MITRAIS

continuous commitment

MITRAIS WHITE PAPER
GOOGLE FLUTTER COMPARISON WITH NATIVE MOBILE DEVELOPMENT

VER.1

Table of Contents

Table of Figures

1. Overview

This White Paper was developed to evaluate Google Flutter (a Google's mobile UI framework for developing high-quality native interfaces on iOS and Android in record time) against native mobile development (Android and iOS Swift).

We decided to publish this evaluation to provide early information of Google Flutter that might suit any prospective client(s) and for internal documentation.

2. Evaluated Platforms

	PLATFORM	VERSION
1	Google Flutter	0.3.5 beta 2
2	iOS Swift native	Swift 4
3	Android Java native	Java 7 / 8
4	React Native	0.55

Table 1 – Evaluated Platforms

3. Other Platforms

Other platforms which are not evaluated but may be included in the future are listed below:

- iOS Objective C native
- Android Kotlin native

4. Evaluation Criteria

The evaluation criteria are based on "Mobile Web-App Framework Evaluation Standard" that were proposed by Heitkotter, et all (2013). The evaluation consists of two points of view. First is the standard from the developer's perspective and the second is the standard from the user's perspective. There are seven standards from the developer's viewpoint and four standards from the user's viewpoint.

The developer's viewpoint standards are:

1. **License and Costs**
   Does initial cost occur to introduce framework?
2. **Long-term Feasibility**
   Is it a framework that can be managed and used continuously?
3. **Documentation and Support**
   Is it well documented and supported for a developer?
4. **Learning Success**
   Is it a framework where the concept is already familiar to the developer?
5. **Development Effort**
   Does a configuration such as development environment help a developer minimize the effort of development?

6. **Extensibility**
   Is it possible to extend a framework?
7. **Maintainability**
   Are source codes well modularized?

The user's viewpoint standards are:

1. **User Interface Elements**
   Is UI composition optimized for a mobile application?
2. **Native Look and Feel**
   Does the framework give the same experience that native application does?
3. **Load Time**
   Does it provide the same loading time as a native application?
4. **Runtime Performance**
   Is its response time short and of high performance?

NB: This document uses a weighting schema which is currently relevant for Mitrais use.

4.1. Developer's View Point Evaluation Criteria

CRITERIA	MITRAIS WEIGHTING
License and Costs	40
Long-term Feasibility	40
Documentation and Support	30
Learning Success	40
Development Effort	50
Extensibility	30
Maintainability	40

Table 2 – Developer's View Point Evaluation Criteria

Criteria used:

1. **License and Costs**
   Costs for obtaining a framework and employing it in commercial apps influence whether a framework is suitable for a certain app or a particular company. Hence, this criterion examines licensing costs that accrue for developing and publishing a commercial app based on the respective framework.
2. **Long-term Feasibility**
   The decision for a framework represents a significant investment because specific know-how needs to be acquired and source code of apps will be tied to the framework. Hence, developers will prefer a framework that will most likely be available in the long term. A framework needs continuous updates, especially in view of rapidly changing browsers and Web technologies. Indicators of long-term feasibility are popularity, update behaviour, and the development team. Popularity can be assessed through a high diffusion rate among app developers and recognition in the developer community, for example through reviews. A positive update behaviour is marked by short update cycles and regular bug-fixes. A framework with a strong development team, ideally backed by several commercial supporters, is more likely to continue to exist in the future.

3. **Document and Support**
   Documentation and further support channels assist developers in learning and mastering a framework. Assistance is not only required when starting to use a framework, but also to efficiently employ its API and advanced concepts. Therefore, a documentation of good quality provides tutorials and a comprehensive, well-structured reference. For popular frameworks, textbooks might provide a good starting point. Besides, other means of support such as community-driven forums or paid assistance help in case of special problems.
4. **Learning Success**
   Time and effort needed to comprehend a framework directly affect its suitability. While good documentation may enhance learning success, learning inherently depends on the inner characteristics of a framework, i.e., its accessibility and comprehensibility. Hence, the learning success is examined separately. It mainly depends on the subjective progress of a developer during initial activities with a framework. Intuitive concepts, possibly bearing resemblance to already known paradigms, can be mastered quickly. To a minor extent, this criterion also considers the effort needed for learning new concepts after initial orientation.
5. **Development Effort**
   The cost for developing apps mostly depends on the development effort needed, assuming a basic familiarity with the framework. While certain development phases such as requirements elicitation or design are largely independent of the framework used, it directly influences the implementation. Hence, the development effort is characterized by the time needed for implementing apps with the framework. Indicators for a framework that ease development are expressive power, an easy-to-understand syntax, reusability of code, and good tool support. The latter includes an Integrated Development Environment (IDE), which facilitates implementation and possibly GUI design, as well as debugging facilities.
6. **Extensibility**
   In view of both evolving requirements and a changing environment, it may be necessary to extend a framework with additional functionality, either during initial implementation or in later iterations. This will be easier and more stable if a framework offers corresponding features such as a plug-in mechanism. As a last resort, app developers might adapt the source code of the framework itself, provided it is available. Besides considering the existence of extensibility measures, this criterion assesses their usefulness and accessibility.
7. **Maintainability**
   Mobile apps can and will be updated regularly. Therefore, their implementation must be maintainable over a longer period. This criterion is positively correlated with comprehensibility of the source code and its modularity. Both indicators depend on the framework used to implement the app. A framework that allows for concise but understandable code will improve comprehensibility. Modularity requires the possibility to separate different parts of an app into distinct units of code.

