BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 28 No. 4. 2023

UDC 004.45
DOI: 10.62660/2306-4412.4.2023.19-27

Comparative analysis of frameworks for mobile application development: Native, hybrid, or cross-platform solutions

Oleksii Zarichuk*
Computer and Information Systems Manager
LLC "Fides"
02000, 11a E. Sverstyuk Str., Kyiv, Ukraine
https://orcid.org/0009-0009-0771-8465

Abstract. In the modern digital world, mobile application development is a key area of information technology, and choosing the optimal approach to their development is crucial for effective market implementation. The purpose of this study is to conduct a comparative analysis of various frameworks for mobile application development: native, hybrid, and cross-platform solutions. To achieve this purpose, methods of analysis, synthesis, and comparison were used. Characteristics of different frameworks for mobile application development, including their performance, cost, and access to device capabilities, were analysed. The study disclosed that native frameworks are distinguished or their highest performance and the ability to provide a maximally native look and functionality of the application. However, this approach has limitations as it requires separate development for each platform, leading to increased time and resource costs. Hybrid solutions proved to be cost-effective, allowing the use of a single codebase for creating applications for different platforms. This simplifies the development and maintenance process. Nevertheless, hybrid applications may have limited performance due to the use of WebView for interface display and restricted access to device capabilities. Cross-platform frameworks, on the other hand, provide a balance between performance and resource efficiency. They allow using a single codebase for creating applications for multiple platforms and can achieve satisfactory performance. However, they may have limited access to certain device capabilities and application appearance. This study makes a new contribution to science by providing a detailed comparative analysis of different approaches to mobile application development and the frameworks used for their creation. The results obtained can be used to make informed decisions regarding the choice of a framework for mobile application development

Keywords: Java; software development; performance; native look; codebase

Article's History: Received: 29.08.2023; Revised: 03.11.2023; Accepted: 18.12.2023

INTRODUCTION

In the modern information society, mobile application development has become one of the most important and relevant fields of software engineering. With each passing year, the number of mobile devices increases, along with the demand for high-quality and effective applications. Choosing the right approach to mobile application development is a critically important task for developers and businesses. The issue under study involves determining which type of framework for mobile application development is the most effective and meets the needs of the modern software market. This issue becomes increasingly relevant due to the growing number of frameworks and approaches available to developers, expanding the spectrum of possibilities for mobile applications.

In several studies, various aspects of mobile application development have been analysed and compared, including the choice between native and hybrid approaches, defining best practises in cross-platform development, and exploring methods for choosing the

Suggested Citation:
Zarichuk, O. (2023). Comparative analysis of frameworks for mobile application development: Native, hybrid, or cross-platform solutions. *Bulletin of Cherkasy State Technological University,* 28(4), 19-27. doi: 10.62660/2306-4412.4.2023.19-27.

*Corresponding author

optimal platform for cross-platform development. In their study, H.O. Kozub and Yu.H. Kozub (2022) explored the development of cross-platform mobile applications using Kotlin Multiplatform and Jetpack Compose for different operating systems, including Android, Windows, Linux, and macOS. They express the importance of these tools and the potential for reducing development time and preventing errors through their use. In addition, the authors emphasise the necessity of employing a declarative approach for creating user interfaces.

O. Karatanov *et al.* (2021) conducted a comparative analysis of two important frameworks for modular testing in the Java programming language – JUnit and TestNG. They identified the main functions and advantages of both frameworks, noting that TestNG has more extensive functionality, making it more flexible for complex projects. In the study by M. Singh and G. Shobha (2021), the relevance of cross-platform mobile application development was highlighted. Various frameworks for this purpose were discussed, and a comparative analysis of their functionality was conducted. The main conclusions of the study emphasise the need for a well-founded framework choice based on the specific project requirements, and there is no universal framework for all situations.

In the paper by T. Zohud and S. Zein (2021), the authors investigate the approaches of development teams to cross-platform mobile application development. They use qualitative research, including the case study method, interviews, and group discussions, to gather information from four software development companies in Palestine. The results of the study show that developer experience is a key factor in the development process. The React Native framework is recognised as promising and dominant. A. Kaczmarczyk *et al.* (2022) considered a comparison between native and hybrid mobile applications for the Android operating system, with a focus on using BLE (Bluetooth Low Energy) and Wi-Fi (Wireless Fidelity). It was noted that mobile applications are becoming increasingly popular in the era of smartphones and tablets. The authors conducted a comparative analysis aimed at determining the efficiency of data processing in both technologies. The study by P. Lachgar *et al.* (2022) solves the problem of developing cross-platform mobile applications due to the growing popularity of mobile devices. The authors propose a new framework for choosing the optimal platform for cross-platform development, using multi-criteria decision-making methods such as the Analytic Hierarchy Process (AHP) and the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS).

