UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE CUENCA

CARRERA DE TELECOMUNICACIONES

DESARROLLO DE UNA APLICACIÓN MÓVIL PARA MONITOREO DE TRANSPORTE DE ESTUDIANTES

Trabajo de titulación previo a la obtención del título de Ingeniera en Telecomunicaciones

AUTOR: EVELYN MICHELLE ILLESCAS ORDOÑEZ

TUTOR: ING.JUAN PAUL INGA ORTEGA, MgT.

Cuenca – Ecuador

2024

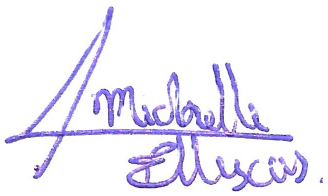
CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Yo, Evelyn Michelle Illescas Ordoñez con documento de identificación N° 0107985269; manifiesto que:

Soy la autora y responsable del presente trabajo; y, autorizo a que sin fines de lucro la Universidad Politécnica Salesiana pueda usar, difundir, reproducir o publicar de manera total o parcial el presente trabajo de titulación.

Cuenca, 24 de julio de 2024

Atentamente,



Evelyn Michelle Illescas Ordoñez
0107985269

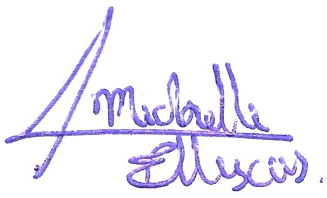
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Yo, Evelyn Michelle Illescas Ordoñez con documento de identificación N° 0107985269, expreso mi voluntad y por medio del presente documento cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autor del Proyecto Técnico: "Desarrollo de una aplicación móvil para monitoreo de transporte de estudiantes", el cual ha sido desarrollado para optar por el título de: Ingeniera en Telecomunicaciones, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribo este documento en el momento que hago la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, 24 de julio de 2024

Atentamente,



Evelyn Michelle Illescas Ordoñez
0107985269

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Juan Paúl Inga Ortega con documento de identificación N° 0104166491, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: DESARROLLO DE UNA APLICACIÓN MÓVIL PARA MONITOREO DE TRANSPORTE DE ESTUDIANTES, realizado por Evelyn Michelle Illescas Ordoñez con documento de identificación N° 0107985269, obteniendo como resultado final el trabajo de titulación bajo la opción proyecto técnico que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

Cuenca, 24 de julio de 2024

Atentamente,

Firmado electrónicamente por:
JUAN PAUL INGA ORTEGA

Ing. Juan Paúl Inga Ortega, MgT.
0104166491

AGRADECIMIENTOS

Primeramente, doy gracias a Dios, quien ha forjado mi camino y me ha dirigido por el sendero correcto, siendo mi guía y fortaleza a lo largo de mi formación académica.

A mi mamá Gladys, que con su amor incondicional, fortaleza y dedicación incansable siempre estuvo ahí. Por ser el pilar más importante de mi vida, y cada logro que alcanzo es un reflejo de su sacrificio y esfuerzo. Gracias por ser mi guía, mi apoyo y mi inspiración en cada paso del camino. Esta tesis es tan suya como mía, y no habría sido posible sin su inquebrantable fe en mí.

A mi adorada hermana Talia, por ser mi compañera de vida, mi confidente y mi amiga. En los momentos de alegría y en los retos, siempre has estado a mi lado, dándome su apoyo incondicional. Su fuerza y espíritu me motivan a ser mejor cada día. Gracias por ser mi roca y por creer en mí incluso cuando dudaba.

A mi papá José, que a pesar de la distancia física, siempre ha estado presente con su amor, apoyo y palabras de aliento. Gracias por tus sabios consejos, por tus constantes ánimos y por demostrarme que la distancia nunca disminuye el amor de un padre. Su apoyo ha sido fundamental para llegar hasta aquí.

A mi abuelo Luis, por sus sabias enseñanzas, su infinita paciencia y su infinito amor. Su presencia en mi vida ha sido un faro que me ha guiado y un ejemplo de vida, una inspiración constante.Por estar siempre dispuesto a compartir su sabiduría y por ser una figura tan importante en mi desarrollo personal y académico.

A mi abuelita Esther, que ahora descansa en el cielo, sus abrazos y sus palabras de aliento siempre estarán presentes en mi corazón. Aunque ya no este físicamente, su espíritu y su amor continúan guiándome.

A mi tío David y primo Anthony, que siempre han estado a mi lado con su amor y

apoyo incondicional. Han sido una figura clave en mi vida, dándome no sólo orientación y consejos, sino también un afecto genuino que siempre he valorado. Su presencia constante y su forma especial de tratarme han sido una fuente de fortaleza. Por estar siempre disponible para escucharme y por ser una influencia positiva en mi vida.

Finalmente, agradezco a todos los docentes que han sido parte de mi formación académica, por transmitirme sus conocimientos a lo largo de este camino. En especial a mi docente tutor Ing. Juan Inga, por su invaluable guía, su paciencia y su apoyo durante la realización de este trabajo de titulación. Gracias por compartir su conocimiento, por sus consejos acertados y por estar siempre dispuesto a ayudarme. Su dedicación y tu compromiso han sido fundamentales para mi crecimiento académico y profesional.

DEDICATORIA

Dedico este trabajo de titulación a mi padres Gladys y José, por su amor incondicional y por ser el pilar fundamental de mi vida. A mi hermana Talia, por su apoyo constante y por ser mi compañera en todas las etapas de la vida. A mi abuelito, por sus sabias enseñanzas y su amor inagotable. A mi abuelita en el cielo, que a pesar de no estar físicamente su espíritu y enseñanzas siguen vivos en mi y me han dado la fuerza para llegar hasta aquí y a todas las personas que me han acompañado durante mi formación académica

Índice general

Agradecimientos I

Dedicatorias III

Índice General IV

Índice de figuras VIII

Índice de tablas IX

Resumen X

Abstract XI

Antecedentes 1

Justificación 3

Objetivos 4

Introducción 5

1. Fundamentación Teórica 8

1.1. Aplicaciones Móviles 8

1.1.1. Desarrollo de Aplicaciones Móviles 9

1.1.2. Desarrollo Híbrido 10

1.2. Backend 10

1.2.1. Servidor Web para la operación de Aplicaciones Móviles 11

1.2.2. Base de Datos . . . 12
1.2.3. Bakend como un servicio . . . 14
1.3. Frontend . . . 15
1.3.1. Plataformas de desarrollo . . . 16
1.4. Geolocalización para dispositivos móviles . . . 16
1.4.1. Sistemas Multiplataforma con Geolocalización . . . 18
1.5. Tecnología RFID . . . 18

2. Desarrollo del Backend y Frontend para la Aplicación Móvil Propuesta 20
2.1. Planificación del Maquetado de la Aplicación Desarrollada . . . 20
2.2. Desarrollo de la aplicación . . . 22
2.2.1. Maquetado de la Interfaz . . . 22
2.2.2. Configuración y actualización del entorno de trabajo . . . 25
2.2.3. Creación del proyecto . . . 28
2.2.4. Desarrollo del Frontend . . . 30
2.2.5. Desarrollo del Backend . . . 38
2.2.6. Integración de Geolocalización y Mapas . . . 45
2.2.7. Implementación del Sistema RFID con el ESP8266 . . . 47

3. Implementación y Pruebas de Operación de la aplicación móvil 50
3.1. Configuración Inicial del Aplicativo Móvil . . . 50
3.1.1. Configuración del Icono de la aplicación . . . 50
3.1.2. Instalación de la Aplicación en dispositivos Android . . . 51
3.2. Pruebas de operación: Módulo del Conductor . . . 52
3.2.1. Inicio de sesión . . . 52
3.2.2. Vista de la Interfaz Principal del Conductor . . . 53
3.2.3. Visualización de la Ruta . . . 55
3.3. Pruebas de operación: Módulos Representante o Padre de Familia . . . 58
3.3.1. Inicio de sesión . . . 58
3.3.2. Vista de Estado de Ruta . . . 59
3.3.3. Vista de Notificaciones . . . 61
3.4. Montaje del Sistema RFID . . . 62

3.5. Pruebas de Rendimiento de la Aplicación 63
3.6. Resultado del consumo de datos 64

4. Conclusiones, Recomendaciones y Trabajos Futuros 66
4.1. Conclusiones 66
4.2. Recomendaciones 67
4.3. Trabajos Futuros 68

Glosario 69

Referencias 73

Índice de figuras

2.1. Proceso de Operación del Sistema. 21
2.2. Prototipo de pantalla de Inicio de Sesion. 23
2.3. Perspectiva de la Aplicación para el padre de Familia y sus notificaciones. 24
2.4. Interfaz de la Aplicación para el conductor. 25
2.5. Verificar la instalación del SDK de Flutter. 26
2.6. Plugins en Android Studio. 27
2.7. Imagen del dispositivo. 27
2.8. Extensión de Flutter en VS Code. 28
2.9. Extensión de Dart en VS Code. 28
2.10. Extensión PlatformIO ID en VS Code. 28
2.11. Opciones para crear nuevo proyecto en Flutter. 29
2.12. Ejemplo de StatefullWidget. 31
2.13. Ejemplo del método Build. 32
2.14. Paquete de ejemplo en pub.dev. 33
2.15. Agregar el paquete mediante comando en la terminal. 33
2.16. Agregar el paquete de manera manual. 34
2.17. Ejemplo de importaciones de librerías. 37
2.18. Panel de configuración de Firebase. 39
2.19. Plan utilizado para el servicio de Cloud Messaging. 40
2.20. Inicializar Firebase en main.dart. 41
2.21. Agregar pluging de google service en seccion dependencias. 41
2.22. Agregar pluging de google service. 41
2.23. Directorio de google-services.json. 42
2.24. Diagrama UML de la base de datos para la Aplicación. 42

2.25. Ejemplo uso Streams. 43
2.26. Habilitar el método de autenticación por correo electrónico y contraseña. 44
2.27. Integracion de autenticación para inicio de sesión. 44
2.28. Inicializar Firebase y configurar FCM. 45
2.29. Función para envío y recepción de notificaciones 45
2.30. Obtención de ubicacion del conductor. 46
2.31. Cálculo de Distancia entre ubicacines. 46
2.32. Duración estimada de viaje. 47
2.33. Circuito. 48
2.34. Configuración para la conexión de Wi-Fi y Firebase. 49

3.1. Configuración del icono. 51
3.2. Instalación apk en Android. 52
3.3. Interfaz de Inicio de Sesión del conductor. 53
3.4. Vista Interfaz Principal del Conductor. 54
3.5. Mensaje de Confirmación para Iniciar la Ruta. 55
3.6. Pagina para la opción ruta de Retorno. 56
3.7. Ruta Escolar. 57
3.8. Pantalla Inicio de Sesión Estudiante. 59
3.9. Pantalla Cuando la Ruta Inicia. 60
3.10. Pantalla cuando la Ruta Finaliza. 61
3.11. Notificaciones Recibidas. 62
3.12. Ensamblaje Fisico del Sistema RFID. 63
3.13. Análisis de rendimiento de la aplicación utilizando Flutter DevTools 64

Índice de tablas

1. Resumen de los trabajos relacionados con el Desarrollo móvil para el transporte de estudiantes. 6

2.1. Esquema de Conexiones.Fuente: El Autor. 49

3.1. Consumo de datos del módulo ESP8266 en distintos intervalos de tiempo. 64

Resumen

El presente trabajo tiene por objeto el desarrollo de una aplicación móvil que contribuya a mejorar la seguridad y eficiencia del transporte escolar mediante tecnologías modernas como GPS y RFID. Esta aplicación, desarrollada e implementada con Flutter, permite el monitoreo del transporte de estudiantes en tiempo real, facilitando a los padres de familia conocer la ubicación exacta de la unidad de transporte escolar tanto en el viaje hacia la institución educativa como en el viaje de retorno. Además, la aplicación permite a los padres notificar al conductor si harán uso o no del sistema de transporte escolar. Como resultado de este proyecto, se ha desarrollado tanto el frontend como el backend de la aplicación móvil, integrando diversos servicios proporcionados por Firebase para garantizar un rendimiento óptimo y una experiencia de usuario fluida. La aplicación se compone de dos interfaces principales: una para los conductores y otra para los padres de familia. El módulo del conductor permite visualizar la ubicación de las paradas de los estudiantes que utilizan su servicio, enviar notificaciones a los padres sobre el inicio y finalización de la ruta, y registrar cuándo el estudiante sube o baja del transporte mediante tarjetas RFID. Por su parte, los padres pueden recibir notificaciones instantáneas del estado del viaje escolar, seguir la ruta del conductor en tiempo real, confirmar la asistencia del estudiante y ver el tiempo estimado de llegada del transporte. Además, la aplicación ofrece un historial de viajes y notificaciones pasadas, lo que permite a los padres revisar la información anterior en caso de necesidad.

Palabras clave: BACKEND; FRONTEND; APP; FIREBASE; NOTIFICACIONES PUSH; GEOLOCALIZACION; RFID.

Abstract

Several educational institutions need help assigning students to their transportation units due to a lack of management or time for administration, and the problem becomes more complicated when the number of units and students increases. The problems arise when student representatives or parents must hire transportation since the manual assignment of students to transportation units only sometimesually optimize the scheduling or routes. This results in more extended pick-up and drop-off times. In addition, if there is no transportation service coordinator, representatives must manage the service directly with the drivers, which can be a complicated and disorganized process.

To address these problems, we propose developing a specialized program to optimize the students' assignments and calculate school transportation routes. This software will use optimization and clustering algorithms to improve resource allocation and route planning in Cuenca - Ecuador. The program reads data from Uben Excel databases, including information on students, drivers, and transportation units. It integrates geolocation functions to map locations and define optimal routes based on distance and time. In addition, it has an intuitive graphical interface to visualize routes on a geo-referenced map. One of the key features of our proposed software is its adaptability. It can easily accommodate changes in the database and automatically recalculate routes to the destination point. This flexibility ensures that the software remains effective and efficient, even in the face of evolving circumstances.

Keywords: BACKEND; FRONTEND; APP; FIREBASE; PUSH NOTIFICATIONS; GEOLOCALIZATION; RFID.

Antecedentes

En el Ecuador, es común que las unidades educativas cuenten con el servicio de transporte escolar para sus estudiantes y algunos padres de familia o representantes se ven en la necesidad de acudir a esta alternativa, ya que su trabajo u otras actividades no les permiten llevar o traer a sus representados.

Por otra parte, la Federación de Transporte Escolar del Ecuador mostraba preocupación por la seguridad de los estudiantes y conductores debido a las diversas situaciones de delincuencia que estaba viviendo el país. Esto se debe a casos como el que dió en julio del 2022, diario el Expreso informaba el caso en que malhechores suben al medio de transporte para robar las pertenencias de los estudiantes y como consecuencia de estos actos vandálicos hubo un estudiante herido y un conductor del expreso escolar fue asesinado [1]. En otro caso, el diario el Comercio indicaba que hay instituciones educativas secundarias que ya están proponiendo el uso de aplicaciones móviles para rastreo de los estudiantes, no obstante, estaban diseñadas para los estudiantes de niveles superiores, ya que la aplicación requiere que cada estudiante cuente con un dispositivo móvil [2].

Debido a estas situaciones, surge la preocupación de los padres de familia o representantes, ya que, durante el viaje de los estudiantes, los padres no poseen información sobre la localización del transporte, sí el estudiante abordó el medio de transporte, el sitio en donde se encuentra el mismo, si el estudiante fue dejado en la unidad educativa o incluso la hora de llegada del estudiante en su destino [1][2].

Otro de los problemas que surgía por parte de los representantes es la falta de conocimiento de la hora de llegada de sus representados, debido a que el medio de transporte no cuenta con una hora exacta para llegar a su destino, esto puede darse a causa de los retrasos en el camino como el tráfico en la ciudad, además de que este

cuenta con varios usuarios donde cada uno de ellos es transportado a diferentes sitios asignados por su representante [1].

Justificación

En los últimos años, los dispositivos móviles han experimentado un notable avance, impulsado en gran medida por las diversas características tecnológicas que incorporan y las distintas aplicaciones disponibles para los usuarios. Estas aplicaciones facilitan múltiples procesos y mejoran la localización de los usuarios.

En el contexto de los transportes escolares, el uso de tecnología puede mejorar significativamente la experiencia del usuario del servicio. Los padres de familia a menudo se preocupan cuando sus hijos utilizan este medio de transporte debido a problemas como la falta de información sobre la ubicación de sus hijos, si han llegado a su destino o la hora de llegada en sus paradas correspondientes. Estos retrasos pueden ser causados por el tráfico o por las rutas seleccionadas. Además, los autobuses escolares suelen tener muchos usuarios y deben pasar por diversas localizaciones, lo que añade otra capa de complejidad.

En virtud a lo mencionado en los antecedentes y como solución a esta problemática, este trabajo busca desarrollar una aplicación móvil que permita el monitoreo del transporte de estudiantes en donde esta notifique a los representantes o padres de familia de los estudiantes el instante en el que cada uno de estos suba o baje del transporte escolar, además de esto que permita un seguimiento de la ruta que recorre el medio de transporte y que notifique el tiempo estimado de llegada a la escuela o la parada destino del estudiante.

Objetivos

Objetivo General

- Desarrollar una aplicación móvil para el monitoreo de estudiantes en el transporte escolar

Objetivos específicos:

- Prototipar la operación de la aplicación móvil.
- Implementar un servidor Backend para la operación de la aplicación móvil.
- Implementar Frontend para la aplicación móvil que permita monitorear el transporte escolar por el cual viajan los estudiantes.
- Evaluar la operación de la aplicación desarrollada para verificar el funcionamiento del servidor de Backend y Frontend.

Introducción

Hoy en día, la mayoría de las unidades educativas ofrecen servicios de transporte escolar. Sin embargo, la falta de información en tiempo real sobre la ubicación y el estado de los estudiantes durante el traslado genera preocupación entre los padres. Además, los métodos convencionales de comunicación, como las llamadas telefónicas, son ineficientes y no proporcionan una solución integral para las necesidades de seguridad y monitoreo.

Existen trabajos relacionados con el tema de este proyecto como es el caso de [3] que está enfocado en dar a conocer a los representantes/padres de familia, la ubicación del transporte escolar y el tiempo que tardará su representado en llegar a su destino tanto de ida como de vuelta. También, el trabajo de Illanes en [4] que también se enfoca en la ubicación del transporte escolar, además de un control de la asistencia de estudiantes en el que el conductor a través de un botón genera una notificación al padre de familia de que el estudiante subió a la unidad de transporte. El trabajo de Pinto en [5] propuso el uso de una base de datos de almacenamiento y consulta sobre la operación para el servicio del transporte para estudiantes universitarios de forma que, estos conozcan la ruta que está siguiente el conductor y cuánto tiempo falta para que llegue a su parada previo a recogerlos.

