UNIVERSIDAD MAYOR REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

FACULTAD DE TECNOLOGÍA

SIS (316)

TÍTULO DEL PRÁCTICO:

Escenario de Alineamiento Estratégico de TI

ESTUDIANTES:

Colque García Ariel Rodrigo

Cepeda Sebastián

Perka Casillas Celedonio

CARRERA: Ing. Ciencias De la Computación

MATERIA: SIS (316) Auditoria de Sistemas

Stack para el desarrollo de aplicaciones móviles para Android e iOS

Introducción

El desarrollo de aplicaciones móviles es un proceso que implica el uso de diferentes tecnologías, herramientas y plataformas para crear productos que funcionen en dispositivos móviles como smartphones y tablets. El stack para el desarrollo de aplicaciones móviles es el conjunto de elementos que se utilizan para construir una aplicación móvil, desde el diseño de la interfaz de usuario hasta la programación del código y la conexión con el servidor.

El stack para el desarrollo de aplicaciones móviles puede variar según el tipo de aplicación que se quiera crear, el público objetivo, las funcionalidades requeridas, el presupuesto disponible y las preferencias del desarrollador. Sin embargo, existen algunos componentes comunes que suelen formar parte del stack para el desarrollo de aplicaciones móviles:

- Front-end: son las tecnologías que se utilizan para desarrollar la interfaz que interactúa con los usuarios finales. Incluye los lenguajes de programación, los frameworks y las librerías que se usan para crear la apariencia y el comportamiento de la aplicación en el dispositivo móvil.
- Back-end: son las herramientas y el software necesarios para crear el procesamiento subyacente en el servidor. Incluye los lenguajes de programación, los frameworks, las bases de datos y las APIs que se usan para almacenar, gestionar y transmitir los datos y la lógica de la aplicación.
- Development: es una plataforma consolidada que proporciona las librerías y las interfaces necesarias para construir la aplicación. Incluye los entornos de desarrollo integrado (IDE), los editores de código, los compiladores, los depuradores y los emuladores que se usan para escribir, probar y ejecutar el código de la aplicación.
- Supporting: son las herramientas y las tecnologías que mejoran la seguridad, la flexibilidad y el rendimiento de la aplicación. Incluye los servicios en la nube, los sistemas de control de versiones, los sistemas de gestión de dependencias, los sistemas de análisis y monitorización y los sistemas de distribución y actualización que se usan para alojar, mantener y optimizar la aplicación.

Criterios para elegir el stack para el desarrollo de aplicaciones móviles

Para elegir el stack adecuado para el desarrollo de aplicaciones móviles, es importante tener en cuenta una serie de criterios que pueden influir en la calidad, la eficiencia y el éxito del producto final. Algunos de estos criterios son:

- Los requisitos generales de la aplicación: se debe definir el objetivo que va a cumplir la aplicación, el tipo de proyecto que se va a realizar, el público al que va dirigido, las funcionalidades que va a ofrecer y las limitaciones que pueda tener.
- La plataforma objetivo: se debe decidir si la aplicación va a ser nativa, híbrida o multiplataforma, es decir, si va a estar diseñada específicamente para un sistema operativo (Android o iOS), si va a usar tecnologías web para funcionar en varios sistemas operativos o si va a usar un framework que permita compilar el código para diferentes sistemas operativos.
- El presupuesto disponible: se debe estimar el costo del desarrollo de la aplicación en función del tiempo, los recursos y las herramientas necesarias. Se debe tener en cuenta que algunas tecnologías pueden ser más caras o más baratas que otras, así como las licencias, las suscripciones y los servicios asociados al stack elegido.
- La experiencia del desarrollador: se debe evaluar el nivel de conocimiento y habilidad del desarrollador o del equipo de desarrollo con respecto a las tecnologías que se van a usar. Se debe tener en cuenta que algunas tecnologías pueden ser más fáciles o más difíciles de aprender y usar que otras, así como la disponibilidad de documentación, tutoriales y soporte técnico.
- La escalabilidad y el mantenimiento: se debe considerar la capacidad del stack elegido para adaptarse al crecimiento y al cambio de la aplicación en función de las demandas del mercado y de los usuarios. Se debe tener en cuenta que algunas tecnologías pueden ser más flexibles o más rígidas que otras, así como la facilidad o dificultad para actualizar, depurar y mejorar la aplicación.