After analysing studies in the field of mobile application development, the need for research comparing different frameworks, including native, hybrid, and cross-platform solutions, was identified. The purpose of the study was to analyse approaches to mobile application development and compare them, considering their limitations and advantages. The most popular frameworks and tools for each approach were also considered to provide objective information that would help developers and organisations make informed choices in mobile application development.

MATERIALS AND METHODS

This study conducted a comparative analysis of frameworks for developing mobile applications using native, hybrid, and cross-platform solutions. To achieve the purpose, the following methods were used: analysis, synthesis, and comparison. The use of these methods contributed to clarifying the advantages and limitations of each of the examined approaches in mobile application development.

As part of the study, two main mobile platforms were identified for analysis: Android and iOS, which are leaders in the modern mobile device market. Three main approaches to mobile application development were considered during the analysis: native, hybrid, and cross-platform. Specialised integrated development environments (IDEs) were used for native application investigation, such as Android Studio for Android and Xcode for iOS. For the study of hybrid applications, frameworks based on web technologies were used, including Apache Cordova, Ionic, and React Native. Cross-platform application frameworks such as Flutter, Xamarin, and Appcelerator Titanium were reviewed. One of the main criteria for selecting frameworks was their popularity and relevance in the mobile application development market.

The study was based on an analysis of open scientific literature that was available on the Internet. Various works, studies, and publications that detailed different approaches and frameworks for mobile application development were used (Biørn-Hansen *et al.*, 2020; Zohud & Zein, 2021; Lachgar *et al.*, 2022). This literature provided the foundation for the analysis and comparison of the selected approaches. The analysis focused on several key aspects, including development productivity, the quality and performance of created applications, development cost, support for different mobile platforms, extensibility, and other important factors. For each of the examined approaches (native, hybrid, and cross-platform solutions), their advantages and limitations were identified and analysed, contributing to the creation of an objective comparative analysis. The synthesis method involved gathering information about each of the reviewed frameworks, including programming languages used, development tools availability, platform support, application quality, extensibility, and other characteristics. This method provided a comprehensive understanding of each approach.

Comparison was conducted by juxtaposing the obtained results to determine the advantages and disadvantages of each approach. Factors considered in the comparative analysis included development productivity, development cost, application quality and performance, support for different mobile platforms, memory

consumption, extensibility, and other parameters. Based on this comparative analysis, the strengths and weaknesses of each of the examined approaches to mobile application development were identified. The choice of analysis, synthesis, and comparison methods for this research is justified by their ability to provide a comprehensive and objective analysis of different approaches to mobile application development. The analysis method allowed for a deeper examination of each approach, identifying its features, technical parameters, and other characteristics. Synthesis combined the gathered information to create a comprehensive understanding of each approach, which was useful for further comparison. Comparison identified the advantages and disadvantages of different approaches, considering various aspects such as productivity, development cost, application quality and performance, support for different mobile platforms, and other factors.

RESULTS

Native frameworks for mobile development. Native frameworks for mobile platforms are specially designed tools that allow developers to create applications optimised for a specific platform or operating system. These frameworks provide access to native capabilities and features of the platform, resulting in the development of high-performance and efficient software products. The key feature of these frameworks is their focus on development for a specific platform, such as iOS or Android. This means that developers have the opportunity to fully leverage all the capabilities of the given platform to create applications that work optimally and reliably. Typically, for iOS, this involves the Swift programming language and the Xcode development tool, while for Android, it includes the Kotlin or Java programming language and the Android Studio development tool.

