UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ESCUELA DE INGENIERÍA EN MECATRÓNICA

TRABAJO DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN MECATRÓNICA

TEMA:

"APLICACIÓN MÓVIL PARA RASTREO DE AUTOBUSES"

AUTOR: DANNY MARCIAL FARINANGO GUAJAN

DIRECTOR: CARLOS XAVIER ROSERO

IBARRA-ECUADOR

DICIEMBRE 2020

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA
AUTORIZACIÓN DE USO Y PUBLICACIÓN
A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educacion Superior, realizo la entrega del presente trabajo a la Universidad Tecnica del Norte para que sea publicado en el Repositorio ´ Digital Institucional, para lo cual pongo a disposicion la siguiente información:

DATOS DEL AUTOR			
CÉDULA DE IDENTIDAD:	172467343 – 7		
APELLIDOS Y NOMBRES:	FARINANGO GUAJAN DANNY MARCIAL		
DIRECCIÓN:	*Cayambe*		
EMAIL:	dmfarinangog@utn.edu.ec - dxnnyfxrx@gmail.com		
TELÉFONO FIJO:	022129122	**TELÉFONO MÓVIL:**	0980431157
DATOS DE LA OBRA			
TÍTULO:	“APLICACIÓN MÓVIL PARA RASTREO DE AUTOBUSES”		
AUTOR:	DANNY MARCIAL FARINANGO GUAJAN		
FECHA (AAAA-MM-DD):	2020-12-10		
SÓLO PARA TRABAJOS DE GRADO			
PROGRAMA:	PREGRADO		
TÍTULO POR EL QUE OPTA:	INGENIERO EN MECATRÓNICA		
ASESOR/DIRECTOR:	CARLOS XAVIER ROSERO C.		

3. CONSTANCIAS

El autor manifiesta que la obra objeto de la presente autorización es original y se la desarrolló sin violar derechos de autor de terceros, por lo tanto la obra es original, y que es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 10 días del mes de Diciembre de 2020.

Danny Marcial Farinango Guajan
C.I.: 172467343 – 7

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS
CERTIFICACION

En calidad de director del trabajo de grado "APLICACIÓN MÓVIL PARA RASTREO DE AUTOBUSES", presentado por el egresado Danny Marcial Farinango Guajan, para optar por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, Diciembre de 2020

Carlos Xavier Rosero
DIRECTOR DE TESIS

Agradecimiento

A mis hermanos Marcelo, Sergio, Kevin, Oscar y Jamilton; quienes siempre estuvieron ahí para ayudarme, apoyarme y empujarme cuando más los necesitaba. Gracias hermanos son los mejores, Uds. son mi fortaleza para seguir.

A mi segunda familia, Ferretería El Constructor, especialmente al Sr. German Morocho y la Sra. Rebeca Lara estaré eternamente agradecido con ustedes, son el más claro ejemplo a seguir. Sus consejos y enseñanzas me ayudaron y sé que lo seguirán haciendo en el transcurso de mi vida. Sin dejar de lado a mis amigos de la empresa, gracias muchachos por ser tan buenas personas.

A mi director de tesis Carlos Xavier Rosero por compartir sus conocimientos y ayudarme a terminar este trabajo. Además, a los docentes que tuve a lo largo de mi carrera universitaria.

Danny Farinango

Dedicatoria

A mi querida madre Luz Divina por guiarme e inculcarme los valores y enseñanzas que ayudaron a formar mi caracter.

A mis hermanos y a toda mi familia que siempre estuvieron pendientes.

A mis amigos Eric, Cristian, Alex, Vanne, con quienes compartí muchas experiencias extraordinarias durante mi paso por la vida universitaria.

Danny Farinango

Resumen

Uno de los inconvenientes que tienen muchas de las personas al momento de movilizarse de un lugar a otro es desconocer el paradero del autobús que deben abordar, así como también el tiempo que tardará en llegar a determinada parada, causando retrasos en caso de no conocer el horario especifico en el cual el autobús pasará por aquel lugar. Por lo tanto, es importante proporcionar la hora exacta de llegada del autobús, ya que puede reducir el tiempo de espera en las paradas y decidir el plan de viaje. En este trabajo se propone un sistema de seguimiento de autobuses que tiene como objetivo minimizar el tiempo de espera mediante la incorporación de funciones que permiten a los viajeros conocer la ubicación en tiempo real y realizar consultas sobre la hora de llegada del autobús. El teléfono móvil se ha vuelto indispensable hoy en día, por este motivo se aprovecha esta ventaja realizando un sistema de seguimiento de autobuses utilizando tecnologías gratuitas que permitirán escalarlo y mejorarlo con el pasar del tiempo. El Sistema posee tres módulos: administrador, transporte y usuario para el desarrollo se obtiene sus requerimientos mediante un análisis previo, posteriormente con base a esto se realiza el diseño de las aplicaciones. Los módulos se sincronizan entre sí, es decir, el módulo del conductor envía datos de localización a la base de datos de Firebase, la almacena y la entrega al módulo del usuario, el cual lo interpreta y muestra en un mapa la ubicación actual del autobús o autobuses, además, se puede consultar el tiempo en el que la unidad llegará a las paradas establecidas; esta respuesta se realiza en menos de 4 segundos.

Abstract

One of the inconveniences that many of the people have when moving from one place to another is not knowing the whereabouts of the bus they must board, as well as the time it will take to get to a certain stop, causing delays if they do not know the specific time in which the bus will pass through that place. Therefore, it is important to provide the exact arrival time of the bus, as it can reduce the waiting time at the stops and decide the travel plan. This work proposes a bus tracking system that aims to minimize waiting time by incorporating functions that allow travelers to know the location in real time and make inquiries about the arrival time of the bus. The mobile phone has become indispensable nowadays, for this reason this advantage is taken advantage of by making a bus tracking system using free technologies that will allow it to be scaled up and improved over time. The System has three modules: administrator, transport and user. For the development, its requirements are obtained through a previous analysis, then based on this, the design of the applications is carried out. The modules are synchronized with each other, that is, the driver module sends location data to the Firebase database, stores it and delivers it to the user module, which interprets it and shows the current location of the bus on a map or buses, in addition, you can check the time in which the unit will arrive at the established stops; this answer is done in less than 4 seconds.

Índice general

Índice de figuras

Índice de cuadros

Lista de Programas

Capítulo 1

Introducción

En sección se describe la problemática y de qué manera solucionarla, así como también su importancia.

1.1. Problema

La movilización en transporte público constituye uno de los soportes más importantes para el desarrollo de la sociedad. Diariamente las personas utilizan este servicio para ir desde su casa a los lugares de trabajo, estudio, entre otros [1],[2].

Existen dos tipos de información que todos los sistemas de transporte deben proporcionar. El primero es la ruta y su programación (mapas, horarios y conexiones); el segundo es la información básica (política de tarifas, entre otras) [3]. Sin embargo, en la ciudad de Ibarra, dados los volúmenes de pasajeros y la insuficiencia de información básica y de rutas, los usuarios desperdician mucho tiempo al esperar unidades de transporte público.

Para solucionar el problema citado, en el año 2016 se implementaron con un costo elevado, ocho paradas inteligentes ubicadas en la avenida Mariano Acosta, desde el sector del Obelisco hasta el último semáforo del sector de La Florida [4]. En la actualidad, las mencionadas paradas

no funcionan correctamente y se encuentran deterioradas.

Como una alternativa de solución al problema, en diferentes países del mundo existen aplicaciones móviles para seguimiento de autobuses que han disminuido el tiempo de espera de las personas en las paradas. Estas aplicaciones son específicas para los sistemas de transportación de estos países. Así, se plantea como solución al problema en la ciudad de Ibarra, la utilización de teléfonos inteligentes para proporcionar información básica de rutas y de programación a cualquier usuario [5].

1.2. Objetivos

1.2.1. Objetivo General

Crear una aplicación móvil para rastreo de autobuses.

1.2.2. Objetivos específicos

- Determinar la funcionalidad de la aplicación móvil en base a modelos creados en otras plataformas y al funcionamiento de las paradas inteligentes de la ciudad de Ibarra.
- Desarrollar la aplicación móvil de rastreo de autobuses con base en la funcionalidad establecida.
- Validar la aplicación móvil en condiciones reales de uso para la determinación de su desempeño.

1.3. Justificación

El desarrollo de una aplicación móvil para la localización de autobuses de traslado de pasajeros se hace relevante en los siguientes aspectos:

En el ámbito de la movilidad, las personas necesitan trasladarse de forma rápida y segura de un lugar de residencia al sitio donde laboran o estudian, siendo este un factor de alta importancia para el desarrollo e integración de la sociedad.

En el aspecto económico, las paradas inteligentes instaladas en los diferentes lugares de la ciudad de Ibarra, como herramientas que facilitan la movilización de personas locales o extranjeras tienen un alto costo, por lo tanto, es necesario reducirlo. Un teléfono inteligente se ha convertido en una parte integral de la vida de las personas y podría ser aprovechado. Mediante utilización de tecnologías de hardware y software libre el costo podría ser reducido.

Según el VII censo nacional realizado en el año 2010, de los 14.483.499 habitantes que tiene el Ecuador, la población del cantón Ibarra alcanza el 181.175 [6]. El 80 % de esta ciudad utilizan el transporte público para movilizarse [7], convirtiéndolos en beneficiarios directos del sistema de seguimiento de autobuses, al igual que las organizaciones a cargo del transporte público urbano, quienes pueden vigilar las unidades con mayor facilidad.

En el aspecto tecnológico, la creación de sistemas de seguimiento podría generar nuevos temas de desarrollo útiles en diferentes aplicaciones.

1.4. Alcance

Se desarrollará una aplicación para teléfonos inteligentes para rastreo de autobuses, que tomará datos desde un sitio web o desde repositorios. Proveerá en tiempo real información de navegación de autobuses.

La aplicación contará con diferentes herramientas de visualización, los usuarios tendrán información de rutas y tiempos de llegada de los autobuses a las paradas. Este sistema será aplicado para un número de rutas determinado.

1.5. Estructura del documento

Además de la presente introducción el documento está conformado por cuatro capítulos y un apéndice. El Capítulo 2 presenta el estado del arte y en él se realiza una ánalisis de los sistemas existentes, tecnología necesaria y metodología a usar.

En el Capítulo 3, se describe y analiza las herramientas utilizadas para el desarrollo del sistema.

En el Capítulo 4,se describe la implementación del sistema del sistema y las pruebas realizadas.

El Capítulo 5 muestra las conclusiones de este proyecto y además se bosquejan algunas posibles líneas de trabajo futuro.

Finalmente, el Apéndice muestra el código del proyecto completo.

Capítulo 2

Revisión Literaria

En este capítulo se realiza un estudio del estado del arte de los diferentes sistemas de seguimiento así como también de las tecnologías usadas.

2.1. Ciudad y transporte inteligente

Una ciudad inteligente es aquella que mejora la calidad de vida de la persona, mediante el mejoramiento de servicios de una forma tecnológica [8] [9].
La inteligencia de una ciudad se muestra principalmente por su capacidad para reunir sus recursos y alcanzar los objetivos enfatizados y especialmente a través de la tasa de satisfacción de las necesidades de sus habitantes [10].

Unas de las posibilidades de las ciudades inteligentes es la movilidad personal, cómoda, barata y sostenible a sus ciudadanos. Los sistemas de trasporte inteligentes (ITS) tienen el objetivo de mejorar la seguridad y movilidad de los transportes, además, de aumentar la productividad de las personas y disminuir efectos nocivos de tránsito [11] [12].

Esa mejora se alcanza a través de la integración de tecnologías de comunicación. Por ende, el sistema de transporte será más seguro y eficiente.

2.2. Aplicaciones móviles de transporte

En el sector del transporte se utilizaba principalmente servicios de navegación y servicios basados en ubicación con aplicaciones móviles. A partir del 2016, las aplicaciones móviles relacionadas con el transporte incluyen la recopilación de datos de tráfico, información de viajes, planificación de rutas, viajes compartidos, entre otros [16] [17].

En la actualidad existe una variedad de aplicaciones móviles para el transporte público, este estudio tiene como objetivo revisar las aplicaciones potenciales en el transporte e identificar sus posibles beneficios y desventajas, para posteriormente mediante un análisis realizar el correcto diseño del sistema de transportación a desarrollar.

2.3. Análisis de sistemas de transporte existentes

2.3.1. Moovit

Es una aplicación gratuita con sede en Israel desarrollada para el área urbana, disponible para Android, IOS, y web. Su objetivo es ayudar a los usuarios a recorrer por la ciudad de manera simple y fácil, gracias a la gran cantidad de información que entrega[18] [19].

En la pantalla principal podemos observar que se muestran en tiempo real las unidades de trasporte dependiendo del lugar en donde se contrató el servicio y el lugar de ubicación del usuario Figura 2.1, además, la aplicacion posee diferentes opciones; por ejemplo la ubicación de usuarios cercanos.

Figura 2.1: Pantalla principal Moovit

Sin embargo, a diferencia de otros servicios, Moovit busca aprovechar a sus propios usuarios como fuente de información valiosa de tránsito. Lo que significa que los usuarios más consistentes pueden informar si el transporte en el que viajan está demasiado lleno o si el conductor está conduciendo muy rápido.

Moovit posee varias funcionalidades [20].
Puede ser usada para planificar excursiones por las ciudades y determinar en cuanto tiempo llegará el autobús o un tren en tiempo real, además busca horarios dependiendo en que ciudad haya sido configurada por el usuario, mantiene al usuario informado de incidencias de las líneas,

cambio de idioma, se conecta a redes sociales.

2.3.2. Movilízate UIO

Es una aplicación gratuita desarrollada por el Municipio del Distrito metropolitano de Quito, disponible para Android y IOS. El objetivo de esta herramienta es brindar información oportuna del transporte público y de BiciQuito a la ciudadanía local o extranjera; de modo que ayudará a mejorar la planificación de viajes [21] [22].

Además, la aplicación trabaja junto a la tecnología de Google Inc, que permite conocer rutas, horarios, generar denuncias en tiempo real acerca del servicio Figura 2.2. Sin embargo, la aplicación no ha tenido mucha aceptación por parte de los usuarios, debido a que por el momento cuenta con poca cantidad de información, además, algunas de las funcionalidades no trabajan correctamente, especialmente la de estimación de tiempo[23].

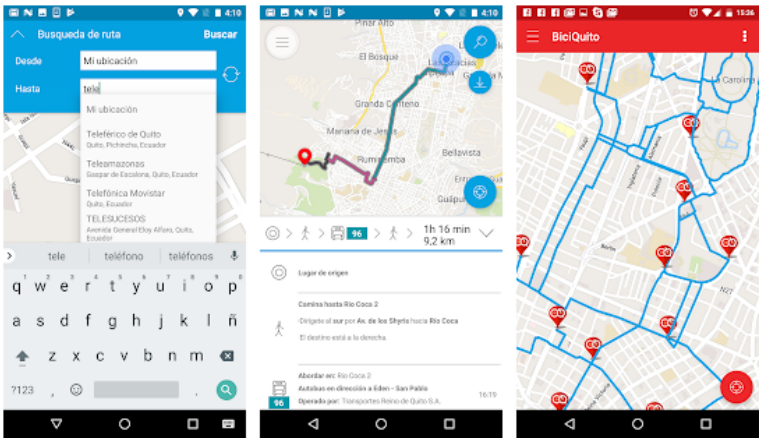

Figura 2.2: Funcionalidades de Movilizate UIO

Funcionalidades que ofrece Movilízate UIO[23]:
Búsqueda de rutas de transporte público; ubicación de estaciones del Sistema de Bicicleta Pública, BiciQuito; reporte de incidencias de los mencionados servicios, mapas, banco de preguntas

frecuentes, búsqueda de direcciones, reporte de accidentes.

2.3.3. Metrovía Guayaquil

Es una aplicación gratuita desarrollada por el consorcio de Tranvía, Operador Tecnológico y de Recaudo del Sistema Metrovía, está disponible para dispositivos Android, su objetivo es dar a conocer el tiempo de llegada de los buses a las diferentes paradas y terminales del Sistema [24], entonces, con este aplicativo permite a los usuarios ahorrar tiempo al trasladarse de un lugar a otro.

En la pantalla inicial de la aplicación Figura 2.3 se muestra la ubicación de las unidades en tiempo real. Para utilizarla basta con seleccionar en el mapa de Google la ubicación actual, luego se selecciona la parada donde se encuentra el usuario y automáticamente aparecerán todos los buses que se encuentran cerca al punto de destino.

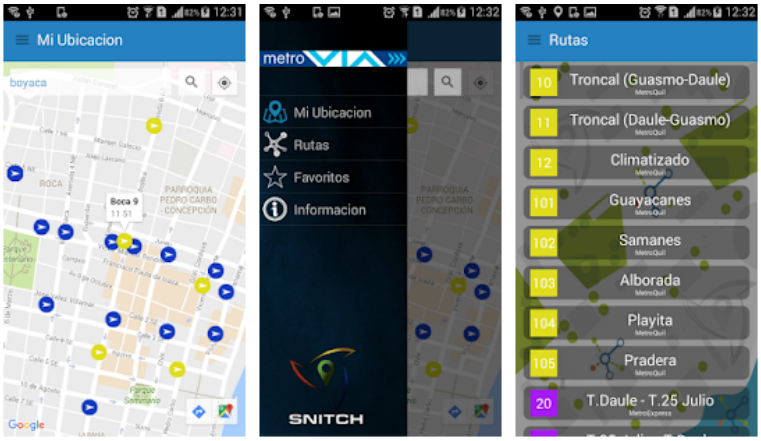

Figura 2.3: Funcionalidades de metrovía Guayaquil

Funcionalidades de Metro Arribo[25]

Visualizar las paradas cercanas a su ubicación actual o cercanas a una ubicación seleccionada.

Entrega los tiempos próximos de llegada de las unidades del sistema a cada punto de parada. Utilizando la herramienta, los usuarios del sistema pueden administrar de una forma más eficiente su tiempo conociendo los tiempos reales de llegada.

Otras funcionalidades que pueden realizar desde la aplicación:

- Ver el listado de las Rutas por Empresa o Troncal
- Ver el listado de Paradas de cada Ruta
- Visualizar en el mapa el recorrido de cada Ruta
- Guardar las paradas favoritas para facilitar las consultas

2.3.4. Paradas inteligentes Ibarra

Las primeras 8 paradas inteligentes fueron instaladas en el 2016, con el pasar de los años estas paradas han ido en aumento, para 2017 se esperaba contar con 50 paradas en la ciudad[26]. El objetivo de las paradas es informar a las personas locales y extrajeras el tiempo de llegada de autobuses a la parada donde se encuentran Figura 2.4, además, podrá visualizar la ruta por donde transitará la unidad que abordará, de modo que de esta forma el usuario ahorre tiempo y pueda desplazarse de forma sencilla.

Figura 2.4: Parada Inteligente Ibarra

Las paradas mencionadas poseen tecnología satelital que funciona mediante un dispositivo GPS instalado en los buses de transporte urbano, el cual proporciona datos de ubicación del bus, para simultáneamente enviar esos datos a la parada establecida, en la cual mediante una pantalla de 32 pulgadas se muestra en un mapa su ubicación, además, en la parte inferior el tiempo de llegada y la distancia a la que se encuentra la unidad.

En cada parada existen dos pantallas las cuales cumplen con diferentes funcionalidades Figura 2.5, la pantalla que se encuentra en la parte superior, se anuncia publicidad de los diferentes negocios de la ciudad. En la segunda pantalla se muestra todos los datos referentes al trasporte público, es decir, el mapa, ubicación del bus y tiempos de llegada.

Figura 2.5: Funcionamiento de paradas inteligentes Ibarra

Para evitar actos de vandalismo por parte de algunos ciudadanos cada parada tiene una cámara conectada al ECU 911[4].

Además, se desarrolló una aplicación para el seguimiento de autobuses en tiempo real llamada CityTouch Figura 2.6 , la cual funciona en teléfonos inteligentes con sistema operativo Android. El objetivo era que el usuario pueda consultar las rutas de autobuses para planear sus destinos y mirar el horario de llegada [4].

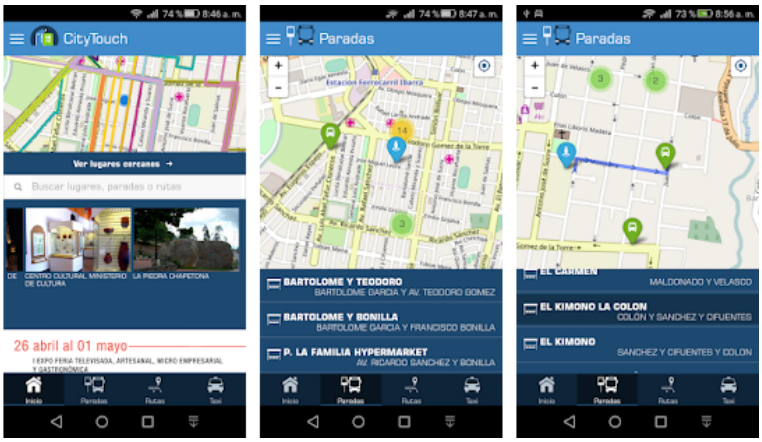

Figura 2.6: Funcionalidades aplicacación CityTouch

Funcionalidades de la aplicación móvil City Touch[27].

