

A Students' Perspective of Native and Cross-Platform Approaches for Mobile Application Development

Paulo Meirelles[1(✉)], Carla S. R. Aguiar[2], Felipe Assis[2], Rodrigo Siqueira[3], and Alfredo Goldman[3]

[1] Federal University of São Paulo (UNIFESP), São Paulo, Brazil
paulo.meirelles@unifesp.br
[2] University of Brasília (UnB), Brasilia, Brazil
caguiar@unb.br, fel.cesar@gmail.com
[3] University of São Paulo (USP), São Paulo, Brazil
{siqueira,gold}@ime.usp.br

Abstract. One of the software industries with the most significant growth nowadays is the mobile application industry. In this paper, we study the differences between two mobile development approaches: (i) native (the development that's specific for each platform) and (ii) cross-platform (when a standard code is shared among different platforms). Inexperienced teams, students, and all newcomers in mobile development might find difficulties in choosing among approaches and frameworks. Our goal is to provide a comparison of these approaches from 3 different surveys with 166 participants, from students to experts perspectives, and study the factors that might influence the choice between them. We focus not only on quantitative aspects but also on qualitative and practical considerations that may also affect this decision (e.g., project requirements, final product category, and platform support). The results can aid students and inexperienced teams to obtain a more unobstructed view of the most recommended mobile development approach in a specific context and guide them to systematize their choice. Through our research, we observed that even when the quantitative aspects point towards a hybrid approach, it may not be the best decision due to other qualitative and practical factors. We found two nontechnical characteristics that may impact the selection of the best approach: previous team experiences with the chosen technology; and how well established and consolidated the technology is. With that in mind, we propose a mapping between project requirements and the benefits/drawbacks of each approach.

1 Introduction

In 2017, Stack Overflow, a popular and respected website among the developers, released the results of a global Developer Survey with 64,000 developers from all around the world. The survey revealed that 23% of the interviewees were mobile

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11623, pp. 586–601, 2019.
https://doi.org/10.1007/978-3-030-24308-1_47

developers. From this group, 64% worked with Android, 57% with iOS, 4.3% with Windows Phone and 0.7% with the Blackberry platform [22]. This data reflects the number of worldwide smartphone sales to end users, where 325 394.4 of units use Android, 71 525.9 use iOS, 4 395.0 use Windows, and 906.9 use Blackberry platform [24]. Therefore, developers and companies that wish their applications to reach a higher number of users will focus mainly on Android and iOS environments.

Applications that need to run on multiple platforms have distinct source codes per platform [4]. An application for a specific platform which has direct access to the device resources is known as a native app [11]. Developing a native app for multiple platforms is complicated and consumes a considerable amount of resources, since the source code must be duplicated for different platforms, possibly increasing project costs [19]. In response, tools that allow developers to implement a single application and export it to multiple platforms emerged. This sort of app is known as a cross-platform app and has the main advantage of simplifying the app life-cycle by centralizing development in a single codebase [4]. Despite this advantage, cross-platform approaches can also have their downsides, e.g., access to device resources and high memory usage [15].

The game development industry is one example where software development must be, as a requirement, multi-platform. The most used and mature game engines, such as Unity and Unreal, allow cross-platform development not only between mobile platforms, but even between consoles, desktop operating systems, smart TVs, and web browsers. In 2016, approximately 59% of the mobile games used cross-platform technologies as their game engine [23].

Native and cross-platform approaches have different trade-offs, and it is necessary to consider them before selecting the most appropriate. Among different contexts, technologies and requirements, it can be difficult to determine which is the best approach for a given project. Evidence of these factors, considering both technical and non-technical aspects of each strategy and technology, is necessary. We aim to provide such evidence through surveys, where we gathered the main qualitative and non-technical aspects when deciding among technologies.

Previous studies have compared both qualitative and quantitative aspects of adopting one of the approaches [14,15]. We compare native and cross-platform development from students' perspectives. We also validated these perspectives with professional developers point of view. We conducted surveys, separating our target into three categories: beginner (undergraduate students with different levels of practice), intermediate (advanced students from graduate courses) and advanced (professionals from different communities of mobile software development). After analyzing the results of our surveys, we inferred that quantitative characteristics might not be the primary factor to consider when selecting between native or cross-platform.

The remainder of this paper is organized as follows. Section 2 discusses the background of mobile development approaches and technologies. Section 3 presents some related works to position our contribution. Section 4 presents the research design of this study, discussing our research questions and methodology.

