UNIVERSIDAD POLITÉCNICA DE SINALOA
PROGRAMA ACADÉMICO DE INGENIERÍA EN INFORMÁTICA

“Implementación de Flutter para el desarrollo de aplicaciones móviles nativas en iOS y Android”

Para cumplir la acreditación de las estadías profesionales y contar con los créditos necesarios para obtener el grado de Ingeniero en Informática

Autor:
Kevin Antonio Lizárraga Osuna

Asesor:
Doctor Luis Javier Mena Camare

Asesor OR:
Ing. Miguel Ángel Basilio de la Paz

Mazatlán, Sinaloa 9 de diciembre de 2020

UNIVERSIDAD POLITÉCNICA DE SINALOA

SINALOA
SECRETARÍA DE EDUCACIÓN PÚBLICA Y CULTURA

C. LIZARRAGA OSUNA KEVIN ANTONIO **Folio 2016030221-2020-A031**

Presente.-

Por medio de la presente me permito comunicarle que **es de aceptarse el tema de tesina**, el cuál se ha solicitado bajo el título:

"Implementación de Flutter para el desarrollo de aplicaciones móviles nativas en iOS y Android"

mismo que usted desarrollará con objeto de dar lugar a los tramites conducentes para la acreditación de la asignatura de Estadías Profesionales de la Unidad Académica de:

Ingeniería en Informática

Así mismo. Le comunico que para el desarrollo de la citada tesina le ha sido asignado como director de la misma a:Dr. Luís Javier Mena Camaré, y como asesores a **Dr. Rodolfo Ostos Robles** y **M.C. Alejandro Pérez Pasten Borja**.

Sin otro particular por el momento, aprovecho la ocasión para enviarle un cordial saludo.

Atentamente

Dr. Rodolfo Ostos Robles
Director del Programa Académico de Ingeniería en Informática
Universidad Politécnica de Sinaloa

DIRECCIÓN INFORMATICA

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

"2020, Año de Leona Vicario, Benemérita Madre de la Patria".

UTP

Carretera Municipal Libre Mazatlán Higueras Km. 3, Col. Genaro Estrada. C.P. 82199. Mazatlán, Sin. Tel (669) 1800695 y 96
www.upsin.edu.mx

UNIVERSIDAD POLITÉCNICA DE SINALOA

SINALOA
SECRETARÍA DE EDUCACIÓN PÚBLICA Y CULTURA

C. LIZARRAGA OSUNA KEVIN ANTONIO **Folio 2016030221-2020-A031**
Presente.-

Por este conducto le envío un cordial saludo y aprovecho la ocasión para notificarle que el jurado que le fue asignado para evaluar la tesina desarrollada en las estadías profesionales denominada ***"Implementación de Flutter para el desarrollo de aplicaciones móviles nativas en iOS y Android"*** y que después de ser revisada en reunión de sinodales, ante la Dirección de la Unidad Académica de Ingeniería en Informática, integrada por:

PRESIDENTE DEL JURADO: Dr. Luis Javier Mena Camaré

SINODAL: Dr. Rodolfo Ostos Robles

SINODAL: M.C. Alejandro Pérez Pasten Borja

Ha decidido autorizar y aceptar la digitalización de la misma por el participante, conforme a la normatividad vigente y cumpliendo con los requisitos para tal caso.

Agradeciendo la atención a la presente, le reitero a Usted mi atenta consideración y respeto.

Atentamente

Dr. Rodolfo Ostos Robles
Director del Programa Académico de Ingeniería en Informática
Universidad Politécnica de Sinaloa

DIRECCIÓN INFORMÁTICA

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

"2020, Año de Leona Vicario, Benemérita Madre de la Patria".

UTP

Guadalajara, Jalisco, a 07 de septiembre de 2020

LIC. EDUARDO CRESPO CAMPA
DIRECTOR DE VINCULACIÓN, DIFUSIÓN Y EXT. UNIVERSITARIA
UNIVERSIDAD POLITÉCNICA DE SINALOA

PRESENTE

Por este medio, hago de su conocimiento que el alumno el C. Kevin Antonio Lizárraga Osuna, con número de matrícula 2016030221, de la carrera de Ingeniería en Informática, ha sido aceptado para realizar su estadía práctica, en esta empresa, durante el período que comprende del 7 de septiembre al 4 de diciembre, para cubrir un total de 600 horas.

Dicho alumno realizará actividades dentro del Área de Tecnología, bajo la supervisión del Ing. Miguel Basilio de la Paz, Director de Tecnología.

Sin otro particular, le envío un cordial saludo.

Nextline S.C.
Av. Circunvalación Jorge Álvarez del Castillo #1471,
Lomas del Country, C.P: 44610, Guadalajara, Jalisco.
contacto@nextline.mx
Tel. (33) 15 62 66 31
www.nextline.mx

Atte.

Ing. Miguel Angel Basilio De la Paz
Director de Tecnología

nextline®

Guadalajara, Jalisco, a 04 de diciembre de 2020

LIC. EDUARDO CRESPO CAMPA
DIRECTOR DE VINCULACIÓN, DIFUSIÓN Y EXT. UNIVERSITARIA
UNIVERSIDAD POLITÉCNICA DE SINALOA

PRESENTE

Por este medio, hago de su conocimiento que el alumno el C. Kevin Antonio Lizárraga Osuna, con número de matrícula 2016030221, de la carrera de Ingeniería en Informática, ha cumplido con 600 horas correspondientes a estadía final, en esta empresa, durante el período que comprende del 07 de septiembre al 04 de diciembre.

Dicho alumno realizó actividades dentro del Área de Tecnología, bajo la supervisión del Ing. Miguel Basilio de la Paz, Director de Tecnología.

Sin otro particular, le envío un cordial saludo.

Nextline S.C.
Av. Circunvalación Jorge Álvarez del Castillo #1471,
Lomas del Country, C.P: 44610, Guadalajara, Jalisco.
contacto@nextline.mx
Tel. (33) 15 62 66 31
www.nextline.mx

Atte.

Ing. Miguel Angel Basilio De la Paz
Director de Tecnología

Agradecimientos

Quero agradecer principalmente a mi familia, ya que son ellos quiénes me han apoyado personal y económicamente en mis estudios y gracias a su motivación y apoyo me ha sido posible llegar hasta aquí.

Asímismo, agradezco a la Universidad Politécnica de Sinaloa por darme la oportunidad de estudiar lo que me gusta y por las oportunidades que me ha otrogado, y agradezco también a los docentes por las valiosas enseñanzas que me dejaron.

Resumen

Debido al auge de las aplicaciones móviles, cada vez es más exigente el tiempo de desarrollo y abarcar el mayor porcentaje del mercado. Debido a esto surge la necesidad de utilizar frameworks para el desarrollo de aplicaciones híbridas, ya que se disminuye el tiempo de codificación y se obtiene un mayor valor al poder compilar para los sistemas operativos deseados. En el presente trabajo, se propone el uso del framework Flutter para el desarrollo de una aplicación para Android y se destacan las ventajas de su uso durante el proceso de codificación de la aplicación.

Palabras clave: Framework, Flutter, Android

Abstract

Due to the rise of mobile applications, development time has become more demanding and it has to cover the largest percentage of the market. Due to this, the need arises to use frameworks for the development of hybrid applications, since coding time is reduced and greater value is obtained by being able to compile for the desired operating systems. In this work, the use of the Flutter framework is proposed for the development of an Android application and the advantages of its use during the application coding process are highlighted.

Keywords: Framework, Flutter, Android

Índice General

Lista de Imágenes

Lista de Figuras

Lista de Tablas

Capítulo 1

Antecedentes y planteamiento del problema

Introducción

El mercado de las aplicaciones móviles se encuentra en auge, cada vez es más relevante para cualquier servicio el tener presencia en el mercado de las aplicaciones móviles, ya que de esta manera se tiene un mayor alcance a posibles usuarios y aumenta la visibilidad de dicho servicio.

Esto mismo ha ocasionado que la importancia de tener una aplicación móvil en este momento aumente, y esta misma necesidad se refleja en la demanda de programadores móvil y en los tiempos de desarrollo cada vez menores para dichos proyectos, lo que lleva consigo la necesidad de desarrollar e implementar nuevas tecnologías para lograr cumplir las necesidades del mercado.

El principal aspecto que complica los tiempos de desarrollo de las aplicaciones móviles es la presencia de dos sistemas operativos móviles en el mercado: iOS y Android. Siendo que cada uno de éstos tiene su propio lenguaje de programación para codificar una aplicación para ellos, lo que genera la problemática de tener que desarrollar dos aplicaciones para un mismo servicio.

Este problema ha ocasionado el surgimiento de frameworks para el desarrollo de aplicaciones híbridas, es decir, aplicaciones en las que con un mismo código fuente se pueden generar las aplicaciones para los distintos sistemas operativos. Siendo frameworks jóvenes, cada vez tienen una mayor recepción y su soporte y capacidades han estado en aumento.

En este momento, estos frameworks ya se encuentran lo suficientemente maduros como para ser utilizados en proyectos de diferentes ámbitos, logrando una mejora en el tiempo de desarrollo de una aplicación móvil sin comprometer la eficiencia y la funcionalidad de la misma.

En el presente trabajo de investigación, se tratará la factibilidad de emplear un framework de desarrollo híbrido en conjunto con metodologías ágiles para el desarrollo de una aplicación, y se mencionarán las facilidades que proporciona dicho framework con respecto a una programación nativa.

1.1 Antecedentes

“Nextline es una empresa mexicana que ofrece soluciones inteligentes para promover el desarrollo económico y el crecimiento sustentable de una región. A partir de fomentar la empleabilidad, desarrollar al talento y promover la atracción de inversión al integrar en su Plataforma de Desarrollo Económico la participación de la cuarta hélice” [1].

Nextline surgió a inicios del 2014 a razón de conocer a profundidad y mediante experiencia propia, la brecha que separa al mundo laboral de los recién egresados y universitarios.

Su propuesta de valor nació al conformarse como una red profesional dinámica que colabora en resolver la presente problemática, mediante la vinculación estratégica entre la oferta laboral capacitada y la creciente demanda en las empresas para encontrar a los perfiles indicados.

“Actualmente, en Nextline se incentiva la formación de la denominada cuarta hélice, donde el sector de la población se incorpora como un eje estratégico en el desarrollo económico de la región al colaborar de manera directa con el Gobierno, Academia e Industria, mediante su incorporación como talento calificado” [2].

1.1.1 Ubicación

Nextline cuenta con unas oficinas ubicadas en Plaza Country Club, en Avenida Circunvalación Jorge Álvarez del Castillo #1471 CP. 44610, en el municipio de Guadalajara, estado de Jalisco.

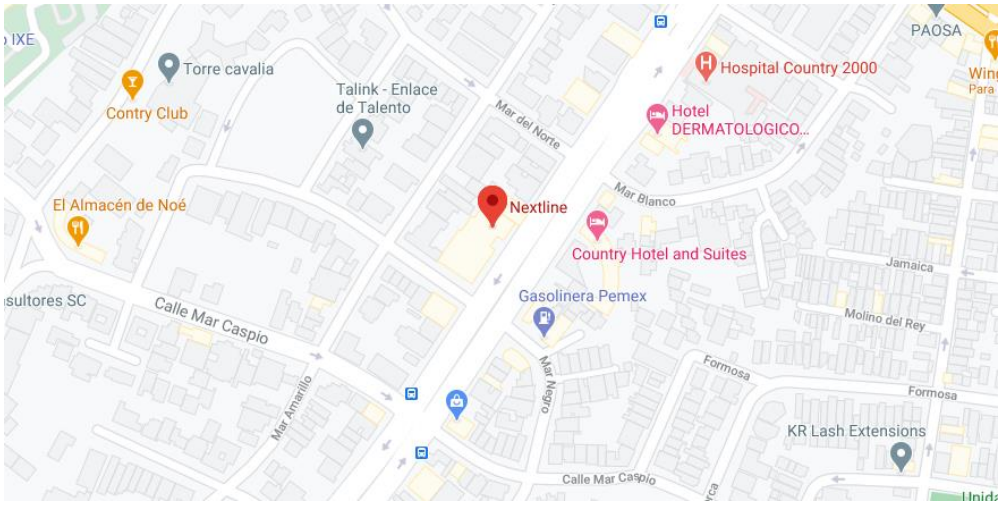

Imagen 1
Ubicación de Nextline

1.1.2 Organigrama

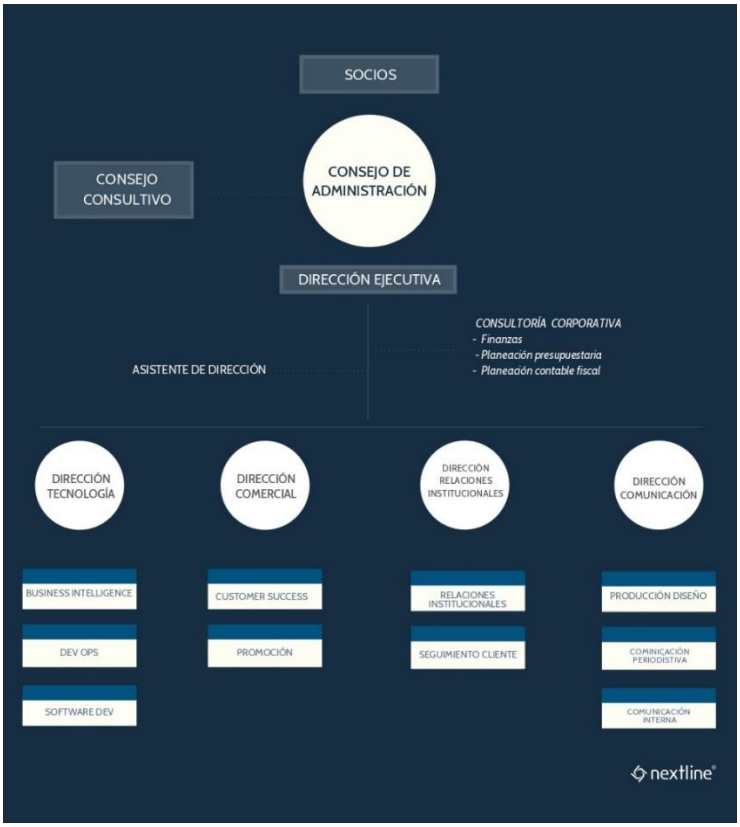

Imagen 2
Organigrama de Nextline [2]

1.1.3 Visión

“Nuestra visión es generar alianzas estratégicas por medio de la implementación de soluciones óptimas que coadyuven a mejorar la empleabilidad a nivel nacional en el año 2025.” [2].