- Consulta las rutas de buses para planificar trayectos.
- Cálcula el tiempo estimado de llegado del autobús.
- Visualizar las paradas en el mapa.

Sin embargo, la empresa española Impacto encargada del manejo y mantenimiento del sistema inteligente de transportación, decidió dejar de prestar el servicio debido a que no pudo financiar el servicio con publicidad. Por lo que, MOVILDELNORT asumió el manejo del sistema[28].

Por otro lado, se analiza diferentes aplicaciones que poseen funcionalidades similares a las descritas anteriormente, en la la Tabla 2.1 se muestra las características clave de cada una de ellas.

Tabla 2.1: Comparación de sistemas de seguimiento

Nombre	Características clave
Uber,Cafiby, Easy Taxi.	Utiliza datos de ubicación tanto de usuario como de conductor/ muestra distancia de objetivo y rutas de destino en alguno de los casos.
ETM Madrid	Ofrece encontrar todo lo referente al transporte de tu ciudad de forma gratuita, incluido por ejemplo los tiempos de espera si vas a coger alguna línea de bus.
Citymapper	Se utiliza en todos medios de transporte, busca el itinerario más eficiente permite guardar paradas, rutas, acceso a horarios ofrece ruta más rápida.
Urban Step	Utiliza diferente versión dependiendo de la ciudad, consulta de horarios, rutas, planificar recorridos y si activa el GPS se encuentra la parada más cercana.
RACC Trips	Información sobre transportes metropolitanos, bicicletas, coches, motos, tiempos de llegada de transporte, rutas.
EMT Valencia	Ofrece información sobre transporte sostenible, horarios o paradas, rutas, tiempos de llegada, la aplicación también permite recargar el bonobús o el abono mensual.

Una vez realizado el estudio de diferentes plataformas de seguimiento de autobuses nacionales e internacionales, se concluye que es necesario utilizar un dispositivo que genere los datos de posición del bus, los envié a una base de datos(servicio web) y posteriormente se visualice en un dispositivo los datos de interés.

En el sistema a desarrollar es muy importante la velocidad de actualización de los datos de posición del autobús; por lo cual, es necesario realizarlo en intervalos de tiempos cortos, para no tener inconvenientes al entregar información de tiempos de llegada a estaciones a los usuarios.

Además, para evitar confusiones cada autobús tendrá una ruta diferente, si todos los autobuses se mueven en un solo mapa, los usuarios pueden entrar en confusión al buscar el autobús que desean tomar, por lo tanto, una de las ventajas de los sistemas presentados anteriormente es diferenciar las rutas del autobús en un mapa diferente, de modo que se toma como base aquella función.

2.4. Desarrollo de aplicaciones móviles

Para un desarrollo eficaz de una aplicación móvil se debe tomar en cuenta cinco etapas: etapa de análisis, donde se obtienen, se clasifican los requerimientos y se personaliza el servicio; etapa de diseño, momento en el que se define el escenario tecnológico, se estructura la solución por medio de algún diagrama o esquema; etapa de desarrollo, cuando se implementa el diseño en un producto de software; etapa de prueba de funcionamiento, donde se emula y simula el producto ajustando detalles, se instala en equipos reales, se evalúa el rendimiento para posteriormente se evaluar el potencial de éxito; y finalmente, en la etapa de entrega, se define el canal de distribución de la aplicación, con el propósito de adecuar la aplicación al mismo. [29].

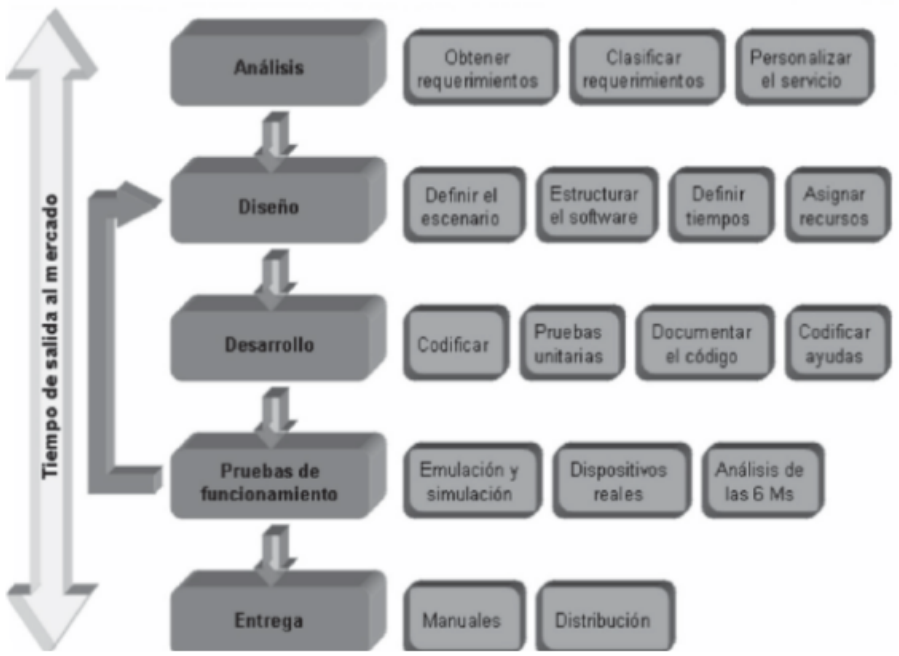

Figura 2.1: Etapas de la metodología para el desarrollo de aplicaciones móviles

2.5. Gestión de Base de Datos

Es un conjunto estructurado, organizado y categorizado de datos que representa entidades y sus interrelaciones. La representación será única e integrada, a pesar de que debe permitir utilizaciones varias y simultáneas [30] [31]. Los usuarios no necesitan conocer los detalles del almacenamiento, solo requieren tener una vista abstracta de los datos [32].

2.5.1. Sistemas de gestión de base de datos

Un SGBD(Sistemas de gestión de base de datos) es una herramienta de propósito general útil para estructurar, almacenar y controlar los datos ofreciendo interfaces de acceso a la base de datos basada en una arquitectura de tres niveles(externo conceptual e interno) [32].

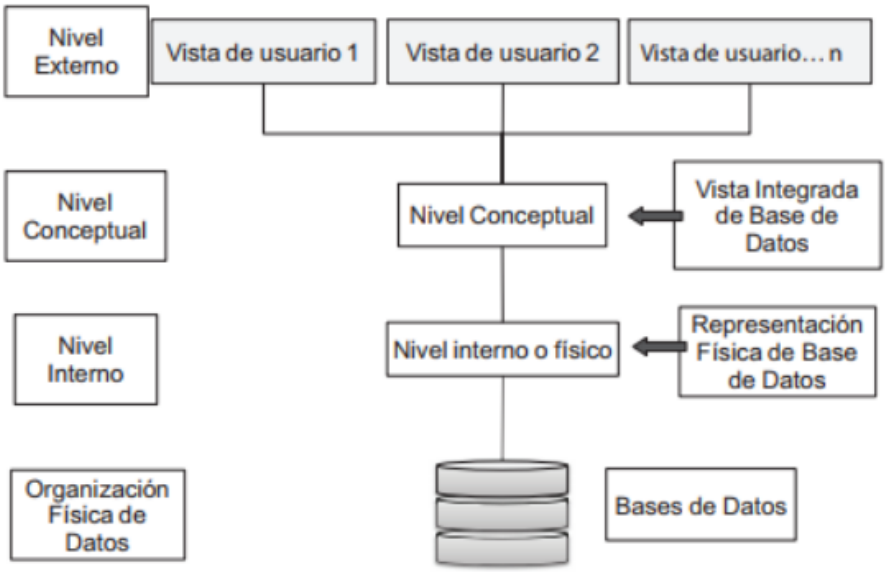

Figura 2.2: Arquitectura ANSI-SPARC

2.5.2. Base de datos relacional

Es un repositorio compartido de datos. Permite establecer interconexiones entre ellos los cuales están guardados en tablas y a través de ellas relacionarlos [33]. Este tipo de modelo es el

más utilizado para implementar aplicaciones.

2.5.3. Base de datos no relacional

NoSQL es un sistema de gestión de base de datos que difiere de modelo clásico del sistema de gestión de base de dato relacionales [33]. Los datos no requieren estructuras fijas como tablas, además, almacenan la información sin cumplir el esquema entidad–relación [30]. Este modelo se utiliza principalmente cuando es necesario manejar grandes volúmenes de datos dando la mayor importancia al rendimiento que a la coherencia [11].

2.6. Sistema operativo para desarrollo de aplicaciones móviles

Mediante el estudio realizado en [34] se logra determinar que los sistemas operativos dominantes en el mercado son Android, IOS, y Windows Phone, Por lo tanto, es necesario realizar una comparación de las plataformas que tengan mayor demanda en el mercado Tabla 2.2, es decir, describir las características principales de las plataformas móviles disponibles en la actualidad, para posteriormente elegir una que se adapte a nuestras necesidades.

Tabla 2.2: Comparación de plataformas de desarrollo

	Aple iOs 9	**Android**	**Windows Phone**
Núcleo del SO	Mac OS X	Linux	Windows NT
Año de lanzamiento	2007	2008	2010
Variedad de los dispositivos	Muy alta	Muy alta	Media
Soporte memoria externa	No	Si	Si
Motor de navegador web	Webkit	Webkit/chromium (blink)	Trident
Tienda de aplicaciones	App store	Google play	Windows MarketPlace
Número de aplicaciones	2.400.000 (sept 2016)	2.000.000 (junio 2016)	700.000 (oct. 2016)
Coste publicar	$ 99/año	$25 una vez	$99 / año
Soporte 64 bits	Si	Si	No
Máquina virtual	No	Dalvik /ART	.net
Lenguaje de programación	Objective-C, Swift	Java, C++	C#,visual Basic, C++
Plataformas de desarrollo	Mac	Windows, Mac, Linux	Windows
Multiusuario	No	Si	No

Como se puede apreciar en la tabla de comparación, el Sistema IOS y Android, poseen una variedad de dispositivos en el mercado, sin embargo, para el sistema a desarrollar es necesario reducir al máximo los costos, por lo tanto, la plataforma que cumple con este requerimiento es Android, además, es de código abierto y se lo puede desarrollar en el lenguaje de programación Java, en el ya se tiene experiencia.

Otro aspecto fundamental a la hora de comparar las plataformas móviles es su cuota de mercado. En la Figura 2.3 se muestra la evolución del mercado de los sistemas operativos para móviles según el número de terminales vendidos. Podemos destacar la desaparición de la plataforma Symbian de Nokia, el declive continuo de BlackBerry, el estancamiento de la plataforma de Windows, y el afianzamiento de la cuota de mercado de Apple en torno al 15 %. Finalmente,

cabe señalar el espectacular ascenso de la plataforma Android, que en cinco años ha alcanzado una cuota de mercado superior al 80 %.

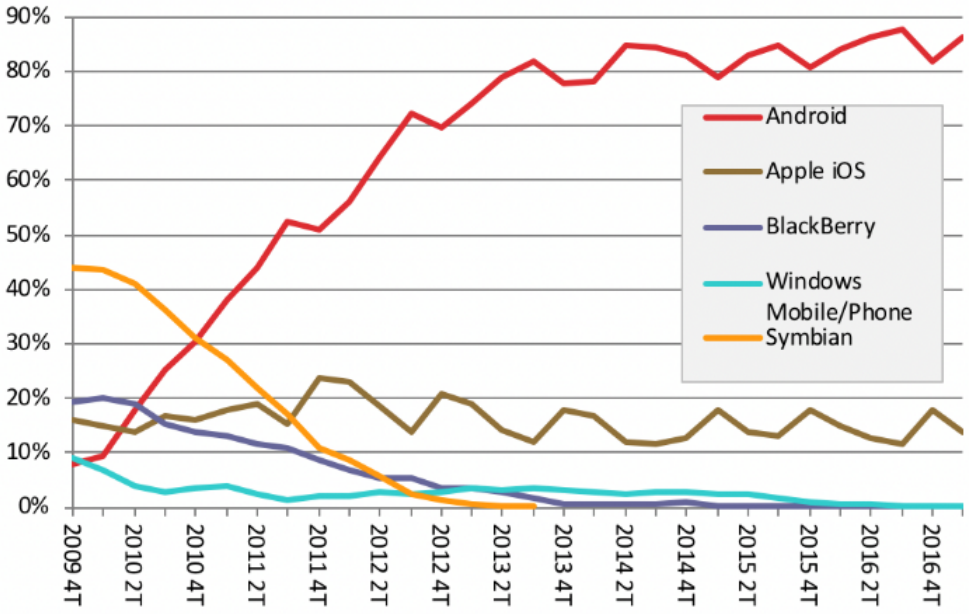

Figura 2.3: Porcentaje de teléfonos inteligentes vendidos en todo el mundo hasta el primer trimestre de 2017, según su sistema operativo [35]

.

2.7. El Sistema de Posicionamiento Global (GPS)

Es un sistema de localización, diseñado con fines militares para proporcionar estimaciones precisas de posición, velocidad y tiempo; que utiliza conjuntamente una red de ordenadores y satélites para determinar mediante triangulación, la altitud, longitud y latitud de cualquier objeto en la superficie terrestre [13] [14].

2.8. Propuesta

Con base en el análisis realizado se plantea una solución para el desarrollo de la plataforma de seguimiento de autobuses, en la que se toma en cuanta diferentes aspectos como: costo, escalabilidad y facilidad de uso. Por lo cual, para cumplir con las funcionalidades requeridas es necesario tener como principales características para el correcto funcionamiento: dos aplicaciones, del conductor y del cliente, además, de contar con una base de datos en tiempo real.
Con lo presentado se propone una plataforma de código abierto para desarrollar el sistema, que permita gestionar los de datos en tiempo real.

Capítulo 3

Diseño del sistema

En esta sección se presenta la caracterización del sistema, además, el diseño de las funciones con las que cuenta, por último, se describe cada de ellas para su correcta interpretación.

3.1. Descripción general

Este sistema se diseña con el propósito de obtener datos de geolocalización de conductores de autobuses con ayuda de un dispositivo móvil, mediante la utilización del GPS que tiene integrado, para posteriormente procesar y almacenar la información de ubicación en una base de datos alojada en un servidor web, ubicarlo en un mapa y realizar el seguimiento en tiempo real, de igual manera, se calcula y estima el tiempo de llegada a las diferentes estaciones establecidas.

En la Figura 3.1 se muestra la arquitectura que se usa para el desarrollo del Sistema.

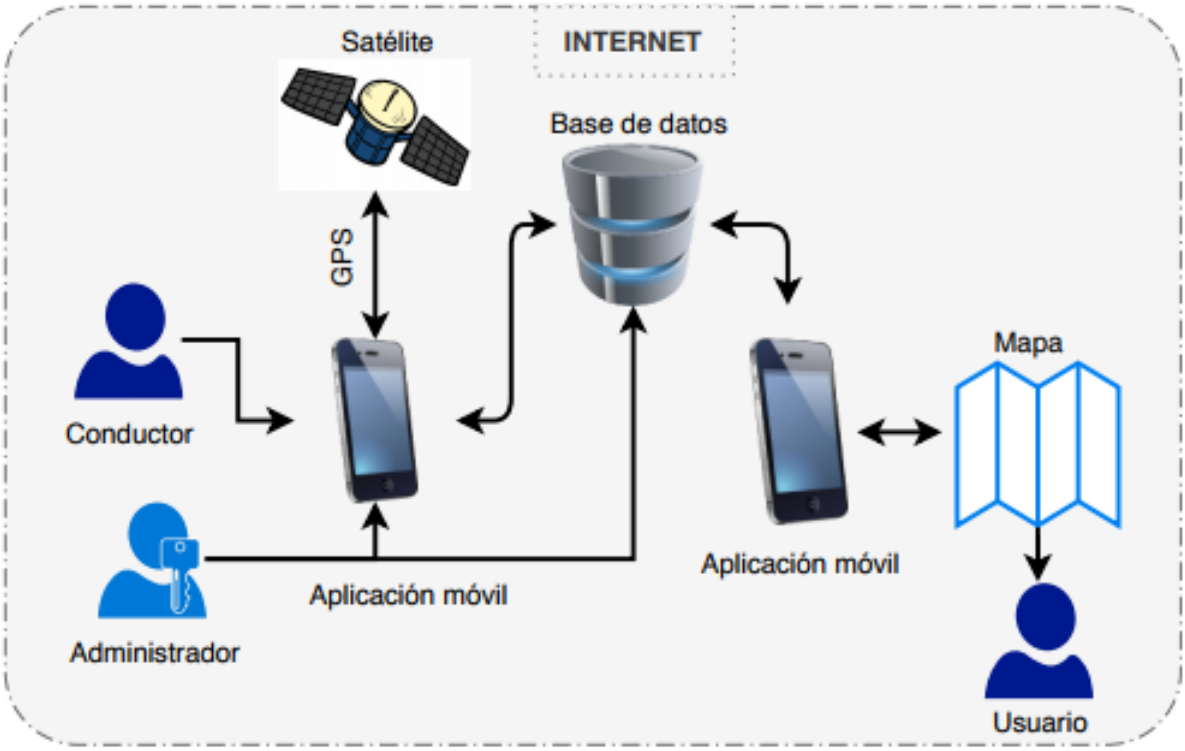

Figura 3.1: Arquitectura del Sistema

Los usuarios que tengan instalada la aplicación en su teléfono móvil tienen acceso a la información que está disponible en todo momento en un servidor web. Por otro lado, solo el administrador puede agregar, modificar y eliminar información de un conductor.

3.1.1. Requerimientos del sistema

En esta sección se enumera los diferentes requisitos necesarios tanto para la parte de servicio web como para la parte de aplicación móvil del proyecto, distinguiremos entre: 1) requisitos funcionales, que son aquellos que establecen cómo debe funcionar nuestra aplicación, 2) requisitos no funcionales describen aspectos del comportamiento de un sistema, capturando las propiedades y restricciones bajo las cuales un sistema debe operar y 3) requisitos técnicos, que enumeran de una manera más específica las tecnologías a usar [41].

3.1.1.1. Requisitos funcionales

Servidor web: El servidor web utilizado permite realizar funciones CRUD (crear, leer, actualizar, borrar) en la base de datos, por lo tanto, las modificaciones realizadas se deben visualizar en tiempo real. Además, es necesario tener un administrador que gestione toda esta información.

Aplicación móvil: La aplicación móvil creada es acceso gratuito, cuenta con servicios de Google de manera que el mapa funcione, de igual manera, el servicio de internet es indispensable para poder visualizar en tiempo real el movimiento del autobús de interés. Se muestra en la pantalla del usuario la posición actual del transporte y el tiempo de llegada a las diferentes paradas establecidas.

3.1.1.2. Requisitos no funcionales

De acuerdo con el esquema de Mamani [41] es recomendable separar los requisitos no funcionales en tres grupos: calidad, alcance y operación. Con base en esto se realiza la descripción de cada unos de los grupos tanto para el servicio web como para la aplicación móvil.

Servidor web

Calidad

- Precisión: :Entregar datos fiables y minimizar errores al momento de enviarlos desde el servidor web a la aplicación móvil.
- Seguridad: Desarrollar un sitema de validación de usuarios.
- Rendimiento: Gestionar los datos recibidos por los dispositivo rapidamente.
- Usabilidad: Esta definida por estándares internacionales y se mide en términos como la efectividad, eficiencia y satisfacción con los cuales se alcanzan los objetivos en el contexto especifico de uso.

Alcance

- Almacenamiento de datos: El servidor web utilizado es quien almacena los datos de los conductores de autobuses, además, guarda y procesa la información entregada por estos de forma temporal.
- Trasferencia de datos: Reducir los retrasos o latencia en el momento de la sincronización de dispositivos.

Operación: Almacena y sincroniza la información con nuestra base de datos alojada en la nube. Los datos se sincronizan con todos los clientes en tiempo real [40].

Aplicación móvil: Los requisitos no funcionales esenciales para las aplicaciones móviles son los siguientes:

Calidad

- Usabilidad: Fácil de usar para que el usuario pueda interactuar de mañera rápida y sin complicaciones sobre ella.
- Mantenimiento: Adaptación y evolución con ayuda de retroalomentación de los mismos usuario [42].
- Portabilidad: Es un requisito muy importante debido a que queremos que la aplicación se pueda ejecutar en la mayoría de los celulares para abarcar el mayor número de usuarios.

Alcance: Trasferencia de datos: Rápida respuesta en la recepción y envío de información.

3.1.1.3. Requisitos técnicos

Servidor web

- Base de datos de acceso gratuito

Aplicación móvil

- Dispositivo con sistema operativo (SO) Android con versiones superiores a la 5.0.
- Paquete configuración de los servicios de Google

3.2. Caracterización del sistema

Se determinan las tecnologías y componentes que se implementan en el sistema para asegurar un correcto desempeño.