In the sequence, Sect. 5 summarizes our results, in particular, based on surveys conducted with students and mobile software development professionals. Finally, Sect. 6 concludes the paper, highlighting its main contributions and providing paths for future work.

2 Background

Mobile applications have particularities in their development due to hardware features, software distributions, and release cycles. Developers have to handle issues related to storage capacity, mobile specifications, mobility, user experience, security, and privacy [11]. Mobile applications are commonly dynamically developed, released in small cycles, and they require small storage and have a low cost to the final user [9].

Native applications are designed for a particular Operating System (OS), e.g., iOS or Android, which means there is no portability between platforms. Commonly, these apps are available at digital stores (e.g., App Store for iOS and Play Store for Android), responsible for downloading the app in the devices and make it available to the end user [18]. Native apps tend to follow the interface (look and feel) and the technical patterns supported by the specific platform. This sort of app has direct access to device resources such as sensors, cameras, GPS (Global Positioning System) [11]. This approach delivers both high performance and specialized UI/UX for users [18]. However, it requires a specific framework and SDK [15].

An alternative to developing an app for multiple platforms is the use of cross-platform technologies. This strategy produces a single code base that enables app distribution for various platforms. Two different cross-platform technologies are being considered: Web apps and Hybrid apps. **Web apps** are applications developed using web technologies customized for mobile restrictions. The device needs to access these apps through a browser (e.g., Firefox, Safari, Chrome). They provide a uniform experience across different mobile platforms [18]. This technique follows the client-server model, in which the source code runs in a remote server. However, it uses modified front-end technologies that can detect mobile device restrictions and adapt the UI. A vast number of frameworks exist to support this approach, as illustrated in Table 1, and most of them use well-known front-end technologies for the web.

Hybrid applications combine native and Web App approaches. This approach enables developers to use web technologies to target multiple mobile platforms from a single code base and produce an app that works similarly as a native app. Hybrid apps installed in the device, internally use the WebKit rendering engine, an engine that depends on the browser of the OS, and is available in Safari, Mail, App Store, and many other apps on macOS, iOS, and Linux [2]. It shows web content inside the application while using native components to improve the user experience [1]. In this paper, we define a hybrid app as a web app, primarily built using HTML5 and Javascript. A native container wraps the web app providing access to native platform features [20].

Table 1. Examples of cross-platform technologies.

Platform	Languages
Alpha Anywhere	C++, C#, Javascript, VisualBasic
Cocos 2D	C++, C#, JS, Swift, Objective-C
Corona	Lua
Ionic	Javascript, HTML, CSS
PhoneGap	Javascript, HTML, CSS
Qt	C++
Sencha	Javascript, HTML, CSS
Titanium	Javascript
Unity	C#, UnityScript
Unreal	C++
Xamarin	C#, ruby

According to Ciman [8], the battery consumption is significantly increased when using hybrid technologies, in some cases exceeding 250% when compared with an equivalent native app. Also, according to Rakesh [20], compared to the native approach, hybrid apps usually consume a significantly more significant amount of RAM, they handle animations with less fluidity and often appear sluggish during page and state transitions.

Deciding among native, web app, and hybrid approaches require a careful analysis of the technical and nontechnical characteristics of the project. However, they are not equally weighted, since they depend on a variety of external factors such as development experience on the target OS, project schedule, software requirements, and the number of platforms to be released.

3 Related Work

We examined related works on mobile development. We observed in the past 13 years a significant discussion, research, and development of new approaches to improve mobile development [13]. We were looking for alternatives that could lower the costs and development time with the least possible impact on other aspects such as app performance, UI/UX, and security. Some of these studies have goals and methodologies similar to ours. We compared some approaches adopted by other researchers with ours.

Charland et al. [7] present a theoretical comparison of cross-platform and native application development on aspects like UI coding, performance on each platform and compliance with conventions. They concluded that hybrid solutions have several advantages over native ones when dealing with cross-platform development. Following the same motivation, we also considered non-quantitative factors in our research, in particular, empirical (e.g., project requirements).

Prezotto [21] makes a distinction between native, web app, and hybrid approaches. The author made proof of concept by building a hybrid app. After the implementation of the solution, Prezotto noticed different behavior between platforms that executed the application. We note that there is a concern in verifying the capacity of the hybrid apps in front of the Native approach. In our surveys, we aimed to highlight those differences to beginner and advanced developers, analyzing their perspectives.

