ASCE MAGAZINE ISSN: 3073–1178

Revista ASCE Magazine, Periodicidad: Trimestral Abril - Junio, Volumen: 4, Número: 2, Año: 2025 páginas 1301- 1316

Doi: https://doi.org/10.70577/ASCE/1301.1316/2025

Recibido: 2025-04-15
Aceptado: 2025-05-15
Publicado: 2025-06-26

Análisis comparativo de la productividad en el desarrollo móvil: Flutter vs Kotlin vs Swift

Mobile Development Productivity Benchmark: Flutter vs. Kotlin vs. Swift

Autores:

Diego Geovanny Falconí Punguil
https://orcid.org/0009-0000-2398-8849
diego.falconi4@utc.edu.ec
Universidad Técnica de Cotopaxi
Latacunga-Ecuador

David Mauricio Cordonez Arias
https://orcid.org/0009-0004-5714-1290
davidsuax@gmail.com
Universidad Técnica de Cotopaxi
Latacunga-Ecuador

Edwin Fabricio Damacela Calero
https://orcid.org/0009-0002-7126-5316
faby98dc@gmail.com
Universidad Técnica de Cotopaxi
Latacunga-Ecuador

Edison Isaias Pallo Cuchiparte
https://orcid.org/0009-0005-2992-5693
isaiaspallo11@gmail.com
Universidad Técnica de Cotopaxi
Latacunga-Ecuador

Cómo citar

Falconí Punguil, D. G., Cordonez Arias, D. M., Damacela Calero, E. F., & Pallo Cuchiparte, E. I. (2025). Análisis comparativo de la productividad en el desarrollo móvil: Flutter vs Kotlin vs Swift . *ASCE*, *4*(2), 1301–1316.

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

Resumen

El desarrollo de aplicaciones móviles se basaba en una sola plataforma, y los desarrolladores debían generar las aplicaciones para iOS y Android por separado. Sin embargo, la productividad de desarrollo móvil multiplataforma, goza de un amplio reconocimiento en el mercado. El objetivo de la investigación fue; realizar un análisis comparativo de la productividad en el desarrollo móvil: Flutter vs Kotlin vs Swift. La investigación se llevó a cabo mediante; diseño no experimental, alcance descriptivo, método analítico, y enfoque cualitativo; se realizó una evaluación comparativa, de las ventajas de productividad de cada uno de los desarrolladores. Se logró conocer que, la elección de la tecnología usada para el desarrollo de aplicaciones móviles impacta directamente en la productividad del equipo ya sea Android o iOS; Flutter (desarrollador de aplicaciones híbridas) se posiciona como el líder en productividad, para proyectos multiplataforma, por lo que es idóneo cuando se busca maximizar la eficiencia y reducir costos; Kotlin y Swift (desarrolladores de aplicaciones nativas) son la opción productiva específicamente para el desarrollo de aplicaciones nativas, es decir sirve para Android o iOS pero no para los dos en conjunto, lo que les permite aprovechar al máximo todas las funcionalidades del sistema operativo seleccionado.

Palabras clave: Aplicaciones Móviles; Comparaciones Multiplataforma; Desarrolladores Móviles; Flutter; Kotlin; Sistemas Operativos; Swift

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

Abstract

Mobile app development used to be based on a single platform, and developers had to create apps for iOS and Android separately. However, cross-platform mobile development productivity is widely recognized in the market. The objective of the research was to conduct a comparative analysis of mobile development productivity: Flutter vs. Kotlin vs. Swift. The research was conducted using a non-experimental design, descriptive scope, analytical method, and qualitative approach. A comparative evaluation was conducted of the productivity advantages of each developer. It was learned that the choice of technology used for mobile app development directly impacts team productivity, whether Android or iOS. Flutter (hybrid app developer) is positioned as the leader in productivity for cross-platform projects, making it ideal when seeking to maximize efficiency and reduce costs. Kotlin and Swift (native app developers) are the productive choice specifically for native app development. That is, they work for Android or iOS, but not both together, allowing you to take full advantage of all the features of the selected operating system.