1.1.4 Misión

“Nuestra misión es crear una conexión estratégica a través de una herramienta inteligente entre la población, academia, industria y Gobierno e impulsar la colocación efectiva del talento para contribuir con el desarrollo económico.” [2].

1.1.5 Pilares

Los valores de Nextline surgen como principios que guían a la institución en el alcance de su misióno.

Ante la sociedad:

Colaboramos con pasión. La pasión involucra el florecimiento de competencias personales con compromiso y dedicación por generar algo valioso que resalte la identidad propia. Por ello, como equipo nos une la pasión por crear soluciones que inspiren y mejoren las condiciones sociales en el entorno.

Cooperación y fortaleza de las alianzas. El aprendizaje mediante la cooperación forma un desarrollo integral que fortalece los lazos de las relaciones humanas y su crecimiento social. En Nextline, transformamos nuestros esfuerzos en lazos de confianza para conectar e incentivar la participación de la denominada “cuarta hélice” como un desarrollo inteligente.

Integridad como núcleo de acción. Creemos que lo vínculos fuertes se forman gracias a la entereza moral en todas las acciones y decisiones. Solidificamos nuestra forma de colaborar con responsabilidad social, conciencia y transparencia en dirección a vivir cada relación, cada proyecto con integridad y respeto.

Intraempresarial

Mejora continua e innovación. La confianza es la llave que abre el progreso, incluso si se desconocen los resultados. Al confiar en nuestro talento, colaboramos en la

búsqueda de soluciones que superen de manera creativa nuestros logros e involucren nuestro pacto con la comunidad y razón de ser.

Enfoque y eficiencia. Tenemos el compromiso de crear un espacio hacia la mejora de oportunidades laborales de todos. Aunado a esto, nuestro enfoque es actuar con responsabilidad y consciencia al priorizar actividades y movernos con rapidez para agilizar el tiempo de respuesta ante cualquier solicitud.

De servicio

Accesibilidad, bienestar e inclusión. Reconocemos la importancia de la participación de cada eslabón como una sociedad multicultural. Nuestro incentivo es brindar el acceso práctico e inclusivo al eje laboral con la finalidad de activar las capacidades de crecimiento colectivo y empoderar a la gente, así como respetar la diversidad cultural.

Actitud positiva y respeto. Promover una competitividad sana se logra por la cohesión en las metas y buena actitud. Nuestra ruta y esencia corporativa es la amabilidad, honestidad, trato digno y respeto al ser un equipo horizontal que se esfuerza por honrar a las personas y fomentar una actitud de excelencia y profesionalismo.

Acompañamiento y seguridad. La forma en la que vemos al mundo es a través de los ojos de nuestros clientes. Por ello, brindamos atención personalizada, capacitación y acompañamiento en el proceso a modo de oportunidad de crecimiento e impulso para mejorar nuestras soluciones de acuerdo a las necesidades requeridas [3].

1.2 Planteamiento del Problema

Actualmente, tener presencia en el mercado de las aplicaciones móviles resulta crucial para cualquier empresa, ya que resulta en una mayor accesibilidad para los potenciales usuarios y facilita el uso de cualquier servicio, sin embargo, el desarrollo de una aplicación móvil requiere de preparación y de tener bien definido que plataformas se quieren abarcar, ya que en esta cuestión radica la principal problemática del desarrollo móvil.

Al desarrollar una aplicación móvil, se encuentra el factor del sistema operativo, típicamente Android e iOS, así como las diferentes versiones y los diferentes

dispositivos, hechos que aumentan la complejidad del desarrollo, especialmente si se quiere tener la aplicación para ambas plataformas. Combinando esto con el creciente uso de metodologías ágiles para el rápido desarrollo de software, se requiere el uso de herramientas que agilicen el desarrollo multiplataforma.

1.3 Hipótesis

Utilizar un framework de desarrollo móvil híbrido permitirá agilizar el desarrollo de una aplicación móvil multiplataforma, y cumplir con todos los objetivos de la misma sin impactar el desempeño o la experiencia de usuario en su uso.

1.4 Objetivos

A partir del planteamiento del problema y su hipótesis, se estableció el siguiente objetivo general y objetivos específicos.

1.4.1 Objetivo general

Desarrollar una aplicación móvil para los sistemas operativos con mayor uso, Android y iOS, utilizando un mismo código fuente con desempeño casi nativo en un periodo breve de tiempo siguiendo metodologías ágiles.

1.4.2 Objetivos específicos

- Implementar pruebas del software.
- Desarrollo ágil.
- Control de versiones y publicación en la tienda de aplicaciones.

1.5 Justificación

Desarrollar una aplicación móvil permite a cualquier empresa alcanzar una mayor cantidad de usuarios potenciales gracias a la accesibilidad que las plataformas móviles facilitan en comparación con un sitio web. El desarrollar una aplicación para los dos sistemas operativos móviles más relevantes permite alcanzar el mayor número posible de usuarios.

Además, utilizar tecnologías o herramientas actuales permiten un desarrollo más ágil, al tener presentes facilidades que permitan la reducción de líneas código y de configuración, para poder enfocarse únicamente en el desarrollo.

1.6 Limitaciones del Estudio

El presente estudio considera únicamente el desarrollo utilizando Flutter, React Native o Ionic, los cuales permiten el desarrollo de aplicaciones móviles nativas con el mismo codigo fuente.

La metodología ágil planteada es Scrum, con el uso de sus prácticas y artefactos correspondientes.

Aunque se plantea el desarrollo para ambas plataformas, el estudio se encuentra enfocado en Android, contemplando una compatibilidad con la versión 6 de Android en adelante.

1.7 Definición de términos

Una app móvil es una aplicación de software pensada para dispositivos móviles y tabletas. El término *app* es una abreviatura de la voz inglesa *application* y tiende a utilizarse para referirse a una aplicación informática para dispositivos móviles y tabletas.

Las aplicaciones móviles híbridas son una combinación de tecnologías web como HTML, CSS y JavaScript, que no son ni aplicaciones móviles verdaderamente nativas, porque consisten en un WebView ejecutado dentro de un contenedor nativo, ni tampoco están basadas en Web, porque se empaquetan como aplicaciones para distribución y tienen acceso a las APIs nativas del dispositivo.

Las Apps nativas son aquellas aplicaciones que están desarrolladas para un equipo o plataforma determinada. Es decir, funciona en el equipo sin necesidad de ningún programa externo ya que se ha desarrollado en el lenguaje de programación específico de cada equipo. El término de App Nativa está habitualmente asociado a los dispositivos móviles y por tanto hay Apps Nativas para cada sistema operativo como iOS o Android [4].

Capítulo 2

Marco referencial y marco teórico

2.1 Sistemas operativos móviles

Un sistema operativo móvil es un conjunto de programas de bajo nivel que permiten la abstracción de las peculiaridades del hardware específico. A diferencia de los sistemas operativos de escritorio, éstos se encuentran más orientados a la conectividad inalámbrica, formatos multimedia y las diferentes maneras de introducir información en ellos.

Existen diferentes sistemas operativos para dispositivos móviles, pero actualmente el mercado se encuentra dividido en su mayoría en dos: Android y iOS, teniendo una presencia en el mercado durante el periodo de octubre 2019 y octubre 2020 del 73.86% y del 25.35% respectivamente [5].

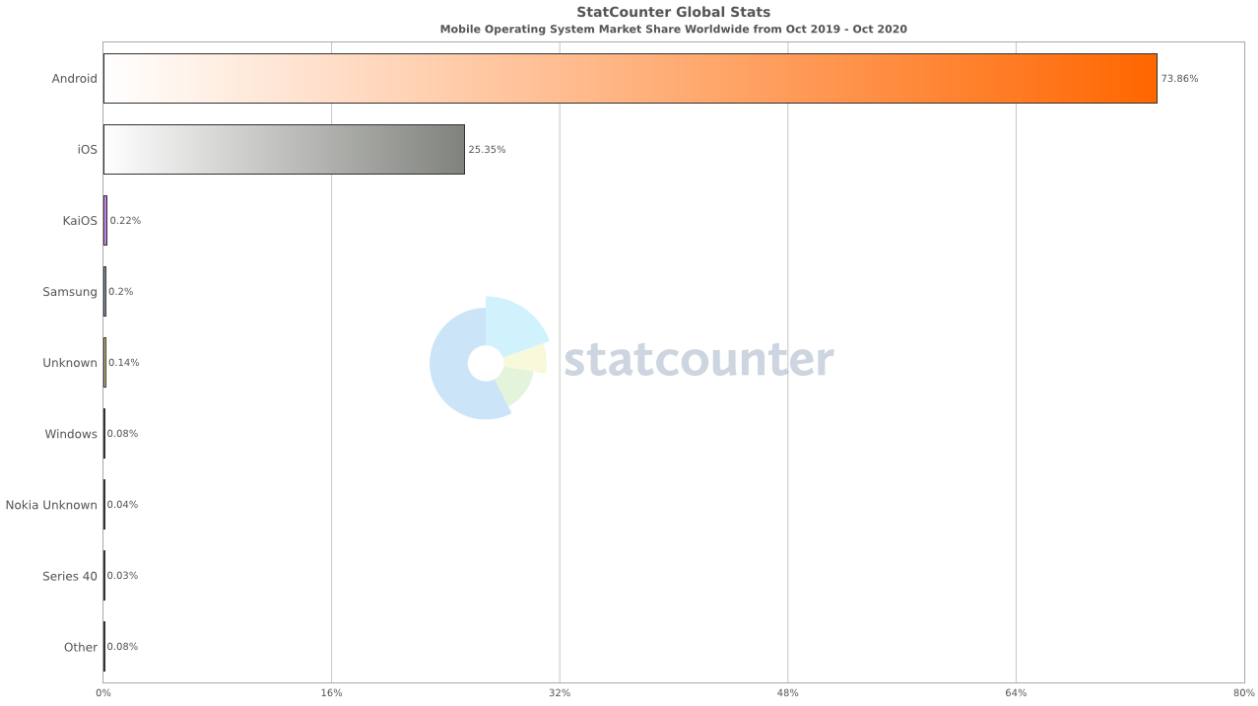

Figura 1
Uso de sistemas operativos móviles [5]

2.1.1 Android

Android es un popular sistema operativo para teléfonos móviles basado en Linux desarrollado por Google y presentado por primera vez en el año 2007. El sistema operativo Android se encuentra diseñado para dispositivos con pantalla táctil, como teléfonos inteligentes, tabletas, relojes inteligentes, televisores, entre otros.

Imagen 3
Logotipo de Android [6]

Android es un proyecto de código abierto ampliamente adoptado. Google desarrolla activamente la plataforma de Android, pero ofrece una parte de ella de forma gratuita a los fabricantes de hardware y operadores de telefonía que desean utilizar Android en sus dispositivos.

El sistema permite el desarrollo de aplicaciones móviles con los lenguajes Java y Kotlin, así como también ofrece una serie de herramientas para facilitar el desarrollo en su kit de desarrollo de software (SDK) tales como debuggers, librerías, emuladores y documentación.

Asímismo, el sistema operativo facilita la interacción con el hardware del dispositivo móvil para poder acceder a los componentes del mismo como lo sería la cámara, llamadas telefónicas o manejo del GPS.

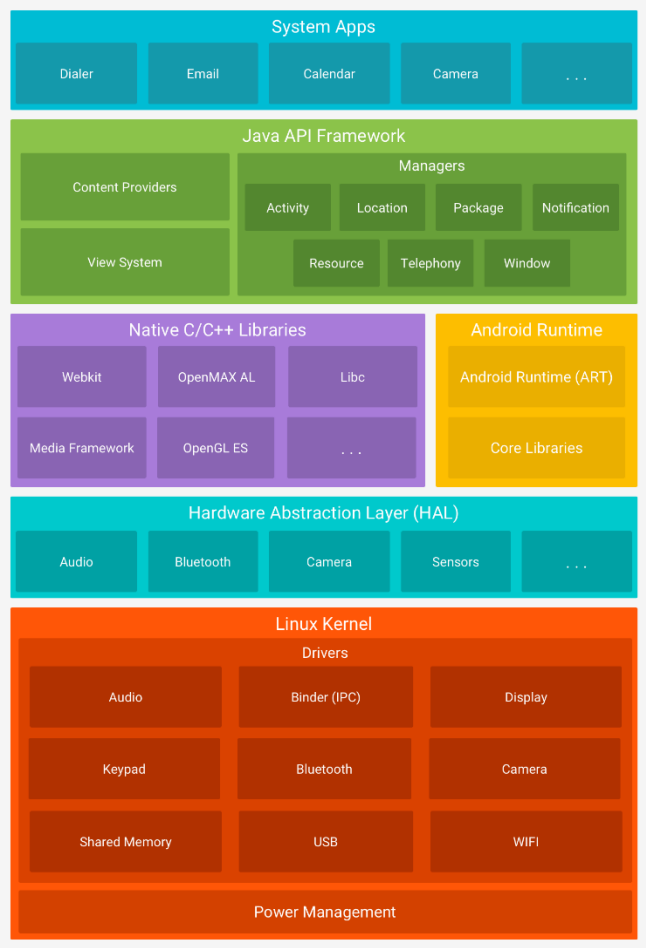

Figura 2
Arquitectura de Android [7]

Al ser un sistema operativo accesible para los desarrolladores al tener libre sus herramientas de desarrollo, desarrollar una aplicación en Android resulta sencillo y no requiere hardware especial para ello. [8]

2.1.2 iOS

iOS es un sistema operativo móvil de Apple lanzado oficialmente el 29 de junio de 2007 derivado de macOS, que a su vez se deriva de Darwin BSD, siendo por lo tanto un sistema operativo tipo Unix.

iOS cuenta con cuatro capas de abstracción: la capa del núcleo del sistema operativo, la capa de "Servicios Principales", la capa de "Medios" y la capa de "Cocoa Touch".

