



INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

FLUTTER VS NATIVE PLATFORM

Ashish Rawat¹, Brijesh Chauhan²

MCA student, MCA student,
Uttaranchal University, Dehradun, Uttarakhand

ABSTRACT: We looked at both flutter (a single codebase that can be used on different platforms) and native (a single codebase that can be used on multiple platforms) (multiple codes for each platform). Analyse the speed and performance of a flutter application as well as native Android and iOS applications. The conclusions are based on a comparison of many activities over time to assess if flutter technology is appropriate for mobile app development. In some cases, native programmes outperformed the flutter technique, but the flutter strategy saves massive development time without compromising user experience in others.

Introduction

The global smartphone applications approx. the market value was \$106 billion in 2018 and projected growth at a CAGR is 18.4% from the year 2019 to 2026. Android-powered 70.75% of smartphones, while the iOS market is 28.53% and 0.38 % Samsung, 0.14 % kaiOS, 0.02 % of Nokia, 0.13% of unknown [1].

Every mobile app developer wants their apps to reach as many users as possible. Apps must be designed separately for each platform due to the nature of the mobile operating system industry and the issue of platform compatibility.

Mobile apps that need to function on a variety of platforms have various source codes depending on the platform. A native application is an app that has access to the device's and is designed for a specific platform. Because the source code of different platforms must be replicated, developing native software for many platforms takes longer and consumes more resources, potentially increasing project expenses. Developer tools that allow them to construct a single code base and deploy across several mobile platforms. One of these types of apps is a flutter app, which has the advantage of streamlining the app life-cycle by centralising development in a single code base. Flutter has two drawbacks: access to device resources and excessive memory usage.

“As the community releases new versions of the Flutter SDK, Flutter applications are no longer experiencing slow performance or crashing issues.”

Using a rapid flutter application development tool, you can create multiple apps with a reusable codebase (90%) that can leverage native UI

and access smartphone functionality, control elements, and adhere to the Interface and Material Design criteria [1].

Choosing between flutter and native apps is frequently a matter of cost, effort, and development methodology. Flutter is an open-source user interface toolkit that leverages the Dart programming language to create mobile apps with a single code base that can be compiled for both Android and iOS platforms. Flutter SDK was launched by Google in 2018 and, according to Google, allows developers to create applications that have the same appearance and feel as if they were designed as native mobile apps. The research report compares the performance of the Flutter application to that of native programmes in numerous areas, including CPU performance, code complexity, graphics, and the code required to do its mission.

Native and flutter application

Mobile applications have specific development characteristics due to software distributions, hardware features, and release cycles. Developers must deal with a variety of issues such as storage capacity, mobile specifications, user experience, security, and privacy [2].

Native apps are designed specifically for a particular OS, such as android or iOS, and cannot be used on other platforms. These apps are typically found in their respective application stores such as the App Store for iOS and the Google Play Store for Android applications), which are accountable for installing and making the application available to the customer [3]. Native applications are more likely to match the platform's UI (feel and look) and technology patterns. This type of app has full access to the device's sensors, cameras, and GPS [4]. This method provided consumers with both great performance and a customized UI/UX. It does, however, necessitate the use of a specialized platform foundation and SDK [5].

Flutter technology provides an alternative to designing an app for many platforms. This method results in a single code base that can be used to create apps for a variety of platforms [2].

Flutter applications combine native and web-based functionality. By merging mobile and web capabilities, this method allows developers to target numerous mobile operating systems from a single code base and create an app that operates similarly to a native app. Apps that run on both iOS and Android are known as hybrid apps [2].

Analyzing the performance of flutter and native programmes

To examine the performance of native and flutter apps, an analysis was carried out to verify the execution time of selected jobs in native and flutter apps. The Android To examine the performance of native & flutter apps, an analysis was carried out to verify the execution time of selected jobs in s was built in the Flutter environment.

The exam was broken into four halves. Time is the most important aspect determining the experience for users that arises from the use of mobile apps.

1. **numerical test** – This test measures the time it takes to compute a number 0 to 1000 decimal points.
2. **data storage and retrieval from a folder and file** – the test that assesses reading and writing durations in a file that contains the number computation result from the previous test.
3. **network test** – The time it took to download a 7 MB file from the Internet was assessed in this test. Both applications in the research use the same file, which is stored on a publicly accessible server. The Internet connection for all of the smartphones was the same.
4. **selection of a location** – the time-based test for determining the latitude and longitude of a mobile phone the test was conducted on both devices with default settings due to the location of mobile devices can be determined using GPS [6].

On the same cellphones, applications are used. It was the Xiaomi for Android and the iPhone simulator 8 pro for iOS. The apps were tried on recent OS available to the devices to take advantage of all of the device's features. It was Android version 9.0 "Pie" for Android and iOS version 15.2 for iOS. The "release" compiler setting was used to create both apps, which were then mounted on the devices. For both systems, every test was repeated two to three times, and the network tests were conducted using a similar internet connection. The location tests remained all done in a similar place.

