UNIVERSITY OF GOTHENBURG

PhoneGap: Potentials of a Mobile Cross-Platform Application

Bachelor of Science Thesis in the Programme Software Engineering and Management

BATBILIG BAVUUDORJ
IONUT TRANCIOVEANU

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2013

PhoneGap: Potentials of a Mobile Cross-Platform Application

BATBILIG BAVUUDORJ
IONUT TRANCIOVEANU

Examiner: ROGARDT HELDAL

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2013

PhoneGap: Potentials of a mobile cross-platform application

Batbilig Bavuudorj
University of Gothenburg
Göteborg, Sweden
gusbavba@student.gu.se

Ionut Trancioveanu
University of Gothenburg
Göteborg, Sweden
gustraio@student.gu.se

Abstract - **In spite of its low performance over native development approach, cross-platform applications developed using PhoneGap framework are getting popular. On the other hand, company A is looking for a solution on informing their customers about current electricity interruption through a mobile application that can support multiple platforms. One way to solve the problem was focusing on the real world by building and evaluating an artifact using hybrid approach in cross-platform development. Results show that PhoneGap framework is a viable solution that saves considerable resources such as time, budget, and developer's effort yet can support multiple mobile platforms without or with little modification to the original code-base. Also, if the applications are crafted well it can be a strong alternative (to native applications) both in performance and usability for light business type applications.**

Keywords **-** ***cross platform; hybrid applications; mobile web; cross compiled applications; interpreted applications; PhoneGap; PhoneGap build***

.

1. INTRODUCTION

In the early stages of mobile application development, often one of the first constraints was whether an application was going to be portable across multiple mobile devices and which platform to support first. This issue may sound simple but it is important as it includes the following problems behind: development and support for all popular platforms, for various versions being very expensive in terms of budget and time; choosing a single platform would only limit the number of potential customers [1]. Mobile applications, as with any other software applications, are characterized by the output and customer satisfaction. Often the customer satisfaction in mobile products is defined by the product's usability and performance.

These problems are being solved by mobile cross-platform development using the web application technologies such as: HTML5, CSS and JavaScript. However, these mobile web applications are limited in taking advantage of native resources of mobile devices and they cannot be distributed through application markets [2].

Frameworks such as the PhoneGap bridges web applications and mobile devices by using standard web technologies along with native resources of mobile devices. It is known from the literature that the applications developed using PhoneGap framework perform slower than the pure native applications [4, 6, 20]. Despite this drawback the PhoneGap development is one of the most popular alternatives in cross-platform development framework [1, 20, 22]. Why is that so?

This research aims to identify the factors influencing user acceptance and popularity of PhoneGap framework by developing a prototype using the hybrid approach.

- What are the factors that influence user acceptance in cross-platform applications that use PhoneGap framework? Is PhoneGap framework a viable solution for light business type applications?

- What are the challenges involved in developing cross-platform mobile applications using PhoneGap framework?

The scope of the research is limited to cross-platform development. It excludes native development or other types of frameworks except PhoneGap. Target mobile operating systems are Android and iOS along with various version of the two platforms.

This research is done in collaboration with company A. The company specializes in the area of energy services, heating, and electricity supply. It is also interested in the feasibility of development of a low-cost mobile application that supports wide range of mobile operating systems. However, due to the limitations of specific native development knowledge and cost and effort required, the company did not have the opportunity to undertake this project.

2. RESEARCH METHOD

This research is based on the idea of 'problem-driven approach' and has four assumptions about an underlying problem: the problem should be framed, it should be an instance of a problem, it is stable, and it has unambiguous goals [3].

The method also falls into the framework of design science. Design science is the study of artifacts in context [3, 17]. In Hevner's design science framework an "environment is the source of design goals and a budget to achieve them.

In return, the design researcher delivers artifacts that can be used to solve problems in the environment, i.e. to achieve goals in the environment" [3]. Wieringa and Morali make distinction between activities in this framework such as "solving improvement problems" and "answering knowledge questions" [3].

A. Solving improvement problems - building and evaluating artifacts. The word "improvement" here means a difference between actual state of the world and the world as desired by a stakeholder.

B. Answering knowledge question - a lack of knowledge about an aspect of actual real world and it includes development of theories as well as development of rules of thumb or design guidelines [3].

The tasks of "solving improvement problems" include activities such as identifying relevant stakeholders, their goals, criteria for the improvement, designing a prototype to change the real world in the direction of a stakeholder. The evaluation of "solving improvement problems" includes an effectiveness criteria (has a change been achieved?), and utility (has the change led to an improvement?) [3].

This research activities include:

- to *review and analyze* existing literature about cross-platform mobile development and PhoneGap framework
- to *build* an IT artifact or prototype to address a stakeholders concern
- to *evaluate* the artifact to determine the progress if there is any
- to *determine* why and how the artifact worked or did not work

Evaluation

The evaluation of the prototype will be conducted in two phases. In the first phase, the prototype will be evaluated at the university environment and the user experience feedback will be collected. In the second phase, the prototype will be evaluated at the company A by key stakeholders and the necessary feedback will be collected through questionnaires (See Appendix A3) and interviews. Results from this evaluation will be analyzed and presented.

Research delimitations

This research is aware of the existence of alternative methods such as ISO Metrics and Product Reaction Cards for measuring user-experience and usability. However, the use of simple method approach is preferred due to the primary stage of the research along with time and budget constraints.

3. BACKGROUND / THEORETICAL FRAMEWORK

This chapter describes various cross-platform development approaches in the market today. First, an overview of the present mobile web approach and hybrid approach, and motivation for choosing the hybrid approach over the mobile web approach will be explained by contrasting the advantages and disadvantages. Second, a detailed discussion on PhoneGap framework, its architecture and inner workings is deliberated. Lastly, this chapter discusses about various tools, languages and libraries necessary for this research. These descriptions are intended to help the reader to get a broad overview of different development approaches for cross-platform and their tools.

