Analysis of companies’ experience with cross-platform development compared to native development for mobile devices

Author: Jan Steczko
Supervisor: Jesper Andersson
Semester: Spring 2016
Subject: Computer Science
Abstract

Today, there are a handful of different platforms for mobile phones. Writing an application natively for each mobile operating system is time consuming and expensive. This situation has created a need for using cross-platform frameworks, that allow programmers to create an application once and run it on all platforms. The problem is that it is not certain whether cross-platform apps can fully replace native ones, or if by using cross-platform tools some desired qualities are lost. Investigating this issue would allow to find out which one of these two application development methods is better, or in which situation it is better to choose one over the other. Such knowledge would allow to decrease development time and costs. Companies that create mobile applications on a daily basis have expertise in this area. Thus, thirteen interviews were done with thirteen distinct businesses in order to research this problem. The results showed that native development produces higher quality applications, but there are some situations where it might be better to use cross-platform frameworks.

Keywords: Mobile development, cross-platform development, cross-platform frameworks, native apps, cross-platform apps, hybrid apps, web apps, Android, iOS, Windows Phone
Contents

1 Introduction ................................................. 5
   1.1 Background ........................................... 5
   1.2 Previous research .................................... 7
   1.3 Problem formulation ................................. 7
   1.4 Motivation ............................................. 8
   1.5 Research Question .................................... 8
   1.6 Limitation ............................................. 9
   1.7 Target group .......................................... 9
   1.9 Outline ................................................ 9

2 Theory and background .................................. 10
   2.1 Definitions ............................................ 10
   2.2 Software development methods ....................... 11
   2.3 Mobile platform analysis ............................ 12
   2.4 Cross-platform frameworks .......................... 16
   2.5 Previous research review ............................ 17

3 Method ...................................................... 20
   3.1 Scientific Approach ................................... 20
   3.2 Method Description ................................... 20
   3.3 Reliability and Validity ............................... 22
   3.4 Ethical Considerations ................................. 22

4 Results ..................................................... 23
   4.1 Data from introductory questions .................... 23
   4.2 Differences between cross-platform projects and native projects 25
   4.3 Comparison of native apps and cross-platform apps .......... 28
   4.4 Other results .......................................... 31

5 Analysis .................................................... 32
   5.1 Analysis of questions 1 - 4 .......................... 32
   5.2 Analysis of questions 5 - 10 ......................... 32
   5.3 Analysis of questions 11 - 14 ....................... 33
   5.5 Summary of the analysis ............................. 34

6 Discussion ................................................ 36
   6.1 Discussion about previous research ................. 36
   6.2 Answers to research questions ........................ 36

7 Conclusion ............................................... 38
   7.1 General conclusion .................................... 38
   7.2 Future Research ....................................... 38

References .................................................. 39

Appendix 1 .................................................. 43

Appendix 2 .................................................. 46
Appendix 3 ................................................................. 48
Appendix 4 .................................................................. 50
Appendix 5 .................................................................. 52
Appendix 6 .................................................................. 54
Appendix 7 .................................................................. 56
Appendix 8 .................................................................. 58
Appendix 9 .................................................................. 60
Appendix 10 .................................................................. 62
Appendix 11 .................................................................. 64
Appendix 12 .................................................................. 66
Appendix 13 .................................................................. 68
1 Introduction

When creating an application for mobile devices, there are a few technologies to choose from. A programmer can decide to develop an application natively or with a cross-platform framework. Choosing the right programming method can save time and can help reduce costs. This thesis contains an analysis that compares these two development methodologies with each other and shows the advantages and disadvantages of both of them.

1.1 Background

Nowadays, many people own a smartphone. These phones are like a personal computer that can be worn inside a pocket. Smartphones can be used for various purposes like calling, sending text messages, surfing the internet, playing games, navigation or work in general.

1.1.1 Mobile platform market

There are a handful of different mobile platforms that exist today. The most popular ones are iOS, Android and Windows Phone. Currently, the most dominant one is Android from Google that held 82.8% of the mobile market share worldwide in 2015. The second dominant one is iOS from Apple which had 13.9% of global market share, and the third one is Windows Phone from Microsoft that held 2.6% of the market in 2015. There are other platforms, like Blackberry or Symbian but their presence in the mobile world is minimal [1].

For some developers creating applications for only the Android platform may sound like a good choice, because Android’s 82.8% global market share may look satisfactory. However, other developers who want to target as many users as possible, have to create applications for other platforms. By focusing on only iOS and Android, developers would reach 96.7% of the worldwide market, which for some programmers might sound as a better option than Android’s 82.8% market share, and thus would want to create applications for more than just Android.

Furthermore, the situation of the mobile platform market depends on the country. In some regions of the world the fragmentation of mobile platforms may look differently. For example, in the United States, Android had 52.3% of the market share and iOS had 43.6% of the market share in 2015 [2]. Together they have 95.9% of the market in the United States. This shows that a developer that wants to target users in the US, is forced to develop for both Android and iOS, because otherwise only half of all mobile phone users will be able to use the app. Therefore, before deciding on which mobile platforms to focus on, the country or region where the application users are should be taken into account.

1.1.2 Native development compared to cross-platform development

These developers that wish to create an application for more than just one platform, need to use separate APIs and programming environments for every mobile platform. Because of this they are forced to create the same application more than once. This enlarges the costs of development and increases the time needed to develop a mobile application [3].

This has produced a demand for creating cross-platform solutions. By using cross-platform programming a developer could write an application only once, and then release it on all platforms. With cross-platform solutions it would be possible to reduce costs and time required to develop applications. Moreover, cross-platform development would result in having only one codebase and just one programming environment [3].

Native programming for Android, iOS and Windows Phone requires knowledge in three different programming languages, that is Java [4], Swift [5] and C# [6] respectively. By using cross-platform solutions it would be possible to develop a mobile program in only one language.

There are various cross-platform technologies today. One of such technologies are cross-platform frameworks which allow the developer to write code in a certain programming language, and these frameworks later compile that code into native programming language [7]. Thus, the
cross-platform framework is a middle layer between the developer and the native code. As a consequence, when using cross-platform tools a developer receives in the end native code, just as if he or she was using regular native programming.

One example of such cross-platform framework is Xamarin. This tool allows developers to write code in C#. It supports creating apps for Android, iOS and Windows Phone. It is possible to write applications in Xamarin with help of their own special platform, or by using a plug-in to Microsoft Visual Studio. It allows creating client and server side code. Moreover, it has a special tool built in to help developers create user interfaces [8].

Another example of a cross-platform framework is Appcelerator Titanium. It enables programmers to write applications with help of Javascript and XML. It supports creating apps for Android and iOS. It compiles its Javascript code into a mix of native code and Javascript code [9].

Apache Cordova is another cross-platform framework. It lets programmers create apps with such web technologies as: Javascript, CSS and HTML. With Apache Cordova it is possible to write applications for iOS, Android, Windows Phone, Blackberry, Firefox OS, Ubuntu and others [10].

Facebook created its own cross-platform framework and named it React Native. It makes use of Javascript and React library, and this library is also developed by Facebook. This framework supports Android and iOS [11].

There are also highly specialized cross-platform frameworks, that focus on creating only one type of applications. Unity is such a framework. It enables programmers to create 2D and 3D games and its programming language is C#. It supports multiple platforms like: iOS, Android, Windows Phone, PS4 and Xbox to name a few [12].

There are also other cross-platform solutions, that do not require the use of cross-platform frameworks. One of such solutions is to create a web application and adjust it to smartphones’ screen sizes. Such programs have to be launched inside a web browser, and usually they cannot be installed on a mobile phone [13].

A third cross-platform solution is to create a hybrid application. Such programs consist of a web application that is written in for example Javascript, CSS and HTML and then such web app is wrapped inside a small native layer. Then, a potential user can install this application on his or her mobile phone, but when the user launches this application, the native layer of the app starts the web part of the whole application. Thus, such programs are a mix of web technologies and native technologies. These programs are called hybrid applications [14].

Both native and cross-platform technologies have their benefits and drawbacks. It would be beneficial for a developer to learn about the properties of both native and cross-platform before creating a mobile application, because then he or she could write apps that are better adapted in a given context [15].

1.1.3 The process of creating mobile applications

Since each native programming language for mobile platforms requires different range of skills and each cross-platform technology requires different skill set as well, the organization of a project that uses native languages could look differently than the arrangement of a project where cross-platform tools are used. This in turn means that a company that creates native apps may hire totally different programmers than a company where cross-platform apps are produced [16].

There are also various development methodologies that different companies follow when creating software. A few examples include agile, scrum and waterfall. In agile development, the application is created in short time frames, which are called iterations. These iterations can last from one to four weeks. The goal is to release new or updated software at the end of every iteration. Scrum is somewhat similar to agile. In scrum, each iteration is called a sprint. At the end of each sprint a goal must be achieved by the team developing the software. Scrum projects usually have a log of prioritized work to be done, regular meetings where process is discussed, and meetings where people reflect about the past sprints. Waterfall development method is different then the previous two methodologies. The process of creating the software is rigid and linear. A waterfall project has a
set of distinct phases, and at the end of each phase some goals must be met. When one phase is completed, the development goes to the next phase. Usually it is not possible to go back to the previous phase [17], [18].

It is uncertain whether the choice between cross-platform development and native development could affect the choice of development methodology. It is also unknown if one development method is better than the other when producing applications for mobile platforms.

1.2 Previous research

Several studies were done in the past that investigated different means of developing mobile applications. Some of this previous academic work has been used as a basis for gaining a deeper understanding of development for mobile platforms.

In the article "A comparative analysis of cross-platform development approaches for mobile applications", Spyros Xanthopoulos and Stelios Xinogalos analyze cross-platform development approaches and try to find their advantages and disadvantages [7].

Inside the conference article "Comparison of Cross-Platform Mobile Development Tools" written by Manuel Palmieri, Inderjeet Singh and Antonio Cicchetti, authors compare four cross-platform tools: Rhodes, PhoneGap, DragonRad and MoSync. The goal of their research was to get an overview of these frameworks and their APIs, programming languages, supported mobile operating systems, and their environments. By doing so the authors of this research wanted to help developers in choosing a cross-platform framework that suits best their needs [19].

The article "Survey, Comparison and Evaluation of Cross Platform Mobile Application Development Tools" whose authors are Isabelle Dalmasso, Soumya Kanti Datta, Christian Bonnet and Navid Nikaein shows a comparison and assessment of a few cross-platform frameworks. The tools that were analyzed were: PhoneGap, Titanium and Sencha Touch. In this study, the authors created test applications with these frameworks and measured their CPU performance, memory usage and power consumption. Their findings showed that PhoneGap used less memory, CPU and power than other frameworks that were used in this research [20].

The paper "Cross-Platform Mobile Development: Challenges and Opportunities" created by Arianit Kurti and Suyesh Amatya presents a literature study of cross-platform development for mobile platforms. Their results show that web apps and hybrid apps are the best cross-platform solutions. Their findings also pointed out that even though cross-platform frameworks are not fully matured, these tools have great potential [21].

The paper "Evaluation of cross-platform frameworks for mobile applications" whose authorship belongs to Andreas Sommer and Stephan Krusche reveals a study where native SDKs were compared with cross-platform frameworks. To perform the study, authors created a sample application with Android SDK, iOS SDK, Titanium, Rhodes, PhoneGap and Sencha Touch. The authors found that cross-platform tools can be recommended in general, but if there are high requirements regarding to quality, performance or user experience, then native solutions are a better choice [22].

The article "Review of Multi-Platform Mobile Application Development Using WebView: Learning Management System on Mobile Platform" presents a study that was performed by Timothy Yudi Adinugroho, Reina and Josef Bernadi Gautama. Their research investigated strengths and weaknesses of hybrid applications for mobile platforms. The results of their study showed that whether to use hybrid apps or not depends on the application’s requirements. They also found that if simplicity of development plays an important role, then hybrid applications are a good replacement for native SDKs [23].

1.3 Problem formulation

Since there are various mobile operating systems in the mobile industry, it is somewhat appealing to develop an application for multiple platforms. A programmer can either develop an application in
the native language or with a cross-platform framework. Nonetheless, the decision to develop with cross-platform or native may not always be straightforward.

Cross-platform development is clearly simpler and cheaper, but it is not certain if this technology is mature enough to replace native development. Moreover, it is not safe to believe that cross-platform applications will have the same performance or customization abilities as native ones. When using a middle layer tool like a cross-platform framework, there is a risk that the resulting application may be somehow limited or slower [3], [15], [24].

Furthermore, there are many different cross-platform frameworks to choose from, for example Unity, Appcelerator Titanium, Xamarin, React Native or Apache Cordova. The creators of each one of these frameworks claim that their tool can replace native development. Therefore, it might be difficult to choose the best cross-platform framework.

Native programming has a few positive and negative sides as well. One downside of native is that it increases time that it takes to deliver an application, because development and testing efforts require more time. The maintenance of native applications forces programmers to release updates and bug fixes to all platforms. Instead of focusing on a single codebase, the effort is split into each platform. All of this increases costs. On the other hand, with native development programmers can be sure that they will have access to all functionalities and APIs of a given platform [3], [15], [24].

Because of these various issues with both native and cross-platform, it is not certain which one is better than the other.

