UNIVERSIDAD CATÓLICA DE CUENCA

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLÓGICA.

CARRERA DE INGENIERÍA EN SISTEMAS DE INFORMACIÓN

PROTOTIPO DE UN APLICATIVO MÓVIL DE GEOLOCALIZACIÓN DEL RECOLECTOR DE DESECHOS EN BASE A SU RECORRIDO PARA LOS USUARIOS DE LA EMMAIPC-EP

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS DE INFORMACIÓN.

AUTOR: BRYAN STALIN SOLÓRZANO ESPINOZA.

DIRECTOR: ING. LUIS FERNANDO PINOS CASTILLO.

CAÑAR - ECUADOR

2022

DIOS, PATRIA, CULTURA Y DESARROLLO

UNIVERSIDAD CATÓLICA DE CUENCA

Comunidad Educativa al Servicio del Pueblo

UNIDAD ACADÉMICA DE INFORMÁTICA, CIENCIAS DE LA COMPUTACIÓN E INNOVACIÓN TECNOLÓGICA.

CARRERA DE INGENIERÍA EN SISTEMAS DE INFORMACIÓN

PROTOTIPO DE UN APLICATIVO MÓVIL DE GEOLOCALIZACIÓN DEL RECOLECTOR DE DESECHOS EN BASE A SU RECORRIDO PARA LOS USUARIOS DE LA EMMAIPC-EP.

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS DE INFORMACIÓN.

AUTOR: BRYAN STALIN SOLÓRZANO ESPINOZA.

DIRECTOR: ING. LUIS FERNANDO PINOS CASTILLO.

CAÑAR - ECUADOR

2022

DIOS, PATRIA, CULTURA Y DESARROLLO

DECLARACIÓN

Yo, Bryan Stalin Solórzano Espinoza, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que he consultado las referencias bibliográficas que se incluyen en este documento.

La Universidad Católica de Cuenca extensión Cañar puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y la Normativa actual de la institución.

Sr. Bryan Stalin Solórzano Espinoza

Alumno

UNIVERSIDAD CATÓLICA DE CUENCA.

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por el Est. Bryan Stalin Solórzano Espinoza, bajo mi supervisión.

Ing. Luis Fernando Pinos Castillo

DIRECTOR DEL TRABAJO INVESTIGATIVO

UNIVERSIDAD CATÓLICA DE CUENCA.

DEDICATORIA

El presente trabajo de titulación lo dedico primeramente a mi madre ya que ella fue el pilar fundamental que me ayudo a motivarme y seguir adelante a pesar de los obstáculo y adversidades, fue la persona que siempre estuvo ahí conmigo en todo momento, formándome con sus buenos sentimientos, enseñanzas y valores.

También a mi padre que desde el cielo me ilumina y es mi inspiración para alcanzar todas mis metas y objetivos que me plantee durante mi carrera profesional.

A mis abuelitos que me apoyaron y me sirvieron de guía y orientación durante el trayecto de mis estudios, a través de sus buenos consejos, aportando a mi formación profesional y moral como ser humano.

AGRADECIMIENTO:

En primer lugar, a Dios por darme salud y sabiduría para cumplir mis sueños y a mi madre, a mi padre desde el cielo y a mis abuelitos por su amor, cariño y apoyo incondicional.

A esta prestigiosa casa de estudio y catedráticos de la Unidad Académica de Tecnologías de la Información, por la impartición de sus conocimientos y buenos consejos.

Al Ing. Luis Pinos que me guio y aconsejo como tutor durante el desarrollo y culminación de mi trabajo de titulación.

De la misma manera, a la empresa EMMAIPC-EP por brindarme la oportunidad de poder realizar mi trabajo de titulación dentro de la misma.

APROBACIÓN DE TRIBUNAL DE GRADO

El tribunal designado por el honorable consejo directivo de la Universidad Católica de Cuenca Extensión Cañar, Facultad de Ingeniería de Sistemas instalado para receptar la sustentación del trabajo final de investigación con el tema: “PROTOTIPO DE UN APLICATIVO MÓVIL DE GEOLOCALIZACIÓN DEL RECOLECTOR DE DESECHOS EN BASE A SU RECORRIDO PARA LOS USUARIOS DE LA EMMAIPC-EP”, transcurrido el tiempo reglamentario procede a consignar la calificación de (_____/100).

Cañar, _____de_____ del 2022

PRESIDENTE	**DIRECTOR**
DELEGADO	**SECRETARIO**

RESUMEN

El presente proyecto tiene como objetivo el diseño de un prototipo móvil de geolocalización del recolector de desechos en base al recorrido para los usuarios de la empresa EMMAIPC-EP, con la finalidad de facilitar la identificación de la llegada del recolector a su ubicación, el tipo de desecho y horario correspondiente a su ruta de recorrido. El desarrollo del prototipo se ha llevado a cabo mediante la metodología Mobile-D haciendo uso de herramientas como Android Studio y Java como su lenguaje de programación, además de la codificación de un API REST en lenguaje de programación PHP para la interacción con la base de datos y el desarrollo de un sistema de información de escritorio en lenguaje de programación Java, que permite para crear, actualizar, eliminar, buscar y visualizar registros de rutas de recorrido, vehículos recolectores, usuarios, zonas, sub zonas y campos adicionales de recolección. Almacenando estos datos en MySQL, utilizando el servidor Traccar Open Source para enviar y obtener los datos de geolocalización del dispositivo GPS. Los resultados del prototipo permiten identificar que en caso de implementación del prototipo de aplicativo móvil de geolocalización de recolector de desechos beneficia a la empresa EMMAIPC-EP y a los usuarios en general.

***Palabras Clave*:** prototipo móvil, java, traccar, mobile-d, php.

ABSTRACT

This article seeks to design a mobile geolocation prototype for garbage trucks based on their routes and to be used for the EMMAIPC-EP company users to identify the truck's arrival at its location, the type of waste, and the appropriate schedule according to its route. The prototype has been carried out through the Mobile-D methodology, and as its programming language tools such as Android Studio and Java were used, a REST API code in PHP programming language and the development of a desktop information system in Java programming language were also used to interact with the database. It allows to create, update, delete, search and display records of travel routes, collection vehicles, users, zones, sub-zones and additional collection fields; storing this data in MySQL using the Traccar Open Source server to send and get the geolocation data from the GPS device. Results show that once the mobile geolocation prototype for garbage trucks is carried out, the EMMAIPC-EP company and users, in general, would benefit.

***Keywords*:** mobile prototype, java, traccar, mobile-d, php.

ÍNDICE DE CONTENIDO

TABLA DE ILUSTRACIONES

ARTÍCULO CIENTÍFICO

Prototipo de un aplicativo móvil de geolocalización del recolector de desechos en base a su recorrido para los usuarios de la EMMAIPC-EP

Prototype of a mobile application for the geolocation of the waste collector based on its route for the users of the EMMAIPC-EP

Bryan Solórzano
Estudiante, Universidad Católica de Cuenca, Ecuador
bssolorzanoe51@est.ucacue.edu.ec
Luis Pinos
Docente, Universidad Católica de Cuenca, Ecuador
lfpinosc@ucacue.edu.ec
ORCID

RESUMEN

El presente proyecto tiene como objetivo el diseño de un prototipo móvil de geolocalización del recolector de desechos en base al recorrido para los usuarios de la empresa EMMAIPC-EP, con la finalidad de facilitar la identificación de la llegada del recolector a su ubicación, el tipo de desecho y horario correspondiente a su ruta de recorrido. El desarrollo del prototipo se ha llevado a cabo mediante la metodología Mobile-D haciendo uso de herramientas como Android Studio y Java como su lenguaje de programación, además de la codificación de un API REST en lenguaje de programación PHP para la interacción con la base de datos y el desarrollo de un sistema de información de escritorio en lenguaje de programación Java, que permite para crear, actualizar, eliminar, buscar y visualizar registros de rutas de recorrido, vehículos recolectores, usuarios, zonas, sub zonas y campos adicionales de recolección. Almacenando estos datos en MySQL, utilizando el servidor Traccar Open Source para enviar y obtener los datos de geolocalización del dispositivo GPS. Los resultados del prototipo permiten identificar que en caso de implementación del prototipo de aplicativo móvil de geolocalización de recolector de desechos beneficia a la empresa EMMAIPC-EP y a los usuarios en general.

