

A Comparative Analysis :

Moving to Flutter
vs. Xamarin and
.NET MAUI

Introduction	3
The Xamarin evolution	3
Is .NET MAUI the same as Xamarin?	4
Performance and speed	5
Development environment and ease of learning	6
Platform-specific customisation	6
Community and ecosystem	7
Integration and compatibility	7
Cost and licensing	8
Dart vs C#: Language comparison	11
Xamarin to Flutter: Potential migration challenges	14
Apple's response to Flutter	15
Xamarin vs. Flutter: Conclusion	16

Introduction.

With Microsoft's support for all Xamarin software development kits (SDKs) ending on 1st May 2024, this paper aims to explore the differences between Xamarin, .NET MAUI and Flutter.

Overall, MAUI promises better performance, increased productivity and a more streamlined developer experience than its predecessor, Xamarin. But while .NET MAUI boasts significant improvements in some areas, if you want to build high-performance apps with a fast development cycle and a stunning user experience, we'd recommend considering Flutter for your next web or mobile app.

While the transition to Flutter may present challenges - such as adapting to new programming languages and restructuring codebases - the potential benefits of development speed, user interface quality, significant community support, future-proofing and platform consistency are substantial.

The Xamarin evolution.

Xamarin isn't ending. It's evolving. You're likely already familiar with .NET MAUI - a newer cross-platform development framework from Microsoft, which has replaced Xamarin.Forms. And, as of 1st May 2024, support for all Xamarin software development kits (SDKs) - including Xamarin.Forms - will cease. There are no new APIs planned and Xamarin.Forms projects should be migrated to .NET MAUI, while project types currently using Xamarin.Android, Xamarin.iOS or Xamarin.Mac must be upgraded to .NET SDK-style projects for continued Microsoft support.

If you have an existing Xamarin app - don't panic - there's no need to migrate to .NET MAUI immediately. But, if you're a Xamarin developer or a business using Xamarin for your app development, you might have been wondering about the benefits of the framework evolution to .NET MAUI, whether it's going to solve any of your niggling Xamarin inefficiencies or perhaps whether it's time to make the move to Flutter for your next project (unless you have essential C# dependencies, but we'll address that later). So, let's start by exploring some of the improvements that .NET MAUI offers compared with its predecessor.

Is .NET MAUI the same as Xamarin?

Despite its undeniably powerful and versatile cross-platform development framework, Xamarin has faced particular development challenges throughout its lifecycle, with some of the most documented downsides including:

Cost considerations:

Xamarin is free and open-source, but some features, such as Visual Studio Enterprise, can be expensive for enterprise users.

Performance and size overheads:

Xamarin apps are typically slightly larger and slower than native apps. This is because Xamarin apps use a layer of abstraction to compile to native code.

Limited access to open-source libraries:

Not all open-source libraries are compatible with Xamarin. This can be a limitation for developers who want to use specific libraries in their apps.

Not suitable for apps with heavy graphics:

Xamarin isn't ideal for developing apps with heavy graphics. This is because Xamarin apps use a layer of abstraction that can add overheads.

Slightly delayed support for the latest platform updates:

Xamarin typically supports the latest platform updates slightly later than native development platforms. This is because Xamarin needs to release updates to its own tools and libraries.

Smaller community:

Xamarin has a smaller community than some other cross-platform development frameworks. This can make it more challenging to find help and support.

Steeper learning curve:

Xamarin has a steeper learning curve than other cross-platform development frameworks, such as Flutter. This is because Xamarin developers need knowledge of C# and .NET.

Not suitable for tight turnarounds:

Xamarin apps typically take longer to develop than native apps.

As you'd expect from an evolutionary framework, .NET MAUI offers several advantages over Xamarin, including:

- A unified API for developing mobile and desktop apps
- Support for the latest platform features
- Improved performance and reliability
- A smaller learning curve

Overall, MAUI promises better performance, increased productivity and a more streamlined developer experience than Xamarin. But while .NET MAUI boasts significant improvements in some areas, if you want to build high-performance apps with a fast development cycle and a unique look and feel, we'd recommend considering Flutter for your next project.

While NET MAUI and Flutter are both excellent solutions for building cross-platform applications, the choice between the two will come down to the unique needs and preferences of your project and development team. In this paper, we'll explore the potential advantages and challenges of Xamarin vs Flutter. Without further ado, let's dive in.

Xamarin vs. Flutter: Performance and speed.