1.4 Motivation
Companies and developers that create or want to start creating mobile applications would be interested in knowing whether cross-platform is better than native and vice versa. This knowledge would allow them to offer applications with higher quality. Moreover, it would let them reduce costs and simplify development when possible.

Furthermore, higher knowledge about all differences between cross-platform and native would allow these companies and programmers to deliver applications that better suit the needs of their customers. This is because a cross-platform solution might fit better one customer, whereas a native solution would fit better a different customer.

1.5 Research Question
The purpose of this thesis was to find differences between native development and cross-platform development in general. To find their advantages and disadvantages, and to point out which one is more beneficial for users and developers. The research questions for this thesis were the following:

<table>
<thead>
<tr>
<th>RQ1</th>
<th>How does the process and organization in cross-platform projects differ from native projects?</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQ2</td>
<td>Are there some features or qualities that are worsen or lost when developing with cross-platform frameworks instead of native?</td>
</tr>
</tbody>
</table>

In the second research question, "features" mean access to hardware components of the mobile phone, like geolocation, gyroscope, camera, device orientation, SMS or telephone. "Qualities" mean speed and response time of the app (that is, time it takes for an application to react when a user presses with his or her finger on the screen) and also quality of the graphical user interface (if it makes the application look like a native part of the phone).
1.5.1 Expected findings

The results could vary. One outcome could be that native development is clearly advantageous when compared to cross-platform development and that native is the only viable way to develop applications for smartphones.

Another possible outcome would show the opposite, that programming in cross-platform environment has more benefits than programming in native environment, and that native development is not worth its higher costs and efforts.

A third result could be that cross-platform and native are pretty equal, and that cross-platform is a good solution in some circumstances, whereas native is a good choice in different types of situations.

A forth and last possible outcome could be that there is an external factor that a company or developer cannot affect. It could be for example a client, that makes a decision of whether to use cross-platform or native.

1.6 Limitation

The exact dissimilarities between various cross-platform frameworks or precise differences between web applications, hybrid application and cross-platform applications were not examined. Differences between diverse native programming environments were not inspected as well. The reason for this is that other researches have made such comparisons in the in the past (their work was listed in section 1.2) and the goal of this thesis was to solely focus on analyzing differences between native development and cross-platform development and to find strengths and weaknesses of each one of them.

1.7 Target group

Developers who currently create or want to start creating mobile applications in the near future, might utilize the information that is shown in this report. Business executives who manage one or more teams of mobile application programmers could also be interested in the findings of this thesis.

1.9 Outline

In this paragraph there is an outline of the rest of this thesis. Chapter 2: Method, contains a detailed description of the scientific approach that was used in order to answer the research questions. Inside Chapter 3: Results, outcomes from the research are shown. These results are presented in form of text and tables. In Chapter 4: Analysis, a dissection of the data can be found. Chapter 5: Discussion, contains overall reflections on the discoveries. Moreover, in this section it is also discussed whether the investigation done during this thesis answers the research questions. Chapter 6: Conclusions, includes a summary of the findings. It also presents conclusions that can be drawn from the analysis. Possible future research is also discussed here.
2 Theory and background

This chapter presents and explains the most important frameworks, techniques, definitions and concepts, that may serve as a theoretical basis for the rest of this thesis. 

2.1 Definitions

There are a few terms that are used throughout this thesis. In this section there are the definitions of those terms.

2.1.1 App

This is an abbreviation for Application. It is a program that is designed to run on a mobile device. Its purpose is to execute specific operations on a given smartphone. Usually they can be downloaded from an application store. Developers can create apps that serve their users in various fields like health, tourism or education [14].

2.1.2 Mobile platform

It is an operating system, usually Android, iOS or Windows Phone which runs on a specific mobile phone hardware. Table 2.1 below depicts the differences between three most common mobile platforms, that is Android, iOS and Windows Phone.

<table>
<thead>
<tr>
<th></th>
<th>Virtual machine</th>
<th>Programming language</th>
<th>User interface</th>
<th>Memory management</th>
<th>Application store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>Dalvik</td>
<td>Java</td>
<td>XML files</td>
<td>Garbage collector</td>
<td>Google Play Store</td>
</tr>
<tr>
<td>iOS</td>
<td>-</td>
<td>Swift or Objective-C</td>
<td>Cocoa Touch</td>
<td>Reference counting</td>
<td>iTunes App Store</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>CLR</td>
<td>C#</td>
<td>XAMP files</td>
<td>Garbage collector</td>
<td>Windows Phone Store</td>
</tr>
</tbody>
</table>

Table 2.1: Differences between Android, iOS and Windows Phone platforms [25].

2.1.3 Native app

This is the same thing as Native Application. This is a program for a smartphone that is created in a programming language and API (Application Programming Interface) that is dedicated for a given mobile platform. Native apps are also created in an IDE (Integrated Development Environment) that is devoted to a particular platform. Such IDEs contain all tools that are necessary for building and testing the native applications [7].

2.1.4 Cross-platform app

It is an abbreviation for Cross-platform Application. This is a program designed for a smartphone. In order to create such applications, a cross-platform framework is required. A programmer can develop an app in a cross-platform environment, and later that tool is compiling the code (it could be C# or Javascript) into native code. That framework generates native code for desired mobile platforms. Some cross-platform frameworks allow the programmer to create an application just once, before compiling it to all platforms, so that the final products looks the same on all mobile operating systems. Other cross-platform tools give a possibility to create such codebase, so that the final products looks a bit differently on each mobile platform [8], [9].
2.1.5 Web app
This is a shorthand for Web Application. This is a program that is usually created in HTML, CSS, Javascript, jQuery or a similar web technology. It is possible to run this application only in a web browser, and it requires internet connection. It is not possible to install such apps. Such programs can adjust themselves to the user’s screen size, so that they can run on desktop computers, tablet computers and mobile phones [26].

2.1.6 Hybrid app
It is an abbreviation for Hybrid Application. This is a web app, that in wrapped inside a simple native application. Such hybrid apps can be installed on a mobile device, but when they are launched, the native layer of that application runs the web app that is part of the whole application. Therefore hybrid apps require internet connection in order to function [26].

2.1.7 Software development methodology
It is a framework that is used to structure, plan and control the process of developing software. Over the last few years and decades many different software methods were created. Every methodology has different organizational and technical aspects. Each one of them has its advantages and disadvantages. Some of them are better suited in one context, and others are a better choice in a different context or project [27]. A few examples of software development methods are: waterfall, scrum and agile [18].

2.2 Software development methods
Native applications and cross-platform applications, just like any software, need to be planned, designed, built and tested. When a company wants to create a mobile application, it can decide to whether follow a software development method or not. This section contains a deeper analysis of common development methodologies, that companies which build mobile apps may follow.

2.2.1 Waterfall methodology
Waterfall model is considered being a traditional development method. Waterfall model is divided into several phases. Each phase has a different set of activities and at the end of each phase some deliverables must be produced. After completing one phase, the development goes to the next and most often it is not possible to go back to the previous stage [18]. The number of phases and their actions differ from company to company, and from project to project, but one Waterfall model may look as the following:

<table>
<thead>
<tr>
<th>Number</th>
<th>Phase name</th>
<th>Actions during the phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>System Engineering</td>
<td>High level architecture of the software is produced.</td>
</tr>
<tr>
<td>2</td>
<td>Requirement analysis</td>
<td>Requirements are gathered and specified</td>
</tr>
<tr>
<td>3</td>
<td>Design</td>
<td>Software architecture, tests, and design are specified.</td>
</tr>
<tr>
<td>4</td>
<td>Construction</td>
<td>Program code is produced.</td>
</tr>
<tr>
<td>5</td>
<td>Testing</td>
<td>System and unit tests are done and test reports are created.</td>
</tr>
<tr>
<td>6</td>
<td>Installation</td>
<td>The software is installed and afterwards it is ready to use.</td>
</tr>
<tr>
<td>7</td>
<td>Maintenance</td>
<td>Errors and faults are corrected, and change requests from clients are fulfilled.</td>
</tr>
</tbody>
</table>

Table 2.2: Phases in a Waterfall project [28]
The waterfall methodology is used inside those projects where the requirements are not likely to change and when the user participation in the development process is not necessary. Usually no prototypes of the software are created. Over the last years companies have replaced the waterfall methodology with other development methods, and therefore the popularity of waterfall model has decreased. One of the reasons for this shift is that if during development something new occurs, that nobody has thought about before, it is not possible to go back to previous phase [18].

2.2.2 Spiral methodology

Spiral methodology is an extension of the waterfall model, because it introduces prototyping and users are involved during development. Spiral projects have stages, and at each stage certain activities are done, before going to the next stage. The amount of stages can vary from project to project, and from company to company. One example of a spiral model may look as the following [18], [27]:

1. The system requirements are gathered.
2. A preliminary design of the software is created.
3. A first prototype is created, based on the preliminary design.
4. The first prototype is evaluated. Requirements and design for the second prototype are formed.
5. The second prototype is created.
6. The steps 4 and 5 are repeated until a design has been created that satisfies the customer.
7. The final product is built and tested.

A prototype is an incomplete version of a program and it usually contains only a few features. The final product may to totally different than the prototypes that were released.

2.2.3 Agile methodology

The idea of agile methodology is to minimize risks by developing software in short time frames (iterations). One iteration may last from one to four weeks. During each iteration the following tasks may be performed: planning, requirements analysis, design, coding, testing, and documentation. Therefore iterations are like small projects on their own. At the end of each iteration, a new or updated software is released and the team that worked on the application evaluates their work. Usually agile development methods involve users of the software in the process, and face-to-face communication is preferred over written communication [18].

2.2.4 Scrum methodology

Scrum is an agile method but it is different from agile in some details. Software is developed in short iterations, that are called sprints. A project that uses scrum usually has a scrum master, whose job is to remove any obstacles that would affect negatively the productivity of the team that produces software [29].

A scrum project usually has a log of prioritized work that must be done. Moreover, brief daily meetings are held where progress is discussed and future actions are planned [18].

Scrum allows the creation of self-organizing teams and it encourages verbal communication between teams that are involved in the development process. Scrum accepts that a problem cannot be fully comprehended or defined. Thus, it focuses on maximizing the team’s ability to respond to appearing challenges [18].

2.3 Mobile platform analysis

In this section, there is a deeper analysis of most dominant mobile platforms, that is iOS, Android and Windows Phone. Their history, architecture and development tools are shown.
2.3.1 iOS platform

Apple is the company which creates the iOS platform. In 2007, Steve Jobs introduced iPhone, a mobile phone that had a touch screen that allowed zooming and scrolling. It was also possible to browse the internet on the first iPhone. The second iPhone used 3G network and it opened doors for third party developers to create iPhone apps. In June 2010 Apple renamed its iPhone OS to iOS. In the same year the first iPad was released by Apple. Every upgrade that was released by Apple before 2010 and after 2010 introduced more and more new features to the iOS environment [30].

2.3.1.1 Architecture

The architecture of iOS is layered. It acts as an intermediary between the apps that the developers create and the hardware. Apps cannot communicate with the hardware directly, instead they do it with a collection of defined system interfaces [31]. Figure 2.1 shows the layers that iOS consists of.

![Figure 2.1: Layers of iOS](image)

Cocoa Touch layer contains the most important frameworks for creating iOS apps. These frameworks provide the basic infrastructure and appearance of the apps. This layer contains support for key technologies like multitasking, push notifications, touch input and other system services [32]. Media layer includes graphics, audio, and video technologies that developers use to implement multimedia features in their apps [33]. Core Services layer contains fundamental system services for applications. This layer also includes support for geolocation, networking, social media and iCloud [34]. The last layer, Core OS, contains low-level properties that most other technologies build upon. Usually developers do not deal with this layer directly. In cases when programmers do use this layer, they do so because they explicitly need to handle security and communication issues [35].

2.3.1.2 Development

X-Code is the programming environment that is used when developing applications for the iOS platform. It can be downloaded for free from Apple’s website. It allows the developers to write code in two programming languages: Objective-C and Swift. With X-Code it is possible to write apps for iPhone, iPad and Mac. Moreover, X-Code has special tools built in, that allow testing and checking code coverage [36].

2.3.2 Android platform

Google released its first version of Android in October 2008. The second release of Android (named 1.5 "Cupcake"), took place in April 2009. That version had some features that laid the foundations for future Android versions. It had screen widgets, pull down notification window and Gmail integration. Version 1.5 also had Android market, where third party developers could release their own Android apps. Every release of new Android version introduced new features and expanded the
possibilities of Android. Version 3.0 "Honeycomb" added support for the tablets. Today, Android is the most popular operating system for smartphones worldwide [37].

2.3.2.1 Architecture
Android’s architecture consists of four layers. Each layer contains a set of components that a given layer uses. The picture below shows how the architecture of Android looks.

![Android Architecture Diagram]

Figure 2.2: Android’s architecture [38].

The first layer from top is the Applications layer. All apps that third party developers write are installed inside this layer. Some examples app are: web browser, calendar, games, book readers, etc. The second layer is the Application Framework. It contains Java classes that provide various high-level services to the applications. Third party developers make use of these Java classes. The third layer is called Libraries and it is divided into two sections. The first one is Android Runtime. This section contains a key component named Dalvik Virtual Machine, which uses core Linux features such as memory management and multi-threading. All Android applications run on Dalvik VM. Android Runtime section also contains Core Libraries, which gives Android developers access to standard Java libraries and classes that are specific to just the Android platform. Among other things these frameworks are responsible for graphics drawing, database access, interface building, internet security and others. The last and forth layer is the Linux kernel, which is the most fundamental layer. It contains the most important drivers like camera, WiFi, audio, keypad, display, etc. The kernel also takes care of networking and power management [38].

