Mobile Application Frameworks of Different

Types and Their Comparison with Native Applications

Sidharth  
Lambton College

Author’s Note

Sidharth, Lambton College in Toronto,

pursuing Mobile Application Design and Development Course

Correspondence concerning this paper should be addressed to

Sidharth

Contact: sidharth100895@gmail.com

**Introduction**

Android and iOS are the primary platforms for mobile application improvement. In the event that you select a native mobile application development approach, you should compose distinctive code for the two platforms. This is the primary motivation behind why cross-platform improvement picked up such a great amount of fame in the mobile business since it spares time and effort when building an application.

Hybrid application advancement frameworks, for example, React Native, Xamarin from Microsoft and Flutter from Google have turned out to be progressively mainstream and their utilization will be even expanded in 2019.The fight between hybrid applications and local applications is ceaseless. Both have their very own majors and minors. In any case, frameworks of hybrid platforms are really giving an extreme challenge to the local applications. Individuals have consistently thought little of hybrid application development's incredible usefulness and speed. The efficiency of the hybrid application frameworks is expanding radically.

My research paper portrays different Cross-platforms for mobile application development and which are the best three platforms utilized in the market with their Advantages and Disadvantages. In addition, it gives examples of why hybrid application planning is superior to native.

**Cross Platform Application Development Frameworks**

Mobile applications have been with us for over ten years. Within that time, the mobile development market has essentially changed. In the starting, developers were restricted to native platform-specific applications which were not ready to take a shot at various frameworks. The developing interest for mobile applications brought new thoughts for shortening development time. What's more, that is the means by which cross-platform development was introduced. Near-native platforms, for example, React Native or Flutter, are making strides and offering better chances to fabricate native-like experience.

It is affordable and time-saver. It is easier and faster to deploy as Cross-platform application developers don't have to get familiar with numerous technologies before making their applications; they simply need to ace a couple.

**Some of the Hybrid Frameworks:**

* Apache Cordova

Adobe PhoneGap was recently known by the name Apache Cordova is a cross-platform framework that is possessed by Adobe. This framework uses HTML5, CSS, and JavaScript for the improvement and enables the clients to share the application which they are creating with the group so they can get dynamic input. In the event that the developer needs to construct the application straightforwardly, at that point, it additionally gives a cloud solution. It provides freedom over tangled content directions and access to the outsider tools and components. With the assistance of this mobile development framework, developers can see the adjustment in elements effectively.

* Ionic

Ionic is the most famous framework on the rundown for hybrid application development. It utilizes innovations that most developers know about, for example, HTML, CSS, and JavaScript. The Ionic framework accompanies a ground-breaking CLI which makes it conceivable to make, code, test and convey Ionic application to the platform of your accommodation. The mind-boggling results Ionic has furnished related to Angular has made it a simple pick for building hybrid mobile applications in the ongoing past. Its broad rundown of highlights, predefined components and a huge network of developers are only a portion of the things that made Ionic so mainstream. Because of its capacity to build native like mobile applications, Ionic is additionally an extraordinary decision for PWA improvement. version 4 of as of now in beta rendition, and the production variant is coming soon.

* Framework 7:

Framework 7 is an entirely free and open-source mobile HTML framework that permits creating hybrid mobile applications with local iOS and Android look and feel. This framework likewise helps in building fast application models, which can be a lifeline when you have to show the customers a rough idea of the project. Most likely, the best thing about Framework 7 is that it has no outside reliance on tools like React or Angular, in contrast to other hybrid frameworks.

* Sencha Ext JS:

Ext JS is the most dominant JavaScript-based application improvement platform. It enables you to make information concentrated HTML 5 applications utilizing JavaScript with unrivaled cross-program similarity. Ext JS framework enables us to manufacture endeavor application with brought together client experience utilizing JavaScript without composing any HTML 5 or CSS code. It incorporates worked in topics and UI parts out of the container, which limit advancement time and builds the profitability. Ext JS is a result of Sencha. It is a framework for web applications dependent on JavaScript and Html5. Sencha Touch is a superior HTML5 mobile application framework. Sencha GXT is the quickest, most dominant approach to make rich electronic applications utilizing Java.

