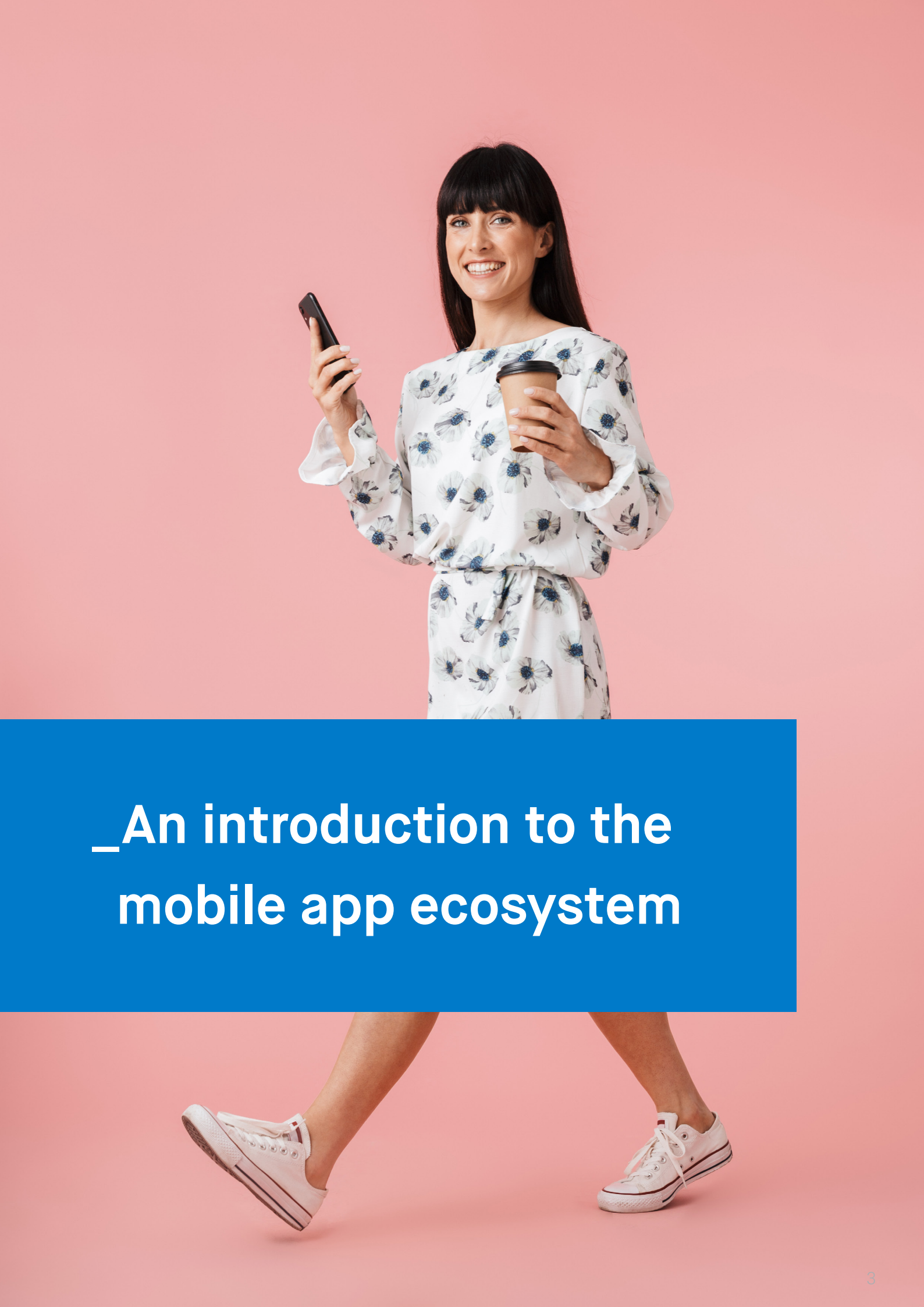
jntive

WHITE PAPER

Cross-platform apps and the future of agile development

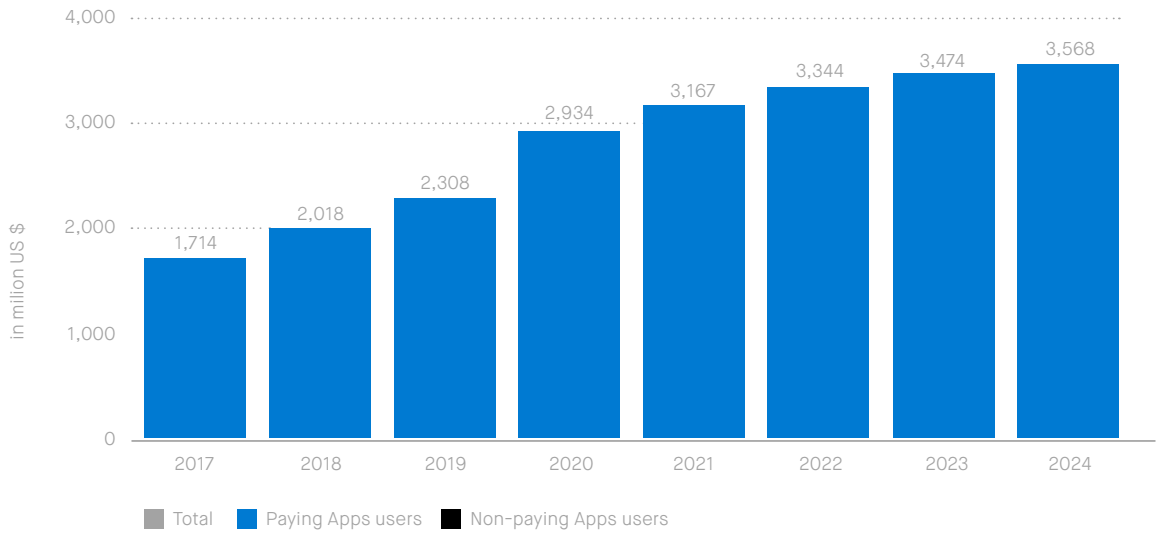
Content



_An introduction to the mobile app ecosystem

The Apple App Store first made its debut in 2008, shortly after the launch of the first iPhone. Although 500 apps seems miniscule by today's standards, that collection - and its immediate success - marked the true start of the fiercely dynamic mobile app scene. After this point, it wasn't long before Google, Blackberry, Microsoft and Android joined the ranks of contenders vying to reach a segment of this market.

Now, Apple and Android are the clear winners of this race, responsible for delivering 1.84 million and 2.57 million mobile apps respectively to consumers globally. Total mobile downloads reached 204 billion in 2019 and latest estimates suggest that 90% of all mobile time is spent in-app, averaging 188.6 minutes per day versus only 9.3 minutes on the mobile web.



Go to source

A number of coding languages thread behind the scenes of these substantial download figures, helping to ensure that users receive a reliable, consistent experience and the latest innovations without delay. These languages make up every detail of how an app will look, feel and function. However, selecting the right code for a project requires careful consideration. On the one hand, apps built in languages native to the platform of choice present the opportunity to build a flawless user experience and take advantage of in-built hardware and software features seamlessly.