3.2.1. Elección de plataformas a usar en el desarrollo de la aplicación

Con base al estudio realizado en el capítulo anterior se logró determinar que la mejor alternativa para desarrollar el sistema de seguimiento de autobuses debido a varios aspectos relevantes como ser de código abierto, evolución creciente, aplicabilidad en multiples dispositivos es Android. Además, la mayoría de modelos de teléfonos inteligentes tienen integrada la función GPS que puede ser aprobechada por el desarrollador, algo importante a tomar en cuenta es que Android posee multiples herramientas proporcionadas por Google, de esta manera haciéndolo más robusto. Por lo tanto, la plataforma Android es elegida para el desarrollo del sistema de seguimiento propuesto. En consecuencia el entorno de desarrollo que se usa es Android Studio, una de las bases de datos con la que se relaciona de forma sencilla es Firebase de Google, cabe decir que esta base de datos es gratuita, para un cierto número de peticiones. A continuación, se describen cada uno de los elementos elegidos.

3.2.1.1. Android Studio

Es el entorno de desarrollo integrado (IDE) oficial para la creación de aplicaciones en Android, basado en IntelliJ IDEA. Es un potente editor de códigos y posee algunas herramientas

para desarrolladores de IntelliJ [36]. Android Studio ofrece incluso más funciones que aumentan la productividad al momento del desarrollo de Apps para Android, en la Tabla 3.1 se presentan sus ventajas y desventajas:

Android tiene a su disposición multitud de herramientas para desarrollar software nuevo y, sobre todo, una gran comunidad para dar soporte a dichas herramientas [37][38].

Tabla 3.1: Ventajas y desventajas de Android Studio

Ventajas	**Desventajas**
Las aplicaciones desarrolladas son compatibles casi en su totalidad con los dispositivos móviles. Un sistema de compilación flexible basado en Gradle[1]. Para la compilación se puede usar un dispositivo virtual o físico. Permite la creación de nuevos módulos dentro de un mismo proyecto. Gran variedad de información para solucionar problemas en conjunto. Reutilizar código de funciones ya realizadas para ahorrar tiempo. Compatibilidad integrada para Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine. Se puede utilizar todas las funciones de un dispositivo móvil.	El permitir muchas aplicaciones abiertas hace que el consumo de la batería sea elevado y que dure mucho menos de lo previsto. El sistema de construcción de proyectos Gradle puede resultar complicado inicialmente. Para que el emulador trabaje correctamente, requiere de una elevada cantidad de recursos. Curva de aprendizaje más lenta para nuevos desarrolladores de Android.

3.2.1.2. Firebase

Uno de los pilares fundamentales en el desarrollo de esta aplicación es la comunicación, almacenamiento y disponibilidad de datos en tiempo real, por lo tanto, una de las herramientas usadas es Firebase, la cual proporciona varias utilidades para las aplicaciones móviles de una manera sencilla y rápida.

Firebase es una plataforma desarrollada por Google que facilita el desarrollo de aplicaciones

móviles, el cual suministra un servidor para el backend de las Apps. Además, el mismo puede ser re-utilizado de forma común por diversas plataformas: Android, IOS y web [39].

Firebase es una solución eficaz frente a problemas de desarrollo, como también de escalabilidad, debido a que utiliza la infraestructura de Google y se escala automáticamente a medida que los datos crecen. Proporciona diferentes funcionalidades como un servicio de autenticación, base de datos en tiempo real, almacenamiento de archivos, solución de errores, funciones backend, testeo y medida de estadísticas recogidas de los usuarios [39] [40].

Para el desarrollo de la aplicación es necesario usar algunas de las funciones que posee firebase. A continuación, se explica las funcionalidades a usar.

• Base de datos en tiempo real: Firebase proporciona una base de datos noSQL que almacena datos en formato JSON y los sincroniza en tiempo real facilitando la comunicación con la aplicación [40]. El sistema guarda los datos de posición mediante una aplicación creada para el conductor y los extrae mediante otra creada para el cliente.

• Autenticación: Es un servicio que simplifica el inicio de sesión y la gestión de usuarios, se puede iniciar mediante el correo y proveedores disponibles como Facebook, Twitter, GitHub y anónimo [40]. En el caso del sistema desarrollado se utiliza la autenticación mediante correo, de modo que es necesario una actividad de registro para poder realizar la validación del usuario.

3.3. Sistema de seguimiento de autobuses

Este sistema se divide en tres módulos; Módulo de transporte, Módulo de administración y Módulo de usuario, cada uno de estos está relacionado con la base de datos en tiempo real. En la Figura 3.7 se puede apreciar su interacción.

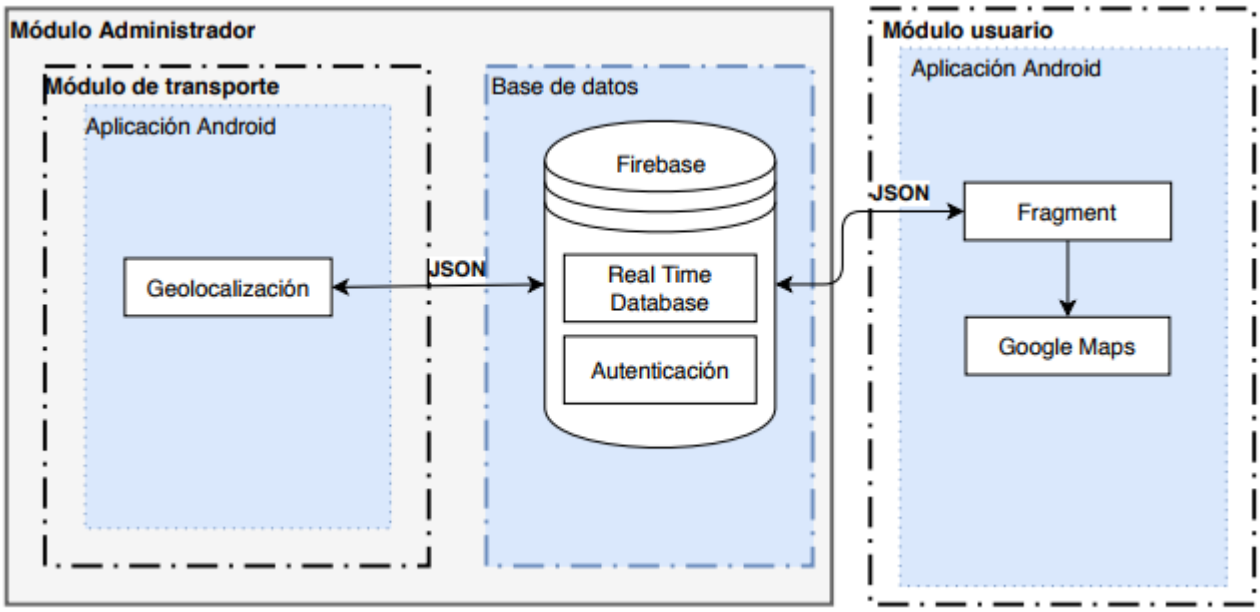

Figura 3.2: Diagrama general del sistema

3.3.1. Módulo del administrador

Este módulo es el responsable de administrar y actualizar toda la informacion estática sobre el autobús, como nuevos autobuses, rutas en la base de datos. Es decir, se encarga de realizar el registro de conductores y datos acerca de sus unidades, mediante una actividad en donde el administrador debe autentificarse para poder tener acceso al registro de datos, los datos del conductor registrado son enviados a la base de datos en Firebase. En donde el administrador podrá visualizar que usuarios estan activos. En la Figura 3.3 se muestra el diagrama de flujo del proceso.

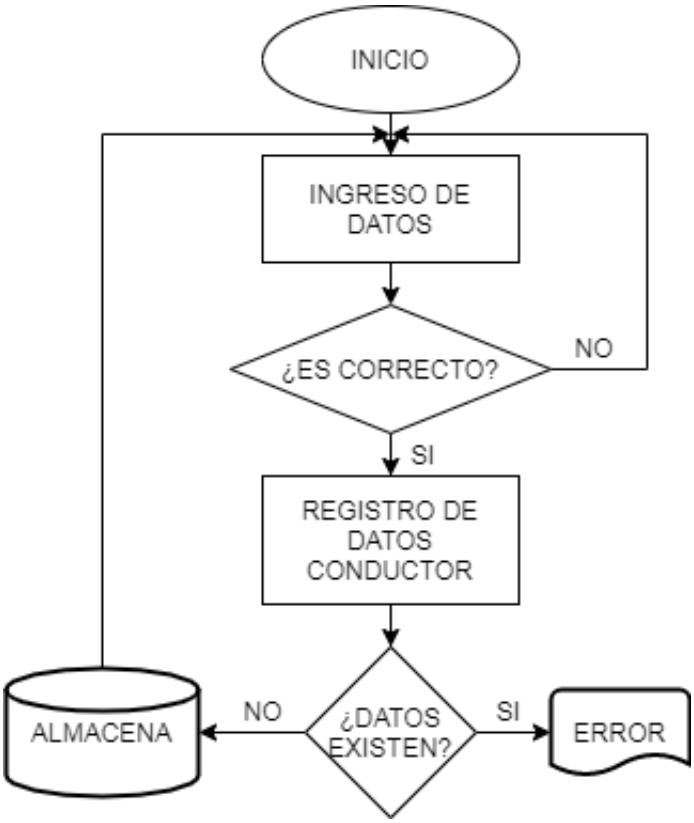

Figura 3.3: Diagrama de flujo del administrador

3.3.1.1. Boceto de la aplicación para el administrador

Figura 3.4: Boceto de actividades administrador

3.3.2. Módulo del conductor

El módulo del lado del conductor o de transporte describe como se rastrea el autobús, es decir, es responsable de entregar los datos y actualizar la ubicación actual del autobús. La Figura 3.5 muestra el proceso en un diagrama de flujo para actualizar la ubicación del bus.

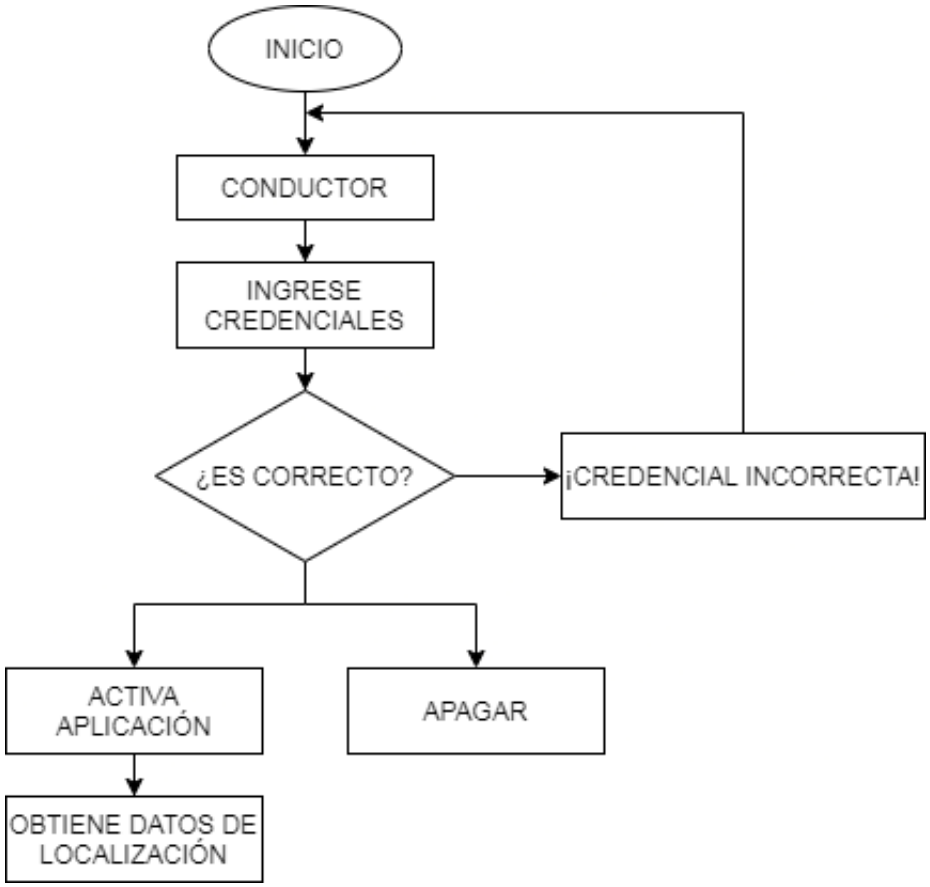

Figura 3.5: Diagrama de flujo del módulo de conductor

3.3.2.1. Geolocalización

Periódicamente, esta aplicación obtiene coordenadas geográficas del receptor GPS integrado en el smartphone. El sistema propuesto envia los datos de ubicación cada cierto tiempo. En la base de datos se crea variables de ubicación Latitud, longitud; dependiendo del usuario con el que se haya iniciado sesión, sin embargo, para el correcto funcionamiento en la captura de datos, es necesario dar permisos de ubicación de forma manual desde el dispositivo móvil.
Además de enviar información de ubicación actual, muestra la localización exacta del conductor, es decir, las calles por donde circula. También se obtiene información de velocidad mediante los datos entregados de posición. Para lo cual, se utiliza la fórmula de Haversine para el cálculo de distacia y de esa forma poder obtener la velocidad de acuerdo a un tiempo establecido.

Para encontrar la distancia es necesario conocer la distancia existente entre las dos coordenadas geográficas, de modo que es prudente tomar en cuenta que los puntos se encuentran localizados en la esfera terrestre, para considerar la curvatura de la tierra en el cálculo se emplea la fórmula de Haversine [43]. Es decir, para dos puntos en una esfera, el Haversine del ángulo central entre ellos está dado por.

$$haversine\left(\frac{d}{r}\right) = havsin(\Phi_1 - \Phi_2) + \cos(Phi_1) * \cos(\Phi_2) * havsin(\lambda_1 - \lambda_2) \tag{3.1}$$

donde *havsin* es la función haversine, dada por

$$havsin(\theta) = \sin^2\left(\frac{\theta}{2}\right) = \frac{1 - cos(\theta)}{2} \tag{3.2}$$

Donde d es la distancia entre los puntos (distancia esférica), r es el radio de la esfera terrestre, el radio en la zona ecuatorial es de 6378.14 km. Mientras que en la polar es de 636,78 *km*. En este caso por estar ubicado en la línea ecuatorial se utiliza el valor de 6378. Φ_1 y ϕ_2 corresponden a la latitud del punto1(bus) y del punto2(parada), y λ_1 y λ_2 representan la longitud del punto inicial y final respectivamente.

En la ecuación (3.1) el termino $\frac{d}{r}$ es el ángulo central, asumiendo que el mismo es medido en radianes, por lo tanto, los demás valores se deben trabajar de la misma manera para no tener errores en el resultado final. Para el cálculo de distancia despejamos la variable d de (3.1).

$$d = r * (havsin)^{-1}(h) = 2r * \arcsin(\sqrt{h}) \tag{3.3}$$

Donde $h = havsin\left(\frac{d}{r}\right)$ y debe ser un número entre 0 y 1 para que d sea real.

$$d = 2r * arcsin\left[\sin^2\left(\frac{\Phi_1 - \Phi_2}{2}\right) + \cos(\Phi_1)\cos(\Phi_2)\ \sin^2\left(\frac{\lambda_1 - \lambda_2}{2}\right)\right]^2 \qquad (3.4)$$

Para este cálculo se crea un algoritmo el cual realiza el procedimiento para cada uno de los puntos establecidos.

Antes de que los datos se almacenen, el subsistema consulta la información del bus a la base de datos. Si los datos existen, la aplicación actualiza la ubicación del bus, de lo contrario, crea una nueva información del bus.

3.3.2.2. Boceto de la aplicación del conductor

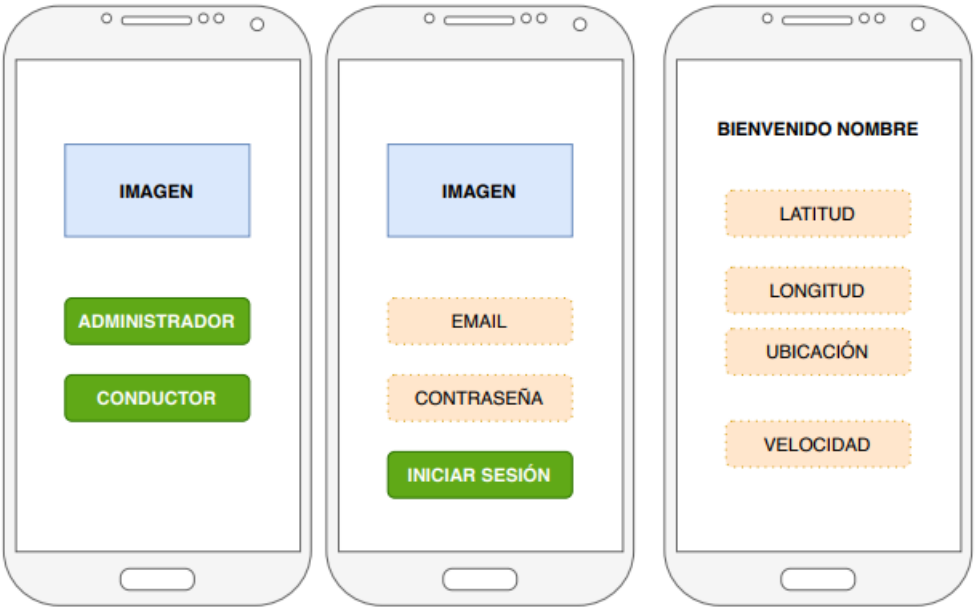

Figura 3.6: Boceto de actividades conductor

3.3.3. Módulo del usuario

El módulo del lado del usuario describe como interactúa el usuario con la funcionalidad, además de todas las interfaces proporcionadas para cada función. La función más relevante es la sincronización con la base de datos de Firebase, la cual proporciona la información de los autobuses almacenada en ella, por consiguiente, el resultado es la visualización de marcadores que representan a cada uno de los autobuses en funcionamiento en el Mapa, toda esta información se maneja en el formato JSON. Para que el mapa funcione correctamente se usa la API de Google Maps para Android Figura 3.7.

3.3.3.1. Fragment

Uno de los componentes escenciales en el desarrollo de la aplicación del usuario es el Fragment, el cual ayuda a ampliar parte de la lógica utilizada para la navegación entre pantallas o

Activities, pudiendo definir varios Fragments dentro de una misma Activity.

Se crea un fragment en donde se inserta el mapa que es usado por las diferentes actividades. Se muestra diferentes características de las actividades más relevantes:

1. En la actividad principal de la aplicación muestra mediante marcadores en el mapa[2] la posición actual de los autobúses.
2. Mediante una actividad de navegación (Navigation Drawer[3]) se desarrolla una opción llamada líneas en la cual se almacenan las rutas de cada uno de los autobuses y las paradas que hay en la ruta, además, se utiliza un algoritmo para el cálculo de tiempo de llegada a cada una de las paradas establecidas, ecuación (3.5).

$$t = d/v \tag{3.5}$$

Donde t es el tiempo calculado en minutos min, d es la distancia entre la ubicación actual del bus y las paradas establecidas calculada en metros m , v es la velocidad con la que el bus se desplaza calculada en m/min.

En el mapa se muestra el movimiento del autobús en tiempo real. La Figura 3.7 muestra el proceso del diagrama de flujo para mostrar todos los datos del autobús.

[2]Marcadores: Indican ubicaciones únicas en el mapa.
[3]Navigation Drawer: Es un panel que aparece desde el lado izquierdo de la pantalla y muestra las opciones principales de navegación de la app.

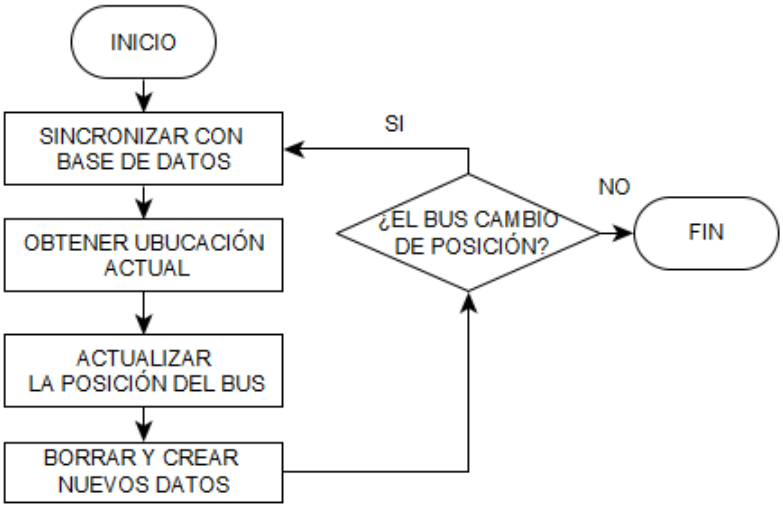

Figura 3.7: Diagrama de flujo cliente

3. Muestra la leyenda en cada uno de los marcadores creados en los mapas para ayudar al usuario a identificar la línea, el autobús y las paradas.

