TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO SUPERIOR DE HUAUCHINANGO

SUBDIRECCIÓN DE INVESTIGACIÓN Y POSGRADO

MAESTRÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

“Aplicación Android para el rastreo del vehículo recolector de basura en Huauchinango Puebla”

TESIS

Que para obtener el grado de

MAESTRO EN TECNOLOGÍAS DE LA INFORMACIÓN

P r e s e n t a:

Lic. Jorge Alfredo Barrón Castillo

Director: **Mtro. Aldo Hernández Luna**

Codirector: **Dr. Jacinto Torres Jiménez**

Huauchinango, Pue., Noviembre del 2023.

ACTA DE REVISIÓN DE TESIS

En la ciudad de Huauchinango, Puebla, siendo las **09:00** horas del día **06-noviembre-2023** se reunieron los miembros del Comité Sinodal designado por el H. Consejo de Posgrado de la Maestría en Tecnologías de la Información del Instituto Tecnológico Superior de Huauchinango, para examinar y evaluar la tesis de grado titulada:

"APLICACIÓN ANDROID PARA EL RASTREO DEL VEHÍCULO RECOLECTOR DE BASURA EN HUAUCHINANGO PUEBLA"

Presentada por el (la) alumno (a):

JORGE ALFREDO BARRÓN CASTILLO, con número de control M21315008

Aspirante al grado de:

MAESTRO(A) EN TECNOLOGÍAS DE LA INFORMACIÓN

Después de intercambiar opiniones los miembros de la comisión manifestaron la aprobación e impresión final de la tesis, en virtud que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

Presidente

Aldo Hernández Luna
Maestro en Ingeniería
Cédula profesional: 11606723

Secretario

Jacinto Torres Jiménez
Doctor en Ciencias en Ingeniería Eléctrica
Cédula profesional: 8112868

Vocal

Carmen Jeannette Sampayo Rodriguez
Maestro en Sistemas Computacionales
Cédula profesional: 8591146

Suplente

Hugo Hernández Cabrera
Maestro en Sistemas Computacionales
Cédula profesional: 8157886

Sello de la institución

INSTITUTO TECNOLOGICO SUPERIOR DE HUAUCHINANGO MAESTRÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Huauchinango, Puebla, 23/noviembre/2023
OFICIO No.: SPeI/126/2023

ASUNTO: AUTORIZACIÓN DE OMPRESIÓN DEFINITIVA DE TESIS

ING. JORGE ALFREDO BARRÓN CASTILLO
ESTUDIANTE DE LA MAESTRÍA EN TECNOLOGÍAS
DE LA INFORMACIÓN
PRESENTE.

Por este conducto tengo el agrado de comunicarle que el jurado designado para que Usted obtenga el grado de **MAESTRO EN TECNOLOGÍAS DE LA INFORMACIÓN**, ha informado a esta Subdirección de Posgrado e Investigación, su aceptación para la impresión Definitiva de su trabajo de tesis, en el cual lleva por título:

"APLICACIÓN ANDROID PARA EL RASTREO DEL VEHÍCULO RECOLECTOR DE BASURA EN HUAUCHINANGO PUEBLA"

Por lo anterior se autoriza la impresión de su trabajo, esperando que el logro del mismo sea acorde con sus aspiraciones profesionales.

ATENTAMENTE
Excelencia en Educación Tecnológica.

C.FERNANDO GONZÁLEZ CRUZ
SUBDIRECTOR DE POSGRADO E INVESTIGACIÓN

INSTITUTO TECNOLOGICO SUPERIOR DE HUAUCHINANGO
SUBDIRECCION DE POSGRADO E INVESTIGACION

c.c. p. C. Carmen Jeannette Sampaye Rodríguez. Coordinación de la Maestría en Tecnologías de la Información
c.c. p. C. Aldo Hernández Luna. Presidencia del Consejo de Posgrado de la Maestría en Tecnologías de la Información
c.c.p. Archivo

INSTITUTO TECNOLÓGICO SUPERIOR DE HUAUCHINANGO

MAESTRÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

“Aplicación Android para el rastreo del vehículo recolector de basura en Huauchinango Puebla”

TESIS

Que para obtener el grado de

MAESTRO EN TECNOLOGÍAS DE LA INFORMACIÓN

P r e s e n t a:

Lic. Jorge Alfredo Barrón Castillo

Director: **Mtro. Aldo Hernández Luna**

Codirector: **Dr. Jacinto Torres Jiménez**

Huauchinango, Pue., Noviembre del 2023.

Resumen

El presente proyecto de investigación está orientado en el desarrollo de una aplicación móvil que permite ubicar a los camiones recolectores de basura del municipio de Huauchinango, Puebla, “Rastreo del servicio recolector”, para que los usuarios puedan deshacerse de su basura solo cuando el camión se encuentre en ruta y cercanos a su ubicación.

En el municipio de Huauchinango, se tiene la problemática de acumulación de basura en la vía pública, esto es originado por el hecho de que la gente no tiene certeza de cuando pasa el camión de la basura o que pase en horarios donde la gente no se encuentra en sus hogares, generando la acumulación de basura en grandes cantidades, lo que origina focos de infección, mala imagen a la ciudad y propagación de plagas de animales por las calles.

El desarrollo de la aplicación inicia con el análisis de la tecnología para la ubicación de objetos geográficamente, lo que llevo a seleccionar un dispositivo móvil con tecnología GPS ya que brinda mayor exactitud. Se realizaron las configuraciones necesarias en los servicios de google para poder usar las API de google maps, así como la configuración de los servicios de Firebase, Que es la plataforma que se encargará de gestionar la base de datos y las notificaciones push. Después de las configuraciones previas se continúa con el diseño y desarrollo de las pantallas que tendrá la aplicación, posteriormente se realizan pruebas de rendimiento de la aplicación. Por último, se realiza una prueba de campo con una prueba piloto de 19 usuarios designados por el departamento de limpia de Huauchinango, Con la finalidad de poder medir la experiencia de usuario.

La aplicación busca generar un alto grado de aceptación en los usuarios, así como del personal que administra el sistema de limpia del H. Ayuntamiento de Huauchinango ya que les permitiría mejorar la logística y monitoreo de los camiones de recolección de basura.

Abstract

This research project is focused on the development of a mobile application that allows locating the garbage collection trucks of the municipality of Huauchinango, Puebla, "Tracking the collection service", so that users can get rid of their garbage only when the truck is en route and close to your location.

In the municipality of Huauchinango, there is a problem of accumulation of garbage on public roads, this is caused by the fact that people are not sure when the garbage truck passes or that it passes at times when people do not found in their homes, generating the accumulation of garbage in large quantities, which causes sources of infection, a bad image of the city and the spread of animal pests through the streets.

The development of the application begins with the analysis of the technology for geographically locating objects, which led to selecting a mobile device with GPS technology since it provides greater accuracy. The necessary configurations were made in Google services to be able to use the Google Maps API, as well as the configuration of Firebase services, which is the platform that will be in charge of managing the database and push notifications. After the previous configurations, the design and development of the screens that the application will have continues, then performance tests of the application are carried out. Finally, a field test is carried out with a pilot test of 19 users designated by the Huauchinango cleaning department, in order to measure the user experience.

The application seeks to generate a high degree of acceptance among users, as well as the staff that manages the cleaning system of the Huauchinango City Council since it would allow them to improve the logistics and monitoring of garbage collection trucks.

Dedicatoria y lema

El presente trabajo está dedicado a mi familia que me han apoyado a lo largo de toda mi vida, a mi madre que todo su esfuerzo se ve reflejado en la persona que soy ahora, a mi padre y hermana por demostrarme que el cielo es el límite, a mi hermano por enseñarme que ninguna idea es absurda y a mi esposa e hijo por apoyarme durante este camino, dándome los ánimos para mejorar cada día.

La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica.

Aristóteles.

Agradecimientos

Gracias al CONAHCYT por el apoyo con la beca de posgrado para la maestría en tecnologías de la información.

Gracias al programa de posgrado, maestría en tecnologías de la información.

Gracias a los cuerpos académicos de tecnología aplicada y computo inteligente del instituto tecnológico superior de Huauchinango por el apoyo y las facilidades para el desarrollo de este trabajo

Gracias a mis profesores que sirvieron de guías en el proceso de desarrollo de este trabajo.

Gracias a mi asesor que ha sido un buen mentor y aliado durante toda la maestría.

Gracias a mi familia, esposa e hijo nuevamente por darme los ánimos y apoyo durante este proceso.

Índice General.

Índice de figuras

Capítulo I: INTRODUCCIÓN

Capítulo II: MARCO TEÓRICO

Capítulo III: MARCO METODOLÓGICO

Capítulo IV: RESULTADOS

Índice de tablas

Capítulo III: MARCO METODOLÓGICO

Lista de acrónimos

IDE.- Entorno de desarrollo integrado (IDE) es un sistema de software para el diseño de aplicaciones que combina herramientas del desarrollador comunes en una sola interfaz gráfica de usuario (GUI).

JAVA.- Plataforma informática de lenguaje de programación creada por Sun Microsystems.

Firebase.- Plataforma para el desarrollo de aplicaciones web y aplicaciones móviles ubicada en la nube, adquirida por Google en 2014.

DB.- Abreviatura de Base de Datos.

APIs.- Interfaz de programación de aplicaciones (Application Programming interface) es una pieza de código que permite a diferentes aplicaciones comunicarse entre sí y compartir información y funcionalidades, es un intermedio entre dos sistemas que permite que una aplicación se comunique con otra y pida datos o acciones específicas.

2G.- Segunda generación de telefonía móvil, se caracteriza por ser digital y disminuir el tamaño para mejorar la portabilidad, no es un estándar o protocolo. Es una forma de marcar el cambio de protocolos de telefonía móvil analógica a digital.

2.5G.- Término para describir la evolución de las redes de datos que surgieron después del 2G pero antes del 3G, tenía casi la misma tecnología que las redes 2G pero tenía dominios de conmutación de paquetes que brindaban fiabilidad y capacidad adicionales.

3G.- Abreviación de tercera generación de transmisión de voz y datos a través de telefonía móvil mediante UMTS, proporcionan la posibilidad de transferir voz y datos no-voz como la descarga de programas, intercambio de correos electrónicos y mensajería instantánea.

4G.- Sigla utilizada para referirse a la cuarta generación de tecnologías de telefonía móvil, está basada completamente en el protocolo IP, siendo un sistema de red que se alcanza gracias a la convergencia entre las redes cableadas e inalámbricas. La principal diferencia entra las generaciones predecesoras es la capacidad de proveer velocidades de acceso mayores a 100 MBPS en movimiento y 1 GB en reposo.

GPRS.- Servicio general de paquetes vía radio (General Packet Radio Service) fue lanzado en 2001 por la red del sistema global de comunicaciones móviles (GSM) para proporcionar acceso a internet a los usuarios de dispositivos móviles, pueden utilizarse para transmitir datos telemáticos a servidores centralizados cuando no se dispone de redes móviles más avanzadas.

UTMS.- El sistema universal de telecomunicaciones móviles (Universal Mobile Telecommunications System) es una de las tecnologías es una de las tecnologías más usadas por los móviles de tercera generación sucesora del GPRS, sus tres grandes características son: multimedia, mayor velocidad de acceso a internet y transmisión de calidad de voz equiparable a las redes fijas.

IP.- Es una dirección única que identifica a un dispositivo en el internet o en una red local.

MBPS.- Siglas que significan megabytes por segundo, es una cantidad de transmisión de datos equivalente a 1024 kilobytes por segundo.

GB.- Cantidad de transmisión de datos equivalente a 1024 megabytes.

GPS.- Sistema de posicionamiento global, es un sistema que permite a un dispositivo receptor localizar su propia posición sobre la tierra con una precisión de hasta centímetros.

GSM.- Sistema global de comunicaciones móviles 2g.

CAPÍTULO 1
INTRODUCCIÓN

1.1 Introducción.

En la actualidad, los equipos de rastreo han tomado gran fuerza en diferentes ambientes, como son la industria automotriz, servicios de transporte, medicina, veterinaria, etc. Sin embargo, donde ha tenido mayor impulso es en el servicio de transporte, donde se requiere saber la ubicación geográfica de sus elementos como son: vehículos, paquetes, equipos delicados y costosos, con el propósito de tener seguridad, seguimiento y logística, como también el saber los recorridos realizados por su flota de vehículos y tener un control de gastos por optimización de rutas.

En cuestión de seguridad, el rastreo GPS también puede ser un factor importante, para la policía, bomberos y ambulancias para actuar más rápidamente ante una emergencia al ubicar sus patrullas en tiempo real y sitio, optimizando su servicio a la ciudadanía.

Otra área de oportunidad donde se pueden utilizar los equipos de rastreo es en el servicio de limpieza municipal, es por eso que este proyecto está enfocado a el rastreo del vehículo recolector de basura, Que permita ayudar a la población de Huauchinango Puebla, saber el momento exacto cuando poder sacar su basura, utilizando las tecnologías de rastreo satelital y añadiendo funciones que pueden enfocar la aplicación al servicio de limpia con la finalidad de evitar la problemática de la acumulación excesiva de basura en diferentes puntos de la ciudad.

Los beneficios que nos puede aportar esta aplicación son:

- Evitar focos de infección.
- Mejorar imagen para la ciudad.
- Disminuir acumulación de basura en las calles.
- Mejorar tránsito por las calles.
- Reducir el número de animales en las calles como perros o ratas.

1.2 Estado del arte.

Existen diferentes investigaciones que hablan sobre el uso de tecnologías de rastreo satelital utilizadas para diferentes fines, las cuales sirven como una base para el presente trabajo de investigación. En la referencia [1] "SISTEMA DE LOCALIZACIÓN MONITOREO Y CONTROL VEHICULAR BASADO EN LOS PROTOCOLOS GPS/GSM/GPRS", se consolida la implementación de un Sistema de Administración de flota de transporte en empresas de consumo masivo , donde se determina que la logística empresarial mejora considerablemente y los costos de transporte se reducen hasta en un 15%, además de tener el control total del vehículo generando reportes estadísticos de los recorridos realizados; el sistema que proponen los autores tiene características avanzadas para redefinir las rutas ya establecidas considerando factores de tiempo y costos. (Astudillo L, Delgado E, 2012).

En la tesis "EVALUACION Y DISEÑO DE UN SISTEMA DE RASTREO SATELITAL PARA EL MONITOREO Y CONTROL DE LAS RUTAS TERRESTRES ASIGNADAS DIARIAMENTE EN TIEMPO REAL" [2] desarrollada por Liliana Grimaneza Guzmán Acán, comenta que el uso de los sistemas de posicionamiento GPS que utilizó en su investigación, consta de 24 satélites artificiales que proporcionan información para el posicionamiento las 24 horas del día sin importar las condiciones del tiempo (NAVSTAR). Utilizó APIS de GOOGLE para poder implementar google maps en páginas web usando dispositivos de hardware con comunicación M2M para que la comunicación sea más eficiente.

Bryan Joshua Murillo Coello en su proyecto de investigación "DISEÑO E IMPLEMENTACION DE PROTOTITPO PARA EL ENCENDIDO Y RASTREO SATELITAL DE UN VEHICULO" [3] de la universidad Técnica estatal de Quevedo, implementó de tecnologías GPS para poder conocer la ubicación de un objeto en tiempo real, así mismo utilizó tecnologías GSM/GPRS que es la que permitió el empleo de mensajes de texto e internet para el envío de notificaciones y demás utilidades de servicio para los dispositivos móviles. Además, utilizó un módulo raspbery PI 2 mediante comunicación remota a través de tecnologías GSM para conocer las coordenadas del vehículo.

Por otro lado, Anthony Limber Moran Cabezas, en su tesis de posgrado "ANALISIS E IMPLEMENTACION DE UN SISTEMA DE RASTREO SATELITAL APLICADO A MASCOTAS MEDIANTE SOFTWARE LIBRE CON TECNOLOGIAS GPS Y GSM" [4], expone el uso de tecnologías GPS para el rastreo satelital por su precisión ante sus 24 satélites orbitando el planeta, para poder hacer rastreos las 24 horas. En la investigación presenta el uso de tecnologías de comunicación móvil GSM mediante compañías telefónicas. Cabe mencionar que explica el uso de vallas virtuales en el sistema para poder delimitar zonas y así poder enviar una alerta a la aplicación móvil para que el usuario sepa que su mascota ha salido de cierta zona en el mapa.

El ingeniero Tapuy Cerda Welington Leonardo en su tesis "DISEÑO DE UN PROTOTIPO ELECTRONICO PARA EL ENCENDIDO DEL VEHICULO RENAULT LOGAN, MEDIANTE TECNOLOGIA NFC Y RASTREO SATELITAL GPS" [5], utilizó tecnologías GPS y GSM para el rastreo y comunicación entre el dispositivo móvil y el hardware al que se quiere comunicar de manera remota. En este caso implementó Arduino para poder comunicar al vehículo. A través de un módulo SIM808 el cual permite conectar la placa Arduino a la red GSM y así mismo a la tecnología GPS.

Gracias a la revisión bibliográfica se ha logrado determinar cuáles son las mejores tecnologías para dar seguimiento satelital, ya que se han tomado como referencia los dispositivos rastreadores, así como el software utilizado. Con el objeto de saber cuáles son las limitantes que tienen cada uno de estos dispositivos, precisión en el rastreo, tiempo de repuesta, costo y escalabilidad.

1.3 Objetivo general.

Desarrollar una aplicación móvil para dar seguimiento a los camiones recolectores de basura del servicio de limpia del municipio de Huauchinango, utilizando el sistema de posicionamiento global de dispositivos móviles, para que los ciudadanos tengan una herramienta que les indique cuando puedan sacar su basura; la aplicación permitirá ubicar si el camión se encuentra en ruta y cercano a sus ubicaciones.

1.4 Objetivos específicos.

Para lograr el objetivo general se requiere la realización y ejecución de los siguientes objetivos específicos.

- Dar seguimiento en tiempo real de los carros recolectores activos.
- Notificar al usuario cuando este cerca el recolector.
- Notificar al usuario cuando haya algún imprevisto y no pase el vehículo recolector de basura.
- Mostrar al usuario el calendario de recolección.
- Mostrar al conductor solo su ubicación en tiempo real.
- Mostrar en la misma aplicación funciones diferentes para usuario y conductor
- Mostrar la información general del usuario y conductor en sus respectivos perfiles.
- Guardar la información de usuario para no estar haciendo login cada que se inicie la aplicación.
- Restringir el registro de conductores mediante claves.

1.5 Justificación.

Durante el desarrollo de la presente investigación, se trabajó en conjunto con el departamento de ecología del municipio de Huauchinango Puebla. Para dar una propuesta de solución a la problemática de las montoneras de basura en la ciudad, para esto se hicieron recorridos junto con el personal de limpia para saber cuáles son los problemas que enfrentan día con día durante su recorrido, para así poder determinar; 1) por qué no existe un horario fijo de recolección y 2) por qué algunas veces no llega el servicio de recolección a las colonias de la ciudad.

Después de culminar los recorridos en el servicio de limpia, se encontraron diferentes problemas que hacen ineficiente la recolección de basura, a continuación, se describen cada uno ellos:

En primer lugar, se tiene la falta de cultura vial, los vehículos recolectores son demasiado grandes para transitar libremente sobre diversas las calles de la ciudad, aunando a lo anterior la ciudadanía no tiene conciencia sobre esto, estacionando vehículos grandes sobre avenidas pequeñas limitando el paso del recolector, véase en la figura 1.1.

Figura 1. 1 Obstrucción de camino

El segundo problema es que no se respetan los límites de estacionamiento en las calles o en algunos casos no hay señalamiento, por lo tanto, los ciudadanos tienen que esperar a que se termine la recolección del punto de basura para poder avanzar, generando tráfico durante el recorrido (figura 1.2).

Figura 1. 2 Límites de estacionamiento

El tercer problema, las banquetas se encuentran obstruidas por arena, material o vehículos mal estacionados, esto limita la circulación de los carriles de la calle, como se muestra en la figura 1.3.

Figura 1. 3 Obstrucción de banquetas

Estos son algunos de los problemas viales encontrados durante los recorridos, siendo unos de los factores para no tener un horario fijo de recolección en las colonias.

La cuarta problemática es que algunas de las personas que se dedican a la recolección de reciclables destruyen las bolsas para poder sacar el material de reciclaje. Esto hace que el

personal de limpia levante con pala todos los residuos (figura 1.4), generando un atraso en el itinerario.

Figura 1. 4 Destrucción de bolsas de basura

Cabe mencionar que el levantar la basura con palas puede convertirse en un riesgo para los trabajadores, ya que se puede contraer alguna infección por los residuos de botellas de vidrio, jeringas etc. Considerando de que no se tiene una cultura para separar los residuos.

Figura 1. 5 Eventos no contemplados

El quinto problema son los eventos no contemplados, un ejemplo es cuando no se tienen unidades suficientes por motivos de alguna falla mecánica (figura 1.5), para resolver el

problema se utilizan camiones de otras rutas para poder terminar el recorrido, sin embargo, esto genera que el camión ya tenga algo de carga de basura para la siguiente ruta, esto hace que no se termine la ruta ya que se alcanza el límite de recolección antes de completar su recorrido.

Figura 1. 6 Desahogo de unidades

Por otra parte, no es posible vaciar los vehículos recolectores de la ruta hasta que esté llena (figura 1.6), ya que no se cuenta con un relleno sanitario y se tiene que transportar la basura hasta el relleno de la ciudad de Tulancingo Hidalgo (figura 1.7).

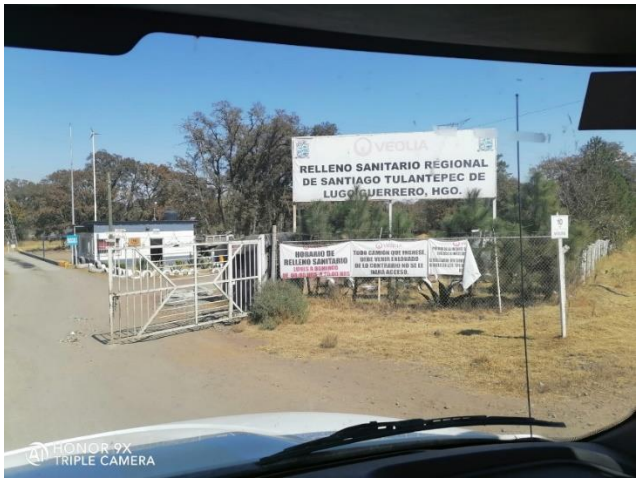


Figura 1. 7 Relleno sanitario

Por último, se tiene el problema sobre las campañas de recolección de residuos especiales, uno de muchos casos se presenta después de la temporada navideña la gente quiere tirar sus

árboles navideños, pero estos no pueden ser levantados junto con toda la basura ya que dañan el sistema de compresión del camión y es por eso que deben ser recolectados en vehículos diferentes.

Aunque se avisa en redes sociales sobre las fechas de recolección de estos residuos, esta información no llega a todas las personas y por ende sacan estos residuos a las calles, aunque el camión no se los va a recolectar (figura 1.8).

Figura 1. 8 Desinformación ciudadana

La presente investigación se enfocará en desarrollar una aplicación móvil que permita a los ciudadanos de Huauchinango saber la posición real de los carros que recogen la basura, con la finalidad de que sean notificados cuando no vaya a haber servicio de limpia o ya se encuentre cerca de su zona el carro de recolección de basura.

Figura 1. 9 Organización ciudadana

Para fortalecer la idea se pueden tomar de ejemplo las colonias que están organizadas y solo sacan su basura de manera ordenada al momento de la llegada del carro de la basura. Para lograr esto, el presidente, comité, o encargado de la colonia es avisado por el personal de limpia cuando ellos ya van en dirección a su colonia, y los colonos se esperan hasta ser avisados para sacar su basura como se puede ver en la figura 1.9.

Esto puede ser un gran indicador de la viabilidad del proyecto, ya que de una manera arcaica se está llevando en práctica la idea general de la aplicación Android, que se propone la presente investigación de tesis, es importante mencionar que la aplicación podrá avisar sobre las campañas de recolección especiales o el incumplimiento del calendario de recolección.

La aplicación que se propone utilizará una base de datos para poder almacenar las ubicaciones de todos los conductores que se encuentren activos, los datos generales de usuarios y conductores, y la calendarización de las rutas de la semana.

La base de datos será gestionada por Firebase, que permitirá gestionar las notificaciones push para dar los avisos necesarios a los ciudadanos.

El rastreo satelital será a través de las APIS de google, utilizando el teléfono celular (Android) como GPS logrando el mejor tiempo de respuesta y la mayor precisión de ∓ 5 metros.

La aplicación puede ayudar a resolver los problemas detectados en el recorrido de la ruta, ya que esta les avisará a los ciudadanos cuando: a) no vaya a haber servicio de limpia b) sea momento de sacar su basura, y c) haya campañas especiales de recolección de residuos.

1.6 Motivación para elaborar la investigación.

En México, específicamente en el estado de Puebla, existen varios municipios con la problemática de no tener certeza de los servicios de recolección de basura municipal, haciendo que se acumule demasiada basura en el hogar debido a que los horarios laborales de los ciudadanos coinciden con los horarios de las rutas de los vehículos recolectores.

La motivación para realizar este proyecto no solo es para desarrollarme profesionalmente expandiendo el conocimiento a nuevas tecnologías que se encuentran en tendencia, también

como padre de familia al adquirir un mayor grado de estudios y poner un buen ejemplo a mi hijo, así también como ciudadano para ayudar a mi ciudad aportando para que sea una ciudad más prospera y dejar un mejor entorno a mi familia con conocimientos de desarrollo en aplicaciones móviles, para esto se desarrollará una aplicación la cual nos pueda ayudar como ciudadanos a mantener limpia la ciudad, así como evitar esos montones de basura los cuales atraen enfermedades, malos olores en las calles y obstrucción de banquetas. Aunque no toda la población cuenta con la posibilidad de un equipo celular, por lo menos el 80% si podrá saber cuál es el momento adecuado para poder sacarla o incluso podría transportarla a otro punto de recolección donde el carro de basura si pasará.

1.7 Organización de tesis.

El proyecto de investigación presentado en este documento consta de 4 capítulos los cuales describen de forma muy puntual cada una de las etapas del proyecto:

- Capítulo I: Introducción, se dará a conocer todos los detalles del proyecto, estado del arte, justificación, hipótesis en general, objetivos del proyecto y motivación para elaborar la investigación.
- Capitulo II: Marco teórico, se muestra el análisis y recopilación teórico práctico del proyecto abordando los temas que se relacionan con el uso de tecnologías de rastreo satelital, bases de datos en la nube, notificaciones push y la importancia de evitar puntos de acumulación de basura en la ciudad, proyectos e investigaciones y/o propuestas que han dado soluciones a problemas similares.
- Capitulo III: Marco metodológico, aquí se lleva a cabo una serie de métodos y técnicas que se utilizan para el desarrollo del proyecto iniciando desde su análisis, diseño de la aplicación, desarrollo de lógica del programa, implementación y puesta a punto del proyecto.
- Capitulo IV: Resultados, en esta parte se dará a conocer el proyecto final, los resultados de la ejecución del mismo, opiniones de un grupo de ciudadanos que utilizarán la aplicación para encontrar sus áreas de mejora.

- Capítulo V: Conclusiones y trabajos futuros, en este capítulo se mostrarán las oportunidades del proyecto y los trabajos futuros para enriquecer el proyecto en su próxima versión.

1.8 Publicaciones que sustenten el trabajo de tesis.

Como parte del desarrollo del proyecto se presenta un artículo en la séptima edición del congreso internacional interdiciplinado de energías renovables, mantenimiento industrial, mecatrónica e informática (CIERMMI) 2022, titulada "**Aplicación Android para el rastreo del vehículo recolector de basura en Huauchinango Puebla**" mismo que se publica en la revista indexada de Tecnologias de la Información de EORFAN, Ver anexo 1 y 2. [6]

También se presenta el artículo "**Configuración Firebase para la integración de un sistema de Geolocalización**" para el segundo congreso internacional de innovación en ingeniería industrial, gestión y computación en la era digital (INGECO) 2022, el cual se publica en la revista indexada FEGLININ, Ver anexo 3 y 4. [7]

CAPÍTULO 2
MARCO TEÓRICO

2.1 GPS.

El **Sistema de Posicionamiento Global** (**GPS**; en inglés, **Global Positioning System**), originalmente **Navstar GPS**, es un sistema que permite localizar cualquier objeto (una persona, un vehículo, etc.) sobre la Tierra con una precisión de hasta centímetros (si se utiliza GPS diferencial), aunque lo común son unos pocos metros. El sistema fue desarrollado, instalado y empleado por el departamento de defensa de Estados Unidos, y actualmente es propiedad de la Fuerza especial de los estados unidos. Para determinar su posición, un usuario utiliza cuatro o más satélites y utiliza la trilateración. [8]

Cuando se desea determinar la posición tridimensional, el receptor que se utiliza para ello localiza automáticamente como mínimo cuatro satélites de la red, de los que recibe unas señales indicando la identificación y hora del reloj de cada uno de ellos, además de información sobre la constelación de satélites que forman parte del sistema. Con base en estas señales, el aparato sincroniza su propio reloj con el tiempo del sistema GPS y calcula el tiempo que tardan en llegar las señales al equipo, y de tal modo mide la distancia al satélite. Mediante el método de trilateración inversa, computa su propia posición. Se calcula también con una gran exactitud en el tiempo, basado en los relojes atómicos a bordo cada uno de los satélites y en el segmento terreno de GPS (figura 2.1).

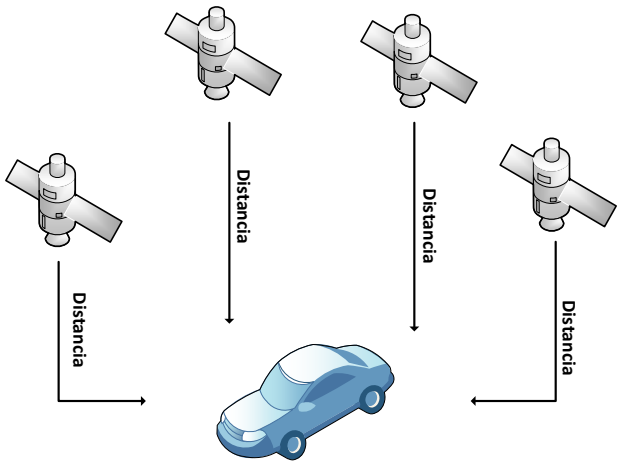


Figura 2. 1 Funcionamiento de la ubicación [9]

2.2 GSM.

Las siglas **GSM** vienen, como siempre, de las palabras anglosajonas Global System for Mobile comunications. Como su propio nombre indica, pues, el GSM no es más que un **estándar de comunicación para la telefonía móvil**, implementado mediante la combinación de satélites y antenas terrestres. A los móviles que usan la tecnología GSM también se les conoce por móviles **2G** o de **segunda generación (figura 2.2)**.

Aunque su principal función es como hemos dicho la telefonía, del mismo modo que antiguamente se podía utilizar la línea telefónica para el modem, también el GSM permite la transmisión de datos por medio de sus canales, siempre y cuando estos se hallen libres. Es un sistema digital, y al ser un estándar usado mundialmente, permite su uso en cualquier lugar con cobertura, incluso en ámbitos internacionales (el llamado roaming o itinerancia).
En un principio GSM venía del francés Groupe Spécial Mobile, puesto que se desarrolló en Europa, si bien se extendió muy rápidamente en el resto del mundo y como hemos dicho, **actualmente es el sistema básico para todas las comunicaciones móviles**, y aún el más usado, aunque debido a sus limitaciones técnicas, está siendo rápidamente reemplazado en los países avanzados por los nuevos sistemas **3G** y **4G**, que usan el estándar UTMS, de mayor rapidez y prestaciones.

El GSM apenas permitía una velocidad de descarga de datos de 100 KBPS, se desconectaba al perder la cobertura, tarificaba por segundos y no por volumen de datos..., características todas ellas muy insuficientes para las aplicaciones demandadas por el público, como el streaming de audio y video, la visualización de contenidos web o videoconferencias... Otro estándar muy conocido y que convive igualmente con los anteriores es el GPRS, que no es más que una optimización del GSM para su uso con datos. Este estándar trabaja con la base del GSM, por lo que suele llamarse tecnología **2.5G.**

Sin embargo, estos nuevos sistemas no han desbancado del todo las antiguas redes GSM, sino que conviven con ellas. La mayor parte de los operadores y móviles permiten el uso dual de estas redes, de modo que, si no hay cobertura en un lugar de 3G, pueden usar la red 2G (GSM) sin ningún problema. Esto se debe a que las infraestructuras 3G y 4G se han realizado sobre las ya existentes 2G, siguiendo en funcionamiento estas últimas. En general podemos

decir que un móvil 4g puede funcionar con 3g y con 2g también, siendo compatible con las redes más antiguas [10].

Por lo demás, decir sobre el GSM que realmente supuso un salto definitivo en las plataformas móviles en el mundo. Las personas ya no sólo podían estar comunicadas desde cualquier lugar, sino que no necesitaban un terminal fijo para acceder a cualquier información existente en la red. Las aplicaciones prácticas de todo esto han sido enormes y aún continuarán sorprendiéndonos durante unas cuantas décadas, aunque el nombre de GSM vaya cayendo cada vez más en desuso.

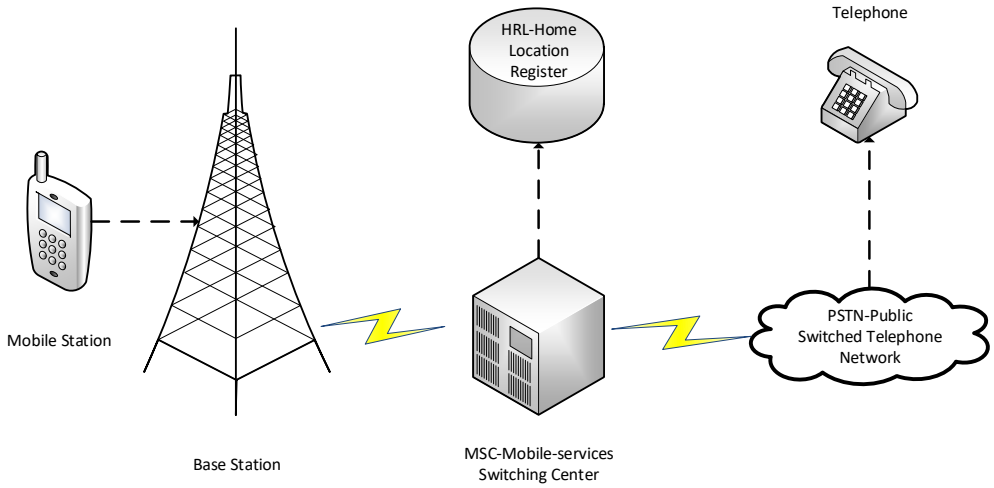


Figura 2. 2 Funcionamiento GSM [11]

2.3 GPRS.

Las siglas **GPRS** son hoy en día muy conocidas por todos aquellos usuarios de servicios de telefonía móvil. Vienen de las palabras inglesas General Packet Radio Service (en castellano Servicio General de Paquetes vía Radio). En su día (años 80) fueron una gran novedad, y aunque hoy ya han sido substituidos (o al menos lo están siendo), por los sistemas **3G y 4G**, todavía son de gran uso en zonas en donde la cobertura de tercera y cuarta generación no es aún completa. El GPRS se basa en el sistema **GSM** de **transmisión de voz**, que fue de por si una revolución mundial, al permitir comunicarse vía satélite, sin necesidad

de cables ni conexión física a dos terminales móviles (el GSM fue diseñado para la llamada segunda generación de móviles).

Usando pues la base del GSM, del mismo modo que se usaban los módems antiguos conectados a la línea telefónica, nació el **GPRS, sistema que permitía mandar y recibir paquetes de datos usando la red de telefonía por satélite.** Internet ya no sólo estaba en casa, en la oficina o al lado de un ordenador con conexión por cable, sino que cabía en el bolsillo. El gesto, ahora natural, de sacar el celular para consultar una página web, o el correo, fue posible gracias a esta tecnología pionera. La gran diferencia entre GSM y GPRS era que la primera estaba orientada a la transmisión de audio y la segunda a la de datos, y además mediante la tarjeta SIM de los celulares permitía asignar una **IP** y por tanto integrar al móvil como un dispositivo más dentro de Internet, con su identificación propia. Es por ello que los sistemas GPRS se llamaron también de **generación 2.5**, al ser una evolución de los GSM tradicionales, y servirían de puente entre el GSM y el UTMS.