Palabras clave: prototipo móvil, java, traccar, mobile-d, php.
Abstract: This article seeks to design a mobile geolocation prototype for garbage trucks based on their routes and to be used for the EMMAIPC-EP company users to identify the truck's arrival at its location, the type of waste, and the appropriate schedule according to its route. The prototype has been carried out through the Mobile-D methodology, and as its programming language tools such as Android Studio and Java were used, a REST API code in PHP programming language and the development of a desktop information system in Java programming language were also used to interact with the database. It allows to create, update, delete, search and display records of travel routes, collection vehicles, users, zones, sub-zones and additional collection fields; storing this data in MySQL using the Traccar Open Source server to send and get the geolocation data from the GPS device. Results show that once the mobile geolocation prototype for garbage trucks is carried out, the EMMAIPC-EP company and users, in general, would benefit.
Keywords: mobile prototype, java, traccar, mobile-d, php.

INTRODUCCIÓN

Las Tecnologías de la Información y Comunicación han tenido un desarrollo exponencial a nivel mundial; facilitando a los usuarios, empresas y organizaciones el disponer de dispositivos tecnológicos con múltiples aplicaciones en los diferentes ámbitos utilizando varias tecnologías que benefician actividades. Una de las tecnologías más utilizadas actualmente es la geolocalización, misma que permite obtener la ubicación geográfica de un objeto en tiempo real. Es por eso que varias empresas han adoptado esta tecnología como una técnica de marketing digital para mejorar experiencias de los usuarios.

La empresa EMMAIPC-EP, ofrece servicio de aseo Integral en Cañar, Biblián, El Tambo, y Suscal, contando con horarios de recolección por zonas. Sin embargo, se puede evidenciar una gran cantidad de desechos que se encuentran botados en las calles, específicamente en el cantón Cañar, generando como resultado evidencias de que los actuales mecanismos o procedimientos de recolección de desechos no son lo suficientemente eficientes. Es por ello que surge la idea de diseñar un prototipo de aplicativo móvil que permita a los usuarios de la EMMAIPC-EP, la visualización grafica de las rutas de recorridos, horarios de recolección y días de desecho orgánico e inorgánico correspondiente a la ruta. Para ello es importante iniciar realizando definiciones sobre las herramientas y metodologías que permiten llevar a cabo el presente proyecto, además de proyectos similares. Tal es el caso de Quintana E. (2018) quien presenta un sistema de geolocalización de alerta de recojo de residuos sólidos utilizando el gps para

obtener coordenadas de latitud y longitud del vehículo y del usuario aplicando la metodología V para su desarrollo.

Gonzáles et al. (2021) desarrolla un sistema de información web para la gestión de datos referente a la recolección de desechos domiciliarios aplicando geolocalización en el GAD Municipal del Cantón Azogues, utilizando la metodología SCRUM y la herramienta .NET para su desarrollo, concluyendo que el sistema sirve como un medio informativo para la colectividad sobre los servicios que brinda la empresa.

Geolocalización

La geolocalización es una nueva dimensión que ha surgido con la aparición del Internet y dispositivos móviles permitiendo situar a personas u objetos en el espacio a través de coordenadas. Permite unir el mundo físico con el digital siendo la geolocalización una herramienta de comunicación (Silva & Aparicio, 2021).

Tipos de Geolocalización:

- *Sistema de Posicionamiento Global (GPS):* Es una tecnología de navegación basado en satélites, funciona ante cualquier condición climatológica a nivel mundial. El receptor GPS puede establecer con gran precisión la posición del usuario, calculándola en tres dimensiones como la longitud, latitud y altitud (Torres & Macias, 2021).
- *Sistema Global para comunicaciones móviles (GSM):* Es un sistema utilizado por la red de telefonía, la cual consiste en la combinación de una red de estaciones transmisoras o receptoras de radio y una serie de centrales telefónicas de conmutación de primer y quinto nivel, permitiendo la comunicación entre dispositivos móviles y teléfonos de red fija (CABRERA, 2018).

Aplicativos móviles

Enríquez et al. (2021) menciona que los aplicativos móviles son piezas de software que se ejecutan en teléfonos móviles y tabletas, las aplicaciones son desarrolladas en base a las limitaciones de los dispositivos. Existen dos categorías que permiten clasificar a las aplicaciones móviles:

- *Aplicaciones nativas:* Desarrolladas únicamente para un tipo de dispositivo y sus S.O, accediendo a funciones propias del dispositivo.
- *Aplicaciones web*: Se ejecutan en servidores, y son accedidas desde cualquier navegador web (Gabriel & Isabel, 2021).

Sistema Operativo Android

Ilustración 1. Sistema Operativo Android. Fuente: (Logo, 2019)

Android es un sistema operativo diseñado para dispositivos táctiles (celulares, relojes inteligentes, televisores, automóviles), creado por Google basado en el sistema Linux.

Este sistema operativo tiene acceso a recursos que pueden ser gestionados tales como controladores de pantalla, cámara, memoria flash, entre otros, siendo su código fuente principal "Android Open Source Project", destacado por ser el sistema operativo más utilizado a nivel mundial (Manuel Báez, 2018) (Zheng, 2022).

Android Studio

Android es un entorno de desarrollo diseñado para una variedad de teléfonos móviles.

"Android Studio cuenta con una estructura y este compuesto por aplicaciones que utilizan el lenguaje de programación Java, contiene bibliotecas para almacenar datos de forma local de bases de datos relacionales SQLite" (Mainez, 2018).

Ofrece elementos que permiten incrementar eficiencia al momento de crear aplicaciones:

- Un marco o sistema de compilación adaptable dependiente de Gradle.
- Un emulador rápido y cargado en elementos.
- Un entorno unificado donde se puede producir para todos los dispositivos con sistema Android.
- Aplicación de cambio para incrustar cambios de código y activos en la aplicación en ejecución sin reiniciarla.
- Integración con GitHub y diseños de código para ayudarle a ordenar las funciones normales de la aplicación y además importar código de prueba.
- Compatibilidad con C++ y NDK.

- Ayuda integrada para Google Cloud Platform, que funciona con la incorporación de Google Cloud Messaging y App Engine (Android, 2021).

Librerías de Android Studio

- Librería Retrofit: Es una librería para Android y Java que permite realizar peticiones al servidor a través de los verbos HTTP (Get, Post, Put, Patch, Delete y Head) de manera síncrona o asíncrona (Mejía, Corea, & Frank, 2019).
- Librería Gson: Es un API de código abierto que permite realizar conversaciones entre objetos de tipo Java y JSON, es utilizado conjuntamente con la librería Retrofit para parsear los datos obtenidos (Morcillo, 2019).
- Material Design: Es un lenguaje visual para generar aplicaciones en Android y directrices de accesibilidad proporcionando recomendaciones que guían su diseño de forma atractiva de manera eficiente e interactiva para con el usuario (Solecki , y otros, 2020).
- Google Maps API: Es el servicio gratuito de Google establecido en un servidor de mapas que permite la visualización de imágenes de cartografías desplazables, así como fotos de satélites de todo el mundo. Siendo posible incorporarlo al desarrollar aplicaciones web y móviles (Montero, 2022).

Lenguajes de Programación para Desarrollo de Aplicaciones Móviles

Los lenguajes de programación son lenguajes que utilizan las computadoras permitiendo el desarrollo de programas de software, aplicaciones, entre otros (Villalba, Moraleda, & González, 2021).

Java

Según Villacres (2022), Java es un lenguaje de programación basado en C y C++, se puede ejecutar sobre cualquier máquina, es un lenguaje orientado a objetos multiplataforma, obtiene una variedad de herramientas que se pueden utilizar de forma gratuita. Se pueden crear aplicaciones para móviles y para ordenadores.

Este lenguaje es ideal para el desarrollo de aplicaciones móviles ya que es simple y tiene una API excelente.

Kotlin

Kotlin es considerado como un lenguaje de programación estático de código abierto que utiliza estructuras y normas de codificación entendibles y fáciles de utilizar siendo flexible (APOLONIA, 2021).

Ilustración 2. Java VS Kotlin. Fuente: (Mariana Berga, 2021)

Cuadro Comparativo de los Lenguajes de Programación para Desarrollo de Aplicaciones Móviles

Se utiliza los indicadores (compatibilidad con SDK de plataforma, excepción marcada, campos no privados, excepciones, tipos de comodines, programación Reactiva Funcional, miembros estáticos), ya que permiten identificar el mejor lenguaje de programación para el desarrollo de aplicaciones móviles en cuanto a rendimiento, eficacia y características de programación.

