zühlke
empowering ideas

Using Flutter

as a key framework for cross-platform front-end development

Zühlke January 2023

Figure 1: Google search results (worldwide) comparison (Source: trends.google.com)

What is a Flutter app?

Flutter is an open-source framework by Google. The underlying programming language is Dart, which was influenced by and has similarities to Java, C++, and C. It provides the ability to build natively compiled applications for the web, Windows, macOS, Linux, iOS and Android using one single codebase. Flutter is supported and used by Google and actively maintained by Google and a large community of open-source developers around the world.

Market prevalence of Flutter

There are currently over 500,000 apps[1] built with Flutter, and the number is increasing rapidly. Many large brands rely on Flutter, and it is a key framework for many of their applications. Brands covering different segments ranging from social apps like WeChat to financial and banking institutions such as Betterment and Nubank, lifestyle apps like Fastic, Tabcorp and trip.com, companion apps like BMW and even public institutions like the Brazilian government use Flutter for their services.

Not only have many brands made use of the Flutter framework and adapted it in their applications, but Flutter also very quickly took the lead in the industry. This can be observed when looking at Google search results, for example, where Flutter is compared directly with other frameworks that pursue the same goal.

[1] https://flutter.dev/showcase

What characterizes a Flutter app?

Single codebase:

The main characteristic of a Flutter app is that no matter how many targets the application has, namely mobile, web and/or desktop, they all have a single codebase. This can simplify the whole development process, as code can be reused for different platforms. With some planning, the layout UI elements can be reused across platforms and the whole business logic, contained in the application, only needs to be written once.

Rich variety of widgets:

The main concept of a Flutter app is the synergy of different widgets. Widgets are elements that can be combined to build the graphical interface of the application. Flutter provides a rich variety of widgets for developers to use or even add to. It is also possible to develop fully custom widgets to achieve a pixel-perfect design.

Rapid app development:

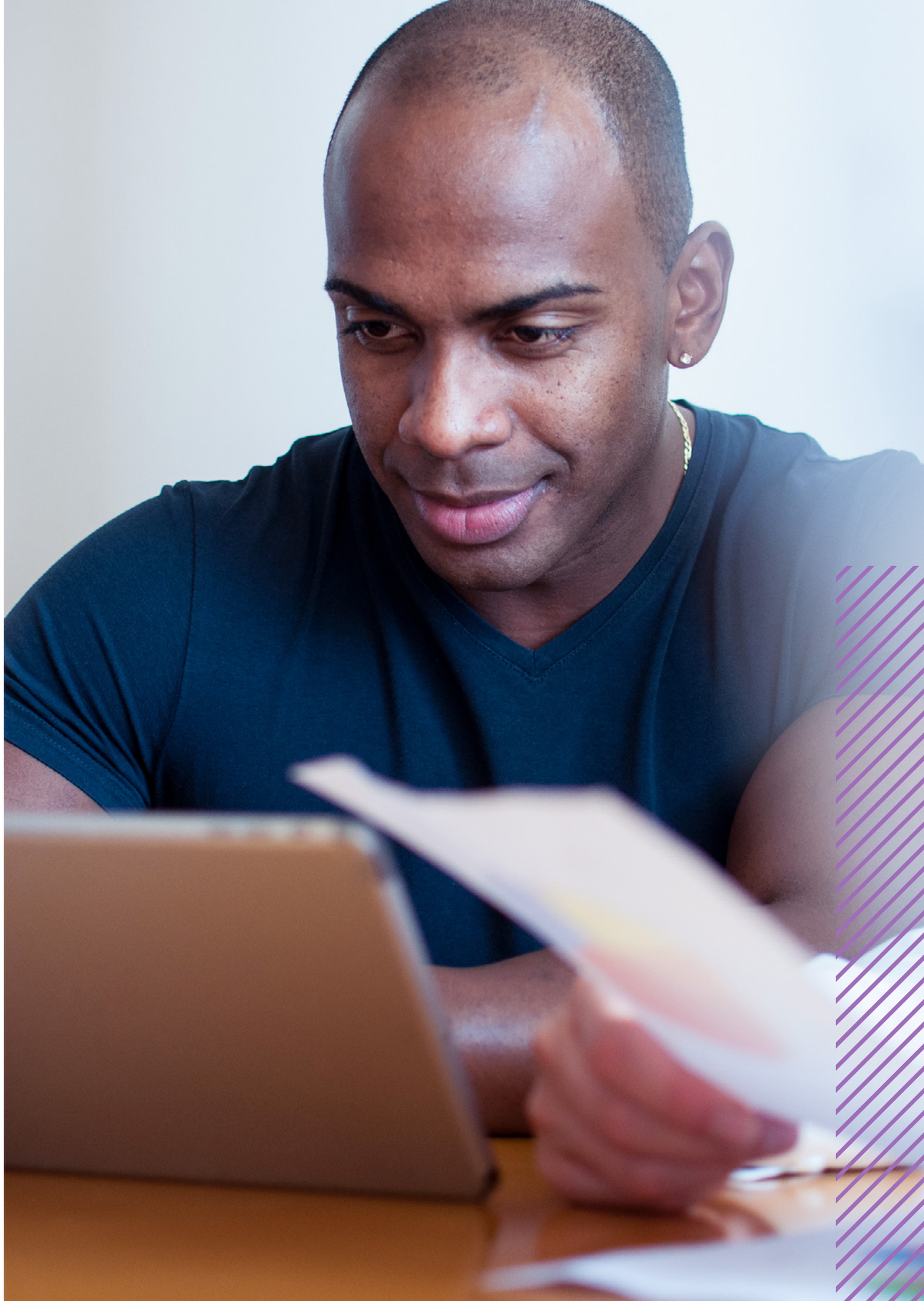
Features like Hot Reload for platforms, which makes changes visible and accessible to developers immediately, enable the developers to rapidly develop, experiment and change applications. Widgets make it possible to quickly adapt and implement novel changes without compromising large parts of the application. This feature, although provided by other frameworks too, works exceptionally well in Flutter.