La tabla 1 un resumen comparativo de los trabajos similares revisados. Se identifica que, en la mayoría de estos, cuenta con ciertas características similares a lo que se propone en el presente trabajo; sin embargo, este trabajo se diferencia en la posibilidad de que el padre de familia o representante de un estudiante pueda notificar al conductor si el estudiante usará o no el sistema de transporte, de esta manera el conductor optimiza la ruta de transporte. También, a través de un nodo de identificación por Radio Frecuencia (RFID, del inglés *Radio Frequency Identification*)

ubicado en la unidad de transporte se envía una notificación de si el estudiante subió o bajo del transporte escolar sin que lo tenga que accionar el conductor en forma manual como sucede en otras aplicaciones.

Tabla 1: Resumen de los trabajos relacionados con el Desarrollo móvil para el transporte de estudiantes.

Referencia	Sistema de Geolocalización	Notificación manual si el estudiante subió o bajó del transporte escolar.	Tiempo promedio estimado para la llegada del medio de transporte	Confirmar si el estudiante va a tomar el medio de transporte .	Notificación automática si el estudiante subió o bajo del transporte escolar.	Novedades durante el viaje.	Monitoreo Padre de Familia.	Monitoreo Institución Educativa.
Autor								
(Pinto, 2021) [5]	✳	✳				✳	✳	
(Sánchez&Yvimas, 2018) [3]	✳	✳				✳	✳	
(Illanes Rumiguano, 2021) [4]	✳						✳	
(Vinueza Carranza, 2019) [6]	✳		✳				✳	
(Forda Nama et al., 2018) [7]	✳						✳	
Mi Buseta (app en el Mercado)	✳		✳			✳	✳	
OnTrack Road (app en el Mercado)	✳		✳			✳		✳
Propuesta de trabajo	✳		✳	✳	✳		✳	

Para abordar esta problemática, se plantea la implementación de una plataforma completa que incluye tanto el desarrollo del backend como del frontend de la aplicación móvil. La aplicación permitirá a los conductores visualizar la ubicación de los estudiantes en tiempo real, notificar a los padres sobre el inicio y finalización de la ruta, y registrar automáticamente cuando un estudiante sube o baja del transporte mediante tarjetas RFID. Por su parte, los padres podrán recibir notificaciones instantáneas, seguir la ruta del conductor en tiempo real, confirmar la asistencia del estudiante y ver el tiempo estimado de llegada del transporte.

También, la aplicación móvil desarrollada en el presente trabajo, usa Flutter para el monitoreo del transporte de estudiantes. Este proyecto surge como respuesta a la necesidad de mejorar la seguridad y eficiencia del transporte escolar, ofreciendo una solución tecnológica que permita a los padres y conductores gestionar y supervisar el traslado de los estudiantes de manera efectiva.

La aplicación se desarrolló utilizando Flutter, un kit de desarrollo de software

(SDK) que facilita la creación de aplicaciones móviles multiplataforma con acceso ágil a elementos nativos de los dispositivos[8], [9]. Para el manejo de datos en tiempo real y la gestión de notificaciones push, se utilizarán servicios en la nube proporcionados por Firebase.

Capítulo 1

Fundamentación Teórica

Este capítulo ofrece un resumen de los conceptos fundamentales para comprender el contexto del proyecto a implementar. Se abordan los diferentes tipos de desarrollo de aplicaciones móviles, las soluciones utilizadas para el desarrollo del *Backend* y Frontend, así como los requisitos para la base de datos y la interacción entre estos componentes para la operación de la aplicación móvil.

1.1. Aplicaciones Móviles

En la era digital actual, las aplicaciones móviles, o "apps", han experimentado un auge sin precedentes, modificando radicalmente nuestra forma de interactuar con la tecnología. Estas herramientas informáticas, diseñadas principalmente para smartphones (aunque no son los únicos dispositivos compatibles), se definen como softwares que pueden ejecutarse en diferentes sistemas operativos y descargarse desde diversas tiendas de aplicaciones.

Si bien muchas apps son gratuitas, esto no siempre implica que sean totalmente libres de costo. En ocasiones, los desarrolladores incluyen publicidad dentro de las aplicaciones gratuitas, o bien ofrecen funciones básicas en la versión gratuita y funciones premium en una versión paga [10].

1.1.1. Desarrollo de Aplicaciones Móviles

Una aplicación móvil es un programa de computadora diseñado y desarrollado para ser utilizada por el usuario en un dispositivo móvil como tabletas, TV inteligentes, teléfonos inteligentes, relojes inteligentes, entre otras. Las Apps móviles ayudan al usuario a realizar diversas tareas y en general tienen por objeto agilizar tiempo que se emplea para realizar una actividad; además, son usadas para gestión y control de eventos, etc [11].

Existen tres tipos de aplicaciones móviles y son las aplicaciones nativas, las aplicaciones híbridas o multiplataforma y las aplicaciones web. Sin importar el tipo de aplicación que se decida utilizar, estas deben brindar calidad de información a los usuarios que la utilizan [12], [9].

Desarrollo nativo

El desarrollo nativo se caracteriza por aprovechar al máximo las funcionalidades y características específicas de cada sistema operativo, utilizando lenguajes y tecnologías propias de la plataforma. Esto permite crear aplicaciones altamente optimizadas, que se integran perfectamente con el hardware y brindan una experiencia de usuario superior.[9].

Este enfoque de desarrollo se centra en lograr una compatibilidad total con dispositivos que poseen sistemas operativos (SO,proveniente del inglés operating system) propios. Esto permite alcanzar un equilibrio óptimo entre el hardware y el software, aprovechando al máximo las capacidades de los dispositivos y brindando al usuario una experiencia superior.[13].

Desarrollo Web

El desarrollo web se define como el proceso integral de creación y mantenimiento de sitios y aplicaciones web. Este campo abarca una amplia gama de actividades, desde la codificación y el diseño de la interfaz de usuario hasta la gestión de contenidos, la seguridad y la administración del servidor web.

Un aspecto fundamental del desarrollo web es su naturaleza accesible y

sencilla, ya que las herramientas necesarias para la programación web se encuentran ampliamente disponibles en línea, eliminando la necesidad de emuladores o dispositivos físicos especializados [9]. Esto permite que desarrolladores de todo el mundo puedan crear sitios web y aplicaciones web sin barreras significativas.

El panorama del desarrollo web ofrece una amplia gama de herramientas con características diversas que se adaptan a las necesidades específicas de cada usuario desarrollador. Estas herramientas, conocidas como *frameworks*, están diseñadas para facilitar el proceso de desarrollo y cumplir objetivos predefinidos. Además, cuentan con librerías integradas que permiten la creación de programas a gran escala y con alta escalabilidad [14].

1.1.2. Desarrollo Híbrido

Este entorno de desarrollo facilita la creación de aplicaciones co-nativas, las cuales se pueden generar a partir de un único código base. Para ello, existen herramientas especializadas en el desarrollo multiplataforma que preprocesan el código y generan aplicaciones completamente nativas [9].

Es decir, no son más que aplicaciones creadas para ser implementadas en diferentes sistemas operativos, evitando la tarea de desarrollar una aplicación para cada sistema operativo. De esta manera, una aplicación híbrida puede ser adaptada a diferentes plataformas sin generar nuevos códigos, pero ajustando cambios operacionales para cada uno de ellos[15].

De acuerdo con lo mencionado más arriba y considerando lo expuesto por Yazbek [9] en donde se expone que las aplicaciones híbridas brindan una ventaja en el tiempo de desarrollo y capacidad para acceder a elementos nativos del dispositivo móvil, este trabajo se despliega haciendo uso de Flutter.

1.2. Backend

Si imaginamos el desarrollo de aplicaciones móviles como la construcción de una casa, el backend sería todo lo que está detrás de las paredes y hace que la casa funcione a la perfección, como la plomería, la electricidad y el sistema de calefacción.

Esta analogía es comúnmente utilizada por los desarrolladores de aplicaciones para explicar la función crucial del *Backend* [16], [17]. El término"*Backend*" se refiere a una capa de software que gestiona el acceso a los datos y permanece invisible o inaccesible para el usuario final. Esta capa es la responsable de la funcionalidad, la seguridad y la optimización de recursos de una aplicación o sitio web. Además, alberga toda la lógica interna que garantiza el correcto funcionamiento de la aplicación, incluyendo la base de datos que almacena los datos en el servidor [18].

En el ámbito del desarrollo de aplicaciones móviles, el *Backend* se refiere a la parte del servidor de la aplicación que se encarga de almacenar y administrar datos, generar la autenticación de usuarios, procesar pagos y proporcionar otras funcionalidades esenciales. Tradicionalmente, los desarrolladores debían crear y mantener su propio *Backend*, lo que podía resultar un proceso complejo y lento [16].

El *Backend* también permite ejecutar o implementar procesos que respaldan el desarrollo del sistema, tales como: las operaciones lógicas, la conectividad de bases de datos, la optimización de recursos y gestión de archivos. Para optimizar la eficiencia del proceso, en el *Backend* se utilizan diversas librerías que realizan funciones como la compresión o previsualización de imágenes.

El *Backend* admite diversos lenguajes de programación del lado del servidor, como Java, PHP, C y Node.JS. Además, existen diferentes tipos de bases de datos relacionales, como MySQL, PostgreSQL y SQL Server, mientras que MongoDB destaca entre las bases de datos no relacionales [18]).

1.2.1. Servidor Web para la operación de Aplicaciones Móviles

Un servidor web se define como un sistema integral que alberga servicios web. Estos servicios se basan en el uso de URL como elementos centrales, lo que facilita a los sitios web y aplicaciones la gestión de los datos de cada usuario mediante el protocolo HTTP [19].

Manejo de Información

Al interactuar con una página web, ya sea al enviar un formulario o realizar una búsqueda, se establece una comunicación entre el navegador del usuario y el

servidor web. Esta interacción se lleva a cabo mediante solicitudes HTTP, que son mensajes que el navegador envía al servidor para solicitar información o realizar acciones específicas. Los componentes de una solicitud HTTP son:

- El localizador uniforme de recursos (URL por sus siglas en ingles *Uniform Resource Locator*) es la dirección única que identifica el recurso web al que se desea acceder. El servidor utiliza la URL para determinar qué página o archivo debe enviar al navegador [19].
- El método HTTP indica la acción que el usuario desea realizar sobre el recurso solicitado. Los métodos HTTP más comunes son:

  GET: Recupera un recurso específico del servidor.

  POST: Envía datos al servidor para crear o actualizar un recurso.

  HEAD: Obtiene metadatos de un recurso, como la fecha de última modificación, sin descargar el contenido completo del mismo. [19].

  PUT: Actualiza un recurso existente en el servidor.

  DELETE: Elimina un recurso específico del servidor.

  TRACE, OPTIONS, CONNECT, PATCH: Estos métodos son para realizar acciones menos comunes o avanzadas, por lo cual no se detallará aquí.

1.2.2. Base de Datos

La información se define como un conjunto de datos que, tras ser analizados, permiten la toma de decisiones. Esta información se procesa y presenta de manera clara para facilitar su interpretación, revelando tendencias o patrones ocultos [20].

En este contexto, una base de datos se define como un conjunto de información, relacionada o no, que puede ser recuperada posteriormente. Estas bases de datos se

almacenan en archivos computarizados ubicados en diferentes sitios, lo que permite el intercambio de información entre distintos usuarios [20].

Base de Datos Relacionales

Las bases de datos relacionales (SQL, del inglés *Structured Query Language*), permiten establecer interconexiones entre diferentes tipos de datos almacenados en tablas. Cada tabla en la base de datos se compone de filas y columnas. Las filas representan registros individuales y cada fila tiene una clave principal denominada *primary key* que la identifica de manera única. Las columnas representan los atributos de los datos y cada registro tiene un valor para cada uno de estos atributos [21].

Esta estructura permite definir relaciones entre las tablas mediante el uso de claves foráneas denominadas *foreign keys*, que son referencias a las claves principales de otras tablas. De esta manera, se pueden establecer relaciones entre los datos, facilitando la organización y el acceso eficiente a la información [21].

El lenguaje de consulta estructurada SQL es un estándar para trabajar con bases de datos relacionales y se utiliza como un lenguaje declarativo de la estructura de una base de datos, lo que permite a los programadores administrar y extraer información de los datos mediante la aplicación de cláusulas y sentencias [21].

Base de Datos no Relacionales

Las bases de datos no relacionales (NoSQL, del inglés *No Structured Query Language*), son ideales para modelos de datos y esquemas flexibles, lo que facilita la creación de aplicaciones sin la necesidad de definir relaciones estrictas entre tablas. En lugar de organizar la información en tablas, estos sistemas utilizan estructuras como archivos basados en notación de JavaScript (JSON, del inglés *JavaScript Object Notation*) para almacenar los datos. Este enfoque es popular debido a su capacidad para permitir un desarrollo ágil, funcionalidad robusta y un rendimiento eficiente a cualquier escala.[10].

NoSQL emplea diversos modelos de datos para gestionar y acceder a la información, y está especialmente optimizado para aplicaciones que manejan grandes volúmenes de datos, requieren baja latencia y demandan esquemas de datos flexibles.

Esto se logra mediante la reducción de ciertas restricciones de coherencia de datos presentes en las bases de datos tradicionales escala [10].

En este proyecto, se utiliza Firebase, una base de datos NoSQL de documentos, para gestionar y almacenar datos en formato JSON, lo que se adapta perfectamente a la estructura dinámica de la información en la aplicación móvil.

1.2.3. Bakend como un servicio

El uso de *Backend* como un Servicio (BaaS, del inglés *Backend as a Service*) proporciona a los desarrolladores acceso a un backend pre-construido y administrado y se trata de un modelo de computación en la nube. Esto significa que el desarrollador no se preocupa por configurar y mantener servidores, bases de datos u otra infraestructura de backend. En cambio, se puede concentrar en lo que realmente importa: crear la mejor experiencia de usuario posible para tu aplicación [22].

Un caso de uso muy desplegado es el de Firebase de google que ofrece una suite completa de herramientas para el desarrollo de aplicaciones móviles. Proporciona una amplia gama de servicios, que incluyen Backend as a service in web development:

- Bases de datos: Firebase ofrece varias opciones de bases de datos, como Cloud Firestore (una base de datos NoSQL) y Realtime Database (una base de datos NoSQL en tiempo real), que te permiten almacenar y administrar datos de forma escalable y segura .

- Autenticación: Firebase Authentication facilita la implementación de un sistema de autenticación seguro para tu aplicación, permitiendo a los usuarios registrarse, iniciar sesión y acceder a sus datos de forma segura[23].

- Almacenamiento de archivos: Firebase Storage permite almacenar y administrar archivos de forma segura en la nube, lo que resulta ideal para almacenar imágenes, videos y otros datos binarios[23].

- Funciones en la nube: Firebase Functions te permite ejecutar código en la nube en respuesta a eventos, como cambios en la base de datos o solicitudes de usuarios. Esto te permite crear aplicaciones más dinámicas y escalables.

- Análisis: Firebase Analytics proporciona herramientas para comprender el comportamiento de los usuarios y el rendimiento de tu aplicación. Permite recopilar datos sobre cómo usan los usuarios tu aplicación, identificar áreas de mejora y tomar decisiones informadas basadas en datos.

1.3. Frontend

El *frontend* es el lado visible de una aplicación móvil, es decir, la interfaz con la que el usuario interactúa de manera directa. Por ello, su diseño debe ser amigable e intuitivo, con el objetivo de brindar una experiencia de usuario satisfactoria.

Es fundamental que los desarrolladores de *frontend* posean conocimientos sólidos en técnicas y principios de diseño de interacción con el usuario, ya que les permite crear interfaces fáciles de usar, navegar y comprender, optimizando la experiencia del usuario.

La colocación estratégica de los elementos dentro de la interfaz es crucial para facilitar la navegación y la interacción del usuario. Los desarrolladores deben considerar aspectos como la accesibilidad, la jerarquía visual y la distribución espacial de los elementos para crear una interfaz intuitiva y fluida.

El *frontend* debe propiciar una interacción natural y fluida entre el usuario y la aplicación. Esto se logra mediante el uso de elementos como botones, menús, formularios y otros componentes interactivos que respondan de manera rápida y precisa a las acciones del usuario.

El *frontend* no solo se trata de funcionalidad, sino también de estética. La combinación armoniosa de elementos como tipografías, colores, imágenes y animaciones crea una interfaz atractiva y agradable a la vista, lo que contribuye a una experiencia de usuario positiva. [24].

El desarrollo de *frontend* actual ofrece una amplia gama de lenguajes de programación y herramientas que permiten crear interfaces modernas, dinámicas y adaptables. Entre los lenguajes más utilizados se encuentran:JavaScript, CSS, HTML, Dart, Java. Para la comunicación entre el frontend y el backend de una aplicación, se utilizan lenguajes de transferencia de datos como:XML, JSON y Ajax

1.3.1. Plataformas de desarrollo

Framework Flutter

Flutter es un framework de código abierto desarrollado por Google que se utiliza para construir aplicaciones móviles multiplataforma, lo que significa que se puede desarrollar aplicaciones para iOS y Android desde una sola base de código, ya que este cuenta con un desarrollo Multiplataforma. Ademas de que flutter se centra en la creación de interfaces de usuario (UI) y proporciona un conjunto completo de widgets personalizados para construir interfaces atractivas y funcionales.

Flutter utiliza **Dart** como lenguaje de programación principal, el cual es un lenguaje moderno y orientado a objetos desarrollado por Google [25].

Android Studio

Android Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) creado específicamente para el desarrollo de aplicaciones Android. Es la herramienta oficial de Google para el desarrollo de aplicaciones móviles en la plataforma Android[25]

Visual Studio Code

Visual *Studio Code*, o VS Code como se le conoce comúnmente, es un editor de código fuente gratuito y de código abierto desarrollado por *Microsoft*. Se ha convertido en una herramienta popular entre los desarrolladores de software debido a su versatilidad, extensibilidad y facilidad de uso. VS Code está disponible para múltiples plataformas de Windows, macOS y Linux [25].

1.4. Geolocalización para dispositivos móviles

Un dispositivo móvil dispone de diferentes vías para determinar su ubicación, siendo algunas mucho más precisas que otras. Sin embargo, en ocasiones al dispositivo no podrá utilizar la técnica correcta y deberá recurrir al método más conveniente según el dispositivo. Esta disponibilidad que está conectado el

dispositivo, y dependiendo de la marca del operador, los métodos geolocalización se dividen en cuatro grupos [26].

- ***Redes Wi-Fi***

  Este método se basa en el uso de bases de datos que contienen la información y situación de gran cantidad de redes Wi-Fi. El método se basa en enviar la dirección MAC de su enrutador Wi-Fi y el SSID (nombre la red) y compararlo con la base de datos que devolverá la ubicación geográfica de la red. De esta manera es posible determinar la ubicación de cualquier dispositivo conectado a la red inalámbrica con una precisión de 30 a 100 metros [26].