Xamarin offers decent performance and speed capabilities, especially for applications prioritising code-sharing across platforms.

However, Xamarin's performance can be impacted by its use of platform-specific UI components, which might lead to subtle differences in behaviour across iOS and Android. Xamarin applications are typically written in C#, which is then compiled to native code using Xamarin.iOS and Xamarin.Android. While this approach provides good performance, it may not match the near-native speed achieved by Flutter.

On the other hand, Flutter utilises Dart as its programming language. It boasts exceptional speed due to its compiled nature and set of widgets, providing a consistent and highly optimised user experience. For a more detailed comparison between C and Dart, see the 'Dart vs C#: Language Comparison' section below.

When it comes to development speed, Flutter's hot reload feature gives it a notable edge over Xamarin. By allowing developers to see instant changes during the development process, the hot reload feature has been reported in several articles and blog posts to result in a 30-40% reduction in time-to-market for mobile apps.

Xamarin vs. Flutter: Development environment and ease of learning.

Xamarin's development environment is robust, particularly for developers familiar with Visual Studio and C#. However, its learning curve can be steep, especially for those new to these technologies. Integrating platform-specific APIs and UI elements often requires additional effort and expertise, making the development process more complex.

In contrast, Flutter offers an incredibly user-friendly development environment. Its simplicity stems from its reactive framework and a rich set of pre-designed widgets, which help to facilitate intuitive app creation.

Moreover, Flutter boasts a [vibrant and supportive community](#). Its extensive documentation, tutorials and active forums provide ample resources for developers of all levels, significantly easing the learning curve.

Flutter's hot reload feature allows developers to experiment and iterate quickly, fostering a more seamless development experience. This combination of a beginner-friendly environment and collaborative community support makes Flutter particularly appealing to developers looking for an accessible yet powerful framework for cross-platform app development.

Xamarin vs. Flutter: Platform-specific customisation.

Xamarin allows developers to leverage native APIs and UI elements. However, the process can be complex and may require extensive knowledge of each platform's intricacies, leading to potential development bottlenecks.

On the other hand, Flutter stands out with its innovative widget system and exceptional platform-specific capabilities. Flutter's platform-aware widgets provide a flexible and consistent way to create custom UI designs across both iOS and Android platforms, allowing developers to effortlessly achieve platform-specific customisations while maintaining a cohesive user experience.

Flutter's approach not only simplifies the development process but also guarantees that apps maintain a native look and feel on both platforms. Its ease of customisation and ability to seamlessly handle platform-specific nuances give developers a significant advantage over Xamarin when creating highly tailored and platform-specific user interfaces.

Xamarin vs. Flutter: Community and ecosystem.

Xamarin boasts an undeniably strong and dedicated developer community with an abundance of available resources and support. However, Flutter has gained remarkable traction recently, establishing itself as a frontrunner in the cross-platform development landscape.

Flutter's community is active and rapidly expanding, with developers worldwide embracing its intuitive framework. One of Flutter's key strengths lies in its extensive open-source packages and plugins collection. These contributions from the community enhance Flutter's functionality, offering solutions for a wide array of tasks, from UI design to backend integrations.

Additionally, Flutter's availability of third-party libraries enables developers to tap into a vast ecosystem of pre-built components and tools. This wealth of resources, combined with Flutter's growing community engagement, provides developers with unparalleled support, making it an increasingly attractive choice for those seeking a vibrant and dynamic development environment.

Xamarin vs. Flutter: Integration and compatibility.

When it comes to their integration capabilities and compatibility, Xamarin and Flutter both excel in supporting seamless integration with backend technologies and providing coherent compatibility with different platforms and technologies.

Xamarin's comprehensive integration options have made it a solid choice for developers who prioritise compatibility across diverse platforms and technologies. Its robust integration options and compatibility with various platforms and technologies have allowed developers to build applications that work consistently across iOS, Android, and Windows devices. It also integrates well with popular IDEs like Visual Studio and Xamarin Studio, ensuring a smooth development experience.

While Xamarin has excelled in its compatibility capabilities, the level of integration might require a deeper understanding of platform-specific intricacies, which can lead to a steeper learning curve and potential challenges for developers unfamiliar with the Xamarin ecosystem.

On the other hand, Flutter provides seamless integration with a wide array of popular development tools and services, including Visual Studio Code and Android Studio.

But Flutter's real strength lies in its intuitive integration with backend technologies and APIs. It connects with REST, GraphQL, and other APIs, enabling developers to establish communication with databases, cloud services and other backend systems effortlessly.

