Cross-Platform Mobile Development using Xamarin and the Evaluation of the Application

Georgios Karagiannis

H00226912

August 2016

Computer Science

School of Mathematical and Computer Sciences

Dissertation submitted as part of the requirements
for the award of the degree of
MSc in Software Engineering

STATEMENT OF NON-PLAGIARISM

I, Georgios Karagiannis, declare that this thesis titled *Cross-Platform Mobile Development using Xamarin and the Evaluation of the Application* and the work presented to it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g. ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: Georgios Karagiannis

Date: August 18, 2016

ACKNOWLEDGEMENTS

I wish to thank my family and all my close people for their encouragement throughout the study and also for the invaluable support they gave me in order to make a new start in my life. Also, I wish to express my appreciation to my supervisor, Dr. Lilia Georgieva, for her invaluable assistance and support throughout this thesis.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF ACRONYMS

OS	Operating System
UX	User Experience
IDE	Integrated Development Environment
API	Application Program Interface
UI	User Interface
GUI	Graphical User Interfaces
SDK	Software Development Kit

ABSTRACT

This thesis is concerned with Cross-Platform applications development for mobile devices. Cross-Platform development is the process of implementing applications for multiple OS. It allows, using same code, the building of an application for multiple platforms (Charkaoui, et al., 2014).

It is understood that the diverging and the rapid growth of mobile industry (GSMA, 2016), created the need for development of applications that can be installed and work in all types of mobile devices and all types of mobiles OS. Furthermore, these applications must provide the same UX among all types of mobile OS. This thesis examines the development of a Cross-Platform Native mobile application using Xamarin and the evaluation of the UX the application provides in Android and iOS mobile devices. The Xamarin is a platform which enables the developers to build applications, using same C# code, for iOS and Android (Xamarin, 2016).

Keywords (in alphabetical order): Android, Cross-Platform Mobile Applications, iOS, Xamarin,

CHAPTER 1
INTRODUCTION

Mobile devices are becoming extremely popular in our society and a variety of mobile platforms have come (Ohrt & Turau, 2012). Consumers are using mobile devices for communication, web access, e-commerce, gaming and social networking. According to the International Data Corporation (Figure 1- Smartphone OS Market Share, 2015 Q2), Android and Apple iOS are the two major OS in the mobile market.

Mobile devices are characterized by great computing power. They have pre-installed applications, for example the application which handles the phone calls, but also users are able to install additional applications, for example the WhatsApp application (WhatsApp Inc, 2016). The distribution of these applications is taking place via the application stores each OS supports, for example the Google Play for Android (Xanthopoulos & Xinogalos, 2013).

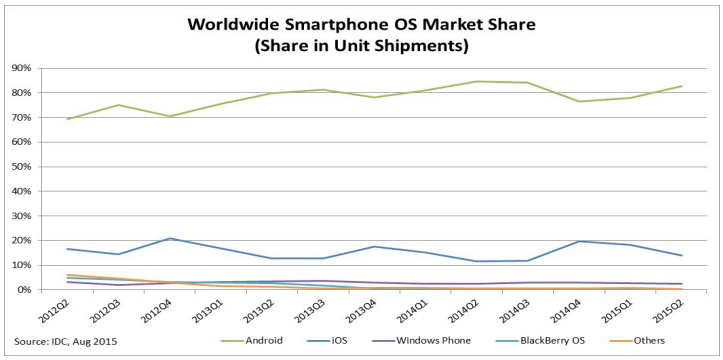


Figure 1- Smartphone OS Market Share, 2015 Q2

Each one of these platforms uses different programming language, different IDE and different APIs. Each one of these platforms may only support applications which are implemented with this technologies. For the development of applications by platform, the developers need to become familiar with the different programming languages, the different IDEs and the different APIs each OS uses. Because of that, the cost for building and maintaining applications by mobile OS is high (Lin & Lee, 2015).

Observing the diversity in the mobile OS, the need for a mobile application development process which will bypass these differences and will help the developers is necessary. Cross-platform development is the process of implementing applications for multiple OS. It allows, using same code, the building of an application for multiple platforms (Charkaoui, et al., 2014). This thesis examines the development of a Cross-Platform Native mobile application using the Xamarin and investigates the UX this application provides in each of the two major mobile OS, in Android and in iOS.

1.1. Motivation

Motivation for this thesis gave me a personal experience. The changing to a new smartphone with different mobile OS, enforced me to change the applications I used, my habits and also made me less productive than I was. Finally, I was enforced to go back to previous mobile OS because the applications I used, was not available in other platforms or if there was available, the UX was not the same. For example, the mobile application of the National Bank of Greece which I used in my everyday life, does not provide the same functionality among the Android and the iOS. Also the UI and the UX is not the same among the previous mobile OS (National Bank of Greece, 2016).

The users want to use the same applications in every mobile device they have, regardless of the mobile OS. The applications should provide the same functionality and the same UX among the two major mobile OS, the Android and the iOS. Furthermore, the developers needs to reduce the cost for building applications and the same time needs to be able to offer their applications to as many users as is possible.

For these reasons, this thesis examines the mobile application development regardless of the mobile OS and investigates if the UX of a Cross-Platform mobile application, which has developed using Xamarin, is the same in Android and in iOS mobile devices.

1.2. Aims and Objectives

The main objectives of this thesis is the development of a fully working mobile application which is able to be installed and run in Android and in iOS mobile devices and the evaluation of the UX, the feeling and the performance the application offers to users between the previous mobile OS.

The current solutions and the best practices for mobile Cross-Platform development will be analysed, a prototype mobile application will be developed using Xamarin and potential users will evaluate this application.

The mobile market is fragmented but the developers need to have as much market share as they can. In order to be achieved that, their applications should be available for Android, and iOS mobile devices. It is hard for the developers to build application by platform because the cost for building and maintaining applications is high (Lin & Lee, 2015). Furthermore the design of the application by platform is not always the same.

Each company which is behind of each mobile OS wants to attract as many developers as it can in order to make their own mobile OS the most popular (Dalmasso, et al., 2013). Developers are the engine of each mobile OS, they produce applications, they produce innovation and because of that, they brink users to the platforms and profits to companies.

Cross-Platform development helps developers to reduce the development time and the building cost of a mobile application (Boushehrinejadmoradi, et al., 2015). Also it helps the mobile OS in terms of gaining more developers and therefore more customers. Cross-

Platform development helps developers in terms on spending time and sources and platforms in terms of gaining developers. It is easy for the developers to release the same application in every platform.

The main goal of the mobile Cross-Platform development is the development of an application which offers the performance that a Native application offers and in the same time is able to run in more than one mobile OS (Xanthopoulos & Xinogalos, 2013). However, developing a mobile application which can be executed in more than one mobile OS may imply several issues to users in terms of the provided UX.

1.3. Report Outline

The first chapter is the introduction and provides an outline of the problem being addressed and the main goal of the thesis. The second chapter will illustrate previous work that have been done on the research area and the associated objectives that will help for the accomplishing of the thesis, It will provide an outline of the mobile platforms, the types of the mobile applications and the Xamarin which will be used for the development phase of this thesis. The third chapter will demonstrate the different methodologies that have been used in this thesis, the implementation and the evaluation methodology.

The fourth chapter refers to the requirements of the project and the requirements of the application will be built. The fifth chapter will show how the application designed and the technologies were used for the development phase. The implementation of the application is following in the sixth chapter. The evaluation and the analysis of the result are in the chapter seven. The chapter eight, which is the final, includes the conclusion of this thesis.

CHAPTER 2
LITERATURE REVIEW

2.1. Mobile Platforms

The smartphones are now an integral part of our society and our everyday life. Millions of people around the world enjoy increased capabilities the smartphones offer. The smartphones provide internet access, personal information management and entertainment applications like games. Ten years ago, these features were available only in personal computers. (Charkaoui, et al., 2014)

The term *smartphone* first appeared in 1997 when Ericsson described the GS88 as smartphone (McNish , 2015), although devices that combine telephony and information technology were conceived in 1973, where Theodore Paraskevakos patented concepts of intelligence combination of data processing and display screens phones (Want, 2014).

After the appearance of desktop computers, various forms of portable computers, notebooks and laptops, began to appear. The last 5 years, in the position of netbooks, appeared a new category of mobile-type device, called tablet (Prey & Weaver, 2014). Tablets are an intermediate state of smartphones and computers, as most of them use the same OS as smartphones and also because of their larger screen is utilitarian and as notebooks.

There are two major mobile OS that used by smartphones and tablets: the Android and the iOS. The OS is responsible to compline the hardware of the smartphones in each device with the software. The mobile OS can be viewed as a stack of layers. This stack of each OS will be analysed. Following a description of each OS highlighting similarities and differences. The understanding of the similarities and the differences of the platforms is important because developers can use common features among platforms at the development process.

2.1.1. Android

In 2005 Google bought a company with the name Android. This company focused in developing software for mobile devices and was developing an OS based on Linux Kernel (Anderson & Hall, 2009). According to (IDC, 2016), Android is the most popular and the faster growing technology platform. Since its launch, the amount of its users increased about 1.5 million per day (Pon, et al., 2014). The source code of the Android platform is released by Google under open source licenses. (Google, 2016). The figure 2 is showing the home screen from a mobile device with the latest version, the Android 6 "Marshmallow".

Apart from Android for mobile phones, Google has built and released the Android TV, for televisions, the Android Auto, for cars, and most recently, the Android Wear for wrist watches.

Android is created on Java platform. The open source nature of the platform, the widespread use of Java developers and familiarity with the Java language make the Android platform the most widely used (IDC, 2016). The development of an Android application is most likely done using the free Google's IDE called Android Studio. Furthermore, documentation for the platform and an online community is available. Android applications can be distributed through the Google's Play Store (Google, 2016).

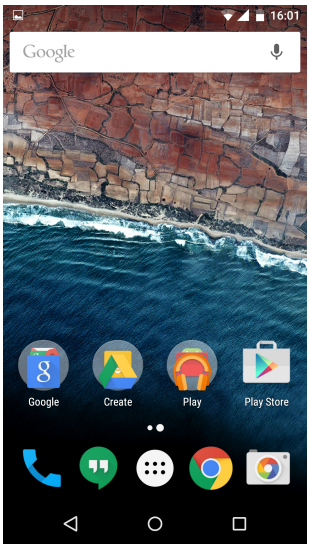


Figure 2 - Home Screen Android 6 (Google, 2016)

2.1.1.1. The Android Software Layers Stack

Android OS has as base the Linux Kernel. The Linux kernel interacts with hardware and contains all essential hardware drivers. The next layer is a set of libraries including SSL for security, SQLite for storing data and WebKit which is the web browser engine. For the execution of Java programs, Android uses a Virtual Machine called Android Runtime, before the version of 4.4 Android used the Dalvik Virtual Machine (Deng, 2015).

In Application framework Layer are located high-level services. The Activity Manager which manages the applications lifecycle, the Resource Manager which handle the access to various resources and the Location Manager which handles the location. At the top of stack there is the Application layer in which all the application will be installed. The figure 3 is showing the software layer stack.

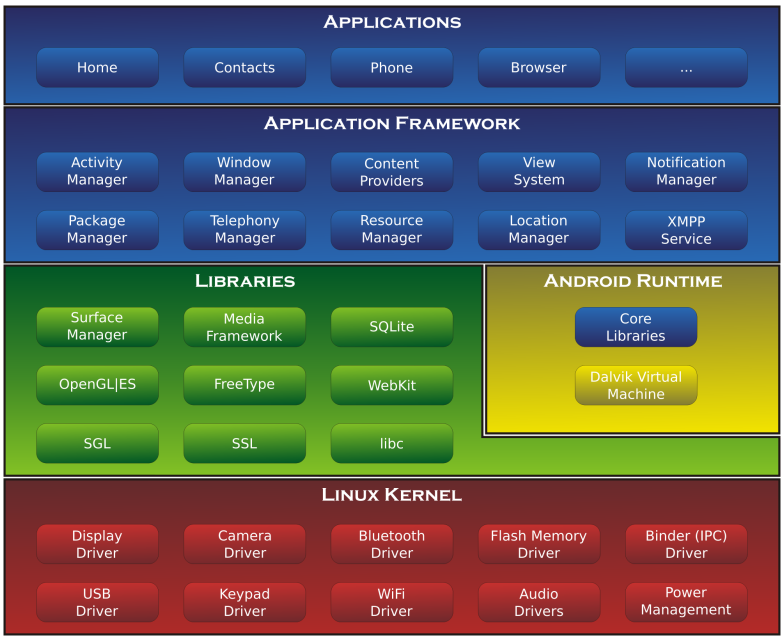


Figure 3 - Android Software layers Stack (Google, 2016)

2.1.2. Apple iOS

The second most popular mobile OS, after Android, is the Apple iOS (IDC, 2016). Released by Apple with the first iPhone in 2007, it is a closed-source platform and runs specifically on Apple's mobile devices like iPhone and iPad. The iOS applications can be distributed through the Apple's App Store (Apple, 2016).

For the development of native iOS applications two programming languages are used, Swift and Objective-C, both of them are Apple's in-house designed programming languages but Swift released as open-source. The latest version is the iOS 9 and the figure 4 is showing the home screen of this version. The development of an iOS application is done using Xcode IDE installed on a Mac running Apples OS X (Apple, 2016).

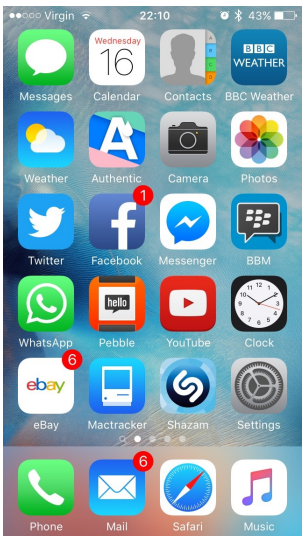


Figure 4 - iOS 9 Home Screen (Apple, 2016)

2.1.2.1. iOS Software Layers Stack

At the button of the layer stack is the Core OS layer. This layer contains the features that the majority of technologies are built upon. This layer is also used for security and communication with external hardware situations. The Core Services layer contains the basic services required by upper layers, like SQLite. The Media Services layer Provides frameworks for graphical and audio technologies. Finally, the Cocoa layer contains the frameworks that are used for building applications (Smyth, 2011). One example is these frameworks are the Message UI Framework which is needed for emails and SMS communications. The figure 5 is showing the iOS software layers stack.

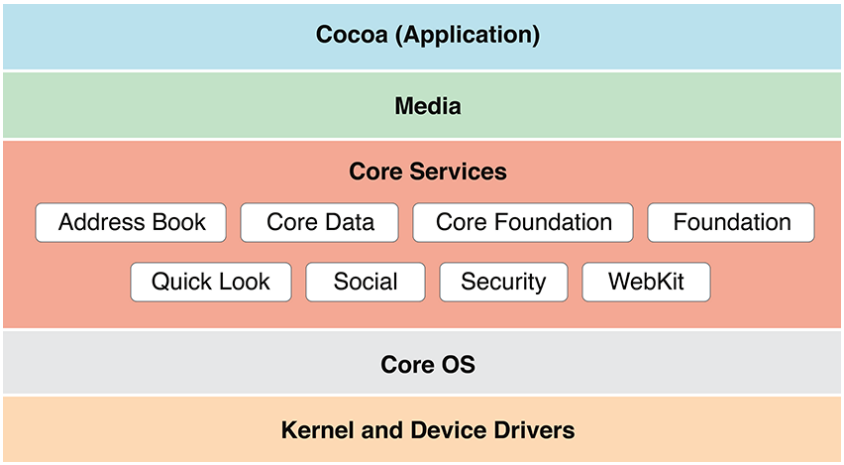


Figure 5 - iOS Software layers Stack (Apple, 2016)

2.2. Mobile Applications Development

There are two main development approaches for mobile applications. The Native development and the Cross-Platform development. The Native development involves developing applications only for a specific platform. The Cross-Platform is the application development approach for more than one platforms and can be separated in three main categories. The Web approach, the Hybrid approach and finally the Cross-Platform Native approach (Delia, et al., 2015). The figure 6 is showing all the approaches for mobile applications development.

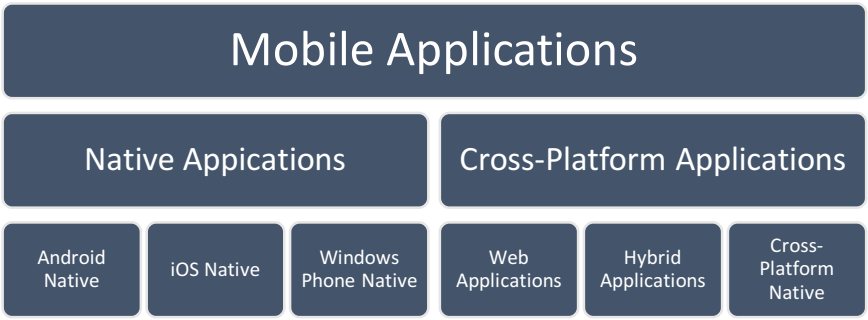


Figure 6 - Mobile Application Categories

The Native applications executed totally on a mobile device and engaged directly and efficiently with the device hardware. The Web applications executed on a Web browsers and most of these applications run on a remote server. The Hybrid applications are like the Web applications but there are not running remotely and finally the Cross-Platform Native applications are compiled natively by creating a specific version of the application for each platform (Charkaoui, et al., 2014).

The main advantages of all the Cross-Platform development approaches compare to Native is the less time that the developers need to spent at the development of the application and specifically in Web and Hybrid approach, the technologies that are be used (HTML5, JavaScript and CSS) may are already known to developers.

It can be detected that the Native Applications are better in performance and can provide access to all features of the device in which they operate, but the cost of implementation and maintenance is high. The Cross-Platform Native applications offer the same features as the Native applications and also, with less cost and less time, the performance and the UI is almost the same (Xanthopoulos & Xinogalos, 2013).

The Web Applications are simpler in implementation, the cost of implementation and maintenance is considerably lower, but have reduced access to the functions of the devices and provide lower performance and UX. The hybrid approach offers a combination of Web and Native applications. In several cases compromises in the hybrid approach is required (Delia, et al., 2015).

A detailed analysis and a comparison of the mobile development approaches is following.

2.2.1. Native Applications

A native application is the one that was designed on the targeted OS and it take advantage of all features of the mobile phone (Charkaoui, et al., 2014).

The native applications consist of executable files that are downloaded to the device. The installation process can be done by the user or the application. The most common way to install a native application is through the online application store each platform provides. After installation, the user can launch the application like the other applications which are pre-installed on the mobile device. Usually, an icon is created on the device's desktop. The user selects this icon each time he wants to use the application.

At the first run, the application will connect directly with the mobile OS, without the need for another, intermediate, software layer. In this way, the user may have access to the functions of the applications supplied with the mobile OS, gaining the entire control of the device. An example which makes a simpler understanding of the above procedure is the camera of device, which is controlled by many applications such as the application of Facebook and Instagram.

For the implementation of a Native application, the developer need to write the source code in the programming language that supports the mobile OS and include whatever else is necessary for the operation of the application, such as images and audio files. Using tools that provided by the company that created the OS, these files are compiled and produced a

file which is the application that will be stored on the device. These tools are the application development environment (SDK) for the mobile OS. The process of developing a Native application is similar between different OSs, but the development environments (SDK's) are different. They have been created to serve the needs of a specific mobile OS and provide different tools (IBM, 2012).

The application, once installed on the mobile device and executed by the user, interacts with the mobile OS of the device making use of the available interfaces. The programming interfaces can be divided into two main categories:

- Low level programming interfaces (low-level APIs).
- High-level programming interfaces (high-level APIs).

The Low-level programming interfaces provides interaction with the touchscreen, the keyboard, the connection to a network, the audio files received through the microphone. These are some of the examples of the possibilities offered by these tools. The high-level programming interfaces offer a range of services. Some of these services are surfing the web, access to calendar, to contacts, to photo gallery and of course the possibility of telephone calls, sending and receiving text messages using the application. Most OSs, like Android and iOS, have pre-installed applications that implement these services, but a set of access methods to these services programmatically (through native application) is available, thus presented applications that combine several of these functions (Chen , et al., 2010).