3.3.3.2. Boceto de las actividades del usuario

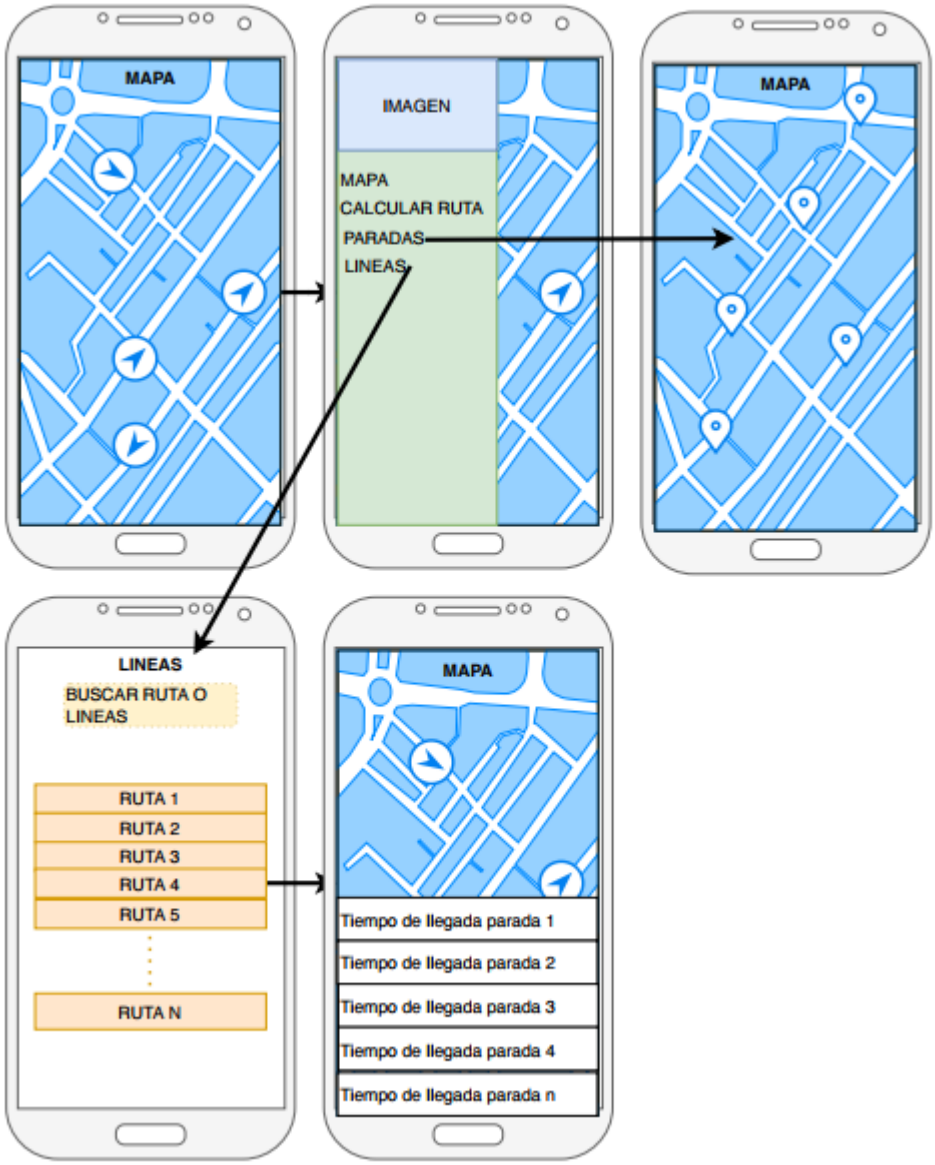

Figura 3.8: Boceto de actividades cliente

3.3.4. Base de datos

Firebase cuenta con dos bases de datos en tiempo real las cuales se prodrían usar para el desarrollo del proyecto, a continuación en la tabla 3.2 se muestra la comparativa.

Tabla 3.2: Comparativa de base de datos de firebase

	Realtime Database	**Cloud Firestore**
Actualizaciones en tiempo real	Si	Si
SDK centrados en el cliente, sin servidores	Si	Si
Modelo de datos	Almacena datos como un gran árbol JSON.	Almacena datos como colecciones de documentos.
Compatibilidad sin conexión y en tiempo real	Si	Si
Presencia	Si	Si
Realizar consultas	Si	Si
Confiabilidad y rendimiento	Disponibilidad zonal, latencia extemadamente baja.	Multiregional, latencia extemadamente baja
Escalabilidad	Escala hasta alrededor de 200,000 conexiones simultáneas y 1,000 escrituras por segundo en una misma base de datos. Para un escalamiento mayor que ese, se deben fragmentar los datos en distintas bases de datos. -Las tasas de escritura en piezas individuales de datos no tienen límites locales.	Escala de forma completamente automática. En este momento, los límites de escalamiento son de 1 millón de conexiones simultáneas y 10,000 operaciones de escritura por segundo. Las tasas de escritura en índices o documentos individuales tienen límites.
Seguridad	Alta	Alta
Precios	Se cobra solo por ancho de banda y almacenamiento.	Se cobra principalmente por operaciones ejecutadas en la base de datos(Lecturas, escrituras, eliminaciones), ancho de banda y almacenamiento.

Las bases de datos de Firebase cuentan con funciones similares, escalabilidad, baja latencia, lectura y escritura, actualizaciones en tiempo real, seguridad; las variables más importantes para el desarrollo del proyecto. En base a la comparativa realizada se elige como base de datos a Realtime Database por su nivel de coste.

La base de datos en tiempo real almacena los datos en formato JSON, siendo este sencillo de leer, almacenar y entender. Todo el sistema de rastreo interactúa con la base de datos enviando y recibiendo información inmediata dependiendo del módulo usado Figura 3.7.

3.3.4.1. Autenticación

Firebase cuenta con la función para autenticación de usuario, la cual es de mucha utilidad en la presente aplicación. El administrador es el encargado de registrar a los conductores mediante su correo electrónico y una contraseña.

3.3.5. Diagrama del funcionamiento del sistema

En la Figura 3.9 se puede observar la interacción de cada unos de los módulos descritos anteriormente,sus actividades desarrolladas, así como tambien se muestra como fluye la información.

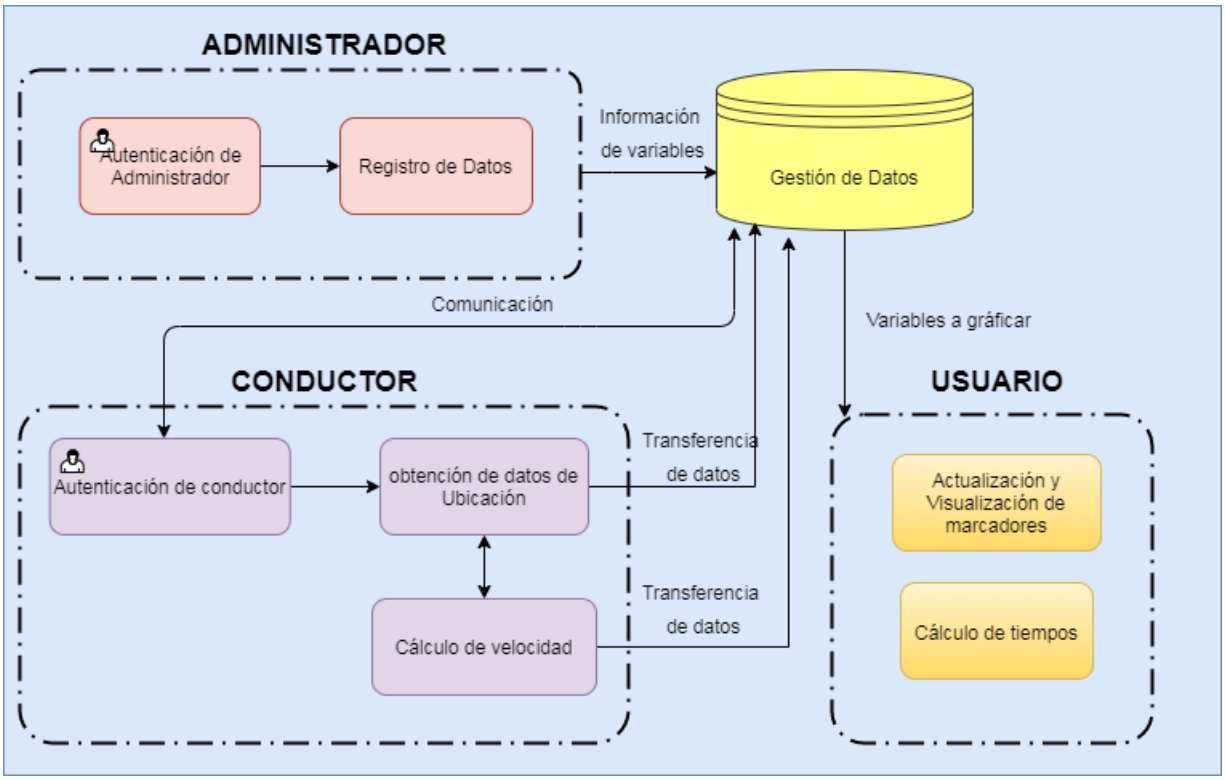

Figura 3.9: Diagrama del funcionamiento del sitema

Capítulo 4

Implementación y pruebas

En este apartado se detalla el proceso de implementación del proyecto, además, se describe los pasos para que el sistema cumpla con los requerimientos establecidos en los apartados anteriores.
El sistema desarrollado está basado en código abierto, lo que permite que sea extensible a cambios y mejoras en un futuro. El lenguaje de programación usado es Java junto con Android Studio como IDE para su desarrollo. Firebase Realtime Database se usa como base de datos en tiempo real para almacenar los datos del autobús y del conductor. El proyecto está disponible en el repositorio $https://github.com/DannyMard$.

4.1. Configuraciones de aplicaciones en Android Studio

En los diferentes módulos creados se utiliza una configuración similar, a continuación, se muestra de manera rápida la creación de un proyecto en Android Studio.

1. Tener la versión más reciente del programa.
2. Crear y nombrar el proyecto.
3. Seleccionar java como lenguaje de programación.

4. Seleccionar la versión más baja con la que Android ejecutará la aplicación.

5. Desarrollamos la aplicación.

4.1.1. Implementación de Mapa

En este proyecto se trabaja con la API de Google Maps, por lo que es necesario, descargar los plugins necesarios para que el proyecto funcione correctamente.

Para la implementación del mapa se puede encontrar información en Google Maps Platform[45]. A fin de utilizar Maps SDK para Android, se debe tener una clave de API. La clave de API es un identificador único que se utiliza para autenticar solicitudes asociadas con el proyecto con fines de uso y facturación[46]. Una vez obtenida la clave API, se habilita los servicios para obtener la ubicación del conductor. En la Figura 4.1 Se muestra las API activadas para este proyecto.

Figura 4.1: API Activadas

La aplicación de seguimiento solicita permisos de ubicación al usuario de forma manual a través del dispositivo móvil, estos son configurados desde el archivo Manisfest.xml. Los permisos usados son los siguietes:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

a) android.permission.ACCESS_FINE_LOCATION: Proporciona una ubicación mejor y más precisa.

b) android.permission.INTERNET: Proporciona a la aplicacion permiso para acceder a internet desde el dispositivo.

c) android:name=android.permission.ACCESS_NETWORK_STATE: Permite obtener el estado de la red, también puede verificar si la red está conectada o no.

4.1.2. Clase usada para servicio de localizacización.

4.1.2.1. LocationManager

Esta clase proporciona acceso a los servicios de ubicación del sistema. Estos servicios permiten que las aplicaciones obtengan actualizaciones periódicas de la ubicación geográfica del dispositivo o que se les notifique cuando el dispositivo entra en la proximidad de una ubicación geográfica determinada.

Todos los métodos de la API de ubicación requieren los permisos ACCESS_COARSE_LOCATION o ACCESS_FINE_LOCATION.

Esta clase tiene una variedad de métodos que pueden ser utilizados [45], no obstante en este proyecto se ha usado el método público Location.

Location: Obtiene la última ubicación conocida del proveedor especificado, o nula si no hay una última ubicación conocida. La ubicación devuelta puede ser bastante antigua en algunas circunstancias, por lo que siempre se debe verificar la antigüedad de la ubicación.

Conocida la ubicación se extrae la información de velocidad, latitud y longitud con un HashMap[1]. Los datos estarán en formato JSON. El cual permite construir una jerarquia de datos.

```
{
    "Velocidad": Dato Variable,
    "Latitud": Dato Variable,
    "Longitud": Dato Variable

  }
```

4.2. Configuración del Servidor web

Para la sincronización del proyecto en Android Studio y Firebase, es necesario:

1. Crear el proyecto en Firebase.
2. Elegir y habilitar la base de datos en tiempo real.
3. Habilitar las reglas de lectura y escritura.
4. Agregar el ID[2] de la aplicación de Android Studio en el proyecto en Firebase.
5. Agregar el certificado de firma SHA-1[3] del proyecto de Android Studio.
6. Registrar la aplicación.
7. Descargar el archivo de configuración .json y lo agregarlo al proyecto en Android Studio.

[1]HashMap: Designa claves únicas para los valores correspondientes que se pueden recuperar en cualquier punto dado.
[2]ID: Permite identificar de manera exclusiva tu app en el dispositivo y en Google Play Store.
[3]SHA-1: Huella digital de depuración.

8. Agregar en el gradle de Android Studio las dependencias necesarias de los servicios de Firebase a usar.

9. Sincronizar el proyecto.

Para habilitar el servicio de autenticación en necesario elegir y habilitar el proveedor, en este caso la autenticación será por email. Además, se debe agregar en el gradle la dependencia necesaria en el proyecto de Android Studio.

Las depencias necesarias usadas en este proyecto son:

```
//Firebase
implementation 'com.google.firebase:firebase-analytics:17.0.0'
implementation 'com.google.firebase:firebase-auth:19.3.1'
implementation 'com.google.firebase:firebase-database:18.0.0'
implementation 'com.google.firebase:firebase-core:17.0.0'
implementation 'com.google.firebase:firebase-storage:18.0.0'

//Location Service
implementation 'com.google.android.gms:play-services-location:17.0.0'
```

Finalmente se debe verificar si los servicios a usar están conectados con Firebase, para esto necesario ir al asistente en que tiene Android Studio para Firebase. El Figura 4.2 se muestra el resultado.

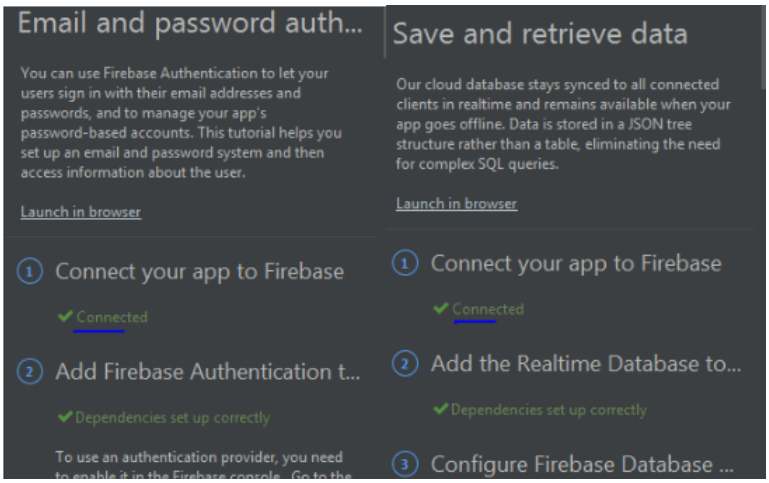

Figura 4.2: Verificación de estado de conexión

4.3. Indicaciones de funcionamiento

4.3.1. Aplicación del conductor

En el aplicativo del conductor se crea diferentes actividades para una mejor comprensión del sistema, con objetivo de facilitar el uso de la misma. Para ejecutar esta aplicación es necesario tener el archivo APK o ejecutable de la aplicación para poder utilizarlo. Debe ser instalado en nuestro dispositivo Android. Una vez instalada la aplicación podremos observar que en la pantalla principal tenemos dos opciones: a) administrador y b) Conductor, tal como se muestra en la Figura 4.3.

Figura 4.3: Actividad inicio de sesión de usuario

Administrador: Esta opción cuenta con dos actividades como se muestra en la Figura 4.4. La primera es de autenticación de administrador, en la que por conveniencia se ha fijado los datos de ingreso como: usuario admin y de contraseña admin. En la siguiente actividad se debe registrar los datos del conductor y del autobús, la información de interés será enviada a la base de datos en Firebase.

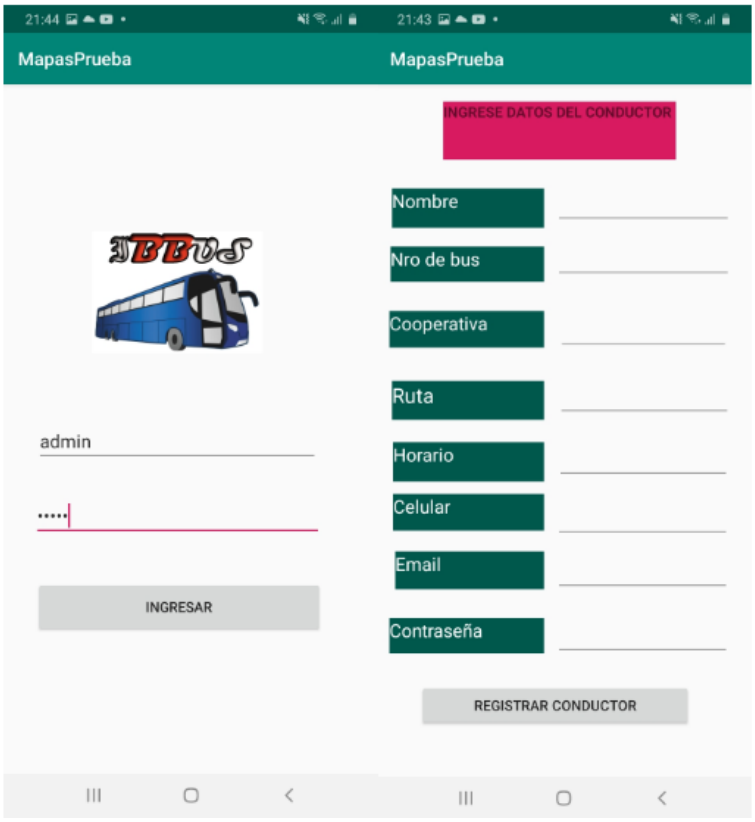

Figura 4.4: Actividad administrador y registro de conductores

Es necesario tomar en cuenta que para la creación del conductor en el sistema se debe realizarlo mediante el email y una contraseña personal, la cual es utilizada para poder iniciar la actividad de rastreo mediante el GPS incorporado del teléfono móvil.

Una vez realizado el registro se verifica si los usuarios están creados. Verificamos el servicio de autenticación en Firebase Figura 4.5, además, es necesario corroborar que la información del conductor también este en la base de datos en tiempo real(RealTime DataBase), tal como se muestra en la Figura 4.6. Todo lo expuesto anteriormente acerca de la plataforma esta controlada por el administrador del sistema.

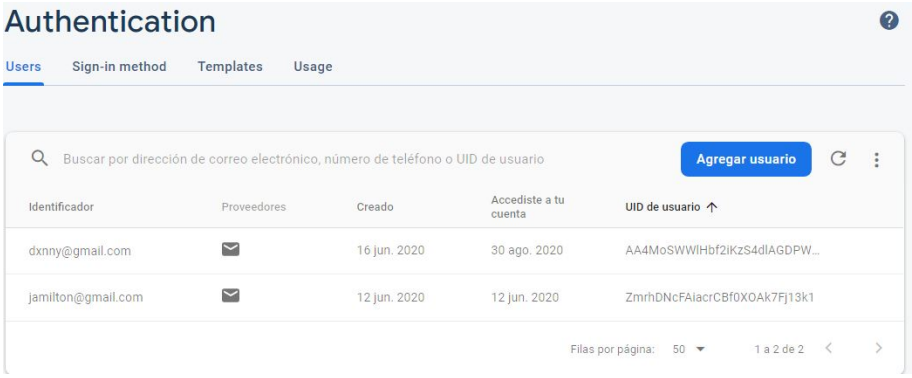

Figura 4.5: Registro de conductores en Firebase

Figura 4.6: Visualización de datos en RealTime DataBase

Conductor: Los conductores ingresan con el correo y contraseña creada por el administrador, seguidamente se inicia la siguiente actividad, en la cual, es necesario dar permisos de ubicación para que obtenga información mediante el GPS que tiene integrado el teléfono inteligente, es decir, hay que activarlo. El proceso se muestra en la Figura 4.7.