Como todos los grandes avances, sin embargo, el GPRS adolecía de muchas carencias. No existía la posibilidad de navegar en itinerancia (movimiento), pues había que conectarse de nuevo después de cada desconexión de red, y la velocidad de transmisión era ridícula, de apenas unos 56 kbps. Por ello en pocos años se vio desplazado por los sistemas de UTMS**,** la llamada tercera generación, que permite conexiones mucho más rápidas, transmisión en itinerancia etc., e incluso estos sistemas 3g ya están quedando obsoletos frente a la llamada cuarta generación (4g).

Pero volviendo al sistema GPRS, digamos que, a pesar de sus defectos, su triunfo vino dado por el hecho de permitir una transmisión de datos muy eficiente teniendo en cuenta la infraestructura de la época. Anteriormente cada vez que se enviaban datos, estos constaban como una llamada, debiendo pagar por tiempo conectado, se transmitiese o no información (siendo como eran hace años las llamadas a móviles bastante caras, el precio podía hacerse prohibitivo para un usuario medio). El GPRS permitía pagar sólo por la información enviada/recibida. Se basaba en mandar la información en pequeños paquetes, que no se enviaban todos juntos, sino cuando las redes estaban libres, aprovechando los espacios huecos de estas. Si la red estaba muy sobrecargada, evidentemente, la transmisión podía demorarse bastante, pero para usos básicos (emails, mensajería instantánea, consulta de

noticias...) la velocidad era más que adecuada, y permitía estar conectado con la web desde cualquier lugar en donde hubiese cobertura, algo realmente sorprendente para aquel entonces (y no hablamos de hace siglos, aunque lo parezca, sino de hace unos 20-30 años). También permitía enviar mensajes multimedia e incluso transmitir imágenes, si bien la velocidad de transmisión no era todavía adecuada para videoconferencias, el esquema de funcionamiento del GPRS se puede observar en la figura 2.3.

Además, tenían una ventaja tremenda para los operadores de telefonía móvil. Actualizar las antenas para dar servicio a GPRS era algo poco costoso, y además los nuevos elementos valían también para la tercera generación. Los costes eran por tanto fácilmente amortizables, y por si esto fuese poco, al utilizar las redes cuando estas no estaban ocupadas, permitía aprovechar mucho mejor todo el ancho de banda que el operador ofrecía y maximizar su rendimiento [12].

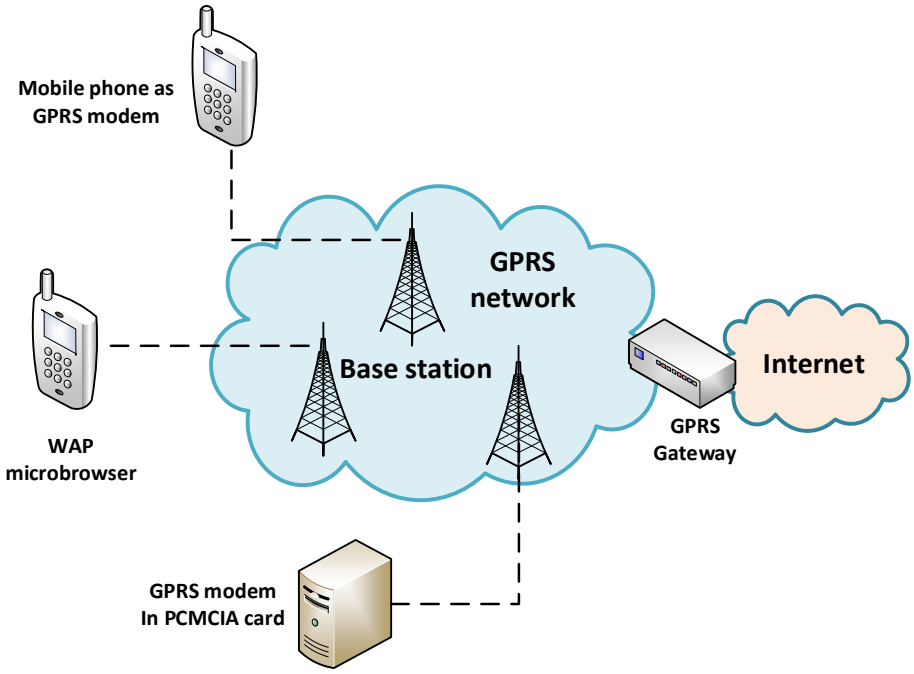


Figura 2. 3 Funcionamiento GPRS [13]

2.4 Base de datos.

Se llama base de datos, o también banco de datos, a un conjunto de información perteneciente a un mismo contexto, ordenada de modo sistemático para su posterior recuperación, análisis y/o transmisión. Existen actualmente muchas formas de bases de datos, que van desde una biblioteca hasta los vastos conjuntos de datos de usuarios de una empresa de telecomunicaciones.

Las bases de datos son el **producto de la necesidad humana de almacenar la información**, es decir, de preservarla contra el tiempo y el deterioro, para poder acudir a ella posteriormente. En ese sentido, la aparición de la electrónica y la computación brindó el elemento digital indispensable para almacenar enormes cantidades de datos en espacios físicos limitados, gracias a su conversión en señales eléctricas o magnéticas.

El manejo de las bases de datos se lleva mediante sistemas de gestión (llamados DBMS por sus siglas en inglés: Database Management Systems o Sistemas de Gestión de Bases de Datos), actualmente digitales y automatizados, que **permiten el almacenamiento ordenado y la rápida recuperación de la información**. En esta tecnología se halla el principio mismo de la informática.

En la conformación de una base de datos **se pueden seguir diferentes modelos y paradigmas**, cada uno dotado de características, ventajas y dificultades, haciendo énfasis en su estructura organizacional, su jerarquía, su capacidad de transmisión o de interrelación, etc. Esto se conoce como modelos de base de datos y permite el diseño y la implementación de algoritmos y otros mecanismos lógicos de gestión, según sea el caso específico [14].

2.5 Base de datos NoSQL.

El término "NoSQL" se refiere a tipos de bases de datos no relacionales que almacenan datos en un formato distinto a las tablas relacionales. Sin embargo, las bases de datos NoSQL se pueden consultar utilizando API de lenguaje natural, lenguajes de consulta estructurados declarativos y lenguajes de consulta mediante ejemplo, por lo que también se les llama bases de datos "no solo SQL". [15]

2.5.1 Uso de una base de datos NoSQL.

Las bases de datos NoSQL se utilizan de forma generalizada en aplicaciones web en tiempo real y big data, ya que sus principales ventajas son los elevados niveles de escalabilidad y disponibilidad. Las bases de datos NoSQL también son la opción preferida entre los desarrolladores, ya que propician por su propia naturaleza un desarrollo ágil dada su rápida adaptación a los requisitos en constante cambio. Las bases de datos NoSQL permiten almacenar los datos de forma más intuitiva y fácil de entender, o más cercanas a la forma en que las aplicaciones utilizan los datos: no necesitan realizar tantas transformaciones cuando almacenan o recuperan datos utilizando interfaces API de NoSQL. Además, las bases de datos NoSQL pueden aprovechar al máximo la nube para evitar por completo el tiempo de inactividad. [15]

2.5.2 Ventajas de una base de datos NoSQL.

La velocidad y la escalabilidad sin precedentes de la interacción digital y el consumo de datos observados en las últimas dos décadas han forzado a las empresas a adoptar un enfoque más moderno y fluido a la forma en que se almacenan los datos y se accede a ellos. En un contexto en que usuarios de todo el mundo exigen un flujo ininterrumpido de contenidos y funciones, no es de extrañar que las bases de datos se hayan tenido que adaptar rápidamente. Teniendo esto en cuenta, compartimos a continuación algunos de los principales motivos clave por los que los desarrolladores eligen bases de datos NoSQL o no relacionales:

Flexibilidad

Con las bases de datos SQL, los datos se almacenan en una estructura predefinida mucho más rígida. Sin embargo, con NoSQL se pueden almacenar de forma más libre sin la rigidez de esos esquemas. Este diseño impulsa la innovación y el rápido desarrollo de aplicaciones. Los desarrolladores pueden centrarse en crear sistemas que ofrezcan un servicio mejor a sus clientes sin preocuparse por los esquemas. Las bases de datos NoSQL pueden manejar

fácilmente cualquier formato de datos, como estructurados, semiestructurados y no estructurados en un único almacén.

Escalabilidad

En lugar de escalar verticalmente agregando más servidores, las bases de datos NoSQL pueden escalarse horizontalmente utilizando hardware básico. De esta forma se puede admitir un mayor volumen de tráfico para satisfacer la demanda sin tiempo de inactividad. Con el escalado horizontal, las bases de datos NoSQL pueden ganar en tamaño y potencia, por lo que se han convertido en la opción preferida para la evolución de los conjuntos de datos.

Alto rendimiento

La arquitectura con escalabilidad horizotnal de las bases de datos NoSQL puede resultar especialmente valiosa cuando aumenta el volumen de datos o el tráfico. Como se muestra en el gráfico siguiente, esta arquitectura garantiza tiempos de respuesta rápidos y predecibles de milisegundos de un solo dígito. Las bases de datos NoSQL también pueden ingerir datos y entregarlos de forma rápida y fiable. Por esta razón, las bases de datos NoSQL se utilizan en aplicaciones que recopilan terabytes de datos todos los días y que requieren una experiencia de usuario altamente interactiva. En el siguiente gráfico, se muestra una tasa de entrada de 300 lecturas por segundo (línea azul) con una latencia de 95 en el rango 3-4ms, y una tasa de entrada de 150 escrituras por segundo (línea verde) con una latencia de 95 en el rango 4-5ms.

Disponibilidad

Las bases de datos NoSQL replican datos de forma automática en múltiples servidores, centros de datos o recursos en la nube. A su vez, esto minimiza la latencia de los usuarios, independientemente de dónde se encuentren. Esta función también sirve para reducir la carga de la gestión de bases de datos. De esta forma, se libera tiempo para dedicarlo a otras prioridades.

Altamente funcional

Las bases de datos NoSQL están diseñadas para almacenes de datos distribuidos que precisan una capacidad de almacenamiento de datos extremadamente grande. Esta característica hace que NoSQL sea la opción ideal para big data, aplicaciones web en tiempo real, cliente 360, compras en línea, juegos en línea, Internet de las cosas, redes sociales y aplicaciones de publicidad en línea. [15]

2.6 Firebase.

Es una plataforma digital que se utiliza para facilitar el desarrollo de aplicaciones web o móviles de una forma efectiva, rápida y sencilla, la cual es utilizada por sus diversas funciones como una técnica de marketing digital para aumentar la base de usuarios y generar mejores beneficios. [16]

Su principal objetivo, es mejorar el rendimiento de las apps mediante la implementación de diversas funcionalidades que van a hacer de la aplicación en cuestión, mucho más manejable, segura y de fácil acceso para los usuarios.

Entre las funciones que se pueden encontrar de Firebase son:

- Lograr una integración dinámica de los usuarios usando Firebase Authentication.
- Lograr que nuestras aplicaciones sean visualizadas y utilizadas utilizando la herramienta de compartir o Dynamic Links.
- Enviar notificaciones a varias plataformas con Cloud Messaging.
- Crear análisis de resultados con Analytics.

2.7 Pruebas de rendimiento y pruebas de campo.

Las pruebas son una parte integral del proceso de desarrollo de una aplicación, para garantizar la máxima calidad y la mejor experiencia de usuario. La integración de pruebas en el proceso de desarrollo ayuda a identificar los problemas en una fase temprana, lo que reduce el tiempo y el costo asociados a la corrección de errores.

El lanzamiento de una aplicación al mercado, puede generar la saturación de un sitio web, o una aplicación móvil ya sea en sus procesos o al acceder a un servidor, si la aplicación no cuenta con un desarrollo óptimo pueden desencadenarse algunas fallas, como el tiempo excesivo de respuesta, sobrecalentamiento de dispositivo, gasto inusual de batería o gasto de datos no necesarios.

Las pruebas de rendimiento tienen la función de evitar ese tipo de defectos, sometiendo al sistema a cargas extremas o poco usuales de uso. De esta manera se sabrá si la aplicación podrá soportarlas en una situación real.

Al finalizar la aplicación debe ser sometida a la prueba de campo, donde se usará en un entorno real, para poder medir el consumo de datos, batería y rendimiento en general bajo el uso de ciudadanos que puedan poner a prueba la aplicación y detectar errores que no hayan sido contemplados durante las pruebas en el desarrollo.

2.8 Experiencia de usuario.

La experiencia de usuario también llamada User Experience o simplemente UX, se define como el conjunto de factores y elementos relacionados con el proceso de interacción de un usuario respecto a un producto o servicio. A menudo, este concepto se aplica a la interacción con páginas web y aplicaciones.

Cuando se desarrolla una aplicación móvil no solo se tienen en cuenta las funcionalidades a desarrollar, el tipo de app o para qué sistema operativo va funcionar, si no que el diseño y la experiencia de usuario son aspectos importantes. Desde el momento en que se comienza a desarrollar la aplicación, se debe tener en cuenta a los usuarios que utilizarán la herramienta y la facilidad para interactuar con ella. [17]

2.8.1 Aspectos fundamentales de experiencia de usuario.

Usabilidad

Es un elemento que ayuda a que los usuarios estén cómodos y les resulte sencillo navegar por una app. A través del desarrollo de una aplicación se pretende solucionar el problema a los usuarios.

Simplicidad

Para una buena experiencia de usuario una app tiene que ser simple, con la información necesaria y de utilidad para los usuarios. No es necesario recargar una aplicación con fuincionalidades que no aportan valor y pueden confundir a las personas que las usen.

El objetivo del desarrollo tiene que ser claro y sencillo desde un primer momento y tiene que verse reflejado en la experiencia del usuario de las aplicaciones móviles.

Navegación

La facilidad de navegación es uno de los puntos más importantes a la hora de mejorar la experiencia del usuario. Una navegación sencilla y sin complicaciones para el usuario hace que pueda acceder al contenido de valor que quiere haciendo que la experiencia en el uso de la app sea positiva.

Además, hay que tener en cuenta el tiempo de carga, aspecto que tambien influye a la hora de acceder a una aplicación móvil. Cuanto mas rápido carguen los contenidos en la app, mejor es la experiencia del usuario.

Elementos en la app

Los botones y llamadas a la acción, los textos o formularios de registro son elementos que suelen estar presentes en casi todas las aplicaciones. Son clave para que la experiencia de usuario se torne positiva si estos estan bien desarrollados y ubicados. [17]

2.9 Android.

Android es un sistema operativo que está basado en Linux principalmente y que es de código abierto, es decir, cualquiera lo puede usar para modificarlo a su gusto.

En un principio fue diseñado para teléfonos móviles y tablets principalmente, aunque después se ha ido incorporando a otros dispositivos como relojes inteligentes con Wear OS, automóviles por medio de Android Auto o Android Automotive o televisores gracias a Android TV.

Android vio la luz en el año 2003 de las manos de Rich Miner, Nick Sears, Chris White y Andy Rubin. Su objetivo era lograr teléfonos móviles más interactivos, mejor posicionados y que tuvieran más opciones para los usuarios.

Después de poco de tiempo desarrollando Android se quedaron sin dinero y si no hubiese sido por la aportación de un préstamo que consiguió Steve Perlman, amigo íntimo de Andy Rubin, jamás hubiera existido este sistema operativo.

En el año 2005 Google compró Android y la hizo entrar como una de sus empresas dejando a Rubin, Miner y White al cargo del sistema operativo, pero bajo su tutela.

No fue hasta el año 2007, en la presentación del consorcio tecnológico Open Handset Alliance, dónde estaban presentes marcas como Samsung, Qualcomm, HTC o la propia Google, cuando se anunció oficialmente Android.

En el año 2008 apareció el primer terminal con este sistema operativo en su interior. Ese honor lo ostenta el HTC Dream. A partir de entonces, muchos otros terminales comenzaron a llevar Android, haciendo que se llegase a nuestros días donde se estima que más del 90% de los teléfonos inteligentes del mercado lo llevan. [18]

2.9.1 Android Studio.

[19] Android Studio es el entorno de desarrollo integrado (IDE) oficial que se usa en el desarrollo de apps para Android. Basado en el potente editor de código y las herramientas para desarrolladores de IntelliJ IDEA, Android Studio ofrece funciones que mejoran la productividad cuando se compila apps para Android como son:

- Un sistema de compilación flexible basado en Gradle
- Un emulador rápido y cargado de funciones
- Un entorno unificado donde puedes desarrollar para todos los dispositivos Android
- Ediciones en vivo para actualizar elementos componibles en emuladores y dispositivos físicos, en tiempo real
- Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra
- Variedad de marcos de trabajo y herramientas de prueba
- Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros
- Compatibilidad con C++ y NDK
- Compatibilidad integrada con Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine

2.10 Lenguaje de programación Java.

Java es un lenguaje de programación creado en 1995 por Sun Microsystems. [20]
El proyecto inicialmente se llamaba Oak, pero posteriormente se cambió el nombre a Java por cuestiones legales.
En sus inicios estaba destinado a ser un lenguaje para televisores, pero debido a lo avanzado que era y el potencial que tenía se continuó desarrollando para ser utilizado como un lenguaje de programación.
Se optó por utilizar una sintaxis parecida a la de C++ puesto que ya era familiar para los programadores de la época y así resultaría fácil de aprender.
Además, se buscaba crear un lenguaje que permitiese desarrollar las aplicaciones una única vez y ejecutarlas en cualquier plataforma sin necesidad de realizar modificaciones sobre las mismas. Gracias a la máquina virtual fué posible, por lo que Java se convirtió en el primer lenguaje multiplataforma de la historia.
Se hizo muy popular y su uso se fue extendiendo de forma muy rápida, siendo hoy uno de los lenguajes de programación más usados en todo el mundo.

2.10.1 Carácterísticas de Java.

Java se caracteriza por ser un lenguaje independiente de la arquitectura, orientado a objetos, interpretado, distribuido, multihilo, robusto y seguro.

Por todo ello Java se ha convertido en un lenguaje muy popular y su uso se ha extendido tanto.

Independiente de la arquitectura

Gracias a la máquina virtual de Java (Java Virtual Machine JVM) se puede ejecutar el mismo programa en cualquier sistema operativo (Linux, Windows, MacOS, etc.) y en cualquier hardware sin necesidad de hacer modificaciones sobre el mismo. Esto convierte a Java en un lenguaje multiplataforma y 100% portable.

Orientado a objetos

Java es un lenguaje orientado a objetos puesto que permite la definición de clases y la instancia de objetos de las mismas.

Además, tiene todas las características comunes de la programación orientada a objetos: polimorfismo, herencia y enlazado dinámico.

También permite definiciones abstractas de clases, lo que se conoce como interfaces.

Interpretado

No es cierto que Java sea 100% interpretado. Hay una fase inicial en la que el código se compila para generar los ficheros .class de tipo Bytecode.

Los ficheros Bytecode generados tras la compilación son los que la Máquina Virtual de Java interpretará durante su ejecución.

Esta fase de interpretado es muy importante puesto que es la que permite ejecutar los programas implementados en Java de forma independiente a la plataforma.

Distribuido

Existen en Java multitud de bibliotecas que permiten a las aplicaciones utilizar protocolos de red como http o ftp. Gracias a esto no es necesario tener una única aplicación corriendo en una máquina. Se pueden implementar múltiples aplicaciones en distintos nodos que se comuniquen a través de la red.

Multihilo

Java es capaz de ejecutar múltiples tareas de forma simultánea, lo que reduce el tiempo de ejecución y mejora el rendimiento del programa.

Aplicaciones que necesitan cargar diferentes tipos de datos (como textos e imágenes) se pueden aprovechar de esta característica para cargar la información de diferente tipo en hilos separados, de forma que se eviten bloqueos por culpa de la carga más lenta de los elementos más pesados.

Robusto y seguro

Tanto la declaración de tipos como la definición de métodos durante la fase de desarrollo ayudan a la detección de errores de forma prematura, como la conversión errónea de datos.

Posteriormente, durante la compilación y la ejecución, Java realiza multitud de comprobaciones para evitar errores.

Gestiona tanto la reserva como la liberación de memoria gracias a su recolector de basura, la comprobación de punteros y la comprobación de límites de arrays. De esta forma se libera al programador de la responsabilidad de realizar esta tarea.

También realiza múltiples comprobaciones de los Bytecode antes de ser ejecutados por la Máquina Virtual de Java para minimizar el número de errores durante la ejecución, como posibles desbordamientos en la pila.

Igualmente, durante la ejecución, Java dispone de una gestión de excepciones para controlar errores imprevistos y reducir el impacto de los mismos durante la ejecución.

Además, Java mantiene su espacio de nombres independiente de los recursos procedentes de la red para evitar así ataques de troyanos.

2.11 Maquetación de aplicaciones móviles.

Hacer un prototipo de app móvil es una de las técnicas más sencillas para previsualizar cómo va a quedar la aplicación a desarrollar, en cuestión de diseño, y sin tener que empezar a programar ni una línea de código.
Hacer un prototipo de app ayuda a ahorrar mucho tiempo y trabajo. Además, permite ver de manera muy fiel cómo va a quedar la app y si es necesario optimizar el diseño, apariencia y usabilidad de la misma. [21]

2.11.1 Justinmind.

Es una de las herramientas más potentes en su ámbito. Tiene una herramienta para hacer wireframes gratuita y la versión completa para hacer prototipos está disponible desde $ 19 al mes (con esta versión más económica tienes prototipos ilimitados). Puedes descargarla de forma gratuita en Mac y Windows.
La herramienta cuenta con una galería de elementos nativos de iOS y Android que te permiten crear un prototipo muy fiel al resultado final en el móvil. Puedes cargar los diseños directamente desde Photoshop o Sketch, entre otras herramientas.
Puedes dotar a los diseños de prácticamente cualquier transición (rotar, tap, voltear, scroll…). Una vez tengas listo tu prototipo, Justinmind te permite ponerlo a prueba en cualquier dispositivo Android, iPhone o iPad a partir de un simulador. [21]

CAPÍTULO 3

MARCO METODOLÓGICO

3.1 Metodología.

De acuerdo a la experiencia laboral obtenida en el desarrollo de aplicaciones de escritorio, la metodología seleccionada será "programación extrema (XP)", esta metodología siempre fue utilizada durante los procesos de desarrollo, dando un resultado deseado en las implementaciones ya que su estructura nos permite establecer tiempos de entrega y agilizar procesos gracias a sus iteraciones cortas[22]. Como se muestra en la figura 3.1.

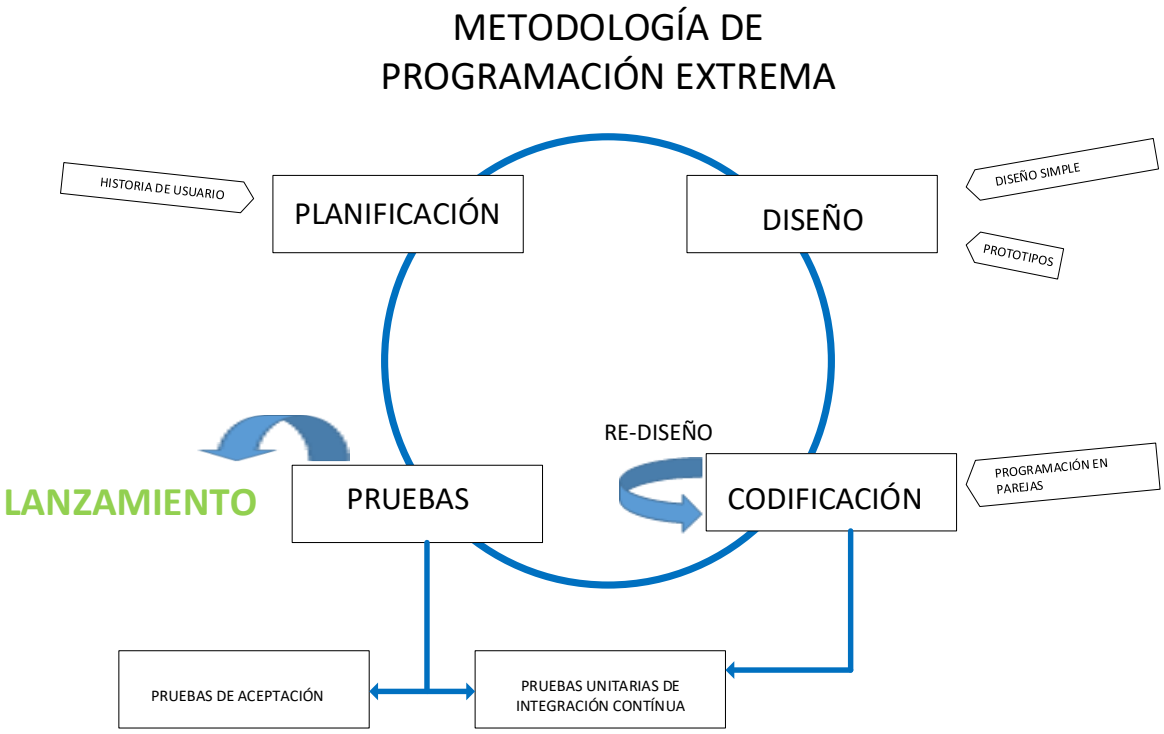


Figura 3. 1 Diagrama de metodología programación extrema. [22]

Las ventajas que ofrece esta metodología, son el desarrollo de trabajo ágil que tiene como finalidad la entrega de valor en periodos cortos de tiempo y para ello se basa en 3 pilares: transparencia, inspección y adaptación.

- TRANSPARENCIA: Permite dar a conocer toda la información correspondiente al proyecto a las partes interesadas o que integran el grupo de desarrollo.

- INSPECCIÓN: Garantiza que las revisiones sean periódicas con la finalidad de detectar y corregir fallos.
- ADAPTACIÓN: Tiene que ver con las características del equipo de trabajo para ajustar tiempos y lograr dichos objetivos.

3.2 Análisis.

La identificación de una necesidad enunciada en términos concretos es el punto de partida para la puesta en marcha de un proyecto y la evaluación de las posibles soluciones que dará viabilidad del mismo. Para el análisis se requiere de 3 etapas fundamentales:

- Conceptualización.
- Planeación.
- Requerimientos del proyecto.

3.3 Diseño.

El diseño comprende cuatro tipos de actividades, diseño de base datos, de arquitectura, procedimental y diseño de interfaces, desde el punto de vista el proyecto evoluciona desde un diseño preliminar a un diseño detallado.

- Diseño de diagramas de flujo.
- Diseño de base de datos.
- Diseño de interfaces de aplicación móvil.

3.4 Desarrollo.

Crear los datos de prueba, interpretar código fuente, implementar código fuente, generar documentación, planificar y realizar la integración de los módulos.

- Creación de tablas de bases de datos.
- Desarrollo de interfaces gráficas ANDROID.
- Codificación de aplicación móvil.
- Conexión de base de datos con aplicación móvil.
- Desarrollo de interface de aplicación de escritorio.
- Desarrollo de aplicación de escritorio.
- Conexión a base de datos con aplicación de escritorio.

3.5 Pruebas iniciales.

Las pruebas que se harán al proyecto son las siguientes:

- Pruebas de dispositivo GPS
    - tiempo de respuesta.
    - precisión de geolocalización.
    - tiempo de batería.
- Pruebas de GPS teléfono móvil
    - tiempo de respuesta.
    - precisión de geolocalización.
    - tiempo de batería.
- Pruebas de codificación
    - pruebas unitarias de módulos.
    - pruebas de integración de módulos.
    - pruebas funcionales.
    - pruebas de carga y estrés.
    - pruebas de usabilidad.

3.6 Puesta a punto de la aplicación.

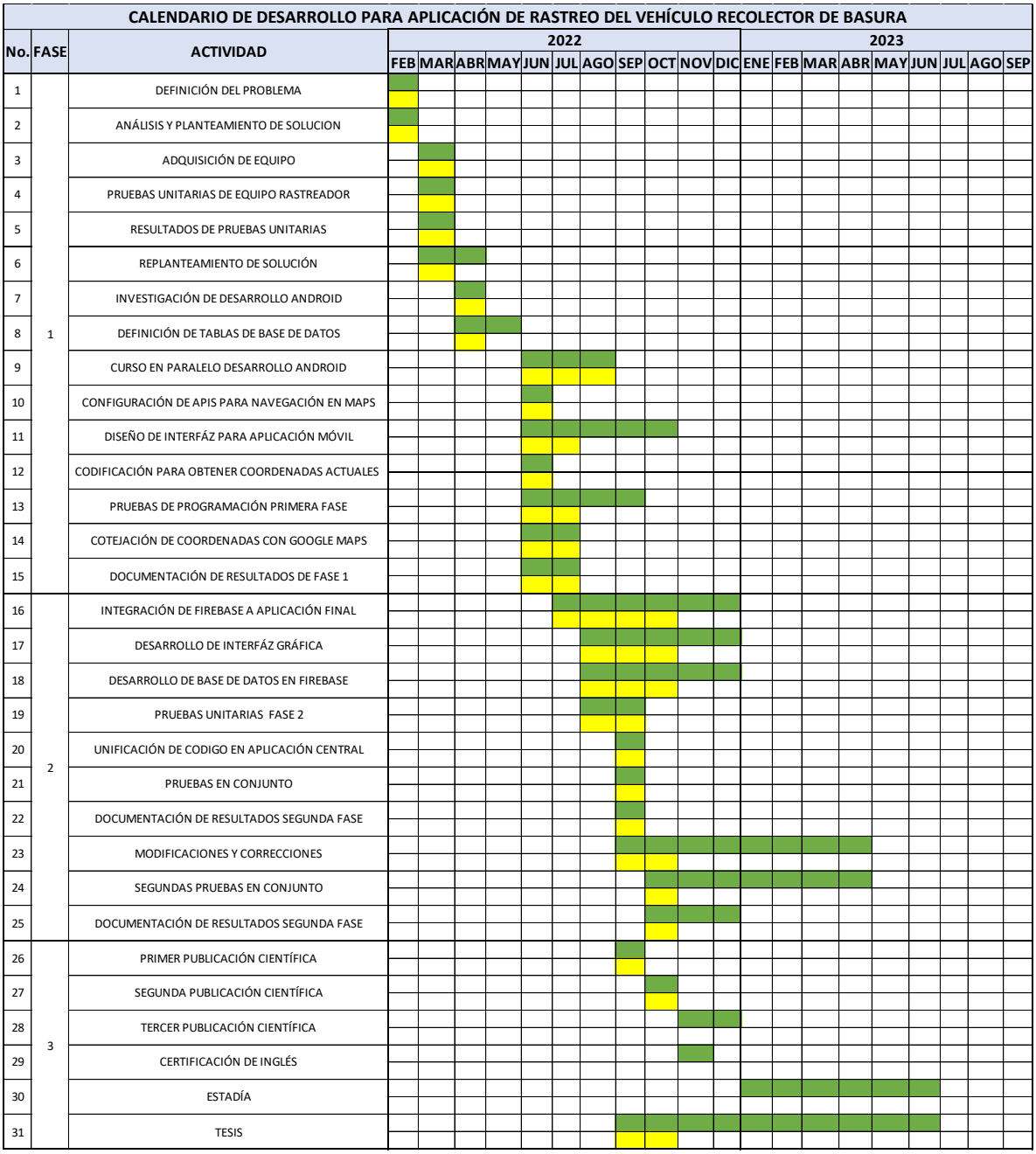
Implementación del sistema y todos sus componentes en un entorno real de funcionamiento y con sujetos de pruebas para la obtención de los resultados deseados.

- Configuración del módulo Real time data base en Firebase.
- Configuración de las APIS de google maps.
- Configuración del módulo cloud messaging de Firebase para las notificaciones push.
- Instalación de la aplicación en dispositivos Android.

3.7 Análisis y planeación.

La planificación del proyecto es una de las etapas más importantes, basadas en la metodología antes mencionada, los resultados deben considerarse de forma rápida y bien estructurada, para eso se han calendarizado las tareas específicas y definido tiempos de desarrollo de cada una de ellas, véase en tabla 3.1.

Tabla 3. 1 Cronograma de actividades para desarrollo.



3.8 Listado de herramientas, materiales y software.

En la tabla 3.2 se muestran las herramientas que se utilizaran para el desarrollo de la aplicación, materiales que se consideraron para la implementación y equipos.

Tabla 3. 2 Lista de Herramientas para el desarrollo de aplicación.

LISTA DE HERRAMIENTAS		
ELEMENTO	TIPO	DESCRIPCIÓN
Mini Rastreador magnetico GF07	Dispositivo GPS	Es un dispositivo GPS de tamaño compacto, el cual tiene una ranura para chip GSM, esto para enviar las ubicaciones mediante la misma red GSM
Android Studio	IDE	Es un entorno de desarrollo integrado oficial para la plataforma Android
Java	Lenguaje de programación	Es un lenguaje de programación y una plataforma informática desarrollada por "Sun Microsystems" que puede ejecutarse en cualquier máquina virtual JAVA sin importar la arquitectura de la computadora subyacente
Firebase	Manejador de Base de Datos	Es una plataforma de desarrollo de apps que ayuda a complilar y desarrollar las apps con el respaldo de google
Google maps API	API	Es una interfaz de google para la integración y visualización de diversos sitios y aplicaciones, dandole diversas acciones al usuario como el posicionamiento global

3.9 Listado de costos.

En la tabla 3.3 se muestra la relación de costos del material y software utilizado para el desarrollo de la aplicación.

Tabla 3. 3 Relación de costos generados para el desarrollo del proyecto.

COSTOS	
ELEMENTO	COSTO
Mini Rastreador magnetico GF07	$400
Android Studio	n/a
Java	n/a
Firebase	ver tabla Realtime Database
Google maps API	$0.007 USD por solicitud

REALTIME DATABASE	
DESCRIPCIÓN	PLAN BLAZE
Conexiones simultáneas	200000 por base de datos
GB almacenados	$5 USD por GB
GB descargados	$1 USD por GB
Varias bases de datos por proyecto	Si

3.10 Análisis de funcionamiento de proyecto.

La aplicación general tendrá dos opciones una para conductor y otra para usuario, se identificarán con su correo electrónico y dependiendo de su registro la aplicación tendrá diferentes funciones para cada tipo de usuario. Del lado del conductor, la aplicación mostrará su ubicación actual y guardará en una base de datos las últimas coordenadas cada que este se mueva se actualizarán los valores.

En el caso del usuario, la aplicación mostrará su ubicación actual y accederá a la base de datos para revisar que conductores están activos, revisará las coordenadas de cada conductor en la base de datos y mostrará en el mapa a todos los conductores dentro de un radio específico, así como al estar cerca de la zona donde se encuentra el usuario este será notificado que ya está cerca el vehículo recolector de basura.

La inserción de los datos será hacia una base de datos en Firebase, así como la autenticación de los usuarios.

La geolocalización será proporcionada por los servicios de google y la comunicación será a través de la red GSM de los equipos móviles, como se muestra en la figura 3.2.

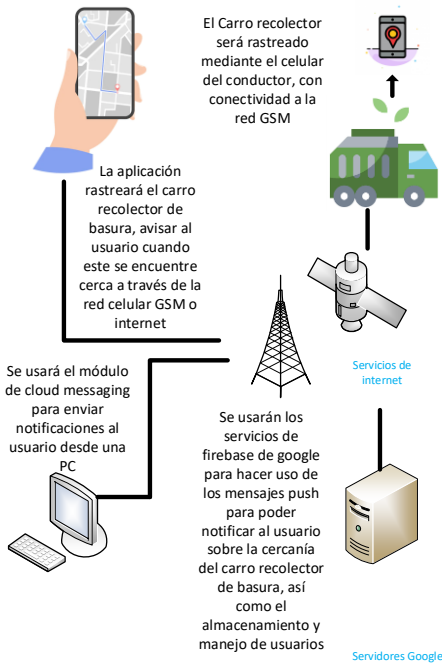


Figura 3. 2 Diagrama de funcionamiento general.

