



UNIVERSIDAD TÉCNICA DE BABAHOYO
FACULTAD DE ADMINISTRACIÓN, FINANZAS E
INFORMÁTICA

PROCESO DE TITULACION
MAYO – SEPTIEMBRE 2022
EXAMEN COMPLEXIVO DE GRADO O FIN DE CARRERA
PRUEBA PRÁCTICA
INGENIERIA EN SISTEMAS
PREVIO A LA OBTENCION DEL TITULO DE INGENIERIA EN
SISTEMAS

TEMA:
ANALISIS COMPARATIVO DE TECNOLOGIAS FLUTTER VS SWIFT
EN EL MODELO DE PROCESOS PARA EL DESARROLLO DE
APLICACIONES MOVILES.

EGRESADA:
KASANDRA THALIA SUAREZ GARCIA
TUTOR:
JOFFRE VICENTE LEON ACURIO

RESUMEN Y PALABRAS CLAVES

Flutter y Swift de momento son dos de las soluciones de desarrollo de aplicaciones móviles más populares hasta la fecha, por cualquiera de los dos marcos de desarrollo de aplicaciones puede ser difícil, debido a que las 2 son preferidas para las empresas que estarían creando aplicaciones para Android y iOS. El objetivo de este proyecto es la realización de un análisis comparativo entre: Flutter vs. Swift, con esta comparación nos permitirá detallar cada uno de sus pro y contras al momento de poner en producción una aplicación móvil, sus características proporcionadas por ambos y el motivo de su popularidad.

Existen herramientas que están orientadas para desarrollar proyectos pequeños y otras que son más escalables, óptimos y tienen un enfoque para aplicaciones de índole empresarial, pero llevar a cabo nuestras soluciones móviles con ayuda de un Framework puede deberse a varios motivos como puede ser que el Framework se adapta a una arquitectura establecida, estas herramientas constituyen algunas librerías basadas en componentes web como lo son Flutter y Swift. Algunos en la actualidad están basados en C++ se usan en el desarrollo de la mayoría de las aplicaciones y siempre están apareciendo herramientas novedosas que buscan liderar el mercado.

Para culminar este documento acerca del desarrollo móvil, más específico en los frameworks basados en C++ puedo decir que el crecimiento que han tenido estas tecnologías multiplataforma en los últimos años es grande, porque las herramientas entregadas al desarrollador y su comunidad de apoyo que trabaja con ellas.

PALABRAS CLAVE: Frameworks, herramientas, compilador, arquitectura, prototipo, Flutter, Swift, librerías.

ABSTRACT

Flutter and Swift at the moment are two of the most popular mobile app development solutions to date, for either of the two app development frameworks it can be difficult, due to the fact that the 2 are preferred for companies that would be creating applications for Android and iOS. The objective of this project is to carry out a comparative analysis between: Flutter vs. Swift, with this comparison, will allow us to detail each of its pros and cons when putting a mobile application into production, its features provided by both and the reason for its popularity.

There are tools that are aimed at developing small projects and others that are more scalable, optimal and have a focus on business applications, but carrying out our mobile solutions with the help of a Framework can be due to several reasons, such as the Framework adapts to an established architecture, these tools constitute some libraries based on web components such as Flutter and Swift. Some are currently based on C++, they are used in the development of most applications and new tools are always appearing that seek to lead the market.

To conclude this document about mobile development, more specifically in frameworks based on C++, I can say that the growth that these multiplatform technologies have had in recent years is great, because the tools delivered to the developer and their support community that works with them .

KEYWORDS: Frameworks, tools, compiler, architecture, prototype, Flutter, Swift, libraries.

INTRODUCCIÓN

El desarrollo de aplicaciones orientadas a dispositivos móviles permite que las personas tengan diversas herramientas en sus smartphones que les ayudan en su día a día, pero la creación de aplicaciones complejas o que están pensadas para realizar muchas tareas puede ser un trabajo muy pesado incluso para un equipo de programación empresarial, por lo que, con el tiempo, se han propuesto distintas alternativas que permiten hacer que los proyectos de programación traspasen las barreras de las plataformas y que así se disminuyan los costes y tiempos de programación.