2.3.2.2 Development
An official development environment for the Android platform is Android studio. It can be downloaded for free from the internet. It gives a possibility for writing code in two languages: Java
and C++, but Java is the primary one. With Android studio it is possible to develop software for all Android related operating systems like: Android phones, Android tablets, Android Wear, Android TV and Android Auto [39].

2.3.3 Windows Phone platform

Microsoft is the company that created Windows Phone. This operating system is a successor of Windows Mobile, but these two platforms are not compatible with each other. Windows Phone was first launched in October 2010 as Windows Phone 7. Two years later, Microsoft released Windows Phone 8. Two additional years later, Windows Phone 8.1 was announced. Each update added new features to Microsoft’s operating system. A few examples of new features that were added in Windows Phone 8.1 are: a voice assistant Cortana, personalized starting screen, notification centre and live lock screen. In 2015, yet another version of this operating system was released, named Windows 10 mobile [40].

2.3.3.1 Architecture

Windows Phone’s architecture consists of a few distinct layers. The figure below shows how these layers and sections are organized.

![Figure 2.3: Windows Phone architecture](image)

The top layer is the Application layer. Inside this layer all apps are installed, including these that are built but third party developers. The second layer contains various Windows services, that the applications from the above layer make use of. Windows Platform Services layer is responsible for different aspects of the system, like graphics, databases, audio, video, radio or telephony. A third layer is the kernel, also called Windows Core. Since the release of Windows 10, all kinds of devices that run on Windows platform, whether it is a desktop, laptop, mobile phone or Xbox, use the same kernel. Therefore all these devices that use Windows 10 have a single codebase. Windows Core is responsible for several things, like memory management, networking, security, file storage, and others [40], [41].

2.3.2.3 Development

In order to develop an app for Windows Phone (or Windows 10 mobile) a developer needs to download and install Visual Studio. It is a programming environment that is created by Microsoft and it is available for free. Furthermore, it is possible to use Visual Studio to develop apps for other mobile platforms, like iOS and Android. There are also extensions for Visual Studio, that can be installed, that allow creating mobile cross-platform apps [43].
2.4 Cross-platform frameworks

A developer that would like to create a mobile application natively for two or more mobile platforms, would have to create the same app several times, once for each platform. In order to avoid this, several companies and organizations have created cross-platform frameworks. Their purpose was to give developers a possibility to create an app only once, and then release it on all desired platforms. In this section, there is an analysis of a few of such cross-platform frameworks. Their history and features are shown.

2.4.1 Xamarin

Xamarin is a company that was founded on 16 of May, 2011 [44]. They created a tool, that allowed developers to write code in C#, and then compile such code into native apps for Android, iOS and Windows Phone [8]. Since then, Xamarin’s popularity was growing and in 2016, more than 1,4 million developers worldwide have used their framework. Furthermore, Xamarin has several offices in 14 countries and it hires more than 350 employees [45]. On 24 of February, 2016, Microsoft announced that it has purchased the company that has built Xamarin [46].

According to Xamarin creators, their framework allows the developers to build apps that have native user interface. They also say that apps created with Xamarin have the same access to platforms’ API as native ones and that the performance of Xamarin apps is the same as native [8].

Furthermore, Xamarin creators claim that in their framework it is possible to adjust some parts of the code for each platform separately. By doing so the app will look differently on each platform, and thus it will have a native user interface, but the applications’s logic will be the same for all platforms. By using this method, it is possible to on average share 75% of app code across all mobile platforms. Xamarin also offers the use of a special tool, called Xamarin.Forms, and by using that tool it is possible share almost 100% of the app code, across all three platforms. The figure below shows how this can be achieved [8].

![Figure 2.4: How app code can be shared across different platforms with Xamarin](image)

Xamarin can be downloaded as a separate platform for building mobile applications, or as an extension to Visual Studio [8].

2.4.2 Appcelerator Titanium

Appcelerator is a company that was founded in 2008. In the beginning their focus was to create an open source platform that helped in creating rich internet applications [47]. In 2009, the company shifted its attention to mobile apps. In that year they released a beta version of Titanium, a framework that allowed creating cross-platform apps for Android and iOS [48]. Appcelerator released Titanium 1.0 in March 2010 [49]. Through out the years, the company was growing. In 2016, the development community of Titanium consisted of more than 600 000 developers [50].

Today, Titanium framework allows creating apps for not just Android and iOS, but also for Blackberry OS and Windows Phone. All apps that are created in Titanium have Javascript codebase.
This framework allows creating not only native apps, but also hybrid apps and web apps. According to Appcelerator, from 60% to 90% of the code can be reused when building an application for multiple platforms [50].

Titanium offers also the use of Alloy framework. It is a tool that allows programmers the use of Model-View-Controller (MVC) pattern in their Titanium applications. Alloy gives a toolset for separating the user interface from business logic [50].

Appcelerator offers other products than Titanium. One of them is Appcelerator Platform. It is a framework that also allows building mobile apps with Javascript, but it has a few additional features. Appcelerator Platform enables automation testing, and gives access to real-time analytics.

### 2.4.3 Apache Cordova and PhoneGap

Nitobi Software, with its headquarters in Vancouver, Canada, was the company that created PhoneGap, an open source framework that allowed building cross-platform applications for mobile phones. In 2011, Adobe Systems purchased Nitobi Software, and PhoneGap together with it [51]. Adobe continued developing PhoneGap, and soon released Apache Cordova, a second framework for building cross-platform apps.

Both PhoneGap and Apache Cordova allows creating native mobile applications with HTML, CSS and Javascript. Yet, there are a few differences between these two tools. Apache Cordova has a few Adobe services integrated. Furthermore, Apache Cordova is like an engine, that powers PhoneGap. Because of this, Apache Cordova was used as an engine for other frameworks, for example Ionic [52].

### 2.4.4 Unity

Unity is a cross-platform game engine. It belongs to a company named Unity Technologies, that was founded in 2004, with its headquarters in Copenhagen, Denmark [53]. In the beginning Unity was available for only OS X, but through out the time Unity supported more and more platforms. Some examples include Android, iOS, Windows Phone, Xbox, Play Station, Linux, Steam OS, Wii, Oculus Rift and many others [12].

Games that are created in Unity are written in C# or Javascript. Unity allows creating 2D games and 3D games, single player and multi player via network. This game engine has also a few additional features, for example it lets programmers insert ads into their games, it provides analytics for checking users behavior, it creates performance reports that provide lists of games’ exceptions, and others [12].

### 2.5 Previous research review

This section contains a review of academic work that was done in the past by other researches, regarding cross-platform frameworks and native applications. Most of them were literature studies or controlled experiments.

#### 2.5.1 A study performed by researchers S. Xanthopoulos and S. Xinogalos

Spyros Xanthopoulos and Stelios Xinogalos did an analysis of trends in developing cross-platform apps. They sorted cross-platform apps into four categories and attempted to find advantages and disadvantages of each category. According to them, there are four cross-platform app categories: web apps, hybrid apps, interpreted apps (applications where native code is used to implement user interface, but underlying logic is implemented with another technology) and generated apps (apps where the whole code of a cross-platform framework is compiled into native code) [7].

Their analysis found that web apps, hybrid apps and interpreted apps have limited access to phone’s hardware, but generated apps had full access. Furthermore, they found that web apps and hybrid apps had a "simulated" look and feel of the user interface, whereas interpreted and generated apps had a native look and feel. They also stated that web apps have low performance, hybrid apps and interpreted apps have medium performance, and generated apps have the highest performance.
Moreover, web applications cannot be distributed via an application marketplace, whereas the other application types can be distributed [7].

In their conclusions, Xanthopoulos and Xinogalos state that there is no best solution when it comes to cross-platform development approach, that should be selected for a mobile app in general. They say that the choice of a cross-platform development approach is depending of several factors. Some examples of such factors are: whether the app needs access to hardware components, whether the app needs to be distributed via an application store or what are the project deadlines [7].

2.5.2 A study performed by researchers M. Palmieri, I. Singh and A. Cicchetti
Manuel Palmieri, Inderjeet Singh and Antonio Cicchetti did a comparison of four cross-platform frameworks. These were: Rhodes, PhoneGap, DragonRad and MoSync. The reason why authors chose these frameworks was that their availability was major, at the time of performing their research. The aim of their study was to analyze these frameworks by comparing their availability of APIs, accessibility to native APIs, programming languages, supported mobile operating systems, licenses, architecture and Integrated Development Environments (IDE) [19].

In their conclusions the authors stated that these cross-platform tools may not be the best choice when developing apps that have very complex business logic or background services. Another thing that the authors say is that these cross-platform frameworks had poor support for 3D graphics. The authors also pointed out some strengths that each of these frameworks had. For example, PhoneGap had the best access to native API and Rhodes was the only framework that supported MVC pattern [19].

2.5.3 A study performed by researchers I. Dalmaso, S. K. Datta, Ch. Bonnet and N. Nikaein
Isabelle Dalmaso, Soumya Kanti Datta, Christian Bonnet and Navid Nikaein did a study where they investigated the differences between three cross-platform frameworks: PhoneGap, Titanium and Sencha Touch. The authors do not give any exact reason why they picked these frameworks, but they wrote that they preferred these frameworks because they were open-source and that these tools allowed the use of web technologies, such as HTML and CSS [20].

Firstly, the authors analyzed the architecture of the frameworks, their documentation, programming environment, deployment, framework stability and also listed their strengths and weaknesses [20].

Secondly, the authors created a few test applications by using the listed cross-platform frameworks, but only for Android platform. Next, they examined how these test applications behave. They investigated the memory usage, CPU usage and power consumption [20].

Their findings showed that PhoneGap was the best framework when it comes to memory usage, CPU usage and power consumption, but they also found that in PhoneGap it was more difficult to create more advanced or sophisticated user interface. Furthermore, they found that in Sencha Touch framework it was easier than in PhoneGap to create desired user interface [20].

2.5.4 A study performed by researchers S. Amatya and A. Kurti
Suyesh Amatya and Arianit Kurti did a literature research regarding cross-platform frameworks and mobile phones. Firstly, they selected their library sources, and then they searched for papers and academic work that contained such keywords as "mobile" or "cross-platform". They found 17 articles that matched their criteria [21].

Their outcomes showed that since 2009 and onwards, cross-platform frameworks were gaining popularity. Authors of the study could not find any articles prior to 2008. They also found that web applications are a prevailing cross-platform solution, and that hybrid applications are catching up. They also state that web apps do not have the same performance as native apps. The authors also said that hybrid apps can be a serious alternative to web apps. Furthermore, cross-platform frameworks, like Titanium or PhoneGap can also be seen as a good option, but that they are still in their initial stages of development [21].
In the end, authors of the study underline the fact that the choice between cross-platform solutions and native solutions depends on the app objectives and business realities that the app tries to address [21].

2.5.5 A study performed by researchers A. Sommer and S. Krusche

Andreas Sommer and Stephan Krusche have done a study where they compared native SDKs with four cross-platform frameworks: Titanium, Rhodes, PhoneGap and Sencha Touch. They created the same application five times. Once with Android, once with iOS, once with Titanium, once with Rhodes and once with PhoneGap and Sencha Touch combined [22].

Afterwards, the authors were comparing the applications with each other. They examined their features like: geolocation, database access, networking, access to hardware components, access to data that was provided by other applications (such as contacts) and lifecycle events (happenings like start, pause, resume or ability to run a background service). The authors also examined usability features, for example: customization of buttons and lists or how apps reacted to touch gestures. Another aspects that the authors examined were: developer support (what kind of help a given framework offered to its developers), reliability and performance (how apps carry out their tasks under a certain condition and a specified period of time) and deployment (the ease of making changes after deploying the app) [22].

The outcomes of the study are the following. Titanium and PhoneGap combined with Sencha Touch had the same developer support as native SDK. Except that, Native SDKs outperform the cross-platform frameworks in every category. In the article, authors summarized their study by saying that cross-platform frameworks have their benefits, like for example simplified development, lower costs or lower knowledge threshold. Despite these positive aspects that cross-platform frameworks have, native is a better option when high performance, functionality, reliability or usability is an important requirement [22].

2.5.6 A study performed by researchers T. Y. Adinugroho, Reina and J. B. Gautama

The following study was conducted by Timothy Yudi Adinugroho, Reina and Josef Bernadi Gautama. Their work investigated the strengths and weaknesses of hybrid apps for mobile phones. No cross-platform frameworks (such as PhoneGap or Titanium) were used.

Firstly, the authors created two hybrid applications. These two hybrid apps were implemented with two distinct methods. Afterwards, the authors measured the performance of these hybrid apps and compared them with a native application, that was carrying out the same tasks.

