Development Approaches for Mobile Applications: Comparative Analysis of Features: Proceedings of the 2018 Computing Conference, Volume 2

Chapter · January 2019

DOI: 10.1007/978-3-030-01177-2_34

CITATIONS
0

READS
105

7 authors, including:

Lisandro Delia
Universidad Nacional de La Plata
13 PUBLICATIONS **23** CITATIONS
SEE PROFILE

Pablo Javier Thomas
Universidad Nacional de La Plata
19 PUBLICATIONS **28** CITATIONS
SEE PROFILE

Juan Fernandez Sosa
Universidad Nacional de La Plata
4 PUBLICATIONS **2** CITATIONS
SEE PROFILE

Germán Cáseres
Universidad Nacional de La Plata
8 PUBLICATIONS **3** CITATIONS
SEE PROFILE

Development Approaches for Mobile Applications: Comparative Analysis of Features

Lisandro Delia, Pablo Thomas, Leonardo Corbalan, Juan Fernandez Sosa, Alfonso Cuitiño, Germán Cáseres, Patricia Pesado

Instituto de Investigación en Informática LIDI (III-LIDI)*
Facultad de Informática – Universidad Nacional de La Plata.
La Plata, Buenos Aires, Argentina
* Centro Asociado Comisión de Investigaciones Científicas de la Pcia. de Bs. As. (CIC)
{ldelia, pthomas, corbalan, jfernandez, acuitino, gcaseres, ppesado}@lidi.info.unlp.edu.ar

***Abstract*—The rise and evolution of mobile devices has had a major impact on Software Engineering. Currently, it is possible to develop mobile applications for a specific platform, using the native approach, or to develop applications for different platforms simultaneously, using web, hybrid, interpreted or cross-compiled approaches. In this paper we identify and analyze different features that affect the choice of development approach to be used.**

Keywords—mobile devices, multi-platform mobile applications, native mobile applications, mobile application development approach

I. INTRODUCTION

Currently smartphones have become everyday and ubiquitous. These devices allow to execute tasks of different complexity, and in some cases critical. For this reason, continuous improvement in computing capacities, availability, efficient performance and other needs is required. The dizzying growth of this technology puts pressure on the adaptation of Software Engineering.

The development of mobile applications has a series of specific features for this activity, not found in traditional software development [1]. The platforms fragmentation, variety of programming languages, development tools, standards, protocols and network technologies, types of device and limited devices capacity, are some of the issues to consider.

It is very common to associate the success of a mobile application with the popularity it achieves. For this, the application should be available for multiple platforms, especially the two most currently used: Android and iOS [2]. To satisfy this purpose, there are two strategies:

i) Develop a specific application for each platform, using the platform-specific tools and languages. This involves carrying out more than one development project. These applications are known as native applications.

ii) Develop products able of operating in more than one operating system but with a unique base of source code, called multiplatform applications.

The Software Engineering community has shown great interest recently, for the development of multiplatform mobile applications, as can be appreciated in different articles.

In [3], a comparative analysis of development approaches for mobile device multi-platform application is presented, and the following taxonomy is proposed: mobile web applications, hybrid applications, interpreted applications and cross-compilation applications.

In [4], the authors of this paper have analyzed the advantages and disadvantages of the multi-platform development methods mentioned in [3], from the point of view of the Software Engineer.

The choice of development approach to be used is crucial and has a strong impact throughout the entire life cycle of the application. This choice can hardly be changed in advanced stages, since it would involve a complete reengineering of the software product. It is for this reason that a thorough analysis should be carried out beforehand to determine the most appropriate approach for a given project.

The criteria for choosing a development approach for mobile applications depend on several factors. One of these, oftentimes essential, is execution time. Recently, the authors of this paper have evaluated the performance of applications generated using the approaches defined in [3], for an application with high computational demand. These results are shown in [5].

Performance is one of the most important decision factors, but not the only one. While the general aspects of multi-platform development frameworks for mobile devices are discussed in [6], [7] and [8], it should be noted that no works have been found that evaluate and compare the key factors for all multiplatform development approaches, following the taxonomy proposed in [3].

This paper introduces the key characteristics to be considered when choosing the development approach to use, and analyzes them individually for 9 development technologies, which cover the 5 approaches mentioned in [3].

