DOI: 10.5604/01.3001.0010.4838

PERFORMANCE ANALYSIS OF NATIVE AND CROSS-PLATFORM MOBILE APPLICATIONS

Paweł Grzmil, Maria Skublewska-Paszkowska, Edyta Łukasik, Jakub Smołka
Lublin University of Technology, Institute of Computer Science

Abstract. *This article presents the performance analysis of a cross-platform mobile application implemented with Xamarin and two native applications for Android and iOS platforms. The results concerning the time analysis for selected activities were compared in order to determine whether cross-platform tools are worth using in mobile application development. Native applications achieved better performance, however in some cases the cross-platform approach allows for significant development time reduction without deterioration in user experience.*

Keywords: mobile applications, cross-platform mobile applications, performance analysis of mobile applications development

ANALIZA WYDAJNOŚCI NATYWNYCH I WIELOPLATFORMOWYCH APLIKACJI MOBILNYCH

Streszczenie. *Artykuł przedstawia analizę wydajności mobilnej aplikacji wieloplatformowej utworzonej za pomocą technologii Xamarin oraz dwóch natywnych dla platformy Android i iOS. Wyniki dotyczące analizy czasów wykonywania wybranych czynności zostały porównane, aby odpowiedzieć na pytanie, czy warto używać rozwiązań wieloplatformowych w wytwarzaniu aplikacji mobilnych. Aplikacje natywne osiągnęły lepsze wyniki, jednakże w pewnych scenariuszach wykorzystanie technik wieloplatformowych pozwoli na znaczne oszczędności czasu, bez spadku poziomu wrażeń odbieranych przez użytkownika.*

Słowa kluczowe: aplikacje mobilne, wieloplatformowe aplikacje mobilne, analiza wydajności wytwarzania aplikacji mobilnych

Introduction

The goal of any mobile application developer is to provide applications to the widest possible audience. In view of the structure of the mobile operating systems market and the lack of compatibility between them, applications must be prepared for each platform separately. The specificity of the market rewards the simultaneous delivery of applications for all systems. For this reason, the development team must be divided into sub-teams responsible for the development of each version. This contributes to an increase in the number of programmers developing the application, and translates directly into an increase in the cost of the project. It is obvious that from a business point of view, reducing costs and delivering applications as soon as possible is a priority in decisions about the type of software produced. To meet these needs, methods of producing cross-platform applications have appeared. Thanks to their application, the application is implemented once, but can nevertheless be run on multiple systems. This allows to cut costs by shortening the development time and reducing the development team [2].

The purpose of this article is to analyze the performance of mobile applications produced natively and the cross-platform way. Three mobile applications were created: one dedicated to the iOS platform (implemented in the Objective-C programming language), one dedicated to the Android platform, and a cross-platform one, implemented in the Xamarin environment. The analysis covers typical tasks in mobile applications, i.e. numerical calculations, file access, downloading an image from the Internet and determining location.

1. Problems with generating mobile applications

Regardless whether applications are created using the native or multi-platform approach, programmers and designers face similar challenges during the development process. The most common problems include [2, 8]: the limited resources of mobile devices, wireless communications and the variety of mobile devices.

1.1. Limited resources

Year after year, mobile devices become more efficient, and their screen resolution increases. To maintain low weight and dimensions, particularly the thickness, the device's battery is considerably limited in size. Accordingly, the process of creating a mobile application requires considerations of optimizing the energy use. Users are reluctant to use applications that cause significant battery drain. Designing the application logic should first and foremost cover the planning activities that the application carries out in the background, the manner it communicates with the server or the limiting CPU-intensive tasks.

1.2. Wireless communication

The most important feature of the smartphone is the ability to use multiple wireless communication links. Besides GSM phone calls and communication with other devices via Bluetooth it is also possible to access the Internet via mobile HSDPA and LTE networks, or by the conventional Wi-Fi network. The implementation of each route, depending on requirements, is only one problem. Wireless data transmission is characterized by the presence of interference, errors in communication and large differences in bandwidth. Therefore, a major challenge is to ensure smooth operation of applications regardless of the connection type and quality. Wireless networks are vulnerable to attacks and therefore, while designing applications that operate on sensitive data, one should be particularly careful to ensure encryption and authentication.