Flutter's versatility and ease of integration make it a preferred choice for developers looking for a streamlined and efficient development process.

Xamarin vs. Flutter: Cost and licensing.

Flutter and Xamarin offer different licensing models, influencing their cost considerations for commercial use. As a Google-ran open-source framework, Flutter is free to use for personal and commercial projects. This open-source nature eliminates initial licensing costs and allows businesses to modify the source code to suit their specific needs without incurring additional charges.

On the other hand, Xamarin, owned by Microsoft, operates under a commercial license. While Xamarin offers a free tier, the full suite of features, especially those tailored for enterprise-level applications, comes at a cost. Businesses using Xamarin for commercial purposes have typically needed to invest in licenses, which could be a significant factor in budget planning.

In terms of potential cost savings, Flutter's open-source nature provides an advantage by reducing upfront expenses significantly.

The extensive collection of open-source packages and plugins available within the Flutter community can save developers valuable time and effort, translating into further cost savings. Xamarin, while powerful, has tended to require a higher initial investment, especially for larger businesses or projects that demand enterprise-level features.

Companies seeking a budget-friendly and highly flexible solution may find Flutter a cost-effective choice due to its open-source nature and the availability of a vast array of free resources. However, the decision to use Flutter should also consider factors such as project complexity, required features and developer expertise.

For a detailed examination of choosing Flutter for your specific app development needs, we recommend reading our in-depth white paper, [Unlocking Business Success with Flutter: A Comprehensive Guide for Business Owners](#).

Xamarin vs. Flutter: Security considerations

Both Flutter and Xamarin offer similar capabilities when it comes to security considerations for developing cross-platform mobile applications. Both are robust and capable frameworks, and the choice will come down to your specific project requirements, development team expertise, and overall preferences. Regardless of the framework you choose, it's essential to follow security best practices, stay updated with the latest guidelines, and maintain secure coding techniques to ensure the security of your mobile applications.



VS



Secure communication

Flutter provides packages like `http` and `dio` for making secure API requests using HTTPS protocols, ensuring secure communication between the app and the server.



Xamarin supports secure communication channels for API requests, using libraries like `System.Net.Http` to make secure HTTP requests, ensuring data transmission over HTTPS.

Code obfuscation

Flutter developers can use tools like ProGuard and R8 for code shrinking and obfuscation, which helps protect the app's source code from reverse engineering.



Xamarin developers can employ code obfuscation techniques using tools like ProGuard, enhancing the security of the app's source code against reverse engineering.

Dependency security



Flutter developers can monitor and update dependencies using flutter pub outdated to identify outdated packages, ensuring that the app uses the latest, most secure versions of libraries.



Xamarin developers can use the NuGet package manager to keep track of dependencies and update packages to their latest versions, addressing security vulnerabilities in outdated libraries.

Authentication and authorisation

Flutter integrates well with various authentication providers, enabling developers to implement strong authentication and authorisation mechanisms within the app.



Xamarin supports integration with various authentication providers, enabling effective tailored authentication and authorisation mechanisms.

Data storage security

Flutter provides plugins like flutter_secure_storage for secure and encrypted storage, allowing developers to store sensitive data in a protected manner.



Xamarin provides secure storage options and supports encryption libraries, enabling developers to store sensitive data securely within the app.



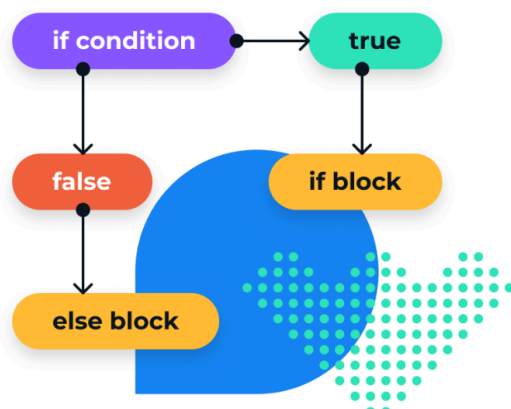
Dart vs C#: Language comparison.

Earlier, we touched on C# dependencies as a potential hindrance for developers moving from Xamarin to Flutter. However, due to their language similarities, it's relatively easy for developers to shift from C# to Dart if they want to use Flutter instead of Xamarin or .NET MAUI. Both languages are object-oriented and have similar syntax. Dart is also a statically typed language which will be familiar to C# developers.

Here are some **key similarities** between C# and Dart



Both languages use **{ curly braces }** to define blocks of code.