Significant community support:

A large community of active developers, steadily increasing interest in the framework and Google backing and leading the development of Flutter make the platform reliable. There is a wide selection of plugins and libraries developed and maintained by the community, which can help to rapidly accelerate development.

Variety of platforms

The Flutter framework allows the user to compile the application for all different kinds of targets. Not only can Flutter be used for desktop, web and mobile, but it also allows users to use the core functionality of these targets with zero compromises, while maintaining full native feel and performance. This feature makes Flutter one of the key frameworks for cross-platform development as code redundancy for different targets can be mitigated.



Pros and cons of Flutter

- Cross-platform capability for almost all desired targets
- Fast-growing community
- Open-source Project
- Massive collection of widgets available
- Support for mobile, web and desktop
- Very good documentation from Google
- Used, backed and continuously developed by Google

- Ever-growing range of tools and libraries
- New iOS and Android features may not be supported.
- Young technology. This can have an impact on the number of available experts and existing libraries.
- Getting the platform-specific look and feel requires extra work
- Larger app size as Flutter has its own rendering engine

Difference to other cross-platform Frameworks

Commonly used cross-platform frameworks such as Cordova or Ionic are based on JavaScript (JS) and web views, basically displaying a web page in a native app. These apps are built using HTML and then get displayed in the apps as web views, which provides more of a mobile-first website than an actual app. One big disadvantage of this approach is that in addition to browsers and JS being slow, implementing native device functionality is sometimes prohibited entirely. Flutter also requires libraries to implement device features, but there is a good variety to choose from. If there is no open-source library covering the functionality, it can always be implemented with native code by the development team. Some alternative cross-platform frameworks, such as React Native, use a bridging layer to translate the layout described in JavaScript into the native format of the target platforms. Xamarin also uses an abstraction layer that manages communication between shared code and underlaying platform code. This often means performance drawbacks when widgets are accessed frequently. Flutter, on the other hand, uses its own rendering engine. The engine provides a low-level API for rendering the app layout, text, and everything else. This gives Flutter the advantage of effectively rendering and controlling every single pixel.

The following figure shows Flutter compared to native development in terms of code redundancy for different layers of the applications. As mentioned, the main drawback of native development is the requirement to develop the UI for all target platforms. There are frameworks, such as Kotlin Multiplatform, that effectively enable the business logic of an app to be shared between different mobile platforms. The figure shows how many codebases are necessary for an application developed for different targets.