1.3. The variety of mobile devices

There is a great variety of mobile devices on the market. Both Android and iOS support a wide range of sizes, aspect ratios and screen resolutions. From the 4-inch mobiles at the lower end, via high-end models of about 5-inch high pixel density screens, to large tablets in which the resolution reaches up to the 4K. Regardless of the device type, mobile applications should work correctly. The set of internal device components is not standardized. By creating a mobile application that uses a particular built-in sensor, one must handle the scenario where there is no such sensor in the device (or it is damaged).

2. Multi-platform applications in the Xamarin environment

A cross-platform mobile application is designed to run on many mobile systems. Such applications are usually created using technologies that provide an additional layer of abstraction above the system API, uniform for each platform [7].

The Xamarin Platform is a solution for creating cross-platform and native applications using the C# programming language. It was developed by the creators of Mono, Mono Touch and Mono for Android - implementations of the .NET Framework for Linux, iOS and Android correspondingly. Xamarin supports application

development for Android, iOS and Windows Phone. In February 2016, Microsoft took over Xamarin and at the same time open-sourced the whole platform [4]. The technology offers two ways to create cross-platform applications.

Xamarin Native is an approach that uses Android and iOS SDKs, which have been mapped to C# classes. Creators provide almost one hundred percent coverage of the Android and iOS API. Because of this, it is possible to realize everything in Xamarin that one can do in a native application. In this approach, the user interface is created separately for each platform, using the appropriate solutions for each of the systems [10].

The second approach to creating applications is using Xamarin Forms. This is a library that provides a platform-independent programming interface. It is thus possible to achieve a maximum amount of code shared between platforms. The application interface is created once for all platforms in the XAML language, using shared controls library. When the application is running, they are mapped to native components of the platform. This allows to achieve native performance and quality of the user interface [5].

These solutions have their advantages and disadvantages. Preparing a user interface in Xamarin Native requires more time, but at the same time provides full control over the views structure. Xamarin Forms allows a much higher degree of code sharing between platforms. The manufacturer recommends using the Native approach in applications that have complicated GUI and require use of multiple functions associated with the system API. Xamarin Forms is proposed for use in situations when the time of product development and code sharing is more important than a robust user interface [11].

The basic and most important advantage of cross-platform application development is shortening the development time. This allows for faster delivery of applications to stores and thus to users. Shortening production contributes significantly to the reduction of the project cost. Saving is also achieved by reducing the number of teams – creating a cross-platform application instead of two or more native ones it is enough to hire one team of programmers. One application means one set of technologies, tools and additional libraries used during the software development process. This simplifies the management of changes in the later stages of the application's life cycle. Sharing code between applications makes it easier to provide the same functionality on all supported platforms. Also, in the case of change of requirements, work on updating needs only be performed once. If you add a supported platform, the conversion of a hybrid application is much faster and easier than creating a native solution from scratch. Depending on the knowledge and experience of the team, the implementation of cross-platform technology can be quite effective. For example, if the programmers are familiar with .NET, it will be easier for them to get to know Xamarin than to learn Objective-C and iOS SDK.

The most often mentioned disadvantage of hybrid solutions is their lower performance compared to native methods. In the case of technologies based on running applications in a browser, this problem is particularly noticeable [6]. Cross-platform applications run slower because of loading the required runtime and numerous libraries at the beginning. Typically, technologies that enable the creation of hybrid applications provide their own wrapper interface for functionality offered by the system. Due to the differences in operating systems for mobile devices, often a cross-platform API provides a set of functionality that is the common for all supported platforms. Potentially, there may be some gaps in relation to the native API [9]. To achieve the performance and aesthetics of the user interface similar to native applications, it may be necessary to adjust some elements specifically for a particular platform. As with any new solution, developers need to have time to familiarize themselves with the technology used or may need some training. The use of hybrid methods in creating applications introduces a dependency on an external library. This can cause problems with support or occurring errors. One should also be aware that in creating a cross-platform application one may use technologies and languages not provided by the system manufacturer. Software developed in this way can have unforeseeable and difficult to detect errors [1, 2].

3. The performance analysis of native and cross-platform applications

In order to compare the performance of mobile applications created natively and in a hybrid-fashion, an examination was carried out to measure execution times of selected tasks. For the experiments the Android and iOS systems were selected due to their largest market share. Globally, both systems are installed on 95% of mobile devices in the world [3]. A cross-platform application has been created in the Xamarin environment. The choice of this technology was dictated by its high popularity and rapid growth [4].

