SILESIAN UNIVERSITY OF TECHNOLOGY PUBLISHING HOUSE

SCIENTIFIC PAPERS OF SILESIAN UNIVERSITY OF TECHNOLOGY 2021
ORGANIZATION AND MANAGEMENT SERIES NO. 153

TECHNIQUES OF MOBILE APPLICATION DEVELOPMENT PROCESS

Mateusz DRUĆ[1*], Ireneusz J. JÓŹWIAK[2], Alicja M. JÓŹWIAK[3], Wojciech M. NOWAK[4]

[1] Wroclaw University of Science and Technology, Faculty of Computer Science and Management, Wroclaw; druc.mateusz@gmail.com
[2] Wroclaw University of Science and Technology, Faculty of Computer Science and Management, Wroclaw; ireneusz.jozwiak@pwr.edu.pl, ORCID: 0000-0002-2160-7077
[3] Wroclaw University of Science and Technology, Faculty of Architecture, Wroclaw; ajozwiak07@gmail.com
[4] Wroclaw University of Science and Technology, Faculty of Architecture, Wroclaw; wojciech98nowak@gmail.com
* Correspondence author

Purpose: The purpose of the paper is to present the approaches of developing mobile applications and the technologies used for this process to facilitate the choice of the right programming approach in the initial phase of an IT project.
Design/methodology/approach: Taking the right and adequate decision on choosing the proper approach to software development should be preceded by searching for available solutions and a there reliable comparison. The collected and compared information comes from current scientific sources and official conferences, which is the basis for making the right decisions at the start of a new project.
Findings: Hybrid approach to the manufacture of mobile applications provide applications to run the most popular mobile operating systems. However, the technologies used in this programming approach are not adopted to the extent they have been in the native approach.
Originality/value: The article summarizes and compares the approaches of developing mobile applications. The presented comparison, together with information on the used technologies, helps system architects and programmers in making a decision about the use of a given programming approach based on the given technologies when creating a new project.

Keywords: mobile application, native, hybrid, Android, iOS.

Category of the paper: Research paper.

1. Introduction

Smartphones are multimedia, portable devices that combine the functions of a portable computer and a mobile phone. They are made of many sensors, modules and sensors such as GPS module, gyroscope, proximity sensor or GSM module. In addition to the distant communication, these devices also enable, for example, e-mail client support, installation of thousands of different applications, high-resolution shooting, video recording and multimedia playback (Charlesworth, 2009). With the development of such devices, which are often equipped with touch screens covering a significant part of their surface area, there was a need to develop operating systems adapted to the needs and expectations of users. Many companies have tried to develop operating systems to find applications in tens of millions of devices around the world to meet demand.

Over the last few years, several main competing solutions have appeared on the market. According to the authors of the article entitled "Evaluating cross-platform development approaches for mobile applications" from 2012 (Heitkötter, Hanschke, & Majchrzak, 2012), until some time ago the mobile technology market was dominated by several companies providing mobile operating systems developed by them. They were: Android developed by Google, Symbian provided by Nokia, iOS available on Apple brand devices, Blackberry by RIM and then evolving Windows Phone developed by Microsoft. However, as today's market analyzes show, Google and Apple (Mobile Operating System Market Share Worldwide) (Mobile, 2020) remain the major corporations with the largest share in the mobile operating system market. According to these analyzes (data from April 2020), the Android operating system currently occupies over 70% of the market, while iOS nearly 29%. The remaining mobile operating systems do not exceed 0.2% of the global market share.

2. Native application development

The presence of two competing mobile platforms on the market creates the need to develop mobile applications that will be able to run on each of them. Providing a mobile application supported on only one of the platforms automatically deprives you at least 30% of potential application users from ever using the application. The basic and primary form of developing a mobile application dedicated to a given platform is the native programming approach. It means the use of programming tools, a programming environment and languages dedicated and specific to a given platform, provided by the designers of a given technology. The main disadvantage of this approach is the final development of an application supported by only one of the available mobile operating systems. This is due to the inability to share native code between platforms (Huynh, Ghimire, & Truong, 2017). Therefore, the desire to deliver the application to the widest possible group of potential recipients means the need to develop two separate native applications. Most often, this requires the launch of two development teams with separate skills and specific knowledge of the given platform.

