UNIVERSIDAD TÉCNICA DE BABAHOYO

FACULTAD DE ADMINISTRACIÓN, FINANZAS E INFORMÁTICA

PROCESO DE TITULACIÓN

JUNIO 2021 – NOVIEMBRE 2021

EXAMEN COMPLEXIVO DE GRADO O DE FIN DE CARRERA

PRUEBA PRÁCTICA

INGENIERÍA EN SISTEMAS

TEMA:

ESTUDIO COMPARATIVO DE LOS FRAMEWORKS DEL DESARROLLO MÓVIL NATIVO "FLUTTER" Y "REACT NATIVE".

EGRESADO(A):

ERICK VICENTE MACIAS VERA

TUTOR:

ING. JOFFRE VICENTE LEON ACURIO

BABAHOYO – ECUADOR - 2021

INTRODUCCIÓN

El conocimiento en la programación de aplicaciones para los dispositivos móviles ha avanzado de una manera excepcional siendo una necesidad debido a la rápida implantación y evolución de estas plataformas móviles.

La rápida evolución crea una problemática al momento de elegir que tecnología se debe utilizar para la programación de aplicaciones móviles, su desarrollo también cuenta en los distintos sistemas en los dispositivos móviles como Android y IOS.

La presente documentación del caso de estudio tiene como título estudio comparativo de los Frameworks del desarrollo móvil nativo o el desarrollo de aplicaciones multiplataforma "Flutter" y "React Native", es decir al escribir solo una base de código nativa se la pueda compartir entre varias plataformas como dispositivos móviles, web y de escritorio, su objetivo principal de esta acción formativa es de un estudio comparativo de los distintos frameworks del desarrollo nativo móvil para averiguar CPU, GPU, uso de memoria y determinar sus características, ventajas, desventajas y realizar ejemplos y pruebas de estas dos tecnologías.

La tecnología Flutter utiliza el lenguaje open source desarrollado por google con el objetivo de permitir a los programadores utilizar un lenguaje orientado a objetos, Dart se diseñó con el propósito de hacer los procesos de desarrollo más cómodos y rápidos para los desarrolladores. La máquina virtual de Dart y su compilación just-in-Time realizan los cambios en el código de forma inmediata.

La tecnología React Native utiliza el lenguaje open source desarrollado por Facebook es un framework JavaScript para la creación de aplicaciones reales nativas para Android y IOS, inspirada en la librería de JavaScript de React en la creación de los componentes virtuales. Las actualizaciones rápidas se dan a notar tan pronto como guarde

los cambios. Con el poder de JavaScript, React Native nos permite iterar a una velocidad favorable.

En la investigación se utilizó la metodología comparativa y documental la misma que ayudo a saber las diferencias de cada de una de las tecnologías móviles, recaudamos la información de sus características y ventajas, se empleó la técnica de la observación con su instrumento de guía de observación.

La línea de investigación presente en este proyecto está encaminada al desarrollo de sistemas de información y comunicación, emprendimiento e innovación correspondiente a la sub línea redes y tecnologías de software y hardware.

DESARROLLO

Los frameworks de desarrollo para aplicaciones móviles nos brindan capacidad de uso a multiplataforma que solo utilizan una única base de código nativa, existen diversas limitaciones como la construcción de sus interfaces de usuario, el hardware ya que se debe desarrollar el diseño de interfaces (UI) para varios dispositivos móviles y su acceso al software. Se suele evidenciar que la mayoría de las aplicaciones desarrollada con estos frameworks proporcionan la esencia de la compilación del código nativo específicas de estas plataformas. Pero surgen muchas diferencias entre los dos frameworks como: la compilación del código, sus sintaxis, como se mapean los elementos en la interfaz de usuario, el acceso a las capacidades del dispositivo y la memoria que se consume en el proceso de compilación, etc.

Por eso la presente documentación tiene como objetivo responder a tres preguntas principales sobre las aplicaciones desarrolladas en los Frameworks Flutter y React Native:

¿Cuáles son sus diferencias de las aplicaciones móviles desarrolladas en los frameworks Flutter y React Native?

¿Cuáles son las ventajas y desventajas de las aplicaciones móviles desarrolladas en estos dos frameworks?

El caso de estudio se centra en la investigación comparativa de multiplataforma móviles entre Flutter y React Native. El marco de desarrollo presenta comparaciones de su sintaxis, su instalación independiente de ambos frameworks comparándolas con hechos sobre su diferencia, ventajas y desventajas.

Debido a las limitaciones de tiempo los ejemplos de pruebas contienen funciones como su introducción al lenguaje Dart, TypeScript, sus componentes, estas funciones solo son un subconjunto de la amplia gama de características que pueden contener ambos frameworks.

Categoría de desarrollo de las aplicaciones.

Se puede decir que el desarrollo de aplicaciones móviles son un conjunto de procesos involucrados en la escritura de un software para distintos dispositivos informáticos, como teléfonos inteligentes y portátiles.

Una aplicación esta lista cuando pasa por varios procesos que son llamados colectivamente ciclos de vida de desarrollo.

Los ciclos de vida para el desarrollo de aplicaciones móviles incluyen los siguientes pasos:

- **Planificación:** identifican la necesidad de la aplicación, especificar los objetivos que requieran los usuarios, especificar las plataformas a utilizar.

- **Análisis:** incluyen los requerimientos funcionales que va tener la aplicación y la predicción de los problemas que pueden suceder dentro del ciclo de vida de la aplicación.
- **Diseño:** debe ser observada desde el punto de vista de cada usuario, define características, componentes y piezas fundamentales de la aplicación.
- **Construcción:** también llamada como codificación, se realiza la programación verificando los requisitos y diseño.
- **Pruebas:** testean la aplicación para encontrar errores. También se debe confirmar que se cumplan la especificación de los requisitos y el diseño.
- **Producción:** se termina todos los procesos cumpliendo las funcionalidades y requerimientos, incluye la documentación o manual de usuario de su uso.
- **Mantenimiento:** supervisan los cambios, actualizaciones y mejoras según la experiencia del usuario.