Another set of tools available to native applications is the graphical interface. The OS provides a set of basic widgets such as buttons, menus, tab bars, alarms and more. Applications that use these widgets, inherit the appearance of the particular OS on which it is installed, so the experience received by the user is within the scope accustomed.

It is important to note that each OS comprises its own special graphics tools, For instance, Android uses XML. Even between different versions of the same operating disparities (Zhu, et al., 2015). The graphical interface is very important part for the success of the OS with as result, there are continued improvements and thus variations (Ghayas, 2013). The difference observed between these tools make it necessary to familiarize the designer of the application, with the tools and components provided.

The programming interfaces required for full use of the mobile device and therefore the OS. Are associated with the OS for which they were created. This adds complexity and cost to the development of a native application with presence in various OSs. Nevertheless, their role is particularly important, as are those that enable the development of highly complex applications.

2.2.1.1. Native Android Application Development

Google provide official guides for Android OS developing. Apart from that, documentation for every framework that Android supports is accessible on-line for every developer. Companies, like LG and Samsung, releases new, more powerful (Antutu, 2016) smartphones very often. Developers need to take advantage of this power that new smartphones provide. New powerful apps and new games with graphics very close to gaming consoles is the target.

The large number of companies that releases smartphones with Android OS, the different characteristics these smartphones have and the wide range of Android versions are major problems for the developers (Joorabchi, et al., 2013). The compatibility with previous versions of Android OS and the same time the need to take advantage of the new features the newest version provide is difficult and drives to more complex code. It is feasible for the developers to use all these devices for testing of the application, virtual machines helps for that in most cases but is still possible the appearance of specific issues in some real devices.

For the developing of a Native Application for Android OS, the Software Development Kit (SDK) is needed. Google provides that for free (Google, 2016). The SDK also includes the latest API version but using the SDK manager the developers can install previous API versions.

The Android Package (APK) contains all the content and the data the application needs. The SDK compile the code the developers write into an APK file which is used for the installation of the application. Apart from that, the APK file contains certificates, raw files and also the manifest file. The manifest is a XML type file and in that are stored important information of the application, the name of the app, the version, the resolution of the screens that the application supports and also the minimum API level that the application needs. One other characteristic is the digital sign that the developer can have in the application (Burnette, 2015).

The installation, the test and the debugging of the application can be done using virtual machines but also in physical devices using the Android Debug Bridge (adb). The adb is an utility which lets the developer to install an application in a real smartphone at the development phase. The virtual machines, or emulators, are handled by the Android Virtual Device Manager which is part of the SDK. The developer can define the settings and the sources of the virtual machines, can choose screen size, resolution, available RAM, if the device has GPS, Accelerometer, Cameras. Furthermore can choose API level, CPU architecture and internal storage.

Android apps are written in Java language, for the execution of Java programs Android OS uses a Virtual Machine called Android Runtime (ART). The ART compiles the application into native code at the installation and not at the runtime. With this method the applications supposed to run faster with less needs for sources (CPU, memory, battery). Furthermore, the Java Development Kit (JDK) must be installed, it easy open source and free to download.

For the distribution of the application, the developer has to register to Google Play (Google, 2016). That is the online application store in which the developer has to upload the application in order the users to have access in this. The fees for the registration is $25 and Google also provides version control and multiple statistics about the distribution of the application.

2.2.1.2. Native iOS Application Development

The iOS platform is less fragmented than Android and devices running older versions are limited (Apple Inc, 2016). For the development of an iOS application a Mac computer

and the IDE Xcode is required. The OS of the Mac and the Xcode should be up to date with the most recent version. Furthermore, as in the Android OS, the latest iOS SDK is needed. The Xcode provides an iOS virtual machine. The developer can test the application in this virtual machine in different versions of iPhones and iPads with different screen sizes. Apple's iOS uses a file similar to Android's Manifest, which is the Info.plist and is used for configuration. It is used to set the version, the name, background images, icons and other information (Neuburg , 2013).

For the realizing of an application on the Apple's app store, there is a review process that the app must pass and usually needs some days for this process. According to the conditions, an application must do something useful, unique or provide lasting entertainment and must be well designed according to Apple's guidelines. An iPhone must be connected to actually build a release version of the application. The Xcode must accept this device as a target and it must first be registered in the Apple Developer member centre. For the enrolment in the iOS developer program, in order the developer to be able to realize applications in to Apple store, the cost is $99 per year. (Apple, 2016)

2.2.2. Web Applications

In the description of native applications was felt the fragmentation size that exists in the field of developing applications for mobile devices. Solution to this problem, attempt to provide the Web applications. Users can access these applications using a web browser. These are built with programming languages like JavaScript and HTML. Usually cannot take full advantage of the mobile phone and the most of the times they need internet connection (Serrano, et al., 2013). This category is not new as an idea, as is known from the applications already exist for desktop computers for several years now. One example is the e-mail application, such as Google's Gmail, using the browser on their representation and implemented with web technologies.

The web applications promise for the solution of the problem of fragmentation is detected in mobile OSs, providing the ability to develop applications that run on different OSs and devices using only web technologies like HTML, CSS and JavaScript. In this way, developers avoid the huge obstacle of learning new programming languages for each different OS that creates an application. (Charkaoui, et al., 2014)

Finally, these applications will be hosted on the server and can be executed by the user visiting the corresponding web address (URL) through the browser, avoiding in this way the various online stores providers of OSs and different requirements that must be met for an application to be made available through these.

The fact that the field of web applications have already several development tools does not mean that they can be used in the field of mobile devices without further development. Mobile devices, as described above, have specific characteristics that render the existing inefficient technologies. The actual start of web applications was the arrival of HTML5 and the continuous evolution of the Browsers who can now take advantage of these new capabilities.

At this point, it should be noticed that unlike the native applications that are as environment executing the OS and therefore directly related to this, the web applications running on the web browser. The web browser is the one with the turn one native application, so it has direct access to the OS APIs, but only a few of them are available to applications running on it. While native applications have full access to the device where performed, web applications have limited access to many features of the device and in some cases access completely absent. The lack of access due to two main reasons. The first is the security of the device. The native applications are available through various marketplaces that before placing an application to users, considering the functions and examine whether it has features that could be harmful to the device or for sensitive user data. In web applications that control is lacking as it is available through webserver. The second reason is the lack of implementation of various programming interfaces, resulting in lack of access to various functions. Web applications depend on the progress of HTML5 and especially JavaScript APIs it supports. Many of them have not yet implemented or not fully supported by browsers (Serrano, et al., 2013).

Particular attention should be paid to the performance of web applications. As it mentioned earlier, the online approach uses a web browser as a runtime environment. This adversely affects the performance since the performance of the application depends not only on the processing power of the device in which it is performed, but also by the browser.

2.2.3. Hybrid Applications

Hybrid applications run locally using web technologies and taking advantage of possibilities of the phone. As having these technologies, there is no need to develop the application each time, because is compatible on all OSs (Serrano, et al., 2013) (Charkaoui, et al., 2014).

Following this approach, developers have the ability to develop the main functions of the application using web technologies while preserving access to the device features via native APIs that work as bridge between the core functions of the application and the characteristics of the device. Thus achieve reuse part of the application that was implemented in web technologies, while you only need to be modified to operate the application and on other mobile OS is the changing of the native APIs used for the functions (Bouras, et al., 2014).

The developers are able to implement their own bridge or use existing tools which provide access to the most common features in mobile devices, for example the GPS, through APIs that can be called using JavaScript programming language.

The part of the mobile application which implemented using web technologies can be a web page located on the web server the application has communication, or a number of HTML5, JavaScript and CSS files gathered in the application and stored on the device (Ronkainen, et al., 2013).

Each of the previous approaches, the Native development approach, Web development approach and the Hybrid development approach has advantages and disadvantage that

should be considered. The solution lies in the combination of these three approaches (Khandeparkar, et al., 2015).

2.2.4. Cross-Platform Native Applications

At the present time there is a number of tools that are available and allow the creation of Native applications using although programming languages that are unknown to the OS in which targeted. These tools do not try to imitate the native applications on the graphical interface, but are designed so that the end result is a true Native application, both visually and functionally and this application is able to run in more than one mobile OS (Perchata, et al., n.d.).

The way it works this approach is the inclusion of a runtime environment along with the application code. When the user installs one cross-compiled application in the device, then installs and execution environment that undertakes to recognize and execute the code that is unknown to the OS. The package that installs the user also includes a set of interfaces that allow the application access to the device features (Wafaa, et al., 2015).

Some of the tools that implement the above architecture are the Xamarin for the implementation of applications for iOS and Android using the C# programming language and the Titanium Appcelerator for the implementation of applications on iOS and Android using the programming language Javascript. These tools allow reuse the part of the code of the application that implements the logic functions (connection to the database, complex calculations etc.) in all versions of the application. Also the part of the application needs access to the device features can be used again after the tools offer homogenized dialling mode these characteristics, so the code remains the same. The piece of the application that should be changed is that which implements the GUI, since each operating has its own characteristic GUI.

2.2.5. Comparison of mobile development approaches

The figure 7 illustrates the relationship among the cost and time of implementation, the performance and quality of the user interphase among all the mobile development approaches which are mentioned previously. The development of Native applications is more difficult because more experience of the developers is needed, on the other hand, the Native applications provide better UX and better performance (Charkaoui, et al., 2014). The main problem in Web Applications is the partial access to device's hardware and data like the GPS and the contacts. However, the development of the HTML5 promises access to device's hardware. Another issue is that the most of the times, the applications need internet access for their operation. The applications is not installed on the device and an extra time is needed for the downloading of source code each time the user runs it (Delia, et al., 2015).

The Web approach does not follows the unique User Interphase style that each platform has. A single one User Interphase style is build and the users of each platform are forced to follow that style even are not familiar with that (Xanthopoulos & Xinogalos, 2013).

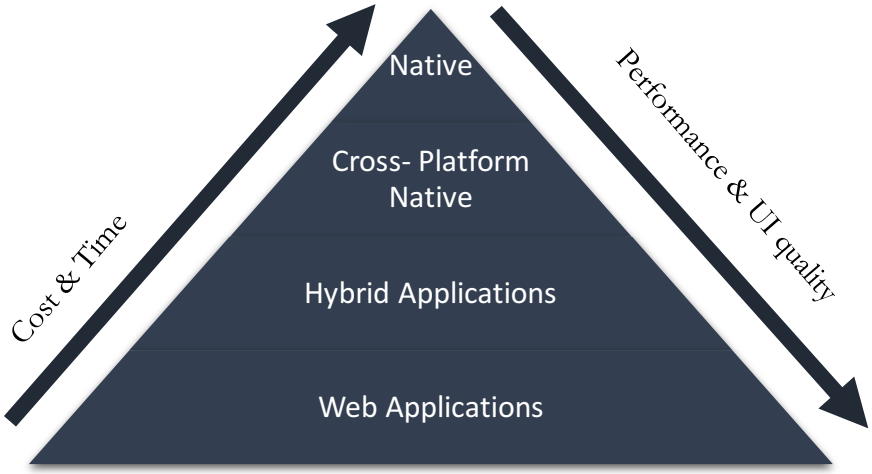


Figure 7 -Comparison among the Mobile Development approaches

The Hybrid Applications have better performance compared to Web Applications but worse than the Native Applications. Hybrid Applications are using same technologies (HTML5, JavaScript, CSS) as the Web but these applications are not running in the web browser of the device, also use APIs for the application logic and for data and hardware access and these application, in contrast of Web Applications, can be released via applications stores.

The main problem in the Native Cross-Platform applications are the dependency of the development environment compared to Web and Hybrid Applications, as example the modification of an applications which has been developed using Xamarin can be done only in Xamarin. Furthermore if one OS like Android release a new feature, that feature can be used in the application only when will be supported by the development environment (Xanthopoulos & Xinogalos, 2013). The Table 1 summarizes the comparison among the mobile development approaches.

	Native	**Cross-Platform Native**	**Hybrid**	**Web**
Development Time	Slower than all types	Medium	Medium	Faster than all types
Maintenance	Most difficult of all types	Medium	Medium	Easier than all types
Device's Features that can	Complete Access to Camera,	Complete Access to Camera,	Complete Access to Camera,	Limited Access

take advantage of	GPS, gyroscope, accelerometer, etc.	GPS, gyroscope, accelerometer, etc.	GPS, gyroscope, accelerometer, etc.	
Off-line operation	Yes	Yes	Yes	Yes (with cache) No (as Website)
Advantages	It allows the creation of applications with rich and demanding graphics.	Faster implementation than Native with access to features of the device. Reusage code on the functionality of the application	It combines the speed of implementation of Web Applications with the access to device's features	It provides fast implementation, easy maintenance and support of the code by a large number of devices and OSs
Disadvantages	Implementation time & costs, maintenance and upgrade, support only from one OS.	Expensive tools, dependency of the development environment	Cannot handle demanding graphics	Cannot handle demanding graphics. No access to all device features.

Table 1 - Mobile application types comparison

None of these approaches is best solution for every development process of mobile application. There are several aspects that needs to be under consideration. For example, if the developer needs to release the application in applications stores, if the application needs to have access to device's hardware like the camera and how time the developers are willing to spend for the development. On the other hand, there is one solution that seems to outperform, this is the Cross-Platform Native approach because it provides close-to-Native UX and performance and extensive reusability of code (Dalmasso, et al., 2013).

2.3. Cross-Platform Mobile Development Frameworks, Technologies & Tools.

This chapter refers to main technologies and to popular frameworks that developers use for mobile Cross-Platform development. The term, or the characteristic, "Cross-Platform" refers to a software that applies interoperate across multiple OSs and hardware platforms. In practice, these tools allow the programmers to develop software for mobile devices writing the code once and with limited or without modification in the code, the software is able to run on multiple OSs.

In Chapter 2, it was analysed the four types of mobile applications, the Native, the Web, the Hybrid and the Cross-Platform Native type. There exist a large number of tools to simplify the building of applications supported by a variety of mobile OS. These tools are trying to find the solution in the same basic problem, the fragmentation of the mobile OS but their priorities and the specific problems they are trying to solve are different (Xanthopoulos & Xinogalos, 2013).

There are two types of Cross-Platform frameworks, the Web-based and the Native frameworks (Boushehrinejadmoradi, et al., 2015). The first type allows the developers to build applications using web languages and technologies like HTML5, JavaScript and CSS3. The Cordova and the Sencha are examples of this type. However, these technologies provide partial access to the device hardware and therefore these tools needs additional libraries to be installed and run with with the application on the mobile device. Furthermore, the first type provides low performance, especially in graphics, and not close to Native applications UX.

The second type allows developers to overcome these limitations. The Xamarin is an example of this type. In Xamarin, programmers develop the application as they would for Windows Phone using the Windows Phone API and the Xamarin provides the translation of these API's to relevant for Android and iOS. A developer using Xamarin can develop apps in C#, using features such as generics and the parallel task library (Bilgin , 2016).

For this thesis, the Xamarin platform, the Visual Studio IDE and the C# programming language will be used. For that reason, thorough analysis will be take place for these technologies. Furthermore, because of the important role they have in Cross-Platform mobile development area, it will be analysed the following frameworks, the Apache Cordova, the Appcelerator Titanium, the Sencha Touch and the technologies these frameworks use, the HTML5, the JavaScript and the CSS3.

2.3.1. Xamarin

The Xamarin was a software development company based in San Francisco and bought by Microsoft. The Xamarin created in 2011 by engineers who created the Mono, the MonoTouch and the Mono for Android. All that are implementations of the Common Language Infrastructure (CLI) and the Microsoft .NET Framework.

Xamarin enables the developers to build mobile applications for Android and iOS with the same, shared, C# code. The UI of the application can be the native UI each platform supports or can be the same among all mobile OS as the figure 8 shows. The Xamarin has his own IDE, the Xamarin Studio, but also can be integrated with Microsoft Visual Studio. (Xamarin, 2016)

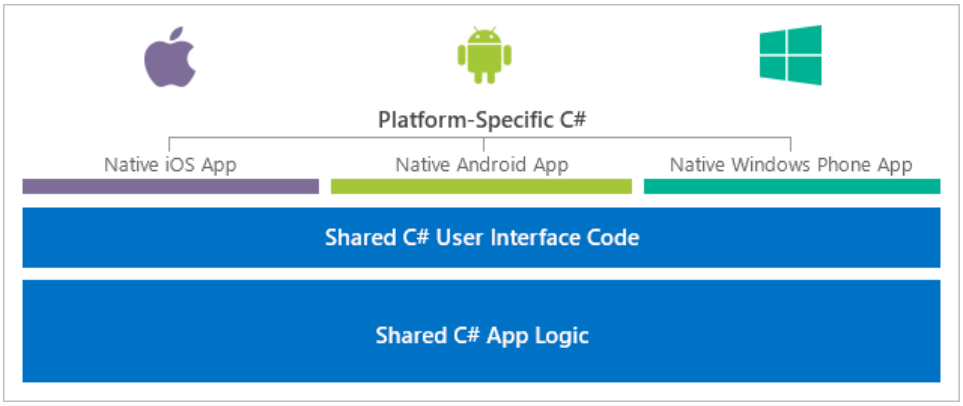


Figure 8 - Shared code which runs on all platforms (Xamarin, 2016)

The Xamarin 2.0 was released in 2013 and consolidated the iOS and the Android development tools in a single platform. The tool Xamarin.iOS and the tool Xamarin.Android make feasible the development of iOS and Android applications in C#. In Cross-Platform mobile development, the developers can reuse and can share important code between the Android and the iOS application. (Xamarin, 2016).

The source code of the application's logic among the platforms is common and according to Xamarin Company is about 80%. The developers for the UI have two options, the first is to build UI per platform and the second to use the tool Xamarin.Form in order to build the UI in C# only once. Using the Xamarin.Form, the UI of the application is the same in Android and in iOS.

The most effective strategy using Xamarin in Microsoft Visual Studio is working at the same time two projects, the first project is the Android application and the second the iOS application (Boushehrinejadmoradi, et al., 2015).

The developers split the application into two pieces, the application core, which is the business logic and contains code that is common in all platforms, and the UI code (Boushehrinejadmoradi, et al., 2015).

2.3.1.1. Xamarin for Microsoft Visual Studio

The Visual Studio is the Microsoft's Integrated Development Environment (IDE). It enables the developing of applications for the .NET Framework. The Visual Studio includes a number of tools to support the development of applications from the first stages until the release of the application. It includes a code editor with autocomplete (IntelliSence), which speeds up the writing and incorporates a debugger for direct debugging. Furthermore, it can be expanded with several additional tools available from Microsoft or third-party providers (Powers & Snell, 2015).

The Xamarin can be integrated in Microsoft Visual Studio as add-on. That enables the developers building applications using code completion and IntelliSense, which allows for exploration of the extensive landscape of iOS and Android API's (Application Programming Interface) simply by typing. In chapter 2.2.1 mention that API's are sets of routines, protocols and tools for development applications. Xamarin also provides extensions in Windows Visual Studio for creating, developing and debugging applications in a simulator or in a mobile device (Microsoft, 2016).

2.3.1.2. Xamarin.Forms

The pages of Xamarin.Forms represent the individual UI screens, is written in C# and XAML and contain layouts, buttons, labels, lists and other common features. At runtime, each page and characteristics assigned to UI components in native platforms. For example, a Xamarin.Forms Entry is UITextView in iOS and the EditText in Android (Xamarin, 2016).

2.3.1.3. Xamarin Studio

Xamarin has its own, standalone, IDE for mobile development, the Xamarin Studio. Released in 2013 and is based on the open source project MonoDevelop. It also includes debugger, code completion in C #, an Android User Interphase builder and integration with Xcode Interface Builder for iOS mobile Development (Xamarin, 2016).

2.3.1.4. Xamarin Test Cloud

The Xamarin Test Cloud enables the test of mobile applications to more than 1000 real devices in the cloud. The developers may choose to test the application in devices depending on the manufacturer, on the OS or even the people of the target market for the application (Powers & Snell, 2015).