3.11 Casos de uso.

Los diagramas de casos de uso nos permiten identificar los componentes principales que forman el sistema y nos ayuda a recolectar los requerimientos del sistema en pocas palabras, cuáles son las funciones que realiza nuestro sistema.

Al crear un caso de uso se determinan que los actores que intervienen son 4, que son el administrador del sistema, dispositivo móvil del usuario, dispositivo móvil del conductor y aplicación. Los casos de uso que corresponden al administrador del sistema permiten la posibilidad de eliminar y editar información, de los usuarios como de los conductores para crear campañas de notificaciones.

La aplicación por parte del conductor se encargará de enviar a la base de datos las últimas coordenadas cada que este se mueva en el mapa, mostrará la ubicación actual, la información general del conductor y el itinerario general de la semana.

La aplicación del usuario señalará en el mapa dónde se encuentran todos los conductores en tiempo real en un radio de 1 km a la redonda de donde se encuentre el usuario ubicado, mostrará su información general, el itinerario general del servicio de limpia, notificará cuando el carro de basura este cerca y si es que se suspende el servicio de limpia, como se muestra en la figura 3.3.

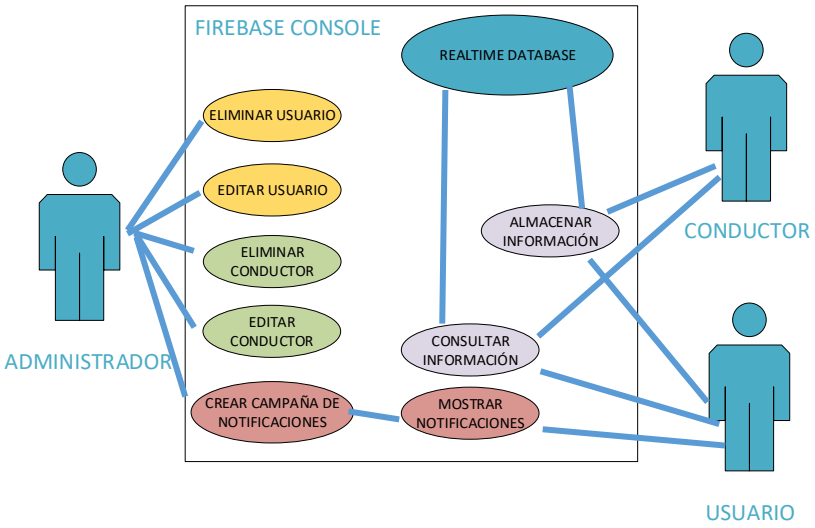


Figura 3. 3 Diagrama de caso de uso.

3.12 Diagrama de flujo.

Esta herramienta se usa para representar los procesos del sistema informático, es de gran importancia ya que permite documentar, planificar y mejorar el proceso de forma clara y sencilla. Para la codificación y análisis de uso, el diagrama muestra la ejecución del programa, mostrando primero la pantalla de bienvenida. Seguido a esto el usuario debe seleccionar si es conductor o ciudadano para así poder mostrar las pantallas correspondientes para cada tipo de usuario. Si el usuario está registrado se muestra la pantalla de login, y si el usuario es nuevo se muestra la pantalla de registro correspondiente, esto para ambos casos de tipo de usuario.

Una vez terminado el registro o login se revisa que el GPS del equipo se encuentre encendido y que se tengan los permisos necesarios para poder ejecutar la aplicación. Al finalizar estas validaciones se muestra la pantalla principal con el mapa correspondiente (figura3.4).

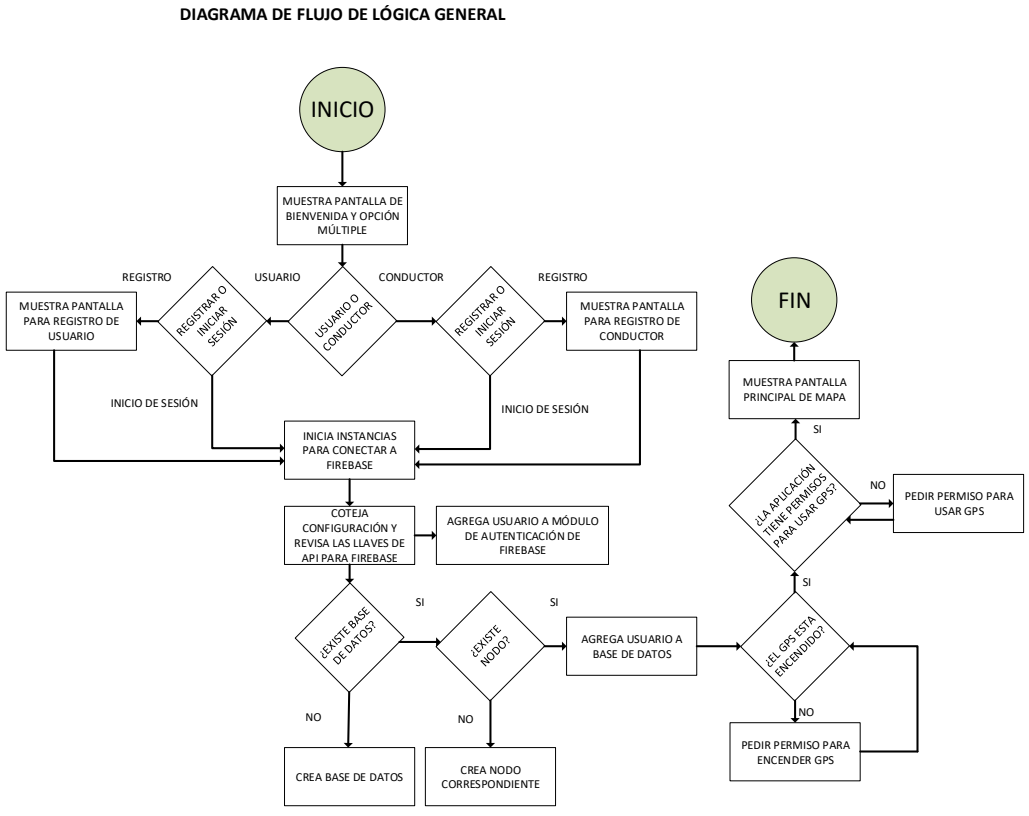


Figura 3. 4 Diagrama de flujo de funcionamiento general.

3.13 Diseño de la base de datos.

Uno de los aspectos más importantes de un desarrollo de software es la base de datos donde será almacenada toda la información necesaria. Firebase Realtime Database es una base de datos NoSQL, por lo cual no lleva un modelo de entidad-relación como las bases de datos tradicionales. Esto ayuda a soportar diferentes estructuras sin incrementar la complejidad de su manejo, escalar para soportar gran cantidad de usuarios, mantener la aplicación con un nivel de respuesta adecuado ante el aumento de carga, tener un esquema no centralizado, tolerar y responder a las fallas en los sistemas y acoplarse al desarrollo ágil con entregas incrementales. Ya que el proyecto trabajará con una base de datos NoSQL en la tabla 3.4 se representan de forma tabular los nodos que serán los elementos, así como sus campos.

Tabla 3. 4 Representación de nodos de base de datos.

BASE DE DATOS PARA SOFTWARE DE RASTREO DEL VEHÍCULO DE BASURA			
CONDUCTOR	USUARIO	TOKEN	ACTIVE DRIVERS
ID	ID	ID	ID
NOMBRE	NOMBRE		LATITUDE
COLONIA	COLONIA		LONGITUDE
TELEFONO	TELEFONO		
SEXO	SEXO		
TIPO DE SANGRE	EMAIL		
NUMERO DE LICENCIA			
NUMERO DE SEGURO SOCIAL			
EMAIL			

3.14 Relación entre nodos.

Las bases de datos NoSQL no tienen una estructura de entidad relación, sin embargo, en la codificación de la aplicación se hace una relación para cotejar los datos de los usuarios mediante su ID. Esta relación se representa en la figura 3.5.

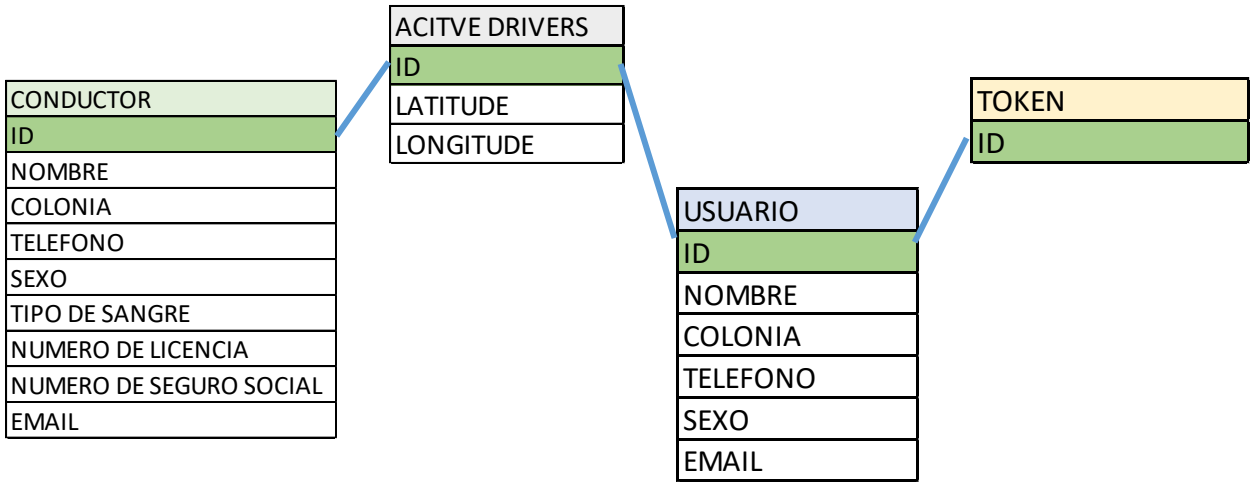


Figura 3. 5 Representación de relación entre nodos mediante ID.

3.15 Desconexión.

La base de datos no tendrá problema si el dispositivo se queda sin conexión, debido a que el SDK conserva los datos en disco. Una vez que se reestablece la conectividad, el dispositivo cliente recibe cualquier tipo de cambio que se haya perdido, sincronizándolo con el estado actual del servidor [23], en la figura 3.6 se muestra el proceso de guardado por cada vehículo recolector de basura, cuando el conductor vuelva a tener conexión se actualizarán sus coordenadas que quedaron pendientes en disco y los usuarios podrán darle continuidad a su rastreo.

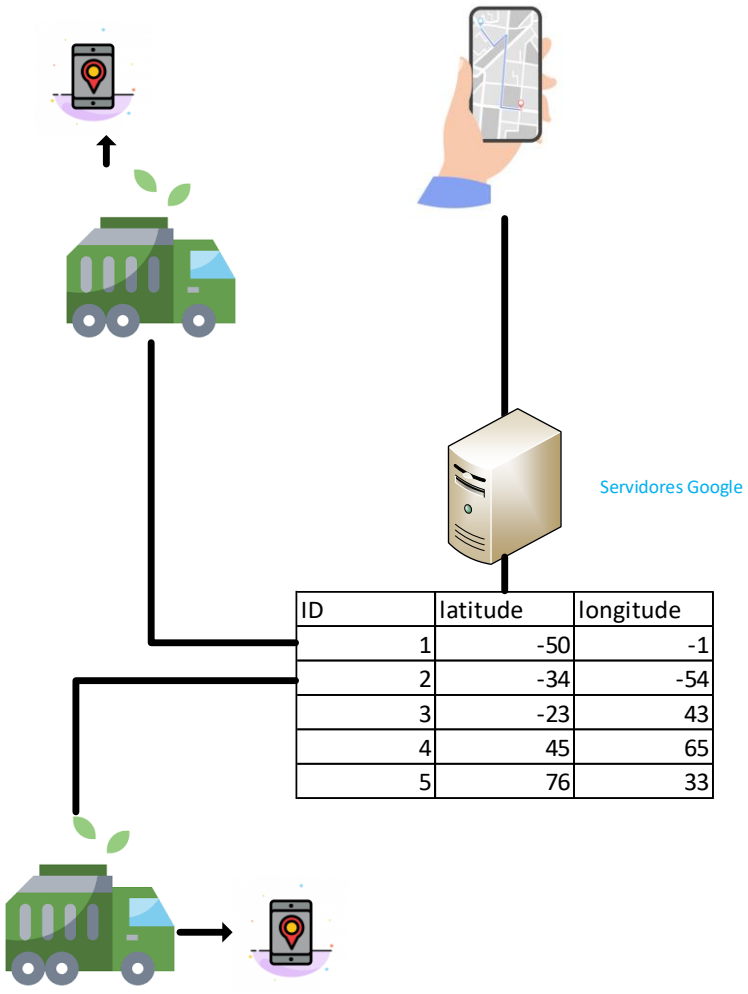


Figura 3. 6 Actualización de coordenadas de los vehículos recolectores.

3.16 Diseño final de base de datos.

Estos datos serán almacenados en los servidores de google así garantizando que la información está almacenada de manera segura por medio de las reglas de seguridad las cuales son:

- Que la lectura y escritura de los datos solo sea permitida a usuarios autenticados.
- Que dichos usuarios pueden solo leer sus propios datos.
- Los usuarios administradores podrán leer, escribir, modificar y eliminar cualquier dato.

3.17 Configuración de Firebase.

Para poder hacer uso de la plataforma de Firebase se requiere de una configuración previa, la cual es proporcionada por google. Esto permitirá a la aplicación funciones como login, base de datos, mensajes push, y almacenamiento de geolocalizaciones actuales de los conductores de camión para que los usuarios puedan ver a los conductores activos.

Los pasos son los siguientes:

1. Se debe dirigir a la consola de configuración de Google Firebase, y agregamos un nuevo proyecto y en la pantalla principal seleccionamos agregar app, se selecciona la plataforma a la que se va a desarrollar y se registra el paquete (véase Figura 3.7).
2. Seleccionar el paquete de la aplicación y dar clic en copiar path.
3. Seleccionar copiar referencia.
4. Pegar la referencia copiada en el registro de la app de Firebase.
5. Abrir CMD de Windows y ejecutar la siguiente instrucción: Keytool -list -v -alias androiddebugkey -keystore %USERPROFILE%\.android\debug.keystore .
6. Copiar el certificado digital sha1 (figura 3.8).

Figura 3. 7 Nuevo proyecto Firebase.

7. Pegar el certificado en el campo correspondiente de Firebase; con esta secuencia de pasos, 2 a 7 se logra el registro del paquete.

Figura 3. 8 Registro de paquete en Firebase.

8. Descargar el archivo de google-services.json.
9. Pegar el archivo json en la carpeta app, Seguido se requieren agregar las siguientes líneas en los siguientes archivos (Figura 3.9).
10. Sincroniza el proyecto y si todo es correcto, se tendrá agregada la dependencia a Firebase.
11. Se deben de habilitar los módulos deseados para nuestro proyecto en la consola de Firebase.

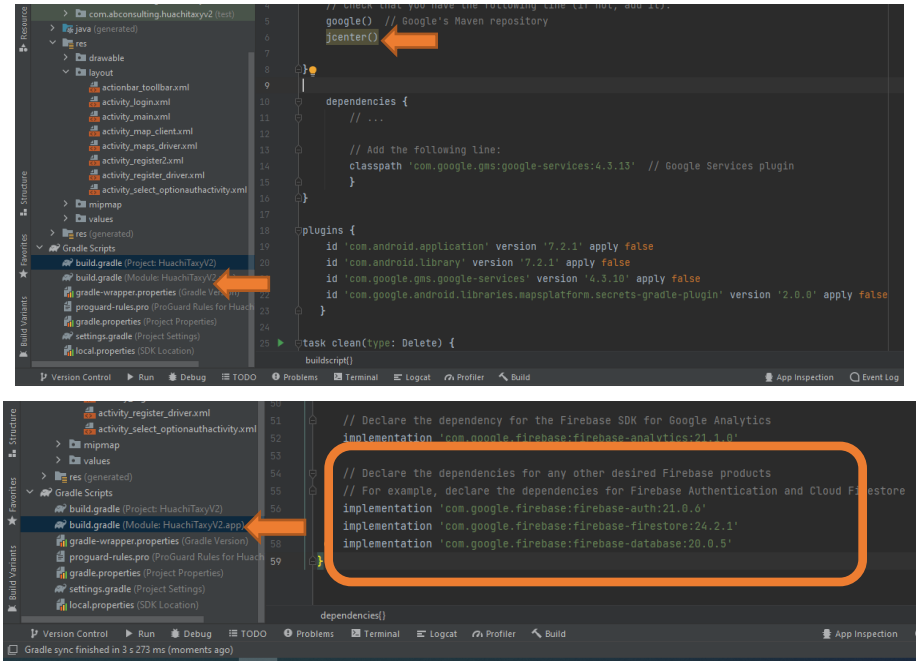


Figura 3. 9 Implementación y librerías de Firebase.

El módulo de autenticación será utilizado para hacer el login de usuarios en la aplicación, y el módulo de Firebase realtime database se utilizará para almacenar los datos de los usuarios en una base de datos. Esto permitirá dar diferentes funciones por cada tipo de usuario ya que el conductor únicamente tendrá un mapa de su ubicación actual, mientras que el usuario podrá saber dónde se encuentran todos los vehículos recolectores en tiempo real, así como también será notificado de cuando este cerca el vehículo recolector de su zona únicamente y también será notificado en caso de que el vehículo recolector no vaya a pasar el día programado.

La lógica del inicio de sesión y registro de usuarios a la base de datos es con el módulo de autenticación realtime database en Firebase, Se hace una selección ya sea entre usuario o conductor, seguido a esto se selecciona entre el registro o el acceso de un usuario registrado, si el usuario es nuevo se crea en la base de datos y en el módulo de autenticación, por otra parte, si el usuario existe se coteja que la información sea correcta y una vez autenticado correctamente se dirige a la pantalla principal de la aplicación véase figura 3.10.

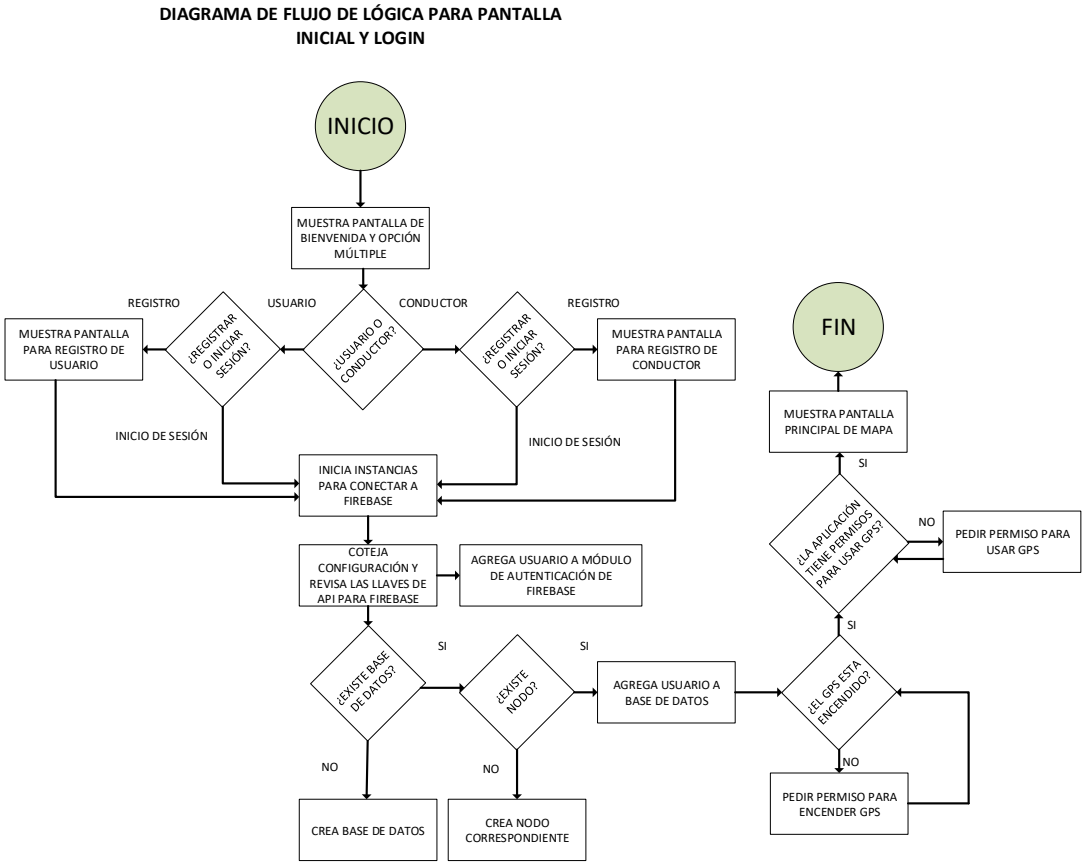


Figura 3. 10 Diagrama de flujo para el funcionamiento de login y registro de usuarios.

3.18 Métricas de diseño de aplicación.

Para el diseño de la aplicación se utilizaron las métricas para la calidad en el desarrollo de aplicaciones proporcionada por Android**,** como experiencia visual, funcionalidad, privacidad y seguridad entre otras. En esta lista de tareas se define el conjunto de criterios principales de calidad y pruebas relacionadas que ayudarán a evaluar la calidad de la app [24].

3.18.1 Experiencia visual.

La app debe proporcionar los patrones de interacción y diseño visual estándares de Android cuando corresponda, a fin de garantizar una experiencia del usuario intuitiva y coherente, las tablas 3.5 y 3.6 se muestran los criterios evaluados.

Tabla 3. 5 Métricas de experiencia visual 1.

EXPERIENCIA VISUAL				
ÁREA	**ID**	**PRUEBAS**	**DESCRIPCIÓN**	**Resultado**
Navegación	VX-N1	CR-3	La app admite la navegación estándar del botón Atrás y no utiliza avisos personalizados en pantalla para este.	**ok**
	VX-N2	CR-3	La app admite la navegación por gestos a fin de navegar a la pantalla principal y volver a ella.	**ok**
	VX-N3	CR-1	La app preserva y restaura correctamente el estado del usuario o la app.	**ok**
		CR-3	La app preserva el estado del usuario o la app cuando abandona el primer plano y evita la pérdida accidental de datos a causa de la navegación hacia atrás y otros cambios de estado.	**ok**
		CR-5	Cuando regresa al primer plano, la app debe restablecer el estado preservado y toda transacción importante con estado que haya quedado pendiente, como los cambios en campos editables, el progreso de un juego, menús, videos y otras secciones de la app o el juego.	**ok**
			1. Cuando se reanuda la app desde el conmutador de Apps recientes, la app regresa al usuario al estado exacto en que se encontraba la última vez que este la utilizó.	**ok**
			2. Cuando se reanuda la app después de la activación del dispositivo (luego de haber estado bloqueado), esta regresa al usuario al estado exacto en que se encontraba en la última utilización.	**ok**
			3. Cuando se reinicie desde Página principal o Todas las apps, la app debería realizar una de las siguientes acciones, según cuánto tiempo haya transcurrido desde la última utilización:	
			Si la app se usó por última vez hace poco tiempo (minutos), restablece el estado de la app lo más cerca posible de su estado anterior.	**ok**
			Si pasó más tiempo desde que se usó la app por última vez, restablécela lo más cerca posible de su estado anterior o iníciala en su pantalla principal o en cualquier otro estado predeterminado.	**ok**
Notificaciones	VX-S1	CR-9	Las notificaciones siguen los lineamientos de Material Design. En particular:	
			1. Las notificaciones no se utilizan para realizar promoción cruzada ni para publicitar otro producto, ya que Play Store lo prohíbe de forma estricta.	**ok**
			2. Define los canales de notificaciones de forma adecuada en virtud de las prácticas recomendadas, en lugar de entregar todas las notificaciones desde un canal único.	**ok**
			3. Selecciona la prioridad de notificación correcta.	**ok**
			4. Cuando sea posible, las notificaciones se consolidarán en un solo grupo de notificaciones.	**ok**
			5. Establece tiempos de espera para las notificaciones cuando sea necesario.	**n/a**
			6. Las notificaciones son persistentes solo si están relacionadas con eventos en curso, como una llamada telefónica o la reproducción de música (obtén más información en la sección Funcionalidades).	**ok**
	VX-S2	CR-9	Para apps de mensajería o sociales y conversaciones:	**n/a**
			1. Usa las notificaciones MessagingStyle para las conversaciones.	
			2. Brinda compatibilidad con la acción de respuesta directa.	
			3. Admite combinaciones de teclas en conversaciones e implementa prácticas recomendadas para obtener la mejor clasificación de uso compartido.	
			4. Admite burbujas.	

Tabla 3. 6 Métricas de experiencia visual 2.

EXPERIENCIA VISUAL				
ÁREA	**ID**	**PRUEBAS**	**DESCRIPCIÓN**	**Resultado**
IU y gráficos	VX-U1	CR-5	La app admite tanto la orientación horizontal como la vertical (si fuera posible).	**no**
			Las orientaciones exponen ampliamente las mismas funciones y acciones, y conservan la paridad funcional. Se aceptan cambios mínimos en el contenido o las vistas.	**n/a**
	VX-U2	CR-5	La app utiliza toda la pantalla en ambas orientaciones y no usa formato de pantalla ancha para compensar los cambios de orientación.	
			Se acepta el uso mínimo de formato de pantalla ancha para compensar pequeñas variaciones en la geometría de la pantalla.	
	VX-U3	CR-5	La app controla correctamente las transiciones rápidas entre las orientaciones de la pantalla sin presentar problemas de renderización ni perder el estado.	
Calidad visual	VX-V1	CR-all	La app muestra gráficos, texto, imágenes y otros elementos de la IU sin distorsión, esfumado ni pixelado notables.	**ok**
			1. La app deberá usar interfaces dibujables en vector cuando sea posible.	**ok**
			2. La app ofrece gráficos de alta calidad para todos los tamaños de pantalla y factores de forma orientados.	**ok**
			3. No se observa suavizado en los bordes de los menús, los botones ni otros elementos de la IU.	**ok**
	VX-V2	CR-all	La app muestra texto y bloques de texto de forma aceptable para cada uno de los idiomas que admite.	**ok**
			1. La composición es aceptable en todos los factores de forma compatibles.	**ok**
			2. No se visualizan letras ni palabras cortadas.	**ok**
			3. No se visualizan ajustes automáticos de línea incorrectos en botones ni íconos.	**ok**
			4. Hay suficiente espacio entre el texto y los elementos que lo rodean.	**ok**
	VX-V3	CR-all	El contenido de la app y el contenido web al que esta hace referencia admiten el tema oscuro.	**ok**

3.18.2 Funcionalidad.

La app está diseñada para ser de manera informativa para el usuario. Por tal motivo, algunos de los criterios que menciona Android en cuanto a la funcionalidad no son aplicables en este caso. Sin embargo, la aplicación deberá implementar el comportamiento funcional mostrado en la tabla 3.7.

Tabla 3. 7 Métricas de funcionalidad.

FUNCIONALIDAD				
ÁREA	**ID**	**PRUEBAS**	**DESCRIPCIÓN**	**Resultado**
Audio	FN-A1	CR-1	La reproducción de audio continúa cuando la app regresa a primer plano, o le indica al usuario que la reproducción está pausada.	n/a
		CR-8		
	FN-A2	CR-1	Si la reproducción de audio es una función principal, la app deberá admitir la reproducción en segundo plano.	
		CR-2		
		CR-8		
	FN-A3	CR-0	Cuando el usuario inicie la reproducción de audio, la app deberá realizar una de las siguientes acciones antes de que transcurra un segundo:	
			1. Comenzar a reproducir el audio	
			2. Proporcionar un indicador visual de que los datos de audio se están preparando	
	FN-A4	CR-0	La app deberá solicitar foco de audio cuando el audio comience a reproducirse y deberá abandonarlo cuando se detenga la reproducción.	
	FN-A5	CR-0	La app deberá controlar las solicitudes de foco de audio de otras apps. Por ejemplo, una app podría bajar el volumen de reproducción cuando otra reproduzca contenido de voz.	
Multimedia	FN-M1	CR-0	Si la app reproduce audio en segundo plano, deberá crear una notificación con estilo MediaStyle.	n/a
		CR-6		
		CR-8		
	FN-M2	CR-0	Si la app reproduce video, deberá admitir la reproducción pantalla en pantalla.	n/a
	FN-M3	CR-0	Si la app codifica contenido de video, deberá hacerlo mediante el estándar de compresión de videos HEVC.	n/a
Se comparte	FN-S1	CR-0	La app deberá usar Android Sharesheet cuando comparta contenido. Puede sugerir objetivos que no están disponibles para las soluciones personalizadas.	n/a
Servicio en segundo plano	FN-B1	CR-6	Si es posible, la app evita que se ejecuten servicios en segundo plano. Para garantizar que el dispositivo del usuario funcione sin problemas, el sistema aplica varias restricciones a los servicios en segundo plano. Los siguientes no se consideran buenos usos de los servicios en segundo plano:	ok
			Mantener una conexión de red para las notificaciones	ok
			Mantener una conexión Bluetooth	ok
			Mantener el GPS encendido	ok
	FN-B2	CR-10	La app admite correctamente las funciones de administración de energía que se introdujeron en Android 6.0 (Descanso y App Standby). Cuando se interrumpe la funcionalidad central por la gestión de energía, solo apps calificadas pueden solicitar una exención. Consulta Compatibilidad con otros casos de uso en Descanso y App Standby.	ok

3.18.3 Rendimiento y estabilidad.

La aplicación debe proporcionar estabilidad, compatibilidad, y capacidad de respuesta que los usuarios esperan, para esto se hacen las pruebas que Android sugiere. Las actividades se muestran en la tabla 3.8.

Tabla 3. 8 Métricas de rendimiento y estabilidad.

RENDIMIENTO Y ESTABILIDAD				
ÁREA	**ID**	**PRUEBAS**	**DESCRIPCIÓN**	**Resultado**
Estabilidad	PS-S1	CR-all	La app no falla ni bloquea el subproceso de IU que provoca errores ANR ("Android no responde"). Usa el informe previo al lanzamiento de Google Play a fin de identificar posibles problemas de estabilidad. Después de la implementación, presta atención a la página de Android Vitals en Google Play Console.	**ok**
		SD-1		
Rendimiento	PS-P1	CR-all	La app se carga rápidamente o le proporciona al usuario comentarios en pantalla (como un indicador de progreso o una señal similar) en el caso de que tarde más de dos segundos en cargarse.	**ok**
		SD-1		
	PS-P2	CR-all	Las apps deberán renderizar los fotogramas cada 16 ms a efectos de alcanzar 60 fotogramas por segundo. Los desarrolladores podrán usar la opción Profile HWUI rendering en las pruebas. En el caso de que se presenten problemas, hay herramientas disponibles que ayudarán a diagnosticar la renderización lenta.	**n/a**
		SD-1		
	PS-P3	PM-1	Cuando StrictMode está activado (consulta Prueba StrictMode a continuación), no se verán destellos rojos (advertencias de rendimiento de StrictMode) durante la prueba de la app. Cualquier destello rojo indica un comportamiento inadecuado en relación con el almacenamiento, el acceso a la red o las fugas de memoria.	**n/a**
SDK	PS-T1	CR-0	La app se ejecutará en la última versión pública de la plataforma de Android sin que se produzca una falla y sin que haya un impacto sobre la funcionalidad principal.	**ok**
	PS-T2	SP-1	La app se orienta al SDK de Android más reciente mediante la configuración del valor targetSdk.	**no**
	PS-T3	SP-1	Se compilará la app con el último SDK estableciendo el valor compileSdk.	**no**
	PS-T4	SP-2	Todos los SDK de terceros usados están actualizados. Cualquier mejora que se realice a estos SDK, como la estabilidad, la compatibilidad o la seguridad, debe estar disponible de forma oportuna para los usuarios.	**ok**
		SP-3	El desarrollador será responsable de toda la base de código de la app, incluidos los SDK de terceros que se usen.	**ok**
	PS-T5	CR-0	La app no usa interfaces que no pertenecen al SDK.	**ok**
Batería	PS-B1	BA-1	La app admite correctamente las funciones de administración de energía que se introdujeron en Android 6.0 (Descanso y App Standby). Cuando se interrumpe la funcionalidad central por la gestión de energía, solo apps calificadas pueden solicitar una exención. Durante el desarrollo, los desarrolladores podrán probar el comportamiento de App Standby y Descanso mediante estos comandos de ADB.	**ok**
			En términos del uso de batería, los desarrolladores podrán usar el generador de perfiles de energía de Android Studio o la herramienta Battery Historian, combinados con el trabajo previsto en segundo plano, a fin de diagnosticar el uso inesperado de la batería.	**ok**

3.18.4 Privacidad y seguridad.

La aplicación deberá administrar de forma segura los datos de usuario y la información personal con el nivel de permisos que resulte adecuado. Véase las tablas 3.9 y 3.10 las métricas a evaluar.

Tabla 3. 9 Métricas de privacidad y seguridad 1.

PRIVACIDAD Y SEGURIDAD				
ÁREA	**ID**	**PRUEBAS**	**DESCRIPCIÓN**	**Resultado**
Permisos	SC-P1	SC-4	La app solicita solo la cantidad mínima absoluta de permisos que necesita a fin de admitir el caso de uso en cuestión. Para algunos permisos, como la ubicación, usa la ubicación aproximada en lugar de la precisa si es posible.	**Ok**
	SC-P2		La app solo deberá solicitar permiso de acceso a datos sensibles (como los SMS, el registro de llamadas o la ubicación) o a servicios que cuesten dinero (como el Teléfono o los SMS) si están directamente relacionados con los casos de uso principales de las apps. Las consecuencias relacionadas con estos permisos deberán divulgarse de manera destacada al usuario.	**Ok**
			Según cómo uses los permisos, puede haber una forma alternativa de cumplir con el caso de uso de tu app sin depender del acceso a la información sensible. Por ejemplo, en lugar de solicitar permisos relacionados con los contactos de un usuario, puede ser más apropiado solicitar acceso mediante un intent implícito.	**Ok**
	SC-P3	CR-0	La app solicita permisos de tiempo de ejecución en contexto, cuando se solicita la funcionalidad, en lugar de hacerlo directamente durante el inicio de la app.	**Ok**
	SC-P4	CR-0	La app deberá diseñar su UX de modo que transmita de forma clara el motivo por el que se necesitan ciertos permisos. Si eso no es posible, la app deberá seguir el flujo recomendado para explicar por qué una función requiere un permiso.	**Ok**
			La app deberá degradarse de manera elegante cuando los usuarios rechacen o revoquen un permiso. La app no deberá impedir que el usuario acceda a ella.	**Ok**
Datos y archivos	SC-DF1	SC-1	Todos los datos sensibles se almacenan en el almacenamiento interno de la app.	**Ok**
	SC-DF2	SC-10	No se registran datos personales o sensibles de los usuarios en el registro del sistema ni en un registro específico de la app.	**Ok**
	SC-DF3		La app no deberá usar ID de hardware que no se puedan restablecer, como el IMEI, para fines de identificación.	**Ok**
Identidad	SC-ID1	CR-0	Brinda sugerencias para autocompletar las credenciales de la cuenta y otra información sensible, como información de tarjetas de crédito, direcciones físicas y números de teléfono.	**Ok**
	SC-ID2	CR-0	Integra One Tap para Android y obtén una experiencia de acceso fluida.	**No**
	SC-ID3	CR-0	Integra la autenticación biométrica a fin de proteger las transacciones financieras y la información sensible, como los documentos importantes de los usuarios.	**N/a**

Tabla 3. 10 Métricas de privacidad y seguridad 2.

