Comparison and Evaluation of Cross Platform Mobile Application Development Tools

Article *in* International Journal of Applied Mathematics Electronics and Computers · December 2020

DOI: 10.18100/ijamec.832673

2 authors:

Mehmet IŞitan
Sakarya University
1 PUBLICATION **3** CITATIONS
SEE PROFILE

Murat Koklu
Selcuk University
111 PUBLICATIONS **887** CITATIONS
SEE PROFILE

IJAMEC
e-ISSN: 2147-8228

INTERNATIONAL JOURNAL OF APPLIED MATHEMATICS ELECTRONICS AND COMPUTERS

www.dergipark.org.tr/ijamec

International Open Access

Volume 08
Issue 04

December, 2020

Research Article

Comparison and Evaluation of Cross Platform Mobile Application Development Tools

Mehmet ISITAN [a], ***Murat KOKLU*** [b]

[a] *Sakarya University, Faculty of Computer and Information Science, Department of Software Engineering*
[b] *Selcuk University, Faculty of Technology, Department of Computer Engineering*

ARTICLE INFO

Article history:
Received 28 November 2020
Accepted 6 December 2020

Keywords:
Cross Platform,
Mobile Development,
Mobile Frameworks,
One Code

ABSTRACT

In order to develop a mobile application, it is necessary to develop software separately for each operating system to be outputted. In response to this problem, frameworks that can give application outputs for more than one operating system by developing applications on only one platform have been developed. With the recent diversification of these systems, which are called cross platform mobile application development tools, which one should be preferred has become a problem for developers. In this study, the cross-platform mobile application development tools that have come to the fore in recent years will be determined and evaluated separately based on the pros and cons of distinguishing parameters. With the help of the applications to be developed, values such as processor, memory, battery and network usage, rendering time, opening time, installation file size, application size will be measured. It is also aimed to help developers find out which framework is more suitable for their needs by comparing them on topics such as popularity, third party software support, operating systems that can be outputted, development languages and ease of use, speed - performance. In the study, it was observed that Flutter and React Native gave more successful results.

1. Introduction

As it is known, there are many mobile operating systems used in the market such as Android, IOS, Windows Phone, Blackberry, Ubuntu, Symbian, RIM OS, BADA, Palm, Maemo, Meego, Verdict, KaiOS, Open WebOS [1]. Considering the programming processes of the programs that will run on these operating systems, it is observed that the software languages and platforms of all three are completely independent from each other. Therefore, each application should be developed in accordance with the language and template determined by the operating system it will run on. This necessity forces mobile application developers to have a very difficult, time consuming and costly process.

With the effect of developing technology in recent years, a solution has been developed that will provide revolutionary convenience in developing applications for mobile devices. Thanks to these tools, which are called cross platform mobile application development tools, software can be developed on only one platform and output can be obtained in accordance with the software development languages and templates determined by all these operating systems. While using these tools, it will be sufficient to develop software by only complying with the software language and format determined by the specified tool. In this way, with the code written using the environment provided by the selected platform, application output can be obtained to the platforms needed faster, easier and with less cost. The previous studies on this subject are given below.

In the study conducted by Allen et al (2010); Frameworks such as PhoneGap and Rhomobile were compared and the processes of creation and publishing in app stores were mentioned [2].

Palmieri et al. (2012) has discussed a comparison between four frameworks: Rhodes, PhoneGap,

* Corresponding author. E-mail address: mehmetisitann@gmail.com
DOI: 10.18100/ijamec.832673

DragonRad, MoSync. Comparison was made according to application interfaces, programming languages, supported mobile operating systems, market share, licenses and integrated development environments [3].

Heitkötter et al. (2013), in their studies; they evaluated web applications, applications developed with PhoneGap or Titanium Mobile and native applications. With this evaluation, it is concluded that PhoneGap product is more suitable for applications using native interfaces [4].

Dalmasso et al. (2013) conducted a survey for PhoneGap, Titanium, and Sencha Touch. Comparisons have been made in terms of CPU, memory usage and power consumption. Test applications have been developed on the Android operating system using these tools. It has been determined that PhoneGap consumes less memory, CPU and power since it does not contain special interface components [5].

Amatya et al. (2013) with their study; They argued that although cross platform frameworks are not fully mature, they show great potential. As a result of the study, it is concluded that the web-based approach offers the best for cross-platform mobile application development [6].

In the study by Gültürk Karlı (2014), a new software framework developed to help developers using cross platform mobile application tools is proposed. The proposed software framework has provided various features to increase the efficiency and quality of the resulting application. The developed software framework has been experimentally applied on data mining applications [7].

Charkaoui et al. (2014) stated that the choice of cross-platform mobile application development framework depends on the following two factors in their study; what kind of mobile application is needed and the requirements of the targeted platforms. It is concluded that the existing tools are insufficient for high capacity applications [8].

