Institutionen för datavetenskap
Department of Computer and Information Science

Final thesis

Multi-platform development of applications for mobile devices

by

Mattias Sjödin

LIU-IDA/LITH-EX-G—12/025—SE

2012-10-18
Final Thesis

Multi-platform development of applications for mobile devices

by

Mattias Sjödin

LIU-IDA/LITH-EX-G—12/025—SE

2012-10-18

Supervisor: Jimmy Jonasson, Combitech AB
Examiner: Simin Nadjm-Tehrani, Linköping University
Abstract

This bachelor thesis has been performed at Combitech, a technology, development and management consultancy company. The purpose of the thesis is to investigate the possibilities for multi-platform development of applications for mobile devices, as well as actually developing a simpler application using multi-platform development. The application is supposed to have functionality for sending an expression of interest for work and thesis work at Combitech, and should be available both as a native application for the Android and iOS operating systems as well as in the form of a web application.

The thesis begins with an investigation of the possibilities for multi-platform development of applications for mobile devices. It is concluded that there is a wide range of available development frameworks that target several mobile platforms. It is decided that an HTML/CSS/JavaScript-based framework is most suited for the application to be developed. Therefore four HTML/CSS/JavaScript-based multi-platform development frameworks targeting mobile devices are analyzed more closely, namely jQuery Mobile, PhoneGap, Sencha Touch and Appcelerator Titanium.

jQuery Mobile and PhoneGap are chosen to develop the application, and the thesis describes the development work as well as the resulting application. It is concluded that HTML/CSS/JavaScript-based frameworks can provide a nice touch screen optimized user interface that is consistent across platforms with e.g. jQuery Mobile, as well as access to native features and the possibility to package a web application as a native application with e.g. PhoneGap. It is also concluded that jQuery Mobile and PhoneGap can be used to create a modular application that, with well defined interfaces, can be run both as a web application and packaged as a native application on different mobile operating systems with no or minor changes to the code.
Contents

1 Introduction 1
  1.1 Background . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
  1.2 Purpose and scope . . . . . . . . . . . . . . . . . . . . . . . 2
  1.3 Application . . . . . . . . . . . . . . . . . . . . . . . . . . . 2
    1.3.1 Requirements . . . . . . . . . . . . . . . . . . . . . . 3
  1.4 Methodology . . . . . . . . . . . . . . . . . . . . . . . . . . 4
    1.4.1 Investigative phase . . . . . . . . . . . . . . . . . . . 5
    1.4.2 Development phase . . . . . . . . . . . . . . . . . . . 5

2 Applications for mobile devices 7
  2.1 Mobile devices . . . . . . . . . . . . . . . . . . . . . . . . . 7
  2.2 Operating systems . . . . . . . . . . . . . . . . . . . . . . . 8
    2.2.1 Android . . . . . . . . . . . . . . . . . . . . . . . . . 8
    2.2.2 iOS . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
  2.3 Application stores . . . . . . . . . . . . . . . . . . . . . . . 8
  2.4 Web applications . . . . . . . . . . . . . . . . . . . . . . . 9

3 Multi-platform development 11
  3.1 Frameworks . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
    3.1.1 Web applications . . . . . . . . . . . . . . . . . . . 12
    3.1.2 Challenges . . . . . . . . . . . . . . . . . . . . . . . 13
  3.2 Analysis . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
    3.2.1 jQuery Mobile . . . . . . . . . . . . . . . . . . . . . 15
    3.2.2 PhoneGap . . . . . . . . . . . . . . . . . . . . . . . . 15
    3.2.3 Sencha Touch . . . . . . . . . . . . . . . . . . . . . 16
    3.2.4 Appcelerator Titanium . . . . . . . . . . . . . . . . 16
    3.2.5 Comparison . . . . . . . . . . . . . . . . . . . . . . . 17
  3.3 Selecting a method . . . . . . . . . . . . . . . . . . . . . . . 18
    3.3.1 Decision . . . . . . . . . . . . . . . . . . . . . . . . 19

4 Application development 21
  4.1 Development plan . . . . . . . . . . . . . . . . . . . . . . . 21
  4.2 Test plan . . . . . . . . . . . . . . . . . . . . . . . . . . . . 21
  4.3 Development environment . . . . . . . . . . . . . . . . . . 22
## CONTENTS

4.4 Frameworks ................................................. 22  
4.4.1 jQuery Mobile ........................................... 22  
4.4.2 PhoneGap ................................................. 24  
4.5 Architecture ................................................. 25  
4.6 Development process ......................................... 26  
4.6.1 Iteration 1 .................................................. 26  
4.6.2 Iteration 2 .................................................. 26  
4.6.3 Iteration 3 .................................................. 27  
4.7 Issues .......................................................... 27  

5 Results ......................................................... 29  
5.1 Application ................................................... 29  
5.1.1 Structure ................................................... 29  
5.1.2 Functionality ............................................... 30  
5.1.3 Limitations ................................................ 33  
5.2 Test reports ................................................... 34  
5.2.1 Native application ......................................... 34  
5.2.2 Web application ........................................... 35  
5.2.3 Extended native application ............................. 36  
5.2.4 Final testing ............................................... 37  

6 Conclusion .................................................... 39  
6.1 Summary and Conclusions ..................................... 39  
6.2 Discussion .................................................... 41  
6.2.1 The developed application ............................... 41  
6.2.2 Native vs portable applications .......................... 41  
6.2.3 Future of multi-platform development .................. 42  
6.3 Future work .................................................... 42  

Bibliography ..................................................... 43
Chapter 1

Introduction

Mobile devices, such as smartphones and tablets, are becoming increasingly popular. According to recent estimates by Canalys [1], in 2011 more smartphones than PCs were sold worldwide.

Today there is a large spread in the market shares of operating systems that are running on these mobile devices. Estimates of the market share vary a lot, but the two major operating systems, the Google-supported Android and Apple’s iOS, have roughly a 50% and 25% market share respectively in an estimate by Gartner [2].

Most operating systems for mobile devices provide Software Development Kits (SDK) that allows third party developers to develop applications that run on them, and gives them access to a number of features on the mobile device, such as sensors like accelerometers, compasses and GPS receivers.

Both the execution and development environment for applications differs largely between operating systems for mobile devices. Applications for the Android operating system are primarily developed using the Android SDK and the Eclipse Integrated Development Environment (IDE) with the programming language Java. The application’s source code is then compiled into bytecode that runs on the Dalvik virtual machine in the operating system on the mobile device [3]. Applications for the iOS operating system are primarily developed using the iOS SDK and the Xcode IDE with the programming language Objective-C. The application’s source code is then compiled to machine code for processors using the ARM architecture that is used on mobile devices running iOS [4].

If an application is to be developed and target both of these operating systems, these differences normally lead to running two separate development projects, targeted at one platform each. This is time consuming and requires different competences which can lead to high costs. It would therefore be desirable to be able to run only one development project, targeted at both platforms, with as little redundant work necessary as possible.
1.1 Background

Combitech is one of Sweden’s leading technology, development and management consultancy companies. Combitech promotes themselves at several labour market fairs and universities every year. Visitors at these fairs can leave an expression of interest, for the moment by filling out a piece of paper with contact information and some information about themselves. However, it would be desirable if these expressions of interest could instead be given digitally. That would be more modern as well as making the given expressions of interest easier to process.

This could be possible by running an application on a mobile device, where an interested visitor could fill out and leave an expression of interest, instead of filling it out on a piece of paper. This application could either run on larger tablets made available in Combitech’s booth at the fair, or on the visitors own mobile devices such as smartphones, e.g. by scanning a QR (Quick Response) code that would open a web application in their browser.

