Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 59 (2015) 291 – 297

Procedia Computer Science

International Conference on Computer Science and Computational Intelligence (ICCSCI 2015)

Review of Multi-Platform Mobile Application Development Using WebView: Learning Management System on Mobile Platform

Timothy Yudi Adinugroho [a], Reina [a], Josef Bernadi Gautama [a]*

[a]*Bina Nusantara University, School of Computer Science, K H. Syahdan No. 9 Kemanggisan-Palmersh , Jakarta Barat and 11480, Indonesia*

Abstract

The advancement of mobile technology and the internet network and their rapid adoption has enabled instant information access without relying on desktop or notebook computers. By using this technology, Learning Management System (LMS) can provide an unimpeded interaction for their users and promote awareness for any information updates. It is crucial to develop mobile application for each major mobile platform to reach most of the LMS user. In this paper, the mobile application development was done by combining native mobile technology and web technology using WebView API. This approach was taken to anticipate the need for creating and maintaining the application on multiple mobile platform and was expected to cut development time frame while retaining consistent interface and still enable platform specific's feature usage. The purpose of this study is to review the strengths and weaknesses of such combination in mobile application development specifically under android platform.

Peer-review under responsibility of organizing committee of the International Conference on Computer Science and Computational Intelligence (ICCSCI 2015)

Keywords: Mobile Application Development; Multi-platform; WebView; Learning Management System;

1. Introduction

Learning Management System (LMS) enables rich interaction between the lecturer and the student by providing useful features such as send/return assignment mechanism, peers discussion platform, immediate feedback on the online quizzes, timeless access to the learning materials, communication with peers and lecturer, collaborative group work, calendar as a reminder, news announcement, and performance dashboard [1]. Web-based learning system is popular in today's college because it allows some support to keep a large number of course resources during higher

* Corresponding author.
E-mail address: tadinugroho@binus.edu

Peer-review under responsibility of organizing committee of the International Conference on Computer Science and Computational Intelligence (ICCSCI 2015)
doi:10.1016/j.procs.2015.07.568

education process[2]. However, students wants to be able to use the new technologies such as mobile phones in conjunction to the LMS in education for the reason that they see and use the technological devices in their everyday life [3].

One of the main issues in mobile application development is that mobile platform is highly fragmented and it is expensive to support multiple platforms with multiple version and variants[4]. An intermediary language is needed to soften differences between such platforms so the developers will be able to focus on the business logic layer instead of worrying application porting between platforms[5]. One simple approach to address this issue is by using web application which benefits from good support over mobile browsers on all platforms [6]. Web technologies such as HTML, CSS, and JavaScript are highly suitable for developing multi-platform applications because they are standardized, popular, reasonably simple but powerful and well-supported. Combined with additional measures to utilize the special capabilities of mobile devices, they fulfill the requirements of most mobile scenarios[7]. WebView API enables developer to reap the benefit of web application in native mobile programming because not only allows applications to display web content, but it also enables applications to interact with the web content itself[8].

This paper structured as follows. Section 2 contains studied related works. Section 3 defines the hybrid mobile application approach used in this study as well as the main components that is vital for our hybrid approach. Section 4 discuss the strengths and weaknesses of the hybrid mobile application that we found over the android application development. And at last, we conclude our findings and thoughts in section 5.

2. Related Works

The available technological options for developing multi-platform mobile applications are web-based apps and hybrid apps. Hybrid apps, though also relying on standardized web technologies, are bundled within a native app "container", which serves as a bridge to access device hardware and functions like ringing, vibrating, notifications, making calls, using the camera, GPS sensor and accelerometer[10]. There are several hybrid multi-platform mobile application that have been developed such as PhoneGap, Titanium, and Xamarin[11]. These frameworks let a developer use one programming language to build a mobile application that support multiple different platforms at once. There is also some web framework such as jQueryMobile and Sencha Touch that enables a web developer to create a mobile website with a similar design as mobile platform's native interface[12].

In the context of user interface, each major mobile platform has their own design pattern. While Android prefer tabbed layout to have their tabs on the upper side of the layout[13], iOS prefer to have the tabs on the bottom side of the layout[14]. With this kind of difference, existing multi-platform framework could not decrease the effort of devising appropriate user interfaces for the different device types and orientations, however it still helped to create a codebase which is usable across multiple platforms[15]. Even if only a single platform is to be supported, a cross-platform approach may prove as the most efficient method due to its low barriers mainly owed to usage of standardized and popular Web technologies such as HTML, CSS, and JavaScript[6].

