Proc. of the 2nd International Conference on Electrical, Communication and Computer Engineering (ICECCE) 12-13 June 2020, Istanbul, Turkey

Framework Choice Criteria for Mobile Application Development

Mohamed Karim KHACHOUCH
SIGER Laboratory
FST of Fez, Sidi Mohamed Ben Abdellah University
Fez, Morocco
mohamedkarim.khachouch@usmba.ac.ma

Ayoub KORCHI
SIGER Laboratory
FST of Fez, Sidi Mohamed Ben Abdellah University
Fez, Morocco
ayoub.korchi@usmba.ac.ma

Younes LAKHRISSI
SIGER Laboratory
ENSA of Fez, Sidi Mohamed Ben Abdellah University
Fez, Morocco
younes.lakhrissi@usmba.ac.ma

Anis MOUMEN
System Engineering Laboratory
ENSA, IbnTofaïl University
Kenitra, Morocco
amoumen@gmail.com

***Abstract*—Mobile applications development is complex. In order to build mobile apps that target all users, developers must put up with all the existing operating systems, software development kits (SDKs), devices various screen resolutions and existing mobile app development technology that is in constant evolution. And if this is not enough, developers must choose the right approach to build their app. Taking those decisions has a huge impact on the eventual cost, time, and success of an application building. In order of importance, it comes right after choosing the functionality of the app itself. If developers fail to match an app demands to the right development approach, it can turn their project into a certain failure. This paper aims to compare between the mobile app development approaches and suggest a decisional framework to choose between them.**

Keywords—mobile development, cross-platform, development approaches

I. INTRODUCTION

Choosing the right technology to develop an application is never easy. The architect has to make a decision depending on many criteria starting with answering some basic question: What is the best framework and technology to use to develop his mobile app?

To answer that, we have to remember what desktop applications development taught us: every app is unique and therefore drive decisions about its architecture.

Many project managers tend to choose mobile as a primary target platform for their apps. How couldn't they since the exponential growth of the mobile app market? The overall revenues in 2018 reached 365.2 billion US dollars [1] and numbers are still breaking the records. A huge number come with great responsibility. Project managers have to take the right decisions depending on the budget and time available. Since there is no silver bullet that can go through all software problems, the best approach must be chosen to fit all the app requirements. Finally, project managers need to choose the technology that will fit their resources.

The remaining of the paper is organized as follows: A presentation of all the mobile development approaches found in section 2. An approach decision metric that aims to help a project manager to make the right decision concerning the approach in section 3. Framework decision metric that aims to bring support in technological decisions in section 4. Finally, a conclusion given in section 5.

II. CROSS-PLATFORM APPRAOCHES

In this section, we explicit the known mobile development approaches which are the native, web, hybrid, cross-platform, modeling, cloud-based and merged approaches with their pros and cons.

A. Native approach

Native mobile apps are built using platform specific tools and languages. Those apps can be executed only on devices with target platform. They also can be downloaded and installed from the store [2].

We show in the following table the advantages and weaknesses of the native approach:

TABLE I. ADVANTAGES AND WEAKNESSES OF NATIVE APPROACH

Advantages	Weaknesses
1) Native look and feel 2) High performance than any other approach 3) Total access to the APIs of the mobile device which allows the use of the various available sensors, access to files, local storage, sending sms, sending calls ...	1) Steep learning curve 2) Effort, cost, time are multiplied by the number of targeted platforms 3) Restrictions and costs associated with developing and deploying to certain platforms. For instance, an Apple developer license and Apple's approval to distribute applications to iTunes Apps Store [3]

B. Web Approach

A web mobile app is built to be executed on a web browser. Such apps are developed using web technologies like HTML, CSS, JavaScipt. In this approach, none of the app components are installed on the mobile device. Rather than that, the app is accessible from a URL and the app data are manipulated by the server [4].

978-1-7281-7116-6/20/$31.00 ©2020 IEEE

The following figure shows the architecture of a web applications [5].