Keywords: Mobile Apps; Cross-Platform Comparisons; Mobile Developers; Flutter; Kotlin; Operating Systems; Swift

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

Introducción

La disponibilidad de diferentes dispositivos móviles y sistemas operativos aumenta la demanda de soluciones intersectoriales; crear e identificar, una aplicación nativa principal diferente para cada plataforma requiere mucho tiempo y no es recomendable (Riaz, 2025).

Pues las aplicaciones móviles son un segmento importante de las soluciones tecnológicas existentes debido a la popularidad de dispositivos como smartphones y tablets. A medida que el uso tradicional de las aplicaciones se desplaza de los navegadores a las aplicaciones móviles en todos los ámbitos, se demandan herramientas de desarrollo de aplicaciones inteligentes, eficientes y multiplataforma, generando la necesidad de identificar el desarrollador más idóneo para los dispositivos (Melovic et al., 2025).

Anteriormente, el desarrollo de aplicaciones móviles se basaba en una sola plataforma, y los desarrolladores debían generar las aplicaciones para iOS y Android por separado, sin embargo, esto ha supuesto un gran reto, ya que la empresa debe contratar a un desarrollador para desarrollar aplicaciones para cada plataforma, lo que eleva el tiempo de desarrollo, los costes y los gastos de mantenimiento (Dos Santos, 2024).

En este caso, la transición hacia el desarrollo multiplataforma elimina los problemas generados por el desarrollo de aplicaciones nativas, ofreciendo a los usuarios soluciones móviles más fáciles de implementar, robustas y eficientes en diversas plataformas (Melovic et al., 2025).

Según Riaz (2025), la productividad de desarrollo móvil multiplataforma, goza de un amplio reconocimiento en el mercado, dentro de los frameword (conjunto de herramientas y componentes que generan una base para desarrollar aplicaciones o sofyware) más destacados en este ámbito están; Kotlin Multiplatform, Flutter y Swift. Cabe considerar que todos presentan fortalezas, debilidades y opciones para la selección de herramientas en diversos tipos de iniciativas móviles. Por ejemplo;

Kotlin Multiplataforma, es una herramienta relativamente nueva lanzada previamente para respaldar el desarrollo de aplicaciones multiplataforma y, al mismo tiempo, conservar la capacidad de Android, este busca la reutilización y el alto rendimiento, lo que se logra compartiendo el código

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

de la capa de lógica de negocio, a la vez que se satisface la necesidad de desarrollo de interfaz del usuario en cada plataforma (Espitia-Acero, 2020).

Flutter, fue lanzado por Google es un increíble conjunto de herramientas de interfaz de usuario para crear aplicaciones multiplataforma utilizando una única base de código, esto hace que su arquitectura basada en widgets sea muy consistente ,y mucho más fluida en dispositivos Android, iOS, web e incluso de escritorio (Stanic y Ćirković, 2024).

Swift, fue lanzado por Apple en 2015, es un lenguaje de propósito general compatible con la programación de sistemas, servicios en la nube, así como aplicaciones de escritorio y móviles, incluye funciones orientadas a objetos como; clases y protocolos, pero también incluye patrones de programación funcional, como mapa y filtro; está diseñado para ser un lenguaje seguro, con características como el lenguaje estricto (Biørn-Hansen et al., 2020).

Frente a lo expuesto anteriormente, es importante reconocer que, el desarrollo de aplicaciones móviles es un campo en rápida evolución, por ella, las demandas y expectativas de aplicaciones de alta calidad con una experiencia de usuario intuitiva y un rendimiento productivo rápido impulsan el desarrollo tecnológico (Singh y Shobha, 2021).

El objetivo de la investigación fue; realizar un análisis comparativo de la productividad en el desarrollo móvil: Flutter vs Kotlin vs Swift, con la finalidad de ayudar a los desarrolladores y organizaciones a tomar decisiones informadas sobre las ventajas de productividad, basándose en su eficiencia, y rendimiento a nivel de soporte de la comunidad.

