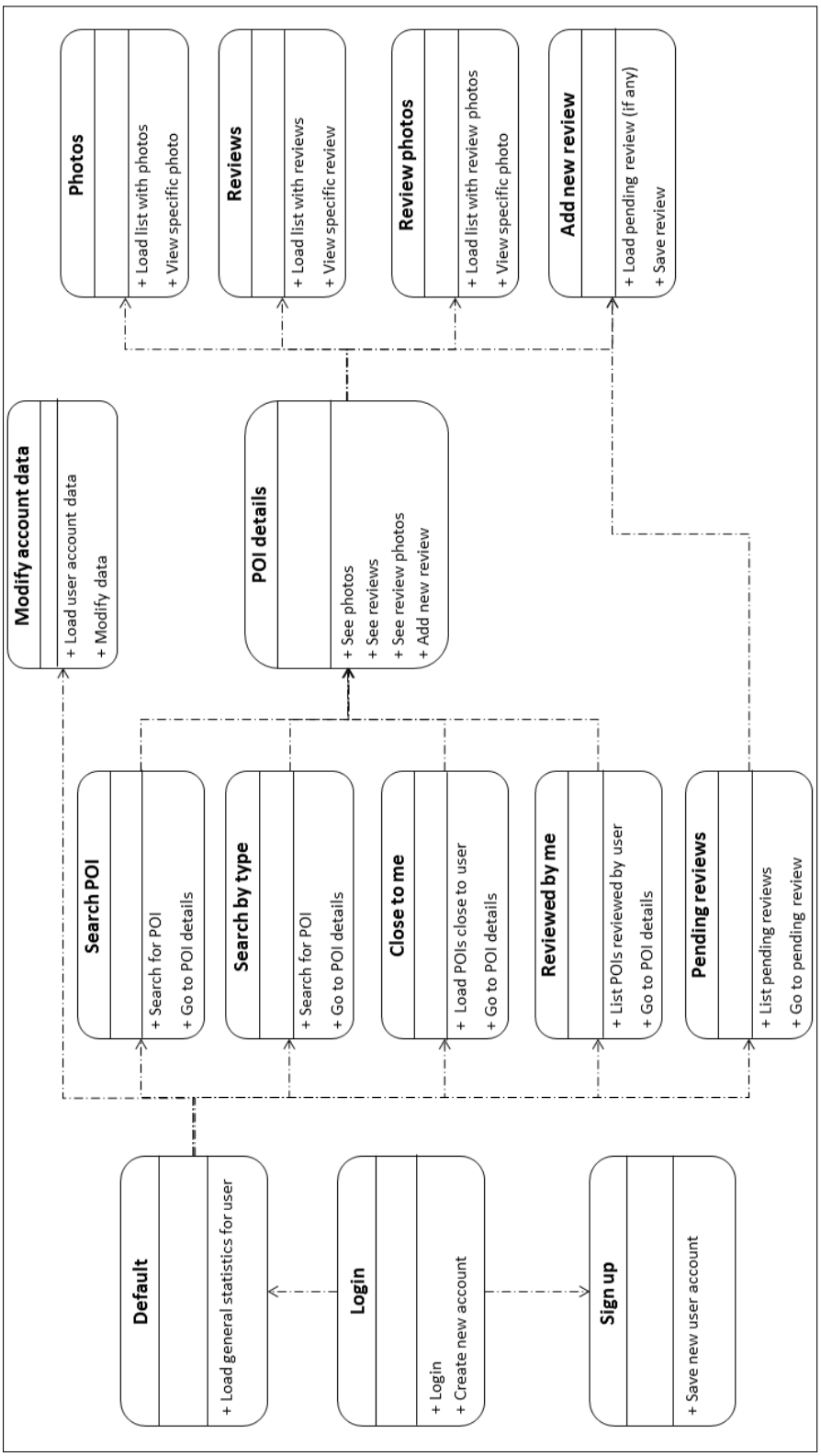

Figure 5.2: Page diagram

5.3 Implementation of WebAPI

Data access - reading and writing in a database always is one of the main challenges during the process of design and implementation of an application. But when an application is mobile, then it becomes even more complicated considering that in this case we are dealing with different operating systems, depending on where the mobile application is installed. To overcome this and access our data regardless of client type, the best solution is webservice. Through web service, app does not access directly database, but it accesses web service and then web service returns to app the required data in a predefined format, recently widely used in JSON.