This paper is structured as follows. Section 2 gives an overview of existing mobile application development approaches; Section 3 describes the most important decision features for choosing the development approach; Section 4 presents a comparative analysis of features in terms of the 5 development approaches, using 9 different technologies; and subsequent sections summarize our conclusions and future work.

II. MOBILE APPLICATION DEVELOPMENT APPROACHES

In recent years, the mobile device market, especially that of smart phones, has seen a remarkable growth. In particular, the operating systems that have grown the most are Android and iOS [2].

Each of these operating systems has its own development infrastructure. The main challenge application providers face is offering solutions for all platforms in the market; however, achieving this goal usually involves high development costs that are often hard to afford [9].

The multiplatform development tries to reduce costs by employing a single project compatible with all operating systems.

The following is a summary of the development approaches used to create mobile applications:

A. *Native Applications*

Native applications are developed to be run on a specific platform, considering the type of device, the operating system and the version to be used.

The source code is compiled to obtain the executable code, similar to the process used for traditional desktop applications.

To distribute the application, the mobile app stores (App Stores) of each platform are used. These stores perform an audit that verifies if the minimum requirements of each platform are met. Then, the product will be ready to be downloaded and installed by the end users.

The native development allows full access to the hardware of the device, i.e. sensors, camera, microphone, among others. In addition, it is possible to execute them in offline mode and/or in background. Finally, native applications have good performance.

Likewise, this benefits user experience, since native components are used for user interfaces, giving them a look and feel that is similar to that of other operating system interfaces.

If covering different platforms is a requirement to be met, this approach will have high costs, since it is not possible to share the code between platforms. Each of these must have a specific development, repeating coding, testing, maintenance and distribution of new versions.

B. *Web Applications*

Mobile web applications run directly from any browser installed on the device. Since web applications are hosted on a server, an active Internet connection is required. The development process is similar to the standard web application development process, using HTML, CSS and JavaScript as base technologies.

In order to make the web application operational, the approval of the application stores is not required, since these do not take part of the process. In addition, users always use the most recent version of the application. When there is a change, these take effect immediately. This facilitates maintenance, since there is no fragmentation of users. Finally, the independence of the platform is its biggest advantage.

On the other hand, both performance and user experience may be affected. Likewise, security restrictions imposed by the execution of the code through a browser result in a more difficult access for the applications to all the features offered by the device [10].

C. *Hybrid Applications*

Although hybrid applications use web development tools (HTML, JavaScript and CSS), they are not executed from a browser. Hybrid applications are installed on the device and through an internal web container, they are executed locally and have access to some of the specific features of the device through an API.

Hybrid applications offer great advantages because they allow code reuse for the various platforms, access to device hardware, and distribution through application stores [8].

Hybrid applications have two disadvantages in relation to native applications:

i) User experience suffers from not using the native components in the interface.

ii) Execution could be slower due to the additional load associated to the web container.

Apache Cordova [11] is one of the most used frameworks. Its architecture is represented in Fig. 1. Another popular framework is Ionic [12], which is built on top of Cordova. Ionic provides a set of front-end components (HTML/CSS/JavaScript and AngularJS) that allows writing an HTML5 app with user interfaces that mimic a native app.

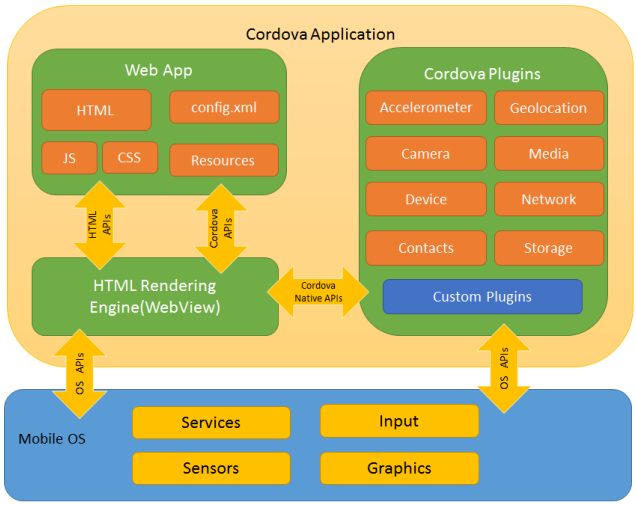

Fig. 1. Architecture of an Apache Cordova application

D. Interpreted Applications

Interpreted applications are built from a single project that is mostly translated to native code, with the rest being interpreted at runtime. Their implementation is platform-independent and uses several technologies and languages, such as Java, Ruby, XML, and so forth.