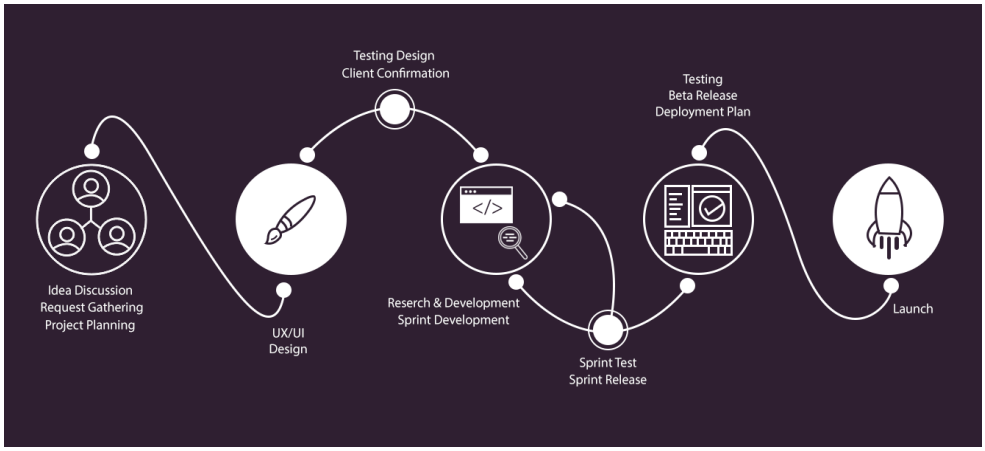

Figura 1 *Ciclo vida del desarrollo de aplicación móvil*

Fuente: https://www.kookyinfomedia.com/about-us

Aplicaciones nativas

Según los autores (Delía, Galdámez, Thomas, & Pesado, 2013) es una aplicación desarrollado para el uso en una plataforma o dispositivos similares ya

sea Windows, Android, IOS y Linux, proceso similar que el utilizado para las tradicionales aplicaciones de escritorio, eso nos permite aprovechar todas las ventajas del SO. Uno de los problemas es que se debe desarrollar una versión del programa para cada dispositivo. Estas aplicaciones son de procesamiento rápido, brindan la excelencia en la experiencia de usuario.

Se denominan aplicaciones nativas porque se desarrollan en el lenguaje nativo del propio dispositivo, es decir si la aplicación es desarrollada en un dispositivo Android utiliza el lenguaje Java, klotin desarrollado por google en la plataforma Android Studio y si es desarrollada en IOS utiliza el lenguaje Objetive C, Swift desarrollados por Apple en la plataforma XCode.

La gran ventaja de las aplicaciones es la interacción con las capacidades de los dispositivos móviles, su ejecución es rápida.

Sistemas Operativos	Fabricante	Lenguaje	Plataforma
Android	Google LLC	Klotin y Java	Android Studio
IOS	Apple Inc.	Swift y Objetive C	XCode

Tabla 1 comparación sistemas operativos
Fuente: Erick Vicente Macias Vera

Aplicaciones hibridas

Estas aplicaciones hibridas tienen una capa de abstracción sobre el sistema que se esté desarrollando esta tiene la función de encapsular una aplicación desarrollada en el entorno de etiquetas o Html5 convirtiéndola en una aplicación nativa.

Según el autor (Triguero, 2017) se puede desarrollar únicamente un código fuente para distintas plataformas reduciendo el desarrollo como el mantenimiento de las aplicaciones. Esta tecnología reduce el plazo de lanzamiento y varios desarrolladores nos dan a conocer de un ahorro de costos de un 30% y 50% respecto al tipo de aplicación nativa.

Esta tecnología es fundamentada en las siguientes bases tecnológicas: Html5, Css y JavaScript. Todos los sistemas nativos cuentan con un navegador web, basado en la norma Webkit encapsulan la aplicación nativa con el webkit. Tienen facilidad de conectar con las API's nativas de cada entorno, por esta manera se tiene acceso a las funcionalidades de las aplicaciones nativas como: notificaciones push, cámara, GPS, sensores, etc.

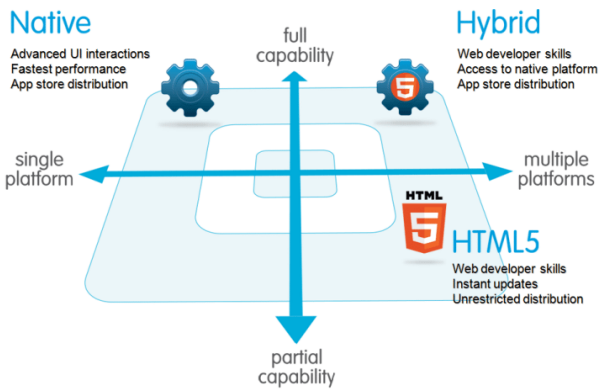

Figura 2 *Aplicaciones Hibridas*

Fuente: https://profile.es/blog/aplicaciones-moviles-hibridas-la-solucion-mas-eficiente-para-el-desarrollo-multiplataforma/

Flutter

El framework que nos facilita un toolkit o conjunto de herramientas el mismo tiene como objetivo crear interfaces de usuario. Creado por la empresa Google, se inició en el año 2015, pero su producción fue en el año 2018. En el principio, este Framework fue desarrollado para el desarrollo aplicaciones que pueden ejecutarse en Android y IOS, y

su segundo enfocado a desarrollar aplicaciones híbridas con rendimiento nativo (Qualitydevs.com, 2019).

En las nuevas actualizaciones como Flutter 2 permite realizar aplicaciones multiplataforma hasta para 6 plataformas que utilizan el lenguaje de programación Dart creado por google.

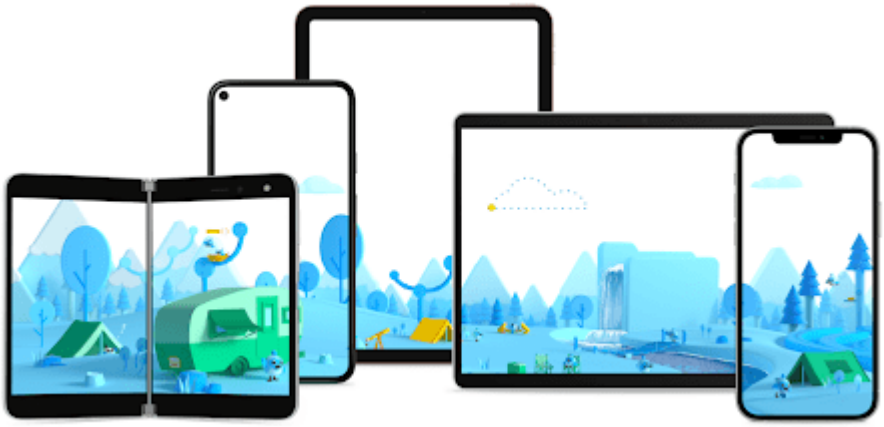

Figura 3 *Flutter multiplataforma*

Fuente: https://esflutter.dev/

Existen diversas características sobre el framework de Flutter que se darán a conocer:

Desarrollo rápido

El desarrollo rápido le da vida a la aplicación en milisegundos con los Stateful Hot Reload (recarga en caliente con estados), utilizando un conjunto de widgets totalmente personalizados para la creación nativas en minutos.