1.2 Purpose and scope

The purpose of this thesis is two-fold:

- To investigate the possibilities for multi-platform (also called cross-platform) development of applications for mobile devices.
- To develop an application for leaving an expression of interest for work and thesis work at Combitech, using multi-platform development.

The investigation will look at what alternatives there are for multi-platform development for mobile devices, how these differ from each other and what their strong and weak sides are in relation to the purpose of the application to be developed.

From this investigation it will then be decided which alternative for multi-platform development that is best suited for an application to give an expression of interest for work and thesis opportunities at Combitech.

An application that has this functionality will then be developed using that method, and will be described in the next section of this chapter.

1.3 Application

The application should provide the ability to leave an expression of interest for work and thesis work at Combitech, by filling out a form with relevant information and sending it to Combitech.

Because there is such a large spread in the market shares of operating systems for mobile devices, the application should work on at least the Android and iOS operating systems as a native application, but preferably also
as a web application. As the focus of this thesis project is to investigate multi-platform development of applications for mobile devices, the application that is developed as a part of this project is not expected to achieve full functionality for its purpose during this project.

1.3.1 Requirements

The requirements for the application are prioritized in three levels, with Priority 1 being the highest. As the application is not expected to achieve full functionality during this ten weeks project, at least the requirements with Priority 1 are expected to be fully completed and all of the requirements with Priority 3 are not expected to be completed during this project.

The application’s development is therefore in effect divided into three parts. The first part, consisting of fulfilling the Priority 1 requirements, produces a basic native application for saving expressions of interests locally on a mobile device running the Android or iOS operating systems. The second part will produce a web application that can be accessed by the visitors of the fair on their own mobile devices, by navigating to a website where they can fill out and leave their expression of interest by sending it to a server on the Internet. The third part, consisting of expanding the native application from the first part with features of priority 3, produces an extended native application to be run on mobile devices provided by Combitech at e.g. labour market fairs.

Priority 1 - Native application

1.1 The application should be developed using multi-platform development.

1.2 Users of the application should be able to fill out an expression of interest for work and thesis work, containing among other things contact information, information about their education, their interest areas and a short description of themselves.

1.3 Expressions of interest that are given using the application should be saved locally on the device running it, in a suitable data format such as e.g. JavaScript Object Notation (JSON) or Extensible Markup Language (XML). The data should then be able to be read from the device, outside the application.

Priority 2 - Web application

2.1 The application should be available as a web application for mobile devices, where expressions of interest instead of being saved locally are sent over the Internet to a server that receives and saves them in a database.
2.2 The web application should work in default mobile web browsers on mobile devices running the Android and iOS operating systems.

Priority 3 - Extended native application

3.1 It should be possible to authenticate as an administrator of the application.

3.2 An authenticated administrator should be able to view expressions of interest that have been saved locally on the device, using the application.

3.3 An authenticated administrator should be able to register the information about what fair that the application is being used at, and that information should be saved with expressions of interest.

3.4 An authenticated administrator should be able to load saved expressions of interest to a computer using the application and then remove them from the device.

3.5 Expressions of interest should besides being saved locally on the device also be able to be sent to an e-mail address specified by an authenticated administrator.

1.4 Methodology

The thesis project is divided into two phases. In a first investigative phase the possibilities for multi-platform development of mobile applications are investigated and a method suitable for developing the application is selected. In a second development phase, the application is developed using the method selected in the first phase. Table 1.1 shows the expected results from each of these two phases.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Expected results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Investigative phase</td>
<td>An investigation report on multi-platform development and a selected method for developing the application.</td>
</tr>
<tr>
<td>Development phase</td>
<td>A report on the development process using multi-platform development and an application with functionality according to the requirements.</td>
</tr>
</tbody>
</table>

Table 1.1: The expected results from the two phases of the thesis project.
1.4. METHODOLOGY

CHAPTER 1. INTRODUCTION

1.4.1 Investigative phase

During this phase multi-platform development will be investigated. What possibilities there are for multi-platform development will be evaluated using scientific articles and documentation for various multi-platform development solutions. A few of the alternatives that seem well suited for the application to be developed will then be investigated more thoroughly and compared using these criteria:

- Operating system (OS) compatibility - As the application should work on at least the Android and iOS operating systems, it is important that the multi-platform method supports these operating systems, but preferably more.

- Native support - The possibility to use some, if not all, of the native functionality provided by the native SDKs, such as sensors, location services and access to the local filesystem, is desired.

- Web application - As it is desirable for the users of the application to be able to use it on their own mobile devices, the possibility to open the application as a web application in a web browser on a mobile device is desirable.

- User interface - A good user interface, which will not have to be rewritten for each individual platform, is desired.

The alternatives will be given a rating of "Bad", "Acceptable" and "Good" in regard to these criteria, relative to each other. This means that the investigated alternative that is best suited in each category will be given the rating "Good" and the other alternatives will be given a rating of "Good", "Acceptable" or "Bad" in relation to that. The alternative that seems most suited for the application to be developed is then selected to be used during the second phase of the thesis project.

1.4.2 Development phase

The application described in the Section 1.3 will be developed during this phase, using the method chosen in the previous investigative phase. The development will be divided into three development iterations as well as a testing phase. Table 1.2 shows the expected results from each of these iterations as well as the testing phase.

In the first development iteration, the Priority 1 requirements will be fulfilled, which will yield a native application running on the Android and iOS operating systems with basic functionality. In the second iteration the application will be adapted to a web application by fulfilling the priority 2 requirements. In the third iteration, the application is polished, and the native application may be extended with additional functionality according to the priority 3 requirements if there is time available.
### 1.4. METHODOLOGY

**CHAPTER 1. INTRODUCTION**

<table>
<thead>
<tr>
<th>Iteration/Phase</th>
<th>Expected results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development iteration 1</td>
<td>A native application with functionality according to the priority 1 requirements.</td>
</tr>
<tr>
<td>Development iteration 2</td>
<td>A web application with functionality according to the priority 2 requirements.</td>
</tr>
<tr>
<td>Development iteration 3</td>
<td>That all priority 1 and most of the priority 2 requirements are fulfilled. Possibly an extended native application according to the priority 3 requirements.</td>
</tr>
<tr>
<td>Testing phase</td>
<td>An application that has the same functionality on both the Android and iOS operating systems as well as a test report summarizing the outcomes of the tests.</td>
</tr>
</tbody>
</table>

Table 1.2: The expected results from the each iteration of the development phase of the thesis project.

After each iteration the functionality added during that iteration will be tested on a mobile device running the Android operating system. During the testing phase the application will be tested thoroughly on devices running both the Android and iOS operating systems to ensure that all functionality is available and works as expected on both platforms. Test reports will be written from both the testing after the iterations and the testing from the testing phase.
Chapter 2

Applications for mobile devices

This chapter gives a brief introduction to mobile devices, their platforms and operating systems as well as applications developed for them.

2.1 Mobile devices

A mobile device can be defined as a small handheld computing device. In this thesis, the term ”mobile device” will refer to smartphones and tablet computers with touch screen user interfaces.

Smartphones typically have screens of sizes 2.5" - 5" which take up most of the front of the device, with common resolutions as low as 240x320 pixels and as high as 1920x1080 pixels. Tablet computers typically have screens of sizes 5" - 10" which also take up most of the front of the devices, with resolutions typically ranging over 800x480 pixels up to 2048x1536 pixels. These devices are most commonly controlled through touch screens, most commonly with the users finger, but also more rarely with a special pen. They may also have a physical keyboard and some form of pointing device, however neither of these are common in new devices at the time of the writing of this thesis.