Al crecer la escala de desarrollar softwares también debe hacerlo las distintas herramientas que facilitan su desarrollo, teniendo una variedad de lenguajes de programación y Frameworks, gracias a aquellas alternativas ayudara a que los desarrolladores web tengan ciertas dudas y dificultades al elegir, porque cada lenguaje tiene sus características, fortalezas y limitaciones que pueden influir o no en el proyecto a desarrollar.

El objetivo de este estudio comparativo de estas aplicaciones es percibir por medio de una investigación extenuada sus diferencias, ventajas, etc. Se realizará a obtener información mediante páginas web, libros, como estas tecnologías ayudan para el desarrollo de aplicación para Android ya que ambas son tecnologías preferidas por las organizaciones que consideran crear aplicaciones para Android y iOS. Flutter y Swift son dos de las soluciones de desarrollo de aplicaciones más populares hasta la fecha desempeñar en un dispositivo móvil y así ir destacando cuál de estos sistemas es los más accesibles o factibles para que el móvil tenga un excelente rendimiento.

Se utilizó el método descriptivo que ayudara para juntar, organizar, planificar resumir, entregar información que se ha llegado a recopilar en distintas páginas web o fuentes bibliográficas. Consiste en describir y evaluar dichas características, desventajas, ventajas, funciones de las aplicaciones. Los fines de este método descriptivo es ir seleccionando información que existe sobre el estudio de estas tecnologías en comparación.

El presente caso de estudio se realizó siguiendo la línea de investigación de Sistemas de información y comunicación, emprendimiento e innovación con referente sub línea de investigación en redes y tecnologías inteligentes de software y hardware, también el estudio comparativo de las herramientas Flutter y Swift para gestión administrativa que tienen como finalidad la optimización de los procesos empresariales, en la investigación se utilizó como técnica la recopilación documental y bibliográfica.

Desde el enfoque de la Ingeniería de Sistemas, una propiedad fundamental que debe poseer un software móvil exitoso es que sea de calidad, la buena selección de las herramientas de desarrollo, puede influir para que un software web se desarrolle eficientemente y pueda cumplir con las expectativas de los usuarios; día a día se vive un debate de qué herramientas realizaran un aporte mayor para un proyecto es por esta razón que se realizará un estudio comparativo entre Frameworks principales como Flutter Vs Swift, con el fin de brindar un mejor conocimiento de acorde con sus características y ventajas que cada uno de estos puede ofrecer.

DESARROLLO

En el presente desarrollo del estudio de caso presentaremos tecnologías interesantes al momento de desarrollar una aplicación móvil, detallando como objetivos a Flutter y Swift, dando una explicación un poco más a fondo de la estructura y componentes que estas usan. Focalizando aquello tenemos a C++, un lenguaje de programación cuyo objetivo de su nacimiento fue extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos, desde el punto de vista de los lenguajes orientados a objetos, C++ es un lenguaje híbrido.

Luego se añadieron facilidades de programación genérica, que se sumaron a los paradigmas de programación estructurada y programación orientada a objetos razón por el cual es un lenguaje de programación multiparadigma.

FLUTTER

Es un SDK de código fuente abierto de desarrollo de aplicaciones móviles creado por Google es usada en el desarrollo de interfaces de usuario para aplicaciones en Android, iOS y Web, así como método primario para crear aplicaciones para Google Fuchsia. En los últimos 3 años obtuvo un exponencial crecimiento en cuanto a su popularidad; debido a su velocidad de desarrollo, experiencia nativa y renderización de las aplicaciones.