3. Hybrid application development

A relatively new and alternative approach to native application development is a hybrid approach. It is a concept the idea of which is to share one application source code between many development platforms (Que, Guo, & Zhu, 2016). The concept of a hybrid approach to application development does not define the use of specific tools, languages or technologies, but covers all concepts of sharing source code across more than one platform. The hybrid approach is gaining more and more popularity in the mobile industry due to the reduced costs of application development (maintaining one team of programmers instead of two) and often shortened development time. Ultimately, to make a mobile application available to 99% of smartphone users in the hybrid process, fewer specialists who do not need to have the broad knowledge required to develop native applications are needed.

Nevertheless, not all mobile applications that are currently being developed using this approach. In the majority of new applications vendors choose not to adopt this relatively new development approach. For companies that develop new applications, achieving satisfaction by their customers when the last uses their applications, is a priority. The native app development approach is well established with a strong technical background and a broad developer community. It provides a satisfying app user experience. In the process of developing mobile applications, positive feelings of the end user during use of the application are one of the key aspects determining success in the market. In addition to the proper operation of the application and providing the expected functionalities, application performance also determines a positive user experience (Ballard, 2007). Low performance manifested by a drop in the number of displayed frames per second, excessive CPU use (resulting in increased device temperature) or excessive battery consumption, very often causes the user to resign from using the application (Mainkar, 2017). Providing the functionalities needed by the user without ensuring the application performance at a satisfactory level will not bring developers success in the mobile application market.

4. Techniques of mobile application programming

4.1. Preliminary remarks

As presented by the authors of the article "Evaluating Cross-Platform Development Approaches for Mobile Applications" (Heitkötter, Hanschke, & Majchrzak, 2012), currently most of the mobile market is occupied by Google, which has developed the Android system, and Apple, which owns iOS. Other platforms popular in recent years have not survived on the market or have lost some of their shares and currently do not play a significant role in the mobile industry. An example of such a platform is the Windows Phone operating system released by Microsoft (Gajewski) in 2010.

In addition to various platforms for which mobile applications are created, there is also a division into approaches to the development of these applications. This division was created as a result of the development of many mobile software implementation techniques. According to Mahesh Panhale, the author of the book "Beginning Hybrid Mobile Application Development” (Panhale, 2016), we distinguish three approaches in the development of mobile applications: native, web and hybrid.

4.2. Native applications

Native applications are those, which include written for a specific operating system, with the usage of technology strictly indicated by the designers of a given platform applications. In case of native Android applications, it is preferable and most common to use Java and Kotlin, while in case of iOS applications – Swift and Objective-C languages are the most relevant choice (Native vs Hybrid – what type of mobile application should you choose?) (Native vs Hybrid, 2020). The most popular development environment for developing native applications for the Android platform is currently Android Studio (Hagos, 2019), and for the iOS platform Xcode (Knott, 2016). Applications designed in a native approach are not compatible with more than one operating system, which makes it necessary to prepare several applications in order to reach a wider group of potential users. The significant advantages of the native approach in the development of mobile applications include (Liu, 2013):

- high performance of the software being built,
- no restrictions on the implementation side,
- potentially less chance of various errors (the application is dedicated to a given platform),
- direct access to operating the device components (e.g. Bluetooth),
- popularity of the approach and the technologies used (developed community network),
- a very large number of publicly accessible libraries.

In the case of disadvantages of the described approach, the following should be mentioned (Panhale, 2016):

- the need to implement several applications (separate for each platform),
- system maintenance (several applications) requires more work,
- the size of the application in terms of the number of lines of code,
- lower reusability of the code created,
- the need for more knowledge.

