Cross-Platform Mobile Development: A Study on Apps with Animations

Matteo Ciman, Ombretta Gaggi, Nicola Gonzo
Department of Mathematics
University of Padua
via Trieste 63, 35121 Padua, Italy
{mciman,gaggi}@math.unipd.it

ABSTRACT

The paper presents a comparison between different frameworks for cross-platform mobile development, MoSync, Titanium, jQuery Mobile and Phonegap, with particular attention to development of applications with animations. We define a set of criteria for the evaluation and we develop the same game as case study app, with the aim to provide an unbias judgement.

Our analysis recommends Titanium as the best framework to develop mobile applications with animations.

Categories and Subject Descriptors

D.2.3 [**SOFTWARE ENGINEERING**]: Coding Tools and Techniques

Keywords

mobile applications, cross-platform development

1. INTRODUCTION

A mobile application may require to be developed several times, one for each supported platform, thus dramatically increasing the required time and skills for developers, and finally, the cost of production. A solution is represented by framework for *cross-platform* development.

Some authors [1] [2] [3] try to highlight a set of criteria to be met by frameworks of high quality. Heitkötter et al. [1], compare jQuery Mobile, Sencha Touch, The-M-Project and Google Web Toolkit+mgwt according to a particular set of criteria, which includes license and costs, documentation and support, learning success, user interface elements, etc. Palmieri et al. [2], evaluate Rhodes, PhoneGap, dragonRAD and MoSync with particular attention to the programming environment and the APIs they provide. Raj and Tolety [3] analyze and classify a set of frameworks for cross-platform development in four approaches: web, hybrid, interpreted and cross compiled. They highlight strength and weakness of each approach, concluding that it does not exist a preferred solution for each kind of application, but the decision about which framework to use should be made considering the features of the application to be developed.

All the addressed works make a critical analysis of the chosen frameworks according to criteria which do not always require to develop any application. They study the frameworks, the supported operating systems and sensors, the provided APIs, etc, but they do not try to implement a case study application to understand the real performances of the frameworks. Sometimes they interview different developers, which have different experiences and skills. Therefore these evaluations may suffer of the bias due to the different situations in which each developer has used the discussed frameworks, i. e., a particular framework may be well suited for a specific application. Moreover, they usually consider *business applications* which do not include animations and transition effects (or they have very poor ones).

In this paper, we try to compare and evaluate performances of different cross-platform frameworks using the same case study, a *serious game* which aims to help children to learn letters. The fishes Nemo and Dory race along a path and the child has to move the Nemo fish to avoid obstacles. This game requires to implement a lot of animations to move the two fishes, Nemo between four positions, the obstacles and an octopus. Moreover, to realized the illusion of the horizontal movement of Nemo, the background is moved during all the play, changing its speed depending on the result of the choices of the player.

2. ANALYSIS GUIDELINES

An impartial comparison between different frameworks for cross-platform development is a challenging goal since evaluations are often biased by programmers' background skills, framework learning time, specific application, etc. For this reason, our evaluation assumes that developers, that were asked to realize the application, already know the programming language used by each framework and have to learn only how to use the framework tools (SDK, APIs, etc).

For each framework, we consider many factors, i. e., licenses and their costs, the variety and quality of APIs available, the presence of tutorials and the size of the community, the complexity of the code needed to implement the case study application, the usability of the IDE, the list of supported mobile devices, the support to create applications with a native user interface and the basic knowledge in term of programming skills and technologies required by each framework.

SAC'14 March 24-28, 2014, Gyeongju, Korea.

http://dx.doi.org/10.1145/2554850.2555104

A comparison between frameworks is reported in Table 1. All frameworks are graded with a number between 0 and 5.

3. MOSYNC

MoSync is a framework developed from MoSync Inc. since 2004. It supports several mobile platforms, and it is particularly interesting since it produces a real native application as final product.

The developer can develop a web-app, using HTML, CSS and JavaScript, or can use C++ or C. The possibility to choose the preferred solution drastically reduces the basic knowledge required to developers.

The IDE provided for MoSync allows to test the application with the integrated simulator in all the supported platforms, to easily build the final application file and to select which are the necessary requirements, i.e. sensors, of the application. Unfortunately, the IDE suffers of a long start-up time and lacks of a complete integration with other tools, e. g., a user interface designer.

The debugger is really poor for the C++ implementation, while it gives much more functionalities using the JavaScript and web-view development, like step-by-step analysis.