- ***Redes de telefonía móvil***

  Existen varios métodos de georreferenciación que permiten determinar la ubicación de un dispositivo que esté conectado a una red de telefonía móvil con una precisión que oscila entre los 50 y los 500 metros. A continuación, se presentará las tres técnicas más relevantes.

  Basados en la red: Estos sistemas utilizan sistemas del proveedor de servicios para localizar el terminal, por lo que no se necesita ninguna aplicación especial para ejecutar en el teléfono móvil. El problema principal de este sistema es que necesita estar cerca del proveedor para que funcione [26].

  Basados en el terminal: Los dispositivos que utilizan estos sistemas poseen un receptor de señales y un software de cliente para determinar la ubicación del terminal a través de las señales externas. Cabe señalar que es necesario instalar una aplicación en un dispositivo, haciendo que el funcionamiento dependa de la adaptación de los diferentes sistemas operativos [26].

  Híbridos: Es una combinación del sistema terminal y el sistema de red. Aunque contenga los métodos más fiables, también cubre los problemas de los dos grupos mencionados anteriormente [26].

- ***GPS***

  Es un sistema de rastreo satelital que permite determinar la ubicación de un dispositivo en cualquier lugar del mundo con una precisión de entre 1 y 15 metros; en el 95 % esta precisión es de 3 metros. El sistema consta de 27

satélites cuya función es transmitir señales con información sobre el tiempo de transmisión y su ubicación para que los receptores GPS las interpreten y utilicen en el cálculo de su situación geográfica [26].

1.4.1. Sistemas Multiplataforma con Geolocalización

Las aplicaciones multiplataforma se caracterizan por ser desarrolladas en un lenguaje de programación que facilita exportarlas y así visualizarlas en cualquier tipo de dispositivo independientemente de su sistema operativo. Por otra parte, con la geolocalización se busca de obtener la localización de un objeto como puede ser un teléfono celular, un automóvil, etc. Para ello se pueden utilizar varios métodos como por ejemplo comprobar el código de la carta postal, la dirección IP de un equipo o el sistema GPS de nuestro teléfono móvil [27].

1.5. Tecnología RFID

Un sistema RFID es una forma de comunicación inalámbrica que permite identificar objetos a través de ondas de radio de manera única y pudiendo captar cientos de objetos a la vez. Esta tecnología funciona mediante la interacción entre dos componentes principales:

- **El Lector**

  Es la parte fundamental de todo el sistema, este envía la señal a la antena para que emita ondas de radiofrecuencia, a su vez una de estas ondas llega a una etiqueta RFID la cual se activa y devuelve los datos que contiene en su interior hacia la antena dichos datos llega al lector el cual se encarga de transformar los datos en información [28].

- **La Etiqueta**

  Es un dispositivo que emite ondas de radio y recibe las señales de respuesta de las etiquetas RFID. El lector decodifica la información almacenada en la etiqueta y la envía a un sistema informático para su procesamiento.

Las etiquetas RFID tienen la función similar a un código de barras pues estas contienen un microchip que guardan un código único de identificación el cual se va a transmitir cada vez que sea requerido, dependiendo del tag. [29]

Capítulo 2

Desarrollo del Backend y Frontend para la Aplicación Móvil Propuesta

En este capítulo detalla las etapas de desarrollo y diseño del *Frontend* y *Backend* de la aplicación móvil propuesta, utilizando el *framework* Flutter. Aborda las configuraciones necesarias para los componentes y librerías, así como las consideraciones y actualizaciones que se deben tener en cuenta para evitar fallos en el proceso. Además, presenta un proceso de maquetado inicial de la aplicación para dar inicio al desarrollo.

2.1. Planificación del Maquetado de la Aplicación Desarrollada

La Figura 2.1 describe el proceso de operación del sistema, explicando los componentes principales y el flujo de información utilizando Firebase.

Figura 2.1: Proceso de Operación del Sistema.
Fuente: El Autor.

- **Paso 1:** Mediante el aplicativo móvil los usuarios (conductor o representante/padre de familia), pueden ingresar a la información la cual se encuentra almacenada en Firebase Firestore.

- **Paso 2:** El paso 2 indica cuando el conductor ha iniciado o finalizado la ruta del viaje de los estudiantes, también el momento en el que el estudiante ha subido o bajado del medio de transporte. Dentro de la aplicación habrá una opción en la cual el conductor pueda observar los estudiantes asignados a su número de bus.

- **Paso 3:** Dicha información se envía a través de Internet al servidor, donde los servicios correspondientes identificarán a los estudiantes asignados a ese conductor. Posteriormente, se enviará una notificación a los padres de familia o representantes informándoles si la ruta ha comenzado o terminado. Además, cuando el estudiante suba o baje del medio de transporte, deberá acercar su tarjeta RFID, lo cual generará una notificación adicional para informar a los padres sobre estos eventos.

- **Paso 4:** El padre de familia o representante del estudiante podrá monitorear en tiempo real la ubicación del conductor, también podrá observar que este se encuentra abordo de la unidad de transporte o no.

2.2. Desarrollo de la aplicación

2.2.1. Maquetado de la Interfaz

Antes de sumergirnos en la codificación de la interfaz de la aplicación móvil, es importante dar forma a su funcionamiento mediante un prototipo. Este prototipo nos permitirá esbozar las páginas o pantallas que componen la aplicación en acción, brindándonos una idea concreta de su apariencia final y cómo se ajustará a los objetivos establecidos en la sección anterior. Cabe destacar que este proceso puede ser iterativo,

Inicio de sesión

La Figura 2.2 presenta un prototipo de la pantalla inicial de la aplicación, donde se visualiza el proceso de inicio de sesión tanto para conductores como para representantes o padres de familia. Cada tipo de usuario podrá acceder a la aplicación utilizando las credenciales que le hayan sido asignadas previamente

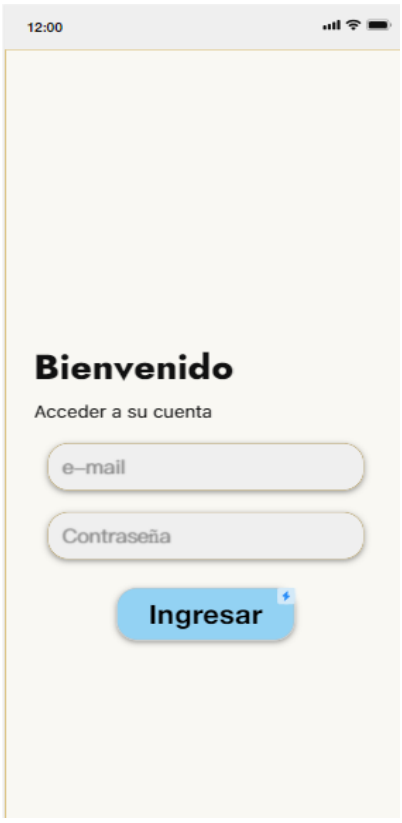
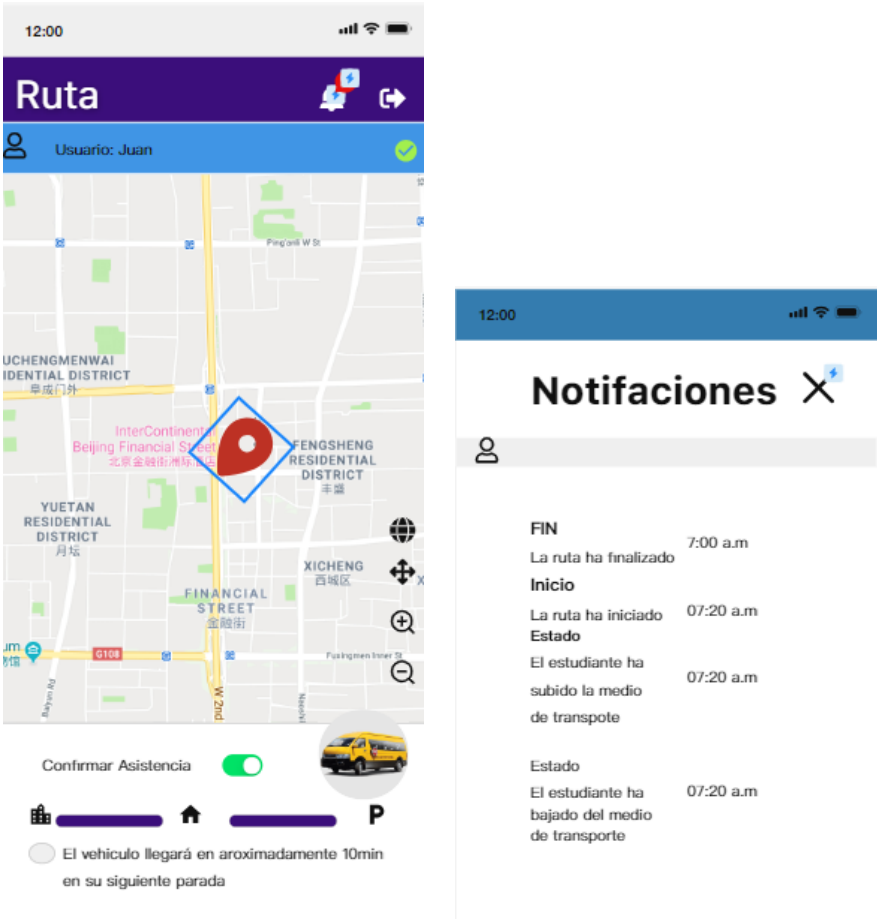


Figura 2.2: Prototipo de pantalla de Inicio de Sesion.
Fuente: El Autor.

Interfaz del Representante/Padre de Familia

El representante o padre de familia del estudiante podrá visualizar la posición del conductor sobre un mapa en tiempo real así como datos de su representado, tiempo estimado de llegada y que el conductor está considerando al estudiante ya que el padre de familia confirmó el uso de la buseta; esto se puede verificar en la Figura 2.3(a). Además, tendrá la opción de confirmar si el estudiante necesitará los servicios del medio de transporte y podrá ver el tiempo estimado de llegada del conductor a su parada. Sin embargo, la Figura 2.3(b) ilustra en forma clara el detalle de las notificaciones recibidas por el representante o padre de familia.

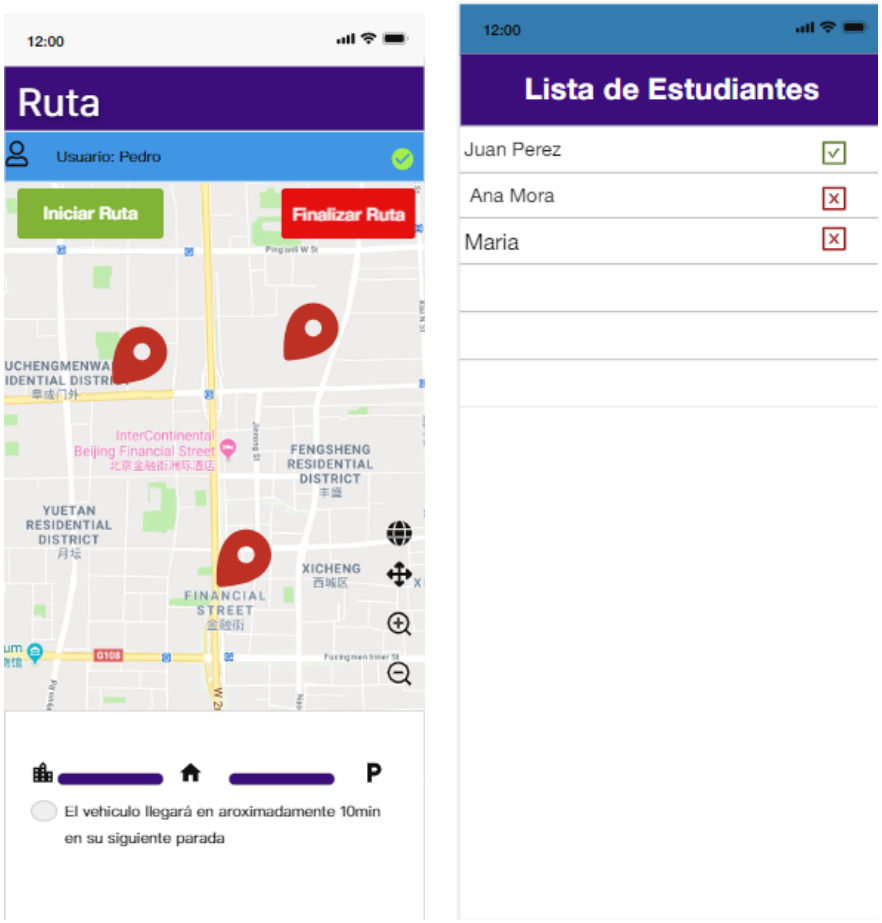


(a) Monitoreo Posición Conductor (b) Notificaciones para el Padre de Familia

Figura 2.3: Perspectiva de la Aplicación para el padre de Familia y sus notificaciones. Fuente: El Autor.

Pantalla del conductor

La Figura 2.4 presenta un maquetado de las principales pantallas que conforma la aplicación móvil para conductores. Estas pantallas abarcan las funcionalidades esenciales para la gestión de viajes y la comunicación con los representantes o padres de familia.



(a) Mapa con la ubicación de los estudiantes.

(b) Lista de estudiantes.

Figura 2.4: Interfaz de la Aplicación para el conductor.
Fuente: El Autor.

La Figura 2.4(a) presenta un mapa con las ubicaciones de los estudiantes asignados al conductor. Además, incluye botones que, al presionarlos, envían una notificación para indicar el inicio o la finalización de la ruta. Por otro lado, en la Figura 2.4(b) se muestra una lista de los estudiantes asignados al conductor, junto con un indicador que señala si el estudiante utilizará o no el medio de transporte.

2.2.2. Configuración y actualización del entorno de trabajo

Instalación SDK de Flutter

- Acceder al sitio web oficial de Flutter y descargar el SDK correspondiente a su sistema operativo.

- Extraer el archivo descargado en una ubicación adecuada en su disco local, por ejemplo, C:\src. Si la carpeta src no existe, deberá crearla.
- Edite la variable de entorno PATH del sistema e incluya la ruta a la carpeta bin dentro de la carpeta de Flutter extraída. Por ejemplo, si Flutter está instalado en C:\src \flutter, agregue la siguiente línea a la variable PATH: C:\src \flutter\bin
- Abra una terminal y ejecute el comando ***flutter doctor***. Este comando verificará la instalación correcta de *Flutter* y mostrará información sobre las herramientas y dependencias necesarias, como se ilustra en la Figura 2.5. Si se presentan problemas durante la instalación, consulte la documentación oficial de Flutter

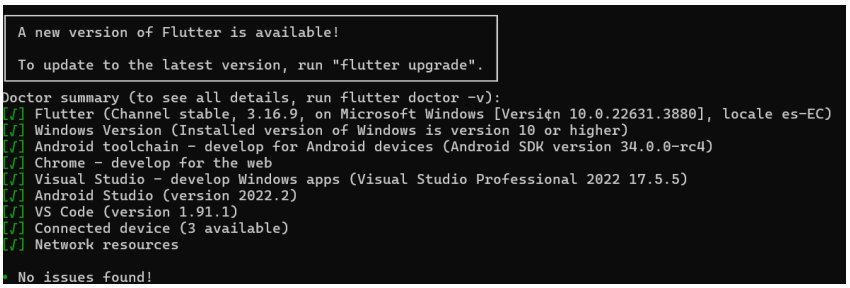


Figura 2.5: Verificar la instalación del SDK de Flutter.
Fuente: El Autor.

Instalación Android Studio

Para compilar la aplicación móvil en Android, se utiliza Android Studio. Por lo tanto, es necesario instalar este software, ingresando al sitio web oficial de Android Studio y descargando la versión correspondiente al sistema operativo.

Una vez instalado, es necesario configurar Android Studio para trabajar con el framework Flutter. Para ello, es necesario instalar el plugin de Flutter y Dart, como se muestra en la Figura 2.6. Estos plugins incluyen todo lo necesario para desarrollar aplicaciones Flutter dentro de Android Studio.

Para probar la aplicación en un dispositivo virtual con Android, se debe descargar una imagen del dispositivo con las características adecuadas desde el Administrador de Dispositivos de Android Studio en la Figura 2.7 se muestra la imagen del dispositivo creado en Android Studio. Esto te permitirá emular diferentes tipos de dispositivos y verificar el funcionamiento de tu aplicación en diversas

condiciones.Es necesario tener en cuenta que Android Studio requiere el JDK para compilar aplicaciones.

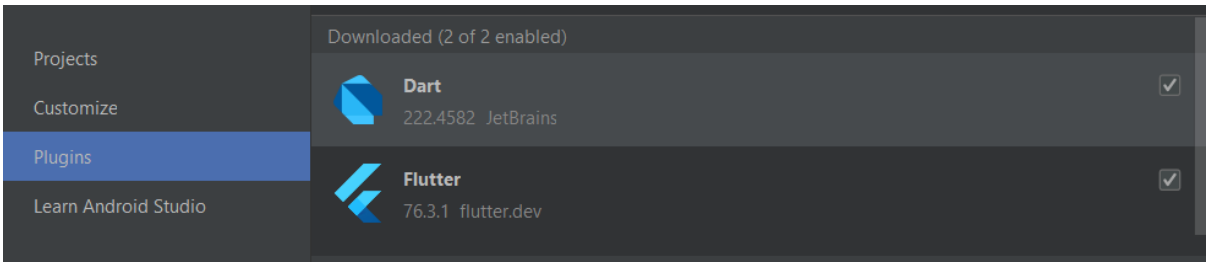


Figura 2.6: Plugins en Android Studio.
Fuente: El Autor.

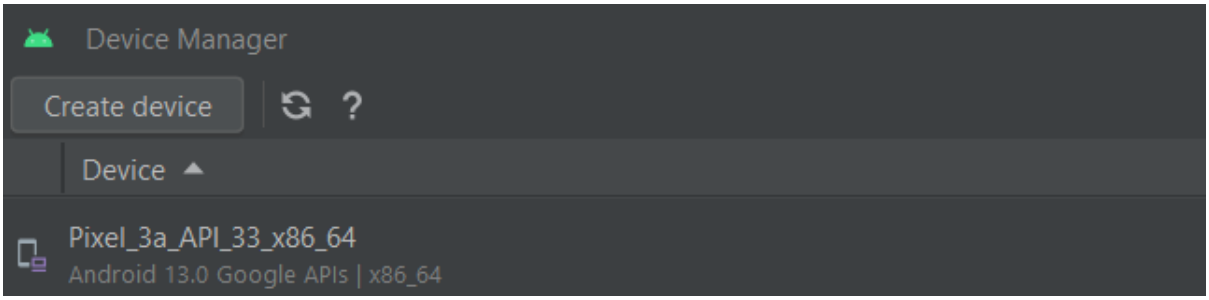


Figura 2.7: Imagen del dispositivo.
Fuente: El Autor.