Dalmasso [10] presented several hybrid app development tools available in the market (e.g., PhoneGap, Titanium, Rhomobile, and JQuery Mobile) and made a comparison through a survey. Dalmasso's work discussed a qualitative study of these frameworks to provide a quick overview to mobile developers. Our research intends to cover not only the cross-platform approaches but also put it in a more empirical comparison against the native approach.

Majchrzak and Grønli [16] discussed how to choose between different hybrid technologies. The author adopted Ionic, React, and Fuse as the objects of his study, considering purely technical and mostly quantitative aspects. Most recently, Biørn-Hansen et al. [5] defined an overview of core concepts and domain terminology regarding cross-platform app development, establishing a taxonomy. The authors argue that the developers face many cross-platform development options to choose and there is no one technology fits all. Other works of Biørn-Hansen and Majchrzak [6,17] suggested Progressive web apps as the final approach to cross-platform development. In our research, we discussed nontechnical issues when determining the ideal strategy for a project. We also discussed how to select a mobile development approach, but we did not consider individual technologies for this decision.

Latif [15] made a case study of how various hybrid technologies work (i.e., language, architecture, and performance) and highlighted their benefits and their challenges. Gaouar [14] conducted a similar work, where the author mapped the technical requirements for cross-platform into six categories: support platforms, documentation, security, access to native resources, look and feel, and resource consumption. Different from those authors, we did not focus on mapping the characteristics of each approach or technology, but to take those already highlighted and mapped topics from the students perspective. In our study, we also compare cross-platform and native development of each Operating System, not only related to the multi-platform SDKs itself.

Carlström [3] conducted a study closer to our goal, a practical experiment that separated results for each predefined use cases defined. In general, the time spent in the development of cross-platform applications were smaller than native development separately. Jiang [12] considered technical, empirical, and practical aspects of the different solutions. The author concluded that there is no universal solution, once there is no extreme difference regarding development processes and products to name a final approach to mobile app development.

4 Research Design

We aim to investigate native, hybrid, and web app approaches regarding mobile applications development for inexperient teams. In the context, we guide this study to answer the principal research question:

#RQ: *What should be considered when choosing the best approach (Native, hybrid, or Web App) to develop a mobile app?*

The following questions emerged to investigate this scenario when respecting the qualitative context:

#Q1: *What are the technical advantages (technical features) of cross-platform (hybrid) frameworks from the perspective of inexperienced mobile developers and experts?* We hypothesized that inexperienced mobile developers would suggest that the strongest technical advantage will be the cross-platform (hybrid) "cost per platform". In counterpart, we expect that experts will not corroborate on all cases. Even with all these considerations, analyzing other practical factors is required, and it led us to the second research question.

#Q2: *What non-technical factors influence decision making regarding the approach and framework used for developing a mobile application?* We hypothesized that several subjective aspects like "project requirements", "technology learning curve" and "previous practice with technology" will not only influence but be decisive in the choice of an approach. We expected that developers would classify nontechnical aspects to be more relevant to decide between approaches than technical ones.

We conducted three questionnaires to collect data to answer these research questions. From the data collected and analyzed from these questionnaires, we elaborated a guideline that guides students with little or no mobile development experience to make the best decision on mobile application development. Also, it could be used as guidelines in mobile development courses with a project-oriented approach.

4.1 Students' Perceptions on Implementing Using Native and Cross-Platform Mobile Solutions

The first questionnaire collected the perception from advanced Computer Science undergraduate and graduate students at the University of São Paulo (USP). Previous to questionnaire application, students developed, through 4 months course, simple features with both Native and Hybrid mobile frameworks. This survey intended to evaluate the main advantages and difficulties faced by intermediate developers when implementing a mobile application in both approaches, and if these challenges are perceived differently in hybrid and native strategies. Also, we could evaluate the learning curve in both technologies and this way answer the #Q1 and #RQ.

The questionnaire was composed of four sections:

- **Section 1 - Participants Profile:** age, gender, prior knowledge of programming languages (such as C, C++, Python, PHP, and Ruby). Ex: "Have you ever developed some software that is being used by other users?";

- **Section 2 - Previous experiences on mobile development:** familiarity level with programming languages and technologies such as Java to Android, Android Studio, HTML, Javascript, Angular 2, and Ionic. Ex: "How many other development platforms have you ever used?";
- **Sections 3 and 4 - Perceptions on Android Studio and Ionic development:** difficulty to install Web Services to implement the login, development of student registration feature. Implementation of Bluetooth and Wifi communications, camera image capture, QR-Code reading, and finally the upload of files using Web Services.