4.3. Web applications

The web application is an application that can be displayed in a web browser on a computer station as well as in a web browser installed on a mobile device. It is designed and made in such a way as to display the information correctly on both types of devices and it also gives the impression of using a traditional mobile application on the smartphone (Internet or mobile application?) (Application, 2020). When building web applications, technologies such as HTML and JavaScript are most often used. In order to achieve the impression of using a typical mobile application, there is also a solution consisting of running such an application in a specially adapted application web view (WebView). Then the user installs the application on the device in the traditional way and the application launches a built-in browser inside the given web application running. The effect achieved in this way can be satisfactory in the case of uncomplicated applications. The advantages of using the web-based approach to developing a mobile application are as follows (Hartman, Rokitta, & Peake, 2013):

- a shared code that can be run on any device using a web browser,
- the ability to update the appearance and functionality of the application without the need to update it on the user's side,
- the application code does not take up space on the mobile device,
- the universality of the technologies used,
- relatively low cost and short application development time,
- the ability to install the application in a traditional way or run it from the level of a web browser.

The disadvantages of the web approach to the development of mobile applications are (Hartman, Rokitta, & Peake, 2013):

- Internet connection required for the application to function,
- difficult process of implementing functionalities using access to the hardware components of the device (such as e.g. a camera),
- relatively low application performance,
- worse adjustment of the user interface to the platform on which the application is run than in the case of a native application,
- the inability to write native code.

4.4. Hybrid applications

A hybrid application is defined as an application resulting from the fusion of two solutions. These solutions are native and web approaches. Thanks to this combination, it is possible to obtain a product that combines the advantages of both aforementioned solutions (Lehman) (Lehman, 2020). The tools that are currently most often used to implement this type of application are:

- Flutter (using the Dart programming language),
- React Native (using the JavaScript programming language),
- Xamarin (using the C # programming language).

Knowledge of one of the mentioned technologies allows you to build an application supported by both platforms (Android and iOS). Undoubtedly, this is one of the advantages of this programming approach. The other advantages are:

1) the ability to write native application components,
2) access to the components of the mobile device as in the case of native applications,
3) the possibility of implementing separate user interfaces for both platforms,
4) higher performance of the implemented application than in the case of the web approach,
5) knowledge of one technology enables the implementation of applications for both platforms,
6) lower cost of development and maintenance of the application than in the case of the native approach,
7) the growing popularity of the hybrid approach, its rapid development and continuous improvement.

However, this approach is not just about advantages. By writing applications using hybrid approach, some elements still need to be implemented separately for each platform. This involves the need to handle the components of a mobile device (such as an accelerometer) differently for different platforms. Google provides different guidelines and requires a different implementation of such a module comparing to Apple. According to the author of the book "Beginning Hybrid Mobile Application Development" by Mahesh Panhale (Panhale, 2016), currently approximately 80% of the hybrid application code is shared by both platforms, while 20% is still native code, written for each platform separately. Other disadvantages of implementing a mobile application in the hybrid approach are:

- the need for frequent adaptation of changes in the design as a result of hybrid technologies' dynamic development and their frequent updates,
- fewer publicly available libraries than in the native approach.

Among the described approaches to the implementation of mobile applications, technologies used in the native and hybrid programming approach will be discussed and compared. Web applications were not considered due to the above-described disadvantages of the web approach to mobile application development.

5. Technologies used in the native programming approach

Android is a comprehensive, completely open source platform for mobile devices. It includes the entire technology stack, from low-level Linux modules to native libraries, from the application framework to complete mobile applications. This platform was launched by Google with the goal of accelerating mobile innovation and offering consumers richer, cheaper and better mobile features. Android is licensed (Apache/MIT) so that it can be freely extended and implemented for personal use, making the source code of the system available to developers (Gargenta, 2011).

Dedicated tools are used when implementing a native application on the Android platform. Google supports and develops the development environment called Android Studio released under the "Apache 2.0" license which is also free for commercial use (Studio, 2017).

Google supports two programming languages in the process of implementing native mobile applications on the Android platform. These languages are Java and Kotlin.

The leading programming language was originally Java, but from 2019 the main language promoted by Google is Kotlin.

The iOS operating system (iOS, 2020) has been developed for the group of Apple mobile devices such as iPhone, iPod touch and iPad, published by Apple. According to its architects, the key to the success of iOS was simplicity and intuitiveness. This system is very popular on the mobile market, but its use is much lower than in the case of Android, due to the very limited number of devices supporting this solution (iOS).

Apple, like Google, supports two programming languages in the process of implementing native applications for iOS. These languages are: Objective-C and Swift.

The official Apple-backed development environment for developing native iOS applications is xCode (Tiano, 2016). The architects of the xCode environment claim that the application was designed to speed up and streamline the work of programmers and ensure a high level of comfort in using this solution.