3.1 Cross-platform mobile applications

Cross-platform application development can be categorized into four types by their development approaches. These are mobile web app, hybrid, interpreted, and cross-compiled approaches [4]. However, as the interpreted and the cross-compiled approaches are not related with this research, it compares mobile web application approach with hybrid approach.

3.1.1 Mobile web application approach

Mobile web applications are simply web pages that have application functionality, and they are invoked in the web browser [2]. Much of the data and logic is processed on the server and the client manages the user interface and the controller. These types of applications do not require installation and can be accessed through URLs. There are number of challenges that exist with the mobile web applications. To name a few, there are limited number of supporting mobile browsers, difference in screen resolutions of the devices, inaccessibility of native devices such as camera and GPS due to application sandboxing, and issues of distributability through application stores [2]. Nevertheless, it has advantages such as one point maintenance on the server side, no installation, and reusable UI.

3.1.2 Hybrid approach

The hybrid choice of development helps to build applications using mix of web and native technology. Hybrid application achieves this by using common mobile web implementation and website technologies (HTML, CSS and JavaScript) that are then wrapped or that run inside a native container which provide access to the native features of the mobile device platform [4]. The hybrid application uses the native browser engine to render and display the HTML content for the users. Unlike the Mobile web applications, the browser is embedded inside the native container application, exposing the application through implementation of the abstraction layer that exposes the device's application programming interface (API). Moreover, hybrid approach bridges the HTML pages with the native APIs of the device operating system[5].

The hybrid applications are slower in performance compared with the native applications because the execution is done by the browser engine. User interface of the hybrid applications cannot always be replicated, and to obtain the look and feel of a native application the platform specific styling may be required. Other limitations of the hybrid application are JavaScript's platform specific behavior and threading model incompatibilities [4].

Compared to mobile web applications, the hybrid applications can be distributed through application stores, downloaded and installed on the devices as native applications. The hybrid applications expose a common set of native features across different platforms. On the other hand, the user interface of a hybrid application is reusable across platforms like a mobile web application. The same application can also be used on different operating systems, without modifications and with the help of hardware abstraction layer.

3.2 PhoneGap

One of the most popular examples of a container that is designed for developing hybrid mobile applications is PhoneGap.

PhoneGap is an open source mobile development framework produced by Nitobi, and purchased by Adobe Systems under Apache 2.0 license. PhoneGap allows to build commercial and open-source applications for free, with the possibility to use any license combinations [6]. With a multitude of mobile platforms such as Android, iOS, Windows Phone OS's, Bada, BlackBerry, Symbian and webOS, developing applications in device-specific languages such as Java, Objective-C or other native languages is difficult and expensive. Through the use of the Apache Cordova library, PhoneGap enables software programmers to interface directly with the mobile device using standard web technology languages such as JavaScript, HTML5 and CSS3 [4]. "Apache Cordova is a set of device APIs that allow a mobile application developer to access native device function such as the camera or accelerometer from JavaScript" [9].

The elements from both native and web application are combined in PhoneGap, making it a hybrid type of application. The application runs in a native container which uses the mobile device's chromeless browser engine to render and process the HTML, CSS and the JavaScript, which makes it neither native nor purely web-based [2]. The application is not purely native since the layout rendering is done via a web-view instead of the native language of a specific operation system. It is also neither purely web-view based because the browser is embedded inside the native container application.

PhoneGap does not provide an IDE (e.g. Eclipse for Android, Xcode for iOS or Visual Studio for Windows Phone) to develop applications. Developers are allowed to choose an environment by themselves without a centralized environment. By adopting PhoneGap approach the application development and maintenance can be performed and enhanced on different operating systems prior the executable application [6].

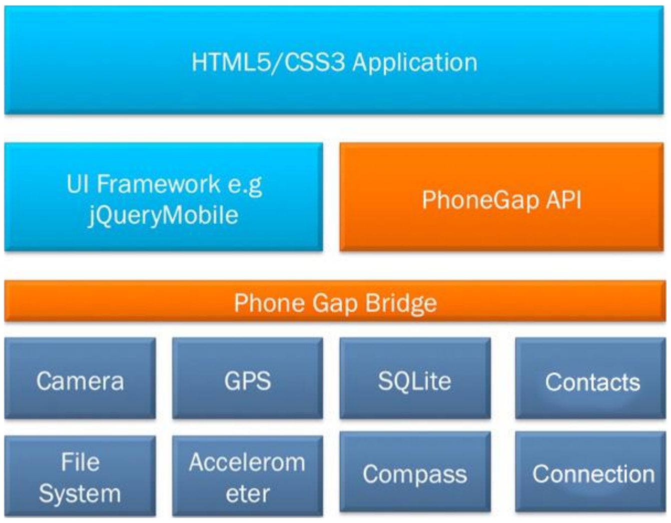


Fig. 1. Generic PhoneGap high level application architecture

PhoneGap framework primarily consists of a JavaScript Library that allows web languages such as HTML, CSS and JavaScript to communicate and access the native device features. As shown in the Figure 1 the top layer of the architecture represents the application source code and other resources. Furthermore, the middle layer is responsible for the interfacing between JavaScript APIs and the native APIs that are used by the operating system. This layer has the most important role of maintaining the relation between JavaScript APIs and native APIs, allowing developers to take advantage of device functionalities such as Camera, Accelerometer, Bluetooth, Calendar, Compass, Network Connection, Contacts, File, GPS, Menu, NFC, and Barcode [6].

Common PhoneGap applications follow the basic architecture. The application behaves as a client for the user to interact with, whereas their client part communicates with an application server for receiving the data. Next, the application server handles business logic which drives the UI and its functionality, and communicates with a back-end data repository [8].