From the performance perspective, a hybrid mobile application using framework such as Phonegap had been known to be fast and giving smooth user interaction comparable with a native application. In comparison, website applications give slower performance due to the fact that a Web app has to be loaded via the Internet. This phenomenon can be limited by keeping an offline copy of the page. At runtime, Web apps profit from the fact that today's smartphone browsers are highly performance-optimized[7].

3. Methodology and Main Components

The mobile application development approach used in this study is a combination of native mobile programming (in this case Android) and mobile web programming without using any predefined hybrid mobile framework such as PhoneGap. We use the native mobile programming to create a container for rendering the web page which contain the main business logic using a component called WebView. The native mobile programming is also useful to access many native functionality such as push notification which is a platform-specific technology and to store required data on the client side.

The business logic for the application is offloaded from the native mobile application into the mobile web application which will be accessed from the native application's WebView. This approach will allow a one-time

development of the business logic over multitude platform with the same consistency and compatibility as each platform's default browser. Next, we will have a brief overview of the main components used in a mobile application development that combines web technology with native mobile programming.

For comparison purpose, we use Galaxy Nexus device using Android 4.2.1 and nexus 5 device using Android 5.1.1. As for the hybrid mobile application, we use Phonegap version 5.0.0-0.28.0 and JQuery Mobile version 1.4.2.

3.1. WebView

WebView is an application programming interface (API) which allow native application to process URL or HTML files and, as a result, render a web page within the application itself without switching into the default internet browser. WebView is a basic component for internet browser application and by using WebView, the native application will behave just like basic internet browser. WebView API is also capable in processing CSS and JavaScript which allow advance customization for functionality and interface of a web page. Usually a WebView is coupled with the platform's default web engine, therefore an update for the web engine will affect the WebView's capability and any application that utilize the WebView API.

3.2. Mobile Web

Mobile web is essentially an ordinary web application which is designed with a responsive user interface. A responsive user interface means that it can handle various small window resolution and resize the application design accordingly. Since mobile web is using the same technology as the ordinary website such as HTML, CSS, and JavaScript, it is well supported by popular mobile platform's default web engine and has good consistency over multitude platform. In our hybrid approach, the mobile web application will be opened within the native mobile application using WebView API.

4. Discussion

In this section, we will discuss the result of developing mobile application using a combination of native mobile programming and mobile web programming in the context of task management, content uniformity and flexibility, application performance, data consumption over network, and changes over platform's web engine.

4.1. Task Management

Nowadays mobiles platforms such as tablet and smartphone are using touchscreen and common touch gestures for a user to navigate and interacts with web application. To be able to cut development time frame, we decide to make use this interaction uniformity and implement it by creating one mobile web application to handle all application's business logic on every mobile platform. However, since there are platform specific function that must be implemented such as push technology, the native side of the application must be built independently for each major mobile platform. For basic functions such as navigating the online mobile web page and accessing native function to receive push notification and store the data, this approach works pretty well and able to cut the development timeframe more than a half compared to recreating every page on each platform using native mobile programming. With the combination of mobile web technology and native programming approach, the developer will need to maintain a single code-base for the business logic and each platform's basic business logic container. This is especially useful when multiple business unit exist on a single mobile application because each business unit's logic can be managed in their own code base. Hybrid framework such as PhoneGap will be able to reduce more effort because the same code can be compiled over different platform and the developer team will only need to maintain a single code-base.