Mobile devices most commonly allow installation and running of third party applications. Mobile devices also commonly have hardware sensors, most commonly geolocation sensors such as GPS receivers, orientation sensors such as gyroscopes and accelerometers, cameras, and a compass which are made available to these applications.

Mobile devices are network oriented, often having Internet access through both WiFi and cellular networks with a web browser preinstalled on the device.
2.2 Operating systems

Mobile devices run operating systems which allow the user to interact with the device. As mentioned in Chapter 1, at the time of the writing of this thesis, there is a large spread in the market shares of these operating systems, with the Google supported Android operating system having the largest market share, followed by Apple’s iOS operating system. These two operating systems will be described in this chapter.

2.2.1 Android

Android is an open source Linux-based operating system, primarily targeted at smartphones and tablet computers, with a market share of roughly 50% [5] [2]. Android devices commonly have access to a market place called Google Play out of the box, offering over 600 000 applications that run on the operating system [6]. However, there are also other application stores available for Android devices, such as GetJar\(^1\) and Amazon Appstore for Android\(^2\). It is also possible to install applications outside of application stores by downloading and opening an application package file. Android devices typically have at least the sensors mentioned in Section 2.1 and Internet access through both cellular networks and WiFi. Android devices have a webkit based default web browser preinstalled.

2.2.2 iOS

iOS is a Unix-based operating system developed by Apple, for Apple’s mobile devices. Apple is at the time of writing this thesis offering a smartphone called iPhone with a 3.5" touch screen and a tablet computer called iPad with a 9.7" touch screen. These devices have all the sensors described in Section 2.1 and Internet access through WiFi, as well as through cellular networks for the iPhone and in some configurations of the iPad. In their default configuration, iOS devices only allow installation of applications through the Apple App Store, offering over 500 000 applications [7]. iOS devices have the Safari Mobile webkit based default web browser preinstalled.

2.3 Application stores

Both Google Play and the Apple App Store allow users with mobile devices running the Android and iOS operating systems respectively to download and install applications on them over the Internet. These applications can either be free or require the paying of a fee, typically in the area of 1-5 USD, however it also common that free applications include advertisements. These applications can then get access to the hardware sensors on the mobile

\(^1\)http://www.getjar.com/
\(^2\)http://www.amazon.com/mobile-apps/b?node=2350149011
devices and interact with the user through them, as well as the touch screen and speakers of the device. A lot of these applications are network oriented and requires communication with a third party over the Internet.

Developers can develop applications for these operating systems using their respective SDKs as mentioned in the Chapter 1. These applications can then either be published in application stores, in the case of the Apple App Store after a screening and approval process, and thereby made available to the users. In the case of the Android operating system, as mentioned, applications can also be made available through an alternative application store or by packaging the application and distributing it outside of application stores.

2.4 Web applications

As most mobile devices have Internet access and a web browser, applications for them can also be developed and made available in the form av web applications. These web applications are often primarily developed using HyperText Markup Language (HTML), JavaScript and Cascading Style Sheets (CSS). They can be made available for mobile devices either through application stores or by hosting them on a web server and then be accessed from a web browser. When a web application is hosted on a web server and accessed through a web browser, it does not need to be installed on the device using it, however it may store data on the device for persistence between sessions and requires an Internet connection to be accessed. On both the Android and iOS operating systems a shortcut to a web application hosted on a web server can be created and saved in the same manner as a shortcut to a normal application obtained through an application store as explained by Ableson [8]. A web browser is launched when the shortcut is accessed, which then navigates to the web application and presents it to the user. The difference between a web application hosted on a web server and accessed through a web browser and a normal application installed through an application store may therefore not be noticed by a user.

Work on providing standardized Application Programming Interfaces (API) for hardware sensors in mobile devices to web applications is in progress by the World Wide Web Consortium (W3C). Some type of sensors, such as Geolocation and motion sensors are already available to web applications on both the Android and iOS operating systems using their default browsers [9].

An interesting trend in web design is responsive design, described by Marcotte [10], where web pages are built to be adaptable to different screen resolutions and maintain the same functionality and accessibility on both mobile devices and normal computers on the very same webpage.
Chapter 3

Multi-platform development

This chapter includes an investigation of the possibilities for multi-platform development of applications for mobile devices. It provides an overview of the possibilities for multi-platform development and what their strengths and weaknesses are. A few methods more suited for the application described in Section 1.3 are then investigated more thoroughly and finally one of them is chosen for the development.

3.1 Multi-platform development frameworks

At the time of writing, a very large number of frameworks and SDKs for application development for mobile devices are available. A Wikipedia article on mobile application development [11] lists more than 50 of them targeting more than one mobile platform.

These multi-platform development frameworks differ in many ways. First and foremost they are available under different licenses, some are available under free licenses, some under commercial licenses and some are available under both types of licenses. The article Cross-Platform Development Tools for Smartphone Applications by Ohrt and Turau [12] presents and compares 9 of these frameworks more thoroughly and concludes that they can be divided mainly into two types. The first type runs purely native on the mobile platform, meaning they execute code native to the mobile platform. The other type contains interpreted code, meaning they may launch an interpreter native to the platform and then interpret non-native code using the interpreter. The interpreted approach means that an interpreter can either be packaged with the application, which is the case with e.g. RhoMobile\(^1\) where an interpreter for the Ruby programming language is wrapped with

\(^1\)http://www.rhomobile.com
the application, or on some platforms be installed separately, which is the case with Adobe AIR on the Android operating system. Another type of interpreted applications are web applications, where the user interface is built using HTML, CSS and JavaScript and may include application logic in JavaScript. The web application is then rendered using a web browser on the mobile device, which also interprets the JavaScript code.

An interesting example is MonoTouch\(^2\), for the iOS operating system, and Mono for Android\(^3\), for the Android operating system. These SDKs, from the same company, Xamarin, allow developers to develop applications using the C# programming language and the .NET Framework. Developers can then interface with the native SDKs through a platform-specific C# API. This allows developers to keep the application logic, written in C# and .NET, independent from the different native platforms, and then interface it with platform specific user interfaces and functions, still developed using the C# programming language. An application developed using Mono for Android is compiled into Common Intermediate Language (CIL) bytecode and packaged as a native Android application together with an implementation of the Mono runtime for the Android operating system, which interprets the CIL bytecode, meaning it is using the interpreted approach \[^{13}\]. However, an application developed using MonoTouch is compiled into native machine code for the ARM architecture used on devices running the iOS operating system, meaning it is using the native approach \[^{14}\].

3.1.1 Web applications

Both the Android and iOS SDKs allow the development of applications using a "WebView" or similar, where the application is a normal web application being viewed using a web browser, but still packaged in a native application \[^{15}\] [\(^{16}\)]. However, they do not provide any JavaScript API or similar interfaces to access the native features of the device, nor any touch screen optimized way to develop the user interface using HTML, CSS and JavaScript. This is solved by some frameworks that utilize this functionality to build multi-platform applications and packaging them as native applications for the different operating systems. Some provide touch screen optimized user interfaces built using HTML, CSS and/or JavaScript. Web applications developed using these frameworks can also be accessed using web browsers on mobile devices, instead of installing a native application. Some frameworks such as PhoneGap\(^4\) also, or only, provide the ability to access native features of the different platforms, such as location services and sensors, through a JavaScript API and platform specific libraries. They also then provide means to package the application into a native application for the supported platforms.

\(^{2}\)http://www.xamarin.com/monotouch/
\(^{3}\)http://android.xamarin.com
\(^{4}\)http://www.phonegap.com
3.1.2 Challenges

Ohrt and Turau [12] discuss some challenges of the different multi-platform development frameworks.