All responses were based on the Likert's scale with the following levels: (1) Strongly disagree; (2) Disagree; (3) Not agree or disagree; (4) Agree; (5) Strongly agree. Finally, the student had two open questions to write open comments on the development using Android Studio and using Ionic.

4.2 Students and Developers' Perspectives on Selecting Mobile Technology

The following two questionnaires intended to answer the #Q2 and #RQ. We conducted surveys in two distinct contexts. First, we wanted to understand the factors developers with little experience or utterly inexperienced in mobile development take into account on their first mobile projects. Second, we validated this set of factors with advanced and experts mobile developers.

The second questionnaire collected the point of view of undergraduate students of the Software Engineering course from the Federal University of Brasília. In this context, we collected data from students that enrolled in a based-project course, where students are expected to choose the technologies they are going to use (there are no restrictions) and to justify their choice. Differently, from students evaluated in the first survey, these students developed an entire mobile application with only the selected approach (native or cross-platform).

The third questionnaire collected the point of view from experienced mobile developers. We provided an anonymous online and shared to developers enrolled in mobile online communities. We consider specialists those who fit into the following requirements:

- 3 or more years of professional experience in software development.
- Worked at least on two published software for mobile.
- Worked on at mobile development with recent technologies (2015 onward).

Both questionnaires collected data about the current knowledge of the participant and also to determine the primary factors when selecting between different approaches, frameworks, and platforms. The questionnaire was composed of four sections:

- **Section 1 - Participants Profile and previous experience:** age, gender, prior knowledge of programming languages (such as C, C++, Java, Javascript, Python, Objective-C, and Swift). Ex: "How many projects were published, how many are still in production?"

Table 2. Nontechnical factors that interfere with the choice mobile approach.

1 - Previous knowledge regarding the platform
2 - Learning Curve
3 - Development project requirements
4 - Stakeholders preferences
5 - Code reuse
6 - Quality of support (Documentation, API, Community, Wiki)
7 - Recommendations of specialists
8 - Google research trend
9 - Final product performance (Battery consumption, response time, graphical performance)
10 - I can't tell. (In this case, do not consider all of the above options)

- **Sections 2 and 3 - arbitrary preferences regarding mobile application development:** cover the currently most used languages and frameworks in mobile development. For example, Xcode (Objective-C and Swift) and Android Studio (Java) for native development, and Ionic framework and ReactNative for hybrid approaches;
- **Section 4 - Factors that interfere with the selection of mobile technology:** the list of the factors evaluated are depicted in Table 2.

When asked about each of those factors, the respondents have different options. They could select the priority level they attribute to each of them when starting a software project. All response options followed a scale from 1 to 5: (1) "I do not know or can not answer"; (2) "little interest or value"; (3) "moderate interest or value"; (4)"big interest or value"; (5) "maximum priority, interest, and value for the solution over the others".

5 Exploratory Data Analysis

This section presents the data collected from the respondents to the questionnaires described in Sect. 4. First, we analyze the learning curve of both approaches, where 40 computer science undergraduate and master students implement, for the first time, simple features in both native Android and Ionic. We evaluate their perceptions and difficulties while configuring the development environment, using IDEs, programming language, testing and deploying a mobile application in both approaches.

In a different University, we analyze which factors 96 undergraduate software engineering students take into account to choose between native and hybrid approaches when developing their first mobile application. Finally, we evaluate if the same factors guide more advanced practitioners when choosing an adequate approach to develop a mobile app. Table 3 gives an overview of the results

Table 3. Overview of the surveys' participants.

Survey	Nr of participants	Profile interviewee	Survey intent
1	40	Undergraduate and Master Students in Computer Science	Evaluate technical comparison between both native (Android) and hybrid (Ionic)
2	96	Undergraduates in Software Engineering	Evaluate the factors an inexperienced developers consider when choosing technology for a mobile project
3	30	Experienced mobile developers	Evaluate the factors experienced developers/experts consider when choosing technology for a mobile project

obtained from these three surveys with 136 students and 30 professionals. From the acquired data, we derived both non-technical and technical aspects to define between native and hybrid approaches when students begin a new mobile project.

5.1 First Survey

We conducted experiments related to Android (Native) and Ionic (hybrid) throughout the Mobile Computing course at the University of São Paulo. Most of the 40 students had a good background in C, Java, and Python programming languages. On C language, 80% declared that they have "good" or "excellent" skills. For Python, 42% reported having the "medium" knowledge. 32.5% of the respondents have "good" to "excellent" domain over the language. More than 65% declared to have "zero" or "minimum" knowledge of Javascript (mostly used by Ionic), PHP, and Ruby programming languages.