Their results showed that there were no big differences between these three apps. They also concluded that hybrid apps have their upsides and downsides. As an upside, hybrid apps render identically on all platforms, thus if it is a requirement to create an application that looks the same on all platforms, then hybrid apps are a good choice. The authors also stated that the native layer of the hybrid apps give access to the whole native API. On the downside, hybrid apps force the developers to manage many separate codebases (a native layer for each platform and a web part of the app). Moreover, hybrid apps have higher data consumption over the network, which in some cases might be not desirable (for example with an unstable mobile carrier network). There are solutions for this, for example it is possible save certain pages offline for later execution, but this solution might not always be feasible. In summary, the choice between hybrid apps and native apps depends on the application’s requirements.
3 Method

In this chapter there is a description of the scientific method that was used in order to answer the research problem.

3.1 Scientific Approach

A survey was chosen as a scientific approach for this thesis. More specifically, it was decided to do interviews with companies that have experience in developing mobile applications. Such companies have an opinion about native and cross-platform development. This is because they need to stay competitive, thus they have to choose the most efficient way of developing applications, no matter if it is cross-platform or native.

Another reason why interviews were chosen as a scientific approach, was that there has been previous academic work about native and cross-platform frameworks, but very often these studies were performing controlled experiments [19], [20], [22], [23]. Some other studies were doing literature research [7], [14], [21]. Probably there are very few studies where researchers have done interviews with firms, and thus such studies are hard to find.

The interviews that were performed in this thesis were semi-structured. This means that there was a set of predefined questions that were asked during each meeting, but during the conversation it was possible to bring up new questions as a result of what the respondent spoke about.

3.2 Method Description

In this section there is a description on how subjects for the survey were selected, how the structure of the interview looked like and what kind of research questions were used.

3.2.1 Selection of subjects

The survey was done among developers, managers and consultants that had experience in creating mobile phone applications. Developers were the people who wrote apps themselves. Managers were the ones who managed one or more teams of programmers that develop mobile applications. Consultants were the people who had experience in mobile development and advised others on how to do create mobile apps.

The size of the company that a given person worked for did not matter. A given interviewee could be a single developer who runs a one-man business, or it could be a firm that hires several dozens of employees. The reason for this is that the firms that were interviewed, these were companies that developed applications for external clients, not for themselves. Therefore, when such a firm was building an app for a customer, the technology that was used (native or cross-platform) was determined by the client’s requirements and client’s budget, not by the company that developed the app. As a consequence of this, the firms that were interviewed could develop all types of apps. It could be a business app, e-commerce app, informational app, a game or something totally different.

The origins of companies were not taken into account. A potential interviewee could be working in a company that is from Sweden or from other countries.

3.2.2 Search of subjects

When looking for potential survey participants, only internet was used. Keywords that were associated with mobile application development were inserted into the Google search engine and then results from it (a list of websites that match the keywords) were inspected. Keywords that were used in the search engine were: iOS development, Android development, mobile app development.

The website of each company was examined in order to estimate if a given company was a good candidate for the survey. If a company made it clear on their website that they offer services which include creating mobile applications, then such company was qualifiable for the research.
Then, such businesses were called and asked for a survey. If the person who answered the phone was eligible for participating in the research, and agreed to take part in it, then the interview started right then on the phone.

If the individual who picked up the phone was not suitable for the interview and could not answer the questions (for example, if it was a receptionist) then that person was asked for contact details of somebody from his or her company who would be qualifiable for the survey. After receiving such contact information, a correct person was called. Once the right individual was on the phone, he or she was asked if she could participate in the survey. If the answer was yes, then the questions were asked.

Some companies did not publish their telephone number on their website. In such circumstances, an email was sent to these businesses, asking for a survey. If a given firm replied that they are interested in doing an interview, then a meeting time was set up and later at that meeting time the company was called.

3.2.3 Interview questions

The survey consisted of fifteen questions that were established in advance. The final amount of questions that were asked during a given survey could change. This is because a person could answer a question before it was asked. The amount of questions could be altered by other reasons. For example, if a respondent said that at his company they do not develop cross-platform apps, then some questions were skipped because that person would not know the answers to them. A respondent could be asked more than fifteen questions, in case something that he or she said was unclear, or when he or she said something intriguing. Below is a list of these fifteen predefined questions.

1. For which mobile platforms do you develop apps?
2. Which mobile apps do you develop more often: cross-platform or native?
3. When it comes to cross-platform solutions, do you develop more cross-platform apps, hybrid apps or web apps?
4. Which cross-platform frameworks have you tested and what kind of results did you get?
5. Which development method do you use, agile or waterfall?
6. Is it possible to reuse code when you develop native apps?
7. Is it possible to reuse code when you develop cross-platform apps?
8. Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?
9. How does the process of creating a new project look like at your company?
10. Is a cross-platform development process different from native development process?
11. In which situations is it better to develop cross-platform apps instead of native apps?
12. Is the response time different between cross-platform apps and native apps?
13. Is there any difference in customer experience between cross-platform and native apps?
14. Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?
15. Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

The questions were divided into three sections. The questions from one to four were introductory and their purpose was to find out what the company thinks about mobile development in general.

The questions from five to ten were there to answer the first research question, which is "How does the process and organization in cross-platform projects differ from native projects?" whereas the questions from eleven to fourteen were created in in order to answer the second research question, which is "Are there some features or qualities that are worsen or lost when developing with cross-platform frameworks instead of native?".
By "code reuse" in questions six and seven it is meant code reuse between different projects, but within the same platform.

The question fifteen was just a finalization of the interview, that allowed the interviewee to share his or her thoughts or ask a question to the interviewer.

3.2.4 Actions taken during and after the interviews

Totally, forty-two distinct firms were contacted. Among these companies, thirteen agreed to participate in the survey. After performing these thirteen interviews, no more were done, because it seemed to be a decent amount. During each interview, notes were taken. Furthermore, eleven interviews were recorded, and two were not recorded because of technical issues. Moreover, a transcript of each recorded interview was created. The notes from all interviews can be seen in the appendices from one to thirteen.

After performing all thirteen interviews, the transcripts and the notes were examined. Because all interviews had very similar (or almost the same) questions, it was possible to compare answers between different companies. By setting side by side various responses to the same questions, it was viable to create tables that present the findings.

Once the raw data from the interviews was written down, an analysis of the results was performed, in order to estimate whether the information gathered during this thesis answers the research questions.

3.3 Reliability and Validity

One reliability threat to this thesis is that not a large amount of companies were interviewed. In order to produce much more reliable results, it would be necessary to perform interviews or hand out questionnaires to a bigger quantity of companies. A satisfying amount of surveys for some researchers could be several dozens. On the other hand, this thesis was a qualitative research, meaning that, in qualitative studies a narrower span of information is captured, instead of a broader range of data. As a compensation of this, in qualitative studies researchers delve deeper into those individuals and settings that were examined [54]. Therefore thirteen interviews were considered as a decent amount.

Another reliability threat is that two interviews were not recorded. This creates a risk that the interviewer did not make notes of everything that was said.

A validity threat is that almost all companies that were interviewed were from Sweden. It could be possible that in other countries or continents, developers think otherwise than those in Sweden, due to cultural differences.

3.4 Ethical Considerations

The names of the companies and names of the individuals who were interviewed were hidden from the gathered data, in order to protect their privacy. The interviewees were aware that they are completely anonymous.
4 Results

In total, thirteen interviews were done with thirteen different companies. After performing all surveys, every answer to a given question was compared with other answers to the same question. The results of this investigation is presented in this chapter, in form of text and tables.

4.1 Data from introductory questions

The first four questions of the interview were universal and introductory. They were asked in order to find out what was a company’s opinion about mobile development in general. The results from these questions are shown in this section.

To the question "For which mobile platforms do you develop apps?" the companies responded as follows. All thirteen companies said that they develop applications for iOS. Among these thirteen firms, twelve of them also developed Android apps. Four of those companies that created Android applications, also produced Windows Phone apps. Table 4.1 shows these findings.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>13</td>
<td>100 %</td>
</tr>
<tr>
<td>Android</td>
<td>12</td>
<td>92 %</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>4</td>
<td>30 %</td>
</tr>
</tbody>
</table>

Table 4.1: Mobile platforms for which companies developed applications. One company could give multiple answers.

The question "Which mobile apps do you develop more often: cross-platform or native?" got the following responses. Seven companies said that they create both native and cross-platform applications, and that it is hard to estimate which apps they create more often. Five interviewees said that they create mostly or only native applications. One firm said that they develop only cross-platform apps. Table 4.2 presents these discoveries.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both</td>
<td>7</td>
<td>54 %</td>
</tr>
<tr>
<td>Mostly or only native</td>
<td>5</td>
<td>38 %</td>
</tr>
<tr>
<td>Only cross-platform</td>
<td>1</td>
<td>8 %</td>
</tr>
</tbody>
</table>

Table 4.2: Types of applications that companies developed. One company could give only one answer.

The question "When it comes to cross-platform solutions, do you develop more cross-platform apps, hybrid apps or web apps?" received the following answers. Among all thirteen companies, four of them said that they do not use any cross-platform solutions, and that they develop only native apps. Another four firms said that they use cross-platform frameworks, and five companies said that they do not use cross-platform frameworks, but that they create hybrid apps or web apps and use that as their cross-platform solution. Table 4.3 shows the answers to this question.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both</td>
<td>7</td>
<td>54 %</td>
</tr>
<tr>
<td>Mostly or only native</td>
<td>5</td>
<td>38 %</td>
</tr>
<tr>
<td>Only cross-platform</td>
<td>1</td>
<td>8 %</td>
</tr>
</tbody>
</table>

Table 4.3: Cross-platform solutions and types of applications. One company could give only one answer.
Answers to the question "Which cross-platform frameworks have you tested and what kind of results did you get?" varied. Most companies said that they tried out cross-platform development. If a company did try using cross-platform tools, they usually tested more than one framework. Table 4.4 shows which cross-platform tools all companies together mentioned.

The most popular frameworks were Xamarin, Appcelerator Titanium and Apache Cordova, which were tested by five, four and three companies respectively. Two respondents said that their firm examined cross-platform frameworks, but that it was a long time ago, and that they do not remember the names of these tools. Three companies said that they have never tested any cross-platform frameworks.

Once a responded said which cross-platform tools they have examined, he or she had to say what kind of results they received when using these frameworks. Table 4.5 shows the answers to that question. Most people just replied briefly that they were satisfied or not satisfied with a given tool. Four people gave not clear answers about their results. Some of them said that everything depends on client’s requirements, others that it was not themselves who personally examined a given framework, and that they do not know the details.
Table 4.5: Companies’ satisfaction from using cross-platform tools. One company could give only one answer.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not satisfied</td>
<td>5</td>
<td>38 %</td>
</tr>
<tr>
<td>Satisfied</td>
<td>4</td>
<td>31 %</td>
</tr>
<tr>
<td>Unknown</td>
<td>4</td>
<td>31 %</td>
</tr>
</tbody>
</table>

Only two respondents said which frameworks they have tested for creating web applications. One person said that their company examined Angular, and that they still use it. Another respondent said that their firm tested MXF and NGiNX, but the results were not as satisfactory as native.

4.2 Differences between cross-platform projects and native projects

In this section there are results from the interview questions from five to ten. The purpose of these questions was to answer the first research question of this thesis, which is: "How does the process and organization in cross-platform projects differ from native projects?"

To the question "Which development method do you use, agile or waterfall?", most companies replied similarly. Ten firms said that they use agile development, or a method that is based on agile development, for example Scrum, or that they use their own iterative method that is founded on agile. The remaining three businesses said that they do not use any development method. One person gave such response because his business was a consultancy company, that only advises during app development. The other two people said that they were a one-man business and therefore they did not need to have any development method. Table 4.6 shows results received from this question.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agile</td>
<td>10</td>
<td>77 %</td>
</tr>
<tr>
<td>None</td>
<td>3</td>
<td>23 %</td>
</tr>
</tbody>
</table>

Table 4.6: Development methods used by companies. One company could give only one answer.

Responses to the question "Is it possible to reuse code when you develop native apps?" were as follows. Nine respondents said that it is possible to reuse code in native development, but only within the same platform. Two people stated that it is not doable to reuse code in native. Two other people declared that they are not sure and therefore they cannot give precise answer to this question. Table 4.7 depicts these responses.
Table 4.7: Companies’ opinion about if it is possible to reuse code in native development. One company could give only one answer.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>9</td>
<td>70 %</td>
</tr>
<tr>
<td>No</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>Not sure</td>
<td>2</td>
<td>15 %</td>
</tr>
</tbody>
</table>

Some of the respondents who said that it is possible to reuse code in native, motivated their answers by giving examples. One person said that integration and structural parts of apps can be reused. Someone else said that architectural fragments of the application can be reused, but this requires planning. Another person stated that backend code of apps could be reused.

The next question "Is it possible to reuse code when you develop cross-platform apps?" received the following answers. Ten people said that it is viable to reuse code when developing cross-platform applications. Some respondents gave examples of components that could be reused, for example login form or backend code. Three people did not give any answer, because their company did not test cross-platform solutions, thus they could not give an honest reply to this question. The answers to this question are displayed in table 4.8.

Table 4.8: Companies’ opinion about if it is possible to reuse code in cross-platform development. One company could give only one answer.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>10</td>
<td>77 %</td>
</tr>
<tr>
<td>Not sure</td>
<td>3</td>
<td>23 %</td>
</tr>
</tbody>
</table>

Replies to the question "Do you have different programmers for each platform or do you have same programmers for all platforms?" varied. Table 4.9 shows the answers to this question.