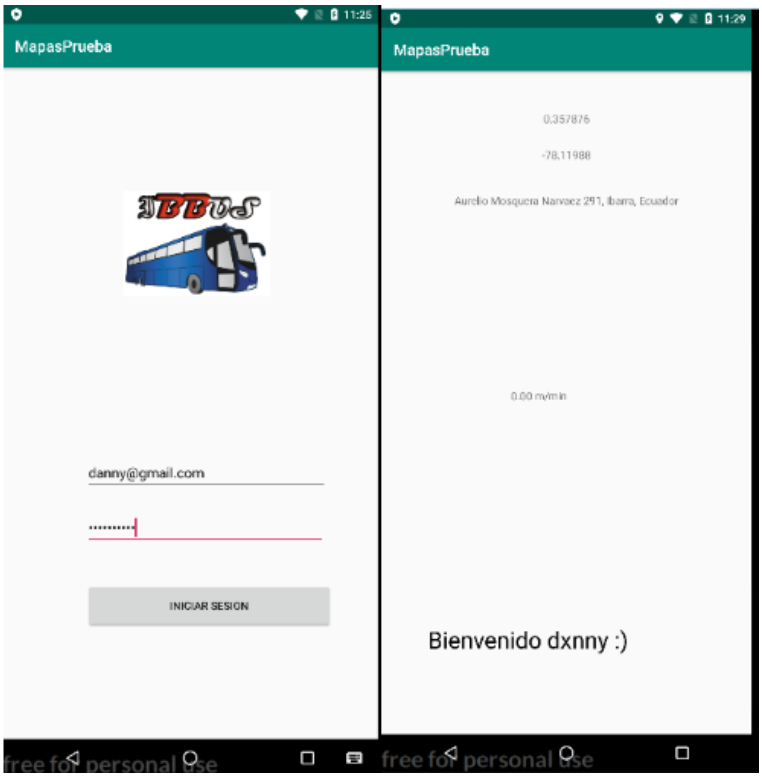

Figura 4.7: Autenticación y obtención de datos GPS

Una vez que la actividad de rastreo comienza a funcionar, en la base de datos de Firebase habrá un incremento de información en los datos del conductor, es decir, los datos de latitud, longitud y velocidad se añadirán al árbol ruta del usuario activo, como se muestra en la Figura 4.8.

Figura 4.8: Actulizacíon de datos de ubicación y de velocidad

Cabe decir que los datos de ubicación y velocidad estan sujetos a cambios periódicos debido a que la aplicación trabaja en tiempo real, por tal motivo la información de cada una de las posiciones entregadas son diferentes cada cierto tiempo.

Para que la aplicación del lado del conductor funcione correctamente es necesario:

- Conectividad a internet
- Permitir el acceso a ubicación, es decir encender el GPS.
- El conductor debe contar con un correo y una contraseña.
- El SO debe ser Android y su versión superior a 6.

4.3.2. Aplicación del cliente

Para ejecutar la aplicación del lado del usuario es necesario contar con el APK o el ejecutable para poderlo instalarlo en el teléfono inteligente, esta aplicación funciona desde la versión de Android 6 en adelante.

Una vez instalada la aplicación se puede observar en la pantalla principal el mapa con las ubicaciones en tiempo real de los buses. Como se muestra en la Figura 4.9.

Figura 4.9: Pantalla principal del subsistema del usuario

Para averiguar la ubicación actual del bus, la aplicación del lado del usuario solicita la información a la base de datos en Firebase, para esto no es necesario que la aplicación del lado del usuario solicite datos de actualización periódicamente porque se sincronizarán automáticamente, posteriormente, los datos son almacenados en las variables creadas en Android Studio para mostrarlas en el IDE. Por lo que, si no hay un conductor en línea, no se muestra ningún bus.

En la parte superior izquierda de la aplicación tenemos un menú Figura 4.10 , en donde se muestra las diferentes opciones, sin embargo, este proyecto se enfoca en creación de rutas, seguimiento de trasporte y tiempos de llegada, por lo que se explica el funcionamiento de la

opción lineas.

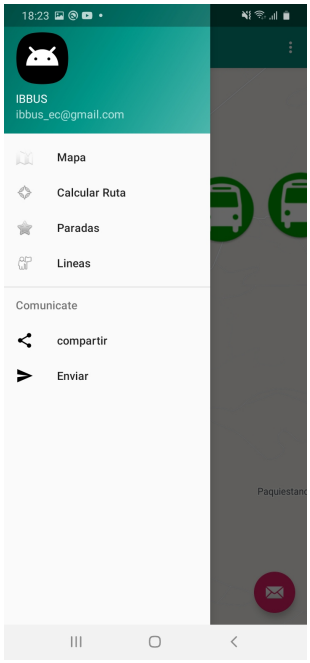

Figura 4.10: Menú de opciones

Lineas: En la opción lineas existe un filtro en donde el usuario puede buscar la ruta por cual viaja Figura 4.11, es necesario mencionar que se puede agregar, modificar y eliminar rutas. El prototipo ha sido desarrollado para dos rutas especificas. Por lo que, al ingresar a una de estas opciones podremos visualizar un mapa con la ruta cargada por donde el bus viaja, las paradas estalecidas y el tiempo de llegada a cada una de ellas. Figura 4.12.

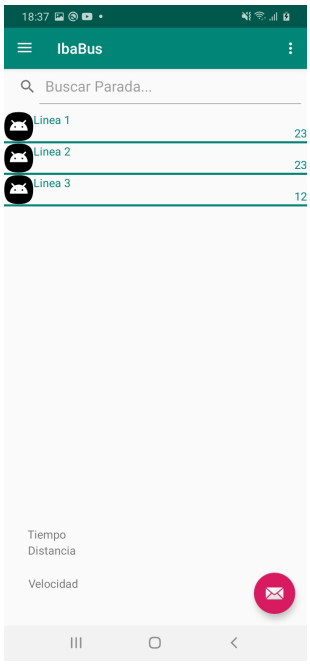

Figura 4.11: Lista de Lineas

Figura 4.12: Seguimiento de autobus, paradas y tiempos de llegada

4.4. Pruebas de funcionamiento

Una vez que se realizada la implementación del sistema, se procede a hacer las respectivas pruebas del funcionamiento para evaluar su rendimiento y validación. La prueba del sistema es un proceso importante y útil en el proyecto porque se puede detectar errores dentro del software. Normalmente, los criterios de prueba se basan en los requisitos del usuario y del sistema, para verificar si el sistema cumple con los requisitos o no [44]. La confiabilidad del sistema es muy importante para los usuarios finales.

Para las pruebas del sistema se divide en dos partes que son: pruebas unitarias y pruebas de integración de módulos. Las pruebas unitarias son un tipo de prueba que se realiza a cada uno de los componentes individuales en un sistema y las pruebas de integración son el testeo del proceso en combinación de todos los subsistemas o módulos [44].
Por esta razón antes de la integración , se realizaron pruebas unitarias en cada uno de los subsistemas para garantizar que funcionen perfectamente. Una vez que todos los módulos pueden comunicarse entre sí, el sistema final está listo y las pruebas de integración probarán el sistema completo.

4.4.1. Pruebas unitarias del sistema

Prueba 1: Registro y Autenticación del conductor (Aplicación del conductor). - Prueba unitaria

Tabla 4.1: Registro y Autenticación del conductor

N°	Objetivo de la prueba	Pasos de prueba	Resultado esperado	Resultado
1	Verificación de cuenta de administrador	1) Ingrese usuario administrador 2) Ingreso de contraseña	Debería dar paso a la siguiente actividad de registro. En caso de no ingresar correctamente los datos Debería dar un aviso.	Aprobado
2	Para garantizar el registro de datos del conductor y almacenamiento en la base de datos Firebase	1) Llenar los campos de los datos del conductor y regístralos. 2) Observar los datos en la base de datos Firebase.	En la base de datos Firebase debería mostrase los datos registrados del conductor. Si los datos no son ingresados correctamente o el usuario ya existe debería dar un mensaje de error.	Aprobado
3	Para garantizar la autenticación correcta del conductor.	1) El conductor debe ingresar el correo registrado por el administrador. 2) Ingrese la contraseña registrado por el administrador.	Debería dar paso a la siguiente actividad de localización de conductor. En caso de ingresar los datos de manera incorrecta debería mostrar el respectivo mensaje de error.	Aprobado
4	Para garantizar que los datos de ubicación y velocidad del conductor sean transferidos correctamente.	1) Permitir el acceso de ubicación o activar el GPS. 2) Observe los datos GPS recibidos del satélite. 3) Observe que los datos de velocidad y de ubicación en la base de datos 4) Mover el teléfono inteligente a otro lugar.	La base de datos debería almacenar los datos de ubicación y velocidad en su respectivo usuario registrado. Los datos de ubicación y velocidad deben estar en constante cambio en la base de datos Firebase. Debería mostrarse el nombre de las calles en el que se encuentra el conductor.	Aprobado
5	Para garantizar el funcionamiento de la aplicación en segundo plano.	1) Minimizar la aplicación del conductor. 2) Observar la base de datos	La aplicación debería seguir entregando los datos de ubicación a la base de datos en Firebase.	Aprobado

Prueba 2: Creación de marcadores de posición de bus en tiempo real – Prueba unitaria (Aplicación del usuario).

Tabla 4.2: Creación de marcadores de posición de bus en tiempo real

No	Objetivo de la prueba	Pasos de prueba	Resultado esperado	Resultado
1	Para garantizar que los marcadores estén creados correctamente en el mapa de la aplicación del usuario.	1) Registrar varios usuarios en la aplicación del conductor. 2) Iniciar la aplicación del conductor para que envié datos de ubicación. 3) Abrir en un navegador la base de datos. 4) Observar los marcadores creados en la aplicación del usuario.	Debería mostrarse los marcadores en el mapa de la aplicación del usuario dependiendo del número de usuarios activos con la aplicación de conductor.	Aprobado
2	Para garantizar que el marcador del bus se actualizará automáticamente con intervalo de 2 segundos.	1) Realizar un viaje por una ruta con la aplicación del teléfono instalada y funcionando. 2) Ejecutar la aplicación del usuario cuando el bus comience a moverse. 3) Abrir la base de datos en el navegador. 4) Observar la actualización del marcador en la aplicación del usuario.	Actualización de datos de ubicación y velocidad del bus en la base de datos. La posición del marcador debería actualizarse en el mapa de la aplicación del usuario.	Aprobado

Prueba 3: Visualización del tiempo de llegada del autobús en tiempo real- Prueba unitaria. (Aplicación de usuario)

Tabla 4.3: Visualización de tiempos de llegada

No	Objetivo de la prueba	Pasos de prueba	Resultado esperado	Resultado
1	Debe mostrar las paradas de ruta escogida.	1) Abrir la base de datos en el navegador. 2) En el menú desplegable seleccionar la opción línea. 3) En la actividad buscar la ruta del bus deseado.	En el mapa de la actividad de la ruta escogida debería mostrarse las paradas establecidas a esa ruta.	Aprobado
2	Para garantizar que el tiempo de llegada del bus se actualizará mientras el bus este en movimiento.	1) Hacer el viaje en bus mediante en la ruta establecida. 2) Ejecute la aplicación del conductor. 3) Ejecutar la aplicación del usuario en ingresar a la línea seleccionada. 4) Observar el tiempo cuando el bus este en movimiento.	Los nombres de las paradas deberían ser correctas. Los tiempos de llegada a las paradas deben variar de acuerdo al movimiento del autobús.	Aprobado

4.4.2. Prueba de integración de módulos

Prueba 5: Prueba de integración de módulos.

Tabla 4.4: Integración de módulos

No	Objetivo de la prueba	Pasos para prueba	Resultados esperados	Resultado
1	Para garantizar que todos los módulos puedan comunicarse entre sí.	1) Correr todos los subsistemas. 2) Observar el marcador en el mapa. 3) Observar los cambios de tiempo	Todos los módulos deberían estar comunicándose correctamente y enviando la información a la aplicación del usuario.	Aprobado

Realizada cada una de las pruebas a los módulos, se logra determinar que todas las carac-

terísticas del sistema funcionan correctamente.

4.4.3. Evaluación de rendimieto de las aplicaciones

El rendimiento es uno de los aspectos más relevantes al momento de evaluar las aplicaciones. DoubleClick de Google determinó que un 53 % de las visitas a sitios móviles se abandonaban si una página tardaba más de 3 segundos en cargarse[47]. Además, en cuestión de costos mientras más latencia tenga una aplicación o sitio web más perdida tendrá. En el sistema desarrollado se realizan las pruebas de rendimiento con las siguientes condiciones de software.

Tabla 4.5: Características de sotfware

Memoria RAM	3 GB
Procesador	Qualcomm Snapdragon 450
SO	Android
Versión de Android	10
Velocidad de subida	4.78 Mbps
Velocidad de bajada	6.63 Mbps
Ping	12 ms

Como este sistema cuenta con dos aplicaciones móviles, se evalua cada una independientemente.

4.4.3.1. Rendimiento de la aplicación del conductor

En la Tabla 4.6 se puede visualizar los tiempos de carga de las actividades o tareas realizadas por el administrador y conductor que oscilan entre 1 segundo, la sincronización de datos de ubicación posee una rápida respuesta.

Tabla 4.6: Tiempo de carga de la aplicación del conductor y administrador

Actividades	Tiempo de carga (s)
Pantalla principal	1.25
Registro de datos	1.62
Validación de usuario	1.52
Sincronización con Firebase	0.59

4.4.3.2. Rendimiento de la aplicación del conductor

En la Tabla 4.7 se puede visualizar los tiempos de carga de las actividades o tareas realizadas por usuario, estas no sobrepasan los tres segundos. En la sincronización de la aplicación con Firebase existe un retardo en entrega de información de 1 segundo, añadiéndole el tiempo de demora de la aplicación del conductor; podemos determinar que los datos se actualizan en menos de tres segundos, lo cual está dentro de los estándares mencionados anteriormente.

Tabla 4.7: Tiempos de carga de actividades del usuario

Actividades	Tiempo de carga (s)
Pantalla principal	2.05
Lineas	0.5
Cargar ruta	1.2
Sincronización con Firebase	1

Además, mediante la herramienta performance de Firebase se puede visualizar si la aplicación tuvo algún inconveniente al momento de su ejecución, en este caso los errores son nulos; incluso se puede observar la cantidad memoria que utiliza al momento de la ejecución de la aplicación es baja. Estos datos serán manejados por el administrador.
La memoria usada momento de la ejecución de la aplicación es de 6.35MB. En la Figura4.13 se puede visualizar que el tiempo de carga de la aplicación oscila por los 2 segundos.

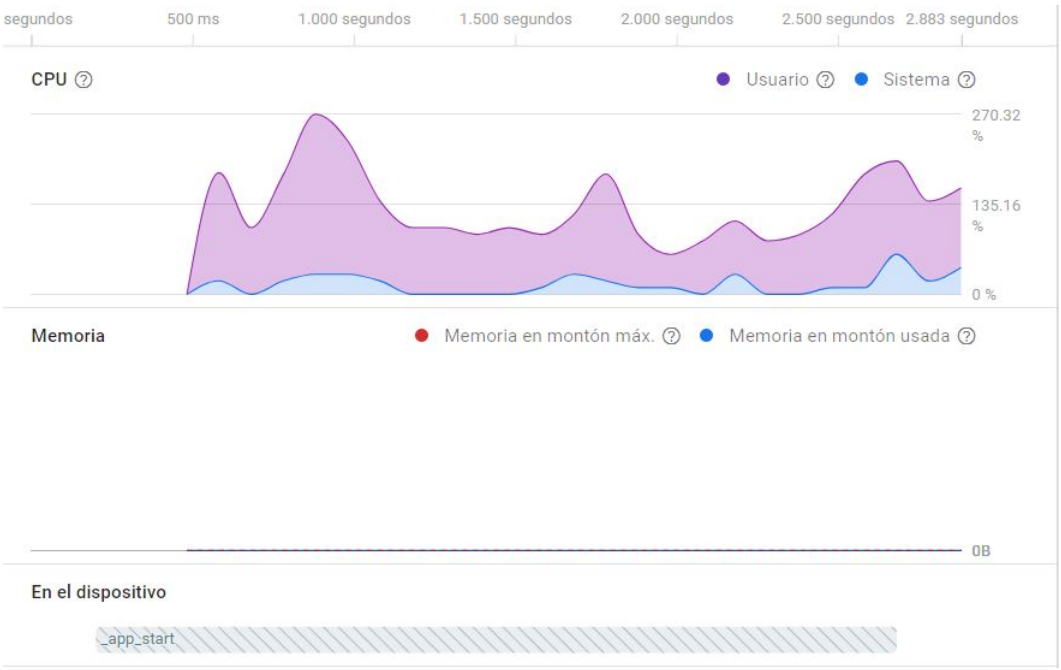

Figura 4.13: Rendimiento de la aplicación del usuario usando Firebase

4.4.3.3. Pruebas de tiempos de llegada

En las pruebas de funcionamiento para validar los tiempos de llegada a las paradas establecidas, se realiza en cuatro días diferentes en un mismo horario, se marca con SI/No, si el autobús llegó en el tiempo mostrado en la aplicación. En la Tabla 4.8 se muestra los resultados.

Tabla 4.8: Evaluación de tiempo de llegada

PARADAS	HORA	DIA 1	DIA 2	DIA 3	DIA 4
Parada 1	6:30	SI	SI	SI	SI
Parada 2	6:35	SI	SI	SI	SI
Parada 3	6:38	SI	SI	SI	NO
Parada 4	6:42	SI	NO	NO	NO
Parada 5	6:45	NO	SI	SI	NO
Parada 6	6:48	SI	SI	SI	NO
Parada 7	6:50	NO	NO	SI	NO
Parada 8	6:53	SI	SI	SI	NO
Parada 9	6:54	SI	SI	NO	NO
Parada 10	7:00	SI	NO	SI	NO

4.4.4. Evaluación de usabilidad

Como se mencionó anteriormente la usabilidad está definida por términos como efectividad, eficiencia y satisfacción. Para medir la usabilidad del sistema respecto al usuario se utiliza SUS(System Usability Scale), es una herramienta metodológica muy similar a la escala de Likert. Aunque esta escala es extraordinariamente simple de usar, diferentes pruebas y tests han demostrado que los resultados obtenidos a partir de la misma suelen ser confiables y acertados, razón por la cual es uno de los métodos de medición de usabilidad más utilizados en experiencia de usuario.En la Figura 4.14 se muestra la escala.

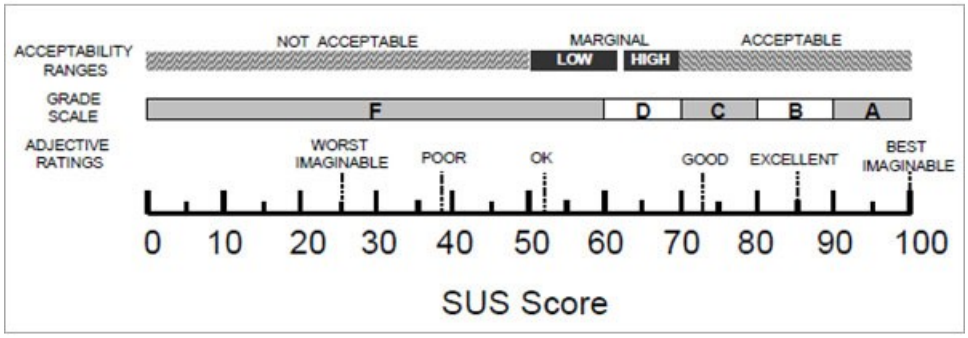

Figura 4.14: Puntaje SUS

La escala en sí consiste en 10 preguntas, cada una de las cuales puede ser puntuada de 1 a 5, donde 1 significa total desacuerdo y 5 significa total acuerdo.

Las preguntas recomendadas y utilizadas son las siguientes:

- Creo que usaría esta aplicación frecuentemente.
- Encuentro esta aplicación innecesariamente compleja.
- Creo que la aplicación fue fácil de usar.
- Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar esta aplicación.

- Las funciones de esta aplicación están bien integradas.
- Creo que esta aplicación es muy inconsistente.
- Imagino que la mayoría de la gente aprendería a usar esta aplicación en forma muy rápida.
- Encuentro que esta aplicación es muy difícil de usar.
- Me siento confiado al usar esta aplicación.
- Necesité aprender muchas cosas antes de ser capaz de usar esta aplicación.

La prueba se realiza con 10 participantes,que tienen la aplicación instalada en su teléfono móvil para poderla utilizar. Se dio una explicación rápida del funcionamiento de la misma.
Con la ayuda de usabilityTest[48] se crea el test de usabilidad de la aplicación y se envía a los participantes, para que puedan calificar su uso. Una vez terminada la evaluación los usuarios enviaran sus resultados al evaluador. Este sitio web nos permite analizar de manera rápida y sencilla los datos que envían los participantes. En la Tabla 4.9 se muestran los resultados obtenidos.

