Future Network and MobileSummit 2013 Conference Proceedings
Paul Cunningham and Miriam Cunningham (Eds)
IIMC International Information Management Corporation, 2013
ISBN: 978-1-905824-36-6

Poster Paper

A Case Study of Cross-Platform Web Application Capability

Sarah FAHLESSON, Dan JOHANSSON
Luleå University of Technology, Campus Skellefteå, Skellefteå, 93187, Sweden
Tel: +46 910 585300, Email:{firstname.surname}@ltu.se

Abstract: This study applies the existing knowledge on native and web applications to a specific case study. We compare the strengths and drawbacks of native applications versus web apps with emphasis on cross-platform functionality. We develop a prototype based on novel web technology (e.g. the emerging HTML5 standard and related frameworks) with the goal of recreating the functionality and graphical properties of an existing native application within the health care area. No changes are made to back-end infrastructure. The prototype is evaluated using cognitive walkthrough, and then we compare the differences of the main types of mobile applications to the results arrived at in this case study. Our beliefs are that the results from this particular case can inform design of cross-platform applications in general.

Keywords: Mobile e-Services, HTML5, Case Study, IT in Health and Care Areas

1. Introduction

Recent statistics from the ITU [1] show that there are almost six billion mobile-cellular subscriptions worldwide. An ever-increasing amount of new subscriptions concern smartphones, being able to run applications, or "apps". Add to this the advent of tablets and the fact that there are more than 1 billion mobile-broadband subscriptions worldwide [1], the mobile landscape is rapidly changing, with apps being one of the novelties paving the way. For companies looking to save time and money, mobile devices with the right applications installed have become excessively valuable.

HTML5 [2] is the much promised newest version of the Hyper Text Markup Language and while not yet officially launched, is already starting to be implemented by many browsers. HTML5 provides new features involving restructuring of documents, media capabilities, a new canvas element for rendering graphics, as well as new form elements to better standardize forms across browsers. Besides the official features of the W3C standard for HTML5, there are a number of related APIs that are often considered to be part of the official HTML5 standard. These include such features as Drag and Drop, more detailed media interaction, and the Geolocation API along with many others in development by various groups. [3]

The word "app" has become a common one in todays language and it generally refers to an application to be used on a mobile device such as a mobile phone or tablet. Most apps that people talk about are native applications, downloaded and installed from an official local such as Google Play or Apple's App Store. A native application typically has native compiled code and is usually faster than interpreted languages like JavaScript (that is commonly used in web applications). Native applications run on one operating system and require being rewritten in a different language when they are run on multiple platforms [3]. The main alternative to a native app though is the web app, requiring no installation or approval from an outside organization. The web app is

created using typical web programming skills and abilities and runs through the user's browser. Nothing needs to be installed, so a web application is able to theoretically run on any platform or device with a browser and internet connection [4]. The world of the web app though is not itself perfect, there are still numerous compatibility issues due to a lack of standardization [5]. A third route of hybrid applications, part native and part web, have been growing in popularity recently. It uses a web view in order to run JavaScript but can also access the mobile's features as though it were a native app. [3]

In this paper we compare the strengths and drawbacks of native applications versus web apps with emphasis on cross-platform functionality. We conduct a case study and develop a prototype based on novel web technology (e.g. the emerging HTML5 standard and related frameworks) with the goal of recreating the functionality and graphical properties of an existing native application. The purpose is to compare the differences of the main types of mobile applications to the results arrived at in this case study. Our beliefs are that the results from this particular case can inform design of cross-platform applications in general.

2. Case: Home Care Application

Our study was conducted in a real world setting, our case being a system developed by a leading Swedish provider of ICT services in the health and welfare sector. The end-user application is an Android app supporting mobile personal working with health and care services. The application is crucial for workers as they can quickly access information while out in the field.

2.1 System Structure

In this situation a company has created a native android application that they then sell as part of a package solution to various other companies and municipalities working in the Swedish sector of home care. This sector involves field workers going out to people's residences and providing support for the elderly and disabled (henceforth known as 'the clients') so that they can continue to live on their home instead of being put into care homes. The workers provide cleaning, food, social contact, connections with doctors, personal hygiene, and even help getting up in the morning. In order to accomplish and document their daily tasks, the workers have been supplied with mobile android phones by their employers along with an already existing native android application. This application connects with a number of other systems as seen in figure 1. The mobile device connects with a web service that then connects to a number of systems and databases in the solution that handle scheduling, documents, notes, and information about clients and the daily schedules for the field workers. ProCapita is a work system with built-in quality control while LapsCare adds back-end planning functionality. The InMovit then meld these systems, creating mobile back-end interfaces for web services and applications. The native application has functions for unlocking doors using Bluetooth, GPS location, writing and reading documentation, daily schedules for the worker whose phone it is, and a large amount of information about the client. The information can be doctor numbers, not to pet the dog (it bites), shower schedule, close contacts, hair appointments, and anything else a worker may need to know. As there are often multiple workers or even substitute workers unfamiliar with the particular client, it is vitally important that information be up to date and accurate.