Stack nativo para Android e iOS

El desarrollo nativo consiste en crear una aplicación específica para un sistema operativo móvil usando las tecnologías propias o recomendadas por el fabricante. Esto implica usar un lenguaje de programación diferente para cada plataforma (Java o Kotlin para Android y Objective-C o Swift para iOS), así como un IDE diferente (Android Studio o Xcode) y unas librerías diferentes (Android SDK o iOS SDK). El desarrollo nativo ofrece algunas ventajas como:

- Un mejor rendimiento: al estar optimizado para cada plataforma, el código nativo suele ser más rápido y fluido que el código híbrido o multiplataforma.
- Una mejor integración: al usar las librerías propias del sistema operativo, el código nativo suele acceder mejor a las funcionalidades del dispositivo como la cámara, el GPS o los sensores.
- Una mejor experiencia: al seguir las pautas de diseño propias del sistema operativo, el código nativo suele ofrecer una interfaz más coherente y familiar para los usuarios.

Sin embargo, el desarrollo nativo también presenta algunos inconvenientes como:

- Un mayor costo: al tener que desarrollar una aplicación diferente para cada plataforma, el código nativo suele requerir más tiempo, recursos y herramientas que el código híbrido o multiplataforma.
- Una mayor complejidad: al tener que manejar diferentes lenguajes, frameworks y librerías para cada plataforma, el código nativo suele ser más difícil de aprender y usar que el código híbrido o multiplataforma.
- Una menor portabilidad: al estar atado a un sistema operativo específico, el código nativo suele ser más difícil de adaptar o migrar a otras plataformas que el código híbrido o multiplataforma.

Un ejemplo de stack nativo para Android e iOS sería:

- Front-end:
    - Lenguaje: Java o Kotlin para Android; Objective-C o Swift para iOS
    - Framework: Android SDK o Jetpack Compose para Android; iOS SDK o SwiftUI para iOS
    - Librería: Material Design Components para Android; UIKit o SwiftUI para iOS
- Back-end:
    - Lenguaje: PHP, Java, Python, Node.js, Ruby, etc.
    - Framework: Laravel, Spring Boot, Django, Express, Rails, etc.
    - Base de datos: MySQL, PostgreSQL, MongoDB, Firebase, etc.
    - API: RESTful, GraphQL, etc.
- Development:
    - IDE: Android Studio para Android; Xcode para iOS
    - Editor: Visual Studio Code, Sublime Text, Atom, etc.
    - Compilador: Gradle o Maven para Android; Xcode Build System o CocoaPods para iOS
    - Depurador: Logcat o Firebase Crashlytics para Android; Xcode Debugger o Firebase Crashlytics para iOS
    - Emulador: Android Emulator o Genymotion para Android; iOS Simulator o TestFlight para iOS
- Supporting:
    - Servicio en la nube: AWS, Google Cloud Platform, Microsoft Azure, etc.

    - Sistema de control de versiones: Git, GitHub, Bitbucket, etc.
    - Sistema de gestión de dependencias: Gradle o Maven para Android; CocoaPods, Carthage o Swift Package Manager para iOS
    - Sistema de análisis y monitorización: Google Analytics, Firebase Analytics, Mixpanel, etc.
    - Sistema de distribución y actualización: Google Play Store o AppGallery para Android; App Store Connect o TestFlight paraiOS

Stack multiplataforma

para Android e iOS

El desarrollo multiplataforma consiste en crear una aplicación única que pueda funcionar en varios sistemas operativos móviles usando un framework que permita compilar el código fuente en diferentes formatos binarios. Esto implica usar un lenguaje común (como JavaScript) así como un framework específico (como React Native o Flutter) que permita generar una interfaz nativa para cada plataforma. El desarrollo multiplataforma ofrece algunas ventajas como:

- Un menor costo: al tener que desarrollar una sola aplicación para varias plataformas, el código multiplataforma suele requerir menos tiempo, recursos y herramientas que el código nativo.
- Una menor complejidad: al tener que manejar un solo lenguaje, framework y librería para varias plataformas, el código multiplataforma suele ser más fácil de aprender y usar que el código nativo.
- Una mayor portabilidad: al estar basado en un framework común, el código multiplataforma suele ser más fácil de adaptar o migrar a otras plataformas que el código nativo.