In addition, the business logic part of the PhoneGap architecture generally uses a single HTML page as a single-page application model. The HTML page receives the necessary data from the application server using AJAX or JSON technology, that is dynamically displayed by updating the HTML Document Object Model (DOM). As the variables are kept in the memory within JavaScript, the single page is never unloaded from memory. However, PhoneGap supports multi-page client-side architecture as well, but it is less recommended because of the loss of variable in memory when loading a different page [8].

While PhoneGap architecture tries to keep a consistent JavaScript interface on the client side, the low level layer may slightly differ between platforms.

3.3 Development tools

For developing cross-platform applications PhoneGap does not provide an IDE or other development tools. Therefore, developers have to set up their environment for each platform accordingly. For setting up the environment on Android, the Android SDK with Eclipse IDE is recommended. For iOS the Xcode IDE with iOS SDK, and for Windows Phone the Visual Studio for Windows Phone, IDE is the ultimate choice as this tool contains the Windows Phone SDK as a bundle.

Mobile OS	PC OS	Software/IDEs	Language
iOS	Mac	Xcode	Objective C
Android	Windows/ Mac/Linux	Eclipse	Java
Windows Phone	Windows mainly	Visual Studio	.NET framework languages

Fig. 2. Development tools and environments for various mobile platforms.

3.3.1 *Eclipse*

Eclipse is a multi-language integrated development environment (IDE) and an extensible plug-in system that helps customize the environment. It is an open source development environment under terms of Eclipse Public License [11]. The Eclipse IDE is mainly used for writing Java programs, but it can be used for writing applications in any other programming languages, e.g. Ada, C, Erlang and C++. The Eclipse software development kit (SDK) is meant for Java developers, whereas for Android, the Android Development Tools (ADT) plug-in is needed (Figure 2).

3.3.2 *Xcode*

Xcode is Apple's integrated development environment for developing software for iOS and OS X platform. The IDE comes with a built in compiler, tester and debugger that supports programming languages like Objective C, C, C++, Java and many more. The IDE requires a registered Apple ID user for its download. On the other hand, for deploying your application to the market or directly to the phone, a developer license and a registered iPhone is necessary. The inbuilt simulator is available with the software purchase [12].

3.3.3 *Visual Studio 2012*

Microsoft integrated development environment Visual Studio is mainly used for developing software and application for Windows. Among them are web applications, web sites, web services and graphical user interface applications. The IDE's built-in language support includes languages like C#, C, C++, HTML/XML; languages supported by .NET framework [13]. Windows Phone SDK comes with Visual studio 2012 with a 30 days trial version and comes with its own emulator that allows for an easier development and testing without using Windows phone [14].

3.3.4 *jQuery*

jQuery is a lightweight JavaScript library created to ease developers work while scripting the client side of HTML. jQuery is an open source software, licensed under MIT License. The modular approach of jQuery is designed for creating interaction and manipulation of HTML and CSS elements [15].

3.3.5 *jQuery Mobile*

jQuery Mobile is a JavaScript web framework, created and optimized specifically for smartphones and tablets. The framework allows developers doing minimal scripting while giving the application a unified look across different mobile operating systems. This is due to the built-in themes and icons which is compatible with all major platforms and desktop web browsers. jQuery Mobile is compatible with other mobile application frameworks such as PhoneGap [16, 19].

3.4 Security in PhoneGap

As with any other mobile applications such as native applications developed in Java or Objective-C, it is also possible to reverse engineer PhoneGap applications. There is no restriction for a dedicated or malicious user to open the PhoneGap's binary application and extract the JavaScript source code, as well as modifying the code and resubmitting it back to app stores for the application phishing purposes [18]. However, there is one security workaround and that is the JavaScript code which can be downloaded during the runtime and get utilized but be removed upon the closing of the application. This ensures the non-existence of the source code when the device is at rest. On the other hand, this security measure has limited usage on applications written in Java or Objective-C [18].

3.4.1 Application security and cross-site scripting

Other bigger issues for mobile development are related to device and network security. As PhoneGap applications are regarded similar to web applications, Open Web Application Security Project (OWASP) guides about authentication and session management and cross-site scripting (XSS) prevention mechanisms should still be valid. To avoid XSS, measures such as cleaning data on both server-side before sending it to the client and also cleaning it on the receiver side should be taken [18]. As a basic rule, when posting data, method Post is recommended instead of method Get. In addition, PhoneGap has domain whitelisting for restricting and allowing accesses to external domains through config.xml file. However, "there is no built-in prevention mechanism against JavaScript injection" in PhoneGap [18].

4 EXISTING LITERATURE

There are number of literature that exist in the area of mobile and cross-platform development. Wasserman's perspective is purely on the software engineering issues regarding mobile applications [23]. Corral et al. noted on a trend of using web technologies in creating mobile applications, and emphasized the evolution of mobile web development to a single cross-platform development effort [24]. Charland and LeRoux discuss the advantages and the disadvantages of the mobile web vs. native applications, and furthermore, introduce PhoeGap as the next potential solution to the existing gap between the web and the native applications [1]. Huy and VanThanh touch on both theoretical and practical sides of mobile application development by defining four mobile application paradigms such as native applications, mobile widgets, mobile web applications, and HTML5 mobile applications, as well as evaluating the paradigms through building prototypes [25]. Moreover, Ohrt and Turau and Palmieri and others categorize and compare the current mobile cross-platform tools [2, 6]. There is not much research available specifically on PhoneGap framework, as it is still a fresh topic. Nonetheless, authors like Ghatol and Patel and Munro discuss PhoneGap's development [7, 21].