4.2. User's View Point Evaluation Criteria

CRITERIA	MITRAIS WEIGHTING
User Interface Elements	40
Native Look and Feel	30

CRITERIA	MITRAIS WEIGHTING
Load Time	40
Runtime Performance	40

Table 3 – User's View Point Evaluation Criteria

Criteria used:

1. **User Interface Elements**
   From an app user's perspective, elements of the UI should be well-designed and optimized for mobile usage. Hence, a mobile app framework needs to provide high-quality elements for important tasks. On the one hand, this criterion assesses whether a framework offers mobile versions of common structural elements, i. e., widgets such as buttons or text fields and their layout in containers, as well as their quality. Structural elements need to address limited screen sizes and particularities of touch-based interaction. On the other hand, a framework should support behavioural UI elements such as animations and gestures
2. **Native Look and Feel**
   User acceptance of a mobile app that is developed with Flutter, also compared to a native app, often depends on a native look & feel. In contrast to a typical mobile app with a native UI that has a platform-specific appearance and behaviour. As this is an often-mentioned requirement of apps, this criterion assesses whether a framework offers support for a native look and feel. Optimally, a framework would provide different, platform-specific themes, at least for Android and iOS. If that is the case, we examine how closely these resemble truly native UIs. Otherwise, the framework should provide means to efficiently style its UI elements and implement themes.
3. **Load Time**
   The time required to load a mobile app is important to users in view of slow and instable network connections on mobile devices.
4. **Runtime Performance**
   The performance at runtime (after loading) informs the overall impression of an app. The UI elements need to react quickly to user interactions, and animations should be smooth for a high-quality user experience

5. Platform Evaluation

PLATFORM	Developer's View Point Evaluation Score							User's View Point Evaluation Score				TOTAL SCORE
	1	2	3	4	5	6	7	1	2	3	4	
Google Flutter	360	280	240	280	450	270	280	320	240	320	320	3360
iOS Swift native	360	360	270	360	350	270	360	360	270	360	360	**3680**
Android Java native	360	360	270	360	350	270	360	360	270	360	360	**3680**
React Native	360	320	270	360	450	270	320	320	270	320	320	3580

Table 4 – Platform Evaluation

6. Framework Recommendation

6.1. Developer's View Point Criteria

6.1.1. License and Costs

SOLUTION	EVALUATION
Google Flutter	Google Flutter is an open source SDK and it is free. https://github.com/flutter
iOS Swift native	iOS Swift native is an open source programming language with Apache License Version 2.0 https://github.com/apple/swift. Like another open source, Swift has free public access.
Android Java native	Java is a programming language that is free to use.
React Native	React Native is licensed under MIT. https://github.com/facebook/react-native/blob/master/LICENSE

Conclusion:

- Google Flutter: 9
- iOS Swift native: 9
- Android Java native: 9
- React Native: 9

6.1.2. Long-term Feasibility

SOLUTION	EVALUATION
Google Flutter	Google Flutter was developed by Google and might have long-term feasibility. The first beta version was released on February 27, 2018 at Mobile World Congress 2018.
iOS Swift native	Swift is a successor of Objective-C that was developed by Apple. Swift reached the 1.0 milestone on September 9, 2014.
Android Java native	Java is the main language used to develop Android applications. Large parts of Android apps are written in Java and its APIs are designed to be called primarily from Java.
React Native	Developed by Facebook and released to the public (v0.5) on June 6, 2015. After that, at the beginning of each month, a new release candidate is created off the master branch on GitHub. Current version (March 2018) is v0.55.