Tabla 1. Cuadro Comparativo de los Lenguajes de Programación para Desarrollo Aplicaciones Móviles. Fuente: Autoría Propia.

	Java	**Kotlin**
Compatibilidad con SDK de plataforma	Si	Si
Excepción marcada	Si	No
Campos no privados	Si	Si
Excepciones	Si	No
Tipos de comodines	Si	No
Programación Reactiva Funcional	No	Si

Miembros estáticos	Si	No

En base a un estudio realizado por Martínez D. (2021) en el cual realiza una matriz comparativa de los lenguajes de programación para el desarrollo de aplicaciones móviles mediante el uso de indicadores mencionados en la tabla 1 indica que Java es el lenguaje más utilizado en la actualidad debido a que brinda una mejor compatibilidad SDK. Además, es un lenguaje que admite su ejecución en tiempo real que se adapta al desarrollo de aplicaciones móviles. Mediante el análisis comparativo se determina que Java cumple con la mayoría de los criterios evaluados, por ende, será el lenguaje de programación a utilizar en el presente trabajo.

Lenguajes de Programación para el Desarrollo de API REST

PHP

Es un lenguaje de código abierto de alto nivel incluido en páginas web y ejecutado en el servidor que incluye etiquetas especiales de comienzo y final. Es un lenguaje de programación simple que permite procesar la información de formularios, generar páginas dinámicas o enviar y recibir cookies (Fossati, 2018).

PHP maneja solicitudes HTTP y provee un gran conjunto de funciones con una gran escalabilidad como mecanismos de seguridad basados en técnicas de cifrado como JSON Web Token, por ello permite construir una API REST (Estrada, Villatoro, García, & Vázquez, 2022).

Java

Es un lenguaje de programación orientado a objetos multiplataforma y multitarea, soporta la encapsulación, herencia y polimorfismo. Teniendo como ventaja que, en la máquina virtual de Java, se aporta a la portabilidad al lenguaje de marca para diferentes arquitecturas, es decir que un programa escrito en Windows puede interpretarse en Linux (Arenaza, 2019).

Java permite la construcción de una API REST mediante la versión API Java, para ello se puede utilizar framework como SpringTool, contando con un gran rendimiento.

Cuadro Comparativo de los Lenguajes de Programación para el Desarrollo de un API REST

Los indicadores presentados en la tabla 2 han sido seleccionados ya que permiten determinar el mejor lenguaje para desarrollar un API REST, analizando ventajas y desventajas de PHP y Java, además de los frameworks que permiten realizar interacción con los servicios web.

Tabla 2. Cuadro Comparativo de los Lenguajes de Programación para el Desarrollo de un API REST. Fuente: Autoría Propia.

	PHP	Java
Bases de datos	Soporta una gran cantidad de bases de datos.	Soporta gran cantidad de base de datos, además elimina tareas repetitivas
Framework	Laravel, lúmenes, Guzzle, Epifanía	Framework Spring, TomEE, Restlet
Protocolos	Cuenta con soporte para comunicarse con otros servicios a través de protocolos como LDAP, IMAP, SNMP, NNTP, POP3 y WDDX para el intercambio de datos entre lenguajes de programación web	Protocolos SSL y TLS
Rendimiento	Permite acelerar el proceso de escritura de aplicaciones, reduciendo el consumo de memoria.	Más rápido y eficiente para escribir aplicaciones empresariales
Ventajas	Útil para el desarrollo de páginas web y desarrollo de aplicaciones complejas	Utiliza un lenguaje simple, interpretado y compilado.
Desventajas	Requiere de un servidor para su funcionamiento	El código de Java es convertido a un código intermedio antes de convertirlo a código máquina

Caiza et al. (2021) realiza un análisis entre el lenguaje de programación Java y PHP para la construcción de un API REST utilizando los indicadores mencionados en la tabla 2 del presente documento. Determinando a PHP como el mejor lenguaje debido a su eficacia en el desarrollo web permitiendo generar cualquier texto en diferentes archivos generando contenido dinámico. Es por ello que para la construcción del API REST se utilizará PHP.

IDE de Desarrollo API REST

*Visual Studio Code**:** Es un editor de código fuente ligero* compatible con varios lenguajes de programación, incluyendo varios editores para diferentes documentos.

Según Plainer (2020) esta herramienta facilita una utilidad para descargar y gestionar extensiones para personalizarlas y fortalecer el sistema. Ofrece código de color, un conjunto de atajos de código siendo una herramienta flexible, contiene control de Git integrado, resultado de sintaxis, fragmentos de código y su refactorización.

Bases de Datos para el Desarrollo de Aplicaciones Móviles

MySQL

Es un sistema de gestión de base de datos Open Source con un alto rendimiento y fiabilidad, dirigida para proyectos web y tradicionales, ofreciendo múltiples protocolos de comunicación en la arquitectura cliente-servidor (Combaudon, 2018).

Ilustración 3. Base de Datos MySQL. Fuente: (AWS, 2022)

PostgreSQL

Es un sistema de base de datos que utiliza la estructura cliente-servidor, caracterizado por la estabilidad ya que es compatible con varios modelos de datos que permiten crear aplicaciones orientadas a objetos de forma escalable con un alto rendimiento (Garrido, López, & Constante, 2020).

Ilustración 4. Base de Datos PostgreSQL. Fuente: (KINSTA, 2022)

Cuadro Comparativo sobre las Bases de Datos para el Desarrollo de Aplicativos Móviles

Tabla 3. Cuadro Comparativo sobre las Bases de Datos para el Desarrollo de Aplicativos Móviles. Fuente: Autoría Propia.

	MySQL	**PostgreSQL**
Licencia	Licencia publica general de GNU	Licencia BSD
SQL	SI	SI
Transacciones	Si	Si
Procesamiento distribuido de consultas	Si	Si
Compatibilidad	Multiplataforma	Multiplataforma
Nro. Máximo de tablas (Capacidad)	16 TB	32 TB
Flexibilidad	Mayor flexibilidad	SI
Fiabilidad y Seguridad	Garantiza la protección de los datos gracias a capacidades como la recuperación inmediata y el autocommit.	Permite realizar copias de seguridad en línea
Replicación y clustering	Replicación asíncrona unidireccional	Replicación sincrónica

Los indicadores mencionados en la tabla 3, fueron obtenidos de la norma ISO 25010 misma que hace referencia al modelo de calidad de producto y en base al estudio realizado por Moreno et al. (2021) quienes realizan un análisis comparativo de rendimiento de los gestores de base de datos MySQL y PostgreSQL, evaluando algunos criterios mencionados, además de la eficiencia de desempeño determinando que MySQL tiene una sintaxis más flexible y consume menos recursos que PostgreSQL. Además, ofrece una velocidad y un rendimiento óptimo dentro del ámbito de aplicaciones web a través de consultas no complejas.

En base a lo expuesto anteriormente y al análisis comparativo, se puede ultimar que MySQL será apto para el desarrollo del prototipo.

Servidor de Geolocalización

Traccar

Es un servidor que proporciona un alto rendimiento y estabilidad en cualquier sistema operativo, ya sea en la nube o en las instalaciones. Permitiendo la selección de múltiples protocolos y rastreadores GPS de toda marca con una alta calidad, trabajando en tiempo real con una variedad de sensores e información adicional (Traccar, 2022).

Traccar Client APK

Esta plataforma de seguimiento por GPS, es compatible con Traccar Server, es una "versión oculta especial que contiene modificaciones para que la aplicación sea menos visible para el usuario, cambiando el nombre a configuración del dispositivo que permite consumir datos de geolocalización" (Traccar, 2022).

OpenGTS

OpenGTS es una plataforma de rastreo satelital de código abierto que facilita servicios de seguimiento GPS basados en la web que permite el rastreo de vehículos/activos (OpenGTS, 2020).

Cuadro Comparativo de los Servidores de Geolocalización

Tabla 4. Cuadro Comparativo de los Servidores de Geolocalización. Fuente: Autoría Propia.