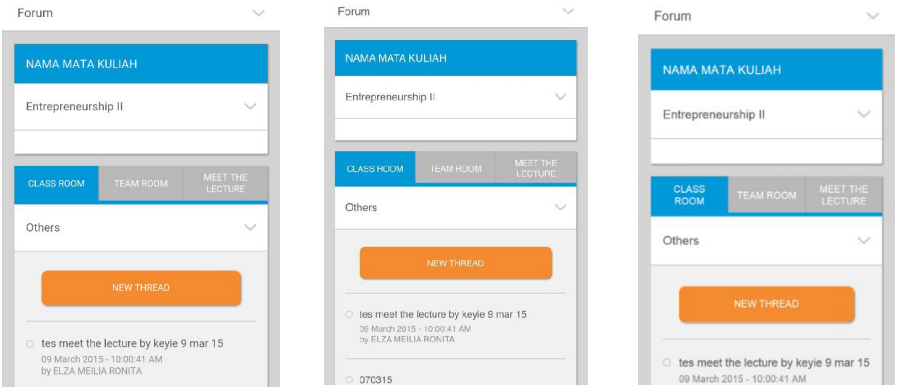

Figure 1 Rendered Mobile Web on Android, iOS, and Windows Phone WebView

4.2. Content Uniformity and Flexibility

In the mobile web part of the mobile LMS development, we use PHP as the server side scripting. For the user interface, we don't use any popular mobile interface framework such as JQuery mobile, in favor of pure performance. As expected, the mobile web interface is rendered correctly across multitude mobile platforms such as Android, iOS, and Windows Phone as shown in Figure 1 from left to right. Using this approach, we choose to implement native design pattern only on the native side for each platform while we use our own distinct design for the mobile web side. When developing mobile application using native programming, every platform has their own design pattern and it is advised to adhere by it. Mobile web and Hybrid framework such as PhoneGap also allow application to know on which platform it is viewed in or installed to. This is useful should the developer choose to follow each platform's design pattern and execute it at application runtime.

Our WebView application approach have a unique advantage in the context of content management flexibility compared to native and hybrid application (see Figure 2). Due to native and hybrid mobile application nature to use web service to fetch required data, these application are only able to change their data but not their layout or content interface. If any changes is needed for the layout or content's interface then the developer should update their application the platform's store which might take extra amount of time. This is not true for our WebView application which allow application developer to update new interface in the app's content from the server side without updating the native application in the platform's application store.

Figure 2 Difference of Application Content Management

4.3. Application Performance

Overall, there is no significant difference in the context of application's performance. The application's performance is measured using an external application called GameBench which measure the application's frame rate. The test result shows that native mobile applicatio are able to reach 59 frame per second in average while hybrid mobile application and WebView application are able to reach 57 frame per second in average. These performance result are considered as pretty smooth in case of user navigation. However, it is noted that the number of memory and computing power consumed by other process might disrupt the application performance. Also, because WebView application need to download the html, CSS and Javascript files to render the page, it will consume more data over the network and slow down the interpage navigation when compared to their native or hybrid counterparts.

There are several things that can be done to optimize WebView application performance such as using paging technique to keep the number of listed object small enough on one page and minifying the Javascript and CSS files to keep the web page size as small as possible.

	Native App	Hybrid App	WebView App
Android 4.2.1	59 fps	57 fps	57 fps
Android 5.1.1	59 fps	57 fps	57 fps

Table 1 Application Performance Measurement

4.4. Data Consumption Over Network

Our approach using combination of mobile web and native technology consumes more data over the network compared to a regular native mobile application or hybrid application. This is because regular native mobile application and the hybrid application has their own layout and interface installed on the platform, therefore they only need to pull the required data in a form of data structure while our approach need to retrieve the whole layout and data in the form of web page which consists of several files such as HTML, CSS, and JavaScript.

While the network traffic usage seems to be bulky in comparison of consuming web services, it is actually no different than a normal case of web browsing. However, if data consumption over the network become a concern, developers heavily utilizing WebView API for such combined mobile web and native approach should take additional steps to cache the web page thus minimizing the data consumption over the network.

4.5. Changes over Platform's Web Engine

The Mobile platform is updated continuously, the same can be said for the underlying web browser engine which provide HTML, CSS, and JavaScript processing into the platform's default internet browser and WebView API. While the continuous update improves the web browser engine's capability and is a necessity in the context of security, the update might also include an abrupt change that affect the WebView API. This change also affect PhoneGap application since it is also using the default web engine.

When building a save offline web page feature on our application, the required native API works nicely on android 4.0 – 4.3 version. However, android 4.4 introduces a new version of WebView which is based on Chromium. This brought format change for saved offline web page from XML to MHT file type and breaks the platform's capability to run offline JavaScript. It is noted that this change doesn't break the platform's default browser and WebView API capability in processing online web page.

While the platform's developer team tried their best in updating the platform without breaking the compatibility of the general use cases, niche features might be overlooked and developers who use the WebView API should always check their application's behavior over every update brought to the web browser engine.