Analytical findings

Each test was broken down into paragraphs in the analysis.

Calculating efficiency

The outcomes of the computational performance trial are shown in Table 1. Native applications outperform flutter applications on both platforms. On Android, however, the

The difference between native and flutter app outcomes is even bigger.

Table 1 displays the findings of the study on calculating efficiency.

Average computing time		Android[s]	iOS[s]
	Native	5.515	4.315
	Flutter	10.515	6.216

Read and write of file access

The outcomes of files read and write is shown in Table 2. Android native app was found to be the fastest in the read file test, while in other tests, both flutter programmes take a long time, but the findings are in milliseconds, so these differences may not be significant in real-time use.

File Access Time time		Android[ms] read	Android[ms] write	iOS[ms] read	iOS[ms] write
	Native	26.000	2.722	0.714	16.945
	Flutter	5.114	3.515	1.484	18.475

Network Test (Image downloading)

The outcomes of the connection performance trial are shown in Table 3. All of the applications yielded comparable outcomes. Temporary fluctuations in the connection bandwidth cause them.

downloading time		Android[s]	iOS[s]
	Native	6.845	6.589
	Flutter	6.786	6.435

selection of a location

The outcomes of the period to fetch location experiment are shown in Table 4. Android devices get a location in a few seconds, while iOS takes a few secs longer. Native applications, as in prior studies, produced the best performance.

Average Fetch Location time		Android[s]	iOS[s]
	Native	3.196	12.109
Flutter	6.000	14.636	

Survey

In our company, NeoSoft Technology and Cynoteck Solutions, we surveyed native and flutter technology in the mobile development department.

In terms of the IDE/Framework, 80% of developers said they had previously used an IDE with Android Studio, whereas 50% said they had no prior experience with Visual Studio or XCode.

According to developers, developing a Flutter app is 40-50 % faster than developing a native app, and maintaining two codebases requires more effort. In addition, the hot reloading feature rapidly resolves the issue in the program.

According to the developers, Flutter community support is just as good as native community support. The performance of a native app is superior to that of a flutter app, although the flutter community releases an update every 2–3 months. In 1–2 years, the performance of flutter apps will be comparable to that of native apps.

Regarding Developers shifting native to flutter, according to the developers, 70 to 80 % are switching from native to flutter development approach due to fast development, compiling and single codebase.

Conclusion

We investigated the elements that influence the decision to design a mobile application using a native platform or a flutter strategy in this study report. Our findings can aid developers in systematising the selection of the best framework for a mobile project. We performed surveys of freshers, intermediate, and expert mobile developers in our firms to answer the main research topic. The data analysis revealed that practitioners value both technical and nontechnical variables such as technological support (documents, wiki, and community), end-product performance, project needs, and learning curve, independent of their experience.

In terms of technical issues, we looked into the problems and benefits of native and flutter development. We experimented with developers who used both native and flutter to implement the functionality. According to our findings, practitioners are strongly predisposed to use a native strategy when it comes to achieving high performance (i.e., low battery consumption, low latency, high graphical performance, low memory usage). flutter, on the other hand, is a superior alternative for designing flutter mobile apps if performance isn't an issue because of its speedy development and hot reloading capabilities.

References

1. Andrei-Klubnikin Medium: cross-platform-vs-native-mobile-app-development, <https://andrei-klubnikin.medium.com/cross-platform-vs-native-mobile-app-development-choosing-the-right-dev-tools-for-your-app-project-47d0abafee81>
2. Paulo Meirelles, Carla Rocha, Felipe Assis, Rodrigo Siqueira of A Students Perspective of Native and Cross-platform Approaches for Mobile Application Development. https://www.researchgate.net/publication/334086675_A_Students'_Perspective_of_Native_and_Cross-Platform_Approaches_for_Mobile_Application_Development
3. Malavolta, I.: Beyond native apps: web technologies to the rescue! (keynote). In: Proceedings of the 1st International Workshop on Mobile Development, Mobile! SPLASH 2016, Amsterdam, Netherlands, October 31, 2016. pp. 1–2 (2016)

4. 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 (2015), <http://www.sciencedirect.com/science/article/pii/S2090447915001276>
5. Latif, M., Lakhri, Y., Nfaoui, E.H., Es-Sbai, N.: Cross-platform approach for mobile application development: A survey. In: 2016 International Conference on Information Technology for Organizations Development (IT4OD). pp. 1–5 (March 2016)
6. Epstein Z.: September data shows the 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).
7. Pawel Grzmil, Maria Skublewska-Paszkowska, Edyta Lukasik, Jakub Smolka: Performance Analysis of native and cross-platform mobile applications. <https://www.researchgate.net/publication/320039474>