2.3.1.5. Xamarin Insights

The Xamarin Insights enables the improvement of applications through real-time monitoring. The Xamarin Insights detects crashes, exceptions and understand the problems seeing who is online, which device is used, which activities used in each session etc. The Xamarin uses a unique algorithm to rank the topics according to the impact in the users, so that developers know where to give priority for the repair of crashes. Developers can see which users affected and the sequence of actions that preceded the crash (Bilgin , 2016).

2.3.1.6. Xamarin Requirements & Available Options

There are two ways for sharing code in Xamarin, the first is the Shared Projects and the second is the Portable Class Libraries. According to Xamarin, for the first method the developers need to have a shared folder with the common code and for the second method the developers should produce libraries with the common code and to share these libraries among the Android project and the iOS project (Xamarin, 2016).

For the development of Cross-Platform applications using Xamarin a personal computer is required. This computer should have installed the Microsoft Visual Studio IDE or the Xamarin Studio IDE. The Xamarin Studio supports development only for Android and iOS. However, for the supporting of the iOS, an Apple computer having the Xcode IDE is needed because the iOS simulator is only available on Apple computers.

2.3.2. .NET Framework & C#

The .NET is a software framework intended for the Windows platform. It consists of a large class library and supports a plurality of programming languages, for example C#, C++, F#, Visual Basic etc. Programs written for the .NET Framework run in an execution environment known as the Common Language Runtime (CLR). The CLR includes a virtual machine which handles the execution of a program and provides a number of important services such as security, memory management and exceptional handling (Capek, et al., 2015).

Programs written for the .NET class use the Class library of .NET, which provides access to the runtime environment. Basic functions such as GUIs, communication with databases, cryptography, web application development and network communications are provided via the Application Programming Interface (API) of .NET and can be combined with code from developers. As a program is limited to use library classes in .NET, can run anywhere is installed the execution environment of .NET. The C# language is one of the languages supported by .NET (Skeet, 2013).

The C# is a modern, high-level, object-oriented programming language general purpose. Designed by Microsoft for the .NET platform. It is the later series C (C, C ++, C #) from which removed which was considered dangerous as global variables, global functions, indirect arithmetic formulas conversions and add missing elements from their ancestors as garbage collector, limits control in tables, integer overflow control and more. As mentioned previously, the C# is based on the .NET platform so that it has access to all .NET libraries. (Skeet, 2013)

The C# is a constantly evolving language and each new version has new features. According to (Microsoft Inc, 2016), some of the main features are:

- **Generics**
  Generics enables the developers to define type-safe data structures without pledging to actual data types.
- **Anonymous Functions**
  Are expressions or statements that may be used anywhere a delegate type can be. The developers can use anonymous functions in order to initialize a named delegate or in order to pass it instead of a named delegate type as a parameter in a method. There are two kinds of anonymous functions, the first kind is the Lambda Expressions and the second kind is the Anonymous Methods
- **LINQ**
  LINQ is a set of features which enable querying and updating data in C# code
- **Var**
  Var is a data type which is determined at the execution time and not at compile time as the other data types.
- **Delegates**
  Delegate is a data type (reference type) which provides a secure way to encapsulate functions

2.3.3. *Apache Cordova (PhoneGap)*

The Apache Cordova, former PhoneGap, is a popular platform for developing hybrid Cross-Platform mobile applications.

As it can be seen in the figure 9, the Apache Cordova allows software developers to create applications for mobile devices using JavaScript, HTML5 and CSS3, instead of relying on the API of each platform. Allows wrapping of CSS3, HTML5 and JavaScript code depending on the device platform (Wafaa, et al., 2015). Extends the capabilities of HTML5 and JavaScript to work with the device. The resulting applications are hybrid, meaning it is neither truly native mobile applications because the entire device performance take part via web-views instead of the natural UI framework of the platform nor purely Web-based because it is not just Web application, but is packaged as parent applications that have access to the APIs of the device.

The core of Apache Cordova applications using HTML5 & CSS3 for the UI and JavaScript for the logic. The HTML5 provides access to the underlying hardware, such as an accelerometer, camera, and GPS (Liu, et al., 2015).

Figure 9 - Apache Cordova (Microsoft, 2016)

The Apache Cordova incorporates HTML5 code into a native Web View to the device using a foreign Operation Interface to gain access to the inherent resources of the device. The Apache Cordova may have extensions as plug-ins. These plug-ins allow developers to add functionality using JavaScript, allowing the communication between the physical layer and the HTML5 page. It includes the key links that allow access to the device's features, for example access to accelerometer, camera, microphone, and more.

Although, the use of Web-based technologies has resulted, the Apache Cordova applications running slower than Native applications with similar functionality (Dalmasso, et al., 2013).

The PhoneGap Build is the cloud service of Apache Cordova/PhoneGap. It allows developers to upload HTML5, CSS3 and JavaScript code into a cloud compiler that produces applications for each supported platform like iOS and Android. Using PhoneGap Build, the developers do no need to install and maintain multiple native SDK's for local development applications except Cordova SDK. The programmer can target iOS and Android with a single code base, so maximizes productivity while reducing production time. The PhoneGap Build provides another tool for users, the Hydration having two main advantages. It reduces the time needed to compile and debugging and updates are forwarded directly to the application installed on the device (Adobe, 2016).

2.3.4. Appcelerator Titanium

The Appcelerator is an IT company based in California, which introduced the Titanium platform in 2008.

The Titanium is an open-source framework which enables the creation of applications for desktops, tablets, mobiles using programming languages for the web such as HTML5 and JavaScript. The Titanium consists of an SDK that provides tools, compilers and API's for development applications. The Titanium is available for Mac, Linux and Windows. For the developing of iPhone applications a Mac computer and an iPhone's is needed according with the iOS SDK. Developing for Android requires Android's SDK (Appcelerator Inc, 2016).

The main features of Appcelerator Titanium include:

- A cross-platform API for accessing in UI ingredients such as navigation bars, menus and dialog boxes and access to native device function, including the file system, the network, the accelerometer, and the GPS.
- Transparent access to native features that are not already covered by the API.

All source code of the application is sent to mobile device where interpreted using a JavaScript engine. Loading programs created with Titanium needs more time than those developed with native SDK's as the interpreter requires extra libraries to be loaded (Dalmasso, et al., 2013).

2.3.5. Sencha Touch

The Sencha Touch 2 is an open source framework for mobile cross-platform applications. The first version of the Sencha Touch released on 2010. The Sencha Touch 2 is the second version. The Sencha Touch produces hybrid applications which are written in HTML5, CSS3 and JavaScript. The aim of the framework is to allow developers to create easily and quickly applications for Android and iOS. For development process is required a web server which runs locally, in the developer's personal computer. Furthermore, Sencha Touch takes advantage of hardware acceleration (Sencha Inc, 2016).

The Sencha Touch includes a set of GUI controls for use within the mobile web-based applications. These components include buttons with themes, effects, text fields for example

e-mail, date and address, slider, selectors, combo-boxes, control momentum for scrolling, index bar, toolbars, menus, movable tabs, bottom toolbars and a map component with support for multi-touch gestures such as pinch and zoom. All components may be formulated according to the device for which intended. This is done using SASS (Syntactically Awesome StyleSheets), a stylesheet language built on the CSS. The Sencha Touch has a built-in effects transition as slide above or below the current item, pop, flip and cube. Supports ordinary tap touch gestures, double tap, swipe, scroll and pinch (Ji, et al., 2013).

2.3.6. HTML5, CSS3, & JavaScript

The HTML5 is the new web standard and provides new functions for applications more powerful and more flexible. The HTML is the primary mark-up language for web applications and websites. The HTML is written in the form of data consisting of labels enclosed in symbols acting pairs, for example <html> </ html>. When a web server opens an HTML data translated into appropriate characteristics results in the appearance and functionality of the page file (W3C, 2016).

The HTML5 is the new and more modern standardization of HTML, XHTML and HTML DOM. Its development, is still ongoing, but many modern browsers, support many of the new features that incorporates the HTML5.

For the implementation of the HTML5, the Word Wide Web Consortium established some new rules (W3C, 2016). For example the new features needs be based on HTML, CSS, DOM (Document Object Model) and JavaScript, the development process should be visible to the public, better error handling, more mark-up which will replace the scripts etc. The HTML5, introduces several new modern elements, such as the element of canvas for painting, the items *video* and *audio* for media playback, better support for local offline storage, new elements specifically for content like article, header, footer, navigation, new operations in forms, such as calendars, date, time, email, url, search etc (McLaughlin, 2011).

The CSS is a style sheet language used to control the appearance of a web application. Changes can be made directly in the HTML document using the tag <style> or creating a CSS file and passing a label on what we want to change, either class or id we can do to change (Sin, et al., 2012).

The JavaScript was created in 1995 and it is an object-oriented, interpreted, scripting programming language, not mark-up such as HTML. The work of JavaScript is communication with the client (client-side), although in recent years have created Node.js type models that do and the server-side (Joyent, 2016). The JavaScript is used mainly in the side of the customer, and is implemented as an integrated component of the web browser, allowing the development of enhanced interfaces and dynamic websites and applications (Stefanov, 2010).

2.4. Professional, Legal, Ethical & Social Issues

This thesis takes under consideration the Professional, the Legal and the Social Issues. Also, it respects all the human rights, the laws, the rules and the values that the academic world and the laws sets.

2.4.1. Professional Issues

At the evaluation of the application, the users will be called to participate in study and to answer questions about the applications. The users should not be pressured or enforced. The evaluation process will be handled with professionalism. Furthermore, the quality of the applications must be in high level.

2.4.2. Legal Issues

Copyright issues must be adhered. For all the software that will be used for this thesis, there will be a legal license. The Xamarin and the Microsoft Visual Studio are commercial software and for the developing of the application should be used a free student license from Microsoft DreamSpark program. If for the development of the application will be used third party code then this usage should be by the terms and the contrition that are provided.

2.4.3. Social Issues

Users will take part in the evaluation of the prototype application. The personal data of these users will not be recorded and the all the data from the evaluation process will be handled with respect to Data protection law.

2.4.4. Ethical Issues

The privacy and the security that the application is able to provide are two significant issues. The security of data storage and data transfer as well as the privacy the users are receiving, for example the location tracking violates the user's privacy. The applications are developed by Cross-Platform tools do not provide the security level that the Native applications provide (Dalmasso, et al., 2013). The application will interact with the OS and it will be ensured that the OS and the user's data is not effected.

CHAPTER 3
METHODOLOGY

This thesis involves two phases, the first is the development of the Cross-platform application using Xamarin and the second is the evaluation of this application in terms of the UX this application provides in iOS and Android. This chapter illustrates the development and the evaluation methodologies.

3.1 Waterfall Methodology

The waterfall methodology was chosen for this thesis. The waterfall is one of the most popular development methodology. The main idea is that the development of the software passes across successive sub-phases, each of these is considered concluded with the production of certain deliverables. Each individual sub-phase is completed with a work verification/validation of the deliverables. The main characteristic of the waterfall methodology is that, in order to begin a phase, the previous phase must be fully completed. The growth in this manner is characterized sequential because the individual phases of which are discrete passes and follow one another (Larman & Victor, 2003).

The first stage is the definition of the requirements and the detailed design of the software is following. During the design, the definition of the units of the software is taking place. The development of the units is following and finally the combination of the units and the final control of the entire system. The waterfall methodology is mainly useful in cases where the requirements of the software are already known and are not changed during the development.

The advances of this methodology are:

- It is ideal for teams and developers with little relevant experience.
- The sequence of stages of development and the rigorous testing ensure the quality, the reliability and the maintainability of the software.
- The progress of system development is measurable.

The disadvantages are:

- The large structure and the strict controls which are driving to slow and costly development.
- The project does not include many steps where feedback can be used.
- It depends on early identification and specification of requirements, but these requirements may not be able to be clearly defined before the start of the project.

- The system performance can not be tested until the system is finished and it is difficult for a problem to be solved when the system is in production (Larman & Victor, 2003).

For this thesis, the first stage was the gathering and the analysis of the requirements for the Cross-Platform application. The kind of the mobile application was chosen and designed in this stage. The second stage was the initial design of the application according to the requirements which was gathered in the first stage and also according to initial requirements about the Cross-Platform nature of the application which analyzed in the Literature review. The third stage was the implementation of the application. The functionality of the application splinted in four modules. The core functionality developed and the next increments added one by one. The fourth stage was the evaluation of the application. The figure 10 illustrates these stages.

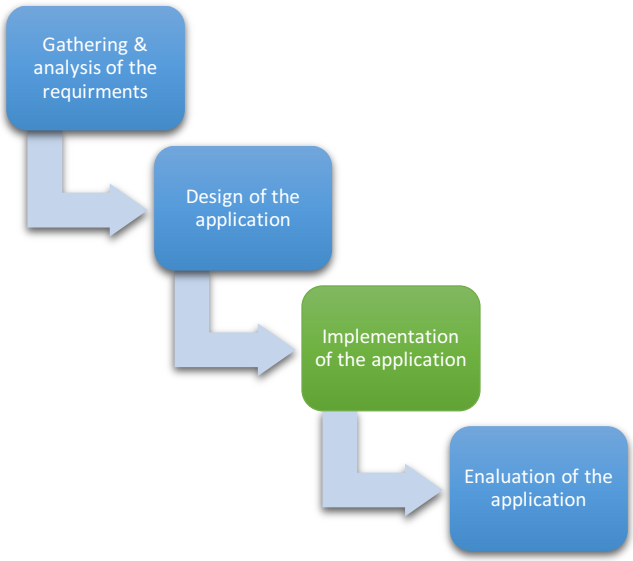


Figure 6 – Waterfall Methodology Stages for this thesis

3.2 Evaluation Methodology

This section is concerned with the evaluation methods adopted in this study, why they are chosen and how they were designed.

This thesis evaluates the UX a Cross Platform applications, which is developed using Xamarin, provides in Android and iOS mobile OS. A multimethod design was used including a usability test and two questionnaires. Collecting data from users who are using mobile devices like smartphones and tablets will provide a rounded view of the UX and the similarity of the application in Android and in iOS.

3.2.1 Usability testing

According to (Shneiderman, et al., 2009), a widely accepted method for the evaluation of UI and the UX is the usability testing which is concerned with performing tasks by representative users on the application. In this case, real users will perform tasks on the mobile application in both of the platforms and they will provide their opinion about the application. According to (Dumas, 2003), the people that take part in a usability test are possible users and at the end of the procedure, the data gathered is analysed for results. Issues like the technical equipment should be taken into consideration and the users should be free while performing the tasks so that the results are not contaminated by the observer.

3.2.2 Test Objectives

The test was conducted to evaluate the prototype Cross-Platform mobile application's:

- **Functionality**
  Does the application supports the same functions and provide the same features in Android and iOS?
- **UX**
  Is the provided User Experience the same in both of the platforms?
- **UI Similarity per platform**
  Does the User Interface of the Android application look like as other Android applications?
  Does the User Interface of the iOS application look like as other iOS applications?

3.2.3 Selecting the participants

The participants of the study should be potential users of a Cross-Platform application. According to (Dumas, 2003) the ideal number of the users involved is six but for more accurate results, the application should be tested by more users. In this study, the application was tested by 11 users.

3.2.4 Selecting the tasks

The tasks the participants will perform should cover the main features of the application, should be realistic and not be time consuming (Nielsen & Levy, 1994).The participants received a list of eleven tasks which were chosen to cover all the functions of the application. It is vital to be declared to the users that the application is tested and not them. The tasks are discussed in next section.

3.2.5 Test Administrator Tools

Two questionnaires were used, the first one before the performing of the tasks and the second one after the performing of tasks. The first questionnaire had as purpose the gathering on demographic data about the participants and the second one the gathering of the participants' opinions about the Cross-Platform application in both of the platforms. The timing equipment was two Garmin™ chronometers. The margin of error allowed between the two chronometers was 0.3 seconds.

3.2.6 Procedure

Each participant was retold that he has been asked to take part in a usability test about a mobile application in two mobile devices. The test is involved of three phases and the participants could withdraw their participation at any phase. An informed agreement form received and signed from each participant and the first questionnaire was given and was filled. The next phase was the performing of the tasks in the two mobile devices and the last phase was the filling of the second questionnaire.

3.2.7 Questionnaires

According to (Saunders, et al., 2009) a questionnaire is a well established method of gathering data. In questionnaire, participants are requested to answer to the identical set of questions in a prearranged way and not customized to each individual.

The aim of this study was the evaluation of a Cross-Platform mobile application, which developed using the Microsoft's Xamarin, in Android and iOS mobile OS. The main concern is the UX the application provides in both of the mobile OS and the functionality the application is able to provide in each platform. The first questionnaire was about gathering demographic data from the users and the second one was asking questions about the aim of the study. The two questionnaires as long as the user's answers can be shown in the Appendix.

CHAPTER 4
APPLICATION REQUIREMENTS

4.1 Initial requirements of the project

The application needs to be able to be installed and run in the latest versions of Android and iOS.

The application needs to be able to offer the same functionality in the previous mobile OS.

The UI of the application in both of the mobile OS should be similar.

The users must operate the application immediately after the installation in the device.

The application needs to have access to mobile device's features like the Camera, the GPS, etc. and must take advantage of these features.

The application needs to be able to be released via the application stores that each mobile OS supports, via Google play for Android and via App Store for iOS.

The reusability of the code must be the maximum possible.

The application has to be optimized for the sources that needs and must respect the battery of the mobile device.

For the development of the application must be used the Xamarin and the C#.

The application needs to respect people with disabilities and to provide access to them.

4.2 Decision about the type of the application

According to the requirements of the application the Cross-Platform prototype application should take in advantage the features of the mobile device such as the camera, the speaker and the GPS of the device.

It was chosen to be built an application which helps the productivity of the people. According to the Google's and Apple's application stores (Google, 2016) (Apple, 2016), most of the applications in the productivity category are notepads. The most famous is the

Evernote (Evernote Corporation, 2016) and the OneNote (Microsoft, 2016) which are released for Android and iOS mobile devices. The users are familiar with the notepad applications and the implementation and the evaluation of a Cross-Platform application of this kind could be very interested.

The functional requirements of the prototype application are following.

4.3 Functional requirements of the application

The user of the application shall be able to:

- Keep text notes with a title for each note.
- View the list of the text notes.
- Edit and delete a text note.
- Keep tasks with a title for each task.
- View the list of the tasks.
- Edit and delete a task.
- Set a task's status to "done" or "not done".
- Save the current address of the device giving a title to the address.
- View the list of the addresses.
- Edit and delete an address.
- Take driving directions to a saved address.
- Take a photo using the camera of the device.

4.4 Accessibility

According to (European Commission, 2014), 10% of the population in developing countries are people with Disabilities. In these people need to be added the growing proportion of elderly, the people how facing substantial difficulties in using a particular service or device and also those who have difficulties with new Information and Communications Technologies (ICT).

Accessibility is the possibility for a service to be used from everyone, regardless any disability he may have and it is embedded in a broader scientific field of Human-Computer

Interaction (HCI) and in particular the "Universal Design" or "Design for all". According to (Nielsen & Landauer, 1993) accessibility is by definition a matter of usability because anything that is not accessible to a user is not handy. Types of disabilities include various physical and mental impairments, one type is the vision disability. In our society there are people with low vision or blindness or colour-blindness. For these people was chosen to be able to listen the saved material of the application. The description of the implementation of this feature if following in the chapter 6.

CHAPTER 5
APPLICATION DESIGN & TECHNOLOGIES

In this chapter it will be demonstrated how the application was designed according to the requirements, the technologies which were used for the implementation and the decisions were taken about the design and the technologies. It will also be described the design of the prototype application. It is important an initial plan for the functionality of the application before coding.

5.1 Description of the application

As it was mentioned in the previous chapter, that the prototype application is a Notepad Application, like Microsoft's OneNote, and it targets Android and iOS mobile devices. The application uses a local SQLite database and it takes in advantage the camera, the GPS and the speaker of the mobile device.