Dhillon et al (2014); they compared PhoneGap, Appcelerator Titanium, Adobe Air and MoSync tools. As a result of the study, they stated that Adobe Air and Appcelerator Titanium gave the best results and PhoneGap gave the worst results [9].

Tunali et al. (2015) compared the PhoneGap, Xamarin, Appcelerator Titanium, and Smartface App Studio tools on a theoretical table [10].

In the study conducted by Boushehrinejadmoradi et al. (2015); a testing tool called X-Checker has been developed. Xamarin, a popular framework that enables Windows Phone applications to be cross compiled on native Android and iOS applications, has been tested [11].

Jiang (2016); It has evaluated Xamarin and Cordova tools by scoring them with parameters such as ease of use, setup, productivity, memory and power consumption, security, project size and achieved total scores close to each other [12].

Latif et al (2016); worked on a questionnaire to investigate the basic requirements of cross platform mobile application development tools. It is concluded that the MDA (Model Driven Architecture) approach is superior [13].

In the study conducted by Öberg (2016), Xamarin and Cordova tools were evaluated on an application called Teknisk Förvaltning. CPU and RAM usage, development speed, application launch speed and the views of tools such as DatePicker and AlertDialog were compared [14].

Cristiane et al. (2018) conducted a study involving comparison between PhoneGap, Sencha Touch and Titanium frameworks and native applications. With the research, they aimed to measure their current maturity status. Memory usage and performances were measured with an application that takes photographs and accesses multimedia sources. It is concluded that native applications perform better in all respects and the Titanium framework is slower than others [15].

In the study of Shah et al. (2019), native applications and applications written with cross platform frameworks were compared theoretically. As a result, it has been concluded that while the desired kind of applications can be developed with native applications, it is concluded that large-scale applications such as the Asphalt game cannot be developed with cross platform frameworks. [16].

Application developers spend a lot of time to determine which mobile application development tool is the most suitable for them. Because after the developers choose one of these tools, they first enter the learning and professionalization process, and then the application development process.

In this study, the pros and cons of each of them, including the recently released cross-platform mobile application development tools, have been evaluated separately on the basis of distinctive parameters. An application was developed and measured on the empty foundation of each platform. It is also aimed to help developers find out which mobile application development tool is more suitable for their needs by comparing the processor, memory, battery and network usage, the platforms it supports, popularity, third party software support, rendering and opening time, speed-performance.

2. Material and Method

There are many frameworks for developing cross platform mobile applications. Some of them are React Native, Flutter, Xamarin, Nativescript, Ionic Framework, Unity 3D, Cocos 2D, Titanium, Phonegap, Sencha Touch, Appcelerator Titanium, Apache Cordova, Rhodes, Onsen UI, Framework 7, Kony, Jasonette, iFactr, FeedHenry, Qt Corona [17]. The use of many of them has decreased considerably in recent years and a few of them have started to be preferred. In this study, first of all, frameworks that are trending in the last period will be determined and then

performance tests will be made between the applications to be developed. In order to determine the most preferred ones, the 3 systems from which the developers get the most help will be used. These; GitHub is Stackoverflow and Google Trends.

StackOverflow (SO) is the most popular community for getting answers to software development questions and is a rapidly growing knowledge base on topics ranging from algorithms to languages and tools [18]. The graph showing the cross platform mobile application development tools that have been the most sought after in SO in recent years is given in Figure 1.

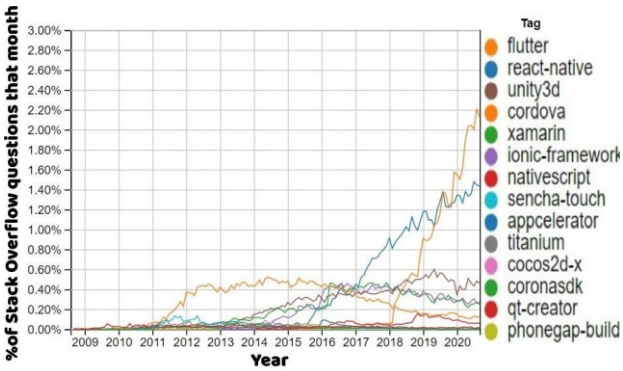

Figure 1. Trending frameworks of recent years on the Stackoverflow site [19]

According to the data announced by SO, Flutter is the 3rd and React Native is the 8th in the list of the most popular frameworks by developers. The complete list is given in Figure 2. According to these data, the number of records opened on the SO about the trending frameworks of recent years is given in Figure 3.