Conclusion:

- Google Flutter: 7
- iOS Swift native: 9
- Android Java native: 9
- React Native: 8

6.1.3. Document and Support

SOLUTION	EVALUATION
Google Flutter	Although it's a new one, Google Flutter has good and well-structured documentation and also already has a Support team. We could not find any textbooks about Flutter for now. We think it is just a matter of time.
iOS Swift native	Swift has great documentation since it was released almost four years ago. There are also plenty of free online tutorials (text and/or video) of Swift. It also has large community support and some textbooks related to Swift have already been published. Of course, Swift has a support team from Apple.
Android Java native	Java Android has great support and documentation. We could find a number of learning sources, documentation, forums and textbooks about Java android.
React Native	Facebook creates great documentation, tutorials, blogs and discussion forums for React Native. https://facebook.github.io/react-native/docs/getting-started.html

Conclusion:

- Google Flutter: 8
- iOS Swift native: 9
- Android Java native: 9
- React Native: 9

6.1.4. Learning Success

SOLUTION	EVALUATION
Google Flutter	Flutter use a new language called Dart, previously it was not popular. Also it is still a bit hard to find Flutter best practice and examples. Sometimes we discovered bugs that are still in the process of being fixed. Google Flutter is now growing fast, and it should be easy to find best practice in a few months or even weeks.
iOS Swift native	IOS swift native is a language that was developed by Apple to support their own device product. Apple already has provided complete documentation and a tutorial for Swift. Since Swift is mature enough and used by a lot of iOS mobile developers, they have a large community, so we can ask or search for references on that forum. It helps us as developers if we are facing issues regarding Swift language.
Android Java native	Android Java Native uses Java language as its basis; it is one of the more well-known languages and has many learning sources. It has a very easy to follow tutorial and success develop first application.
React Native	React Native uses JavaScript code, CSS-like stylesheets and HTML-like tags for layout. It makes it easier to on-board a new developer with basic JavaScript knowledge to develop native apps quickly. React Native also provides a user experience that no other JavaScript based mobile solution has been able to provide before.

Conclusion:

- Google Flutter: 7
- iOS Swift native: 9
- Android Java native: 9
- React Native: 9

6.1.5. Development Effort

SOLUTION	EVALUATION
Google Flutter	With Flutter we would only need to build it once and it would already work in Android and iOS. It significantly reduces development effort to build an application in multiple platforms. Less boilerplate compares to native language. With Hot Reloading, we can even run new code while retaining the application state.
iOS Swift native	With storyboard in XCode, we can more easily create UI and manage the navigation for each page in iOS native application. Obviously, we cannot develop this as an Android app.
Android Java native	We can easily and flexibly customize an Android widget using Android Java Native. Obviously, we cannot develop an iOS app here.
React Native	With the principle of *learn once implement everywhere*, we can easily build mobile apps for iOS and Android super quickly and intuitively. React Native lets us build an app faster. Instead of recompiling, we can reload an app instantly. Similarly with Flutter, we can even run new code while retaining the application state.

Conclusion:

- Google Flutter: 9
- iOS Swift native: 7
- Android Java native: 7
- React Native: 9

6.1.6. Extensibility

SOLUTION	EVALUATION
Google Flutter	Flutter enables the creation of modular code that can be shared easily.
iOS Swift native	Swift supports modular application, so we can create common modules that may be able to be used by another application.
Android Java native	Android Java Native supports modular application. We can create plugins or customize packages for widget or any Android API.
React Native	React Native uses the same fundamental UI building blocks as regular iOS and Android apps. We just put those building blocks together using JavaScript and React. It's also easy to build part of an app in React Native, and the other part using native code directly. React Native combines smoothly with components written in Objective-C, Java, or Swift.

Conclusion:

- Google Flutter: 9
- iOS Swift native: 9
- Android Java native: 9
- React Native: 9

6.1.7. Maintainability

SOLUTION	EVALUATION
Google Flutter	Flutter is still in Beta Release and there will definitely be many updates and changes in the future. Based on experience, when we develop apps which are in beta version, it will be a bit hard to maintain the apps. Also Flutter offers no separation between templates, styles, and data.
iOS Swift native	IOS app that is developed using Swift is maintainable. When iOS or Swift have an updated version, Apple will provide complete documentation about their updates. So, as developers, it will be easier to maintain.
Android Java native	The maintainability will depend on coding technique. So far, every Android application that is developed using Java is maintainable. Because every update of Android SDK is well documented, it is not hard to maintain it.
React Native	React native has dedicated tools and documentation for updating application into newer versions. However some major versions have breaking changes and require a lot of manual work.