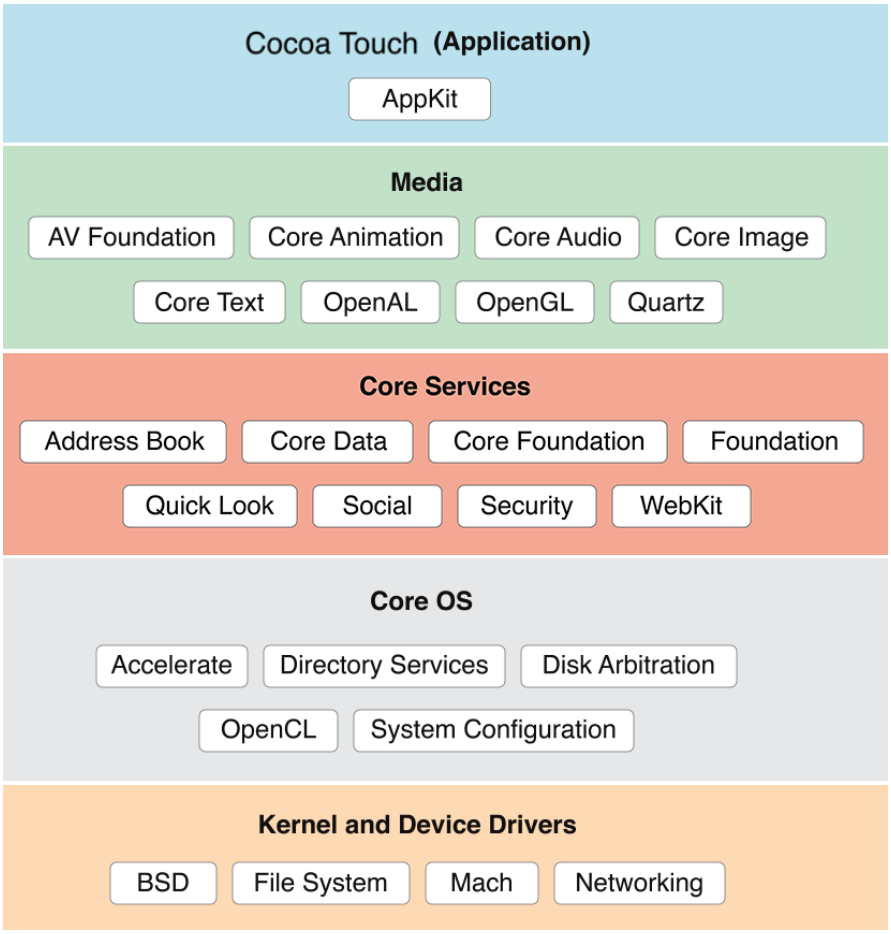

Figura 3
Arquitectura de iOS [9]

A diferencia de Android, este sistema operativo se encuentra únicamente en los equipos desarrollados por la empresa como lo son sus iPhones, iPods, iPads o Apple Watch. Asímismo, el kit de desarrollo de este sistema operativo únicamente es accesible desde los equipos del fabricante, hecho que potencialmente aumenta los costos de desarrollo al requerir de hardware específico [10].

2.2 Aplicaciones móviles

Una aplicación móvil diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. Este tipo de aplicaciones permiten al usuario efectuar un variado conjunto de tareas, facilitando las gestiones o actividades a desarrollar.

Por lo general, se encuentran disponibles a través de ciertas plataformas de distribución, o por intermedio de las compañías propietarias de los sistemas operativos móviles tales como Android, iOS, BlackBerry OS, Windows Phone, entre otros.

2.2.1 Aplicaciones nativas

Las aplicaciones nativas son aquellas aplicaciones que están desarrolladas para un equipo o plataforma determinada. Es decir, funciona en el equipo sin necesidad de ningún programa externo ya que se ha desarrollado en el lenguaje de programación específico de cada equipo. El término de App Nativa está habitualmente asociado a los dispositivos móviles y por tanto hay Apps Nativas para cada sistema operativo como iOS o Android.

Este tipo de aplicaciones se programan con el código nativo de la plataforma en cuestión utilizando las herramientas del kit de desarrollo proporcionado por el equipo responsable del sistema operativo. La instalación de estas aplicaciones se realiza directamente en el dispositivo, usualmente a través de una tienda de aplicaciones como lo es la Play Store de Google.

La principal ventaja de las Apps Nativas es que se adaptan en su totalidad al dispositivo, hecho que le da un buen rendimiento, y pueden utilizar sin mayores problemas todas las funcionalidades del dispositivo (Rendimiento gráfico, GPS, cámara, acelerómetro).

2.2.2 Aplicaciones híbridas

Las aplicaciones híbridas aprovechan al máximo la versatilidad de un desarrollo web y tienen la capacidad de adaptación al dispositivo como una app nativa. Actualmente, existen múltiples herramientas/frameworks que permiten el desarrollo de aplicaciones híbridas, dentro de los cuales se encuentra React Native que utiliza JavaScript y Flutter que utiliza Dart.

Este tipo de aplicaciones han ido recibiendo una mayor adoptación por parte de los desarrolladores, hecho que ha conllevado en una mejora en el proceso de desarrollo y desempeño de la misma.

A diferencia de las aplicaciones nativas, este tipo de aplicaciones no se realiza en el lenguaje nativo del sistema, sino que lo que suele suceder es que se convierte el código fuente en código nativo o se emula la vista, de tal manera que funciona correctamente sin perjudicar el desempeño de la misma.

2.2.3 Web App

Son aquellas aplicaciones desarrolladas con tecnologías web que se pueden utilizar accediendo a un servidor web a través de Internet mediante un navegador.

Estas aplicaciones no se instalan en el dispositivo, sino que se requiere obligatorialmente tener acceso a internet para acceder a ella. Estas aplicaciones tampoco pueden interactuar con el hardware del dispositivo y ofrecen una experiencia al usuario menos negativa, no obstante, su desarrollo es más ágil y pueden cumplir con las funcionalidades requeridas, así como también cuentas con todas las herramientas del desarrollo web.

2.2.4 Comparativa de tipo de aplicaciones

A continuación, se muestra una tabla que compara los diferentes aspectos a considerar al momento de seleccionar un tipo de aplicación móvil para desarrollar [11].

	Nativa	Híbrida	Web App
Costo de desarrollo	Alto	Medio	Bajo
Tiempo de desarrollo	Largo	Medio	Corto
Mantenimiento	Complejo	Medio	Fácil
Experiencia de usuario	Excelente	Bastante buena	Buena
Acceso al dispositivo	Completo	Alto	Parcial
Velocidad	Muy rápida	Rápida	Medio
Portabilidad de código	Nula	Alta	Completa
Seguridad	Alta	Normal	Normal

Tabla 1
Comparativa de tipos de aplicaciones

La selección de un tipo de aplicación móvil se encuentra definida en gran medida por los requerimientos que se tengan para la aplicación en cuestión, ya que para el mejor desempeño se realizaría una aplicación nativa, no obstante, esto conlleva un mayor costo y tiempo de desarrollo.

Asímismo, con el paso de los años han surgido nuevas tecnologías como lo son Flutter y React Native que han mejorado el desempeño y la facilidad de desarrollo de aplicaciones móviles, logrando obtener las ventajas de una aplicación híbrida y con un desempeño casi nativo.

2.3 Lenguajes para el desarrollo en Android

En este momento, el sistema operativo Android cuenta con dos lenguajes oficiales para su desarrollo, siendo estos Java y Kotlin.

2.3.1 Java

Java es un lenguaje de programación orientado a objetos cuya compilación se lleva a cabo en una máquina virtual Java (JVM), fue comercializado por primera vez en 1995 por Sun Microsystems y adquirido en 2010 por la compañía Oracle. Desde inicios de Android, Java es el lenguaje de programación para el desarrollo de aplicaciones nativas.

Imagen 4
Logotipo de Java [12]

En este momento, Java cuenta con una extensa comunidad de desarrolladores, así como una extensa cantidad de librerías para cumplir diferentes funciones, no obstante, es un lenguaje complejo que requiere de líneas excesivas y repetitivas de código para realizar funciones simples.

Es uno de los lenguajes más utilizados gracias a la compatibilidad multiplataformas y a la versatilidad del mismo.

2.3.2 Kotlin

Años después del lanzamiento de Android, surgió Kotlin como una alternativa no excluyente para el desarrollo de aplicaciones móviles en esta plataforma, y logró posicionarse como el lenguaje preferido y recomendado por Google para esto.

Imagen 5
Logotipo de Kotlin [13]

Este lenguaje desarrollado por JetBrains logra tiempos de compilación tan rápidos como Java, pero reduciendo considerablemete las líneas de código requeridas para el desarrollo y facilitando enormemente la lectura y manejo del código, teniendo consigo todos los beneficios de Java y solventando los principales problemas de este.

Asímismo, el código de Java funciona al utilizar Kotlin, funcionando este mismo con diferentes clases realizadas en Java. Esta razón permite la inclusión de este lenguaje en cualquier aplicación previamente desarrollada en Java y permite la reutilización de código.

2.4 Lenguajes para el desarrollo en iOS

Al igual que Android, iOS cuenta con dos lenguajes oficiales para el desarrollo de aplicaciones, los cuales son Objective-C y Swift. Este lenguaje tiene una sintaxis similar a C y es posible compilar código de C en el mismo.

2.4.1 Objective-C

Objective-C es un lenguaje de programación orientado a objetos. Fue el único lenguaje de programación para desarrollo de los años 2007 a 2014.

Imagen 6
Logotipo de Objective-C [14]

Teniendo una sintaxis complicada y un alto nivel de complejidad, este lenguaje es complicado de utilizar, no obstante, partes importantes del SO se encuentran desarrollados en este lenguaje, así como una gran cantidad de librerías y conocer este lenguaje permite la optimización y detección de problemas de bajo nivel.

2.4.2 Swift

Swift es un lenguaje de programación multiparadigma creado por Apple enfocado en el desarrollo de aplicaciones para iOS y macOS. Fue presentado en la WWDC 2014 y está diseñado para integrarse con los Frameworks Cocoa y Cocoa Touch; puede usar cualquier biblioteca programada en Objective-C y llamar a funciones de C.

Imagen 7
Logotipo de Swift [15]

Es posible desarrollar código en Swift compatible con Objective-C bajo ciertas condiciones. Swift tiene la intención de ser un lenguaje seguro, de desarrollo rápido y conciso. Usa el compilador LLVM incluido en Xcode 6. Fue presentado como un lenguaje propietario, pero en el año 2015, con la versión 2.2, pasó a ser de código abierto con la Licencia Apache 2.0. 2.5 Herramientas para desarrollo multiplataforma

Este lenguaje simplifico considerablemente el desarrollo de aplicaciones móviles, reduciendo considerablemente la curva de aprendizaje de Objective-C y simplificando

el desarrollo mientras se mantiene la eficiencia, además de ser un lenguaje con sintaxis moderna, lo que facilita la adopción del mismo [16].

2.5 Lenguajes para desarrollo híbrido

2.5.1 Flutter

Flutter es un SDK de código fuente abierto de desarrollo de aplicaciones creado por Google para crear hermosas aplicaciones compiladas de forma nativa para dispositivos móviles, web y de escritorio desde una única base de código usando Dart como lenguaje de programacicón.

Imagen 8
Logotipo de Flutter [17]

En cuanto al funcionamiento de Flutter, la aplicación se compila y corre nativamente en la plataforma en la que se esté utilizando. Flutter proporciona un *shell*, que aloja la máquina virtual de Dart. El *shell* es específico de la plataforma, brinda acceso a las API de la plataforma nativa y aloja el establecimiento del lienzo relevante de la plataforma.

Los *shells* también ayudan a proporcionar comunicación a los IME relevantes (por ejemplo, el teclado) y los eventos del ciclo de vida de la aplicación del sistema.

Posteriormente, el *engine* proporciona *Dart Runtime, Skia, Platform Channels* y mucho más. Dicho engine corre dentro de la plataforma en la que se ejecuta la aplicación.

Los widgets son los componentes básicos de su aplicación. Flutter no tiene controles o componentes nativos. Flutter dibuja la salida de la interfaz de usuario en un Skia Canvas. Esto reduce la complejidad drásticamente, ya que Flutter solo tiene widgets. Los widgets son controles de IU que puede usar en su aplicación. Toda la aplicación estará compuesta por widgets sin estado o con estado.

