GUIDE FOR EXECUTIVES

NATIVE VS HYBRID/MULTIPLATFORM

Making the Right Call for Your Company's Mobile App Future

SIDE KICK INTERACTIVE

Updated 2023 edition

Executive Summary

This guide is designed to assist you, a key decision-maker in your organization, in navigating the landscape of mobile application development.

Embarking on the app development journey requires a critical decision upfront: whether to choose native or hybrid/multiplatform development frameworks. This decision has a significant impact on the application's budget, performance, functionality, user experience, and overall success. **And most important of all, significant financial ramifications will result from this decision.**

The aim of this guide is to provide a comprehensive comparison of native (Swift for iOS and Kotlin for Android) and hybrid/multiplatform (Ionic, Flutter, Xamarin and ReactNative) frameworks.

It's designed to equip you with the critical insights needed to make informed decisions about your mobile app development strategy and to provide you with persuasive arguments to secure approval and support from your board and fellow executives.

SPOILER: Native apps are superior for most business cases, but there are exceptions! Read on to learn more.

WHY THIS DECISION MATTERS?

Building a mobile app is a major undertaking with large budget and timeline. Projects can range from hundreds of thousands of dollars to millions and can take more than 2 years to build.

Making a wrong decision can lead to a subpar application, but worst, can also lead to having to re-build your app from scratch (as AirBnB has learned the hard way).

This decision will impact hiring, budgeting, and your ability to evolve your app through time. And its success can increase the top and bottom lines.

In-Depth Look at Native Applications

Native applications are designed and developed to run on a specific platform, such as iOS or Android. Because these apps are optimized for the specific operating system they are developed for, they can directly access and utilize the device's hardware and native features, such as the camera, microphone, GPS, and more. This means they can deliver a highly optimized, fast, and smooth user experience.

Native apps also adhere to the specific UI standards of the platform, offering a more integrated and intuitive user interface. Furthermore, native app frameworks are meticulously crafted and maintained by the respective hardware developers of each platform (such as Apple or Google), guaranteeing continuous support and ongoing enhancements as time progresses, which can avoid costly re-engineering in the short to medium term.

KEY BENEFITS OF NATIVE APPLICATIONS

There are several advantages of choosing native development for your business app:

1. **Overall Project Budget:** While native development might require more upfront resources and time due to the necessity of creating separate apps for each platform, the long-term maintenance costs for native apps can be lower, as they are more stable and less susceptible to issues arising from OS updates. These factors become particularly critical for complex apps or apps aiming to deliver a premium user experience over many years.
2. **Security:** Native apps have the advantage of the platform's built-in security features. Additionally, since they are not dependent on third-party platforms, there are fewer vulnerabilities that can be exploited by hackers.
3. **Technological Independence:** With native apps, you are choosing not to rely on third parties to power your applications. You can take your source code and change development teams, and a problem by a third-party won't affect you.

4. **Performance:** Since native apps are developed specifically for a platform, they are highly optimized for it. They can make full use of the device's hardware and the operating system's features, resulting in a highly responsive, fast, and smooth performance. This can lead to a significantly better user experience, especially for apps that require complex computations or handle substantial amounts of data.
5. **User Experience:** Native apps can access the native UI components of the platform they are developed for. This means they can provide a UI that is consistent with the user expectations for that platform, resulting in an intuitive and seamless user experience.
6. **Scalability:** Native apps can handle more complex functionalities and larger amounts of data, making them more scalable. As your business and user base grows, your app will be able to scale to meet increased demands.
7. **Size of Community:** Both Swift and Kotlin have large, active developer communities. This means that support, in the form of tutorials, forums, and third-party tools, is readily available. This can greatly aid the development process and help resolve any issues quickly.
8. **Ease of Recruiting Developers:** Given the popularity of Swift and Kotlin, finding skilled developers for these languages is typically easier compared to hybrid/multiplatform frameworks. Moreover, developers for these languages are often more specialized, which can contribute to higher quality code and better application performance. According to our research, there is a 5-1 ratio of native developers to hybrid/multiplatform developers in North-America. (for example, there are 5,500 “Flutter” developers on LinkedIN in Canada, and there are 26,000 “Swift” or “Kotlin” developers)
9. **Maintenance of Programming Languages:** Swift and Kotlin are maintained and continuously improved by Apple and Google, respectively. This ensures that these languages stay up-to-date with the latest technological advances and standards, and can provide optimized and efficient app performance. Native apps can also be quicker and easier to update or upgrade in line with new OS versions, compared to hybrid/multiplatform apps.
10. **Enhanced User Satisfaction and Reviews:** Native apps offer superior user experiences, performance, and quality, resulting in heightened user satisfaction. Consequently, this translates into higher app store ratings, and therefore more prominence, better visibility, and more downloads.