Interfaces de usuario expresivas

Se crean hermosas aplicaciones rápidas con la colección de widgets de Material Design y cupertino (IOS) Visuales, estructurados e interactivos del framework Flutter, con la API de diseño enriquecido.

Rendimiento Nativo

Los widgets de Flutter incorporan una alta interacción de la plataforma, como las animaciones, la navegación, los iconos y sus fuentes para proporcionar el rendimiento nativo tanto en Android como en IOS.

¿Cuáles son sus ventajas y desventajas de Flutter?

Ventajas

- Con solo una base de código se puede ejecutar para diversas plataformas de destino.
- Lenguaje de programación Dart de fácil aprendizaje.
- Bibliotecas amplias con elementos prefabricados de interfaz gráfica.
- El Hot Reload acelera las pruebas durante el desarrollo.
- Ejecución potente de la aplicación nativa en los dispositivos móviles.

Desventajas

- En caso de actualizar el diseño de la aplicación, hay que actualizar los módulos de Flutter.
- Como es un lenguaje nuevo no está muy extendido y cuenta con una comunidad reducida.

Dart

El lenguaje Dart de Google empezó a desarrollarse en 2010 y se presentó un año después. Como los navegadores no podían, ni pueden, trabajar con este

lenguaje de forma natural, y JavaScript puede ejecutarse en todos los navegadores actuales, existe el compilador Dart2js, es decir, "Dart para JavaScript". El lenguaje Dart se asemeja a los ya establecidos lenguajes de programación orientados a objetos, entre los que se encuentran Swift, C# o Java, que se subordinan a determinados paradigmas de programación. Las reglas para combinar caracteres definidos, es decir la sintaxis, son similares al lenguaje C. Esta semejanza facilita enormemente el aprendizaje, de manera que es posible iniciarse en él sin tener que enfrentarse a grandes problemas de lenguaje (Digital Guide IONOS, 2020).

Estructura de Dart

Este lenguaje dispone de variables, operadores, condiciones, bucles, funciones, clases, objetos y listas, etc. Ofrece los conceptos importantes de la programación orientada a objetos como herencia y programación genérica, si deseamos aprender este lenguaje debemos comenzar con las pautas básicas desde la plataforma gratuita DartPad, podemos ver algunos ejemplos:

Lo primero se debe hacer es buscar en un navegador web **www.dartpad.dartlang.org** y veremos su sintaxis imprimiendo mediante la sentencia for un Hola Mundo Dart.

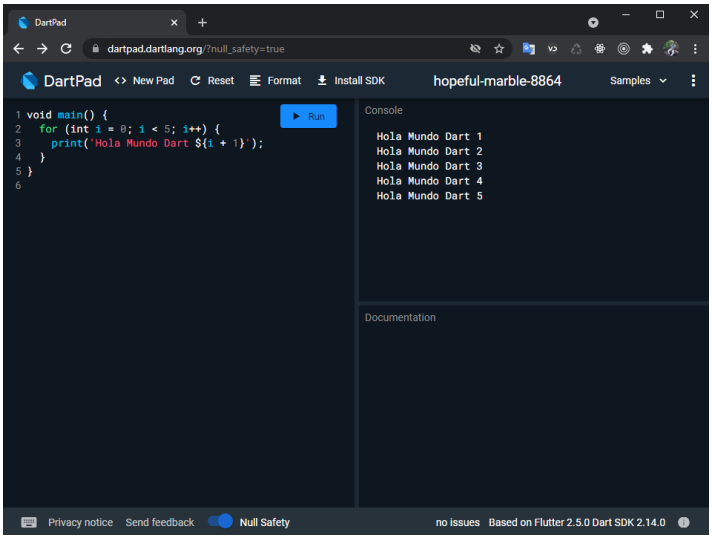

Figura 4 código DartPad

Fuente Erick Vicente Macias Vera

¿Cuáles son sus ventajas y desventajas de Dart?

Ventajas

Es un lenguaje de programación desarrollado por Google de código abierto, debido a su sintaxis este lenguaje es fácil de aprender para los desarrolladores ya que ellos han simplificado muchas características de otros lenguajes como el lenguaje C# y Java no tardaran en familiarizarse con Dart, el mismo que funciona en todos los navegadores móviles y de escritorio actuales.

Desventajas

La única desventaja es que este lenguaje es relativamente nuevo, lo que implica que su comunidad de soporte es bastante baja, pero día a día existen programadores que les incentiva desarrollar múltiples aplicaciones solo es de esperar que crecerá y mejorará a corto plazo. Existen muchos criterios negativos de la existencia de este nuevo lenguaje en el mercado en lugar de perfeccionar los lenguajes ya existentes.

Widgets

El autor (Ferrer, 2020) nos dice se puede considerar a los widgets unos bloques de código reusables que implementan el diseño de la interfaz de usuario (UI). Esto lo hacen a partir del estado de cambio o interacción de los mismos, se determinan en estático es decir que no cambia y dinámico que cambia con la función que se les programe.

Los widgets son construidos usando un moderno framework. La idea principal es la construcción de tu UI de widgets, estos se describen como deberían verse, dada su configuración y estado actual. Cuando el estado cambia, se reconstruye la aplicación para determinar los cambios necesarios en el árbol de renderizacion subyacente para la transición del siguiente estado.

Tipos de Widgets

En la actualidad tienen dos tipos de widgets se clasifican en; Stateless y el Stateful.

Stateless Widgets

Es considerado como los widgets sin estado, no guardan o no cambian, es decir, no contienen valores estáticos que puedan cambiar. Un ejemplo claro, sea un hola mundo, o al crear una aplicación en Flutter nos muestra por defecto una app de un contador.