Four respondents stated that their firm uses same developers for all platforms. Three people said that their company uses different programmers for each platform, and that each programmer works only within his or hers platform. Three people said that they have programmers that are specialized in one platform, but they sometimes ask their developers to go outside of their area of expertise, in order to help their colleagues that create applications for another platform. Two people told that their company is a one-man business, thus they are the only developer at their firm. One person did not give any precise answer to this question.

Table 4.9: Companies’ opinion about if they have different programmers for each platform or do they have same programmers for all platforms?

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>1</td>
<td>100 %</td>
</tr>
<tr>
<td>No</td>
<td>2</td>
<td>40 %</td>
</tr>
<tr>
<td>Not sure</td>
<td>4</td>
<td>60 %</td>
</tr>
</tbody>
</table>
Replies to the question "How does the process of creating a new project look like at your company?" were similar to each other. All respondents answered by describing such process in a few steps. These steps were:

Phase 1: A potential client contacts the company and tells what kind of mobile application he or she would like to be created.

Phase 2: This step is about analysis and planning. During this phase, companies gather requirements and scope what the application should do. They check complexity and difficulty of the app. They do a technical analysis and try to get a clear view of the mobile application that they are supposed to produce. They create scenarios. Some companies give at this stage a price proposal to the potential customer.

Phase 3: The development company sketches and plans the app. They design the application and create wireframes.

Phase 4: A prototype or a simple version of the app is created and presented to the client. The prototype should show approximately how the final application will look like.

Phase 5: Develop the application. Most companies use an iterative (agile) approach during this stage.

Phase 6: When the application is created, the company publishes the app or assists the customer with publishing it.

The answers to the question "Is a cross-platform development process different from native development process?" were as follows. Six people said that there is no difference. Two people thought that cross-platform development process is faster than native development process. One person stated that there are small differences when planning the app. Five people did not give any answer to this question. This is because either their company did not develop cross-platform apps or because they personally were not dealing with cross-platform frameworks in the past, and that it was somebody else at their company who tested cross-platform tools. Table 4.10 shows a summary of responses to this question.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Same programmers for all platforms</td>
<td>4</td>
<td>31 %</td>
</tr>
<tr>
<td>Different programmers for all platforms</td>
<td>3</td>
<td>23 %</td>
</tr>
<tr>
<td>Different programmers, but allow mixing between platforms</td>
<td>3</td>
<td>23 %</td>
</tr>
<tr>
<td>Single developer (one-man business)</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>No answer</td>
<td>1</td>
<td>8 %</td>
</tr>
</tbody>
</table>

Table 4.9: How companies organize the work of their programmers between different platforms. One company could give only one answer.
Table 4.10: Companies’ answers to whether a cross-platform development process is the same as native development process. One company could give multiple answers.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>The process is the same</td>
<td>6</td>
<td>46 %</td>
</tr>
<tr>
<td>No answer</td>
<td>5</td>
<td>38 %</td>
</tr>
<tr>
<td>Cross-platform process is faster</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>Small differences in planning the app</td>
<td>1</td>
<td>8 %</td>
</tr>
</tbody>
</table>

4.3 Comparison of native apps and cross-platform apps

In this section there are results from the interview questions from eleven to fourteen. The purpose of these questions was to answer the second research question of this thesis.

Responses to the question "In which situations is it better to develop cross-platform apps instead of native apps?" varied. Nearly all respondents replied by giving a handful of different causes about when it is a good choice to develop a cross-platform app.

The most often mentioned reason was client’s low budget, it was brought up by five respondents. Other reasons that were referenced frequently by companies were: in situations where the app is supposed to be simple or informational, when it is necessary to save time, when it is important to develop an application quickly or that everything depends on customer’s requirements.

Two respondents said that they would at all times develop native applications, and two other companies said they always recommend cross-platform development. One man said that an application can be developed as cross-platform, when it is supposed to receive updates very often. One person did not give a clear reply. Table 4.11 that is on the next page summarizes the answers to this question.
Table 4.11: Reasons when it is better to develop cross-platform applications. One company could give multiple answers.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low budget</td>
<td>5</td>
<td>38 %</td>
</tr>
<tr>
<td>Save time and develop app quickly</td>
<td>3</td>
<td>23 %</td>
</tr>
<tr>
<td>Informational app</td>
<td>3</td>
<td>23 %</td>
</tr>
<tr>
<td>Depends on the requirements</td>
<td>3</td>
<td>23 %</td>
</tr>
<tr>
<td>Simple app</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>Always native</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>Always cross-platform</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>Internal company app</td>
<td>1</td>
<td>8 %</td>
</tr>
<tr>
<td>App that is updated often</td>
<td>1</td>
<td>8 %</td>
</tr>
<tr>
<td>No answer</td>
<td>1</td>
<td>8 %</td>
</tr>
</tbody>
</table>

Table 4.11: Reasons when it is better to develop cross-platform applications. One company could give multiple answers.

The next question, "Is the response time different between cross-platform apps and native apps?" received the following replies. Five people said that there is a difference, and that native applications are faster than cross-platform apps. Three respondents pointed out that native is faster, but the difference is very small. Two people said that it depends on what the application does, and two others indicated that there is no difference. One person did not give any precise reply to this question. The table 4.12 shows a summary of these answers.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes, native is faster</td>
<td>5</td>
<td>39 %</td>
</tr>
<tr>
<td>Yes, native is faster, but difference is small</td>
<td>3</td>
<td>23 %</td>
</tr>
<tr>
<td>It depends on what the app does</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>No, there is no difference</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>No answer</td>
<td>1</td>
<td>8 %</td>
</tr>
</tbody>
</table>

Table 4.12: Companies’ opinions about if there is a difference in response time between native and cross-platform applications. One company could give only one answer.

Answers to the question "Is there any difference in customer experience between cross-platform and native apps?" were as follows. Nine people said that native applications give better user experience. Three people said that they do not see any distinctions.
Table 4.13: Companies’ opinion about if there is a difference in customer experience between native and cross-platform applications. One company could give only one answer.

Some of these ten respondents who said that they do see a difference in customer experience, also expanded their answers and explained why they think like that. They gave reasons to support their opinion. Table 3.14 shows those reasons. The most popular motive was that native apps are faster. This reason was brought up by six people. The second motive was that graphical user interface (GUI) in cross-platform apps does not look and feel as good as GUI of native applications. This reason was brought up five times.

Table 4.14: Support claims of these companies who did see a difference in customer experience between native and cross-platform apps. One company could give multiple answers.

One person who said that their company does not see any distinctions in customer experience, between cross-platform and native, supported that claim by saying that this can be achieved only by using Xamarin framework.

The question "Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?” got various answers. Table 4.15 shows all differences that respondents talked about.

A reply that was spoken out most often, was that cross-platform development, testing and maintenance is faster and simpler. This reply was raised eight times. The second most popular response, that was brought up four times, was that in native development a programmer has better access to phone’s functionality, hardware and API. Three people said that in cross-platform development there is only one code base, whereas in native programming it is not possible. Two respondents said that native development requires more knowledge, because then a developer needs to know two separate programming languages, and two separate programming environments. Thus, the threshold to get into native development is higher.

Two people responded by saying that recruitment of programmers is different between native and cross-platform. The first person motivated his answer by saying that sometimes it might be easier to find programmers with one competence than the other. The second person supported his claims by saying that it is easier to find C# developers, and such programmers can use the cross-platform framework Xamarin.

Two respondents did not give any precise answers. One person said that their company does not see any differences when accessing functionality, hardware or API with cross-platform frameworks.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Native gives better user experience</td>
<td>10</td>
<td>77 %</td>
</tr>
<tr>
<td>There is no difference</td>
<td>3</td>
<td>23 %</td>
</tr>
</tbody>
</table>

Table 4.13: Companies’ opinion about if there is a difference in customer experience between native and cross-platform applications. One company could give only one answer.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI of cross-platform apps does not look good</td>
<td>5</td>
<td>38 %</td>
</tr>
<tr>
<td>Native apps are faster</td>
<td>6</td>
<td>46 %</td>
</tr>
</tbody>
</table>

Table 4.14: Support claims of these companies who did see a difference in customer experience between native and cross-platform apps. One company could give multiple answers.

One person who said that their company does not see any distinctions in customer experience, between cross-platform and native, supported that claim by saying that this can be achieved only by using Xamarin framework.

The question "Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?” got various answers. Table 4.15 shows all differences that respondents talked about.

A reply that was spoken out most often, was that cross-platform development, testing and maintenance is faster and simpler. This reply was raised eight times. The second most popular response, that was brought up four times, was that in native development a programmer has better access to phone’s functionality, hardware and API. Three people said that in cross-platform development there is only one code base, whereas in native programming it is not possible. Two respondents said that native development requires more knowledge, because then a developer needs to know two separate programming languages, and two separate programming environments. Thus, the threshold to get into native development is higher.

Two people responded by saying that recruitment of programmers is different between native and cross-platform. The first person motivated his answer by saying that sometimes it might be easier to find programmers with one competence than the other. The second person supported his claims by saying that it is easier to find C# developers, and such programmers can use the cross-platform framework Xamarin.

Two respondents did not give any precise answers. One person said that their company does not see any differences when accessing functionality, hardware or API with cross-platform frameworks.
Table 4.15: Companies’ opinions about differences between native and cross-platform development, from a programmer’s perspective. One company could give multiple answers.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cross-platform development, testing and maintenance is faster and simpler</td>
<td>8</td>
<td>66 %</td>
</tr>
<tr>
<td>Native has better access to phone’s functionality, hardware or API</td>
<td>4</td>
<td>31 %</td>
</tr>
<tr>
<td>One code base in cross-platform</td>
<td>3</td>
<td>23 %</td>
</tr>
<tr>
<td>Higher threshold, more knowledge needed in native development</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>Recruitment of programmers is different</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>No answer</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>No differences in access to functionality, hardware and API</td>
<td>1</td>
<td>8 %</td>
</tr>
</tbody>
</table>

Table 4.15: Companies’ opinions about differences between native and cross-platform development, from a programmer’s perspective. One company could give multiple answers.

4.4 Other results

During the interviews, respondents said whether they prefer native or cross-platform development. Some of them stated clearly that they like one of these two development methods more, others said that they are not really sure which one to prefer. Most often these statements were an official statement of what their company thinks.

Out of thirteen respondents, nine of them preferred native development and two of them liked better cross-platform development. The remaining two people did not say clearly which one they prefer. In order to motivate their uncertainty, they said that everything depends on a few factors, like client’s budget, project type or application type. The table 4.16 displays how many firms preferred each development type.

Table 4.16: Companies’ preferences between native and cross-platform. One company could give only one answer.

<table>
<thead>
<tr>
<th>Answer</th>
<th>Number of answers</th>
<th>Percent of respondents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prefer native</td>
<td>9</td>
<td>70 %</td>
</tr>
<tr>
<td>Prefer cross-platform</td>
<td>2</td>
<td>15 %</td>
</tr>
<tr>
<td>Not sure</td>
<td>2</td>
<td>15 %</td>
</tr>
</tbody>
</table>
5 Analysis

In this chapter there is an analysis of the results that were shown in the third chapter. Conclusions and meanings that can be drawn from the data is presented.

5.1 Analysis of questions 1 - 4

In this segment the following interview questions are analyzed:

1. For which mobile platforms do you develop apps?
2. Which mobile apps do you develop more often: cross-platform or native?
3. When it comes to cross-platform solutions, do you develop more cross-platform apps, hybrid apps or web apps?
4. Which cross-platform frameworks have you tested and what kind of results did you get?

From the interview results it is clear that Android and iOS were the most popular platforms, for which companies developed apps. Only a handful of firms created Windows Phone applications. This is most likely caused by the fact that Windows Phone market share is very small [1].

A vast majority of the companies that were interviewed said that they have tested cross-platform frameworks, but only four of them said that they were satisfied with the results and still use these cross-platform tools. It is hard to estimate why it is so, because it is unknown what kind of applications explicitly the interviewed companies were creating. Some companies might develop apps that are more dependent on hardware aspects, such as sensors or GPS. Other firms might develop more games or informational apps. In order to estimate with certainty whether cross-platform frameworks give satisfactory results or not, it would be necessary to find out what apps the various companies were building.

A few firms could not point out precisely if they were satisfied with cross-platform frameworks or not. This could be caused by the fact that cross-platform frameworks have their positive and negative sides. Cross-platform development is faster and it lets businesses save money. It also allows them to offer lower prices to their customers, which in turn could result in more orders. Cross-platform frameworks make updating and fixing bugs easier, because there is only one code base. On the other hand, those firms wanted to offer products with as high quality as possible to their clients. Taking all of these points into consideration, it is understandable that for some people the choice between native and cross-platform is not an easy one.

Another interesting fact is that some companies used hybrid apps or web apps as their cross-platform solution. This could mean that if these firms tested cross-platform frameworks, then these businesses were not fond of them. Therefore they decided to continue or switched to using hybrid apps or web apps.

5.2 Analysis of questions 5 - 10

In this section the following interview questions are investigated:

5. Which development method do you use, agile or waterfall?
6. Is it possible to reuse code when you develop native apps?
7. Is it possible to reuse code when you develop cross-platform apps?
8. Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?
9. How does the process of creating a new project look like at your company?
10. Is a cross-platform development process different from native development process?