5 PHONEGAP PROTOTYPE

This chapter illustrates the steps to constructing the prototype application, starting from the requirements, design of the user interface and the implementation of various features and programming modules (Figure 3).

5.1 Stakeholders requirements

Functional requirements

- The prototype application is supposed to inform the company A customers about electricity interruptions
- It should have necessary contact information
- It should inform the user about the current price of the electricity
- It should have one button instant calling to the customer's service
- It should have the accessibility to the customer's page / Login page
- Extra feature such as a map, however, not a requirement

Non-functional requirements

Cross-platform support

- It should support mobile platforms such as Android and iOS and their various versions

Usability and User Interaction

- Intuitive interface
- Achieve unified interface across different platforms such as Android and iOS
- Support different screen sizes (tablets, phones)

Performance

- Page load should take no more than 1 second
- Similar user experience in terms of performance on different platforms
- The application should use native resources such as local storage, notifications, and networking

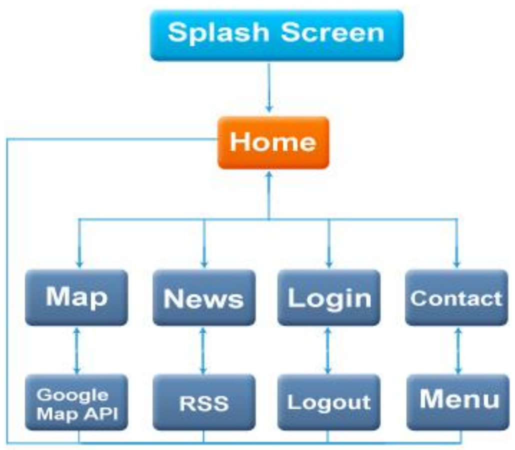


Fig. 3. Application navigation flow

5.2 Description of the prototype

The company A's IT department is presently informing their customers about current electricity interruptions on their blog. The current approach to the information access is not the ideal method for customer interaction. It is not ultimately convenient for the customers to having have to go to the company website and navigate through pages to read about electricity interruptions on a daily basis. Instead, by having an instant access to the same information through a mobile application, users will have the opportunity to be informed and updated faster and better. Appendix A1 for more information on stakeholder requirements.

5.2.1 Home

The requirements for the home screen page specify that a brief description for the various pages and functionality should be included on the home page. Moreover, four shortcuts have been created on the home screen to increase the navigation efficiency. Thus, the aim was to have a familiar feel and look while the user can easily learn and adapt as navigating through the prototype application pages. Navigation to other pages is done through a persistent menu at the bottom of the application. This menu is created with the help of jQuery Mobile's Ajax navigation system.

5.2.2 Map

Within the map page the user is able to select from the dropdown list the area he/she lives in. After selection, the data is saved in the local storage and is used later to be compared with coming new location information. By comparing the local data with incoming news feed or location information, the system decides if the user should get or should not get pushdown notifications in the future.

5.2.3 News Feed

Another essential page for the prototype application is the news feed page (Avbrott) which gets updated at Avbrott page's initial load. The news page presents relevant information from the organization blog and in the background acts as a web feed reader. The organization's blog information exists in a XML file format. The XML file is saved into a local storage and eventually get parsed into items that are presented dynamically in the DOM with the help of jQuery's collapsible menu.

5.2.4 Login

The prototype's Login page communicates with the existing backend. The objective of this prototype was to develop a stand-alone client with its own local storage mechanism with views and controllers. The prototype does cross-site login through custom tailored login screen but it does not change data on the server.

5.2.5 Contact

Contact page, based on the requirements specification, aims to inform customers with necessary information such as addresses and customer service phone numbers. Plus, there is a page that displays the actual price list of the current electricity market. The price list page is updated in a similar way to the news page by reading and parsing the data from the company website and presenting it in tables.

5.3 Testing environment and debugging

Each development environment supported by PhoneGap has a mechanism for testing and debugging the PhoneGap application. In general, during development process the prototype application has been tested in two ways. The most common way of testing was done with the help of the integrated emulator into the IDEs. The second way of testing was done by running the prototype on the physical devices.

5.3.1 Testing on simulators

The main advantage of using simulators was that they could mimic specific device functionalities and features such as geo-location, network, screen sizes or other devices like tablets. On the other hand, the main drawback of the simulators were with their lack of feedback and behavior that one can get from physical device performance.

5.3.1.1 iOS simulator

For testing and debugging on the iOS simulator a Mac computer was required. The iOS simulator provides the ability to simulate five devices: iPhone, iPhone (Retina 3.5-inch), iPhone (Retina 4-inch), iPad, and iPad (Retina).

5.3.1.2 Android emulator

Android Virtual Device (AVD) helped to modulate the needed devices by defining the hardware and operating system versions to be emulated. The only issue regarding the Android emulator was that the performance and user experience could not be evaluated as same as a physical device. On the other hand, the iOS and Windows Phone emulators could closely resemble their actual devices in terms of performance and usability.

5.3.2 Testing on devices

Testing on devices has been conducted intensively on Android platform mainly because of the hardware variations it has. The wide variety of screen sizes and form factors determines the choice of design approach that will scale up or down from device to device without compromising the UI elements. On the contrary, for iOS devices testing has been conducted by the use of one device since the hardware does not offer as much variety as Android devices. Performing tests on real devices can be costly and time consuming mainly because of the many devices and platforms some operating system can support. However, testing on a real device has been the most efficient way of investigating the quality and limitation of the application.