Both languages have similar **control flow** statements, such as **'if'**, **'else'**, **'for'**, and **'while'**.



Both languages use **classes** and **objects** to organise code.



Both languages support **generics**.

There are also some **key differences** between C# and Dart

Dart	C#
Single-inheritance language	Multiple-inheritance language
Has no null reference type	Does have a null reference type
Has a built-in type system for asynchronous programming	Does not have a built-in type system

Overall, the similarities between C# and Dart outweigh the differences, meaning that C# developers should be able to pick up Dart quickly.

We'd suggest starting by learning the basics of the Dart language, such as syntax, data types, and control flow statements. Once you have a basic understanding of the Dart language, learn about the Flutter framework. Flutter has comprehensive documentation and tutorials that can help you get started.

Don't be afraid to ask for help. There are many online resources and communities where you can get help from other Dart and Flutter developers. With a little effort, C# developers should be able to transition to Dart and Flutter smoothly.

Xamarin to Flutter: Potential migration challenges.

Migrating from Xamarin to Flutter can be a transformative but challenging process for businesses and developers.

One of the key challenges lies in the fundamental differences between the two frameworks: Xamarin utilises C# and relies heavily on native components, whereas Flutter uses Dart and has its own widget-based architecture.

Consequently, developers must adapt to new programming languages and paradigms, which can require time and effort. Migrating existing codebases and ensuring feature parity can be complex, especially for large, intricate applications.

However, the benefits of shifting to Flutter are substantial. As discussed, Flutter's hot reload feature significantly accelerates the development cycle, allowing for faster iterations and bug fixes. The platform's expressive UI components and platform-specific customisation options enhance the user experience and facilitate the creation of engaging, eye-catching apps.

Furthermore, Flutter's single codebase for iOS and Android reduces maintenance efforts and ensures consistent performance across platforms, ultimately saving time and resources in the long run.

Despite the challenges, the benefits of improved development speed, enhanced user interface, and platform consistency make the migration to Flutter a compelling choice for many businesses.

Apple's response to Flutter.

While there's no official statement from Apple on its thoughts on Flutter, in 2019, Apple's vice president of worldwide marketing, Phil Schiller, said that Flutter was "a very interesting technology" and that Apple was "watching it closely."

In terms of how Apple views Flutter as a hybrid approach to creating mobile apps that are suitable for native iOS apps, our views are speculative. But of course, Flutter could prove either a competitor to Apple's own native development tools, such as Xcode or SwiftUI, or - more positively - a solution to enabling developers to create cross-platform apps quickly, easily and effectively.

Here are some of the pros and cons of using Flutter for hybrid iOS development:

Pros

Flutter allows developers to create cross-platform apps with a **single codebase**, which can save a lot of time and effort.

Flutter apps are native to the platform they are running on, so they have the same **look and feel as native** iOS apps.

Flutter has a **large and active community**, so there are plenty of resources available to help developers get started and troubleshoot problems.

Cons

Flutter is a relatively **new technology** that isn't as well-established as some other hybrid development frameworks.