According to most companies that were interviewed, code in both native and cross-platform applications can be reused. When it comes to native, reusing the code is not so simple as in cross-
platform. It requires more planning, but it is still viable. In native development, only certain components can be reused (like architecture of the app). The logic or graphical interface of applications cannot be reused. Moreover, reuse is possible only within the same platform. In cross-platform development, code reuse is easier, because in cross-platform programming it is possible to reuse not only architecture or backend code, but graphical components as well (for example login form or registration form).

The amount of code that can be reused in a cross-platform app, probably depends on the chosen cross-platform solution. The company that develops Xamarin says that with their tool it is viable to reuse from 75% to almost 100% of code [8]. Appcelerator claims that in Titanium it is possible to reuse from 60% to 90% of the code [9]. In hybrid apps only the native layer needs to be adapted for each platform, while the web layer of the app can be fully reused across every platform [23]. When it comes to web apps for mobile devices, their code reuse is usually 100% on all mobile platforms, because these applications are just a web app that is adjusted to all screen sizes.

The process of creating applications looked almost the same at every company. Each business went through very similar phases, from planning and designing the app, through developing it and publishing it. Moreover, this process seemed to be almost the same when developing native applications and cross-platform applications. There were some minor differences, for example cross-platform development was faster than native. The reason for this is that in cross-platform programming there is only one code base, and the programmers have to develop the application only once, and they do not need to do it twice or more times. Native development needs more resources in form of time. Native programming requires also more knowledge about distinct programming environments.

Additionally, creating a cross-platform app requires a bit different planning. This is because in cross-platform, programmers usually have to design an application in such a way, so that the app looks good on all mobile operating systems. This is different in native, because when planning an app, a programmer has to design an application so that it looks and feels well on a specific platform, not on all platforms.

Furthermore, most companies used agile development, or a method that was similar to agile (two companies stated that they used scrum). Some other companies said that they have their own method that was based on agile, and a small portion of firms stated that they do not have any special development methodology. The companies which said that they do not use any development method, were small firms, usually one-man businesses. This suggests that when a company is very small (for example it has only one developer) then such firms do not have any need for using a development method.

None of the companies that were interviewed admitted that they used waterfall methodology, spiral methodology or any other methodology. This could suggest that agile is the most preferred development method when building mobile applications.

5.3 Analysis of questions 11 - 14

Inside this segment these interview questions are analyzed:

11. In which situations is it better to develop cross-platform apps instead of native apps?
12. Is the response time different between cross-platform apps and native apps?
13. Is there any difference in customer experience between cross-platform and native apps?
14. Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?

According to the vast majority of companies that were interviewed, cross-platform solutions were better than native only in special cases. If a business was supposed to create an app that is cheap and the quality was not as important as low price, then it was a good choice to use cross-platform solutions.
If a mobile program was supposed to be an informational app, or simple app, then such application could be develop with cross-platform frameworks. An informational app in this case is something that only presents information to the user, in form of text and images. Thus, such mobile applications are not very sophisticated or hardware demanding. On the other hand, if a company had to create an advanced and hardware demanding app, or an application that had to have high quality, then it was better to choose native development.

As stated by most companies, native applications have lower response time and they are generally faster. Perhaps this difference in response time is small, or this distinction might depend on what the application actually does, but it exists and it might discourage some people from using cross-platform frameworks.

Native applications also have better user experience. Their graphical user interface can be adjusted to a particular platform, which makes them look nicer and they feel like a part of the phone. This cannot be achieved so easily in a cross-platform environment. A cross-platform application needs to be adapted for all mobile operating systems. Because of this, cross-platform apps may look sometimes like a website, or like a web app and not like a real, native program.

One thing to bear in mind, is that there could be some companies out there which would prefer to have an app that looks identically on all platforms. Such approach could be used by these companies that want to create their own look, that is universal across all platforms. It might be easier to achieve such effect by creating a web app, hybrid app or cross-platform app.

According to the interviewed companies, developing a cross-platform application is easier and faster. A company has then only one code base, which makes maintenance and testing easier as well. Native development requires more knowledge and more work. It is also more difficult to find programmers that know how to create applications for more than one platform. It could be easier to find people who are good at only C# or only Javascript, and thus can build cross-platform apps.

5.5 Summary of the analysis

Both cross-platform and native have their positive and negative aspects. Tables 5.1 and 5.2 present a summary comparison of native and cross-platform.

Only one difference between native and cross-platform is not mentioned in tables 5.1 and 5.2. It is access to phone’s functionality, hardware and API. During the survey, four respondents said that in native development programmers have better access to these features, whereas two people said that access to these aspects is the same. These numbers are too close to each other, therefore it is not possible to say for certain who is right.

Moreover, the creators of Xamarin [8] and Appcelerator Titanium [9] claim that their tools have 100% access to platforms’ native API. This means that it is possible to use hardware features or at least some parts of the hardware features with help of their cross-platform frameworks.

Furthermore, in chapter 2 of this report there is a review of previous studies, and one of these studies found that certain cross-platform frameworks do indeed have full access to mobile platforms’ native API [7].

When taking all these points into consideration, it seems that certain cross-platform frameworks do have full access to platforms’ API. Therefore, the developers who would like to use a cross-platform tool, need to first make sure that their framework of choice gives the possibility of full access to native API.
Cross-platform applications

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low cost</td>
<td>Worse response time</td>
</tr>
<tr>
<td>Faster development</td>
<td>Worse graphical user interface</td>
</tr>
<tr>
<td>One code base</td>
<td></td>
</tr>
<tr>
<td>Low threshold</td>
<td></td>
</tr>
<tr>
<td>Staff recruitment easier</td>
<td></td>
</tr>
</tbody>
</table>

Table 5.1 Advantages and disadvantages of cross-platform development for mobile platforms, when compared to native development.

Native applications

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Better response time</td>
<td>High cost</td>
</tr>
<tr>
<td>Better graphical user interface</td>
<td>Slower development</td>
</tr>
<tr>
<td></td>
<td>Separate code bases</td>
</tr>
<tr>
<td></td>
<td>High threshold</td>
</tr>
<tr>
<td></td>
<td>Staff recruitment harder</td>
</tr>
</tbody>
</table>

Table 5.2 Advantages and disadvantages of native development for mobile platforms, when compared to cross-platform development.

The biggest advantage of cross-platform is its low costs. Other benefits of cross-platform is faster development and lower threshold for gaining knowledge that is needed to enter mobile programming. The disadvantages are lower response time and worse graphical user interface, that is, lower quality in general. When it comes to native applications, everything is the opposite. Native apps have higher quality, but they are more expensive.

Despite the fact that native programming has more disadvantages than advantages, the benefits that native gives are much more powerful than the benefits provided by cross-platform tools. Because of this, most companies choose native development.
6 Discussion

In this chapter there is a discussion about the findings of this thesis, and if the research questions of this thesis were answered.

6.1 Discussion about previous research

There have been several studies in the past that examined the same problem as this thesis. Most previous academic work performed controlled experiments, and some others did literature reviews. Unfortunately it is very hard to find such studies where qualitative data was gathered. Therefore in this thesis, surveys among companies were carried out, in order to examine this problem from a different angle than before. Another thing that differentiates the research in this thesis from other, preceding studies, is that previous academic work did not examine the differences between cross-platform projects and native projects.

Academic work done in the past showed that that native apps usually have higher quality and that native apps are a better choice when performance, reliability, developer support and functionality is an important factor.

Previous research also pointed out that cross-platform frameworks have their strengths and weaknesses. Its strengths are simpler development, one codebase and lower knowledge threshold. Its weakness was worse performance.

Preceding studies pointed out that it is important to remember that the choice of development approach depends heavily on the context and on the requirements. They showed that there are many factors that come into play when choosing between native and cross-platform solutions. In certain context a cross-platform solution might give the same performance and quality as a native app, whereas in some other cases native apps outperforms cross-platform applications.

The research that was carried out during this thesis confirms what other researchers have found. Cross-platform solutions are still not mature enough to fully replace native development. Because of this, each project should be evaluated individually, whether to use cross-platform solutions or native solutions, because each project will have different context and business needs.

Because previous academic work, that was shown in chapter 2, did not investigate the differences between cross-platform projects and native projects, it is not possible to compare this part of results from this thesis with other academic studies.

6.2 Answers to research questions

The results from the thirteen surveys that were conducted give some answers to the research questions of this thesis. Below is first research question:

*RQ 1: How does the process and organization in cross-platform projects differ from native projects?*

The answer to this question is the following. According to a larger part of the companies that were interviewed, cross-platform projects and native projects do differ, but the differences are small. Cross-platform projects require a little bit different planing at the inception of the project. Moreover, the development during these projects is faster, easier and simpler. But differences end there. The overall project structure and workflow stays the same, which looks as follows.

First, a client contacts a development firm. Second, the development company gathers requirements and does a technical analysis. Third, the company designs the application. Forth, the company creates a prototype and shows it to the customer. Fifth, the firm produces the app, usually by using agile development method. Sixth and last, the application is published.
RQ 2: Are there some features or qualities that are lost or worsen when developing with cross-platform frameworks instead of native?

The answer to this research question is as follows. According to the majority of the companies that were interviewed, when using cross-platform frameworks, some small qualities are worsen. The most important one is the speed of the application. Cross-platform apps may in some cases have higher response time. Another quality that is worsen, is that graphical user interfaces of cross-platform apps may not have a good look and feel. This in turn affects the user experience. One thing to bear in mind though is that this is just an opinion of the companies that were interviewed. It is possible that if an another subset of companies were examined, they would respond differently. To be fully sure, what is companies‘ opinion about native and cross-platform, a wider range of firms should be interviewed. Another thing to remember is that a lot depends on the requirements of the app that is to be created. Some applications could be built with cross-platform solutions, and that would not decrease their overall quality, whereas in some situations when high performance and quality is necessary, then it is better to use native development.
7 Conclusion

In this chapter there is an overall conclusion and summary of the whole thesis. There is also some information about for whom this report could be useful. Furthermore, there is some information about possible future research.

7.1 General conclusion

The conclusions from this report are the following. Both native and cross-platform have their advantages and disadvantages. In the light of the data that was gathered during this thesis, native has stronger advantages than cross-platform frameworks. Most of the companies that were interviewed preferred native development and recommended it to their customers.

According to some companies, native applications are faster and provide better user experience. This compensates the fact that they are more expensive. Currently, cross-platform frameworks do not live up to the expectations of some companies and do not provide as high quality as native apps. On the other hand, cross-platform tools allow to create applications with lower cost, and the development of these apps is faster and simpler. Therefore, the choice between cross-platform and native depends on several factors, like: complexity of the app, budget that can be spent on the app, desired quality, etc.

Furthermore, most companies that were interviewed preferred agile development methodology, and a few stated that they do not have any development method.

Other researchers have done similar studies in this field. Their results also showed that native apps have higher quality than cross-platform apps. Previous studies also point out that when choosing a development approach, each project should be evaluated individually, whether to use native or cross-platform, because each context is different.

The information from this report is valuable to individual programmers and companies that create applications for smartphones. If quality is an important issue for a customer, then the programmer or company should choose native development. However, if the price is a major concern for that client, then cross-platform apps could be a better option.

Individuals and firms that want to use services of companies that create apps, could also use the information in this report. It could protect them from being falsely persuaded. Some companies might offer creating cross-platform applications, and they might market these apps as if they were as good as native ones. Currently, there is still a gap between cross-platform applications and native apps. Any company that would claim otherwise would be exaggerating.

The knowledge from this report can also be applied by single developers and companies that create their own applications, for their own needs.

There is one thing that could be done differently, that could deliver better results for this thesis, and that thing is: better follow up questions during the interviews. Some respondents did not give precise answers or their answers were too short. Proper follow up questions would gently push the interviewees into giving more details, which in turn would increase the quality of gathered data.

7.2 Future Research

Future research in this area could examine the same topic, but on a larger scale. In this thesis, only thirteen companies were interviewed. A bigger amount of interviewed companies would increase the trustworthiness of results.

Similar research could be done in other countries, in order to check if cultural differences could affect a company’s choice between cross-platform and native.

An inspection and comparison of companies’ culture, size and structure could also help researchers in estimating if there are some other, hidden aspects that influence the choice between native and cross-platform technologies.
References


Microsoft to acquire Xamarin and empower more developers to build apps on any device. [Online]. Available: http://blogs.microsoft.com/blog/2016/02/24/microsoft-to-acquire-xamarin-and-empower-more-developers-to-build-apps-on-any-device/#sm.00015xwaky15fiddbeuclexxd4qgpbu


Appendix 1

This appendix contains notes from an interview with the first company.

For which mobile platforms do you develop apps?

Android and iOS.

Which mobile apps do you develop more often: cross-platform or native?

Native.

When it comes to cross-platform solutions, do you develop more cross-platform apps, hybrid apps or web apps?

We have developed web apps in the past. We used a few different technologies and frameworks, but we haven’t found any good ones that we would like to go further with. We haven’t been able to have a project with any newer frameworks yet, like React. We’re waiting to get into such projects to see what we can do with that.

Which cross-platform frameworks have you tested and what kind of results did you get?

We tested Sencha Touch as a web framework for hybrid development. We tested Titanium, and I don’t remember the names of the other ones, it was a while ago and we weren’t fond of them. They weren’t really what we needed.

Which development method do you use, agile or waterfall?

Agile.

Is it possible to reuse code when you develop native apps?