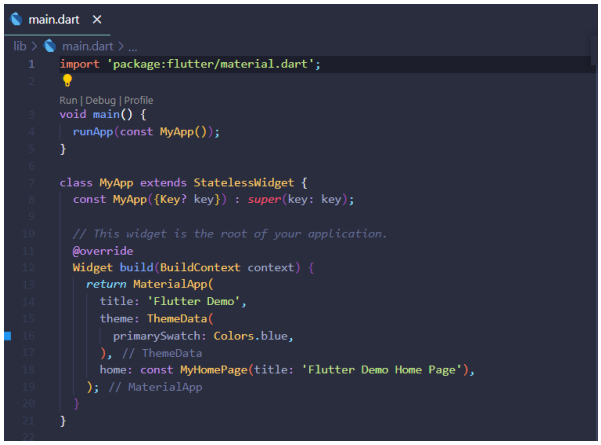

Figura 5 *Código ejemplo Stateless Widget*

Fuente Erick Vicente Macias Vera

Stateful Widgets

Es considerado como los widgets con estado, realizan cambios que se pueden modificar la interfaz de usuario con la función de interactuar en la aplicación. Por ejemplo, los botones son un ejemplo de los Stateful widget.

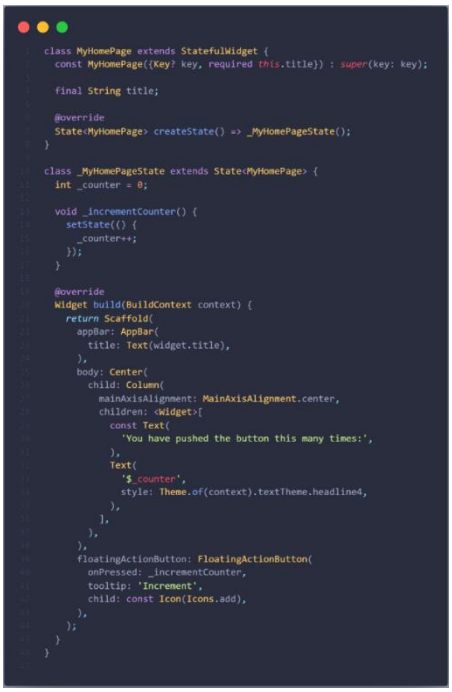

Figura 6 *Código ejemplo Stateful Widget*

Fuente Erick Vicente Macias Vera

Podemos ver la estructura de los widgets empezando desde el Scaffold, siguiendo una estructura de padre-hijo al TabBar, AppBar, el container y los widgets personalizados por el desarrollador.

Scaffold: este widget es el de la estructura principal de la interfaz de usuario.

AppBar: muestra una barra de iconos que interactúan en la navegación como menú ya sea inferior o superior de la aplicación.

Center: un widget de tipo layout, su función es central el contenido del widget padre.

Column: también es de tipo layout, que permite pone a una lista de widgets verticalmente.

Text: muestra un texto en pantalla.

FloatingActionButton: presenta un botón en la pantalla.

Figura 7 *Árbol de Widgets*

Fuente Erick Vicente Macias Vera

React Native

El framework se utiliza para la creación de aplicaciones móviles nativas para las plataformas de Android y IOS basada en el lenguaje de programación de JavaScript con ReactJS usando el paradigma de construir sus bloques UI, creando componentes virtuales, modificando el objetivo de cada componente, en lugar de ejecutarlos en los navegadores, ejecutarlos directamente sobre las plataformas móviles nativas. En pocas palabras, en lugar de implementar una aplicación web híbrida, lo que se obtiene es el resultado de una aplicación nativa, distintas a las que se puede desarrollar con tu código en el lenguaje Java (React Native: ¿Qué es y para que sirve?, 2019).

Figura 7 *React Native* **Autor:** *Raul Fraile*

Fuente: https://www.futurespace.es/react-native-en-apps-multiplataforma-como-primera-opcion

Existen diversas características sobre el framework de React Native que nos proporcionan las siguientes funcionalidades:

Compatibilidad multiplataforma

Las APIs de React Native son multiplataforma, esto ayuda a que los desarrolladores creen sus aplicaciones puedan ser ejecutadas tanto en Android y IOS con la misma base de código.

Funcionamiento nativo

Las aplicaciones desarrolladas con código nativo funcionan de la misma manera que un framework. La unión de React Native con JavaScript permite que su ejecución sea más suave, mejorando el rendimiento de las apps y sin el uso de un WebView.

Actualización instantánea

Con JavaScript los programadores contienen flexibilidad de ejecutar los cambios de contenidos en las actualizaciones directamente al móvil de usuario sin pasar por tiendas de aplicaciones propias y sus molestosos ciclos de procesos obligatorios. Pero esto tiene una desventaja en el desarrollo o al testear la app, es un proceso ilegal, pueden llegar a tener castigos o en la retenida de la aplicación si realizan cambios directos sobre código con las aplicaciones en producción. La compañía Apple lleva un control profundo sobre este tipo de prácticas.

Sencillez en el aprendizaje

El framework React Native es muy fácil de entender, leer y sencillo de aprender o captar los conceptos del lenguaje JavaScript, nos provee un amplio rango de componentes.

Experiencia para el desarrollador

Aquí nos ofrece varias características como el Hot reloading este refresca la app en el momento que se guardan los cambios, tiene una gran ventaja para el desarrollo y el testing de las nuevas versiones, también el uso de la herramienta debugger del navegador Google Chrome, ayudando a la depuración o compilación de código.

Estas son las cinco características fundamentales de React Native más adelante se describirá su funcionamiento y componentes.

Como funciona React Native

React Native "VirtualDOM"

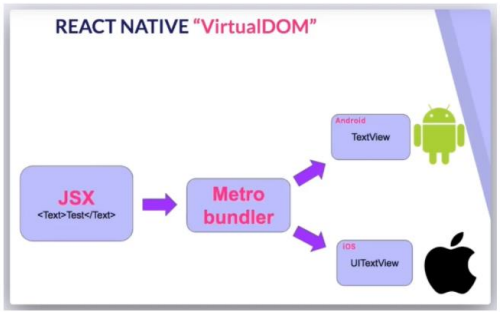

Figura 8 React Native **"**VirtualDOM"

Fuente: https://openwebinars.net/blog/react-native-que-es-para-que-sirve/

Como se observa en la figura, el VirtualDOM es el que tenemos en el apartado de JSX, en este se define los documentos HTML, y esto se cambian en componentes del navegador mediante JavaScript.

Es más, un patrón que una tecnología específica, los desarrolladores a menudo le dan significados diferentes. En el mundo de React, el patrón "VirtualDOM" es asociado con los componentes de React ya que son objetos representando la interfaz de usuario. Sin embargo, React también usa objetos internos llamados "fibers" su principal objetivo es renderizar el incremento del VirtualDOM, para mantener la información del árbol de componentes.