Su primera versión de Flutter es conocida como "Sky" y corrió en el sistema operativo de Android, se lanzó en el Dart developer summit de 2015, declarado de ser capaz de hacer un render a 120 fps (fotogramas por segundo). (Ron, 2021).

Según (Google, 2020) para junio de 2020 las empresas Canonical y Google dieron a conocer que se unificarán para llevar dicho entorno al sistema operativo Linux, creando

primeramente un sistema Beta para probarlo. A continuación detallaremos cada uno de sus componentes de Flutter:

- Dart platform
- Flutter engine
- Foundation library
- Design-specific widgets
- Flutter Development Tools (DevTools)

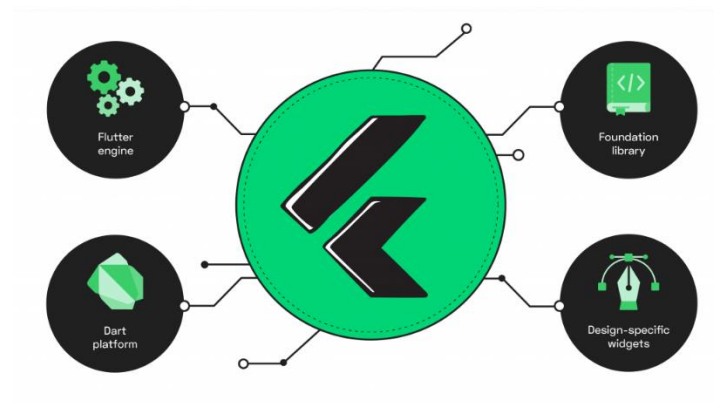


Figura. Componentes Flutter.

Fuente: <https://www.dart.dev>

Dart platform: Las apps de Flutter están escritas en Dart (lenguaje de programación) y hace uso de muchas de las características más avanzadas.



Figura 1: Logo Dart.

Fuente: <https://www.dart.dev>

Flutter engine: Flutter engine, está escrito principalmente en C++, proporciona un soporte de bajo-nivel para renderización que utiliza Google Skia. Además, se vincula con SDKs de Android e iOS, utilizando MethodChannels y EventChannels que permiten la comunicación entre el Flutter engine y el nivel nativo del sistema operativo. Los lenguajes que pueden parecer a Dart y con los que se puede familiarizar la implementación, además de C++ serían: Kotlin, Swift y TypeScript. (Flutter, 2021).

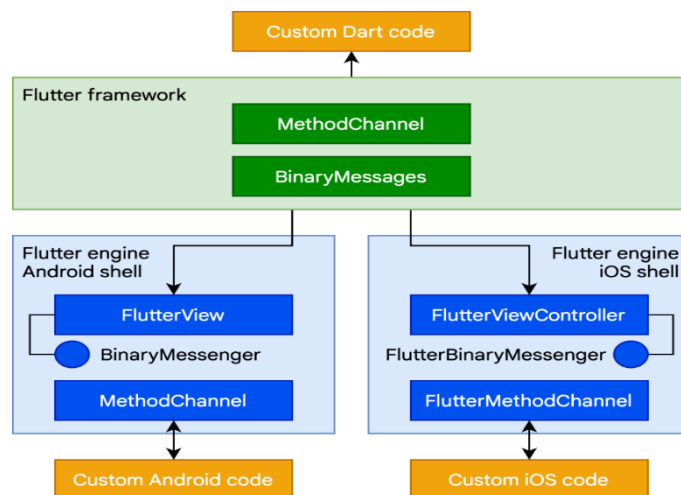


Figura 2: Arquitectura de Flutter.

Fuente: <https://docs.flutter.dev>

Foundation library: Está escrito en Dart, proporciona clases básicas y funciones las cuales suelen construir las aplicaciones que utilizan Flutter, como APIs para comunicar con el motor.