Tabla 4.9: Resultados del test de usabilidad

Personas	**Preguntas**										**SUS**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	4	2	5	1	4	2	4	2	4	2	80
2	5	1	4	2	4	2	4	2	4	2	80
3	4	2	4	2	5	1	3	4	3	3	68
4	4	2	3	2	4	1	4	1	5	2	80
5	4	2	4	2	4	2	4	2	4	2	75
6	5	4	5	2	5	2	5	1	5	2	85
7	4	2	2	2	4	4	4	2	4	2	65
8	4	2	4	2	4	2	4	2	4	3	73
9	5	2	4	2	4	1	4	2	3	3	75
10	4	2	2	2	4	4	4	2	4	2	65
							SUS Promedio:				**75**

4.4.5. Discusión

El sistema desarrollado es de código abierto, esto permite que cualquier persona con el conocimiento necesario pueda modificarlo y mejorarlo. Además, para el desarrollo de este proyecto se utiliza herramientas gratuitas de Firebase para el seguimiento en tiempo real y autenticación de usuario. Mediante la utilización de la herramienta autenticación se da seguridad a los datos, es decir, solo pueden ingresar al sistema personas que sean registradas por el administrador, de esta forma evitando confusión y mal interpretación de información.

Realtime DataBase de Firebase permite realizar funciones CRUD, lo cual lo convierte en una herramienta muy potente para el seguimiento en tiempo real de las unidades de transporte. Mediante diferentes herramientas que posee Android Studio para localización, se puede obtener datos fiables de ubicación. En esta aplicación se hizo uso de Location, la cual permite capturar la ubicación de un dispositivo.

Las pruebas de funcionamiento se realizan para garantizar que los módulos trabajen deacuerdo a lo planeado, cada módulo tiene funciones diferentes. En los resultados presentados se puede apreciar que todas las funciones trabajan correctamente. Seguidamente se analiza el tiempo de respuesta de las aplicaciones en condiciones reales de uso, se logra apreciar que el tiempo de respuesta oscila entre los tres segundos, estando dentro de los estándares esperados. Además, no existen retrasos considerables en el cálculo de estimación de tiempos de llega a las paradas establecidas. Por otro lado, se logra apreciar que la aplicación del lado del usuario no consume mucha memoria al momento de su ejecución.

Con respecto a la estimación de tiempos de llegada del autobús a las paradas establecidas, se puede observar en la Tabla 4.8 que las unidades no llegan a tiempo a diferentes paradas, debido a que existen una variedad de factores como: tráfico, equipaje de algunos pasajeros, espera de autobús, daño de autobús. En el día 4 se puede observar que el autobús no llega a tiempo a la mayoría de paradas esto debido a que sufre daño mecánico, evento fortuito que puede suceder en el transcurso del viaje, por lo que, es necesario que el conductor en su aplicación cuente con una función de interacción con el usuario, para comunicar cualquier novedad que pueda surgir en el transcurso del viaje y de esa forma puedan tomar una decisión para no llegar retrasados al

lugar de destino.

Finalmente se realiza la prueba de usabilidad de la aplicación, utilizando SUS como herramienta de análisis de la información obtenida por parte de los usuarios. En donde tenemos como resultado que la aplicación tiene un puntaje de 75. Según la Figura 4.14 le asigna una ponderación de C y lo califica como un sistema bueno.

Capítulo 5

Conclusiones y trabajo futuro

En este capítulo muestra las conclusiones del presente proyecto, además se propone posibles líneas para trabajo futuro.

5.1. Conclusiones

- En base a análisis realizado, uno de los pilares fundamentales de esta investigación fue definir la funcionalidad del sistema para determinar su funcionamiento, de esta forma logrando precisar las características más relevantes de cada una de ellas, es decir, para el desarrollo del sistema fue conveniente establecer los requerimientos funcionales y no funcionales, a partir de las necesidades que se tenía para el desarrollo.

- El sistema se desarrolló para la plataforma Android, debido al auge que está viviendo, además, por el gran número de herramientas que posee y la compatibilidad con diferentes tecnologías de código abierto ayudando esto a que el sistema de seguimiento sea modificable y extensible.

- La aplicación tanto del conductor como del usuario se deben sincronizarse automáticamente, para lo cual, se utiliza la plataforma Firebase que posee varias herramientas, que

permiten gestionar los datos de manera ordenada. Se debe realizar la configuración necesaria para la correcta utilización de las dos plataformas Android Studio y Firebase.

- Para someter el sistema a condiciones reales de funcionamiento se realizó pruebas en recorridos cortos y largos, mediante observación se logró determinar que los datos de ubicación y velocidad eran variables, es decir, en un intervalo de tiempo determinado la información se obtiene, se actualiza, se almacena y se muestra de manera correcta.

- Finalmente, mediante pruebas realizadas al sistema, se logra garantizar que todas las tareas funcionen correctamente, con una latencia máxima de 3 segundos, de esta forma se da validez al sistema.

5.2. Trabajo futuro

El sistema de seguimiento tiene gran escalabilidad debido a que sido desarrollado en plataformas de código abierto, es decir, que cualquier persona puede modificar el código y mejorarlo. La mayoría de las aplicaciones móviles inteligentes existentes no se han evaluado por completo para determinar su precisión en la recopilación de datos de viaje. Por lo que mediante técnicas de estimación de posición se puede ser más preciso en la entrega de información.

Las aplicaciones desarrolladas pueden ser mejoradas abarcando a una ciudad incluso provincias. El número de usuarios que podrán acceder a la aplicación estará restringido hasta un cierto valor, pero para extender el servicio se debe pagar un costo adicional a Google, lo cual a grande escala es recomendable.

Una de las mejoras importantes sería crear un programa con inteligencia artificial para estudiar y analizar automáticamente los datos de la ruta del autobús para de esa forma proporcionar datos más confiables. Al aplicar el programa de inteligencia artificial, el sistema será más valioso debido a la precisión de la estimación del tiempo de llegada.

Se puede utilizar Flutter un framework realizado por Google que posee diferentes herramientas que facilitaran y agilizaran el desarrollo de las aplicaciones creadas.

[english]babel [utf8]inputenc

Bibliografía

[1] V. R. S. Sriddharan S, «Designingna smart trasport system appliction for south Indian traffic scenarios,» de Conference on Communication and Electronics Systems, Coimbatore India, 2017.

[2] H. P. Sankarananrayanan S, «Movile Enabled Bus Tracking and Ticketing System,» de Conference on Information and Communication Tecnology, Bandug Indonesia, 2014.

[3] I. B. A. S. Cemil SUNGUR, «Smart Bus Station-Passenger Information System,» de 2015 2nd International Conference on Information Science and Control Engineering, Konya-Turkey, 2015.

[4] D. L. Hora, «La Hora,» La hora, 22 Septiembre 2016. [En línea]. Available: https://lahora.com.ec/noticia/1101985642/noticia. [Último acceso: 08 Agosto 2018].

[5] J. B. Pankaj Verma, «Design and Development of GPS-GSM Based Tracking System with Google Map Based Monitoring,» de International Journal of Computer Science, Engineering and Applications(IJCSEA) Vol.3,No.3,June 2013, India, 2013.

[6] INEC, «Instituto Naciona de Estadística y Censos(INEC),» INEC, 2010. [En línea]. Available: http://www.ecuadorencifras.gob.ec/censo-de-poblacion-y-vivienda/. [Último acceso: 11 19 2018].

[7] D. J. Quishpe Enriquéz y M. A. Reyes Navarrete, DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA BASADO EN GPS PARA EL ANUNCIO DE PARADAS EN LAS RUTAS DE LOS AUTOBUSES URBANOS DE LA CIUDAD DE IBARRA, Ibarra, 2017.

[8] Bismart, «Bismart,» 12 Julio 2017. [En línea]. Available: https://blog.bismart.com/es/que-es-exactamente-una-ciudad-inteligente. [Último acceso: 3 Noviembre 2019].

[9] D. Fernández, «El País,» 12 Enero 2016. [En línea]. Available: https://elpais.com/diario/2010/12/12/negocio/1292162603850215.html.

[10] A. . H. A. FOUNOUN, «Evaluation of the concept of the smart city through local regulation and the importance of local initiative,» de 2018 IEEE International Smart Cities Conference (ISC2), Kansas City, MO, USA, USA, 2018.

[11] W. J. Mitchell, «Ciudades inteligentes,» de LECCIÓN INAUGURAL DEL CURSO 2007-2008 DE LA UOC, Catalunya, 2008.

[12] G. M. (. C. S. F. S. Felipe D. Cunha (PUC-MG), «Sistemas de Transporte Inteligentes:Conceitos, Aplicac¸oes e Desafios,» 2 12 2019. [En línea]. Available: https://bps90.github.io/assets/files/2017-05-15-sbrc-minicurso-its.pdf. [Último acceso: 4 11 2019].

[13] A. M.-A. L. D. F. A.Pozo-Ruz, «SISTEMA DE POSICIONAMIENTO GLOBAL (GPS): DESCRIPCIÓN, ANÁLISIS DE ERRORES, APLICACIONES Y FUTURO,» de Instituto de Automática Industrial Consejo Superior de Investigaciones Científicas, Arganda. Madrid.

[14] A. K. B. A. Sturza, «Vehicle tracking system employing global positioning system (gps) satellites». USA Patente 5,225,842 , 6 Julio 1996.

[15] U. N. A. y. a. Distacia, «Universidad Nacional Abierta y a Distacia UNAD,»

[16] M. Dakic, «Data Driven Investor,» 2019. [En línea]. Available: https://medium.com/datadriveninvestor/how-to-develop-a-transportation-mobile-app-f97666398c09.

[17] J. M. S. Siuhi, «Opportunities and challenges of smart mobile applications in transportation,» KeAi, vol. 3, pp. 582-592.

[18] G. Kumparak, «TenchCrunch,» 20 Diciembre 2012. [En línea]. Available: https://techcrunch.com/2012/12/20/social-transit-app-moovit-launches-in-new-york-city-just-in-time-to-help-all-those-hapless-tourists/.

[19] Moovit, «Google Play,» 12 Noviembre 2019. [En línea]. Available: https://play.google.com/store/apps/details?id=com.tranzmate.

[20] M. Inc, «Guía de Usuario para Android,» Moovit, Israel, 2014.

[21] M. D. D. M. QUITO, «POLÍTICAS DE USO Y PRIVACIDAD PARA EL APLICATIVO MÓVIL: "MOVILIZATE UIO",» Quito, 2017.

[22] Q. Informa, «Quito Informa,» 19 Septiembre 2017. [En línea]. Available: http://www.quitoinforma.gob.ec/2017/09/19/municipio-de-quito-lanza-movilizate-uio-la-primera-aplicacion-movil-de-transporte-publico-del-ecuador/.

[23] [6] M. UIO, «Google Play,» 21 Diciembre 2019. [En línea]. Available: https://play.google.com.

[24] E. Universo, «El Universo,» 13 marzo 2017. [En línea]. Available: https://www.eluniverso.com/guayaquil/2017/03/13/nota/6088816/metrovia-guayaquil-habilita-su-app-ubicar-buses-mas-cercanos.

[25] M. Arribo, «Google play,» 12 Diciembre 2019. [En línea]. Available: https://play.google.com/store/apps/details?id=ec.transvia.snitchhl=es419.

[26] E. Norte, «EL NORTE,» 11 Julio 2016. [En línea]. Available: https://www.elnorte.ec/paradas-inteligentes-estaran-bajo-prueba/.

[27] I. S. T. Empresariales, «Google Play,» 23 Diciebre 2019. [En línea]. Available: https://play.google.com/store/apps/details?id=ec.inprise.citytouchhl=es419.

[28] Expectativa, «Expectativa,» 31 Mayo 2018. [En línea]. Available: http://www.expectativa.ec/empresa-espanola-que-administraba-paradas-inteligentes-alzo-vuelo/.

[29] L. L. C. A. B. M. D. Maira Cecilia Gasca Mantilla, «Metodología para el desarrollo de aplicaciones móviles,» Tecnura: Tecnología y Cultura Afirmando el Conocimiento, ISSN-e 0123-921X, , vol. Vol. 18, nº Nº. 40, pp. págs. 20-35, 2014.

[30] D. V., PLATAFORMA PARA EXPERIMENTACION NATURALISTICA EN BICICLETAS: SOFTWARE DE ADQUISICION Y TRATAMIENTO DE DATOS, Ibarra: UTN, 2018.

[31] C. S. L. A. C. C. D. G. G. M. E. C. M. C. M. Camps P. Rafael, Bases de datos, Catalunya: Eureca Media, SL, 2005.

[32] M. E. Millán, FUNDAMENTOS DE BASES DE DATOS, Cali: Programa Editorial Universidad del Valle, 2017.

[33] C. B. Cordová Rosa, Analisis comparativo entre bases de datos relacionales con base de datos no relacionales, Cuenca: Universidad Salesiana, 2013.

[34] Comprar Móvil, «Tu mejor móvil,» s.f.. [En línea]. Available: https://tumejormovil.com/sistemas-operativos/.

[35] C. F. Martín, Desarrollo de una aplicación móvil para la gestión y recomendación de información de actualidad, Leganés: Universidad Carlos III de Madrid, 2017.

[36] A. Studio, «Developers,» Android Studio, [En línea]. Available: https://developer.android.com/studio/intro?hl=es-419. [Último acceso: 12 11 2019].

[37] J. B. K. Malave, «.Android"Google´s aperatind system for mobile devices,» Negotium, vol. 19, pp. 79-96, 2011.

[38] F. A. W. F. S. Bohm, «Impact of the Mobile Operating System on Smartphone Buying Decisions: A Conjoint-Based Empirical Analysis,» RheinMain University of Applied Sciences.

[39] M. C. G., «Desarrollo de una aplicación Android de apuestas utilizando Firebase para la sincronización de datos,» Universitat Jaume I, Jaume, 2017.

[40] Google, «Firebase,» Google, [En línea]. Available: https://firebase.google.com. [Último acceso: 20 11 2019].

[41] E. Y. Marco, Aplicación Web y Móvil para el seguimiento de autobuses escolares, Valencia: Universitat Politècnica de València, 2014/2015.

[42] A. L. S. G. C. Challiol, «Diseño de aplicaciones móviles basadas en posicionamiento: un framework conceptual,» de XIV Workshop de Ingeniería de Software (WIS), 2017.

[43] C. Veness, «Movable Type Scripts,» [En línea]. Available: https://www.movable-type.co.uk/scripts/latlong.html. [Último acceso: 12 Diciembre 2019].

[44] C. C., BUS TRACKING SYSTEM, University Tunku Abdul Rahman, 2013.

[45] Developers, «Google Maps Platform,» [En línea]. Available: https://developers.google.com/maps/documentation/android-sdk/map-with-marker?hl=es. [Último acceso: 12 Agosto 2020].

[46] Developers, «Google Maps Platform,» [En línea]. Available: https://developers.google.com/maps/documentation/android-sdk/get-api-key?hl=es. [Último acceso: 12 Agosto 2020].

[47] Developers, «Web,» [En línea]. Available:https://developers.google.com/web/fundamentals/performan performance-matters?hl=es-419 .

[48] Usability, «usability.gov,» [En línea].https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. [Último acceso: 12 Agosto 2020].

Apéndice

.A. Aplicación en Android Studio

.A.1. Aplicación del administrador y conductor

.A.1.1. Manifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.mapasprueba">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".RegistroConductor"></activity>
        <activity android:name=".AdministradorPassword" />
        <activity android:name=".Registro" />
        <activity android:name=".Ubicaciones" />
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

.A.1.2. MainActivity

```
import android.app.ProgressDialog;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {


    private Button btnAdministrador;
    private ProgressDialog progressDialog;
    private  Button botonConductor;
    Button btnPrueba;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);



        //referenciamos
        btnAdministrador = (Button) findViewById(R.id.btn_administrador);
        botonConductor = (Button) findViewById(R.id.btn_Driver);

        progressDialog = new ProgressDialog(this);

        //Asociamos un oyente al evento onClick
        btnAdministrador.setOnClickListener(this);
        botonConductor.setOnClickListener(this);

    }

    private void admin(){
        Intent ubicacion = new Intent(getApplication(),AdministradorPassword.class);
```

```
        startActivity(ubicacion);
    }


    private void Driver(){
        Intent ubicacion = new Intent(getApplication(),RegistroConductor.class);
        startActivity(ubicacion);
    }


    @Override
    public void onClick(View view) {

        switch (view.getId()){

            case R.id.btn_administrador:
                //invocamos al metodo
                admin();
                break;
            case R.id.btn_Driver:
               Driver();

            break;
        }
    }
}
```

.A.1.3. Administrador

```
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
```

```
import android.widget.Toast;

public class AdministradorPassword extends AppCompatActivity implements View.OnClickListener {

    private EditText txtPassword;
    private EditText txtadmin;
    private Button btnIngresar;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_administrador_password);


        txtPassword = (EditText)findViewById(R.id.tv_passwordAdmin);
        txtadmin = (EditText)findViewById(R.id.tv_admin);
        btnIngresar = (Button)findViewById(R.id.btn_InicioAdmin);

        btnIngresar.setOnClickListener(this);
    }

    private  void ingreso() {

        String password = txtPassword.getText().toString().trim();
        String admin = txtadmin.getText().toString().trim();

        //verificamos que las cajas de texto no esten vacias

        if (admin.equals("admin") && password.equals("admin")){

            Intent  siguiente = new Intent(getApplicationContext(), Registro.class);
            startActivity(siguiente);

        }else
        {
            Toast.makeText(getApplicationContext(), "Usuario o contrase a erronea", Toast.
                LENGTH_SHORT).show();
        }

    }

    @Override
```

```
    public void onClick(View v) {
    ingreso();
    }
}
```

Registro de Conductores

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.util.HashMap;
import java.util.Map;

public class Registro extends AppCompatActivity implements View.OnClickListener{

    private EditText txtNombreC;
    private EditText txtnroBus;
    private EditText txtCooperativa;
    private EditText txtRutaC;
    private EditText txtHorario;
    private EditText txtCelular;
    public EditText txtEmail;
    public EditText txtContrase a;
    private Button btnRegistrar;
```

```
//variables de registro

private String nombre = "";
private String bus = "";
private String cooperativa = "";
private String ruta = "";
private String horario = "";
private String celular = "";
private String contrase a = "";
private String email = "";

// Variables de Firebase

public  FirebaseAuth mAuth;
public DatabaseReference mDatabase;
private ProgressDialog progressDialog;


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_registro);

    mAuth = FirebaseAuth.getInstance();
    mDatabase = FirebaseDatabase.getInstance().getReference();

    txtNombreC = (EditText)findViewById(R.id.tv_nombreConductor);
    txtnroBus = (EditText)findViewById(R.id.tv_numBUs);
    txtCooperativa = (EditText)findViewById(R.id.tv_cooperativa);
    txtRutaC = (EditText)findViewById(R.id.tv_Ruta);
    txtHorario = (EditText)findViewById(R.id.tv_horario);
    txtCelular = (EditText)findViewById(R.id.tv_celular);
    txtEmail = ( EditText)findViewById(R.id.tv_email);
    txtContrase a = (EditText)findViewById(R.id.tv_passwordConductor);
    btnRegistrar = (Button)findViewById(R.id.btn_registrar);

    btnRegistrar.setOnClickListener(this);

    progressDialog = new ProgressDialog(this);

}
```

```
private void registroDatos(){

    nombre = txtNombreC.getText().toString().trim();
    bus = txtnroBus.getText().toString().trim();
    cooperativa = txtCooperativa.getText().toString().trim();
    ruta = txtRutaC.getText().toString().trim();
    horario = txtHorario.getText().toString().trim();
    celular = txtCelular.getText().toString().trim();
    email = txtEmail.getText().toString().trim();
    contrase a = txtContrase a.getText().toString().trim();

    // preguntamos si los campos no estan vacios

    if(!nombre.isEmpty() && !bus.isEmpty() && !cooperativa.isEmpty() && !ruta.isEmpty()
    && !horario.isEmpty() && !celular.isEmpty() && !email.isEmpty() && !contrase a.
        isEmpty()) {

        if(contrase a.length() >= 6 ){

            if(celular.length() == 10){

                registrarUsuario();
                progressDialog.setMessage("Registrando Conductor...");
                progressDialog.show();

            }else {
                Toast.makeText(getApplicationContext(), "El nro debe tener 10 d gitos",
                    Toast.LENGTH_SHORT).show();
            }

        }else{
            Toast.makeText(getApplicationContext(), "La contrase a debe contener almenos
                6 caracteres", Toast.LENGTH_SHORT).show();
        }



    }

    else {
        Toast.makeText(getApplicationContext(), "Debe llenar todos los Datos", Toast.
            LENGTH_SHORT).show();
    }
```

```
        progressDialog.dismiss();

    }

    private void registrarUsuario() {
        mAuth.createUserWithEmailAndPassword(email,contrase a).addOnCompleteListener(new
            OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {

                if(task.isSuccessful()){

                    //traemos los valores que queremos almacenar en la base de datos

                    HashMap<String, Object> datos = new HashMap<>();
                    datos.put("Nombre", nombre);
                    datos.put("Nro Bus", bus);
                    datos.put("Cooperativa", cooperativa);
                    datos.put("Ruta", ruta);
                    datos.put("Horario", horario);
                    datos.put("Celular", celular);
                    datos.put("Email", email);
                    datos.put("Contrase a", contrase a);