Tested Mobile Devices

	OS	CPU	GPU	Display
Samsung Galaxy SIII	Android 4.1.2	Quad-Core 1.4GHz	Mali-400MP	720 x 1280
Samsung Galaxy SII	Android 2.3.3	Dual-core 1.2GHz	Broadcom	480 x 800
HTC Desire	Android 4.0	600 MHz Cortex-A5	Adreno 200	320 x 480
Motorola RAZR	Android 2.3.6	Dual-Core 1.2 GHz	SGX540	540 x 960
iPhone 4	iOS 6.1	1GHz Cortex-A8	SGX535	640 x 960

Fig. 4. Tested Mobile devices

The prototype application on physical devices has been tested by generating and uploading application binaries through the IDEs or using PhoneGap Build's cloud services. However, there was an unexpected requirement encounter while building the application package through the PhoneGap Build. Unlike other mobile platforms, iOS platform requires a registered developer's certificate to be purchased. As it costs considerable amount and was not considered in the budget, testing on the actual devices running iOS was postponed for some time. Figure 4 presents the list of devices used during the testing of the prototype app.

The Windows Phone emulator provided comparable performance to an actual device. Since a physical device with the Windows Phone was not available, the prototype was tested only on the Windows Phone 7 emulator.

6 RESULT

This chapter sums the result gained during the *prototyping* and the *literature reviewing* part. It also includes the *evaluation* phase of this research.

6.1 Electricity interruption

The electricity interruption was the first feature the prototype aimed to have. It was implemented in the News page. The Map page was an extra feature which aimed to inform the user about electricity interruption by visualizing the affected area on the map. Currently, this extra feature has not been achieved because of timeframe and lack of additional information provided by the company.

6.2 Necessary contact information and the current price of electricity.

These functionalities are implemented under the Contact page and they are discussed in detail in Section 4 PhoneGap prototype part.

6.3 Button instant call

Invoking call to customer service on a phone device was implemented using jQuery Mobile.

6.4 Accessibility to customers page

This functionality is half-implemented and the customer can login and check his or her electricity consumption along with other information. It cannot download or make changes on the server currently.

6.5 Cross-platform support

The multiple mobile platform support is the biggest selling point of the PhoneGap framework. The prototype does run on all versions of Android and iPhone with little or no modification. Once configuring the PhoneGap libraries for each different platform correctly, there were no problems or discrepancies observed between these two platforms. In contrast, on Windows Phone 7 it did not run well. One of the reasons for failure was incompatibility of jQuery Mobile's libraries with Windows Phone platform, especially with Ajax Navigation system which the prototype uses. When the pages were linked with traditional 'href' linking the navigation worked fine, but other programmatic features were failing. However, due to the time frame, testing on Windows Phone platform was limited.

6.6 Usability and User interaction

Usability and user interaction requirement is presented in the following subsections such as intuitive-interface, consistent user-experience, designing for multiple screens, and screen orientation.

6.6.1 Intuitive user-interface

During prototyping one of the most important parts, besides the functionality, was the user experience. The upper layer of the software which is responsible for user interface was developed using HTML/CSS, and JavaScript was used for displaying content, navigation and functionality. Using PhoneGap as a cross-platform solution requires extra tools such as jQuery Mobile framework for creating the interface layer. Thus, the aim was to investigate its possibilities, and eventually enhance the view layer, while keeping a consistent user experience over various operating systems. The native Android application uses an XML based layout file to define the user interface, while iOS has the Interface Builder for user interface creation [21]. With jQuery Mobile framework the user interface is a unified CSS and JavaScript library that supports the majority of mobile device platforms [7]. The HTML pages were populated with 'div' elements that contain 'data-role' attributes. These attributes, in turn, call the library for presenting the UI widgets such as buttons, form elements, dialogs and much more.

6.6.2 Consistent User Experience

As mentioned in the requirements, the look across different mobile operating systems should be unified (See Appendix A1). For example, the end-user may own an Android phone and an iPad tablet. Therefore the application has to provide a consistent user experience across mobile platforms. For the application to keep a unified look across different mobile operating systems and platforms, jQuery Mobile framework was recommended by literature and the framework was later proven to be a competent solution [16]. Upon migrating from Android platform to iOS platform, there were no observable problems from the user interface and interaction, as long as developers followed the documentation correctly and made required configurations. The configurations are specific for each platform's native SDK [10]. When running the binaries on Android and iOS platforms also no problems encountered, but Windows Phone OS proved to be challenging. On Windows Phone, user interface was preserved after some modification on page transition functionality. See Appendix A2 for examples of the prototype look on different platforms.

6.6.3 Designing for multiple screens

Maintaining density and aspect ratio on devices that run Android and Windows Phone is a challenge with PhoneGap development, for the majority of devices have different screen sizes and densities. As a result, the user interface will be affected. In contrast, the iPhone has only two screen sizes, the new model of iPhone which has an increased screen height of 4.0' compared with old models of 3.5', iPods and older versions of iPhone.

jQuery Mobile framework provides, in general, an optimized UI for different screen sizes and densities [16]. However, specific images, icons or maps have to be configured through optimization of tags and CSS. This approach provided a robust compatibility between most of the tested renderings. The most accurate rendering with PhoneGap was achieved on Android OS and iOS.

6.6.4 Screen Orientation

PhoneGap with version up to 2.5.0 does not handle screen orientation change, since the majority of browsers already have this functionality [8]. Testing the orientation feature of prototype on actual mobile devices caused the prototype to restart on orientation changes. Small modification made in Manifest.xml file improved the functionality.

6.7 Performance

The jQuery Mobile does not limit the number of pages within particular HTML page file. Moreover, the HTML page can stand as separate HTML pages [16]. From the maintenance viewpoint the prototype took an organized approach and separated HTML pages. Furthermore, jQuery Mobile includes AJAX navigation system to support animated transitions between pages. The AJAX navigation changes pages dynamically by calling JavaScript "$.mobile.changePage" function and by setting a valid transition attribute instead of changing HTML pages through static linking. In the prototype, the effect for page transition has been set to "none" to increase the performance. In the jQuery Mobile's CSS library the default animation duration value is 350 ms [16]. However, the prototype was performing slower than the default duration. The speed of various functionalities such as page transitioning, loading and displaying the news feed, and rendering of maps were consistent and have had similar performances across Android and iOS platforms.