PRIVACIDAD Y SEGURIDAD				
ÁREA	**ID**	**PRUEBAS**	**DESCRIPCIÓN**	**Resultado**
Componentes de la app	SC-AC1	SC-5	Solo se exportan los componentes de la aplicación que comparten datos con otras apps o aquellos que otras apps deberían invocar.	**N/a**
			Esto incluye actividades, servicios; receptores de emisión; y, en especial, proveedores de contenido.	
			Siempre define el atributo android:exported de forma explícita a fin de minimizar la confusión sobre el valor predeterminado.	
	SC-AC2	CR-0	Todos los intents y transmisiones siguen las prácticas recomendadas:	**N/a**
		SC-4	1. Usa intents explícitos si la aplicación de destino está bien definida.	
			2. Usa intents a efectos de diferir los permisos a una app diferente que ya cuente con permiso.	
			3. Comparte datos de manera segura entre apps.	
			4. Los intents que contienen una carga útil se verifican antes de su uso.	
			5. Si necesitas pasar un intent a otra app para que la app receptora pueda invocar y esperar una devolución de llamada en la app que realiza la llamada, no incluyas un intent anidado en los extras. Usa un PendingIntent.	
			6. Cuando configures tus PendingIntents, establece explícitamente la marca inmutable si corresponde.	
	SC-AC3	SC-3	Todos los proveedores de contenido que comparten contenido entre tus apps usan android:protectionLevel="signature" para los permisos personalizados. Esto incluye actividades; servicios; receptores de emisión; y, en especial, proveedores de contenido. La mayoría de las apps no deben depender del acceso a una lista de paquetes instalados. A partir de Android 11, se restringió el acceso.	
Redes	SC-N1	SC-9	Todo el tráfico de red se envía mediante SSL.	
	SC-N2	SC-6	La aplicación declara una configuración de seguridad de red.	**No**
	SC-N3		Si la aplicación utiliza los Servicios de Google Play, se inicializará el proveedor de seguridad cuando lo haga la aplicación.	**Ok**
Bibliotecas	SC-U1	SP-2	Todas las bibliotecas, los SDK y las dependencias están actualizados.	**Ok**
	SC-U2		No se incluyen bibliotecas de depuración en la app de producción. Esto puede causar problemas de rendimiento y de seguridad.	**Ok**
WebViews	SC-W1	SC-6	No uses setAllowUniversalAccessFromFileURLs() para acceder al contenido local. En su lugar, usa WebViewAssetLoader.	**N/a**
	SC-W2	SC-7	Las WebViews no deben usar addJavaScriptInterface() con contenido que no sea de confianza.	**N/a**
			En su lugar, en Android 6.0 y versiones posteriores, usa los canales de mensajes HTML.	**N/a**
Ejecución	SC-E1		La app no carga código dinámicamente desde fuera del APK de la app. Los desarrolladores deberán usar Android App Bundles, que incluyen Entrega de funciones en Play y Play Asset Delivery.	**Ok**
			A partir de agosto de 2021, el uso de Android App Bundles será obligatorio para todas las apps nuevas en Google Play Store.	**Ok**
Criptografía	SC-C1		La app usa un generador de números aleatorios y algoritmos criptográficos fuertes y proporcionados por la plataforma. Además, la app no implementa algoritmos personalizados.	**N/a**

3.19 Diseño de interfaz gráfica.

Para el diseño de la interfaz de usuario, se utilizó la herramienta JustinMind (figura 3.11) para su maquetado, de acuerdo a las métricas de diseño de UI proporcionadas por google, la aplicación debe ser lo más sencilla, pero al mismo tiempo intuitiva para el usuario; es por eso que las pantallas de la aplicación deberán constar de simples diseños los cuales muestren solo la información requerida para cada actividad.

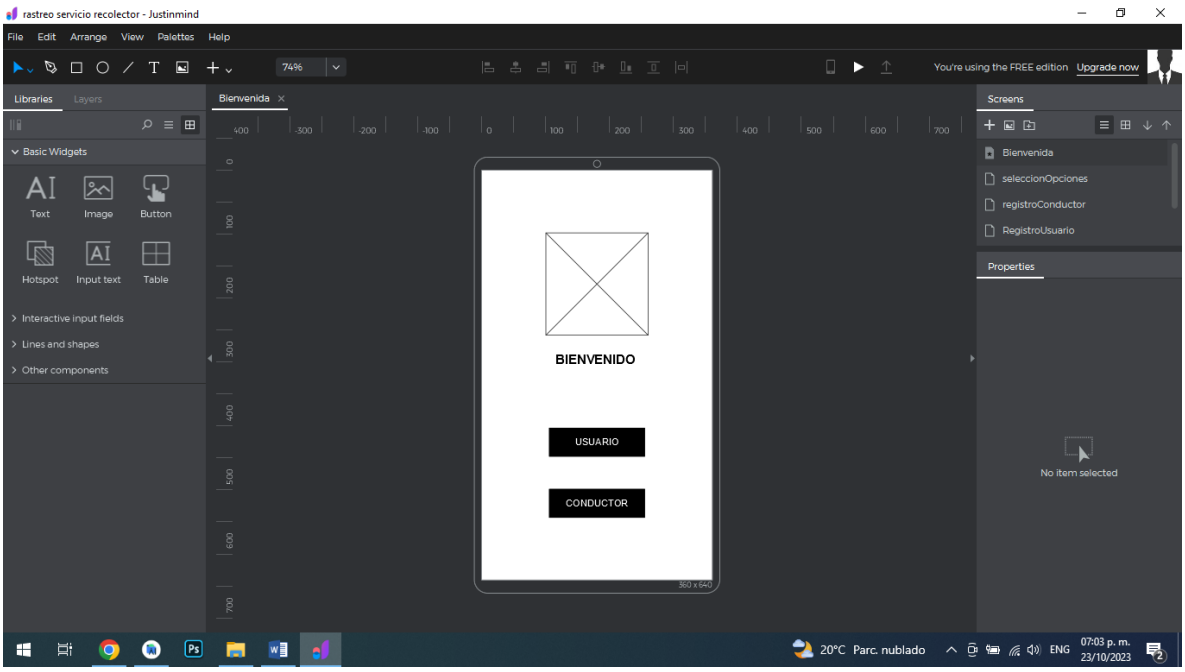


Figura 3. 11 Pantalla principal Justinmind.

3.19.1 Pantalla de bienvenida.

La pantalla de bienvenida es donde se puede seleccionar el tipo de usuario, éste puede seleccionar si será un conductor del vehículo recolector de basura o un ciudadano. El cual quiere dar seguimiento y ser informado sobre la situación actual del servicio de recolección. Esta pantalla solo aparece por única vez hasta que el usuario decida cerrar su sesión como se muestra en la figura 3.12.

Figura 3. 12 Pantalla de inicio.

3.19.2 Selección de registro o iniciar sesión.

Una vez seleccionado el tipo de usuario, la aplicación debe de dar la opción de registro o iniciar sesión si es que el usuario ya se encuentra registrado (figura 3.13).

Figura 3. 13 Selección de registro o inicio de sesión.

3.19.3 Registro de usuarios.

El registro de los usuarios dependerá del tipo de usuario seleccionado en la pantalla principal, ya que cada uno tendrá diferentes campos en la base de datos que correspondan a la información que se requiera saber de ellos. Las diferencias se muestran en las figuras 3.14 y 3.15.

Figura 3. 14 Registro de ciudadano.

Figura 3. 15 Registro de conductor.

3.19.4 Inicio de sesión.

Si los usuarios ya tienen cuenta, la pantalla de login es única para todos los tipos de usuario, ya que solo se requiere autenticar con su correo electrónico y su contraseña, una vez hecho esto la aplicación no volverá a pedirle la información de registro o autenticación; la información quedará almacenada en la configuración de la aplicación haciendo así más sencillo el uso de la misma ya que con solo abrir la aplicación se dirigirá a la pantalla principal véase en la figura 3.16.

Figura 3. 16 Inicio de sesión.

3.19.5 Pantalla principal.

En la pantalla principal es donde se mostrará el rastreo satelital del vehículo recolector de basura, notificaciones, campañas e información general para el ciudadano, y en la versión del conductor solo desplegará su ubicación actual como se muestra en la figura 3.17.

Figura 3. 17 Pantalla principal.

3.20 Desarrollo de interfaz gráfica.

El desarrollo de las pantallas y funcionamiento es desarrollado en Android Studio, tomando como base el maquetado previo. Dando como resultado las pantallas mostradas en la figura 3.18.

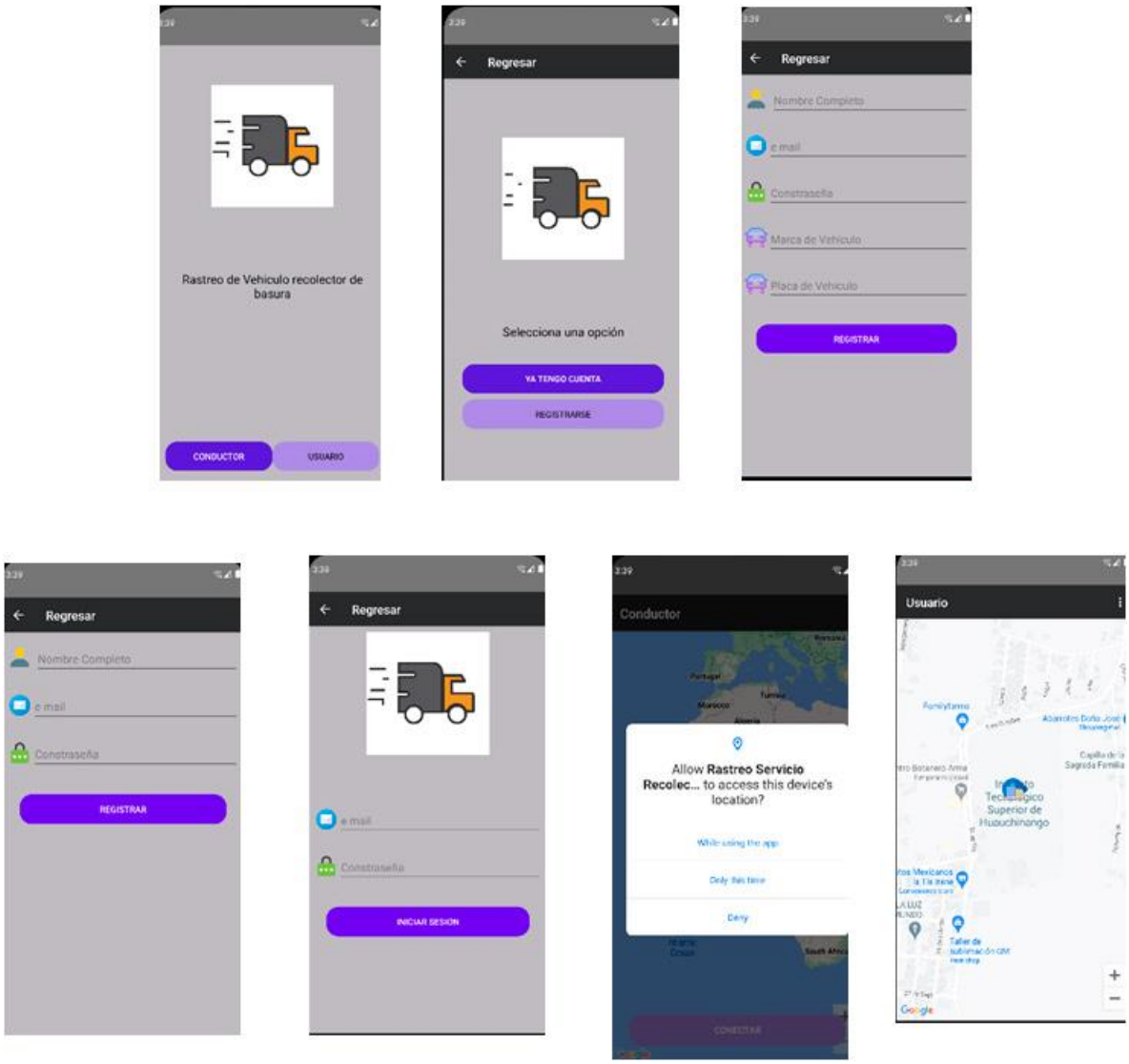


Figura 3. 18 Diagrama de pantallas.

En la siguiente sección se describirá a detalle cada una de las pantallas y la construcción de las interfaces y clases que las conforman.

3.20.1 Pantalla de bienvenida.

La pantalla de bienvenida consta de una animación descargada del sitio LootieFiles, el cual muestra un vehículo en movimiento [25], un textview que da el mensaje de introducción de la aplicación y 2 botones los cuales dirigirán al tipo de usuario correspondiente.

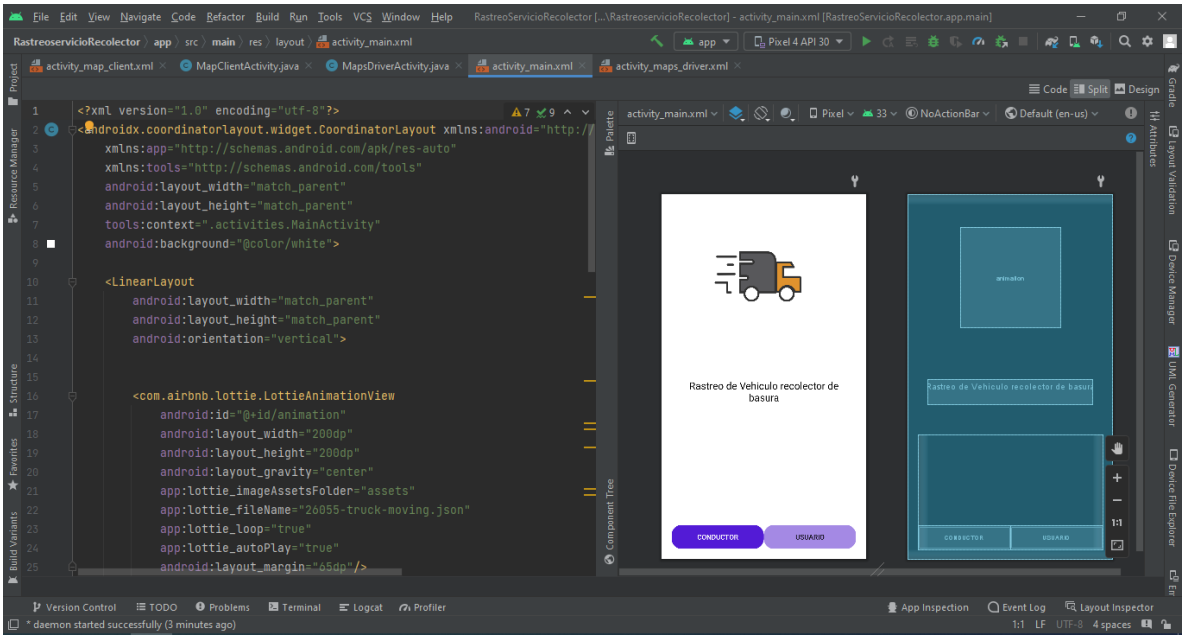


Figura 3. 19 Pantalla de bienvenida (Construcción).

La pantalla debe hacer una selección simple entre si es un ciudadano y un conductor, cada botón tiene su codificación de acción para así poder diferenciar entre los diferentes tipos de usuario.

Lo primero es importar todas las librerías necesarias, estas se importan de manera automática al momento de escribir el código.

```
package com.abconsulting.RastreoServicioRecolector.activities;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
```

```
import android.widget.Button;
import
com.abconsulting.RastreoServicioRecolector.activities.Driver.MapsDriverAc
tivity;
import com.abconsulting.RastreoServicioRecolector.R;
import
com.abconsulting.RastreoServicioRecolector.activities.Client.MapClientAct
ivity;
import com.google.firebase.auth.FirebaseAuth;
```

Se deben declarar variables que serán el enlace entre la clase java y el diseño xml, estas variables serán de el tipo de control que se está ocupando en el diseño.

```
public class MainActivity extends AppCompatActivity {

    Button conductorBtn,usuarioBtn;
    SharedPreferences mpref;
    SharedPreferences.Editor editor;
```

En el método oncreate se inicializan las variables que identifican los controles de el diseño, usando la clase R para identificar los controles en el diseño. Para guardar el tipo de usuario que está ejecutando la aplicación, se utiliza el método sharedpreferences.

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

        conductorBtn=(Button) findViewById(R.id.btn_conductor);
        usuarioBtn=(Button)findViewById(R.id.btn_usuario);

        mpref=
getApplicationContext().getSharedPreferences("typeUser",MODE_PRIVATE);
        editor = mpref.edit();


    }
```

Se hace uso del método onstart para que al momento de iniciar la aplicación se ejecute por única vez este segmento de código, en este método se hace instancia a la base de datos de Firebase y al módulo de autenticación, para saber cuál fue el tipo de usuario que utilizó por última vez la aplicación y poder ejecutar la pantalla principal que le corresponda.

```
    @Override
    protected void onStart(){
        super.onStart();
        if(FirebaseAuth.getInstance().getCurrentUser()!=null){
            String user = mpref.getString("user","");
            Intent intent;
            if(user.equals("client")){
                intent = new Intent(MainActivity.this,
MapClientActivity.class);
            } else{
                intent = new Intent(MainActivity.this,
MapsDriverActivity.class);
            }
            intent.addFlags(intent.FLAG_ACTIVITY_NEW_TASK |
intent.FLAG_ACTIVITY_CLEAR_TASK);
            startActivity(intent);
        }
    }
```

En el caso de que no se tenga la información de algún usuario previo, se ejecuta el método correspondiente por cada botón, usando la variable de tipo sharedpreferences se almacena el tipo de usuario que está ejecutando la aplicación y enseguida mediante un intent se ejecuta la siguiente actividad.

```
    public void goToSelectOptions(View view){
        Intent intent = new
Intent(MainActivity.this,SelectOptionauthactivity.class);
        startActivity(intent);
        editor.putString("user","client");
        editor.apply();
    }

    public void goToSelectasDriver(View view){
        Intent intent = new
Intent(MainActivity.this,SelectOptionauthactivity.class);
        startActivity(intent);
        editor.putString("user","driver");
        editor.apply();
    }

}
```

3.20.2 Pantalla de selección.

La pantalla de selección consta de la misma animación para hacer referencia a un menú de opciones [26], también consta de un textview con el texto de selección y dos botones los cuales dirigirán al usuario a la opción deseada.

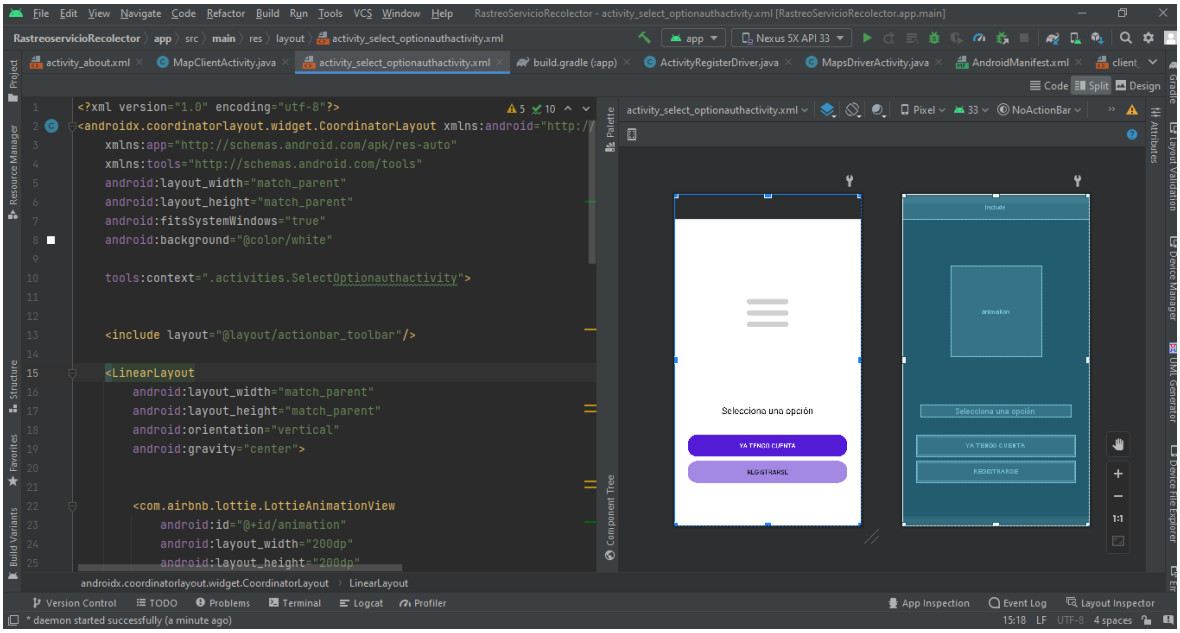


Figura 3. 20 Pantalla de selección (Construcción).

Se importan los paquetes necesarios de manera automática durante la codificación.

```
package com.abconsulting.RastreoServicioRecolector.activities;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import com.abconsulting.RastreoServicioRecolector.R;
import
com.abconsulting.RastreoServicioRecolector.activities.Client.RegisterActi
vity2;

com.abconsulting.RastreoServicioRecolector.activities.Driver.ActivityRegi
sterDriver;
```

```
import com.abconsulting.RastreoServicioRecolector.includes.MyToolbar;
```

Se declaran las variables globales, 2 de tipo botón para interactuar con el diseño y una para instanciar la clase SharedPreferences para obtener el tipo de usuario que está ejecutando la aplicación.

```
public class SelectOptionauthactivity extends AppCompatActivity {


    Button Registradoboton,RegistrarBoton;
    SharedPreferences mpref;
```

En el método onCreate se inicializan las variables haciendo referencia entre el diseño y la clase java con la clase R, se obtiene el valor del objeto SharedPreferences para saber qué tipo de usuario está ejecutando la aplicación y se hace instancia al toolbar que da la opción de regresar a la pantalla anterior.

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

        MyToolbar.show(SelectOptionauthactivity.this,"Regresar",true);

        Registradoboton = (Button) findViewById(R.id.Registrado_btn);
        RegistrarBoton = (Button) findViewById(R.id.Registrarse_btn);
        mpref = getApplicationContext().getSharedPreferences("typeUser",
MODE_PRIVATE);


    }
```

Si se selecciona el botón de inicio de sesión entra el método GotoLogin, este inicia una nueva actividad utilizando le método Intent y si se selecciona el método Gotoregister primero se obtiene el valor del tipo de usuario del objeto de tipo SharedPreferences, seguido se hace

una selección donde dependiendo del tipo de usuario, será la nueva actividad a abrir mediante el método Intent.

```
    public void GotoLogin(View view){
        Intent intent = new
Intent(SelectOptionauthactivity.this,ActivityLogin.class);
        startActivity(intent);
    }

    public void GotoRegister(View view){
        String userType = mpref.getString("user","");
        Intent;
        if(userType.equals("client")){
            intent = new Intent(SelectOptionauthactivity.this,
RegisterActivity2.class);
        }
        else{
            intent = new Intent(SelectOptionauthactivity.this,
ActivityRegisterDriver.class);
        }
        startActivity(intent);

    }



}
```

3.20.3 Pantalla de registro de usuario.

La pantalla de registro de usuario llevará información básica, como nombre [27], email [28], contraseña [29], número celular [30], sexo [31] y colonia [32]. El activity cuenta con 3 textfield, 1 textfield numérico, 2 radiobutton, un spinner y un botón para almacenar toda la información.

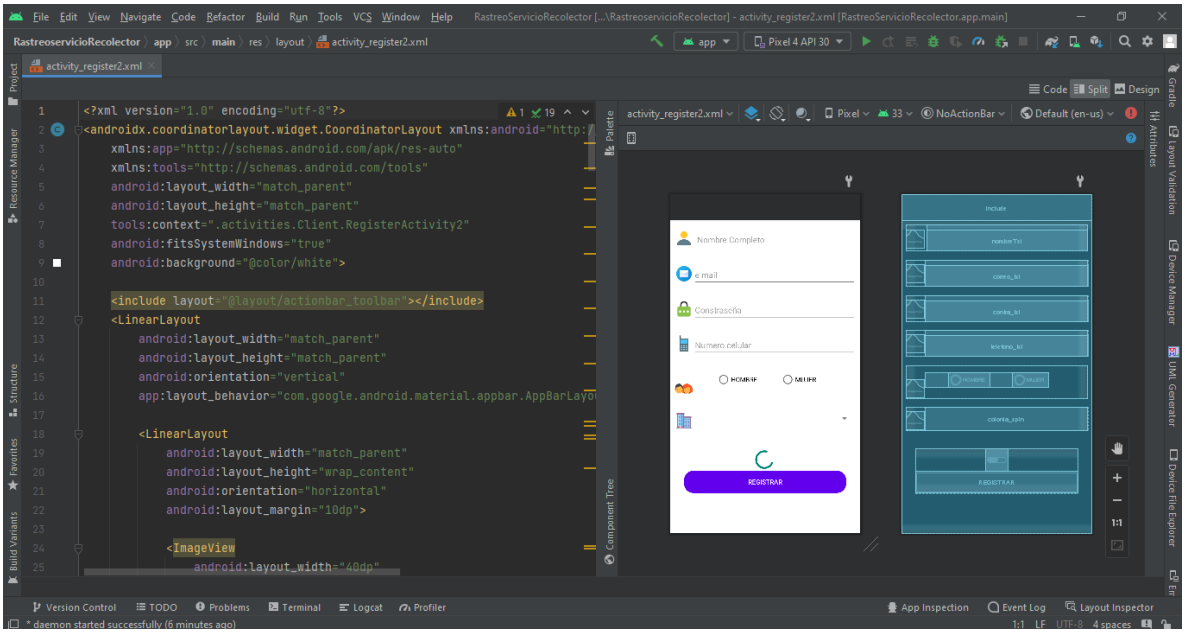


Figura 3. 21 Pantalla de registro de ciudadano (Construcción).

Lo primero que se hace es declarar las variables que serán la conexión entre el diseño y la codificación. Variables del tipo de control que están en el diseño de la actividad seleccionada y que serán las que interactúen con el usuario.

```
public class RegisterActivity2 extends AppCompatActivity {


    Button mButton;
    EditText mail, nombre, pass,tel;
    Authprovider mAuthprovider;
    ClientProvider mClientprovider;
    ColProvider;
    ProgressBar progress;
    //AlertDialog mdialog;
    Spinner coloniasSpinner;
    RadioButton Hradio,Mradio;
```

En el método onCreate se inicializan las variables que van a interactuar con el usuario mediante la clase R, se inicializa el toolbar que tendrá la opción de regresar al activity anterior y se llenará el spinner que corresponde a la lista de las colonias mediante una lista con cada una de las colonias a seleccionar.

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register2);

        MyToolbar.show(RegisterActivity2.this, "Regresar", true);


        mButton = (Button) findViewById(R.id.regBtn);

        nombre = (EditText) findViewById(R.id.nombreTxt);
        mail = (EditText) findViewById(R.id.correo_txt);
        pass = (EditText) findViewById(R.id.contra_txt);
        tel = (EditText) findViewById(R.id.telefono_txt);

        Hradio = (RadioButton) findViewById(R.id.hombre_radio);
        Mradio = (RadioButton) findViewById(R.id.mujer_radio);

        mAuthprovider = new Authprovider();
        mClientprovider = new ClientProvider();
        colProvider= new ColProvider();
        coloniasSpinner=(Spinner)findViewById(R.id.colonia_spin);

        progress = (ProgressBar) findViewById(R.id.progressBar);
        progress.setVisibility(View.GONE);
        //mdialog = new
SpotsDialog.Builder().setContext(RegisterActivity2.this).setMessage("Espe
re un momento").build();

        //LLENADO DE SPINNER PARA COLONIAS
        List<colonia> colonias = colProvider.cargaColonias();
        ArrayAdapter<colonia> arrayAdapter=new
ArrayAdapter<>(getApplicationContext(),
android.R.layout.simple_spinner_dropdown_item,colonias);
        coloniasSpinner.setAdapter(arrayAdapter);
    }
```

Al momento de seleccionar el botón de registrar, se obtienen los valores introducidos por el usuario mediante el método gettext y se almacenan en variables de tipo cadena, para almacenar el sexo se hace una validación sobre que RadioButton está seleccionado y de acuerdo a la selección, se asigna una cadena con el valor correcto. Por último, se hace una validación para verificar que todos los campos hayan sido llenados y en caso de que falte alguno, avisarle al usuario que debe llenar todos los campos para continuar con el registro.

```
    public void clickregister(View view) {
        String name = nombre.getText().toString();
        String correo = mail.getText().toString();
```

```
        String contrasenia = pass.getText().toString();
        String sexoString;
        String telefono = tel.getText().toString();
        String colonia = coloniasSpinner.getSelectedItem().toString();

        if (Hradio.isChecked()==true){
            sexoString = "Hombre";
        } else{
            sexoString="Mujer";
        }

        if (!name.isEmpty() && !correo.isEmpty() &&
!contrasenia.isEmpty() && !colonia.isEmpty()) {
            if (contrasenia.length() >= 6) {
                progress.setVisibility(View.VISIBLE);
                register(name, correo,
contrasenia,colonia,telefono,sexoString);
            } else {
                Toast.makeText(this, "La contraseña debe tener al menos 6
caracteres", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(this, "Ingrese todos los campos",
Toast.LENGTH_SHORT).show();
        }
    }
```

El método register se ejecuta cuando se ha validado que todos los campos han sido llenados de manera correcta, este recibe los valores como parámetros y utilizando el objeto mAuthprovider se hace la validación con los servicios de Firebase en el módulo de autenticación. Si todo es correcto se ejecuta de manera automática el método onComplete donde se valida que la tarea se haya terminado de manera correcta, si es así se guarda el id que Firebase le asigna al usuario en una variable de tipo cadena y se crea un nuevo objeto de tipo cliente, el cual recibe como parámetros los datos del usuario incluyendo el id generado.

```
    private void register(String nombre, String mail, String pass,String
colonia,String tel,String sexo) {
        mAuthprovider.Register(mail, pass).addOnCompleteListener(this,
new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(task.isSuccessful()){
                    String id =
FirebaseAuth.getInstance().getCurrentUser().getUid();
                    Client cliente = new Client(id, nombre,
mail,colonia,tel,sexo);
```

```
                    create(cliente);
                }else{
                    progress.setVisibility(View.GONE);
                    Toast.makeText(RegisterActivity2.this, "No se pudo
crear usuario", Toast.LENGTH_SHORT).show();
                }
            }
        });

    }
```

Por último, se ejecuta el método create este recibe el objeto cliente como parámetro. Utilizando el objeto mClientProvider declarado en el inicio de nuestra clase para agregarlo en la base de datos de Firebase y en el módulo de autenticación. Si todo es correcto automáticamente entra el método onComplete en el cual se valida que la tarea se haya terminado de ejecutar correctamente, si esto es así se muestra un mensaje al usuario que este fue creado correctamente y mediante la clase intent se hace el llamado a la siguiente pantalla que es el mapa del cliente, en caso de que haya ocurrido algún error durante la ejecución de la creación, se muestra un mensaje al cliente el cual le avisa que el usuario no pudo ser creado.

```
private void create(Client client) {

        mClientprovider.create(client).addOnCompleteListener(this, new
OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()){
                    progress.setVisibility(View.GONE);
                    Toast.makeText(RegisterActivity2.this, "Usuario
Creado correctamente", Toast.LENGTH_SHORT).show();
                    Intent = new Intent(RegisterActivity2.this,
MapClientActivity.class);
                    intent.addFlags(intent.FLAG_ACTIVITY_NEW_TASK |
intent.FLAG_ACTIVITY_CLEAR_TASK);
                    startActivity(intent);
                }else{
                    progress.setVisibility(View.GONE);
                    Toast.makeText(RegisterActivity2.this, "No se creo
usuario en base de datos", Toast.LENGTH_SHORT).show();
                }
            }
        });
    }
}
```

3.20.4 Pantalla de registro de conductor.

El registro del conductor tiene más campos que el ciudadano común, ya que este si requiere llevar más datos como son: nombre, email, contraseña, sexo, número celular, tipo de sangre [33], número de licencia [34], número de seguro social [35] y colonia. Para eso la pantalla debe contar con 5 textfield, 2 radiobutton, 1 textfield numérico, 1 spinner para selección de colonia y un botón para aceptar y guardar los cambios.

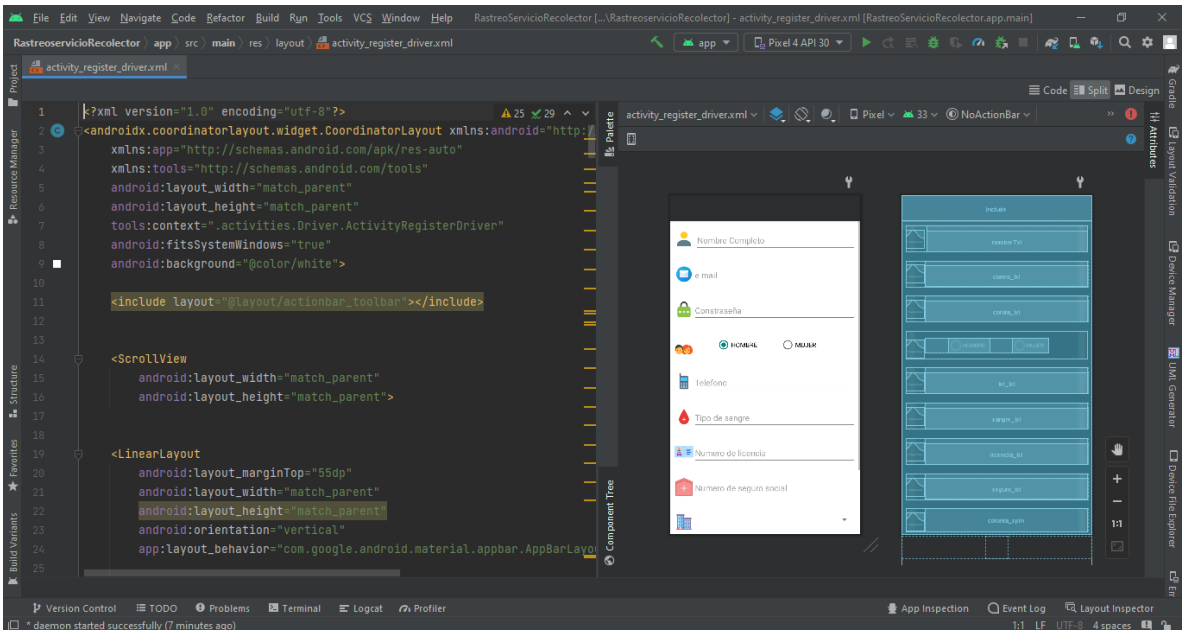


Figura 3. 22 Pantalla de registro de conductor (Construcción).

En primera instancia se declararán las variables que serán la conexión entre el diseño y la codificación. Variables del tipo de control que están en el diseño de la actividad seleccionada y que serán las que interactúen con el conductor.