                    String userId = mAuth.getCurrentUser().getUid();

                    mDatabase.child("Conductores").child(userId).setValue(datos).
                        addOnCompleteListener(new OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task2) {
                            if(task2.isSuccessful()){

                                Toast.makeText(getApplicationContext(), "El conductor fue
                                    registrado", Toast.LENGTH_SHORT).show();

                                Intent principal = new Intent(getApplication(), MainActivity.
                                    class);
                                startActivity(principal);
                                finish();
                            }else {
                                Toast.makeText(getApplicationContext(), "Los datos no fueron
                                    registrados", Toast.LENGTH_SHORT).show();
                            }
```

```
                    }
                });
            }else {
                Toast.makeText(getApplicationContext(), "NO se pudo registrar el usuario
                    Verifique los datos", Toast.LENGTH_SHORT).show();
            }
        }
    });
}


@Override
public void onClick(View v) {
    registroDatos();
}
}
```

.A.1.4. Conductor

Autenticación

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.app.ProgressDialog;
import android.content.Intent;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
```

```
import com.google.firebase.auth.FirebaseAuthUserCollisionException;

public class RegistroConductor extends AppCompatActivity implements  View.OnClickListener{


    private EditText txtUser;
    private EditText txtPassword;
    private ProgressDialog progressDialog;
    private FirebaseAuth mAuth;
    private  Button btnIniciarSesion;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_registro_conductor);


        mAuth = FirebaseAuth.getInstance();

        //referenciamos
        txtUser = (EditText) findViewById(R.id.tv_email);
        txtPassword = (EditText) findViewById(R.id.tv_passwordConductor);
                btnIniciarSesion = (Button) findViewById(R.id.btn_login);

        progressDialog = new ProgressDialog(this);


        //Asociamos un oyente al evento onClick

        btnIniciarSesion.setOnClickListener(this);


    }



    private  void iniciarSesion(){

        final String email = txtUser.getText().toString().trim();
        String password = txtPassword.getText().toString().trim();

        //verificamos que las cajas de texto no esten vacias
        if(TextUtils.isEmpty(email)){
```

```
        Toast.makeText(this, "Se debe ingresar email", Toast.LENGTH_SHORT).show();
        return;
    }
    if(TextUtils.isEmpty(password)){
        Toast.makeText(this, "Se debe ingresar password", Toast.LENGTH_SHORT).show();
        return;
    }

    //barra de progreso
    progressDialog.setMessage("Realizando consulta en linea...");
    progressDialog.show();
    //iniciar secion
    mAuth.signInWithEmailAndPassword(email,password)
            .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    //
                    if(task.isSuccessful()){
                        //mostramos solo los caranteres antes del @
                        int posicion = email.indexOf("@");
                        String user= email.substring(0,posicion);

                        Toast.makeText(RegistroConductor.this, "bienvenido "+ txtUser.
                                getText(), Toast.LENGTH_LONG).show();
                        //pasamos de actividad
                        Intent ubicacion = new Intent(getApplication(),Ubicaciones.class);
                        ubicacion.putExtra(Ubicaciones.user, user);
                        startActivity(ubicacion);


                    } else {
                        if(task.getException() instanceof
                                FirebaseAuthUserCollisionException){//verificamos si hay
                                alguna colision
                                Toast.makeText(RegistroConductor.this, "El Usuario ya existe",
                                        Toast.LENGTH_SHORT).show();
                        }else {
                            Toast.makeText(RegistroConductor.this, "No se ha registrado el
                                    email", Toast.LENGTH_LONG).show();
                        }
                    }
                    progressDialog.dismiss();
                }
```

```
                });
    }

    @Override
    public void onClick(View view) {

        switch (view.getId()){

            case R.id.btn_login:
                iniciarSesion();

                break;
        }
    }
}
```

Obtención de localización

```
import androidx.appcompat.app.AppCompatActivity;
import androidx.core.app.ActivityCompat;

import android.Manifest;
import android.content.Context;
import android.content.Intent;
import android.content.pm.PackageManager;
import android.database.sqlite.SQLiteDatabase;
import android.location.Address;
import android.location.Geocoder;
import android.location.Location;
import android.location.LocationListener;
import android.location.LocationManager;
import android.location.LocationProvider;
import android.os.Bundle;
import android.provider.Settings;
import android.util.Log;
import android.widget.Button;
import android.widget.SeekBar;
import android.widget.TextView;
```

```
import com.firebase.geofire.GeoFire;
import com.firebase.geofire.GeoLocation;
import com.firebase.geofire.LocationCallback;
import com.google.android.gms.location.LocationResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import java.io.IOException;
import java.util.HashMap;
import java.util.List;
import java.util.Locale;
import java.util.Map;

public class Ubicaciones extends AppCompatActivity {

    public static final String user = "names";
    TextView txtUser;

    TextView latitud,longitud;
    TextView direccion;

    DatabaseReference midatos;

    //variables velocidad
    LocationManager locationManager;//
    SQLiteDatabase db;//initialize variable of sqlite database type
    TextView mvelocidad;
    SeekBar seekbar1, seekbar2;
    Button button1,button2;
    long start,finish,time;//
    double lat1,lon1,lat2,lon2,alt1,alt2,time1, speed=1;
    public String[] radiuspois;
    int counts;
    // para  calculo de velocidad promedio

    double[] velocidad_Prome = new double[3];
    double suma_velo;
    double prome_vel=1;

    double distance = 0;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_ubicaciones);
    txtUser = (TextView) findViewById(R.id.txt_comunicacion);

    String user = getIntent().getStringExtra("names");
    txtUser.setText(" Bienvenido " + user + " :)");

    midatos = FirebaseDatabase.getInstance().getReference();
    mvelocidad = (TextView)findViewById(R.id.tv_velocidad);
    latitud = (TextView) findViewById(R.id.txtLatitud);
    longitud = (TextView) findViewById(R.id.txtLongitud);
    direccion = (TextView) findViewById(R.id.txtDireccion);



    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
            Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED)
            {
            ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.
                ACCESS_FINE_LOCATION,}, 1000);
    } else {
        locationStart();
    }


}
private void locationStart() {
    LocationManager mlocManager = (LocationManager) getSystemService(Context.
        LOCATION_SERVICE);
    Localizacion Local = new Localizacion();
    Local.setMainActivity(this);
    final boolean gpsEnabled = mlocManager.isProviderEnabled(LocationManager.GPS_PROVIDER)
        ;
    if (!gpsEnabled) {
        Intent settingsIntent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
        startActivity(settingsIntent);
    }
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermission(this,
```

```
        Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED)
        {
        ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.
            ACCESS_FINE_LOCATION,}, 1000);
        return;
    }
    mlocManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 0, 0, (
        LocationListener) Local);
    mlocManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, (
        LocationListener) Local);
    latitud.setText("Localizacin agregada");
    direccion.setText("");
}
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
    grantResults) {
    if (requestCode == 1000) {
        if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
            locationStart();
            return;
        }
    }
}

public void setLocation(Location loc) {
    //Obtener la direccion de la calle a partir de la latitud y la longitud
    if (loc.getLatitude() != 0.0 && loc.getLongitude() != 0.0) {
        try {
            Geocoder geocoder = new Geocoder(this, Locale.getDefault());
            List<Address> list = geocoder.getFromLocation(
                    loc.getLatitude(), loc.getLongitude(), 1);
            if (!list.isEmpty()) {
                Address DirCalle = list.get(0);
                direccion.setText(DirCalle.getAddressLine(0));
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
/* Aqui empieza la Clase Localizacion */
public class Localizacion implements LocationListener {
    Ubicaciones mainActivity;
    public Ubicaciones getMainActivity() {
```

```
            return mainActivity;
        }
        public void setMainActivity(Ubicaciones mainActivity) {
            this.mainActivity = mainActivity;
        }
        @Override
        public void onLocationChanged(Location loc ) {



            String distaciaaaaa = String.valueOf(prome_vel);
            Log.e("velocidad ",distaciaaaaa);


//prueba de velocidad mediante speed integrada por android.
           /* if (loc==null){

                mvelocidad.setText("1.0 km/h");
            } else {
                double nCurrentSpeed = loc.getSpeed() * 3.6f;
                speed = nCurrentSpeed*16.66666;
                mvelocidad.setText(String.format("%.2f", speed)+ " m/min" );
                //mvelocidad.setText(Double.toString(nCurrentSpeed ));
            }*/

//// sacamos la velocidad de cambio de puntos mediante calculo con formula de haversine
            if(counts==0) {
                start = System.nanoTime();// comienza timer
                lat1 = loc.getLatitude();//tguarda coordenadas
                lon1 = loc.getLongitude();
                distance = 0;
                counts +=1;
            }
            else if (counts == 5){
                finish = System.nanoTime();//finaliza timer
                lat2 = loc.getLatitude();//guarda coordenadas
                lon2 = loc.getLongitude();
                time = finish - start;//guardamos el tiempo en nano segundos
                time1 = (double)time / 1_000_000_000.0;//en segundos
                distance = distance + measureDistance(lat1,lat2,lon1,lon2);//calculamos la
                    distacia entre dos puntos llamando al metdo de calculo de distancia
                speed = distance / time1;//calculo de distacia en m/s
                speed = speed * 3.6;//de m/s a km/h
```

```
        speed = (int) speed;//quitamos decimales convirtiendo a entero
        mvelocidad.setText(Double.toString(speed));// se muestra la velocidad
        start = 0;//reiniciamos el contador
        finish = 0;
        counts=0;

    }
    else{
        lat2 = loc.getLatitude();
        lon2 = loc.getLongitude();
        distance = distance + measureDistance(lat1,lat2,lon1,lon2);
        lat1 = lat2;
        lon1 = lon2;
        counts +=1;
    }

    //
    for (int i=0;i<velocidad_Prome.length;i++){
        velocidad_Prome[i]=speed;
        Log.e("velocidad333 ", String.valueOf(velocidad_Prome));
    }
    for (int i=0;i<velocidad_Prome.length;i++){

        suma_velo+=velocidad_Prome[i];
        String distaciasuma = String.valueOf(suma_velo);
        Log.e("sumavelo ",distaciasuma);
    }
    prome_vel= suma_velo/velocidad_Prome.length;

    // Este metodo se ejecuta cada vez que el GPS recibe nuevas coordenadas
    // debido a la deteccion de un cambio de ubicacion
    loc.getLatitude();
    loc.getLongitude();

    String sLatitud = String.valueOf(loc.getLatitude());
    String sLongitud = String.valueOf(loc.getLongitude());
    latitud.setText(sLatitud);
    longitud.setText(sLongitud);
    this.mainActivity.setLocation(loc);
```

```
    String userId = FirebaseAuth.getInstance().getUid();
    DatabaseReference ref = FirebaseDatabase.getInstance().getReference("Conductores")
        ;

    GeoFire geoFire = new GeoFire(ref);
    //estos ser s los hijos de userid
    geoFire.setLocation("you", new GeoLocation( loc.getLatitude(), loc.getLongitude())
        );

    Map<String,Object> velo = new HashMap<>();

    velo.put("Velocidad",speed);
    velo.put("latitudd", loc.getLatitude());
    velo.put("longitudd", loc.getLongitude());



    midatos.child("Conductores").child(userId).updateChildren(velo);
   // midatos.child("velocidad").updateChildren(velo);


  prome_vel=0;

}
@Override
public void onProviderDisabled(String provider) {
    // Este metodo se ejecuta cuando el GPS es desactivado
    latitud.setText("GPS Desactivado");
}
@Override
public void onProviderEnabled(String provider) {
    // Este metodo se ejecuta cuando el GPS es activado
    latitud.setText("GPS Activado");
}
@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
    switch (status) {
        case LocationProvider.AVAILABLE:
            Log.d("debug", "LocationProvider.AVAILABLE");
            break;
        case LocationProvider.OUT_OF_SERVICE:
            Log.d("debug", "LocationProvider.OUT_OF_SERVICE");
            break;
```

```
                case LocationProvider.TEMPORARILY_UNAVAILABLE:
                    Log.d("debug", "LocationProvider.TEMPORARILY_UNAVAILABLE");
                    break;
            }
        }
    }


    public double measureDistance(Double lat1,Double lat2,Double lon1,Double lon2){
        final int R = 6371; // Radio de la tierra
        double latDistance = Math.toRadians(lat2 - lat1);
        double lonDistance = Math.toRadians(lon2 - lon1);
        double a = Math.sin(latDistance / 2) * Math.sin(latDistance / 2) + Math.cos(Math.
            toRadians(lat1)) * Math.cos(Math.toRadians(lat2)) * Math.sin(lonDistance / 2) *
            Math.sin(lonDistance / 2);
        double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
        double distance = R * c * 1000; // convierto a metros
        return distance;
    }

    public  double promVelocidad(){

        for (int i=0;i<velocidad_Prome.length;i++){
            velocidad_Prome[i]=speed;
            Log.e("velocidad333 ", String.valueOf(velocidad_Prome));
        }
        for (int i=0;i<velocidad_Prome.length;i++){

            suma_velo+=velocidad_Prome[i];
            String distaciaaaaa = String.valueOf(suma_velo);
            Log.e("sumavelo ",distaciaaaaa);
        }
        prome_vel= suma_velo/velocidad_Prome.length;

        return prome_vel;
    }
}
```

.A.2. Aplicación del Usuario

.A.2.1. Manisfest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.ibabus">
    <!--
         The ACCESS_COARSE/FINE_LOCATION permissions are not required to use
         Google Maps Android API v2, but you must specify either coarse or fine
         location permissions for the 'MyLocation' functionality.
    -->
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <uses-feature
        android:glEsVersion="0x00020000"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".Rutas2"></activity>
        <activity android:name=".Rutas" />
        <!--
             The API key for Google Maps-based APIs is defined as a string resource.
             (See the file "res/values/google_maps_api.xml").
             Note that the API key is linked to the encryption key used to sign the APK.
             You need a different API key for each encryption key, including the release key
                 that is used to
             sign the APK for publishing.
```

```
            You can define the keys for the debug and release targets in src/debug/ and src/
                release/.
        -->
        <meta-data
            android:name="com.google.android.geo.API_KEY"
            android:value="@string/google_maps_key" />

        <activity
            android:name=".MapsActivity"
            android:label="@string/title_activity_maps" />
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <meta-data
            android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />

        <uses-library
            android:name="org.apache.http.legacy"
            android:required="false" />
    </application>

</manifest>
```

.A.2.2. MainActivity

```
import android.location.LocationManager;
import android.os.Bundle;
import android.view.Menu;
```

```
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;

import androidx.appcompat.app.ActionBarDrawerToggle;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import androidx.core.view.GravityCompat;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.fragment.app.FragmentManager;

import com.google.android.gms.location.FusedLocationProviderClient;
import com.google.android.gms.location.LocationServices;
import com.google.android.material.floatingactionbutton.FloatingActionButton;
import com.google.android.material.navigation.NavigationView;
import com.google.android.material.snackbar.Snackbar;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {


    ContactoFragment cargaMapa;


    private int MY_PERMISSIONS_REQUEST_READ_CONTACTS ;
    private FusedLocationProviderClient fusedLocationClient;
    private LocationManager ubicacion;

    private DatabaseReference miDataBase;
    TextView mVelocidad;

    ///seleccion de cada item
    int lineas= 0;




    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
setContentView(R.layout.activity_main);
Toolbar toolbar = findViewById(R.id.toolbar);
setSupportActionBar(toolbar);
FloatingActionButton fab = findViewById(R.id.fab);
fab.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                .setAction("Action", null).show();
    }
});
DrawerLayout drawer = findViewById(R.id.drawer_layout);
NavigationView navigationView = findViewById(R.id.nav_view);
ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
        this, drawer, toolbar, R.string.navigation_drawer_open, R.string.
            navigation_drawer_close);
drawer.addDrawerListener(toggle);
toggle.syncState();
navigationView.setNavigationItemSelectedListener(this);

// instancias o objetos de la clase aqui es donde validare to
//todos los Fragments



miDataBase = FirebaseDatabase.getInstance().getReference();



cargaMapa = new ContactoFragment();


/** Para que aparesca el mapa o cualquier actividad en inicio
 * lo debo declarar de esta forma
 */
getSupportFragmentManager().beginTransaction().replace(R.id.content_main, cargaMapa).
    commit();

//poner colores a los iconos
navigationView.setItemIconTintList(null);
///fused  locatiom
fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);
```

```
    //almacenamiento en firebase


}//FIN oncreate


@Override
public void onBackPressed() {

    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    if (drawer.isDrawerOpen(GravityCompat.START)) {
        drawer.closeDrawer(GravityCompat.START);
    } else {
        super.onBackPressed();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle action bar item clicks here. The action bar will
    // automatically handle clicks on the Home/Up button, so long
    // as you specify a parent activity in AndroidManifest.xml.
    int id = item.getItemId();

    //noinspection SimplifiableIfStatement
    if (id == R.id.action_settings) {
        return true;
    }

    return super.onOptionsItemSelected(item);
}

@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    // Handle navigation view item clicks here.
```

```
        int id = item.getItemId();


        if (id == R.id.Mapa_principal) {

            // realizamos la asignacion del contenedor y validacion
            ContactoFragment contactoFragment = new ContactoFragment();
            FragmentManager manager = getSupportFragmentManager();
            manager.beginTransaction().replace(R.id.content_main, contactoFragment).commit();

        } else if (id == R.id.nav_gallery) {

        } else if (id == R.id.nav_slideshow) {

        } else if (id == R.id.nav_tools) {
            Lineas lineas = new Lineas();
            FragmentManager manager = getSupportFragmentManager();
            manager.beginTransaction().replace(R.id.content_main, lineas).commit();

        } else if (id == R.id.nav_share) {

        } else if (id == R.id.nav_send) {
        }
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }
}
```

.A.2.3. MapsActivity

```
mport android.os.Bundle;

import androidx.fragment.app.FragmentActivity;

import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
```

```
import com.google.android.gms.maps.SupportMapFragment;

public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {



    //POsicionamiento de posicion actual
    private GoogleMap mMap;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_maps);
        // Obtain the SupportMapFragment and get notified when the map is ready to be used.
        SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
                .findFragmentById(R.id.map);
        mapFragment.getMapAsync(this);

    }


    @Override
    public void onMapReady(GoogleMap googleMap) {
        mMap = googleMap;

    }
}
```

.A.2.4. Fragment

```
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolygonOptions;
import com.google.android.gms.maps.model.PolylineOptions;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;

public class ContactoFragment<distace> extends Fragment implements OnMapReadyCallback {

    private static int CAMERA_ZOOM = 14;
    GoogleMap mMap;// varible de mapa

    private DatabaseReference midatabaseReference;

    private ArrayList<Marker> anteriorMarker = new ArrayList<>();
    private ArrayList<Marker> actualMarker = new ArrayList<>();

    private ArrayList<Double> actualDistancia = new ArrayList<>();
    private ArrayList<Double> anteriorDistancia = new ArrayList<>();
```

```
    LatLng posicionmove;

    //prueba
    //LatLng Parada1, Parada2;
    TextView mdistancia;
    TextView mVelocidad;
    TextView mTiempo;
    String velocidad;
    String dista;

    double speed=1;
    double distance=1;
    double [] distancia = new double[100];
    double [] tiempo_array = new double[5];

    LatLng Parada1 = new LatLng(0.088571666, -78.092111166);
    LatLng Parada2 = new LatLng(0.0934533,-78.078876);

    LatLng[] Paradas = {Parada1,Parada2};

/// calculo de datos para promedio

    double tiempo_llegada =0;
    double suma_distance;
    double promedio_tiempo =0;
    double suma_tiempo;




    public ContactoFragment() {

        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View root = inflater.inflate(R.layout.fragment_contacto, container, false);
```

```
        return root;
        //creaccion de movimiento marcador


    }

    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        SupportMapFragment mapFragment = (SupportMapFragment) getChildFragmentManager()
                .findFragmentById(R.id.mapasfragment);
        mapFragment.getMapAsync(this);

        midatabaseReference = FirebaseDatabase.getInstance().getReference();

        mVelocidad = (TextView)getActivity().findViewById(R.id.Tv_velocidad);
        mTiempo = (TextView)getActivity().findViewById(R.id.tv_tiempo);




        lecturaDatos();

    }

    private List<WeatherSample> weatherSamples = new ArrayList<>();

    private void lecturaDatos() {
        InputStream is = getResources().openRawResource(R.raw.prueba2ruta);
        BufferedReader reader = new BufferedReader(new InputStreamReader(is, Charset.forName("
            UTF-8")));
        String line = "";
        try {
            //paso sobre las cabeceras

            reader.readLine();

            while (((line = reader.readLine()) != null)) {
                Log.d("Myactividad", "Line" + line);
                // split creamos una matriz
                String[] tokens = line.split(";");
```