Unlike the aforementioned multiplatform development approaches, with this approach native interfaces are achieved, this being one of the main advantages.

Some of the most popular interpreted development environments are Appcelerator Titanium [13] and NativeScript [14].

Appcelerator Titanium is an open source framework for the development of mobile applications on iOS and Android. The JavaScript programming language is used, which is interpreted at runtime in the operating system of the device. When using the Titanium API, each element of the JavaScript code is assigned to its corresponding native element. Therefore, the Titanium API acts as a bridge, providing user interfaces built with native controls.

NativeScript is an open source project to build native applications using JavaScript or TypeScript. This framework allows to create interfaces with native components and access the functionalities of the device. When the application is compiled, part of the code is translated into native code, while the rest is interpreted at runtime. Fig. 2 shows a representation of the internal architecture of NativeScript.

E. Applications Generated by Cross-Compilation

These applications are compiled natively by creating a specific version for each target platform. Some examples of development environments used to generate applications by cross-compilation are Xamarin [15] and Corona [16].

Xamarin allows to generate native applications for iOS, Android and OS X sharing the same base code written in C#. The interfaces must be programmed specifically for each target platform (see Fig. 3). Statistical studies carried out by Xamarin report that the reuse of the code is close to 85%.

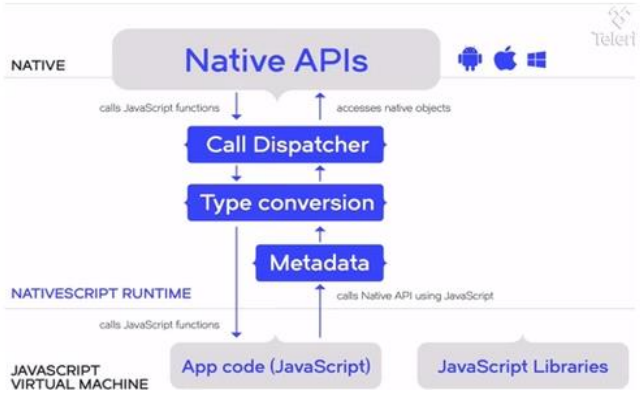

Fig. 2. Interpretation process with NativeScript.

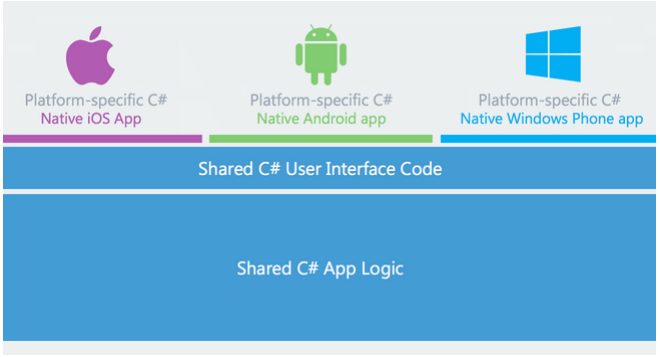

Fig. 3. Xamarin's unique development approach.

Corona is a cross-platform framework for creating general purpose applications and games for the main platforms, including OS X, Windows, iOS, Android, Kindle, Windows Phone 8, Apple TV and Android TV. The programming is done with Lua, which is a simple scripting language. A single base code is used, but unlike Xamarin, no rewriting of interfaces is required for each platform.

III. FEATURES FOR CHOOSING THE DEVELOPMENT APPROACH.

In the previous section the different approaches for the development of applications for mobile devices were presented.

One option is to develop native mobile applications, which make use of all the capabilities of the mobile device and allow a better user experience as outstanding positive aspects. However, development cannot be reused to support other operating system families, with higher development and maintenance costs as a result.

In contrast to native development, multiplatform approaches allow developing a single product and running it on different platforms. Each of these approaches has certain characteristics that naturally differentiate them, with advantages and disadvantages.

Taking into account that mobile application providers need to materialize their ideas in the shortest possible time, and try to reach the greatest number of end users, it is advisable to carry out an in-depth analysis of which is the most suitable approach for a given project. This choice is critical, and can lead to the success or failure of the software product.