Although the total amount of supported APIs is really high, they are not developed for recent versions of the platforms: iOS is supported till version 4.3, Android till version 4, Blackberry is in beta till version 6 and Windows Phone till version 7.5. But, also for supported versions, not all the provided functionalities are available through the APIs, e. g., Bluetooth for iOS and Google Maps for Android. Moreover, the programming language chosen for the development affects the set of available APIs.

MoSync does not provide any API for animations. For this reason, it is necessary to build them manually, redrawing the objects every frame. It is really easy to include other APIs in the JavaScript implementation, while using C++ it is essentially impossible to add new libraries.

The documentation is complete only for the JavaScript implementation, while for C++ is extremely elementary and sometimes with big misses, i.e. the developer needs to look into the code of the library to understand the class that provides image scalability.

The creation of a "native like" application is very easy using several JavaScript functions that build at run-time the right GUI. It is even possible to use third party APIs and features that improve support to native interfaces. Two libraries NativeUI (native User Interface only for iOS and Android) and MAUI (single UI for every platform and device) allow the creation of native GUI in C and C++ applications.

MoSync supports nine mobile operating systems: Android, Blackberry, iOS, J2ME, Moblin, MeeGo, Symbian, Windows Phone and Windows Mobile. But, the supported versions are quite old and the support is not complete.

The test application developed with MoSync was written in C++, to reach the best performances. Compared with other frameworks, the MoSync implementation required much more lines of code and working hours, showing a high complexity of the framework. In particular, the most time consuming task is to develop animations that are not natively provided by the framework.

To conclude, MoSync has shown to be a really wide and powerful framework. The possibility to develop an application using different languages is clearly a positive aspect. Even the performances of the application are great. Unfortunately, the available APIs are not complete, have a low level of support and are sometimes not well documented.

4. TITANIUM

Titanium is a cross platform framework developed since 2006 by Appcelerator. It provides the possibility to develop mobile applications (in particular for iOS and Android) using JavaScript. Titanium produces a native application, so performances are very good, even if the deployed application contains essentially a bridge between JavaScript calls and native APIs of the target device.

Titanium requires to developers to know only JavaScript. All the APIs of this framework can be called simply using this language, that is a mandatory prerequisite. The Titanium Studio IDE provides several tools for developing and debugging, but requires several native tools, e.g., the Android DDMS to access to the log of the application, to fully support the developer.

An high number of APIs are available for this framework. They are even well documented, containing a description of every method, its input and output parameters and, sometimes, an usage example. They cover transactions, rotations and scaling, and the possibility to combine them together. They do not allow to know the exact position of a moving object. The use of a timer to change the position of the object decreases the fluidity of the application.

Considering the main platforms supported by Titanium, the same application can have completely different User Interface if deployed on an iOS or Android device.

The platforms supported by this framework are iOS (from 5.0[1]), Android (from 2.3.3[2]), Blackberry, Tizen and Web Mobile, but the APIs almost cover Android and iOS.

Compared to the MoSync framework, the application built with Titanium required much less time and lines of code. The total lines of code were 37% less, i. e., more than one quarter less of total amount of time. This clearly show a simplicity of the framework that decreases the amount of time necessary to develop an application.

From our experience, Titanium has shown several really good aspects, e. g., the high number of APIs and its documentation, that simplify the development an application. Therefore, Titanium must be considered when developing a cross platform application, even for the frequent updates of the platform and APIs.

The negative aspect of this framework relies on the support essentially limited on iOS and Android.

5. JQUERY & JQUERY MOBILE

jQuery is a JavaScript library that provides several facilities to developers, like DOM manipulation, event handling, animations and AJAX calls in a cross-browser and easy way. jQuery mobile is a framework, built on jQuery, that provides several graphics element designed to be used with mobile devices.

Unlike other frameworks, jQuery and jQuery mobile allow the developer to built web applications that will work either on desktop and mobile web browsers, but not native mobile applications. Therefore the user must access the application through the web browser.

[1]only 1.4% of the devices do not have at least this version.
[2]only less than 4.9% do not reach this minimum version.

This framework requires HTML, CSS and JavaScript as mandatory programming languages. No other languages are required, and, due to their enormous diffusion, the requirements are extremely low.

jQuery provides a wide set of different APIs for every purpose, e. g., adding or deleting a new element, handling click events, object's style manipulation, etc. It also provides a set of APIs for animations, e. g., fade in/out, animate, etc., but they work very well with desktop browsers, but suffer an high performances degradation in the mobile environment.