Given this issue and also taking into account that one of the comparison factors for this thesis is also consuming services, we decided to use webservice for data access in our case-study. Our service (named: PoiPolimi) is of type: REST (REpresentational State Transfer) and it is implemented through Microsoft ASP.NET technology - ASP.NET Web API.

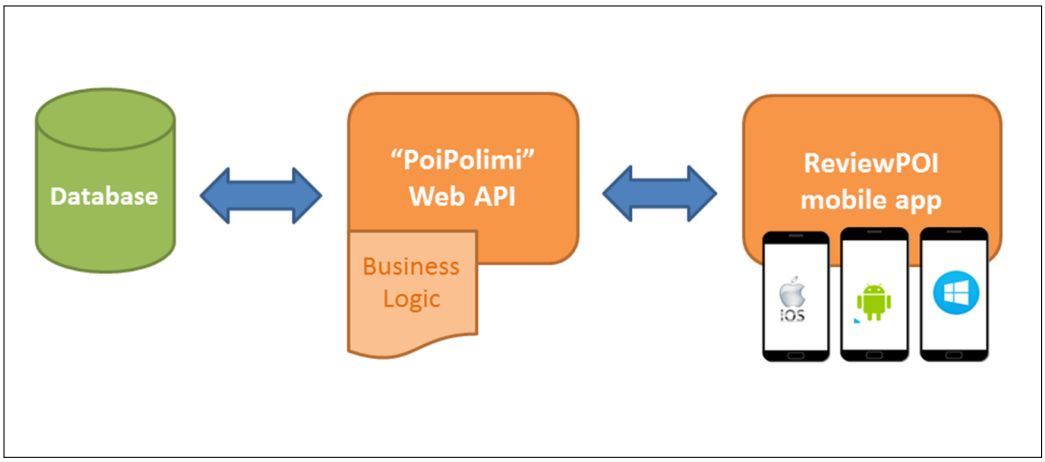

Figure 5.3: WebApi in our case study (PoiPolimi)

PoiPolimi is developed in programming language C# through Visual Studio 2015. Currently it is hosted in personal hosting: politema.ardianisufi.info, a host offered by Godaddy. Data are presented to client in JSON. Below we will give a brief description separately for each of the following webapi main parts: database, model, controllers and calling webapi from the client.

Database - is implemented through Microsoft SQL Server and it is hosted in personal account: ardianisufi.info. All tables in the database are identical with the objects designed in our data model. Database contains in total six table: Poi, Poitype, Poiphoto, Review, Reviewphoto and User. Each of these

tables contains a number of columns, identical with the number of attributes described in our data model.

Model - is used to represent data in webapi. In our case the model represents objects of our database implemented in SQL Server. Besides having six classes by representing our tables in database, for the purpose of not exposing database entities to the clients we created DTOs – Data Transfer Objects. By using DTO we were able to remove circular references, hide properties that we thought user should not see and also use flatten object that contains nested objects. Thus te response given to client is much more understandable and clean. In our model we have 9 DTOs - GeneralStat, Poictm, PoiDetails, PoiGeneral, PoiPhoto, PoitypeGeneral, ReviewGeneral, Reviewphoto and User.

Controller - Controllers main task is to handle all HTTP requests. Through controller we can return either a list of records or only a single record. For querying data we used LINQ, and all queried data are put in DTOs and thus in this format presented later to the client. In our case-study we have implemented seven controllers: HomeController - querying general data shown in default page of application, PoisController - handling getting and storing pois, PoitypesControlles - showing poi types information, PoiphotosController - handling photos of each poi, ReviewsController and ReviewPhotosController handling getting and storing review information and UsersControllers handling users information. Each controller represents a main group of data of mobile application and contains a number of public methods with or without expecting input parameters.

Calling WebApi from a client - by default is: *[host]/api/controller/id*, where *controller* is the name of controller and *id* is id of specific data. Through default pattern we can have maximum two get actions-all records or single record by id. To have more flexible, better and friendly controller design, we modified route definition. In this case we modified route to the following: *[host]/api/controller/action/id*. In this case, right after controller name we added customized action name, so now it is possible to have more than one get action for controller. Moreover now we can have customized public methods. For example, in our PoisController, we have six customized get actions, each of them giving to the client a specific set of data. One example in our webapi, could be searching all POIs close to specific geolocation. In this scenario we have an action called: *searchforpoiclosetome* with two input parameters: longitude and lattidue of specific location. The right way of calling webapi through the client would be: *[host]/Pois/api/0/searchforpoiclosetome/Lattitude/Longitude*.