However, native apps only function with the platform they were designed for. If a mobile app needs to reach multiple user bases, the code must be written in multiple languages and developed in tandem with design, marketing and developers to ensure the finished product is consistent and meets the end-goals of the product. Although possible, companies often have to negotiate a fine balancing act between the available budget and the project deliverables.

Cross-platform solutions are gaining significant traction due to their ability to address this challenge by running across multiple operating platforms from a single code base.

As a result, the overall complexity, length, and costs of mobile development projects have the potential to be reduced. In reality, the right choice will always depend on project specifics and requires thorough planning and thought.

This whitepaper explores the coding languages used to create mobile apps, the realities of development projects, and the new generation of cross-platform solutions on offer.

The great iOS - Android divide

Apps have become as indispensable as the smartphones on which they run, and are the operational mode of choice for mobile. Following the early mix of providers, development camps have clearly split into two dominant streams: Apple's iOS and Google's Android mobile operating system.

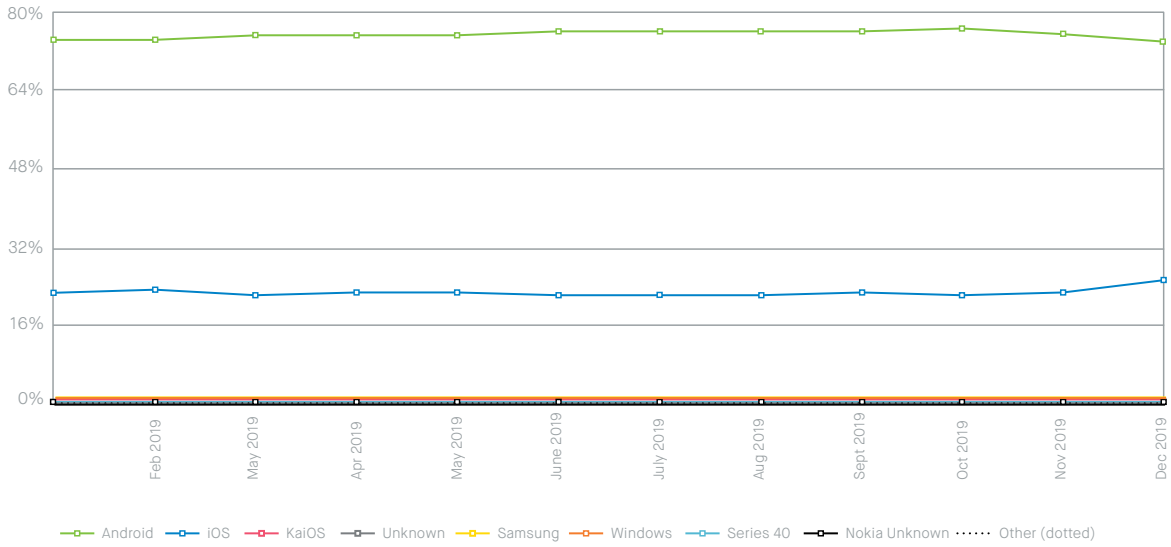
Google Android and Apple iOS jointly possess over 99% of the global market share, controlling 72.52% and 26.8% respectively, although the balance between these percentages varies across countries and regions.