Instalación de Visual Studio Code

Para la codificación de la aplicación, se utiliza el entorno de desarrollo integrado (IDE) Visual Studio Code (VS Code). Por esta razón, es necesario instalar este software, ingresando al sitio web oficial de Visual Studio Code y descargando la versión correspondiente al sistema operativo.

Para el desarrollo eficiente con Flutter y Dart, así como para el uso del módulo ESP8266, es necesario la instalacion de las siguientes extenciones en VS Code:

(a) **Extensión de Flutter**

Para la ejecución, refactorización y recarga de la aplicación móvil, es esencial instalar la extensión de Flutter para VS Code, como se muestra en la Figura 2.8 . Esta extensión proporciona todas las herramientas necesarias para desarrollar con Flutter de manera eficiente.

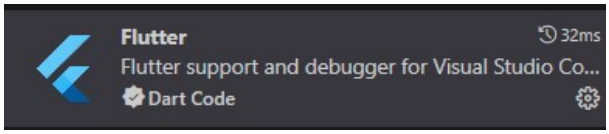


Figura 2.8: Extensión de Flutter en VS Code.
Fuente: El Autor.

(b) **Extensión de Dart**

Dado que Dart es el lenguaje de programación utilizado por Flutter, la extensión de Dart para VS Code se instala automáticamente junto con la extensión de Flutter, como se muestra en la Figura 2.9 . Esta extensión proporciona soporte adicional para el lenguaje Dart, incluyendo resaltado de sintaxis, autocompletado y herramientas de análisis de código.

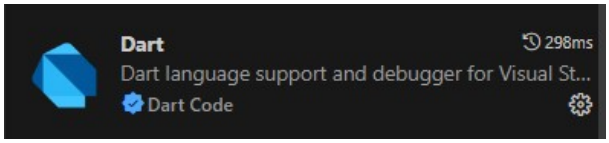


Figura 2.9: Extensión de Dart en VS Code.
Fuente: El Autor.

(c) **Extensión PlatformIO IDE** Debido a que en este proyecto se utiliza el módulo ESP8266 también se instaló la extensión PlatformIO IDE. Esta extensión facilita el desarrollo y la programación de microcontroladores al proporcionar un entorno que está integrado con herramientas de compilación, depuración y gestión de proyectos, como se muestra en la Figura 2.10.

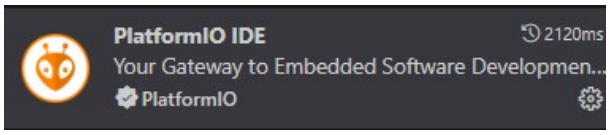


Figura 2.10: Extensión PlatformIO ID en VS Code.
Fuente: El Autor.

2.2.3. Creación del proyecto

Para crear un nuevo proyecto de Flutter mediante el IDE de VS Code, se debe utilizar la paleta de comandos (Command Palette). Esta se puede abrir desde el menú en la opción “Ver” o mediante el atajo de teclado Ctrl+Shift+P. Primero, ingresa el

comando ***Flutter: New Project***. En la Figura 2.11 se muestran las diferentes opciones para la creación del proyecto; en este caso, usaremos la primera opción.

Al comenzar un nuevo proyecto en Flutter, es necesario asignar un nombre al proyecto y elegir la ubicación donde se guardará el paquete de la aplicación. Este paquete, al inicio, contiene varios directorios, aunque algunos no se utilizarán en gran medida debido a que incluyen funciones nativas para distintas plataformas. El directorio más importante es el llamado ***lib***, ya que abarca tanto el código lógico como la interfaz gráfica de la aplicación móvil.

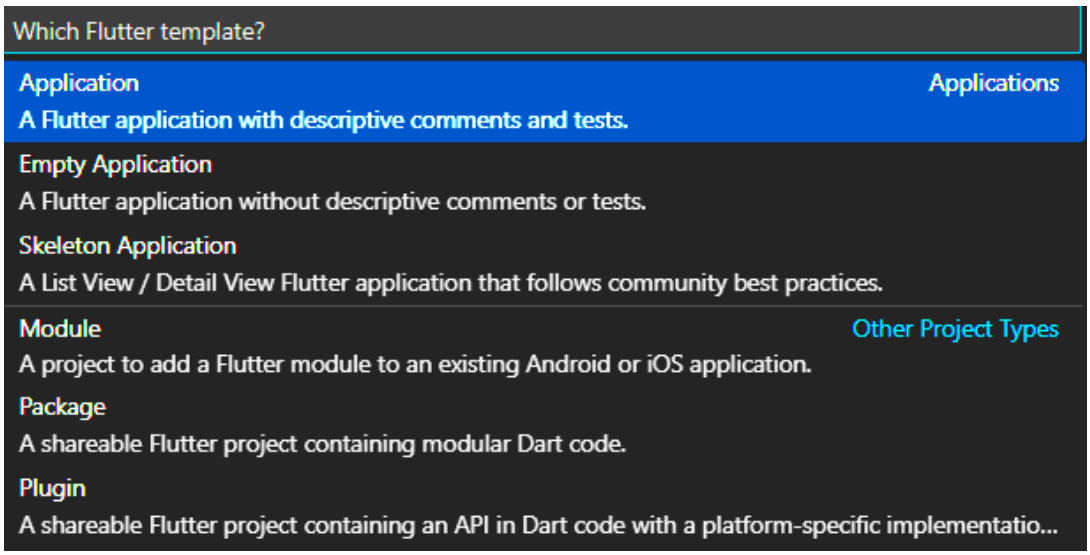


Figura 2.11: Opciones para crear nuevo proyecto en Flutter.
Fuente: El Autor.

En un proyecto creado con el lenguaje Dart, la función principal que se ejecuta es *main*, similar a otros lenguajes de programación. En Flutter, se emplea el método *runApp* para invocar la función *main*, la cual puede estar acompañada de otras funciones según la complejidad del proyecto. Flutter fomenta el desarrollo de aplicaciones mediante la reutilización de código en pequeños bloques llamados "widgets". Los desarrolladores de Flutter recomiendan el uso de estos bloques de código que realizan funciones específicas, lo que permite construir aplicaciones más grandes a partir de componentes individuales [30].

2.2.4. Desarrollo del Frontend

Widgets de Flutter

Los widgets son bloques de construcción fundamentales en Flutter, ya que todo en Flutter es un widget. Desde elementos de interfaz simples como botones y texto hasta disposiciones complejas como columnas, filas y listas, los widgets se componen y combinan para crear la interfaz de usuario completa de la aplicación.

En Flutter, hay dos tipos principales de widgets:

- **StatelessWidget:** Estos widgets no cambian una vez que se han construido. Son inmutables y se utilizan para representar elementos estáticos de la interfaz de usuario.
- **StatefulWidget:** Estos widgets tienen un estado interno que permite gestionar cambios. Son mutables, lo que significa que pueden cambiar su estado o mantenerlo, incluso cuando Flutter redibuja el componente.

El proyecto utiliza widgets de tipo StatefulWidget. Los métodos principales que permiten gestionar su estado son:

- **setState:** Este método se utiliza para notificar que el estado ha cambiado, lo que provoca que el widget se vuelva a redibujar con el nuevo estado.
- **initState:**Este método se utiliza para inicializar todos los datos necesarios en el estado inicial del widget, antes de que se dibuje en la pantalla.
- **dispose:** Se asegura de liberar recursos y limpiar animaciones cuando el widget se destruye.

En la Figura 2.12 , se presenta un ejemplo de un StatefulWidget utilizando parte del código del proyecto:

```dart
class FindingRidePageWidget extends StatefulWidget {
  const FindingRidePageWidget({
    super.key,
    this.studentDetails,
    required this.notifications,
  });

  final DocumentReference? studentDetails;
  final List<NotificationMessage> notifications;

  @override
  State<FindingRidePageWidget> createState() => _FindingRidePageWidgetState();
}

class _FindingRidePageWidgetState extends State<FindingRidePageWidget> {
  late FindingRidePageModel _model;

  final scaffoldKey = GlobalKey<ScaffoldState>();

  @override
  void initState() {
    super.initState();
    _model = createModel(context, () => FindingRidePageModel());
  }

  @override
  void dispose() {
    _model.dispose();

    super.dispose();
  }
```

Figura 2.12: Ejemplo de StatefullWidget.
Fuente: El Autor.

En el ejemplo, initState se utiliza para inicializar el modelo de la página cuando se crea el widget, y dispose se utiliza para limpiar el modelo cuando se destruye el widget.

Método Build

La aplicación contiene el método **Build**, que se encarga de ejecutar el código necesario para mostrar el widget en la pantalla. Este método es esencial para renderizar diferentes elementos de la interfaz de usuario, como botones y campos de texto. Dado que toda la aplicación está compuesta por widgets, es crucial tener un mecanismo que determine donde se localiza de cada widget. Este mecanismo se llama **BuildContext**.

BuildContext proporciona información sobre la posición y el entorno del widget dentro del árbol, permitiendo que el framework gestione y actualice la interfaz

de usuario de manera eficiente.

En la figura 2.13, se presenta un ejemplo del método Build en un StatefulWidget.

```
@override
Widget build(BuildContext context) {
  return StreamBuilder<List<backend.RideRecord>>(
    stream: queryRideRecord(
      queryBuilder: (q) =>
          q.where('uid', isEqualTo: currentUserReference?.id),
      singleRecord: true,
    ),
    builder: (context, snapshot) {
      if (!snapshot.hasData) {
        return Scaffold(
          backgroundColor: FlutterFlowTheme.of(context).primaryBackground,
          body: const Center(
            child: SizedBox(
              width: 50.0,
              height: 50.0,
              child: CircularProgressIndicator(
                valueColor: AlwaysStoppedAnimation<Color>(
                  Colors.black,
                ), // AlwaysStoppedAnimation
              ), // CircularProgressIndicator
            ), // SizedBox
          ), // Center
        ); // Scaffold
      }
```

Figura 2.13: Ejemplo del método Build.
Fuente: El Autor.

Librerías Utilizadas en Flutter

Dentro de la carpeta principal del proyecto, se encuentra un archivo denominado **pubspec.yaml**. Este archivo especifica la versión del SDK que Flutter utilizará durante el proceso de compilación, así como las bibliotecas y paquetes externos que se implementarán en nuestra aplicación móvil. Además, incluye las rutas de los archivos de configuración y de contenido multimedia que se incorporan en la aplicación.

Para la gestión e implementación de bibliotecas y paquetes externos, Flutter utiliza Pub, el administrador de paquetes para el lenguaje de programación Dart.

Estos paquetes se pueden encontrar en el sitio web oficial de Pub.dev, que alberga una amplia colección de bibliotecas y paquetes reutilizables para Flutter y programas generales de Dart. Cada paquete en Pub.dev incluye instrucciones detalladas para su instalación y ejemplos de uso, como se muestra en la Figura 2.14.

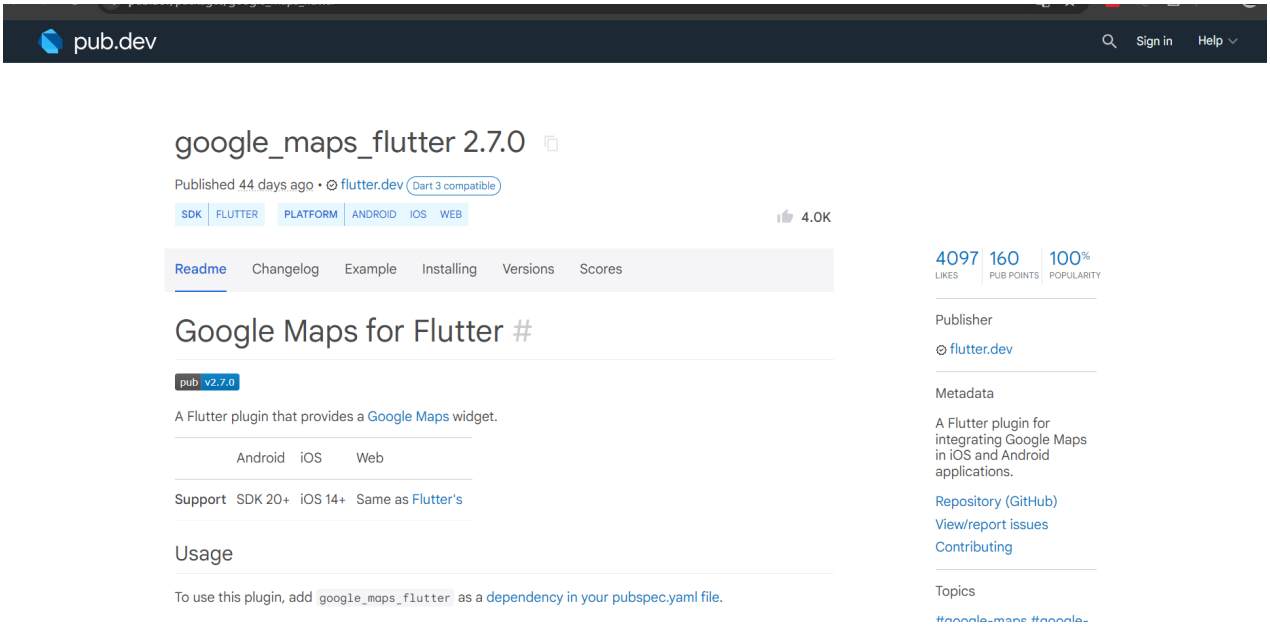


Figura 2.14: Paquete de ejemplo en pub.dev.
Fuente: El Autor.

Existen dos métodos principales para instalar estos paquetes:

- **Instalación mediante comandos en la terminal:** La Figura 2.15. ilustra el proceso de instalación utilizando comandos en la terminal del proyecto.

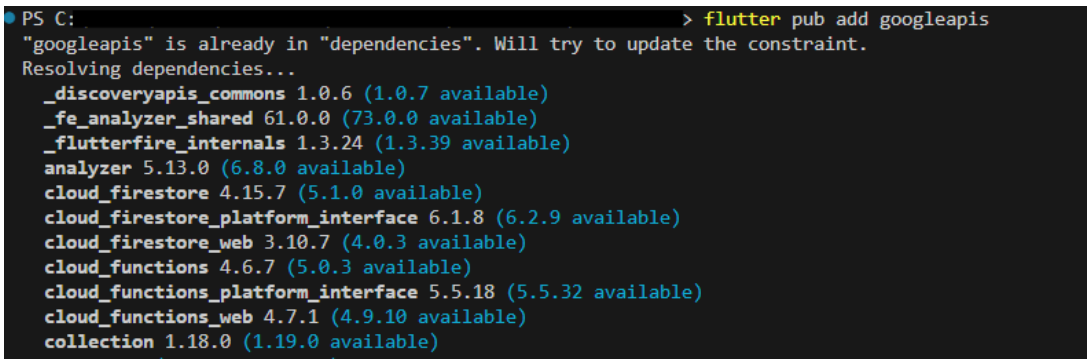


Figura 2.15: Agregar el paquete mediante comando en la terminal.
Fuente: El Autor.

- **Instalación manual:** La Figura 2.16. muestra cómo agregar manualmente un paquete y su versión correspondiente en el archivo **pubspec.yaml** en la sección

de dependencies. Después de esto, se debe ejecutar el comando ***flutter pub get*** para completar la instalación. En el caso del IDE de VS Code, basta con guardar el archivo para que la instalación se realice automáticamente.

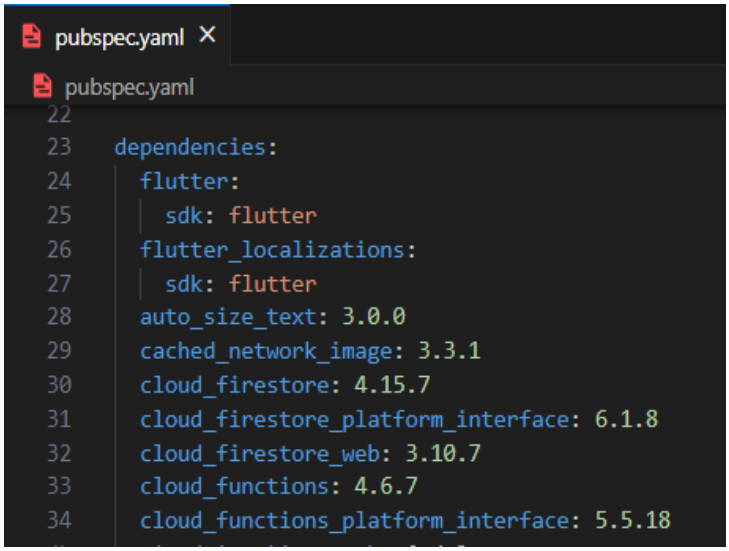


Figura 2.16: Agregar el paquete de manera manual.
Fuente: El Autor.

Para evitar problemas de compatibilidad y vulnerabilidades de seguridad asociados con paquetes obsoletos, es recomendable utilizar el comando ***flutter pub outdated***. Este comando mostrará una lista de todas las dependencias en el archivo ***flutter pub get***, junto con su versión actual y la última versión disponible, permitiendo mantener las dependencias actualizadas y seguras.

A continuación, se detallan las librerías externas y las versiones utilizadas en el desarrollo de la aplicación móvil.