Material y métodos

Características del estudio

La investigación se llevó a cabo mediante un diseño no experimental pues no existió manipulación de las variables de estudio; el alcance del estudio fue descriptivo para realizar la descripción de las características y comparación de los desarrolladores móviles (Futter – Híbrido multiplataforma, Kotlin – Nativo Android y Swift – Nativo iOS); a la vez, se trabajó con el método analítico el

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

mismo que permite analizar de forma minuciosa los hechos y propiedades de manera particular; el enfoque del estudio fue cualitativo, pues la información recopilada fue producto de la evaluación de cualidades y virtudes de los desarrolladores móviles (frameword) mencionados.

Método

Se realizó una evaluación comparativa, la cual se refiere al proceso de identificar y caracterizar las ventajas de productividad de cada uno de los desarrolladores, a la vez se realizó una breve definición de estos.

En cada uno de los frameword, se registraron y analizaron métricas como:

- Tiempo de desarrollo (horas trabajadas)
- Cantidad de líneas de código
- Rendimiento de la app (uso de CPU y RAM, velocidad de carga, etc.)
- Facilidad de implementación de interfaces de usuario (UI)
- Cantidad de errores o bugs detectados durante las pruebas

Dichos datos fueron registrados de forma manual con herramientas tales como Toggl (para medir el tiempo), Android Studio/Visual Studio Code (para análisis del rendimiento) y Firebase Performance para algunas métricas de prueba en dispositivos móviles.

Con la información obtenida, se construyó dashboards en Excel para visualizar de mejor forma los resultados y proceder a comparar los frameworks. Finalmente, la información recopilada fue expuesta de forma precisa, clara y efectiva desde el punto de vista del usuario, a través de realizar una comparación de elementos de forma sencilla.

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

Resultados

Descripción de cada uno de los desarrolladores evaluados en la investigación:

1. *Flutter*

Flutter es un Interfaz de Usuario (UI) toolkit de código abierto desarrollado por Google para construir aplicaciones compiladas de forma nativa para móvil, web y escritorio desde una única base de código. Utiliza el lenguaje de programación Dart (Figura 1) y es considerado un desarrollador móvil de aplicaciones híbridas.

Ventajas en Productividad:

- Hot Reload & Hot Restart: Característica estrella que permite a los desarrolladores ver los cambios en la interfaz de usuario y la lógica de negocio al instante, sin perder el estado de la aplicación. Esto acelera drásticamente el ciclo de desarrollo y depuración.
- Código Único para Múltiples Plataformas: La mayor ventaja en productividad, escribir código una sola vez y desplegarlo en Android e iOS (y más) reduce el tiempo de desarrollo a la mitad, o incluso más, en comparación con el desarrollo nativo dual.
- Widgets Preconstruidos y Personalizables: Flutter ofrece un rico conjunto de widgets Material Design y Cupertino que son altamente personalizables y fáciles de usar, acelerando la construcción de UI complejas.
- Excelente Documentación y Comunidad: Una comunidad en rápido crecimiento y documentación exhaustiva facilitan la resolución de problemas.
- Rendimiento Cercano al Nativo: Al compilar directamente a código nativo, Flutter ofrece un rendimiento muy cercano al de las aplicaciones nativas, reduciendo la necesidad de optimizaciones complejas post-desarrollo.

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

Figura 1

Frameword: Flutter herramienta de desarrollo multiplataforma (SOFCAVI, 2023).

Cabe recalcar, que Futter es una herramienta de desarrollo multiplataforma, que se clasifica dentro de los desarrolladores de aplicaciones hibridas, estos suelen desarrollarse en HTML5, CSS y un framework basado en JavaScript, y principalmente incluyen componentes web que conforman la interfaz de usuario (UI) de la aplicación, que a su vez utilizan complementos o bibliotecas para interactuar con funciones específicas del sistema operativo, como la cámara, la geolocalización y otros sensores de hardware.

Sin embargo, la aplicación híbrida se centra en el uso de las API (Interfaz de Programación de Aplicaciones) del sistema operativo y no en el código o la arquitectura subyacentes, por lo que puede abstraerse para utilizar dichas bibliotecas.