5.4 Implementation with Xamarin

Writing mobile applications code through C# and same time be able to reuse, share and implement this code natively depending on targeting platform, is making Xamarin as one of the most attractive cross-platform framework in the market. Taking into account our development experience in C# programming language, it did not take too much time to be familiar with Xamarin and programming through it. Since the product is developed in C#, in particular - programming the business logic part of mobile application is same as programming business logic in any other application in Visual Studio IDE.

General description of our case-study implementation experience is divided in following sections, for the purpose of better understanding:

- Development environment,
- Use of programming language and technologies, and
- Project structure (classes, pages and methods).

Development environment

With Microsoft purchase of Xamarin platform, a lot has changed for the better in terms of mobile applications development through Xamarin, especially for software developers who are familiar with or want to learn C# programming language. Now Xamarin is available for free of charge through two ways: as part of the Visual Studio Community IDE (Integrated Development Environment) primarily for the Windows OS users, or as Xamarin Community Studio IDE for Mac users.

In our case-study, since the operating system of the computer where development has taken place is Windows 8.2, we installed Microsoft Visual Studio Community IDE (free of charge) to write and compile code and also run in Android emulator our mobile application based in case-study requirements. All necessary Xamarin libraries are either installed together with IDE or can be downloaded later at any stage.

Programming language and APIs

C# is the programming language we used to create our mobile application in Xamarin – Visual Studio. C# is object-oriented programming language with the aim of building applications primarily through Microsoft Visual Studio.

Because the main logic behind the Xamarin cross-platform framework is to divide the process of mobile applications development in main two groups:

Business Logic in one side - including: business logic classes, database access, network and other common features, and User Interfaces in other side, also the implementation of our application follows this pattern. As for software architectural pattern we used MVVM : Model-View-ViewModel. All UI are designed and implemented in XAML (Extensible Application Markup Language) through Xamarin.Forms. Through Xamarin.Forms we were able to create native User Interfaces with access to all native APIs, but this UIs were created and maintained in only one place - shared code part of the application. In other words, we created native UI only once and not specifically for each targeting platform, but we were able to use them in all targeted platforms.

Because besides the other requirements of our case study were also the ability to take photo by using mobile device camera, get location coordinates from mobile device GPS, save data locally and the ability to consume webservices, to implement these features in our project we had to use also other plugins that are especially implemented for Xamarin for these particular reasons.

To serialize and de-serialize our objects to and from JSON because the response from our webservice is always in JSON, we used Newtonsoft.Json. For photos, particularly to be able to use device camera we used the special plugin: Media, while for location coordinates we used plugin: Geolocator. For local data storage we used SQLite.Net package.

Project structure

Xamarin is known as cross-platform mobile development that gives to developer the opportunity to develop mobile applications where its controls are mapped to native elements of targeted platform. Our project is created as Xamarin.Forms project which means that through this way according to Xamarin, we will be able to share over 96% of code across platforms.

When we created a Xamarin.Forms based project, named: ReviewPOI, four main parts (folders) were created: Portable - where all the shared code is placed and .Droid, .iOS and .WinPhone where any platform specific control or feature is placed. In our case, besides implementing local data storage through SQLite where we had to create an interface to allow us access in specific platform native functionality, all other work is done in shared part of the project.

Using MVVM architecture pattern, forced us to divide the code of the project in three main parts: Model, View and ViewModel. For each part of the model a folder is created and respective classes are placed within that folder. This make the project maintenance easier, because whenever we want to change something we know where the class is located.

Below we describe every part of our shared code in our project:

- **Model** - In Model we have all the classes that represent data that we get or send to our data webservice. More precisely, we have six classes that deal with data from web service - Poi, Poitype, Poiphoto, Review, ReviewPhoto and User and the other only left class: ReviewLocalDB interacting with our local database SQLite.