	Traccar	**OpenGTS**
Dispositivos GPS	Posee compatibilidad con una gran variedad de dispositivos GPS del mercado	Posee compatibilidad limitada con dispositivos GPS del mercado
Herramientas	Traccar Client, Traccar Web, Traccar API, Traccar Manager	Basic GTS Enterprise GPS Tracking Web
Alertas	Permite alertas externas en casos de comportamiento de manejo severo, como eventos de exceso de velocidad, combustible y	Admite alertas por mensaje si el vehículo sale de la zona establecida

	mantenimiento, geo-cercas y muchos otros tipos de alertas.	
Compatibilidad	Multiplataforma	Multiplataforma
Servicio de Mapas personalizable	Map Layers, Google Map Layer	Soporte para OpenLayers / OpenStreetMap, Google Maps, Microsoft Virtual Earth y Mapstraction.
Informes	Admite informes sencillos de historial de ubicación, viaje, gráficos y resúmenes con proyección del mapa	Informes basados en XML, detallados y resumidos
Bing Maps	Si	No

Un análisis comparativo de servidores de geolocalización realizado por León et al. (2017), utilizan los indicadores mencionados anteriormente en la tabla 4 ya que permiten determinar de mejor manera la eficacia de los servidores de geolocalización y su rendimiento, mencionando a Traccar como el mejor al contar con un mayor número de características. Es así que, de la misma manera, en base al cuadro comparativo de servidores de Geolocalización, Traccar será utilizado ya que provee un alto rendimiento y estabilidad multiplataforma. Así como también, la posibilidad de la interacción con sus datos a través de su API REST, disponibilidad de otras herramientas complementarias y la compatibilidad con una gran cantidad de dispositivos GPS del mercado.

Metodologías para el desarrollo de Software

Mobile-D: Es una metodología que contiene un ciclo de desarrollo pequeño optimizado para un equipo de menos de diez desarrolladores que entreguen un aplicativo móvil funcional en poco tiempo menos de diez semanas, teniendo como fases las siguientes:

- *Exploración:* Se genera un plan y se establece las características del proyecto, definiendo los actores, el alcance y el establecimiento de proyectos.
- *Iniciación:* Los desarrolladores preparan e identifican los recursos necesarios definiendo la planificación inicial, el día de prueba y el día de salida.
- *Producción:* Se repite la programación de tres días, preparando pruebas de interacción, desarrollando el código e integrándolo con repositorios existentes.
- *Estabilización:* Fase que permite llevar el proceso de integración para asegurar el sistema y su funcionamiento

- *Pruebas del sistema:* Tiene como fin disponer del sistema funcional eliminando errores encontrados (Santiago & Isarael, 2022).

Metodología en cascada: La metodología en cascada denominada también Waterfall, es usada en proyectos de desarrollo de software de forma secuencial. Contiene cinco etapas, las cuales deben ejecutarse una tras otra, en un determinado orden de arriba hacia abajo. Siendo sus fases:

- *Requerimientos del Software:* se determina el problema, se analiza y se definen los requisitos y objetivos que se pretende obtener
- *Diseño*: La información recolectada se realiza a través de una estructura algorítmica de datos.
- *Implementación*: El diseño del sistema se implementa mediante una tecnología seleccionada.
- *Integración*: En esta tarea se encuentra los fallos para que estos sean corregidos.
- *Mantenimiento*: Se corrige los errores que no se detectaron antes, adaptándose al entorno de trabajo, mejorando la aplicación en beneficio de los interesados (Gallego, 2022, págs. 16-17).

Cuadro Comparativo de las Metodologías para el Desarrollo de Software

Tabla 5. Cuadro Comparativo de las Metodologías para el Desarrollo de Software. Fuente: Autoría Propia.

	Mobile-D	**Cascada**
Etapas	1. Exploración 2. Inicialización 3. Producción 4. Estabilización 5. Prueba y reparación	1. Análisis de requerimientos 2. Diseño 3. Implementación 4. Pruebas 5. Mantenimiento
Tipo de proyecto de Software	Software móvil	Proyectos de reingeniería, aplicaciones, y proyectos compuestos con requerimientos claros.
Características	Las etapas de dividen en ciclos de tres días para planificar, para ejecutar el proyecto y para presentar los resultados.	Se puede retroceder en las etapas, los requerimientos son específicos.

Programador	Interactúa con el usuario de manera periódica.	Interactúa con el usuario solamente en la toma de información.

Del estudio que realiza Alvarado et al. en el año (2022) se determinaron las 2 metodologías más usadas actualmente para el desarrollo de aplicaciones móviles (Mobile-D, Cascada) dicho estudio analiza cada uno de los indicadores mencionados en la tabla 5, en donde obtiene que Mobile-D permite realizar proyectos en un período de tiempo corto interactuando con los usuarios finales a diferencia de la metodología en cascada que excluye a los clientes y dificulta los cambios hasta obtener el software final. Es por eso que se establece que la metodología a utilizar para el desarrollo de software móvil es Mobile-D, ya que contiene un ciclo de desarrollo pequeño e interactúa continuamente con el cliente y se enfoca en el desarrollo de aplicaciones móviles.

METODOLOGÍA

Enfoque de la investigación

Para la elaboración del presente trabajo se han tomado en cuenta variables cualitativas y cuantitativas, que permiten la identificación de aspectos importantes para la elaboración del prototipo móvil.

Nivel de investigación

Siendo su nivel de investigación de carácter descriptivo en vista de que se realiza la recolección de información para el desarrollo adecuado del prototipo.

La metodología a utilizar para el desarrollo del prototipo es Mobile-D, metodología analizada en la tabla 5 del presente documento.

Población y muestra

El universo de la investigación está centrado únicamente en el Gerente y el encargado del Departamento de TI de la empresa EMMAIPC-EP, por lo cual no existe muestra.

El método de investigación es deductivo, ya que el proceso del desarrollo del prototipo móvil va de lo general a lo particular. Utilizando como instrumento de investigación el método de la entrevista.

Con la finalidad de determinar las necesidades de la empresa EMMAIPC-EP para la utilización de un aplicativo que permita visualizar la ruta de los carros recolectores de basura a sus

usuarios, se realizó un levantamiento de información mediante la aplicación de una entrevista, tal y como se muestra en la tabla 6.

Tabla 6. Entrevista Realizada a la Empresa EMMAIPC-EP de la Ciudad de Cañar. Fuente: Autoría Propia.

Pregunta	Respuesta
1. ¿Actualmente su empresa cuenta con un aplicativo móvil? ¿Cuál es la función que desempeña el aplicativo móvil?	Actualmente la empresa no cuenta con ningún aplicativo móvil para brindar servicios tecnológicos a los usuarios.
2. ¿Conoce usted sobre la utilización de los sistemas GPS y de los GIS?	Se tiene conocimiento acerca de los dispositivos GPS y GIS, debido a que la empresa tiene implementado los dispositivos GPS dentro de cada uno de los vehículos recolectores para llevar el control de su ubicación.
3. ¿Está de acuerdo que los sistemas GIS pueden ayudar a ubicar la ubicación exacta en tiempo real de los vehículos recolectores?	Si ya que a través de herramientas tecnológicas permite ubicar de manera óptima los vehículos recolectores y realizar un seguimiento de manera segura, fácil y eficiente.
4. ¿Considera usted que es un problema para la empresa sacar los desechos fuera del horario planificado? ¿Considera usted que informando a tiempo la llegada del recolector se solucionaría este problema?	Si debido a que esto afecta realizar el proceso de recolección de desechos de una manera eficiente, a más de eso se pierde material de reciclaje. Si, se vería solventada esta problemática ya que los usuarios estarían pendientes de la llegada del vehículo recolector y les permitiría sacar a tiempo los desechos o incluso depositar directamente ya en el vehículo recolector que sería lo más óptimo.
5. ¿Cómo cree que los aplicativos móviles podrían servir para informar a los usuarios acerca de la llegada del vehículo recolector	A través de una interfaz amigable e intuitiva proporcionando al usuario la visualización de los horarios de recolección, días de desechos orgánicos e inorgánicos, rutas de recolección e identificando y notificando la distancia y tiempo estimado de llegada del vehículo recolector a la ubicación del usuario.

RESULTADOS

El desarrollo del prototipo se lleva a cabo a través de las siguientes fases de la metodología Mobile-D.