Revenue in the apps segment

Mobile Operating System Market Share Worldwide - May 2020

Android	iOS	Samsung	KaiOS	Unknown	Windows
72,52%	26,8%	0,21%	0,21%	0,11%	0,03%

Mobile Operating System Market Share Worldwide Jan -Dec 201



Go to source

Apple

Apple continues to deliver apps through the App Store, although the availability on offer is now significantly higher. Its entire infrastructure runs on iOS, and Swift and Objective-C (also known as Obj-C are the coding languages at play.

Obj-C is a subset of the C programming language and has been used to support the Apple platforms for over 30 years. The latest codebase, Swift, was brought into the picture in 2014. Both languages are still supported by Apple, but Swift has largely replaced Obj-C in broad range and is now the leading language for iOS development, and highly enforced by Apple itself. The language is less verbose, boasts better security, reliably supports newer feature integrations such as wearable tech, and offers a modern, interactive experience.

As Apple continues to retain ultimate control over their hardware and software, the closed system supports high usability. Mobile app developers can roll out updates and features knowing that languages will interact with all users using iOS. What's more, developers can take advantage of Apple's strong security features such as FaceID, and connect to in-built hardware like cameras to offer a smooth user experience.

It's worth noting that Apple's app approval process is notoriously lengthy and ever-changing, which can sometimes increase overall development time and place limitations on flexibility.

Android

Initially launched independently, Google leads the Android Open Source Project. The whole Android OS is not hardware specific and is used by a wide range of smartphone manufacturers and has become notorious for its open-source approach to development. This provides mobile app developers in this sphere with a wide range

of readily available frameworks, plugins and features which can be used to get new projects off the ground quickly and economically.

However, the staunch dedication to a free and open system also creates its own unique set of problems. Android app developers must navigate the challenge of writing code which is compatible with the wide variety of hardware integrations and systems in use by the countless brands. This issue also spills over into testing and software updates which are hard to deliver with consistency across the entire ecosystem at once. Aside from impacting development time, the open-source framework has been plagued with malware and hacker attempts which could easily compromise customer data if not handled well.

Android platforms historically run on Java, which performs well across web, desktops, mobile and IoT products, making it a diverse coding language with many other potential applications. Android also supports Kotlin, a new, more concise programming language to better serve native mobile app development. Kotlin is a continuation of the Java codebase that boasts a simpler language and delivers better type handling, modern error handling and greater flexibility, which Google confirmed as the official second language for Android in 2017.

A new generation of development tools

Even with the most enticing app on offer, most people are unlikely to switch platforms. Users are historically married to one camp, with Android devotees vocally passionate about its open-source modular nature and no-nonsense approach, and die-hard Apple fans dedicated to its slick and notoriously high-performance software.

Although the specific KPIs will vary according to use case and industry, it's evident

that mobile applications continue to drive revenue growth across sectors and - in today's remote economy - have become increasingly business-critical. Apple and Android both boast access to millions of mobile users, but maximizing this figure to its highest potential invariably means delivering a solution that works for both. However, this also means that developers need to deliver dual code bases, dual design interfaces and maintain dual updates, significantly increasing the scope of work. But it doesn't need to remain that way.

In efforts to bridge the gap between the iOS and Android OS user bases in a more efficient way, a new generation of cross-platform mobile app development solutions has emerged.

These promise to offer a non-platform specific solution that can also deliver a slick, seamless in-app experience without the headache of developing native apps in tandem.

This new generation falls into three main camps:

Hybrid app development

Hybrid solutions use standard HTML, CSS and JavaScript web technologies as a codebase then wrap these in a native shell to connect to mobile operating systems. Ionic, Apache and Phone Gap are developer tools that work using a hybrid solution.

Cross-platform app development

Cross-platform solutions use a native rendering engine to enable non-platform specific functionality. Apps are built in a single codebase which then uses a bridge architecture to link with the native OS. React Native, Xamarin, Flutter, and Kotlin Native are all examples of cross-platform development options.

Progressive web applications

Progressive web applications are the newest contender. Technically they're closer to mobile websites and built using JavaScript, yet they appear on mobile with the same icon interface that apps deliver. Angular, React, and Polymer are the current options for progress web applications.