Flutter actualiza dicha interfaz de usuario a 60 fps y usa la GPU para la mayor parte del trabajo. Si bien esto no afecta la forma en que se crea la aplicación, es la razón por la que la interfaz de usuario de Flutter es muy fluida. El código de la aplicación basado en Dart continuará ejecutándose a través de la CPU y en el subproceso de interfaz de usuario especializado [18].

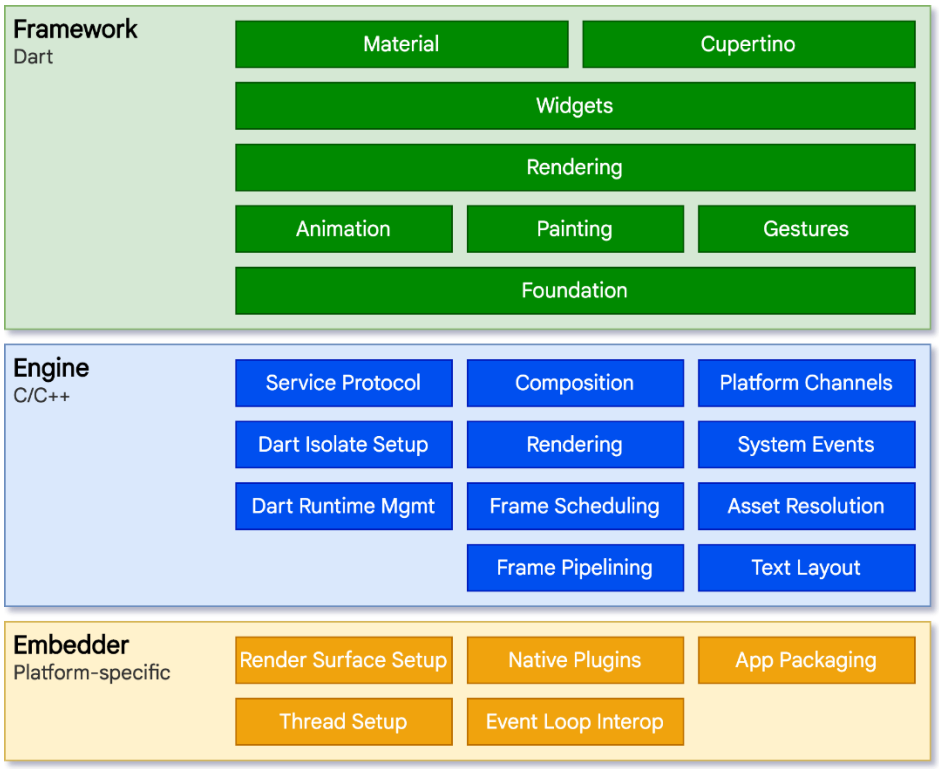

Figura 4
Estructura de Flutter [17]

Ventajas de Flutter

- Recarga caliente: al hacer algún cambio en el código se podrán ver los efectos reflejados inmediatamente, sin tener que compilar la aplicación de nuevo y sin perder el contexto en el que estábamos.
- Renderizado de vistas rápido y constante: Flutter se ha planteado objetivos de renderizado muy altos para ciertos dispositivos, lo que supera con creces a cualquiera otra solución de desarrollo móvil híbrido.

- Desarrollo multiplataforma: no es necesario construir por separado para las dos plataformas: Android y IOS. Flutter ya genera un código base que sirve para ambas plataformas.
- Acceso a las funciones nativas: algunas funciones específicas de la plataforma, como la cámara y la geolocalización, requieren acceso a funciones nativas. Estas funciones deben implementarse mediante lenguajes nativos, y Flutter da la sensación de desarrollarse en la plataforma nativa. Flutter permite reutilizar código existente de Java, Swift y Objective-C para acceder a las funciones nativas y SDK en iOS y Android.

Desventajas de Flutter

- Dart necesario: para poder usar Flutter es necesario aprender el lenguaje de programación Dart.
- Framework muy joven: y aún no tiene una gran comunidad detrás, por lo que se deberán afrontar los problemas que nos encontremos con menos ayuda que en otros frameworks.
- Está enfocado solo a móvil: por el momento solo hay una versión oficial de Flutter y solo está enfocada para móvil. Así si nuestra aplicación va a tener un sitio web tendremos que desarrollarlo paralelamente a la versión de móvil.
- Librerías limitadas: las bibliotecas a las que pueden acceder los desarrolladores de aplicaciones móviles están muy limitadas en Flutter. No siempre proporcionan todas las funcionalidades que necesita el desarrollador. Dichas funcionalidades deben ser desarrolladas por los desarrolladores de aplicaciones por sí mismas.

2.5.2 React Native

React Native es un framework JavaScript anunciado y mantenido por Facebook en marzo de 2015 el cual permite crear aplicaciones reales nativas para iOS y Android, basado en la librearía de JavaScript React para la creación de componentes visuales, cambiando el propósito de los mismos para, en lugar de ser ejecutados en navegador, correr directamente sobre las plataformas móviles nativas, en este caso iOS y Android.

Imagen 9
Logotipo de React Native [19]

React Native tiene su propio proceso para la generación de una aplicación con componentes nativos e interacción nativa con las funciones del sistema, dicho proceso se puede visualizar en la siguiente figura.

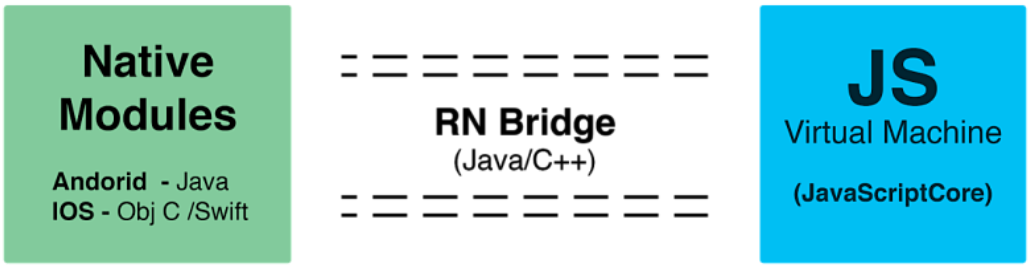

Figura 5
Arquitectura de ejecución de React Native [19]

1. **Módulos/códigos nativos**: son todos los módulos necesarios (tanto de iOS como de Android) para que cuando hagamos nuestra RN app no tengamos que preocuparnos de escribir código nativo.
2. **Javascript VM**: es la Virtual Machine de Javascript que ejecutará nuestro código JS. Tanto en iOS como en Android se utiliza JavascriptCore, que es el motor Javascript que utiliza Webkit para Safari. Esta pieza de software ya viene incluida en los dispositivos iOS, pero no en Android. Por lo que RN empaquetará esta pieza dentro del apk, lo que incrementa el tamaño en 3-4 megabytes.
3. **React Native Bridge**: es un bridge React Native escrito en C++/Java y es responsable de comunicar los hilos de Javascript y de nativo. Entre ellos se hablan en un protocolo de mensajes [19].

Ventajas de React Native

- Recarga caliente: al hacer algún cambio en el código se podrán ver los efectos reflejados inmediatamente, sin tener que compilar la aplicación de nuevo y sin perder el contexto en el que estábamos.
- Desarrollo multiplataforma: con el mismo código base es posible compilar para iOS y Android.
- Acceso a las funciones nativas: es posible acceder a funciones nativas del sistema como lo son la cámara o el almacenamiento.
- Javascript: React Native se programa en Javascript, lo cual le otorga una facilidad al momento de aprenderlo gracias a la relevancia actual del lenguaje.
- Comunidad: React cuenta con una extensa cantidad y una extensa cantidad de librerías para los desarrolladores.

Desventajas de React Native

- Framework joven: el framework sigue en desarrollo, por lo que se encuentra sujeto a posibles cambios con el tiempo.
- Componentes definidos: Son pocos los componentes que te ofrece el marco de desarrollo, por lo que es necesario realizarlos manualmente.

2.5.3 Ionic

Ionic es un SDK completo de código abierto para el desarrollo de aplicaciones móviles híbridas creado por Max Lynch, Ben Sperry y Adam Bradley de Drifty Co. en 2013. Ionic proporciona herramientas y servicios para desarrollar aplicaciones móviles híbridas utilizando tecnologías web.

Imagen 10
Logotipo de Ionic [20]

Es sencillo crear una aplicación utilizando tecnologías web que se pueden presentar más adelante en las tiendas de aplicaciones nativas para los dispositivos multiplataforma. Para acceder a las funciones de los sistemas operativos, como la

cámara, la ubicación o el audio, etc., Ionic utiliza complementos de Cordova que permiten el acceso a estas funciones.

Ionic no genera componentes nativos, sino que el código de HTML/CSS/JS se muestra en un WebView, similando la apariencia y funcionalidad de una aplicación nativa.

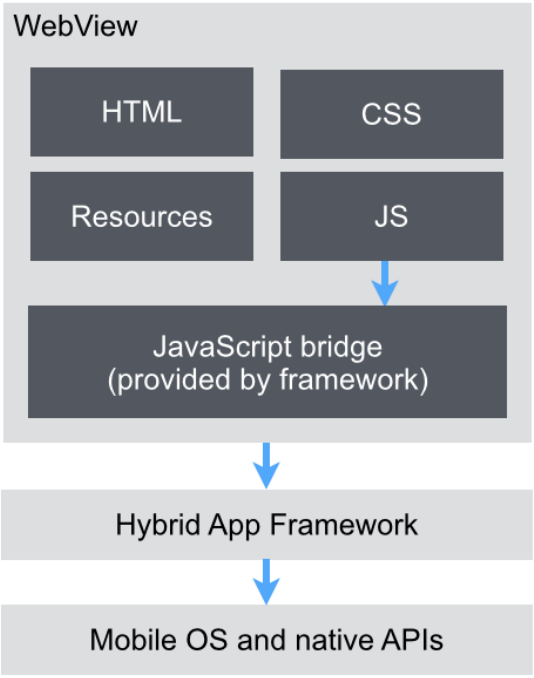

Figura 6
Arquitectura de Ionic [21]

Ventajas de Ionic

- Lenguaje de desarrollo: Ionic utiliza los lenguajes de web, hecho que hace la adaptación al mismo muy sencilla para los que ya conozcan este paradigma.
- Reutilización: el código de la aplicación es reutilizable en entornos web.
- Desarrollo multiplataforma: la aplicación desarrollada es compatible con todas las plataformas.
- Acceso a las funciones nativas: se tiene acceso a funciones nativas del sistema.

Desventajas de Ionic

- Requiere internet: al ser una página web embebida, se requiere de acceso a internet.

- Desempeño: Su desempeño es muy inferior a las alternativas previas, ya que no hace uso de componentes nativos.

2.5.4 Comparativa de frameworks

A continuación, se muestra una tabla con una comparativa de las principales características de cada uno de los frameworks mencionados previamente [22].

	Flutter	React Native	Ionic
Lenguaje de programación	Dart	Javascript	Javascript
Reusabilidad	Alta	Media	Alta
Librerías	Media, en aumento	Alta	Alta
Popularidad	Alta	Alta	Media
Comunidad	Extensa	Muy extensa	Media
Desempeño	Excelente	Bueno	Medio
Costo	Sin costo	Sin costo	Sin costo, paquete pro de paga
Tiempo de desarrollo	Medio	Alto	Bajo

Tabla 2
Comparativa de frameworks

2.6 Controlador de versiones

“Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante.” [23].

En el desarrollo de aplicaciones móviles, la agilidad y el trabajo colaborativo son los aspectos fundamentales, ya que cada vez se requieren desarrollos más veloces, los cuales solo son posibles con un equipo bien comunicado y manejando el código con un controlador de versiones para evitar conflictos entre las diferentes partes trabajadas y permitir trabajar un mismo proyecto e integrar los cambios sencillamente.

El software utilizado para este es Git, el cual “es un software de control de versiones diseñado por Linus Torvalds, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando éstas tienen un gran número de archivos de código fuente. Su propósito es llevar registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.” [24].

Imagen 11
Logotipo de Git [23]

Al usar Git, se requiere de un repositorio de código en el cual se almacenan los archivos del proyecto a trabajar, este repositorio se configura en un servidor propio con Git instalado o en un prestador de servicio. Este repositorio es clonado por todos los interesados en sus máquinas locales, y ya es posible manejar cambios localmente y subirlos al repositorio cuando estén listos, cambios que estarán disponibles para todo el equipo de trabajo.

Además de esto, Git ofrece facilidades para poder manejar diferentes entornos como lo son un entorno de desarrollo, uno de pruebas y producción. Estás funciones hacen que Git sea imprescindible en un proyecto de desarrollo de software.

2.7 Metodologías ágiles

Estas metodologías nacieron en la industria del desarrollo de software, cuando las compañías de este sector comprendieron que la forma tradicional de trabajo retrasaba mucho la entrega del producto final. Unos procesos basados normalmente en un contrato cerrado, con escasa comunicación de los trabajadores, que conducían a entregables de mala calidad.

Los princicpios de las metodologías ágiles quedaron definidos en el “Manifiesto ágil”, el cual propone un modelo de mejora continua en el que se planifica, se crea, se comprueba el resultado y se mejora. Algo que es constante y rápido, con plazos de entregas reducidos que buscan evitar la dispersión y centrar toda la atención en una tarea encomendada [25].

Las principales ventajas que este modelo ofrece son:

- Mejora la calidad: Minimiza los errores en los entregables y mejora la experiencia y las funcionalidades para el cliente.
- Mayor compromiso: Mejora la satisfacción del empleado y genera conciencia de equipo.
- Rapidez: Acorta los ciclos de producción y minimiza los tiempos de reacción y toma de decisiones.
- Aumento de la productividad: Al asignar mejor los recursos, y de forma más dinámica, mejora la producción según las prioridades que tenga la empresa.

