Preview:

Mobile technology has revolutionized how we interact with the internet, with most of the internet usage now done via mobile devices. As a result, businesses have had to adapt and provide most of their services through mobile apps. But what is the best way to build these apps?

The decision of whether to use cross-platform or native technology can be a tricky one. While cross-platform technology offers cost and time efficiency, certain industries may still opt for native development over cross-platform.

Don't miss out on our latest article https://bit.ly/44GzyQN where we delve into the pros and cons of native development and take a closer look at the top programming languages for native apps.

Preview - INDIA:

As the world becomes increasingly mobile-dependent, businesses are looking for ways to provide their services through mobile devices.

While cross-platform technology has become popular in recent years due to its cost and time efficiency, there are still cases where native development is the way to go. They include applications requiring specific performance, native functionality, or security.

To gain a deeper understanding of this topic, check out our latest article at https://bit.ly/44GzyQN , where we delve into the advantages and disadvantages of native development, and examine the top programming languages for native apps.

For twitter:

Mobile tech revolutionized internet usage, pushing businesses to adapt with mobile apps. Choosing between cross-platform and native tech can be tricky. Check our article at https://bit.ly/44GzyQN for the pros/cons of native dev and top languages.

For threads:

Businesses go mobile, using cross-platform tech for efficiency, but native development is essential for certain apps needing performance, native features, or security. Read our article at https://bit.ly/44GzyQN for more on native dev pros/cons and top languages.

Businesses go mobile, choosing between cross-platform and native tech. Explore native app pros/cons and top programming languages athttps://www.scrumlaunch.com/blog/top-programming-languages-for-native-apps .

Top programming languages for native apps

Peter Rojas, Co-founder of Engadget and Gizmodo, once said, "The majority of internet usage will be done via a mobile device, and for most people, the mobile web will be their primary, if not their only way of experiencing the internet". And this time has come. According to a Statista survey, 63% of organic search traffic comes from mobile phones. So, mobile is the way businesses now provide most of their services.

As we mentioned in our previous article, "The State of the Mobile Market: React Native Supremacy, the Fall of Ionic and What To Expect from Flutter?", the choice of whether to use a cross-platform or native technology when building a mobile application in 2023 isn't very difficult. With the appearance of React Native and Flutter frameworks, cross-platform technology will be the right choice in about 90% of cases. The main reason that stands behind cross-platform development popularity is that it is cost and time efficient. With one code required to run an app on several systems, the development costs go down, time to market gets shorter, and an app reaches a wider audience easier. Due to these advantages offered by cross-platform technology, some applications have their chance to be developed, as otherwise, that would not even be possible.

Nonetheless, there are exceptions in cases where some very specific performance, particular native functionality, or security is required. Even with the availability of React Native and Flutter frameworks, native development offers higher security measures and the possibility for an app to work offline. It also provides a better user experience, achieved by full access to a device, its operating system, camera, microphone, GPS, and more. Any of the above can be a crucial reason for some businesses to opt for native development despite higher costs and longer time to market. For instance, iGaming, FinTech, and healthcare applications usually contain confidential or sensitive data concerning banking transactions

or health states. Native mobile apps are more secure than cross-platform ones as they leverage the in-built security features of a device's operating system. Security measures include authorization methods like biometrics and two-factor authentication implemented through device-native functions (like Apple's Face ID) and SSL certificate pinning. For other applications, it might be crucial that they work offline. For instance, travel applications are dependent on GPS navigation. Native development allows such applications to continue to work efficiently without an Internet connection. A great example of such an application is Google Maps. It allows users to download maps for a particular region locally without an internet connection, which is a highly relevant feature for any travel app.

So, if budget and urgency are not crucial, and you have ambitious plans to develop a stable, reliable, scalable product with the best security from the very start, then consider going for native development.

When it comes to the choice of programming languages, there can be multiple programming languages for building native iOS and Android apps. Yet **iOS native app developers would use Objective-C or Swift**, while **native Android apps are typically created using Java or Kotlin** language.

The iOS programming languages battle is mainly between Objective-C and Swift, as Apple backs up both languages. Apple has been supporting Objective-C and continues to do it now. It also created its own language Swift. And it is evident that Apple integrates the best functionalities into its programming tools. The development of iOS apps started with Objective-C, and for years, it was used for creating high-quality, dynamic iOS apps. But technology tools are not static. Objective-C supremacy ended in 2014 when Apple introduced Swift at Apple's Worldwide Developers Conference as a replacement for its earlier programming language Objective-C. Since then, Swift has been the go-to option for building iOS, Mac, Apple TV, and Apple Watch apps. It is a powerful and intuitive programming language, and, as suggested by its name, it offers a faster development process. It also enhances security and performance and automates memory management.