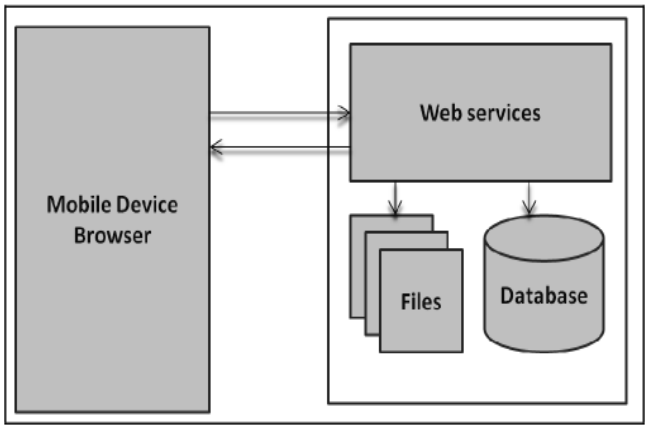

Fig. 1. Web application architecture [5].

We shows in the following table the advantages and weaknesses of the web approach [5]:

TABLE II. ADVANTAGES AND WEAKNESSES OF WEB APPROACH

Advantages	Weaknesses
1) The app does not have to be installed on the device to use it 2) App updates do not depend on the mobile device 3) User interface code is reusable in multiple platforms thanks to the standardization of web browsers	1) Web mobile apps cannot be published in app stores [6] 2) Web apps performance rely on internet connection speed 3) Web apps cannot have access to mobile hardware such as the various available sensors 4) Monetizing web apps is not as simple as native applications

C. *Hybrid approach*

This approach combines between native and web approaches. This approach depends on web technology such as HTML CSS and JavaScript for User Interface and some native wrapped code to access mobile hardware [7].

The figure 2 shows the hybrid app architecture, which is based on Cordova framework [8]:

Fig. 2. Hybrid app architecture.

In the following table we list the advantages and weaknesses of the hybrid approach [5].

TABLE III. ADVANTAGES AND WEAKNESSES OF HYBRID APPROACH

Advantages	Weaknesses
1) The app can be published in an app store 2) User interface can be reused in different platforms 3) The app can access mobile hardware through the hardware abstraction layer 4) The app can rely on mobile computing capabilities in runtime	1) Performances can't match with native apps because they're executed in the platform web engine 2) Lack of native look and feel 3) Hybrid applications suffer from platform specific behavior of JavaScript and threading model incompatibilities with JavaScript.

D. *Cross-platform approach*

The cross-platform approach has many categories that some research papers tried to regroup them in subcategories. We took the categorization from this research paper [2] because we think it is the closest to our categorization. However, there is some point of divergence that we highlight further

1) Compilation approach

The authors divided this sub-approach in two separate groups:

- Cross-compiler approach

- Trans-Compiler approach

The trans-compiler is defined as a tool that translate a high level programming language into another language on the same level. Other works relate such as this research paper [9] where he describes the transpiler as a set of tools that generates the targeted source code after a transpilation process from the original high level programming language.

2) Virtual machine approach

It is a sub-approach of a bigger approach, which is the interpretation approach. However, that interpretation approach encompasses the hybrid approach, which is not our opinion.

This approach consists on running the app on a virtual machine installed on the device. The best example to illustrate this approach is the MobDSL framework. With MobDSL, mobile apps are built using DSL and ran on a dedicated virtual machine [10]

E. *Modeling approach*

This approach is based on abstract models that describes the app functionalities using a textual and/or graphical concrete syntax [11]. Then, with transformations chain, those models are transformed to a targeted platform's source code [12].

F. *Cloud-based approach*

This approach consists on hosting all the app backend on a server so that all the computations and processing are executed on a distant server. Many companies, such as Amazon Web Services, Heroku, offer that kind of service called Baas (Backend as a service). Well-known companies have chosen this approach for their products such as Facebook, Google (Google Drive for instance), Salesforce, etc.

G. Merged approach

As it is shown in another research paper [13] several approaches from those mentioned previously can be combined together so developers can get benefit from the advantages of each one of them.

III. Approach decision metric

When evaluating the mobile apps development approaches, we often come across a comparison of the kind "hybrid vs native" or "low code vs full code". However, a project manager many more constraints than that. Therefore choosing the right approach depends on many more criteria than a pacific checklist.

We propose in our work a decision framework that can help anyone confronted with this situation.

A. Related Works

Haire and al. tried to answer that problematic by setting a checklist to make the right choice [14].