La figura 2 es una arquitectura para el Frameword Ionic, que es un marco de aplicación híbrido ampliamente utilizado en la plataforma móvil, para desarrollar aplicaciones para la web, Android e iOS simultáneamente utilizando una única base de código. En este caso, la lógica de la aplicación está escrita como una aplicación web con HTML, CSS y Angular, un framework basado en JavaScript para la creación de aplicaciones web. El framework utiliza Cordova y Capacitor para interactuar con las API nativas y usar las funciones nativas de la aplicación, como la brújula, la cámara, la geolocalización y las notificaciones push, entre otras; la aplicación utiliza las API del sistema operativo para interactuar con este hardware y las funciones nativas.

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

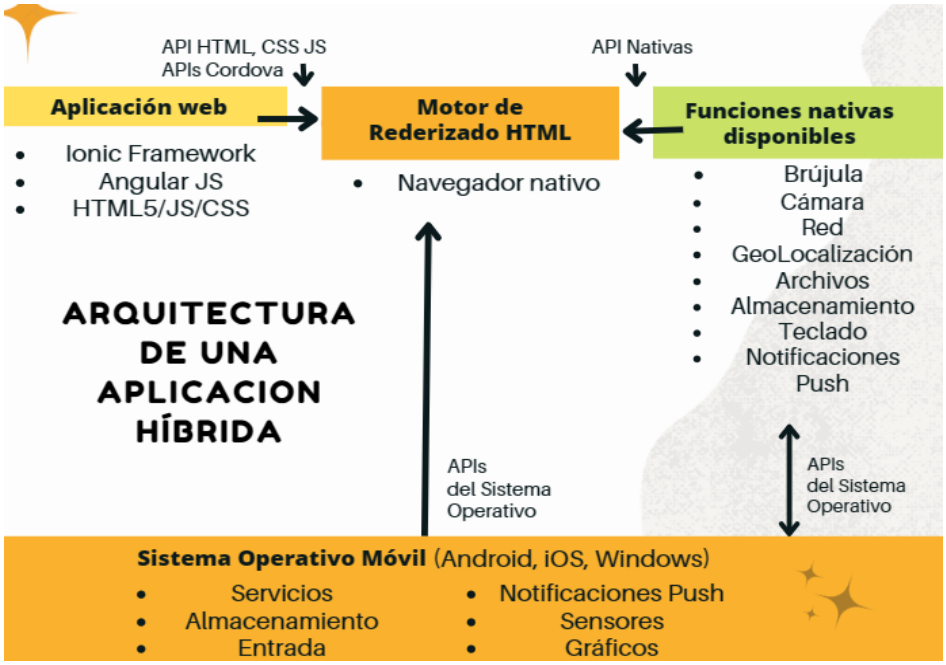

Figura 2

Arquitectura de una aplicación hibrida (Singh y Shobha, 2021).

2. *Kotlin*

Kotlin es un lenguaje de programación de tipo estático, desarrollado por JetBrains, totalmente interoperable con Java y diseñado para JVM. Es el lenguaje preferido por Google para el desarrollo de aplicaciones Android (Figura 3), es un desarrollador de aplicaciones específicamente nativas.

Ventajas en Productividad:

- Concisión y Seguridad de Código: Kotlin es un lenguaje moderno que requiere menos código que Java para lograr las mismas funcionalidades, lo que reduce el tiempo de escritura y la probabilidad de errores (ejemplo: nulidad segura).
- Excelente Soporte IDE (Android Studio): Android Studio ofrece un soporte inigualable para Kotlin, con autocompletado inteligente, refactorización avanzada y potentes herramientas de depuración.
- Gran Ecosistema y Librerías: Acceso completo al vasto ecosistema de Android y bibliotecas Java, lo que significa que rara vez hay que "reinventar la rueda".

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

- Rendimiento Óptimo: Las aplicaciones Kotlin se compilan directamente a bytecode de JVM y aprovechan al máximo las optimizaciones de la plataforma Android, garantizando un rendimiento nativo.
- Control Fino: Permite un control muy granular sobre cada aspecto de la aplicación Android.