Exploración

Para el desarrollo del prototipo móvil de geolocalización del recolector de desechos en base a su recorrido para los usuarios de la EMMAIPC-EP es necesario definir a las personas involucradas en el proyecto:

- Usuarios de la empresa EMMAIPC-EP: Personas que se beneficiarán de las funcionalidades y harán uso del prototipo.
- Gerente de la empresa EMMAIPC-EP: Proporciona los requisitos iniciales para desarrollar el prototipo.
- Desarrollador: Persona encargada de realizar la codificación y desarrollo del prototipo del aplicativo móvil basado en los requisitos.

Especificación de requerimientos

Para ejecutar adecuadamente el prototipo móvil, se realizó una entrevista anexada en la metodología dirigida al gerente de la empresa EMMAIPC-EP y al director de TI. En donde se definieron los requerimientos especificados a continuación:

- ***Requerimientos funcionales:***

Tabla 7. Requerimientos Funcionales. Fuente: Autoría Propia.

Id	Requerimiento	Descripción	Prioridad
RF1	Visualización Horarios de Recolección	El usuario de la EMMAIPC-EP al ingresar a la pantalla principal del aplicativo selecciona la zona, sub zona y ruta correspondiente a su ubicación para visualizar los horarios de recolección pertenecientes a la ruta.	Alta
RF2	Visualización Días de Desecho Orgánico e Inorgánico	El usuario de la EMMAIPC-EP al ingresar a la pantalla principal del aplicativo selecciona la zona, sub zona y ruta correspondiente a su ubicación para visualizar los días correspondientes a los desechos orgánicos e inorgánicos pertenecientes a la ruta.	Alta

RF3	Visualización Gráfica Ruta de Recolección	El usuario de la EMMAIPC-EP al ingresar a la pantalla principal del aplicativo selecciona la zona, sub zona y ruta correspondiente a su ubicación, visualizando a su vez, de manera gráfica dentro del mapa la ruta de recolección.	Alta
RF4	Visualización Estado del Vehículo Recolector en Tiempo Real	El usuario de la EMMAIPC-EP una vez que selecciona la ruta de recolección correspondiente pasará a la pantalla de geolocalización del vehículo recolector del aplicativo, en donde, le permitirá visualizar el estado, distancia y tiempo estimado en tiempo real existente entre el usuario y el vehículo recolector.	Alta
RF5	Visualización Vehículo Recolector en Tiempo Real	Dentro de la pantalla de geolocalización del vehículo recolector del aplicativo, el usuario de la EMMAIPC-EP podrá visualizar el icono del vehículo recolector dentro del mapa en tiempo real cuando este se encuentre a una distancia menor a 500 m del mismo.	Alta
RF6	Notificación en Tiempo Real de Llegada del Vehículo Recolector	Dentro de la pantalla de geolocalización del vehículo recolector del aplicativo, el usuario de la EMMAIPC-EP podrá activar la función de notificaciones en segundo plano del aplicativo, permitiéndole informarle al mismo, si el vehículo recolector ya se encuentra a una distancia cercana de él, menor a 500 m, así como también, el estado en tiempo real del vehículo recolector.	Alta

- ***Requerimientos no funcionales:***

Tabla 8. Requerimientos No Funcionales. Fuente: Autoría Propia.

Id	Requerimiento	Descripción	Prioridad
RNF1	Lenguaje de Desarrollo	El prototipo del aplicativo móvil será desarrollado en lenguaje de programación Java.	Alta
RNF2	Plataforma	El prototipo del aplicativo móvil funcionara en dispositivos móviles que cuenten con sistema operativo Android.	Alta
RNF3	Interfaz de Usuario	El prototipo del aplicativo móvil tendrá una interfaz adaptable, sencilla, amigable y fácil de manejar.	Alta
RNF4	Topología de Base de Datos	El prototipo del aplicativo móvil funcionara con una base de datos relacional para el almacenamiento de información.	Alta
RNF5	Servidor de Geolocalización	Para él envió de datos de geolocalización del vehículo recolector y la obtención e interacción con dichos datos por parte del prototipo de aplicativo móvil se utilizará el servidor open source de geolocalización Traccar.	Alta
RNF6	Idioma	El prototipo del aplicativo móvil tendrá por defecto el idioma castellano.	Alta
RNF7	Disponibilidad	La información del aplicativo móvil debe estar disponible dentro de los horarios de recolección.	Alta

El alcance del proyecto tiene sus presupuestos y limitaciones de desarrollo tales como:

- Los dispositivos móviles que utilicen este prototipo de aplicativo móvil deberán contar con una versión de Android superior o igual a Oreo, para su óptimo funcionamiento y contar con el componente GPS para su funcionamiento.

- El prototipo del aplicativo móvil requiere de conexión a Internet para la obtención de los datos e información necesaria del servidor, proporcionando su visualización al usuario.

El proyecto se llevará a cabo con las herramientas seleccionadas en el marco teórico del presente trabajo además de trabajar con NetBeans IDE 8.2 y un servidor Apache.

Casos de Uso

Una vez definidos los requerimientos, se analiza la arquitectura del prototipo mediante diagramas de caso de uso, definiendo además la usabilidad de la aplicación y las acciones de ejecución de quién está a cargo de ejecutarla.

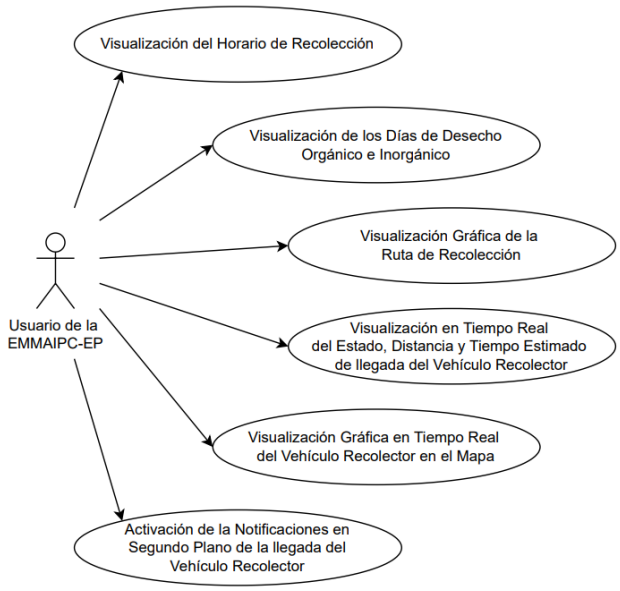


Ilustración 5. Diagrama de Caso de Uso Usuario de la EMMAIPC-EP. Fuente: Autoría Propia.

Inicialización

Esta fase se realiza las actividades de diseño y desarrollo del prototipo del aplicativo móvil, tomando en cuenta diferentes configuraciones:

Configuración del Ambiente de Desarrollo

- Configuración Prototipo de Aplicativo Móvil:

  Permisos: Ubicación, Acceso a Internet, Ejecución en Segundo Plano.

  Configuraciones: Uso de las dependencias de Retrofit, Material Desgin, GSON, Google Play Services Location API y Maps SDK for Android.

- Configuración Sistema de Información:

  Tipo de Proyecto: Sistema de Información de Escritorio para la gestión del aplicativo móvil.

Configuraciones: Uso de las librerías Retrofit, OkHttp, Java API for KML, GSON y Absolute Layout.

- Configuración API REST:

  Tipo de Proyecto: API RESTful

  Lenguaje de Codificación: PHP.

  Servidor: Apache.

  Protocolo: HTTP.

Preparación del Ambiente

Para la preparación del ambiente se ha instalado las siguientes herramientas para el desarrollo: Android Studio IDE, NetBeans IDE 8.2, Visual Studio Code, XAMPP, Servidor Traccar, Postman.

Planificación Inicial

- ***Definición de Módulos***

  Módulo Pantalla Selección de Ruta de Recorrido

  - Selección de las zonas, sub zonas y rutas de recolección.
  - Obtención y visualización de la ubicación actual del usuario.
  - Visualización de los horarios de recolección, días de desecho orgánico e inorgánico correspondiente a la ruta seleccionada.
  - Visualización gráfica de la ruta de recolección en el mapa.

  Módulo Pantalla de Obtención de la Ubicación Real del Vehículo Recolector

  - Comprobación de la ubicación real y exacta del vehículo recolector dentro de la ruta de recorrido.
  - Verificación del paso o no del vehículo recolector por la ubicación del usuario.