- **auto_size_text 3.0.0:** Permite que el tamaño del texto dentro de un container cambie automáticamente según el espacio disponible.
- **cached_network_image 3.3.1:** Permite cargar y almacenar imágenes desde la web en caché, lo que mejora el rendimiento y la experiencia del usuario.
- **cloud_firestore 4.15.7:** Permite el almacenamiento y la sincronización de datos en la nube a través de Firestore, la base de datos en tiempo real de Google.
- **cloud_functions, 4.6.7:** Permite invocar funciones en la nube de Firebase desde una aplicación Flutter

- **collection, 1.18.0:** Incluye herramientas y clases adicionales para la manipulación de colecciones de datos.
- **firebase_auth, 4.17.7:** A través de Firebase, proporciona métodos de autenticación de usuarios como correo electrónico y contraseña,
- **firebase_core, 2.26.0:** Esta dependencia fundamental permite la conexión con diversos servicios de Firebase, como Firebase Authentication, Cloud Firestore, Realtime Database y Cloud Storage.
- **firebase_messaging, 14.7.18:** Proporciona el servicio Firebase Cloud Messaging (FCM), una solución de mensajería multiplataforma para enviar notificaciones push confiables en dispositivos Android e iOS.
- **flutter_cache_manager, 3.3.1:** Mejora el rendimiento y la eficiencia de almacenamiento al facilitar la gestión de la caché para archivos descargados en una aplicación Flutter.
- **flutter_polyline_points, 1.0.0:** Decodifica cadenas de polilíneas codificadas por Google, convirtiéndolas en listas de coordenadas que representan la ruta entre dos ubicaciones geográficas.
- **geolocator, 10.1.0:**

  Ofrece una API multiplataforma (iOS y Android) para acceder a funciones de ubicación, como el GPS, la brújula y la altitud.
- **google_fonts, 6.1.0:** Simplifica la integración y personalización de las fuentes de Google Fonts en las aplicaciones Flutter, permitiendo una amplia variedad de estilos tipográficos.
- **google_maps_flutter, 2.1.1:** Facilita la integración de Google Maps en aplicaciones Flutter, permitiendo la visualización de mapas, la búsqueda de ubicaciones, la definición de rutas y la implementación de marcadores.
- **intl, 0.18.1:** Maneja la internacionalización y localización de la aplicación, incluyendo el formato y análisis de fechas y números, el texto bidireccional y otros aspectos relacionados con la adaptación cultural

- **json_path, 0.6.2:** Proporciona herramientas para navegar y consultar datos en estructuras JSON.
- **plugin_platform_interface, 2.1.8:** Ofrece una interfaz básica para la creación de plugins multiplataforma en Flutter, asegurando la compatibilidad y la consistencia entre diferentes plataformas.
- **provider, 6.0.5:** Proporciona un patrón de diseño simple y escalable para la inyección de dependencias, lo que facilita la gestión del estado en las aplicaciones Flutter.
- **shared_preferences, 2.2.2:** Permite el almacenamiento y recuperación de datos básicos de preferencias del usuario en almacenamiento persistente.
- **stream_transform, 2.1.0:** Incluye utilidades para la transformación y manipulación de flujos de datos Dart
- **tuple 2.0.2:** Proporciona una clase Tuple que puede empaquetar varios valores en una estructura única, lo que es útil para transferir datos agrupados.
- **url_launcher, 6.2.5:** Permite el lanzamiento de una URL Admite esquemas en línea, mensajes de texto, teléfonos y correos electrónicos.
- **cupertino_icons, 1.0.0:** Esta dependencia proporciona acceso a una colección de iconos diseñados por Apple, siguiendo el estilo de la interfaz de usuario de iOS.
- **geocoding 2.0.4:**Ofrece funciones de geocodificación directa e inversa para convertir direcciones en coordenadas geográficas y viceversa.
- **firebase_database 10.4.8:** Brinda acceso a Firebase Realtime Database, permitiendo almacenar y sincronizar datos en tiempo real entre clientes.
- **flutter_local_notifications 17.1.2:** Permite la visualización y programación de notificaciones locales, que se pueden adaptar a cada plataforma (iOS, Android).
- **googleapis_auth 1.4.1:** Facilita el acceso a las APIs de Google mediante autenticación OAuth2, lo que facilita la integración con los servicios de Google en las aplicaciones Dart.

- **googleapis 13.1.0:** Permite a los clientes acceder a las APIs de Google, lo que les permite interactuar con varios servicios de Google a través de las aplicaciones Flutter.
- **flutter_launcher_icons 0.13.1:** Facilita la actualización del icono de inicio de la aplicación.

Importación de Librerías y Paquetes

Las librerías y los paquetes son esenciales para ampliar las funcionalidades de la aplicación, es por esto que la mayoría del código fuente de la aplicación incluye la importación de librerías. Estas librerías pueden ser externas, internas o librerías Dart, así los widgets que vienen con el SDK de Flutter de forma predeterminada.

Las librerías externas permiten utilizar código que ha sido probado y optimizado por la comunidad, lo que le ahorra tiempo y esfuerzo durante el desarrollo. Las librerías internas permiten dividir el código en módulos, lo que facilita el mantenimiento y la reutilización. Finalmente, la librería Dart proporciona herramientas y utilidades necesarias para el desarrollo de aplicaciones.

La Figura 2.17 presenta un ejemplo práctico de cómo se importan diferentes tipos de librerías en un archivo Dart:

```dart
import 'package:flutter/material.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:cloud_firestore/cloud_firestore.dart';
import '/backend/backend.dart'; // Ensure to import backend.dart
import 'rider_page_model.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
```

Figura 2.17: Ejemplo de importaciones de librerías.
Fuente: El Autor.

Ejecución y Depuración de la Aplicación

Para ejecutar la aplicación, es necesario seleccionar el dispositivo que se utilizará para la misma. Esto se puede hacer desde la barra de estado en la parte inferior de VS Code o mediante el atajo de teclado Ctrl+Shift+P y seleccionando el comando ***Flutter: Select Device***, donde se muestra el nombre del dispositivo

conectado o la opción para seleccionar un dispositivo disponible. En caso de no haber dispositivos conectados, se puede utilizar un emulador de Android o iOS, que se inicia desde Android Studio.

Para ejecutar la aplicación, se puede hacer clic en **Run Start Debugging** desde la ventana principal del IDE VS Code o presionar F5. La aplicación se compilará y se ejecutará en el dispositivo seleccionado, ya sea un emulador o un dispositivo físico conectado.

Para depurar la aplicación, se puede iniciar el modo de depuración seleccionando **Run Start Debugging** o presionando F5. Esto iniciará la ejecución de la aplicación en modo de depuración, en el cual se puede pausar la ejecución, establecer puntos de interrupción (breakpoints) y revisar el estado de las variables y la pila de llamadas.

Los breakpoints se pueden establecer haciendo clic en el margen izquierdo del editor de código, junto a la línea donde se desea pausar la ejecución. Cuando la ejecución alcanza un breakpoint, la aplicación se pausa y se puede inspeccionar el estado de las variables y la pila de llamadas en el panel de depuración.

2.2.5. Desarrollo del Backend

Para el desarrollo del backend de la aplicación se ha utilizado Firebase, ya que, este proporciona una solución escalable y completa para la gestión de datos en tiempo real, autenticación de usuarios, almacenamiento de datos de usuarios, y notificaciones push.Además, se intengró un sitema RFID con un modulo ESP8266 para registrar cuando los estudiantes suban o bajen del transporte escolar y posteriormente notificar al representante o padre de familia

Configuración de Firebase

Para usar los servicios de Firebase en la aplicación, fue necesario realizar una serie de configuraciones iniciales tanto en la consola de Firebase como en el proyecto de Flutter. A continuación, se presentarán las configuraciones que se realizaron para este proyecto.

- **Crear un Proyecto en Firebase**

  Para iniciar un proyecto en Firebase, primero se debe acceder al sitio web oficial de Firebase. Tras ingresar los parámetros necesarios para la creación del proyecto, aparecerá una pantalla donde se debe elegir el tipo de aplicación a la que se desea agregar Firebase. Para este proyecto, se selecciona una aplicación de Android y se registran los parámetros requeridos. Esto permite descargar el archivo *google-services.json*, que debe integrarse y configurarse en el proyecto de Flutter, ya que contiene la información necesaria para utilizar los servicios de Firebase en la aplicación móvil.

  Una vez completados estos pasos, se añade el servicio de Cloud Messaging al proyecto de Firebase desde el panel en la parte izquierda, tal como se muestra en la Figura 2.18.

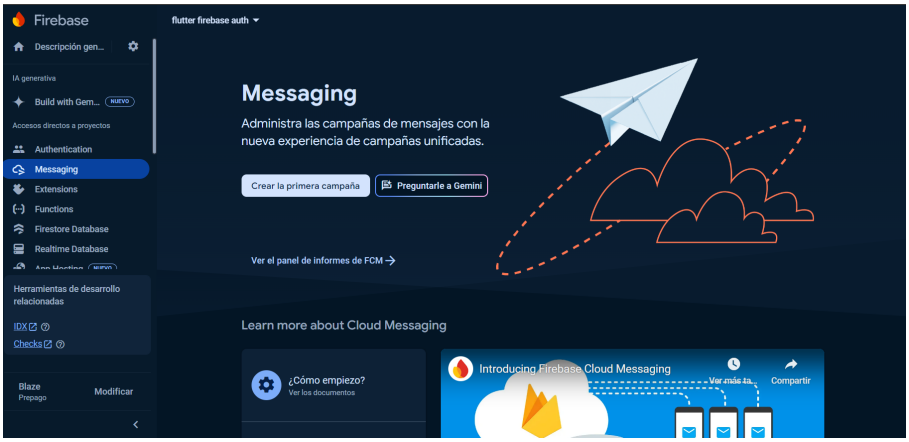


Figura 2.18: Panel de configuración de Firebase.
Fuente: El Autor.

Para utilizar el servicio de Cloud Messaging en Firebase, es necesario tener activado el plan Blaze. Este plan de prepago proporciona acceso a una mayor cantidad de recursos y servicios avanzados de Firebase. Además, permite enviar notificaciones ilimitadas a través de Cloud Messaging, lo cual es esencial para aplicacion debido a que esta va a enviar notificaciones en tiempo real cuando el conductor inicia o finaliza una ruta, y cuando el estudiante sube o baja del transporte.La Figura 2.19 muestra el detalles del plan utilizado para este servicio.

Figura 2.19: Plan utilizado para el servicio de Cloud Messaging.
Fuente: El Autor.

- **Configuracion en el proyecto de flutter**

  Para integrar Firebase en el proyecto, se utiliza la CLI de Firebase y FlutterFire, lo que facilita su integración y configuración. En primera instancia, se debe instalar la CLI de Firebase a través de una terminal utilizando el comando ***npm install -g firebase-tools***. Una vez realizada la instalación de Firebase CLI, es necesario iniciar sesión en la cuenta de Google asociada al proyecto de Firebase con la ayuda del comando ***firebase login***. Luego, se debe realizar la instalación de FlutterFire CLI con el comando ***dart pub global activate flutterfire_cli***. Finalmente, hay que agregar el proyecto a la aplicación con la ayuda del comando ***flutterfire configure***, este mostrará una lista de proyectos existentes en Firebase y se deberá seleccionar el proyecto que se utilizará para el desarrollo de la aplicación móvil.

  Una vez realizadas dichas instalaciones, se debe inicializar Firebase en el proyecto de Flutter en el archivo **main.dart**, como se muestra en la Figura 2.20.

```
Run | Debug | Profile
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await initFirebase();

  FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBackgroundHandler);

  final appState = FFAppState(); // Initialize FFAppState
  await appState.initializePersistedState();

  runApp(ChangeNotifierProvider(
    create: (context) => appState,
    child: const MyApp(),
  )); // ChangeNotifierProvider
}
```

Figura 2.20: Inicializar Firebase en main.dart.
Fuente: El Autor.

Para integrar Firebase en Android, es necesario agregar el pluging de **google services** en el directorio **android/build.gradle** en la sección de **dependencias**, como se muestra en la Figura 2.21.

```
buildscript {
  dependencies {
    classpath 'com.google.gms:google-services:4.3.10'
  }
}
```

Figura 2.21: Agregar pluging de google service en seccion dependencias.
Fuente: El Autor.

Luego, en el directorio **android/app/build.gradle**, se aplica el plugin de **google services** al final del archivo, como se muestra en la Figura 2.22.

```
flutter {
    source '../..'
}
apply plugin: 'com.google.gms.google-services'
```

Figura 2.22: Agregar pluging de google service.
Fuente: El Autor.

Finalmente, el archivo ***google-services.json*** descargado desde el proyecto de Firebase se debe colocar en el directorio **android/app/** como se mestra en la Figura 2.23.

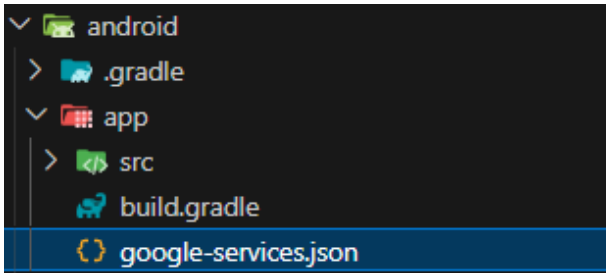


Figura 2.23: Directorio de google-services.json.
Fuente: El Autor.

Estructura de la Base de Datos

La estructura de la base de datos en Firestore está diseñada para proporcionar una gestión eficiente y en tiempo real de la información relacionada con el transporte de estudiantes. A través de las colecciones "students" y "ride", se puede rastrear y notificar a los padres sobre la ubicación y el estado de los estudiantes de manera efectiva. En la Figura 2.24 se muestra el diagrama final de la base de datos.

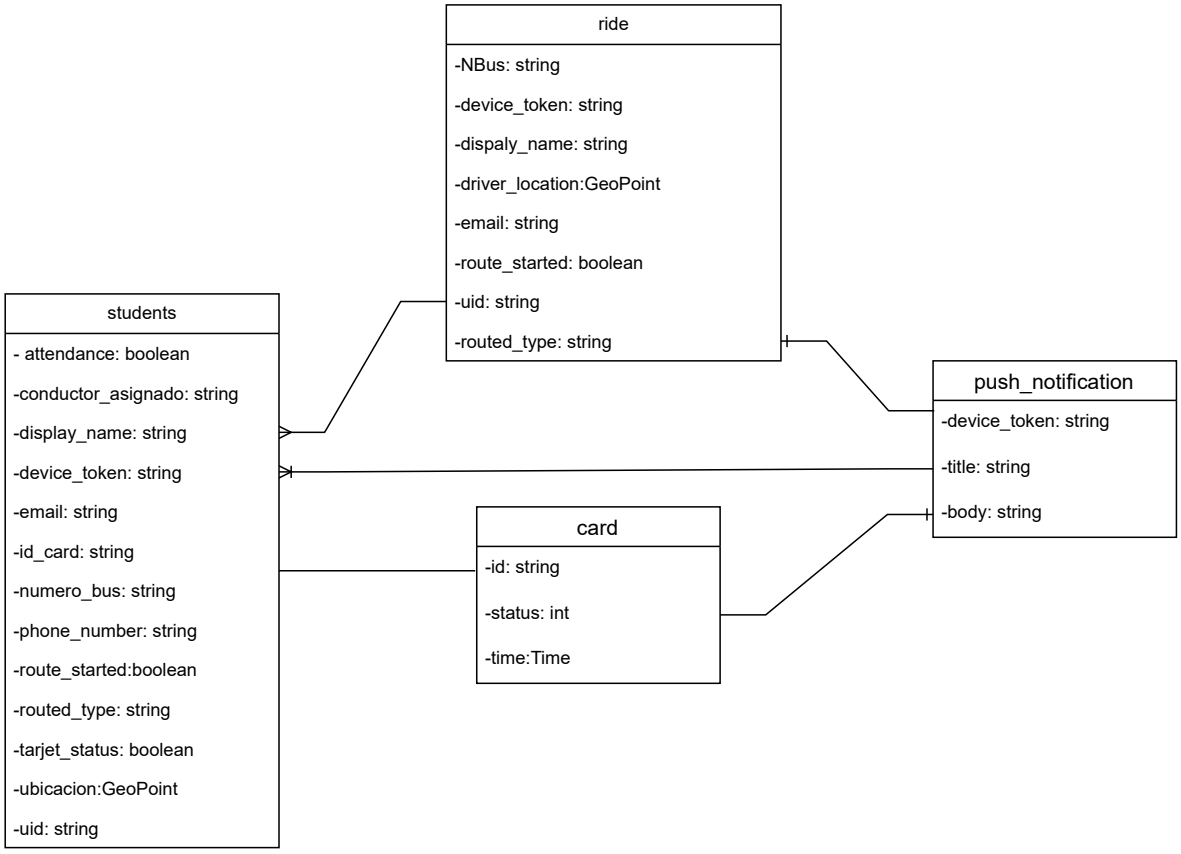


Figura 2.24: Diagrama UML de la base de datos para la Aplicación.
Fuente: El Autor.

Gestión de Datos en Tiempo Real

El uso de Firebase Firestore permite la gestión de datos en tiempo real, lo cual es esencial para nuestra aplicación, ya que necesita una actualización constante. Esto es especialmente importante porque los representantes o padres de familia van a monitorear la ubicación en tiempo real del conductor en un mapa.

Los **Streams** dentro del código de la aplicación son fundamentales para manejar datos en tiempo real. Un Stream proporciona una secuencia de datos asincrónicos. En el contexto de Firestore, los Streams se utilizan para escuchar cambios en tiempo real en los documentos o colecciones, permitiendo que la interfaz de usuario se actualice automáticamente cuando los datos cambian.

Para implementar Streams con Firestore, se utiliza el método ***snapshots()*** de una colección o documento, que devuelve un Stream de datos. La Figura 2.25 muestra un ejemplo del uso de Streams para escuchar cambios en una colección y actualizar la interfaz de usuario en tiempo real, el cual utiliza ***StreamBuilder*** para construir un widget basado en el último valor emitido por el Stream de Firestore. Cada vez que hay un cambio en la colección students, el StreamBuilder reconstruye su contenido, mostrando la lista actualizada de estudiantes.

```
Stream<List<StudentRecord>> queryStudentRecord({
  required Query Function(Query) queryBuilder,
}) {
  return queryBuilder(FirebaseFirestore.instance.collection('estudiantes'))
      .snapshots()
      .asyncMap((snapshot) async {
    List<StudentRecord> students = [];
    for (var doc in snapshot.docs) {
      students.add(await StudentRecord.fromFirestore(doc));
    }
    return students;
  });
}
```

Figura 2.25: Ejemplo uso Streams.
Fuente: El Autor.

Autenticación de Usuarios

La autenticación de usuarios es una parte esencial de la aplicación, ya que permite identificar y autorizar a los conductores y a los representantes o padres de familia. Para lograr esto, se ha utilizado Firebase Authentication, que proporciona una solución segura y eficiente para el inicio de sesión y el registro de usuarios.

Para implementar Firebase Authentication, hay que navegar a la consola de

Firebase, seleccionar el proyecto, ir a la sección “Authentication >Método de acceso” y habilitar el método de autenticación por correo electrónico y contraseña, como se muestra en la Figura 2.26.

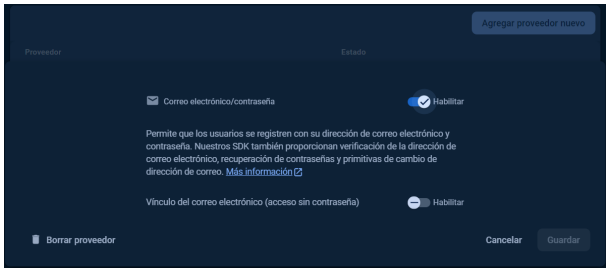


Figura 2.26: Habilitar el método de autenticación por correo electrónico y contraseña.
Fuente: El Autor.