* MonoCross:

It is a C# .NET Model-see controller (MVC) framework where the Model and Controller are shared crosswise over platforms, and the View is actualized for each particular platform as well as target design. It permits the improvement of both local and HTML5 web applications that offer business rationale and information code. The example empowers cross-platform transportability of business rationale and information access code, (Model + Controller), while supporting full, local, and additionally platform-explicit introduction. This allows partition of the presentation layer, (View) from the application code (Model + Controller). This adjustment empowers convenience of use code over numerous platform targets.

* Kony App Platform:

Kony is a cross-platform mobile application development framework. The fundamental component is to compose the code once and keep running on various platforms like Android iOs Windows Blackberry. It additionally bolsters various channels like mobile, tablet, work area web, mobile web. Kony utilizes JavaScript as front end improvement and Java for backend development. Visualizer is a useful tool made by Kony to plan the user interface. This item is these days sought after for mobile application advancement as a result of its code reusability and easy to create mobile application.

* Convertigo:

Convertigo MADP is an Open Source Mobile Application Development Platform to help Enterprises to create and run cross-platform mobile applications associated with big business information back-closes. Convertigo would mobile be able to empower any endeavor business application, and too, blend and join information and procedures from a few venture applications to a huge assortment of mobile devices.

* Appcelerator:

It is an enterprise-focused development platform that gives developers a chance to compose JavaScript with its Alloy MVC framework. The utilization of JavaScript is useful, the same number of developers will as of now be acquainted with the language, diminishing the need to invest energy getting to grasps with new code. Appcelerator additionally offers a virtual private cloud choice for big business clients, which is extraordinary news on the off chance that you work with touchy organization information. The undertaking item likewise accompanies investigation, execution of the board, and observing and committed foundation. In any case, you'll need to reach them straightforwardly for valuing.

**Native vs Hybrid:**

Cross platform applications or Hybrid applications are a better choice than a native application because of the following factors:

* Budget Friendly:

The primary motivation to pick cross platform application development over native application development is its cost-adequacy.

* Ease of maintenance:

As the hybrid apps are built on the same codebase, so it is easy to maintain.

* Minimum Requirements:

HTML, CSS, and JavaScript are the main languages required to develop completely functional Hybrid mobile applications and web applications. Despite what might be expected, the native application development approach utilizes much more languages, similar to Java, Swift, and Objective-C.

* Offline Support:

In any event, when the application isn't connected with the web, hybrid applications can work splendidly disconnected. Users can load the application and access the recently loaded information.

A screenshot of text

Description automatically generated

This picture represents different parameters and how they differ for both cases. The source of the picture is <https://bit.ly/2InsLkx>.

**Top 3 Cross platform Frameworks:**

1. **React Native:**

React Native is an open-source, cross-platform mobile application development tool developed by Facebook. It follows "learn once, write everywhere" protocol. JavaScript is the base language which offers a chance for native functionality. Once coded, you can move your application to both Android and iOS using the same base code.

The functionality that helps React Native to be different is that, even though they are cross-platform, apps built in this platform will have a look of a native application, and will offer an interactive user experience.

It is a framework that enables quick prototyping. Moreover, it is very easy to learn the basics and get started in this framework if you are skilled in JS. While this sounds incredible, the codes in JavaScript use a great amount of memory for calculation-intensive tasks. It is a excellent choice for mobile development and turns out to be more effective in many cases

**Pros and Cons:**

**PROS:**

* FASTER TO BUILD: The significant selling purpose of React Native is the shorter development time. The system gives various prepared to-apply segments that can quicken the procedure. React Native still does not have a few solutions, so you might need to construct them from scratch, yet React Native is based on JavaScript, which offers access to the biggest package ecosystem available in the market, npm. Having access to such an immense base of packages, you can spare a lot of time, and it will show signs of improvement. As the RN community group is developing, and Facebook normally presents new updates, we may discover prepared components for the greater part of the problems we need one day.

A lot of savings can occur with developing for more than one platform. As RN lets us share a big part of a codebase between operating systems within some hours, you can save more time and money with react native.

* ONE PLATFORM,MULTIPLE PLATFORMS: React Native allows us to reuse the codebase or just a part of it multiple times between iOS and Android. Practically, complete cross-platform development is achievable to some degree, depending upon how many native modules you have used in your application. Some of the features will be available in npm packages, but others will need to be written from the scratch. It will definitely get much better in the future. React Native community actively supports the framework and adding new tools to its open source.