They propose to answer the following questions:

1) Skilled team availability

The approach must be based on your team's skill on the first place or hiring developers nearby. Choosing an appealing technology but not having the human resources can be a huge obstacle for the project success.

2) Platform choice: Walled garden or open ecosystem

The term walled garden is used wisely. It referred to the low-code mobile apps development platforms such as Mendix or Kony. Everything is made available to the developer, from the front-end development with a library of components to the data management. They usually offer some third-party services such as Google Maps API.

The inconvenient of those walled garden is that you are limited to what they propose as development tools. You will never have full control of your app.

Open ecosystems on the other hand are a good solution for those who do not want to be limited by the platform. In fact, they offer you an SDK that can be used without any boundaries.

3) Long-term partner

Choosing an approach will tie you to the development platform for a long time. The reason is the support needed in case of platform bugs occurrence.

4) Design & UX consistency across teams and projects

In some contexts, there is a need of having shared UI library across all targeted platforms (mobile, desktop …). The origin of this problematic is because enterprises want to keep a unique design spec. Therefore, the solution is to choose an approach that is compatible with many frameworks.

5) Ecosystem evolution

At this point, we need to sheds light on the changing trend over the years. The project manager decision must be based on its maturity, then choose the right approach for it.

This checklist can help the developer to know more about his project but authors did not have an objective point of view because he was promoting for Ionic framework.

In the other hand, Todd Anglin proposed his own decision framework [15] where he answered almost the same questions. However, we think that Todd's framework is not complete because it lacks of some approaches we mentioned earlier which are the cloud-based, merged and cross-platform approaches.

B. Decision Framework

To set a correct framework, we propose the following questions about the app:

1) Does it need access to native mobile features?

This question is decisive as its answer could exclude the web approach. However, several frameworks implementing the other approaches provide APIs allowing full access to the mobile native features.

2) Does it need high performance?

The question is important because its answer leaves the decision maker with only one choice, namely the cloud-based approach.

3) Will it be published as free app or under a license?

This question is decisive because its answer may strongly influence the application architecture. For instance, if the decision maker opts for a licensed application on web platform, he must think about implementing a membership subscription for the targeted users. Otherwise, he can rely on the app store (depending on which platform he targets).

This question has a strong correlation with the following one.

4) What are the targeted platforms?

This question is as essential as the previous one (Will it be published as free app or under a license?) as its answer makes it possible from the beginning the native or web approach if the decision maker is targeting a single platform.

5) Does it need support after publishing it?

Thinking about support is important because being linked to a technology for life (at least before thinking of an overhaul) suggests the maturity of the latter. However, this question is not important in the choice of approaches because all the frameworks implementing these approaches offer good support.

6) Does it need a native look and feel?

Native look and feel may influence the user experience (UX) but the answer to this question is not very important since the frameworks implementing the other approaches (except the web and hybrid approaches) offer a UX close to the native approach one.

7) Does it need to consume little power from the phone battery?

Battery drain is not very important because the frameworks implementing all the approaches generate applications that consume battery in moderation. The problem lies in hybrid applications that consume the phone's energy significantly.

8) Does it have to implement an advanced security standard?

Security is well managed by all approaches except the web approach where further development is required. In this case, call to third-party libraries is a necessity

9) Does it need to be scalable?

The scalability of the application primarily affects the project budget. However, the native approach is the only

approach to require a more significant budget than the other approaches because of the lack of components reusability.

Those questions have not the same importance. Therefore, we assign a weight for each one.

On a scale from one to three, here is the weight given to each level of importance:

1: Not so important

2: Important

3: Very important

In order to find the approaches that answer those questions, we choose a framework that implement the hybrid and cross-platform approaches and see if it satisfies the needs described by the question.

For the hybrid approach, we choose Ionic Framework, and for the cross-platform approach, we choose Xamarin.