Figura 3

Frameword: Kotlin (Desarrollo Nativo Android) herramienta de desarrollo multiplataforma (GENBETA, 2017).

3. *Swift*

Swift es un lenguaje de programación multiparadigma (Figura 4) compilado y de tipo estático desarrollado por Apple. Es el lenguaje principal para el desarrollo de aplicaciones para iOS, macOS, watchOS y tvOS (Figura 5), es un desarrollador de aplicaciones de origen nativo.

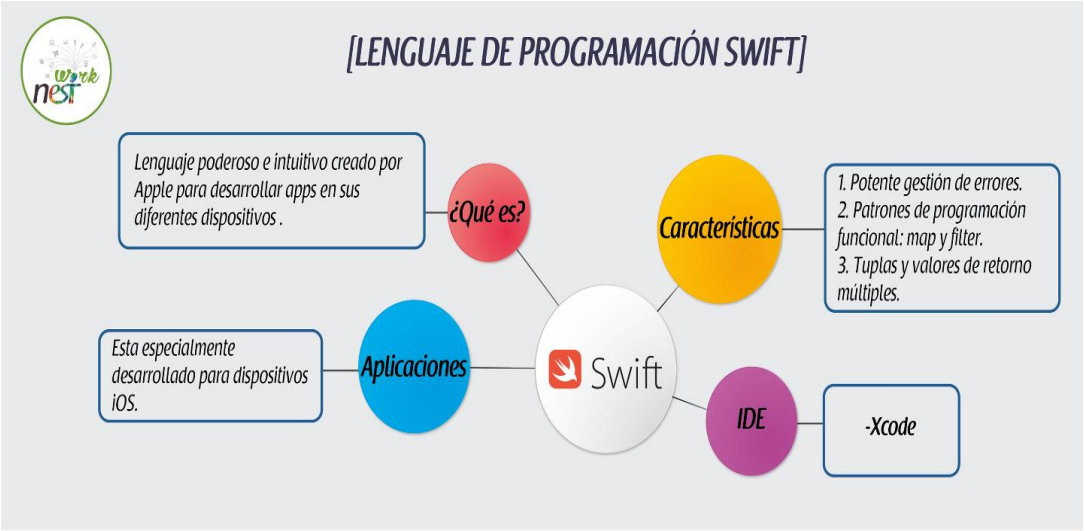

Figura 4

Características del lenguaje de programación SWIFT (Work Nest Oficial, 2020).

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

Ventajas en Productividad:

- Diseño de Lenguaje Moderno y Seguro: Swift es un lenguaje intuitivo, seguro y con sintaxis concisa, lo que acelera la escritura de código y reduce errores (ejemplo: manejo de opcionales).
- Excelente Soporte IDE (Xcode): Xcode, el IDE de Apple, ofrece un entorno de desarrollo integrado con potentes herramientas de depuración, un constructor de interfaces (Interface Builder) y simuladores eficientes.
- Frameworks Potentes de Apple: Acceso directo a todos los frameworks nativos de iOS, como UIKit y el más reciente SwiftUI, que facilitan la construcción de interfackes de usuario y la integración con las características del sistema operativo.
- Rendimiento Nativo y Optimización: Las aplicaciones Swift se compilan directamente a código máquina, ofreciendo el máximo rendimiento y una experiencia de usuario fluida.
- Acceso a Últimas Características de iOS: Los desarrolladores de Swift son los primeros en acceder a las nuevas características y APIs de iOS tan pronto como se lanzan.

Figura 5

Frameword: Swift (Desarrollo Nativo iOS) herramienta de desarrollo multiplataforma (Work Nest Oficial, 2020).

Discusiones

Las principales plataformas en las que funcionan los dispositivos son Android (propiedad de Google), iOS (propiedad de Apple) y algunos son Windows, como se ha expuesto con ayuda de la información sobre las ventajas de la productividad de los desarrolladores móviles, se conoció que,

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

el desarrollo de aplicaciones para estas plataformas se convierte en un desafío al considerar la creación para todas las plataformas existentes, pues cada una posee, su propio conjunto de herramientas y bibliotecas, e incluso diferentes lenguajes de programación.