Let's examine the primary languages used for native development:

1. **Swift for iOS:** Swift is a modern, powerful and intuitive programming language developed by Apple. Swift is designed to be easy to read and maintain, and it offers robust, efficient, and responsive app development. Swift is interoperable with C, Objective-C, C++ and also other languages.

2. **Kotlin for Android:** Kotlin is a modern, statically typed programming language that is officially supported by Google for Android app development. It is fully interoperable with Java, but more concise, expressive, and safer.

In-Depth Look at Hybrid/Multiplatform Frameworks

KEY BENEFITS OF HYBRID/MULTIPLATFORM APPLICATIONS

Despite some limitations, hybrid/multiplatform apps offer several advantages that might make them a suitable choice for certain projects:

1. **Cost and Time to Market:** Since a single codebase is used for all platforms in hybrid/multiplatform app development, it can lead to faster development and deployment times compared to native development. If your app needs to be available on both iOS and Android, and time-to-market is a critical factor, hybrid/multiplatform development might be beneficial.
2. **Unified Development Process:** Since the same codebase is used for all platforms in hybrid/multiplatform development, you will not need separate iOS and Android teams. This can significantly reduce the development team size, making it a great option for startups or companies with tight budgets. One or a small team of developers can handle the development for both iOS and Android, contributing to cost efficiency in the short term.
3. **Platform Diversity:** One of the significant benefits of hybrid/multiplatform applications is their ability to operate across multiple platforms. If your business targets more than two platforms, such as iOS, Android, Windows, etc., then hybrid/multiplatform development starts to make a lot of sense. With a single codebase, hybrid/multiplatform frameworks can facilitate the creation of an application that works seamlessly across multiple platforms. This can save considerable time and resources, making hybrid/multiplatform development an attractive choice for businesses aiming for a broad, multi-platform reach. It is important to note that while the aforementioned advantages hold true for applications designed for different OS, the same cannot be said for applications that span across multiple devices within a platform ecosystem. For instance, if you intend to release an app within the Apple ecosystem encompassing iOS, MacOS, WatchOS, and tvOS, native apps can seamlessly extend to each device with minimal effort, unlike hybrid/multiplatform applications that may face challenges in achieving the same level of seamless integration.

*While the terms "hybrid" and "multiplatform" development frameworks often get used interchangeably, they technically refer to different concepts in the realm of mobile app development. For this guide, we will refer to them as one development framework concept, as their benefits and drawbacks are similar. However, here is more detail explaining both terms:

Hybrid Apps

Hybrid applications are essentially web applications (typically developed using HTML5, CSS, and JavaScript) enclosed in a native app shell. They run inside a WebView (a native container that can display web content), but can also interact with native device features via various plugins.

Multiplatform Development Frameworks

On the other hand, multiplatform development frameworks enable developers to write code once and compile it to multiple platforms – producing a "native" app for each platform (without native code).

Let's delve into these primary frameworks used for hybrid/multiplatform development:

1. **Ionic:** Ionic is an open-source framework for developing mobile apps. It utilizes web technologies and focuses on performance, usability, and platform-specific design. Its components allow the developer to create apps with a native-like feel.

2. **ReactNative:** ReactNative, developed by Facebook, allows developers to build mobile applications using only JavaScript and React. It translates your markup to real, native UI elements, leveraging existing means of rendering views on whatever platform you are working with.

3. **Flutter:** Created by Google, Flutter is a UI toolkit for crafting natively compiled applications for mobile, web, and desktop from a single codebase. It uses the Dart programming language and is known for its fast development.