6. Technologies used in the hybrid programming approach

Currently, one of the most widely used technologies in the hybrid programming approach are (Fayzullaev, 2018): React Native and Xamarin and Flutter.

React Native is a technology based on a library developed by Facebook, used to build user interfaces called React. However, applications developed in React Native technology are not dedicated to web browsers (as is the case with React technology), but to mobile devices. React Native is a JavaScript framework that allows you to compile a written application to native code. This technology allows you to build applications in a similar way as in the case of React, but allows you to display them on mobile devices in such a way that they appear native. The JavaScript programming language and its JSX extension are used to write applications in React Native, which technically resembles the XML markup language and is used for simple composing of user interface elements (Eisenman, 2015). The React Native technology also allows you to write application fragments in native languages for the Android and iOS platforms. As a result, critical application areas, exposed to high loads and performance degradation, can be written with native code and thus match the performance of native applications.

Figure 1 shows a diagram of the React Native technology architecture. As shown in the Figure 1, the React Native architecture consists of several main components. Code written in JavaScript language is properly interpreted by the React Native development platform and then separate projects are created for each of the target platforms. The result of the standard compiler work on the Android platform is the native code of the application intended for the Android platform, while the standard iOS compiler generates a native application intended for the iOS platform as a result. Another of the technologies mentioned is Xamarin. It is a development platform that allows you to write cross-platform (Android, iOS, Windows Phone) code using the C# programming language and the XAML user interface description language (Co to jest środowisko Xamarin?) (2020). A dedicated development environment for developing applications using Xamarin is Visual Studio and Xamarin Studio (in the case of the macOS operating system). The code written with this technology, as in the previous case, is compiled to native code for each of the supported platforms. Similarly, when writing some functionalities, it is also necessary to write native code fragments for each of the mobile operating systems separately. Figure 2 shows the architecture of the discussed technology for the case of compiling the application for the Android platform.

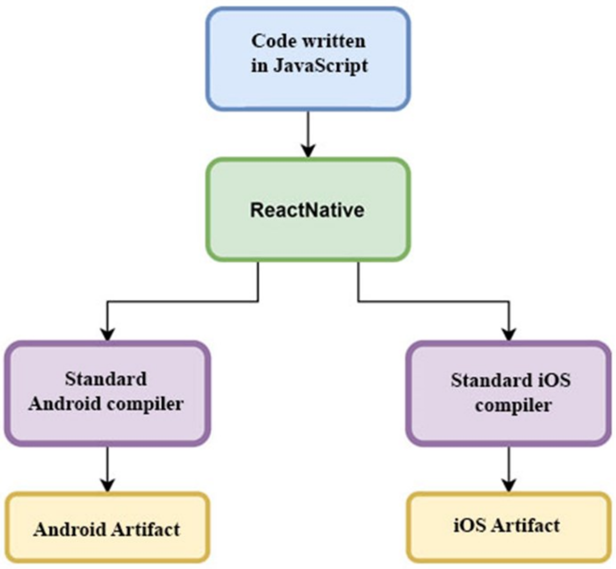

Figure 1. React Native architecture diagram (Frachet, 2020).

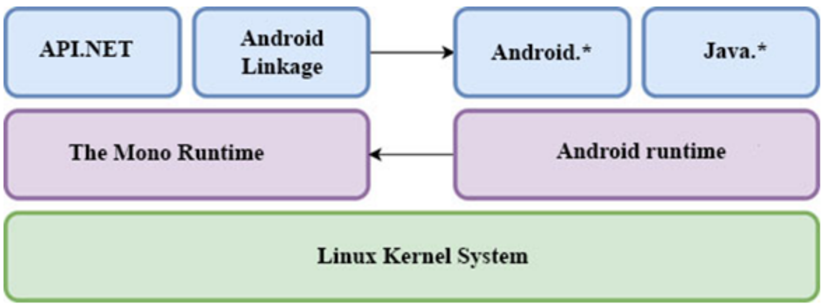

Figure 2. Xamarin technology architecture diagram when compiled for the Android platform (Flutter vs Xamarin vs Native trends, 2020).