Interpreters

Some drawbacks of applications developed with multi-platform development frameworks using the interpreted approach is that the installed size of the application may increase greatly if the interpreter is packaged with the application and another drawback is that performance may suffer. The authors compared the application package size, memory usage and launch time of a very basic application for the Android operating system, developed using both the native Android SDK and nine multi-platform development frameworks. It was concluded that performance varied greatly between the different frameworks. When it comes to launch time, some launched almost as fast as the application developed using the native SDK, but most used significantly more memory. The application package size varied greatly, with some being roughly equal to that of the native application and others being significantly larger.

One disadvantage with this study was that the tested application was very basic, only containing a text label. As some of the applications include a virtual machine or runtime to run and interpret the application code, the relative increase in memory usage because of that may be smaller in a more advanced application. This was also shown in the article [12] when compared to the more advanced application Qype developed using the native Android SDK, which used significantly more memory than the very basic application developed for the study.

User expectations

Another disadvantage with multi-platform development frameworks discussed by Ohrt and Turau [12] is regarding user expectations. While some multi-platform development frameworks allow developers to develop a single uniform user interface that looks roughly the same on all platforms, this also means that the application in some cases will not look like an application native to the platform, as it does not utilize the native user interface of that platform, which the users may be accustomed to. Another drawback regarding user expectations may be features that differ between the platforms, such as a physical "Back"-button present on devices running the Android operating system, but not on e.g. iOS, or "Live Tiles" (dynamic links to applications) on the Windows Phone 7 operating system, which some frameworks may not make use of.
3.2. ANALYSIS

CHAPTER 3. MULTI-PLATFORM

Native features

A final disadvantage with multi-platform development frameworks is access to platform specific system APIs and therefore native features such as sensors, like accelerometers, cameras and platform specific features such as notifications. Most frameworks provide access to either all or a subset of these in some form, either through platform specific APIs, such as the case with the Mono for Android and MonoTouch frameworks, or through a platform independent API, such as PhoneGap’s JavaScript Cordova API.

Apple App Store

Another important challenge of multi-platform development frameworks is the policies for the Apple App Store, which are being updated continually and historically puts some restrictions on how applications to be released in it are supposed to be developed. Previously the policies stated that applications had to be “originally written in Objective-C, C, C++ or JavaScript” to be allowed in the App Store. However, for the time being, there are only restrictions against applications that download executable code or interpret code that is not contained in the application’s archive [12].

3.2 Analysis

The application described in Section 1.3 will mainly consist of a form, where an applicant can fill out some information and then have it either be sent to a remote server or saved locally in a file or database on the mobile device. This means the application logic will be relatively thin, and the focus on the application will be on the user interface and interfacing with either the local file system or a remote server.

This implies that multi-platform development frameworks that have their focus on making heavier application logic platform-independent, but where the user interface has to be implemented on each platform, such as MonoTouch, Mono for Android and Marmalade SDK are less suited for the application to be developed. However, HTML/CSS/JavaScript-based frameworks seem well suited for the application. With these frameworks the user interface will only need to be implemented once, and look relatively uniform on all platforms. Also JavaScript and HTML is well suited for interfacing with a remote server. Using this kind of frameworks will also introduce the possibility to run the application in a web browser on mobile devices without the need to install it as a native application, which was a desired feature of the application. In the rest of this section a few HTML/CSS/JavaScript-based multi-platform development frameworks will be presented and analyzed in more detail.
3.2.1 jQuery Mobile

jQuery Mobile\(^5\) is a web framework built upon the popular JavaScript library jQuery, but optimized for use on mobile devices. It is available under GPL or MIT licenses. It allows developers to develop touch-screen optimized web applications with a unified user interface. It is widely used for developing web pages and web applications aimed at web browsers on mobile devices [17]. It has broad support for both desktop browsers, as well as mobile devices [18].

jQuery Mobile has a strong point in its elaborate and feature-rich user interface. The user interface has support for theming. The user interface uses responsive design, meaning it scales automatically with the size of the screen. It is event-driven with support for among other things touch and mouse events. It utilizes AJAX (Asynchronous JavaScript and XML) for navigation between pages and supports animated page transitions. jQuery Mobile has a markup-based system for defining and configuring the user interface and to apply behavior and theming. jQuery Mobile builds upon and therefore provides the entire jQuery JavaScript library for use with the application, which is a good feature of the framework.

On its weak side jQuery Mobile offers no support for packaging web applications created using the framework into native applications for mobile devices. It also therefore does not include support for using any native features of the mobile devices (other than touch-events). jQuery Mobile is however often combined with PhoneGap to package a web application into a native application for different platforms, and make use of native features of the devices using PhoneGap’s Cordova API.

3.2.2 PhoneGap

PhoneGap\(^6\) is an open-source mobile development framework available under the Apache license. It allows developers to build their applications using HTML, CSS and JavaScript and package it as a native application for several mobile platforms. Through PhoneGap’s JavaScript Cordova API developers gain access to a wide range of native features on the different mobile platforms [19]. A wide range of operating systems for mobile devices are supported by PhoneGap, however not all features of the Cordova API are supported on all of them [20]. This allows developers to target several mobile platforms with a single unified codebase using HTML, CSS and JavaScript.

The native SDK for all targeted platforms are required to compile and package the applications [12]. The application’s user interface is then rendered and the JavaScript code interpreted on the mobile devices using a built-in web browser, however packaged as a normal application, and the difference from a purely native application may not be visible to the user. Native features such as sensors are then accessed through the Cordova API.

\(^{5}\)http://jquerymobile.com/
\(^{6}\)http://www.phonegap.com
3.2.3 Sensa Touch

Sensa Touch\(^7\) is an HTML5- and JavaScript-based mobile development framework available under free commercial and open source (GPL) licenses. Installation of the Sensa Touch SDK and the Sensa SDK Tools is required. An IDE is not included, but the SDK requires a web server to run and test the application during development. The user interface is declared programatically using JavaScript, and produces an HTML/CSS/JavaScript-based user interface that looks uniform on the different platforms. It has a look similar to that of the native user interface of the iOS operating system and has support for theming. An application built using Sensa Touch is therefore entirely built using JavaScript, however closely tied to the SDK, requiring it to build, run and package the application.

A JavaScript library for accessing native features on mobile devices is provided with the SDK, however it only gives access to four features: Connection, Notification, Orientation and Camera. The native library also only supports Android and iOS and only when the application is packaged into a native application for these operating systems [21]. This means it provides a lot less native features, on a lot fewer platforms than PhoneGap mentioned in the previous section. However, Sensa Touch can make use of PhoneGap to access the native functions it provides.

An application built using Sensa Touch can be run on a web server, where it can be reached as a normal web application using web browsers on mobile devices. However the SDK also provides the possibility to package an application built using Sensa Touch into a native application for the iOS and Android operating systems, in the case of Android requiring the installation of the Android SDK.

3.2.4 Appcelerator Titanium

Appcelerator Titanium\(^8\) is a JavaScript-based SDK for developing applications targeted at the Android and iOS operating systems, however support for Blackberry devices is in development as well. The Titanium Studio Eclipse-based IDE is provided, however requires registration of an account and log in to use. The SDK is licensed under the Apache Public License, meaning it is free for both commercial and personal use. However, some features of the SDK as well as support requires a paid license.

\(^7\)http://www.sencha.com/products/touch
\(^8\)http://www.appcelerator.com/
Libraries for accessing native features from JavaScript are included with the SDK. Support for using native user interface features of Android and iOS is included, however this requires some double work for the different platforms. Support for defining the user interface using HTML and CSS is included, but using the native user interface features of the SDK is recommended. An application developed using Titanium Appcelerator and native user interface features therefore becomes heavily coupled with the Titanium SDK and relies on its further development to keep compatibility with newer versions of the Android and iOS operating systems.