The study was divided into 4 parts. Time is the element having the greatest impact on the user experience resulting from the use of a mobile application. Each of the experiments concerns another area of the system API use:

- numerical calculations – the test measured the time to calculate the number π to ten thousand decimal places;
- saving and reading from a file – the test measured the writing and reading times of a text file containing the result of the calculation of the number π from the previous test. The data to be written to the file are calculated once and kept in memory. The calculation time is not included in the result of the experiment;
- network support – the test measured the time needed to download from the Internet a large graphic file (about 7 MB). In the study, each application uses the same file placed on a publicly accessible server. Smartphones were connected to the same Internet connection via Wi-Fi;
- determination of location – the test measured the time to determine latitude and longitude using the mobile phone's sensors. In modern mobile devices the location can be determined using the GPS system, the BTS transmitter location or on the basis of the available Wi-Fi networks. In the absence of the choice of location method in iOS, the test in both systems was carried out using the default system settings.

The time was counted from the moment of execution of the first instruction of the application logic. Updating the user interface and presentation of the result was not included in the results. For each study a separate view was prepared in the application. In order to simulate conditions closest to the real ones, every test run is preceded by re-entering the appropriate screen in the application.

Six test applications were set up for the study – three for each system: native, cross-platform using the Xamarin Native approach and cross-platform using the Xamarin Forms libraries. Due to the differences in the systems architecture and the performance of the mobile components, the results obtained in the study will be compared only within the system platform. Figure 1 shows a comparison of the user interface in the native and multi-platform application.

In order to ensure the reliability of the results, each test was carried out with attention to the repeatability of the conditions. Applications ran on the same smartphones. For Android it was the LG G4, and for iOS – iPhone 6 16GB. In order to use all features of the system, the applications were tested on the latest operating system versions available for the devices. In the case of Android it was Android 6.0 "Marshmallow", and for iOS it was version 9.3.2. Applications were built with the "release" compiler setting and installed on the phone. Each test was performed 20 times for each system and each application. The networking tests were made using the same connection. Location tests were performed in the same location.

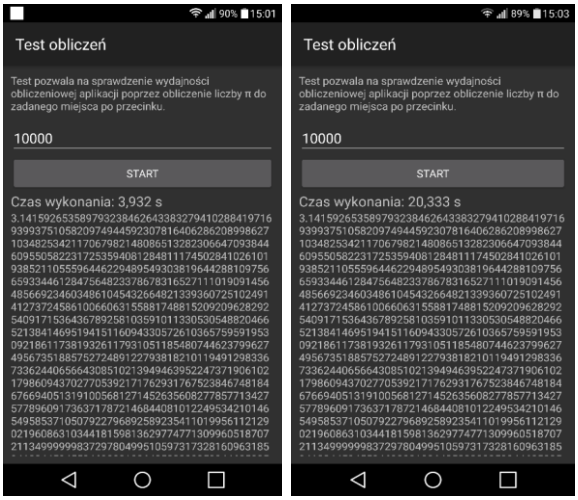

Fig. 1. Comparison of the user interface of a native application for Android (left) and a cross-platform one (right)

4. Results of analysis

The analysis was divided into subsections – one for each test.

4.1. Computing performance

Table 1 shows the results of the computing performance test. In both systems, native applications coped with the test much better than the cross-platform ones. However, the difference between the results of the native and cross-platform applications is much higher on Android. The calculations were performed about five times faster natively than in the cross-platform fashion. In the case of iOS, this difference was approximately 60%.

Table 1. Results of the computing performance study

		Android [s]	iOS [s]
Average computing time	Native	4.515	3.215
	Xamarin Native	22.315	5.116
	Xamarin Forms	20.804	5.121

4.2. File access

The results of the file write and read speeds are described in Table 2. In the file read test the Android native application turned out to be the slowest, but in other tests both multi-platform applications achieved worse results. The differences between the application versions are even several times, but the results are in milliseconds, so in real-life use these differences may not be so important.