Widgets: UI Diseño en Flutter implica reunir y/o crear varios widgets. Un widget en Flutter representa una descripción inmutable de parte de la interfaz de usuario; todos los gráficos, incluyendo texto, formas, y las animaciones están creadas utilizando widgets. Complejos widgets pueden ser creados combinando otros más sencillos. (Flutter, 2021).

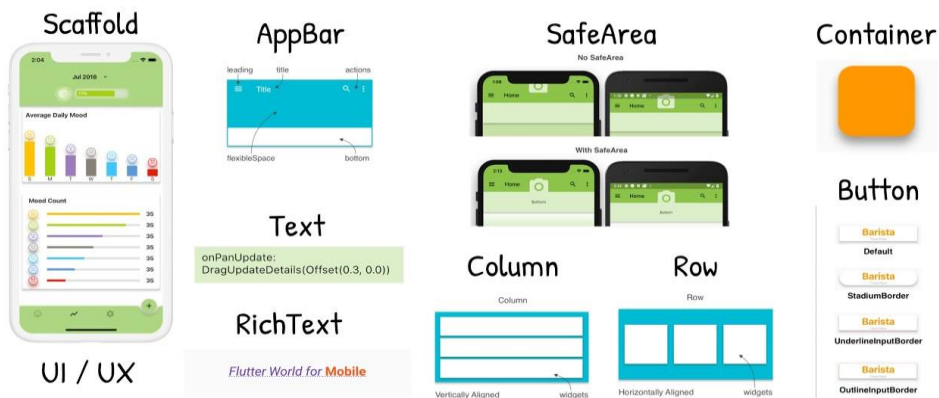


Figura 3: Widgets de Flutter.

Fuente: <https://docs.flutter.dev>

Design-specific widgets: Flutter framework contiene dos conjuntos de widgets que conforman un lenguaje de diseño concreto. Material Design Widgets implementa el lenguaje de diseño de Google del mismo nombre, y Cupertino widgets imita el diseño de Apple iOS.

```

1 import 'package:flutter/material.dart';
2
3 void main() => runApp(HelloWorldApp());
4
5 class HelloWorldApp extends StatelessWidget {
6   @override
7   Widget build(BuildContext context) {
8
9     return MaterialApp(
10      //El nombre de nuestra aplicación se determina con "title"
11      title: 'Aplicación Hello World',
12
13
14      home: Scaffold(
15        appBar: AppBar(
16          title: Text('Hola'),
17        ),
18        body: Center(
19          // imprime "Hello World" en la pantalla.
20          child: Text('Hello World'),
21        ),
22      ),
23    );
24  };
25 }
26 }

```

Figura 4: Fragmento de código Hola Mundo en Flutter.

Fuente: <https://docs.flutter.dev>

SWIFT

Es un lenguaje de programación multiparadigma creado por Apple enfocado en el desarrollo de aplicaciones para iOS y macOS, presentado en la Conferencia Mundial de Desarrolladores de Apple 2014 diseñado para integrarse con los Frameworks Cocoa y Cocoa Touch; puede usar cualquier biblioteca programada en Objective-C y utilizar funciones de C, es posible desarrollar código en Swift compatible con Objective-C bajo ciertas condiciones. Su intención de ser un lenguaje seguro, de desarrollo rápido y conciso, se presentó como un lenguaje propietario, para luego en el año 2015, con la versión 2.2, pasó a ser de código abierto con la Licencia Apache 2.0. (Swift, 2021).



Figura 5: Logo de Swift.

Fuente: <https://www.applesfera.com>

Los tipos de datos se dividen principalmente en dos grupos que son:

✚ **Tipo de valor:** aquí se guarda una copia de su contenido, es recomendable usarlo cuando se va a copiar su información o se va a trabajar en múltiples hilos.

✚ **Tipo por referencia:** aquí suele asignarle una instancia compartida que es mutable aún si son usadas en constantes, quiere decir que modificar una instancia se verá reflejado en todas las variables y constantes que compartan, es recomendable usarlo cuando se requiera compartir datos mutables.