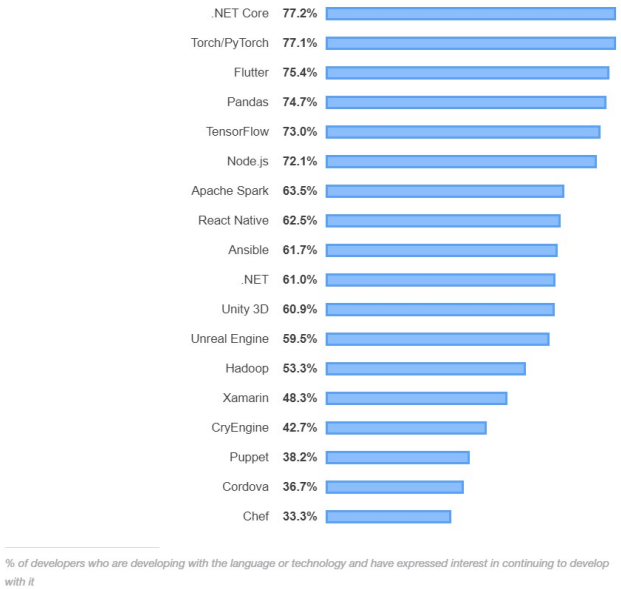

Figure 2. List of most popular frameworks based on Stackoverflow 2019 data [20]

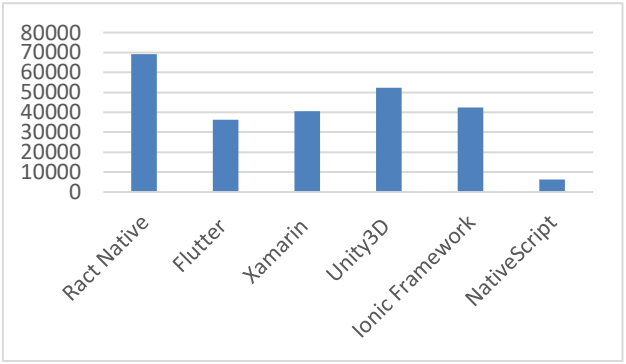

Figure 3. Number of records opened on Stackoverflow about frameworks specified so far

Google Trends (GT) gives the number of searches done on Google. When the same research is done on GT, the graphic in Figure 4 is obtained. According to this graph, it can be easily seen that the development tools that software developers use the most and seek support the most in recent years are React Native and Flutter.

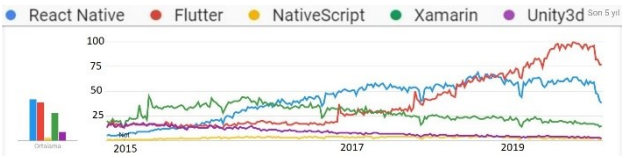

Figure 4. Comparison of the last 5 years according to Google Trends

GitHub is the most popular web-based collaboration platform for software developers. It is the world's largest open source software platform with over 28 million users [21]. The number of 3rd party applications shared on GitHub and calculated based on the keyword are given in Table 1 [22].

Table 1. Number of projects on GitHub [22]

Platform name	Number of projects on GitHub
React Native	21,832
Flutter	15,140
Nativescript	687
Xamarin	3,129

On GitHub, users star the applications they like and inform other users that they are useful. Accordingly, the number of stars is also of great importance. The star numbers of these platforms shared with open source are given in Table 2 below.

Table 2. Star numbers on GitHub [22]

Platform name		Number of stars on GitHub
React Native		90.7k
Flutter		105k
Nativescript		19.1k
Xamarin	Xamarin.Forms	4.9k
	Xamarin-macios	1.9k
	Xamarin-android	1.5k

In line with all these data, it was decided to compare React Native, Flutter, Xamarin and Nativescript, which are among the most preferred cross platform mobile application development tools in recent years. Comparison process; The same application will be written on each mobile application development platform and the loading performance of these applications will be measured on the same device. These apps contain a customized list of 1000 elements in total with styles. The written applications will be tested on the emulator of the Google Pixel 3 device with Android 10 operating system. Results will be obtained while viewing all of the objects in this application on the screen.

In Figure 5, the basic function of the applications written is given in the React Native platform. Then, screenshots of the build.gradle files of these applications are given. The basic implementation of the Xamarin framework does not contain a build.gradle file. However, if desired, the basic settings here can be changed later. Therefore, the gradle settings in the basic application of all the specified versions are the same.

```
export default function App(){
  var myloop = [];
    for (let i = 0; i < 1000; i++) {
      myloop.push(
        "Mehmet" + i
      );
    }
  return (
    <View style={styles.container}>
      <FlatList style={{ flex: 0 }} initialNumToRender={myloop.length}
        keyExtractor={(item) => item}
        data={myloop}
        renderItem={ ({item}) => (
          <View>
            <Text style={styles.item}>{item}</Text>
          </View>
        ) }
      />
    </View>
  );
}
```