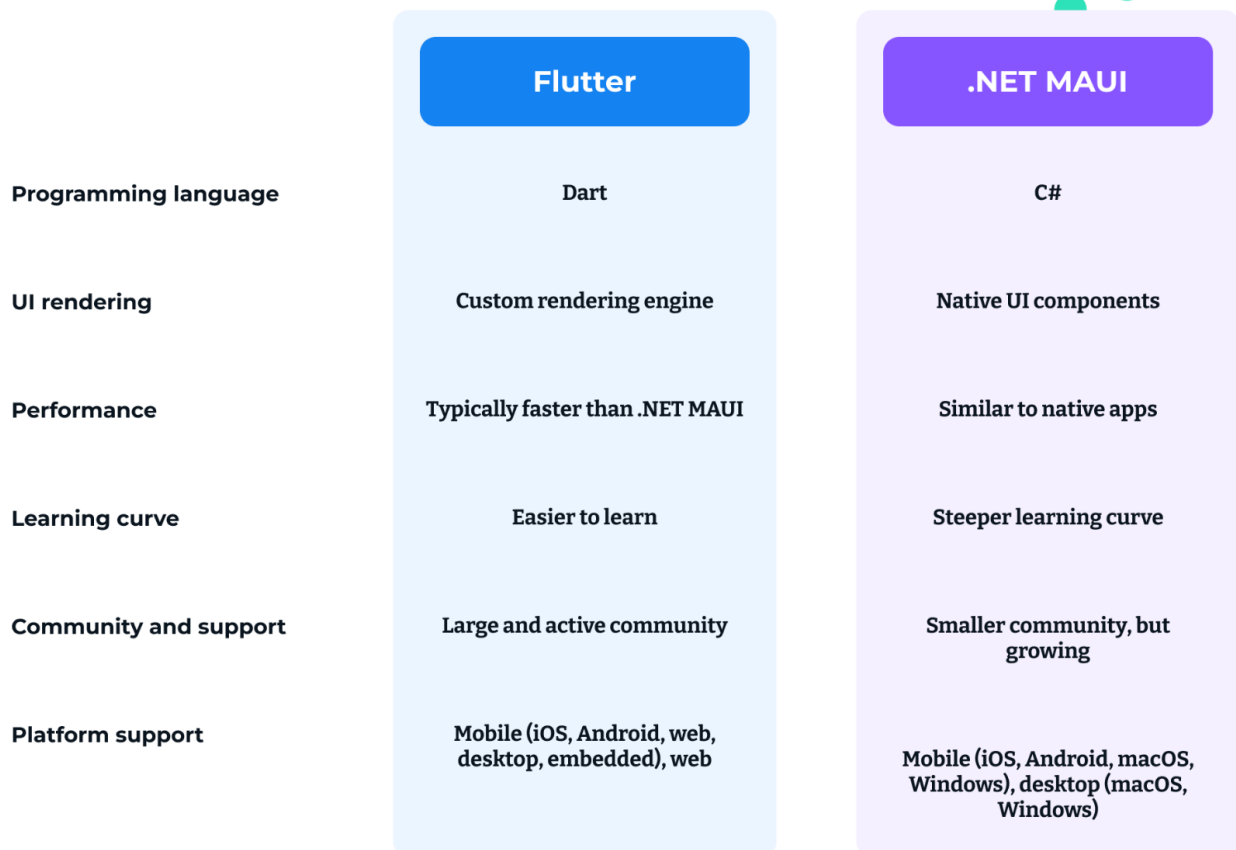
Flutter apps can be **larger in size** than native iOS apps due to the fact that Flutter includes its own rendering engine.

Some Flutter features are not yet fully supported on iOS, such as **native navigation bars and tab bars**.

Of course, the Flutter framework is still developing and there have already been some significant examples of popular iOS apps that have been built using Flutter such as Google Pay, BMW Connected, Alibaba, eBay Motors and Hamilton. These apps clearly demonstrate that Flutter can be used to create high-quality, native-looking iOS apps.

Xamarin vs. Flutter: Conclusion.

With the Xamarin end of support fast approaching, here is a summary of the key differences between Flutter and .NET MAUI:



	Flutter	.NET MAUI
Programming language	Dart	C#
UI rendering	Custom rendering engine	Native UI components
Performance	Typically faster than .NET MAUI	Similar to native apps
Learning curve	Easier to learn	Steeper learning curve
Community and support	Large and active community	Smaller community, but growing
Platform support	Mobile (iOS, Android, web, desktop, embedded), web	Mobile (iOS, Android, macOS, Windows), desktop (macOS, Windows)

The decision to migrate from Xamarin to Flutter represents a significant choice that requires careful consideration of various factors. While the transition may present challenges - such as adapting to new programming languages and restructuring codebases - the potential benefits of development speed, user interface quality and platform consistency are substantial.

As the app development landscape evolves, businesses and developers must align their projects with frameworks that best meet their specific needs and goals. Flutter's robust features, intuitive development environment and active community support make it a compelling choice for those seeking a seamless and efficient cross-platform development experience.

While both frameworks allow for cross-platform development, the choice between Flutter and .NET MAUI often depends on factors such as preferred programming language, desired level of customisation and integration requirements with existing Microsoft technologies.

As one of the first 13 companies to be listed on [Google's Flutter directory](#), we're recognised for our Flutter experience and have currently used the framework to successfully deliver 7 mobile apps: [Dodl by AI Bell](#), [Brawn](#), [52N](#), Misthos, Warburtons, Great Rail Journeys and [Railguard](#).

Through our [unique perspectives and in-depth Discovery process](#), we can support your business to thoroughly evaluate project requirements and ensure you're equipped with the most suitable tools to create exceptional, high-performing applications that cater to the demands of today's competitive digital market.

Interested in having a chat? [Get in touch with our team](#).