COMPARATIVA MAS RELEVANTES ENTRE FLUTTER vs. SWIFT

Características principales de Flutter

OpenSource: Flutter al ser una plataforma opensource o de código abierto lo ayuda a transformar sus estrategias de aplicaciones creativas en una solución de aplicación elegante y rentable con el apoyo de una empresa confiable de creación de aplicaciones Flutter, sus widgets, API de movimiento de Cupertino y los diseños de materiales integrados son de gran utilidad para crear aplicaciones fáciles de usar haciéndose digno de ser utilizado por desarrolladores que pueden

explorar libremente varias funciones mientras crean una aplicación de la forma más personalizada posible.

Carga Realtime: Tiene una ventaja debido a que se puede cambiar el proceso de creación de la aplicación, incluidos los códigos y las ideas, haciendo que estos cambios sean visibles en la pantalla al instante. Muestra las actualizaciones en los proyectos en curso y permite a los desarrolladores experimentar fácil y rápidamente.

Widgets: Su gran catálogo de widgets lo convierte en una opción ideal para los desarrolladores ya que pueden crear una interfaz de usuario expresiva y elegante combinando diferentes widgets de su elección. También pueden crear sus widgets personalizados e incorporarlos a su proyecto actual, sin dejar de tener la seguridad del rendimiento nativo de la aplicación.

Soporte: Gracias a Google los desarrolladores pueden aprovechar al máximo el soporte de Firebase de Google como backend. Pueden utilizar esta ayuda para crear aplicaciones escalables y fáciles de usar.

Ventajas de Flutter

Herramientas: Se programa usando Dart, que es un lenguaje de programación bastante poderoso, su cadena de herramientas posee pub, un potente administrador de complementos y una gran cantidad de otros complementos disponibles a través del repositorio pub.dev.

Extensión: Algunos desarrolladores que requieren una función para la que no existe ningún complemento en Flutter, codificar uno no es demasiado difícil con la extensión create.plugin.

Rendimiento: El rendimiento de la aplicación es de suma importancia, lo que hace que cualquier tarea de optimización del rendimiento sea conveniente con este marco de desarrollo de aplicaciones.

Fácil de aprender: Flutter proporciona videos y tutoriales, lo que facilita que los usuarios aprendan e implementen estrategias de desarrollo antes de lanzar aplicaciones en el mercado.

Integraciones: Flutter ofrece integraciones de editor populares y fáciles de usar, incluido Android Studio. Los usuarios tienen acceso a otras instrucciones disponibles necesarias para configurar VSCode, Emacs o IntelliJ.

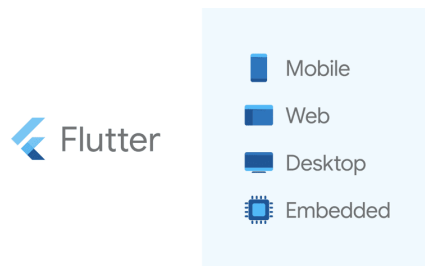


Figura 6: Logo Flutter.

Fuente: <https://www.platzi.com>

Desventajas de Flutter

- El código del programa puede volverse confuso al integrar los widgets.
- En caso de actualizar los módulos Flutter se integran en el programa de manera fija, se compila el programa e instalarlo en los dispositivos.
- Es un lenguaje nuevo y poco extendido, cuenta con una comunidad reducida.

Características principales de Swift

El marco de desarrollo de aplicaciones Swift incluye una variedad de características que facilitan a los programadores la lectura, escritura y evaluación de código, al tiempo que les brinda el control total requerido en un verdadero lenguaje de programación de sistemas.

Swift renderiza tipos inferidos para hacer que el código sea transparente y menos propenso a errores, y los módulos reducen los encabezados y ofrecen espacios de nombres. Con Swift, la memoria se administra automáticamente y los desarrolladores no necesitan escribir punto y coma. Swift toma prestado de otros lenguajes de programación, como Objective-C.