Moreover, it does not support specific features of mobile devices, e. g., Bluetooth, accelerometer, etc., since jQuery mobile support is limited to what is accessible through web browsers. This is a big issue, because most of the games relies on these features to improve the user experience.

A CSS template for Android and iOS is freely available to build a native User Interface.

jQuery supports the majority of the mobile devices, i. e., iOS, Android, Windows Phone[3], Blackberry etc., and even several mobile browsers, i. e., Firefox Mobile, Chrome and Opera Mobile. Even if all these mobile systems and browsers are supported, support for complex web applications is only theoretical due to performance degradation.

The development using jQuery and jQuery mobile has been fast, thanks to the big number of provided APIs and plugins freely available. Most time has been dedicated to cross-browser testing, because there are no automatic tools able to provide information about compatibility.

To conclude, jQuery and jQuery mobile provide the possibility to develop a web application either for desktop and mobile browsers. Several APIs exist but their performances decrease with mobile browsers. Access to sensors is missing.

6. PHONEGAP

Phonegap is a framework that allows to wrap a web application inside a webkit engine that will run on the selected device looking like a native application.

Phonegap requires HTML, CSS and JavaScript as mandatory programming languages. No other languages are required, and, due to their enormous diffusion and knowledge, Phonegap requirements are really low.

Phonegap defines several APIs that provide the possibility to access the most important sensors and tools of mobile devices. In particular, it provides access to the accelerometer, the camera, geolocalization, contacts, calendar etc. Phonegap does not provide any particular API for animations, that are the core of our case study game, but it can be extended with other frameworks, i.e. jQuery. The result that we got from our experience is that even in this case, as in the case of jQuery Mobile, the application is fluid and very usable if accessed through a laptop, while it encounters performance problems if used in a mobile device.

Phonegap does not provide any IDE to develop applications. The developer has to rely on IDE for native applications, i.e. XCode for iOS or Eclipse for Android. Moreover, it does not provide the possibility to have Native User Interface, therefore these two criteria are not applicable.

Phonegap supports the most common mobile operating system: Android, iOS, Blackberry and Windows Phone. The most complete APIs support is provided for Android and iOS.

[3]iOs from version 3., Android from version 2.1, Windows Phone from version 7.5-8.

	MoSync	Titanium	jQuery	Phonegap
Licenses	5	4	5	5
API	3	4.5	2	2
Community	3	5	5	5
Tutorials	4	5	4	5
Complexity	2	5	4	4
IDE	4	5	-	-
Devices	2	4	5	4
GUI	4.5	5	3	-
Knowledge	5	5	5	5

Table 1: Final remarks comparison between frameworks

The total amount of time necessary to develop an application with Phonegap is almost the same of a web-application using jQuery or jQuery mobile: Phonegap simply requires to change (or to add) very few JavaScript functions (in particular the ones related to smartphones sensors and features).

To conclude, Phonegap is a powerful framework that allows to wrap a web application into a native one. It provides access to the most common sensors and tools of the different smartphones. Actually, the most important issue regards the performances of the final application, that are far from the ones of a native application.

7. CONCLUSIONS AND FUTURE WORKS

This paper presents a comparison between different frameworks for cross-platform mobile development, with particular attention to development of application which requires animations. For this reason we use, as case study, a simple game.

Our evaluation considers MoSync, Titanium, jQuery Mobile and Phonegap. The results are reported in Table 1 which compares the evaluation gained by each framework against all the chosen criteria.

At the moment, the best framework for development of mobile applications with animations is Titanium, because it natively supports, with some limitations, animations and transition effects, and its performances are good and promising even for applications more complex than the case study.

Future works will be devoted to the analysis of other frameworks, e. g., we are currently studying Sencha Touch, which does not natively support animations but has great results in terms of performances.

8. REFERENCES

[1] H. Heitkötter, S. Hanschke, and T. A. Majchrzak. Comparing cross-platform development approaches for mobile applications. In *Proceedings of the 8th International Conference of Web Information Systems and Technologies*, WEBIST '12, pages 299–311, 2012.

[2] M. Palmieri, I. Singh, and A. Cicchetti. Comparison of cross-platform mobile development tools. In *16th International Conference on Intelligence in Next Generation Networks*, ICIN '12, pages 179–186, 2012.

[3] R. Raj and S. Tolety. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In *Annual IEEE India Conference*, INDICON '12, pages 625–629, 2012.