Xcode is an integrated development environment (IDE) designed for creating software for iOS and macOS platforms. It is specifically tailored for programming languages recommended by Apple, including C, C++, Objective-C, Swift, Java, AppleScript, Python, and Ruby. Xcode offers advanced capabilities and tools that facilitate the development process, including code refactoring features (Tkachuk & Bulakh, 2022). The modern Swift programming language was developed by Apple for iOS, macOS, watchOS, and tvOS platforms. It aims to provide safety, performance, and code expressiveness, offering a more convenient and efficient alternative to the Objective-C language (Fojtik, 2019). Swift comes integrated with the Xcode development environment in the macOS operating system. This integration allows developers to develop, test, and debug their Swift programmes in a unified environment. Xcode provides a range of tools and features that ease the development process, such as a visual interface builder, debugger, and simulator for testing applications on various devices (Ziyodullayevich *et al.,* 2019).

The advantage of using Xcode and Swift to develop iOS applications is the ability to use native features and functions of Apple devices. Native development allows creating applications optimised for specific iOS hardware and software, resulting in improved performance and user experience. However, it requires developers to have knowledge of different programming languages and development environments (Biørn-Hansen *et al.,* 2020). The primary programming languages for Android application development include Java and Kotlin. Java is a traditional programming language for Android, while Kotlin is a modern alternative that provides efficiency and convenience for developers. Android Studio serves as an integrated development environment (IDE) specifically designed for creating Android applications. It supports both major programming languages, Java and Kotlin, and provides developers with the necessary tools for application development and debugging. In addition, for scenarios requiring maximum performance, the Native Development Kit (NDK) is available, allowing developers to use native code, such as C and C++, in their applications. The Android Software Development Kit (SDK) offers tools and resources for application development, testing, and documentation for Android. It also includes emulators for testing applications on various devices and Android versions (Sun *et al.,* 2021). All these components of the Android application development infrastructure enable developers to create efficient and reliable applications for different devices, working at an optimal level of performance and utilising the powerful capabilities of the Android platform.

The development of native mobile applications has several key features that distinguish it from other development approaches. One of the main advantages is the ability to harness the full potential of the underlying operating system and device hardware. Native applications have direct access to device-specific features and APIs (application programming interfaces), enabling developers to create optimised and productive applications. This level of control allows the development of multifunctional and engaging user interfaces with smooth animations, fast responsiveness, and access to advanced functionalities (Dittrich *et al.,* 2023). Another key feature of native application development is the ability to follow specific design recommendations for user interface components. Native applications can provide a consistent user experience by adhering to design principles and templates of the target operating system (Thamutharam *et al.,* 2021). This ensures that the application looks and feels like it is native to the platform, improving user experience and satisfaction.

Native application development also offers better integration with the device ecosystem. Developers can easily access device capabilities such as the camera, GPS, accelerometer, and push notifications (Raeesi *et al.,* 2022). This enables the creation of applications that can fully utilise the device's potential, providing features like location-based services, augmented reality, and

real-time functionality. In addition, compared to other development approaches, native applications typically offer better performance. Since they are built using native programming languages and tools, they can operate more efficiently. However, native application development also has limitations and challenges. One of the main drawbacks is the need to develop separate codebases for each target platform (Masaad Alsaid *et al.,* 2021). This can increase development time and costs. Moreover, updates and bug fixes may require separate deployments for each platform, leading to increased maintenance efforts.

Hybrid mobile app development frameworks. Hybrid development of mobile applications allows developers to create mobile applications using web development technologies, such as HTML (Hybrid Mobile App Development Frameworks), CSS (Cascading Style Sheets), and JavaScript. This method combines the advantages of other approaches and offers a compromise between native applications and web applications, allowing the production of applications that simultaneously provide high functionality for a specific platform and compatibility with different platforms (Wu *et al.,* 2022).

One of the most popular tools for hybrid development is Apache Cordova. It initialises a native application using WebView, an embedded web browser. It acts as a bridge between native code and application web components, enabling developers to write business logic in JavaScript and create user interfaces using HTML and CSS. This approach allows the development of applications that can work on various platforms, including Android and iOS, using a single codebase. Hybrid application development frameworks provide a wide range of features and capabilities to simplify the development process. These frameworks often include components and libraries that allow developers to create visually appealing and fast user interfaces (Singh & Shobha, 2021). They also provide access to device features and APIs through plugins, allowing developers to use native functionality in their hybrid applications.

In addition to Apache Cordova, other popular hybrid frameworks include Ionic (which uses Angular) and React Native (based on React). Each of these frameworks has its unique features and applications (Table 1), allowing developers to choose the one that best suits their needs.