```
public class ActivityRegisterDriver extends AppCompatActivity {

    Button mButton;
    EditText mail, nombre, pass,telefono,tiposangre,licencia,seguro;
    Authprovider mAuthprovider;
    Driverprovider mDriverprovider;
    ColProvider;
```

```
    ProgressBar progress;
    AlertDialog mdialog;
    Spinner coloniasSpinner;
    RadioButton hombre,mujer;
```

En el método onCreate se inicializan las variables que van a interactuar con el usuario mediante la clase R, se inicializa el toolbar que tendrá la opción de regresar al activity anterior y se llenará el spinner que corresponde a la lista de las colonias mediante una lista con cada una de las colonias a seleccionar.

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

        MyToolbar.show(ActivityRegisterDriver.this, "Regresar", true);


        mButton = (Button) findViewById(R.id.regBtn);

        nombre = (EditText) findViewById(R.id.nombreTxt);
        mail = (EditText) findViewById(R.id.correo_txt);
        pass = (EditText) findViewById(R.id.contra_txt);
        telefono = (EditText) findViewById(R.id.tel_txt);
        tiposangre = (EditText) findViewById(R.id.sangre_txt);
        licencia=(EditText)findViewById(R.id.licencia_txt);
        seguro=(EditText)findViewById(R.id.seguro_txt);
        hombre=(RadioButton)findViewById(R.id.hombre_radio);
        mujer = (RadioButton)findViewById(R.id.mujer_radio);

        mAuthprovider = new Authprovider();
        mDriverprovider = new Driverprovider();
        colProvider= new ColProvider();
        coloniasSpinner=(Spinner)findViewById(R.id.colonia_spin);

        progress = (ProgressBar) findViewById(R.id.progressBar);
        progress.setVisibility(View.GONE);

        //mdialog = new
SpotsDialog.Builder().setContext(RegisterActivity2.this).setMessage("Espe
re un momento").build();

        //LLENADO DE SPINNER PARA COLONIAS
        List<colonia> colonias = colProvider.cargaColonias();
        ArrayAdapter<colonia> arrayAdapter=new
ArrayAdapter<>(getApplicationContext(),
android.R.layout.simple_spinner_dropdown_item,colonias);
        coloniasSpinner.setAdapter(arrayAdapter);

    }
```

Al momento de seleccionar el botón de registrar, se obtienen los valores introducidos por el usuario mediante el método gettext y se almacenan en variables de tipo cadena, para almacenar el sexo se hace una validación sobre que RadioButton está seleccionado y de acuerdo a la selección, se asigna una cadena con el valor correcto. Por último, se hace una validación para verificar que todos los campos hayan sido llenados y en caso de que falte alguno, avisarle al usuario que debe llenar todos los campos para continuar con el registro.

```
    public void clickregister(View view) {
        String name = nombre.getText().toString();
        String correo = mail.getText().toString();
        String contrasenia = pass.getText().toString();
        String TelefonoString = telefono.getText().toString();
        String sangreString = tiposangre.getText().toString();
        String licenciaString = licencia.getText().toString();
        String seguroString = seguro.getText().toString();

        String colonia = coloniasSpinner.getSelectedItem().toString();

        if (!name.isEmpty() && !correo.isEmpty() &&
!contrasenia.isEmpty() && !TelefonoString.isEmpty() &&
!sangreString.isEmpty() && !colonia.isEmpty() &&
!licenciaString.isEmpty() && !seguroString.isEmpty()) {
            if (contrasenia.length() >= 6) {
                progress.setVisibility(View.VISIBLE);
                register(name, correo,
contrasenia,TelefonoString,sangreString,colonia,licenciaString,seguroStri
ng);

            } else {
                Toast.makeText(this, "La contraseña debe tener al menos 6
caracteres", Toast.LENGTH_SHORT).show();
            }
        } else {
            Toast.makeText(this, "Ingrese todos los campos",
Toast.LENGTH_SHORT).show();
        }
    }
```

Por último, se ejecuta el método register, Utilizando el objeto mAuthProvider declarado en el inicio de nuestra clase para agregarlo en la base de datos de Firebase y en el módulo de autenticación. Si todo es correcto automáticamente entra el método onComplete en el cual se valida que la tarea se haya terminado de ejecutar correctamente, si esto es así se obtiene el id generado por Firebase y se almacena en una variable de tipo cadena, la cual es utilizada para la creación del objeto conductor y se manda llamar el método create.

```
private void register(String nombre, String mail, String pass,String
telefono, String sangre,String colonia,String licencia, String seguro) {
            mAuthprovider.Register(mail,
pass).addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                @Override
                public void onComplete(@NonNull Task<AuthResult> task) {
                    if (task.isSuccessful()){
                        String id =
FirebaseAuth.getInstance().getCurrentUser().getUid();
                        Driver conductor = new Driver(id, nombre,
mail,telefono,sangre,colonia,licencia,seguro);
                        create(conductor);
                    }else{
                        progress.setVisibility(View.GONE);
                        Toast.makeText(ActivityRegisterDriver.this, "No
se pudo crear usuario", Toast.LENGTH_SHORT).show();
                    }
                }
            });


    }
```

En el método create utilizando el objeto mDiverprovider se hace la inserción del registro en la base de datos de Firebase y en el módulo de autenticación. Si las instrucciones se ejecutan de manera correcta en el servidor, muestra un mensaje al usuario que este fue creado correctamente y mediante la clase intent se hace el llamado a la siguiente pantalla que es el mapa del conductor, en caso de que haya ocurrido algún error durante la ejecución de la creación, se muestra un mensaje al cliente el cual le avisa que el usuario no pudo ser creado.

```
private void create(Driver driver) {
        mDriverprovider.create(driver).addOnCompleteListener(this, new
OnCompleteListener<Void>() {
            @Override
            public void onComplete(@NonNull Task<Void> task) {
                if (task.isSuccessful()){
                    progress.setVisibility(View.GONE);
                    Toast.makeText(ActivityRegisterDriver.this, "Usuario
Creado correctamente", Toast.LENGTH_SHORT).show();
                    Intent intent = new
Intent(ActivityRegisterDriver.this, MapsDriverActivity.class);
                    intent.addFlags(intent.FLAG_ACTIVITY_NEW_TASK |
intent.FLAG_ACTIVITY_CLEAR_TASK);
                    startActivity(intent);
                } else{
                    progress.setVisibility(View.GONE);
                    Toast.makeText(ActivityRegisterDriver.this, "No se
pudo crear usuario en base de datos", Toast.LENGTH_SHORT).show();}}});}
```

3.20.5 Pantalla de inicio de sesión.

La autenticación en la base de datos solo será con el correo electrónico registrado y la contraseña registrada para el conductor se muestra una pantalla previa para validar que sea trabajador, se muestra una animación de seguridad [36] y se pide la contraseña proporcionada por el departamento de limpia [37]. Por tal motivo solo se requiere introducir dichos datos y validar en la base de datos y el módulo de autenticación de Firebase que el usuario este registrado de manera correcta. La pantalla cuenta con 2 textfield, la animación principal y un botón para actuar cuando el usuario haya llenado los datos de inicio de sesión.

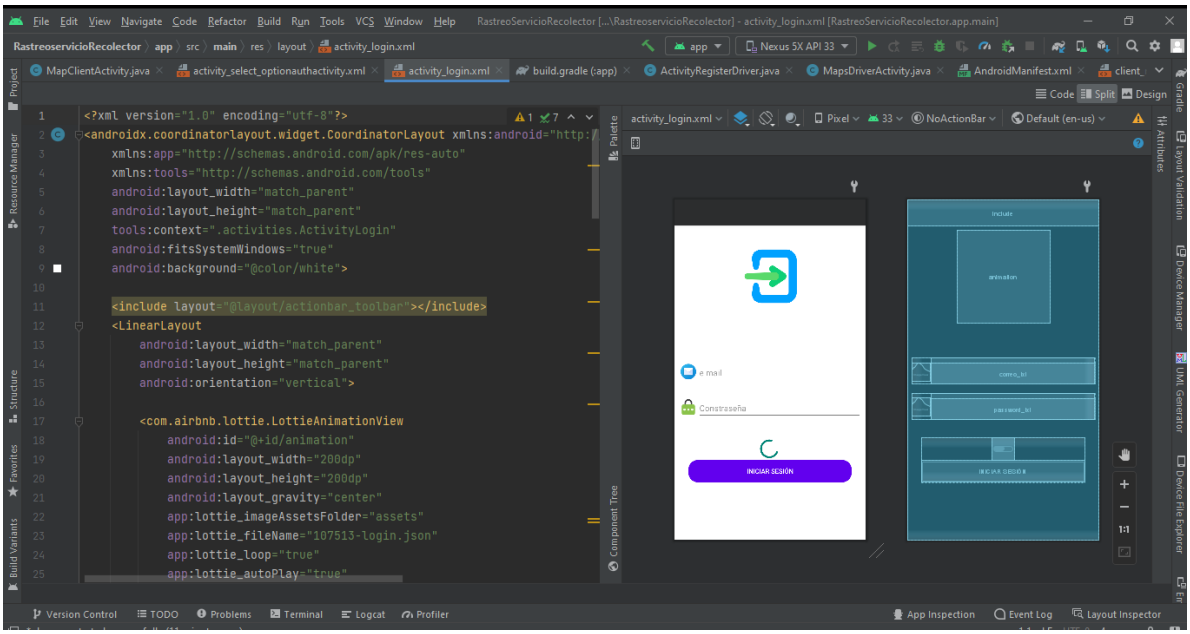


Figura 3. 23 Pantalla de inicio de sesión (Construcción).

Se declaran las variables del tipo de objeto que se encuentra en el diseño, y las variables que hacen referencia a las clases Firebase Auth, DatabaseReference y sharedPreferences

```
public class ActivityLogin extends AppCompatActivity {

    EditText Correo,Password;
    Button login;
    FirebaseAuth mAuth;
    DatabaseReference mDatabase;
```

```
    SharedPreferences mpref;
    ProgressBar progress;
    //AlertDialog mdialog;
```

Seguido, en el método onCreate se inicializan las variables haciendo la conexión con el diseño mediante la clase R, y se generan las instancias con las clases Firebase Auth, DatabaseReference y sharedPreferences para poder hacer uso de los métodos que ayudarán a hacer la conexión con la base de datos y el módulo de autenticación de Firebase.

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

        Correo = (EditText) findViewById(R.id.correo_txt);
        Password = (EditText) findViewById(R.id.password_txt);
        login = (Button) findViewById(R.id.login_btn);
        mAuth = FirebaseAuth.getInstance();
        mDatabase = FirebaseDatabase.getInstance().getReference();
        progress = (ProgressBar) findViewById(R.id.progressBar);
        progress.setVisibility(View.GONE);
        //mdialog = new
SpotsDialog.Builder().setContext(ActivityLogin.this).setMessage("Espere
un momento").build();

        mpref = getApplicationContext().getSharedPreferences("typeUser",
MODE_PRIVATE);

        MyToolbar.show(ActivityLogin.this,"Regresar",true);

    }
```

El método login se ejecuta cuando el usuario presiona el botón correspondiente, primero se obtiene el valor tecleado por el usuario almacenándolos en variables de tipo String. Seguido, se valida que ninguna de las variables tenga un valor vacío, después se valida que la longitud de la contraseña sea mayor a 6 caracteres, si todo es correcto se muestra el control de progress para notificarle al usuario que la aplicación está trabajando y haciendo uso de la clase mAuth se hace la conexión al módulo de autenticación de Firebase usando como parámetros el email y la contraseña que escribió el usuario.

```
    public void Login(View view){

        String emailString = Correo.getText().toString();
        String passwordString = Password.getText().toString();

        if(!emailString.isEmpty() && !passwordString.isEmpty()){
            if (passwordString.length()>=6){
                progress.setVisibility(View.VISIBLE);
                //mAuth.signInWithEmailAndPassword(emailString,
passwordString);

mAuth.signInWithEmailAndPassword(emailString,passwordString).addOnComplet
eListener(this, new OnCompleteListener<AuthResult>() {
```

Si la autenticación fue correcta se obtiene el tipo de usuario utilizando la clase mpref y dependiendo del tipo de usuario, se abre la actividad correspondiente mediante un intent, por último, se oculta el objeto progress. En caso de que la autenticación sea fallida, se muestra un mensaje al usuario notificándole que este no existe.

```
                    @Override
                    public void onComplete(@NonNull Task<AuthResult>
task) {
                      if (task.isSuccessful()){
                          Toast.makeText(ActivityLogin.this, "Login
Exitoso", Toast.LENGTH_SHORT).show();
                          String userType = mpref.getString("user","");
                          Intent intent;
                          if(userType.equals("client")){
                              intent = new Intent(ActivityLogin.this,
MapClientActivity.class);
                          } else{
                              intent = new Intent(ActivityLogin.this,
MapsDriverActivity.class);
                          }
                          progress.setVisibility(View.GONE);
                          intent.addFlags(intent.FLAG_ACTIVITY_NEW_TASK |
intent.FLAG_ACTIVITY_CLEAR_TASK);
                          startActivity(intent);
                      } else{
                          progress.setVisibility(View.GONE);
                          Toast.makeText(ActivityLogin.this, "Usuario no
existe", Toast.LENGTH_SHORT).show();
                        }
                    }
                });

            }
            else
            {
                Toast.makeText(ActivityLogin.this, "La contrtaseña debe
```

```
de tener mas de 6 digitos", Toast.LENGTH_SHORT).show();
            }

        }
        else
        {
            Toast.makeText(ActivityLogin.this, "El usuario debe escribir
un usuario y contrtaseña", Toast.LENGTH_SHORT).show();
        }
    }
}
```

3.20.6 Pantalla principal.

La pantalla principal es muy parecida para ambos tipos de usuario, sin embargo para los conductores únicamente se encargara de mostrar su ubicación actual en tiempo real con un icono cuando está en movimiento [38] y otro icono cuando tiene más de 1 minuto en el mismo lugar [39], Por último mandar hacia la base de datos todo el tiempo su ubicación actual, a diferencia del ciudadano que su pantalla principal mostrará su ubicación actual [40], la ubicación actual de todos los conductores que se encuentren en línea, notificaciones de cuando el conductor ya se encuentre en un radio de 1 km de distancia y notificará también si hay algún aviso directo de los administradores del servicio de limpia.

Ambas pantallas constan de un control google maps, y un menú superior desplegable. Sin embargo, en la pantalla del conductor también se le agrega un botón de acción el cual conectará o desconectará al conductor para que este aparezca o desaparezca del mapa.

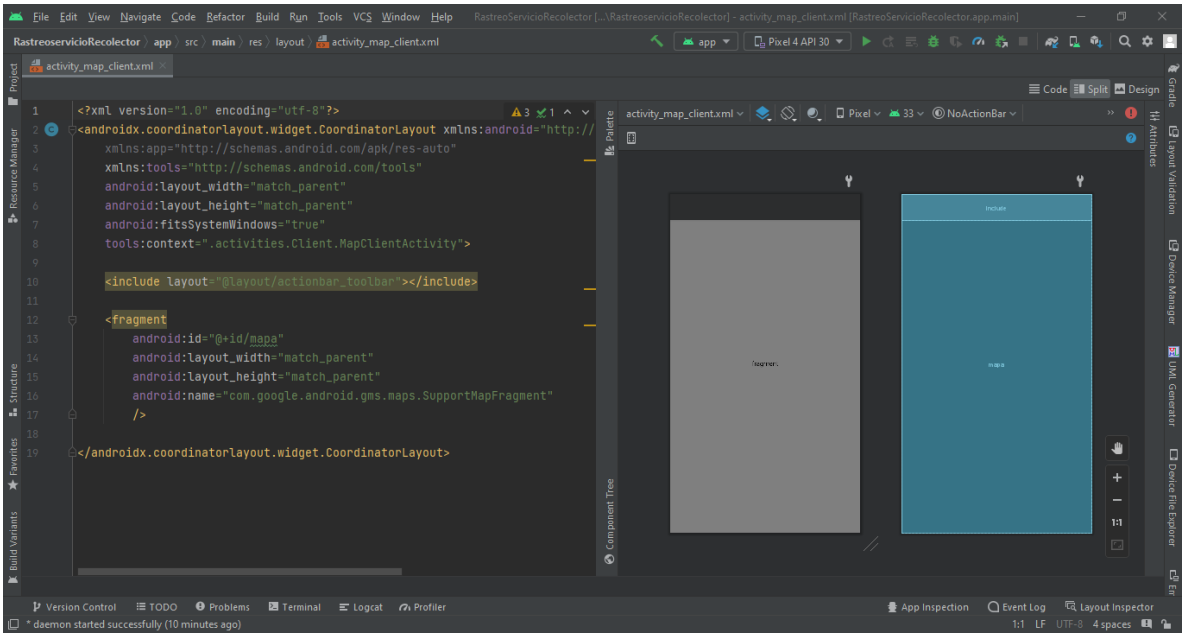


Figura 3. 24 Pantalla principal usuario (Construcción).

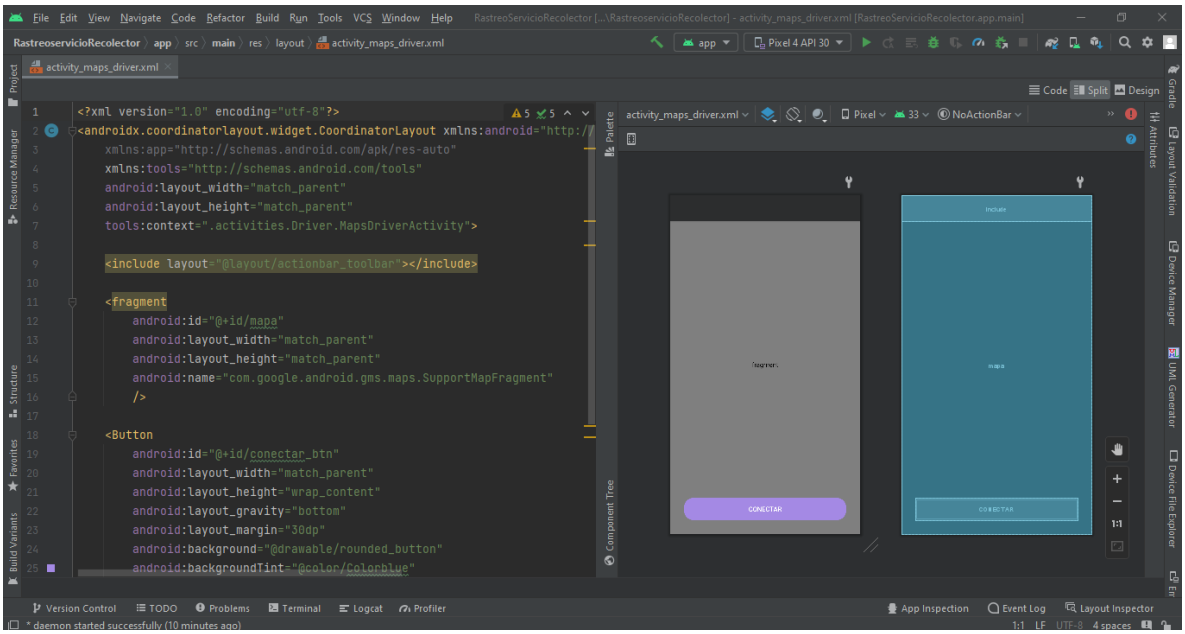


Figura 3. 25 Pantalla principal conductor (Construcción).

En el constructor de la clase se declaran las variables globales para hacer referencia a las clases que harán la conexión a la base de datos, así como el llamado a los servicios de google maps para el rastreo vehicular. En esta ocasión no se hará inicialización de objetos que interactúen con el usuario ya que este no tendrá interacción más que con el control de mapas.

```
public class MapClientActivity extends AppCompatActivity implements
OnMapReadyCallback {

    private GoogleMap mMap;
    private SupportMapFragment mMapfragment;
    private Authprovider mAuthprovider;
    private LocationRequest mLocationRequest;
    private FusedLocationProviderClient mFusedlocation;

    private final static int LOCATION_REQUEST_CODE = 1;
    private final static int SETTING_REQUEST_CODE = 2;

    private Marker mMarker;
    private LatLng mCurrentLatLong;
    private GeofireProvider mGeofireProvider;
    private TokenProvider mTokenProvider;

    private List<Marker> mDriversMarkers = new ArrayList<>();
    private boolean mIsFirstTime=true;

    private double mRadius = 0.1f;

    private String ChannelID="canal";
    private PendingIntent pendingIntent;

    private boolean notificationSowed = false;
```

En el mismo constructor, se hace una llamada en respuesta al evento mLocationCallback este da una respuesta cada que se detecta un movimiento del dispositivo en tiempo real, al detectarse por primera vez, se quita el marcador que pone el control de google maps por default, para en su lugar poner el marcador personalizado. En este caso será el icono del usuario mostrando la ubicación actual inicial y se manda a llamar el método para obtener a los conductores activos.

```
    LocationCallback mLocationCallback = new LocationCallback() {
        @Override
        public void onLocationResult(@NonNull LocationResult
locationResult) {
            super.onLocationResult(locationResult);

            for (Location location : locationResult.getLocations()) {
                if (getApplicationContext() != null) {
```

```
                    if (mMarker != null){
                        mMarker.remove();
                    }
                    mCurrentLatLong = new
LatLng(location.getLatitude(),location.getLongitude());
                    // AGREGAMOS EL MARCADOR PERSONALIZADO EN ESTE CASO
EL ICONO ES DESCARGADO DE https://iconos8.es/icons/set/map-pin
                    mMarker = mMap.addMarker(new
MarkerOptions().position(
                                new
LatLng(location.getLatitude(),location.getLongitude())
                    ).title("Posicion
actual").icon(BitmapDescriptorFactory.fromResource(R.drawable.usuario_ico
no_marcador)));

                    //OBTENER LA LOCALIZACION DEL USUARIO EN TIEMPO REAL

mMap.moveCamera(CameraUpdateFactory.newCameraPosition(new
CameraPosition.Builder().target(new LatLng(location.getLatitude(),
location.getLongitude()))
                                        .zoom(17f)
                                        .build()
                                )
                    );
                    if(mIsFirstTime){
                        mIsFirstTime=false;
                        getActiveDrivers();
                    }

                }

            }
        }
    };
```

En el método onCreate se hace la inicialización de las variables y la instancia a las clases que se mandan llamar, así como la generación por única vez del token de la aplicación.

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

        mAuthprovider = new Authprovider();
        mGeofireProvider = new GeofireProvider();
        mFusedlocation =
LocationServices.getFusedLocationProviderClient(this);
        mTokenProvider = new TokenProvider();

        mMapfragment = (SupportMapFragment)
```

```
getSupportFragmentManager().findFragmentById(R.id.mapa);
        mMapfragment.getMapAsync(this);
        MyToolbar.show(this, "Usuario", false);

        generarToken();
    }
```

El método getActiveDrivers devuelve en una lista, todos los conductores que se encuentran activos en la base de datos. Por cada conductor encontrado, se agrega un marcador personalizado, en este caso se agrega un camión de basura al mapa. Utilizando las coordenadas que se encuentran almacenadas en la base de datos.

```
    private void getActiveDrivers(){

mGeofireProvider.getActiveDrivers(mCurrentLatLong,5).addGeoQueryEventList
ener(new GeoQueryEventListener() {
            @Override
            public void onKeyEntered(String key, GeoLocation location) {
                //AGREGAREMOS LOS MARCADORES DE LOS CONDUCTORES QUE SE
CONECTEN A LA APLICACION
                for(Marker marker:mDriversMarkers){
                    if(marker.getTag()!=null){
                        if(marker.getTag().equals(key)){
                            return;
                        }
                    }
                }
                LatLng driverLatlong = new
LatLng(location.latitude,location.longitude);
                Marker marker = mMap.addMarker(new
MarkerOptions().position(driverLatlong).title("Vehiculo
recolector").icon(BitmapDescriptorFactory.fromResource(R.drawable.camioni
cono)));
                marker.setTag(key);
                mDriversMarkers.add(marker);
            }
```

Cuando se detecta que un conductor se ha desconectado de su aplicación, se vuelve a recorrer la lista de marcadores, para buscar y eliminar de la misma el marcador y las coordenadas del conductor que se ha salido.

```
            @Override
            public void onKeyExited(String key) {
                //QUITAMOS LOS MARCADORES DE LOS CONDUCTORES QUE SE
DESCONECTEN A LA APLICACION
                for(Marker marker:mDriversMarkers){
                    if(marker.getTag()!=null){
                        if(marker.getTag().equals(key)){
                            marker.remove();
                            mDriversMarkers.remove(marker);
                            return;
                        }
                    }
                }
            }
```

Para actualizar la posición de los conductores, en el evento onKeymoved se actualizan las nuevas coordenadas y se manda llamar el método getCloresDriver que mostrará todos los conductores cercanos en un radio de 100 metros.

```
            @Override
            public void onKeyMoved(String key, GeoLocation location) {
                //ACTUALIZAR LA POSICION DE CADA CONDIUCTOR
                for(Marker marker:mDriversMarkers){
                    if(marker.getTag()!=null){
                        if(marker.getTag().equals(key)){
                            marker.setPosition(new
LatLng(location.latitude,location.longitude));
                            getCloserDriver();
                        }
                    }
                }
            }

            @Override
            public void onGeoQueryReady() {

            }

            @Override
            public void onGeoQueryError(DatabaseError error) {

            }
        });
    }
```

Para obtener los conductores cercanos se hace uso de la clase mGeoProvider enviando como parámetros la ubicación actual del usuario y el radio en el que se quiere mostrar los conductores próximos al usuario, en cuanto la clase nos dé una respuesta onKeyEneterd retornará los conductores que estén dentro del radio especificado. Primero se valida una variable booleana la cual especifica si ya se ha notificado al usuario anteriormente de que el vehículo recolector ya se encuentra cerca, eso se usa ya que el método se manda llamar cada que se detecta cualquier movimiento de los conductores, siendo así el evento se mandaría llamar N veces de manera continua. Por tal motivo se debe limitar a solo 1 llamado al método.

```
//METODO PARA REVISAR SI HAY ALGUN CONDUCTOR BASTANTE CERCANO PARA
NOTIFICAR AL USUARIO

    private void getCloserDriver(){

mGeofireProvider.getActiveDrivers(mCurrentLatLong,mRadius).addGeoQueryEve
ntListener(new GeoQueryEventListener() {
            @Override
            public void onKeyEntered(String key, GeoLocation location) {
                if(notificationSowed==false){
                    sendNotification();
                    notificationSowed=true;
                }

            }

            @Override
            public void onKeyExited(String key) {

            }

            @Override
            public void onKeyMoved(String key, GeoLocation location) {

            }

            @Override
            public void onGeoQueryReady() {

            }

            @Override
            public void onGeoQueryError(DatabaseError error) {

            }
        });
    }
```

Al inicializarse el control de google maps entra automáticamente el evento onMapReady aquí se configura el mapa y se inicializa la variable mMap que es el objeto que se utilizará para la interacción con el control de mapas.

```
    @Override
    public void onMapReady(@NonNull GoogleMap googleMap) {
        mMap = googleMap;
        mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
        mMap.getUiSettings().setZoomControlsEnabled(true);

        mMap.addMarker(new MarkerOptions()
                .position(new LatLng(0, 0))
                .title("Marker"));
        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);

//mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        mLocationRequest.setPriority(Priority.PRIORITY_HIGH_ACCURACY);
        mLocationRequest.setSmallestDisplacement(5);

        startLocation();
    }
```

Al ejecutarse la aplicación también se ejecuta de manera automática el método onRequestPermissionsResult, en este método se revisan los permisos de la aplicación, en caso de que la aplicación aun no tenga los permisos otorgados por el usuario para hacer uso del GPS del dispositivo, le manda una notificación al usuario la cual le pide conceder los permisos necesarios para que la aplicación se ejecute de manera correcta.

En caso de que el usuario otorgue o ya haya otorgado con anterioridad los permisos, se manda llamar el método gpsActivated para verificar que el dispositivo GPS se encuentre encendido en el teléfono celular.

```
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
```

```
grantResults);
        if (requestCode == LOCATION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                if
(ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
                    if(gpsActived()){

mFusedlocation.requestLocationUpdates(mLocationRequest,
mLocationCallback, Looper.myLooper());
                    }
                    else{
                        showAlertdialogNOGPS();
                    }

                } else {
                    checkLocationPermissions();
                }
            } else {
                checkLocationPermissions();
            }
        }

    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == SETTING_REQUEST_CODE && gpsActived()) {
            if
(ActivityCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                return;
            }
            mFusedlocation.requestLocationUpdates(mLocationRequest,
mLocationCallback, Looper.myLooper());
        }
        else{
            showAlertdialogNOGPS();
        }
    }
```

El siguiente método se usa para construir el AlertDialog, personalizarlo y ejecutar la acción correspondiente a la respuesta del usuario. En este caso si el usuario decide activar su GPS

desde esta alerta, automáticamente se abre la configuración del sistema para que pueda encender su GPS.

```
    private void showAlertdialogNOGPS(){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Por favor activa tu GPS para continuar")
                .setPositiveButton("Configuraciones", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int
which) {
                        startActivityForResult(new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS),SETTING_REQUEST_CODE);
                    }
                })
                .create().show();
    }

    private boolean gpsActived(){
        boolean isActive = false;
        LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);


if(locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)){
            isActive= true;
        }
        return isActive;
    }
```

Para iniciar la localización de los vehículos, primero se hace una revisión de que versión de Android tiene el dispositivo ya que en las versiones anteriores a Android OREO la codificación se ejecuta de diferente manera para las versiones previas.

```
    private void startLocation(){
        if(Build.VERSION.SDK_INT>=Build.VERSION_CODES.M){
            if(ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION)==PackageManager.PERMISSION_GRAN
TED){
                if(gpsActived()){

mFusedlocation.requestLocationUpdates(mLocationRequest,mLocationCallback,
Looper.myLooper());
                } else{
```

```
                    showAlertdialogNOGPS();
                }

            }
            else{
                checkLocationPermissions();
            }
        }
        else{
            if(gpsActived()){

mFusedlocation.requestLocationUpdates(mLocationRequest,mLocationCallback,
Looper.myLooper());
            }else{
                showAlertdialogNOGPS();
            }
        }
    }

    private void checkLocationPermissions(){
        if(ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION)!=PackageManager.PERMISSION_GRAN
TED){

if(ActivityCompat.shouldShowRequestPermissionRationale(this,Manifest.perm
ission.ACCESS_FINE_LOCATION)){
                new AlertDialog.Builder(this)
                        .setTitle("Proporciona los permisos para
continuar").setMessage("Esta aplicacion requiere de los permisos de
ubicacion para utilizarse")
                        .setPositiveButton("OK", new
DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog,
int which) {

ActivityCompat.requestPermissions(MapClientActivity.this,new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},LOCATION_REQUEST_CODE)
;
                            }
                        })
                        .create()
                        .show();
            }
            else {

ActivityCompat.requestPermissions(MapClientActivity.this,new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},LOCATION_REQUEST_CODE)
;
            }
        }
    }
```

En la parte superior de la pantalla se agrega un menú desplegable, el cual tiene las opciones de información de usuario, información de la programación de la semana para el servicio de recolección de basura y la opción de cerrar sesión.

```
    //AGREGAMOS EL MENU DESPLEGABLE EN LA PARTE SUPERIOR DE LA PANTALLA
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.client_menu,menu);
        return super.onCreateOptionsMenu(menu);
    }
    //ACCIONES A TOMAR DE ACUERDO A LA OPCION SELECCIONADA DEL MENU
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if(item.getItemId()==R.id.action_logout){
            salir();
        }
        if(item.getItemId()==R.id.action_info){
            Intent intent = new
Intent(MapClientActivity.this,InfoUsuario.class);
            startActivity(intent);
        }
        if(item.getItemId()==R.id.action_routes){
            Intent intent = new
Intent(MapClientActivity.this,RutasActivity.class);
            startActivity(intent);
        }
        return super.onOptionsItemSelected(item);
    }
```

Si el usuario selecciona salir, se hace la desconexión de la clase mFusedLocarions para dejar de enviar las actualizaciones de ubicación actual, también se manda llamar el método que permitirá cerrar la sesión en la base de datos y el módulo de autenticación de Firebase, por último, mediante un Intent se regresa a la pantalla de bienvenida y se finaliza el proceso de la pantalla con el mapa de google.

```
    private void salir(){
        desconectar();
        mAuthprovider.logout();
        Intent intent = new Intent(MapClientActivity.this,
MainActivity.class);
        startActivity(intent);
        finish();
```

```
    }

    private void infoUsuario(){

    }

    private void desconectar(){
        mFusedlocation.removeLocationUpdates(mLocationCallback);
    }

    void generarToken(){
        mTokenProvider.create(mAuthprovider.getID());
    }
```

Para poder mostrar las notificaciones al usuario, también se debe de hacer una revisión sobre que versión de Android se está corriendo la aplicación. Ya que también la manera de mostrar las notificaciones es diferente entre las versiones anteriores a Oreo.

```
    private void sendNotification(){

        if (Build.VERSION.SDK_INT>= Build.VERSION_CODES.O){
            showNotificacion();
        } else{
            showNewNotificacion();
        }

    }

    @RequiresApi(api = Build.VERSION_CODES.O)
    private void showNotificacion(){
        NotificationChannel channel = new
NotificationChannel(ChannelID,"NEW",
NotificationManager.IMPORTANCE_HIGH);
        NotificationManager manager = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE);
        manager.createNotificationChannel(channel);
        showNewNotificacion();
    }
    private void showNewNotificacion(){
        setPendingIntent(MapClientActivity.class);
        NotificationCompat.Builder builder= new
NotificationCompat.Builder(getApplicationContext(),ChannelID)
                .setSmallIcon(R.drawable.camionicono)
                .setContentTitle("Cambion de Basura cerca")
                .setContentText("El Vehiculo recolector ya se encuentra
cerca de tu zona")
                .setPriority(NotificationCompat.PRIORITY_HIGH)
                .setContentIntent(pendingIntent);
        NotificationManagerCompat managerCompat =
```

```
NotificationManagerCompat.from(getApplicationContext());
        managerCompat.notify(1, builder.build());
    }

    private void setPendingIntent(Class<?>clsActivity){
        Intent intent = new Intent(this,clsActivity);
        TaskStackBuilder stackBuilder= TaskStackBuilder.create(this);
        stackBuilder.addParentStack(clsActivity);
        stackBuilder.addNextIntent(intent);
        pendingIntent =
stackBuilder.getPendingIntent(1,PendingIntent.FLAG_UPDATE_CURRENT);
    }


}
```

La pantalla para el conductor tiene un código similar al del usuario, ya que esta también guarda las coordenadas actuales del conductor en la base de datos para poder ser rastreado, la diferencia que llevan entre las pantallas es que esta no puede ver a los demás ciudadanos ni a los demás conductores, tampoco recibe notificaciones de estar cerca de los ciudadanos, su única función es actualizar su posición actual para que los ciudadanos puedan darle rastreo.

Lo primero es declarar las variables globales para poder hacer uso de mapas, módulo de autenticación y base de datos.

Se hace una llamada en respuesta al evento mLocationCallback este da una respuesta cada que se detecta un movimiento del dispositivo en tiempo real, se quita el marcador que pone el control de google maps por default, para en su lugar poner el marcador personalizado. En este caso será el icono del carro de basura mostrando la ubicación actual inicial y se manda a llamar el método para obtener a los conductores activos.

```
public class MapsDriverActivity extends AppCompatActivity implements
OnMapReadyCallback {


    private GoogleMap mMap;
    private SupportMapFragment mMapfragment;
    private Authprovider mAuthprovider;
    private LocationRequest mLocationRequest;
    private FusedLocationProviderClient mFusedlocation;
    private Marker mMarker;
    private Button mButonConnect;
```

```
    private Boolean isConected = false;
    private GeofireProvider mGeofireProvider;

    private TokenProvider mTokenProvider;

    private final static int LOCATION_REQUEST_CODE = 1;
    private final static int SETTING_REQUEST_CODE = 2;

    private LatLng mCurrentLatLong;

    LocationCallback mLocationCallback = new LocationCallback() {
        @Override
        public void onLocationResult(@NonNull LocationResult
locationResult) {
            super.onLocationResult(locationResult);

            for (Location location : locationResult.getLocations()) {
                if (getApplicationContext() != null) {

                    if (mMarker != null){
                        mMarker.remove();
                    }
                    mCurrentLatLong = new
LatLng(location.getLatitude(),location.getLongitude());

                    mMarker = mMap.addMarker(new
MarkerOptions().position(
                            new
LatLng(location.getLatitude(),location.getLongitude())
                    ).title("Posicion
actual").icon(BitmapDescriptorFactory.fromResource(R.drawable.camionicono
)));

          //OBTENER LA LOCALIZACION DEL CONDUCTOR EN TIEMPO REAL

mMap.moveCamera(CameraUpdateFactory.newCameraPosition(new
CameraPosition.Builder().target(new LatLng(location.getLatitude(),
location.getLongitude()))
                                    .zoom(18f)
                                    .build()
                            )
                    );

                    updateLocation();

                }

            }}};
```

En el método onCreate se inicializan las variables y se hace instancia a las clases que ayudarán a obtener las ubicaciones, el acceso a la base de datos y módulo de autenticación, y la generación del token único. Este token será el identificador para que los servicios de google registren la aplicación para los servicios de notificaciones push.