In the first place, it is necessary to decide whether to perform a native or multiplatform development. The natural and possibly ideal answer is to choose the native approach, and to develop the same application on each platform. But how is planning affected by the simultaneous development of different projects? Does the company have the human resources needed to develop applications on each platform? Is the technical team trained in the different technologies inherent to each platform? How is project budget affected by embarking on parallel developments?

Possibly large companies can afford the organizational costs to carry out different versions of the same product. However, for a small and medium sized company the situation

could be entirely different. In this context, it is important to assess the different alternatives for cross-platform development, to determine which of them is best suited to the needs of the project. The choice of development approach to be used impacts the entire life cycle of the application. Once this choice is made, changing it in advanced stages is very difficult, since a complete reengineering of the software product would have to be carried out, resulting in higher costs.

As part of the present work, a classification of features that should be analyzed exhaustively in order to determine the most suitable approach for a specific application is presented.

Table 1 describes non-functional characteristics to be taken into account for the development of mobile applications, Table 2 details technical features that are of interest to application developers, and Table 3 summarizes the features related to software project management.

TABLE I. NON-FUNCTIONAL FEATURES

NF1 User Experience This is the set of factors related to user satisfaction when using a software product. The result is a positive or negative opinion about the product, affecting user emotions and expectations. Response speed, user interface design and usability, among others, are significant features, and similarity to the operating system standard is a plus. Considering that user experience is a subjective perception, a survey was distributed to 16 participants, who interacted with applications developed using the technologies being studied. After analyzing their responses, the technologies that would seem to favor end user experience can be identified. *Possible values: Very high, high, medium, low or very low.*
NF2 User Interfaces Even though user interfaces affect NF1, the diversity of component types used to build user interfaces should be analyzed on its own. As already mentioned in Section 2, choosing the development approach to be used defines whether interfaces are created using native or non-native components, and, if non-native components are used, they can be decorated to simulate native ones. *Possible values: Native or web.*
NF3 Performance Factor analyzing task execution and time required for resolution. Depending on the technology used, different performance levels may be obtained. This factor affects user experience, among others. The authors of this article carried out experiments related to mobile application performance in [5]. Based on these experiments, five possible values are suggested. *Possible values: Very high, high, medium, low or very low.*
NF4 Installation mode How an application is accessed varies based on the development approach used. In the case of web applications, no installation will be required; the user simply accesses an URL through a browser. This can benefit mass software product delivery, but it can be impractical for intensive use. The remaining development approaches analyzed generate applications that are distributed through app stores and that must be downloaded and then installed to be used. *Possible values: Download and install, or No installation.*
NF5 Battery use Mobile device battery life is perhaps the slowest-evolving component in relation to other device capabilities. Knowing which technologies generate an efficient energy consumption is a significant factor that should not be overlooked. Currently, the authors of this paper are carrying out experiments to assess energy consumption in applications generated using the different development approaches. The results of these experiments will be published in an article devoted specifically to discussing those issues. *Possible values: Very high, high, medium, low or very low.*
NF6 Disk usage / App size Another aspect to take into account is the space required for the installation of the application. *Possible values: Very high, high, medium, low or very low.*
NF7 Image rendering Depending on the type of application to be developed, it may be of interest to study how each technology behaves in the image rendering process. *Possible values: Very high, high, medium, low or very low.*
NF8 Initial boot time The goal should be achieving the lowest time possible since the application launches until it is shown on the front plane. Currently, the authors of this paper are carrying out experiments to assess initial boot time in applications generated using the different development approaches. The results of these experiments will be published in an article devoted specifically to discussing those issues. *Possible values: Very high, high, medium, low or very low.*

TABLE II. FEATURES FOR DEVELOPERS

D1 Integrated development environment (IDE) Software that assists programmers when developing applications. Some technologies offer their own IDE, while others use general-purpose IDEs. Productivity may vary from one IDE to another. *Possible values: Specific based on the case.*
D2 Programming Languages The programming languages, frameworks and/or libraries required to develop applications are different for the different approaches. *Possible values: Specific based on the case.*
D3 Open source / License and cost The type of licenses required to use the different technologies, whether they are sold at a cost or they are open to the community, should be considered in relation to error reporting and application evolution. *Possible values: Specific based on the case.*
D4 GUI Design Process of creating the graphical user interface (GUI), especially its software-support. Some tools provide WYSIWYG editors and the possibility to develop and test the user interface without having to constantly "deploy" it to a device or an emulator. *Possible values: Specific based on the case.*
D5 Learning curve The learning curve varies depending on the approach selected because the different technologies used have different levels of complexity. The quality and completeness of available documentation should also be considered, as well as the existence of a community of developers that collaborate with each other. To analyze the learning curve for each technology, a brief survey was carried out with 8 developers, who offered their opinions for the different cases. *Possible values: Very high, high, medium, low or very low.*
D6 Access to advanced device-specific features Use of development tools to access device features, such as camera, sensors, and so forth. *Possible values: Very high, high, medium, low or very low.*