6.8 Evaluation

The evaluation of this prototype was conducted by observing and doing informal interviews and distributing questionnaires. The aim of evaluation was in finding out how the change was achieved, in other words, whether the prototype using PhoneGap was accepted as a solution for a business type of application by fulfilling the stakeholder's requirements.

The evaluation was conducted in two places, the university and the company A. At the university, students or evaluators were asked to run the prototype application on the Samsung Galaxy S3 with Android v4.1.2 and iPhone 4 with iOS v6.1 mobile phones. Participant's interaction and behavior was closely observed during the evaluation. The participants were encouraged to try every feature of the prototype for 10 minutes and afterwards were asked to give feedback on the prototype's features, usability, performance, and the general acceptability. Furthermore, the evaluation at the company A was done in two phases. Initial trial evaluation took place shortly after early version of the mock prototype, from there, further elaboration on stakeholder's requirements were given. During the second evaluation phase, stakeholders gave positive reviews on the implemented features and on the general usability, and were pleased with the overall performance of the prototype.

In this evaluation, in total of 10 people participated and the participants were equally divided by gender in order to bring in more balanced results. All participants were comfortable with mobile devices and technologies that they were provided with. Six questions were given and were designed to answer in a close and open ended manner.

The main interest in the evaluation was to find out the general acceptability of the prototype application by asking participants to evaluate the strengths and weaknesses of the prototype. Figure 5 sums the result of evaluation in terms of stakeholder's non-functional requirements such as Usability and User Interaction, Cross-platform support and Performance as the user acceptance criteria. The results were obtained by counting the participant votes, based on feedbacks received from both the University and the Company A. The Performance aspect was drawn from the question 2 of the questionnaire (See Appendix A3). Likewise, the Usability and User Interaction and the Cross-platform aspects were drawn from the questions number 1 and 3. This is described in Fig.5 and only covers the 'countable' part or the close-ended questions. Later under the Discussion section the 'un-countable' part or the open-ended questions will be discussed separately as it would be erroneous to convert subjective feedback into numbers.

Users	Cross-platform	Usability and User Interaction	Performance
Company A	5 of 5	4 of 5	5 of 5
University	5 of 5	4 of 5	5 of 5

Fig. 5. Evaluation results

The questionnaire's close-ended questions such as YES and NO answers were counted, as YES being a positive answer adding 1 point and NO being a negative answer subtracting 1 point from the total number of the participants' responses.

7 DISCUSSION

The literature on PhoneGap often mentions that applications which use this framework perform slower than those developed by native approach [4, 6, 20]. In addition, there were not many available researches done directly related to the PhoneGap framework that could also reveal the popularity of this framework regardless of its low performance. Therefore, it was a plausible choice to take the qualitative approach in the design research methodology to reveal the deeper truth of the problem by focusing on the building of the artifact. By building a real artifact using the PhoneGap with web development technologies and taking feedback from the users, gave us the better experience and insight into the actual problem.

Moreover, the PhoneGap framework was chosen because of its 'popularity' gained in recent years (though there was no clear evidence in the beginning of the research) and its long term support in the future, as the framework has been acquired lately by Adobe. The framework was also recommended by the industrial best practices' course instructor during the class discussions. All in all, it was a good choice considering the experience obtained during the research. The framework was not complicated to install and configure and it was straightforward to use as long as the user followed the guidelines and documentations correctly regarding different operating systems. Plus, it is lenient with the choice of integrated development environments, which eliminates the overhead of learning new development environments and tools.

For the preparation and conducting of this research it was necessary to learn a few web development technologies as an addition to the existing limited knowledge of web development. In fact, the first assumption regarding the development of the cross-platform prototype was to have the very basic web development knowledge such as HTML and CSS. Along the way, there was a need to learn some UI framework such as jQuery that could support multiple platforms and JavaScript language. The jQuery and jQuery Mobile proved to be consistent and powerful frameworks for the research purpose.

Based on the results the following views were drawn. The speed or performance was not that important as in the case of entertainment or performance demanding applications. Comparing native applications with the PhoneGap prototype application in terms of performance confirmed the findings from literature [1, 4, 6]. The prototype did run a bit sluggish in performance compared to native applications. The prototype users, however, were not complaining about the performance such as speed of transition between pages and general flow of the prototype's performance. They were more curious about various features and functionalities in the prototype. For them it looked and felt like an ordinary mobile application. It also falls with the requirement of the stakeholder that the prototype's usability and effectiveness are the first priority regardless of the speed. Nevertheless, there were few things that confused the participants such as the Map feature, Contacts feature, or the shortcut functionalities on the Home page. For example, one participant was wondering about the goal of the Map feature. This 'failure' was related to the incompleteness of the implementation and the lack of information provided to the user. Another participant at company A noted about the Contacts page containing much more content than the page's title. Despite giving positive "yes" answers to close-ended questions, such as whether the prototype was responsive and performing, some participants, when later were asked an open-ended question such as "what is not appealing?" gave specific and in-depth responses such as: *"Missing the loading info when a button is clicked, had to* *tap twice to get the feedback of a page loading."* The 'developer' participant's first impression and understanding was attributed to the Performance aspect of the prototype. However, when it was explained to the participant user the lag reason was not due to the performance of the PhoneGap application per se, and that it was related with the design of the prototype, the participant gave his suggestion on further improvement as follows: "*Show next page then show loading image then wait to complete loading the page info."*

Another participant was curious about the 'redundant functionalities' at Home page. These functionalities were shortcut buttons, meant to connect to nested menus that were not accessible from the general navigation. The shortcuts had not been implemented as intended at the time of evaluation. In addition, most of the participants were looking for a 'back' button after pressing a list menu and finding themselves on a nested page. The intention was to train users to utilize the general navigation buttons for going back to the original page, for the 'redundancy' of the 'back' button. Participant users found this new workaround awkward and unfriendly. Making even small changes to already accustomed user-interaction idioms (like changing the functionality of 'back' button) heavily influences the general usability and acceptability of a product as opposed to considerable tolerance on inferior performance of the application comparing to native applications.