Figure 5. Application codes written with React Native

```
buildscript {
    ext {
        buildToolsVersion = "29.0.2"
        minSdkVersion = 16
        compileSdkVersion = 29
        targetSdkVersion = 29
    }
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath("com.android.tools.build:gradle:3.5.3")
    }
}
allprojects {
    repositories {
        mavenLocal()
        maven {
            url("$rootDir/../node_modules/react-native/android")
        }
        maven {
            url("$rootDir/../node_modules/jsc-android/dist")
        }
        google()
        jcenter()
        maven { url 'https://www.jitpack.io' }
    }
}
```

Figure 6. React Native application's gradle.build file

```
buildscript {
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.5.0'
    }
}
allprojects {
    repositories {
        google()
        jcenter()
    }
}
rootProject.buildDir = '../build'
subprojects {
    project.buildDir = "${rootProject.buildDir}/${project.name}"
}
subprojects {
    project.evaluationDependsOn(':app')
}
task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Figure 7. Flutter application's gradle.build file

```
buildscript {
    def initialize = { ->
        def userDir = "${rootProject.projectDir}/../.."
        apply from: "$rootDir/gradle-helpers/user_properties_reader.gradle"
        apply from: "$rootDir/gradle-helpers/paths.gradle"
        rootProject.ext.userDefinedGradleProperties = getUserProperties("${getAppResourcesPath(userDir)}/Android")
    }
    initialize()
    def computeKotlinVersion = { -> project.hasProperty("kotlinVersion") ? kotlinVersion : "1.3.72"
    }
    def kotlinVersion = computeKotlinVersion()
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.6.4'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlinVersion"
    }
}
allprojects {
    repositories {
        google()
        jcenter()
    }
    beforeEvaluate { project ->
        if (rootProject.hasProperty("userDefinedGradleProperties")) {
            rootProject.ext.userDefinedGradleProperties.each { entry ->
                def propertyName = entry.getKey()
                def propertyValue = entry.getValue()
                project.ext.set(propertyName, propertyValue)
            }
        }
    }
}
task clean(type: Delete) {
    delete rootProject.buildDir
}
```

Figure 8. Gradle.build file of Nativescript application

3. Experimental Results

In this section, the applications written are run in debug mode separately on each platform and measurements are made based on parameters such as application size, creation time, use of device resources.

3.1. Application Size

The applications developed in this section were made operational and their sizes were examined.

3.1.1. The size of the application resource on disk

In this section, the size of the source files of the application created in debug mode to a specified file path is examined, excluding the resources of each mobile application development tool. After the applications are installed and run, the total dimensions of the application resources on the computer are given in Table 3.

Table 3. The size of the source files on the disk in the measurement platforms

Platform Name	Source Files on Disk (MB)	Version
React Native	467	0.63.3
Flutter	401	1.22.1
Nativescript	434	7.0.10
Xamarin	133	16.7.000.456

3.1.2. The size of the application's installation file