### 3.2.5 Comparison

Here follows a comparison of the analyzed multi-platform development frameworks, against the criteria defined in Section 1.4.1. Table 3.1 shows the analyzed multi-platform development frameworks and their respective ratings. As jQuery Mobile and PhoneGap should be used together to achieve the functionality desired for the application to be developed, they will be compared both as separate alternatives and together as a single alternative.

<table>
<thead>
<tr>
<th>Framework</th>
<th>OS compat.</th>
<th>Native support</th>
<th>Web application</th>
<th>User interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>jQuery Mobile</td>
<td>Good</td>
<td>Bad</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>PhoneGap</td>
<td>Good</td>
<td>Good</td>
<td>Bad</td>
<td>Bad</td>
</tr>
<tr>
<td>Sencha Touch</td>
<td>Acceptable</td>
<td>Bad</td>
<td>Good</td>
<td>Good</td>
</tr>
<tr>
<td>Appcelerator Titanium</td>
<td>Acceptable</td>
<td>Acceptable</td>
<td>Acceptable</td>
<td>Acceptable</td>
</tr>
<tr>
<td>jQuery Mobile &amp; PhoneGap</td>
<td>Good</td>
<td>Good</td>
<td>Acceptable</td>
<td>Good</td>
</tr>
</tbody>
</table>

Table 3.1: Comparison of the analyzed multi-platform development frameworks.

**Operating system compatibility & Native support**

PhoneGap has support for accessing native features on a wide range of platforms. jQuery Mobile has active support for a wide range of platforms and web browsers and also receives a rating of Good. Both Sencha Touch and Appcelerator Titanium mainly target the Android and iOS operating systems, but as these have such a strong market share they receive ratings of Acceptable.

jQuery Mobile offers no native support, other than that offered by some web browsers, and Sencha Touch also has limited native support, so they
both get a rating of Bad. PhoneGap and Appcelerator Titanium both have good support for accessing native features. However, PhoneGap has support for native features on a lot more platforms and therefore receives a higher rating.

Web application

A jQuery Mobile application is a pure web application and therefore gets a rating of Good, but Sencha Touch also runs natively in a web browser and gets the same rating. The documentation for Appcelerator Titanium says it is possible to package and build an application developed using their SDK for the Web, however it is not clear what adaptations has to be made and it therefore receives a rating of Acceptable [22]. A PhoneGap application is built using HTML, CSS and JavaScript, but as PhoneGap is primarily used to access native features of mobile devices, which is not possible through a web browser, it receives a rating of Bad.

User interface

jQuery Mobile has the best user interface of the frameworks, being easily built using markup and being very feature rich. Sencha Touch also has a well developed user interface, but as it is declared programatically it is more tightly coupled to the framework. PhoneGap offers no support for building the user interface, and therefore gets a rating of Bad. Appcelerator Titanium offers support for using native user interface features, but as the user interface then has to be created for each platform it only receives a rating of Acceptable.

3.3 Selecting a method

All of the frameworks presented in the previous section have their strengths and weaknesses. jQuery Mobile is a widely used web framework for mobile devices, used for the official mobile web pages of a lot of larger companies [17]. PhoneGap’s open-source Cordova API is also very widely used and feature rich. For example, the Marmalade cross-platform SDK includes an implementation of the API and Sencha Touch’s documentation suggests that its own less feature rich Native API can be complemented by it [23] [21].

Sencha Touch is interesting as it is offering a sort of combination of jQuery Mobile and PhoneGap, meaning it includes both a user interface optimized for touch screens and an API for accessing native features of devices, however quite limited. It is also interesting since the application is built up entirely programatically using JavaScript. This, however, makes the application closely tied to the Sencha Touch SDK. A strong point with Sencha Touch is the possibility to package the application into native applications for Android and iOS using its SDK. But the fact that it only supports two
operating systems, even though the two with the largest market shares, makes it less appealing.

As it is unclear what adaptations has to be made to make an Appcelerator Titanium application work as a web application, as well as the fact that the application will be very closely tied to the Appcelerator Titanium SDK, it is not an appealing alternative.

Therefore, as shown in Table 3.1, using jQuery Mobile together with PhoneGap is arguably the strongest alternative for developing the application, as they get the highest rating overall. jQuery Mobile gets the best rating for user interface, being easily built using markup and producing a pure web application. PhoneGap’s Cordova API is well documented and has support for a lot of different native features on many platforms, using only one codebase. PhoneGap also makes it possible to package the application for several different platforms, and therefore gets the best rating for native support.

The weak side with using jQuery Mobile together with PhoneGap is that as a web application is to be developed besides the native application, as described in Section 1.3, the JavaScript functionality related to PhoneGap’s Cordova API has to be removed and rewritten. However this has more to do with the web application being different in functionality than being a downside with choosing this alternative. This should be possible to resolve by building the application modularly. The fact that both jQuery Mobile and PhoneGap support several mobile operating systems well, and gets the best rating in OS compatibility, also makes them the strongest alternative from the aspect of portability, which in fact is what you are aiming for with multi-platform development.

3.3.1 Decision

As discussed in the previous section, it was decided that the following frameworks are best suited for developing the application described in Section 1.3, and were therefore chosen for the application development:

- **jQuery Mobile** - To develop the user interface for the application.
- **PhoneGap** - To package the application as a native application for the Android and iOS operating systems and access native features using the Cordova API.
Chapter 4

Application development

This chapter includes a practical evaluation of multi-platform development for mobile devices, primarily aimed at devices running the Android and iOS operating systems. In the previous chapter the frameworks jQuery Mobile and PhoneGap were chosen to develop an application, described in Section 1.3, for leaving an expression of interest for work and thesis work at Combitech. This chapter describes the development process using these frameworks for multi-platform development.

4.1 Development plan

As this is a practical evaluation done under limited time, external code libraries was be used whenever possible, instead of writing all functionality from scratch. The development did not follow a strict model, but was divided into three development iterations and a final testing phase, as described in the Section 1.4.2.

4.2 Test plan

As the application is supposed to work as a native application on the Android and iOS operating systems, and preferably also as a web application on at least these two operating systems, the application had to be tested thoroughly. During development, testing was primarily done both in the web browser Google Chrome on the development computer running Windows 7, as well as on the SGSII Android phone. After each iteration, more extensive testing was done on the SGSII to ensure that the functionality that was added during the iteration worked as intended, and a test report was written. The application was then tested thoroughly on both the Android and iOS operating systems after the last development iteration, as described in Section 1.4.2.
4.3 Development environment

The application was developed primarily on a computer running Windows 7. The documentation for jQuery Mobile \(^1\) and jQuery \(^2\) was used, together with their JavaScript libraries for developing the user interface and core functionality of the application using JavaScript, HTML and CSS. For accessing the native features of the different devices and packaging the application into native applications for the Android and iOS operating systems the PhoneGap Cordova API \(^3\) and its documentation was used. As this requires the installation of the Android SDK \(^4\) and the iOS SDK \(^5\), they as well as the documentation for them were also used.

For writing the HTML markup, JavaScript code and PHP code the Notepad++ text editor \(^6\) as well as the Eclipse IDE \(^7\) with the JDT plug-in for JavaScript, Web Tools plug-in for HTML and PDT plug-in for PHP was used. The JavaScript code for the web application and user interface part of the application was then tested and debugged primarily in the Google Chrome web browser running on a computer running the Windows 7 operating system. Google Chrome provides a JavaScript console for debugging and analyzing JavaScript code. A Samsung Galaxy S II phone (SGSII) running version 4 of the Android operating system was used as a primary testing device for the native features of the application, but the application was also later tested on a device running the iOS operating system.