Sin embargo, el desarrollo multiplataforma también presenta algunos inconvenientes como:

- Un peor rendimiento: al estar basado en una capa de abstracción, el código multiplataforma suele ser más lento y menos fluido que el código nativo.
- Una peor integración: al usar un framework genérico, el código multiplataforma suele acceder peor a las funcionalidades del dispositivo como la cámara, el GPS o los sensores.
- Una peor experiencia: al seguir unas pautas de diseño genéricas, el código multiplataforma suele ofrecer una interfaz menos coherente y familiar para los usuarios.

Un ejemplo de stack multiplataforma para Android e iOS sería:

- Front-end:
    - Lenguaje: JavaScript
    - Framework: React Native o Flutter

    - Librería: React o Dart
- Back-end:
    - Lenguaje: PHP, Java, Python, Node.js, Ruby, etc.
    - Framework: Laravel, Spring Boot, Django, Express, Rails, etc.
    - Base de datos: MySQL, PostgreSQL, MongoDB, Firebase, etc.
    - API: RESTful, GraphQL, etc.
- Development:
    - IDE: Visual Studio Code
    - Editor: Visual Studio Code, Sublime Text, Atom, etc.
    - Compilador: React Native CLI o Flutter CLI
    - Depurador: React Native Debugger o Flutter DevTools
    - Emulador: Android Emulator o iOS Simulator
- Supporting:
    - Servicio en la nube: AWS, Google Cloud Platform, Microsoft Azure, etc.
    - Sistema de control de versiones: Git, GitHub, Bitbucket, etc.
    - Sistema de gestión de dependencias: npm o yarn
    - Sistema de análisis y monitorización: Google Analytics, Firebase Analytics, Mixpanel, etc.
    - Sistema de distribución y actualización: Google Play Store o AppGallery para Android; App Store Connect o TestFlight para iOS

Comparación entre stack nativo y stack multiplataforma

Para comparar entre el stack nativo y el stack multiplataforma, se pueden usar los siguientes criterios:

- Rendimiento: el stack nativo suele ofrecer un mejor rendimiento que el stack multiplataforma, ya que está optimizado para cada plataforma y no depende de una capa de abstracción. Sin embargo, el stack multiplataforma puede ofrecer un rendimiento aceptable si se usa un framework adecuado y se optimiza el código correctamente.
- Integración: el stack nativo suele ofrecer una mejor integración que el stack multiplataforma, ya que usa las librerías propias del sistema operativo y puede acceder mejor a las funcionalidades del dispositivo. Sin embargo, el stack multiplataforma puede ofrecer una integración suficiente si se usan plugins o módulos específicos para cada plataforma.
- Experiencia: el stack nativo suele ofrecer una mejor experiencia que el stack multiplataforma, ya que sigue las pautas de diseño propias del sistema operativo y ofrece una interfaz más coherente y familiar para los usuarios. Sin embargo, el stack multiplataforma puede ofrecer una experiencia

satisfactoria si se usa un framework que genere una interfaz nativa para cada plataforma o si se personaliza la interfaz según las preferencias del usuario.

- Costo: el stack multiplataforma suele ofrecer un menor costo que el stack nativo, ya que requiere menos tiempo, recursos y herramientas para desarrollar una sola aplicación para varias plataformas. Sin embargo, el stack nativo puede ofrecer un costo razonable si se aprovechan las ventajas del desarrollo ágil y las herramientas de automatización.
- Complejidad: el stack multiplataforma suele ofrecer una menor complejidad que el stack nativo, ya que requiere manejar un solo lenguaje, framework y librería para varias plataformas. Sin embargo, el stack nativo puede ofrecer una complejidad manejable si se usa un lenguaje moderno y sencillo como Kotlin o Swift y se siguen las buenas prácticas de programación.
- Portabilidad: el stack multiplataforma suele ofrecer una mayor portabilidad que el stack nativo, ya que permite adaptar o migrar la aplicación a otras plataformas con facilidad. Sin embargo, el stack nativo puede ofrecer una portabilidad aceptable si se usa un framework que permite reutilizar parte del código entre plataformas como Jetpack Compose o Swift.