The application is spited in four core parts, the MyNotes, the MyTasks, the MyAddress and the MyPhotos. The user, in the main screen of the application can choose one of these four modules. In the MyNotes, the user can write and save a text note giving a title. The user can also delete or modify these text notes any time he wants. In the MyTasks, the user can store tasks he wants to do and set them as "done" or "not done". Furthermore, the users are able to modify or delete these tasks. The third module is the MyAddress. In this module the user can save his local address using the GPS of the device and can retrieve this address later. For instance, the user can store the exactly address he parked his car and in a later time can receive navigation in order to go back to his car. Using the MyPhotos module, the user can take photos using the camera of the device directly from the application.

The application is able to read to the user the Notes, the Tasks and the Addresses which are saved. The Appendix B illustrates the mock-up GUI of the application.

5.2 Why Xamarin.Forms

The two reasons that the Xamarin.Forms was chosen for the implementation is the maximum reusability of the code that provides and also that the UI of the application is developed once and in each platform looks native, for example the UI of the Android version of the application looks like the other Android applications. It was discussed in

Literature Review Chapter that the Xamarin.Forms translates the form UI into a native implementation on the devices. Thus, the user gets fast and fluid output on the screen that looks like other native apps (Hermes, 2015).

Xamarin is a way to make mobile apps for all platforms with a single, shared C# code base. There are two approaches for Cross Platform Development using Xamarin. In traditional Xamarin development, the same application in Android and in iOS is sharing the same business logic in a form of a portable class library or a shared project or a combination of the two. The code is sharing in this approach typically includes things such as view models, business logic, web service access code, and data structures. According to (Xamarin, 2016) the sharing code is often about 75% to 90% of the code base, depending on how complicated the application is. The rest 10% to 25% of the code base that ends up being platform specific and defined in a platform specific project that targets iOS and Android.

For this thesis was used the second approach, the Xamarin.Forms. According to (Hermes, 2015) the Xamarin.Forms is not always the best approach to use for apps. It depends on the type of application. If the application is mostly composed of data forms and the reusability of the code needs to be maximum then the Xamarin.Forms is highly recommended and for these reasons it was used.

The sharing of the code between the platforms must be the maximum possible and the application must provide native look in each platform. The Xamarin.Forms provides a library of common controls which are implemented across the target platforms and it allows the definition of the UI in the shared code. At runtime these shared controls create native views for each platform. Because of that, the application still get the native look and feel but the developer have to write the UI once and have it shared on iOS and Android (Snider, 2016). As it can be seen in the following figure, both of the devices shows a screen with a search bar. The same code used for the implementation of this bar but the presentation depends on platform and it is different.

Figure 11 - Native Views

Xamarin.Forms provides support for building fully interactive UI by including support for a standard set of controls such as buttons, edit fields, labels, pickers and sliders. Also, it provides a layout system for the position of these elements on the screen The Xamarin.Forms supports the use of XAML for the definition of the UI in a mark-up format, and also a full data binding infrastructure which allows for separation patterns such as MVVM to be used to connect the data to the UI. These technologies will be described in detail in next section.

5.3 Sharing Code approaches - Code Reusability

A key factor of building Cross-Platform mobile applications is ability of sharing code across different platform projects. There are two different ways for sharing code between cross-platform applications in Xamarin.Forms, the first way is by using a Shared Projects and the second is by using a Portable Class Library (Snider, 2016). For the prototype of this thesis, a Portable Class Library was used. This section explains why this approach has been chosen and how a Portable Class Library work.

Using a Portable Class Library as sharing code approach has advantages and disadvantages comparing to Shared Projects approach. According to (Microsoft, 2016) the advantages are: the centralized code sharing, the writing and the testing of the code is taking place in a single project which can be expended by other libraries. The refactoring of the code is easy because it affects immediately the platform-specific projects. Furthermore, the Portable Class Library can be referenced by other projects in the future. The key disadvantage is that platform-specific libraries cannot be referenced. For instance, it is not allowed in the Portable Class Library the using of specific Android libraries.

According to (Hermes, 2015) that disadvantage can be bypassed using Dependency Service for coding in the platform projects and also an interface class that is defined in the Portable Class Library. The section 6.4 explains how Dependency Service works.

The Shared Project approach has as disadvantage that the developer cannot share code with other solutions or other projects in the future. The maximum reusability of the code is critical and because of that, the Portable Class Library approach used. This approach involves a project per platform (one project for the Android application and one project for the iOS application) and a Portable Class Libraries.

The solution for the application has the following projects:

- Portable Class Libraries
- AndroidApplication
- iOSApplication

The figure 12 illustrates the projects of the application. The Portable Class Library is the "Mnimi" project.

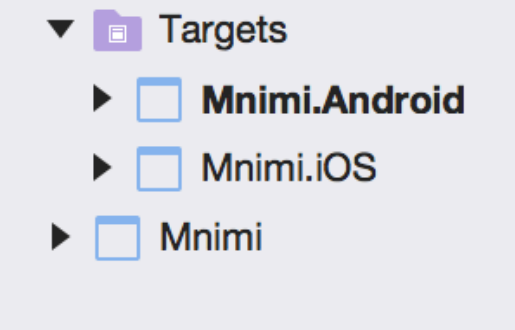


Figure 12 - Project Hierarchy

The reusability of the code is affected by the architecture and by the OOP principles, for instance the Encapsulation ensures that the code for the UI is responsible only for displaying data in the screen and receiving inputs from users, the UI is never interacting with the database straight. A well structure architecture of the code has as result an application with code separated into layers (Microsoft, 2016). For example the Data Layer which is the data of the application, for the prototype the Data Layer is a SQLite database. The Data Access Layer which provides access to the Data Layer without revealing operation details to the caller. The UI Layer which contains the views of the user. The figure 13 presents the content of the Portable Class Library and the two projects.

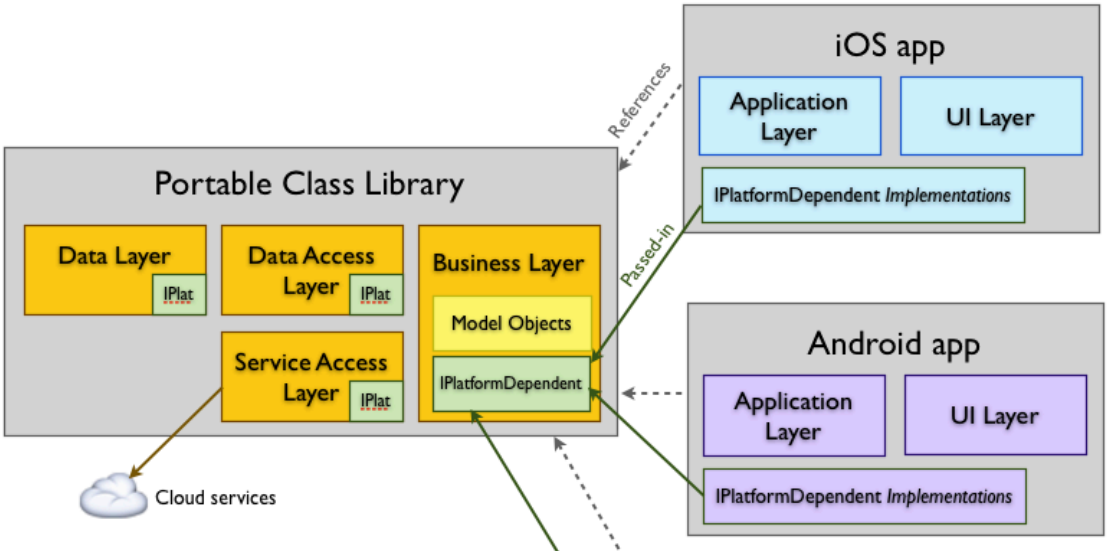


Figure 13 - Portable Class Library (Xamarin, 2016)

5.4 User Interface Design

The design and the development of the GUI of the application can be effected by a number of factors such as: the size of the screen the device has, the hardware of the devices and the input methods of the mobile application (Brooks, 2014). In particular, the development of a good looking application with large images and animations is not desired because this kind of GUI needs significantly processing power of the mobile device. In

mobile devices with entry level hardware this can decrease the usability and the user satisfaction.

Having that in mind, the GUI of the Cross-Platform application tried to find the balance between the performance and the presence. The UI is following in a minimalist design, having although enough quality to increase user satisfaction. The initial mock-up of the GUI with a description of every screen is in the Appendix B.

Regardless the diversity of the mobile OS, there are features of mobile applications that are common between the mobile OS and can therefore be the foundation of the application's design. For instance: Feature selection via tabs or menus, Lists of data and scrolling, Single views of data, Editing single views of data and Navigating back. The application uses these features and is expected by the users to be familiar with these. For example, the titles of the Notes are illustrated in a list and is expected the users to know how to chose one of these and how to go back to the previous screen.

Xamarin.Forms are based on the concept of pages, layouts and controls. The pages are the screens of the application. Controls are the UI elements shown on a page. For example the buttons. In addition to controls, the layout panels help arrangement of the controls on a page.

In Xamarin the elements shown on the smartphone screen are called visual elements. They come in several categories. An app consists of one or more pages. The application consist of ten pages. The page is a screen of the application. Pages usually occupy the entire screen. The children of the page are usually contained in at least one layout element. Layouts are responsible for arranging and positioning their children in to the desired location. The children of the layout consist of views and additional layouts. Xamarin calls their interactive elements views. A cell is a specialized element used for items in a table. They are intended for use in list view and table view elements (Xamarin, 2016).

By default, all pages created from the Xamarin templates are content pages. Then there is the master detail page, the navigation page, and the multi page classes, the tab page and the carousel page. Layouts are the controls that are responsible for arranging elements on the page. Some layouts have a single child to manage while other layouts compose multiple child elements. There are layouts that stack elements, ones that use absolute positions, and others that use a grid metaphor, moving elements in to columns and rows (Hermes, 2015).

For instance, the MyTask screen (page) of the application illustrates in a Stack Layout (one-by-one) all the titles of the Tasks (elements). This screen is also a navigation page because the user can visit this page and can go to the previous page he was before.

5.5 Data Model

From the requirements of the application, it is already known the items which need to be saved. The items are the NoteItem, the TaskItem and the GpsItem. The properties for each item in the table 2 below.

NoteItem	TaskItem	GpsItem
• **int ID** • string Name • string Notes	• **int ID** • string Name • string ToDo • bool Done	• **public int ID** • public string Name • public string Latitude • public string Longitude • public string Address

Table 2 - Data Model

Each NoteItem has a unique (key) ID which is an integer, the Name (the title of the note) which is a string and finally the Notes the user writes which is also a string. The TaskItem is similar to the NoteItem but it has one more property, a Boolean property with the name Done. This property represents if the tasks has or has not be done. The final item is the GpsItem which keeps the address of the user. It has as properties a unique ID, the Name the user gave to that address, the Latitude and the Longitude of the address as also as the specific address of that position. The application uses SQLite.NET and the chapter 6.2 describes in detail the implementation.

The photos that the application takes are stored directly into the external storage space of the device because it is not recommended the conversion of the photo is binary format in order to be stored in the database (Feiler, 2015). If the device does no have external storage, for example does not support SD cards, then the photos are stored in the internal storage of the device.

5.6 XAML & C#

The XAML is a declarative programming language for building and initializing objects. It is similar to XML, but XAML has a set of rules on data and characteristics and matching them to objects, their properties and the values of these properties. The XAML is consisting of rules for how compilers should analyze the XML and has keywords, but does not specify any interested components alone.

The specifications of XAML define the rules that match .NET types, properties, and events, to XML types, properties and events. The declaration of an XML element in XAML (known as element object), is equivalent to the initialization of a .NET object by a constructor in C#. Giving a value to an attribute, it is like giving value to the corresponding property of the same name of the object, or as to add to the button methods call list for the click, an event with the same name (Microsoft, 2016).

For instance:

XAML code

```
<Button Text="Save" Clicked="saveClicked"/>
```

C# Code

```
void saveClicked(object sender, EventArgs e)
      {
          var NoteItem = (NoteItem)BindingContext;
          App.Database.SaveItem2(NoteItem);
          this.Navigation.PopAsync();
      }
```

The Button with the text "Save" is defined in the XAML code and at the clicking of this button, the method "saveClicked" runs. This method has place in the C# file. The next sections describes in details the separation of the XAML code versus C# code and the MVVM pattern which is behind of this.

5.7 Data Binding & MVVM

The term data is generally used to describe an arbitrary .NET object like a table of a database, a custom object or even a control like a Button. The data binding is a way to tie arbitrarily .NET objects. Data-binding connects two objects called the source and the target. The source object provides the data for the binding. The target object is the destination for the data and usually, this is a visible object like a label in the XML file. As a result, the bound data is shown on the page.

There are times when it is sensible to have a two-way Binding. For example the Entry view which has editable text. The user can change it. With a two-way Binding, the data can flow back into the data source. The Binding starts by showing the data from the Description property. When the user changes the data in the Entry view, the Binding updates the Binding source, the Data Base The main benefit of data-binding is the synchronising of the data between the views of the application and the data source. The changes in the source object are automatically pushed to the target object. Plus, changes in the target object can optionally be pushed back to the source object (Snider, 2016)

A dominant pattern that has emerged for GUI creation technologies with XAML, is the Model-View-ViewModel (MVVM). The MVVM is an architectural pattern that splits the logic of the GUI from the logic of the other functions of the application running behind it. The model leverages the capabilities of data binding in XAML technologies and enables loose coupling between the View and the ViewModel, so the ViewModel does not need to handle the View directly. This virtually eliminates the need for C# code behind every XAML page (Microsoft, 2016)

The basic elements of MVVM pattern are:

- Model - The model is responsible for the management and delivery of data.
- View - The view is responsible for the demonstration of data.
- ViewModel - The bridge between the model and the view.

The ViewModel is responsible for managing the state and behaviour for the view. The view binds to properties on the viewmodel. The viewmodel implements

INotifyPropertyChange, so when a value changes in the viewmodel, the binding engine updates the view.

The viewmodel works with the models to get the necessary view data. The viewmodel exposes these values as bindable properties. This keeps a layer of separation between the model and viewmodel. The model doesn't have any view-specific features which makes the model more flexible when using with other service layers or UI clients. The viewmodel adds other properties that makes sense for the view, for example, the ImageURL property is important for showing pictures in the UI. Finally, the viewmodel can contain code which can be invoked by the view.

In normal situations, the button would have an event handler in the view code but MVVM is about separation of concerns. To keep the separation, MVVM uses the command concept. A command is class that can invoke code. The viewmodel exposes the command as a property on the viewmodel. The button and other elements have a bindable target property which can be bound to the command. Now when the button is pressed, the binding engine invokes the command.

The MVVM is based on assigning a ViewModel as DataContext on a View. This is usually done in the View constructor. With the award of the ViewModel in the DataContext of View, the properties can be used as binding expressions in the XAML of each page.

The following code is for the illustration of all the Notes in a list. Data Binding is used for the representation of the name of each Note. Furthermore, for the representation (View) of all the data from the selected Note (Model), the Data Binding is used.

XAML code:

```
<ListView x:Name="listView" ItemSelected="listItemSelected">
      <Label Text="{Binding Name}"
```

Model:

```
public class NoteItem
    {
        [PrimaryKey, AutoIncrement]
        public int ID { get; set; }
        public string Name { get; set; }
        public string Notes { get; set; }
    }
```

The method for the selection of the data:

```
void listItemSelected(object sender, SelectedItemChangedEventArgs e)
      {
            var noteItem = (NoteItem)e.SelectedItem;
            var notePage = new NoteItemPageX();
            notePage.BindingContext = noteItem;
            ……
            ……
      }
```

CHAPTER 6
IMPLEMENTATION OF THE APPLICATION

In this chapter it will be demonstrated how the application was developed, compiled and tested across all systems using the Xamarin.Forms.

For the building and the deploying of an application, a specific process should be followed. The developer environment needs to be setup, a project must be created and the code is written. During the development phase the code is written and the application modules or class libraries are created. The resource files, for example the images for the application are also assembled and added in the project. The Build, Debug and Test phase is the building of the project into a binary package. Then, that binary is transferred to the emulator or device. Once the app binaries are ready, the application is debugged and tested (Snider, 2016)

The work for setting up the development environment for Xamarin is done through a pair of installers. One for Mac OS and one for Windows. Microsoft provides for free these installers. For this Thesis, the free Community Edition of the Visual Studio was used. For the compile of the Android application, the Android SDK tools are needed and the compile process took place in the same Windows machine.

For the compile of the iOS application, Apple's developer tools were needed. All these tools are integrated in Apple's IDE named Xcode. Xamarin allows the build process using Visual Studio and Windows, delegating the required parts to a networked Mac machine. This is done through a software server process that was installed on the Mac machine known as the Xamarin Mac Agent. The Visual Studio connects to this process over the network using SSH and delegates a portion of the build to the Mac.

There are two ways to run and test the application. The first is onto physical devices and the second is onto simulated devices on the machine through the use of simulators. For this Thesis, simulators and two real devices were used for testing during the development process. An iPhone 5s with iOS 9.3 and a Huawei MediaPad T1 with Android OS, v4.3 (Jelly Bean).

It was noticed that simulating the environment tends to be a faster than using a real device, particularly for the build, run and test. Also, simulators can replicate a variety of form factors and versions, allowing the test the the application on different types of devices that was not physically own. However, according to (Hermes, 2015) is important the testing in real devices for making sure that everything works as expected.

For the development of the application, the Visual Studio 2015, the Microsoft's developer IDE, used. There are three editions, the free Community edition and the licensed Pro and Enterprise editions. Xamarin works in all three editions. One Mac is needed for the compiling of the iOS apps from Visual Studio.

6.1 Core Functionality

The prototype application is a combination of four modules the MyNotes, MyTasks, MyAddress and MyPhotos. The section 5 describes the functionality of these parts in details. This chapter describes the implementation of theses modules.

The code of the application can be separated in two parts. The first part is the common code of the application and the second part is the specific code of each platform. The common code covers the code which is the same between Android and iOS and this is the code for the creation of the database, the code for the data items, the views of the application and a part of the code for the MyPhotos module which handles the camera of the device and a part of the code for the "speak" feature that the application provides. For the reusability of the code, as it mentioned if the chapter 5.3, the common code is taking place in a Portable Class Library. The other two projects which are reference in the Portable Class Library are the Android and the iOS project.

6.2 Data Storage

According to (Feiler, 2015) there are advantages of using an SQL database in a mobile application. SQL databases allow well-organized storage of structured data, furthermore explicit data can be extracted with queries. The results of the queries can be sorted and can be combined

In the prototype application, an open source library, the SQLite-net, used for the needs of storing data on the mobile devices locally. The SQLite-net is a library that enables the application to store data in an SQLite database (Feiler, 2015).

This library is available in software packages of NuGet. To be add must in Solution Explorer to be done right click on every project and select Manage NuGet Packages. Then the sqlite-net needs to be found and installed. The figure 14 below illustrates this procedure.