In this section, the dimensions of the setup files (.apk) created in release mode of the most basic applications for each platform are examined. Table 4 shows the dimensions of the installation files of these applications for the Android operating system.

Table 4. The size of the installation files of the applications written

Platform Name	Size of Setup Files (MB)
React Native	23.4
Flutter	15.2
Nativescript	23.5
Xamarin	11.1

3.2. Render Time

After this section (including this section), measurements continued through the application mentioned in section 2 for comparison operations. The time from running the mentioned applications in debug mode to seeing the last element in the list is given in Table 5.

Table 5. Application render times

Platform Name	Time(s)
React Native	51
Flutter	28
Nativescript	14
Xamarin	36

When the application written with React Native is run, the elements in the list are created on the screen by default as 20. As such, it takes 3 minutes and 26 seconds in total to open the application and load the entire list. When the settings were made that allow the entire list to be loaded at the same time, this time was measured as 51 seconds. When the last list item is displayed, the list still continues to show the empty list downstream and after 3 seconds the empty items at the bottom of the list are automatically discarded and restricted to show the correct list. No such problem has been encountered with others.

3.3. Use of Device Resources

In this section, device resource consumption was measured while running mobile applications written separately for each platform on the device and displaying all objects on the screen. Profiler feature of Android Studio application was used in the measurement process.

3.3.1. CPU usage

After the applications written with each of the cross platform mobile application development tools were run sequentially, a thread containing the name of the current application was created in the list of threads running on the device. Measurements have been made based solely on the amount of CPU used by the current application thread. Measurements were made until all the objects in the lists in each application were loaded so that they could be displayed on the screen.

First, the application written with React Native was run and a thread called test (com.test) was created in the thread list. This thread required a maximum of 79% of the processor's resources for 51 seconds as shown in Figure 9. The vertical axis in the graphs indicates the rate of the CPU's resource used in percent, and the horizontal axis indicates the time in seconds.

When the application written with Flutter was run, a thread called flutter_app (ple.flutter_app) was created in the thread list. This thread has used the processor for 28 seconds as shown in Figure 10 and needed a maximum of 75% resources.

When the application written with Nativescript was run, a thread called nativescriptIlk (org.nativescript.nativescriptIlk) was created in the thread list. This thread used the processor for 14 seconds as in Figure 11 and needed a maximum of 70% resources. The graphic of Xamarin with the same process is given in Figure 12. This process took 36 seconds and required a maximum of 62% of resources.

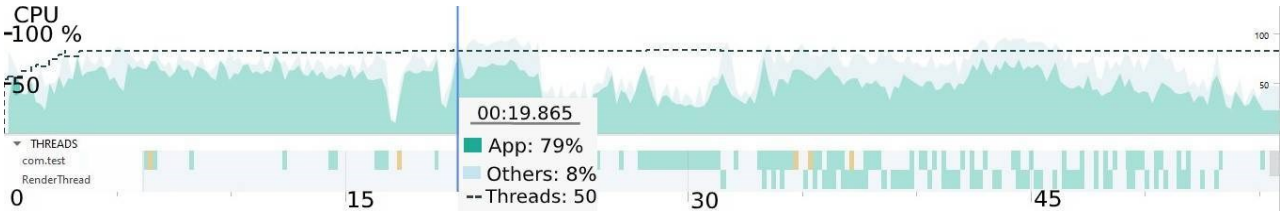

Figure 9. CPU rates used by the application written with React Native

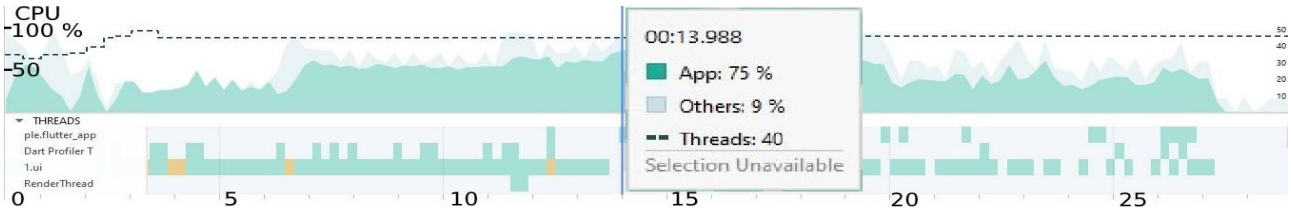

Figure 10. CPU rates used by the application written with Flutter

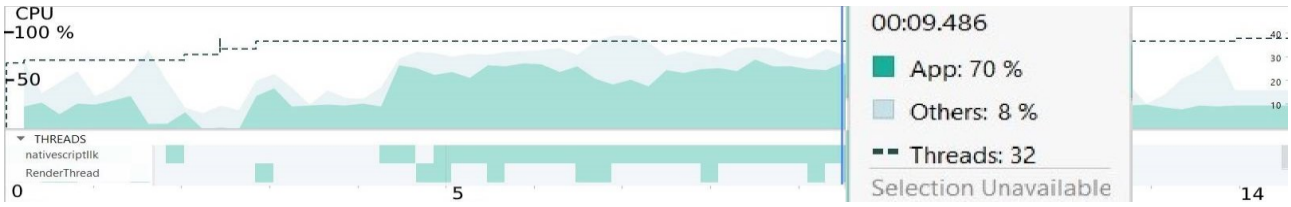