JavaScript

JavaScript es un lenguaje de programación o de secuencias de comandos que te permite implementar funciones complejas en páginas web, cada vez que una página web hace algo más que sentarse allí y mostrar información estática

para que la veas, muestra oportunas actualizaciones de contenido, mapas interactivos, animación de Gráficos 2D/3D, desplazamiento de máquinas reproductoras de vídeo, etc., puedes apostar que probablemente JavaScript está involucrado. Es la tercera capa del pastel de las tecnologías web estándar, dos de las cuales (HTML y CSS) hemos cubierto con mucho más detalle en otras partes del Área de aprendizaje (MDN Web Docs, 2021).

JavaScript es uno de los lenguajes de programación más utilizados y conocidos, ya que este permite crear páginas dinámicas y llamativas en las que se puede interactuar más con los usuarios; como, por ejemplo, gracias a este lenguaje la experiencia visual del usuario más atractiva al momento de actualizar tus estados de Facebook, Instagram, Twitter y otras redes sociales. Cabe resaltar que JavaScript se ejecuta en el ordenador del usuario y actualmente también se ejecuta en el servidor (Barrera, 2021).

Ventajas y Desventajas de JavaScript

Ventajas

- **Velocidad:** JavaScript es muy rápido porque al instante se ejecuta en el navegador. Mientras no requiera recursos externos, no se retrasa por las peticiones del servidor.
- **Simplicidad:** Su sintaxis está inspirada por el lenguaje Java y es sencillo de aprender.
- **Compatibilidad:** Puede ser usado en cualquier página web.
- **Server Load:** la ventaja que JavaScript es client-side, esto reduce la demanda de las peticiones al servidor.

Desventajas

- **Seguridad Client-Side:** cuando el código JavaScript es ejecutado en el client-side, bugs y descuidos pueden ser explotados con malos propósitos.
- **Soporte del navegador:** Mientras que el server-side script produce el mismo valor, los distintos navegadores interpretan el código de manera distinta.

Librerías de Componentes de React Native

Podemos decir que con la popularidad de React y el aumento del desarrollo de aplicaciones móviles nativas, React Native te alienta a construir la interfaz de usuario (UI) usando componente aislados. Los componentes y los toolkit de UI ayudan a reducir el tiempo y construir apps más rápido, usando componentes prefabricados los cuales con:

- **Native Base:** Es una librería de componentes UI, que proporciona muchos componentes de multiplataforma para React Native.
- **Shoutem:** es un kit UI que consta de tres partes: temas, animaciones y componentes UI. La librería tiene un conjunto de componentes multiplataforma para Android y IOS, estos componentes son diseñados para ser componibles y personalizados.
- **React Native Material UI:** librería con un conjunto de componentes UI personalizables, que implementa el material design de Google. La librería usa un único objeto JS llamado uiTheme, que pasa a través de contexto para una máxima personalización.

- **React Native Elements:** un kit de UI multiplataforma personalizable y construido por JavaScript.

Ventajas y Desventajas de React Native

Ventajas

- **Alto rendimiento:** Esta tecnología es útil y nos ayuda a mejor el rendimiento de las apps con control nativo.
- **Actualizaciones más rápidas:** agiliza el proceso de las actualizaciones. Reduce el tiempo drástico de la para publicar la aplicación.
- **Arquitectura modular:** esta función aprueba la agregación de funciones de diferentes componentes intercambiables estos módulos son reutilizables de forma a la API móvil y web.
- **Cambios inmediatos:** se puede decir que la recarga en vivo o Hot Reload de React Native permite a los desarrolladores ver al instante el cambio en la interfaz.

Desventajas

- Necesita desarrolladores Nativos.
- No tiene muchos componentes propios sino librerías externas.
- Está en la versión beta.
- Los mensajes de error no son muy claros.

Comparación y similitudes rápida de los Frameworks Flutter y React Native

Similitudes de Flutter y React Native

Comparando los dos Frameworks ampliamente. Conoceremos la igualdad de ambos marcos.

- Los dos son multiplataforma.
- Son de código abierto.

- Recarga activa
- Desarrollo Nativo.

Comparación de Flutter y React Native

Al pasar de los años los desarrolladores se están adaptando a estas tecnologías. La demanda en el desarrollo de estos frameworks va aumentando la curva de aprendizaje. Para estar actualizados hacia la necesidad de los clientes, el desarrollo de las nuevas plataformas se convierte en una necesidad. Las mismas que nos permiten interactuar con múltiples dispositivos.

Al implementar una aplicación móvil, para el actual modelo de negocios. Los desarrolladores necesitan una tecnología que sea adaptable y robusta que simplifique el tiempo y genere eficiencia en la codificación. Para cumplir este objetivo, muchos programadores están usando las aplicaciones multiplataforma. Porque pueden aprovechar todas sus características para diseñar aplicaciones e-commerce, apps interactivas y apps sociales. (Simões, 2019)

Podemos observar en la **Figura 10** el tiempo que se lleva cada tecnología en la creación del proyecto. React Native lleva mucho tiempo hasta que se ejecute el comando **npx React-native init** y el nombre del proyecto, se lleva 10 minutos hasta crear todos los paquetes dentro de la carpeta, terminado este proceso se levanta ya sea un emulador o un dispositivo físico ejecutando el comando **npx react-native run-android** desde la PowerShell, este levanta el node.js y lanza la app al dispositivo móvil. Mientras que en la **Figura 11** Flutter el tiempo que lleva en crear los paquetes dentro de la carpeta es de 6.7 segundos ejecutando desde visual studio code **ctrl + shift + p** se elige el comando **Flutter: New Application Project**. Por lo tanto, Flutter tiene la ventaja en el entorno del soporte.