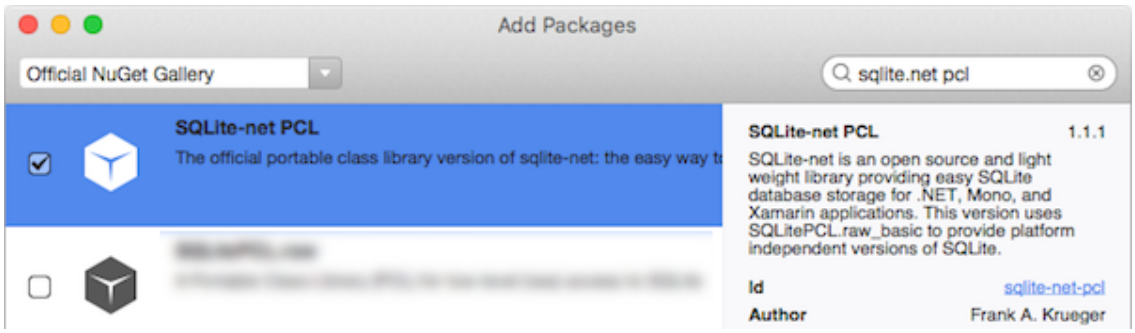


Figure 14 - Download a NuGet Package

The SQLite is a Database Engine that allows for data storage without the need to run as a service. The database is stored in a file that is stored locally in the Storage of the application. The main issue at the storing of the data is that each OS stores the database in different locations. The path is different in Android and in iOS. For the creation of the database firstly this path is needed be found. The following code returns the path for the iOS device.

```
public SQLite.SQLiteConnection GetConnection ()
        {
            var sqliteFilename = "Mnimi.db3";
            string documentsPath = Environment.GetFolderPath(Environment.SpecialFolder.Personal); /
/ Documents folder
            string libraryPath = Path.Combine (documentsPath, "..", "Library"); // Library folder
            var path = Path.Combine(libraryPath, sqliteFilename);

            var conn = new SQLite.SQLiteConnection(path);

            // Return the database connection
            return conn;
        }
```

A reference to database can be created by passing the file path the SQLiteConnection class constructor.

```
var db = new SQLiteConnection (path);
```

For the creation of tables, firstly is needed the creation of classes with properties the fields which are needed to be stored. One class is equivalent to a table. All properties containing represent the columns in the table . For example:

```
public class GpsItem
    {

        [PrimaryKey, AutoIncrement]
        public int ID { get; set; }
        public string Name { get; set; }
        public string Latitude { get; set; }
        public string Longitude { get; set; }
        public string Address { get; set; }

    }
```

The above class represents a GPS Item which is a location. The ID is the primary key, each location has its own, unique ID which is an Integer and takes auto increment values. The user gives the Name of the location. The Latitude, the Longitude is taken by the GPS sensor of the device.

The creation of the tables is following. The code bellow illustrates how a table is created.

```
database.CreateTable<GpsItem>();
```

In SQLite -net there is no Select method. In order to retrieve the data from a table of the database, the table needs to be converted into a List. For example:

```
public IEnumerable<GpsItem> GetItems()
 {
     return (from i in database.Table<GpsItem>() select i).ToList();
  }
```

The above functions returns the List with all the saved locations.

Though it is possible the storage of images in the database on a mobile device, according to (Burnette, 2015) it is recommended the storage of them directly in the file-system.

6.3 Page Navigation

The users are able to navigate through pages of the application, forwards and backwards, as desired. The class NavigationPage provides that functionality. That class implements a stack. The active page is in the top of this stack by pushing from the application. When the users press the Back button, the application will pop this page from the navigation stack, and the new top page will become the active page.

The first page added to the navigation stack is the root page, the HomePage is the root page and the following code illustrates how this is implemented:

```
var nav = new NavigationPage (new HomePage ());
MainPage = nav;
```

The following code drives the user to the NoteItemListX page by pushing it in the top of the stack

```
Navigation.PushAsync(new NoteItemListX());
```

The active page can be popped from the navigation stack by pressing the Back button on the Android device or the on-screen button in the iOS device. The iOS devices does not have physical button for returning on the previous page. The following code drives the user to the previous page by popping the active page from the stack

```
this.Navigation.PopAsync();
```

6.4 Dependency Service

The Xamarin.Forms allows access to the native features of the iOS and Android from the Portable Class Library. It includes a Dependency Service to allow shared code to easily resolve Interfaces to platform-specific implementations (Hermes, 2015). When the native functionality of each platform is essential, an Interface and the Dependency Services will load the right implementation.

The Dependency Service involves three parts: The first part is the Interface which defines the functionality outside, the definition of the Interface is taking place in the shared code. The second part is the registration of the previous Interface, it is the implementation of the functionality per platform. The third part is of the calling of the previous implementation by using DependencyService.Get.

For the implementation of the Accessibility function and the Camera functionality the Dependency Service used, the following code illustrates the implementation of the Words-to-Talk functionality.

Step 1 – Interface

```
public interface IWordToTalk
    {
        void Speak (string text);
    }
```

Step 2 – Registration (The implementation of the code per platform)

```
[assembly: Dependency (typeof (WordToTalk _iOS))]

namespace Mnimi
{
    public class WordToTalk _iOS : IWordToTalk
    {

        float volume = 0.6f;
        float pitch = 1.1f;

        public void Speak (string text)
        {
            var speechSynthesizer = new AVSpeechSynthesizer ();

            var speechUtterance = new AVSpeechUtterance (text) {
                Rate = AVSpeechUtterance.MaximumSpeechRate/4,
                Voice = AVSpeechSynthesisVoice.FromLanguage ("en-US"),
                Volume = volume,
                PitchMultiplier = pitch
            };

            speechSynthesizer.SpeakUtterance (speechUtterance);
        }
    }
}
```

With the annotation on the top of the class, the DependencyService knows that this is the implementation for the iOS platform. The AVSpeechSynthesizer is a precise type for the iOS.

Step 3 – Calling

The XAML code for the button with the text "Speak", when the user press the button, the method `speakClicked` is called. In this method the DependencyService will trace the exact implementation depending on the platform and the user will listen the note from his mobile device.

```
<Button Text="Speak" Clicked="speakClicked"/>
```

```
void speakClicked(object sender, EventArgs e)
    {
        var noteItem = (NoteItem)BindingContext;
        DependencyService.Get<IWordToTalk>().Speak(noteItem.Notes);
    }
```

6.5 MyNotes & MyTasks

For the MyNotes part of the application, two XAML files are responsible to illustrate the list of the Notes and the details of each Note the user selects. The NoteItemList.xaml file illustrates all the NotesItems. The functionality is separated from the view and the functionality code is in the NoteItemList.xaml.cs file.

At the appearing of the page in the mobile device, the method OnAppearing() requests from the database the NoteItems and creates a list of the titles of them. This method does not have direct access to database. The request is taking place throw the App.cs class.

```
protected override void OnAppearing()
    {
        base.OnAppearing();
        listView.ItemsSource = App.Database.GetItems2();
    }
```

This list is represented. This page has also a toolbar with a plus (+) icon for the adding of a new Note. If the user will select a note or press the plus (+) icon then the NoteItemPage.xaml appears. The previous section describes the page navigation in details.

```
tbi = new ToolbarItem("+", null, () =>
                {
                    var noteItem = new NoteItem();
                    var notePage = new NoteItemPageX();
                    notePage.BindingContext = noteItem;
```

```
                Navigation.PushAsync(notePage);
            }, 0, 0);
```

The NoteItemPage retrieve the data from the database of the application using Data Binding and has no direct access to the database. The chapter 5.7 describes in details the Data Binding and because of the usage of the MVVM, the views, which are the pages, does not have direct access to the data.

The NoteItemPage has also buttons to Save, to Delete, and to Speak. The representation of the buttons is in the .xaml file and the functionality is in the .cs file. For example, the Save button has the following XAML code for the representation and the CS code for the functionality.

The XAML code for the button:

```
<Button Text="Save" Clicked="saveClicked"/>
```

The C# code for the method how save a noteItem to database:

```
void saveClicked(object sender, EventArgs e)
    {
        var noteItem = (NoteItem)BindingContext;
        App.Database.SaveItem2(noteItem);
        this.Navigation.PopAsync();
    }
```

The Speak button uses the DependencyService functionality. The code for every platform is not the same. The previous chapter describes this technic in details.

The MyTask part is also consist of two .xaml files for the illustration of all the Tasks and the illustrations of a specific Task or for the adding of a new Task. The functionality is also in two .cs files and none of the files has direct access to the database.

6.6 MyAddress

Accessing GPS services from a Portable Class Library is not easy. Each platform implements GPS services differently, so dependency service would typically be required. Additionally, the implementation between the platforms is very different and complex (Boggan, 2015).

Xamarin.Forms supports libraries that expose a set of APIs for taking in advantage the common device functionality across the platforms. These libraries are called plugins and for the implementation of MyAddress the Geolocator plugin used. Furthermore, for taking advantage of the GPS sensor of the devices, the permission of each platform is needed.

The following method uses the Geolocator plugin in order to find the current Latitude and Longitude of the device. In additional, the possibleAddresses keeps the specific name of the address. That name will be used later in order the user to receive direction to go back to that address.

```
 public async void FindLocationClicked(object sender, EventArgs e)
        {
           geoCoder = new Geocoder();

           var test = await CrossGeolocator.Current.GetPositionAsync(300000);

           var fortMasonPosition = new Position(test.Latitude,test.Longitude);
           var possibleAddresses = await geoCoder.GetAddressesForPositionAsync
(fortMasonPosition);

            foreach (var address in possibleAddresses)
                Address.Text += address+ " " ;

        }
```

The user takes direction to a saved address with the following code. In case the device has Android OS, the Google maps is giving directions and in case the device has iOS the Apple maps is giving directions.

```
              switch (Device.OS)
            {
                case TargetPlatform.iOS:
                    Device.OpenUri(
                      new Uri(string.Format("http://maps.apple.com/?q=
{0}", WebUtility.UrlEncode(address))));
                    break;
                case TargetPlatform.Android:
                    Device.OpenUri(
                      new Uri(string.Format("geo:0,0?q={0}", WebUtilit
y.UrlEncode(address))));
                    break;
            }
```

6.7 MyPhotos

The implementation of the MyPhotos part of the application is based on Dependency Service because the native functionality of each platform is necessary. The section 6.4 explain in detail how the Dependency Service works.

The Portable Class Library of the application has the definitions of the following:

The asynchronous interface.

```
  public interface ICameraProvider
  {
```

```
        Task<CameraResult> TakePictureAsync();
    }
```

The code for the result of the camera (the picture) and the path where the picture is saved is following. It was mentioned in the section 6.1 that the pictures will be saved in the file-system of the device.

```
    public class CameraResult
    {
        public ImageSource Picture { get; set; }

        public string FilePath { get; set; }
    }
```

The code for the usage of the camera is the following. The "await" keyword is because the interface of the camera is asynchronous.

```
var photoResult = await cameraProvider.TakePictureAsync();
Picture = photoResult.Picture;
```

In each specific platform project, the interface has to be implemented and has also to be registered in the Xamarin Dependency Service. The idea for the asynchronous interface is the usage of the native camera functionality of each platform and the returning of a task back to the portable library. The code for the Android platform is the following:

```
public Task<CameraResult> TakePictureAsync()
        {
            Intent intent = new Intent(MediaStore.ActionImageCapture);

            pictureDirectory = new File(Environment. GetExternalStoragePublicD
irectory(Environment.DirectoryPictures), "Mnimi");

            if (!pictureDirectory.Exists())
            {
                pictureDirectory.Mkdirs();
            }

            file = new File(pictureDirectory, String.Format("Mnimi_{0}.jpg", G
uid.NewGuid()));

            intent.PutExtra(MediaStore.ExtraOutput, Uri.FromFile(file));

            var activity = (Activity)Forms.Context;
            activity.StartActivityForResult(intent, 0);

            tcs = new TaskCompletionSource<CameraResult>();

            return tcs.Task;
        }
```

The TaskCompletionSource is for the representation of the async operation of a user interacting with the camera. Once the user's interaction is done, the task will be updated with the result. The class also creates a directory with the name "Mnimi", all the photos will be saved in this directory.

CHAPTER 7
EVALUATION

The evaluation method involves user interaction with the application. The study is following three steps. These steps are the preparation of the evaluation, the conduct of the evaluation and analysis of results.

The preparation phase is important for the success of the study and the quality of the results. This phase includes the familiarity of the users with the two mobile devices and with the different mobile OS. The first mobile device runs Android OS and the second runs iOS. This step is done before conducting any study. Also the limitation of the problems caused by the OS, regardless of the application being evaluated is important.

At the conduct of the evaluation, the users will use the same prototype Cross-Platform application in an Android and in an iOS mobile devices. The users will be asked to interact with the application and to complete same tasks in each mobile device. After that, the users will be asked to complete a questionnaire and to provide the experience they received from the application among the different mobile OS.

The data from studies with real users is necessary to be analysed in order to lead to conclusions about the UX and the usability of the mobile application between the two mobile OS, the Android and the iOS.

7.1 Precondition state

The application had be installed in both of the mobile devices, in the iPhone and in the Android device. Both of the devices had already enable the GPS sensor and the level of the speaker was high. Furthermore, none other application was running in the same time in the background, the battery level of the devices was over of 50% and the brightness of the screen in highest level. The application had no Tasks and no Notes. Only one Location with the name "Maria's house" was saved. The participants were left alone to complete the tasks while they were encouraged to think aloud. The observer was close to participants.

7.2 Procedure

Each participant was retold that he has been asked to take part in a usability test about a mobile application in two mobile devices. The test is involved of three stages and the participants could withdraw their participation at any stage. An informed consent form

received and signed from each participant and the first questionnaire was given and was filled. The next step was the performing of the tasks in two mobile OS and the last step was the answering of the last questionnaire.

7.3 Results of the Evaluation

The feedback received from the participants of the evaluation provided all the necessary information for the evaluation. This section analyses the results from the research methods adopted. For the purposes of this study, 11 participants answered two in paper questionnaires. The two questionnaires, the results of the tasks and the answers of the questionnaires can be found in the Appendix.

The pre-task questionnaire with the demographic characteristics indicates the following: The 64% of the participants were Male and the rest 36% were Female. 46% of the participants were at the age group of 31-40, 36% were under 30 years old and the rest 18% were over 40 years old. 63% of the participants were well educated, they have at least a Bachelor degree, the education level of the participants is considered high. All of the participants are using mobile devices like smartphones or tablets and the majority of them, the 55%, are using a mobile device with Android OS, 27% are using Apple iOS and the rest 18% are using other mobile OS such as Windows Phone and BlackBerry OS. In the question 6, if they have change to another mobile OS than their previous, 45% of the participants had changed and 40% of them had as reason a mobile application for that change. 73% of the participants are using the same mobile application in more than one mobile OS and 62% of them believe that this application is not the same among these mobile OS.

All the participants were invited to perform the following 11 tasks on two mobile devices, one mobile device with Android and one mobile device with iOS. Each of the following tasks done once in the iPhone and once in the Android device. The order is random in order the users not to be more familiar with the one of the two mobile OS. For example the Task 1 firstly done in the Android device and after in the iOS device, the Task 3 firstly in iOS device and secondly in Android device. The average time for the performing of the previous tasks by the participants was 166.6sec in the Android OS and 163.2 sec in the iOS, the difference between the time was 2.12%.

The post-task questionnaire with the questions about the application indicates the following: The majority of the users, 82% of them, managed to finish all the tasks without any problem in both of the devices. 18% of them did not manage to accomplish the Task 7 which has to do with the GPS of the device. The observer noticed that the devices in two cases did not manage to find the location of the user in an accepted time. The observer assumed that this is not a problem of the application but has to do with the GPS sensor of the device as also as with the weather condition and the Wi-Fi network. Moving on, 82% of the participants found the application very similar in both of the devices. Additionally, 73% of the participants found easy the accomplishment of the unique task in both of the devices. 70% of the participants who were Android users in the past think that that the general style

of the Android version of the application (the buttons, the pickers and the camera) is very close to other Android applications. The same for the iOS version of the application, 71% of the participants who were iOS users in the past think that the general style of the iOS version of the application is very close to other iOS applications. Finally, the 73% of the participants thinks that the application is equal in both of the platforms.

CHAPTER 8
CONCLUSION

This chapter gives the final conclusion of this thesis, the achievements and also recommendations about future work by the author. This thesis began with gathering information about the mobile platforms. The researcher then designed a cross-platform mobile application which is able to be installed in Android and iOS mobile devices. The implementation of the application was the next step and the evaluation of the application was the last one.

8.1 Recommendations

The mobile developers should adapt the cross-platform development process in order to build applications for the major mobile platforms. Using this method they can have as result high quality mobile applications with all the feature the mobile devices provide in less time and less cost than the traditional per-platform development process. Furthermore, the Xamarin is a powerful framework that can offer cross-platform applications which are providing the same features and the same UX in Android and iOS mobile devices.

The following can be considered as future work. According to the number of the users, the Windows Phone is the third mobile platform following the Android and the iOS (GSMA, 2016). The Xamarin.Forms supports this platform and the application, with changes, could be able to run in Windows Phone mobile devices. The application may support some more features. For example, one feature that the system is missing is the ability the users to delete the photos directly from the application. One other feature which could be interesting is the access to the microphone of the device, the users could record voice-notes. Finally, the data of the application is stored locally in the device, this data could be stored on-line in a cloud database. A user could have an account and signing-in to this account, the application could retrieve the data from the cloud and illustrate it to the current device.

8.2 Conclusion

The main problem in mobile industry is the fragmentation of the platforms (Delia, et al., 2015). There are two major mobile OS in mobile industry: the Android and the iOS. The Cross-Platform mobile development may reduce this fragmentation in terms of the application each platform provides. The Xamarin is framework for Cross-Platform mobile

development which enables the developers to build applications for Android and iOS using only C#.

According to analysis of the results, the user feedback was very positive. In details, the key points of the study was the following: the problem exists, a notable number of people tend to change mobile devices and mobile OS, in addition, a significant number of people uses the same mobile application in more than one mobile OS. The participants managed to perform the tasks in both of the platforms spending almost the same time, the difference was only 2.12%. This can points that, a Cross-Platform mobile application developed using Xamarin.Forms is able to provide the same UX and the same functionality between Android and iOS mobile OS.

The project was manageable within the program, and all essential knowledge and skills like the C# programming skills, were available to develop the development part of the project. The regular communication with the supervisor was very helpful in order the thesis to be finished in time. As the responses were mainly positive, the project can be considered effective, however some changes and improvements may be essential.

REFERENCES

Adobe, 2016. *Adobe PhoneGap Build.* [Online]
Available at: https://build.phonegap.com
[Accessed 15 March 2016].

Anderson, E. & Hall, S., 2009. Operating systems for mobile computing. *Journal of Computing Sciences in Colleges,* 25(2), pp. 64-71.

Antutu, 2016. *Ranking - AnTuTu Benchmark.* [Online]
Available at: http://www.antutu.com/en/Ranking.shtml
[Accessed 4 March 2016].

Appcelerator Inc, 2016. *Mobile App Development Platform & MBaaS | Appcelerator.* [Online]
Available at: http://www.appcelerator.com
[Accessed 15 March 2016].

Apple Inc, 2016. *App store Support - Apple Developer.* [Online]
Available at: https://developer.apple.com/support/app-store/
[Accessed 15 March 2016].

Apple, 2016. *iOS 9 - Apple.* [Online]
Available at: http://www.apple.com/uk/ios/?cid=wwa-uk-kwg-features

Apple, 2016. *iTunes - Browse the top apps on the App Store - Apple.* [Online]
Available at: http://www.apple.com/itunes/charts/free-apps/
[Accessed 2 March 2016].

Apple, 2016. *Mac Developer Library - Kernel and Device Drivers Layer.* [Online]
Available at:
https://developer.apple.com/library/mac/documentation/MacOSX/Conceptual/OSX_Technology_Overview/SystemTechnology/SystemTechnology.html
[Accessed 20 February 2016].

Apple, 2016. *Submitting Your App to the Store - Apple.* [Online]
Available at:
https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/SubmittingYourApp/SubmittingYourApp.html

Apple, 2016. *Xcode - Downloads - Apple Developer.* [Online]
Available at: https://developer.apple.com/xcode/
[Accessed 1 March 2016].

Bilgin , C., 2016. *Mastering Cross-Platform Development with Xamarin.* 1st Edition ed. s.l.:Packt Publishing.

Boggan, P., 2015. *blog.xamarin.com.* [Online]
Available at: https://blog.xamarin.com/geolocation-for-ios-android-and-windows-made-

easy/
[Accessed 2 7 2016].

Bouras, C., Papazois, A. & Stasinos, N., 2014. *A Framework for Cross-platform Mobile Web Applications Using HTML5.* Barcelona,, IEEEE, pp. 420-424 .

Boushehrinejadmoradi, N., Ganapathy, V., Nagarakatte, S. & Iftode, L., 2015. *Testing Cross-Platform Mobile App Development Frameworks.* Lincoln, NE, IEEE.

Brooks, I., 2014. *The Importance of User Experience.* First ed. s.l.:Amazon Media EU.

Burnette, E., 2015. *Hello, Android: Introducing Google's Mobile Development Platform.* Frisco: Pragmatic Bookshelf.

Capek, P., Kral, E. & Senkerik, R., 2015. *Towards an Empirical Analysis of .NET Framework and C# Language Features' Adoption.* Las Vegas, IEEE, pp. 865 - 866.