Figure 11. CPU rates used by the application written with Nativescript

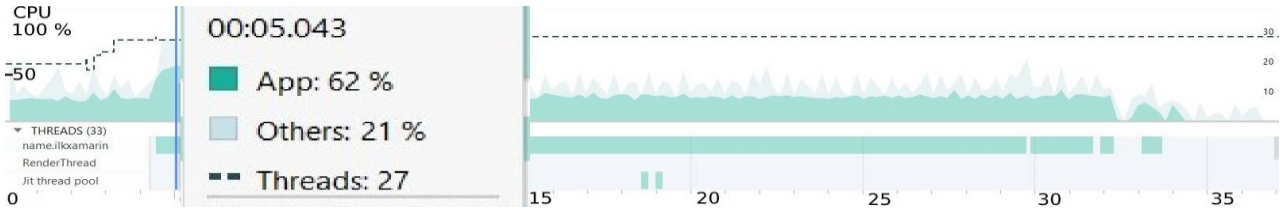

Figure 12. CPU rates used by the application written with Xamarin

3.3.2. Memory usage

In order to measure the memory usage, applications written for each platform were run one by one and the memory usage amount during the loading of all 1000 objects was measured.

In the application written with React Native, when this process was performed, the graphic in Figure 13 appeared and the application required a maximum resource of 33.2 MB. The green part in the graph shows the amount used by the application. The vertical axis indicates the amount of memory used, and the horizontal axis indicates the time in seconds.

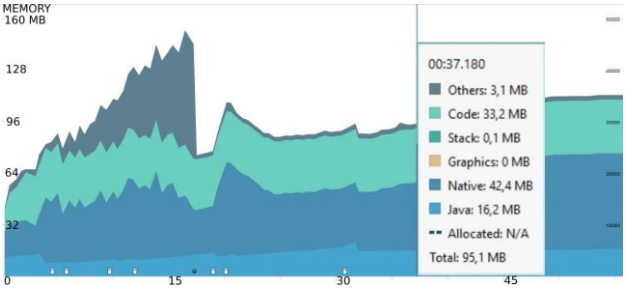

Figure 13. Memory rates used by the application written with React Native

When the same operation was performed in the application written with Flutter, the graphic in Figure 14 appeared and the application required a maximum resource of 23.4 MB. The graph of Nativescript is given in Figure 15 and it needs a maximum of 26.8 MB of resources, and finally the graph of Xamarin is given in Figure 16 and it needs a maximum of 18.3 MB of resources.

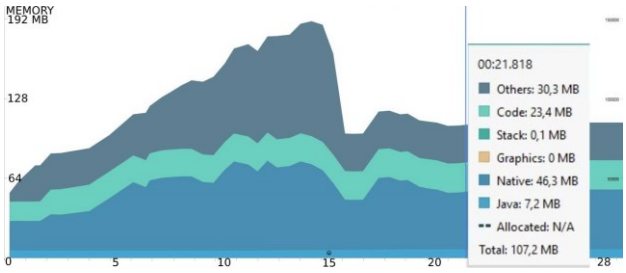

Figure 14. Memory rates used by the application written with Flutter

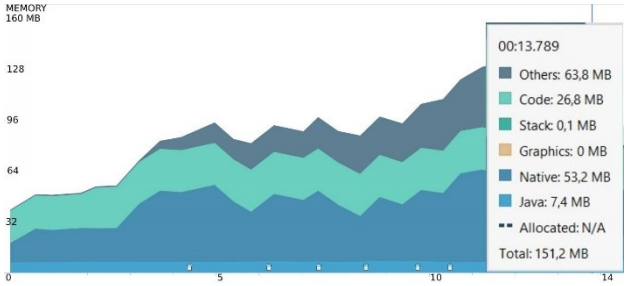

Figure 15. Memory rates used by the application written with Nativescript

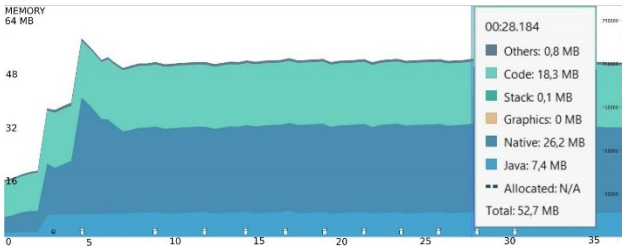

Figure 16. Memory rates used by the application written with Xamarin

3.3.3. Energy consumption

In order to measure energy consumption, or in other words, battery usage, the status of applications during loading all objects was examined.

Battery usage data during the loading of the content of the application written with React Native is given in Figure 17. The vertical axis in this graph indicates the usage rate of the battery and the horizontal axis indicates the time in seconds. The graphic of Flutter is given in Figure 18, Nativescript graphic is given in Figure 19 and finally the graphic of Xamarin is given in Figure 20.

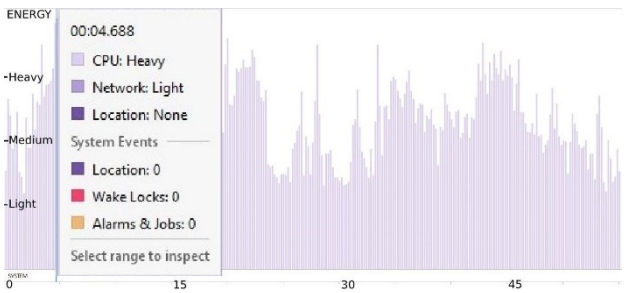

Figure 17. The amount of energy used by the application written with React Native

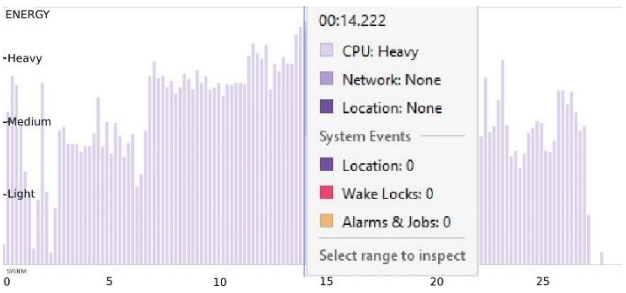

Figure 18. The amount of energy used by the application written with Flutter

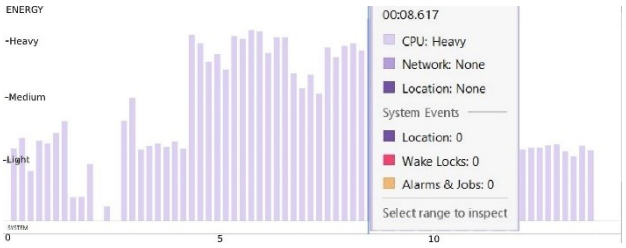

Figure 19. The amount of energy used by the application written with Nativescript

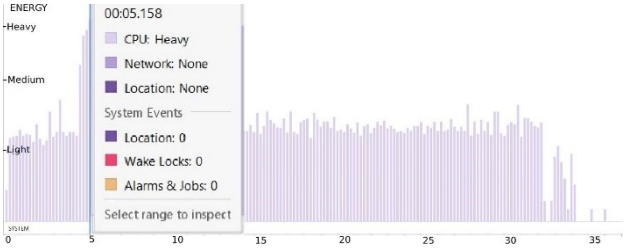

Figure 20. The amount of energy used by the application written with Xamarin

3.3.4. Network usage

In this section, network usage was checked until the whole list was loaded. No network request is included in any of the applications. In this case, the application code is expected not to make any network exchanges.

In the application written with React Native, network requests are made intermittently. The related graphic is given in Figure 21. In this graph, the vertical axis indicates the amount of network usage per second, and the horizontal axis indicates the time in seconds.

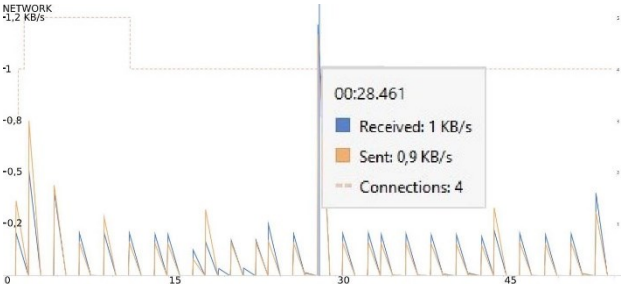

Figure 21. Network rates used by the application written with React Native

Applications written with other mobile application development tools did not perform any network

communication.

4. Results and Recommendations

Table 6. Some features of popular cross platform mobile app development tools

