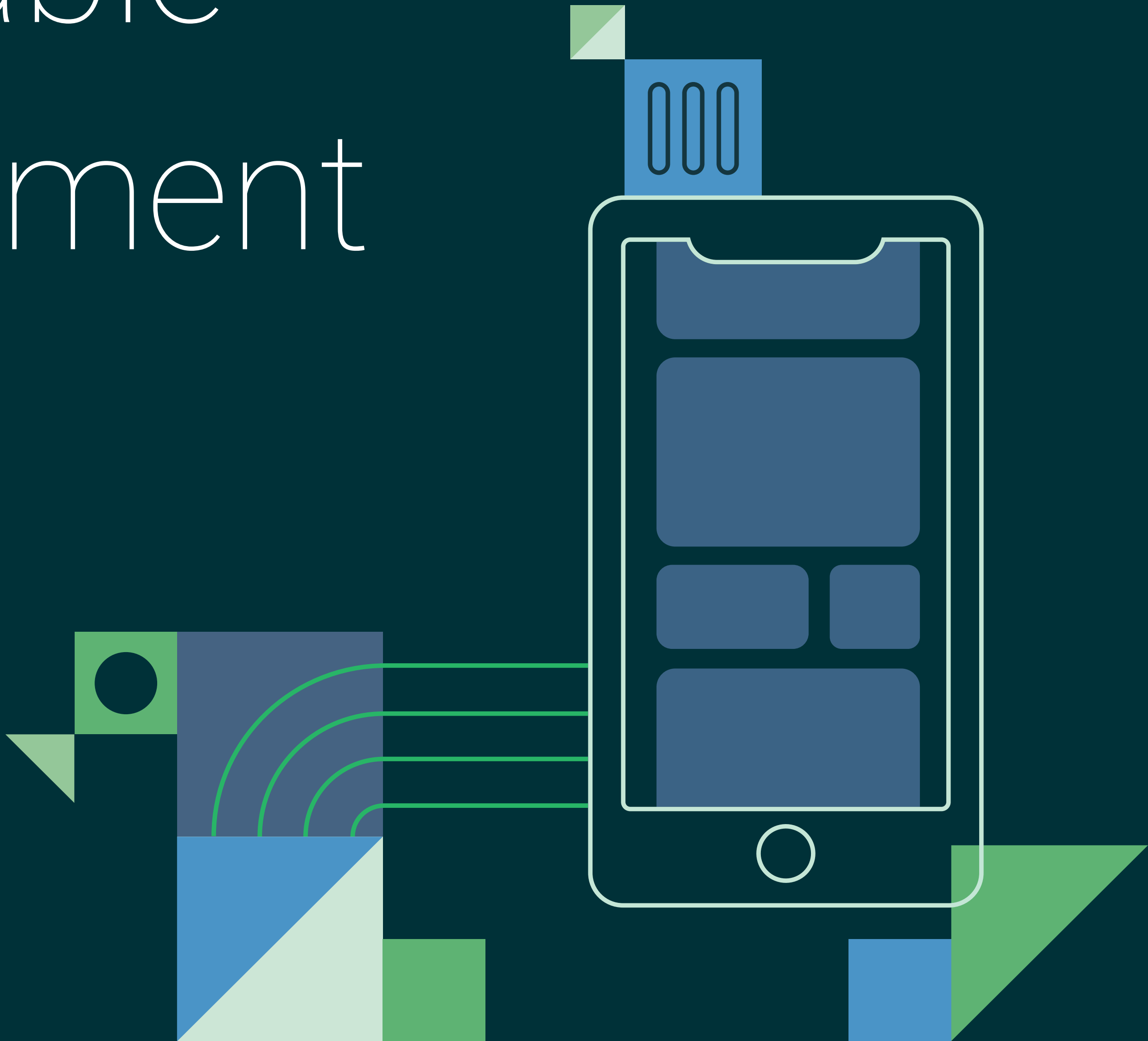
circleci

Building a Capable Mobile Development Tech Stack



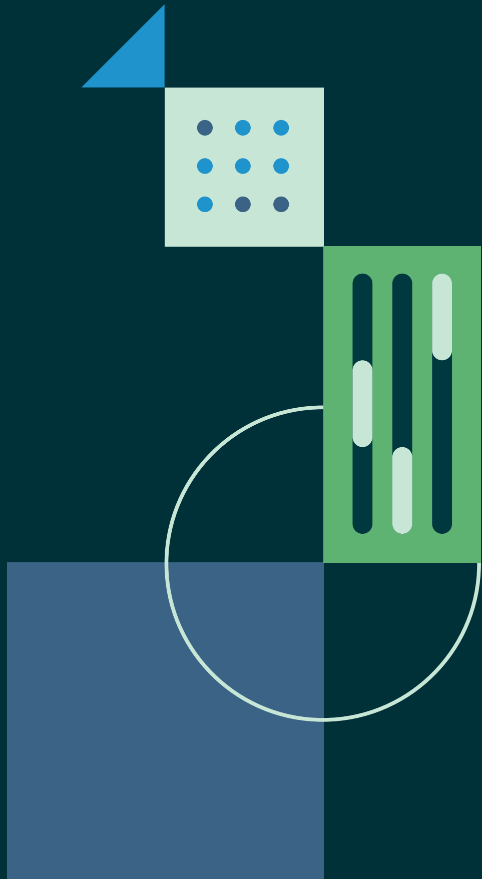
Introduction

Developing a successful software solution starts with using the best building materials and tools for the job. You must select the right programming language, database, framework, user interface — and the list goes on. It is a challenging task that requires many decisions, and it can quickly become overwhelming.

But selecting the technology stack is a crucial first step in any software development project — especially for mobile applications. The tech stack you pick needs to support both a development-friendly environment and a production-ready application. Getting it right is critical to the type of application you can build, its scalability, viability, and ultimately, profitability.

There are several tech stacks to pick from when beginning a project, and they are not all made equally. Knowing how to select the right one helps you ensure the quality of your product and that it meets business needs and customer expectations. It can close the gap between a good idea and a successful mobile app.

Defining mobile app dev goals



You may know what features you want in your mobile application but may not have as much clarity regarding your development goals. Choosing the right tech stack starts with defining app goals and objectives.

Every business has unique objectives, requirements, and challenges, and the best technology stack for one app will not necessarily be the same for another. So, to decide what works best for your team, you should consider the following things when determining your development goals.

Application requirements

Start by determining the application's requirements. Prepare a requirements specification document outlining the application's technical, functional, and non-functional requirements. This provides a solid basis for choosing the most appropriate technology for the application's future development.