```
                //lectura de datos
                WeatherSample sample = new WeatherSample();
                sample.setLatitud((float) Double.parseDouble(tokens[0]));
                sample.setLongitud((float) Double.parseDouble(tokens[1]));
                weatherSamples.add(sample);

                Log.d("mi activida", "solo creado:  " + sample);
            }


        } catch (IOException e) {
            Log.wtf("My", "error leyendo datos" + line, e);
            e.printStackTrace();
        }


    }







    @Override
    public void onMapReady(GoogleMap googleMap) {

        mMap = googleMap;


// Asyntask



                midatabaseReference.child("Conductores").addValueEventListener(new
                    ValueEventListener() {

                    @Override
                    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

                        for(Marker marker: actualMarker){

                            marker.remove();
```

```
    }

    for(DataSnapshot snapshot: dataSnapshot.getChildren()){
       // List<Object> map = (List<Object>) dataSnapshot.getValue();

        ObtencionDatos ubicaciones = snapshot.getValue(ObtencionDatos.
            class);
        Double locationLat = ubicaciones.getLatitudd();
        Double locationLng = ubicaciones.getLongitudd();

         posicionmove = new LatLng(locationLat, locationLng);
        //prueba de datos en logcat
        String daaa = String.valueOf(locationLat);
        Log.e("longitud ",daaa);



        MarkerOptions markerOptions = new MarkerOptions()
                .position(new LatLng(locationLat,locationLng));
        markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawable
            .ic_bus_round));
        markerOptions.title("hola ");
        //guadamos marcadores en primer array
        anteriorMarker.add(mMap.addMarker(markerOptions));
       // mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(
            posicionmove, CAMERA_ZOOM));



    }

    /// borramos marcador
    actualMarker.clear();
    //cambiamos de array
    actualMarker.addAll(anteriorMarker);

    timer();

}

@Override
public void onCancelled(@NonNull DatabaseError databaseError) {
```

```
                }


            });



        // Ahora que el Mapa estalisto:
        // Agregamos un MArker en laUniversidad Tecnica del norte, Ibarra y movemos la camara.
        setMarker(Posiciones.CayambeRuta, "Universidad Tecnica del Norte", "Distrito: Ibarra")
            ;


        //Dibujamos una Polylinea
        //drawPolilyne(Posiciones.POLILINEA);
        //Dibujamos una Polygono
        drawPoligono(Posiciones.POLIGONO);


    }

    //timer
    private void timer(){

        new CountDownTimer(1000, 1000) {

            public void onTick(long millisUntilFinished) {
                Log.e("seconds remaining: ", "" + millisUntilFinished / 1000);

            }

            public void onFinish() {
                //Toast.makeText(getActivity(), "puntos actualizados", Toast.LENGTH_SHORT).
                    show();
                onMapReady(mMap);

            }
        }.start();

    }
```

```
    public void setMarker(LatLng position, String titulo, String info) {
        // Agregamos marcadores para indicar sitios de intereses.
        mMap.addMarker(new MarkerOptions()
                .position(position)
                .title(titulo)  //Agrega un titulo al marcador
                .snippet(info));   //Agrega informacion detalle relacionada con el marcador
                //.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.
                    HUE_GREEN))); //Color del marcador
        mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(position, CAMERA_ZOOM));

        //mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(position, CAMERA_ZOOM));
        //PARADAS
        //Parada1 = new LatLng(0.0889989,-78.092222);


    }

    //public void drawPolilyne(PolylineOptions options){
       // mMap.addPolyline(options);
  //  }

    public void drawPoligono(PolygonOptions opts){
        mMap.addPolygon(opts);
    }
}
```

.A.2.5. Modelo

```
public class Modelo {
    //odo lo que va ir en el raw

    String nombre;
    String edad;
    int image;

    // constructor
```

```
    public Modelo(String nombre, String edad, int image) {
        this.nombre = nombre;
        this.edad = edad;
        this.image = image;
    }

    //setters y getters

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getEdad() {
        return edad;
    }

    public void setEdad(String edad) {
        this.edad = edad;
    }

    public int getImage() {
        return image;
    }

    public void setImage(int image) {
        this.image = image;
    }
}
```

.A.2.6. Adaptador de lista

```
public class ListAdapter extends BaseAdapter implements Filterable {
```

```
private Context context;
private List<Modelo> mList;
private LayoutInflater inflater;
List<Modelo> mValue;
ValueFilter valueFilter;

public ListAdapter(Context context, List<Modelo> mList) {
    this.context = context;
    this.mList = mList;
    mValue = mList;
}


@Override
public int getCount() {
    return mList.size();
}

@Override
public Object getItem(int position) {
    return mList.get(position);
}

@Override
public long getItemId(int position) {
    return position;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {

    if(inflater == null)
    {
        inflater = (LayoutInflater) context.getSystemService(Context.
            LAYOUT_INFLATER_SERVICE);

    }
    if (convertView ==  null)
    {
        convertView = inflater.inflate(R.layout.item_raw,null);
    }
    TextView txtNombre = convertView.findViewById(R.id.txt_Nombre);
    TextView txtEdad = convertView.findViewById(R.id.txtView_Edad);
```

```
    ImageView image = convertView.findViewById(R.id.image_View);

    Modelo modelo = mList.get(position);
    String nombre = modelo.getNombre();
    String edad = modelo.getEdad();
    int image1 = modelo.getImage();

    txtNombre.setText(nombre);
    txtEdad.setText(edad);
    image.setImageResource(image1);


    return convertView;
}

@Override
public Filter getFilter() {

    if (valueFilter == null)
    {
        valueFilter = new ValueFilter();
    }
    return valueFilter;
}

private class ValueFilter extends  Filter {
    @Override
    protected FilterResults performFiltering(CharSequence constraint) {
        FilterResults results = new FilterResults();
        if (constraint != null && constraint.length() > 0)
        {
            ArrayList<Modelo> filterList = new ArrayList<Modelo>();
            for (int i = 0; i<mValue.size();i++)
            {
                if((mValue.get(i).getNombre().toUpperCase()).contains(constraint.toString
                    ().toUpperCase()))
                {
                    Modelo modelo = new Modelo(mValue.get(i).getNombre(),mValue.get(i).
                        getEdad(),mValue.get(i).getImage());
                    filterList.add(modelo);
                }
            }
            results.count = filterList.size();
```

```
                results.values = filterList;

            } else
            {
                results.count = mValue.size();
                results.values = mValue;
            }
            return results;
        }

        @Override
        protected void publishResults(CharSequence constraint, FilterResults results) {

            mList = (ArrayList<Modelo>)results.values;
            notifyDataSetChanged();

        }
    }
}
```

.A.2.7. Obtención de datos

```
class WeatherSample {
    private float latitud;
    private float longitud;

    public float getLongitud() {
        return longitud;
    }

    public float getLatitud() {
        return latitud;
    }

    public void setLatitud(float latitud) {
        this.latitud = latitud;
    }
```

```
    public void setLongitud(float longitud) {
        this.longitud = longitud;
    }

    @Override
    public String toString() {
        return "WeatherSample{" +
                "latitud=" + latitud +
                ", longitud=" + longitud +
                '}';
    }
}
```

.A.2.8. Lineas

```
public class Lineas extends Fragment  {

    //lista
    SearchView searchView;
    private ListView mlistView;
    private List<Modelo> mLista = new ArrayList<>();
    ListAdapter mAdapter;

    String [] nombre = {"Linea 1", "Linea 2","Linea 3"};/// se puede tambien importar de desde
         strings
    String [] edad = {"\n 23","\n 23","\n 12"};
    int[] image = {
            R.mipmap.ic_paradas,
            R.mipmap.ic_paradas,
            R.mipmap.ic_paradas};

    public Lineas() {
        // Required empty public constructor
    }
```

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    return inflater.inflate(R.layout.fragment_lineas, container, false);
}

@Override
public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);


    searchView = getActivity().findViewById(R.id.buscador);
    mlistView = getActivity().findViewById(R.id.listView);



    mLista = new ArrayList<Modelo>();

    for (int j= 0;j<nombre.length;j++)
    {
        Modelo modelo = new Modelo(nombre[j],edad[j],image[j]);
        mLista.add(modelo);
    }

    for (int i = 0;i<nombre.length;i++)
    {
        Modelo modelo = mLista.get(i);
        String s = modelo.getNombre();
        Log.e("Nombre",s);

    }


    mAdapter = new ListAdapter(getActivity().getApplicationContext(),mLista);
    mlistView.setAdapter(mAdapter);
    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override
        public boolean onQueryTextSubmit(String s) {
            mAdapter.getFilter().filter(s);

            return false;
        }
```

```
            @Override
            public boolean onQueryTextChange(String s) {
                mAdapter.getFilter().filter(s);
                return false;
            }
        });

        mlistView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            public void onItemClick(AdapterView<?> arg0, View v, int position, long id) {
                if (position == 0) {
                    Intent myIntent = new Intent(getActivity().getApplicationContext(),Rutas.
                        class);
                    startActivity(myIntent);
                }
                if (position == 1 ) {
                    Intent myIntent = new Intent(getActivity().getApplicationContext(),Rutas2.
                        class);
                    startActivity(myIntent);
                }
            }
        });



    }

}
```

.A.2.9. Rutas, paradas y tiempos de llegada

```
import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;
import android.icu.text.Collator;
import android.location.Location;
```

```
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolygonOptions;
import com.google.android.gms.maps.model.PolylineOptions;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.nio.charset.Charset;
import java.util.ArrayList;
import java.util.List;

public class Rutas2 extends AppCompatActivity implements OnMapReadyCallback {

    private static int CAMERA_ZOOM = 16;
    GoogleMap mMap;// varible de mapa

    private DatabaseReference midatabaseReference;

    private ArrayList<Marker> anteriorMarker = new ArrayList<>();
    private ArrayList<Marker> actualMarker = new ArrayList<>();
```

```
private ArrayList<Double> actualDistancia = new ArrayList<>();
private ArrayList<Double> anteriorDistancia = new ArrayList<>();

//prueba
//LatLng Parada1, Parada2;
TextView mdistancia;
TextView mVelocidad;
TextView mTiempo_p1,mTiempo_p2,mTiempo_p3,mTiempo_p4,mTiempo_p5,mTiempo_p6;
String velocidad;
String dista;

double speed=240;


// creacion variables de paradas
//cachiloma
LatLng Parada1 = new LatLng(0.091219, -78.097099);
//escuela cariacu
LatLng Parada2 = new LatLng(0.091363, -78.099116);
//Barrio florida
LatLng Parada3 = new LatLng(0.089284, -78.101006);
//barrio central
LatLng Parada4 = new LatLng(0.085816, -78.102374);
//barrio Sanpedro
LatLng Parada5 = new LatLng(0.083713, -78.105152);
//villamarin
LatLng Parada6 = new LatLng(0.078894, -78.108357);
LatLng Parada7 = new LatLng(0.071909, -78.106922);
LatLng Parada8 = new LatLng(0.072780, -78.109810);
//calle rodrigez lara
LatLng Parada9 = new LatLng(0.071987, -78.110278);
// escuela paquiestancia
LatLng Parada10 = new LatLng(0.070881, -78.111267);
//tanque de leche paquiestancia
LatLng Parada11 = new LatLng(0.070270, -78.113881);

LatLng Parada12 = new LatLng(0.062296, -78.136094);
//colegio nelson torres
LatLng Parada13 = new LatLng(0.050655, -78.141960);
//rio blanco
LatLng Parada14 = new LatLng(0.048717, -78.142546);

LatLng[] Paradas = {Parada1,Parada2};
```

```
//tiempos
double tiempo_p1 =0;
double tiempo_p2 =0;
double tiempo_p3 =0;
double tiempo_p4 =0;
double tiempo_p5 =0;
double tiempo_p6 =0;
double tiempo_p7 =0;
//distancias
double distance_p1=1;
double distance_p2=1;
double distance_p3=1;
double distance_p4=1;
double distance_p5=1;
double distance_p6=1;
double distance_p7=1;
//botones
Button mAtras;
private Collator FirebaseAuth;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_rutas);


    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map_ruta);
    mapFragment.getMapAsync(this);

    midatabaseReference = FirebaseDatabase.getInstance().getReference();

    mVelocidad = (TextView) findViewById(R.id.tv_parada6);
    mTiempo_p1 = (TextView) findViewById(R.id.tv_parada1);
    mTiempo_p2 = (TextView) findViewById(R.id.tv_parada2);
    mTiempo_p3 = (TextView) findViewById(R.id.tv_parada3);
    mTiempo_p4 = (TextView) findViewById(R.id.tv_parada4);
    mTiempo_p5 = (TextView) findViewById(R.id.tv_parada5);
    mTiempo_p6 = (TextView) findViewById(R.id.tv_parada6);
```

```
}


@Override
public void onMapReady(GoogleMap googleMap) {

    mMap = googleMap;



    midatabaseReference.child("Conductores").child("AA4MoSWWlHbf2iKzS4dlAGDPWOy2").
        addListenerForSingleValueEvent(new ValueEventListener() {

        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {

            for(Marker marker: actualMarker){

                marker.remove();


            }



            ObtencionDatos ubicaciones = dataSnapshot.getValue(ObtencionDatos.class);
            Double locationLat = ubicaciones.getLatitudd();
            Double locationLng = ubicaciones.getLongitudd();

            LatLng posicionmove = new LatLng(locationLat, locationLng);
            //prueba de datos en logcat
            String daaa = String.valueOf(locationLat);
            Log.e("longitud ",daaa);

            //obteniendo datos posicion bus
            Location pos_bus = new Location("");
            pos_bus.setLatitude(posicionmove.latitude);
```

```
pos_bus.setLongitude(posicionmove.longitude);

//calculo de distncias y movimiento de paradas
//parada 1

//obnteniendo datos posicion distacia
Location loc1 = new Location("");
loc1.setLatitude(Parada1.latitude);
loc1.setLongitude(Parada1.longitude);

//c lculo de distancia
distance_p1 = loc1.distanceTo(pos_bus);
String distaciaaaaa = String.valueOf(distance_p1);
Log.e("distacia ",distaciaaaaa);
//tiempo
tiempo_p1 = (int) (distance_p1/speed);
String vel = String.valueOf(speed);
Log.e("longitud ",vel);
mTiempo_p1.setText("El bus llegar  Cachiloma en "+ tiempo_p1 +" min");

//parada 2

//obnteniendo datos posicion distacia
Location loc_p2 = new Location("");
loc_p2.setLatitude(Parada2.latitude);
loc_p2.setLongitude(Parada2.longitude);

//c lculo de distancia
distance_p2 = loc_p2.distanceTo(pos_bus);

//tiempo
tiempo_p2 = (int) (distance_p2/200);
mTiempo_p2.setText("El bus llegar  a Escuela Luis Napoleon Dillon en "+
    tiempo_p2 +" min");

//parada3
Location loc_p3 = new Location("");
loc_p3.setLatitude(Parada3.latitude);
loc_p3.setLongitude(Parada3.longitude);

//c lculo de distancia
distance_p3 = loc_p3.distanceTo(pos_bus);
```

```
//tiempo
tiempo_p3 = (int) (distance_p3/speed);
mTiempo_p3.setText("El bus llegara a parada de camionetas en "+ tiempo_p3 +"
    min");

//parada4
Location loc_p4 = new Location("");
loc_p4.setLatitude(Parada4.latitude);
loc_p4.setLongitude(Parada4.longitude);

//c lculo de distancia
distance_p4 = loc_p4.distanceTo(pos_bus);

//tiempo
tiempo_p4 = (int) (distance_p4/speed);
mTiempo_p4.setText("El bus llegar  barrio Central Cariacu en "+ tiempo_p4 +"
    min");


//parada5
Location loc_p5 = new Location("");
loc_p5.setLatitude(Parada5.latitude);
loc_p5.setLongitude(Parada5.longitude);

//c lculo de distancia
distance_p5 = loc_p5.distanceTo(pos_bus);

//tiempo
tiempo_p5 = (int) (distance_p5/speed);
mTiempo_p5.setText("El bus llegar  barrio Central Cariacu en "+ tiempo_p5 +"
    min");

//parada4
Location loc_p6 = new Location("");
loc_p6.setLatitude(Parada6.latitude);
loc_p6.setLongitude(Parada6.longitude);

//c lculo de distancia
distance_p6 = loc_p6.distanceTo(pos_bus);

//tiempo
tiempo_p6 = (int) (distance_p6/speed);
```

```
            mTiempo_p6.setText("El bus llegar  barrio Central Cariacu en "+ tiempo_p6 +"
                min");



            MarkerOptions markerOptions = new MarkerOptions()
                    .position(new LatLng(locationLat,locationLng));
            markerOptions.title("hola ");
            markerOptions.snippet("hola ");
            markerOptions.icon(BitmapDescriptorFactory.fromResource(R.drawable.
                ic_bus_round));
            //guadamos marcadores en primer array
            anteriorMarker.add(mMap.addMarker(markerOptions));
            mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(posicionmove,CAMERA_ZOOM)
                );




            /// borramos marcador
            actualMarker.clear();
            //cambiamos de array
            actualMarker.addAll(anteriorMarker);



            //anteriorMarker.clear();
            timer();
            //calculo de tiempo

        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {

        }


});
```

```
    // Ahora que el Mapa estalisto:
    // Agregamos un MArker en laUniversidad Tecnica del norte, Ibarra y movemos la camara.
    setMarker(Posiciones.CayambeRuta, "Universidad Tecnica del Norte", "Distrito: Ibarra")
        ;


    //Dibujamos una Polylinea
    drawPolilyne(Posiciones.prueba_cayambe);
    //Dibujamos una Polygono
    drawPoligono(Posiciones.POLIGONO);


}

//timer
private void timer(){

    new CountDownTimer(1000, 1000) {

        public void onTick(long millisUntilFinished) {
            Log.e("seconds remaining: ", "" + millisUntilFinished / 1000);

        }

        public void onFinish() {
            //Toast.makeText(getActivity(), "puntos actualizados", Toast.LENGTH_SHORT).
                show();
            onMapReady(mMap);

        }
    }.start();

}
```

```
public void setMarker(LatLng position, String titulo, String info) {
    // Agregamos marcadores para indicar sitios de intereses.
    mMap.addMarker(new MarkerOptions()
            .position(position)
            .title(titulo)  //Agrega un titulo al marcador
            .snippet(info)   //Agrega informacion detalle relacionada con el marcador
            .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN)
                ));  //Color del marcador

    //mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(position, CAMERA_ZOOM));
    //PARADAS
    //Parada1 = new LatLng(0.0889989,-78.092222);

    // agrgamos icono
    mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
        drawable.ic_estacion))
            .anchor(0.0f,1.0f).position(Parada1).title("Cachiloma-Barrio Oriente").snippet
                (String.valueOf("parada 1")));
    //Parada2 = new LatLng(0.934533,-78.078876);

    // agrgamos icono
    mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
        drawable.ic_estacion))
            .anchor(0.0f,1.0f).position(Parada2).title("Escula Luis Napoleon Dillon"));

    mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
        drawable.ic_estacion))
            .anchor(0.0f,1.0f).position(Parada3).title("Parada de camionetas - Cariacu"));

    mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
        drawable.ic_estacion))
            .anchor(0.0f,1.0f).position(Parada4).title("Tanque Porgreso -Barrio Central-
                Cariacu"));

    mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
        drawable.ic_estacion))
            .anchor(0.0f,1.0f).position(Parada5).title("Barrio San Pedro Cariacu"));

    mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
        drawable.ic_estacion))
            .anchor(0.0f,1.0f).position(Parada6).title("Hacienda los Alamos Cariacu"));
```

```
        mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
            drawable.ic_estacion))
                .anchor(0.0f,1.0f).position(Parada7).title("Hacienda la Jesus- Cariacu"));

        mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
            drawable.ic_estacion))
                .anchor(0.0f,1.0f).position(Parada8).title("Camino al condor- Paquiestancia"))
                    ;

        mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
            drawable.ic_estacion))
                .anchor(0.0f,1.0f).position(Parada9).title("Escuela Paquiestancia"));

        mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
            drawable.ic_estacion))
                .anchor(0.0f,1.0f).position(Parada10).title("Tanque de leche- Paquiestancia"))
                    ;

        mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
            drawable.ic_estacion))
                .anchor(0.0f,1.0f).position(Parada11).title("Santa fe- Paquiestancia"));

        mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
            drawable.ic_estacion))
                .anchor(0.0f,1.0f).position(Parada12).title("Parque Puluvi- Ayora"));

        mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
            drawable.ic_estacion))
                .anchor(0.0f,1.0f).position(Parada13).title("Colegio Nelson Torres"));

        mMap.addMarker(new MarkerOptions().icon(BitmapDescriptorFactory.fromResource(R.
            drawable.ic_estacion))
                .anchor(0.0f,1.0f).position(Parada14).title("Rio Blanco"));



}
```

```
    public void drawPolilyne(PolylineOptions options){
        mMap.addPolyline(options);
    }

    public void drawPoligono(PolygonOptions opts){
        mMap.addPolygon(opts);
    }

}
```