Yes, definitely. If it’s an integration part of the app, or some kind of structural part of the app that we use often we create a library for it. Like Java libraries or Cocoa Touch libraries.

Is it possible to reuse code when you develop cross-platform apps?

We haven’t developed cross-platform apps for a while but I’m sure you’re able to create libraries there as well.

Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?

We have different developers focused on their platform. We would like to find developers who have broad competence in many different areas, whether it be databases, Android or iOS. They should have focus and deep knowledge on one thing, e.g. iOS, and they shouldn’t be afraid of going outside their area of expertise, in case we need resources elsewhere.
How does the process of creating a new project look like at your company?

Usually the client comes to us with an idea or a concept. We talk to them on how we can solve it and what resources we need. If we come to terms to develop an app with the client, we start with wireframes and design, after we scoped what needs to be done within the app. Then we show it to the customer so that they can see how the product will look like when it’s done. When we have enough content and feel comfortable to start development, we give it to our developers. They break it down into tasks and start developing. Usually the client is a participant in all these steps. We prefer the client to talk directly to the developers, so that the client can see how the project goes along, what comes out of it and can make changes during the development. This process is the same for native development and cross-platform development.

Is a cross-platform development process different from native development process?

Well, I can’t really tell you because we didn’t have cross-platforms projects for a while, but I don’t think there’s a big difference in time. Cross-platform apps should be compile once, run everywhere on different platforms, but a thing that we noticed was that if you’re going to do something that is special and not within the scope of cross-platform development, you will have to test it on all platforms anyway. So it won’t save you that much time.

In which situations is it better to develop cross-platform apps instead of native apps?

I would always recommend doing native development. It will take somewhat more time, but it will be a lot better packaged. Cross-platform apps are quite bloated. It will be very big in size, even if it’s small. You will always lose something because you have a layer in between. In native you have higher threshold to get into it. You need more knowledge, ins and outs of a platform but you will get closer to the hardware. So there are a lot of pros to go native.

Is the response time different between cross-platform apps and native apps?

I think native apps have better response time. They are pretty good at doing what they are supposed to do.

Is there any difference in customer experience between cross-platform and native apps?

I would say native gives better experience. If you develop a cross-platform app you will also have good customer experience, however not all scenarios that we have from our clients fit within that description, so that’s why we haven’t gone cross-platform all the way.

Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?

I think I mentioned all differences.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

Regarding cross-platform and native, it will be fun to see what will happen in the future with the cross-platform frameworks like React Native and the other ones.
But as with anything, if you make such a large step from the original platform or original thought, then you will take something away, and it won’t do any good.

With the languages that we have, Java and mostly Swift, they are not hard to learn. Before Swift we had Objective-C which was a little harder to learn, so it was a bigger argument to use cross-platform, because the threshold was higher to get into Objective-C. But now languages are relatively easy to learn, there’s no use for cross-platform for us at least.
Appendix 2

This appendix contains notes from an interview with the second company.

**For which mobile platforms do you develop apps?**

Android and iOS.

**Which mobile apps do you develop more often: cross-platform or native?**

Native.

**When it comes to cross-platform solutions, do you develop more cross-platform apps, hybrid apps or web apps?**

Web apps.

**Which cross-platform frameworks have you tested and what kind of results did you get?**

We don’t use any cross-platform frameworks that transform code into native code. We develop web apps without any special frameworks.

**Which development method do you use, agile or waterfall?**

Agile.

**Is it possible to reuse code when you develop native apps?**

Yes, between different projects but within the same platform.

**Is it possible to reuse code when you develop cross-platform apps?**

Yes.

**Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?**

Same programmers for all platforms.

**How does the process of creating a new project look like at your company?**

First we go through requirements, after that we give a price proposal to the customer. If they accept it we start the development. We work in sprints, we divide the work into parts. After each sprint we show the results to our customer. If they are satisfied then we go to the next step. When the app is ready we help the customer to publish the application to App Store and Google Play.

**Is a cross-platform development process different from native development process?**

No.
In which situations is it better to develop cross-platform apps instead of native apps?

We can develop a web app if it should be an informational app that doesn’t have to feel like a native app. Or if it’s a simple app or when the low price is very important to the customer then we can develop a web app. But in most situations we recommend native apps.

Is the response time different between cross-platform apps and native apps?

Yes, definitely. In native you develop in a language that is planned for a given platform, and when you develop a cross-platform app you have a middle layer that translates one code into native code. That conversion of code will not make the app faster, only slower.

Is there any difference in customer experience between cross-platform and native apps?

Yes, for sure. Let’s take GUI as an example. In cross-platform development, you will create something that looks like iOS on both iOS and Android, or you will get something that looks like Android on both iOS and Android, or you will build something that looks like neither iOS or Android. If you have skilled designers then you will probably be able to make it work, but users on each platform are usually picky and they are used to how a native application looks and feels like. If you won’t reach to that level you might get unsatisfied customers.

Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?

Well, from my perspective, a person who recruits programmers, then this can vary. The choice between cross-platform and native might depend on how easy or difficult it is to find people with the right competence. From a programmers perspective it depends on which environment you prefer. There are cross-platform frameworks that use Javascript for example. Some programmers like it, but I personally think that it is a language that tends to create messy code. I prefer more object oriented programming languages. For that Swift is very good. We are impressed by it. Java is also good.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.
Appendix 3
This appendix contains notes from an interview with the third company.

For which mobile platforms do you develop apps?

Android and iOS.

Which mobile apps do you develop more often: cross-platform or native?

Both.

Which cross-platform frameworks have you tested and what kind of results did you get?

We use Appcelerator Titanium, and the results are good.

Which development method do you use, agile or waterfall?

We don’t use any special method.

Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?

Different programmers.

How does the process of creating a new project look like at your company?

We first do a technical analysis of what the customer wants. We design the app and create a prototype so that the client can see how the app will look like in the end. After that we develop the app.

Is a cross-platform development process different from native development process?

In general it is the same.

In which situations is it better to develop cross-platform apps instead of native apps?

Everything depends on the client’s budget and on how advanced the app should be. We do a technical analysis in order to estimate if an app could be built with cross-platform frameworks and also on client’s budget.

Is the response time different between cross-platform apps and native apps?

Yes. There is a difference. Native apps are a bit faster, cross-platform apps are a bit slower. This difference is not so big but it’s there.

Is there any difference in customer experience between cross-platform and native apps?

I don’t think so.
Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?

Speed. It’s just faster to develop cross-platform apps.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.
Appendix 4
This appendix contains notes from an interview with the fourth company.

For which mobile platforms do you develop apps?
Android and iOS.

Which mobile apps do you develop more often: cross-platform or native?
Only native.

Which cross-platform frameworks have you tested and what kind of results did you get?
We tested a few but we decided to not use them.

Which development method do you use, agile or waterfall?
Agile.

Is it possible to reuse code when you develop native apps?
Yes, sometimes.

Is it possible to reuse code when you develop cross-platform apps?
I think that it is possible.

Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?
Same programmers.

How does the process of creating a new project look like at your company?
We get requirements, create wireframes, then we create design and we work according to the scrum model.

Is a cross-platform development process different from native development process?
Developing cross-platform is faster, but the quality of the app is worse than when using native.

In which situations is it better to develop cross-platform apps instead of native apps?
We always recommend native.

Is the response time different between cross-platform apps and native apps?
Yes, but it was a very small difference.
Is there any difference in customer experience between cross-platform and native apps?

We thought that the user experience wasn’t so good.

Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?

No.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.
Appendix 5

This appendix contains notes from an interview with the fifth company.

Could you tell a bit about your company?

I run my own one-man business. Currently it’s not a full-time job. Right now I go on a training course so my company is a secondary task. My hope is that my business will be enough to support myself for now. I have my own apps that are my products and currently I have one client project.

For which mobile platforms do you develop apps?

Android and iOS.

Which mobile apps do you develop more often: cross-platform or native?

I haven’t tested cross-platform development myself. I develop only native.

Which development method do you use, agile or waterfall?

I don’t use any specific method, because I am a single developer at my company.

Could you tell why you haven’t tested cross-platform development?

I started developing mobile applications five years ago. Back then I thought that cross-platform technology is not good enough, it didn’t give the quality I wanted. I checked different frameworks but when I looked at cross-platform apps, they felt more like a website or a web app. In my opinion a mobile app should give a feeling that it’s part of the platform. This applies more to a philosophical discussion about what is an app. I can explain this so that you understand how I think. In my opinion mobile websites and web apps have come a long way. If the purpose of your app is to present information to the user, then you can use web technology. But native is better if you want to have an app that is a tool or something that you use on a daily basis. As an example I always use public transport. Let’s say you have an app about public transport in your home town. You stand on a bus stop, you press the button and the app quickly opens. You see a map, a button where you can buy a ticket and so on. This is an app that you require to work quickly, the user interface should be clear and the user should experience it as a part of the telephone. In such situations native is better and superior. When it comes to cross-platform, you could use a website and the result would be the same. Everything I said is based on the analysis that I made a few years ago and I know that the technology has moved on, so it is possible that today’s cross-platform environments are better and more effective than in the past.

Is it possible to reuse code when you develop native apps?

Yes, you can actually re-use more than what you would think. In the past when I had an iOS app, I rewrote the iOS code to Android code line after line. From the architecture’s perspective I try to do the following. When I first write the code in iOS, I think how this could be re-used in Android, I identify classes and components that could be reused from architecture’s view. Then it is much easier to write the same app in Android because the structure is ready. This is much faster than creating the whole architecture from scratch again.
How does the process of creating a new project look like at your company?

Well, since I am a single developer the process is a little bit ad-hoc. Most of the apps that I have created, are my own products, and not client projects. So when I create my own product it’s a bit ad-hoc, I get an idea and I develop the app. When it comes to clients’ projects, I first get an inquiry to develop an app, I get requirements from the customer and I tell them how much this is going to cost. After that I follow an iterative process when developing.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.
Appendix 6
This appendix contains notes from an interview with the sixth company.

For which mobile platforms do you develop apps?

Android, iOS and Windows Phone.

Which mobile apps do you develop more often: cross-platform or native?

We develop both types of applications, but we try to develop as much cross-platform as possible.

When it comes to cross-platform solutions, do you develop more cross-platform apps, hybrid apps or web apps?

We use Xamarin and React Native, so the applications are compiled into native by these frameworks. Sometimes we develop hybrid apps.

Which cross-platform frameworks have you tested and what kind of results did you get?

We have used Xamarin in the last two years. We try to use it as much as possible.

Which development method do you use, agile or waterfall?

We use our own method, but it is based on Agile development.

What kind of benefits are there with Xamarin?

In some situations Xamarin is faster, you develop in C# instead of Java. Another benefit is that you have one code base, which you develop and use on all three platforms. You also have lower costs. And when if you find a bug you can fix in it one place, instead of fixing it in three different places.

Have you tested other cross-platform frameworks than Xamarin?

Yes, we have tested Cordova, React Native, HTML5 apps.

And what kind of results did you get with those frameworks?

We got the best results with Xamarin.

Is it possible to reuse code when you develop native apps?

No.

Is it possible to reuse code when you develop cross-platform apps?

Yes.
How does the process of creating a new project look like at your company?

We create a project model in the beginning. We create scenarios and we develop with agile methods, we use scum.

Is a cross-platform development process different from native development process?

Developing with Xamarin is faster and easier, so there is a difference.

When you develop apps, and you have to make a choice to develop a cross-platform app or a native app, how do you make that choice?

The customer decides.

Is it always the customer who decided that?

We always advice the customer to use cross-platform, but it is the customer who decided that in the end.

Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?

Different programmers, but we mix them sometimes.

In which situations is it better to develop cross-platform apps instead of native apps?

We always use cross-platform and Xamarin, because in Xamarin you can do the same things as in native. You have 100% access to the API, so there is no difference. With cross-platform and Xamarin it is much easier to find C# developers than finding developers who know Java or developers who know all three languages that could develop apps on all three platforms.

Is the response time different between cross-platform apps and native apps?

In Xamarin you can get faster response time in some situations. So yes, there is a difference.

Is there any difference in customer experience between cross-platform and native apps?

No, if you use Xamarin.

Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?

No. There are probably some other differences but I can’t come up with anything else right now.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.
Appendix 7

This appendix contains notes from an interview with the seventh company.

For which mobile platforms do you develop apps?

iOS and Android.

Which mobile apps do you develop more often: cross-platform or native?

Native.

Which cross-platform frameworks have you tested and what kind of results did you get?

We tested a little bit. We tried Cordova and Xamarin.

And what were the results?

Difficult to say. I like native more, but everything has its pros and cons.

What were the cons with cross-platform?

Well, in Cordova it was the performance. I cannot say anything about Xamarin because it wasn’t me personally who tested it.

Which development method do you use, agile or waterfall?

We have our own method, we use some agile.

Is it possible to reuse code when you develop native apps?

In general, no. Only a few parts.

Is it possible to reuse code when you develop cross-platform apps?

A little bit.

Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?

It depends. We mix our programmers.

How does the process of creating a new project look like at your company?

First we sketch our app in some kind of simple tool. It depends on how much time we have but we plan and sketch the app. After that we get requirements and technical specifications and then we create the prototype. From that point it depends on how big the project is but we then develop the app.
Is a cross-platform development process different from native development process?

No.

In which situations is it better to develop cross-platform apps instead of native apps?