Por ejemplo, Swift toma prestados parámetros con nombre de Objective-C y los expresa en una sintaxis limpia que hace que las interfaces de programación de aplicaciones en Swift sean fáciles de leer y mantener.

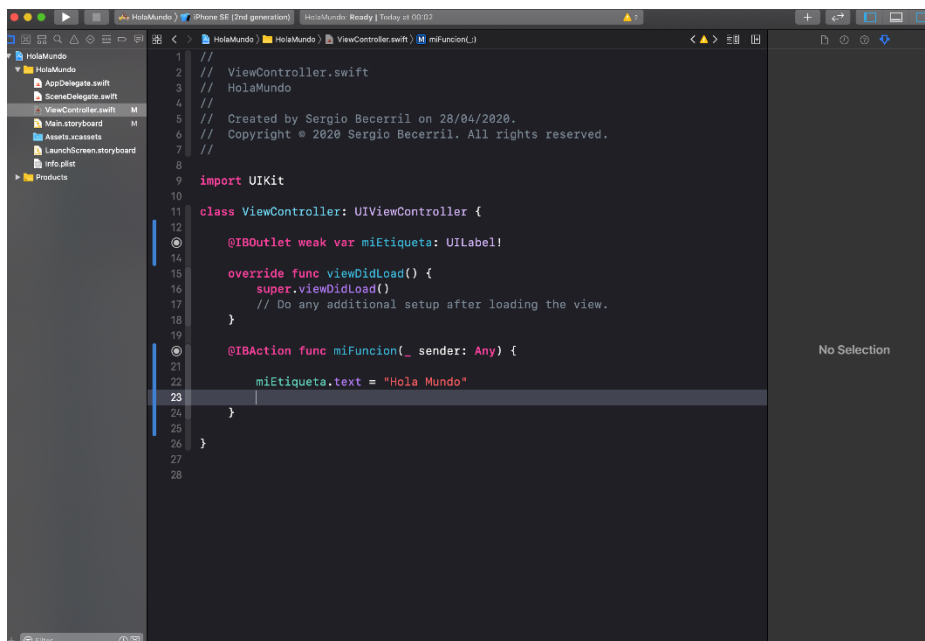
Además de las increíbles características mencionadas anteriormente, hay algunas más que hacen de Swift un poderoso lenguaje de programación. Éstos incluyen:

- Múltiples valores devueltos
- Iteración concisa
- Patrones de programación funcional, por ejemplo, filtro y mapeado
- Cierres unificados con parámetros de función
- Genéricos
- Estructuras que admiten protocolos, métodos y extensiones.

Ventajas de Swift

OpenSource: Al ser una plataforma de desarrollo de código abierto, evolucionó a través del tiempo con la sólida comunidad de desarrolladores, mejorando su experiencia de codificación con el lenguaje Swift explorando varias características y funciones de diseño, contribuyendo a la corrección de errores y agregando plataformas y características exclusivas.

Mantenimiento del código: Swift es un lenguaje de programación simple que requiere que escriba algunas líneas de código, mientras que tiene una sintaxis simple durante todo el proceso de codificación. Cuando se trata de mantener el código, ningún otro lenguaje puede vencer a Swift. Por ejemplo, no es necesario agregar comas después o entre paréntesis, por lo que no hay bucles y puede continuar escribiendo códigos fácilmente sin agregar punto y coma, por lo que no hay errores.



```
1 //
2 // ViewController.swift
3 // HolaMundo
4 //
5 // Created by Sergio Becerril on 28/04/2020.
6 // Copyright © 2020 Sergio Becerril. All rights reserved.
7 //
8
9 import UIKit
10
11 class ViewController: UIViewController {
12
13     @IBOutlet weak var miEtiqueta: UILabel!
14
15     override func viewDidLoad() {
16         super.viewDidLoad()
17         // Do any additional setup after loading the view.
18     }
19
20     @IBAction func miFuncion(_ sender: Any) {
21
22         miEtiqueta.text = "Hola Mundo"
23     }
24 }
25
26 }
27
28
```

Figura7: Fragmento de código Hola Mundo en Swift.