Table 1. Comparative characteristics of popular hybrid frameworks

Framework	Description	Programming language	Main features
Apache Cordova	Uses web technologies to develop mobile applications.	HTML, CSS, JavaScript	Supports many plugins for device functionality.
Ionic	A hybrid framework based on Angular. Designed specifically for creating beautiful and functional mobile applications	HTML, CSS, JavaScript (with Angular)	Pre-built components and tools for the user interface.
React Native	Allows using React to create mobile applications with the appearance and functionality of native applications.	JavaScript (with React)	Ability to reuse code between platforms and high performance.

Source: compiled by the author

One of the main advantages of hybrid development is the ability to leverage existing skills and resources of web developers. Developers proficient in web technologies can easily transition to hybrid development as they can use their knowledge of HTML, CSS, and JavaScript. This can lead to shortened development timelines and reduced costs compared to developing separate native applications for each platform. However, hybrid development also has its limitations. Since hybrid applications rely on WebView for displaying the user interface, they may not achieve the same level of performance as native applications (Hu *et al.,* 2023). In addition, using plugins may be necessary to access certain device features and APIs, introducing additional complexity and potential compatibility issues. Overall, hybrid mobile application development offers a compromise between native and web applications. It allows developers to create applications for different platforms using web technologies while providing access to native device features. Apache Cordova is a popular tool for hybrid development, enabling the use of HTML, CSS, and JavaScript to create applications that can work on various platforms. Although hybrid applications may not have the same performance as native applications, they offer advantages in terms of development efficiency and cost-effectiveness.

Efficient Cross-Platform Software Development Solutions. Cross-platform mobile application development is a modern approach in the field of mobile device software. It involves creating applications that can run on different platforms, such as Android and iOS, using a unified codebase and other shared resources. Cross-platform development has gained popularity due to its ability to effectively utilise shared code for creating applications on different platforms. This allows developers to save time and resources by avoiding the need to maintain separate code for each platform.

The main advantage of cross-platform development is the reduction of costs for application development and maintenance since shared code can be used across all platforms. In addition, developers can use a single programming language to create application

functionality. Cross-platform frameworks like Flutter, Xamarin, and others (Table 2) provide tools for developing mobile applications with appearances and behaviours similar to native applications on each platform (Martínez, 2019). This ensures high quality and consistent functionality on different operating systems.

Table 2. Overview of popular cross-platform frameworks

Framework	Programming language	Description
Flutter	Dart	A framework from Google for creating beautiful mobile applications with a single code base and using its own rendering engine.
Xamarin	C#	A framework from Microsoft that allows developers to use the C# programming language to create mobile applications for Android and iOS.
Appcelerator Titanium	JavaScript	A framework that uses JavaScript to develop cross-platform mobile applications and provides access to native features.

Source: compiled by the author

Cross-platform frameworks for mobile application development, while offering advantages such as a unified codebase and reduced development costs, also come with a set of drawbacks. Firstly, one of the main drawbacks is the limited access to native functions and platform APIs (Application Programming Interface). Cross-platform frameworks attempt to abstract the device's functionality, but sometimes they may not fully replicate all the capabilities available on a specific platform. The second disadvantage is performance and speed. Mobile applications developed using cross-platform frameworks may run slower and require more resources compared to native applications. In addition, delays in updates may lead to difficulties in utilising new features. Finally, cross-platform frameworks may not be as well-adapted to the specific look and behaviour of certain platforms, which may necessitate additional customisation and adaptation. Therefore, cross-platform frameworks, while allowing for time and resource savings in mobile application development, have limitations and drawbacks that need to be considered when choosing a development approach.

Comparison of the Effectiveness of Native, Hybrid, and Cross-Platform Solutions. In the modern context of mobile application development, choosing the optimal approach is crucial. Developers face different capabilities and limitations when choosing between native, hybrid, and cross-platform solutions. Table 3 provides a comparative analysis of these development approaches.