- **View** - is presentation of our mobile application to the user. In other word here we save the layouts/pages of our application. All the pages are created in XAML. In total we have 15 pages, each of them having a specific duty. The names of the views are self descriptive. *SignUp* is the page through which user create an account; *Login* presents the login page; *MainPage* is the default page after successful login; *MasterPage* is the master page of the project; *SearchPOI* is giving user opportunity to search for POIs; *SearchByType* is giving user opportunity to search for POIs by their type; *CloseToMe* lists POIs that are close to the user in 5,000 m radius; *ReviewedByMe* lists POIs already reviewed by the user; *PendingReview* lists pending reviews; *Contacts* give the user oppurtunity to change some information from his account; *PoiDetails* contains main data for specific POI. From here user can view reviews, photos or add new review; *PhotoPoi* list photos for specific POI. Beside photo there is also a description of review and author of the photo; *Photo* list review photos for specific POI. Beside photo there is also a description of review and author of the photo; *Reviews* list all the reviews of specific POI; *AddReview* gives the opportunity to the user to add new review with or without photo;

- **ViewModel** - Represents classes that serve as a link between Model and View. In ViewModel we bind all the commands and public properties of the View, such as: buttons, lists, input text fields, image sources etc. In our project we have created a VM class for each View. Also here names are self descriptive, i.e *LoginViewModel* is VM for *Login* View, *CloseToMeViewModel* is VM for ClosetoMe View etc.

- **Other general classes and interfaces** - Here we have one interface and one class, both of them exclusively dealing with handling of local database. Interface: *ISQLite* gives access to SQLite and class: *ReviewLocalDatabase* that represent databaseby containing methods for creating new local review, get the list of local reviews and delete local review.

5.5 Implementation with PhoneGap

Implementation of our case study in PhoneGap it was interesting for us and this is because we did not have to much experience before in client side programming. Because the final product developed in PhoneGap is known as hybrid mobile application, then the necessary knowledge for this project development were at least basic knowledge in HTML, CSS and JavaScript. We used all of them in our case-study implementation.

In order to be better understood, we will divide the description of our case-study implementation experience into following sections:

- Development environment,
- Use of programming languages and technologies, and
- Project structure (pages and functions).

Development environment

Because besides installation of the platform, PhoneGap does not offer any of its development code editors, then this is left up to developers. To write a project in PhoneGap a text editor for HTML, CSS and JavaScript it is a must. As much professional is the editor, the easier is the work for developers, especially if the editor offers the possibility of detecting the errors during the programming process. But the editor is not the only necessary step. Indeed, in addition to writing the code it is also necessary to create and test the project through PhoneGap platform.

In our case-study, to write code we used Microsoft Editor known as Visual Studio Code, while the process of installing PhoneGap has been a straight forward and very user friendly, because now, Adobe PhoneGap offers a more practical solution than it used to be before. While, previously for development, we had to install PhoneGap CLI, now we have the possibility to install - PhoneGap Desktop App, a solution that offers a 'user-friendly' GUI to create and maintain the project, off course by giving access to the all desired plugins. After coding with one of the text editors, users can test the product directly to a smart phone (an emulator is not necessary anymore). So, all that we needed for our case-study implementation it was: Installing PhoneGap Desktop on the computer, and installing PhoneGap Developer App in our smart phone. We used smartphone: Samsung A5 with operating system, Android, for testing the behavior of case-study implementation in PhoneGap.

Programming languages and technologies

HTML is the language we used to create pages of our mobile application that is developed based on the requirements of case-study. As a script language to code different functions we used: jQuery (version: 1.11.3.min) and JQuery Mobile (version: 1.4.5.min.js). For design and look of the pages, we used: jQuery mobile CSS (version: 1.4.5.min), a css file already offered by JQuery with the aim for mobile application to look as much native as it can.

Since our application is intended also to interact with the camera and also the geolocation feature (GPS) of the mobile device, then we have used two PhoneGap plugins for this reason: camera-plugin and geolocation-plugin, each of them offering a connection between our mobile application and the specific feature (taking photos and getting coordinates of current location) of the mobile device on which our application is running.

Project structure

Our project in PhoneGap consists of several files, each of these files has one or more tasks to perform, so our application to be functional. Below we are explaining main tasks of the files.

- **index.html** - because our application in PhoneGap is designed and implemented in Singe Page, this is the main page dealing with interaction between user and our mobile application. In this html are all specific pages that we designed for this project. As in other standard HTML document, this file is composed of: head and body. Besides implementing forms and lists, necessary for our project it has also the header with the option to go to the main menu or back button. Main menu is designed independent of the pages within this file and it is called the same menu by every page. Through this file we do also the import of all script files (javascript files), necessary to take care of all functionalities of our application.