A typical requirements specification document includes your business goals and objectives, user stories and user roles, features, functional and non-functional requirements, UI/UX designs, infrastructure, tech requirements, and dependencies and constraints.

Application lifespan

Your tech stack must be flexible enough to accommodate changes that increase functionality throughout your product's expected lifecycle. A good tech stack must meet not only your project's current needs but also its future needs. The faster your technology stack ages, the more often you will need to renovate, upgrade, or migrate your application. That is why you need to consider the life expectancy of your application, along with current development trends and how they might evolve over the next few years.

Security

These days, exponential increases in cyber threats mean that businesses are under intense pressure to improve application security. Security is one of the most important factors when choosing a technology stack.

It is crucial for apps that rely heavily on sensitive user data to use a secured technology stack that guarantees protection against unauthorized access. Evaluate your app's security requirements and ensure that adequate advanced authentication and authorization measures are in your selected tech stack.

Scalability

There are two types of scalability in application development:

Horizontal scalability

The ability to maintain optimal performance with an increased workload — more users, more requests — without complete restructuring or redesign

Vertical scalability

The ability to add new features and functionalities without disrupting the app's structure

For an application to be scalable, the technology stack must be able to handle increasing traffic as the application grows. Choosing a technology stack that can support your project through every phase of change will ensure it can continue to succeed as it scales.