4. **Xamarin:** A Microsoft-owned framework, Xamarin allows you to build Android, iOS, and Windows apps with .NET and C#. It gives you the ability to share code across all platforms.

Hybrid/multiplatform apps can face challenges when it comes to performance, especially for complex applications or those requiring extensive use of device features. Hybrid/multiplatform apps can also present difficulties with maintenance over time. Compatibility with new OS versions or hardware features may require additional work compared to native apps, which can increase maintenance costs in the long run.

Why Native Applications are Superior for Most Business Cases

In the world of mobile app development, executives often face the challenge of choosing between native and hybrid/multiplatform app development. While both have their advantages, **native applications often provide a superior return on investment (ROI) for most businesses**, particularly over the long term. Here's why:

1. Optimized Performance: Native applications deliver unmatched performance because they're built specifically for the platform they're running on, using the platform's core programming language and APIs. This means faster load times, smoother transitions, and an overall user experience that leaves a lasting positive impression on your customers, ultimately driving higher user engagement and conversion rates.

2. Unparalleled User Experience (UX): By offering features and design consistent with the platform's UI/UX guidelines, native apps feel intuitive to users, leading to increased usage and customer satisfaction. Better user experience means more repeat customers, directly contributing to your bottom line.

3. Scalability and Futureproofing: Native apps are more scalable and adaptable to the growth of your business and changes in user requirements. The robustness of native development allows the app to handle more complex functionalities and larger amounts of data, protecting your investment as your business evolves. Plus, they are directly supported by the companies building the phones the apps are installed on.

4. Security and Compliance: As data privacy regulations continue to tighten globally, native apps offer the advantage of superior security features inherent to the platform. This reduces the risk of costly data breaches and non-compliance penalties, protecting your company's reputation and bottom line.

5. Better Support and Maintenance: Swift and Kotlin are actively maintained and improved by Apple and Google, respectively, ensuring your app stays current with the latest technology trends and standards. This results in lower maintenance costs and better long-term support, significantly improving your ROI over time.

6. Larger Developer Pool: Recruiting is easier for native platforms as the developer communities for Swift and Kotlin are large and well-established. This can result in faster development times, lower costs, and better overall app quality.

7. Apps for Only One Platform: Many apps are built for only one platform (for example, if your employees all have iPads and you wish to supply them with a tool on their devices in the form of an app). In this case, building a hybrid/multiplatform app serves no purpose and brings MORE costs and time than using native.

8. Costs of Continuous Development and Improvements: Native apps will provide apps that are much easier to support in time should you wish to make changes, both to the functionality and look and feel. Their platform independence also makes it easier to update apps yearly and to debug apps.

While hybrid/multiplatform apps might seem attractive due to initial cost and time savings, the long-term ROI of native apps tends to be significantly higher for most businesses. By investing in a native app, you're ensuring your mobile strategy is built on a solid foundation that will support your business's growth and adapt to future trends.

Why Most Large Enterprises and Funded Startups Opt for Native Apps: Insights from Employment Data

Employment data serves as a robust indicator of the preference for native apps within large enterprises and well-funded startups. An analysis of hiring trends in these organizations reveals a substantial lean towards employing developers with expertise in native app development over those with skills in hybrid/multiplatform development.

In contrast, developers with skills in hybrid/multiplatform development are more commonly found within digital agencies. These agencies often work with a broader range of clients, including smaller businesses and projects with more constrained budgets, where the cross-platform capabilities and potentially lower upfront development costs of hybrid/multiplatform apps can be appealing.

However, for large enterprises and well-funded startups, the immediate cost savings offered by hybrid/multiplatform development are often outweighed by the long-term benefits of native development, which include superior performance, improved user satisfaction, and lower long-term maintenance costs. This preference underscores the value that these businesses place on the distinct advantages offered by native apps. Thus, their hiring practices reflect a strategic investment in the quality and longevity of their mobile applications, cementing native app development as their preferred choice.