Por ello, si el desarrollador desea lanzar una aplicación en múltiples plataformas específicas de cada proveedor, como iOS y Android (basado en Google), debe aprender muchos aspectos, como los lenguajes de programación, las herramientas y las bibliotecas de cada una. Según Ahmad (2023), esto genera mucha confusión entre los desarrolladores que se inician en la creación de aplicaciones, considerando la popularidad de Android e iOS, que al parecer son mercados viables y excelentes para los desarrolladores.

Por lo tanto, los desarrolladores pueden optar por el desarrollo de aplicaciones nativas (Kotlin y Swift), que son específicas para un sistema operativo en particular (Zarichuk, 2023). Otra opción es el desarrollo de aplicaciones híbridas, que parece una opción lucrativa, donde habrá una opción de utilizar una única base de código para múltiples plataformas (Futter), conocido como un desarrollador de aplicaciones híbridas debido a su desarrollo multiplataforma, siendo la principal ventaja, la compatibilidad con Android e iOS (Suri et al., 2022).

Para una mejor compresión en la presente investigación se realiza una comparación entre las aplicaciones nativas (Kotlin y Swift) e híbridas (Futter), analizando la productividad basándose en ventajas y desventajas (Tabla 1).

Tabla 1

Comparación de las aplicaciones nativas (Kotlin y Swift) e híbridas (Futter) evaluadas en la investigación.

Nativas		Hibridas
Ventajas	Desventajas	Ventajas
Las aplicaciones nativas se ejecutan más rápido porque están escritas en el lenguaje nativo y	La falta de flexibilidad de las plataformas es un problema importante en las aplicaciones nativas. Los desarrolladores	Las aplicaciones híbridas pueden ofrecer componentes de interfaz de usuario enriquecidos que

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

pueden utilizar el hardware del dispositivo de manera eficiente	tienden a limitarse a una sola plataforma, ya que la curva de aprendizaje asociada es muy alta para cada una.	se han tomado de sus contrapartes web y aprovechan elementos nativos e híbridos.
A diferencia de las aplicaciones web, que generalmente requieren conexión a la red para funcionar, las aplicaciones nativas pueden funcionar sin conexión a internet y reanudar la conexión cuando se restablece la red. Esto se logra mediante el almacenamiento en caché y la compatibilidad con bases de datos locales para aplicaciones nativas.	El desarrollo de aplicaciones nativas es un proceso costoso si una empresa o negocio desea lanzar su aplicación para múltiples plataformas. Necesitan equipos de desarrollo separados para las distintas plataformas a las que se dirigen. Los costos de mantenimiento también son elevados para las aplicaciones nativas.	Gracias a una única base de código que puede ser utilizada por múltiples plataformas, ofrecen un alcance de audiencia más amplio. El coste de desarrollo se reduce ampliamente ya que se reduce el tiempo de desarrollo y la base de código.
Los elementos y el diseño de la aplicación están estrechamente vinculados con la apariencia del sistema operativo para el que están diseñados. Esto se debe a que la mayoría de estos elementos han sido diseñados por el propietario del sistema operativo.	Como cada plataforma requerirá diferentes bases de código, el tiempo necesario para desarrollar y llevar el producto al mercado puede ser mucho mayor en comparación con el modelo de aplicación híbrida.	El tiempo de desarrollo se puede reducir considerablemente ya que solo se debe desarrollar una base de código para múltiples plataformas.

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

Las aplicaciones nativas admiten una mayor cantidad de sensores de hardware, como GPS, Bluetooth y cámara, y ofrecen un mejor control para el uso de estas funciones.	Las correcciones de errores y las actualizaciones deben realizarse a través de las diferentes tiendas de aplicaciones, lo que puede llevar tiempo y es un proceso lento y costoso de mantener.	El mantenimiento del código es más fácil en comparación con las aplicaciones nativas.

Fuente: Singh y Shobha (2021).