Charkaoui, S., Adraoui, Z. & Benlahmar, E. H., 2014. *Cross-platform mobile development approaches.* Tetouan, IEEE.

Charkaoui, S., Adraoui, Z. & Benlahmar, E. H., 2014. *Cross-platform mobile development approaches.* Tetouan, IEEE, pp. 188 - 191.

Chen , T., Shi, Q., Lou, X. & Hu, W., 2010. *A case study of course design for software development on mobile phone.* Qingdao, IEEE, pp. 59 - 64.

Dalmasso, I., Datta, S. K., Bonnet, C. & Nikaein, N., 2013. *Comparison and Evaluation of Cross Platform Mobile Application Development Tools.* Cagliari, IEEE, p. 1.

Delia, L. et al., 2015. *Multi-Platform Mobile Application Development Analysis.* Athens, IEEE, pp. 181 - 186.

Deng, L., 2015. *Towards mutation analysis of Android apps.* Graz, IEEE.

Dumas, J. S., 2003. *The Human-Computer Interaction Handbook: Fundamentals.* s.l.:CRC Press.

European Commission, 2014. *Web Accessibility.* [Online]
Available at: http://ec.europa.eu/ipg/standards/accessibility/index_en.htm
[Accessed 12 7 2016].

Evernote Corporation, 2016. *The note-taking space for your life's work | Evernote.* [Online]
Available at: http://www.Evernote.com
[Accessed 29 6 2016].

Feiler, J., 2015. *Introducing SQLite for Mobile Developers.* First ed. s.l.:Apress.

Feiler, J., 2015. *Introducing SQLite for Mobile Developers.* 1st ed. s.l.:Apress.

Ghayas, S., 2013. *Qualitative study to identify icons characteristics on mobile phones applications interfaces.* Kuching, IEEE, pp. 310 - 315.

Google, 2016. *Android Apps on Play Store.* [Online]
Available at: https://play.google.com/store/apps?hl=en_GB
[Accessed 1 March 2016].

Google, 2016. *Android Interfaces and Architecture.* [Online]
Available at: https://source.android.com/devices/
[Accessed 25 February 2016].

Google, 2016. *Android Open Source Project.* [Online]
Available at: https://source.android.com/
[Accessed 2 February 2016].

Google, 2016. *Download Android Studio and SDK Tools.* [Online]
Available at: http://developer.android.com/sdk/index.html
[Accessed 3 March 2016].

GSMA, 2016. *GSMA Mobile Economy 2016.* [Online]
Available at: http://www.gsmamobileeconomy.com/
[Accessed 3 March 2016].

Hermes, D., 2015. *Xamarin Mobile Application Development: Cross-Platform C# and Xamarin.Forms Fundamentals.* 1st ed. s.l.:Apress.

IBM, 2012. *Native, web or hybrid mobile-app development,* s.l.: IBM.

IDC, 2016. *IDC: Smartphone OS Market Share.* [Online]
Available at: http://www.idc.com/prodserv/smartphone-os-market-share.jsp
[Accessed 1 March 2016].

Ji, Z., Zhang, X., Ganchev, I. & ODroma, M., 2013. *Development of a Sencha-Touch mTest Mobile App for a mLearning System.* Beijing, IEEE, pp. 210 - 211.

Joorabchi, M. E., Kruchten, P. & Mesbah , A., 2013. *Real Challenges in Mobile App Development.* Baltimore, MD, IEEE.

Joyent, I., 2016. *Node.js.* [Online]
Available at: https://nodejs.org/en/
[Accessed 1 March 2016].

Khandeparkar, A., Gupta, R. & Sindhya, B., 2015. An Introduction to Hybrid Platform Mobile Application Development. *International Journal of Computer Applications,* 118(15).

Larman, C. & Victor, B., 2003. Iterative and Incremental Development: A Brief History. *Computer,* 23(9), pp. 47-56.

Lin, H.-C. & Lee, G., 2015. *Building a Secure Cross Platform Enterprise Service Mobile Apps Using HTML5.* Taipei, IEEE, pp. 162-166.

Liu, F., Xiao, M. & Feng, W., 2015. *Design of Cordova Based Message Push Module for Cross-Platform Smart Home Application.* Qinhuangdao, IEEE, pp. 635 - 639.

MarketLine, 2015. *Mobile Phones Industry Profile: Global,* London: MarketLine.

McLaughlin, B., 2011. *What Is HTML5.* s.l.:O'Reilly Media.

McNish , J., 2015. *Losing the Signal.* 1st Edition ed. s.l.:Flatiron Books.

Microsoft Inc, 2016. *C# Programming Guide.* [Online]
Available at: https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx
[Accessed 15 March 2016].

Microsoft, 2016. *Concepts and architecture for Windows Phone 8.* [Online]
Available at: https://msdn.microsoft.com/en-us/library/windows/apps/ff967549%28v=vs.105%29.aspx
[Accessed 15 February 2016].

Microsoft, 2016. *Cross Platform Mobile Development.* [Online]
Available at: https://www.visualstudio.com/en-us/features/mobile-app-development-vs.aspx
[Accessed 2 March 2016].

Microsoft, 2016. *Cross Platform Mobile Development with Visual Studio.* [Online]
Available at: http://blogs.msdn.com/b/uk_faculty_connection/archive/2014/07/29/cross-platform-mobile-development-with-visual-studio.aspx
[Accessed 25 February 2016].

Microsoft, 2016. *Developer program FAQ - Windows app development.* [Online]
Available at: https://dev.windows.com/en-us/programs/faq

Microsoft, 2016. *Free Dev Tools - Visual Studio Community 2015.* [Online]
Available at: https://www.visualstudio.com/products/visual-studio-community-vs
[Accessed 4 February 2016].

Microsoft, 2016. *Microsoft OneNote | The digital note-taking app for your devices.* [Online]
Available at: www.onenote.com
[Accessed 29 6 2016].

Microsoft, 2016. *Personalising your Start screen and theme colour on Windows Phone | Windows Phone.* [Online]
Available at: http://www.windowsphone.com/en-gb/how-to/wp8/settings-and-personalization/personalize-my-start-screen-and-theme-color

Microsoft, 2016. *Visual Studio - Microsoft Developer Tools.* [Online]
Available at: https://www.visualstudio.com/
[Accessed 29 February 2016].

Microsoft, 2016. *Windows Phone 8.1 for Developers–Converging the App Models-Dr IInspectable & Mr Native.* [Online]
Available at: http://blogs.msdn.com/b/thunbrynt/archive/2014/03/31/windows-phone-8-1-for-developers-converging-the-app-models.aspx
[Accessed 20 February 2016].

National Bank of Greece, 2016. *Mobile Banking - NBG.* [Online]
Available at: https://www.nbg.gr/el/i-bank/retail/mobile-banking
[Accessed 10 March 2016].

Neuburg , M., 2013. *iOS 7 Programming Fundamentals: Objective-C, Xcode, and Cocoa Basics.* 1st Edition ed. Sebastopol: O'Reilly Media.

Nielsen, J. & Landauer, T., 1993. *A mathematical model of the finding of usability problems.* Amsterdam, s.n., pp. 206-213.

Nielsen, J. & Levy, J., 1994. Measuring Usability — Preference vs. Performance. *Communications of the ACM,* April, 37(4), pp. 66-75 .

Ohrt, J. & Turau, V., 2012. Cross-Platform Development Tools for Smartphone Applications. *Computer,* 45(9), pp. 72 - 79.

Perchata, J., Desertot, M. & Lecomte, S., n.d. Component Based Framework to Create Mobile Cross-platform Applications. *Procedia Computer Science,* Issue 19, p. 1004 – 1011.

Pon, B., Seppala, T. & Kenney, M., 2014. Android and the demise of operating system-based power:Firm strategy and platform control in the post-PC world. *Telecommunications Policy,* 38(11), p. 979–991.

Powers , L. & Snell, M., 2015. *Microsoft Visual Studio 2015 Unleashed.* 3rd Edition ed. s.l.:Sams.

Prey, J. & Weaver, A., 2014. Tablet PC Technology: The Next Generation. *IEEE Computer Society,* September.pp. 32-33.

Quinn Patton, M., 2015. *Qualitative Research & Evaluation Methods: Integrating Theory and Practice.* s.l.:SAGE Publications, Inc.

Ronkainen, J., Eskeli, J., Urhemaa, T. & Koskela-Huotari, K., 2013. *Experiences on Mobile Cross-Platform Application Development Using PhoneGap.* Venice, IARIA.

Rui, H., ZhiGang, J. & BaoLiang, W., 2013. *Comparison of Windows Phone 8 & Windows 8.* Shenyang, IEEE, pp. 55 - 57 .

Saunders, M., Thornhill, A. & Lewis, P., 2009. *Research Methods for Business Students.* 5th ed. s.l.:Pearson Education.

Sencha Inc, 2016. *Cross-platform Mobile Web App Development Framework for HTML5 and JS | Sencha.* [Online]
Available at: https://www.sencha.com/products/touch/#overview
[Accessed 15 March 2016].

Serrano, N., Hernantes, J. & Gallardo, G., 2013. Mobile Web Apps. *Software,* 30(5), pp. 22 - 27.

Shneiderman, B., Plaisant , C., Cohen, M. & Jacobs , S., 2009. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* 5th ed. s.l.:Pearson.

Sin, D., Lawson, E. & Kannoorpatti, K., 2012. *Mobile Web Apps - The Non-programmer's Alternative to Native Applications.* Perth, IEEE, pp. 8 - 15.

Skeet, J., 2013. *C# in Depth.* 3rd Edition ed. s.l.:Manning Publications.

Smyth, N., 2011. *iPhone iOS4 Development Essentials - Xcode 4 Edition.* 1st Edition ed. Oxford: Payload Media.

Snider, E., 2016. *Mastering Xamarin.Forms.* 1st ed. s.l.:Packt Publishing.

Stefanov, S., 2010. *JavaScript Patterns.* s.l.:O'Reilly Media.

Sutherland, J., 2015. *Scrum: The Art of Doing Twice the Work in Half the Time.* s.l.:Random House Business.

Szostak, T., 2013. *Windows Phone 8 Application Development Essentials.* 1st Edition ed. Birmingham: Packt Publishing.

W3C, 2016. *HTML5.* [Online]
Available at: https://www.w3.org/TR/html5/
[Accessed 1 March 2016].

Wafaa, E.-K., Bassem, A., Ahmed, Y. & Ayman, W., 2015. Taxonomy of Cross-Platform Mobile Applications Development Approaches. *Ain Shams Engineering Journal,* pp. 1-28.

Want, R., 2014. Smartphones: Past, Present, and Future. *IEEE Pervasive Computing,* pp. 89-92.

WhatsApp Inc, 2016. *WhatsApp Download.* [Online]
Available at: http://www.whatsapp.com/download/
[Accessed 15 March 2016].

Xamarin, 2016. *Announcing Xamarin 3.* [Online]
Available at: https://blog.xamarin.com/announcing-xamarin-3/
[Accessed 23 February 2016].

Xamarin, 2016. *Mobile App Development & App Creation Software - Xamarin.* [Online]
Available at: https://xamarin.com/
[Accessed 1 March 2016].

Xanthopoulos, S. & Xinogalos, S., 2013. *A comparative analysis of cross-platform development approaches for mobile applications.* s.l., ACM, pp. 213-220 .

Zhu, J., Wu, Z., Guan, Z. & Chen, Z., 2015. *Appearance similarity evaluation for Android applications.* Wuyi, IEEE, pp. 323 - 328.

Appendix A : Project Planning

MSc Project Timetable

Task Name	Week	Duration(Days)	Start	Finish
Analyze Requirements & Design of the App	1	7	09/05/2016	13/05/2016
Development phase	1, 2, 3, 4, 5, 6, 7, 8, 9	53	14/05/2016	05/07/2016
Design User Study	9	5	06/07/2016	10/07/2016
Testing App	10	7	11/07/2016	17/07/2016
Write the Methodology Chapter	10	7	11/07/2016	17/07/2016
Evaluation phase	11, 12	10	18/07/2016	27/07/2016
Analyze Evaluation Data	12	4	28/07/2016	31/07/2016
Write the Discussion Chapter	13	7	01/08/2016	07/08/2016
Submit Draft of the thesis to Supervisor	14	1	08/08/2016	08/08/2016
Creating the Poster	14	5	08/08/2016	12/08/2016
Feedback from Supervisor	14	1	12/08/2016	12/08/2016
Submit Final thesis	15	1	18/08/2016	18/08/2016
Poster Session	16	1	25/08/2016	25/08/2016

Risk Management Plan

Risk	Explanation	Risk Level	Impact
Bad application design	The evaluation of the application will be in terms of the UX	Medium	Effects in the evaluation
Delay in learning of Xamarin and C#	The author has no experience in Xamarin and limited experience in C#	High	Delay in development phase
Small sample of user for the evaluation phase	More than 10 people will be used for the evaluation	Medium	Effects in the evaluation
Hardware problem	For this thesis needs to be used one Windows laptop, one Mac, one Android phone and one iPhone	High	Effects the development and the evaluation

Gantt chart

The figure 12 shows the Gantt chart with the timeline for this thesis.

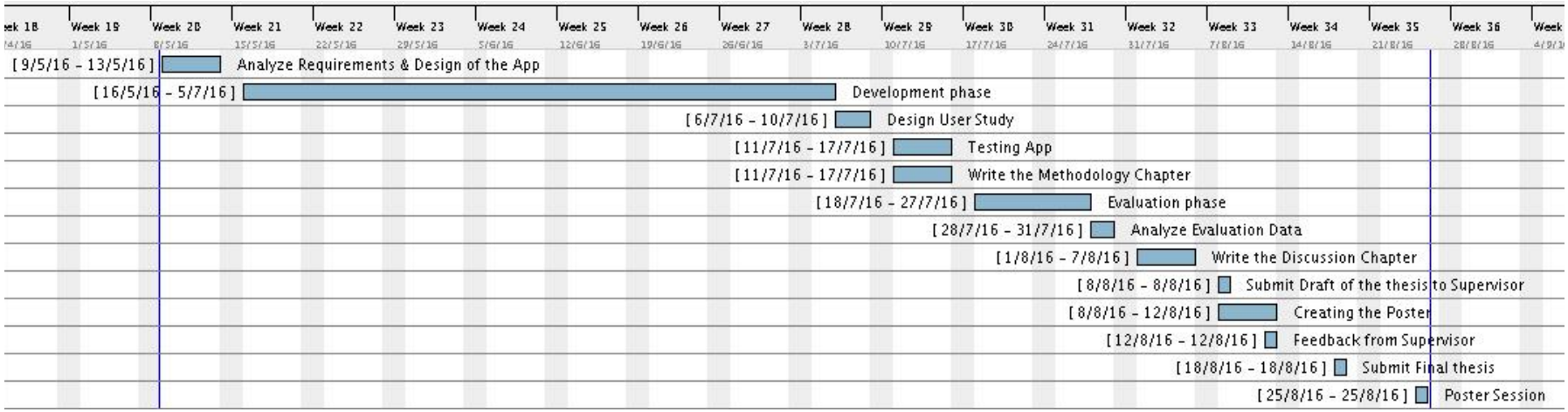
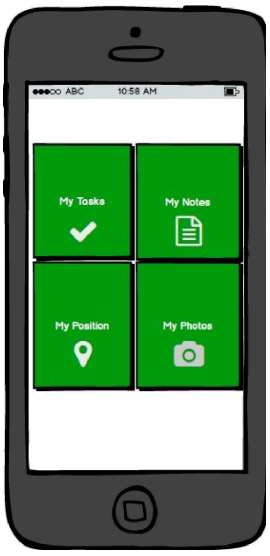


Figure 7 - Gantt Chart

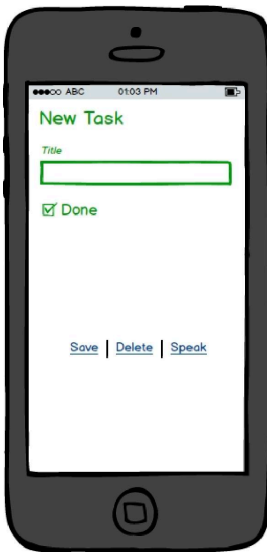
Appendix B : Mock-up

The initial Mock-up of the application.

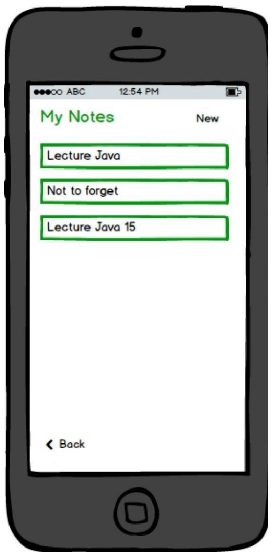


Home Screen of the Application.

"My Task" part. The button "New" is for a new task.

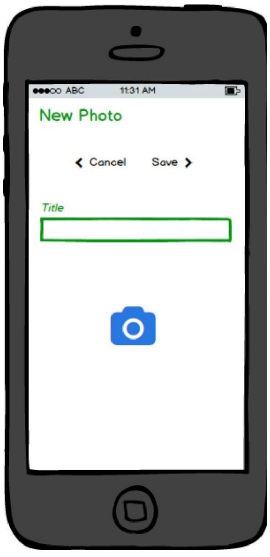


New Task screen or the screen of a saved task

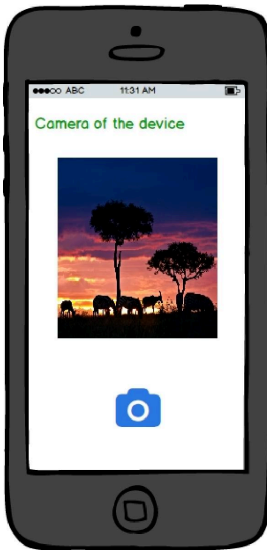


"My Notes" part. The button "New" is for a new note.

New Note screen or the screen of a saved Note.

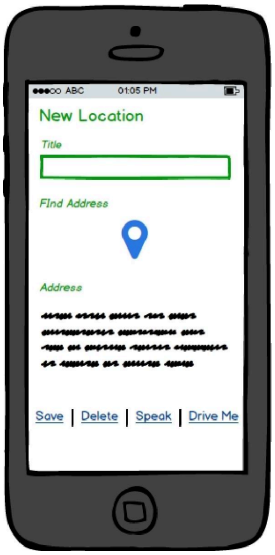


"My Photos" part. The blue button drives to camera

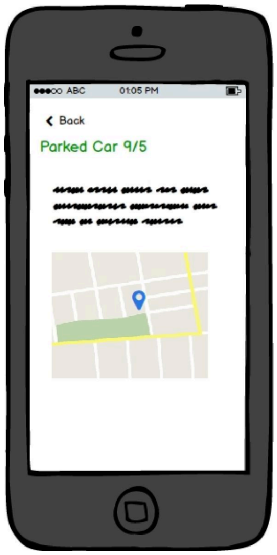


The screen with the camera of the device.

"My Location" part. The button "New" is for a new location

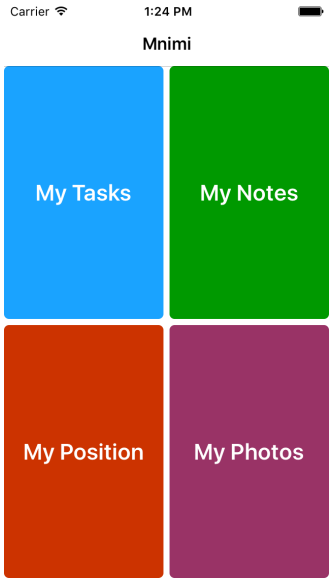


New Location screen
or the screen of a saved location.