TABLE III. FEATURES RELATED TO SOFTWARE PROJECT MANAGEMENT

G1 Target platforms Software providers must decide on which platforms available in the market they wish to be present. For this, the number of active users in each platform and their ongoing evolution is usually considered. Depending on the approach chosen, they may be able to develop applications for more than one platform. *Possible values: Supported operating systems.*
G2 Speed and Cost of Development Development costs may vary substantially depending on whether solutions need to be coded specifically for each platform or if code can be reused. This affects the number of technical teams that are required. *Possible values: Very high, high, medium, low or very low.*
G3 Maintainability Correcting errors or adding new functionalities may require coding in a specific way based for each platform. Additionally, the operating system internal fragmentation strongly affects application maintenance due to the cost of keeping an operating product for different platform versions. *Possible values: Very high, high, medium, low or very low.*
G4 Degree of maturity. Long-term feasibility There is a different maturity level based on the technology selected. Ideally, there should be comprehensive documentation and availability of support services to ensure operation in the long term and error correction. This also affects how the technology reacts to target platform changes and new versions. *Possible values: Very high, high, medium, low or very low.*
G5 Mobile apps categories A mobile application usually can be classified into one of the following categories: Social, Productivity, Tourism, Games, Multimedia, Institutional [6]. Depending on the category to which the application to be developed belongs, a specific approach could be preferable over the others. *Possible values: Social, Productivity, Travel, Games, Multimedia or Institutional*
G6 Offline usage The need for use of the software product to be developed without a connection to the Internet should be considered. *Possible values: Yes or No.*
G7 Code reuse In the cases of applications to be developed for more than one platform, it should be noted that not all approaches allow this. At this point, the amount of code that can be reused to

produce applications that run on different platform becomes relevant. *Possible values: Null, Low, High or Total.*
G8 Distribution / Access Depending on the approach used, the application can be distributed through app stores or directly accessed through a web browser. *Possible values: Through app stores or Through a web browser.*
G9 Potential Users Depending on the development approach used, a larger target user population may be reached. In the case of the web approach, the user population will be all those who have a browser on their mobile device and an internet connection. *Possible values: Limited to platform users, High, Very high or Unlimited.*

IV. COMPARATIVE ANALYSIS OF FEATURES IN TERMS OF THE DEVELOPMENT APPROACHES

A comparative analysis of features for mobile application development approaches was performed.

For each approach the analysis was performed using popular technologies at the time of writing this work. The native approach was studied based on the behavior of the applications developed for Android and iOS, which currently are the platforms with more users. For the mobile web approach, the study was done using HTML, Javascript and CSS. The hybrid approach was studied using Cordova [11] and Ionic [12]. As regards the interpreted approach, the behavior of Appcelerator Titanium [13] and NativeScript [14] was analyzed. Finally, the cross-compilation approach was studied using Xamarin [15] and Corona [16].

Table 4 presents a comparative analysis of non-functional features in mobile applications, Table 5 extends the analysis to include technical features of interest for developers, and Table 6 compares features related to software project management.

