Chapter 1

Introduction

A web application has a lot of advantages compared to mobile applications. They run on all plattforms have instant updates and are simple to maintain [5]. However, a web application can't access some native mobile features such as the mobile's accelorometer. A company who want to access such native features need to develop a mobile application. In addition as much as 76% mobile applications users agree that all companies should have mobile applications to make interacting with the companies easier [4].

Developing a mobile application can be very expensive. Small to medium sized project for an Android mobile is estimated to cost from 20 000 - 40 000 dollars [3].

If a company has a web application they can choose to develop a mobile application from the ground up. The users activity in the mobile and web application can be synchronized with by the use of the same back-end interface, see figure **??**. That will cost them a huge amount of development time and also a lot of time maintaining several platforms.

Another, less costly alternative, is to create a mobile application using the existing web application. The web application is run within a shell of the mobile application, see figure **??**. The web application can then access the mobile's native functions. This will save development time and instead of maintaining several platforms for every feature the mobile application only need to be maintained in regards of the use of the native functions.

The aim of this thesis is to evaluate two different development methods for encapsulating an existing web application in a mobile application to utilize native functions. With focus on evaluating development effort.

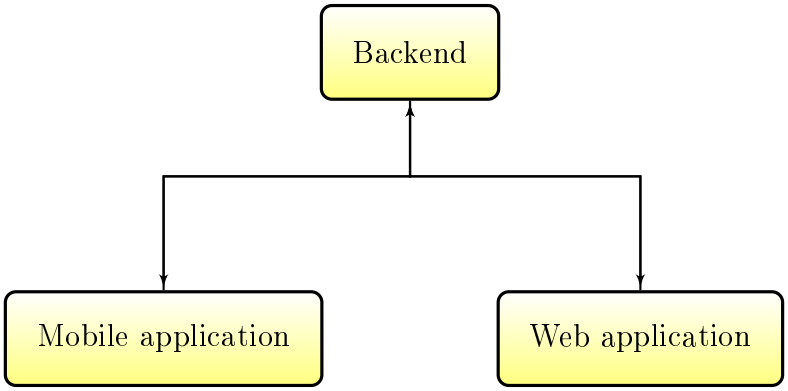

Figure 1.1: Seperate mobile and web application connected with a common backend.

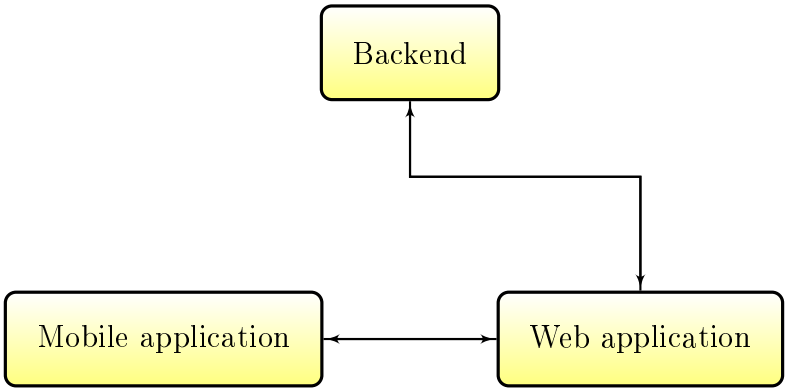

Figure 1.2: Seperate mobile and web application connected with a common backend.

1.1 Motivation

There are a number of researches of developing a native or hybrid mobile application using different methods. The researches focuses on aspects such as cost, effort and advantages and disadvantages for developing an entirely new mobile application. However, it is very hard to find research on extending an existing web application with mobile native features.

The cost for developing a small to medium sized project in native Android or PhoneGap is estimated to cost around 20 000 - 40 000 dollars[3].

One of the huge advantages developing in PhoneGap compared with developing natively is that PhoneGap can be compiled to multiple platforms. Making PhoneGap an attractive development framework for companies with less development resources. If a mobile application is developed natively for Android, IOs and Windows Phone, maintaining the mobile application must be done in three different native applications. If the mobile application is instead developed in PhoneGap only one application needs to be maintained.