  Módulo Pantalla de Geolocalización del Vehículo Recolector

  - Visualización en tiempo real del estado, distancia y tiempo estimado entre el vehículo recolector y el usuario.
  - Visualización en tiempo real del vehículo recolector dentro del mapa.
  - Obtención y visualización de la ubicación actual del usuario.

Módulo de Notificaciones en Segundo Plano de la Geolocalización del Vehículo Recolector

- Notificación en segundo plano y en tiempo real del estado y distancia del vehículo recolector con respecto al usuario.

- ***Planificación de fases***

Tabla 9. Planificación de Fases. Fuente: Autoría Propia.

Fase	Iteración	Descripción
Exploración	Iteración 0	Establecimiento del proyecto, los grupos de interés, limitaciones, dependencias y supuestos.
Inicialización	Iteración 0	Análisis de los requisitos iniciales.
Producción	Iteración 1	Implementación del módulo pantalla selección de ruta de recorrido. Refinamiento de las interfaces. Ejecución de y creación de pruebas de aceptación.
	Iteración 2	Implementación del módulo pantalla de obtención de la ubicación real del vehículo recolector. Refinamiento de las interfaces. Ejecución de y creación de pruebas de aceptación.
	Iteración 3	Implementación del módulo pantalla de geolocalización del vehículo recolector. Refinamiento de las interfaces. Ejecución de y creación de pruebas de aceptación.
	Iteración 4	Implementación del módulo notificación en segundo plano de la geolocalización del vehículo recolector. Refinamiento de las interfaces. Ejecución de y creación de pruebas de aceptación.
Estabilización	Iteración 5	Refactorización del módulo pantalla selección de ruta de recorrido. Definición y establecimiento de las interfaces finales. Ejecución de pruebas de aceptación.
	Iteración 6	Refactorización del módulo pantalla de obtención de la ubicación real del vehículo recolector.

		Definición y establecimiento de las interfaces finales. Ejecución de pruebas de aceptación.
	Iteración 7	Refactorización del módulo pantalla de geolocalización del vehículo recolector. Definición y establecimiento de las interfaces finales. Ejecución de pruebas de aceptación.
	Iteración 8	Refactorización del módulo notificación en segundo plano de la geolocalización del vehículo recolector. Definición y establecimiento de las interfaces finales. Ejecución de pruebas de aceptación.
Pruebas del Sistema	Iteración 9	Realización de la evaluación de las pruebas del sistema y análisis de resultados.

Producción y Estabilización

En este punto se combinaron las dos fases para la realización del desarrollo del prototipo en base a cada una de las iteraciones anteriormente planteadas dentro de la fase de inicialización. Con el objeto de realizar las revisiones respectivas y poder asegurar el correcto funcionamiento de cada uno de los módulos.

Como primer punto se obtuvo el diseño o modelado de la base de datos que se utiliza para el almacenamiento de información. Teniendo el siguiente diagrama de base de datos en donde se presentan las tablas y los atributos para la ejecución del prototipo.

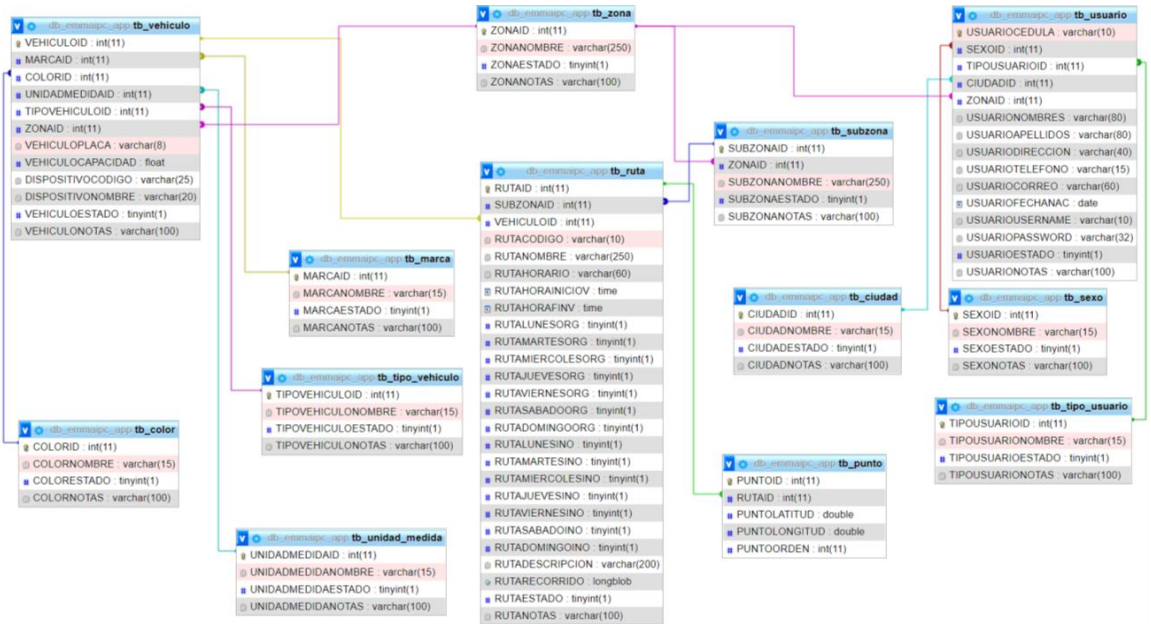


Ilustración 6. Diagrama de Base de Datos. Fuente: Autoría Propia.

De la misma manera se diseñó el diagrama de base de datos incorporado al Servidor Traccar:

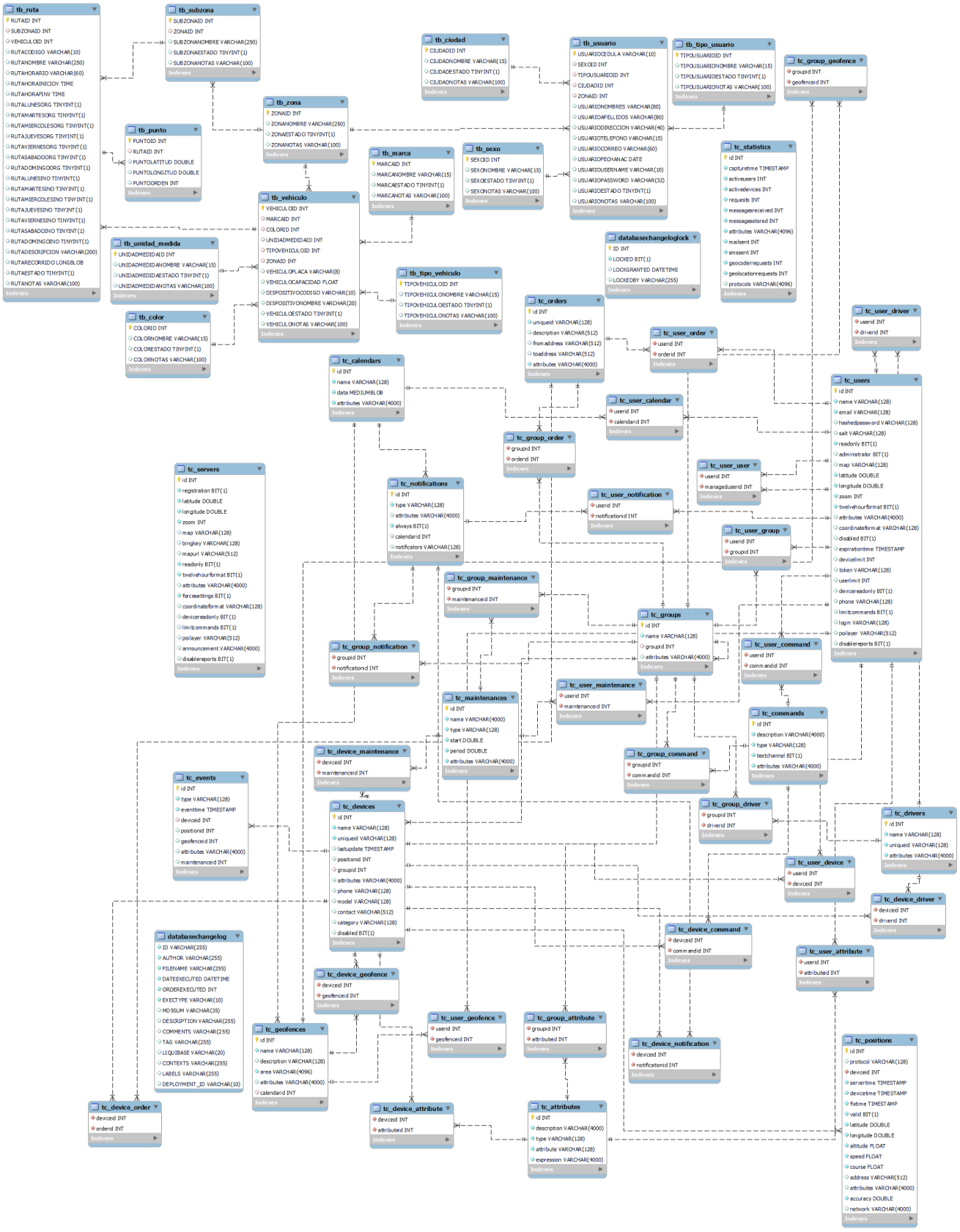


Ilustración 7. Diagrama de Base de Datos Incorporado el Servidor Traccar. Fuente: Autoría Propia.