Existen diferentes metodologías ágiles, todas respetando lo establecido en el manifiesto ágil, pero con ideas y artefactos propios los cuales logran una diferencia notable entre estas metodologías, las metodologías que más destacables por su uso son: Extreme programming, Kanban, Lean y Scrum.

Sobre las metodologías mencionadas anteriormente, cabe destacar Scrum sobre las demás por sus amplía recepción y fácil integración en los equipos de trabajo, razones que hacen de ella una metodología muy importante en el desarrollo de software, razón por la cual se mencionará a mayor detalle su funcionamiento.

2.7.1 Scrum

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

En Scrum un proyecto se ejecuta en ciclos temporales cortos y de duración fija (iteraciones que normalmente son de 2 semanas, aunque en algunos equipos son de 3 y hasta 4 semanas). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite [26].

2.8 Interfaz de Programación de Aplicaciones

Una Interfaz de Programación de Aplicaciones (API por sus siglas en inglés) es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones.

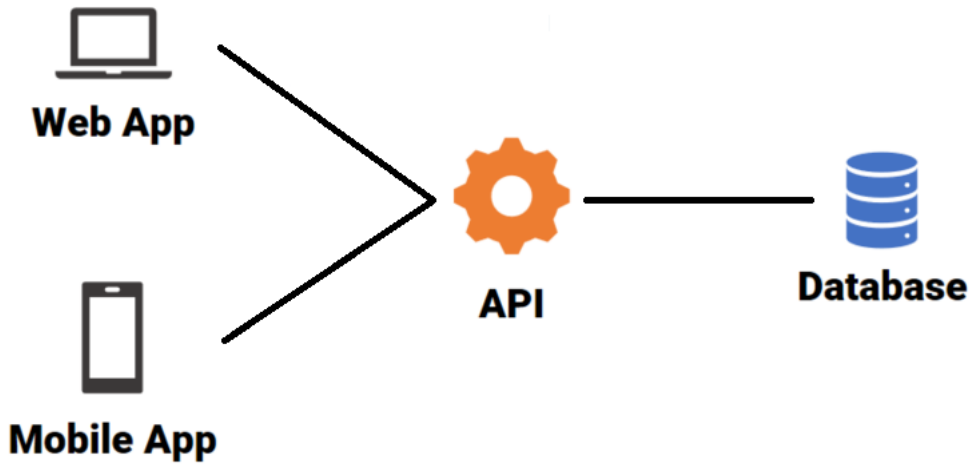

Figura 7
Funcionamiento de una API

Las API permiten que sus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Esto simplifica el desarrollo de las aplicaciones y permite ahorrar tiempo y dinero. Las API le otorgan flexibilidad; simplifican el diseño, la administración y el uso de las aplicaciones, y proporcionan

oportunidades de innovación, lo cual es ideal al momento de diseñar herramientas y productos nuevos (o de gestionar los actuales) [27].

Las API son ampliamente utilizadas para el desarrollo de aplicaciones web y aplicaciones móviles, ya que aunque si es posible acceder a la base de datos desde este tipo de aplicaciones, esto no es seguro ni recomendado ya que se pone en riesgo la seguridad de la base de datos, es por esto que se utiliza una o varias APIs para comunicarse con los diferentes servicios que se requieren.

Capítulo 3

Metodología

3.1 Análisis

Al momento de diseñar una aplicación multiplataforma, es relevante contar con un análisis de las funciones que se quieren abarcar en la aplicación, y definir la parte estética, tomando en consideración las funcionalidades típicas y componentes nativos de las plataformas a tratar, en este caso, iOS y Android. Es importante tener bien definida esta parte ya que idealmente se debe mantener la experiencia del usuario según el ecosistema que utiliza.

La implementación de una metodología ágil como lo es Scrum en conjunto con sus artefactos facilita la comunicación del equipo, permitiendo la planificación de las tareas en el tiempo definido, de tal manera que se pueda desarrollar la aplicación con el menor número de impedimentos y se pueda tratar cualquier problema que surja de manera eficiente por el constante flujo comunicativo.

Contar con los artefactos de Scrum, permite conocer el desempeño del equipo de trabajo en un periodo de tiempo dado, por lo que se pueden optimizar las entregas según el desempeño del equipo, así como poder consultar un historial de todo lo que se ha realizado y lo que aún falta de realizar.

Existiendo actualmente una variedad de frameworks o herramientas a elegir, es importante tomar diferentes aspectos en cuenta, se optó por usar el framework Flutter para el presente ejemplo gracias a la facilidad de desarrollar aplicaciones de manera ágil, el desempeño de las mismas, la popularidad del mismo y la baja curva de aprendizaje del framework.

Para la aplicación, se utilizará una API Rest para consumir los servicios requeridos y el manejo de la información en cuestión, dicha API se realizará en con el entorno NodeJS y el lenguaje JavaScript, y se comunicará con una base de datos relacional MySQL.

Por motivos del tema a desarrollar, no se tomará en cuenta el desarrollo del backend y administración de la base de datos, únicamente se tratará la aplicación en cuestión y se tomara que las demás partes requeridas para el desarrollo ya se encuentran terminadas.

3.2 Diseño

Para disminuir los posibles cambios que puedan ocurrir en el desarrollo de las interfaces de usuario, es importante desarrollar los diferentes diseños de las vistas que tendrá la aplicación, de esta manera se reducen los posibles cambios a futuro y se simplifica el desarrollo de la aplicación en cuestión, al solo requerir maquetar y programar la lógica.

Existen diferentes herramientas para trabajar los diseños, para el ejemplo en cuestión, se utilizará Figma, la cual es una herramienta web que permite diseñar interfaces web y móviles con facilidad de manera colaborativa y sin requerir equipos con hardware específico al utilizarla por medio de un navegador web.

3.3 Herramientas de desarrollo

3.3.1 Instalación de Flutter

La instalación de Flutter en Windows requiere de las siguientes especificaciones y herramientas:

- Sistema operativo: Windows 7 SP1 o siguientes (64 bits)
- Espacio de almacenamiento: 1.32 GB
- Herramientas: Windows Powershell 5.0 y Git para Windows versión 2.x.

Primeramente, se descarga el Flutter SDK del sitio oficial: https://flutter.dev/docs/get-started/install/windows. Este es un archivo .zip que se extrae en la carpeta deseada. Seguido de esto es posible ejecutar comandos de Flutter desde el cmd estando en la ubicación del SDK, o es posible configurar Flutter en las variables de entorno para acceder a sus comandos desde cualquier ubicación.

Para configurar Flutter, en las variables de entorno, es necesario editar las variables de entorno del sistema, agregar una nueva variable, y establecer la ruta de flutter\bin como el valor de la misma.

Flutter tiene una serie de requerimientos para su correcto funcionamiento, estos requerimientos se pueden visualizar utilizando el comando *flutter doctor* desde la terminal de Windows.

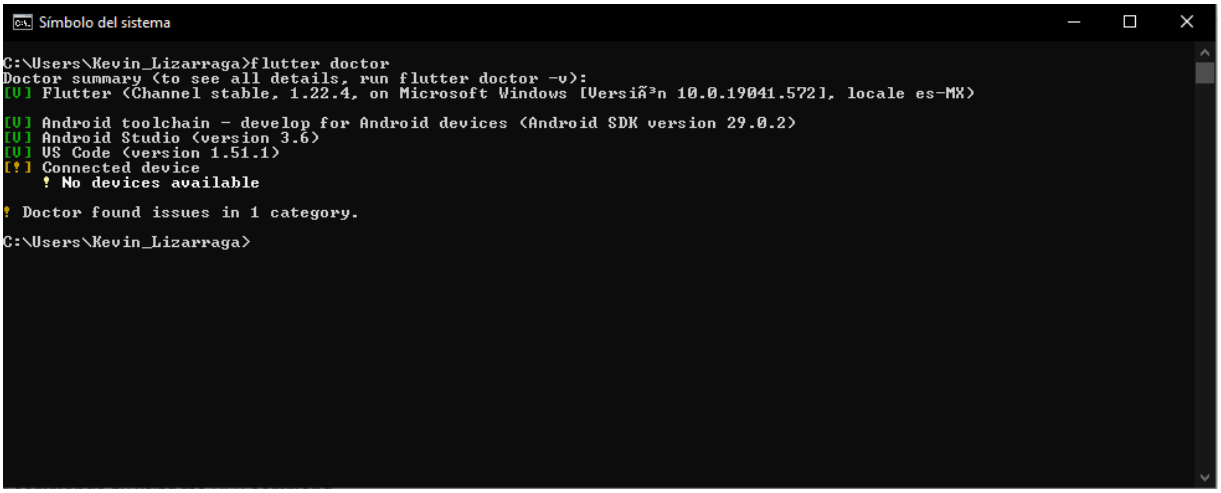

Figura 8
Comando flutter doctor

Los requisitos que pide Flutter son los siguientes:

- Instalar Android Studio. Se requiere del IDE de Android Studio y este incluye el SDK de Android. Este se puede descargar del sitio oficial, y la instalación consiste de utilizar el instalador del software para la instalación de los componentes requeridos.

Imagen 12
Instalador de Android Studio

- Instalar Visual Studio Code. La programación de Flutter se lleva a cabo en este IDE idealmente, gracias a las herramientas que ofrece. Se puede descargar del sitio oficial.
- Preparar un dispositivo físico. Para esto, en este caso se ejemplificará con un dispositivo Android. Para este dispositivo es necesario entrar en las opciones de desarrollador del dispositivo y activar la opción “USB Debugging”. Para visualizar como acceder a dichas opciones, se debe ver la documentación de la versión de Android en cuestión [28].

3.3.2 Creación del proyecto

Para crear un proyecto en Flutter, se abre la terminal del sistema operativo en la ubicacón en la que se desee crear el proyecto y se ejecuta el comando flutter create <my-app>, reemplazando <my-app> por el nombre del proyecto.

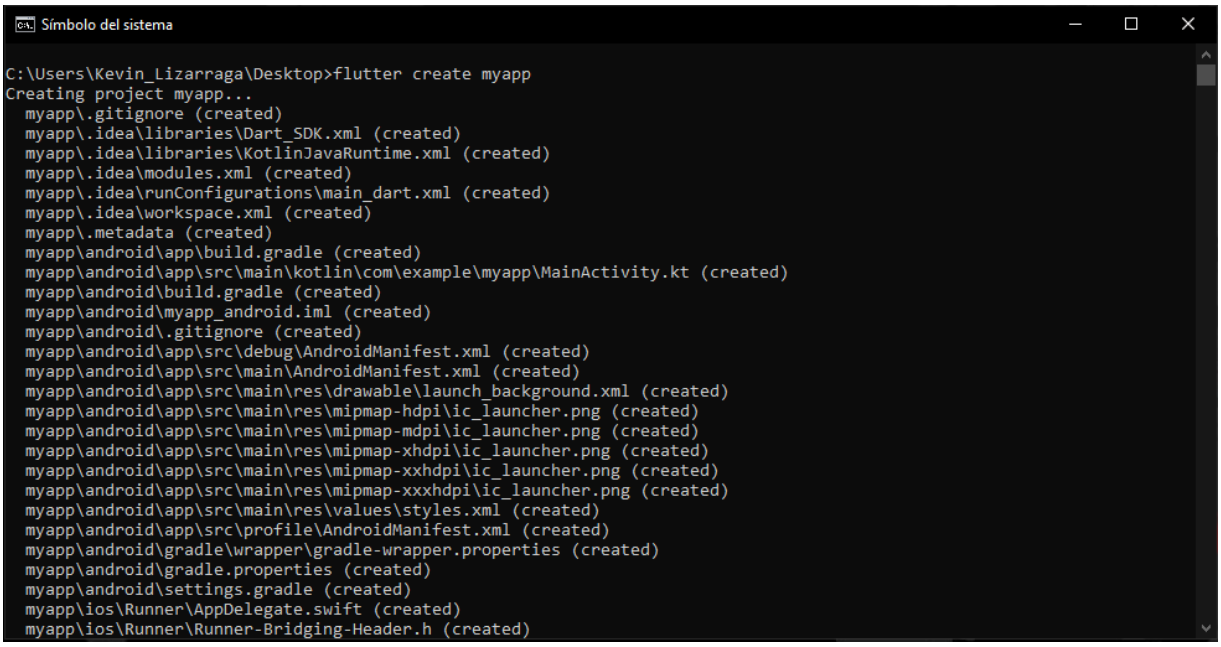

Figura 9
Creación de proyecto en Flutter

Al abrir el proyecto previamente generado, se encuentra configurado con un proyecto inicial, y con todos los archivos necesarios para desarrollar el código compilable para

iOS y Android [29]:

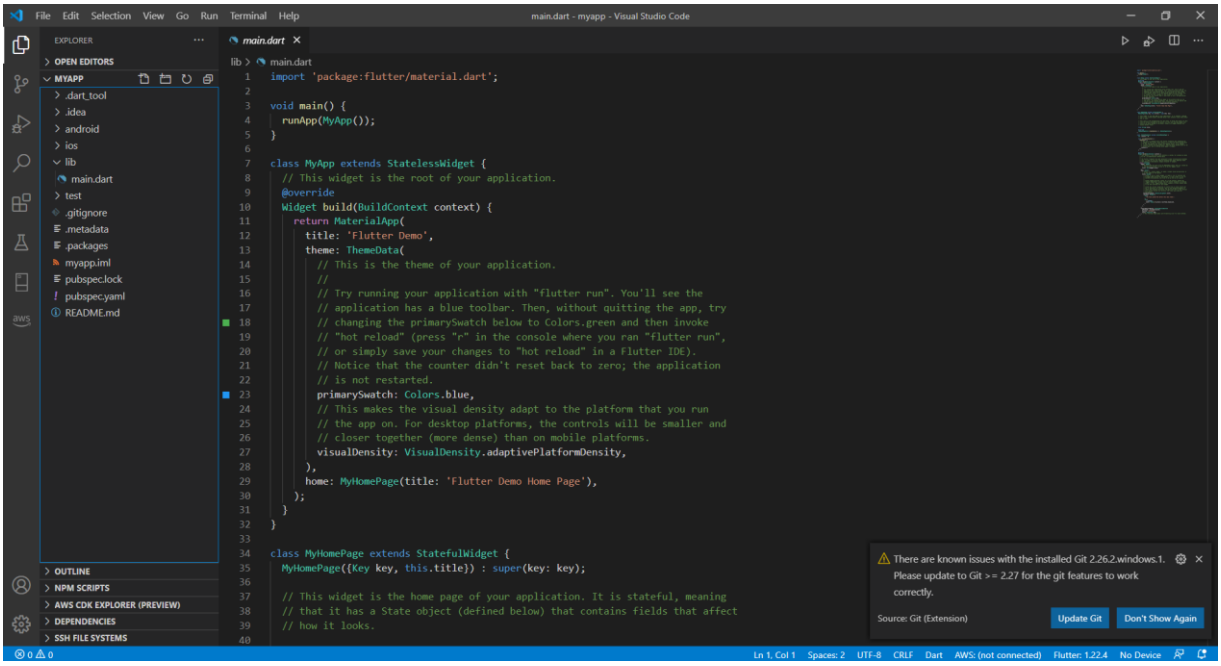

Figura 10
Visualización del proyecto de Visual Studio Code

3.3.3 Configuración de Git

Al momento de crear el proyecto con el comando *flutter create <my-app>*, este por defecto contiene un archivo ".gitignore", dicho archivo define las carpetas o archivos que no serán agregados al momento de subir cambios en un repositorio, usualmente se agregan en este archivo los directorios de las dependencias, constantes o llaves de servicios externos y cualquier archivo que se configure con direcciones de archivos del usuario.

Previo a subir el código a un repositorio, es necesario crear uno en un servidor de controlador de versiones como lo es Github, Gitlab, Bitbucket o CodeCommit. Tras crear dicho repositorio, es necesario obtener la URL del mismo para terminar de vincular el proyecto con el repositorio.

A continuación, se necesitan ejecutar los siguientes comandos en la raíz del proyecto:

- git init: Inicializa git en el proyecto
- git remote add origin <URL del repositorio>: Vincula el proyecto con el repositorio

- git add *: Agrega todos los cambios del proyecto
- git commit –am “comentario”: Se hace un commit de los cambios agregados con un comentario
- git push origin master: Se suben los cambios en el repositorio

Dependiendo del proyecto, se puede necesitar trabajar con diferentes ramas como lo puede ser un entorno de desarrollo (dev) o de control de calidad (qa). Estas ramas se pueden crear directamente desde el servicio de git utilizado, o localmente utilizando el comando “git checkout –b <nombre>”.

3.4 Desarrollo

Para demostrar la hipótesis planteada en el capítulo 1 del presente docucmento, se usará de ejemplo una aplicación móvil desarrollada en Flutter, con enfoque en el sistema operativo Android, de tal manera que se abarque el desarrollo de la misma en un periodo corto de tiempo, haciendo uso de buenas prácticas y de la metodología ágil Scrum.

3.4.1 Información de la aplicación

La aplicación en cuestión debe contar con las siguientes funcionalidades:

- Registro como perfil de asesor y asesorado
- Inicio de sesión
- Gestión de asesorías según el perfil:
    - Asesor: Aplicar, iniciar y finalizar una asesoría.
    - Asesorado: Crear, modificar y gestionar una asesoría.
- Historial de asesorías

Es importante recordar que los diseños de la aplicación y la API para la consulta al servidor backend no se consideran en el presente documento, estos elementos se consideraran como ya proporcionados y únicamente se mencionara las cuestiones pertinentes al desarrollo.

Otras características inherentes a una aplicación con las funcionalidades previamente mencionadas, es que esta debe:

- Guardar la sesión localmente.
- Validar accesos a módulos por perfil.
- Respetar la interacción con el usuario de acuerdo a la plataforma.

3.4.2 Desarrollo de la aplicación

Primeramente, se limpia el código de la aplicación por default y se establecen las diferentes carpetas en las que se van a organizar los diferentes archivos de la siguiente manera:

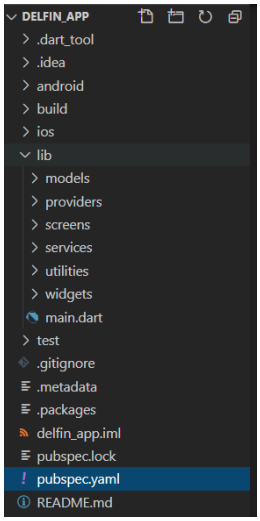

Figura 11
Estructura del proyecto

El motivo de dicha configuración es poder gestionar y ubicar de una manera más eficiente los diferentes archivos de la aplicación, separando los modelos de datos, los providers, las pantallas completas, los componentes utilizados, los servicios http en un archivo único y cualquier otro archivo del proyecto.

Para establecer la comunicación con los servicios proporcionados por una API Rest, se instala un plugin para el manejo de peticiones por el protocolo HTTP llamada httpd, especificando el nombre del plugin y la versión a emplear en el archivo pubspec.yaml, ubicado en la raíz del proyecto, como se muestra a continuación (ver figura 11).

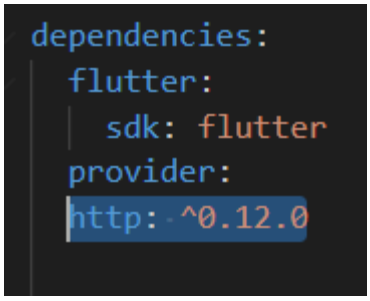

Figura 12
Dependencias del proyecto

Una vez preparada la configuración, se inicia el desarrollo de la aplicación. En el prersente caso, se comienza con una vista que permite seleccionar entre la opción de registrarse o iniciar sesión.

Un aspecto importante que simplifica cualquier cambio en los diseños, es la declaración de variables globales para los colores primarios y colores de acento de la aplicación, así como los estilos de los textos y tipografía.

En las vistas de la aplicación que se visualizarán a continuación, se mostrará en el lado izquierdo las variables de la pantalla, los métodos definidos y el árbol de componentes que conforman la UI; y en el lado derecho, la interfaz de usuario final.

En el caso de esta primera vista, no se requirío codificar manualmente ninguno de los componentes mostrados; únicamente se hizo uso de los widgets proporcionados por Flutter para los elementos visuales y el posicionamineto de los mismos.

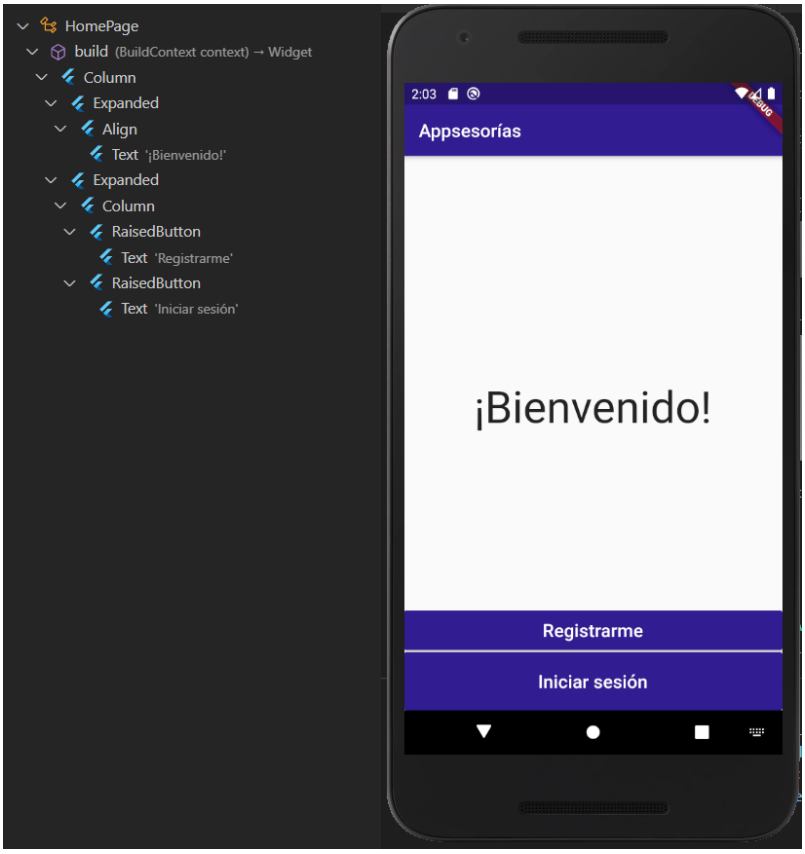

Figura 13
Vista inicial de la app

Para la segunda vista, se necesita que el usuario seleccione el tipo de usuario con el cual quiera registrarse, para ello se estilizo un widget de *RaisedButton* que proporciona Flutter, el cual se reutiliza en otras interfaces de la aplicación. Tras seleccionar una de las dos opciones establecidas, se abre la vista con el formulario correspondiente.

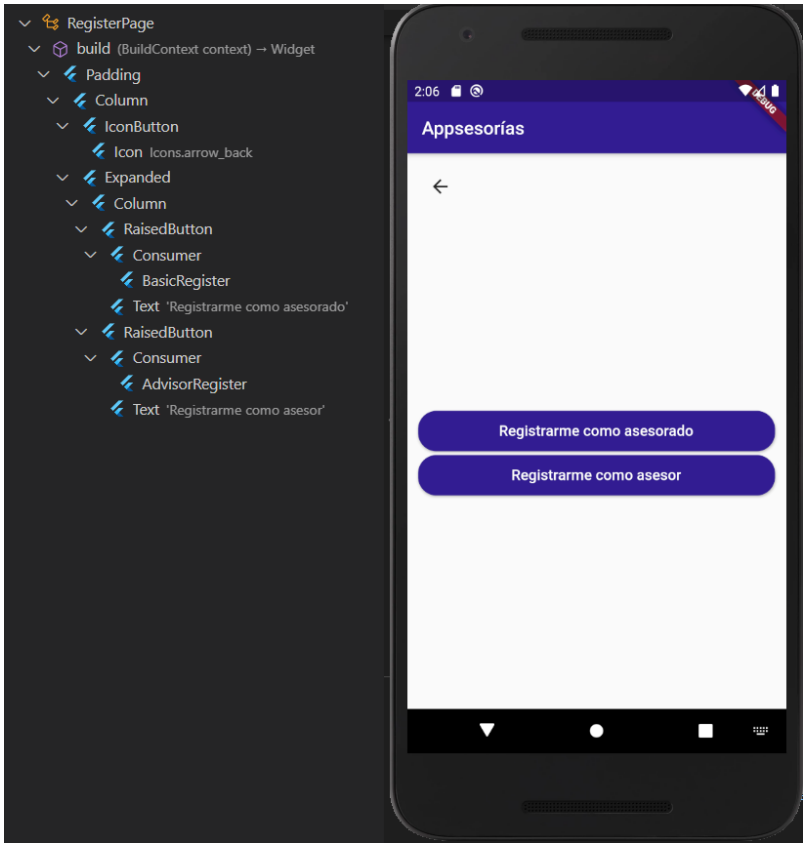

Figura 14
Selección de tipo de registro

En el registro como asesorado, se visualiza un formulario estilizado; gracias a que Flutter proporciona todas las herramientas requeridas para manejar y validar los diferentes tipos de información que se puede requerir de un usuario. Por ello únicamente se requirió colocar los elementos, conectarlos con la API Rest del proyecto y estilizar la interfaz de usuario, el estilo utilizado para los inputs se reutiliza ampliamente en la aplicación.

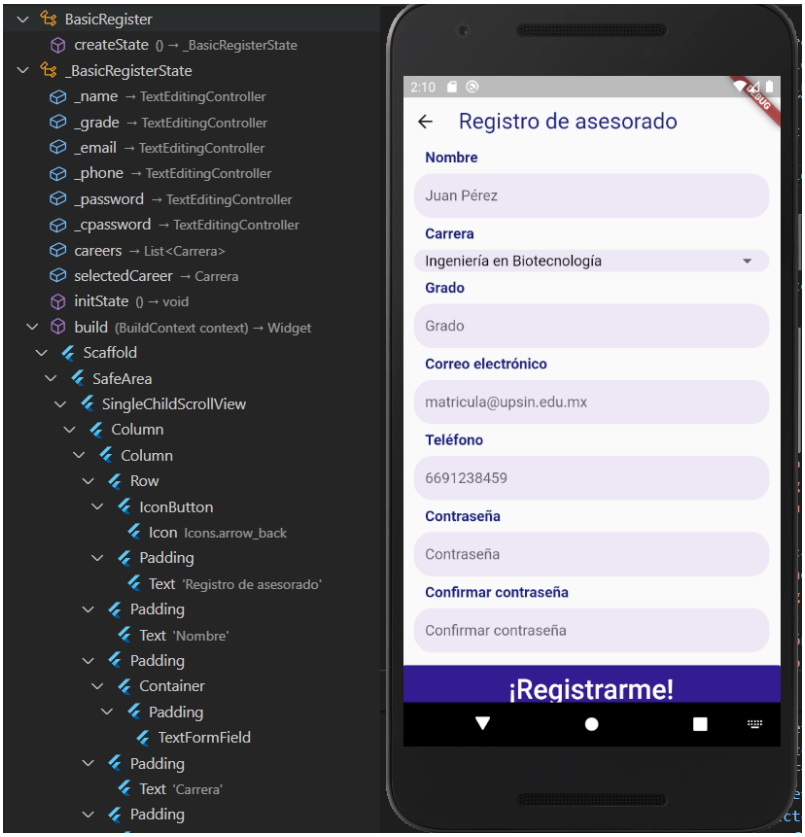

Figura 15
Registro de asesorado

Para el registro como asesor, se reutilizo en su mayoría la vista del registro de asesorado. Pero esta vista requiere de un mayor dinamismo ya que en ella se le solicita al usuario ingresar una cantidad indeterminada de asignaturas. En esta pantalla se aplica una de las ventajas de Flutter, la cual es que, al alterar el estado de un componente, el cambio se refleja el cambio en la pantalla automáticamente, lo que significa que la interacción del usuario no se interrumpe por estos cambios, y a la vez, el equipo de desarrollo no necesita realizar código extra para actualizar la pantalla.