4.4 Frameworks

4.4.1 jQuery Mobile

When building applications with jQuery Mobile, one creates an HTML document and first includes a CSS file for the jQuery Mobile Framework and then two JavaScript files, one for the jQuery JavaScript library and then one for the jQuery Mobile JavaScript library. Then one defines one or more “pages”, which includes user interface widgets and information, all using markup in the HTML document, as shown in Listing 4.1. The user interface is declared using normal HTML user interface elements, such as forms and buttons, which are then touch screen optimized, as well as some user interface elements that are declared using HTML and then converted into interactive user interface elements by the jQuery Mobile JavaScript library.

\(^1\)http://www.jquerymobile.com
\(^2\)http://www.jquery.com
\(^3\)http://www.phonegap.com
\(^4\)http://developer.android.com
\(^5\)https://developer.apple.com/devcenter/ios
\(^6\)http://www.notepad-plus-plus.org
\(^7\)http://www.eclipse.org
Listing 4.1: A single-page template for a jQuery Mobile application.

As the jQuery JavaScript Library is required for using jQuery Mobile, one has access to all of the features of jQuery when building a jQuery Mobile application, which helps a lot when e.g. manipulating the DOM (Document Object Model). jQuery Mobile then gives access to some JavaScript methods that are specific to the mobile framework, such as changing between pages with support for animated page transitions. jQuery Mobile also gives access to some events using JavaScript, such as touch screen gestures like swipes, and orientation changes of the screen of the device running the application.
4.4.2 PhoneGap

When using PhoneGap to package an HTML/CSS/JavaScript-based application into a native application, one gains access to native features of the different platforms through the PhoneGap Cordova JavaScript API. This is simply done by including a JavaScript-file into the web application and then using the functions provided by it. As the API is platform-neutral, one does not have to write several versions of the application for the different platforms, as illustrated in Figure 4.1. One only has to check if the functionality to be used is available on the platform the application is running.

Applications are then built for each platform, using its native SDK. The application is then run inside a "WebView" on the different native platforms, meaning it is run inside a native application environment on the platform, but is being rendered by a Web Browser which is also interpreting the JavaScript code of the application.

PhoneGap Build

PhoneGap also provides a build service, which allows developers to upload their application to the service, which will then build and package the application into an application store ready package for several of the supported platforms. At the time of writing, the service supported building for Apple iOS, Google Android, Windows Phone 7, Palm, Symbian and BlackBerry. The service is free for an unlimited number of public applications, and one private application with one collaborator (account with access to the service). A running license is required if support for more collaborators or private applications are desired [24]. The PhoneGap Build service was tested during this thesis project, and worked very well.
4.5 Architecture

The application is built modularly and divided into three layers as illustrated in Figure 4.2. First is a common user interface layer, built using jQuery Mobile, jQuery, HTML, CSS and JavaScript. Second is a common shared fundamental application logic layer built using JavaScript. Third is the application specific logic built using JavaScript, which differs between the native application and the web application and is interfaced with the previous layer. In the case of the native application, this layer interfaces with the local filesystem using the Cordova JavaScript API. In the case of the web application this layer interfaces with a PHP script running on the web server hosting the application. The priority 3 requirements for extended functionality to the native application may affect all layers and therefore break against the modular architecture.

Figure 4.2: The architecture of the application.
4.6 Development process

This section describes the development process with the selected method for multi-platform development and how the development went.

4.6.1 Iteration 1

A mock up of the user interface, consisting mainly of a touch optimized HTML form, was created using jQuery Mobile and Codiqa. After that, some modifications to the mock-up user interface was made, using both JavaScript and HTML. The JSON format was chosen for saving the data, as it is suitable when working with JavaScript. The jQuery library including a plug-in called jQuery Validation was used to write JavaScript code to validate the entered form data and then convert it to the JSON format when the form is submitted. The user interface and form validation was tested in the Google Chrome web browser during the development and also in a Samsung Galaxy S II phone running Android 4.0 (SGSII).

An interface was created between the fundamental JavaScript application logic to make a separation between the native application and the web application easy. After this the work began on saving the JSON-formatted form data onto a file on the local file system on the device running the application. This was done by extending the application to interface it with PhoneGap using JavaScript and PhoneGap’s Cordova API to write to the local filesystem. The functionality was tested on the SGSII by packaging the application into a native application for the Android operating system using PhoneGap. It was confirmed that submitted form data was appended to a preconfigured file on the local filesystem in the JSON format.

4.6.2 Iteration 2

In Iteration 2 of the development, the application created during Iteration 1 was adapted to a web application that could be opened in a web browser on a mobile device. With this web application, expressions of interests will be sent to the server hosting the web application. Therefore the application created during Iteration 1 was first adapted to being a web application by instead of interfacing the fundamental application logic with the PhoneGap Cordova API, write new JavaScript code to instead post the form data to a PHP page on a web server when an expression of interest is sent.

A development web server with PHP and a MySQL database was set up to be used for the application. The JavaScript application was adapted to send the form data in the JSON format to a PHP page on the web server. A PHP page was then developed to receive the form data in the JSON format, validate it and write it to a file on the web server in the same JSON format.

---

8http://www.codiqa.com/
9http://docs.jquery.com/plug-ins/Validation
This functionality was tested both from a desktop computer running the web application as well as the SGSII. It was confirmed that the form data was being sent to the web server and saved locally in a file.

Then a MySQL database with appropriate tables and fields were set up on the web server. The PHP script was updated to also add an entry in a database table with information from received expressions of interest. The functionality was then tested to ensure the correct data was saved in the database. After that, the web application was tested on the SGSII to ensure the web application worked as expected on a mobile device.

4.6.3 Iteration 3

The user interface created in the first iteration was polished to look more attractive and to feel more like an interactive application rather than an HTML form. This was done by splitting up the jQuery Mobile user interface into several pages, with sliding page transitions between them. The possibility to change between the different pages by swiping the finger over the screen was added.

After this, the way the form data was saved in the JSON format was slightly altered to make the form data more adaptable, as it was desired by Combitech to make changes to the fields in the forms easier.

Then a small web application was built using JavaScript with the jQuery and jsRender libraries and PHP to present the JSON Data that is saved by the native and web applications. After this, time was running out and the final testing phase begun, unfortunately without all priority 3 requirements being fulfilled.

4.7 Issues

This section describes issues that were experienced during the development process. One issue was that debugging of code using the PhoneGap Cordova API was not possible when testing the application on the SGSII. However, messages written to console in JavaScript were written to the Android SDK LogCat console by the Cordova API, which was a nice feature that helped alleviate this problem.

There were a lot of issues with the jQuery Validation plugin for jQuery. It was decided to use this library to save time during development, instead of developing form validation functionality from scratch. However a lot of time was wasted on troubleshooting the plugin, so it is uncertain if any time was saved by using it. The main issue was rooted in that the documentation for the plugin was very sparse, and that the plugin was not actively developed and updated, which is common for third party libraries.

Another issue related to the chosen method for multi-platform development was with the user interface development. As different browsers renders web pages differently, it can be really time consuming to get the design to
look coherent across different browsers as well as making the design look as desired when it was being built using HTML, CSS and JavaScript.

A time consuming issue was setting up an environment with a MySQL database and PHP scripts, which was not originally a part of the goals for this thesis but consumed a lot of time.

There were also some issues with jQuery Mobile, mainly concerning some bugs in the framework, which had been reported but not fixed in the latest available release of the framework.
Chapter 5

Results

This chapter contains the results from the development phase of the thesis project. The chapter begins by presenting the resulting application and then presents test reports showing that the requirements on the application were fulfilled.

5.1 Application

This section describes the resulting application that was developed during this thesis project, its functionality, structure and limitations.