In general, participants found the user-interface attractive and the prototype simple to use. The literature also recommends the data over decoration in mobile applications [1]. The performance of the prototype was acceptable enough for the current business type application. This performance fact not only related to the responsiveness and general performance of the prototype but it also considers the cross-platform aspect of it. The speed difference of network connections was another important factor that influenced the prototype's performance. For applications that often request data from the web, the network speed is important as it influences the general performance of those applications. Although, this factor is not that obvious for users, it is very relevant for PhoneGap developed applications for its inherent inferior performance.

Finally, the research proved that considerable resources and efforts can be saved both in terms of human and material means, by reusing existing web development knowledge and skills without learning platform-specific languages to enable the cross-platform support.

8 Conclusion

The authors have developed an electricity interruption prototype that can run on a cross-platform for the validation of acceptability of the PhoneGap framework within the specific business application. As a result of this research, the following factors that influence the popularity of the PhoneGap have been revealed. These include cheap development both in terms of developer's effort and material resources, acceptable performance for a light business type application, and cross-platform support. Other attractive factors for users are the use of native resources such as push-down notifications, geo-location, and local storage, consistent user-interface across platforms (with little or no modification), good user-experience when finding balance between decoration and data, and fast network connection with small payload that decreased the latency and resulted in the increase in performance. In addition, for the organization or the developers the maintenance cost is considerably low due to one code-base.

Thus, despite the PhoneGap application's low performance compared to the native applications and the framework's current level of maturity, the PhoneGap framework is a viable solution for the business applications that have moderate to average resource consumption.

8.1 Future Work

There are several possible improvements for this research. First, to do an extensive user experience testing by releasing a production type application in the application stores and doing a survey on end-users. Second, to extend the support for other devices and platforms that are not covered in this work. Third, in context of design and performance, to try other UI frameworks with PhoneGap than jQuery mobile, such as Sencha Touch or Kendo UI.

Acknowledgments

The authors would like to thank supervisor Morgan for his advice and guidance in this work. We also would like to extend our gratitude to the staff of the company A for providing the opportunity and motivation to do the practical side of this research.

References

[1] Charland, A. and B. Leroux (2011). "Mobile application development: web vs. native." Commun. ACM 54(5): 49-53.

[2] Ohrt, J. and V. Turau (2012). "Cross-Platform Development Tools for Smartphone Applications." Computer 45(9): 72-79.

[3] Wieringa, R. and A. Moralı. (2012). "Technical action research as a validation method in information systems design science" In *Proceedings of the 7th international conference on Design Science Research in Information Systems: advances in theory and practice* (DESRIST'12), Ken Peffers, Marcus Rothenberger, and Bill Kuechler (Eds.). Springer-Verlag, Berlin, Heidelberg, 220-238.

[4] Raj, R. and S. B. Tolety (2012). A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. India Conference (INDICON), 2012 Annual IEEE.

[5] PhoneGap. Available at: http://phonegap.com/about/. [Accessed May 16, 2013]

[6] Palmieri, M., I. Singh and A. Cicchetti (2012). "Comparison of cross-platform mobile development tools" Intelligence in Next Generation Networks (ICIN), 2012 16th International Conference on.

[7] Ghatol, R. and Y. Patel. (2012). "Beginning Phonegap: Mobile Web Framework for JavaScript and HTML5" Apress.

[8] PhoneGap Documentation. Available at: http://docs.phonegap.com/en/2.7.0/index.html. [Accessed May 16, 2013]

[9] About Apache Cordova. Available at: http://cordova.apache.org/index.html#about. [Accessed May 16, 2013]

[10] PhoneGap. Platform Settings. Available at: https://github.com/phonegap/phonegap/wiki/Platform-Settings. [Accessed May 16, 2013]

[11] Legal Resources. Available at: http://www.eclipse.org/legal/. [Accessed May 16, 2013]

[12] Xcode 4. Available at: https://developer.apple.com/xcode/. [Accessed May 16, 2013]

[13] Visual Studio. Available at: http://www.microsoft.com/visualstudio/eng/downloads. [Accessed May 16, 2013]

[14] Windows Phone Developer Center. Available at: http://dev.windowsphone.com/en-us. [Accessed May 16, 2013]

[15] jQuery Introduction. Available at: http://www.w3schools.com/jquery/jquery_intro.asp. [Accessed May 16, 2013]

[16] jQuery Mobile Docs. Available at: http://jquerymobile.com/demos/1.2.0/docs/about/intro.html. [Accessed May 16, 2013]

[17] Hevner, A., S. March, J. Park, and S. Ram (2004). "Design science in information system research" MIS Quarterly 28(1): 75–105.

[18] Platform Security. Available at: https://github.com/phonegap/phonegap/wiki/Platform-Security. [Accessed May 16, 2013]

[19] Introduction – jQuery Mobile. Available at: http://view.jquerymobile.com/1.3.1/dist/demos/intro/. [Accessed May 16, 2013]

[20] Trice, A. Performance & UX Considerations For Successful PhoneGap Apps. Available at: http://www.tricedesigns.com/2013/03/11/performance-ux-considerations-for-successful-phonegap-apps/. [Accessed May 16, 2013]

[21] Munro, J. (2012). "20 recipes for programming PhoneGap: Cross-platform mobile development for Android and iPhone." Sebastopol, CA, O'Reilly Media.