Table 3. Comparative analysis of native, hybrid, and cross-platform mobile application development solutions

Parameter	Native applications	Hybrid applications	Cross-platform applications
Performance and speed	High performance, optimised speed using native components and programming languages (Java/Kotlin for Android, Swift/Objective-C for iOS).	Moderate performance and speed due to using WebView for UI display and some abstraction.	Moderate performance but can be improved with specialised cross-platform frameworks (e.g., Flutter).
Cost and development time	High costs and longer development time due to the need to create separate code for each platform.	Lower costs and faster development by using a single codebase for both platforms. Some applications may require adjustments for a specific platform.	Lower costs and less time, but development may take slightly longer compared to hybrid solutions due to choosing the right cross-platform framework and team training.
Access to device capabilities	Full access to all device capabilities and native APIs.	Limited access, requires the use of plugins to access specific device functions.	Average access; cross-platform frameworks provide APIs for simplified access to device capabilities but may not always have a full range of features.
Visual appearance	Native appearance on each platform, ensuring high-quality and polished design.	Less native appearance; the interface may be less attractive and less aligned with the design standards of each platform.	Typically less native appearance, but the ability to use specialised frameworks to enhance design.
Platform support	Platform-specific (Android and iOS).	Both platforms (Android and iOS) from a single codebase.	Both platforms (Android and iOS) from a single codebase.
Updates and support	Separate for each platform, separate updates and maintenance.	Updates and maintenance from a single codebase, but there may be a need for updates for each OS (Operating System) separately to support new platform features.	Updates and maintenance from a single codebase, the possibility of additional work to support new features.

Source: compiled by the author

Native applications are characterised by high performance and speed due to the use of native components and programming languages for each platform separately. For developing native applications on the Android platform, programming languages such as Java or Kotlin are used, while on the iOS platform, Swift or Objective-C are utilised. This approach provides full access to all device capabilities and native APIs, allowing the creation of applications with excellent performance and speed. However, it requires more resources and development time due to the need for separate code for each platform, leading to increased costs and project duration. On the other hand, hybrid applications allow cost and development time reduction by using a single codebase for both platforms. They often rely on web technologies such as HTML, CSS, and JavaScript, simplifying development for web-experienced developers. One popular platform for hybrid development is Apache Cordova, which initialises a native application using WebView, an embedded web browser. Hybrid applications may exhibit lower performance due to the use of WebView and limited access to device capabilities. Plugins can be used to access specific device functions, such as the camera or geolocation.

Cross-platform applications combine the advantages of both previous approaches, enabling the use of a single codebase for both platforms. With the right choice of a cross-platform framework, satisfactory performance can be achieved. One such framework is Flutter, developed by Google, which uses the Dart programming language and its own engine for UI rendering. Another popular option is Xamarin, a framework from Microsoft that allows the use of the C# programming language. Cross-platform applications may also use specialised frameworks to enhance the design and appearance of applications. However, they may look less native and have limited access to platform-specific features. The choice of a development approach should be well-founded and depends on the specific project needs, budget constraints, resources, and the importance of performance, speed, and native application appearance.

DISCUSSION

As a result of the study of three main approaches to mobile application development, namely native, hybrid, and cross-platform solutions, a detailed analysis of each of these approaches was conducted, considering the use of different frameworks. The overall analysis of the findings indicates that the choice of a specific approach to mobile application development should be based on the specific requirements of the project, budget constraints, and resources, as well as the importance of performance, speed, and the native look of the application. Adequate evaluation of these factors before choosing an approach is a crucial stage in mobile application development.

The study by S.R. Uplenchwar *et al.* (2022) focuses on exploring Flutter as a cross-platform framework for mobile app development, the Dart programming language, and Firebase technology. The author notes that the development of wireless technologies and mobile devices has a significant impact on everyday life. Consequently, many aspects of life become digital, and to reduce manual work, more tasks are performed using mobile applications. Flutter, as mentioned by the author, is a popular User Interface (UI) framework for mobile application development from Google. It provides a set of user interface elements such as sliders, buttons, and text fields. Developers building mobile applications using Flutter use the Dart programming language. The author also emphasises Firebase technology, which provides tools for tracking analytics, app crash reports, and conducting marketing experiments. This article explores the possibility of using Flutter, Dart, and Firebase for mobile application development to reduce time and resource costs and to ensure consistent functionality on both Android and iOS platforms. The general conclusion from the study by S.R. Uplenchwar *et al.* (2022), which can be agreed upon, is that Flutter, Dart, and Firebase are powerful tools for mobile application development. Flutter allows the creation of beautiful and functional applications with a single codebase for both platforms. Dart is a programming language specifically designed for working with mobile applications and has its package manager for convenient dependency management. Firebase provides extensive capabilities for analysing and enhancing applications. Together, these technologies can significantly simplify and improve the process of mobile application development.