TABLE IV. COMPARATIVE ANALYSIS OF NON-FUNCTIONAL FEATURES

Technology / **Feature**	**Native applications**		**Mobile web applications**	**Hybrid applications**		**Interpreted applications**		**Cross-compilation applications**	
	Android	**iOS**		**Apache Cordova**	**Ionic**	**Appcelerator Titanium**	**NativeScript**	**Xamarin**	**Corona**
NF1 User Experience	Very high	Very high	Very low	Low	Medium	High	High	High	Low
NF2 User Interfaces	Native	Native	Web	Web	Web	Native	Native	Native	Native
NF3 Performance	Very high	Very high	Very low	High (Android). Low (iOS).	High (Android). Low (iOS).	Very high (Android). Low (iOS).	Very high (Android). Low (iOS).	Medium	Very low (Android). High (iOS).
NF4 Installation mode	Download from the app store and install	Download from the app store and install	No installation required. It is accessed through a browser.	Download from the app store and install	Download from the app store and install	Download from the app store and install	Download from the app store and install	Download from the app store and install	Download from the app store and install
NF5 Battery use	High	Very low	Low	Low	Low	Very low	Medium	High	Very high
NF6 Disk usage / App size	Very low	Medium	Very low	Low (Android) Low (iOS)	Medium (Android) Medium (iOS)	Very high (Android) Very high (iOS)	Very high (Android) Very high (iOS)	Medium (Android) Very high (iOS)	High (Android) Low (iOS)
NF7 Image rendering	High	High	Very high	Very high	Very high	High	Medium	High	High
NF8 Initial boot time	Very low	Low	Very low	Low	High	Low	High	Medium	Very low (Android). Very high (iOS).

TABLE V. COMPARATIVE ANALYSIS OF FEATURES FOR DEVELOPERS

Technology / Feature	Native applications		Mobile web applications	Hybrid applications		Interpreted applications		Cross-compilation applications	
	Android	**iOS**		**Apache Cordova**	**Ionic**	**Appcelerator Titanium**	**NativeScript**	**Xamarin**	**Corona**
D1 Integrated development environment (IDE)	Android Studio (Free). IntelliJ IDEA (Free and Licensed editions)	XCode (Free) AppCode (Licensed)	No official IDE. Multiple options available.	No official IDE. Multiple options available.	No official IDE. Multiple options available.	Appcelerator Studio IDE (Free and Licensed editions)	NativeScript for Visual Studio Code (Free)	Visual Studio (Free and Licensed editions)	No official IDE. Multiple options available.
D2 Programming Languages	Java, Kotlin	Objective C, Swift	HTML, CSS, JavaScript plus another server-side language	HTML, CSS, JavaScript	HTML, CSS, JavaScript	JavaScript	JavaScript o TypeScript	C#	Lua
D3 Open source / License and cost	Free	Free	Free	Apache 2.0 Software license (Free)	Free and paid versions	Free and paid versions	Apache 2.0 Software license (Free)	MIT license (Free)	Corona SDK (Free) Corona Enterprise and Corona Enterprise (Paid versions)
D4 GUI Design	XML Files. Free WYSIWYG editor.	XML Files. Free WYSIWYG editor.	HTML, CSS, JavaScript.	HTML, CSS, JavaScript.	HTML, CSS, JavaScript, Angular. Free and paid WYSIWYG editor	XML, TSS. Paid WYSIWYG editor	XML, CSS.	XML Files. Free WYSIWYG editor.	Lua
D5 Learning curve	High	High	Very low	Low	Medium	Medium	High	Very high	High
D6 Access to advanced device-specific features	Very high	Very high	Very low	High	High	Medium	Medium	Medium	Low

TABLE VI. COMPARATIVE ANALYSIS OF FEATURES FOR SOFTWARE PROJECT MANAGEMENT

Technology / **Feature**	**Native applications**		**Mobile web applications**	**Hybrid applications**		**Interpreted applications**		**Cross-compilation applications**	
	Android	**iOS**		**Apache Cordova**	**Ionic**	**Appcelerator Titanium**	**NativeScript**	**Xamarin**	**Corona**
G1 Target platforms	Android.	iOS.	All.	Android, iOS, Windows, Blackberry, Ubuntu, FirefoxOS, webOS, and FireOS.	Android, iOS and Windows.	Android, iOS, Windows and Blackberry.	Android and iOS	Android, iOS and Windows.	Android, iOS and Windows.
G2 Speed and Cost of Development	Very high	Very high	Very low	Low	Low	Medium	Medium	High	High
G3 Maintainability	Very high	Very high	Very low	Low	Low	Medium	Medium	Medium	High
G4 Degree of maturity. Long-term feasibility	Very High	Very High	Very High	High	High	Medium	Medium	Low	Very Low
G5 Mobile apps categories	Social, Productivity, Travel, Games, Multimedia or Institutional	Social, Productivity, Travel, Games, Multimedia or Institutional	Social, Travel or Institutional	Travel or Institutional	Travel or Institutional	Social, Productivity, Travel or Institutional	Social, Productivity, Travel or Institutional	Social, Productivity, Travel, Games, Multimedia or Institutional	Social, Productivity, Travel, Games, Multimedia or Institutional
G6 Offline usage	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes
G7 Code reuse	Null	Null	Total	High	High	High	High	Low	High
G8 Distribution / Access	Through app stores	Through app stores	Through a web browser	Through app stores	Through app stores	Through app stores	Through app stores	Through app stores	Through app stores
G9 Potential Users	Limited to platform users	Limited to platform users	Unlimited	Very high	High	High	High	High	High