Table 2. The results of the file read performance test

		Android [ms] read	Android [ms] write	iOS [ms] read	iOS [ms] write
Average time	Native	26.502	2.722	0.714	16.945
	Xamarin Native	5.114	6.751	1.484	18.475
	Xamarin Forms	8.256	10.309	5.614	21.246

4.3. Image downloading

Table 3 shows the results of the network connection performance test. All applications achieved similar results. The differences are insignificant. They are most likely caused by temporary variations in the connection bandwidth, that was used for the test.

Table 3. Results of testing the download speed of an image.

		Android [s]	iOS [s]
Average down-load time	Native	6.853	6.726
	Xamarin Native	6.667	6.476
	Xamarin Forms	7.039	6.830

4.4. Determining location

The results of the time to location fix (TTF) test are presented in Table 4. The most important is the scale of the results difference between the two systems. Android determined the location within a few seconds whereas iOS needed for it a few tens of milliseconds. As in previous tests, native applications achieved the best results.

Table 4. Results of the determining location test

		Android [s]	iOS [ms]
Average TTF	Native	3.196	12.109
	Xamarin Native	9.617	14.363
	Xamarin Forms	4.589	52.136

5. Conclusions

The very rapid development of the mobile market creates a huge demand for mobile applications. Among mobile operating systems the largest share falls to Android and iOS. They are installed on 95% of all mobile devices [3]. For software developers the need to reach the largest possible audience is obvious. In the current situation the market requires that applications be produced for both platforms. However, the production of native applications is quite expensive. Xamarin, by providing the ability to create cross-platform software, can significantly save time and resources.

Running the same program on many systems had to be paid for by certain compromises. Analyzing the results of performance comparison between native a hybrid applications, one can easily see the differences in the speed of implementation of the basic tasks. Tables 1–4 and Figures 2 and 3 present the results of a study broken up with regard to system platforms.

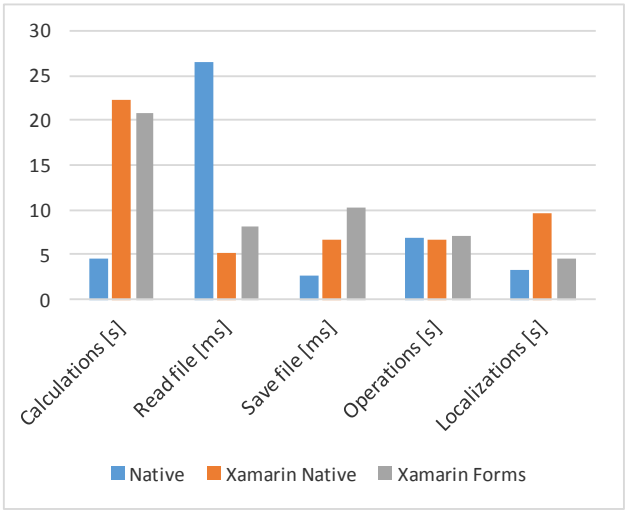

Fig. 2. Results of tests on the Android platform

In the network performance test the differences between the particular versions of the applications are negligible. The native iOS application achieved better results in every test compared to hybrid applications.

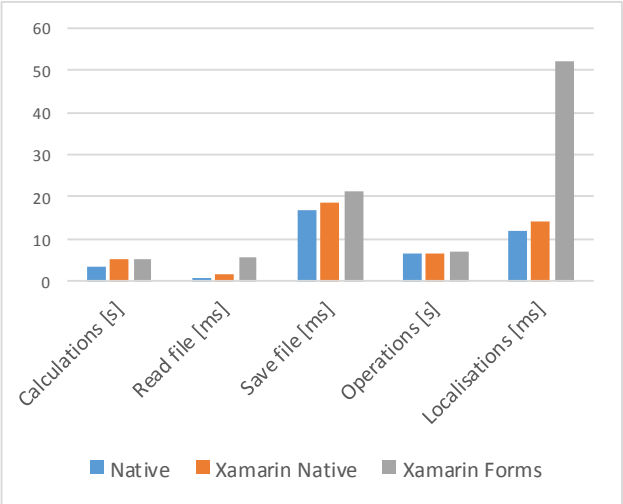

Fig. 3. Results of tests on the iOS platform

On Android only the file reading test came out worse off than in the multi-platform application. On the Google OS there are also more significant differences between the application versions than on iOS. In the computing performance test the native iOS application was approximately 60% faster than the hybrid, while on Android this test natively executed five times faster. Similarly, in the file recording test iOS performed natively about 10% faster, while the Android hybrid application performed the test almost three times more slowly. It should be noted that in the case of the file access and location determination tests in the iOS the results were counted in milliseconds, so that even several times difference in performance may be imperceptible to application users.