5. Conclusion

Our approach in mobile application development which combines native mobile programming and mobile web by utilizing Web API has its own upside and downside. On the upside, mobile web technology is truly versatile and correctly rendered across multi-platform as long as it is used for online viewing and this approach still let the developer use any native functionality while at the same time utilizing the mobile web's versatility. This approach does not only let developer to get the best from the world of native mobile programming and the world of mobile web programming but also simplifying the code base management because all of business logic is gathered in a single code base. When multiple business unit exist in a single mobile application, each business unit can have their own managed business logic code base. WebView approach also let the business logic developer to flexibly update their content without having the need to update the mobile platform's store application. On the downside, the developer still needs to manage multiple code-base of the native side of the application which depends on how many mobile platform they target. For application performance, this research shows that WebView application and Hybrid application is smooth enough compared to the native mobile application, this result is in accordance with other research which states that while web technology stack has not achieved the level of performance that can be attained with native code, it is getting close [9]. Data consumption over a network might also become an issue especially with an unstable mobile carrier network. The mobile application will become more prone to error when consuming larger data over the network, and consuming data using web services is obviously more efficient rather than consuming the whole web page. Regarding other issues, currently niche features such as saving offline web page completely for later execution might not be directly achievable through platform's API on some version. While it is possible to find a roundabout way to overcome this issue, the time taken to solve such problem might cause a setback in the project's timeline. Although this issue might be solved externally by the platform's developer team, but there is no way to verify the exact date of the next update. In the end, depends on the application's requirement it might be sensible to trade over the small print network consumption with the development simplicity over multitude mobile platform by using combination of native mobile programming and WebView API.

References

1. N. Cavus and S. Kanbul, "Designation of Web 2.0 tools expected by the students on technology-based learning environment," *Procedia Social and Behavioral Sciences 2,* p. 5824–5829, 2010.
2. J. Penga, D. Jianga and X. Zhanga, "Design and implement a knowledge management system to support web-based learning in higher education," *Procedia Computer Science ,* vol. 22, p. 95 – 103 , 2013.
3. N. Cavus, "Investigating mobile devices and LMS integration in higher education: Student perspectives," *Procedia Computer Science 3,* p. 1469–1474, 2011.
4. A. L. Wasserman, "Software Engineering Issues for Mobile Application Development," *FoSER '10 Proceedings of the FSE/SDP workshop on Future of software engineering research,* pp. 397-400, 2010.
5. J. Perchat, M. Desertot and S. Lecomte, "Component Based Framework to Create Mobile Cross-platform Applications," *Procedia Computer Science 19,* p. 1004 – 1011, 2013.
6. H. Heitkötter, S. Hanschke and T. A. Majchrzak, "Comparing Cross-platform Development Approaches for Mobile Applications," in *Proceedings of the 8th International Conference on Web Information Systems and Technologies (WEBIST)*, Porto, Portugal, 2012.
7. H. Heitkötter, S. Hanschke and T. A. Majchrzak, "Evaluating Cross-Platform Development Approaches for Mobile Applications," *Web Information Systems and Technologies,* pp. 120-138, 2013.
8. T. Luo, H. Hao, W. Du, Y. Wang and H. Yin, "Attacks on WebView in the Android System," *27th Annual Computer Security Application Conference,* pp. 343-352, 2011.
9. A. Charland and B. Leroux, "Mobile application development: web vs. native," *Communications of the ACM,* vol. 54, no. 5, pp. 49-53, 2011.
10. A. Nitze and A. Schmietendorf, "Cross-Platform Mobile Application Development," in *User Conference for Software Quality, Test and Innovation*, 2013.

11. L. Corral, A. Sillitti and G. Succi, "Mobile Multiplatform Development: An Experiment for Performance Analysis," *The 9th International Conference on Mobile Web Information Systems,* vol. 10, p. 736–743, 2012.
12. Google, "Pure Android," Google, [Online]. Available: http://developer.android.com/design/patterns/pure-android.html. [Accessed 5 March 2015].
13. P. Miravet, F. Ortin, I. Marin and A. Rionda, "Using standards to build the DIMAG connected mobile applications framework," *Computer Standards & Interfaces,* vol. 36, no. 2, p. 354–367, 2014.
14. Apple, "TabBarControllers," 15 11 2014. [Online]. Available: https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewControllerCatalog/Chapters/TabBarControllers.html. [Accessed 9 March 2014].
15. C. d. A. Freire and M. Painho, "Development of a Mobile Mapping Solution for Spatial Data Collection Using Open-Source Technologies," *Procedia Technology,* vol. 16, p. 481–490, 2014.