Regarding the IDE/Framework, 87.5% of the respondents answered that they did not have any previous experience with Android Studio and 80% claimed the same for Ionic. After developing in both technologies (Android and Ionic), they were asked the question "How easy is to set up the development environment?". 67.5% of the participants said that it is easy to configure Android Studio and only 35% had the same opinion about the Ionic environment. More than half of students (55%) claimed the Ionic environment is unintuitive to set up.

In general, the results show some pieces of evidence that developing simple tasks, both Android and Ionic are quite similar. We were also interested in investigating the differences between Android and Ionic environments when using particular hardware resources.

We verified the use of Bluetooth, Wifi, and a camera device on both platforms. Our results indicated that developers found more complicated to access the hardware in Ionic than in Android Studio.

There were two questions about their perception of the advantages and disadvantages of Android Studio, and the second one, on the aspects of Ionic. Most comments were favorable toward Android IDE like: "very useful IDE", "excellent

debugger" and "good log terminal and refactoring tool". However, many of them complained about the quite steep learning curve, the difficulties when manually configuring the development environment and with deprecated methods.

Concerning Ionic, students were quite unanimous in its use for straightforward applications. There were comments as "Easy to do HTML interfaces and to navigate among pages" and "Easy for developers with knowledge on HTML and Javascript". However, there were lot more critics related to several topics as: "Not good to use the smartphone resources", "hard to find the correct libraries", "large dependence on third-party plug-ins". Students also complained about the framework being quite recent with not enough documentation, the "Lack of comprehensive documentation and lack of support", "hard to configure it", "sensation of adopting a'Frankenstein like' application with different functional parts", "the final application ran slower than the native correspondent", "hard to debug", and "It is frustrating to make HTTPS requests work".

5.2 Second Survey

We conducted experiments related to native, hybrid, and web app mobile development throughout the Software Development Methods course at the University of Brasília. The survey intended to understand how the 96 students enrolled in this course chose the development techniques and tools for their first mobile projects. The students evaluated are mostly men (85%) and have an age range between 18 and 25 years old. Most of them (77.1%) have 0 to 1 year of practice in mobile software development.

When asked about their current knowledge regarding programming languages, students declared a medium to a good experience in C/C++ (74%), Java (58.4%), and Python (52.1%). Opposing to that, the iOS platform languages, Swift and Objective-C, where the most uncovered among all students, presenting 88.5% of the answers as "zero knowledge". We observed the Javascript had well-distributed sample space with 29.2% of respondents with "low or zero knowledge", 44% with "medium knowledge" and 26.1% with "high knowledge".

We asked them about specific frameworks, platforms, and IDEs that they use during hybrid development. 42.7% of the students have practice at least in Ionic, React Native, PhoneGap, Xamarin, or Rails. As for the native approaches, Android was the most used native platform among the students with 49% of the responses. According to this data, we can observe that the students have practice with cross-platform technologies.

We also asked the students to select three factors (from a predefined list showed in Table 2) that would be decisive when choosing what platform, framework, technology, or approach to decide when developing a mobile project. 67.7% of the respondents indicated the "quality of support (wiki, community, documentation, API)" and the "project requirements" as the factors with the highest score. The second most selected factors were "previous knowledge" and "learning curve" (both nontechnical topics), with 53.1% and 44.8% of responses, respectively. It is evident that, for beginners, technical and community support available are crucial for their framework evaluation. It means that inexperience mobile

developers are unlikely to be early adopters of mobile emerging technologies since it increases the risks. Equally vital to beginners are project requirements and previous experience in technology. However, when it comes to projects requirements, they tend to choose the hybrid approach when it is required availability in multiple platforms. This fact is due to rigid deadlines, limited budgets, and risk management. Consequently, if the project requirement is to have the application available on only one platform, inexperienced developers tend to choose the native approach.

5.3 Third Survey

We conducted a third survey to collect data from 30 advanced developers who actively participate in the developers' community to verify our founds from the two previous questionnaires. The participants are mostly men, and they have an age range from 22 to 35 years old (80%).