In the table above, we consider the following:

- Native approach : N

- Web approach : W

- Hybrid approach : H

- Cross-platform approach : CP

- Cloud-Based approach : CB

- Merged approach : M

TABLE IV. WEIGHT ATTRIBUTION AND APPROACHES RESULT TO EACH QUESTION

Question	Weight	N	W	H	CP	CB	M
Does it need access to native mobile features?	2	Yes	No	Yes	Yes	Yes	Yes
Does it need high performance?	2	Yes	No	No	No	Yes	Yes
Will it be published as free app or under a license?	3	Yes	No	Yes	Yes	Yes	Yes
What are the targeted platforms?	3	Yes	No	Yes	Yes	Yes	Yes
Does it need support after publishing it?	2	Yes	Yes	Yes	No	No	No
Does it need a native look and feel?	1	Yes	No	No	Yes	Yes	Yes
Does it need to consume little power from the phone battery?	1	Yes	Yes	No	Yes	Yes	Yes
Does it have to implement an advanced security standard?	2	Yes	No	Yes	Yes	Yes	Yes
Does it need to be scalable?	2	No	Yes	Yes	Yes	Yes	Yes

C. *Decision diagram*

Based on the table 4 where we attributed a weight to each question and distinguish between the approaches who answer those questions and the approaches that do not, we were able to establish the following UML activity diagram where we present the decision schema that helps to determine the best development approach to be taken. We chose the activity diagram where every question is an activity and transitions between the activities represent the answers to those questions. Starting from the diagram entry point, the developer must answer all of the questions so he can reach the final activity where he will find the best approach for his application. The final activities shown in green are the decisions to take. We note that there is no case where an activity may leads to two possible activities. Therefore, answering wisely the questions represented by the activities leads to a unique final decision.

IV. CONCLUSION

This paper presents a framework allowing to choose the best mobile development approach for a mobile project in a given context. This framework is based on answering several questions, which answer of each of those questions influences the framework's final result. Therefore, the framework returns a final decision for each mobile project context.

In case of absence of constraints either in term of cost or human resources, the native approach remains the best solution to opt for because of its advantages in terms of quality, performance and ergonomics.

In our future works, we will propose a modeling approach based solution to generate mobile applications using a textual or graphical modeling language.