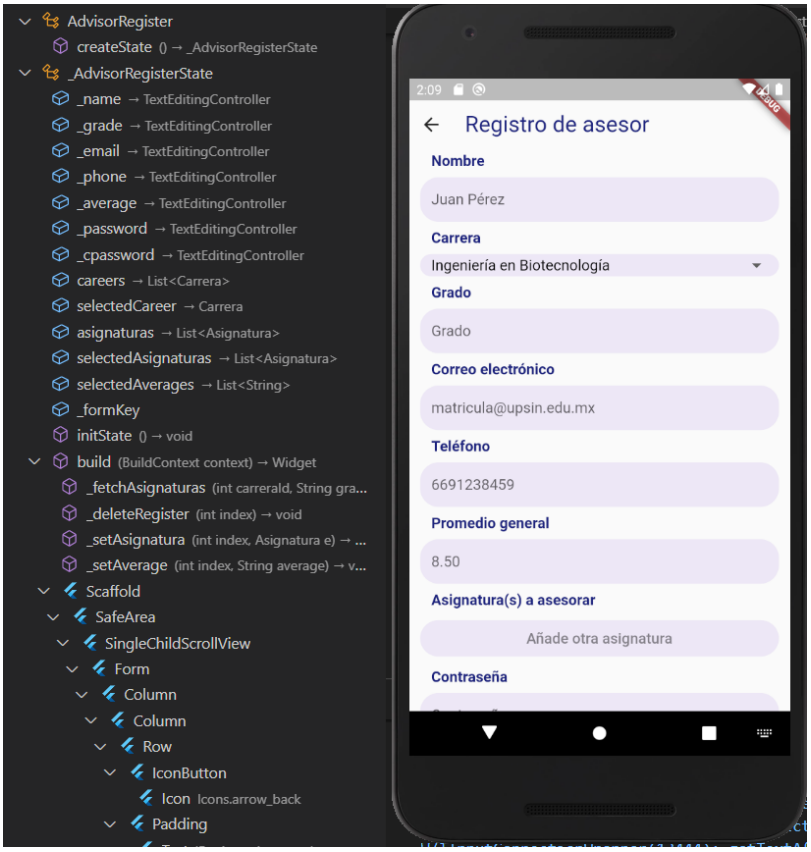

Figura 16
Registro de asesor

Para la vista de inicio de sesión, se reutilizaron los estilos para los campos de texto y para el botón de iniciar sesión, por lo que únicamente se validan los campos con facilidades que otorgan los componentes y se llama al servicio para iniciar sesión. La parte importante de este componente radica en el guardado de los datos tras poder iniciar sesión éxitosamente.

Para el manejo de los datos, se utiliza una de las características de Flutter al utilizar un paquete llamado Provider, el cual permite centralizar la información de la aplicación y poder llamar e interactuar con ésta desde cualquier parte subsecuente a dicho provider después de su inicialización. Esto facilita el manejo de la información entre pantallas, así como poder reflejar cambios en cualquier pantalla tras cambiar cualquier información almacenada, permitiendo de esta manera garantizar que siempre se muestre información actualizada al usuario.

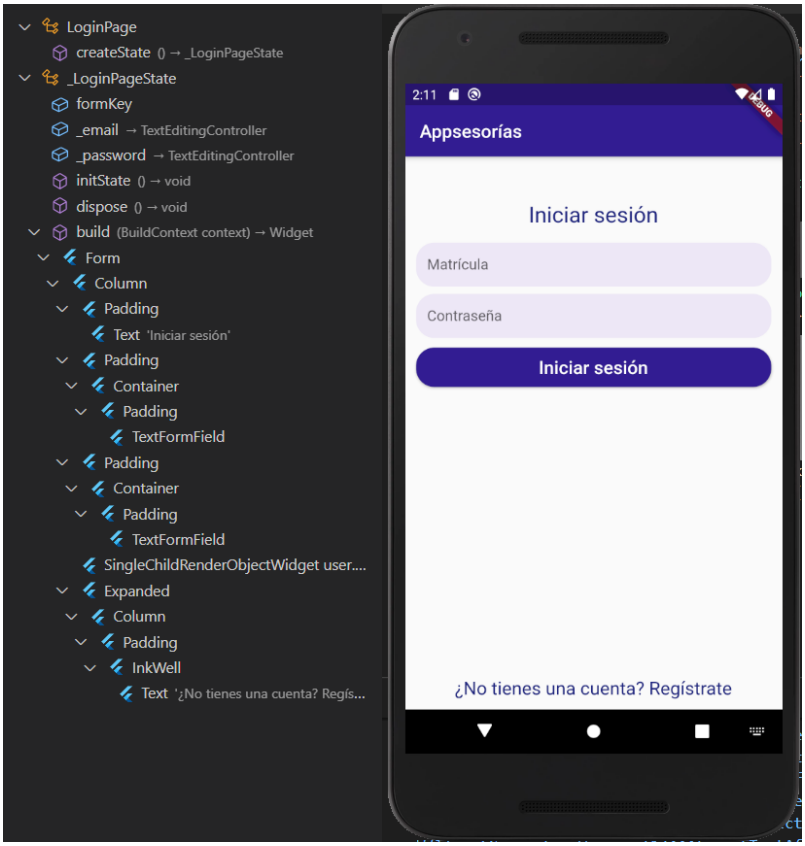

Figura 17
Inicio de sesión

Tras iniciar sesión, se valida que únicamente se muestren los apartados accesibles desde el tipo de perfil en cuestión. Cada uno de los módulos es accesible desde un menú lateral al cual se accede con el ícono ubicado en la esquina superior izquierda, para validar los apartados a mostrar se hace mediante operadores ternarios.

En la vista inicial, se muestran las asesorías que el perfil ha solicitado, para esto se realizo el componente en cuestión para mostrar brevemente la información relevante, dicho componente se reutiliza en diferentes vistas. En caso de presionar uno de estos elementos, se lleva a una vista detallada.

Asímismo, es posible crear un nuevo registro con el botón flotante en la esquina inferior derecha. Este botón flotante, la barra de navegación y el menú desplegable son elementos ya incluidos en Flutter y su configuración resulta muy sencilla y son ampliamente personalizables.

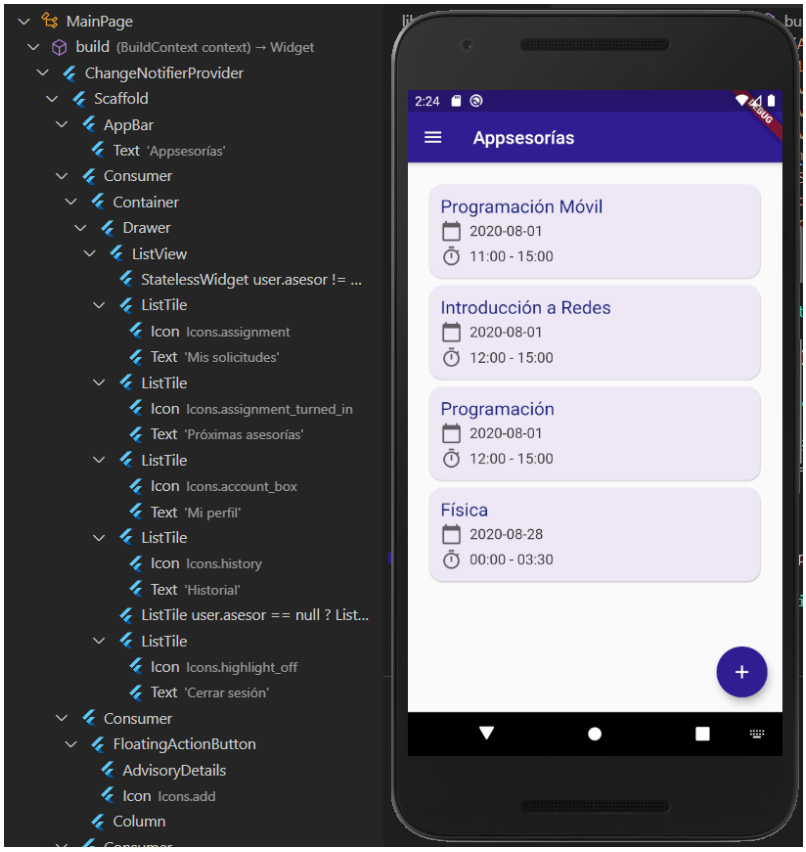

Figura 18
Vista inicial / Asesorías solicitadas

La vista de perfil muestra la información básica del perfil que inició sesión en la aplicación. Esta vista no requiere de ninguna llamada a la API ya que despliega la información que se obtuvo al iniciar sesión en la aplicación. Debido a esto, la vista es sumamente rápida e independiente de cualquier otra pantalla. Esta es una de las ventajas de manejar el estado de la aplicación con el paquete provider, ya que permite el acceso a la información de este mismo desde cualquier parte de la aplicación tras haber iniciado sesión.

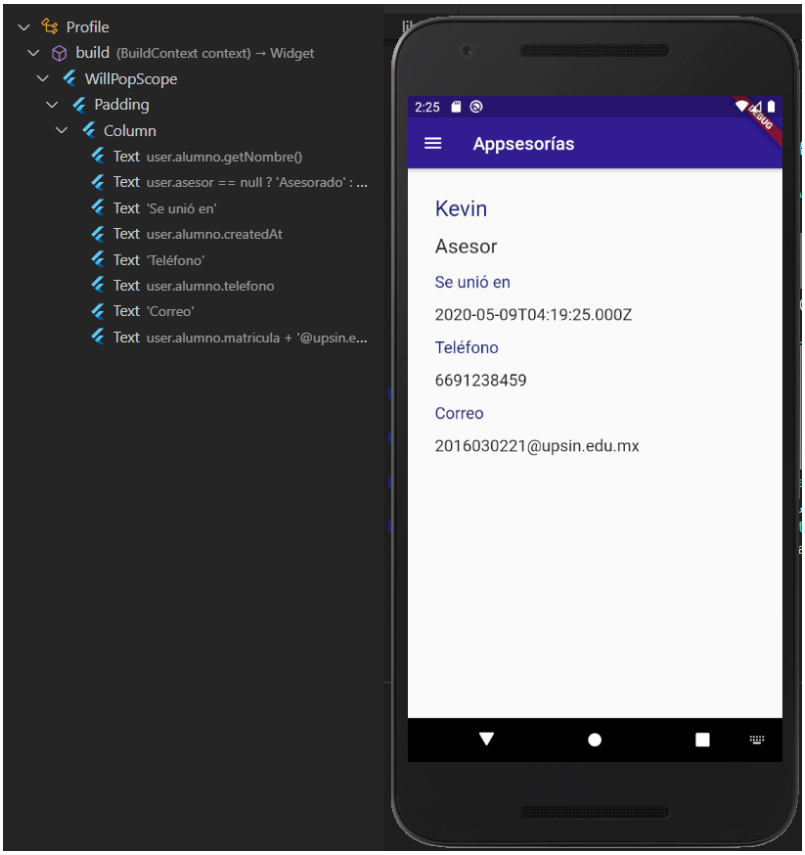

Figura 19
Perfil

La vista de asesorías en proceso únicamente es visible para un perfil de asesor. En esta vista se muestran las asesorías solicitadas y las asesorías a las que el asesor se ha registrado para impartirlas. En cuanto funcionalidad y diseño, se obtienen las listas en cuestión y se reutilizan los elementos de la vista de “Asesorías Solicitadas”.

Este es un ejemplo de como se pueden reutilizar elementos y funcionalidades ya programadas en diferentes apartados de la aplicación de manera sencilla y rápida. Para lograr esto, según las buenas prácticas, los *widgets* realizados deben ser lo más reutilizables posibles.