Conclusiones

- Mediante la información descrita anteriormente se logró conocer que, la elección de la tecnología usada para el desarrollo de aplicaciones móviles impacta directamente en la productividad del equipo ya sea Android o iOS.
- Flutter (desarrollador de aplicaciones híbridas) se posiciona como el líder en productividad, para proyectos multiplataforma donde el objetivo principal es un lanzamiento rápido al mercado, mediante una base de código única.

  Su Hot Reload/Restart y la reutilización de código son inigualables para acelerar el ciclo de desarrollo. Es ideal para startups y empresas que buscan maximizar la eficiencia y reducir costos en un entorno multiplataforma.

- Kotlin y Swift (desarrolladores de aplicaciones nativas) son la opción más productiva para el desarrollo nativo puro. Si el objetivo es construir una aplicación de alto rendimiento, altamente optimizada para una plataforma específica (Android o iOS respectivamente), aprovechando al máximo todas las funcionalidades del sistema operativo, estos lenguajes ofrecen la mayor productividad dentro de su propio ecosistema.

  Son ideales para aplicaciones que requieren un control muy granular, integraciones profundas con el hardware o APIs específicas, o cuando solo se apunta a una plataforma.

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

Referencias bibliográficas

Ahmad, M. (2023). Analysis of cross platform mobile application development frameworks. *The rapid growth of the mobile application industry has necessitated the introduction*.

Biørn-Hansen, A., Rieger, C., Grønli, T., Majchrzak, T., y Ghinea, G. (2020). An empirical investigation of performance overhead in cross-platform mobile development frameworks. *Empirical Software Engineering*, *25*(1), 2997-3040.

Dos Santos, B. (2024). *Partilha de Dados Durante Videoconferência em App Móvel*. Universidade NOVA de Lisboa (Portugal).

Espitia-Acero, J. S. (2020). *Empirical testing for establishing benchmarks: process review and comparison between java, kotlin and dart's performance.* Uniandes.

GENBETA (2017). *Kotlin 1.1 también es para desarrolladores Android*. https://bit.ly/4lhioSd

Melovic, S., Stefanovic, D., y Dakic, D. (2025). Design and Implementation of an Electronic Voting System Using Blockchain Technology. In *2025 24th International Symposium INFOTEH-JAHORINA (INFOTEH), 2*(1), 1-6.

Riaz, M. (2025). *Comparative Analysis of React Native, Kotlin, and Flutter for Cross-Platform Mobile Development.* https://bit.ly/3I1GAcD

Singh, M., y Shobha, G. (2021). Comparative analysis of hybrid mobile app development frameworks. *International Journal of Soft Computing and Engineering (IJSCE)*, *10*(6), 1-12.

SOFCAVI (2023). *Que es Flutter.* https://bit.ly/4jWYZVA

Stanic, N., y Ćirković, S. (2024). Analysis of Approaches to Developing Kotlin Multiplatform Applications and Their Impact on Software Engineering. In *10th International Scientific Conference Technics, Informatics and Education-TIE 2024*, *1*(1), 23-35.

Suri, B., Taneja, S., Bhanot, I., Sharma, H., y Raj, A. (2022). Cross-platform empirical analysis of mobile application development frameworks: Kotlin, react native and flutter. In *Proceedings of the 4th International Conference on Information Management & Machine Intelligence*.

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional

Work Nest Oficial (2020). *Lenguaje de programación Swift.* https://bit.ly/4laVMT6

Zarichuk, O. (2023). Comparative analysis of frameworks for mobile application development: Native, hybrid, or cross-platform solutions. *Вісник Черкаського державного технологічного університету. Технічні науки*, *28*(4), 19-27.

Conflicto de intereses:

Los autores declaran que no existe conflicto de interés posible.

Financiamiento:

No existió asistencia financiera de partes externas al presente artículo.

Agradecimiento:

N/A

Nota:

El artículo no es producto de una publicación anterior.

Esta obra está bajo una Licencia Creative Commons Atribución No Comercial-Sin Derivar 4.0 Internacional