Conclusion:

- Google Flutter: 7
- iOS Swift native:9
- Android Java native:9
- React Native: 8

6.2. User's View Point Criteria

6.2.1.1. User Interface Elements

SOLUTION	EVALUATION
Google Flutter	Flutter has its own UI components, along with an engine to render them on Android as well as iOS platform. Most of these components conform to the guidelines of Material Design and offer complete sets of widgets, for example buttons, modals, forms, and even built-in navigators.
iOS Swift native	Has a full set of UI components for iOS.
Android Java native	Has a full set of UI components for Android.
React Native	React-native already has its own control or components commonly used in iOS or Android Apps like Navigation bar, Side Menu, Tab, DatePickers, etc.

Conclusion:

- Google Flutter: 8
- iOS Swift native: 9
- Android Java native: 9
- React Native: 8

6.2.1.2. Native Look and Feel

SOLUTION	EVALUATION
Google Flutter	Flutter's widgets incorporate all critical platform differences such as scrolling, navigation, icons and fonts to provide full native performance on both iOS and Android. Flutter has its own proprietary UI components, along with an engine to render them on Android (Material Design) as well as iOS (Cupertino) platforms.
iOS Swift native	Since Swift is the language to build the native app in iOS, it will produce native iOS apps.
Android Java native	Application development using Java with Android SDK will produce Native Apps, and it is the basic language. It is designed around Android's capabilities and conventions to give users the best experience.
React Native	React Native apps look and feel like they were custom-developed for the iOS or Android device.

Conclusion:

- Google Flutter: 8
- iOS Swift native: 9
- Android Java native: 9
- React Native: 9

6.2.2. Load Time

SOLUTION	EVALUATION
Google Flutter	Flutter produces native application but the load time is the same as Native. But it is not tested yet for complex and big applications.
iOS Swift native	Has good load time.
Android Java native	Has good load time.
React Native	React Native load time is relatively the same with its Native version.

Conclusion:

- Google Flutter: 8
- iOS Swift native: 9
- Android Java native: 9
- React Native: 8

6.2.3. Runtime Performance

SOLUTION	EVALUATION
Google Flutter	For a small application, it feels like code with Native language. Technically speaking, Flutter should be faster since there is no Javascript bridge for interaction with Native component. However, it is not tested yet to develop it for large and complex applications.
iOS Swift native	Has best performance for iOS App
Android Java native	Has best performance for Android Application
React Native	React Native application feels and performs smoothly like apps with Native language. However, some complex dynamic user interactions and animations still have performance issues in React Native.

Conclusion:

- Google Flutter: 8
- iOS Swift native: 9
- Android Java native: 9
- React Native: 8

7. Conclusion

We can see from the Platform Evaluation, that Android Java Native and Swift native are probably the best tool that will be appropriate for most developers to use for Native application development (iOS and Android), since it has the highest score followed by Google Flutter. Google Flutter is now in Beta version and has much room for improvement. Once it is in Release version, it will be a good choice to develop Native Applications for both Android and iOS with short time frames.

8. References

Flutter - Beautiful native apps in record time (no date). Available at: https://flutter.io/ (Accessed: 14 March 2018).

Flutter vs React Native Comparison for Q1 2018. Available at: https://agileengine.com/flutter-vs-react-native-comparison

Heitkötter, H. *et al.* (no date) 'Evaluating Frameworks for Creating Mobile Web Apps'. Available at: https://pdfs.semanticscholar.org/59e2/950d74d231eaa6889d346b3b7ba7823446d4.pdf (Accessed: 14 March 2018).

Sohn, H.-J. *et al.* (2015) 'Quality Evaluation Criteria Based on Open Source Mobile HTML5 UI Framework for Development of Cross-Platform', *International Journal of Software Engineering and Its Applications*, 9(6), pp. 1–12. doi: 10.14257/ijseia.2015.9.6.01.

Swift.org - Welcome to Swift.org (no date). Available at: https://swift.org/ (Accessed: 15 March 2018).

The Swift Programming Language (Swift 4.1): About Swift (no date). Available at: https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/index.html (Accessed: 15 March 2018).

Copyright

Copyright © 2018 Mitrais

Disclaimer

Any and all information in this document has been compiled and provided for information purposes only. The information provided herein may include information compiled from a variety of third parties. Mitrais will not be liable for any loss, damage, cost or expense incurred in relation to or arising by reason of any person relying on the information in this document or any link to any website provided herein, whether or not caused by negligence on Mitrais' part. While Mitrais endeavours to provide the information up to date and correct, Mitrais make no representations or warranties of any kind, express or implied, about the accuracy, reliability, completeness or currency of the information or its usefulness in achieving any purpose. Readers of this document are responsible for assessing its relevance and verifying the accuracy of the content. In any event, the information provided herein should not be construed as providing advice whether legal or otherwise.