La Figura 2.27 muestra un ejemplo de cómo integrar la autenticación en la interfaz de usuario, permitiendo a los usuarios iniciar sesión. Dentro de esta figura, la Figura 2.27(a) muestra el código de la función utilizada para el inicio de sesión, y la Figura 2.27(b) presenta el código de verificación de datos ingresados por el usuario..

```
Future<User?> signInWithEmailAndPassword(String email, String password) async {
  try {
    UserCredential result = await _auth.signInWithEmailAndPassword(
        email: email, password: password);
    User? user = result.user;
    return user;
  } catch (e) {
    print(e.toString());
    return null;
  }
}
```

(a) Función para Inicio de sesión

```
@override
void initState() {
  super.initState();
  _model = createModel(context, () => LoginPageModel());

  _model.emailAddressTextController ??= TextEditingController();
  _model.emailAddressFocusNode ??= FocusNode();

  _model.passwordLoginTextController ??= TextEditingController();
  _model.passwordLoginFocusNode ??= FocusNode();
}
```

(b) Verificación de datos para inicio de sesión

Figura 2.27: Integracion de autenticación para inicio de sesión.
Fuente: El Autor.

Notificaciones Push

Para la integración de notificaciones push en nuestro proyecto, es necesario importar las librerías ***firebase_messaging*** y ***firebase_core*** en el archivo **main.dart**. Además, se debe inicializar Firebase y configurar FCM (Firebase Cloud Messaging) en el método ***main*** de la aplicación, como se muestra en la Figura 2.28.

```dart
import 'package:firebase_core/firebase_core.dart';
import 'package:firebase_messaging/firebase_messaging.dart';
Future<void> _firebaseMessagingBackgroundHandler(RemoteMessage message) async {
  await Firebase.initializeApp();
  print('Handling a background message: ${message.messageId}');
}

Run | Debug | Profile
Future<void> main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await initFirebase();

  FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBackgroundHandler);

  final appState = FFAppState(); // Initialize FFAppState
  await appState.initializePersistedState();

  runApp(ChangeNotifierProvider(
    create: (context) => appState,
    child: const MyApp(),
  )); // ChangeNotifierProvider
}
```

Figura 2.28: Inicializar Firebase y configurar FCM.
Fuente: El Autor.

La Figura 2.29 muestra la función utilizada para el envío y recepción de notificaciones.

```dart
static Future<void> sendNotificationToSelectedStudent(String deviceToken,
    String messageTitle, String messageBody, String tripID) async {
  final String serverAccessTokenKey = await getAccessToken();
  String endpointFirebaseCloudMessaging =
      'https://fcm.googleapis.com/v1/projects/flutter-firebase-auth-8f7c7/mess

  final Map<String, dynamic> message = {
    'message': {
      'token': deviceToken,
      'notification': {
        'title': messageTitle,
        'body': messageBody,
      },
      'data': {
        'tripID': tripID,
      }
    }
  };

  final http.Response response = await http.post(
    Uri.parse(endpointFirebaseCloudMessaging),
    headers: <String, String>{
      'Content-Type': 'application/json',
      'Authorization': 'Bearer $serverAccessTokenKey',
    },
    body: jsonEncode(message),
  );

  if (response.statusCode == 200) {
    print('FMC message sent successfully');
  } else {
    print('Failed to send FCM message: ${response.statusCode}');
  }
}
```

Figura 2.29: Función para envío y recepción de notificaciones .
Fuente: El Autor.

2.2.6. Integración de Geolocalización y Mapas

La integración de geolocalización y mapas es una parte fundamental del proyecto, ya que permite el seguimiento en tiempo real de la ubicación de los estudiantes y del conductor del transporte escolar. A continuación, se detallan los componentes y funcionalidades clave implementadas para lograr esta integración.

Google Maps y Claves de API

Para la visualización de mapas y rutas, se utiliza la API de Google Maps.Es necesario obtener las claves API desde Google Cloud para diferentes las plataformas (iOS, Android y Web), de esta manera se asegurar la compatibilidad y funcionalidad en todas ellas. Estas claves permiten acceder a los servicios de mapas, direcciones y geolocalización proporcionados por Google.

Obtención de Ubicaciones en Tiempo Real

La aplicación utiliza el paquete Geolocator para obtener la ubicación actual del conductor en tiempo real. Esta ubicación se actualiza periódicamente mediante un temporizador que verifica la posición del conductor y la almacena en la base de datos de Firestore. La Figura 2.30 muestra el codigo que se utilizo.

```
Future<LatLng> getCurrentUserLocation(
    {LatLng defaultLocation = const LatLng(0.0, 0.0)}) async {
  try {
    Position position = await Geolocator.getCurrentPosition(
        desiredAccuracy: LocationAccuracy.high);
    return LatLng(position.latitude, position.longitude);
  } catch (e) {
    log('Error obtaining current user location: $e');
    return defaultLocation;
  }
}
```

Figura 2.30: Obtención de ubicacion del conductor.
Fuente: El Autor.

Cálculo de Distancia

Para proporcionar información precisa sobre la distancia y el tiempo estimado de llegada, se realiza un cálculo de distancia entre la ubicación del conductor y la del estudiante utilizando la fórmula de Haversine. La Figura muestra la implementacion de este.

```
double _coordinateDistance(lat1, lon1, lat2, lon2) {
  var p = 0.017453292519943295;
  var c = cos;
  var a = 0.5 -
      c((lat2 - lat1) * p) / 2 +
      c(lat1 * p) * c(lat2 * p) * (1 - c((lon2 - lon1) * p)) / 2;
  return 12742 * asin(sqrt(a));
}
```

Figura 2.31: Cálculo de Distancia entre ubicacines.
Fuente: El Autor.

Además, se utiliza la API de Google Distance Matrix para obtener la duración estimada del viaje como se muetra en la Figura 2.32

Figura 2.32: Duración estimada de viaje.
Fuente: El Autor.

Marcadores de Estudiantes

Los marcadores en el mapa representan la ubicación de los estudiantes. Estos marcadores se posionan en el mapa en funcion de la distancia de la ubicacion del conductor desde el estudiante mas cercano al mas lejano, ademas que cambian de color en función del estado de la ruta y la posición del conductor. Para rutas escolares, los marcadores cambian a verde cuando el estudiante es el próximo en ser recogido y a rojo una vez ha sido recogido. Para rutas de retorno, el marcador se vuelve verde cuando el estudiante es el próximo en ser dejado y rojo después de ser dejado.

2.2.7. Implementación del Sistema RFID con el ESP8266

La implementación del sistema RFID con el módulo ESP8266 es fundamental para el seguimiento y registro del abordaje y descenso de los estudiantes en el transporte escolar. Este sistema permite a los representantes o padres de familia recibir notificaciones cuando sus hijos o representados suben o bajan del transporte escolar.

A continuación, se detallan los componentes, configuración y el código necesario para implementar este sistema.

Componentes utilizados

- **Modulo ESP8266:** Es un microcontrolador Wi-Fi utilizado para la comunicación con el backend de Firebase.

- **Lector RFID RC522:** Es un dispositivo que lee las etiquetas RFID de los estudiantes, es utilizado para saber si suben o bajan del transporte escolar.
- **Etiquetas RFID:** Tarjetas asignadas a cada estudiante que contienen un identificador único.

Esquema de Conexiones

Es importante realizar las conexiones entre el ESP8266 y el lector RFID RC522 correctamente. La figura 2.33 indica el circuito utilizado para este proyecto, mientras que en la Figura 2.1 se puede visualizar el esquema de conexiones

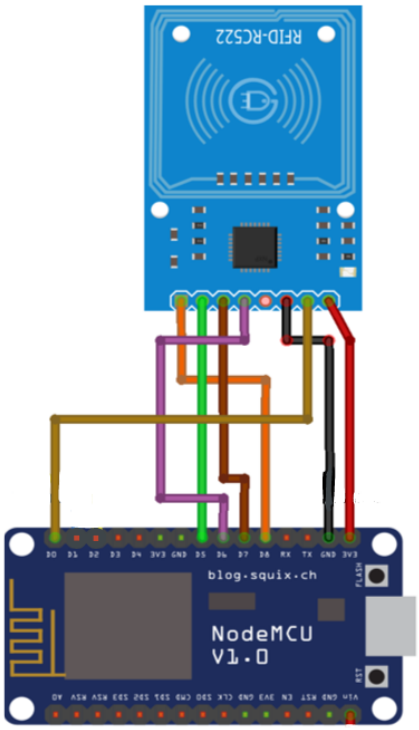


Figura 2.33: Circuito.
Fuente: El Autor.

Código utilizado para el módulo ESP8266

El módulo ESP8266 se encarga de leer las etiquetas RFID, conectarse a la red Wi-Fi, comunicarse con Firebase para almacenar los datos de asistencia y enviar notificaciones push. A continuación, se presentan las partes esenciales del código

Tabla 2.1: Esquema de Conexiones.Fuente: El Autor.

ESP8266 Pin	ESP8266 Pin
D8	SDA (SS)
D0	RST
D5	SCK
D7	MOSI
D6	MISO
GND	GND
3V3	3.3V

La Figura 2.34 muestra las librerías utilizadas para el desarrollo del proyecto, así como la configuración para la conexión de Wi-Fi y Firebase.

```
#include <Arduino.h>
#include <ESP8266WiFi.h>
#include <Wire.h>
#include <SPI.h>
#include <RFID.h>
#include "FirebaseESP8266.h"
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <Time.h>
#include <TimeLib.h>
#include <Timezone.h>

void printTime(time_t t);
String convertirTimeATextoFecha(time_t t);

#define NTP_OFFSET   60 * 60
#define NTP_INTERVAL 60 * 1000
#define NTP_ADDRESS  "pool.ntp.org"

const char *months[] = {"01", "02", "03", "04", "05", "06", "07", "08", "09", "10", "11", "12"};

//////////////////////////
// Conexion con Firebase

#define FIREBASE_HOST "prueba-6acb8-default-rtdb.firebaseio.com"
#define FIREBASE_AUTH "I1QAVdD7JjjXBDxhhVI8crJQdfBaNrpawG76Wr6j"

RFID rfid(D8, D0);
unsigned char str[MAX_LEN];
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP, NTP_ADDRESS, NTP_OFFSET, NTP_INTERVAL);
TimeChangeRule usMST = {"MST", First, Sun, Nov, 2, -360};
Timezone usAZ(usMST, usMST);
time_t local, utc;

const char ssid[] = "SISTELCEL_TALIA";
const char pass[] = "0107983058";
```

Figura 2.34: Configuración para la conexión de Wi-Fi y Firebase.
Fuente: El Autor.

Capítulo 3

Implementación y Pruebas de Operación de la aplicación móvil

En este capítulo se detalla el proceso de implementación y las pruebas de operación de la aplicación móvil desarrollada para el monitoreo del transporte de estudiantes. Se describe la personalización del icono de la aplicación, además del proceso de generación e instalación del archivo APK. Finalmente, se presentan las pruebas de operación realizadas tanto en el módulo del conductor como en el módulo del representante o padre de familia, con el fin de validar la funcionalidad de la aplicación.

3.1. Configuración Inicial del Aplicativo Móvil

3.1.1. Configuración del Icono de la aplicación

Para configurar el icono de la aplicación que se mostrará en varios dispositivos móviles, se recomienda utilizar el paquete ***flutter_launcher_icons*** en el archivo pubspec.yaml para agilizar y simplificar este proceso.

La configuración del paquete se ilustra en la Figura 3.1, donde se especifica la ruta del icono y otras funciones necesarias para su uso en diferentes plataformas.

```
flutter_launcher_icons:
  android: "launcher_icon"
  ios: true
  image_path: "assets/images/Logo_GITEL.jpg"
```

Figura 3.1: Configuración del icono.
Fuente: El Autor.

Después de configurar el proyecto, es necesario ejecutar el siguiente comando en la terminal del proyecto: **flutter packages pub run flutter_launcher_icons**.

Este comando aplica automáticamente las configuraciones realizadas en los distintos directorios de las plataformas iOS y Android.

3.1.2. Instalación de la Aplicación en dispositivos Android

Generación del APK

Para distribuir el aplicativo móvil en un dispositivo android, se generó un archivo APK con la ayuda del comando ***flutter build apk –release***. Una vez completado el proceso, se genera una apk de lanzamiento en la ubicación **/build/app/outputs/apk/** del proyecto de flutter.

Instalación del APK

Una vez generado el APK, se procede a instalarlo en varios dispositivos móviles para realizar las respectivas pruebas de operación.

La instalación se puede realizar utilizando tecnicas como:

- **Conexión USB:**Conectar el dispositivo móvil al ordenador mediante un cable USB y copiar el archivo APK directamente al almacenamiento del dispositivo.
- **Google Drive/Dropbox:** Subir el archivo APK a un servicio de almacenamiento en la nube para posteriormente descargarlo en el dispositivo móvil.
- **Correo Electrónico:** Enviar el APK como un archivo adjunto a un correo electrónico y abrirlo en el dispositivo móvil.

La Figura 3.2 indica como se realizó la instalación del aplicativo móvil

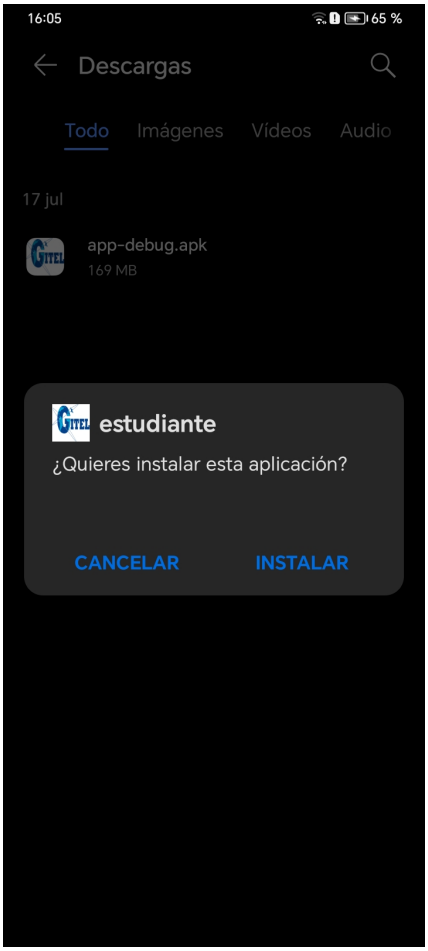


Figura 3.2: Instalación apk en Android.
Fuente: El Autor.

3.2. Pruebas de operación: Módulo del Conductor

3.2.1. Inicio de sesión

Para realizar la prueba de funcionamiento de la aplicación, se utilizará las credenciales de un usuario descritas en la Figura 3.3 el cual tendrá el rol de conductor.

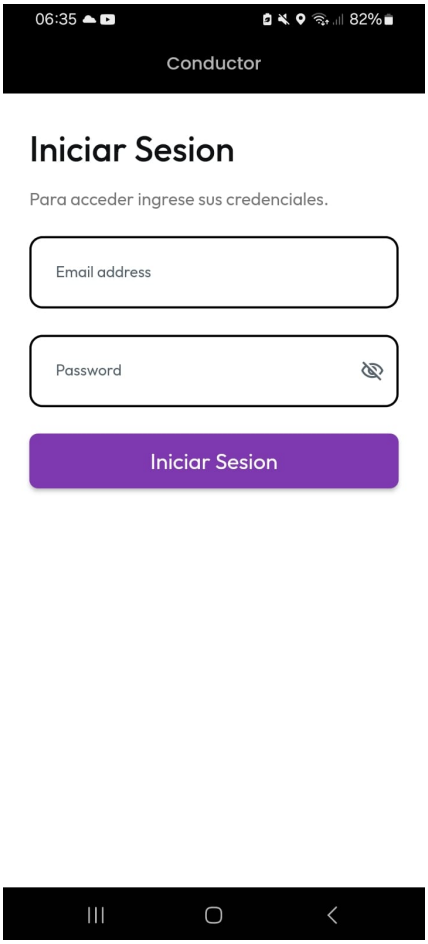


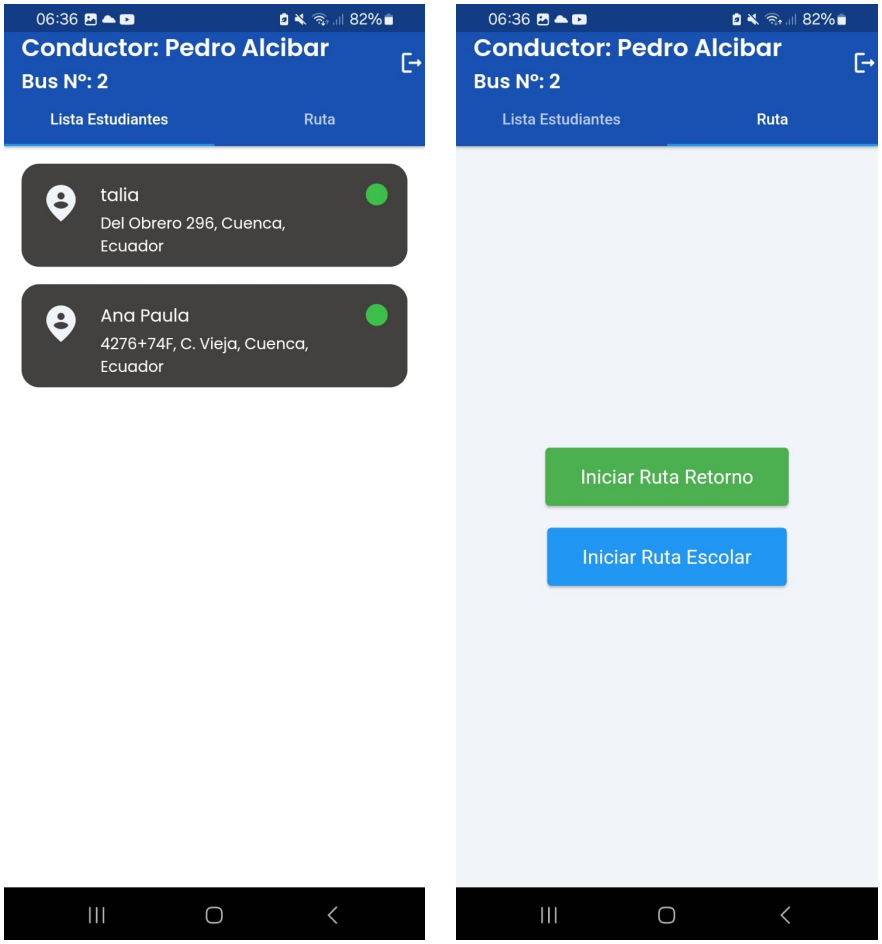
Figura 3.3: Interfaz de Inicio de Sesión del conductor.
Fuente: El Autor.

3.2.2. Vista de la Interfaz Principal del Conductor

Una vez iniciado sesión, el conductor podrá observar sus datos, como su nombre y el número de bus, en la parte superior de la pantalla. La interfaz principal del conductor ofrece dos opciones de cambio de pantalla como lo muestra la Figura 3.4.

La primera opción muestra la lista de estudiantes asignados, como se ilustra en la Figura 3.4(a)

La segunda opción “Ruta” permite al conductor escoger entre iniciar la ruta escolar o la ruta de retorno, como se muestra en la Figura3.4(b).