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

        mButonConnect= (Button)findViewById(R.id.conectar_btn);
        mAuthprovider = new Authprovider();
        mGeofireProvider = new GeofireProvider();

        mFusedlocation =
LocationServices.getFusedLocationProviderClient(this);

        mTokenProvider = new TokenProvider();

        mMapfragment = (SupportMapFragment)
getSupportFragmentManager().findFragmentById(R.id.mapa);
        mMapfragment.getMapAsync(this);
        MyToolbar.show(this, "Conductor", false);

        GenerateToken();
    }
```

El método updateLocation se usa para registrar en la base de datos la ubicación actual del vehículo recolector de basura. Primero se revisa si está registrada la sesión del usuario en el módulo de autenticación y si ya se ha registrado coordenadas, si es así, se mandan las coordenadas y el id del usuario a la base de datos para poder identificar que vehículo recolector que este activo y en movimiento.

```
    private void updateLocation(){
        if(mAuthprovider.existSesion() && mCurrentLatLong!=null){

mGeofireProvider.saveLocation(mAuthprovider.getID(),mCurrentLatLong);
        }

    }
```

En el método onMapReady, es donde se inicializa la configuración del mapa de google, este método solo se ejecuta una sola vez al momento de que se muestre la pantalla de mapa en el dispositivo.

```
    @Override
    public void onMapReady(@NonNull GoogleMap googleMap) {
        mMap = googleMap;
        mMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
        mMap.getUiSettings().setZoomControlsEnabled(true);
        if (ActivityCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
            checkLocationPermissions();
        }


        mMap.addMarker(new MarkerOptions()
                .position(new LatLng(0, 0))
                .title("Marker"));
        mLocationRequest = new LocationRequest();
        mLocationRequest.setInterval(1000);
        mLocationRequest.setFastestInterval(1000);

//mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        mLocationRequest.setPriority(Priority.PRIORITY_HIGH_ACCURACY);
        mLocationRequest.setSmallestDisplacement(5);


    }
```

A partir de las versiones de Android OREO el uso de mapas requiere que siempre se revise que se tengan los permisos necesarios para hacer uso del GPS, así como en la pantalla del ciudadano en caso de tener concedidos los permisos se procede a iniciar la geolocalización mediante la clase mFusedLocation se utiliza un loop para que todo el tiempo se esté ejecutando esta función y la geolocalización sea continua.

```
    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
        if (requestCode == LOCATION_REQUEST_CODE) {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                if
(ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) {
                    if(gpsActived()){
```

```
mFusedlocation.requestLocationUpdates(mLocationRequest,
mLocationCallback, Looper.myLooper());
                        //si se quisiera habilitar el circulo azul en el
mapa, utilizar la siguiente linea en true
                        //y ponerla siempre despues de todas las llamadas
de mFusedLocation
                        mMap.setMyLocationEnabled(false);
                    }
                    else{
                        showAlertdialogNOGPS();
                    }

                } else {
                    checkLocationPermissions();
                }
            } else {
                checkLocationPermissions();
            }
        }

    }
```

Después de revisar que existan los permisos para usar el GPS, también se debe revisar que éste encendido físicamente, si todo esto es correcto se inician las actualizaciones de geolocalización, en el caso contrario si el GPS se encuentra apagado le pide al usuario encenderlo y lo dirige directamente a la configuración del sistema, una vez activo el GPS le pide al usuario otorgar los permisos de uso del periférico.

```
@Override
    protected void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == SETTING_REQUEST_CODE && gpsActived()) {
            if
(ActivityCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
                return;
            }
            mFusedlocation.requestLocationUpdates(mLocationRequest,
mLocationCallback, Looper.myLooper());
        }
        else{
            showAlertdialogNOGPS();
```

```
        }
    }



    private void showAlertdialogNOGPS(){
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setMessage("Por favor activa tu GPS para continuar")
                .setPositiveButton("Configuraciones", new
DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int
which) {
                        startActivityForResult(new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS),SETTING_REQUEST_CODE);
                    }
                })
                .create().show();
    }

    private boolean gpsActived(){
        boolean isActive = false;
        LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);


if(locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)){
            isActive= true;
        }
        return isActive;
    }

    private void startLocation(){
        if(Build.VERSION.SDK_INT>=Build.VERSION_CODES.M){
            if(ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION)==PackageManager.PERMISSION_GRAN
TED){
                if(gpsActived()){
                    mButonConnect.setText("DESCONECTAR");
                    isConected=true;

mFusedlocation.requestLocationUpdates(mLocationRequest,mLocationCallback,
Looper.myLooper());
                } else{
                    showAlertdialogNOGPS();
                }

            }
            else{
                checkLocationPermissions();
            }
        }
        else{
            if(gpsActived()){

mFusedlocation.requestLocationUpdates(mLocationRequest,mLocationCallback,
Looper.myLooper());
```

```
            }else{
                showAlertdialogNOGPS();
            }
        }
    }

    private void checkLocationPermissions(){
        if(ContextCompat.checkSelfPermission(getApplicationContext(),
Manifest.permission.ACCESS_FINE_LOCATION)!=PackageManager.PERMISSION_GRAN
TED){

if(ActivityCompat.shouldShowRequestPermissionRationale(this,Manifest.perm
ission.ACCESS_FINE_LOCATION)){
                new AlertDialog.Builder(this)
                        .setTitle("Proporciona los permisos para
continuar").setMessage("Esta aplicacion requiere de los permisos de
ubicacion para utilizarse")
                        .setPositiveButton("OK", new
DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog,
int which) {

ActivityCompat.requestPermissions(MapsDriverActivity.this,new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},LOCATION_REQUEST_CODE)
;
                            }
                        })
                        .create()
                        .show();
            }
            else {

ActivityCompat.requestPermissions(MapsDriverActivity.this,new
String[]{Manifest.permission.ACCESS_FINE_LOCATION},LOCATION_REQUEST_CODE)
;
            }
        }
    }
```

En la parte superior de la pantalla se encuentra el menú desplegable el cual tiene las opciones de información de conductor y salir de la sesión.

```
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.driver_menu,menu);
        return super.onCreateOptionsMenu(menu);
    }
```

```
    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        if(item.getItemId()==R.id.action_logout){
            salir();
        }
        if(item.getItemId()==R.id.action_info){
            Intent intent = new
Intent(MapsDriverActivity.this,InfoConductor.class);
            startActivity(intent);
        }
        return super.onOptionsItemSelected(item);
    }
```

Si se selecciona la opción de salir, se manda llamar la función desconectar esta función también es llamada cuando el conductor presiona el botón de salir, la diferencia que tiene es que si cierra sesión desde el menú superior la aplicación se dirige a la pantalla de bienvenida y si solo presiona el botón desconectar únicamente se deja de enviar las actualizaciones de ubicación. Pero el conductor sigue logeado en la base de datos y en módulo de autenticación de Firebase.

```
private void salir(){
        desconectar();
        mAuthprovider.logout();
        Intent intent = new Intent(MapsDriverActivity.this,
MainActivity.class);
        startActivity(intent);
        finish();
    }

    private void desconectar(){
        if(isConected){
            if(mFusedlocation!=null){
                mButonConnect.setText("CONECTAR");
                isConected=false;
                mFusedlocation.removeLocationUpdates(mLocationCallback);
                if(mAuthprovider.existSesion()){

mGeofireProvider.removeLocation(mAuthprovider.getID());
                }
            }else{
                Toast.makeText(this,"No se puede
desconectar",Toast.LENGTH_SHORT).show();
            }
        }else{
```

```
            startLocation();
        }
    }

    public void desconectar(View view){
        if(isConected){
            if(mFusedlocation!=null){
                mButonConnect.setText("CONECTAR");
                isConected=false;
                mFusedlocation.removeLocationUpdates(mLocationCallback);
                if(mAuthprovider.existSesion()){

mGeofireProvider.removeLocation(mAuthprovider.getID());
                }
            }else{
                Toast.makeText(this,"No se puede
desconectar",Toast.LENGTH_SHORT).show();
            }
        }else{
            startLocation();
        }
    }

    public void GenerateToken(){
        mTokenProvider.create(mAuthprovider.getID());