A study by N. Varghese and N. Medina-Medina (2021) aimed to develop a new methodology for mobile application development, referred to as "Agile Beeswax." The authors argue for the necessity of such a methodology due to the specificities of mobile application development and the requirements for rapid response to changes in this field. The key elements of the Agile Beeswax methodology include an incremental and iterative approach to development, consisting of two main iteration cycles (sprints): the incremental design cycle and the incremental development cycle. These two cycles are connected by a bridge. The Agile Beeswax methodology is divided into six phases, including strategy and idea, user experience design, user interface design, design to development, handover and technical decisions, development and deployment, and monitoring. The authors indicate that one of the main advantages of their methodology is that it is designed to encompass both academic and business-oriented perspectives, aiming to unite these two communities. The findings show that the Agile Beeswax methodology is focused on addressing specific problems and challenges that arise in the process of mobile application development. It combines Agile and Scrum practises, engineering technical practises,

and operational practises to achieve greater efficiency in this field. The authors emphasise the importance of responding quickly to changes in the market and acknowledge that Agile methodologies are the best approach for mobile application development, as they allow for faster changes and improved product quality. It can be agreed that the Agile Beeswax methodology represents an interesting approach to mobile application development that can be beneficial for both the academic and business-oriented communities, addressing specific problems associated with this type of development.

As a result of the study, it was established that each mobile application development method has its advantages and limitations. Native applications provide high performance and a native look but require more resources and development time. Hybrid applications save time and costs but may have limited access to device functions. Cross-platform applications combine the benefits of both approaches, but the choice of the right framework is crucial. The study by S.R. Uplenchwar *et al.* (2022) underscores the importance of selecting the right cross-platform framework, such as Flutter, for creating applications with a single codebase. Dart and Firebase also play a significant role in easing the development process. The study by H.A. Alrabaiah and N. Medina-Medina (2021) proposes the Agile Beeswax methodology for mobile application development, uniting the best practises of Agile and Scrum. This methodology is aimed at improving mobile application development and can be beneficial for both academic and business-oriented communities. The choice of approach and tools for mobile application development should be based on the specific needs of the project, limitations, and important aspects of productivity and application appearance. The discussed methodologies and frameworks can be useful in achieving the goals of mobile application development. In particular, the choice between native, hybrid, and cross-platform applications should be determined by considering the project's needs for performance, access to device capabilities, and constraints on budget and resources. Research shows that for better mobile application development, it is important to consider the individual requirements and specificity of the project, as well as use appropriate methodologies and tools.

CONCLUSIONS

In the study, various approaches to the development of mobile applications were considered, such as native, hybrid, and cross-platform solutions. Native applications, developed for a specific platform, provide the highest performance and access to device capabilities but require considerable effort and resources. Hybrid applications reduce costs and time by using a shared codebase but may have limited access to device functions. Cross-platform applications, combining the advantages of both approaches, can be effective with the right choice of a framework. Different frameworks for mobile application development were analysed in the study, including Flutter, Xamarin, Apache Cordova, React Native, and Ionic. Each has its own features and advantages. Flutter, developed by Google, allows creating high-quality interface and performance applications with a single codebase. Xamarin by Microsoft uses the C# programming language and integrates with the Microsoft ecosystem. Apache Cordova is based on web technologies and initialises a native application through WebView. React Native from Facebook enables creating applications with a native look and fast coding using React and JavaScript. Ionic uses HTML, CSS, and JavaScript for hybrid development with the option to use Angular. The results of the study show that the choice of a framework should be based on the specific project requirements, considering performance, budget, and resources. Each framework has its advantages and limitations, and it is essential to consider them in the context of a particular project.

Recommendations include selecting the development approach for a mobile application based on specific project requirements, financial constraints, and resources. It is crucial to evaluate the performance, speed, and native appearance of the application correctly to achieve successful results in mobile application development. The obtained results can be useful for mobile application developers and organisations planning to create mobile applications, helping them choose the optimal approach and frameworks for development. Further research can focus on a detailed analysis and comparison of different cross-platform frameworks for mobile application development, exploring the impact of the development approach choice on the quality and performance of applications, and developing new methodologies and tools to optimise the mobile application development process, considering current technological and business needs.