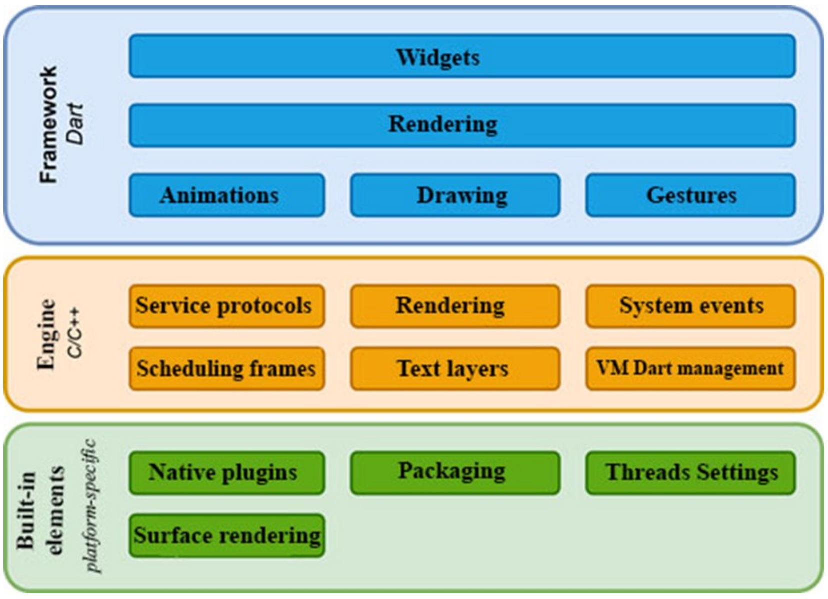

Figure 3. Diagram of the Flutter technology architecture (Flutter internals, 2019).

In the case of compiling a Xamarin application (in the Android version), it is done from C# to the intermediate language and then the code is compiled into a native assembly immediately before the execution of the given code fragment (just-in-time compilation). These applications run in the mono runtime using the Android Runtime virtual machine. Each of these runtime environments runs on top of the Linux kernel.

The third of the listed technologies is Flutter. It is a mobile application development toolkit (mobile SDK), built and provided by Google. Flutter is a free tool, also provided for commercial usage. It is currently the newest available hybrid technologies, which enjoys great and constantly growing interest of programmers. The user interface in Flutter is constructed of elements called widgets, which can adapt to each one of the supported platforms and thus maintain a fully native appearance without the need to write several implementations. Flutter designers strive to ensure that once written code can be run on any platform and operating system. Currently, the application written in Flutter can work with platforms such as: Android, iOS and ChromeOS. However, the system designers note that in the near future, the application will also be available in the web version as well as on any of the popular operating systems. Flutter is a complete tool that provides programmers with a rendering engine, ready-made user interface components as well as development platform for testing applications (Windmill, 2020).

Figure 3 shows a diagram of the Flutter technology architecture. When writing an application in Flutter using the Dart language the programmer moves on the level of the programming platform marked in the diagram with a blue color. This level interacts with the Flutter engine through an abstraction layer called Window. In turn, this layer provides a number of interfaces for communication with the device (Flutter internals) (2019). The abstraction layer is also used to notify the programming platform (framework) when:

- a significant action occurs at the device level (e.g. device orientation change, memory problem, settings change),
- there is an action on the part of the user (e.g. a gesture on the screen),
- Flutter engine is ready to render the new frame.

7. Conclusion

The previous chapter described three popular technologies currently used for application development in a hybrid approach. On the ground of the information collected and the data obtained from the Google Trends website, it is possible to decide on the technology that should be used to implement the application in a hybrid approach. Over the last year, a technology called Flutter (Flutter vs Xamarin vs Native trends, 2020) has been the most popular. The technology provided by Google is relatively new and has recently become more and more popular.

The last choice that had to be made was the choice of language for both of the most popular technologies described. In the case of Flutter, the only supported language is the Dart programming language, the use of which is inevitable when it comes to implementing the application in a hybrid approach. In the case of Android, there are two programming languages, which are mainly used today. These are Java and Kotlin. Due to the fact that during the Google I/O 2019 conference Kotlin was officially the main language for the development of Android technology (Lardinois, 2019), it is a natural choice to use this language when implementing an application in a native approach.