[22] PhoneGap Stats and Metrics. Available at: http://www.slideshare.net/AndreCharland/phone-gap-stats-growth. [Accessed May 24, 2013]

[23] Wasserman, A. I. (2010). "Software engineering issues for mobile application development" Proceedings of the FSE/SDP workshop on Future of software engineering research. Santa Fe, New Mexico, USA, ACM: 397-400.

[24] Corral, L., A. Sillitti, G. Succi, A. Garibbo and P. Ramella (2011). "Evolution of Mobile Software Development from Platform-Specific to Web-Based Multiplatform Paradigm" Proceedings of the 10th SIGPLAN symposium on New ideas, new paradigms, and reflections on programming and software. Portland, Oregon, USA, ACM: 181-183.

[25] Huy, N. P. and D. vanThanh (2012). "Evaluation of mobile app paradigms" Proceedings of the 10th International Conference on Advances in Mobile Computing &#38; Multimedia. Bali, Indonesia, ACM: 25-30.

Appendix A1

Requirement Specification

General:

- The prototype application is supposed to inform the company A customers about electricity interruptions
- It should have necessary contact information
- It should inform the user about the current price of the electricity
- It should have one button instant calling to the customer's service
- It should have the accessibility to the customer's page / Login page
- Extra feature such as a map, however, not a requirement
- The prototype application should support cross-platform
- The prototype application should use native resources like local storage, notifications, geo-location, and networking
- Achieve unified interface across different platforms such as Android and iOS; support different screen sizes
- Similar user experience in terms of performance on different platforms

Detailed:

➢ Map:

- the company should provide necessary information about electricity shortage on its News Feed in terms of coded Area with status Flag (e.g., 'area 1, 2, 4' - 'not solved' or '3, 6, 7, 8' - 'solved'); "Item/Area nr - string/'avbrott'"
- customer should be able to select his/her area of residence for further popup notifications in case of electricity shortage
- customers selection should persist for next visit by using local database
- markers/balloons should show up on the map accordingly with the provided info from the News Feed and local database

o Implementation specification for Map feature:
1. Get alert/fix from the feed (xml)
2. Parse and save locally (check if already exist, validate data)
3. Read local DB and show on the map
4. Check if the new alert matches with user's current location
   a) If yes: notify the user;
   b) If no: put a marker on the given coordinate;

➢ News Feed:

- The prototype application should be able to parse the xml
- parsed information should be saved in local storage for offline usage
- update the page information each time the page is visited
- present the company blog information in organized labels

➢ Log in:

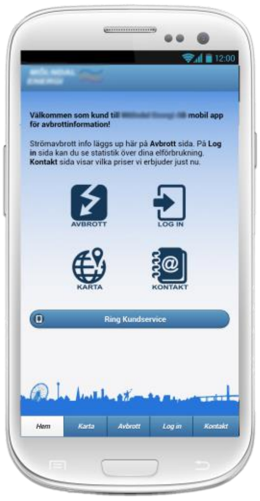
- user should be able to log in to the customer's page
- be able to access his or her information as a customer
- be able to download related information from the server (pdf files)

o Simple use case:
  - the application shows a login-form to a customer
  - customer enters the his/her credentials
  - the application connects to the server through web-app (no visible session-id)
  - the customer can access customer related information
  - customer can log-out

➢ Contact:

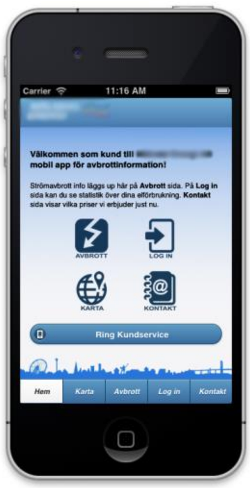
- show contact information
- show current price of electricity
- show description about application
- show questions and answers as a collapsible list

Appendix A2


Välkommen som kund till [redacted] mobil app för avbrottinformation!

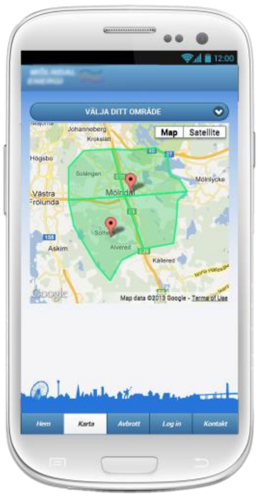
Strömavbrott info läggs up här på **Avbrott** sida. På **Log in** sida kan du se statistik över dina elförbrukning. **Kontakt** sida visar vilka priser vi erbjuder just nu.

1. Android Home page

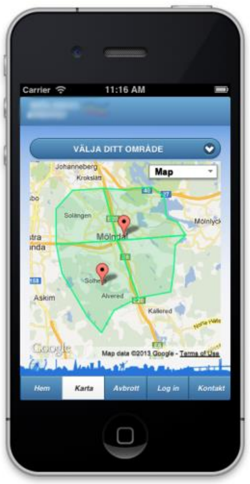

Välkommen som kund till [redacted] mobil app för avbrottinformation!

Strömavbrott info läggs up här på **Avbrott** sida. På **Log in** sida kan du se statistik över dina elförbrukning. **Kontakt** sida visar vilka priser vi erbjuder just nu.

2. iPhone Home page



3. Android Map page



4. iPhone Map page

Appendix A3

Example of the questionnaire

#	QUESTIONS	YES	NO
1	**Did the prototype help to achieve the requirement?**		
2	**Was the prototype application responsive and performing?**		
3	**Was the prototype application informative?**		
4	**What is not appealing?**		
5	**What did you like most from the prototype application?**		
6	**Further improvement?**		