    }

}
```

CAPÍTULO 4
RESULTADOS

4.1 Resultados y discusión.

4.1.1 Base de datos.

Se hicieron pruebas para verificar la correcta configuración de los módulos de autenticación y base de datos de Firebase. Así como el uso de la llave en el metadato de la aplicación y la implementación de las librerías de Firebase.

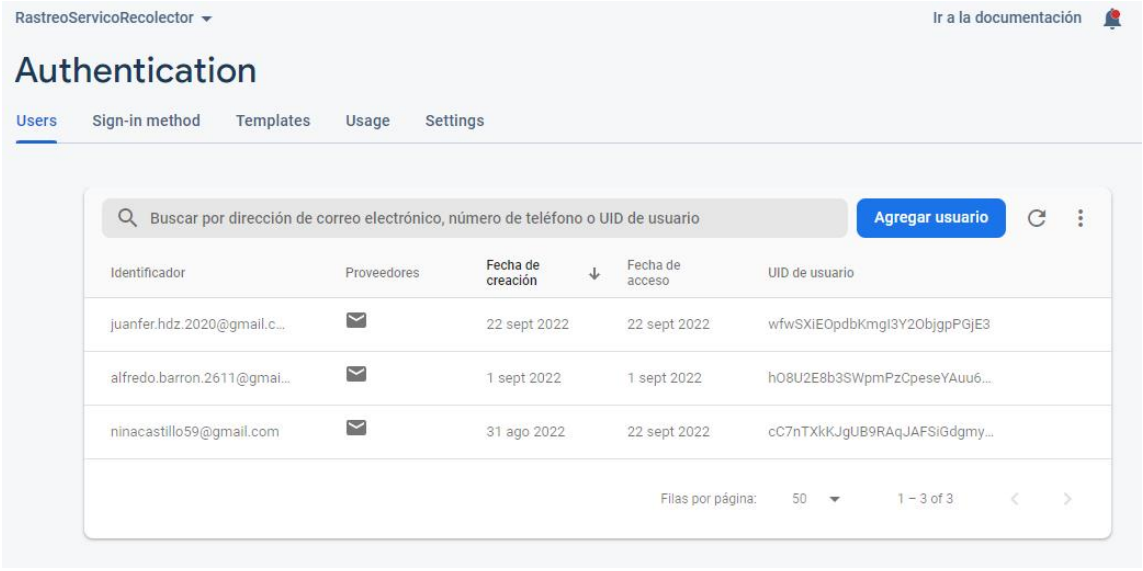


Figura 4. 1 Módulo de autenticación de Firebase.

En la figura 4.1 se muestra que los registros fueron insertados de manera correcta desde la aplicación móvil.

La segunda validación es en el módulo RealtimeDatabase de Firebase, en el cual se verifica que se agreguen los nodos y datos correspondientes.

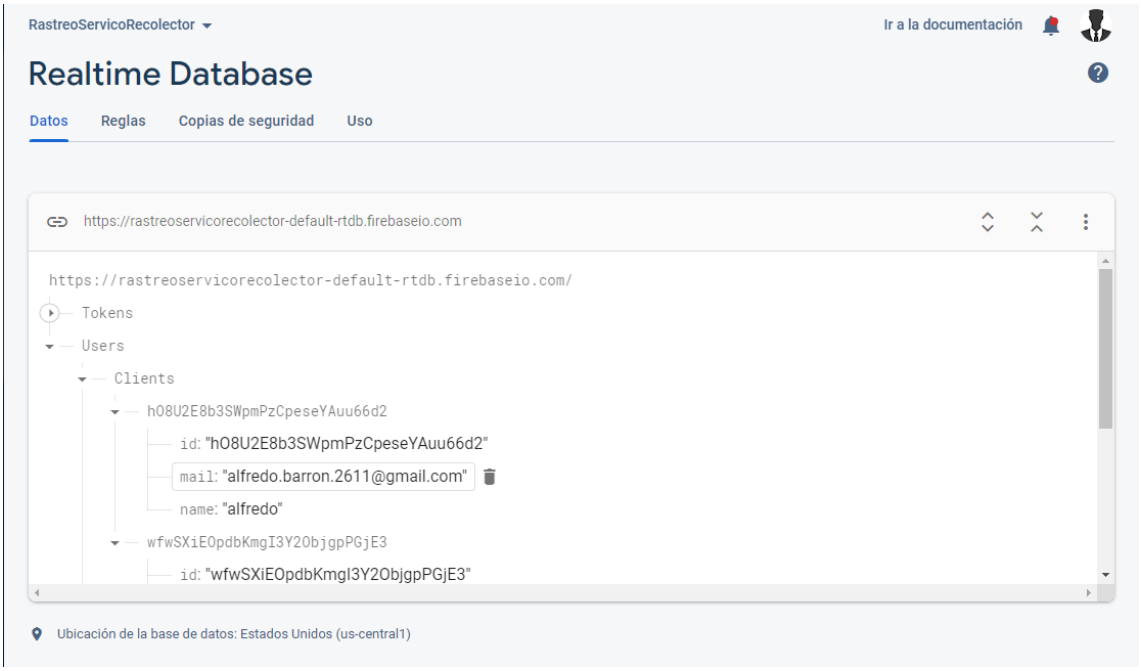


Figura 4. 2 Módulo Realtime Database de Firebase.

En la figura 4.2, se puede observar que se crearon los nodos correspondientes al tipo de usuario seleccionado en la aplicación. Así como los datos de registro que fueron insertados en dicha aplicación.

4.1.2 Aplicación móvil.

El desarrollo de la aplicación móvil dio los resultados esperados, se hicieron pruebas de las cuatro secciones que conforman la aplicación, que son: Base de datos, Pantallas y transiciones, Geolocalización y notificaciones push.

Las primeras pruebas con respecto a la inserción en la base de datos fueron exitosas como se muestra en las figuras 4.1 y 4.2.

En cuanto a la funcionalidad de las pantallas y transiciones de las mismas, se hace una serie de pruebas para verificar su funcionamiento los resultados se muestran a continuación.

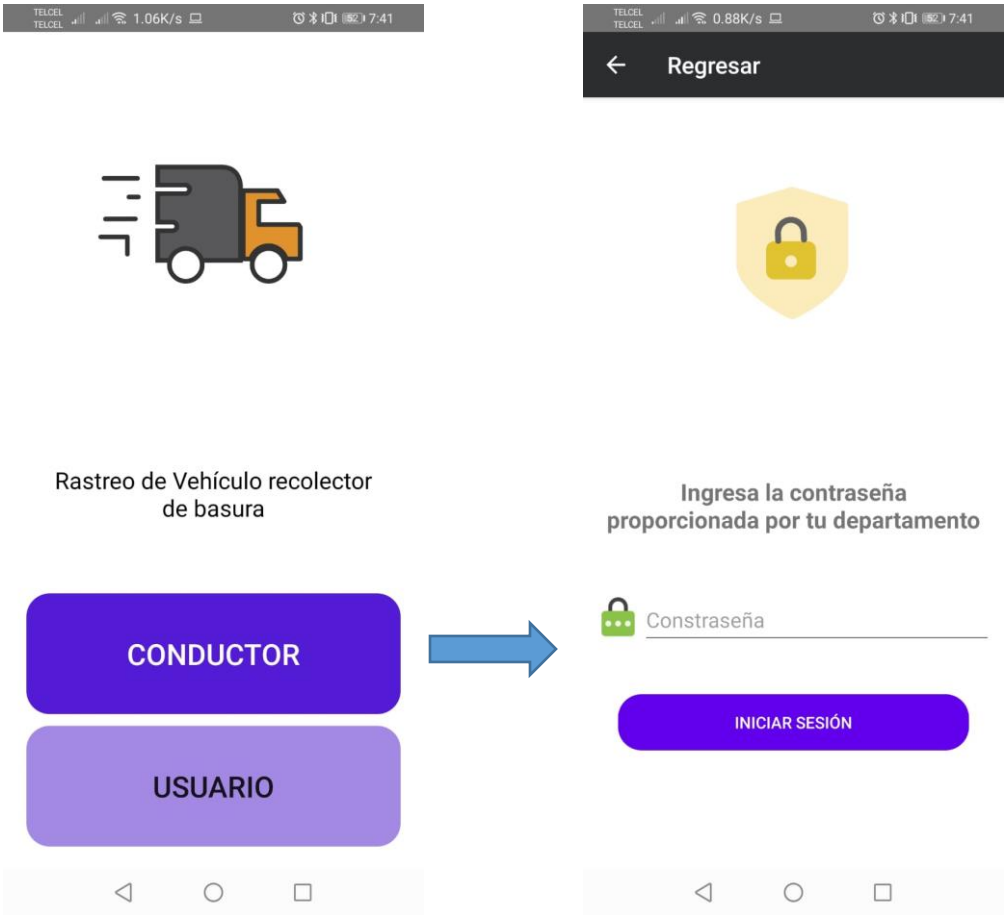


Figura 4. 3 Selección de conductor.

En la primera pantalla (figura 4.3) se hace de manera correcta la transición utilizando el botón de conductor y muestra la pantalla de acceso a opciones de usuario, también se ejecuta de manera correcta las animaciones de cada una de las pantallas.

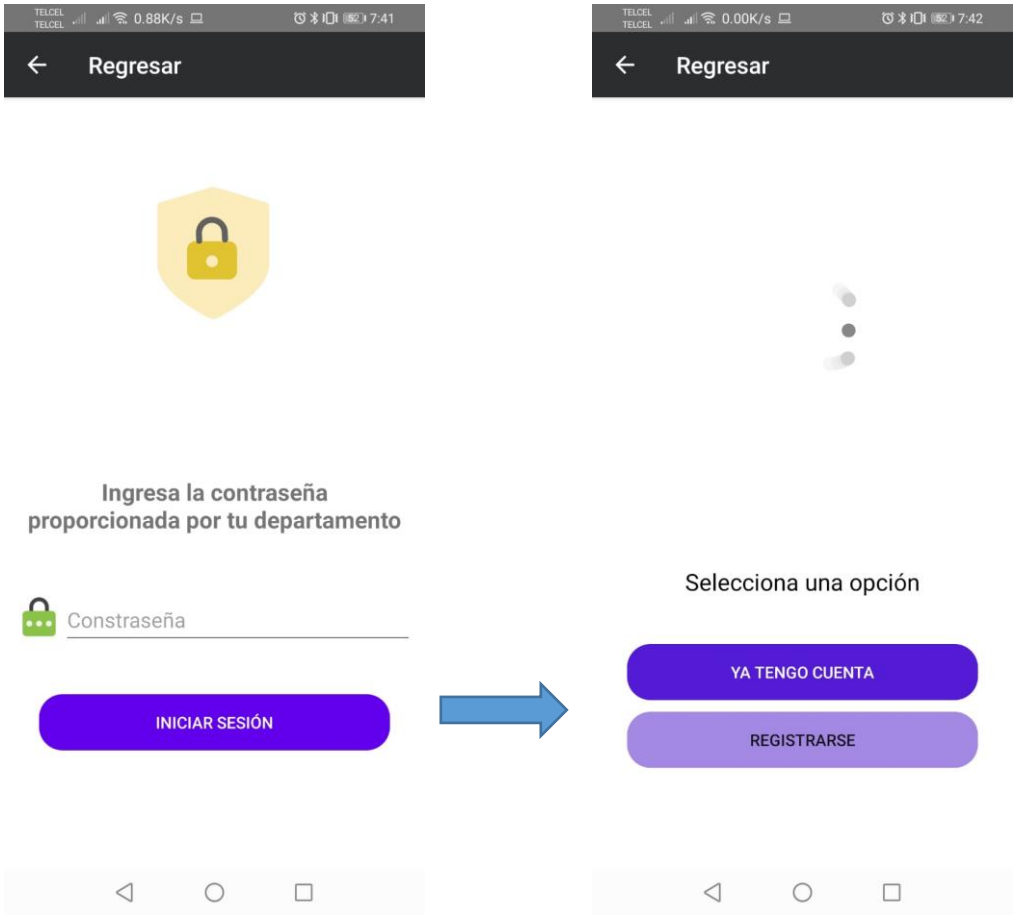


Figura 4. 4 Opciones de conductor

Se realiza de manera exitosa la validación de la contraseña proporcionada por el departamento de limpia, esta contraseña se usa para restringir el registro o acceso a los ciudadanos que no sean trabajadores del servicio de limpia, para esto la aplicación revisa en la base de datos cual es la contraseña proporcionada por el departamento, esta contraseña puede ser editada en la base de datos para siempre tener el control sobre los usuarios nuevos. Y si es la correcta dirige al conductor a las opciones del mismo, como se muestra en la figura 4.4.

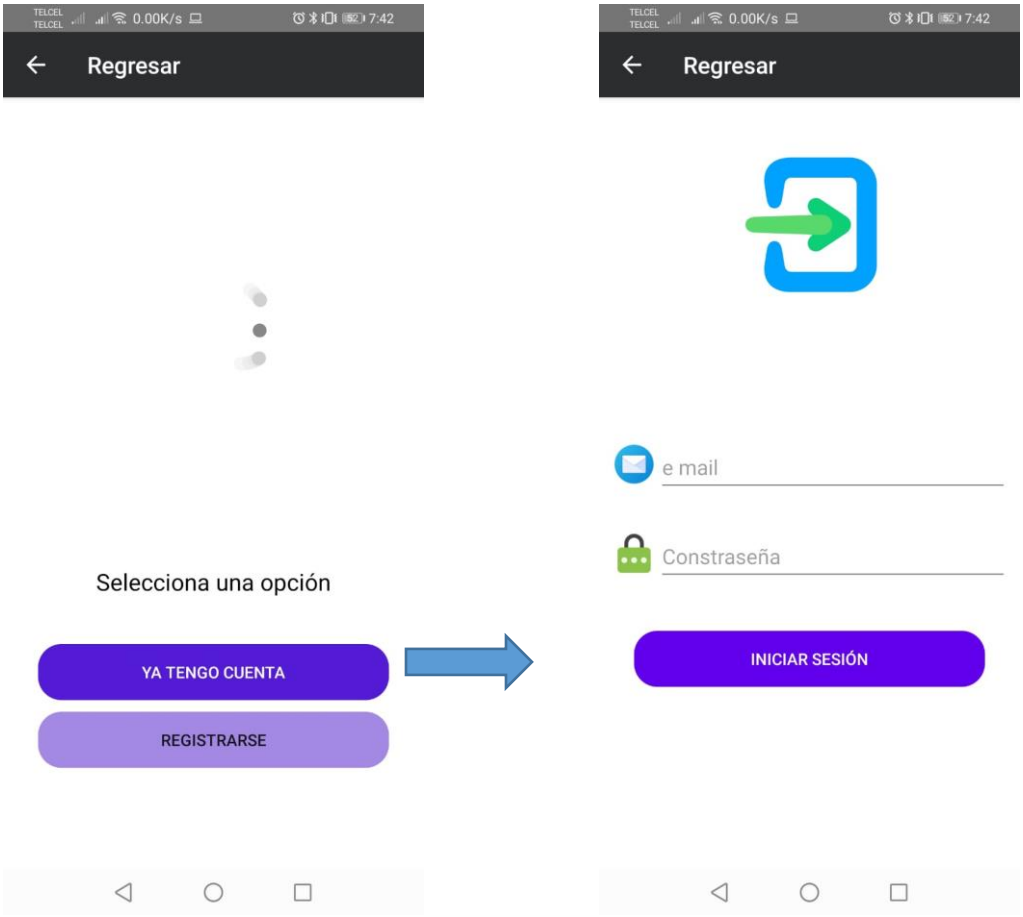


Figura 4. 5 Inicio de sesión.

Si el conductor selecciona que ya se ha registrado anteriormente, la aplicación lo dirige a la pantalla de Login, se corrobora que también la animación correspondiente a cada una de las pantallas se ejecute de manera correcta, véase la figura 4.5.

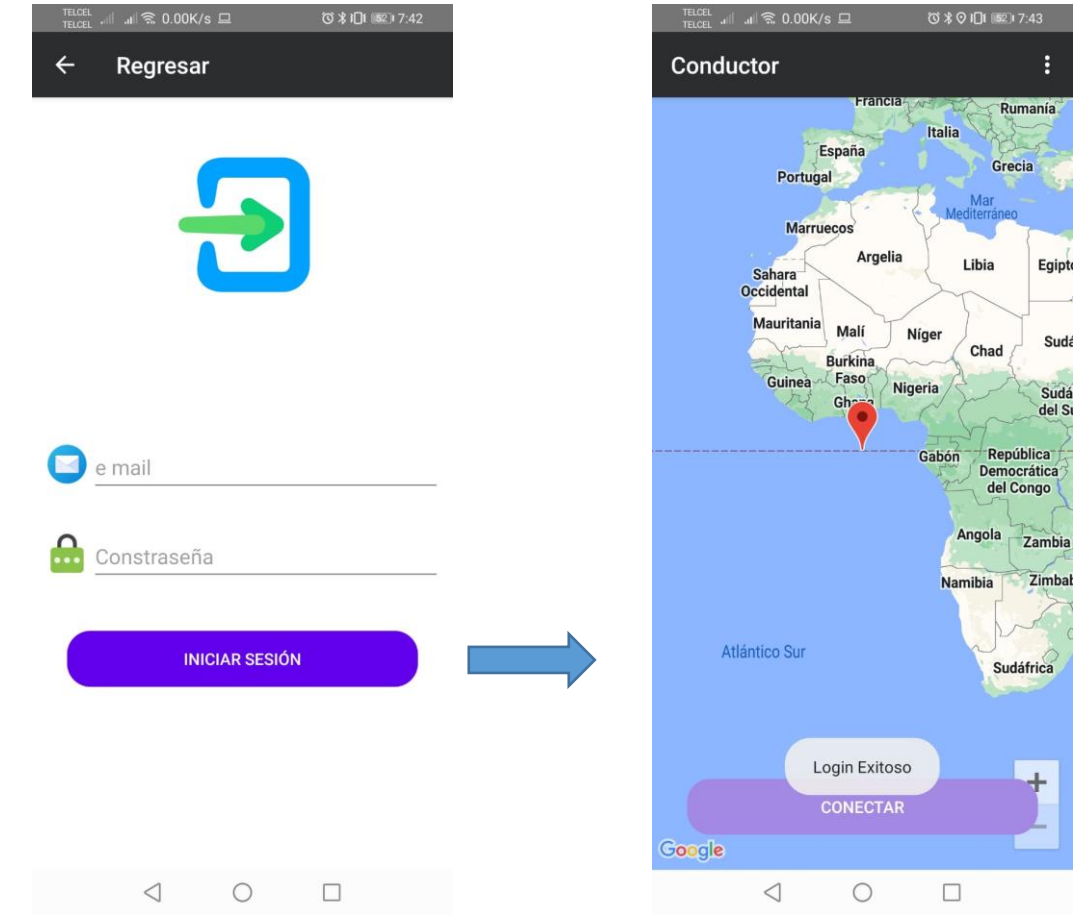


Figura 4. 6 Inicio de sesión correcto.

Se confirma que la conexión a la base de datos se hace de manera correcta, la pantalla Login hace una conexión correcta y revisa en el módulo de autenticación de Firebase que se encuentre registrado el usuario y que su contraseña sea la correcta. Siendo esto verdadero se muestra la pantalla principal del conductor (figura 4.6), la cual también se confirma su correcto funcionamiento. Dicha pantalla hace una conexión correcta al módulo RealtimeDatabase de Firebase, para insertar el registro a la base de datos de las coordenadas del conductor.

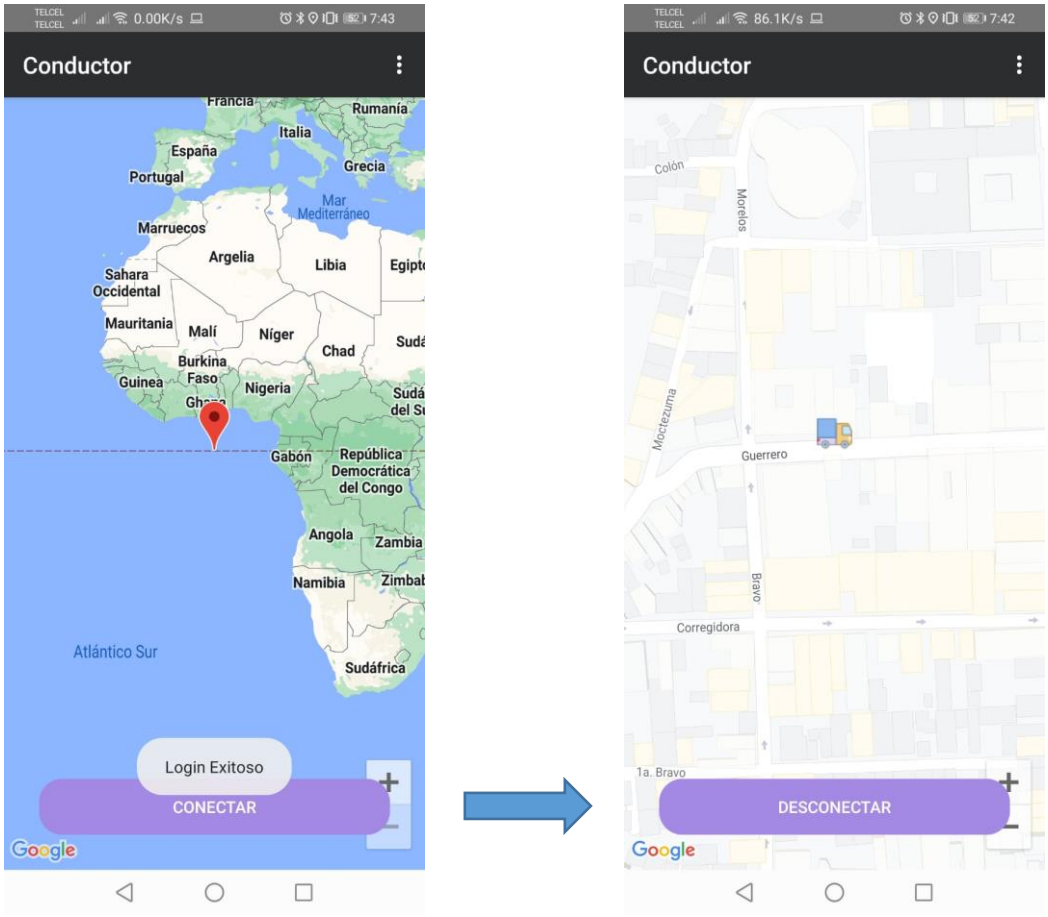


Figura 4. 7 Conexión a servicios de google maps.

Para la verificación del correcto funcionamiento de la pantalla principal del conductor se presiona el botón “conectar”, dicho botón inicia la comunicación con las API de google para hacer uso del servicio de maps. Se puede corroborar que funciona de manera correcta ya que la pantalla se actualiza a las coordenadas actuales en tiempo real del vehículo recolector, como se muestra en la figura 4.7.

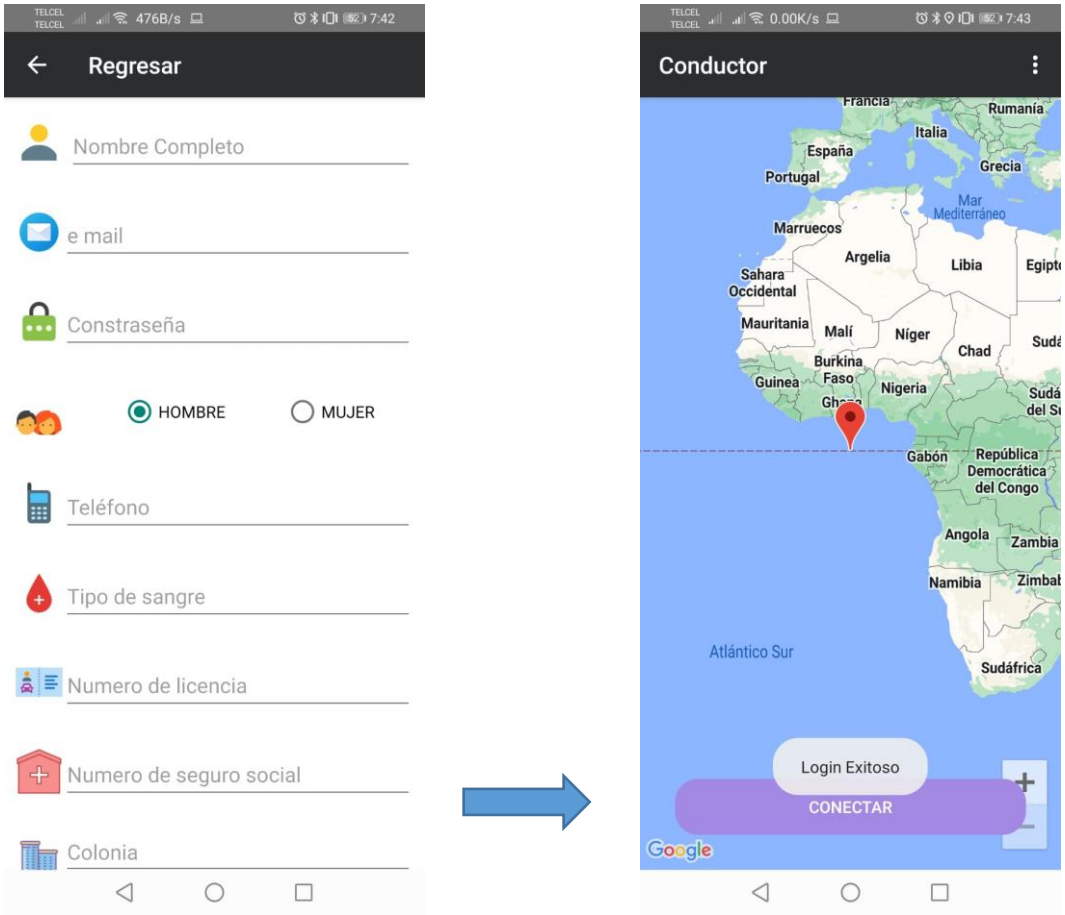


Figura 4. 8 Prueba de registro de usuario.

El registro de los conductores se efectúa de manera correcta, validando que todos los campos se hayan llenado y que la contraseña sea mayor de 6 dígitos, si esto es correcto se guardan los datos del usuario en el módulo de autenticación de Firebase, así como en el módulo Realtime Database. Después del registro exitoso la aplicación muestra la pantalla principal del conductor para poder iniciar con su recorrido, véase la figura 4.8.

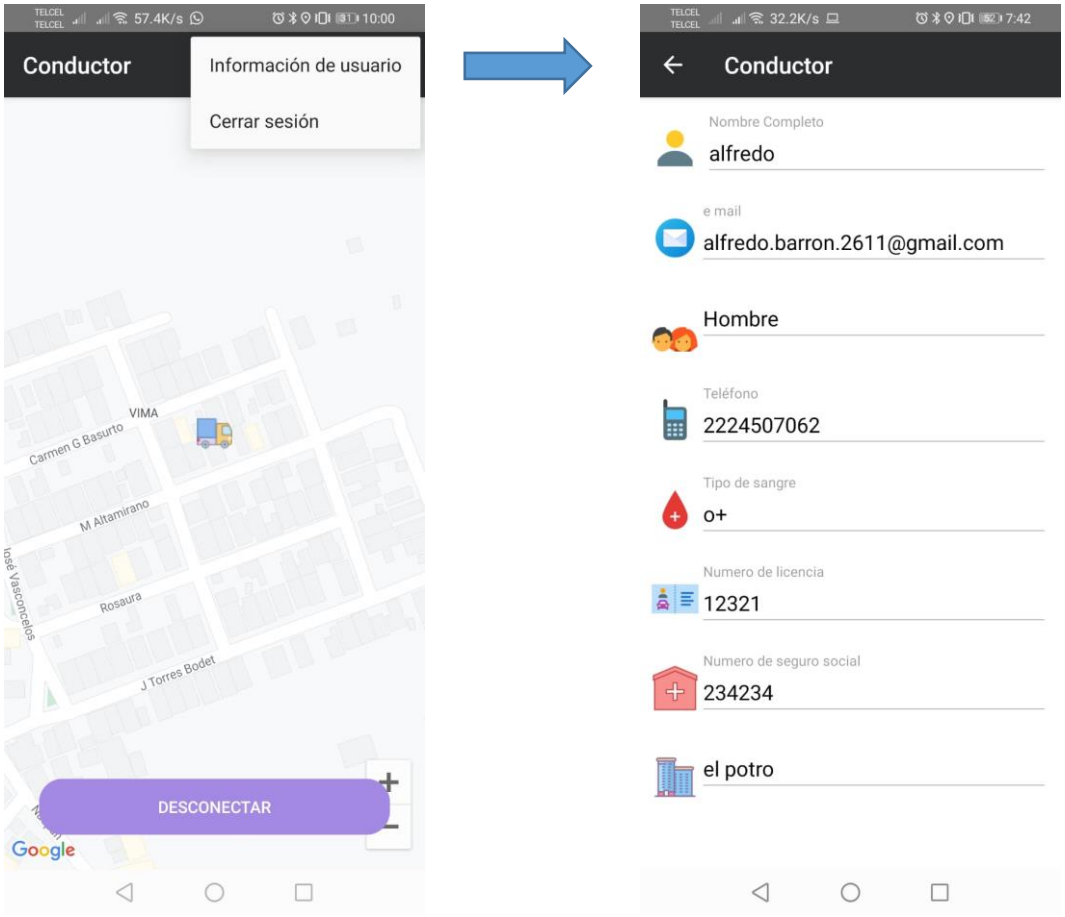


Figura 4. 9 Información de conductor.

Por último, en el lado del conductor, se realiza de manera exitosa el despliegue del menú, el cual dirige a la pantalla de información de usuario y cerrar sesión. La pantalla de información de usuario se conecta de manera correcta a la base de datos para mostrar la información almacenada que registró el conductor durante su registro (figura 4.9).

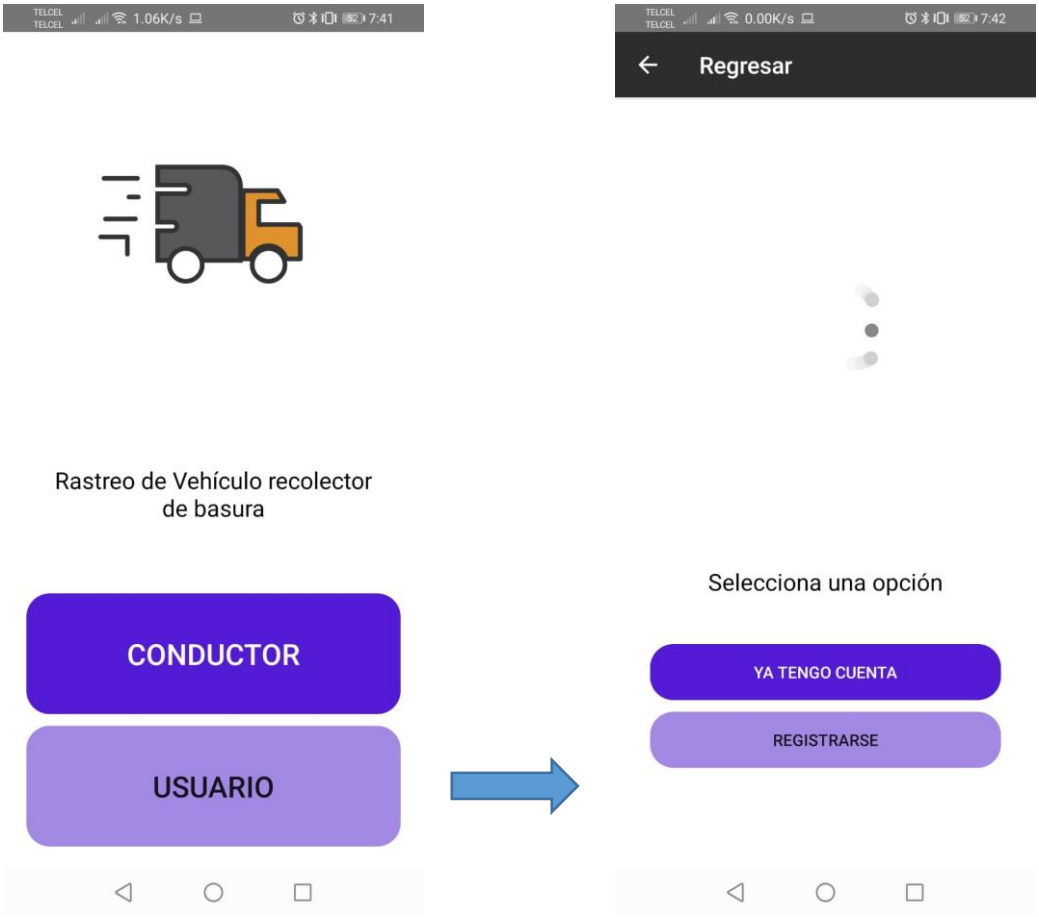


Figura 4. 10 Opciones de inicio de sesión de usuario.

Para el usuario común (ciudadano), no se muestra la pantalla que restringe el uso al registro de nuevos conductores, en esta parte cualquier ciudadano puede registrarse o iniciar sesión si este se encuentra registrado en la base de datos (figura 4.10).

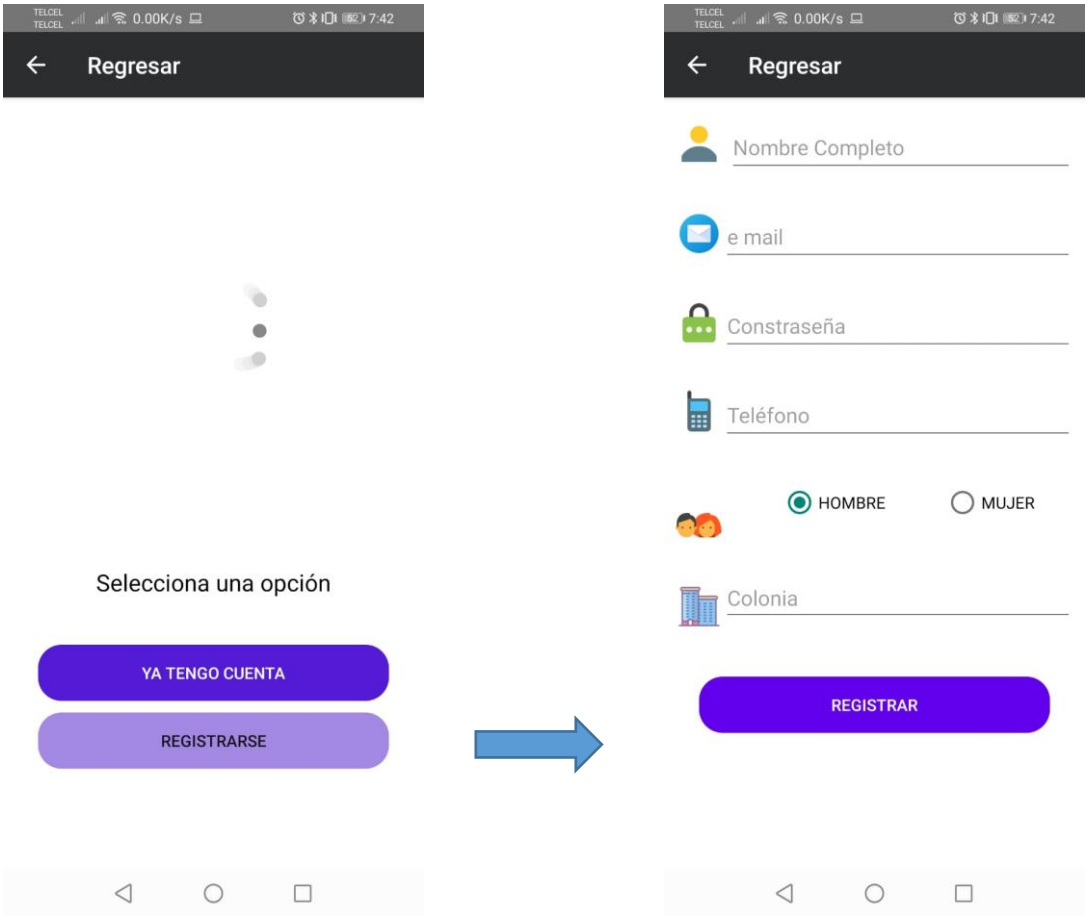


Figura 4. 11 Registro de usuario (Ciudadano).

El registro de los usuarios de efectúa de manera correcta, validando que todos los campos sean rellenados de manera correcta y que la contraseña del usuario sea mayor a 6 dígitos, después de estas validaciones se hace una inserción de manera correcta a la base de datos, así como al módulo de autenticación (figura 4.11).

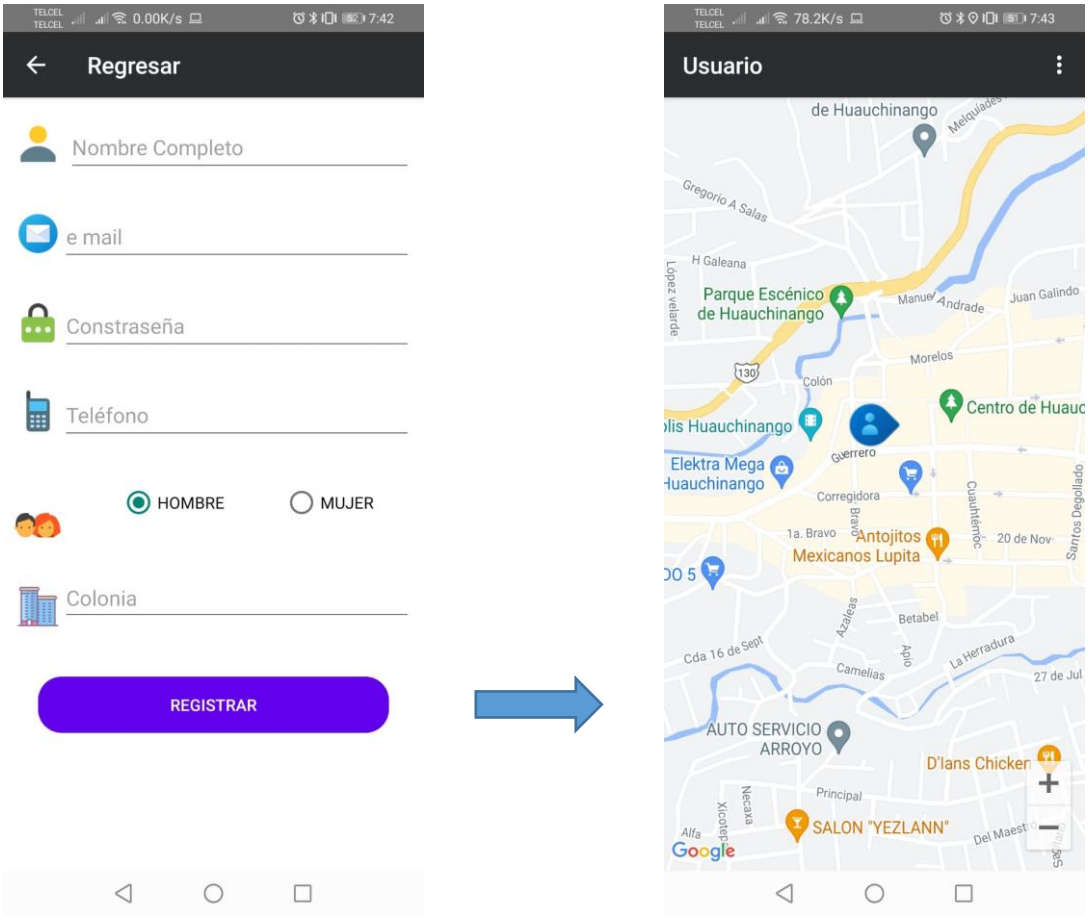


Figura 4. 12 Registro exitoso.

Al igual que no el conductor, si el registro es exitoso, se muestra la pantalla principal del usuario común, esta se conecta de manera automática mostrando que ha funcionado correctamente la conexión con las APIS de google para mostrar la ubicación, véase la figura 4.12.

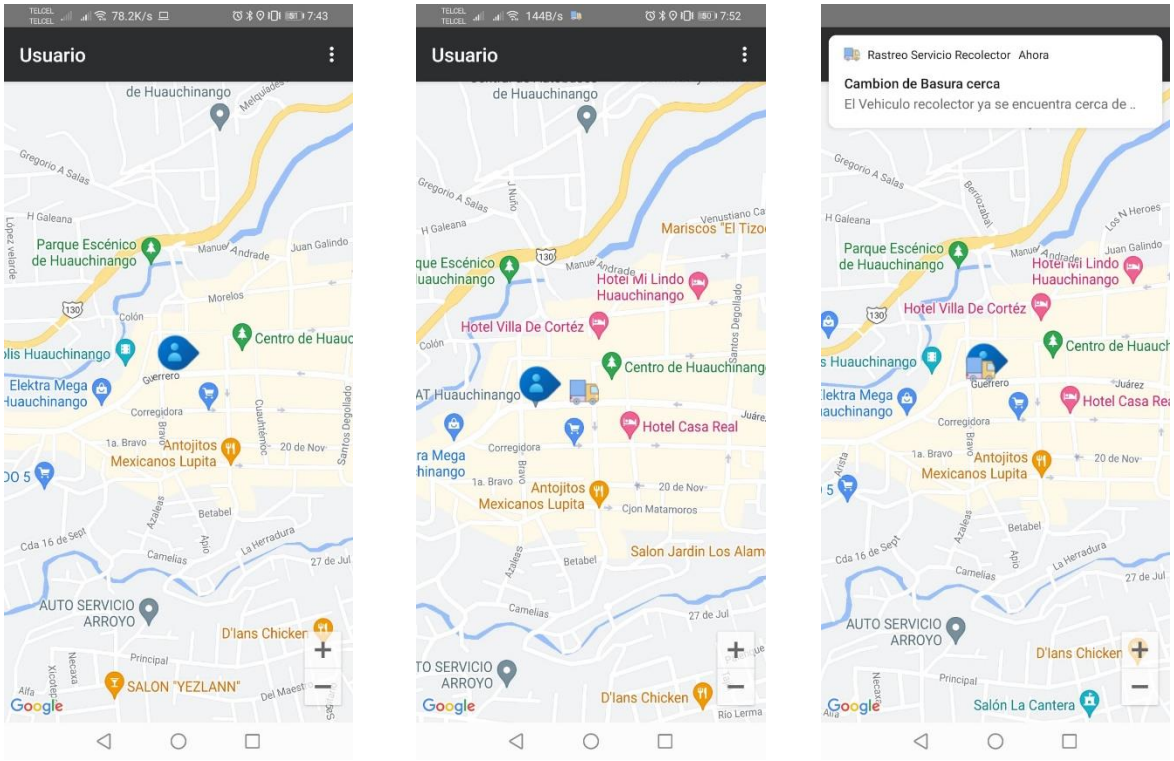


Figura 4. 13 Rastreo del vehículo de basura.

En la pantalla del usuario se muestran los recolectores de basura activos, para esto la aplicación se conecta a la base de datos y obtiene una lista de todos los conductores que se encuentren activos, y recibe las coordenadas de cada uno de ellos para así agregar los marcadores de cada uno de los conductores al mapa, como se observa en la figura 4.13. Cuando algún vehículo se encuentre cerca del usuario, la aplicación manda una notificación para avisar que el recolector esta por pasar en su zona, y así el usuario, aunque no esté al pendiente de su pantalla, este pueda estar notificado y pueda sacar su basura en el momento oportuno.

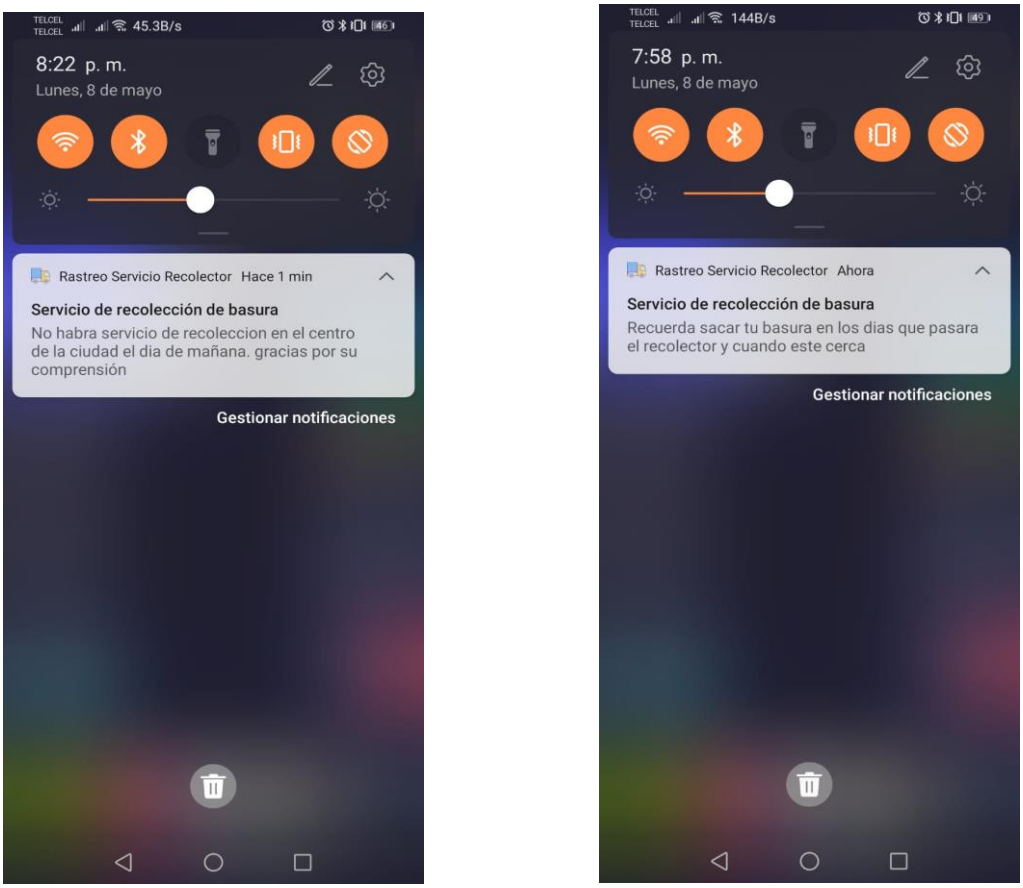


Figura 4. 14 Notificaciones al usuario.

El uso de notificaciones push, permite al usuario que, aunque no tenga la aplicación abierta, se les pueda mandar información de eventos no programados con el servicio de limpia, o también campañas de concientización a los ciudadanos. Estas notificaciones pueden ser programadas para que lleguen una sola vez por evento o que sea recurrente en campañas, como se puede ver en la figura 4.15.

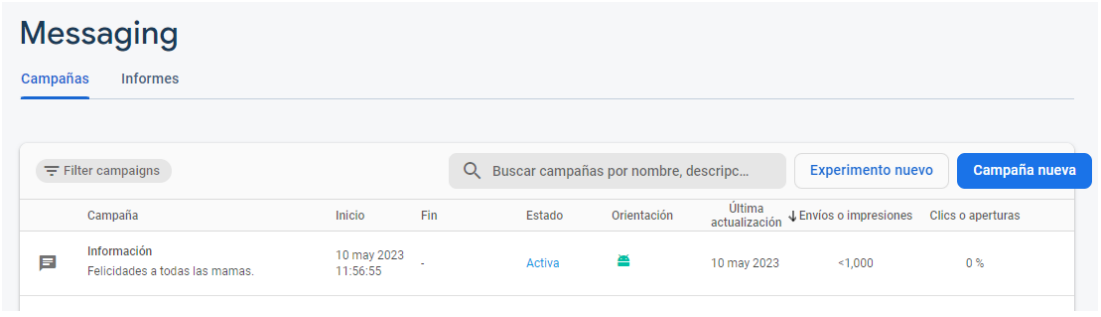


Figura 4. 15 Consola de Cloud Messaging de Firebase.

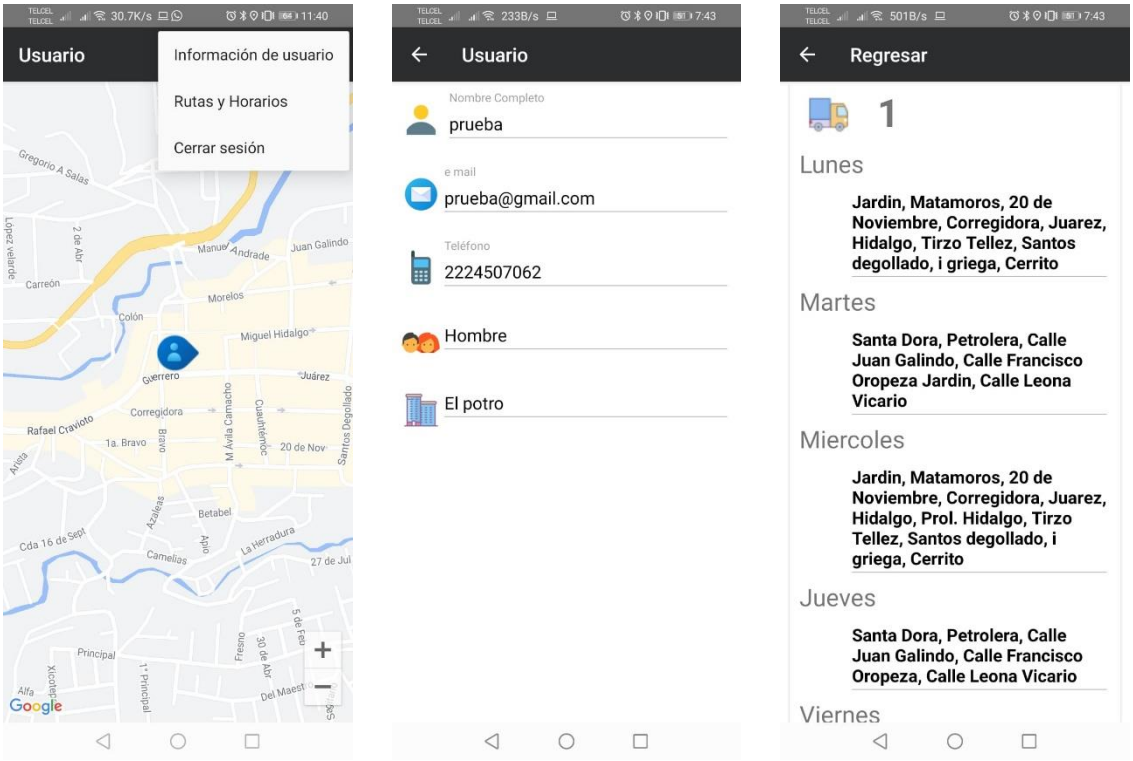


Figura 4. 16 Opciones de usuario.

Del lado del usuario (ciudadano), se corrobora que las opciones funcionan de manera correcta, la primera opción es la información del usuario que está en sesión activa.

La aplicación despliega el menú y abre la nueva ventana, en la cual se accede a la base de datos desde su ejecución para traer la información registrada, como se muestra en la figura 4.16.

La segunda opción y muy importante, es el calendario de recolección para toda la semana, este calendario puede ser editado desde la base de datos y se pueden agregar más rutas o quitar las mismas como se observa en la figura 4.17.

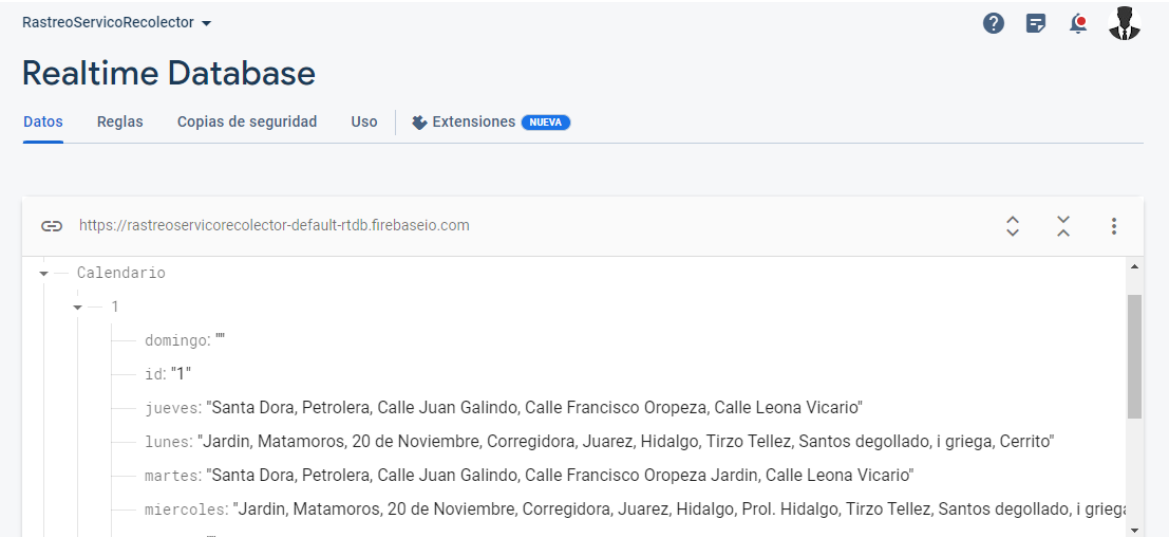


Figura 4. 17 Calendario de recolección.

4.1.3 Rendimiento.

Para realizar las pruebas de rendimiento, se utiliza la aplicación AIDA64 [41] y las estadísticas de Android, para saber el consumo real de memoria, almacenamiento, batería y datos transmitidos.

Se usan dos dispositivos para realizar las pruebas, Honor con 6 GB de memoria RAM y 128 GB de almacenamiento y Samsung a52 con 6 GB d memoria RAM y 128 GB de memoria interna, véase en la figura 4.18.

Figura 4. 18 Características de dispositivo

El tipo de batería del dispositivo es Litio, con un total de 4000 mAh, El consumo de batería es bastante aceptable, ya que la tasa de descarga de la aplicación es de 285mA y la temperatura de trabajo es de 26°C. Como se puede observar en la figura 4.19.

Figura 4. 19 Consumo de batería.

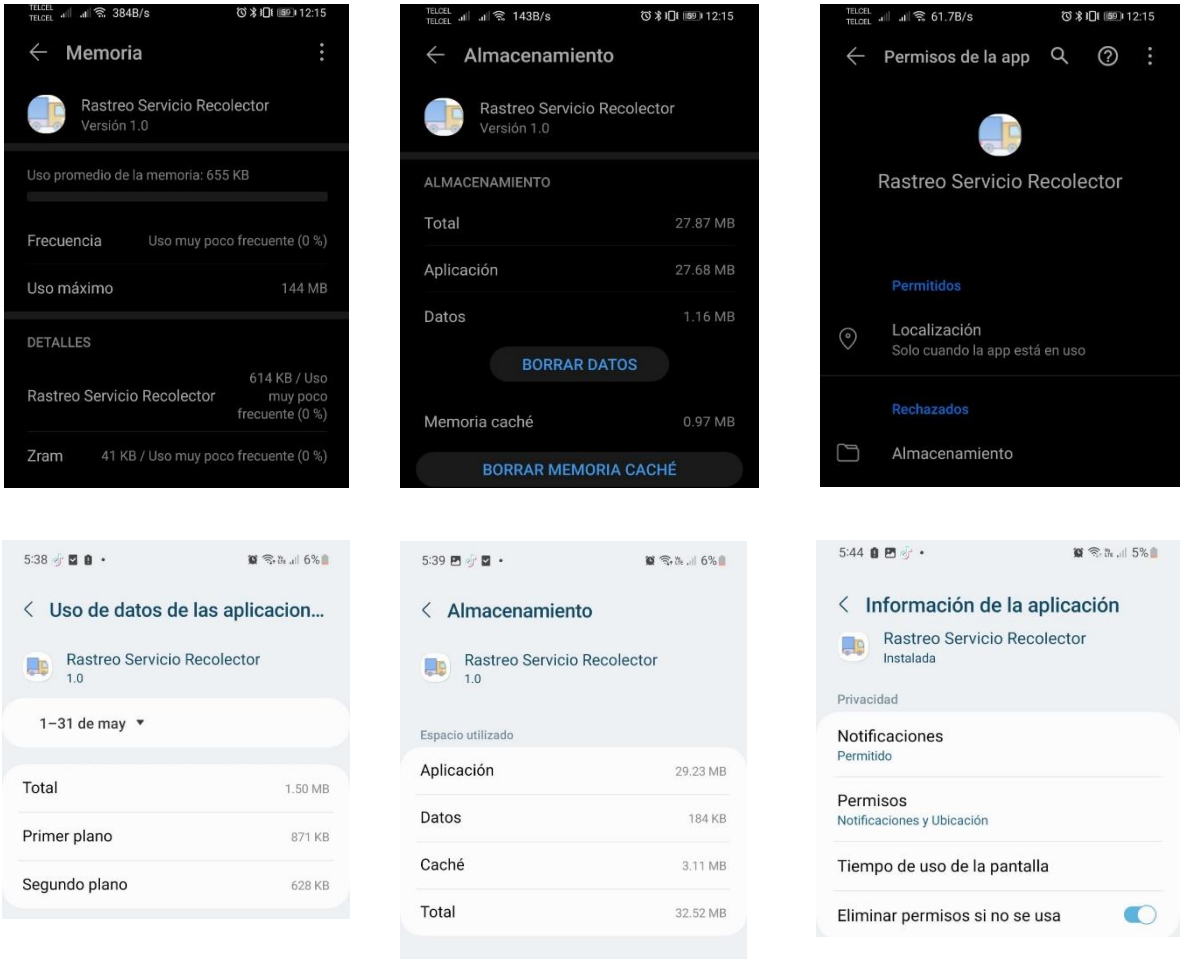


Figura 4. 20 Consumo de memoria, almacenamiento, datos y permisos.

Como se puede ver en la figura 4.20, el uso promedio de memoria RAM de la aplicación es de 655 KB, y el uso máximo es de 144 MB. El espacio de almacenamiento máximo utilizado total es de 32.52 MB. Esto en el caso del modelo SAMSUNG y en el caso del modelo HONOR el consumo total es de 27.87 MB, en cuanto a los permisos necesarios para ejecutar la aplicación solo es el dispositivo de geolocalización. Por último, el consumo de datos fue de 1.50 MB en un periodo de 15 días con la aplicación en segundo plano. Esto indica que los requerimientos son mínimos, así como los permisos necesarios, haciendo que la aplicación pueda ser ejecutada con equipos de capacidades mínimas.

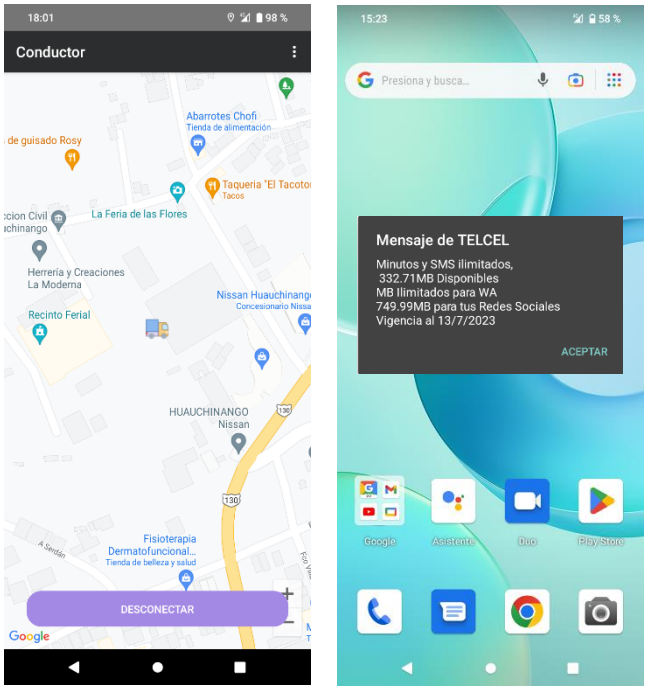
4.1.4 Prueba de campo.

Se realizó una prueba de campo con condiciones reales, para el conductor se utilizó un dispositivo de gama baja para verificar que la aplicación puede correr en casi cualquier dispositivo Android, con las siguientes características:

- Marca: WIKO.
- Versión de Android: 11.
- Memoria RAM: 2GB.
- Almacenamiento: 64 GB.

Y la aplicación cliente fue instalada por el personal del departamento de ecología del H. Ayuntamiento de Huauchinango Puebla, con el fin de medir el consumo real de datos y batería en un recorrido del camión de basura y el consumo de datos en Firebase cuando varios usuarios se conecten a la plataforma. La prueba se realizó durante 2 días, en los cuales se dio seguimiento al vehículo recolector con la siguiente configuración.

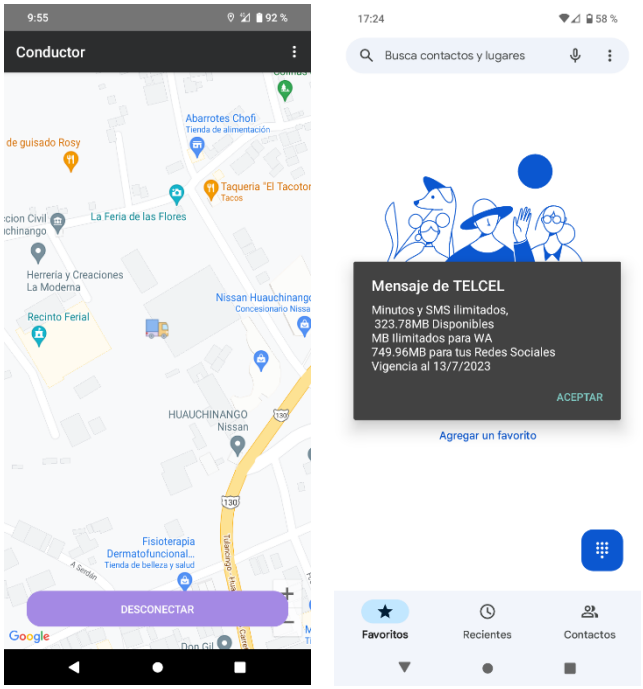
Conductor: La aplicación se ejecutó todo el tiempo del recorrido, el conductor no tuvo necesidad de interactuar con la aplicación ya que todo se ejecuta de manera automática, al finalizar cada recorrido se recogió el dispositivo celular para revisar el consumo de batería, y de datos durante cada turno, los resultados son satisfactorios véase en las figuras 4.21 y 4.22.



Ruta Vespertina

- **3:00 p.m. a 10:30 p.m.**
- Batería inicial 98%
- Batería final 58%
- **Consumo de batería 40%**
- Saldo inicial 345 MB
- Saldo final 332.71 MB
- **Saldo consumido 12.29 MB**

Figura 4. 21 Resultados turno vespertino.



Ruta Matutina

- **7:00 a.m. a 12:00 p.m.**
- Batería inicial 92%
- Batería final 68%
- **Consumo de batería 24%**
- Saldo inicial 332.71 MB
- Saldo final 323.78 MB
- **Saldo consumido 8.93 MB**

Figura 4. 22 Resultados turno matutino.

Usuario: En el caso de los usuarios, el criterio evaluado no fue el rendimiento de batería y consumo de saldo, ya que este no necesariamente tiene la aplicación todo el tiempo en ejecución en primer plano. Para su evaluación se realizó una encuesta de experiencia de usuario con una muestra de 19 usuarios, ya que el departamento de limpia opto hacer la prueba con su personal, ya que no es una aplicación oficial no podían mostar a la ciudadanía la aplicación. Dando los siguientes resultados.

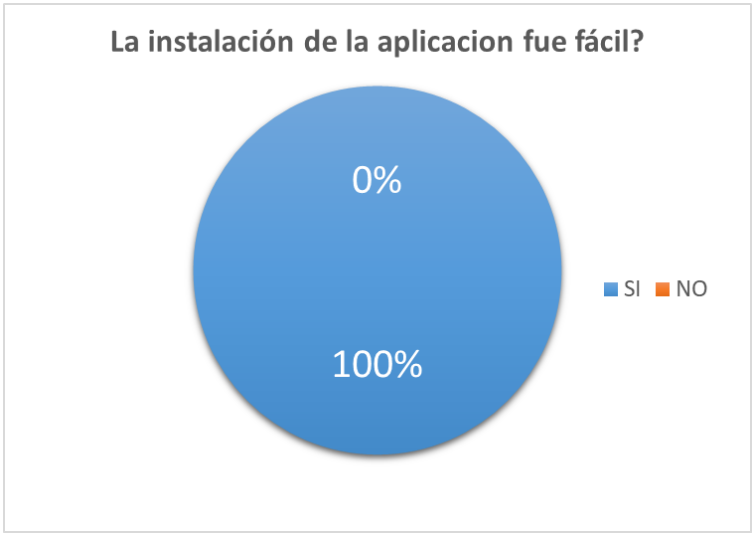


Figura 4. 23 Instalación.

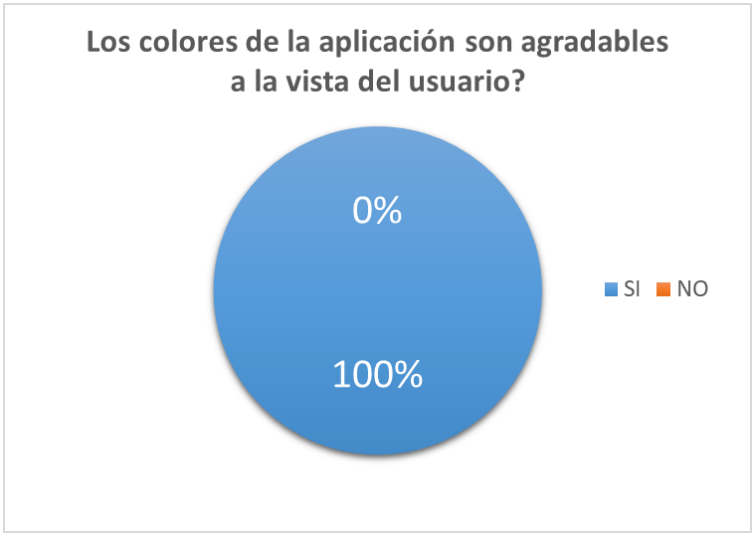


Figura 4. 24 Colores.

Figura 4. 25 Pantallas.

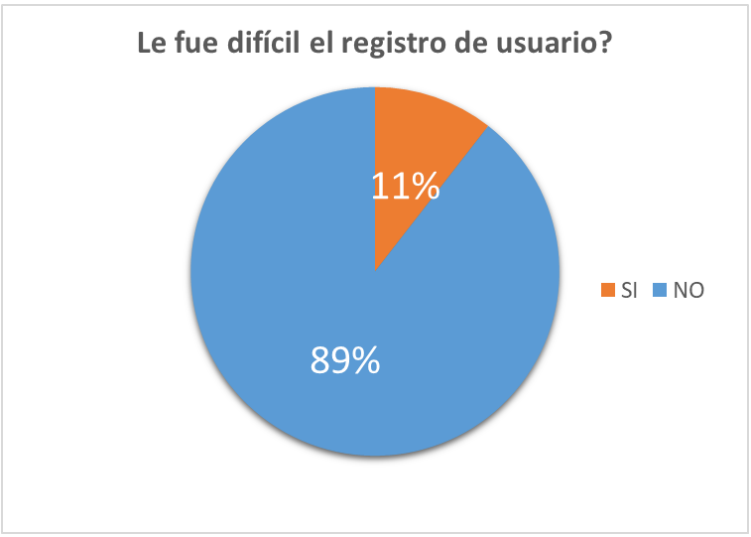


Figura 4. 26 Registro.

Figura 4. 27 Inicio de sesión gráfico.

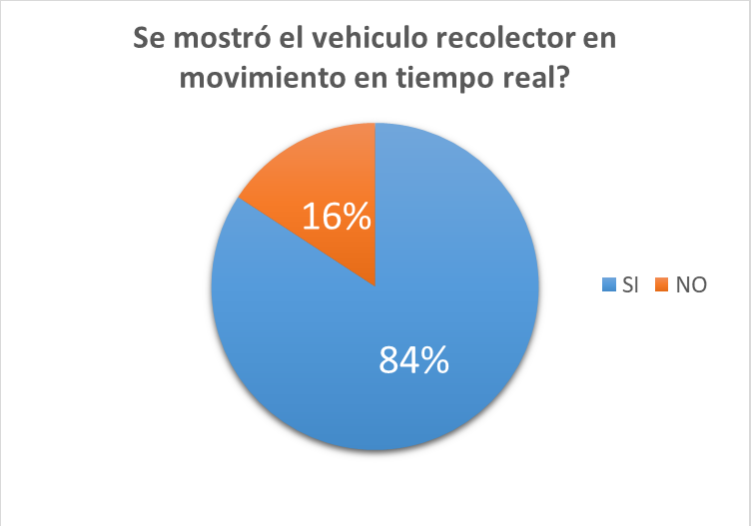


Figura 4. 28 Movimiento de recolector.

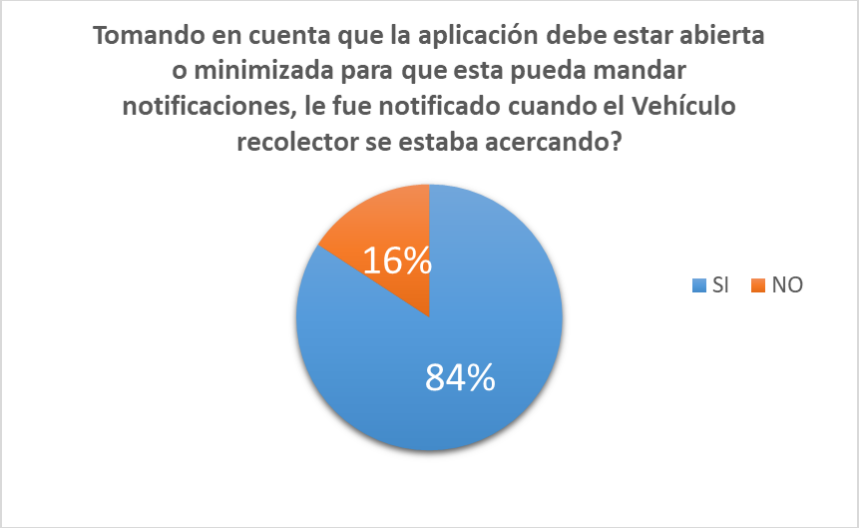


Figura 4. 29 Notificación de recolector.

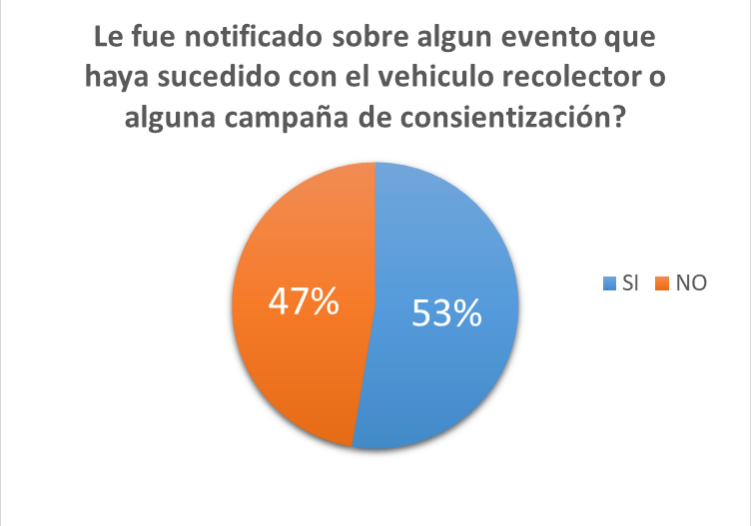


Figura 4. 30 Notificación de campaña.

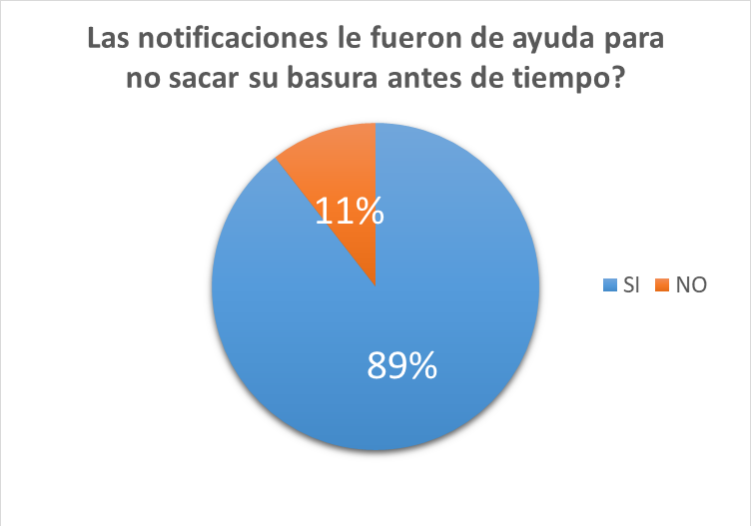


Figura 4. 31 Notificaciones.

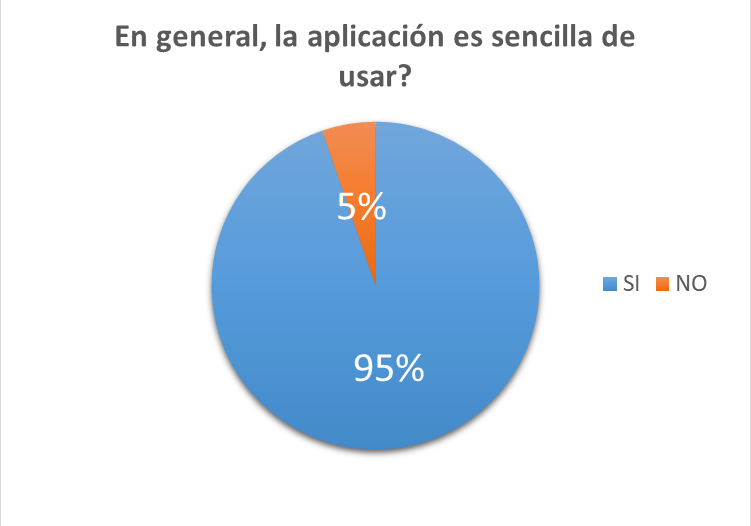


Figura 4. 32 Aplicación fácil de usar.

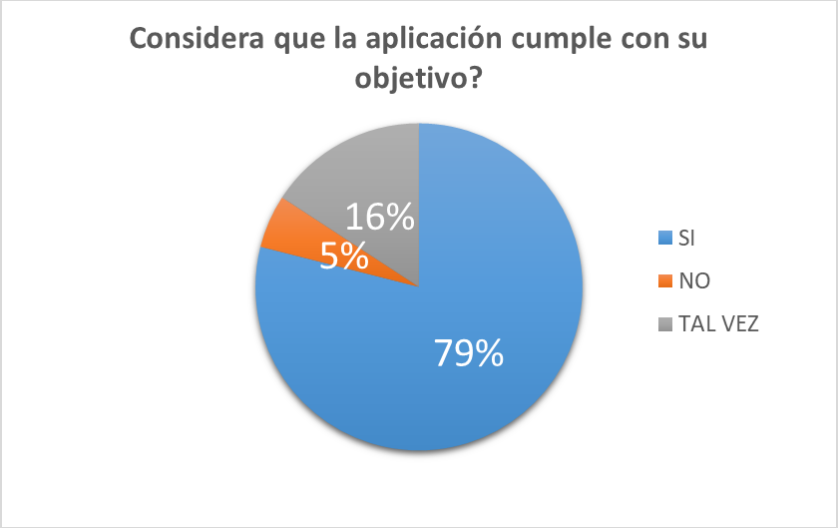


Figura 4. 33 Objetivo de aplicación.

Las respuestas otorgadas por los usuarios dan un mejor panorama de cuáles son las áreas de oportunidad de la aplicación y que puntos son favorables con lo ya desarrollado. Los usuarios concuerdan que la aplicación puede cumplir con su función.

4.1.5 Discusión.

Los ingenieros Juan Pablo Astudillo León y Edgar Gustavo Delgado Tello en su tesis "Sistema de localización monitoreo y control vehicular basado en los protocolos GPS/GSM/GPRS concuerdan que el uso de tecnologías GPRS son mejor aplicables para proyectos de rastreo satelital, ya que la su respuesta es inmediata a diferencia de GPS que utilicen servicio SMS, estos no garantizan que los datos al instante que son enviados con las coordenadas correspondientes.

En su trabajo utilizaron software libre para la programación de su plataforma web y servidores para darle seguimiento a las unidades desde una computadora mediante el navegador de internet para que pueda ser compatible con cualquiera de los sistemas operativos, a diferencia de este proyecto que el desarrollo fue nativamente para una aplicación Android, para que cualquier persona que tenga un dispositivo móvil con este sistema operativo pueda acceder al rastreo vehicular desde cualquier lugar que se encuentre sin necesidad de estar conectado a una red wifi o Ethernet, también se agrega la función de guardado de datos, ya sea para la información de los usuario y conductores así también para guardar y actualizar las rutas programadas de la semana desde la base de datos y añadir notificaciones push a la aplicación para avisar a los usuarios cuando se encuentre el vehiculo recolector cerca o avisar cualquier anomalía con la ruta.

Por otro lado, al igual que Liliana Grimaneza Guzmán se llega a la misma conclusión que en su tesis "Evaluación y diseño de un sistema de rastreo satelital para el monitoreo y control de las rutas terrestres asignadas diariamente en tiempo real", donde se hace mención el uso de las API de google maps para el rastreo en tiempo real del vehículo, agregando que en este proyecto se añaden módulos de Firebase de google, para el uso de base de datos y autenticación de usuarios, añadiendo que con esta herramienta se puede medir el uso de la aplicación, cuantificando el número de usuarios registrados, cantidad de MB que son almacenados y descargados diariamente para así dar una mejor perspectiva de cuantas personas estan utilizando la aplicación.

CAPITULO 5

CONCLUSIONES Y TRABAJOS FUTUROS

5.1 Conclusiones.

En el presente trabajo se presentó el desarrollo de una aplicación móvil para dispositivos ANDROID para dar seguimiento en tiempo real a los vehículos recolectores de basura en el municipio de Huauchinango Puebla, esto para disminuir los puntos de acumulación de basura en la ciudad. Previniendo enfermedades, disminución de animales portadores de infecciones, evitar mala imagen a la ciudad, agilizar el tráfico vehicular y el proceso de recolección de basura.

La aplicación puede ayudar a la ciudadanía para que su uso sea de la manera más sencilla e intuitiva posible, pero que al mismo tiempo cumpla con su objetivo, la aplicación propuesta tiene las siguientes características:

- Funciones diferentes para ciudadanos y conductores de vehículos recolectores.
- Contraseñas dinámicas para conductores, para que solo se puedan registrar trabajadores del departamento de limpia como conductores, editable desde la plataforma de google Firebase.
- Registro de ciudadanos mediante correo electrónico y contraseña.
- Información de usuarios de los perfiles que están usando la aplicación.
- Información del calendario de recolección programada, editable desde la plataforma de google Firebase.
- Seguimiento en tiempo real de los vehículos recolectores de basura.
- Notificación a los ciudadanos cuando el vehículo recolector este cerca (cuando la aplicación este en ejecución o minimizada).
- Notificación a los ciudadanos de campañas de concienciación, avisos incluso si la aplicación no se encuentra en ejecución enviadas desde la plataforma de google Firebase.

Con las pruebas realizadas se puede concluir que la aplicación es apta para usarse en teléfonos Android que cuenten con un mínimo de 1 GB de memoria RAM y 8 GB de almacenamiento interno, actualmente está especificación es de las más bajas en el mercado, aunque preferentemente se recomienda usar un teléfono Android con por lo menos 3 GB de

memoria RAM y 32 GB de almacenamiento interno para que tenga un óptimo funcionamiento con las transiciones de pantalla.

También se hace notar el uso de Firebase para gestionar la base de datos y los mensajes push para notificar al usuario, esta es una excelente opción para este tipo de aplicaciones, por la sencillez en su configuración, incluso se puede seguir paso a paso desde el mismo Android Studio, por su adaptabilidad ya que la base de datos puede ir creciendo de manera dinámica a medida que vayan registrándose más usuarios, su fácil manejo por ser una estructura de tipo NoSQL, y al ser una herramienta alojada en los servidores de google, permite siempre estar disponible y segura la información.

Por último, se llega a la conclusión de la fácil configuración de sus APIs para hacer uso de Google Maps y poder dar a la aplicación la funcionalidad principal y cumplir con el objetivo de la misma.

5.2 Recomendaciones.

Como todas las aplicaciones en la actualidad existen ciertas limitantes, así como recomendaciones de uso, las aplicaciones requieren de mantenimiento y métodos de uso, a continuación, se listan algunas para que la aplicación funcione de manera óptima.