Cross-platform app development means tangible gains:

_Reduced need for domain experts

_Reduced development costs

_Faster time-to-market

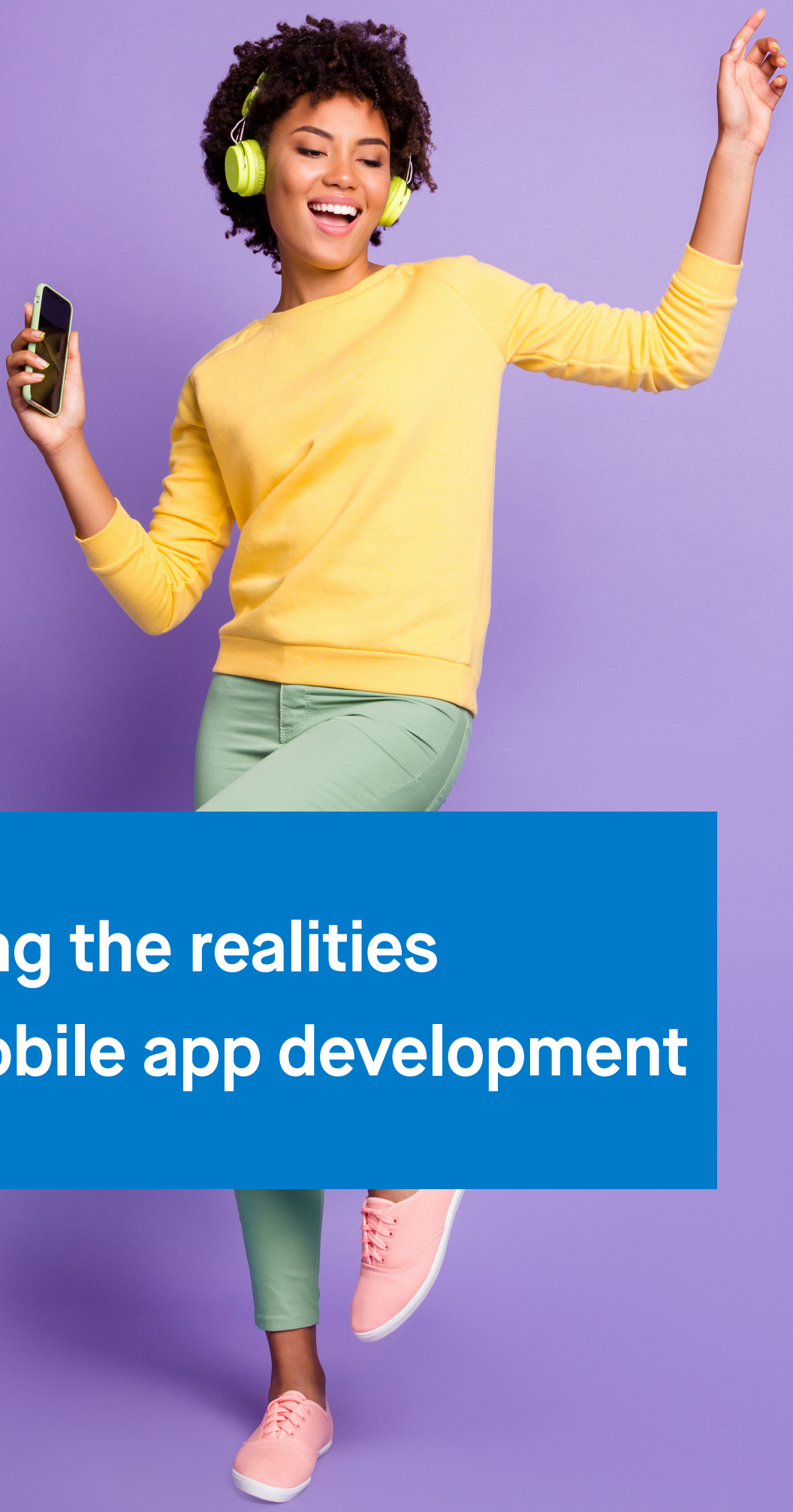
_Multi-platform support

All the users with less effort?

Maximizing user figures without increasing overheads sounds like the ideal scenario, but do these newer mobile technologies live up to expectations, or are native solutions still the best choice?

In reality, there are benefits and drawbacks to each approach which must be closely considered in alignment with the goals of the app, the expectations of the users and the external market demands, to name a few.

To fully understand the right pathway, we need to understand the realities of mobile app development.



_Facing the realities of mobile app development

Maintaining native apps and their codebase

Native apps offer developers the chance to produce an app that's "just right" for the target platform. All features and design elements can be completely optimized and integrated with device hardware smoothly.

Native offers a clear edge for high performance apps, but the same dedication and input must be replicated in two separate sets of codebase if the project aims to reach both iOS and Android users for maximum exposure.

This means that a dual native experience will require a significantly larger scope of work, especially if the main driver for going native is to focus on a complex, high performance app. The product will need to be developed in full using both iOS and Android languages and replicate this duality to cover any subsequent software updates or new features. In addition, app providers usually strive to deliver the same product experience for both camps of native users which requires a tailored approach and thorough understanding of the coding languages in use.