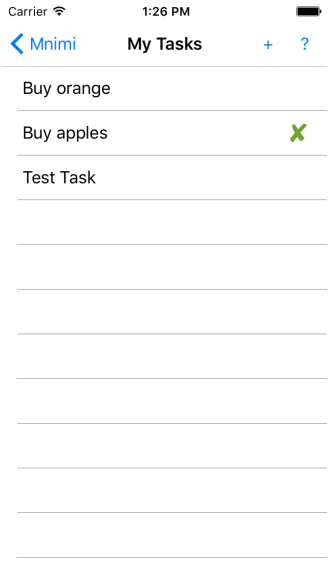


The "Drive me" screen. The user takes directions to a saved location

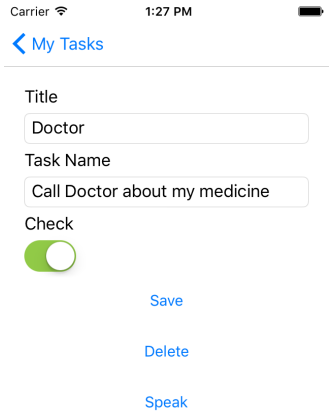
Screenshots from iOS version of the application.



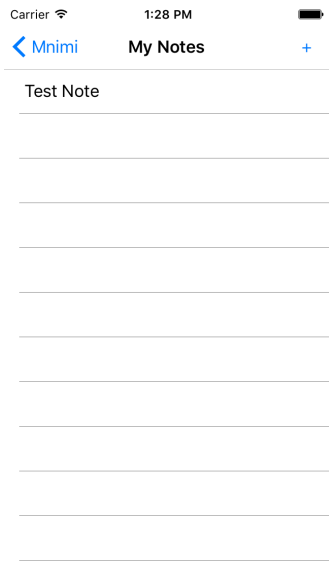
Home Screen of the Application.



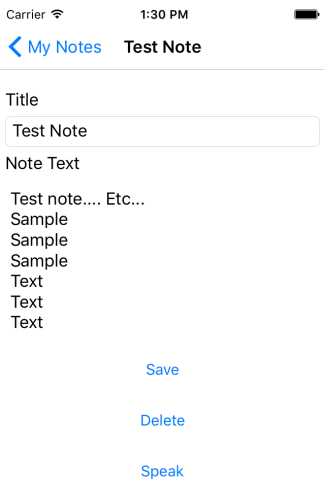
"My Task" part. The button "+" is for a new task. The button "?' reads the tasks have not done.



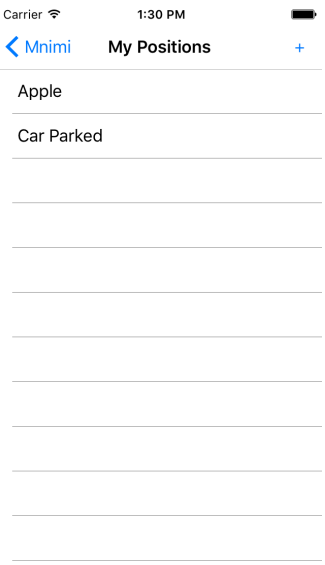
The screen of a task. The Check controller is for "done" or "not done" of a task



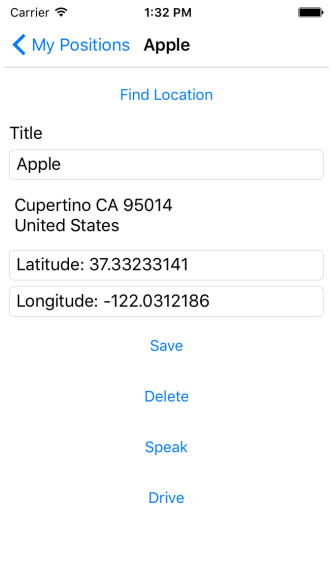
The list of the notes



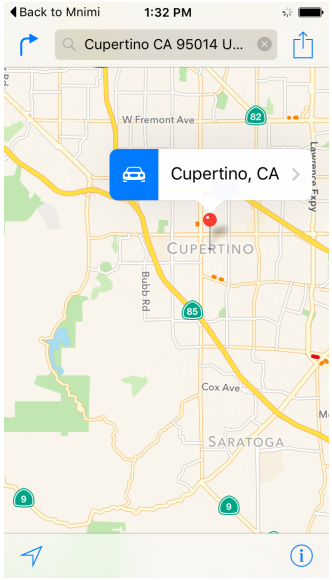
The screen of a note.



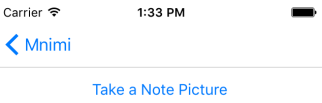
The list of the positions, the "+" button is also for a new position.



The screen of a position. The "Find Location" returns Latitude, Longitude and the name of the address. The "Drive" button drives the user to take direction in order to visit this address

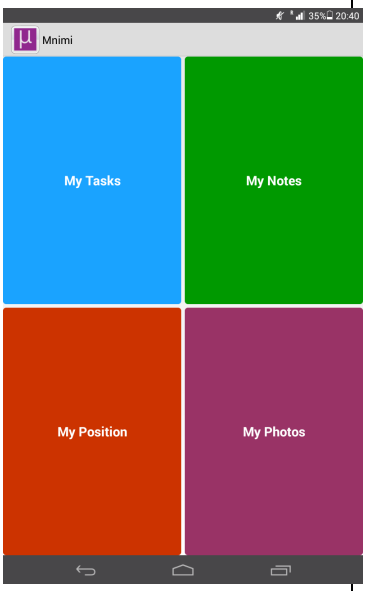


The screen with the Apple Maps where the user takes directions.

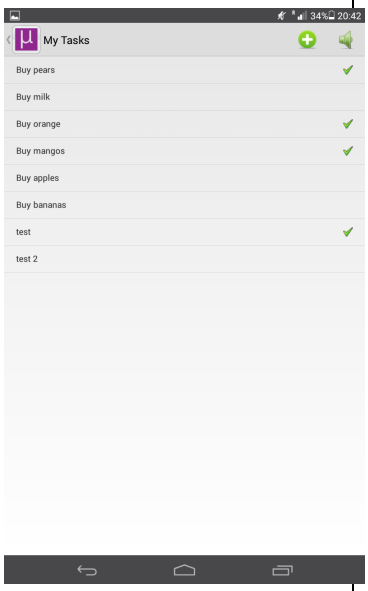


The camera screen, the button "Take a Note Picture" opens the camera of the device.

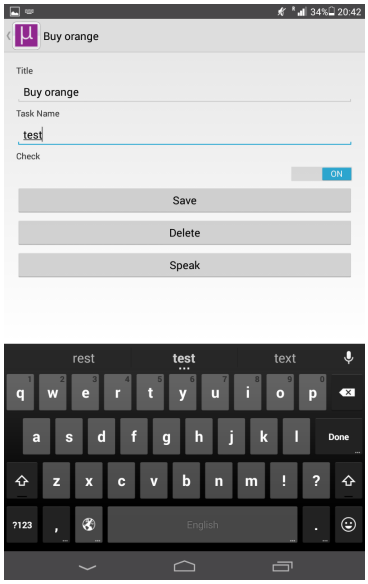
Screenshots from Android version of the application.



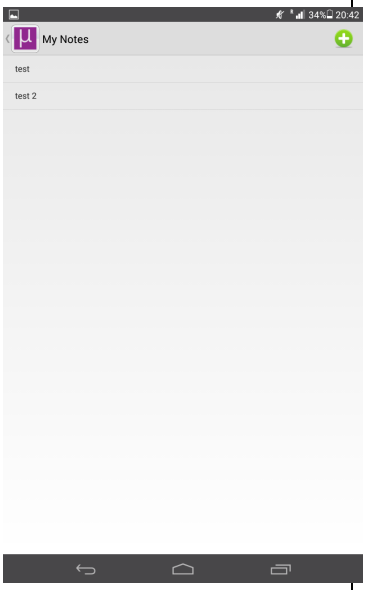
Home Screen of the Application.



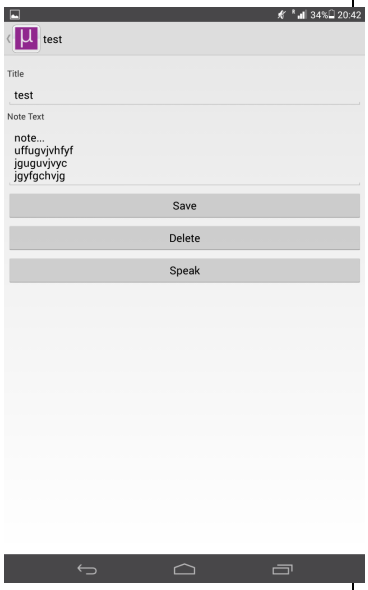
"My Task" part. The button "+" is for a new task. The button "?' reads the tasks have not done.



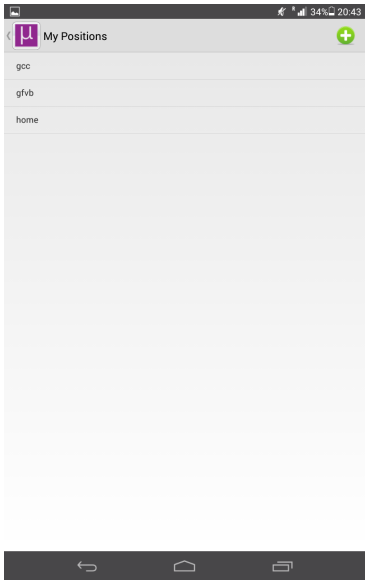
The screen of a task. The Check controller is for "done" or "not done" of a task



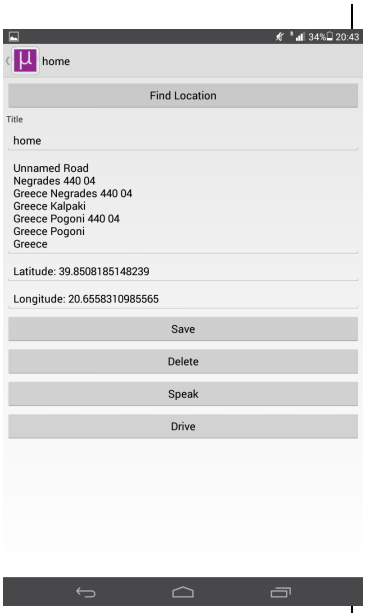
The list of the notes



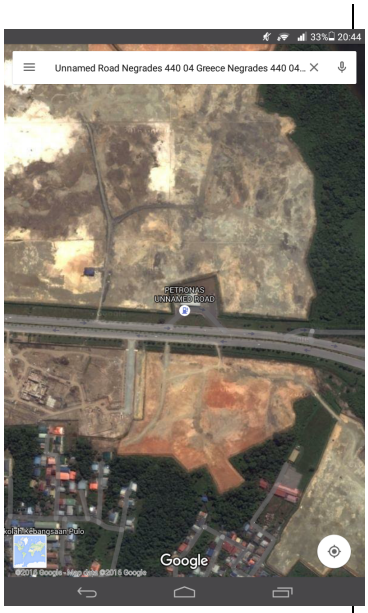
The screen of a note.



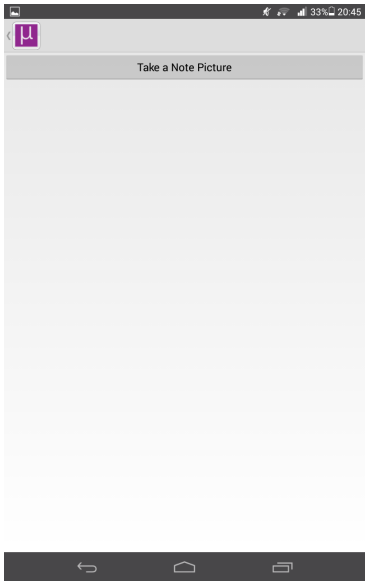
The list of the positions, the "+" button is also for a new position.



The screen of a position. The "Find Location" returns Latitude, Longitude and the name of the address. The "Drive" button drives the user to take direction in order to visit this address



The screen with the Google Maps where the user takes directions.



The camera screen, the button "Take a Note Picture" opens the camera of the device.

Appendix C : Questionnaires and Tasks

Introduction

Thank you for showing interest in this project. Please read the information carefully before deciding whether or not to participate. If you decide not to take part or withdraw your participation at any stage, there will be no disadvantages to you of any kind.

This project is being undertaken as part of the requirements for the Postgraduate degree of Software Engineering at Heriot Watt University. The purpose of this study is to extract the opinion of smartphone users concerning the same mobile application in two mobile platforms and investigate the usability issues that may experience.

Should you agree to take part in this project you will be asked to perform some tasks, on two mobile devices, one iPhone and one Android device and give some feedback about your thoughts on the tasks completed. Data will be collected by using a closed questionnaire. There will be a pre-test question part as well as a post-test question part. The tasks and the questionnaire should no take more than 20 minutes of your time.

Any information you supplied is confidential, no information that could identify the participant is collected. Thank you for your cooperation.

Pre-Task Questionnaire

1. **Do you identify as:**

Male

Female

2. **Which category includes your age?**

20 or younger

21-30

31-40

41-50

50 or more

3. **What is the highest degree or level of school you have completed?**

High School Graduate

Technical/Trade/Vocation Training

Bachelor's Degree

Master's Degree

Doctorate Degree

4. **Are you a smartphone or tablet user?**

Yes

No

5. **If you answered "Yes" in the previous question, what kind of smartphone or a tablet do you use as main device?**

iOS device (iPhone, iPad)

Android device

Windows phone

BlackBerry

Other

6. **Have you ever changed to another smartphone or tablet with different mobile OS from your previous smartphone or tablet?**

Yes

No

7. **If you answered "Yes" in the previous question, the reason of that change had to do with the apps of the mobile OS?**

Yes

No

8. **Do you use the same mobile app in more than one mobile OS?**

Yes

No

9. **If you answered "Yes" in the previous question, did you notice any difference in this app between the mobile OS?**

Yes

No

Tasks

1. Add a new task with title “Supermarket”.
   The participants need to add a new task
2. Add a new note with title “Vacations 2016” and main text “The hotel in Crete costs 55 euros per night”, save the task.
3. Set the task “Supermarket” as “Done” and save it
4. Do back and listen the note “Vacations 2016”.
5. Delete the Task “Supermarket”.
6. Modify the note “Vacations 2016” and set the cost of the hotel as “65 euros per night” and save it.
7. Find and save your current location giving the title “Car”.
8. Take directions in order to go to Maria’s house which is saved as location. Do not follow the directions.
9. Take a picture.
10. Delete the location “Car”
11. Listen where is the location “Maria’s house”.

Post-Task Questionnaire

Introduction question. How did the execution go? Was it pleasing?

1. **Did you manage to finish all the tasks in both of the devices without help?**

Yes

No

2. **If you answered "No" in the previous questions, in which task did you have a problem?**

1

2

3

4

5

6

7

8

9

10

11

3. **How similar is the app between the two mobile devices?**
   (1 means very different and 5 means exactly the same)

1

2

3

4

5

4. **How easy was for you to accomplish the same task in the different devices?**
   (1 means very difficult and 5 means very easy)

1

2

3

4

5

5. **Have you ever used an Android device in the past?**

Yes

No

6. **If you answered "Yes" in the previous question, how close is the general style of the app (the buttons, the pickers and the camera) to the apps you were using in the Android device.**
   (1 means very different and 5 means exactly the same)

1

2

3

4

5

7. **Have you ever used an iOS device in the past?**

Yes

No

8. **If you answered "Yes" in the previous question, how close is the general style of the app (the buttons, the pickers and the camera) to the apps you were using in the iOS device.**
   (1 means very different and 5 means exactly the same)