Exceptions: When to Choose Hybrid/Multiplatform Applications

While native applications often prove superior in most scenarios, there are certain circumstances where choosing a hybrid/multiplatform approach may be more beneficial. **Here are some scenarios where a hybrid/multiplatform application might be a better choice:**

1. **Simple Apps on Many Platforms:** If your app has simple functionalities and doesn't need advanced native features or high-performance, and needs to be deployed to many platforms, a hybrid/multiplatform app could be an effective choice. They offer sufficient functionality for basic applications, such as content distribution apps or simple informational apps.
2. **Leveraging Complex Web Infrastructure:** When firms seek to transport an existing complex web infrastructure's functionality to mobile, feature for feature, choosing a hybrid/multiplatform approach can prove beneficial (For example, if a company has a large complex React website with a large React development team, then ReactNative could prove a good choice). This route may simplify the transition, but the trade-offs concerning adding new features should be weighed in the decision.
3. **Quick Market Entry:** If getting your app to the market as quickly as possible is a priority, hybrid/multiplatform app development can expedite this process. Since a single codebase is used for all platforms, you might be able to reduce the development time.
4. **Minimal Device Feature Usage:** If your app doesn't rely heavily on device features like GPS, camera, or microphone, hybrid/multiplatform development might suffice. Though hybrid/multiplatform apps can access device features, native apps leverage these with greater efficiency and performance.
5. **No Connected Devices:** If your app doesn't need to connect or communicate with other hardware devices (like wearables, IoT devices, connected cars, etc.), a hybrid/multiplatform app could be a feasible choice.

6. **Short App Life:** For apps intended to serve only in the short-term (less than 1 year), a hybrid/multiplatform app can suffice, given its faster time-to-market and lower initial development costs (an example would be a seasonal marketing app).
7. **More than 3 Operating Systems:** If your business is targeting multiple platforms beyond just iOS and Android (for example, web, Windows, Linux, other TV operating systems, etc) - hybrid/multiplatform applications start to make sense. Since these apps share a common codebase, they can be deployed across these various platforms, offering substantial time and cost efficiencies. This allows for a broader reach to your audience, irrespective of the device or operating system they use. However, keep in mind that each additional platform may introduce its own complexities and variations, which could impact the overall user experience if not properly addressed.

It's essential to carefully assess the unique requirements, objectives, and constraints of your project before deciding on the development approach. While native apps offer a range of benefits, hybrid/multiplatform apps do have their place for certain projects and situations.

Leveraging Large Web Teams: Evaluating the Trade-offs of Hybrid/Multiplatform Choices

While companies with significant web development teams may initially be inclined towards hybrid/multiplatform solutions, it's crucial to recognize the inherent complexities of this strategy. Adding a mobile app development project would likely require expanding the team, rather than solely relying on existing resources. While these frameworks might enable use of familiar web technologies like JavaScript, HTML, and CSS, it's not feasible for your existing team to concurrently maintain web tools and create a new app without potentially impacting quality or delivery timelines.

Moreover, while hybrid/multiplatform technologies offer some degree of flexibility, it's important to note that not all web technologies are readily transferable to the mobile environment. There can be limitations, particularly when attempting to access advanced device-specific functionalities or in delivering a seamless, native-like user experience.

So, while pursuing a hybrid/multiplatform strategy could initially seem like an efficient way to capitalize on your current team's skills, it's essential to evaluate the potential trade-offs. The need for team expansion, the limitations in transferring all web technologies to mobile, and the challenges of concurrently maintaining web tools and developing a new app, should all be carefully considered in the decision-making process.

"Why Most Software Agencies Focus on Hybrid/Multiplatform Apps

In the world of digital agencies, you'll find most software agencies advocating for hybrid/multiplatform solutions. While there are certain situations where this can be beneficial for the client (see list of exceptions), it's crucial to understand the reasons behind this recommendation and how it can favor the agency's interests. Here's why agencies might lean towards selling you a hybrid/multiplatform app:

1. **Agency Profitability:** Developing hybrid/multiplatform apps can be more profitable for agencies. Given the single codebase, it is quicker to develop and deploy apps on multiple platforms, therefore the lower initial budget can lead to higher sales conversion. Conversely, the very high support and maintenance costs of hybrid/multiplatform apps will mean higher long-term revenues for the agency.
2. **Customer Dependence:** Choosing a hybrid/multiplatform framework often results in clients being "handcuffed" to the agency's tech stack, as there are less available teams that can take on the specific hybrid/multiplatform language that was chosen. This dependence can make it harder for the client to switch agencies in the future or to recruit internal talent to take over the app.
3. **Developer Recruitment:** It's typically easier for agencies to recruit and manage developers if they specialize in a single hybrid/multiplatform framework. Specializing in a specific hybrid/multiplatform framework means they only need to search for developers proficient in one technology stack, rather than having separate teams for Swift (iOS) and Kotlin (Android), which can streamline their hiring process and give them more flexibility. Good for the agency, but what’s in it for the client?
4. **Ongoing Maintenance Fees:** Hybrid/multiplatform apps generally have higher maintenance costs. They don't age as gracefully as native apps, often requiring updates and fixes with each new OS update to ensure they continue to function as expected. These ongoing costs can be a continuous source of revenue for the agency.

It's important to remember that a reputable agency will consider the best interests of their clients and recommend a solution that meets the project's unique needs, whether that's a hybrid/multiplatform or native approach. However, understanding these motivations can help you enter discussions with an agency better informed, and ensure you're getting the solution that best fits your needs, not just the agency's.

The Longevity of Native

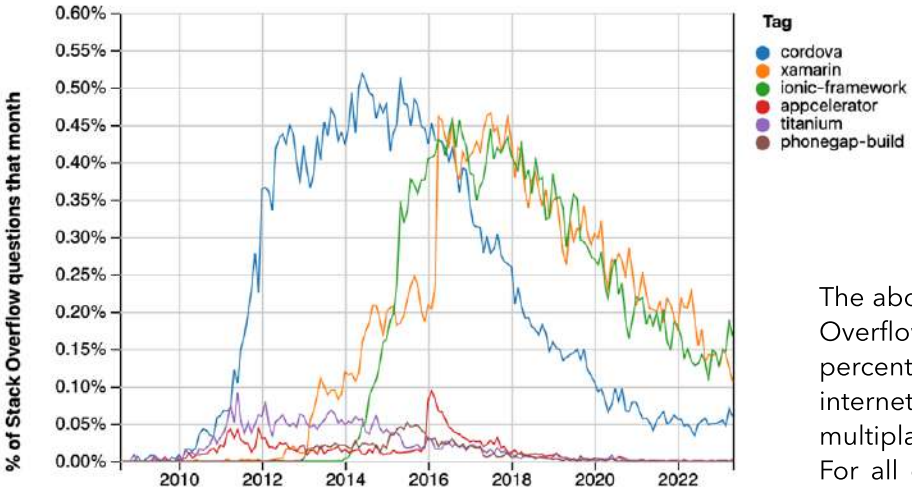
We previously mentioned React Native, Xamarin, and Flutter as hybrid/multiplatform development frameworks. We could have also cited PhoneGap, Cordova, Appcelerator, and Titanium, among others. Over the past few years, these have emerged on the market, one after the other, each reaching its peak of popularity quite rapidly. However, many ended up having to shut down after just a few years.

For example, even though it was acquired by Adobe in 2011, PhoneGap is no longer maintained as of 2020.

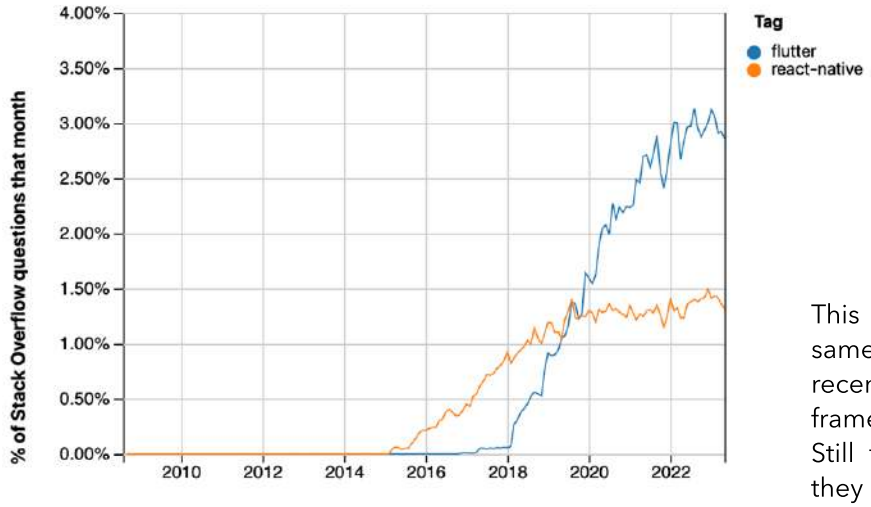
All these hybrid/multiplatform technologies fall prey to trends. Even though they seem innovative at the time of their respective releases, these platforms eventually lose the interest of app creators after a while.