While those not versed in back-end frameworks may assume all coders can easily switch between languages, in reality most specialize early on in their careers. Attempts to write code in a native language without proper training and experience run the risk of the final product being plagued with design and performance issues such as images overflowing screens and sluggish performance. With mobile app users expecting slick, high-performance products that also look pleasing to the eye, these issues are practically unthinkable for any app that wants to prioritize quality above all else.

Cross-platform apps are able to address a number of these issues and offer both native look and feel and consistent UI and UX for both operating systems. What's more, they do this while keeping the backend structure simple resulting in fast-loading content, data and graphics.

Developers can choose to employ comprehensive development frameworks that connect the app UX to device specific functionalities. The UX remains seamless even while fixes and updates are underway and users experience a fluid, native feel with no difference from one platform to the next.

The single code base of cross-platform apps also makes maintenance across the two platforms significantly simpler. Software updates only need to be written once and can be deployed and synced across platforms and devices in one unified process. This also applies to patch updates and bug corrections in the common code base of which 80% can be reused.

1/3

”

Roughly one third of mobile developers use cross-platform technologies or frameworks

Go to source

The need for agile development

Technology trends move quickly. The time in which it takes to respond to a significant shift in consumer preference or the market can make or break the success of an app. Security issues and hacker techniques also develop at a rapid pace. Quite often, new and additional app features need to be implemented in rapid turnaround times to respond to external demands, leaving limited time for writing and testing code.

Whether building a new product from scratch or developing a new in-app feature, doing so across two native codebases can hinder a fast response as efforts need to be duplicated in full. This either means moving swiftly just isn't possible or that additional investment needs to be pumped into the project to get more developer hands on deck.

Every mobile app should aim to deliver quality and consistency, yet when speed is of the essence, updates for dual native projects can become a lengthy task. Native apps are built and tested before they are deployed through the platform specific stores and therefore changes must be polished to perfection before updates can be launched. Do this too often, or for too long, and user numbers will start to dip.

Delivering an app for iOS and Android OS using a cross-platform solution reduced delivery times by up to 70% when compared to a native build in each language.

This is because developers no longer need to coordinate development between platforms. The reduced load in coordination - plus the quicker roundtrip during production - drives speed forward significantly, although it's important to note that apps should always be tested on both platforms completely for quality-assurance.

The true cost of developers

As native apps run on entirely different codebases, a team of varied domain experts need to be hired. Developers highly skilled in both native platforms do exist, but they're in high demand and require a top-draw salary. When choosing to develop dual native app solutions, most often two coding teams develop and run the app.

However, this approach creates its own set of challenges as the teams must ensure they work in tandem to achieve the same results for both platforms at the same time. As team members come and go, stringent management of the development processes and code changes must be implemented to avoid work taking place in silo and ensure that any new developers inherit clean and well-organized code.

To maintain a great product, domain experts still remain key across all facets of mobile app development.

However, with cross-platform, the size and complexity of the team structure required to make a strong app can be significantly reduced.

Early stage software as a service and low budget apps

Award-winning apps don't always require the biggest budget, but the majority of the most successful ones have a proof of concept, otherwise known as a prototype. Prototypes of apps serve to showcase the potential of an app idea, its level of popularity, and allow developers to iron out user kinks and optimize the experience. Ultimately, this initial testing is done to increase the likelihood of success once the app is fully

launched, and does this by allowing developers to continuously improve the product based on the feedback of real users.

Quite often, prototype apps are developed on a native code base. But, with all the work that goes into this, it does beg the question: Why invest the time and resources into creating two codebases for an app that's essentially a live test?

For those companies that experience successful prototype tests or proof of concepts that take off, their app will need to be rolled out quickly to meet the demands of both Apple's iOS and Google's Android OS users and maintain momentum.

For a new or early-stage company, the costs of quickly rolling out a native app for each platform can often be too high. Cross-platform development has a clear use case here, both by allowing for rapid dual app-building so companies can quickly test the potential of their product, and minimize costs while doing so.

How cross-platform development drives business gains

The benefits of cross-platform development for the development teams are clear. How do these relate to overall business gains?

Reduced need for domain experts

While native app development requires multiple domain experts for each platform, the need for these specialists is reduced when building a cross-platform app.

Native app experts are still necessary within cross-platform development: In order to build apps - even cross-platform ones - developers need to have good knowledge of the framework and platform implementation, including how aspects of the app such as audio output and interaction with notifications occur on the device itself. There also always needs to be another developer with knowledge of the same native platform so that code can be reviewed.

However, the level of in-depth knowledge and number of experts is reduced with cross-platform development. Native app developers don't come cheap or easy to source, so being able to trim costs on this front will aid those businesses building an app within a certain budget.

Reduced development costs

While not universally applicable, the age-old cross-platform mantra of "write once, run everywhere" does hold some value. A single codebase for cross-platform development allows businesses to significantly reduce the costs associated with launching and maintaining an app in multiple platforms. That said, it's important to note that the level of applicability of this claim depends on the scale of the project and the number of necessary integrations.