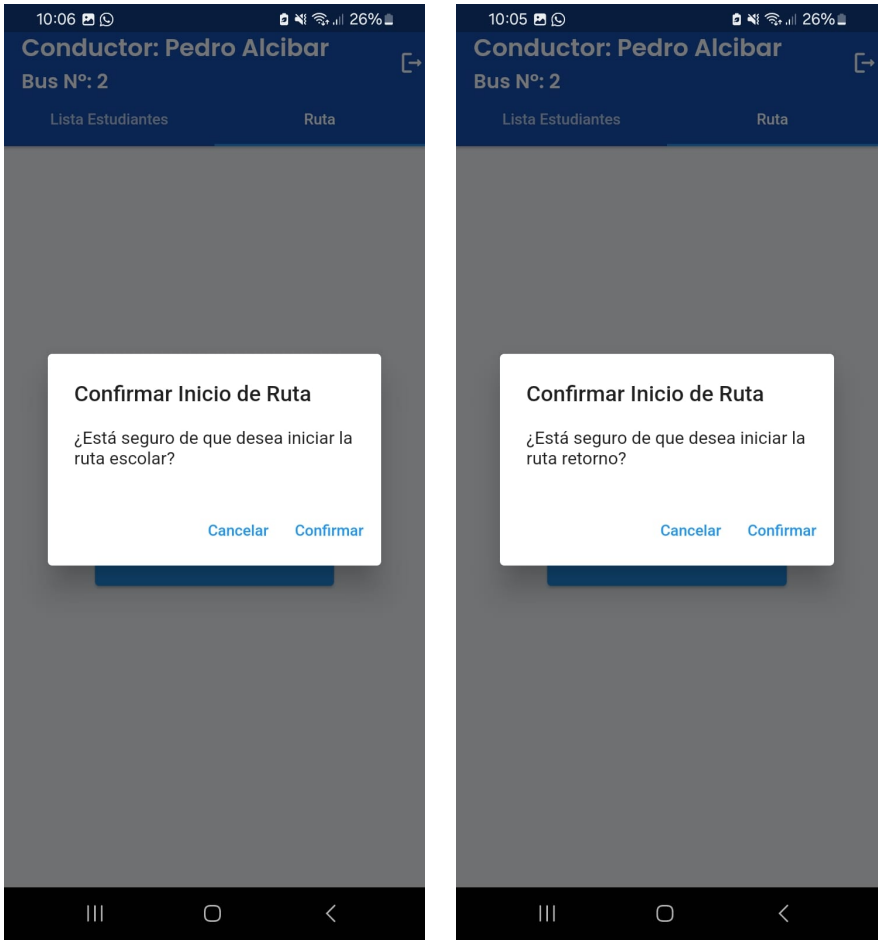


(a) Estudiantes Asignados al Conductor.

(b) Opción para Tipo de Ruta

Figura 3.4: Vista Interfaz Principal del Conductor.
Fuente: El Autor.

Al seleccionar una de estas opciones, aparecerá un mensaje de confirmación para asegurar que el conductor desea iniciar la ruta seleccionada. Este proceso se ilustra en la Figura 3.5



(a) Confirmación Ruta Escolar (b) Confirmación Ruta Retorno

Figura 3.5: Mensaje de Confirmación para Iniciar la Ruta.
Fuente: El Autor.

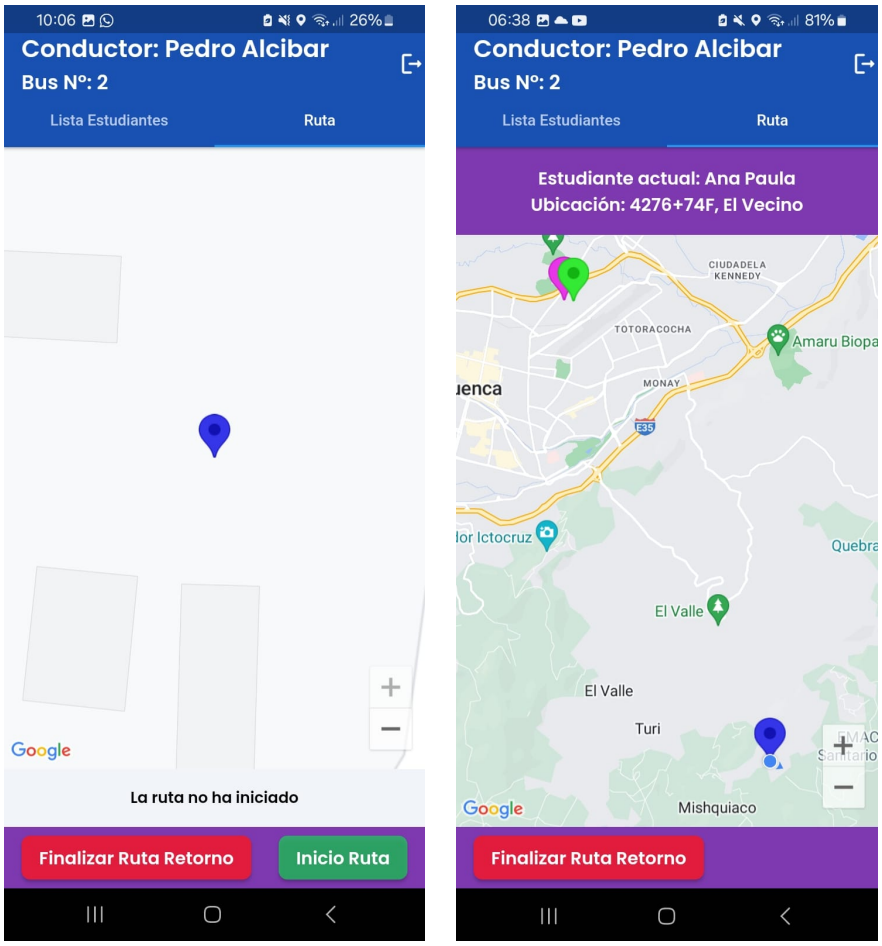
3.2.3. Visualización de la Ruta

Una vez el conductor haya iniciado la ruta y dependiendo de la opción que haya elegido, podrá observar su ubicación actual y la ubicación de los estudiantes.

- **Ruta de Retorno**

  El conductor podrá ver el mapa únicamente con su ubicación actual y un botón con la opción de “Finalizar Ruta” o “Iniciar Ruta” como se muestra en la Figura 3.6. A medida que los estudiantes suben y marcan su tarjeta, irán apareciendo los marcadores con la respectiva ubicación de los estudiantes. Además, se notificará a los padres de familia que el estudiante ha subido al medio de transporte. Cuando el conductor presiona Ïniciar Ruta", los marcadores cambiarán de color,

con el estudiante más cercano en verde y los demás en naranja. Se enviará una notificación al padre de familia indicando que la ruta de retorno ha iniciado. Una vez el estudiante baje y marque la tarjeta, se enviará una notificación al padre de familia y el marcador con la ubicación del estudiante se colocará en rojo, mientras que, el marcador con la ubicación del siguiente estudiante en ser dejado en su parada se pondrá en verde.



(a) Ruta de Retorno no Iniciada (b) Ruta de Retorno Iniciada

Figura 3.6: Pagina para la opción ruta de Retorno.
Fuente: El Autor.

- **Ruta Escolar**

Al iniciar esta ruta, el conductor podrá visualizar su ubicación actual y la ubicación de los estudiantes que debe recoger como se muestra en la Figura 3.7. Una vez que el estudiante suba al transporte escolar y marque la tarjeta, el marcador con la ubicación del estudiante se colocará en rojo, y se enviará una

notificación al padre de familia indicando que su hijo ha subido al autobús y al mismo tiempo el marcador del siguiente estudiante en ser recogido se pondrá en verde, y se enviará una notificación al padre de familia indicando que este será el próximo en ser recogido. Al llegar a la institución educativa, el conductor deberá presionar el botón "Finalizar Ruta"para enviar una notificación a los padres indicando que los estudiantes han llegado a la institución.

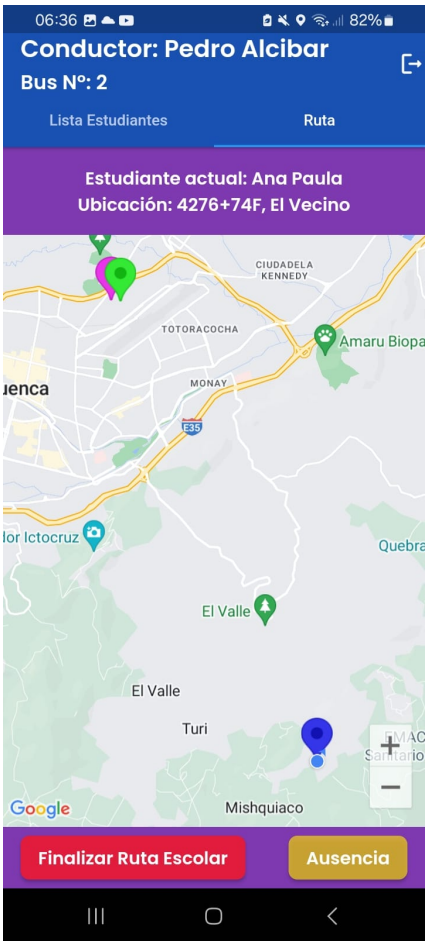


Figura 3.7: Ruta Escolar.
Fuente: El Autor.

Notificaciones a los padres de familia o representantes

Durante la operación, se enviarán notificaciones a los padres en diferentes momentos clave:

- **Inicio de la Ruta de Retorno:** Se notificará a los padres que la ruta de retorno ha comenzado.

- **Subida del Estudiante:** Al marcar la tarjeta, se notificará al representante o padre de familia que su hijo ha subido al autobús.
- **Bajada del Estudiante:** Al marcar la tarjeta al bajar, se notificará al representante o padre de familia que su hijo ha bajado del autobús.
- **Próximo estudiante en ser Recogido:** Se notificará al representante o padre de familia que su hijo será el proximo en ser recogido de su parada.
- **Finalización de la Ruta:** Al finalizar la ruta, se notificará a todos los representantes o padres de familia que la ruta ha terminado y que todos los estudiantes han sido entregados en su parada.

3.3. Pruebas de operación: Módulos Representante o Padre de Familia

3.3.1. Inicio de sesión

Para probar el funcionamiento de la aplicación, se utilizarán las credenciales de un usuario, descritas en la Figura 3.8, que tendrá el rol de Padre de Familia o Representante.

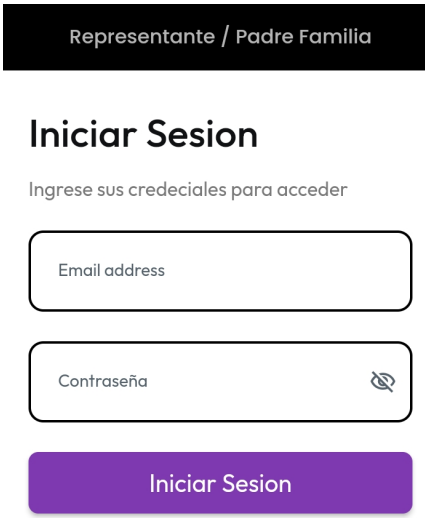


Figura 3.8: Pantalla Inicio de Sesión Estudiante.
Fuente: El Autor.

3.3.2. Vista de Estado de Ruta

Una vez que el padre de familia o representante del estudiante inicie sesión, podrá monitorear el recorrido del conductor en tiempo real en el mapa, siempre y cuando el conductor haya iniciado la ruta. En esta vista, se puede observar el tipo de ruta (escolar o retorno) en la parte superior de la pantalla, además de un icono para las notificaciones y uno para cerrar sesión.

También podrá ver el estado de la ruta, es decir, si ha iniciado o no. En la parte inferior de la pantalla, se mostrarán los datos del conductor de la buseta escolar, como el nombre y número de buseta, además del tiempo estimado de llegada de la buseta y la distancia a la que se encuentra.

También se podrá observar si el estudiante está o no a bordo de la buseta y si ha confirmado o no el uso del transporte escolar como se muestra en la Figura 3.9.

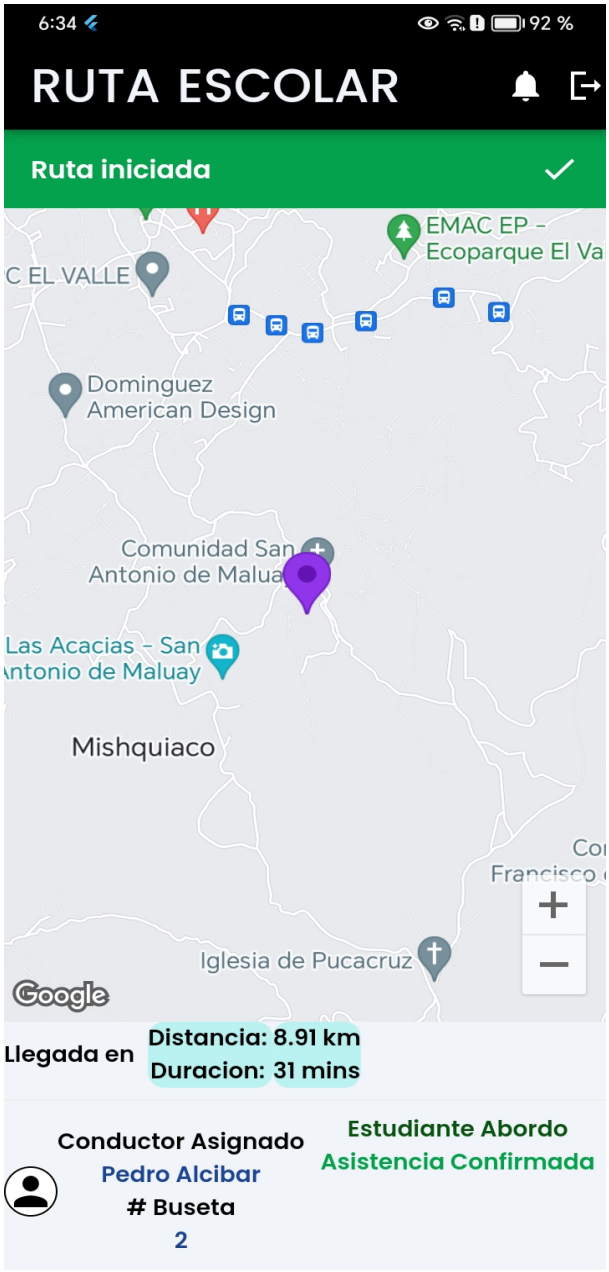


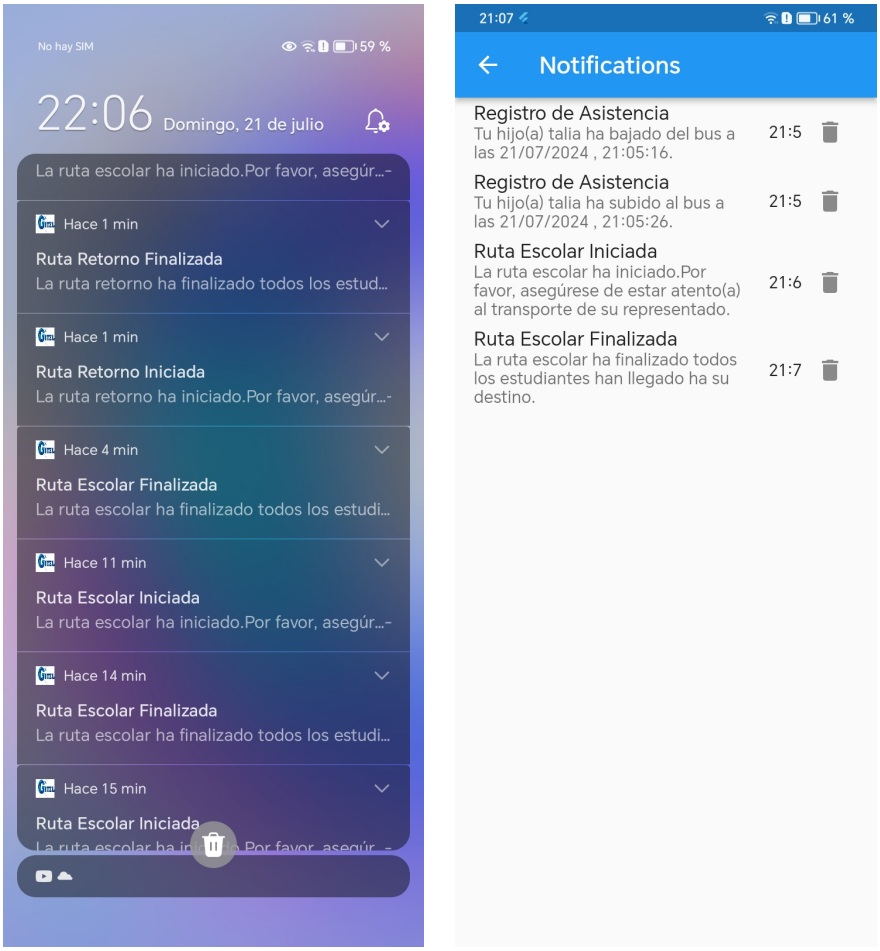
Figura 3.9: Pantalla Cuando la Ruta Inicia.
Fuente: El Autor.

En caso de que la ruta aún no haya iniciado, el padre de familia verá una pantalla con los datos del estudiante, y la opción de confirmar asistencia, la cual sirve para saber si va a ocupar o no el transporte escolar, como se muestra en la Figura 3.10

Figura 3.10: Pantalla cuando la Ruta Finaliza.
Fuente: El Autor.

3.3.3. Vista de Notificaciones

Las notificaciones llegarán al padres de familia o representante del estudiante, ya sea que la aplicación esté abierta o no, como se muestra en la Figura 3.11(a). Los padres podrán ver todas las notificaciones recibidas haciendo clic en el icono de notificaciones, como se ilustra en la Figura 3.11(b). Esta funcionalidad asegura que los representantes o padres de familia estén siempre informados sobre el estado y ubicación de sus hijos durante el trayecto en el transporte escolar.