A disadvantage of developing in PhoneGap is when their is a use of many native features. PhoneGap relies on a development framework and the provided features when building a mobile application. Hence, if the framework is not up to date with the latest new features, the developer will not be able to partake of the features until the framework is updated. If the application is dependent on many native features a hybrid application may have limitations[3].

1.2 Problem formulation

The goal of this project is to evaluate two different development methods for encapsulating an existing web application in an Android mobile application. When evaluating the methods the focus is on development effort. The purpose of the mobile application is to give the web application access to the mobile's native functions. An example of such a native function would be accessing the mobile's accelerometer. It is important that the logic of the web application can be written in a general way (non-platform dependant) adapting to the device accessing the web application. So that the same web application code is used for the web application and mobile application.

Antoher goal is that the mobile application only provides data from native functions. In order to achieve that the web application and mobile application should have a master slave relationship. Where the web application acts as the master and the mobile application as the slave. To get data from the mobiles native functions the web application asks for the data and the mobile application passes the data back.

To enable this master and slave relationship the mobile and web application layer must be able to communicate. Passing commands and data between the layers. It is

important the mobile and web application layers has a way of communicating that is developer friendly.

To limit the scope of the research the development methods is only evaluted for the mobile operating system Android. Android was choosen since it is the most widely used today for mobiles.

There are two development methods of creating an Android application that will be evaluated. The first development method is to develop natively in Android. The second method is to develop using the framework PhoneGap.

The methods will be evaluated and compared from a developers perspective, with the preconditions defined section 1.2.1.

1.2.1 Preconditions

The study is done from a company's perspective and is based on the following prerequisites:

- There are only a few developers, three or less.
- There is already an existing web-application that is to be used by the mobile-application.
- The company has a need and/or desire to extend the functionality of it's web-application with native functions.

1.3 Contribution statement

Throughout this work we have been working together very closely. We have reviewed and improved each others work continuously. During development we often been pair programming. With that said we can attribute parts of our work to one of us more then the other. Even though most of the work has been created together.

Philip designed the code of the web application and the mobile application built natively in Android. David designed the code for the PhoneGap application. David researched different techniques for the communication between the PhoneGap layer and the web application layer. David wrote the Approach section and Philip wrote the Introduction section in this paper.

1.4 Report organization

1.5 Terminology

Mobile application
Refers to the application being developed, excluding code loaded from remote websites.

Web application
If nothing else is specified it refers to the client-side of the web application.

Development methods
Refers (in the context of this paper) to the methods of mobile application development methods being evaluated. I.e developing the mobile application with PhoneGap or natively in Android.

Native function
A hardware function which a device has. Lets say a mobile has a camera and an accelerator. When writing software for that mobile there are functions to interact with the camera and accelerator. Those functions are called native functions.

Web/Mobile application layer
Refers to the application layer belonging specifically to the mobile application or the web application. A mobile application can encapsulate a web application which then is a a part of the mobile application. The mobile application then consists of a mobile and web application layer.

1.6 Background

Android is an operating system used on a wide range of devices. For example a mobile device or tv device. Developing Android applications are written in the programming language Java.

A hybrid application is a native application which is partially written with web technologies. I.e HTML5, CSS and JavaScript. The part of the application written with web technologies runs within a browsers engine in a so called WebView. A web application running in a browser does not have access to a device native functions such as the bluetooth. However a website running within a WebView can through interaction with the application's native code access a device native functions.

PhoneGap is a framework for developing mobile applications. The code is written with web technologies. The code can be compiled to different platforms, such as Android or IOs specific code. The resulting application is a hybrid application. An

example of a mobile application built in PhoneGap is Wikipedia's mobile application.