Fuente: <https://www.cfeapps.com>

Anteriormente, el lenguaje C requería que los desarrolladores mantuvieran dos archivos de código para mejorar la eficiencia y el tiempo de compilación. En Swift, los programadores no necesitan mantener dos archivos de código y todo el contenido de la implementación y los archivos de encabezado se combinan en un solo archivo. Esto acelera el proceso de desarrollo de la aplicación y mejora la eficiencia.

Compatibilidad: La interoperabilidad es una característica que permite a los desarrolladores escribir código en un lenguaje y usarlo a su conveniencia en otro lenguaje dicha interoperabilidad es posible entre Objective-C y Swift debido a que los desarrolladores que crean aplicaciones de Apple importan el marco de Objective y usan sus métodos y clases usando la sintaxis Swift. Por lo tanto, los desarrolladores pueden hacer uso de la función de interoperabilidad de Swift y utilizar las API de Objective-C para crear aplicaciones.



Figura 8: Componentes Objective-C.

Fuente: [https:// ferestrepoca.github.io](https://ferestrepoca.github.io)

Soporte Multiplataforma: El lenguaje de programación Swift creado por Apple Inc es compatible con dispositivos iPhone, iPad, Apple Watch, Apple TV y Mac, también admite la creación de aplicaciones para dispositivos Windows y Linux. De hecho, según las últimas noticias publicadas por Google, el próximo sistema operativo de Google, Fuchsia, admitirá el lenguaje de programación Swift de Apple.

Mejoras de Ejecución: Según el anuncio hecho por Apple, Swift es 3,4 veces más rápido que Objective-C, en otros estudios demuestran que la ejecución de código con Swift es más rápida que Objective-C; procesa múltiples dispositivos y bibliotecas dinámicas que aceleran el desarrollo de aplicaciones y los procesos de actualización.

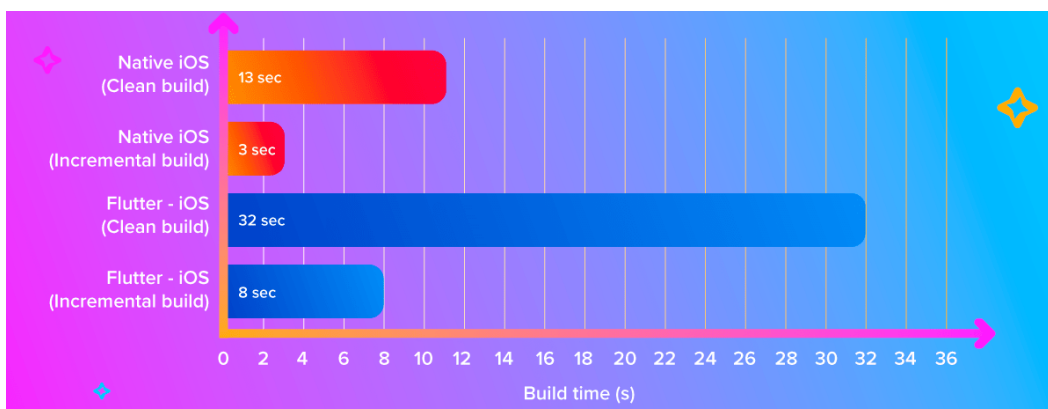


Figura 9: Comparación de Tiempo de Construcción Swift vs Flutter.

Fuente: <https://blog.codemagic.io>

Desventajas de Swift

- Falta de soporte para versiones anteriores de iOS
- Aprender el idioma requiere de tiempo y esfuerzo.