Swift achieves a faster development process via its simplified syntax and grammar. It is very concise, meaning about 60% less code is required to perform the same task compared to its predecessor, Objective-C. Using Swift both on the back-end and front-end allows extensive code sharing and reusing. The development process becomes faster, meaning fewer expenses on the development efforts. Apple website states Swift is up to 2.6x faster than Objective-C. Swift dynamic libraries increase an application's performance as they can be uploaded directly to memory, reducing the application size. The programming language was designed from the outset to be safer than C-based languages. Its robust typing system and error handling prevent code crashes and errors in production. It eliminates entire classes of unsafe code. Variables are always initialized before use, and arrays and integers are checked for overflow. Thus, Swift allows developers to see the errors in the code instantly and fix them right after detection. It reduces the time and effort required for bug fixing and excludes the risks of deploying low-quality code. Swift uses Automated Reference Counting (ARC), tracking and managing the app's memory usage. It defines instances no longer in use and

deletes them, so developers don't need to spend time and effort doing that manually. With the cleaned-up memory, the app performance increases significantly.

Objective C, a predecessor of Swift, is now considered an old-school player. Being in the market for a long time, Objective C is very stable for development. Before Swift was released, it was one of the best iOS development languages. But it has been essentially unchanged since it was built in 1984, and currently, it lacks some features and functionalities that modern projects demand. So, it's time that its progressive successor Swift replaces it. JetBrains statistics published in 2021 confirms Swift dominance, showing that out of all developers participating in their survey, 76% use Swift, 13% use both Swift and Objective-C and only 11% use Objective-C. Stack Overflow survey results released in 2022 also confirm that over 63% of programmers use Swift for mobile app development and love using it, whereas, for Objective-C, the percentage is 23%. It is worth mentioning that Swift language is absolutely compatible with Objective-C. So for large projects that are being extended or updated, it is possible to use the existing Objective-C codebase, supplementing it with features in Swift.

So, **Swift is the apparent winner in the iOS programming languages battle**. It has undoubted benefits over its main competitor Objective-C. It has a gentle learning curve. It also has a large and rather vibrant developer community that quickly formed around the language. It is your right option for iOS native development in 2022.

The programming languages battle in Android native development is somewhat similar to the iOS battle. It is between the two most popular languages intended specifically for Android - old and mature Java vs. young and modern Kotlin. But unlike the iOS battle, here, the competition is more rigid. Based on the survey done by Stack Overflow in 2022, already mentioned earlier in the article, over 63% of programmers are using Kotlin for mobile app development and love using it, whereas, for Java, the percentage is 46%. If we compare these results with the Stack Overflow 2021 survey, it appears Kotlin percentage has risen about 2%, while Java, on the contrary, has decreased by about 1%. The difference is insignificant, yet it shows some tendency and increases the total usage gap between these two languages.

So, old and mature **Java** is an object-oriented programming language released in 1995. It is the world's third most popular programming language, after Python and C – according to the TIOBE index, which evaluates programming language popularity. It has gained incredible popularity because of its performance, simplicity, open-source libraries, and wide range of possible applications. Java helps not only with mobile development but also with many different tasks, including desktop development, back-end operations, and video games. It's not always the best choice for everything, but it can do almost anything.

Java was the only way to develop Android apps at the beginning of Android. In 2022 it continues to remain the top Android app development language with tremendous community support. It is an independent language running on any platform. It is stable and has strong security measures as well as reusability features. It supports multi-threading,

which means executing several tasks simultaneously, thereby allowing CPU usage to the maximum. At the same time, Java code structure is very complex to be the programming language of choice in 2022. It also consumes a lot of device memory, and overall, developers consider it a pretty slow programming language.

Its young counterpart **Kotlin** was developed by JetBrains s.r.o. in 2011. But it started gaining extreme popularity only after Google announced Kotlin as the official language for Android app development in 2017. Kotlin is a statically typed language that can run on the Java virtual machine. It is a cross-platform open-source multipurpose programming language that combines both acquisitive and functional programming features. In contrast to large Java code, Kotlin code is about 20% shorter. It significantly speeds up the development and deployment processes. With less code, developers make fewer mistakes, and as a result, codes have fewer bugs. It also supports many modern features like lambda, smart class, operator overloading, null safety, etc.

Kotlin compiles with Java and is interoperable with it. In a nutshell, Kotlin is a streamlined version of Java, its advancement, rather than an entirely new language. It has part Python, part Swift and part Java-like syntax and other Java-similar capabilities improving on many weak spots. However, its learning curve is not that easy. On top of that, Kotlin has a young community, and finding answers to questions that may arise during development can be challenging.

Unlike in the iOS battle, in the Android battle, there is no ultimate programming language winner. In 2022 and expectantly in 2023, both Java and Kotlin are good options for native development. Each language has its share of dedicated fans. From the developers' view, both programming languages should be in Android developers' arsenal. Without Java, there would be no Kotlin, and with Java knowledge, a developer can easily shift from Java to Kotlin. From the future application owners' view, Kotlin may seem more modern and offer faster development. Still, there are not so many experienced Kotlin developers for now that it can have just the opposite effect on the development time.