When asked about their current knowledge regarding programming languages, the majority 50% declared from medium to good experience in Objective-C/Swift (56,7%), Java (66,6%), Javascript (53,5%); other languages were also mentioned, such as PHP, Ruby, C# e Python, C++. More specifically when argued about their empirical preferences for each technology (outside of any specific use case), we can observe that between cross-platform development technologies, the most significant community of developers are Xamarin and React Native, and Web Apps, all with just over 30% of any level of acceptance between low and maximum. The preference for native alternatives (iOS and Android) remain at 90% of any level of choice.

As in the previous questionnaire, among a list of factors, the developers had to choose up to 3 factors that influence the choice of the framework/technology when starting a software development project mobile. Just like the students, "quality of the support" (documentation, Wiki, APIs) with 46.7% was the most relevant aspect for choosing a technology, but also the "Performance of the final product" aspect also 46.7% in contrast to previous research. "Learning curve" (36.7%) and "prior knowledge of technology" (33.7%) were also considered again resembling the result of previous questionnaires.

In the questionnaire, we presented to the participants with mobile applications hypothetically developed with the iOS and Android platforms. In each option, they should choose which technology (Native, Hybrid, Web App) would have the preference for software development. The types of applications were: (1) "Electronic game", (2) "CRUD based application", (3) "Application with complex UI/UX elements and animations", (4) "Application that relies heavily on device features (GPS, Accelerometer, Gyroscope, iCloud, Native Login, Touch ID, Push Notifications, etc.)". For alternatives (3) and (4), a significant preference over the native approach was evident (90%); for CRUD-based applications, the majority had a choice for cross-platform approaches, with 40% hybrid and 13.13% Web-App. Finally, for electronic games, there was a preference of just over 50% for the native approach and 3% for hybrid approaches (Unity, Unreal Engine, and Game Engine).

6 Discussion

In this section, we discuss some of the insights and results obtained from the collected data. Our results show that the most critical factor to be considered when choosing a mobile development approach is "how well established and consolidated the technology is". It means that developers, in general, should check how good or how extensive is the technical documentation (e.g., code documentation, wiki, forums) and how active is their support and community in a project context. This factor is directly linked to the stability and consolidation of mobile technology in the developer community in general. The previous practice and learning curve of the development team regarding the candidate technology also influence this decision. The surveys evidence that students intuitively prioritize, with a slightly different priority, some of the same factors of experts.

We hoped that, contrary to what literature and recent studies point to, hybrid approaches are not the best choice for most cases. This statement seems to be true even if several hybrid technologies are already well consolidated. The adoption tends towards native approaches as the application requires UI/UX features and non-standard animations. The same choice is preferable as more native resources (not common to other devices) are needed.

We observed from the collected data that there is no optimal choice for all situations, where each project and each kind of software has its particularities. From the results obtained from data analysis, we proposed a general set of recommendations depicted in Table 4 and described as follows.

Native approach presents the best UI/UX graphical performance, easier to implement complex animations. It has maximum conformity with the selected platform (Android or iOS) design guidelines and standards. The most used native Operating Systems excels in providing extensive documentation, technical support for bug fixes and extensive community of developers already with ongoing projects. For the native approaches, the vendors offer a variety of well-established development tools. With the proper implementation of the features, optimization for low battery consumption, RAM usage, video memory, and latency are viable. We recommend the native development approach for:

- Projects that need high performance;
- Projects that require technical support from the technology providers;
- Projects that need a faster learning curve for inexperienced developers.

Hybrid approach presents an acceptable UI and UX compliance for each platform for the basic native graphical elements. However, it imposes a higher difficulty in implementing new (non-standard) or more complex animations and graphical elements. This approach excels in the cost-benefit regarding the budget and the schedule in front of the number of platforms that can be implemented. It happens due to its main advantage concerning code reuse among different platforms. Even though a specific platform code is still needed for features that use specific resources. Not all of the hybrid technologies provide a well-established development environment (e.g., IDEs, SDKs, CLIs). The current

Table 4. Recommended criteria to choose between different approaches in mobile application development.