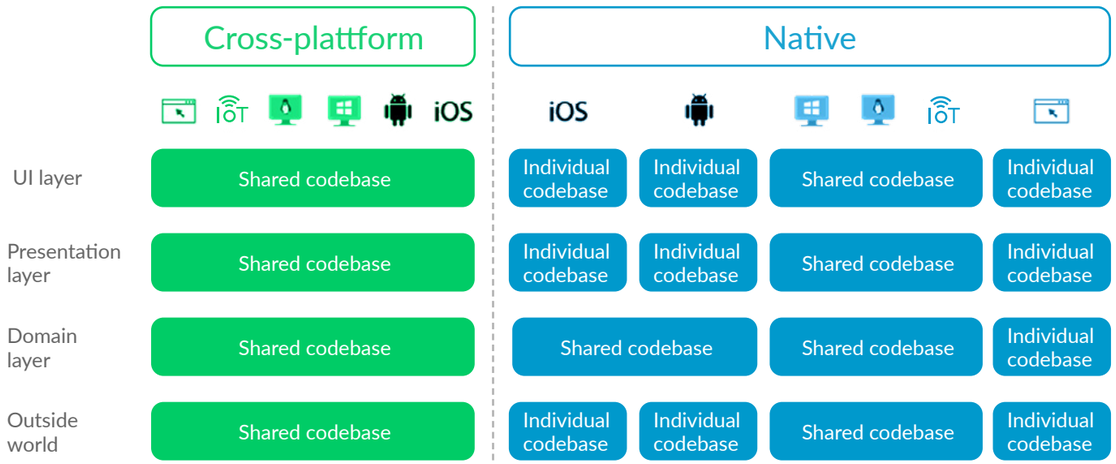


Figure 2: Comparison of codebase required for Flutter vs native

Flutter compared to other cross-platform solutions

	Flutter	React Native	Cordova/ Ionic	Xamarin
macOS	x		x*	
Windows	x		x*	x
iOS	x	x	x	x
Android	x	x	x	x
Linux	x		x*	x
Web	x	x	x*	
Native look and feel	x	x		x
Native performance	x			x
Access to platform-specific services	x	x	x*	x
Hot Reload support	x	x	x	x
SDK built-in testing infrastructure	x	x*	x	x*

Table 1: Comparison of cross-platform solutions

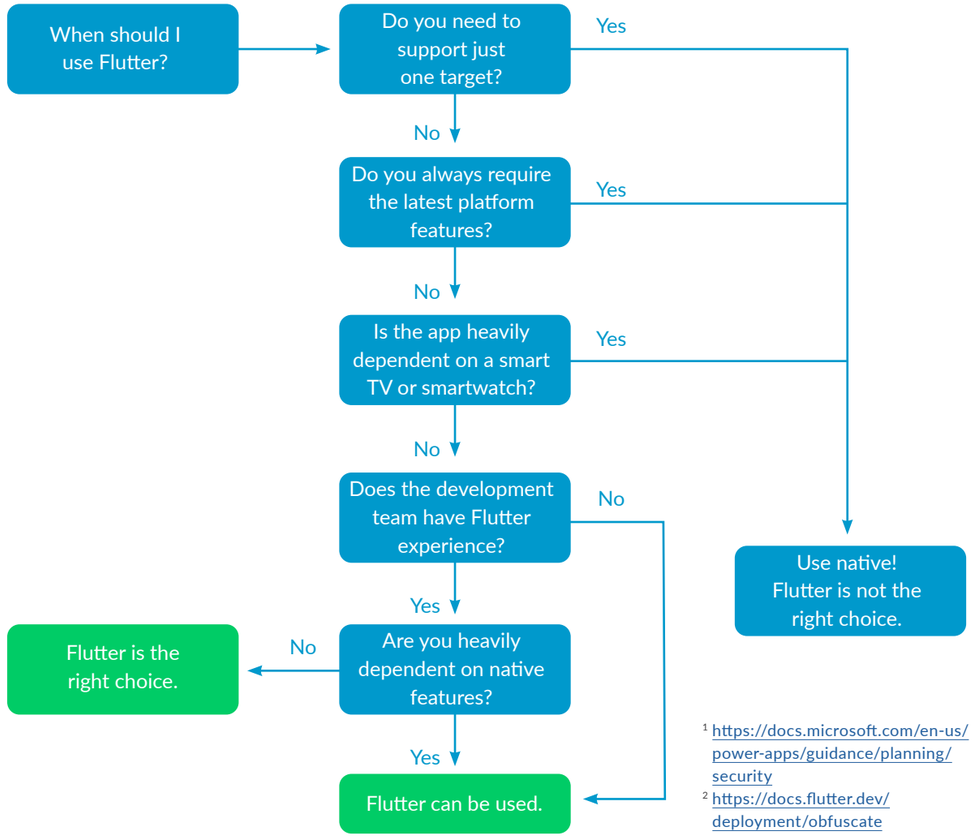
* Requires extra work

Security

Flutter provides a handful of tools and functionality for security. In addition to secure app design[1], which should always be a key factor, code obfuscation can be used on Flutter projects. Code obfuscation hides functions, classes, and everything else from the source code and produces compiled Dart code, which makes it harder to reverse-engineer the app. This feature is currently supported for mobile and macOS applications[2]. The implementation of native security features provided by the target system is always possible. If they are not already implemented by default, they can be integrated with a library extension, thereby effectively achieving the same security standard as with a native app. Libraries from untrusted sources or with outdated implementations often pose the highest risk. It is therefore important to always verify the source when using third-party libraries. That said, there are also great libraries that help to verify and maintain app security. Flutter's security abilities are in line with other frameworks and there are no security trade-offs required.