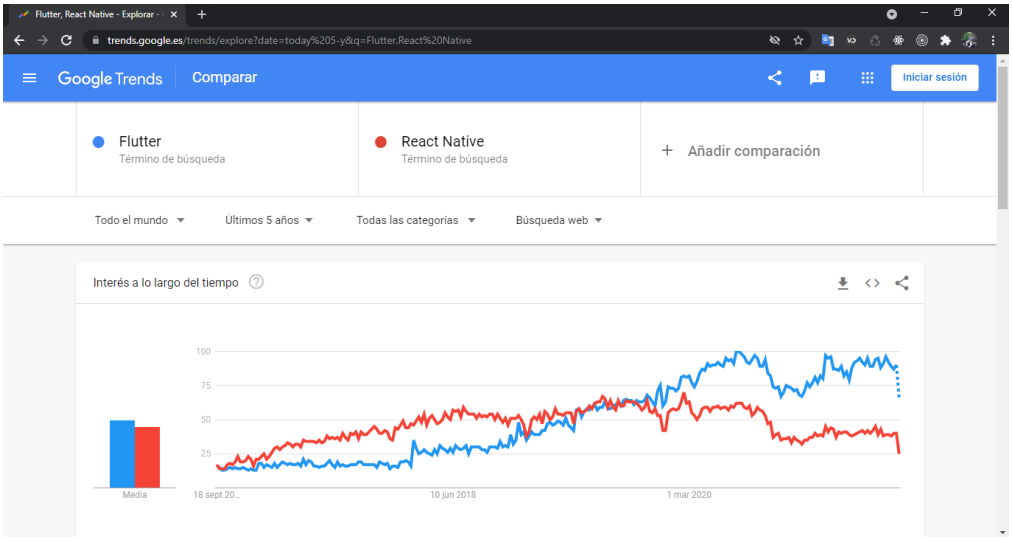

Figura 9 *Comparación de Flutter & React Native (Google Trends)*

Fuente:https://trends.google.es/trends/explore?date=today%205-y&q=Flutter,React%20Native

Podemos observar en la **Figura 9** se realizó mediante Google Trends sobre Flutter y React Native una comparación de los mismo en los últimos 5 años hasta la actualidad nos muestra como estas dos tecnologías están en constante competencia, pero los desarrolladores prefieren más al Framework Flutter, el mismo en la actualidad está en versión estable.

Interfaz de Usuario

Al desarrollar la interfaz de usuario de la aplicación y crear los componentes de mismo existe una gran brecha entre los dos Frameworks. Flutter funciona con su árbol o conjuntos de widgets, el widget personalizado de un usuario es más interesante y es de lo mejor al obtener una interfaz de usuario proporcionando dinamismo y soporte nativo, mientras que React Native se basa en componentes nativos, se puede decir, existen inconvenientes en la interacción de componentes y puede dar problemas en la experiencia de usuario. Observando en la **Figura 9** Flutter tiene una gran ventaja por su interacción y animaciones entre los widgets mejorando la experiencia de usuario.

Documentación y Kit de Desarrollo

El proceso de la documentación en la experiencia de cada desarrollador es algo muy complejo y lento, por lo tanto, Flutter es bueno para facilitar su documentación desde su página oficial flutter.dev/docs, facilitando en el entorno de desarrollo de las aplicaciones móviles esto respecta a su documentación y sus extensas herramientas, Flutter es el fuerte aquí.

La documentación de React Native no es tan organizada se la encuentra en su página oficial devdocs.io/react_native/, además el framework elimina los componentes útiles para facilitar los procesos, se evidencia la dura competencia que tiene estas dos tecnologías. El framework Flutter tiene IDE y herramientas extensas, compatibilidad con Android Studio y Visual Studio Code.

¿Cuál es su rendimiento?

Mostrando los ejemplos que se creó se puede dar una breve comparación al momento de la creación de las aplicaciones nativas de Flutter y React Native veremos el rendimiento de ambas plataformas.

Flutter	El framework es útil para los desarrolladores que crean apps animadas muy pesadas, su rendimiento es superior en comparación a otras plataformas.
React Native	Este no es el adecuado para operaciones con uso el uso intenso de la CPU, el mismo que al crear su app tiene un tiempo de 10 minutos y el uso de CPU alta. Es mejor que se elija Flutter para apps que requieran más memoria y CPU.

Tabla 2 Comparación rendimiento de Flutter y React Native

Fuente: Erick Vicente Macias Vera

¿Cuál tecnología es fácil de aprender?

Todo depende de la experiencia que el desarrollador tenga con JavaScript.

El Framework Flutter debido a su reciente lanzamiento se puede decir que su tecnología casi no la conocen, su lenguaje Dart. Es de fácil aprendizaje, lo mismo con Flutter de fácil aprendizaje, ya que su documentación es sencilla y si números de recursos gratuitos, incluyendo artículos, videos, código de interfaces UI en GitHub.

En React Native si tiene un nivel alto de JavaScript este es tu fuerte, su acceso es sencillo y lo más importante reutiliza el mismo código al igual que Flutter ofrece documentación amplia y videos de aprendizaje.

Ahora podemos decir con la experiencia previa del desarrollador con JavaScript su curva de aprendizaje será similar y un poco más favorable para Flutter.

¿Diferencias al publicar en las tiendas de las aplicaciones?

Existe el mismo proceso para ambas tecnologías. La diferencia se basa en el proceso de la publicación tanto en Google Play como en Apple Store.

Apple Store

Esta tienda requiere a los usuarios paguen una tarifa anual de $100 por cada desarrollador solo así puede distribuir sus productos móviles y formar parte de un programa de desarrolladores Apple. Los miembros de este programa tienen el beneficio al acceso a varias tecnologías modernas para la creación de aplicaciones atractivas estos beneficios son: Apple Pay, Apple Maps, HomeKit y HealthKit. La misma empresa proporcionara a los programadores los SDK y APIs para crear interfaces de usuario atractivas.

Google Play Store

Se necesita una tarifa de $25 para el registro para obtener la cuenta de Google Publisher Account. Su diferencia es que el usuario solo tiene que pagar este costo una sola vez, no como la tienda de Apple que le pide a sus usuarios pagar su tarifa anualmente.

La siguiente tabla nos proporciona la descripción general de ambos frameworks.

Tecnologías	Framework Flutter	Framework React Native
Creado por	Google Chrome	Facebook
Fecha de lanzamiento	Mayo de 2017	Marzo de 2015
Lenguaje de Programación	Dart	JavaScript
Curva de aprendizaje	Si conoces C/C++ y Java es Ligeramente de aprender.	Si ya conoces JavaScript y React, es fácil de aprender
Biblioteca de componentes	En la actualidad muy Amplia (Widgets)	Muy grande (por desarrolladores externos)
Componentes adaptivos	Soporte para Android y IOS	Soporte para Android y IOS
Arquitectura principal	Bloc	Redux
Hot Reload	Activa	Activa
Precio	Open Source	Open Source

Tabla 3 Comparación rápida de Flutter y React Native

Fuente: Erick Vicente Macias Vera

CONCLUSIONES

Concluido el análisis comparativo de los frameworks nativos Flutter y React Native nos da información sobre el potencial de ambos, se ha enumerado una comparación detallada para saber lo populares que son entre las dos empresas Google y Facebook para el desarrollo de las aplicaciones nativas móviles. Ambos tienen puntos positivos como diferencias, ventajas, desventajas, sus lenguajes, documentación y otros aspectos como lo hemos mencionado.