TABLA COMPARATIVA DE CARACTERÍSTICAS DE FRAMEWORKS

| | Flutter | Swift |
|---------------------------------|--|---|
| Descripción | Marco de desarrollo multiplataforma de Google. | El lenguaje de programación iOS más nuevo de Apple. |
| Lenguaje de programación | Dart | Swift |
| Multiplataforma | Sí | No |
| Desarrollo nativo | No | Sí |
| Rendimiento | Muy bueno | El mejor |
| Popularidad | Muy popular | popular |
| Interfaz de usuario | Casi nativo | Nativo |

| | | |
|----------------------|---|--|
| Precio | Gratis | Gratis |
| Tecnología | De código abierto | De código abierto |
| Documentación | Muy buena | Muy buena |
| Costos | Más económico al momento de desarrollo multiplataforma. Similar solo para el desarrollo de iOS. | Más caro para el desarrollo multiplataforma. Similar solo para el desarrollo de iOS. |

Tabla 1: Comparativa de características de Flutter vs. Swift.

Fuente: Kasandra Suarez

CONCLUSIONES

Para finalizar este estudio de caso, se analizó los enfoques de desarrollo nativo, interpretado, híbrido y generado entre Flutter vs Swift, para realizar un análisis comparativo de las características, ventajas, desventajas, rendimiento, construcción, herramientas y pruebas, que pueda ser de utilidad para el desarrollador al momento de tomar la decisión de cuál enfoque usar en un nuevo proyecto, se diseñaron una serie de casos de prueba.

Para ello, se estudiaron las principales problemáticas del desarrollo de aplicaciones móviles, y se eligieron dos factores que pueden resultar claves para el éxito de una aplicación, impactados con sus recursos tales como el consumo de energía de la batería, y el uso de espacio de almacenamiento persistente.

En primer lugar, el enfoque nativo, que implica utilizar las herramientas provistas por los fabricantes de cada una de las plataformas a las que se quiera llegar. Esto trae las desventajas de tener que encarar un desarrollo por separado para cada plataforma. Apuntando a mejorar esta situación, es que surgen los frameworks multiplataforma. Estos a su vez tienen diferentes enfoques para la generación de aplicaciones en base a un único desarrollo.

Al momento de analizar el consumo de energía, se desarrolló 3 aplicaciones diferentes, cada una de ellas implementada en todos los enfoques de desarrollo estudiados. En la primera se realizó un procesamiento intensivo, la segunda, una de reproducción de video, y finalmente una de reproducción de audio. Al tener una comparación entre resultados, se tiene que Flutter es favorecido en mayores puntos que Swift, cabe destacar que existen otros que tienden a ser los mejorcitos, posibilitando a futuro un interesante análisis comparativo.

ANEXOS



Babahoyo, 12 de agosto del 2022

CERTIFICACIÓN DE PORCENTAJE DE SIMILITUD CON OTRAS FUENTES EN EL SISTEMA DE ANTIPLAGIO

En mi calidad de Tutor del Trabajo de Investigación de la Srta.: **SUAREZ GARCIA KASANDRA THALIA** cuyo tema es **ANÁLISIS COMPARATIVO DE TECNOLOGIAS FLUTTER VS SWIFT EN EL MODELO DE PROCESOS PARA EL DESARROLLO DE APLICACIONES MÓVILES**, Certifico que este trabajo investigativo fue analizado por el Sistema Antiplagio Compilatio, obteniendo como porcentaje de similitud de **[6%]**, resultados que evidenciaron las fuentes principales y secundarias que se deben considerar para ser citadas y referenciadas de acuerdo a las normas de redacción adoptadas por la institución y Facultad.

Considerando que, en el Informe Final el porcentaje máximo permitido es el 10% de similitud, queda aprobado para su publicación.



Por lo que se adjunta una captura de pantalla donde se muestra el resultado del porcentaje indicado.

ING. LEÓN ACURIO JOFFRE VICENTE.
DOCENTE DE LA FAFI.