Faster time-to-market

When it comes to launching apps, he who is faster, wins. By being able to build an app on multiple platforms faster than their native-building counterparts, companies can release their minimum viable product before their competitors. With a single codebase, companies can easily make changes or customize the app once it is out, giving them a double edge against competing products.

Multi-platform support

As cross-platform apps are able to support and work seamlessly across multiple mobile platforms, they are likely to receive support for new operating systems and their updates.

Where cross-platform development falls short

While it holds many benefits, cross-platform app development is not a fix-all solution with universal applicability. It's vital to consider the projects in which a cross-platform approach is not yet the best choice for businesses looking to build a premium-quality app.

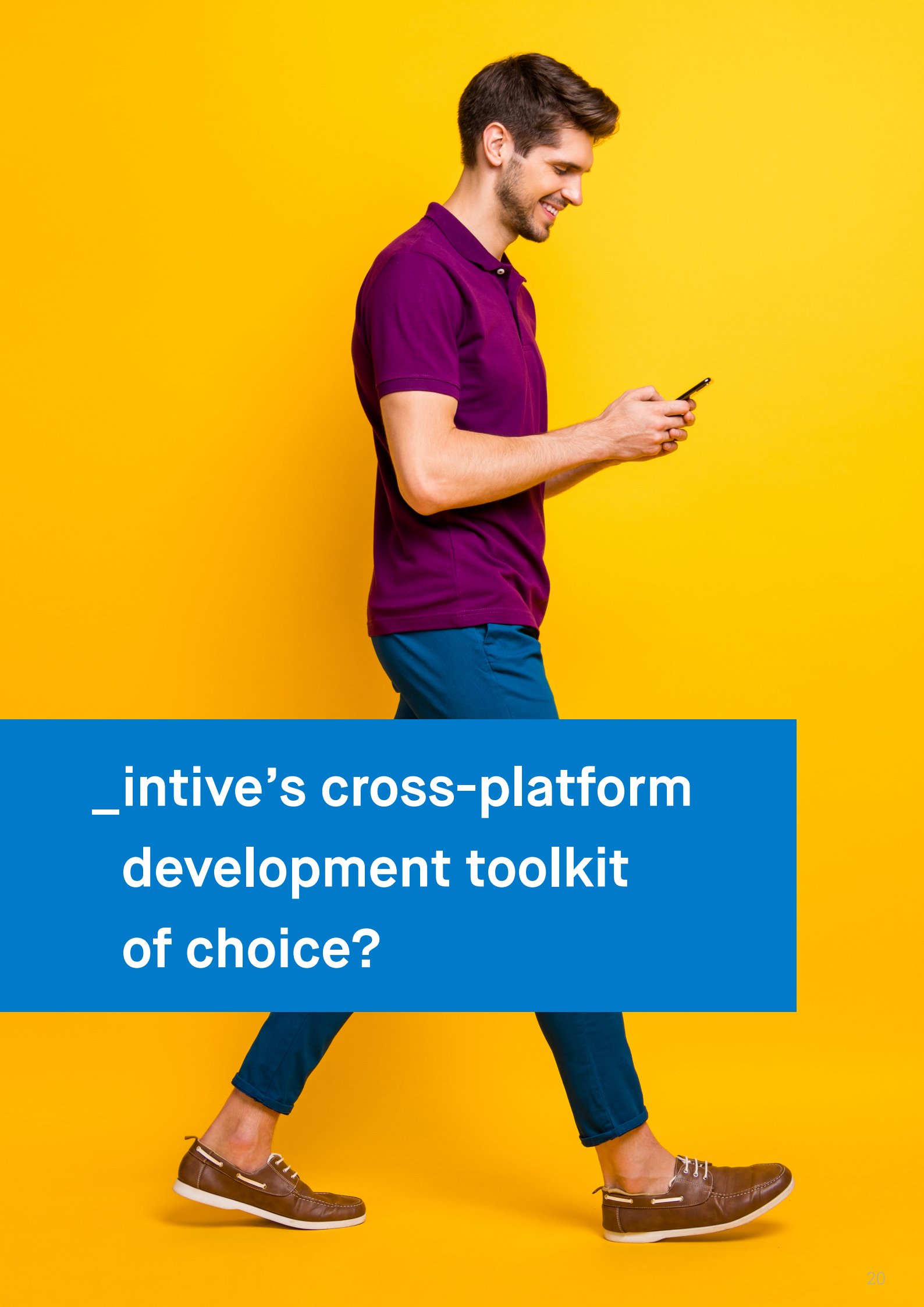
For example, app features that leverage augmented reality (AR), virtual reality (VR), or other forms of 3D rendering are not suitable for cross-platform development. This is because these features would need to integrate with fast rendering using the native framework. In addition, apps that require use of cutting-edge features of mobile phones like cameras and audio recording, or have planned integration with other hardware such as wearables or a TV, would not be appropriate for cross-platform development.