- **jquery.mobile-1.4.5.min.css** - this is the deafult css file offered by JQuery Mobile. We did not touch this file and indeed we just used it as it is.

- **jsp file (jquery scripts)** - we have in total three main files that we use: jsOnLoad.js, jsOnClick.js and fillMainMenu.js. In jsOnLoad we have all the functions that return the results when the page is loaded. jsOnClick has all the functions, run on click event of any button. fillMainMenu has a duty to fill the main menu of application. Consume of rest web-service for data handling is implemented in the functions in both script files: jsOnLoad.js and jsOnClick.js through Ajax.

Chapter 6

Comparison results

6.1 Introduction

The whole experience of design and implementation of our case-study in two cross-platform mobile development frameworks and REST webservice is done with the sole aim, so we could give a solution to the problem definition, defined in the beginning of this master thesis.

Because the problem definition is to compare cross-platform development frameworks according to specific factors, we will do a comparison for each factor individually based on our implementation experience in Xamarin and PhoneGap and then based on this result we will try give an answer for each criteria individually where our subject stand and also to give an answer to our problem definition.

6.2 Graphical User Interface

Graphical User Interface (GUI) is the first interaction between the user and mobile application. A good, well designed and looking native mobile app GUI is one of the factors to increase the user satisfaction.

Xamarin

All the pages and controls we created through Xamarin.Forms. In this way we were able to create pages in one place and use them in all three main platforms - Android, iOS and Windows Phone. In terms of page and navigation types, Xamarin support all the pages in our criterion. Indeed we were able to use two of them in our project: master-detail and navigation page. In terms of page layouts, Xamarin supports also the types mentioned in our criterion and here also we were able to use more than one layout. In

Figure 6.1: Same GUI implementation of our main page in Xamarin(A) and PhoneGap(B).

our implementation we used: linear and grid. In terms of control, Xamarin controls are truly native. And finally, in terms of theme, Xamarin uses native theme of operating system.

PhoneGap Taking into account that PhoneGap mobile app is 'hybrid' application its GUI is composed of Web User Interface, meaning HTML and Javascript (mobile JQuery in our case). This means that its UI is not native. In terms of pages types and layouts they are all HTML pages. Even the controls are HTML controls (input, button etc.). Even though lately there are different packages or css that try to make the UI looks native, still the look and feel are not native.

Comparison result Without any doubt, if the aim of application is to look and behave native, for Graphical User Interface the choice is Xamarin platform. And this is because pages through Xamarin.Forms are created and rendered using native controls of targeted platforms. In the user point of view, these pages and their controls are truly native, so users has truly native experience.

6.3 Architecture

Good and well-organized software structure is not only important for the application functionalities but also for the maintain phase. This applies also to mobile application development, and depends heavily on the tools and technologies we use to develop.

Xamarin

By using Xamarin platform, users are able to develop mobile applications by using the programming language C#, that automatically means we are dealing with: Object Oriented Programming. Through Xamarin it is possible to use latest software design and development - architectural patterns such as MVC (Model-View-Controller) and MVVM (Model-View-ViewModel). Using these patterns means that we can have separation of responsibilities. In our case study we used MVVM pattern and were able to divide organize classes and views accordingly. Also we were able to share the code more than 90% of the case, where the only implementation that we had to implement an interface for each cross-paltform was local data storage. In terms of complexity for the developer, the process is very easy and straight forward if there is experience in object oriented programming.

PhoneGap

HTML, CSS and script files (JavaScript) is all that is necessary to know to make a mobile application in PhoneGap. This means that here we do not have any clear separation in terms of application architecture (excluding javascript platforms such as: AngularJS). All functionalities are within javascript files.

Comparison result

For this particular criterion, Xamarin support all the features defined in our comparison methods, while PhoneGap obviously fails to do so. Through Xamarin you use very high grade of code-sharing, latest architecture patterns and share of responsibilities is very well implemented. But again, if we want a simple and lighter weight application then PhoneGap is good option. If however we want more complex application, with good structure then Xamarin platform is the one.

6.4 Services and sensors

Accessing device APIs and consuming webservices is another important feature when designing and implementing a mobile application. A lot of applications require to access at least GPS or camera and a lot of them nowadays consume data through webservice.