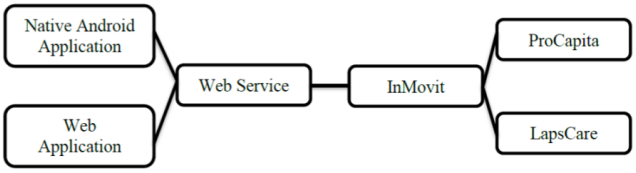

Figure 1: Information Flow.

2.2 Implementing the Web Application

The previously mentioned capabilities are currently present as part of a native Android application that is downloaded and installed on the mobile phones as needed. This system has drawbacks though in that it can only be run on mobiles running the android operating system and cannot be used on any of Apple's products such as the iPad or iPhone nor on Windows mobiles, etc. A solution to this was proposed with creating a web-based application instead of the native application.

Our web application is designed to copy the look and feel as well as functionality of the original Android specific app – i.e. the app should display log-in functionality, client browsing, individual scheduling and ability to access and read documentation about the clients. The GUI should be as close to the native app as possible. Also, our web app should support the opening of a native application through the web app, as there are other related systems (e.g. systems for opening and locking doors) that is still not available other than in native form. The application is written using HTML5, JSON (to create simple machine readable messages), and jQuery (to help and improve interaction with HTML elements and the server). There is also code designed to display information after it is received by the browser, e.g. navigation and GUI adaptation. Included are also a number of helper functions, e.g. handling date object conversion.

3. Evaluation and Results

The application was evaluated using simplified cognitive walkthrough [6], testing the application's functionality through pre-defined tasks such as logging in, choosing clients, editing schedules etc. Two iterative test/design rounds where carried out during the study. To capture the cross-platform properties of the applications, nine heterogeneous testbeds where used, combining different devices (e.g. Amazon Kindle, Apple iPad, HTC, LG and Samsung smartphones), different OSs (namely different versions of Android and iOs) and different browsers (e.g. Silk, Safari, Dophin Mini, Firefox mobile).

Concerning functionality, recreation of the log-in routine was simple, but as the native system demands unicode support for encryption, some browsers did not offer support. In terms of client browsing, individual scheduling and ability to access and read documentation about the clients, this functionality was fully copied and implemented, as was support for opening of a third party native app through the web app.

Working from screen shots taken in the native Android application we could produce near pixel accurate reproductions of the proportions of various parts of the screen. Colors where also very similar to those displayed in the native version of the app. Some fonts differ between the versions, and this is due to the company's explicit desire. Neglecting this, we could easily have recreated the look of the original fonts. Another

dissimilarity between the look of the different apps is the background image, replaced in our web app version to reduce the data traffic involved in downloading a picture instead of going for a solid colored background.

The web application allows any mobile device with internet and a browser that supports Unicode font encoding and HTML5's localStorage ability, to run the device. The type of operating system does not matter.

There is however still challenges to overcome when developing cross-platform applications. Browsers that could not support localStorage or other important HTML5 features did not deliver 100% functionality and/or look and feel. This problem is gradually decreasing though as more browsers add in HTML5 features, phones update their browsers, and mobiles become more capable of handling complicated websites. Our tests also showed that though functionality sometimes was not completely recreated, the web app could be opened and used on almost any platform. By not writing separate native applications a company avoids having to make potentially expensive hardware upgrades as well as requiring only one set of programmers instead of many programmers with speciality skills for each and every platform they would like to have an application work with. An advantage of having only one application, a web-based application, is that a company only needs to keep one application version updated.

It is our belief that emerging web standards like HTML5 and related frameworks can play an important roll in creating mobile systems and applications with cross-platform functionality. Future work will include user testing in a real use case, comparisons with other similar systems and projects and information of application design.

Acknowledgement

This work was supported by the NIMO project (Nordic Interaction and Mobility Research Platform, see nimoproject.org), funded by the EU Interreg IVA North program.

References

[1] ITU, "Ict data and statistics," jun 2012.

[2] W3C, "Html 5.1 nightly. a vocabulary and associated apis.," apr 2013.

[3] A. Charland and B. Leroux, "Mobile application development: web vs. native," *Commun. ACM*, vol. 54, pp. 49–53, May 2011.

[4] K. Buettner and A. Simmons, "Mobile web and native apps: How one team found a happy medium," in *Design, User Experience, and Usability. Theory, Methods, Tools and Practice* (A. Marcus, ed.), vol. 6769 of *Lecture Notes in Computer Science*, pp. 549–554, Springer Berlin / Heidelberg, 2011.

[5] M. Anttonen, A. Salminen, T. Mikkonen, and A. Taivalsaari, "Transforming the web into a real application platform: new technologies, emerging trends and missing pieces," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, (New York, NY, USA), pp. 800–807, ACM, 2011.

[6] C. Wharton, J. Rieman, C. Lewis, and P. Polson, "The cognitive walkthrough method: A practitioner's guide.," in *Usability Inspection Methods* (J. Nielsen and L. Mack, R, eds.), pp. 105–141, New York: John Wiley & Sons, 1994.