This is because native app development runs "closer to the metal," allowing for more optimal hardware integration. Ultimately, the more hardware and operating system features that are required for the app's functionality, the more it is suited to native app development.

Before making the decision between cross-platform and native, businesses must

consider not only the additional dependencies that the app may have on the hardware and operating system, but also the lifetime of the app.

While Apple's iOS and Google's Android OS are unlikely to disappear in the next few years, those additional tools used to build cross-platform apps do not have the same guarantee. This means organizations looking to build an app that's good for years into the future should consider whether using the cross-platform approach - which means having an additional technology layer to depend on - is right for them.



_intive's cross-platform development toolkit of choice?

Flutter

Flutter is a software development kit (SDK) from Google that enables cross-platform mobile app development and brings a number of new benefits to development teams, including expressive and flexible UI and native performance.

Flutter is intive's toolkit of choice for cross-platform app development for many reasons.

Hot Reload feature

Flutter's Hot Reload feature enables a much quicker development cycle than compiled languages, allowing developers to easily and quickly experiment, build UIs, fix bugs, and add features. It works by inserting updated source code files into the active Dart Virtual Machine (VM).

The platform contains just-in-time (JiT technology, which allows for a real-time connection between the development tool and the simulator. This means that all the developer has to do is change the code and hit "save," and they can instantaneously see the edit to the code in action. This allows them to make real-time changes which would have previously had developers waiting to see them visibly functioning.

This boosts the overall time it takes for developers to build an app, making Flutter's time-to-market length difficult to beat.

Cross-platform unity

Due to its single code base and customizable widgets, Flutter allows for cross-platform app development, enabling developers to maintain a natural look and feel on each platform.

The range of widgets available on Flutter, including the Material Design and Cupertino widgets, imitate the platform itself and lets developers build seamless, native interfaces more efficiently than ever before.

Dart is concise, powerful, and relatively easy to learn

Dart, the coding language used within Flutter, was specifically designed to speed up development tasks, meaning it's concise and relatively easy to learn. Because Dart contains features that are familiar to both static and dynamic language users, developers find it simple to grasp. This helps trim down the overhead time that they might usually spend to become fluent in a new, complex coding language.

With its rich set of libraries and tools, developers can build highly responsive applications. The language also simplifies common programming tasks and supports inheritance, interfaces, and optional typing features. This is further boosted by Dart's support of Ahead of Time (AoT) compilation, to create fast, native code which allows for easy customization. Dart can also be Just In Time (JIT), which allows for truly fast development cycles and agile workflow.

When is Flutter the right choice?

Flutter is an excellent choice for MVP versions of apps that need to be built quickly. Its single codebase means that apps built with Flutter can be put to market faster, allowing you to focus on the overarching product idea and not worry about the framework specifics. A single development team here brings added consistency and agility.

With Flutter, there is far less overhead when rendering 2D interfaced. In fact, it's even more efficient in rendering interfaces than many native apps.

Where is Flutter not yet ready?

Using Flutter to develop apps that feature technologies such as AR or VR would not produce results equivalent to those achieved through native development, and would likely complicate processes more than necessary. It's also worth keeping in mind that while Flutter is able to take care of basic maps functionalities, such as showing annotation markers or drawing polygons on the map, it does not yet offer the same range of capabilities as native libraries.

Flutter also has limited support for digital rights management (DRM) and video playing. While there's little doubt that Google will ensure full functionality to accommodate this in the future, videos played on apps developed with Flutter are not as seamless as those built on native platforms.

In addition, Flutter may not be the best option for companies building enterprise applications for internal use. While Flutter is able to leverage native APIs, the effort to do this is more than would be spent doing the same during native development. If a company chooses to use only one type of operating system (that of Apple or Android), there is no need for a cross-platform toolkit. Traditionally, these types of apps are built on the same operating system for employees' work-specific devices and would have a longer lifespan than most other commercially available apps, making them better suited to native development.

The power of Flutter is still being realized. However, there's no doubt that its potential will only expand further and empower businesses of all sizes to creatively and efficiently build high-performing apps.

Where can we see Flutter in action?

Flutter has already proven itself as capable of handling big, demanding applications. Let's take a look at a few applications that are built with Flutter.

Google Ads

This app allows users to manage their Google Ad campaigns directly from their smartphone. The app's features include campaign stats, options to update bids and budgets, real-time notifications, and keyword editing. It also allows users to contact a Google expert through the app.

Tencent

Chinese conglomerate Tencent uses Flutter in the development of several of its apps, including AITeacher, Now Live, K12, Mr. Translator, QiDian, and DingDang. The company specializes in internet entertainment and AI, and leverages Flutter to help it deliver seamless app experiences to its customers.