	Native	**Hybrid**	**Web App**
UI/UX performance	High	Low	Low
UI/UX customization	High	Medium to Low	Relative to web technology adopted.
UI/UX platform guideline compliance	Maximum	Medium to High	None
Documentation, support, community	High	Low to Medium (relative to the selected hybrid technology)	Relative to web technology adopted. But low in the context of web app
Hardware stress (battery consumption, RAM, GPU, response time)	Low	Medium to High, (Very perceptive in some cases)	High to Maximum
Optimized product output (binary or generated source)	Can be very optimized (relative to developer)	Not optimized	Not optimized
Relative cost per number of platform.	Maximum cost per platform.	medium or low (relative to the selected hybrid technology)	Minimum cost per platform
Other comments	Easy to set up the working environment, a high number of developer tools; maximum optimization for game software.	Usually longer to set up the working environment (with exceptions);	Limited access to the device resources
Recommended for	For projects with a lower priority in the budget, projects that do not need to be deployed on multiple platforms, apps that need high or maximum performance are required as well as a need for proper support from the technology providers and needs a faster learning curve in the development team.	For projects targets, two or more platforms, require access to primary device resources, projects that do not require excellence in graphical performance or compliance and have a high priority of a lower budget.	For apps that have simple UI/UX requirements do not need to be stored in the device, apps that need to run in an undetermined number of devices and applications that do not require device-specific features (camera, accelerometer, touch-id, gyroscope, compass, proximity, microphone).

most used hybrid approaches also provide good and extensive documentation, wiki, and support from their communities. The final output code or binary application is not optimized for an individual platform, leading to higher hardware stress compared to its native equivalent. We recommend the hybrid approach for:

- Projects targeting two or more platforms;
- Projects that require access to essential device resources;
- Projects that do not require excellence in graphical performance or compliance;
- Projects that have budget restrictions.

Web App approach presents the lowest UI/UX compliance, as its frameworks do not comply with the specific platform guidelines. This approach excels

for use cases that have already a production environment and have simple end-user features (e.g., CRUD-based applications). This approach has the best level of code reuse. In counterpart of its higher compatibility, the access to the device resources is very limited. We recommend this approach for:

- Projects that have simple UI/UX requirements;
- Projects that do not need to be completely stored in the device;
- Projects that need to run in an undetermined number of devices;
- Projects that do not require specific device features (e.g., camera, accelerometer, touch-id, gyroscope, compass, proximity, microphone).

7 Conclusion

In this paper, we studied the considered factors to choose a better approach, either a native or cross-platform, to develop a mobile application. Our findings can support students and professors to systematize the choice of the most suitable framework in a mobile project during an undergraduate or graduate course related to the software engineering field as well. To answer the principal research question, we conducted surveys with 166 participants from three groups: inexperienced, intermediate, and expert mobile developers. From the data analysis, we concluded that practitioners, no matter their experience, regard both technical and nontechnical requirements such as the support offered by the technology (documentation, wiki, and community), the performance of the final product, project requirements, and learning curve.

When it comes to technical aspects, we studied the advantages and challenges of the native, hybrid, and web app mobile approach. We performed an experiment with intermediate developers where they implemented features in both native Java Android and Ionic and later, compared their experience. Students came to similar conclusions of experts, where we concluded that practitioners are strongly inclined to choose a native approach when it comes to achieving high performance (i.e., low battery consumption, low latency, high graphical performance, animations, low memory usage). However, if performance is not obligatory, and there are few hardware/device access, hybrid solutions are considered a better approach to develop mobile apps for multiple platforms.

Based on these results, we recommended a set of technical and nontechnical aspects to help newcomers of mobile development to systematize the choice of the mobile software development strategy for their projects. We should stress that we found no evidence that there is an optimal mobile approach choice, and it is strongly influenced by both technical and nontechnical factors presented in this papers, and also by the context of the application.

7.1 Threats to Valid and Future Work

As threats to validity, all of the surveyed students are from Brazil as well as most of the expert participants are from the Brazilian industry and communities. We

developed the questionnaires from our experiences in software engineering and software mobile development and teaching. Thereby we did not formally evaluate the reliability of the surveys.

Regarding the related work, we conducted a systematic literature review during the first phase of this study in 2017. After the analysis of the questionnaires, we complete our review with a few related papers from 2018.

As future work, we need to complete our analysis collecting data of developers in other countries to confirm our findings from different perspectives. Another study should perform a quantitative analysis comparing both native and cross-platform mobile approaches to validate the results. We also suggest using the profiling tools to have a clear understanding of the performance difference between implementations of the same feature in native and cross-platform applications.

Acknowledgements. The authors are indebted to the students from USP and UnB, as well as, the professionals for answering the questionnaires providing valuable data for this study.