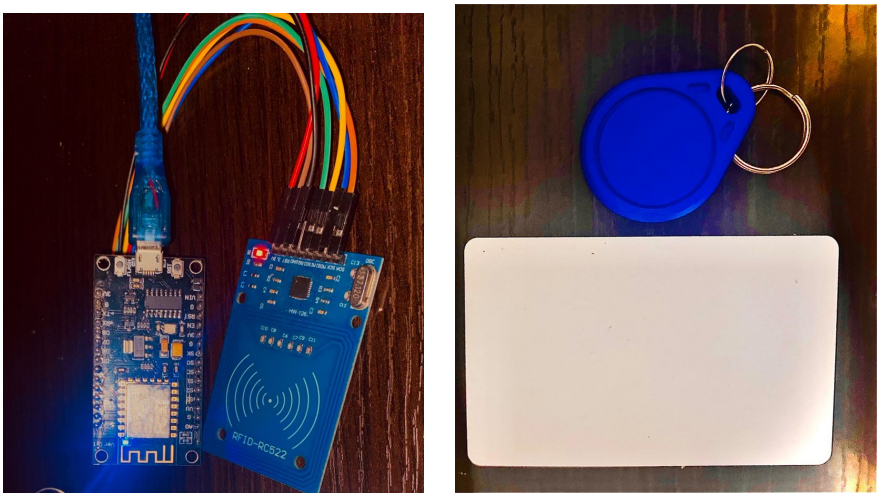
(a) Visualización de Notificaciones recibidas dentro de la aplicación

(b) Visualización de Notificaciones recibidas fuera de la aplicación

Figura 3.11: Notificaciones Recibidas.
Fuente: El Autor.

3.4. Montaje del Sistema RFID

Se implementó el sistema RFID con ESP8266 para enviar las notificaciones a los padres de familia o representantes de los estudiantes cuando estos suben o bajen del transporte escolar real. La Figura 3.12(a) muestra el montaje físico del sistema, mientras que la Figura 3.12(b) indica las tarjetas que utilizaran los estudiantes.



(a) Módulo ESP8266+RC55 (b) Tarjeta RFID

Figura 3.12: Ensamblaje Fisico del Sistema RFID.
Fuente: El Autor.

3.5. Pruebas de Rendimiento de la Aplicación

Para evaluar el rendimiento de la aplicación desarrollada, se utilizó Flutter DevTools, una herramienta avanzada proporcionada por Flutter que ofrece diversos recursos para mejorar el rendimiento y depurar aplicaciones. Flutter DevTools permite analizar la velocidad de la aplicación, identificar cuellos de botella y optimizar el código de manera eficiente.

La prueba de rendimiento se llevó a cabo ejecutando la aplicación en un entorno controlado y utilizando las funcionalidades de Flutter DevTools para monitorear el tiempo de procesamiento de los fotogramas (del ingles *frames*), la rasterización, y la compilación de shaders. Además, se prestó especial atención a los indicadores de *jank*, que representan retrasos en el renderizado y afectan la fluidez de la aplicación.

A continuación, se presenta la Figura 3.13, que muestra los resultados del análisis de rendimiento de la aplicación utilizando Flutter DevTools. La imagen ilustra el tiempo de procesamiento de la interfaz de usuario (UI), la rasterización y los fotogramas lentos (*jank*) durante la ejecución de la aplicación.

Las barras azules claras representan el tiempo dedicado al procesamiento de la UI, las barras azules oscuras muestran el tiempo de rasterización, y las barras rojas indican los fotogramas lentos que no lograron mantener la tasa deseada de 60 FPS. Las

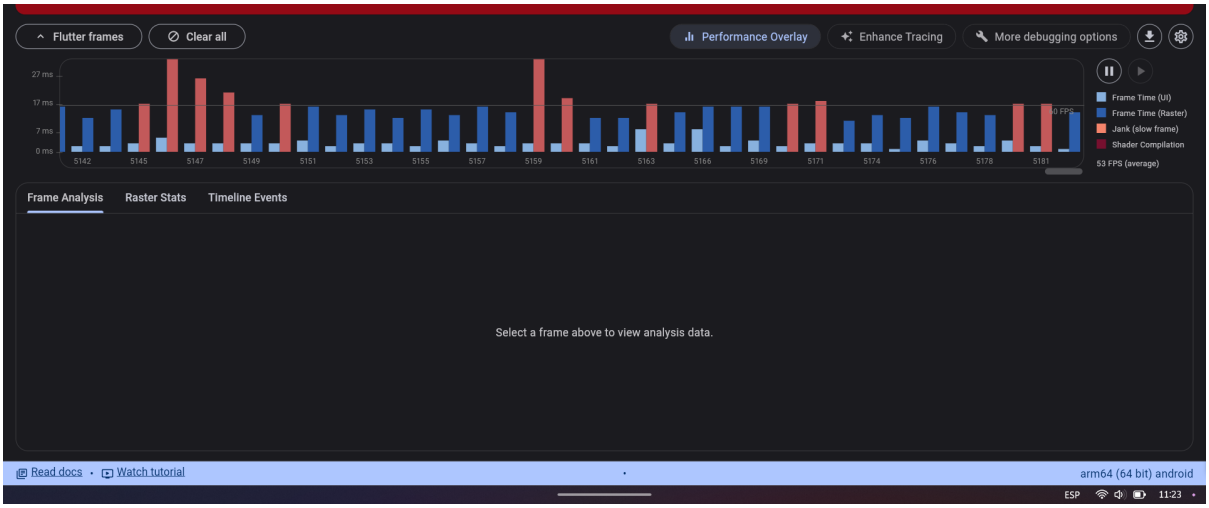


Figura 3.13: Análisis de rendimiento de la aplicación utilizando Flutter DevTools
Fuente: El Autor.

barras rosas representan el tiempo dedicado a la compilación de shaders.

Estos resultados destacan la importancia de optimizar tanto el código de la UI como los gráficos para asegurar una experiencia de usuario fluida y eficiente. Mediante el uso de Flutter DevTools, se pueden identificar y corregir problemas de rendimiento específicos, mejorando así la calidad general de la aplicación.

3.6. Resultado del consumo de datos

Para evaluar el consumo de datos del módulo ESP8266 durante la operación, se realizaron pruebas de monitoreo utilizando una conexión a Internet proporcionada por un teléfono celular. La tabla 3.1 presenta el consumo de datos en diferentes intervalos de tiempo.

Tabla 3.1: Consumo de datos del módulo ESP8266 en distintos intervalos de tiempo.

Tiempo	Consumo de Datos
5 min	780 KB
10 min	1.32 MB
15min	1.78 MB

El análisis del consumo de datos se realizó considerando tanto la operación del módulo ESP8266, encargado de la lectura de etiquetas RFID y la comunicación con Firebase, como la aplicación de monitoreo utilizada por el conductor para seguir la

ruta y administrar la asistencia de los estudiantes.

El consumo de datos fue medido en diferentes intervalos de tiempo y se observó que sigue un patrón lineal, como lo indica la tabla 3.1. Este patrón lineal permite prever de manera precisa el consumo total en escenarios de uso prolongado.

Dado que el conductor utiliza la aplicación de monitoreo durante aproximadamente 4 horas al día, el consumo diario estimado es de 28.48 MB. Extrapolando este consumo a un uso mensual, se estima un total de aproximadamente 854.4 MB.

Este análisis incluye tanto la transferencia de datos del módulo ESP8266 al servidor como la actualización continua de la ubicación del conductor y la interacción con la interfaz de la aplicación.

Capítulo 4

Conclusiones, Recomendaciones y Trabajos Futuros

4.1. Conclusiones

Se logró prototipar efectivamente la operación de la aplicación móvil, creando esquemas y diseños los cuales ayudaron para el desarrollo del proyecto. Los prototipos incluyeron las interfaces de usuario tanto para los conductores como para los padres de familia, asegurando que dicha interfaz sea amigable con el usuario. El prototipo inicial permitió realizar ajustes y mejoras antes de proceder con el desarrollo completo, lo que contribuyó a una implementación más eficiente para la interfaz del usuario.

Se implementó un servidor *backend* utilizando Firebase, el cual proporciona una infraestructura robusta para la autenticación de usuarios, almacenamiento de datos y envío de notificaciones push. Además el uso de este permite manejar los datos de manera eficiente y segura, garantizando que la información de ubicación y asistencia de los estudiantes estuviera siempre actualizada y accesible para los usuarios autorizados.

El *frontend* de la aplicación se desarrolló en Flutter, proporcionando una interfaz de usuario intuitiva y responsiva. La aplicación permite a los conductores visualizar la ubicación de los estudiantes, registrar subidas y bajadas mediante tarjetas RFID, y notificar a los padres sobre el inicio y finalización de la ruta. Los padres pueden recibir

notificaciones instantáneas, seguir la ruta del conductor en tiempo real y confirmar la asistencia de sus hijos. La integración de Google Maps y la API de Geolocalización fue esencial para mostrar ubicaciones de los estudiantes y para que el padre de familia pueda visualizar la ruta que lleva el conductor además del tiempo de llegada del transporte escolar en la parada del estudiante.

Se verificó el funcionamiento tanto del *backend* como del *frontend*. Las pruebas demostraron que la aplicación puede gestionar datos en tiempo real de manera efectiva, proporcionando actualizaciones precisas y notificaciones inmediatas a los usuarios.

La implementación del sistema RFID utilizando el módulo ESP8266 ha demostrado ser una solución eficiente para el registro y notificación de la asistencia de los estudiantes. El ESP8266 se conecta a la red Wi-Fi del dispositivo móvil y comunica los eventos de RFID al Firebase, lo que permite enviar notificaciones push a los padres de familia. Este sistema ha funcionado de manera confiable durante las pruebas, registrando correctamente las subidas y bajadas de los estudiantes.

El desarrollo de una aplicación móvil de monitoreo de transporte escolar requiere una combinación de habilidades en desarrollo de software, diseño de interfaz de usuario, administración de bases de datos y geolocalización.

4.2. Recomendaciones

Se recomienda mantener la actualización con las últimas versiones de Flutter y *Firebase* para aprovechar nuevas funcionalidades y mejoras de seguridad. Revisar la documentación para asegurarse que la aplicación se mantenga actualizada.

Para el caso de las notificaciones se recomienda ver que este habilitada API de *Firebase Cloud Messaging* (V1) para que de esta manera no haya problemas al momento de enviar notificaciones a los diferentes usuarios, ademas asegurarse de que cada usuario tenga un token único.

Algunas vulnerabilidades pueden aparecer en las versiones antiguas y estás expuestas a violaciones de seguridad, es por esto que se recomienda realizar pruebas de seguridad para identificar y reducir estas vulnerabilidad en la comunicación de

datos entre el dispositivo móvil y servicios de backend.

4.3. Trabajos Futuros

Como trabajo futuro para la aplicación móvil se sugiere implementar un botón en la interfaz del conductor para que él pueda notificar en forma rápida si existe algún imprevisto durante el viaje de transporte escolar, así los padres de familia y la institución educativa estarían notificados frente a una posible demora.

También, se podría agregar la visualización de rutas sugeridas en el mapa mediante polilíneas para que los conductores vean la ruta óptima sugerida directamente facilitando la navegación y mejorando la eficiencia del recorrido.

Además, se podría desarrollar una interfaz específica para los administradores de la institución educativa, puedan seguir en tiempo real la ruta de los conductores.

Glosario

APP Aplicación móvil.

BaaS Backend as a Service – Backend como un servicio.

Backend Es el encargado de conectar el servidor que utiliza un sitio web o aplicación con la base de datos..

Framework Entorno de trabajo.

Frontend Área de un sitio web o aplicación que el usuario puede interactuar..

GPS Sistema de posicionameinto Global.

OS Operating System – Sistema operative..

RFID Radio Frequency Identification – Identificación por radiofrecuencia.

Referencias

[1] C. Vasconez, *Transporte escolar, buses urbanos y taxis tendrán cámaras de seguridad*, jul. de 2022. dirección: https://www.expreso.ec/guayaquil/transporte-escolar-buses-urbanos-taxis-tendran-camaras-seguridad-130787.html.

[2] I. Montanilla, *Colegios de Quito usan 'app' para monitorear transporte escolar - El Comercio*, mar. de 2022. dirección: https://www.elcomercio.com/actualidad/quito/colegios-quito-app-transporte-escolar.html.

[3] J. G. Sánchez y J. G. Yvimas, «Desarrollo de una aplicación móvil para el monitoreo de rutas de transportes escolares,» Tesis doct., Universidad Central de Venezuela, 2018. dirección: http://saber.ucv.ve/bitstream/10872/19660/1/TEG%20JosueSanchez%2C%20JoseYvimas.pdf.

[4] F. S. Illanes Rumiguano, «Desarrollo de una aplicación Web-móvil para rastreo, seguimiento y reporte de contingencias del transporte escolar. Caso de estudio: Unidad Educativa San José La Salle,» 2021.

[5] P. C. W. Esteban et al., «Herramienta tecnológica para el control y monitoreo de las rutas escolares del Colegió Príncipe de Paz,» *Universidad Nacional Abierta y a Distancia*, 2021. dirección: https://repository.unad.edu.co/bitstream/handle/10596/41334/wepintoc.pdf?sequence=1&isAllowed=y.

[6] J. M. Vinueza Carranza, «Aplicación web-móvil para la geolocalización del recorrido escolar de los estudiantes de la unidad educativa Espíritu Santo,» *Universidad Regional Autónoma de los Andes*, 2020. dirección: https://dspace.uniandes.edu.ec/bitstream/123456789/11095/1/PIUASIS001-2020.pdf.

[7] G. F. Nama, F. H. Rasyidy, R. A. Setia Pribadi et al., «A Real-time Schoolchild Shuttle Vehicle Tracking System Base on Android Mobile-apps-Full Cover,» *International Journal of Engineering &Technology (IJET)*, vol. 7, n.º 3.36, págs. 40-44, 2018.

[8] E. Windmill, *Flutter in action*. Simon y Schuster, 2020.

[9] F. G. Yazbek Almeida, «Implementación del Backend y el Frontend para una empresa de servicio de mantenimiento de equipos de Laboratorio Clínico,» B.S. thesis, Universidad Politécnica Salesiana, 2022.

[10] M. M. Mora, «Çonstruir una aplicación móvil para Caarry Soluciones SA que se encargue de juntar los servicios de personas independientes y que tengan conocimientos en una determinada área",» *Universidad Católica de Santiago de Guayaquil*, 2021. dirección: `http://repositorio.ucsg.edu.ec/handle/3317/17566`.

[11] J. A. Cuadrado Hidalgo e I. M. Zambrano Domínguez, «ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL Y UN SISTEMA WEB PARA GESTIONAR EL INGRESO DE VISITAS A LA URBANIZACIÓN "LA GEMA",» B.S. thesis, Universidad Española de Madris, 2022. dirección: `http://repositorio.unemi.edu.ec//handle/123456789/6360`.

[12] J. Garrido Cobo, «Desarrollo de aplicaciones móviles,» *Universitat Oberta de Catalunya*, 2013. dirección: `https://openaccess.uoc.edu/handle/10609/18528`.

[13] L. L. N. Delía, «Desarrollo de Aplicaciones Móviles Multiplataforma,» *Facultad de Informática Universidad Nacional de La Plata*, 2017. dirección: `https://sedici.unlp.edu.ar/handle/10915/60497`.

[14] Y. Fain, V. Rasputnis, A. Tartakovsky y V. Gamov, *Enterprise web development: Building HTML5 applications: from desktop to mobile*. .ºReilly Media, Inc.", 2014.

[15] D Villalón, «Crear y desarrollar una aplicación de alto rendimiento con bajo coste utilizando Flutter y Firebase,» *Escola Tècnica Superior d'Enginyeria Informàtica Universitat Politècnica de València*, 2021.

[16] I. Costa, J. Araujo, J. Dantas, E. Campos, F. A. Silva y P. Maciel, «Availability evaluation and sensitivity analysis of a mobile backend-as-a-service platform,» *Quality and Reliability Engineering International*, vol. 32, n.º 7, págs. 2191-2205, 2016.

[17] F. Gropengießer y K.-U. Sattler, «Database backend as a service: automatic generation, deployment, and management of database backends for mobile applications,» *Datenbank-Spektrum*, vol. 14, págs. 85-95, 2014.

[18] K. S. W. Herrera, «Rediseño de front-end y back-end de una aplicación web de un sistema de administración escolar,» *niversitat Politècnica de Sinaloa*, 2020. dirección: `http : / / repositorio . upsin . edu . mx / Fragmentos / tesinas / A077WILSONHERRERAKENNETHSTONEY10302.pdf`.

[19] M. Chinchay Cuenca, *Desarrollo de una aplicación móvil Android para la búsqueda de plazas disponibles en un parqueadero*, 2015.

[20] T. Escalante, J. Llorente, D Espinosa y J. Soberón, «Bases de datos y sistemas de información: aplicaciones en biogeografía,» *Rev. Acad. Colomb. Cienc*, vol. 24, n.º 92, págs. 325-341, 2000.

[21] L. A. Saltos Pérez, «Estudio comparativo entre bases de datos relacional y no relacional,» B.S. thesis, Babahoyo: UTB-FAFI. 2022, 2022.

[22] S. Uunonen, «Backend as a service in web development,» 2023.

[23] A. Ramírez Osuna et al., «Aplicación móvil de ayuda a la comunidad celiaca a través Flutter y Firebase,» 2022.

[24] S Graciela, P Ibarra, R Quispe, F. Mullicundo, D. Lamas y L Presente, «Herramientas y tecnologías para el desarrollo web desde el FrontEnd al BackEnd,» en *XXIII Workshop de Investigadores en Ciencias de la Computación (WICC 2021, Chilecito, La Rioja)*, 2021, págs. 963-968.

[25] R. I. Andy Alvarado y L. E. Andy Alvarado, «IDENTIFICACIÓN DE LAS COMPETENCIAS DIGITALES REQUERIDAS POR LOS ESTUDIANTES PARA EL DESARROLLO DE LA ASIGNATURA DE DESARROLLO DE APLICACIONES MÓVILES EN EL ISTTENA,» Tesis doct., Instituto Superion Tecnológico Tena, 2023. dirección: `http://localhost:8080/jspui/handle/123456789/318`.

[26] J. Sánchez, *Aplicación Móvil para georreferenciación y búsqueda de farmacias utilizando tecnología multiplataforma*, 2017. dirección: `https : / / dspace . unl . edu . ec / jspui / bitstream/123456789/19376/3/SC3A1nchez20Cuenca2C20Johanna20Cecibel.pdf`.

[27] D. A. Pantoja Pino, «Sistema multiplataforma para la gestión de rutas y monitoreo de los buses de la Universidad Técnica de Ambato.,» B.S. thesis, Universidad Técnica de Ambato. Facultad de Ingeniería en Sistemas . . ., 2023.

[28] L. Serrato Mares, R Villegas Téllez y D. Torres Fraustro, «Aplicación movil para el control de asistencia a eventos aplicando tecnologia RFID,» *Jóvenes en la ciencia*, vol. 3, págs. 156-159, 2020.

[29] Z. V. Vargas Vergara, «Sistema de control de acceso y monitoreo con la tecnología RFID para el departamento de sistemas de la universidad Politécnica Salesiana sede Guayaquil,» B.S. thesis, Universidad Politécnica Salesiana, 2013.

[30] K. L. Osuna, «Implementación de Flutter para el desarrollo de aplicaciones móviles nativas en iOS y Android,» *UNIVERSIDAD POLITÉCNICA DE SINALOA PROGRAMA ACADÉMICO DE INGENIERÍA EN INFORMÁTICA*, 2020.