Nubank

Mobile-first Brazilian fintech Nubank chose Flutter as its main technology for app-building. This choice allows the development process to scale and remain agile in the face of a lack of native mobile specialists. Learning one cross-platform platform instead of writing for two reduces the entry barrier for backend developers to contribute to the mobile frontend. Members of the Nubank team concluded that Flutter's testing ability fit well with their mindset and objectives.

As of the company's April 2020 update, Flutter has two million users and counting. There are around 50,000 apps published in the Play Store which have been built using Flutter, with 10,000 added in March 2020 alone. Top development countries using Flutter include India, China, and the US. Dozens more examples of Flutter use cases can be found in Flutter's showcase section of its website.

How to select the right development toolkit

As covered in this paper the history, and future, of mobile app development covers a complex and ever-evolving landscape. Selecting the perfect toolkit for mobile app development is highly dependent on the project scope.

In order to make an informed decision, a comprehensive development brief should always be the first step.

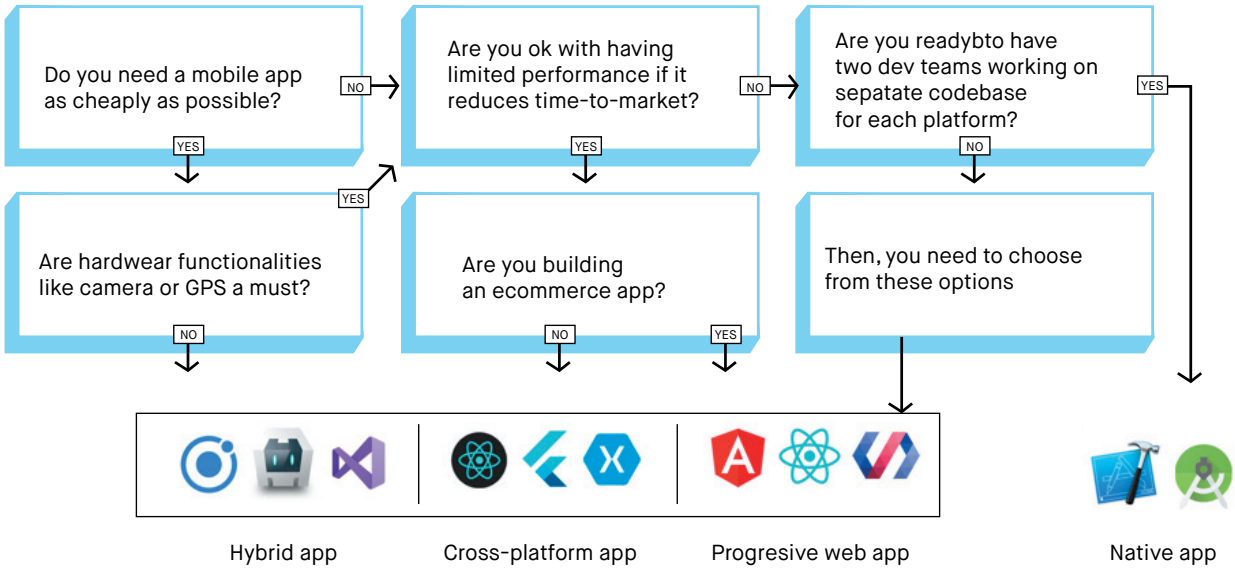
An ideal brief will examine the use of the app, its target market and competitors, which hardware integrations are required for special features and the capabilities of the operating systems of choice. The length of the project, time to market, available budget and planned lifetime of the app also influence this decision. Finally, outside influences such as the amount of competition for developers in the chosen code base, the support offered by each of the technology providers and the stability of these providers are also essential to make the smartest development plans for an app.

New cross-platform solutions have significantly increased the variety of options available for mobile app development, which makes it even more important to fully understand the whole ecosystem and influence the range of choices will have on the final product.

Cross-platform offers consistency, speed and reliability for users and expresses a flexible UI which appears as native for both camps.

However, cross-platform also performs best with simpler apps that don't require complex hardware integrations. Ultimately, careful planning and research is essential for all mobile app development in order to get off the ground with a solution that will provide results and longevity.

Choose a dev approach for your mobile app



intive

WHITE PAPER

Authors:

Florian Fetzer, Principal Software Engineer, intive

Klaus Busse, Senior Principal Software Engineer, intive

CONNECT WITH US AT: intive.com

About intive

intive is a global digital powerhouse headquartered in Munich, Germany. With a challenging, exploratory and agile mindset and international teams of software, design and business experts, intive partners with market leaders in accelerating digital transformation for products and services of tomorrow. intive's solutions driven by world-class innovation add value along the whole application lifecycle – from product conceptualization to its maintenance. The company's operations include 12 development centers and 6 design studios in Germany, Poland and Argentina, as well as regional offices in the UK and the USA. intive won the trust of the world's largest companies such as Audi, BASF, BMW, Credit Suisse, Discovery, Disney, Esprit, Facebook, Google, ING, Viacom and Vorwerk. Learn more at www.intive.com