1

2

3

4

5

9. **Do you think that the app is better in one of the two platforms?**

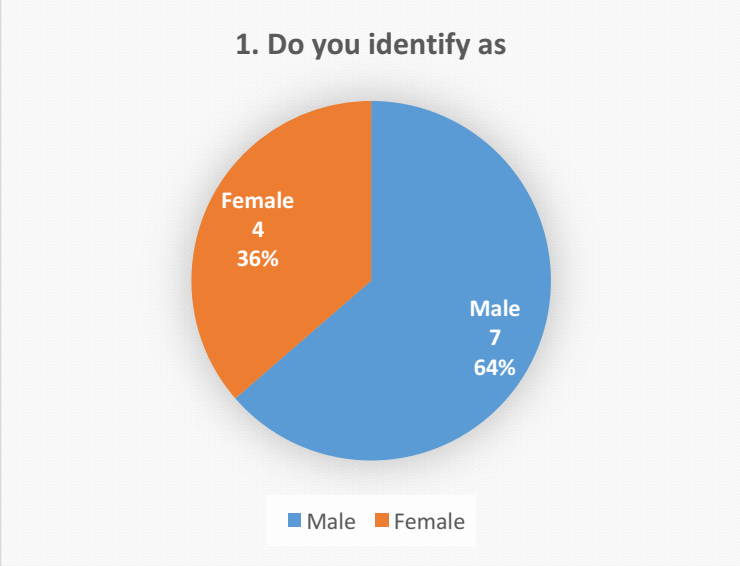
Yes

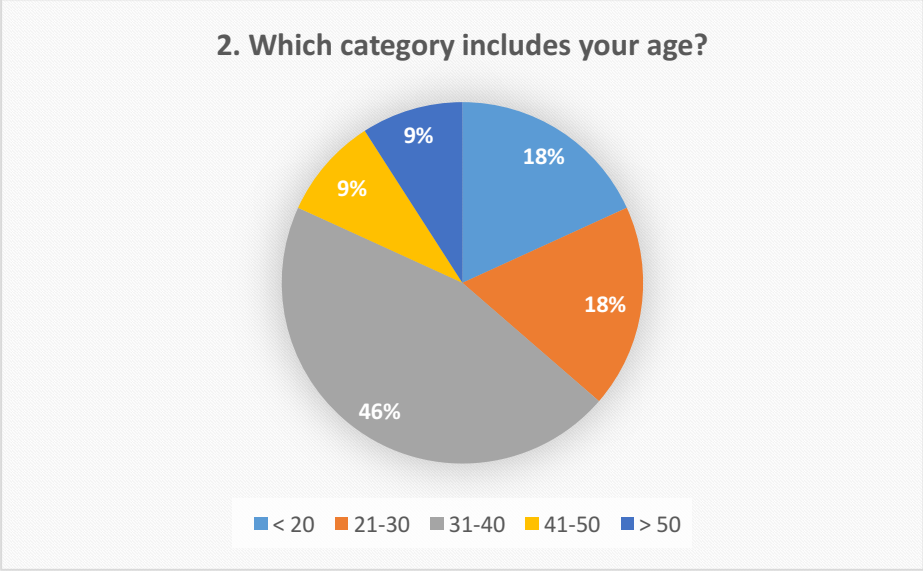
No

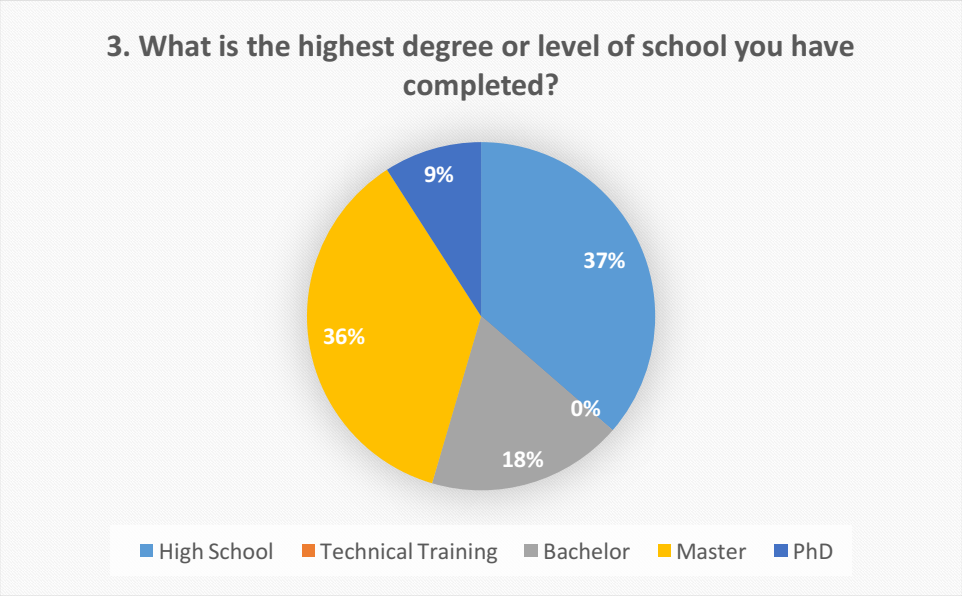
Appendix D : Results of the study

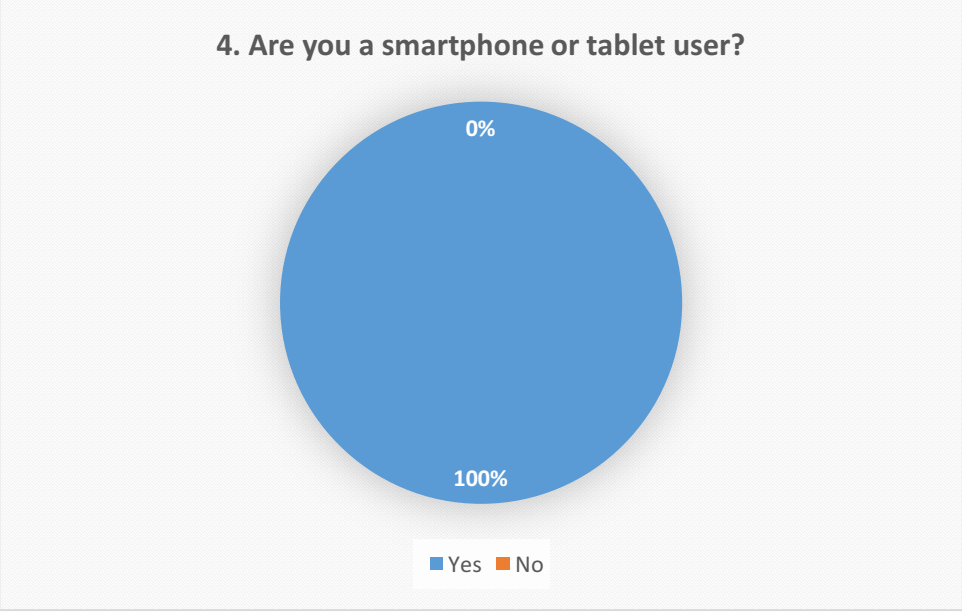
Pre – Task answers

Pre - Task Questions									
	Question 1	*Question 2*	*Question 3*	*Question 4*	*Question 5*	*Question 6*	*Question 7*	*Question 8*	*Question 9*
Participant 1	Famale	<20	High School	Yes	iOS	Yes	Yes	Yes	Yes
Participant 2	Female	31-40	Bachelor	Yes	iOS	No		Yes	Yes
Participant 3	Male	31-40	Master	Yes	Android	No		Yes	Yes
Participant 4	Male	31-40	PhD	Yes	Windows	Yes	No	Yes	No
Participant 5	Male	41-50	High School	Yes	iOS	Yes	No	No	
Participant 6	Male	<20	High School	Yes	Android	No		Yes	Yes
Participant 7	Male	21-30	Master	Yes	Android	No		Yes	Yes
Participant 8	Female	21-30	Master	Yes	Android	No		Yes	No
Participant 9	Male	31-40	Master	Yes	Android	Yes	Yes	Yes	No
Participant 10	Female	>50	High School	Yes	BlackBerry	Yes	No	No	
Participant 11	Male	31-40	Bachelor	Yes	Android	No		No	

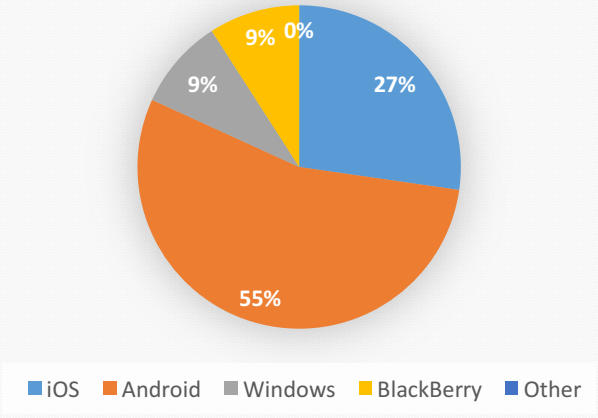



2. Which category includes your age?
9%
18%
9%
18%
46%
< 20
21-30
31-40
41-50
> 50

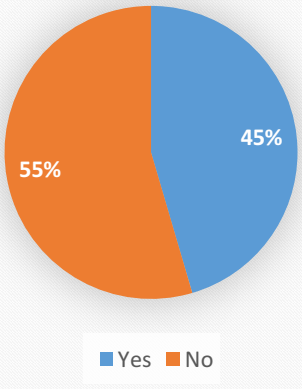

3. What is the highest degree or level of school you have completed?
9%
37%
36%
0%
18%
High School
Technical Training
Bachelor
Master
PhD


4. Are you a smartphone or tablet user?
0%
100%
Yes
No

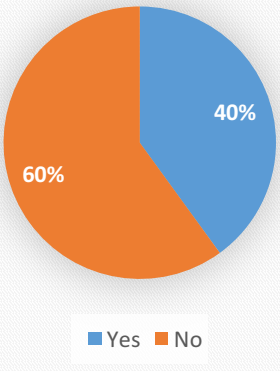
5. If you answered "Yes" in the previous question, what kind of smartphone or a tablet do you use as main device?

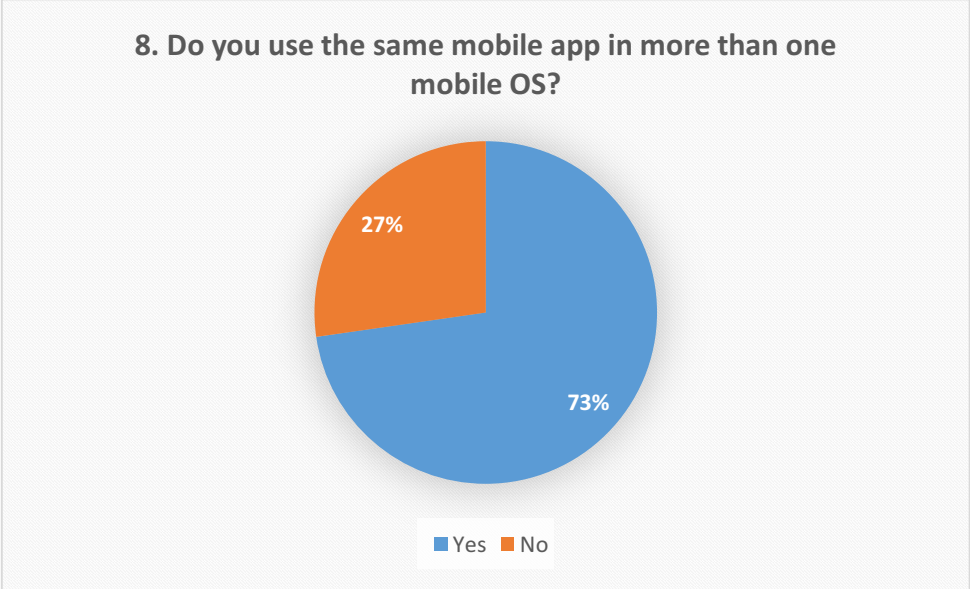


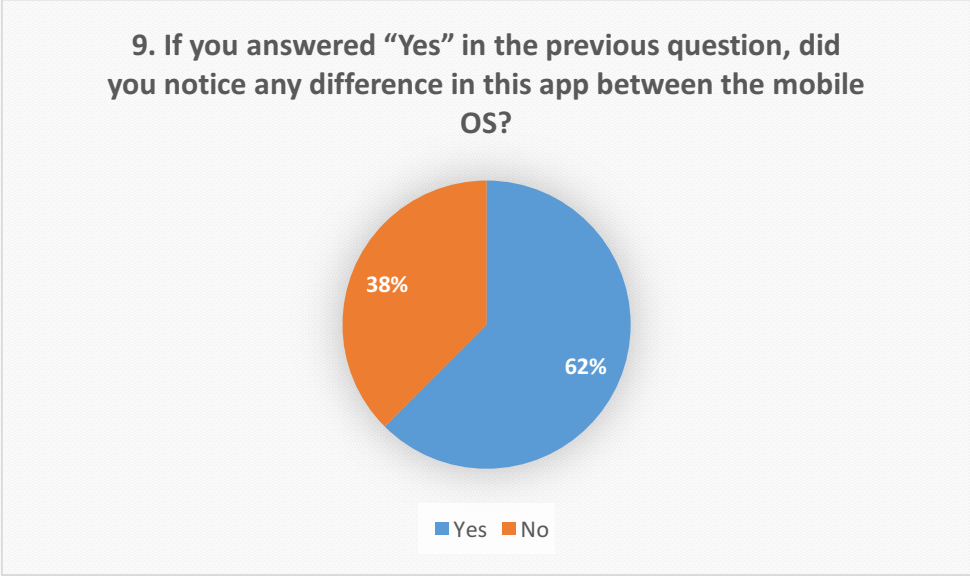
6. Have you ever change to another smartphone or tablet with different mobile OS than your previous smartphone or tablet?



7. If you answered "Yes" in the previous question, the reason of that change had to do with the apps of the mobile OS?




8. Do you use the same mobile app in more than one mobile OS?
27%
73%
Yes
No


9. If you answered “Yes” in the previous question, did you notice any difference in this app between the mobile OS?
38%
62%
Yes
No

Results of Tasks

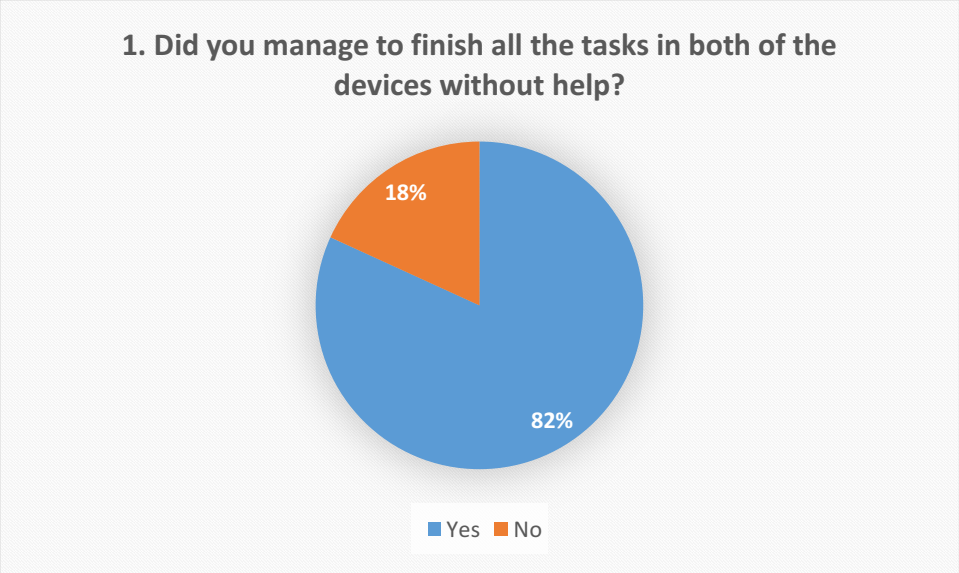
Time for each task in seconds											
Android											
	Task 1	*Task 2*	*Task 3*	*Task 4*	*Task 5*	*Task 6*	*Task 7*	*Task 8*	*Task 9*	*Task 10*	*Task 11*
Participant 1	13	16	11	15	9	14	21	14	7	18	11
Participant 2	12	17	8	18	11	15	22	16	8	19	14
Participant 3	11	21	9	13	13	13	25	17	9	11	16
Participant 4	14	23	12	14	13	13	24	19	8	10	19
Participant 5	13	25	11	15	15	13	21	15	8	15	12
Participant 6	16	22	9	16	12	15	30	15	8	15	14
Participant 7	18	18	12	14	14	18	32	19	5	13	17
Participant 8	12	16	15	17	13	21	22	20	5	10	14
Participant 9	15	21	10	20	14	14	28	12	11	20	13
articipant 10	12	24	10	14	10	15	36	15	7	18	18
articipant 11	19	21	12	14	8	12	24	13	4	14	19
verage Time	**14,1**	**20,4**	**10,8**	**15,5**	**12,0**	**14,8**	**25,9**	**15,9**	**7,3**	**14,8**	**15,2**

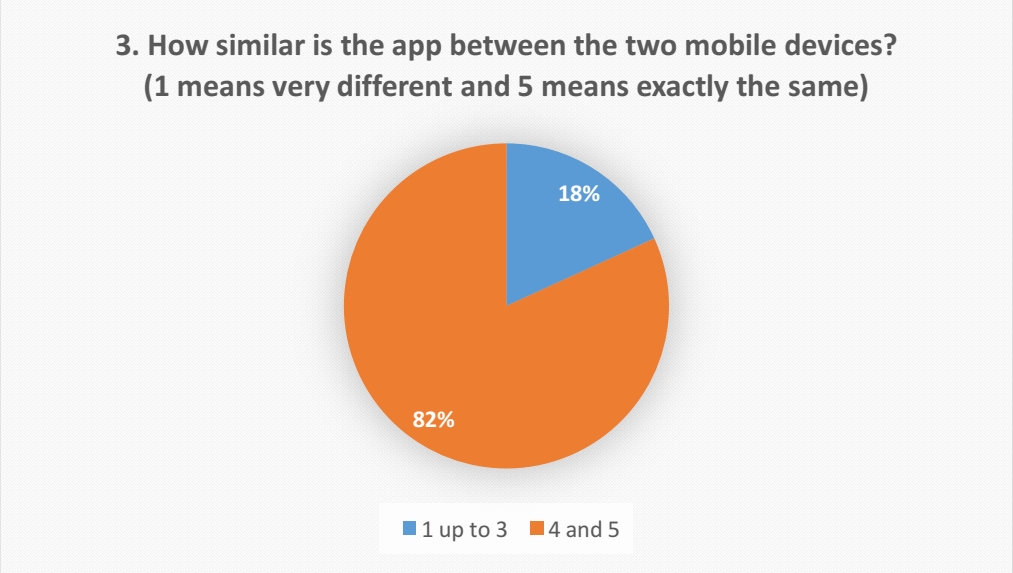
iOS											
	Task 1	*Task 2*	*Task 3*	*Task 4*	*Task 5*	*Task 6*	*Task 7*	*Task 8*	*Task 9*	*Task 10*	*Task 11*
Participant 1	11	18	8	18	14	12	27	18	8	12	12
Participant 2	13	17	8	14	15	12	31	13	9	17	14
Participant 3	12	16	9	14	13	14	34	15	7	18	13
Participant 4	12	18	13	16	15	1	35	16	6	12	18
Participant 5	12	21	10	18	17	15	36	19	7	20	12
Participant 6	10	19	8	11	12	11	27	19	8	11	11
Participant 7	9	15	8	13	11	12	26	14	5	13	14
Participant 8	15	19	9	14	13	18	28	15	9	14	18
Participant 9	16	17	13	10	10	15	21	12	10	12	19
articipant 10	15	23	10	9	14	11	25	15	11	11	22
articipant 11	16	21	6	15	12	17	30	19	6	9	29
verage Time	**12,8**	**18,5**	**9,3**	**13,8**	**13,3**	**12,5**	**29,1**	**15,9**	**7,8**	**13,5**	**16,5**

Average Total time Android (in seconds)	166,6
Averge Total time in iOS (in seconds)	163,2
Difference	2,12%

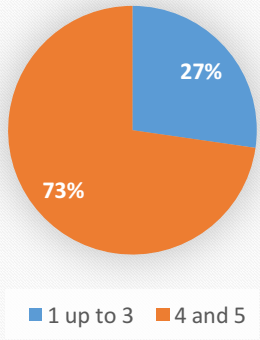
Post – Task answers

Post - Task Questions									
	Question 1	*Question 2*	*Question 3*	*Question 4*	*Question 5*	*Question 6*	*Question 7*	*Question 8*	*Question 9*
Participant 1	Yes		2	4	Yes	3	Yes	4	No
Participant 2	Yes		4	5	Yes	4	Yes	5	No
Participant 3	Yes		4	2	Yes	5	Yes	4	No
Participant 4	Yes		5	3	Yes	5	Yes	2	Yes
Participant 5	Yes		5	5	Yes	5	Yes	2	Yes
Participant 6	No	7	5	4	Yes	4	Yes	4	No
Participant 7	Yes		4	5	Yes	4	No		No
Participant 8	Yes		2	4	Yes	4	No		No
Participant 9	Yes		5	5	Yes	2	Yes	4	No
Participant 10	No	7	4	3	No		No		Yes
Participant 11	Yes		5	5	Yes	2	No		No

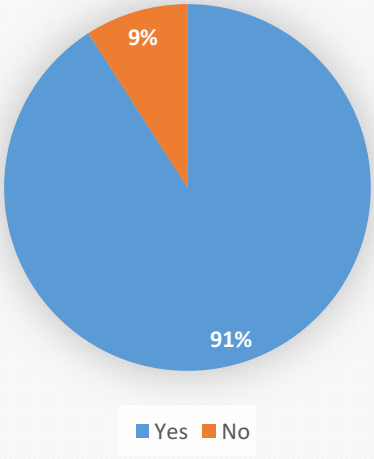




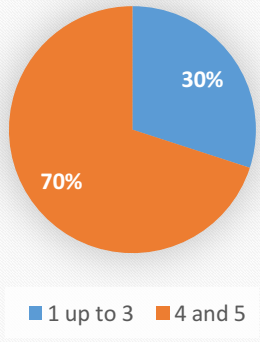
4. How easy was for you to accomplish the same task in the different devices?
(1 means very difficult and 5 means very easy)



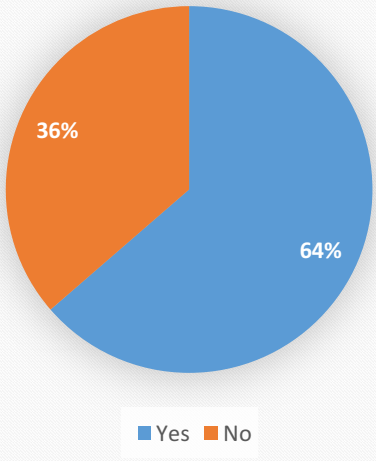
5. Have you ever used an Android device in the past?



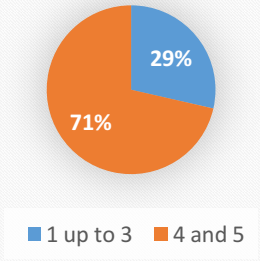
6. If you answered "Yes" in the previous question, how close is the general style of the app (the buttons, the pickers and the camera) to the apps you were using in the Android device.
(1 means very different and 5 means exactly the same)



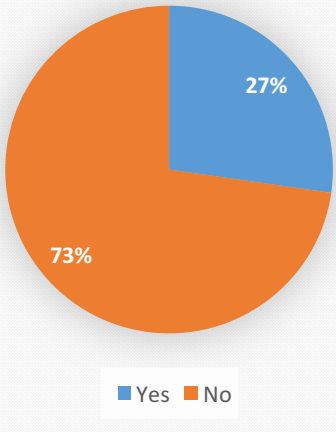
7. Have you ever used an iOS device in the past?



8. If you answered "Yes" in the previous question, how close is the general style of the app (the buttons, the pickers and the camera) to the apps you were using in the iOS device.
(1 means very different and 5 means exactly the same)



9. Do you think that the app is better in one of the two platforms?



MACS Risk Assessment Form (Project)

Student:
Georgios Karagiannis

Project Title:
Cross-Platform Mobile Development using Xamarin and the Evaluation of the Application

Supervisor:
Dr Lilia Georgieva

Risks:

Risk	Present (give details) (tick if present)	Control Measures and/or Protection
Standard Office environment- includes purely software projects	Nothing	Nothing
Unusual peripherals e.g. Robot, VR helmet, haptic device, etc.	Nothing	Nothing
Unusual Output e.g. Laser, loud noises, flashing lights etc.	The wrong participants responses or misleading answers can lead to wrong output	More explanations of the questions and better preparation.
Other risks	Nothing	Nothing