Developing applications with JavaScript also gives an opportunity to share the codebase not only between mobile platforms but also with React web applications. Active development of tools like React Native Web appears to lead in this direction. This also allows the developers to work on both web and mobile apps, as the technologies are very related. Whereas the stability of such solutions still leaves a lot to be aspired, the feasibility of sharing non-UI dependent code may still bring cognitive benefits. Besides shortening the development time, it could help promote the consistency of your app’s business logic among all supported platforms.

* SMALLER TEAMS: Native development needs two separate units for Android and iOS. It can hinder the communication between developers and, accordingly, stall the development. Native iOS and Android teams have their own projects, and they often have varying development rates and plans. As a result, two projects can easily become completely different and inconsistent. Due to this, the developed application may have different behavior, features, and user interface. If you pick React Native, you will necessarily need JavaScript developing team who can write and implement the code for both platforms iOS and Android. Obviously, if you want to build an application with more native features in it, then you would definitely need help from native developers of the required platforms. That said, in most cases, the team will be smaller in size, and thus easier to manage.
* HOT RELOADING: Because of hot reloading, a developer can keep the app running while implementing the latest versions and improving the UI; this makes changes immediately visible without the need for the developer to rebuild the application. It is a significant advantage for app developers. Not only do they save time on the compilation, but they also don’t lose any state of the app while making changes. It boosts productivity and reduces application development time.
* SPEED OF APPLICATION: Many people say that implementing the application in React Native reduces the application’s performance. JavaScript cannot be as fast as native code, but you will not see any difference in most cases, anyway. A simple application, written in React Native and Swift. Both apps achieve very similar performance results. Discrepancies in performance are sparse, nearly unnoticeable to an ordinary user. When you try to develop more complex applications, then the framework may be less efficient. However, you can always shift some code to a native module, and it will not be a problem.
* SIMPLE UI: React Native is firmly based on creating a mobile User Interface. In native development, you will need to build a series of actions in the application. React Native uses declarative programming in which such an order of implementing actions is no longer used. Because of this, it is very easier to detect bugs in the ways a user can take.

**CONS:**

* LESS SMOOTH NAVIGATION: React Native has been tested in a production environment by many giants such as Facebook, Skype, but it's still in a beta phase. There are regularly happening changes in tools and dependencies among versions. React Native developers usually stumble upon many weird issues with hot reloading failures, debugging tools, or the compatibility of packages, which can hinder the development significantly, mainly for amateur developers.
* LACK OF CUSTOM MODULES: Despite its maturity, React Native still needs some more components. Rest of them are underdeveloped. The possibilities are you will not have a problem with it, as the majority of custom modules you may need are accessible, well-documented, and running correctly. Still, it might appear that you will have to develop your solution from scratch or attempt to hack an existing one. When developing your custom modules, you could end up with three codebases (React Native, Android, and iOS) for a component rather than only one. In all of those codebases, there can be variations in behaviour and user interface. Luckily, those states don’t come frequently.
* STILL NEED NATIVE DEVELOPERS: If you want to implement native features and modules in your applications, then detailed knowledge of a particular platform is required. The absence of outstanding support for many native app functionalities (e.g., camera, push notifications) used to be a vital issue with React Native development. As the community expands, more and more open-source libraries give clear access to native platform features. Nonetheless, the implementation of some more high-level features may still need help from native platform developers. The input of native developers depend on the complexity of your project, but you need to negotiate for them when starting with React Native. This may be a problem for small companies, in which application developers don’t have any native mobile experience.

1. **Xamarin:**

Xamarin, developed by an independent organization in 2011, was acquired by Microsoft in 2016 and is now controlled by the Redmond giant. This mobile app development technology is a little different than React Native because Xamarin’s C# code is directly compiled into native languages like Android and iOS. This results in it being a bit more challenging to get started with than React Native. In Xamarin code is written in C#, so developers need to have knowledge of this language, which is stable, but also more troublesome to learn than JavaScript. Same as with React Native, some experience of native tech is needed for more high-level projects. Xamarin’s main advantages include excellent performance (apps run fast), access to platform-specific APIs (full hardware support), and fewer delays in introducing platform updates when compared to React Native.