Xamarin

There is no problem for Xamarin-based mobile app to access native capabilities such as camera, geolocation, files etc. Also consuming data webservice

Figure 6.2: Same camera access implementation of our case-study in Xamarin(A) and PhoneGap(B).

is very easy process. In order for the developer to access native APIs, special plugins are available so we use them in shared code but we access native capabilities for every targeted platform. For camera and geolocation we used two plugins: Plugin.Media and Plugin.Geolocator. By referencing them in our project, with just few lines of codes we were able to get coordinates from device, take photo and store in our application. Consuming webservice was also very easy process. We had only to reference: Newtonsoft.JSON to serialize and de-serialize the response. Calling webservice is possible through HttpClient.

PhoneGap

By using plugins PhoneGap makes it very easy process accessing geolocation (GPS) and camera of mobile device. We can access this capabilities through the following plugins: plugin-camera to access camera of mobile device and plugin-geolocation to access - get coordinates from GPS of mobile device. For consuming data webservice we use ajax. All this plugins should be used within script file and putting them in action is very easy process by writing just few lines of code.

Comparison result For this particular factor, based on our case-study implementation, both platforms were almost equal and this is because we were able to access camera, geolocation and consume service without any problem. However we say almost equal because even here Xamarin has slightly an advantage especially on consuming data service, because sending, receiving and working with data from webservice is implemented in data access layer. Other than that for camera and GPS there was no big difference.

6.5 Local data storage

Is not always a case that mobile applications read or write data to a database located somewhere on external server or data web-service. In many cases there are different scenarios where users want for a specific purpose either temporarily or permanently preserve some data locally in mobile device. This can occur for many reasons, and one of these may be, for certain processes to avoid the need for Internet connection. In our case-study, we are dealing with the scenario of saving review locally in mobile device storage before submitting it to the server.

Xamarin

Xamarin offers minimum two ways of saving local data storages. For little data we can use Application.Current.Properties through which we can same simple Key-Value data pairs. Other Option is SQLite. This is particularly important if we are dealing with big and structured data. In our case study we used both ways. We stored in Application.Current.Properties few data such as username of the user after logged in. This option was very easy to implement by just declaring it. On the other side, implementing SQLite even though was was easy to implement, it was not straight forward process - we used SQLite to store and retrieve data about pending reviews of the user. First we had to add in references sqlite.net, then to create and interface that will give us access to SQLite. In our model we created the class that represent our data, and also another class that handles main functionalities (add and delete record or show the list of specific records). But as final step before everything worked, was to add a class that will implement the interface that we created before. This class will be created in every targeting platform (Android, iOS), because we need to save the database locally. Then we called methods to insert/delete sqllite data in our shared code normally and without any problem.

PhoneGap

PhoneGap supports different local data storage options. Two of them that are widely used are: LocaStorage and WebSQL. In our case-study we used LocalStorage as our option. It is very easy implementation indeed, because it is enough to declare *localStorage.[name of varible]* and give the value. Variable can be any type: string, int or array. We used array to store pending reviews, and string and integer for user information. The only downsize is the maximum limit which is 5 MB.

Comparison result

Taking into account our implementation experience, for small amount of local data there is no big difference between the cross platform mobile development frameworks, since both of them supports Key/Value pair local data storage. However if we work with bigger amount of data and we want to implement queries and keep data in structured way, Xamarin is better option with SQLite option, especially if we want to query or use transactions.

6.6 Development efforts

Time efforts and cross-platform complexity are very important to developer or development team, while performance beside for the developer is also very important to the user of mobile application.

Xamarin

For a software developer with solid experience in object oriented programming, programming through Xamarin with C# is not a big issue. Programming in Xamarin does not take more time than programming any other application in development environment with object oriented programming (C#) - even access to APIs is implemented very easily through plugins. Because Xamarin is offering its Integrated Development Environment (IDE) through Visual Studio or Xamarin Studio, working on it and with it we found very easy and user-friendly. In terms of documentation, a lot of documentation of the platform and with really good implementation examples is available. All this documentation is free of charge and can be downloaded any time through Xamarin webpage. Because the product is native, performance is very good. In terms of testing, Xamarin is offering the possibility to test the product through emulator during development, but also lately has implemented: Xamarin test cloud, a place where you put the project and it will be tested in around 2,000 real devices. This feature is free only for one device hour per day, while other options are with payment.