Producción de Iteraciones

- ***Modulo Pantalla Selección de Ruta de Recorrido***

La ilustración 8 permite visualizar las zonas, sub zonas y rutas a través de un contenedor, así como cargar sus rutas dependiendo se la selección de la sub zona de forma gráfica.

Se puede también visualizar los horarios de recolección, días de desecho orgánico e inorgánico respectivo a la ruta de recolección seleccionada.

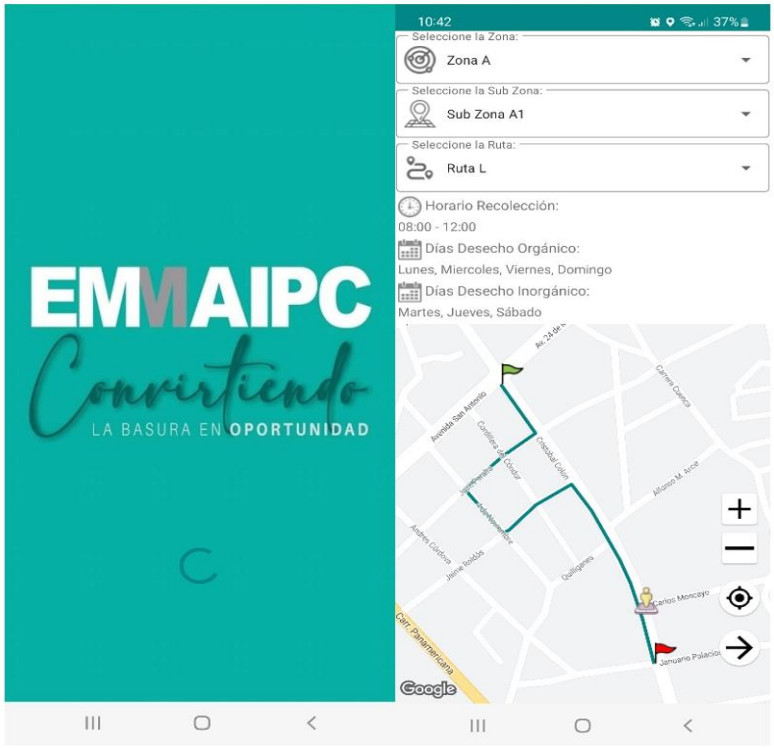


Ilustración 8. Módulo Pantalla Selección de Ruta de Recorrido. Fuente: Autoría Propio.

- ***Modulo Pantalla de Obtención de la Ubicación Real del Vehículo Recolector***

La siguiente ilustración permite visualizar una pantalla de espera para la obtención de la ubicación real y precisa del vehículo recolector.

Ilustración 9. Modulo Pantalla de Obtención de la Ubicación Real del Vehículo Recolector. Fuente: Autoría Propia

- ***Modulo Pantalla de Geolocalización del Vehículo Recolector***

El presente módulo contiene las siguientes características:

- Visualizar en tiempo real el estado, distancia y tiempo estimado entre el vehículo recolector y el usuario.
- Visualizar en tiempo real el icono del vehículo recolector dentro del mapa.

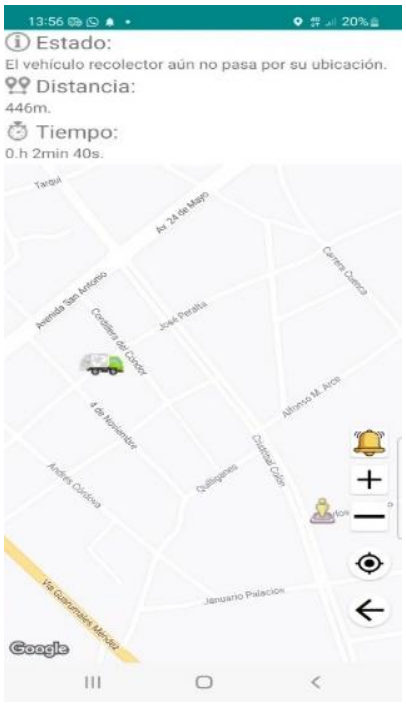


Ilustración 10. Módulo Pantalla de Geolocalización del Vehículo Recolector. Fuente: Autoría Propio.

- ***Módulo de Notificaciones en Segundo Plano de la Geolocalización del Vehículo Recolector***

Las notificaciones representadas en la siguiente tabla permiten visualizar las diferentes notificaciones que proporciona el prototipo. Notificando en segundo plano el estado en tiempo real del vehículo recolector con respecto a la ubicación

del usuario y una notificación de manera continua cuando el vehículo recolector se encuentra dentro de un rango de distancia cercano al usuario.

Tabla 10. Notificaciones del prototipo móvil. Fuente: Autoría Propia.

Notificación	Descripción
El vehículo recolector está a: 229 m, de su ubicación	Notificación que avisa la distancia del recolector de desechos al usuario
Notificaciones Activas	Indica que las notificaciones del aplicativo se encuentran activas
El vehículo recolector ya pasó por su ubicación	Notificación que indica que el vehículo recolector ya pasó por la ubicación del usuario
El horario de recolección de la ruta ha terminado	Indica al usuario que la ruta del vehículo recolector ha terminado
El vehículo recolector se encuentra inactivo	Indica que el vehículo recolector no se encuentra en actividad

Pruebas del Sistema

En esta fase se verifica que cada uno de los requerimientos funcionales descritos en la fase 1 de la presente metodología se han cumplido correctamente.

Tabla 11. Pruebas del Sistema. Fuente: Autoría Propia.

Id	Requerimiento	Cumplimiento
RF1	Visualización Horarios de Recolección	Hecho
RF2	Visualización Días de Desecho Orgánico e Inorgánico	Hecho
RF3	Visualización Gráfica Ruta de Recolección	Hecho
RF4	Visualización Estado del Vehículo Recolector en Tiempo Real	Hecho
RF5	Visualización Vehículo Recolector en Tiempo Real	Hecho
RF6	Notificación en Tiempo Real de Llegada del Vehículo Recolector	Hecho

CONCLUSIONES

En conclusión, en base al análisis de servidores se determinó que Traccar es el software óptimo para el almacenamiento, envió e interacción de los datos de geolocalización de la empresa EMMAIPC-EP, debido a que es un software de código abierto que permite ser implementado, modificado y acoplado al entrono desarrollado para el funcionamiento del prototipo. Proporciona herramientas óptimas para la gestión de los dispositivos a través de una interfaz

completa, segura y amigable. Además, permite la compatibilidad con una gran variedad de dispositivos GPS del mercado.

Con el desarrollo del prototipo de aplicativo móvil, se logró solventar una necesidad de los usuarios de la EMMAIPC-EP de identificar con facilidad los horarios de recolección, días de desecho orgánico e inorgánico, ruta de recolección, estado, distancia y tiempo estimado de llegada del vehículo recolector, así como la geolocalización y notificaciones del mismo. Permitiendo a la empresa EMMAIPC-EP realizar su proceso de recolección mucho más eficiente, gracias a esta herramienta tecnológica que se encuentra a disposición de sus usuarios.