Front-end y Back-end:

Lenguaje	**Framework**	**Explicación**	**Curva de aprendizaje**	**Ventajas**	**Desventajas**	**Escalabilidad**
JavaScript	Flutter	Es un framework de Google que permite crear aplicaciones nativas para Android e iOS usando el lenguaje Dart y un motor de renderizado propio.	Media-Alta	Permite crear interfaces nativas con un solo código, ofrece un alto rendimiento y una gran variedad de widgets y plugins.	Requiere aprender un nuevo lenguaje (Dart), puede tener problemas de compatibilidad con algunas librerías nativas y puede consumir más recursos del dispositivo.	Alta
JavaScript	Node.js	Es un entorno de ejecución que permite usar JavaScript en el lado del servidor, usando el motor V8 de Google Chrome.	Baja-Media	Permite usar el mismo lenguaje en el front-end y el back-end, ofrece una alta velocidad y escalabilidad gracias al modelo de eventos y la programación asíncrona y tiene	Puede tener problemas de seguridad y estabilidad si no se usa correctamente, puede ser difícil de depurar y testear y puede ser menos eficiente para operaciones	Alta

				una gran comunidad y ecosistema.	intensivas en CPU.	
SQL	MySQL	Es un sistema de gestión de bases de datos relacionales, que permite almacenar, consultar y manipular datos usando el lenguaje SQL.	Baja-Media	Es uno de los sistemas más populares y usados en la web, ofrece una alta fiabilidad, seguridad y compatibilidad y tiene una gran documentación y soporte.	Puede tener problemas de rendimiento y escalabilidad si no se diseña correctamente el esquema de la base de datos, puede ser menos flexible que otros sistemas no relacionales y puede tener limitaciones en cuanto a las funciones y los tipos de datos.	Media-Alta

SUPPORTING

	Supporting	**Explicación**	**Curva de aprendizaje**	**Ventajas**	**Desventajas**	**Escalabilidad**
Servicio en la nube	AWS	Es un proveedor de servicios en la nube que ofrece soluciones para alojar, gestionar y escalar aplicaciones web y móviles usando diferentes recursos como servidores, bases de datos, almacenamiento, etc.	Media-Alta	Ofrece una alta disponibilidad, seguridad y escalabilidad para las aplicaciones web y móviles, ofrece una gran variedad de servicios y opciones para adaptarse a las necesidades de cada	Puede tener un alto costo si se usan muchos recursos o servicios, puede tener una mayor complejidad y dificultad para configurar y gestionar los recursos y servicios y puede tener algunos problemas de latencia o disponibilidad según la región o el proveedor.	Alta

				proyecto y tiene una gran documentación y soporte.		
Sistema de control de versiones	Git/GitHub	Son sistemas de control de versiones que permiten gestionar los cambios en el código fuente de forma colaborativa y eficiente.	Baja-Media	Permiten mantener un historial de los cambios en el código fuente, ofrecen una gran facilidad para crear y fusionar ramas y tienen una gran comunidad y soporte.	Pueden tener algunos problemas de seguridad si no se usan correctamente, pueden requerir una mayor configuración e instalación de dependencias y pueden ser menos intuitivos o amigables para algunos usuarios.	N/A
Sistema de gestión de dependencias	npm/yarn	Son sistemas de gestión de dependencias que permiten instalar y actualizar las librerías y los módulos necesarios para el desarrollo de aplicaciones web y móviles.	Baja-Media	Permiten automatizar y simplificar la instalación y la actualización de las dependencias, ofrecen una gran variedad de librerías y módulos disponibles y tienen una gran documentación y soporte.	Pueden tener algunos problemas de seguridad si no se usan correctamente, pueden requerir una mayor configuración e instalación de dependencias y pueden generar conflictos o incompatibilidades entre las dependencias.	N/A
Sistema de distribución y actualización	Google Play Store y	Son sistemas de distribución y actualización que	Media-Alta	Permiten llegar a un amplio	Pueden tener algunos requisitos o restricciones para	N/A

	App Store.	permiten publicar y actualizar las aplicaciones móviles en las tiendas oficiales o alternativas de cada plataforma.		mercado potencial de usuarios, ofrecen una gran facilidad para publicar y actualizar las aplicaciones y tienen una gran documentación y soporte.	publicar o actualizar las aplicaciones,	