	React Native	Flutter	Xamarin	Ionic Framework	NativeScript
Open Source	Yes	Yes	Yes, paid	Yes, paid	Yes, paid
Company	Facebook	Google	Microsoft	Drifty.co	Telerik
Technologies	React.js, Javascript	Dart	C# (Objective-C, Java and C++ based libraries)	Javascript, TypeScript	Javascript, Angular, TypeScript
Target platforms	iOS, Android	iOS, Android	iOS, Android, Windows	iOS, Android	iOS, Android
Release date	2015	2017	2012	2013	2015

Looking at SO and GT data, it is seen that one of the most preferred frameworks is Unity3D. Unity3D, unlike other prominent development tools, is suitable for developing applications that contain more games or graphics. A developer who wants to develop such a cross-platform mobile application is recommended to choose the Unity3D framework. The theoretical comparison of frameworks other than this framework is given in Table 6.

The two most important criteria for choosing the cross platform mobile application development tool are the application developed has a performance and error-free operation and there is sufficient developer support behind it. In this context, considering SO, GitHub and GT data, React Native and Flutter have the biggest popularity and developer support in recent years. According to this criterion, Flutter has surpassed React Native in the last few years. In terms of performance, it is seen that Flutter is ahead of React Native.

If these tools are used to develop only one application, one of the important criteria to look at is 3rd party software support. The application to be developed should be divided into modules and it should be checked whether these modules were developed for the vehicle to be selected before. If you already have web technologies development experience, you should check out the development languages to learn fast. In this sense, React Native is one step ahead as it can be developed with Javascript. For Flutter, Dart, its own development language, must be learned.

As a result of all these evaluations, it is recommended to choose any of the React Native and Flutter tools, but the final decision belongs to the developer.

References

[1] NetMarketshare. (2020, 03.12.2020). *Market Share Statistics for Internet Technologies*. Available: https://netmarketshare.com/operating-system-market-share.aspx?options=%7B%22filter%22%3A%7B%22%24and%22%3A%5B%7B%22deviceType%22%3A%7B%22%24in%22%3A%5B%22Mobile%22%5D%7D%7D%5D%7D%2C%22dateLabel%22%3A%22Trend%22%2C%22attributes%22%3A%22share%22%2C%22group%22%3A%22platform%22%2C%22sort%22%3A%7B%22share%22%3A-1%7D%2C%22id%22%3A%22platformsDesktop%22%2C%22dateInterval%22%3A%22Monthly%22%2C%22dateStart%22%3A%222019-11%22%2C%22dateEnd%22%3A%222020-10%22%2C%22segments%22%3A%22-1000%22%7D

[2] S. Allen, V. Graupera, and L. Lundrigan, *Pro smartphone cross-platform development: iPhone, blackberry, windows mobile and android development and distribution*. Apress, 2010.

[3] M. Palmieri, I. Singh, and A. Cicchetti, "Comparison of cross-platform mobile development tools," in *2012 16th International Conference on Intelligence in Next Generation Networks*, 2012, pp. 179-186: IEEE.

[4] H. Heitkötter, S. Hanschke, and T. A. Majchrzak, "Evaluating cross-platform development approaches for mobile applications," in *International Conference on Web Information Systems and Technologies*, 2012, pp. 120-138: Springer.