ACKNOWLEDGEMENTS

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Alrabaiah, H.A., & Medina-Medina, N. (2021). Agile beeswax: Mobile app development process and empirical study in real environment. *Sustainability*, 13(4), article number 1909. doi: 10.3390/su13041909.

[2] Biørn-Hansen, A., Rieger, C., Grønli, T.M., Majchrzak, T.A., & Ghinea, G. (2020). An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*, 25, 2997-3040. doi: 10.1007/s10664-020-09827-6.

[3] Dittrich, F., Albrecht, U.V., Scherer, J., Becker, S.L., Landgraeber, S., Back, D.A., Fessmann, K., ... Klietz, M.L. (2023). Development of open backend structures for health care professionals to improve participation in app developments: Pilot usability study of a medical app. *JMIR Formative Research*, 7, article number e42224. doi: 10.2196/42224.
[4] Fojtik, R. (2019). Swift a new programming language for development and education. In T. Antipova & Á. Rocha (Eds.), *Digital Science 2019* (vol. 1114; pp. 284-295). Cham: Springer. doi: 10.1007/978-3-030-37737-3_26.
[5] Hu, J., Wei, L., Liu, Y., & Cheung, S.C. (2023). ωTest: Webview-oriented testing for android applications. In *ISSTA 2023: Proceedings of the 32nd ACM SIGSOFT international symposium on software testing and analysis* (pp. 992-1004). New York: Association for Computing Machinery. doi: 10.1145/3597926.3598112.
[6] Kaczmarczyk, A., Zając, P., & Zabierowski, W. (2022). Performance comparison of native and hybrid android mobile applications based on sensor data-driven applications based on Bluetooth low energy (BLE) and Wi-Fi communication architecture. *Energies*, 15(13), article number 4574. doi: 10.3390/en15134574.
[7] Karatanov, O., Yena, M., Bova, Y., & Ustymenko, O. (2021). Comparison of popular test frameworks JUnit and TestNG. *Young Scientist*, 5(93), 164-170. doi: 10.32839/2304-5809/2021-5-93-31.
[8] Kozub, H., & Kozub, Yu. (2022). Declarative method for creating multiplatform applications. *Bulletin of the Eastern Ukrainian National University named after Volodymyr Dahl*, 5(275), 10-15. doi: 10.33216/1998-7927-2022-275-5-10-15.
[9] Lachgar, M., Hanine, M., Benouda, H., & Ommane, Y. (2022). Decision framework for cross-platform mobile development frameworks using an integrated multi-criteria decision-making methodology. *International Journal of Mobile Computing and Multimedia Communications*, 13(1), 1-22. doi: 10.4018/ijmcmc.297928.
[10] Martínez, M. (2019). Two datasets of questions and answers for studying the development of cross-platform mobile applications using Xamarin framework. In *2019 IEEE/ACM 6th international conference on mobile software engineering and systems (MOBILESoft)* (pp. 162-173). Piscataway: Institute of Electrical and Electronics Engineers. doi: 10.1109/mobilesoft.2019.00032.
[11] Masaad Alsaid, M.A.M., Ahmed, T.M., Sadeeq, J., Khan, F.Q., Mohammad, & Khattak, A.U. (2021). A comparative analysis of mobile application development approaches: Mobile application development approaches. *Proceedings of the Pakistan Academy of Sciences: A. Physical and Computational Sciences*, 58(1), 35-45. doi: 10.53560/PPASA(58-1)717.
[12] Raeesi, A., Khajouei, R., & Ahmadian, L. (2022). Evaluating and rating HIV/AIDS mobile apps using the feature-based application rating method and mobile app rating scale. *BMC Medical Informatics and Decision Making*, 22, article number 281. doi: 10.1186/s12911-022-02029-8.
[13] Singh, M., & Shobha, G. (2021). Comparative analysis of hybrid mobile app development frameworks. *International Journal of Soft Computing and Engineering*, 10(6), 21-26. doi: 10.35940/ijsce.f3518.0710621.
[14] Sun, C., Ma, Y., Zeng, D., Tan, G., Ma, S., & Wu, Y. (2021). μDep: Mutation-based dependency generation for precise taint analysis on android native code. In *IEEE transactions on dependable and secure computing*, 20(2), 1461-1475. Piscataway: Institute of Electrical and Electronics Engineers. doi: 10.1109/TDSC.2022.3155693.
[15] Thamutharam, Y.N., Mustafa, M.B.P., Musthafa, F.N., & Tajudeen, F.P. (2021). Usability features to improve mobile apps acceptance among the senior citizens in Malaysia. *ASM Science Journal*, 16. doi: 10.32802/asmscj.2021.686.
[16] Tkachuk, A., & Bulakh, B. (2022). Research of possibilities of default refactoring actions in swift language. *Technology Audit and Production Reserves*, 5(2(67)), 6-10. doi: 10.15587/2706-5448.2022.266061.
[17] Uplenchwar, S.R., Denge, U.S., Bajoriya, A.S., & Bachwani, S.A. (2022). Review on detail information about flutter cross platform. *International Journal for Research in Applied Science and Engineering Technology*, 10(1), 1016-1022. doi: 10.22214/ijraset.2022.39977.
[18] Wu, C., Pérez-Álvarez, J.M., Mos, A., & Carroll, J.M. (2022). Codeless app development: Evaluating a cloud-native domain-specific functions approach. In *Proceedings of the 56th annual Hawaii international conference on system sciences, HICSS 2023* (pp. 6904-6913). Washington: IEEE Computer Society. doi: 10.48550/arxiv.2210.01647.
[19] Ziyodullayevich, A.Q., Babakulovich, Z.R., Bakhodirovna, M.Z., Bekmurodovich, S.A., & Akif-ogli, M.R. (2019). Efficient and convenient application to determine the functions and analysis of the reliability of the device. *International Journal of Innovative Technology and Exploring Engineering*, 9(2), 1804-1809. doi: 10.35940/ijitee.b7323.129219.
[20] Zohud, T., & Zein, S. (2021). Cross-platform mobile app development in industry: A multiple case-study. *International Journal of Computing*, 20(1), 46-54. doi: 10.47839/ijc.20.1.2091.