5.1.1 Structure

The application is built up of a number of files, as shown in Listing 5.1. First and foremost is the "index.html" file, which is built using jQuery Mobile and is the start-point of the application. It includes several JavaScript and CSS-files. First it includes the library files. These are contained in the "lib"-directory. It includes one JavaScript-file for jQuery, one JavaScript-file for jQuery Mobile, one JavaScript-file for the jQuery Validation plug-in, one JavaScript file for localization of the jQuery Validation plug-in and on the native version of the application also one JavaScript-file for the PhoneGap Cordova API. Then it includes two custom JavaScript-files written during this thesis project, contained in the "app" directory. First one JavaScript-file for the shared application logic and then one JavaScript-file for the application specific logic, which is different for the native and web application versions of the application. Two CSS-files are included, one for the jQuery Mobile framework in the "lib" directory and one for custom CSS for the application in the "app" directory.
5.1. APPLICATION

The application’s user interface is entirely in Swedish, however it is prepared for being made available in other languages as well. The user interface is built up by a greeting page where users can select their interest, in either or both of work and thesis work. After this, the user can change to the next step by either swiping their finger across the screen or pressing a “next” button, and the page will change after an animated sliding page transition. A couple of pages follow, where the user can enter among other things contact information. A page indicator at the bottom of the pages indicates what page the user is at. If the user has not filled out all required parts of the form at each page, an error message is shown and the user is not able to change to the next page in the application until all required parts are filled out. When the user enters the last page of the form, they can enter a short description of themselves and then choose to send the expression of interest to Combitech. When they do so a loading screen appears, and if the expression of interest was successfully sent, a message saying so is printed and the user gets taken back to the greeting page after an animated page transition. Otherwise, an error message is shown, and the user can try to send the the expression of interest again.

Listing 5.1: The application file structure.

```
app
  ..app.js
  ..app.handler.js
  ..my.css
lib
  ..images
  ...
  <imagefiles>
  ..cordova−1.7.0.js
  ..jquery.mobile−1.1.0.min.css
  ..jquery.mobile−1.1.0.min.js
  ..jquery.validate.min.js
  ..jquery−1.7.1.min.js
  ..messages_se.js
index.html
```
Native application

When the application is run as a native application, it takes up the whole screen without showing any address bar. Figure 5.1 shows the native application running on the Android operating system. It therefore is both launched and runs as a purely native application. When the expression of interest is sent in the last step of the form, it is in fact saved to a file in the local filesystem in the JSON format.

![Native Application on Samsung Galaxy S II](image)

Figure 5.1: The application running as a native application on a Samsung Galaxy S II Android smartphone, in a resolution of 480x800 pixels (including the status bar).
5.1. APPLICATION

CHAPTER 5. RESULTS

Web application

When the application is run as a web application, it is accessed through a web browser. Figure 5.2 shows the web application running in the default web browser on the Android operating system. When the expression of interest is sent in the last step of the form, a connection to a PHP script is opened in the background and the form data is sent to it in the JSON format. This is invisible to the user, as they never navigate to the script, and an error message is shown in the loading screen if the connection to the PHP script did not work as expected.

![Web application screenshot](image)

Figure 5.2: The greeting page of the application, rendered by the default web browser on a Samsung Galaxy S II Android smartphone.
5.1. APPLICATION

CHAPTER 5. RESULTS

5.1.3 Limitations

As not all priority 3 requirements were fulfilled during this thesis project, the functionality of the application is quite limited. As shown in Figure 5.1-5.3 the user interface is rendered slightly differently on the different platforms. Therefore more work could be done on the user interface, making it more attractive and consistent across the different platforms. However, this is somewhat limited by the fact that it is a web application in its core, and different web browsers will thereby render the user interface differently, which is a downside with this technique for multi-platform development. It should however be possible to minimize the inconsistencies across different web browsers using CSS and JavaScript.

Figure 5.3: Error messages being shown by the web application. Rendered by the Google Chrome web browser running on a Windows 7 computer. The window was set to a size of 480x800 pixels.
5.2. TEST REPORTS

5.2 Test reports

As described in the previous chapter, test cases were written after each iteration to test the functionality added during the iteration. Test reports were then written for these test cases. During the final testing phase test cases for the application was written and tested on devices running both the Android and iOS operating systems. This section contains some of the test reports from these tests, that shows that the requirements described in Section 1.3.1 were fulfilled.

5.2.1 Iteration 1 - Native application

Requirement 1.1

Purpose Make sure that the application is developed using multi-platform development.

Result The application is developed using the frameworks jQuery Mobile and PhoneGap with JavaScript, CSS and HTML. It can therefore be packaged into native applications for several platforms and is therefore developed using multi-platform development.

Requirement 1.2

Purpose Make sure that users of the application are able to fill out an expression of interest for work and thesis work, containing among other things contact information, information about their education, their interest areas and a short description of themselves using the application.

Environment The application was installed and run on a Samsung Galaxy S II phone running version 4.0.3 of the Android operating system.

Procedure

- Launch the application.
- Inspect the user interface.

Result The application contains several pages with forms for filling out an expression of interest for work and thesis work at Combitech.

Requirement 1.3

Purpose Make sure that expressions of interest given using the application are saved locally on the device running it, in a suitable data format such as JSON or XML and that the data is possible to be read from the device, outside of the application.

Environment The application was installed and run on a Samsung Galaxy S II phone running version 4.0.3 of the Android operating system.
5.2. TEST REPORTS  

CHAPTER 5. RESULTS

Procedure  
- Launch the application  
- Fill out the forms  
- Send the expression of interest  
- Look for a file named combitech.json in the local filesystem on the device  
- Open the file and view its contents  

Result  
The forms were successfully filled out and the expression of interest was successfully sent using the application. A file named ”combitech.json” was found under ”/mnt/sdcard” in the local filesystem. It contained the sent expression of interest in the JSON format.

5.2.2 Iteration 2 - Web application  

Requirement 2.1  

Purpose  
To make sure that the application is available as a web application for mobile devices, where expressions of interest instead of being saved locally are sent over the Internet to a server that receives and saves them in database.

Environment  
A web application version of the application was available on a running web server. It was accessed from a Samsung Galaxy S II phone running version 4.0.3 of the Android operating system using the stock web browser.

Procedure  
- Navigate to the web server hosting the application.  
- Fill out the forms  
- Send the expression of interest  
- Inspect the database on the server  

Result  
The application was successfully loaded in the web browser and had the same functionality as the native application. The forms were successfully filled out and the expression of interest was successfully sent. The database on the server contained the sent expression of interest.
5.2. TEST REPORTS

CHAPTER 5. RESULTS

Requirement 2.2

Purpose To make sure that the web application works in default mobile web browsers on mobile devices running the Android and iOS operating systems.

Environment A web application version of the application was available on a running web server. It was accessed from a Samsung Galaxy S II phone running version 4.0.3 of the Android operating system using the stock web browser, as well as an Apple iPad using the default Safari Mobile web browser.

Procedure

• Navigate to the web server hosting the application.
• Fill out the forms
• Send the expression of interest

Result The forms were successfully filled out and the expression of interest was successfully sent using the application on both the default web browser on the Android operating system, and using the Safari Mobile web browser on the iOS operating system.

5.2.3 Iteration 3 - Extended native application

As none of the priority 3 requirements were fulfilled during Iteration 3, other than a simple web application for viewing expressions of interests sent to the web application, only that feature was tested after Iteration 3.

Requirement 3.2

Purpose To make sure that the web application for viewing saved expressions of interests works as intended.

Environment A web application for viewing saved expressions of interests was available on a running web server. It was accessed from a Samsung Galaxy S II phone running version 4.0.3 of the Android operating system using the stock web browser.