1.7 Source lines of code

Source lines of code is a measurement tool for software development. Source lines of code, also abbreviated as SLOC, is very easy to obtain and is a fairly accurate predictor of development effort[1, p. 63]. Measuring SLOC simply means you count the number of lines of code. There are many different ways to measure SLOC, such as Halsted's approach, function points, physical SLOC and Logical SLOC.

Physical SLOC is the length of the code excluding comments and blanks. Function points measure functionality and can therefore be measured before the design and coding if the requirement specification is complete[1, p. 187]. Halstead's uses measurable properties such as operands and operators and uses them to identify properties of software. Such as the length, difficulty and effort of the program. Fenton and Bieman describes Halstead's software science measures as a confused and inadequate measurement. Particularly for other attributes then size[2, p. 345].

Logical SLOC measures the number of statements that carry over one or more physical lines. For languages with terminators, this can be counted more easily and quickly. As an example, in Java you could count the logical SLOC by counting the number of line-terminating semicolons and closing curly brackets. Logical SLOC represents the programming instructions and data declarations which are converted into executable instructions, i.e. the implementation of the software design. Another positive aspect of of logical SLOC is that it better handles differences in formatting and style conventions than physical SLOC[1, p. 155].

To compare size between two different languages a size conversion table can be used. The table can be used to estimate how many SLOC a program coded in one programming language would have in another language. In the table constructed by Galorath and Evans you can compare a third generation language, a fourth generation language, Ada, Assembly or Pascal[1, p. 163]. A third generation language compared to another third generation language would have no conversion rate.

Source lines of code is a measurement tool for software development. Source lines of code, also abbreviated as SLOC, is very easy to obtain and is a fairly accurate predictor of development effort[1, p. 63]. Measuring SLOC simply means you count the number of lines of code. There are many different ways to measure SLOC, such as Halsted's approach, function points, physical SLOC and Logical SLOC.

Physical SLOC is the length of the code excluding comments and blanks. Function points measure functionality and can therefore be measured before the design and coding if the requirement specification is complete[1, p. 187]. Halstead's uses measurable properties such as operands and operators and uses them to identify properties of software. Such as the length, difficulty and effort of the program. Fenton and Bieman describes Halstead's software science measures as a confused and

inadequate measurement. Particularly for other attributes then size[2, p. 345].

Logical SLOC measures the number of statements that carry over one or more physical lines. For languages with terminators, this can be counted more easily and quickly. As an example, in Java you could count the logical SLOC by counting the number of line-terminating semicolons and closing curly brackets. Logical SLOC represents the programming instructions and data declarations which are converted into executable instructions, i.e. the implementation of the software design. Another positive aspect of of logical SLOC is that it better handles differences in formatting and style conventions than physical SLOC[1, p. 155].

To compare size between two different languages a size conversion table can be used. The table can be used to estimate how many SLOC a program coded in one programming language would have in another language. In the table constructed by Galorath and Evans you can compare a third generation language, a fourth generation language, Ada, Assembly or Pascal[1, p. 163]. A third generation language compared to another third generation language would have no conversion rate.

1.8 Related work

Bibliography

[1] Daniel D. Galorath and Michael W. Evans, *Software Sizing, Estimation, and Risk Management*, Auerbach Publications, Taylor and Francis Group, 2006.

[2] Norman Fenton, Queen Mary University of London, UK, James Bieman, Colorado State University, Fort Collins, USA, *Software Metrics: A Rigorous and Practical Approach*, CRC Press, Taylor and Francis Group, 3rd edition, 2015

[3] Bernard Kohan and Joseph Montanez, *Native vs Hybrid/PhoneGap App Development Comparison*, `http://www.comentum.com/phonegap-vs-native-app-development.html`, Comentum, San Diego, January 26, 201EffectiveUI5

[4] Online survey from Harris Interactive on behalf of EffectiveUI, United States, September 30 - October 4, 2010

[5] A white paper from Michaels, ross & cole, ltd, *Native mobile apps: The wrong choice for business?* `http://www.mrc-productivity.com/research/whitepapers/NativeAppsWrongChoice.pdf`, Lombard, IL, January 2013