**Pros and Cons:**

**PROS:**

* CODE SHARING: The likelihood to construct single-code answers for iOS and Android (just as different stages) is the principle selling purpose of Xamarin and comparable systems. While a 100% shared codebase between altogether different platforms is yet to be accomplished, Xamarin is one of the best choices in this office, giving 60% to 95% of reusable code with local execution, UI, and controls, all written in C# inside the .NET system.
* SUPPORT BY MICROSOFT: Regardless of regular bias toward partnerships, the way that Xamarin is a piece of Microsoft's product improvement group puts it above numerous contenders on account of Ensured ceaseless designer support. Wide learning chances. Steadiness and execution, particularly for items like Test Cloud. The majority of this is understandable in light of the fact that Xamarin has the support of one of the greatest tech mammoths on the planet. Despite the fact that Microsoft formally quit supporting Windows Mobile a year ago, Xamarin is as yet thought to be a top-need item with no possibility of it being resigned or relinquished at any point in the near future.
* FLEXIBILITY OF C# AND .NET: Xamarin's primary programming language is C#. In case you're with an advancement organization that has its fingers in numerous generation pies simultaneously, odds are you're as of now utilizing this language in some limit. Given the idea of C# and .NET, just as Microsoft's in general .NET for everything stand, a solitary advancement group ought to have the option to effortlessly switch between applications for different stages by plan.

Considering this attitude of the extensive .NET advancement environment and the way that nowadays essentially all undertaking level improvement organizations have C#/.NET authorities ready, deciding on Xamarin can regularly be just legitimate.

In the event that we talk about developing for enterprises explicitly, over 80% of code overall are reusable between iOS, Android, and different stages in Xamarin and C#. This implies lower improvement expenses and greater adaptability for your group. While some can consider C# a detriment, its adaptability and the basic truth that you can do nearly anything in one language while never expecting to contact Objective-C, Swift, or Java, effectively exceeds the disadvantages.

* MINIMUM TIME TO MARKET: Time to market is fundamental when you build up an exceptionally aggressive business application, or you're on a tight cutoff time. With Xamarin's code sharing abilities and Test Cloud administration, you can without much of a stretch run tests through 2,000 devices and rapidly find out pretty much all conceivable code irregularities inside your application. You shouldn't belittle the significance of this chance, mainly when you have restricted equipment and human resource.
* FAST GUI PROTOTYPING AND CODE SHARING WITH Xamarin.Forms: Xamarin.Forms is at present, perhaps the most straightforward approaches to build cross-platform UIs that are agreeable with the rules. One of the numerous advantages, beside every single local component and controls support, is C# and rearranged XAML coding, with no compelling reason to take into account a particular platform.

Utilizing Xamarin.Forms can drive the rate or reused code up to an incredible 95%, giving a definitive code sharing knowledge. This skyrockets improvement speed and drives down expenses to a large degree, making the "compose once, use all over the place" mantra significantly closer to the brutal reality. Xamarin.Forms continues showing signs of improvement, as well, with the fourth form turned out in May 2019, bringing various redesigns. These incorporate Shell with new alternatives for streamlined advancement with a completely envisioned chain of importance, shared route, and coordinated inquiry choices.

* CODE MAINTAINANCE: Keeping up an application in a working condition is particularly dubious when you have more than one stage included. Each and every utilitarian change must be engendered through different stages. Xamarin is an unequivocal preferred position here, given compatibility of codebase between iOS, Android, and different stages.

This diminishes the time required to look after, investigate, or update code when important. The cross-stage experience empowered by Xamarin can spare you hours and days that could some way or another have been spent on overhauling each application for every stage independently. Subsequently, everything keeps awake to-date and time and cash get spared, particularly on the off chance that you re-appropriate to an outsider engineer.

**CONS:**

* PRICING: The main drawback of Xamarin is the cost. All alone, Xamarin comes free and open source as a piece of the .NET stage. Business improvement, however, calls for utilizing the Microsoft Visual Studio integrated development environment (IDE); and its licensing doesn't come cheap. The yearly expenses for the Microsoft IDE aren't an issue for an expert programming improvement organization, yet it tends to be an interesting point when going with in-house advancement.
* PLATFORM SPECIFIC LIMITATION: Regardless of how close Xamarin gets to local improvement when building single-code answers for numerous stages, a few constraints will undoubtedly surface. For instance, some workarounds are vital for multi-touch usefulness and platform-specific gestures.

These have been watched for a very long time, so designers brought it into their very own hands given Xamarin's open-source nature. You'll discover bunches of Xamarin holes filled gratitude to developers who became weary of sitting tight for official support.

From one viewpoint, things like that are extraordinary as the network individuals are extending the library. Then again, if there's a function that you need in the application and it's not bolstered, you'll need to define it yourself.