Procedure

• Navigate to the web server hosting the application for viewing saved expressions of interests.
• Log in using a predefined password
• View saved expressions of interests

Result The saved expressions of interests were successfully displayed.
5.2.4 Final testing

There were some issues with testing the application as a native application on an iOS device. An Apple iPad was available for testing the application. However, to be able to compile and install the application on the device, both using the iOS SDK and Xcode and using PhoneGap’s Build service, a paid subscription for the Apple Developer Program was required. A subscription was not retrieved in time and the application was therefore never tested running as a native application on an Apple iOS device, but instead only as a web application using the preinstalled Safari Mobile Web Browser, as shown in the test case for requirement 2.2.

Nevertheless, since the web application shares the exact same user interface and fundamental application logic, as explained in the Architecture section of the Application development chapter, and the native application works on an Android device, as shown with the test cases for requirement 1.1-1.3, it is highly unlikely that the application would not work as a native application on an iOS device as well. Especially as PhoneGap officially supports writing to the local filesystem on both the Android and iOS operating systems using the same API functions.

No inconsistencies were noticed when running the application on the iPad, compared to running it on the Android-based Samsung Galaxy S II phone, apart from the user interface being rendered slightly differently.
Chapter 6

Conclusion

This chapter contains conclusions that can be drawn from this thesis, as well as a discussion on the shortcomings of the thesis project.

6.1 Summary and Conclusions

There are mainly three conclusions that can be drawn from the thesis, in regard to its purpose.

- There are many alternatives for multi-platform development of applications for mobile devices.

As concluded in Chapter 3, there are over 50 different multi-platform development frameworks targeting mobile devices. This can be seen as a verification of the fact that the large spread in the market shares of operating systems for mobile devices, and the large differences in application development for them, is a problem.

These multi-platform development frameworks can according to Ohrt and Turau [12] be divided into two main groups, those who interpret platform-independent code on the different platforms, and those who compile platform-independent code to native code for the different platforms. However another possible division could be into three groups.

The first group being frameworks such as MonoTouch and Marmalade SDK that allow application development in a high level language, such as C++ or C#, and then compiles into native code for the different platforms. They may require some double work with interfacing with the different platforms by only providing an interface API to the different platforms SDKs. This gives the upside of e.g. being able to use the different platforms native user interface elements, as well as the possibility of less performance overhead, which may be introduced by an interpreter.
The second group being frameworks such as RhoMobile and Adobe AIR that allow the whole application to be developed in a more platform independent way, often with a user interface that is not native to the different platforms, but instead being consistent between the different platforms. They then either package an interpreter with the application or require the installation of an external interpreter.

The third group being a special case of the second group, namely frameworks such as Sencha Touch and PhoneGap, that allow HTML/CSS/JavaScript-based web applications to be developed to either be accessed from web browsers on the different platforms, and/or packaging them into native applications with access to native features through JavaScript APIs.

Frameworks of the first group may therefore be more suitable to applications having a heavy application logic and where performance is important, but where a lot of user interface development is not required, such as e.g. games. The second and third group may be more suitable to applications that do not have a very heavy application logic or where performance is not very critical, or where a user interface that is consistent between platforms is desirable. The third group having the special upside of the possibility of allowing the application to run as a web application as well.

- HTML/CSS/JavaScript-based multi-platform development provides nice user interfaces through e.g. jQuery Mobile with access to native functionality through e.g. PhoneGap.

In this thesis, four different frameworks belonging to the third group were analyzed more closely; jQuery Mobile, PhoneGap, Sencha Touch and Appcelerator Titanium. The frameworks jQuery Mobile and PhoneGap were chosen to develop an application, and thereby analyzed from a more practical point of view as well.

jQuery Mobile was used to build the user interface for the application, providing a touch screen optimized and relatively consistent user interface in function and look between platforms. Some fundamental JavaScript application logic was then added to the web application developed with jQuery Mobile.

- HTML/CSS/JavaScript-based applications are easily modularised to be able to be run both as native and web applications.

The application was to be available as a web application and a native application, with slightly differing functionality. This was easily accommodated by writing a common JavaScript interface that separated these two applications, but allowing the reuse of the same user interface and fundamental JavaScript application logic in both of them.

In the native application, PhoneGap’s Cordova API was called through the interface to provide access to native features, in this case to write to the local filesystem on the device. In the web application, a PHP script located
on a web server was instead called through the interface to provide access to a MySQL database running on the web server.

Building applications for mobile devices with HTML/CSS/JavaScript-based frameworks therefore allows this kind of modularisation, where the application can easily be adapted to run as both as a web application and a native application. Some frameworks may even allow applications to run as web applications without having to make any changes to the code.

6.2 Discussion

This section contains a discussion regarding the shortcomings of the thesis project as well as multi-platform development for mobile devices in general.

6.2.1 The developed application

As the thesis project was done under limited time, and included an investigation on multi-platform development, the application development was done under very limited time. Therefore the application could have used more work to fulfill more of the priority 3 requirements, described in Section 1.3.1, and have been tested more thoroughly.

jQuery Mobile and PhoneGap proved to be a good choice for the application, as there was not a lot of work required to adapt the native application that used PhoneGap's Cordova API into a pure web application. However, e.g. Sencha Touch may have been able to make this even easier, without having to make adaptations to the code, and instead just build it as a web application. Had more time been available, the different frameworks could have been analyzed more thoroughly before starting the application development, which could have led to a different choice of framework for the development.

The application serves though as a nice proof of concept for multi-platform development of applications for mobile devices.

6.2.2 Native vs portable applications

If we look at desktop computers, web applications are now a natural choice for many types of applications that could instead be run as native applications directly in the operating system. Web applications have the large advantage of not requiring installation on the computer and being available on different operating systems, with the downside of normally requiring a network connection.

As mobile devices are network oriented, often being constantly connected to the Internet, web applications should be well suited to them as well. One downside with web applications developed for use on desktop computers is that their user interface may be difficult to use on mobile devices that have touch screens instead of a mouse and keyboard, and smaller screens with
lower resolution. The advent of responsive design, described in Chapter 2, and touch screen optimized web application frameworks such as jQuery Mobile may however transcend this problem.

When it comes to access to sensors on mobile devices, which is something a lot of native applications make good use of, the W3C is working to provide standardized APIs for web applications to access them, as described in Chapter 2.

As explained in Chapter 3, a shortcut to a web application may be saved in the same manner as a shortcut to a native application on the Android and iOS operating systems (and probably others as well), which can lead to users not noticing the difference between a web application and a purely native application.

This may lead to web applications being able to provide the exact same features as native applications on the different platforms, while being developed in a single project with a single code base for all platforms.

However, a native application may provide features, such as a consistent user interface across applications for the operating system. More importantly, native applications may provide better performance, which has not been analyzed in this thesis. An interpreter may introduce an overhead in the performance when executing interpreted code.

6.2.3 Future of multi-platform development

If web applications have user interfaces that function equally well with those of native applications on mobile devices, and have access to the same features, such as sensors, they may be a more natural choice for some types of applications for mobile devices in the future. This may lead to the same scenario as on desktop computers, where e.g. applications that mainly provide information through a network connection may primarily be developed as web applications.

6.3 Future work

Here are some suggestions for topics of future work that can be done on the subject of multi-platform development for mobile devices:

- Performance comparison between HTML/CSS/JavaScript-based applications and native applications.
- User experience comparison between HTML/CSS/JavaScript-based applications and native applications.
- Development process comparison between running two (or more) native development projects and one multi-platform development project.
- Investigation of the possibilities of usage of native functionality such as sensors and location services from a mobile web application.
Bibliography


På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida: http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Mattias Sjödin