Development time

It is not only essential for you to build a high-quality digital solution for your customers, but you also want to do it within a reasonable timeframe. A quick development time means a short time to market, allowing you to collect feedback from early adopters, aiding future development, and enabling you to adjust your product goals and identify features that need improvement.

If your intended time-to-market is short, choosing a tech stack that supports preexisting solutions is crucial — you do not want to spend time reinventing the wheel.

You might prefer to use the Minimum Viable Product (MVP) approach, whereby you aim to first launch your product with the minimum feature set that delivers customer value, and then deliver additional features later. This means you need a technology stack with features that give you a working prototype as quickly as possible. For example, some tech tacks provide reusable code snippets or other ready-made solutions that make integration easy.

Team skill set

Choosing the right tech stack should be influenced by the available resources, including human resources. Consider the team's skill set, expertise, and familiarity with a technology stack before selecting it.

There may be good reasons to use a newer technology stack, but remember that it will come with a steeper learning curve for your team, increasing development time and training budget.

Also, consider the community of developers and users around the technology stack, as this indicates how easy it will be to hire developers with appropriate skills. If you prefer to outsource your project development, a strong developer community means you should not have too much trouble doing so.

Single-platform versus cross-platform

Mobile apps developed for a specific mobile operating system, like iOS or Android, are known as single-platform or native applications since they are designed in the native language of the target operating system.

Alternatively, you can develop apps with a single codebase that runs on multiple platforms, in which case they are called cross-platform applications.

Ultimately, either approach may be viable for your project, but each has its pros and cons.

Single-platform

Pros

- ✓ It's possible to target audiences on specific devices, such as iOS or Android.
- ✓ Apps are built in the platform's native language and therefore fully optimized for that platform.
- ✓ You can easily access a wide range of device-specific capabilities and features, such as GPS, camera, and gestures.
- ✓ Native applications provide a richer, more responsive user experience.
- ✓ Platform-specific development and testing tools are better supported and more widely available.

Cons

- ✗ Limitations regarding reaching audiences on other platforms due to compatibility issues.
- ✗ Different tools and languages are required for each platform.
- ✗ Costly to create a single-platform application because separate developers are required for each platform.
- ✗ Your development team must develop and maintain different codebases for each platform.

You should consider a single-platform if:

- You want to use specific hardware, and your application requires complete access to the mobile device resources.
- You want your application to be highly responsive and performant.
- Your application must be scalable, easy to update and have the capacity to have new features added as it is developed.

Cross-platform

Pros

- ✓ They can be developed with a single development workflow, speeding development time and reducing time-to-market.
- ✓ A single codebase means less effort to develop and maintain.
- ✓ Only one team of developers is needed, saving on development costs.
- ✓ You can target a wider audience, increasing your revenue potential.

Cons

- ✗ There may be difficulties accessing native smartphone functions and features such as platform specific UI features, screen gestures, or taking advantage of specific hardware optimizations.
- ✗ There is less seamless integration of cross-platform applications with the target operating systems, compared with native applications.
- ✗ The app may not communicate effectively with the device's native components, which can result in sub-optimal performance.
- ✗ These apps are slower than native apps due to the added abstraction layer and rendering process required for multiple platform integration.

You should consider a cross-platform if:

- Your budget is tight and you are willing to accept a less performant app.
- You do not need to use a lot of native hardware resources or create complex logic.
- Your app idea must be launched within a short timeframe and you want to reach both iOS and Android users.

Languages and frameworks

Some languages and frameworks are unique to different mobile operating systems. These frameworks and languages are crucial in developing all types of platform-specific applications and other cross-platform mobile applications.

Let's look at the tech stacks for creating different applications for Android-specific, iOS-specific, and multi-platform apps.

Android

Android Studio

Android Studio is the official Android IDE. It offers a complete suite of development tools for building and debugging Android apps.

Java

Java is the official language of Android. If you understand Java programming, you can develop Android applications.