The Xamarin Platform performs very well in many situations. If one is aware of the disadvantages of cross-platform applications and the characteristics of the project allow to accept them, then Xamarin is worth using. The interface created by this technology is almost as good as the native solution. This is all the more important given that users expect from an application an aesthetic and smoothly-functioning interface. The Xamarin Platform offers the creation of high-quality products, while allowing for shortening the delivery to market time, and thus also for reducing costs.

Bibliography

[1] Corral L., Janes A., Remencius T.: Potential advantages and disadvantages of multiplatform development frameworks – A vision on mobile environments. Procedia Computer Science 10/2012, 1202–1207.
[2] El-Kassas W. S., Abdullah B. A., Yousef A. H., Wahba A. M.: Taxonomy of Cross-Platform Mobile Applications Development Approaches. Ain Shams Engineering Journal 8(2)/2017, 163–190.
[3] Epstein Z.: September data shows sharpest iOS market share drop in months as Android gains. http://bgr.com/2015/10/01/iphone-market-share-q3-2015-android/ (available: 16.05.2016).
[4] Foley M. J.: Microsoft open sources Xamarin's software development kit. http://www.zdnet.com/article/microsoft-open-sources-xamarins-software-development-kit/ (available: 21.05.2016).
[5] Jensen D.: Xamarin.Forms Succinctly, Syncfusion Inc., 2015.
[6] Kataria M.: Native vs cross platform development – Performance & limitations. https://www.simform.com/blog/native-vs-cross-platform-development (avaible: 26.05.2016).
[7] Nitze A., Schmietendorf A.: Cross-Platform Mobile Application Development, ASQT, 2013.
[8] Redda Y. A.: Cross platform Mobile Applications Development. Institutt for datateknikk og informasjonsvitenskap, 2012.
[9] Optimus Information, Cross-Platform Framework Comparison: Xamarin vs Titanium vs PhoneGap, http://www.optimusinfo.com/cross-platform-framework-comparison-xamarin-vs-titanium-vs-phonegap/ (avaible: 26.05.2016).
[10] Xamarin, Introduction to Mobile Development. https://developer.xamarin.com/guides/cross-platform/getting_started/introduction_to_mobile_development/ (avaible 28.05.2016).
[11] Xamarin, Which Xamarin approach is best for your app? https://www.xamarin.com/forms (avaible 28.05.2016).

Eng. Paweł Grzmil
e-mail: pawel.grzmil@gmail.com

Graduate of Computer Science at the Lublin University of Technology. From 2012 to 2014 Microsoft Student Partner and President of the scientific circle Grupa.Net at the University. Organizer of the IT Academic Days conference there and speaker at the Check.it conference. Professional developer of mobile applications.

Ph.D. Eng. Maria Skublewska-Paszkowska
e-mail: maria.paszkowska@pollub.pl

Academic employee at the Institute of Computer Science, Faculty of Electrical Engineering and Computer Science at the Lublin University of Technology. Received her master's degree there. Obtained her doctoral degree at the Silesian University of Technology. Her research activities include: traffic acquisition methods, 3D motion data analysis, 3D algorithms, mobile programming

Ph.D. Edyta Łukasik
e-mail: e.lukasik@pollub.pl

Graduated in mathematics at the Marie Skłodowska-Curie University in Lublin. Received her Ph.D. from the Faculty of Mathematics, Physics and Computer Science there. Since 1998 member of the academic staff of the Lublin University of Technology. Her research interests are mainly programming languages and algorithmization, data structures, numerical and optimization methods, mobile applications as well as acquisition methods of 3D motion.

Ph.D. Eng. Jakub Smolka
e-mail: jakub.smolka@pollub.pl

Research worker at the Institute of Computer Science, Faculty of Electrical Engineering and Computer Science at the Lublin University of Technology. Earned his master's degree there, and his doctoral degree at the Silesian University of Technology. His research activity is in the area of processing digital images, in particular their segmentation and compression, 3D motion analysis and mobile programming.

otrzymano/received: 20.06.2016 *przyjęto do druku/accepted: 01.06.2017*