If we have to create an application that will have to change frequently, be updated very often or an internal app that will be used within a company then it can be really a benefit to use cross-platform. Some code can be reused then and in many situations you can develop the app faster, like when releasing new versions and so on.

Is the response time different between cross-platform apps and native apps?

It depends on which cross-platform tools you use. If you use Xamarin you can compile code into native and the response time and performance will not be very different compared to regular native. But if you use Cordova then you get something that has web performance and web view and not native. I personally, as a developer can see quickly the difference and I can imagine that the user can do that too. The user might get a negative experience from the app.

Are there some other differences between cross-platform and native apps, but from a programmer’s perspective?

There are some technical differences. In native you can access all hardware components, like GPS, camera or orientation detector. You can access it directly in native. If you use cross-platform then you have to make sure that the cross-platform tool allows that.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.
Appendix 8

This appendix contains notes from an interview with the eighth company.

For which mobile platforms do you develop apps?

Android and iOS.

Which mobile apps do you develop more often: cross-platform or native?

We develop both.

When it comes to cross-platform solutions, do you develop more cross-platform apps, hybrid apps or web apps?

We use all of them, but mostly we write the code in HTML and Javascript and compile it into native code. So it will be native app in the end but we build it in HTML.

Which cross-platform frameworks have you tested and what kind of results did you get?

We tested PhoneGap, Xamarin and Apache Cordova. They work well. It all depends on which requirements and wishes the client has.

Which development method do you use, agile or waterfall?

We use SCRUM.

Is it possible to reuse code when you develop native apps?

Yes, to some extent. You have fronted part that is unique for each platform. Then you also have backend code, and you can reuse it.

Is it possible to reuse code when you develop cross-platform apps?

Yes, you can reuse everything.

Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?

Our programmers know more than one platform, but they have one certain platform which they are good at. But they still work in the same team, and help each other and communicate with each other.

How does the process of creating a new project look like at your company?

It starts with an order from the customer. We get requirements, we analyze them, like how the application will be used, how it will be used, who will use it. Once you got ll requirements then you create a prototype, it could be a simple app which shows approximately how the final application will look like. Then you start developing the app and at the end you deliver it.
Is a cross-platform development process different from native development process?

It is almost the same. There are small differences when sketching and planning the app.

In which situations is it better to develop cross-platform apps instead of native apps?

It depends on the customer and their requirements about security, storage, and so on. It is simpler and cheaper to develop cross-platform app, you create it once and can use it everywhere. But there are some benefits with native, you can access the hardware, like the compass.

Could you give some examples on apps that are better to develop cross-platform than native?

Apps where you just have to present information in form of text and pictures, there apps are better to develop as cross-platform. But if you will have to develop an app that is hardware demanding, like GPS, gyroscope, then native apps are better.

Is the response time different between cross-platform apps and native apps?

Yes, but this also depends on what the app does.

Is there any difference in customer experience between cross-platform and native apps?

Yes, this also depends on what the app does. Most often cross-platform apps have a little bit higher response time.

Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?

If you want to develop a native app then you need to have higher knowledge about programming languages and their environment. If you want to develop an iOS app then you need to have a Mac computer and Mac software, that is Xcode. If you use cross-platform then it is enough with just one development environment and a Windows computer.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.
Appendix 9
This appendix contains notes from an interview with the ninth company.

For which mobile platforms do you develop apps?
For all mobile platforms.

Which mobile apps do you develop more often: cross-platform or native?
We create native apps and web apps.

And which apps do you develop more often, native apps or web apps?
We create native more often.

Which cross-platform frameworks have you tested and what kind of results did you get?
We tested React Native but we thought it was too odd. We looked at Xamarin but we didn’t like it. We looked at Ionic but we were not satisfied with it.

Which development method do you use, agile or waterfall?
Agile.

Is it possible to reuse code when you develop native apps?
I think it’s possible to do within the same platform. But most often you have to develop new apps from scratch so there’s not much re-use of code.

Is it possible to reuse code when you develop cross-platform apps?
Yes, it is possible. But it depends on how the app looks like. The point of using web apps is to get higher re-use. Some components can be reused like login form. But you also have functions that are unique for each app, so you can’t reuse that. So some parts can be reused and some can’t.

Do you use any frameworks when you develop web apps?
Yes, we use Angular.

Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?
We have different programmers for each platform.

How does the process of creating a new project look like at your company?
We start by getting the requirements, we sketch the app. Then we try to create a prototype as fast as possible so that you can see what functionality the app will have. And then you develop the app.
Is a cross-platform development process different from native development process?

The process is the same.

In which situations is it better to develop web apps instead of native apps?

It depends on what kind of functionality you want the app to have. There are certain things that you can’t do in a web app and then you have to create a native app.

Could you give some examples on situations when it is better to create a web app rather than native app?

When you need to release a prototype very quickly and when you need to develop the app faster and when you need to save money.

Is the response time different between cross-platform apps and native apps?

We don’t create cross-platform apps, but the point of using it is to reduce costs.

Is there any difference in customer experience between web apps and native apps?

Well, in a web app you need connection to the internet, which slows down the app. When looking at this from a programmer’s perspective, the development process is faster and simpler. Testing and maintenance is simpler.

Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?

No, I mentioned everything. The main difference is the response time and the functionality which you can access.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.
Appendix 10
This appendix contains notes from an interview with the tenth company.

For which mobile platforms do you develop apps?

iOS and Android.

Which mobile apps do you develop more often: cross-platform or native?

Native. We also develop hybrid apps and web apps.

Which cross-platform frameworks have you tested and what kind of results did you get?

We don’t use any frameworks. We create hybrid apps.

So you haven’t tested any cross-platform frameworks?

No.

Which development method do you use, agile or waterfall?

Agile.

Is it possible to reuse code when you develop native apps?

Not within different platforms. But within same platform it is possible.

Is it possible to reuse code when you develop cross-platform apps?

Yes.

Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?

Same programmers.

How does the process of creating a new project look like at your company?

It is a bit different from project to project. First we try to get as clear view of what we are to create as possible. After that we create a prototype. When the client is satisfied with it then we develop the app.

Is a cross-platform development process different from native development process?

It is the same.

In which situations is it better to develop cross-platform apps instead of native apps?

This depends on the budget.
Is the response time different between cross-platform apps and native apps?

Yes, some things are different. In a web app you are a little bit limited, you don’t have access to the hardware functionalities. But there are some ways to get around this. Apps that are not graphically demanding could be implemented as a web app.

You said that there are "some ways to get around" the fact that you don’t have access to all hardware functionalities in a web app. Could you explain what did you mean by that?

You can create a native layer that will handle the user input and it will communicate with the web part of the app.

Is there any difference in customer experience between cross-platform and native apps?

You always try to avoid such differences when you develop apps. But there might come up differences like higher response time and execution time in hybrid apps.

Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?

Yes, in cross-platform the development process is simpler, but the result might not be satisfactory.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.
Appendix 11

This appendix contains notes from an interview with the eleventh company.

Could you tell a little bit about your company and yourself?

I am an independent developer. I run my own one-man business. I develop apps, programs and games for iOS, sometimes Windows Phone. I used to create Android apps in the past but I didn’t like developing apps for that platform.

How does the process of creating a new project look like at your company?

I create my own products. I get my revenue from selling those applications but also from ads and in-app purchases.

Which mobile apps do you develop more often: cross-platform or native?

It depends, but I prefer native. I sometimes create hybrid apps or web apps.

Which cross-platform frameworks have you tested and what kind of results did you get?

I tested Titanium Appcelerator, Unity and Xamarin, but the apps created with them were too slow. I tested also MXF and NGiNX, and these frameworks were a bit better but it’s still not the same thing as native.

Why do you prefer native, and why you don’t use cross-platform frameworks?

Because cross-platform frameworks slow down the apps. Take a look a SoundCloud app for example. It is not a native app, and thus it is so slow. It freezes all the time.

Which development method do you use, agile or waterfall?

I do not use any method.

Is it possible to reuse code when you develop native apps?

Yes.

Is it possible to reuse code when you develop cross-platform apps?

I don’t develop cross-platform apps so I can’t tell for sure but I can imagine that it is possible as well.

In which situations is it better to develop cross-platform apps instead of native apps?

I would use some kind of cross-platform framework or a web framework to create a simple game. That would save me a lot of time, but using such frameworks limits your possibilities. I develop mostly for just iOS, so I don’t feel the need to use cross-platform frameworks.
Is the response time different between cross-platform apps and native apps?

Cross-platform apps are not as fast as native apps.

Is there any difference in customer experience between cross-platform and native apps?

I don’t develop cross-platform apps. But from my point of view cross-platform don’t give that good look and feel and speed that you would like to have in an app.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.
Appendix 12
This appendix contains notes from an interview with the twelfth company.

Could you tell a bit about your company or yourself?

I am a consultant. I cooperate with bigger companies, I go to their offices and I assist them in creating their mobile apps, though out the whole process.

For which mobile platforms do you develop apps?

Android and iOS.

Which mobile apps do you develop more often: cross-platform or native?

Mostly native. Sometimes web apps.

Did you use cross-platform in the past?

Yes, I tried it. But we prefer native development for iOS and Android.

Which cross-platform frameworks have you tested and what kind of results did you get?

I haven’t tested so many cross-platform frameworks. I tested some frameworks for developing games but I didn’t continue testing them because it was not needed in the projects that I was participating in.

Which development method do you use, agile or waterfall?

I don’t develop so many apps personally. I am a consultant. But the people I cooperate with use agile.

Have you tested any frameworks for web apps?

No, I haven’t. I tried HTML5 only but we didn’t continue using it.

Is it possible to reuse code when you develop native apps?

Yes, it’s possible. It depends on how complex the project is. It’s not possible to reuse logic of the app but there are some other components that you can reuse.

Is it possible to reuse code when you develop cross-platform apps?

I imagine that it’s possible.

How does the process of creating a new project look like at your company?

In the projects where I develop apps myself, then I first check what are the wishes of the customer. I also look at the complexity and difficulty of creating the app. Once I have the requirements I tell the
customer how much this is going to cost. If the project is complex then I create a prototype of the app. I use agile development when developing the app.

**You mentioned that you currently don’t develop cross-platform apps. Could you explain why?**

To some extend it’s maybe because I don’t have enough experience in cross-platform development. It was a bit difficult to implement everything I wanted to have in my cross-platform apps. Perhaps I should have spent more time on testing the cross-platform tools. Cross-platform apps felt more like a web app, and not like a real app. I thought that it was difficult to create the UI I really wanted, the final result had too much "web view". Some clients could have said that "this is not an app but a web app".

**In which situations is it better to develop cross-platform apps instead of native apps?**

This depends on the price. It depends on what the customer wants. If the clients wants to have one code-base and they want it cheap, then cross-platform could be used.

**Is the response time different between cross-platform apps and native apps?**

Yes, there are differences. It depends on what the application does. You can see that the app behaves a bit differently. Some web apps might be seen more like a web-view and not a real app. I have seen web apps which were designed so that they loaded some parts before the user requested them, and such web apps worked good, the difference in response time was not so big. But even such apps it’s difficult to make look as real and native. Everything depends on what the app does but if you store everything locally, like in a native app, then the response time will be better.

**Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?**

In cross-platform the programmers have less work and responsibility. If the app is complex then you have a lot of work, but if you have to develop that for two platforms, then you have twice as much work. As a consultant I can say that it’s cheaper to find cross-platform developers because they don’t cost much.

**Thank you, these were all the questions. Would you like to comment on something or say something before we finish?**

No.
Appendix 13
This appendix contains notes from an interview with the thirteenth company.

For which mobile platforms do you develop apps?

Android, iOS and Windows Phone.

Which mobile apps do you develop more often: cross-platform or native?

We use cross-platform frameworks.

Which cross-platform frameworks have you tested and what kind of results did you get?

We currently use Sencha, PhoneGab and Titanium. We develop apps in those tools and then we compile the code into native code.

Why do you prefer cross-platform development?

Because cross-platform apps have the same efficiency as native apps, and you also save a lot of time. Costs are smaller. Developing the app goes much faster.

You mentioned three different cross-platform frameworks. Why did you choose these?

We liked the results that these tools give. There is no difference between a native app and an app created in cross-platform environment. In the end you will have native code, it’s just that you use a tool to create it, and you don’t create the app directly in native.

Which development method do you use, agile or waterfall?

Agile.

Is it possible to reuse code when you develop native apps?

No.

Is it possible to reuse code when you develop cross-platform apps?

Yes.

Do you have different programmers for each platform, like Android or iOS, or do you have same programmers for all platforms?

We have same programmers for all platforms, because we develop our apps in cross-platform frameworks.
How does the process of creating a new project look like at your company?

We first meet with the customer and get the requirements. We do a careful analysis of their needs. After that we sketch the app and create the design of the app. If everything is ok we develop the app. If the customer likes it we deliver it to them.

In which situations is it better to develop cross-platform apps instead of native apps?

We always develop cross-platform apps.

Is the response time different between cross-platform apps and native apps?

No. By using cross-platform you also get native code so there is no difference.

Is there any difference in customer experience between cross-platform and native apps?

No.

Are there some other differences between cross-platform and native apps when looking from programmer’s perspective?

You have one code-base and one environment and you save time when developing the application. The maintenance of the app is also easier and cheaper.

Thank you, these were all the questions. Would you like to comment on something or say something before we finish?

No.