Recomendación

Teniendo en cuenta los supuestos necesarios y las recomendaciones dadas para elegir el stack para el desarrollo de aplicaciones móviles para Android e iOS, se puede recomendar lo siguiente:

- Si se busca crear una aplicación con un alto rendimiento, una buena integración y una excelente experiencia para los usuarios finales, se recomienda usar un stack nativo con los siguientes elementos:
    - Front-end:
        - Lenguaje: Kotlin para Android; Swift para iOS
        - Framework: Jetpack Compose para Android; SwiftUI para iOS
        - Librería: Material Design Components para Android; UIKit o SwiftUI para iOS
    - Back-end:
        - Lenguaje: Node.js
        - Framework: Express
        - Base de datos: MongoDB
        - API: GraphQL
    - Development:
        - IDE: Android Studio para Android; Xcode para iOS
        - Editor: Visual Studio Code
        - Compilador: Gradle o Maven para Android; Xcode Build System o CocoaPods para iOS
        - Depurador: Logcat o Firebase Crashlytics para Android; Xcode Debugger o Firebase Crashlytics para iOS

        - Emulador: Android Emulator o Genymotion para Android; iOS Simulator o TestFlight paraiOS
    - Supporting: Servicio en la nube: AWS Sistema de control de versiones: Git Sistema de gestión de dependencias: Gradle o Maven paraiOS Sistema de análisis y monitorización: Firebase Analytics Sistema de distribución y actualización: Google Play Store paraiOS
- Si se busca crear una aplicación con un bajo costo, una baja complejidad y una alta portabilidad para varios sistemas operativos móviles, se recomienda usar un stack multiplataforma con los siguientes elementos:
    - Front-end: Lenguaje: JavaScript Framework: React Native Librería: React Back-end: Lenguaje: Node.js Framework: Express Base de datos: MongoDB API: GraphQL Development: IDE: Visual Studio Code Editor: Visual Studio Code Compilador: React Native CLI Depurador: React Native Debugger Emulador: Android Emulator paraiOS Simulator Supporting: Servicio en la nube: Google Cloud Platform Sistema de control de versiones: Git Sistema de gestión de dependencias: npm Sistema de análisis y monitorización: Google Analytics Sistema de distribución y actualización: Google Play Store or AppGallery for Android; App Store Connect or TestFlight for iOS.

Referencias consultadas

1 Uptech Team. (2021). A Definitive Overview on Mobile App Technology Stack. Retrieved from https://www.uptech.team/blog/mobile-app-technology-stack

2 Code&Care. (2022). The Best Tech Stack for Mobile App Development for 2022. Retrieved from https://code-care.com/blog/technology-stack-for-app-development/

3 React Native. (n.d.). React Native · Learn once, write anywhere. Retrieved from https://reactnative.dev/

[4] Flutter. (n.d.). Flutter | Beautiful native apps in record time. Retrieved from https://flutter.dev/

[5] Android Developers. (n.d.). Android Mobile App Developer Tools – Android Developers. Retrieved from https://developer.android.com/

[6] Apple Developer. (n.d.). Xcode - Apple Developer. Retrieved from https://developer.apple.com/xcode/

[7] Jetpack Compose. (n.d.). Jetpack Compose | Android Developers. Retrieved from https://developer.android.com/jetpack/compose

[8] SwiftUI. (n.d.). SwiftUI - Apple Developer. Retrieved from https://developer.apple.com/xcode/swiftui/

[9] Documento Investigado:

https://docs.google.com/document/d/1dafftDjRkOURggYOb6XKEL0YdaO5IIgpLJnt4cpiGYM/edit?usp=sharing

Conclusión

En este documento se ha presentado una introducción al stack para el desarrollo de aplicaciones móviles para Android e iOS, así como una comparación y una recomendación entre el stack nativo y el stack multiplataforma. Se ha visto que el stack nativo ofrece un mejor rendimiento, una mejor integración y una mejor experiencia que el stack multiplataforma, pero también implica un mayor costo, una mayor complejidad y una menor portabilidad. Por otro lado, se ha visto que el stack multiplataforma ofrece un menor costo, una menor complejidad y una mayor portabilidad que el stack nativo, pero también implica un peor rendimiento, una peor integración y una peor experiencia. Por lo tanto, se debe elegir el stack más adecuado según los requisitos y las características de cada proyecto.