Decision guidance

This guidance is only advisable as a general first decision matrix. Every case should be looked at in detail. Get in contact with us for a more profound decision.



[1] https://docs.microsoft.com/en-us/power-apps/guidance/planning/security
[2] https://docs.flutter.dev/deployment/obfuscate

Device capabilities

Flutter, like many other frameworks aimed at mobile development, lags behind when it comes to the implementation of the new capabilities launched by Apple or Google every year. For example, the home screen widget feature recently presented by Apple is not yet natively available in Flutter. The solution for such missing requirements is that they can always be implemented with native code or libraries in the Flutter project.

Flutter in general supports the implementation of all kinds of natively coded features for all its different target platforms. This helps to mitigate possible missing platform features with little extra work.

Accessibility

Accessibility is a hot topic in today's software development. It comes as no surprise that the Flutter framework supports all kinds of features to help with this topic. The most important one is the availability and integrability of screen readers. With attention from the developers, it is possible to help visually impaired users to navigate through an app with the help of screen readers. The screen reader can tell the user what the text is, if a button is currently selected, or what will happen once the button is pressed. UI features such as adaptive text size or sufficient contrast are also supported. Native implementations typically still enable better accessibility.

Testing

For the development of high-quality and scalable Flutter applications, testing is one of the most important factors. Flutter has built-in testing infrastructure for unit, integration, and end-to-end tests without the need to render to the screen. You also have the option of implementing tests using Flutter Driver. Flutter Driver connects to the developed application and interacts with it using defined test cases like a user would. It is com parable to Selenium, which is used to automate web applications for testing purposes. All these tests can also run on modern CI/CD pipelines such as Real Device Cloud or BrowserStack.

Experience with Flutter

Time/cost comparison

For the purpose of comparing Flutter to other cross-platform frameworks in terms of time and costs, an internal workshop was held, in which different teams consisting of mobile developers with limited prior knowledge of Flutter tried to develop an app. The goal of the app was to target mobile and web and have as consistent a look across those targets as possible. The predefined goal of the app was to using graphs, show the best time to use electrical power from the grid. The best time was calculated using the electrical grid information on when the power grid was fed with the most renewable energy. The Flutter app development was extremely fast and a prototype deployable to iOS, Android and web was rapidly developed.

Flutter at Zuehlke

At Zuehlke Engineering we have made extensive experiences with Flutter for a variety of different projects. We are enabling clients with large scale applications to transition from native to Flutter and others developing their applications from scratch in Flutter. Flutter is ideal for a transition as Flutter code can be integrated and replace existing code in a native application step by step. This ensures a smooth and fluent transition. But not only our clients also our developers love Flutter. The ease of use, fast development and a decent learning curve makes it easy to adapt.

Platform low-level API’s

We developed an app which could be deployed to mobile and Windows. The app could read and write serial bytes from and to NFC device memory. To this end, we had to make use of the NFC features of the devices. This required the use of low-level APIs. For mobile, we were able to use an already existing plugin that enabled the connection and use of the Android and iOS APIs in Flutter. For the Windows app, we made use of the Dart FFI, which could transfer our requests to the underlying Windows API. The app set an example of how device-specific features and capabilities can be implemented using Flutter and how an entire app can use one business logic.

Rapid app development for an existing website

We also developed an app for a customer in the medical field that is a good example of Flutter's rapid cross-platform solution. The existing web application was successfully rebuilt from scratch in iOS and Android apps within the pre-defined time frame. The existing features of the web app, such as video and audio players, were easily mirrored in the mobile app with the existing Flutter plug-ins, in addition to the network operations with the current backend. The customer’s change requests were applied in both platforms and were delivered efficiently for testing in a short amount of time. The final app showed that rapid development and cost reduction is possible by using Flutter instead of other technologies.

Contact

Doris Rogger, Head of Mobile

doris.rogger@zuehlke.com

Zühlke – empowering ideas.

Zühlke is a global provider of innovation services. We develop ideas and create new business models for our customers by developing services and products based on new technologies – from the vision and concept to implementation, production and operation. Zühlke employs 1,700 staff in Austria, Bulgaria, Germany, Hong Kong, Portugal, Serbia, Singapore, Switzerland, the United Kingdom and Vietnam.

Imprint

Zühlke Engineering AG
Zürcherstrasse 39J
8952 Schlieren (Zürich)
Switzerland

info@zuehlke.com
Managing Director:
Nicolas Durville

Pictures:
Getty Images Deutschland GmbH