* TECHNOLOGY AND IDE LOCK-IN:

In the event that one day your organization changes its course to working with various toolsets, IDEs or dialects, you're in for some harsh time. Any code written in Xamarin must be utilized inside the.NET advancement stack. This implies you can't reuse it for a local application or for any HTML5 applications, which can constrain code life span and ease of use.

1. **FLUTTER**

This is a Google-possessed portable application SDK (software development kit) created as a total package with widgets, tools, and frameworks to simplify the way toward building and conveying versatile applications on the two iOS and Android stages. As a cross-platform development stage, it implies that you don't have to isolate the coding for your Android or iOS stages. You basically utilize one codebase for the two platforms. Additionally, it likewise has the accompanying qualities:

It is free and open source, Based on Dart, Design is based on reactive programming.

**Pros and Cons:**

**PROS:**

* FAST CODE WRITING:

For developers, Flutter proposes quicker and increasingly powerful portable application development. We can make changes in the code and see them straight away in the application! This is the purported Hot reload, which generally just takes (milli)seconds and assists groups with including features, fix bugs, and test quicker.

Hot reload is additionally entirely agreeable in developer-designer cooperation when we need to improve or explore different avenues regarding an application's look and check the consequences for the spot. At the end of the day, with Flutter, your designer or analyzer can cooperate with the developer on the UI, making changes – for instance, "Put it 2 pixels right" or "Make the liveliness quicker" – and see them right away.

* CODE REUSE:

Developers compose only one codebase for your 2 applications – covering both Android and iOS platforms. Flutter doesn't rely upon the stage, since it has its own widgets and designs. This implies you have the equivalent application on two platforms. However, what's significant is that, in the event that you need to separate your applications – it's conceivable.

* LESS TESTING:

In the event that you have the equivalent application for 2 platforms, it implies less testing! The Quality Assurance procedure can be quicker. Due to one codebase, the developers compose programmed tests just once. Furthermore, Quality Assurance authorities have less work to do, in light of the fact that they have only one application to check. Obviously, if your applications have a few contrasts, they should be tried on the two platforms.

* FASTER APPS:

Flutter applications work smoothly and quickly, without hanging and cutting while at the same time scrolling over.

* NEW DESIGNS:

Flutter is designed to make it simple to make your own widgets or tweak the current widgets. You can peruse a list of Flutter's widgets and view, for instance, Material Design widgets and Cupertino widgets.

* EASY TO SETUP:

Flutter is anything but difficult to set up and start coding on low-end machines. You should not be stressed that your operating system won't have the option to deal with Flutter. A remarkable opposite, Flutter doesn't require an incredible machine.

**CONS:**

* LESS LIBRARIES FOR NATIVE DEVELOPMENT:

Google support for Flutter is excellent, and there are numerous useful libraries with functionalities fit to be actualized. Be that as it may, Flutter is still new, and few out of every functionality that you require can be located in these libraries. This implies your developers would need to construct them independent from anyone else, which can be very tedious.

* INTEGRATION SUPPORT:

Flutter it isn't broadly bolstered by CI platforms like Travis or Jenkins. Along these lines, to accomplish programmed building, testing, and arrangement, your development group needs to utilize and keep up custom scripts.

* MOBILE ONLY:

Flutter takes into account portable applications just and isn't bolstered by internet browsers. At the point when you pick the technology stack, you need to work with, this can impact your choice. If you need most extreme device adaptability, at that point, Flutter may not be the thing you're searching for.

**Some of the Applications based on hybrid are:**

* UBER:

Indeed, the mobile application you are at present, utilizing to make your trip agreeable and simple to access, is one of the most mainstream hybrid application models. The simple navigation and simple UI make this application very well known among its rival applications.

* INSTAGRAM:

Embracing the hybrid approach has enabled developers to build an application that supports both offline data and rich media. The best case of this is the short videos. The application gives the client a chance to access the media in any event, when they are disconnected or possibly shows an error message.

* EVERNOTE:

Evernote is the best guide to show that performance isn't an issue now for hybrid applications. Notwithstanding its excellent design, Evernote is very useful in driving productivity.

**React Native vs Flutter vs Xamarin:**

* APP SPEED:

Application speed is constantly a worry for the application proprietors so it must be brisk and responsive. Flutter offers quicker application advancement with quicker application speed at the yield. Different platforms like React Native and Xamarin additionally performs better in the application speed however while contrasting and the Flutter they are somewhat more slow. The application speed likewise relies upon the improvement procedure, capacities, highlights, and different terms so it's difficult to look at on the moment yet we can finish up the Flutter as a quicker cross-platform application advancement framework.