Luego de realizar una comparación en una herramienta llamada Google Trends es una herramienta que permite seguir la evolución del número de búsquedas por una determinada palabra clave, podemos observar en la figura 9 la comparación de los últimos 5 años no era tan reconocido como es ahora en la actualidad. Puedo decir que Flutter con sus nuevas actualizaciones desde la versión 2.0 puede realizar aplicaciones hibridas (móvil, web, desktop) con la misma base de código, ofrece más soporte, su documentación es estable, ofrece muchas funcionalidades en el ámbito del desarrollo de las aplicaciones móviles. Además, React Native exige estilos específicos para que la interfaz de usuario funcione a la perfección en distintas plataformas, otra acotación es al diseñar la interfaz si por alguna ocasión se ocasiono un error no da especificaciones concretas para poder solucionarlo en si se debe programar casi a la perfección y se debe saber lo que se va a implementar.

El desarrollo de esta investigación sirve de gran ayuda hacia nuestro perfil profesional de todo Ingeniero, permitiendo investigar, formular propuestas, liderar y analizar las formas de desempeño de aplicaciones móviles nativas sean nuevas o ya reconocida por muchos desarrolladores, teniendo la facilidad y agilidad en estos entornos.

Bibliografía

Barrera, A. (2021). *CONOCE LAS VENTAJAS Y DESVENTAJAS DE JAVASCRIPT*. Obtenido de next_u: https://www.nextu.com/blog/conoce-las-ventajas-y-desventajas-de-javascript/

Delía, L. N., Galdámez, N., Thomas, P. J., & Pesado, P. M. (17 de Diciembre de 2013). *Un análisis experimental de tipo de aplicaciones para dispositivos móviles.* Obtenido de XVIII Congreso Argentino de Ciencias de la Computación: http://sedici.unlp.edu.ar/handle/10915/32397

Devs, Q. (05 de Julio de 2019). *Qualitydevs.com*. Obtenido de Qué es Flutter y por qué utilizarlo en la creación de tus apps: https://www.qualitydevs.com/2019/07/05/que-es-flutter/

Digital Guide IONOS. (15 de 10 de 2020). *Dart de Google: Una introducción al lenguaje Dart*. Obtenido de Desarrollo web: https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/lenguaje-de-programacion-dart-de-google/

Ferrer, R. (11 de Enero de 2020). *Conoce cómo se construye en Flutter: los Widgets*. Obtenido de Raulferrergarcia.com: https://www.raulferrergarcia.com/conoce-como-se-construye-en-flutter-los-widgets/

MDN Web Docs. (10 de Septiembre de 2021). *¿Qué es JavaScript?* Obtenido de MDN Web Docs: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript

React Native: ¿Qué es y para que sirve? (18 de Junio de 2019). Recuperado el 24 de Septiembre de 2021, de OpenWebinars.net: https://openwebinars.net/blog/react-native-que-es-para-que-sirve/

Simões, C. (18 de Junio de 2019). *React Native vs Flutter. ¿Cuál es mejor para mi producto?* Obtenido de Itdo.com: https://www.itdo.com/blog/react-native-vs-flutter-cual-es-mejor-para-mi-producto/

Triguero, D. (19 de Octubre de 2017). *Aplicaciones móviles híbridas: la solución más eficiente para el desarrollo multiplataforma*. Obtenido de Profile.es: https://profile.es/blog/aplicaciones-moviles-hibridas-la-solucion-mas-eficiente-para-el-desarrollo-multiplataforma/

ANEXOS

Anexo 1

Guía de observación

No.	Aspectos a Evaluar	Si	No	Observaciones
1	Ambos frameworks son para crear apps móviles de forma rápida	**X**		Se pudo crear tanto en Flutter como en React Native
2	Este estudio se desarrolló con la finalidad de apoyar a las personas que están aprendiendo	**X**		Se desarrolló para puedan ver sus novedades actuales y así elegir bien estas metodologías
3	Los dos frameworks son open source	**X**		Su licencia es accesible para cualquier usuario.
4	Se describió lo más importante de estas herramientas	**X**		Se dio conocer lo más relevante de estas dos tecnologías.
5	Se dio conocer el rendimiento de estas dos tecnologías	**X**		En ambos se describió lo que consumen en la creación de sus paquetes
6	Se describió su material o estructura	**X**		Tanto en Flutter y React Native se explicó su estructura, sus widgets y componentes