PhoneGap

PhoneGap mobile application is hybrid application, and even though is just HTML, CSS and Javascript to design and develop it you need these skills. So if these skills are in place then it is not a problem to do a mobile application. Indeed in our case-study from the moment we finalized the website (application without api) it took less than a week to make it ready to use in Android phone. In terms of IDE, PhoneGap does not offer any IDE but developers can use any HTML/Script supported editors. But the lack of documentation is still a issue for PhoneGap even though every day

the documentation is improving. PhoneGap based application in our case, especially when we load data is taking a bit time to retrieve them. In terms of testing, actually it is very user friendly because the product is tested in real device. In our case we were able to test in our Samsung A5 phone. All you need is PhoneGap Desktop and PhoneGap Developer installed in the mobile device. It is worth mentioning that also Adobe PhoneGap offers their cloud testing, but for free it is limited to only one private project.

Comparison result

In terms of Integrated Development Environment (IDE) complexity and easiness of use with no doubt Xamarin is better one, because is user-friendly and compact IDE with all what you need. The same thing we can say also for documentation, because Xamarin documentation is well kept and has a lot of real examples. In terms of time PhoneGap it might take less time to design and develop if developer has basic knowledge in client programming but if the developer has knowledge of object oriented programming then this does not stand - easier with PhoneGap. From visual point of view, performance is better in Xamarin.

6.7 Comparison conclusion

Implementation of our example in our two compared cross-platforms mobile development framework has been a very positive experience but also not forgetting the fact that had its own challenges. After the implementation and analyzing our factors individually defined at the beginning of this master thesis in problem definition, we can conclude that even though according to our comparison method criteria the clear winner is Xamarin, because it supported more features from our comparison criteria, still a priori we cannot give an simple answer to the problem definition discussed at the beginning of this master thesis. To decide which platform is more appropriate for developers of mobile applications, we think that it depends on several factors such as: the size of the application, the type of user interface, the targeted performance, complexity and application structure and also the skills of the development team. If the aim is to create a well-structured application, with a native UI and a foreseen easy maintenance then definitely solution will be Xamarin, on the other hand if the aim is to create a application of smaller size (light application) where is not so critical the performance and native UI then we still can go with PhoneGap.

Chapter 7

Conclusion

Introduction of mobile devices, has increased rapidly the number of mobile users. Such an increase of mobile users is making them important user group in terms of user segmentation. In terms of development, coding mobile applications for mobile devices with operating systems requests well trained staff on specific platform - framework. To overcome this problem, cross-platforms mobile development frameworks were born. Cross-platform mobile development refers to the development of mobile applications, ready to be installed and run in mobile devices with different operating systems.

Throughout this thesis we were focused on comparing cross-platform mobile development approaches, more precisely two cross platforms: Xamarin and PhoneGap, each of them representing different mobile development approach. They were chosen specifically because of different platform types, more precisely the type of product that is developed through them. The product - mobile application of PhoneGap is hybrid mobile application, while Xamarin product is cross-platform native application.

After initial discussions and literature review, we decided that the best way to do a proper comparison, besides relying on relevant work was to design and implement a case-study in both platforms. And indeed, we implemented a case study, a review mobile application - called: ReviewPOI. ReviewPOI is a mobile application that allows users to see what other users thought when they visited a specific touristic point of interest. More precisely it is dealing with reviews of any touristic point of interest. Besides viewing what other users said, users themselves can also give a grade and textual opinion about the touristic points.

Implementation of our example in two compared cross-platforms mobile development frameworks has been a very positive experience but same time also it was a challenge. Besides sending and receiving data, application is accessing also camera, photo albums, geolocation and local data storage of

mobile device. Based on this gained experience, besides writing briefly about implementation in each technology, we tried to do a comparison according to the following factors: Graphical User Interface, architecture, service and sensors, local data storage and development efforts.

After implementing and analyzing our comparison factors individually, we came to conclusion that a priori we cannot give an simple answer which framework is better to be used to implement a mobile application. To decide which platform is more appropriate for developers of mobile applications, we think that it depends on several factors such as: the size of the application, the type of user interface, the targeted performance, complexity and application structure and also the skills of the development team. If the aim is to create a well-structured application, with a native UI and a foreseen easy maintenance then definitely solution will be Xamarin, on the other hand if the aim is to create a application of smaller size (light application) where is not so critical the performance and native UI then we can go with PhoneGap.

Bibliography

[1] *Global market share held by the leading smartphone operating systems in sales to end users from 3rd quarter 2009 to 3rd quarter 2015.* statista.com, 2016.