This points to the enduring value and stability offered by native application development, which continues to evolve while maintaining its relevance in the face of changing trends. Also, it is important to mention that Native frameworks are published, maintained, and marketed by the companies that are selling the smartphone themselves (Apple and Google). Therefore, it makes sense that they would favor these frameworks.

Stack Overflow is a question-and-answer website focused on computer programming.



The above graph (1), created on Stack Overflow, shows the evolution of the percentage of questions asked by internet users about the different multiplatform frameworks over time. For all of them, we notice a peak of interest at their release, followed by a stabilization period before a rapid decline. This illustrates the trend effect that we mentioned earlier.



This other graph (2) compares the same information for the two most recently sought-after multiplatform frameworks: Flutter and React Native. Still the subjects of many questions, they both seem to have recently reached their peak popularity on Stack Overflow.

The main problem with hybrid/multiplatform technologies is therefore their lack of stability. Choosing hybrid/multiplatform means taking the risk of opting for a development framework that will no longer be supported after a few years. This is less of a concern for a short-term and/or disposable project, but for a long-term project, the death of the development framework forces you to start the app development from scratch.

Conclusion

Choosing the right framework for your mobile app is a decision that impacts not just your development process, but your business as a whole.

Native applications, built specifically for iOS or Android platforms using Swift or Kotlin respectively, provide superior performance, a seamless user experience, and access to the full capabilities of the device. They offer a level of scalability, security, and control that makes them the choice of many tech companies and organizations building in-house apps. The robust community and ease of hiring developers for these languages are other key advantages.

Hybrid/multiplatform applications, on the other hand, offer a faster time-to-market and can be more cost-effective initially, especially for throw-away projects or simpler app requirements. However, one should be mindful of potential higher maintenance costs and the necessity of frequent updates to keep up with OS changes.

It's essential to take a holistic view of your app project. Consider your business's unique needs, resources, and long-term goals when deciding between native and hybrid/multiplatform development. While agencies might often advocate for hybrid/multiplatform apps for their benefits, it's crucial to understand how this decision impacts your business in the long run.

Ultimately, the choice between native and hybrid/multiplatform development isn't about which one is universally superior, but rather which is the best fit for your specific circumstances, objectives, and constraints. Use this guide as a starting point, consult with your development team and partners, and make an informed decision that best serves your business and your users.

Appendix

Glossary of key terms.

1. **User Experience (UX):** This refers to the overall experience a user has while interacting with a product such as a website or a mobile app.
2. **Operating System (OS):** An operating system is system software that manages computer hardware, software resources, and provides various services for computer programs. iOS (Apple) and Android (Google) are the two primary mobile operating systems.
3. **Scalability:** Scalability is the capability of a system to handle a growing amount of work, or its potential to accommodate growth.
4. **Maintenance Cost**: This refers to the cost associated with regular updates, bug fixes, adding new features, and ensuring the smooth functioning of an app after it has been deployed.
5. **Time-to-Market:** This refers to the total time it takes from a product's conception to its launch in the market.
6. **Tech Stack:** A tech stack refers to the combination of software products and programming languages used to create a web or mobile application.
7. **Device Features:** Device features refer to the functionalities of a mobile device that apps can access, such as the camera, microphone, accelerometer, GPS, etc.

These definitions should provide clarity on some of the main technical terms used throughout the guide. If you come across any additional terms you're unfamiliar with, don't hesitate to ask for further clarification.