Anexo 2

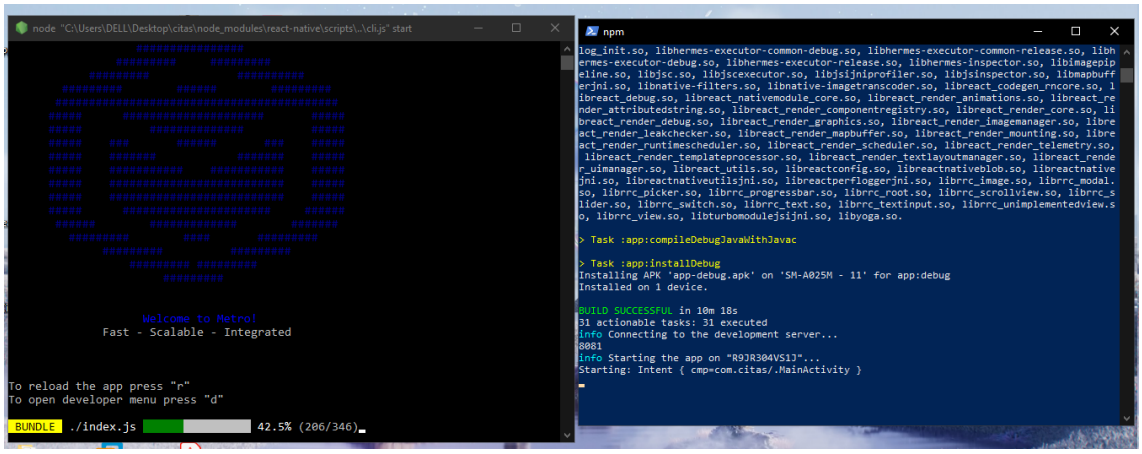

Figura 10 *Tiempo de creación de una Aplicación React Native*

Autor: Erick Vicente Macias Vera

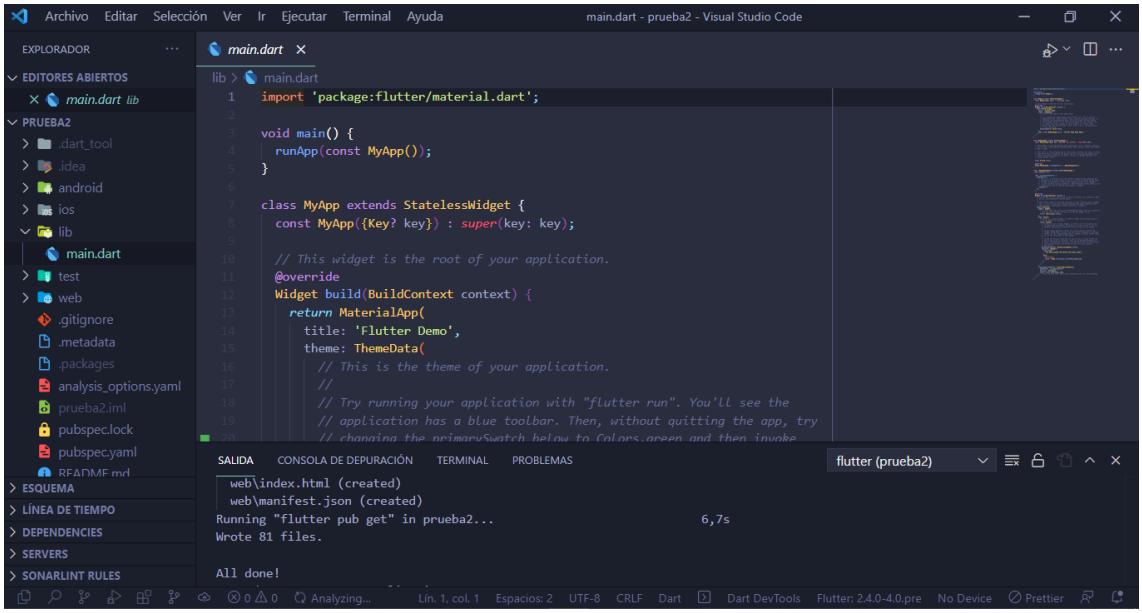

Figura 11 *Tiempo de creación de una Aplicación Flutter*

Autor: Erick Vicente Macias Vera

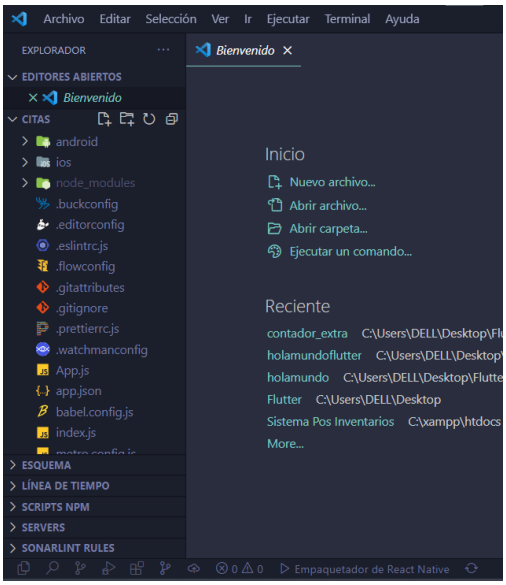

Figura 12 *Entorno de una Aplicación React Native*

Autor: Erick Vicente Macias Vera

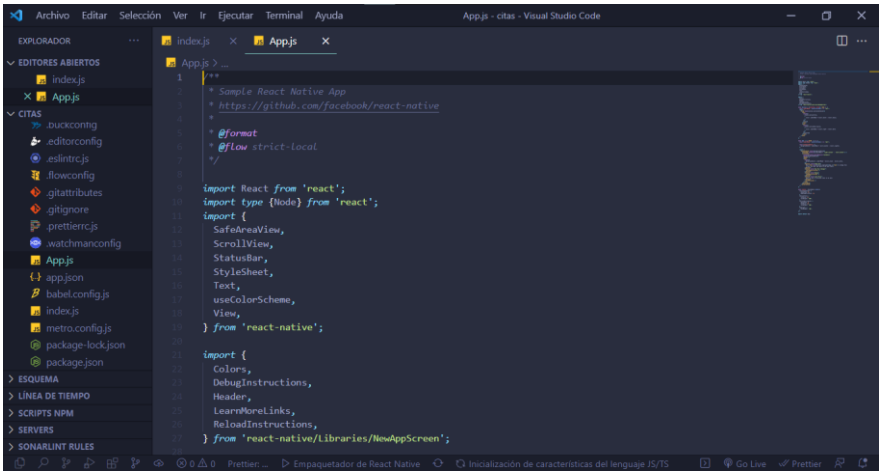

Figura 13 *Clases de una Aplicación React Native*

Autor: Erick Vicente Macias Vera

Figura 14 *Interfaz de la Aplicación Flutter*

Autor: Erick Vicente Macias Vera

Figura 15 *Interfaz personalizada de la Aplicación Flutter*

Autor: Erick Vicente Macias Vera

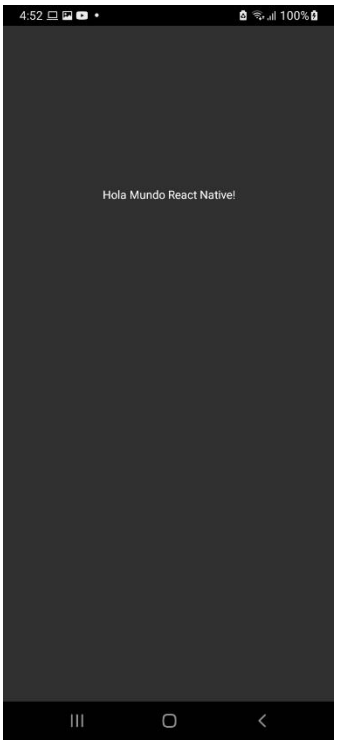

Figura 16 *Interfaz de la Aplicación React Native*

Autor: Erick Vicente Macias Vera