V. CONCLUSIONS

There are different alternatives for developing mobile applications. On the one hand, it is possible to develop native applications specifically for each platform. On the other hand, there are four development approaches that allow developing an application and distributing it in different platforms: mobile web applications, hybrid applications, interpreted applications and cross-compilation applications

In this work, different features of the mobile application development process were analyzed and compared, using different development approaches. Each feature was analyzed using different technologies, thus covering all the development approaches mentioned.

This work is not intended to pick a winner development technology or approach, since this depends on each specific case. Instead, its purpose is acting as a support tool for choosing the approach to be used, offering information on how the different technologies behave in relation to each feature, and thus serve as a guide for future developments. The features analyzed can be seen in tables grouped under three taxonomies: *Non-functional features*, *features for developers* and *features related to software project management*.

VI. FUTURE WORK

As a future line of work, we are planning to extend the scope of this work, adding more incipient features and technologies, such as React Native [17]. Likewise, we expect to carry out a more in-depth analysis using the feedback received from other experts in the area.

Moreover, we expect to add the concept of Progressive Web Apps (PWA) [18], an emerging category of applications for mobile devices that is being positively reviewed by the Software Engineering community, to this comparative study.

ACKNOWLEDGEMENTS

We would like to thank all the people who contributed to the present work, either by answering surveys or by providing some technical details of the different technologies analyzed.

REFERENCES

[1] Mona Erfani Joorabchi, Ali Mesbah, Philippe Kruchten. Real Challenges in Mobile App Development, ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Baltimore, Maryland, US, October 2013.

[2] Florian Rösler, André Nitze, Andreas Schmietendorf. Towards a Mobile Application Performance Benchmark. ICIW 2014: The Ninth International Conference on Internet and Web Applications and Services, Paris, France.

[3] Spyros Xanthopoulos, Stelios Xinogalos, A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications, BCI' 2013, Greece, 2013.

[4] Delia, L.; Galdamez, N.; Thomas, P.; Corbalan, L.; Pesado, P., Multi-platform mobile application development analysis, Research Challenges in Information Science (RCIS), 2015 IEEE 9th International Conference on, Athens, Greece, 2015. 978-1-4673-6630-4.

[5] Delia, L.; Galdamez, N.; Corbalan, L.; Pesado, P., Thomas, P.; Approaches to Mobile Application Development: Comparative Performance Analysis. SAI Computing Conference 2017. London, UK. 978-1-5090-5442-8.

[6] Yonathan Aklilu Redda, Cross platform Mobile Applications Development, Norwegian University of Science and Technology, Master in Information Systems, June 2012.

[7] Dalmasso I., Datta S.K., Bonnet C. Nikaein N., Survey, comparison and evaluation of cross platform mobile application development tools. 9th International on Wireless Communications and Mobile Computing Conference (IWCMC), Cagliari, Sardina, Italy, 2013.

[8] Henning Heitkötter, Sebastian Hanschke and Tim A. Majchrzak, Comparing Cross Platform Development approaches For Mobile Applications, 8th International Conference on Web Information Systems and Technologies (WEBIST), Porto, Portugal, 2012.

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

[10] Tracy, K.W., Mobile Application Development Experiences on Apple's iOS and Android OS, Potentials, IEEE, 2012

[11] http://cordova.apache.org/ [Accessed 29 Sep 2017]

[12] https://ionicframework.com/ [Accessed 21 Sep 2017]

[13] http://www.appcelerator.com [Accessed 29 Sep 2017]

[14] https://www.nativescript.org/ [Accessed 29 Sep 2017]

[15] https://xamarin.com [Accessed 29 Sep 2017]

[16] https://coronalabs.com/ [Accessed 21 Sep 2017]

[17] https://facebook.github.io/react-native/ [Accessed 25 Sep 2017]

[18] https://developers.google.com/web/progressive-web-apps/ [Accessed 25 Sep 2017]