Kotlin

Kotlin is a cross-platform programming language and a Java alternative that you can use to develop Android apps.

Android SDK

You can use the Android software development kit (SDK) with standalone IDEs, such as Eclipse, to build and compile Android apps.

Android NDK

With the help of the Android NDK, you may use programming languages like C and C++ to implement some aspects of your Android app in native code.

Android UI

For building your app's GUI, Android provides a variety of prebuilt UI components, including structured layout objects, menus, UI controls, dialogs, notifications panels and other UI modules.

Jetpack Compose

Jetpack Compose is a toolkit that allows you to build native user interfaces. With Jetpack Compose, you can develop your Android app quickly, as it simplifies and accelerates UI development on Android.

iOS

iOS SDK

The iOS SDK is the official software development kit that Apple provides for developers to create native applications for iOS devices.

Objective-C

Objective-C is Apple's main programming language for OS X and iOS. The language is a superset of C, which provides dynamic runtime and object-oriented programming features.

Swift

Apple created Swift for building iOS apps, and it is a popular language among iOS developers. Code reusability, better memory management, and easier debugging are the main advantages of this programming language over Objective-C.

Xcode

Xcode is Apple's integrated development environment (IDE) for macOS, which allows developers to develop applications for macOS, iOS, iPadOS, watchOS, and tvOS. It offers a unified workflow for designing interfaces, developing code, testing, debugging, and submitting apps to the App Store.

AppCode

AppCode is an integrated development environment (IDE) designed for developing apps for Apple devices such as Macs, iPhones, and iPads. This smart IDE gives quick-fix suggestions to help developers navigate and analyze code on-the-fly. It is designed to improve developer productivity and has tight integration with Xcode.

UIKit

You can build iOS and tvOS apps with the UIKit framework. UIKit defines the basic behavior of applications and offers many ways to customize that behavior. Displaying content onscreen, interacting with it, and managing interactions with the system are all done with these objects.

SwiftUI

SwiftUI uses Swift's powerful features to make designing iOS user interfaces straightforward while encouraging innovation and developmental creativity.

Multi-platform tech stack

React Native

Facebook created this open source framework for developing Android and iOS applications in 2015. React Native apps are written in JavaScript through React's declarative UI paradigm and then rendered with native code that can interact with native APIs.

Xmarain

Xmarain is a cross-platform and open source app development framework developed by Microsoft. It is used to create native cross-platform Android, iOS, and Windows apps with a single common codebase using C# and .NET. An app built with Xamarin is designed to take advantage of native interface controls for a high-performance user experience.

Flutter

Flutter is Google's open source UI toolkit. You can use it to build natively-compiled applications that work across platforms — iOS, Mac, Android, Windows, and more — from a single codebase.

Unity

The Unity platform is a cross-platform game engine that includes an integrated development environment (IDE) for developing gaming applications using one codebase and deploying across more than 25 platforms, including iOS, Android, Mac, and Windows.

Firebase

Google's Firebase is a backend-as-a-service that synchronizes data in real time. Firebase is an API-based, schemaless data solution that significantly reduces the time required to build multi-platform applications for both web and mobile applications.

Hybrid apps

A hybrid application is an alternative to consider when choosing a tech stack. A hybrid app uses one coding language to build a cross-platform application. They are essentially web apps encased in a native app shell. This shell can connect to all the mobile platform's capabilities via an embedded browser. It behaves just like a native application and can be installed like one.

Hybrid apps are typically written using web technologies like HTML, CSS, and JavaScript. A few frameworks that can be used to build hybrid apps include React, Xamarin, and Flutter.

Hybrid app development is somewhat similar to cross-platform app development but is largely different. The only thing they have in common is the ability to share code. With cross-platform apps, the developer writes code once and reuses it across various platforms, whereas hybrid apps are web apps in native app containers.

Progressive web apps

Websites that function as apps are called Progressive Web Apps (PWAs). Users get an "app-like experience" in their web browser. The application can also access the device's hardware, such as the microphone, camera, and location. However, unlike native and hybrid applications, users do not have to download or install the app.