[5] I. Dalmasso, S. K. Datta, C. Bonnet, and N. Nikaein, "Survey, Comparison and Evaluation of Cross Platform Mobile Application Development Tools," (in English), *2013 9th International Wireless Communications and Mobile Computing Conference (Iwcmc),* pp. 323-328, 2013.

[6] S. Amatya and A. Kurti, "Cross-platform mobile development: challenges and opportunities," in *International Conference on ICT Innovations*, 2013, pp. 219-229: Springer.

[7] G. KARLI, "Cross-platform mobile development," *Dokuz Eylül Üniversitesi,* 2014.

[8] S. Charkaoui and Z. Adraoui, "Cross-platform mobile development approaches," in *2014 Third IEEE International Colloquium in Information Science and Technology (CIST)*, 2014, pp. 188-191: IEEE.

[9] S. Dhillon and Q. H. Mahmoud, "An evaluation framework for cross‐platform mobile application development tools," *Software: Practice and Experience,* vol. 45, no. 10, pp. 1331-1357, 2015.

[10] V. Tunali, S. J. C. B. U. F. o. T. D. o. S. E. Zafer, Maltepe University: Faculty of Engineering, and N. S. D. o. S. Engineering, "Comparison of popular cross-platform mobile application development tools," 2015.

[11] N. Boushehrinejadmoradi, V. Ganapathy, S. Nagarakatte, and L. Iftode, "Testing Cross-Platform Mobile App Development Frameworks (T)," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 441-451.

[12] S. Jiang, "Comparison of native, cross-platform and hyper mobile development tools approaches for iOS and Android mobile applications," *Department of Computer Science and Engineering. University of Gothenburg,* pp. 1-15, 2016.

[13] M. Latif, Y. Lakhrissi, E. H. Nfaoui, and N. Es-Sbai, "Cross platform approach for mobile application development: A survey," in *2016 International Conference on Information Technology for Organizations Development (IT4OD)*, 2016, pp. 1-5: IEEE.

[14] L. Öberg, "Evaluation of Cross-Platform Mobile Development ToolsDevelopment of an Evaluation Framework," ed, 2016.
[15] C. M. S. Ferreira *et al.*, "An Evaluation of Cross-Platform Frameworks for Multimedia Mobile Applications Development," *IEEE Latin America Transactions,* vol. 16, no. 4, pp. 1206-1212, 2018.
[16] K. Shah, H. Sinha, and P. Mishra, "Analysis of Cross-Platform Mobile App Development Tools," in *2019 IEEE 5th International Conference for Convergence in Technology (I2CT)*, 2019, pp. 1-7.
[17] Statista. (2020, 03.12.2020). *Cross-platform mobile frameworks used by software developers worldwide in 2019 and 2020*. Available: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/
[18] A. Bosu, C. S. Corley, D. Heaton, D. Chatterji, J. C. Carver, and N. A. Kraft, "Building reputation in StackOverflow: An empirical investigation," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, 2013, pp. 89-92.
[19] Stackoverflow. (2020, 03.03.2020). *Stackoverflow Trends*. Available: https://insights.stackoverflow.com/trends?tags=react-native%2Cflutter%2Cxamarin%2Cnativescript%2Cionic-framework%2Cunity3d%2Ccocos2d-x%2Ctitanium%2Cphonegap-build%2Csencha-touch%2Cappcelerator%2Ccordova%2Ccoronasdk%2Cqt-creator
[20] Stackoverflow. (2020, 03.03.2020). *Most Loved, Dreaded, and Wanted Other Frameworks, Libraries, and Tools*. Available: https://insights.stackoverflow.com/survey/2019#technology-_-most-loved-dreaded-and-wanted-other-frameworks-libraries-and-tools
[21] T. V. Varuna and A. Mohan, "Trend Prediction of GitHub using Time Series Analysis," *IEEE,* 2019.
[22] GitHub. (2020, 19.10.2020). *Topics on GitHub*. Available: https://github.com/topics/