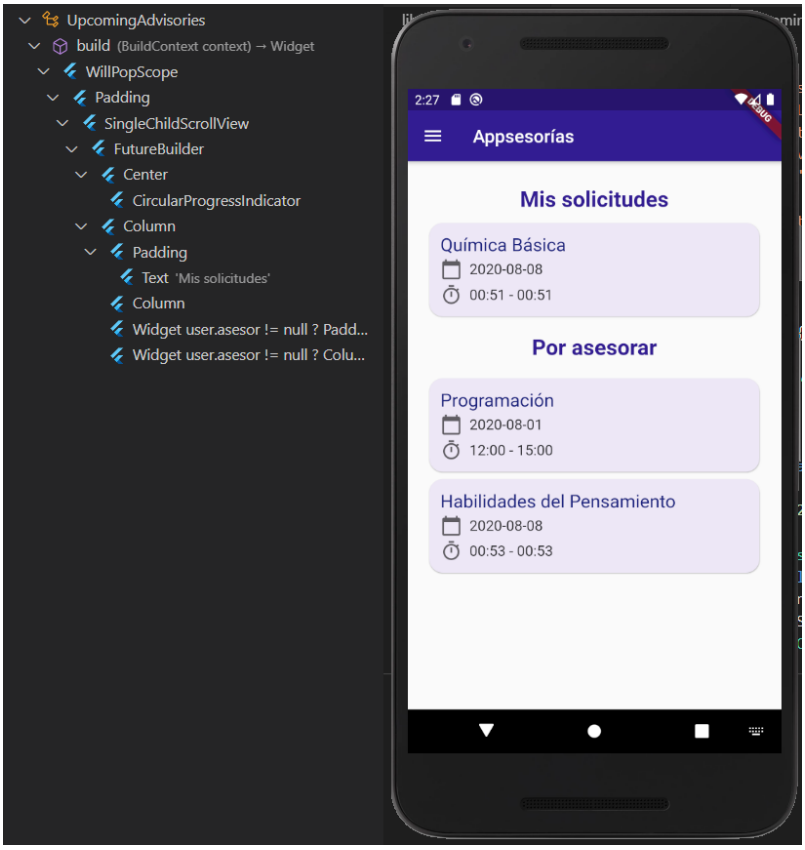

Figura 20
Asesorías en proceso

La vista de formulario para crear una asesoría consta de un formulario con los campos requeridos para solicitar la asesoría. Este es un componente que se reutiliza en muchas pantallas de la aplicación, siendo estas: crear asesoría, modificar asesoría, ver asesoría y asesorar asesoría. La vista recibe ciertos parámetros claves y funciones para el botón inferior para determinar que funcionalidades tendrá dicha vista.

Asímismo, se valida su apariencia dependiendo si el perfil dueño la visualiza, si un asesorado ajeno la visualiza o si alguien con un perfil de asesor la observa.

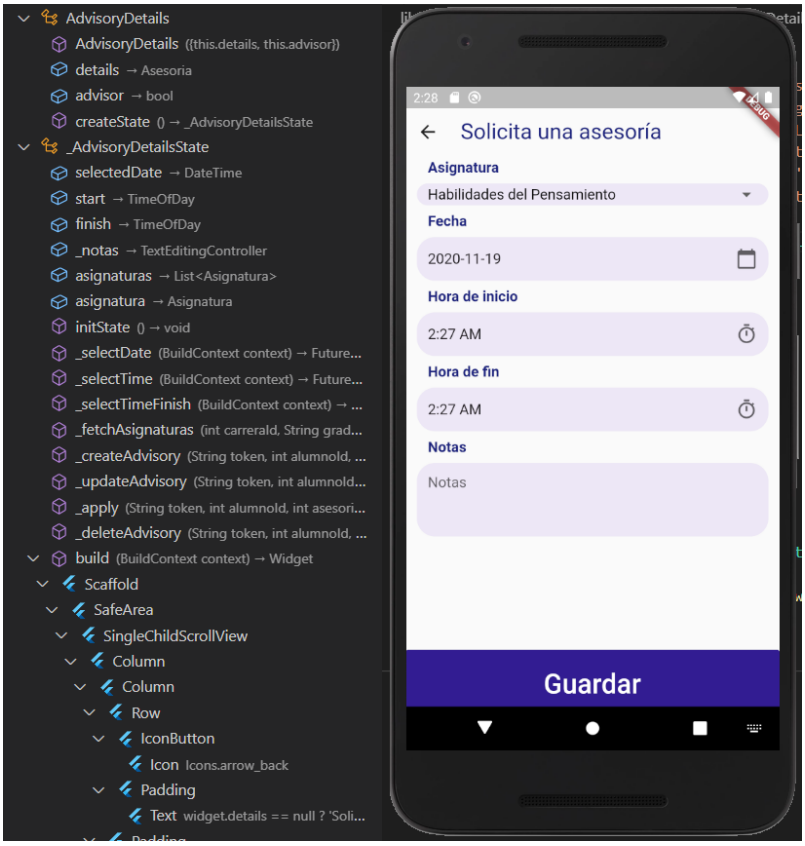

Figura 21
Solicitar asesoría

3.5 Resultados y discusión

El resultado final es una aplicación móvil, la cual es posible compilar para iOS y Android de manera sencilla, dicha aplicación mantuviera el mismo diseño en ambas plataformas (a menos que se configure para respetar la estética del sistema operativo) y accediera a los servicios propios de su respectivo SO como lo es el teclado.

De esta menera, se tiene la posibilidad de generar dos aplicaciones con un mismo código fuente sin requerir cambios en el código para lograr esto. Todo esto obtenido en el tiempo de un desarrollo único, el cual a su vez fue rápido gracias a las harramientas proporcionadas por Flutter.

Teniendo finalizada esta aplicación, se pueden destacar las siguientes ventajas de utilizar Flutter:

- Extenso catálogo de widgets predefinidos.

- Amplia y entendible documentación.
- Alto desempeño.
- Fácil aprendizaje.
- Reutilización de código.

Y teniendo como desventaja el lenguaje Dart, ya que dicho lenguaje es muy poco utilizado pese al soporte de Google sobre el mismo, no obstante, es un lenguaje orientado a objetos que tiene la funcionalidad práctica de lenguajes del mismo paradigma y una sintáxis simple.

En cuanto al entorno de trabajo al momento de trabajar con Flutter, este cuenta con una extensa gama de herramientas que permiten analizar diferentes aspectos de la aplicación, tales como el diseño, inspección de componentes, rendimiento, velocidad de ejecución, debugging, visualización del árbol de elementos, entre otras. Todas estas funciones utilizando Visual Studio Code, el cual es un IDE ligero y gratuito.

En cuanto al mismo procedimiento con otras tecnologías como lo son React Native o Ionic, es posible desarrollar lo mismo con estas tecnologías, siendo la diferencia el lenguaje de programación y las herramientas o facilidades que tiene el entorno, siendo este último factor la mayor ventaja de Flutter con respecto a los demás, ya que al tener un catálogo con las funciones más comunes y siguiendo la línea gráfica de Material Design, es posible iniciar a desarrollar aplicaciones de manera ágil.

3.6 Conclusión

El desarrollo de aplicaciones móviles se encuentra en constante cambio y con una importancia en constante crecimiento, es por ello que es importante reducir los tiempos de desarrollos de una aplicación móvil mientras se mantiene la calidad y eficiencia de la misma, logrando así generar productos de valor con un menor costo.

Pero esta misma situación ha conllevado consigo la necesidad de adoptar nuevas tecnologías para lograr cumplir con esto, siendo aquí donde entra el framework Flutter, el cual pese a ser un framework joven al llevar pocos años en el mercado, ha logrado posicionarse como una tendencia tecnológica y cuyo uso se encuentra extendiéndose

en este momento, siendo posiblemente un framework de buen valor profesional en los años siguientes.

Por el punto mencionado anteriormente, es un buen momento para adentrarse en esta tecnología y darle un adecuado seguimiento, ya que se encuentra creciendo rápidamente y tiene a Google, una de las principales empresas tecnológicas del mundo, como responsable y promotor, hecho que casi garantiza un buen futuro para esta tecnología.

Tras desarrollar la aplicación mencionada, se puede confirmar la hipótesis propuesta en el presente trabajo de investigación, ya que al utilizar Flutter para el desarrollo de una aplicación se logró tener la posibilidad de utilizar el código fuente trabajado para la compilación en dos sistemas operativos diferentes, además de lograr el desarrollo en un periodo breve de tiempo gracias a la incorporación de metologías ágiles.

3.7 Referencias

[1] Nextline, «Mapa Corporativo y Estructura,» 31 Julio 2020. [En línea]. Available: https://drive.google.com/file/d/1bnInIMUYrFuPX-q-WpptxjlZuA1AfKNs/view?pli=1. [Último acceso: 9 Octubre 2020].

[2] Nextline, «Filosofia organizacional,» 21 Enero 2020. [En línea]. Available: https://docs.google.com/document/d/1tUTKbJE9gqMIhiF5G_UyqpTL53WxtbEeQWCdUGobEio. [Último acceso: 9 Octubre 2020].

[3] Nextline, «Nextline,» [En línea]. Available: https://nextline.mx/mission-and-vision. [Último acceso: 14 Noviembre 2020].

[4] Debitoor, «App móvil - ¿Qué es una app móvil?,» [En línea]. Available: https://debitoor.es/glosario/app-movil. [Último acceso: 1 Noviembre 2020].

[5] StatCounter, «statcounter GlobalStats,» [En línea]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201910-202010-bar. [Último acceso: 11 Noviembre 2020].

[6] Android, «Android,» [En línea]. Available: https://www.android.com/intl/es-419_mx/. [Último acceso: 19 Noviembre 2020].

[7] Android, «Developer Android,» [En línea]. Available: https://developer.android.com/guide/platform. [Último acceso: 18 Noviembre 2020].

[8] Google, «Android,» Google, [En línea]. Available: https://www.android.com/what-is-android/. [Último acceso: 11 Noviembre 2020].

[9] Lucideus, «Medium,» [En línea]. Available: https://medium.com/@lucideus/understanding-the-structure-of-an-ios-application-a3144f1140d4. [Último acceso: 18 Noviembre 2020].

[10] Gabit, «Gabit,» [En línea]. Available: http://www.gabit.org/gabit/?q=es/que-es-ios. [Último acceso: 11 Noviembre 2020].

[11] GSOFT, «GSOFT,» [En línea]. Available: https://www.gsoft.es/articulos/que-necesito-web-apps-app-nativa-o-app-hibrida/. [Último acceso: 12 Noviembre 2020].

[12] Oracle, «Oracle,» [En línea]. Available: https://www.oracle.com/mx/java/. [Último acceso: 19 Noviembre 2020].

[13] Kotlin, «Kotlin,» [En línea]. Available: https://kotlinlang.org/. [Último acceso: 19 Noviembre 2020].

[14] Sparklabs, «Sparklabs,» [En línea]. Available: https://www.sparklabs.com.mx/tecnologias/objective-c/. [Último acceso: 19 Noviembre 2020].

[15] Apple, «Developer Apple,» [En línea]. Available: https://developer.apple.com/swift/. [Último acceso: 19 Noviembre 2020].

[16] J. Alarcón, «CampusMVP,» [En línea]. Available: https://www.campusmvp.es/recursos/post/Objective-C-o-Swift-Que-lenguaje-aprender-para-programar-en-iOS.aspx. [Último acceso: 12 Noviembre 2020].

[17] Flutter, «Flutter Dev,» [En línea]. Available: https://flutter.dev/docs/resources/architectural-overview. [Último acceso: 18 Noviembre 2020].

[18] A. Pedley, «BuildFlutter,» [En línea]. Available: https://buildflutter.com/how-flutter-works/. [Último acceso: 12 Noviembre 2020].

[19] React Native, «ReactNative,» [En línea]. Available: https://www.reactnative.guide/3-react-native-internals/3.1-react-native-internals.html. [Último acceso: 18 Noviembre 2020].

[20] Ionic, «Ionic,» [En línea]. Available: https://ionicframework.com/. [Último acceso: 19 Noviembre 2020].

[21] B. Repkins, «codecentric,» [En línea]. Available: https://blog.codecentric.de/en/2014/11/ionic-angularjs-framework-on-the-rise/. [Último acceso: 18 Noviembre 2020].

[22] Sugat, «Medium,» [En línea]. Available: https://medium.com/@sgt.shakya/flutter-vs-react-native-vs-ionic-which-one-will-be-good-for-your-application-f53338633e37. [Último acceso: 12 Noviembre 2020].

[23] Git, «Git --everything-is-local,» [En línea]. Available: https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones. [Último acceso: 19 Noviembre 2020].

[24] Fundación Wikimedia, Inc, «Wikipedia,» [En línea]. Available: https://es.wikipedia.org/wiki/Git. [Último acceso: 19 Noviembre 2020].

[25] Tena, María, «BBVA,» [En línea]. Available: https://www.bbva.com/es/metodologia-agile-la-revolucion-las-formas-trabajo/. [Último acceso: 19 Noviembre 2020].

[26] Proyectos Ágiles, «proyectos agiles,» [En línea]. Available: https://proyectosagiles.org/que-es-scrum/. [Último acceso: 19 Noviembre 2020].

[27] Red Hat, «Red Hat - Qué son las API y para que sirven?,» [En línea]. Available: https://www.redhat.com/es/topics/api/what-are-application-programming-interfaces. [Último acceso: 22 Noviembre 2020].

[28] Flutter, «Flutter,» [En línea]. Available: https://flutter.dev/docs/get-started/install/windows. [Último acceso: 16 Noviembre 2020].

[29] Flutter, «Flutter,» [En línea]. Available: https://flutter-io-deploy-one.firebaseapp.com/getting-started/. [Último acceso: 16 Noviembre 2020].

[30] D. G. Verdugo, «Desarrollo de aplicaciones híbridas con Ionic,» [En línea]. Available: https://solidgeargroup.com/desarrollo-de-apps-hibridas-con-ionic/. [Último acceso: 1 Noviembre 2020].

3.9 Glosario

Framework: Un entorno de trabajo es una estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de *software*, que puede servir de base para la organización y desarrollo de software.

SDK: Software Development Kit.

HTML: HyperText Markup Language.

CSS: Cascading Style Sheets.

WebView: Componente que permite a las aplicaciones mostrar contenido web.

Shell: Intérprete de comandos.

GPU: Graphics processing unit

CPU: Central processing unit

FPS: Frames per second

IDE: Integrated Development Environment

HTTP: Hypertext Transfer Protocol

UI: User Interface. Interfaz de usuario