References

1. Ahti, V., Hyrynsalmi, S., Nevalainen, O.: An evaluation framework for cross-platform mobile app development tools: a case analysis of adobe phonegap framework. In: Proceedings of the 17th International Conference on Computer Systems and Technologies, CompSysTech 2016, Palermo, Italy, 23–24 June 2016, pp. 41–48 (2016)
2. Webkit: Webkit is a trademark of Apple Inc. (2017). https://webkit.org
3. Axelsson, O., Carlström, F.: Evaluation targeting react native in comparison to native mobile development (2016). Student Paper
4. Bernardes, T.F., Miyake, M.Y.: Cross-platform mobile development approaches: a systematic review. IEEE Lat. Am. Trans. **14**(4), 1892–1898 (2016)
5. Biørn-Hansen, A., Grønli, T.M., Ghinea, G.: A survey and taxonomy of core concepts and research challenges in cross-platform mobile development. ACM Comput. Surv. **51**(5), 108:1–108:34 (2018). https://doi.org/10.1145/3241739. http://doi.acm.org/10.1145/3241739
6. Biørn-Hansen, A., Majchrzak, T.A., Grønli, T.-M.: Progressive web apps for the unified development of mobile applications. In: Majchrzak, T.A., Traverso, P., Krempels, K.-H., Monfort, V. (eds.) WEBIST 2017. LNBIP, vol. 322, pp. 64–86. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93527-0_4
7. Charland, A., Leroux, B.: Mobile application development: web vs. native. Commun. ACM **54**(5), 49–53 (2011)
8. Ciman, M., Gaggi, O.: An empirical analysis of energy consumption of cross-platform frameworks for mobile development. Pervasive Mob. Comput. **39**, 214–230 (2017)
9. Corral, L., Janes, A., Remencius, T.: Potential advantages and disadvantages of multiplatform development frameworks - a vision on mobile environments. Procedia Comput. Sci. **10**, 1202–1207 (2012)
10. Dalmasso, I., Datta, S.K., Bonnet, C., Nikaein, N.: Survey, comparison and evaluation of cross platform mobile application development tools. In: 2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 323–328, July 2013

11. El-Kassas, W.S., Abdullah, B.A., Yousef, A.H., Wahba, A.M.: Taxonomy of cross-platform mobile applications development approaches. Ain Shams Eng. J. (2015). http://www.sciencedirect.com/science/article/pii/S2090447915001276
12. Jiang, S.: Comparison of native, cross-platform and hyper mobile development tools approaches for iOS and Android mobile applications. University of Gothenburg (2016)
13. Johnson, H., et al.: Methods and systems for providing platform-independent shared software components for mobile devices. US Patent 6,986,148, 10 January 2006. https://www.google.com/patents/US6986148
14. Gaouar, L., Benamar, A., Bendimerad, F.T.: Desirable requirements of cross platform mobile development tools. Electron. Devices **5**, 14–22 (2016)
15. Latif, M., Lakhrissi, Y., Nfaoui, E.H., Es-Sbai, N.: Cross platform approach for mobile application development: a survey. In: 2016 International Conference on Information Technology for Organizations Development (IT4OD), pp. 1–5, March 2016
16. Majchrzak, T., Grønli, T.M.: Comprehensive analysis of innovative cross-platform app development frameworks. In: Proceedings of the 50th Hawaii International Conference on System Sciences (2017)
17. Majchrzak, T.A., Biørn-Hansen, A., Grønli, T.M.: Progressive web apps: the definite approach to cross-platform development? In: Proceedings of the 51st Hawaii International Conference on System Sciences (2018)
18. Malavolta, I.: Beyond native apps: web technologies to the rescue! (keynote). In: Proceedings of the 1st International Workshop on Mobile Development, Mobile!SPLASH 2016, Amsterdam, Netherlands, 31 October 2016, pp. 1–2 (2016)
19. Martinez, M., Lecomte, S.: Towards the quality improvement of cross-platform mobile applications. In: 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft), pp. 184–188, May 2017
20. Rakesh, P.K., Kannan, M.: Online mobile application development using ionic framework for educational institutions. Int. J. Adv. Res. Methodol. Eng. Technol. **1** (2017)
21. Prezotto, E.D., Boniati, B.B.: Estudo de frameworks multiplataforma para desenvolvimento de aplicações mobile híbridas. Universidade Federal de Santa Maria, Trabalho de Conclusão de Curso (2014)
22. StackOverflow: Developer survey results 2017 (2017). https://insights.stackoverflow.com/survey/2017
23. Unity: Cross-platform game engines (2018). https://unity3d.com/public-relations
24. Woods, V., van der Meulen, R.: Gartner says worldwide smartphone sales grew 9.7 percent in fourth quarter of 2015 (2016). http://www.gartner.com/newsroom/id/3215217