* SECURITY:

Security of the source code and application is an unquestionable requirement thing that is the reason security is recorded in the need activity for all cross-platform developers. As referenced before, React Native helps outsider modules and not have strong security bolsters. Respond Native lose the race of security, so how about we talk about different cross platforms. Flutter and Xamarin based applications are powerful and good with security access.

* USABILITY BY DEVELOPERS:

The Xamarin depends on the C# that is simple and simple to gain proficiency with the language. Flutter uses Dart, an object-oriented programming which is additionally simple to learn for amateur developers. React Native coded in Java that is most generally utilized in any development so taking a shot at Java is easy for each developer. In the compact, Xamarin and React Native are the most effortless to access cross-platform application development framework.

* POPULARITY:

Xamarin is the most famous cross-platform in the rundown, approx 1.4 million developers use Xamarin till the principal quarter of 2017. Because of its capacity to develop native UI code for application development, numerous well-known brands utilize the Xamarin for creating their application. Xamarin creates native codes however it isn't free, so it is considered for the excellent clients. As a large portion of the developers use Xamarin yet at the same time numerous developers depend on the Flutter and React Native.

**Summary**

Nowadays, big organizations don't want to burn through the more significant part of their resources and time on developing numerous applications using native application development. Due to this, Cross-Platform frameworks are coming into the light these days as they stand apart among every single other structure as far as resources and time it needs to create applications. These systems offer extraordinary movability if the right framework is picked by the developing group as indicated by the prerequisite of the application they are creating, and the structure that conveys great and convincing UI highlights. Utilizing cross-platform applications keeps away from regular issues that developers face while building different mobile applications for Android and IOS. It diminishes the pressure that the beginner developers feel. Picking the Hybrid Platform has one more included favorable position, which is that developers don't have to make numerous editions for similar applications on various platforms.

Likewise, with the expanding interest for Cross-Platform framework, we don't have to take in everything without any preparation. Cross-platform advancement devices can spare you a ton of time during your development cycle and assist you with arriving at the most significant number of clients by distributing your application on various platforms utilizing minimal measure of resources. They are certainly not flawless, in any case, and will never turn into a substitute for native application development. There will be exchange offs, yet you win a few, and you lose a few. In any event, when building up your application utilizing a cross-platform improvement device, you are looked at with such a significant number of various tools and frameworks that it very well may be challenging to settle on which one to go with.

An ideal approach to settle on the cross-platform improvement platform that works best for you is to attempt them all and go with the one that works for you. In the event that you don't have the opportunity, which usually is the situation, a little research on a portion of the top cross-platform improvement frameworks can be adequate. Gauge the upsides and downsides of each, and see which one suits your needs, and acknowledge the way that you'll need to take the great and the terrible.

**Citation:**

1. <https://www.netsolutions.com/insights/cross-platform-app-frameworks-in-2019/>
2. <https://instabug.com/blog/cross-platform-development/>
3. <https://www.businessofapps.com/guide/cross-platform-mobile-app-development/>
4. <https://www.quora.com/What-is-the-Kony-framework>
5. <https://marketplace.eclipse.org/content/convertigo-mobile-application-development-platform-enterprises>
6. <https://en.wikipedia.org/wiki/MonoCross>
7. <https://codeburst.io/native-app-or-hybrid-app-ca08e460df9>
8. <https://www.excellentwebworld.com/hybrid-app-examples/>
9. <https://www.thedroidsonroids.com/blog/flutter-in-mobile-app-development-pros-and-cons-for-app-owners>
10. <https://hackernoon.com/flutter-pros-and-cons-for-seamless-cross-platform-development-c81bde5a4083>
11. <http://blog.thinkwik.com/why-choose-google-flutter-for-mobile-app-development-in-2019/>
12. <https://www.altexsoft.com/blog/mobile/pros-and-cons-of-xamarin-vs-native/>
13. <https://www.iflexion.com/blog/xamarin-pros-and-cons>
14. <https://www.netguru.com/blog/react-native-pros-and-cons>
15. <https://csform.com/best-cross-platform-mobile-frameworks-for-2019/>
16. <https://www.acquaintsoft.com/flutter-vs-react-native-vs-xamarin-which-cross-platform-better/>