Порівняльний аналіз фреймворків для розробки мобільних програм: рідні, гібридні чи крос-платформні рішення

Олексій Геннадійович Зарічук
Менеджер комп'ютерних та інформаційних систем
ТОВ «Фідес»
02000, вул. Є. Сверстюка, 11а, м. Київ, Україна
https://orcid.org/0009-0009-0771-8465

Анотація. У сучасному цифровому світі розробка мобільних додатків є ключовою галуззю інформаційних технологій, а вибір оптимального підходу до їх розробки має вирішальне значення для ефективного впровадження на ринку. Метою цієї наукової роботи було проведення порівняльного аналізу різних фреймворків для розробки мобільних додатків: рідних, гібридних і крос-платформних рішень. Для досягнення поставленої мети були використані методи аналізу, синтезу та порівняння. Були проаналізовані характеристики різних фреймворків для розробки мобільних додатків, включаючи їх продуктивність, вартість та доступ до пристроєвих можливостей. Під час проведення дослідження встановлено, що рідні фреймворки вирізняються найвищою продуктивністю та можливістю забезпечити максимально нативний вигляд та функціональність додатку. Однак цей підхід має свої обмеження, оскільки вимагає окремої розробки для кожної платформи, що призводить до збільшення витрат часу та ресурсів. Гібридні рішення виявилися економічно вигідними, оскільки вони дозволяють використовувати єдину кодову базу для створення додатків для різних платформ. Це спрощує процес розробки та підтримки. Однак гібридні додатки можуть мати обмежену продуктивність через використання WebView для відображення інтерфейсу та обмежений доступ до пристроєвих можливостей. Крос-платформні фреймворки, у свою чергу, забезпечують баланс між продуктивністю та ефективністю витрат ресурсів. Вони дозволяють використовувати одну кодову базу для створення додатків для кількох платформ і при цьому можуть досягати задовільної продуктивності. Однак вони можуть мати обмежений доступ до деяких пристроєвих можливостей та вигляду додатку. Це дослідження робить новий внесок у науку шляхом детального порівняльного аналізу різних підходів до розробки мобільних додатків і фреймворків, які використовуються для їх створення. Отримані результати можна використовувати для прийняття інформованих рішень щодо вибору фреймворку для розробки мобільного додатку

Ключові слова: Java; розробка програмного забезпечення; продуктивність; нативний вигляд; кодова база