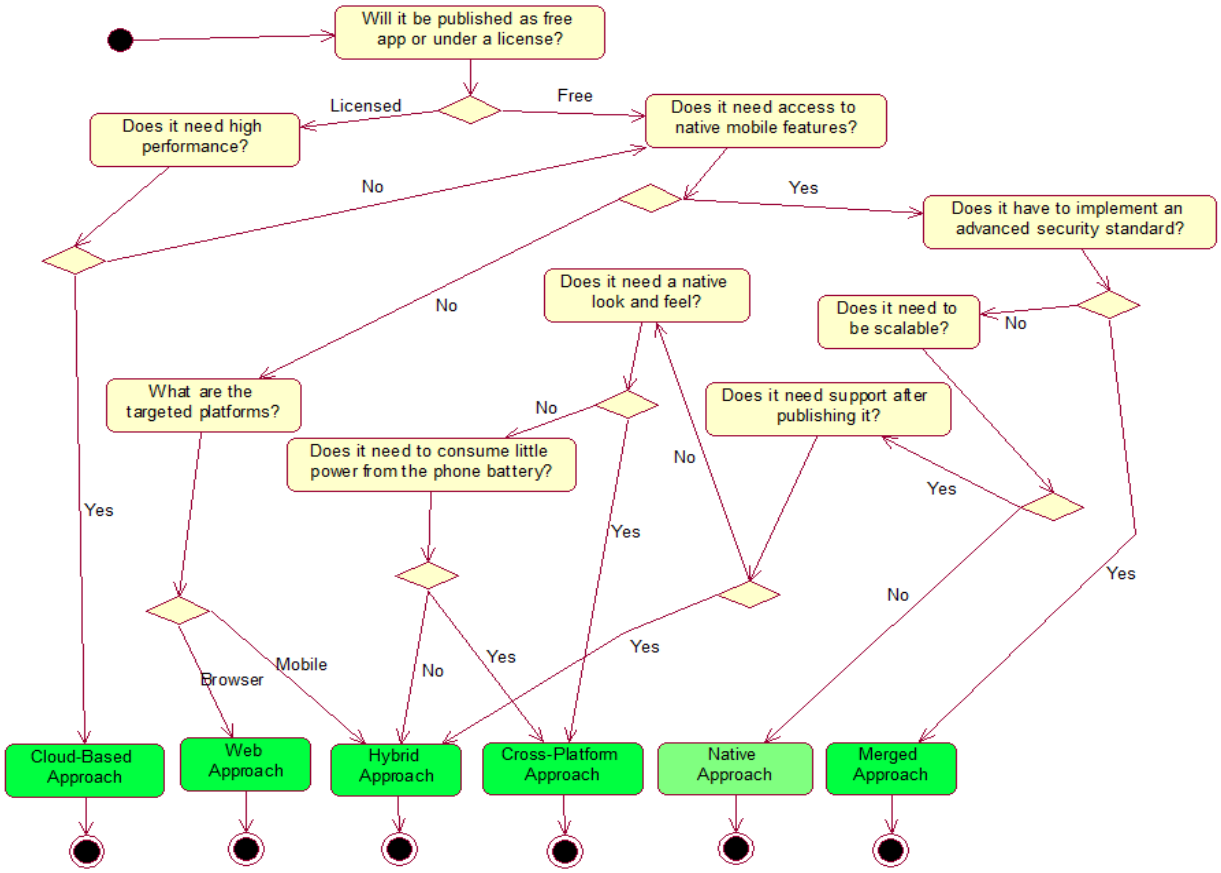

Fig. 3. Decision framework diagram.

REFERENCES

[1] https://www.statista.com/statistics/269025/worldwide-mobile-app-revenue-forecast/. [accessed 02.01.2020].

[2] EL-KASSAS, Wafaa S., ABDULLAH, Bassem A., YOUSEF, Ahmed H., et al. Taxonomy of cross-platform mobile applications development approaches. Ain Shams Engineering Journal, 2017, vol. 8, no 2, p. 163-190.

[3] SMUTNÝ, Pavel. Mobile development tools and cross-platform solutions. In : Proceedings of the 13th International Carpathian Control Conference (ICCC). IEEE, 2012. p. 653-656.

[4] JOBE, William. Native Apps vs. Mobile Web Apps. International Journal of Interactive Mobile Technologies, 2013, vol. 7, no 4.

[5] RAJ, CP Rahul et TOLETY, Seshu Babu. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In : 2012 Annual IEEE India Conference (INDICON). IEEE, 2012. p. 625-629.

[6] FURUSKOG, Martin et WEMYSS, Stuart. Cross-platform development of smartphone applications: An evaluation of React Native. 2016.

[7] PALMIERI, Manuel, SINGH, Inderjeet, et CICCHETTI, Antonio. Comparison of cross-platform mobile development tools. In : 2012 16th International Conference on Intelligence in Next Generation Networks. IEEE, 2012. p. 179-186..

[8] https://www.wavemaker.com/learn/hybrid-mobile/building-hybrid-mobile-apps/ [accessed 02.01.2020].

[9] ANDRÉS, Bastidas F. et PÉREZ, María. Transpiler-based architecture for multi-platform web applications. In : 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM). IEEE, 2017. p. 1-6.

[10] KRAMER, Dean, CLARK, Tony, et OUSSENA, Samia. MobDSL: A Domain Specific Language for multiple mobile platform deployment. In : 2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications. IEEE, 2010. p. 1-7.

[11] BIØRN-HANSEN, Andreas, GRØNLI, Tor-Morten, et GHINEA, Gheorghita. A survey and taxonomy of core concepts and research challenges in cross-platform mobile development. ACM Computing Surveys (CSUR), 2019, vol. 51, no 5, p. 108.

[12] LATIF, Mounaim, LAKHRISSI, Younes, NFAOUI, El Habib, et al. Cross platform approach for mobile application development: A survey. In : 2016 International Conference on Information Technology for Organizations Development (IT4OD). IEEE, 2016. p. 1-5.

[13] ETTIFOURI, El Hassane, RHOUATI, Abdelkader, BERRICH, Jamal, et al. Toward a merged approach for cross-platform applications (web, mobile and desktop). In : Proceedings of the 2017 International Conference on Smart Digital Environment. ACM, 2017. p. 207-213.

[14] https://ionicframework.com/resources/articles/how-to-pick-the-right-mobile-development-approach [accessed 02.01.2020].

[15] Anglin, Todd. (s.d.). telerik: https://www.telerik.com/docs/default-source/whitepapers/choose-right-approach-mobile-app-developmentbb581d10116543e79a9febdb187fd0a3.pdf?sfvrsn=0. [accessed 02.01.2020].