- El teléfono siempre debe tener activado su GPS durante el uso de la aplicación.
- Se deben conceder los permisos necesarios que pide la aplicación para su ejecución.
- Tener conectividad ya sea de señal GSM o WIFI.
- En el caso del conductor, se debe tener activo el teléfono en todo momento para la geolocalización, por lo cual también se recomienda conectarlo a la corriente durante su uso.
- En el caso del usuario, aunque no se muestre en primer plano la aplicación debe permanecer minimizada para poder notificar al usuario sobre la llegada del vehículo recolector de basura, es decir no cerrar del todo la aplicación o eliminar el proceso de la misma.

5.3 Trabajos futuros.

Como parte del proceso de desarrollo de cada proyecto de investigación, una o varias líneas tienen opciones de mejora. Durante el desarrollo de la presente tesis surgen nuevas líneas de investigación las cuales se espera seguir retroalimentando con nueva información, aplicando mejoras y retomarla para proyectos futuros. A continuación, se aportan algunas ideas para apoyar y mejorar la aplicación.

- Avisar al usuario cuando el vehículo recolector se encuentre cerca, aun cuando la aplicación no se esté ejecutando (segundo plano).
- Mostrar nombre de rutas que se muestran en el mapa.
- Editar desde la aplicación la información de los usuarios.
- Recuperar usuario y contraseña de los ciudadanos desde la aplicación.
- Mostrar la distancia y tiempo aproximado de llegada del vehículo recolector de basura.
- Desarrollar un panel de control web propio para enviar notificaciones y gestión de usuarios.
- Validar el registro de los usuarios en dos pasos preferentemente por SMS.
- Cambiar de color el icono del vehículo recolector cuando este haya perdido la conexión, no cuando haya dejado de moverse.
- Usar conexión bluetooth para cuando no exista conexión WIFI o internet móvil.
- Utilizar asistente para que gente con deficiencia visual pueda ser notificada.
- Migrar la aplicación a IOS.

Anexos

Anexo 1 Publicación de artículo en revista indexada de Tecnologias de la Información, https://www.ecorfan.org/bolivia/researchjournals/Tecnologias_de_la_Informacion/vol9num28/Journal_Information_Technology_V9_N28_1.pdf

Article
Journal Information Technology
December, 2022 Vol.9 No.28 1-6

Android application for tracking the garbage collection vehicle in Huauchinango Puebla

Aplicación Android para el rastreo del vehículo recolector de basura en Huauchinango Puebla

BARRON-CASTILLO, Jorge Alfredo†, HERNÁNDEZ-LUNA, Aldo*, TORRES-JIMÉNEZ, Jacinto and LUNA-CARRASCO, Claudia Yadira

Tecnológico Nacional de México/Instituto Tecnológico Superior de Huauchinango, México.

ID 1st Author: *Jorge Alfredo, Barrón Castillo* / **ORC ID**: 0000-0001-8055-3892, **CVU CONACYT ID**: 1162160

ID 1st Co-author: *Aldo, Hernández-Luna* / **ORC ID**: 0000-0002-7717-5314, **CVU CONACYT ID**: 441305

ID 2nd Co-author: *Jacinto, Torres-Jiménez* / **ORC ID**: 0000-0002-8006-6397, **CVU CONACYT ID**: 103469

ID 3rd Co-author: *Claudia Yadira, Luna-Carrasco* / **ORC ID**: 0000-0002-4092-9987, **CVU CONACYT ID**: 368419

DOI: 10.35429/JIT.2022.28.9.1.6

Received: August 10, 2022; Accepted December 30, 2022

Abstract

The present investigation deals with the development of an application for Android devices to monitor in real time the garbage collection vehicle in the municipality of Huauchinango Puebla, the objective of this is to propose a solution for the collection problems in the city of Huauchinango Puebla, and thus be able to avoid sources of infection, bad appearance and accumulation of animals that can be dangerous for the general public. The application will be developed in Android Studio, using the extreme programming methodology for its development. The satellite tracking will be through an Android application which will constantly send the geolocation of the vehicle to a server that in turn will be accessed by a second application to indicate the location of the vehicle, as well as this will show the warnings when the collection vehicle does not Make your route, go late, when you are close and at what time you will be arriving at the collection area in order to remove the waste in a timely manner.

Android, Geolocation, Collection. Application

Resumen

La presente investigación aborda el desarrollo de una aplicación para dispositivos Android para dar seguimiento en tiempo real al vehículo recolector de basura en el municipio de Huauchinango Puebla, el objetivo de esto es proponer una solución para los problemas de recolección en la ciudad de Huauchinango Puebla, y así poder evitar focos de infección, mal aspecto y acumulación de animales que pueden ser peligrosos para la ciudadanía en general. La aplicación será desarrollada en Android Studio, utilizando la metodología de programación extrema para su desarrollo. El rastreo satelital será a través de una aplicación Android la cual mandara constantemente la geolocalización del vehículo a un servidor que a su vez será accedido por una segunda aplicación para indicar la ubicación del vehículo, así como también esta mostrara los avisos cuando el vehículo recolector no realice su recorrido, vaya retrasado, cuando este cerca y a qué hora estará llegando a la zona de recolección para así sacar los residuos en tiempo y forma.

Android, Geolocalización, Recolección, Aplicación

Citation: BARRON-CASTILLO, Jorge Alfredo, HERNÁNDEZ-LUNA, Aldo, TORRES-JIMÉNEZ, Jacinto and LUNA-CARRASCO, Claudia Yadira. Android application for tracking the garbage collection vehicle in Huauchinango Puebla. Journal Information Technology. 2022. 9-28: 1-6

* Author Correspondence (e-mail: aldo.hl@huauchinango.tecnm.mx)
† Researcher contributing as first author.

Anexo 2 Participación en congreso internacional interdisciplinario de energías renovables, mantenimiento industrial, mecatrinónica e informática (CIERMMI 2022).

CIERMMI 2022

OTORGA LA PRESENTE

CONSTANCIA

A

BARRON CASTILLO, Jorge Alfredo

Por su participación en la

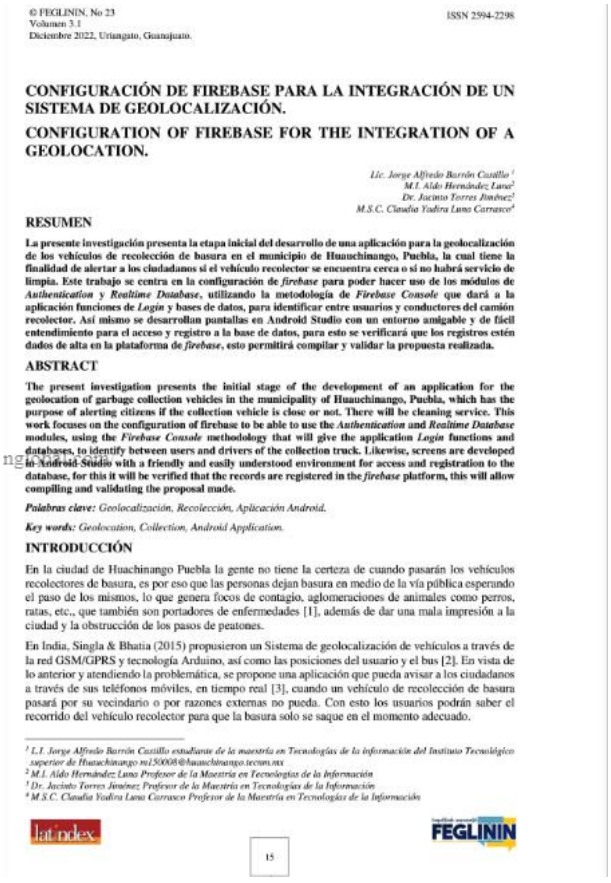
7a Edición del Congreso Internacional Interdisciplinario de Energías Renovables, Mantenimiento Industrial, Mecatrónica e Informática, CIERMMI 2022, Celebrado los días 26, 27 y 28 de octubre del año 2022, en San Juan del Río, Querétaro. México.

Dr. Ángel Marroquín de Jesús
PRESIDENTE
CIERMMI 2022.

Ing. Paola Gómez Vega
COORDINADORA
CIERMMI 2022.

Ing. Luis E. Cruz Carpio
COORDINADOR
CIERMMI 2022.

Anexo 3 Publicación de artículo en revista indexada FEGLININ, https://federacionglobal.com/FEGLININ/No23/dic2022/vol-3.1/access.html



© FEGLININ, No 23
Volumen 3.1
Diciembre 2022, Uriangato, Guanajuato.

ISSN 2594-2298

CONFIGURACIÓN DE FIREBASE PARA LA INTEGRACIÓN DE UN SISTEMA DE GEOLOCALIZACIÓN.

CONFIGURATION OF FIREBASE FOR THE INTEGRATION OF A GEOLOCATION.

Lic. Jorge Alfredo Barrón Castillo [1]
M.I. Aldo Hernández Luna [2]
Dr. Jacinto Torres Jiménez [3]
M.S.C. Claudia Yadira Luna Carrasco [4]

RESUMEN

La presente investigación presenta la etapa inicial del desarrollo de una aplicación para la geolocalización de los vehículos de recolección de basura en el municipio de Huauchinango, Puebla, la cual tiene la finalidad de alertar a los ciudadanos si el vehículo recolector se encuentra cerca o si no habrá servicio de limpia. Este trabajo se centra en la configuración de *firebase* para poder hacer uso de los módulos de *Authentication* y *Realtime Database*, utilizando la metodología de *Firebase Console* que dará a la aplicación funciones de *Login* y bases de datos, para identificar entre usuarios y conductores del camión recolector. Así mismo se desarrollan pantallas en Android Studio con un entorno amigable y de fácil entendimiento para el acceso y registro a la base de datos, para esto se verificará que los registros estén dados de alta en la plataforma de *firebase*, esto permitirá compilar y validar la propuesta realizada.

ABSTRACT

The present investigation presents the initial stage of the development of an application for the geolocation of garbage collection vehicles in the municipality of Huauchinango, Puebla, which has the purpose of alerting citizens if the collection vehicle is close or not. There will be cleaning service. This work focuses on the configuration of firebase to be able to use the *Authentication* and *Realtime Database* modules, using the *Firebase Console* methodology that will give the application *Login* functions and databases, to identify between users and drivers of the collection truck. Likewise, screens are developed in Android Studio with a friendly and easily understood environment for access and registration to the database, for this it will be verified that the records are registered in the *firebase* platform, this will allow compiling and validating the proposal made.

Palabras clave: *Geolocalización, Recolección, Aplicación Android.*

Key words: *Geolocation, Collection, Android Application.*

INTRODUCCIÓN

En la ciudad de Huachinango Puebla la gente no tiene la certeza de cuando pasarán los vehículos recolectores de basura, es por eso que las personas dejan basura en medio de la vía pública esperando el paso de los mismos, lo que genera focos de contagio, aglomeraciones de animales como perros, ratas, etc., que también son portadores de enfermedades [1], además de dar una mala impresión a la ciudad y la obstrucción de los pasos de peatones.

En India, Singla & Bhatia (2015) propusieron un Sistema de geolocalización de vehículos a través de la red GSM/GPRS y tecnología Arduino, así como las posiciones del usuario y el bus [2]. En vista de lo anterior y atendiendo la problemática, se propone una aplicación que pueda avisar a los ciudadanos a través de sus teléfonos móviles, en tiempo real [3], cuando un vehículo de recolección de basura pasará por su vecindario o por razones externas no pueda. Con esto los usuarios podrán saber el recorrido del vehículo recolector para que la basura solo se saque en el momento adecuado.

[1] *L.I. Jorge Alfredo Barrón Castillo estudiante de la maestría en Tecnologías de la Información del Instituto Tecnológico superior de Huauchinango m150008@huauchinango.tecnm.mx*
[2] *M.I. Aldo Hernández Luna Profesor de la Maestría en Tecnologías de la Información*
[3] *Dr. Jacinto Torres Jiménez Profesor de la Maestría en Tecnologías de la Información*
[4] *M.S.C. Claudia Yadira Luna Carrasco Profesor de la Maestría en Tecnologías de la Información*

Anexo 4 Participación en congreso internacional de innovación en ingeniería industrial, gestión y computación en la era digital (INGECO).

TECNOLÓGICO NACIONAL DE MÉXICO

INGECO
CONGRESO DE INNOVACIÓN

La Red de Cuerpos Académicos
Optimización de Procesos con la Industria 4.0

OTORGA EL PRESENTE

RECONOCIMIENTO

A:

Jorge Alfredo Barrón Castillo

Por su participación con la ponencia titulada

"CONFIGURACIÓN DE FIREBASE PARA LA INTEGRACIÓN DE UN SISTEMA DE GEOLOCALIZACIÓN DEL SERVICIO DE LIMPIA EN HUAUCHINANGO PUEBLA"

En el Segundo Congreso Internacional de Innovación en Ingeniería Industrial, Gestión y Computación en la Era Digital, INGECO 2022
con sede en el **Instituto Tecnológico Superior del Sur de Guanajuato**.
Uriangato, Guanajuato, México, celebrado del 08 al 11 de noviembre de 2022.

Dra. Minerva Cristina García Vargas
Presidenta de la Red de Cuerpos Académicos Optimización de Procesos con la Industria 4.0

M.I. Carlos Romero Villegas
Director General
del Instituto Tecnológico Superior del Sur de Guanajuato

ITSCH · Instituto Tecnológico de Zitácuaro · TESVB Tecnológico de Estudios Superiores Valle de Bravo · ITSUR · Universidad Autónoma de Chihuahua · Optimización Industria 4.0

Bibliografía.

[1] Astudillo León, Juan Pablo, Delgado Tello, Edgar Gustavo. (Abr-2012). Sistema de localización monitoreo y control vehicular basado en los protocolos GPS/GSM/GPRS. Universidad Tecnológica Salesiana.

[2] Acán, L. G. G. (Julio – 2018). Evaluación y diseño de un sistema de rastreo satelital para el monitoreo y control de las rutas terrestres asignadas diariamente en tiempo real. Escuela Superior Politécnica De Chimborazo.

[3] Coello, B. J. M. (2015). Diseño e implementación de prototipo de un sistema de seguridad para el encendido y rastreo satelital de un vehículo. Universidad Técnica Estatal De Quevedo.

[4] Cabezas, A. L. M. (2021). Análisis e Implementación de un Sistema de Rastreo Satelital aplicado a mascotas mediante software libre con tecnología GPS y GSM. Universidad Católica De Santiago De Guayaquil.

[5] Cerda, W. L. T. (2019). Diseño de un prototipo electrónico para el encendido del vehículo renault logan, mediante tecnología nfc y rastreo satelital-gps. Escuela Superior Politécnica De Chimborazo.

[6] Jorge Alfredo Barrón Castillo, Aldo Hernández Luna, Jacinto Torres Jiménez, Claudia Yadira Luna Carrasco. (2022). Aplicacion Android para el rastreo del vehículo recolector de basura en Huauchinango Puebla. ECOFRAN, 1-6.

[7] Jorge Alfredo Barron Castillo, Aldo Hernández Luna, Jacinto Torres Jiménez, Claudia Yadira Luna Carrasco. (Diciembre 2022). Configuración de Firebase para la integración de un sistema de geolocalozación. FEGLININ, 15–21.

[8] Wikipedia, C. (2023, octubre 21). GPS. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=GPS&oldid=154773868

[9] rastrearcelularya. (Diciembre 2021). ¿Cómo funciona el GPS del celular? rastrearcelularya. https://rastrearcelularya.com/como-funciona-el-gps-del-celular/

[10] sistemas. (s/f). Definición de GSM. sistemas.com. Recuperado el 11 de enero de 2023, de https://sistemas.com/gsm.php

[11] Vitel Global Communications. (s/f). What is GSM Technology. vitelglobal.in. Recuperado el 11 de enero de 2023, de https://www.vitelglobal.in/blog/what-is-gsm-technology/

[12] sistemas. (s/f). Definición de GPRS. sistemas.com. Recuperado el 2 de enero de 2023, de https://sistemas.com/gsm.php

[13] BasuMallick, C. (2022, Julio). What Is GPRS (General Packet Radio Service)? Meaning, Working, Advantages, and Applications. spiceworks.com. https://www.spiceworks.com/tech/networking/articles/what-is-gprs/

[14] Equipo editorial, Etecé. (2021, Agosto). Base de datos. concepto.de. https://concepto.de/base-de-datos/

[15] ORACLE. (s/f). ¿Qué es NoSQL? oracle.com. Recuperado el 15 de junio de 2023, de https://www.oracle.com/mx/database/nosql/what-is-nosql/

[16] Giraldo, V. (2019, abril). ¿Ya conoces Firebase? La herramienta de desarrollo y análisis de aplicaciones mobile. rockcontent.com. https://rockcontent.com/es/blog/que-es-firebase/

[17] Fernández, C. (2021, mayo 26). UX en las apps y la importancia del diseño. ABAMobile. https://abamobile.com/web/diseno-y-experiencia-de-usuario-ux-en-aplicaciones/

[18] Naranjo, M. (2022, octubre 4). Android: historia, versiones, Google Play y todas sus novedades. Computer Hoy. https://computerhoy.com/reportajes/tecnologia/android-historia-versiones-google-play-todas-novedades-1134319

[19] Introducción a Android Studio. (s/f). Android Developers. Recuperado el 24 de octubre de 2023, de https://developer.android.com/studio/intro?hl=es-419

[20] González, S. M. (2023, abril 6). ¿Qué es Java? Definición, características y usos de Java. Tech Krowd. https://techkrowd.com/programacion/java/que-es-java/

[21] Yeeply. (2022, Diciembre 6). Crear un prototipo de app móvil: cómo definir tu aplicación. Yeeply. https://www.yeeply.com/blog/como-definir-tu-aplicacion-movil-hacer-prototipo-de-app/

[22] Muradas, Y. (2018, marzo 8). Conoce las 3 metodologías ágiles más usadas. Openwebinars.net. https://openwebinars.net/blog/conoce-las-3-metodologias-agiles-mas-usadas/

[23] Firebase Realtime Database. (s/f). Firebase. Recuperado el 24 de octubre de 2023, de https://firebase.google.com/docs/database?hl=es-419

[24] Calidad básica de las apps. (s/f). Android Developers. Recuperado el 24 de octubre de 2023, de https://developer.android.com/docs/quality-guidelines/core-app-quality?hl=es-419

[25] Sanchez, D. (2020). Truck moving. LottieFiles. https://lottiefiles.com/animations/truck-moving-yTOJF6OXuP

[26] Barkade, A. (2021). Menu. LottieFiles. https://lottiefiles.com/animations/menu-Ho18YnpqAf

[27] Iconos, logos, símbolos de Usuario - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/usuario

[28] Iconos, logos, símbolos de Sobre circulado - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/sobre-circulado

[29] Iconos, logos, símbolos de Password - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/password

[30] Iconos, logos, símbolos de Celular - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/celular

[31] Iconos, logos, símbolos de Hombre mujer - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/hombre-mujer

[32] Iconos, logos, símbolos de Apartamentos - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/apartamentos

[33] Iconos, logos, símbolos de Rh plus - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/rh-plus

[34] Iconos, logos, símbolos de Licencia - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/licencia

[35] Iconos, logos, símbolos de Hospital - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/hospital

[36] Bhunwal, M. (2022). Login. LottieFiles. https://lottiefiles.com/animations/login-n9jR7ixeZN

[37] Firmino, N. (2021). Password Security. LottieFiles. https://lottiefiles.com/animations/password-security-6dYDYx3frD

[38] Iconos, logos, símbolos de Camioneta - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/camioneta

[39] Iconos, logos, símbolos de Camion basura - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/camion-basura

[40] Iconos, logos, símbolos de Usuario icono - descarga gratuita PNG, SVG. (s/f). Icons8. Recuperado el 24 de octubre de 2023, de https://iconos8.es/icons/set/usuario-icono

[41] SAMPAYO-RODRÍGUEZ, C. J., GONZÁLEZ-AMBRIZ, R., GONZÁLEZ-MARTÍNEZ, B. A., & ALDANA-HERRERA, J. (2022). Processor and memory performance with design patterns in a native Android application. Journal Applied Computing, 6-18.