[2] P. S. Ptitsyn and D. V. Radko, *Analysis of cross-platform technologies for mobile applications development.* ARPN Journal of Engineering and Applied Sciences, 2016.

[3] V. Tunali and S. Z. Erdogan, *Comparison of popular cross-platform mobile application development tools.* ResearchGate, 2015.

[4] R. Allen, *Mobile Internet Trends Mary Meeker 2015.* smartinsights.com, 2015.

[5] *Number of mobile app downloads worldwide from 2009 to 2017.* statista.com, 2017.

[6] *Statistics and facts about Mobile App Usage.* statista.com, 2017.

[7] *Software Development Discussion Paper.* Blueberry Consultants.

[8] J. Zhang, J. Wilkiewicz, and A. Nahapetian, *Mobile Computing, Applications, and Services.* Springer, 2011.

[9] B. Unhelkar, *Handbook of Research in Mobile Business: Technical, Methodological, and Social Perspectives.* IGI Global, 2009.

[10] D. Steinbock, *The Mobile Revolution: The Making of Mobile Services WorldWide.* Kogan Page Limited., 2005.

[11] L. Darcey and S. Conder, *The Android developer's collection.* Addison-Wesley, 2012.

[12] S. Mann and S. Sbihli, *The Wireless Application Protocol (WAP): A Wiley Tech Brief.* John Wiley and Sons, Ltd., 2000.

[13] J. Annuzi, L. Darcey, and S. Conder, *Introduction to Android Application Development*. Pearson Education, Inc., 2014.

[14] M. Jipping, *Smartphone Operating System Concepts with Symbian OS: A Tutorial Guide*. Symbian Software Ltd., 2007.

[15] S. Olson, J. Hunter, B. Horgen, and K. Goers, *Professional Cross-Platform Mobile Development in C#*. John Wiley and Sons, Ltd., 2012.

[16] E. of the American Heritage Dictionaries, *High Definition - An A to Z Guide to Personal Technology*. Houghton Mifflin Company, 2001.

[17] S. Boardman, M. Caffrey, and S. Morse, *Oracle Web Application Programming for PL/SQL Developers*. Prentice Hall, 2003.

[18] N. Gok and N. Khanna, *Building Hybrid Android Apps*. O'Reilly Media, Inc, 2013.

[19] "Introduction to xamarin." https://developer.xamarin.com/guides/cross-platform/getting_started/introduction_to_mobile_development/. Last access: 2016.

[20] "Xamarin system requirements." http://developer.xamarin.com/guides/cross-platform/getting_started/requirements/. Last access: 2016.

[21] "Understanding the xamarin mobile platform." https://developer.xamarin.com/guides/cross-platform/application_fundamentals/building_cross_platform_applications/part_1_-_understanding_the_xamarin_mobile_platform/. Last access: 2016.

[22] J. M. Wargo, *PhoneGap Essentials: Building Cross-platform Mobile Apps*. Pearson Education Inc., 2012.

[23] V. Gerasimov, S. Bilovol, and K. Ivanova, *Comparative analysis between Xamarin and PhoneGap for .Net*. System technologies, 2015.

[24] M. Prajapati, D. Phadake, and A. Poddar, *Study on Xamarin cross-platform framework*. International Journal of Technical Research and Applications, 2016.

[25] G. Vitols, I. Smits, and A. Zacepins, *Issues of Hybrid Mobile Application Development with PhoneGap: a Case Study of Insurance Mobile Application*. International Baltic Conference, Baltic DB and IS 2014, 2014.

Acknowledgments

I would like to start the acknowledgments by saying a big thank you to all those people who made this thesis possible and an unforgettable experience for me.

A special acknowledgment for Italy and Politecnico di Milano for giving me the opportunity to study abroad. To meet and socialize with different people, to learn different cultures and above all to study in such a well-ranked world class university.

I would like to express my sincere gratitude to my advisor Prof. Marco Brambilla for his continuous support during this research, for his patience, motivation, and knowledge. His guidance helped me a lot in writing of this thesis. I could not have imagined having a better advisor for my master thesis.

A special acknowledgment also for my fellow colleagues for the endless professional discussions. I am very grateful also to my current employer: European Union Rule of Law Mission in Kosovo for encouragement and practical advice.

Last but not least, I would like to thank my family: my wife Ardita and my children Fron and Arb for their patience and endless support during my studies. My parents and sisters, for their on-going support and big influence in my personality. My uncle Florim, for teaching me that failure is never an option.