References

1. A. Studio (2017). *Android Studio. The Official IDE for Android.*
2. *Aplikacja internetowa czy mobilna*? https://mansfeld.pl/programowanie/aplikacja-internetowa-vs-mobilna/, 08.04.2020.
3. Ballard, B. (2007). *Designing the mobile user experience*. John Wiley & Sons.
4. Charlesworth, A. (2009). *The ascent of smartphone*. IET.
5. *Co to jest środowisko Xamarin?* https://docs.microsoft.com/pl-pl/xamarin/get-started/what-is-xamarin, 22.04.2020.
6. Eisenman, B. (2015). *Learning react native: Building native mobile apps with JavaScript.* O'Reilly Media, Inc.
7. Fayzullaev, J. (2018). *Native-like cross-platform mobile development: Multi-os engine & kotlin native vs flutter.* Kaakkois-Suomen ammattikorkeakoulu.
8. *Flutter internals.* https://www.didierboelens.com/2019/09/flutter-internals/, 22.04.2020.
9. *Flutter vs Xamarin vs Native trends*, https://trends.google.com/trends/explore?geo=US&q=react %20native,flutter,Xamarin, 27.04.2020.
10. Frachet, M. (2020). *Understanding the React Native bridge concept*, https://hackernoon.com/understanding-react-native-bridge-concept-e9526066ddb8, 12.05.2020.
11. Gajewski, M. (2017). *Windows Phone umarł. Tym razem formalnie i na dobre*, https://www.spidersweb.pl/2017/07/microsoft-windows-phone.html, 10.05.2020.
12. Gargenta, M. (2011). *Learning android.* O'Reilly Media, Inc.
13. Hagos, T. (2019). *Android Studio IDE Quick Reference: A Pocket Guide to Android Studio Development.* Apress.
14. Hartman, R., Rokitta, C., Peake, D. (2013). *Oracle Application Express for Mobile Web Applications.* Springer.
15. Heitkötter, H., Hanschke, S., Majchrzak, T.A. (2012). *Evaluating cross-platform development approaches for mobile applications.* International Conference on Web Information Systems and Technologies.
16. Huynh, M.Q., Ghimire, P., Truong, D. (2017). Hybrid app approach: could it mark the end of native app domination? *Issues in Informing Science and Information Technology, vol. 14*, pp. 049-065.
17. iOS, https://teamquest.pl/baza-wiedzy/systemy-operacyjne-systemy-mobilne/ios/62.html, 16.04.2020.
18. Knott, M. (2016). *Beginning Xcode*: *Swift 3 Edition.* Apress.
19. Lardinois, F. (2020). *Kotlin is now Google's preferred language for Android app development.* https://techcrunch.com/2019/05/07/kotlin-is-now-googles-preferred-language-for-android-app-development/, 04.05.2020.

20. Lehman, M. (2020). *Aplikacja mobilna – Hybryda, Progresywna, czy Natywna?* https://www.gmi.pl/blog/aplikacja-mobilna-hybryda-natywna/, 03.05.2020.
21. Liu, F. (2013). *Android native development kit cookbook.* Packt Publishing Ltd.
22. Mainkar, P. (2017). *Expert Android Programming: Master skills to build enterprise grade Android applications.* Packt Publishing Ltd.
23. *Mobile Operating System Market Share Worldwide*. Retrived from: https://gs.statcounter.com/os-market-share/mobile/worldwide, 12.04.2020.
24. *Native vs Hybrid – jaki rodzaj aplikacji mobilnej wybrać*? https://appchance.com/pl/blog/native-vs-hybrid-jaki-rodzaj-aplikacji-mobilnej-wybrac, 24.04.2020.
25. Panhale, M. (2016). *Beginning hybrid mobile application development*. Heiderberg: Springer.
26. Que, P., Guo, X., Zhu, M. (2016). *A comprehensive comparison between hybrid and native app paradigms.* 8th International Conference on Computational Intelligence and Communication Networks (CICN).
27. Tiano, J. (2016). *Learning Xcode 8.* Packt Publishing Ltd.
28. Windmill, E. (2020). *Flutter in Action.* Manning Publications.