Some frameworks for building progressive web apps include:

Vue.js

A progressive JavaScript framework for performant and versatile web user interfaces

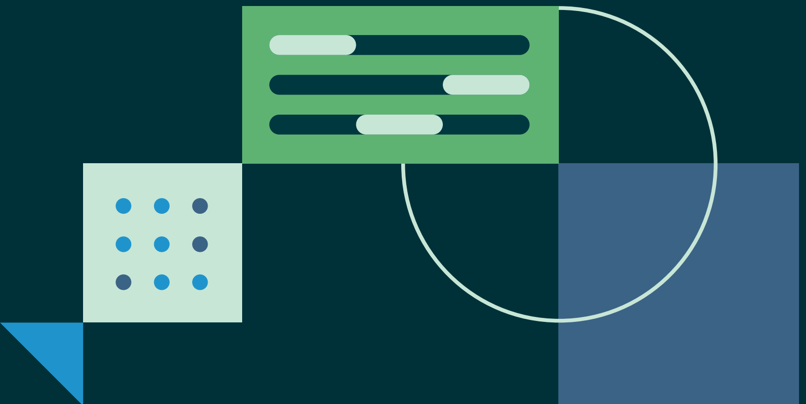
Angular

A framework for building mobile and desktop web applications

Preact

A lightweight JavaScript library with the same modern APIs as React

Accelerating mobile development with CI/CD



A constantly changing market landscape means mobile app development has become even more competitive. To achieve your business goals and reach your app development milestones, your team must iterate quickly and continually adapt the app to meet the needs and feedback of your users.

Therefore, adopting continuous integration and continuous delivery (CI/CD) has become a critical component of modern mobile development. CI/CD is an application development strategy that accelerates the mobile development workflow by allowing for quick iterations, rapid feedback loops, and automation of the entire development pipeline, ensuring system security and stability.

CI/CD also provides continuous monitoring and testing throughout the app's life cycle. Developers can catch minor code errors early rather than spending a lot of time looking through an expansive codebase later. Development pipelines can be significantly improved with CI/CD, increasing the velocity of the application development process.

CI/CD pipeline stages

A typical CI/CD pipeline has four pipeline stages:

Build

When code is checked in, a build process is automatically triggered (app compilation is initiated).

Test

Automated testing is done on the code, and various app functions are verified.

Deploy/Release

The code is validated and made available for deployment to production.

Monitor/Iterate

Real-world testing is performed to ensure that all security, compliance, and other requirements are fulfilled.

Introducing CircleCI

CircleCI is a purpose-built CI/CD platform that brings an integrated CI/CD pipeline to your mobile development workflow. Some of the benefits it brings include:

- Cross-platform support to build for iOS and Android using the same configuration.
- Flexible compute offerings for resource-intensive builds or special requirements like GPU testing.
- Up-to-date Xcode and Android images pre-installed with all required dependencies and tools.
- Support for nested virtualization and x86 Android emulators to enable Android UI testing.
- Seamless integration with third-party tools using orbs.
- Fast feedback and debugging with the Insights dashboard and SSH reruns to help teams resolve issues quickly and maintain high ratings in app stores.

Conclusion

The choice of technology stack should be dictated by your project requirements and the skills of your developers.

Despite the higher development costs, single-platform applications still offer the strongest performance and user experience. Cross-platform apps, by comparison, take less time and effort to develop, but more work is needed to provide the same quality of user experience across all platforms. Hybrid and progressive web apps are close alternatives to cross-platform apps.

However, choosing a tech stack is only half the battle. Today's market demands that you move quickly when developing and iterating your mobile application. There are several approaches to accelerating your application development pipeline, but CI/CD is one of the most productive. It helps your developers to build, test, and deploy new code quickly, all while reducing errors.

Get started with CircleCI today to learn more about using CI/CD to accelerate your mobile development projects.