Finalmente, se desarrolla un sistema de información de escritorio que permitirá al personal de la empresa EMMAIPC-EP poder gestionar los datos de las Zonas, Sub Zonas, Usuarios, Vehículos, Rutas y demás datos relacionados al proceso de recolección, los mismos que serán visibles por el usuario dentro del aplicativo móvil. Permitiendo que el desarrollo de este prototipo de aplicativo móvil sea flexible ante eventos en donde se requiera modificar o realizar cambios en la información de recolección, la misma, que es visible por el usuario a través del prototipo de aplicativo móvil.

BIBLIOGRAFÍA

Android. (17 de 05 de 2021). *developer android*. Obtenido de developer android: https://developer.android.com/studio/intro

APOLONIA, M. V. (01 de 11 de 2021). *dspace.utb.edu.ec.* Obtenido de dspace.utb.edu.ec: http://dspace.utb.edu.ec/bitstream/handle/49000/10535/E-UTB-FAFI-SIST-000244.pdf?sequence=1&isAllowed=y

Arenaza, R. E. (24 de 01 de 2019). *repositorio.une.edu.pe.* Obtenido de repositorio.une.edu.pe: https://repositorio.une.edu.pe/bitstream/handle/20.500.14039/3026/MONOGRAF%c3%8dA%20-%20ROMAN%20ARENAZA.pdf?sequence=1&isAllowed=y

AWS. (01 de 01 de 2022). *aws.amazon.com*. Obtenido de aws.amazon.com: https://aws.amazon.com/es/rds/mysql/

CABRERA, R. M. (01 de 01 de 2018). *repositori.tecnocampus.cat.* Obtenido de repositori.tecnocampus.cat/: https://repositori.tecnocampus.cat/bitstream/handle/20.500.12367/99/Memoria.pdf?sequence=1&isAllowed=y

Combaudon, S. (2018). *MySQL 5.7 Administración y optimización.* España: Ediciones ENI.

Estrada, C. M., Villatoro, K. C., García, V. B., & Vázquez, A. d. (2022). Diseño de un Sistema web para el control de Curriculum Vitae Electrónico de personal docente basado en una arquitectura orientada a servicios (API REST). *RITI*, 28-42.

Fossati, M. (2018). *Introducción a PHP y HTML.* Matias Fossati.

Gabriel, E. J., & Isabel, C. S. (2021). USABILIDAD EN APLICACIONES MÓVILES. *ICT-UNPA*, 25-47.

Gallego, M. T. (05 de 07 de 2022). *openaccess.uoc.edu.* Obtenido de openaccess.uoc.edu: http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf

Garrido, A. P., López, Y. B., & Constante, G. G. (2020). Rendimiento de MariaDB y PostgreSQL. *Revista Científica y Tecnológica UPSE*, 9-16.

KINSTA. (25 de 04 de 2022). *kinsta.com*. Obtenido de kinsta.com: https://kinsta.com/es/base-de-conocimiento/que-es-postgresql/

Logo. (30 de 09 de 2019). *es.logodownload.org*. Obtenido de es.logodownload.org: https://es.logodownload.org/android-logo/

Mainez, Á. P. (01 de 06 de 2018). *oa.upm.es.* Obtenido de oa.upm.es: https://oa.upm.es/52431/1/TFG_ANGELA_PORRAS_MAINEZ.pdf

Mariana Berga, R. F. (10 de 06 de 2021). *www.imaginarycloud.com*. Obtenido de www.imaginarycloud.com: https://www.imaginarycloud.com/blog/kotlin-vs-java/

Mejía, G. J., Corea, D. M., & Frank, M. T. (01 de 02 de 2019). *riul.unanleon.edu.ni:8080.* Obtenido de riul.unanleon.edu.ni:8080: http://riul.unanleon.edu.ni:8080/jspui/bitstream/123456789/7088/1/242054.pdf

Montero, R. (2022). *ANDROID: Desarrollo de aplicacioes.* Bogotá: Ediciones de la U.

Morcillo, L. M. (01 de 06 de 2019). *oa.upm.es.* Obtenido de oa.upm.es: https://oa.upm.es/55698/1/TFG_LUCIA_MONDEJAR_MORCILLO.pdf

OpenGTS. (20 de 02 de 2020). *www.opengts.org*. Obtenido de www.opengts.org: http://www.opengts.org/?fbclid=IwAR0Rj-ngd1gVsrf3M6E9-uZiNzbwAw2H1jgHRob9BO7zts2IYR0xQ9M4V5c

Plainer, M. (01 de 12 de 2020). *www21.in.tum.de.* Obtenido de www21.in.tum.de: https://www21.in.tum.de/teaching/osp/WS20/assets/pr-plainer-vscode.pdf

Santiago, E. B., & Isarael, C. G. (2022). DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN MÓVIL QUE CUMPLA LA FUNCIÓN DE ESTACIÓN EN TIERRA PARA EL MONITOREO DE UAV'S EN EL CENTRO DE INVESTIGACIÓN Y DESARROLLO DE LA FUERZA AÉREA ECUATORIANA. *Departamento de Eléctrica y Electrónica,* 174-175.

Solecki , I., Porto, J., da Cruz Alves, N., Gresse von Wangenheim, C., Hauck, J., & Ferreti Borgatto, A. (2020). Automated Assessment of the Visual Design of Android Apps Developed with App Inventor. *Paper Session: Automated Systems,* 51-57. Obtenido de dl.acm.org: https://dl.acm.org/doi/pdf/10.1145/3328778.3366868?casa_token=keDuh94tCmYAAAAA:p-NI9mEoSoi3oNqkh4jO_K0v5WsIzw6yGuRiDNGYFq1l6oBDe7EYC-uk1E8Y9MUZ6M5mmYxQb96Zhg

TANDAZO, H. S. (01 de 01 de 2022). *dspace.utb.edu.ec.* Obtenido de dspace.utb.edu.ec: http://dspace.utb.edu.ec/bitstream/handle/49000/11689/E-UTB-FAFI-SIST-000343.pdf?sequence=1&isAllowed=y

Traccar. (28 de 05 de 2022). *www.traccar.org*. Obtenido de www.traccar.org: https://www.traccar.org/

Villalba, C. M., Moraleda, A. U., & González, M. Á. (2021). *Lenguajes de Programación.* UNED.

ANEXOS

2022

Propuesta Tecnológica

PROTOTIPO DE UN APLICATIVO MÓVIL DE GEOLOCALIZACIÓN DEL RECOLECTOR DE DESECHOS EN BASE A SU RECORRIDO PARA LOS USUARIOS DE LA EMMAIPC-EP

BRYAN S.

UNIVERSIDAD CATÓLICA DE CUENCA EXTENSIÓN CAÑAR

Anexo 1: Manual del Usuario

Manejo del Prototipo de Aplicativo Móvil EMMAIPC-EP

El prototipo en caso de requerir su implementación a futuro debe ser subido a la tienda de aplicaciones Play Store para que los usuarios de la EMMAIPC-EP puedan descargarlo e instalarlo en sus dispositivos móviles.

Visualización Inicial

Como primer punto se debe de tomar en cuenta que este aplicativo móvil tiene funcionalidad en dispositivos con sistema operativo Android.

El aplicativo una vez instalado en el dispositivo móvil, se lo puede identificar con el siguiente icono dentro del menú de aplicaciones.

Ilustración 11. *Icono Aplicativo. Fuente: Autor.*

Al abrir el aplicativo se muestra una pantalla de carga de 2 segundos aproximadamente con la imagen propia de la empresa EMMAIPC-EP.

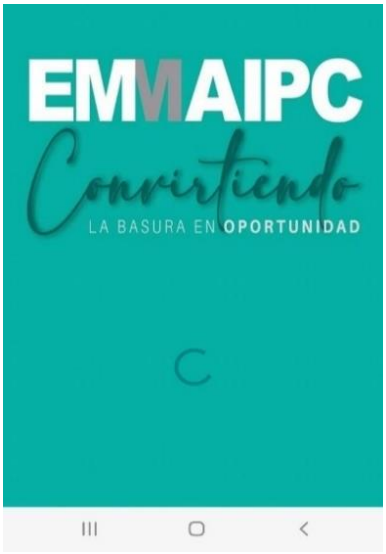


Ilustración 12. Fullscreen Inicial Aplicativo. Fuente: Autor.

Pantalla Selección de Ruta

Al acceder por primera vez al aplicativo móvil, el mismo solicita el acceso a la ubicación del dispositivo. Este permiso es de suma importancia ya que permite identificar la ubicación del usuario y cumplir el aplicativo con su propósito.

Ilustración 13. Solicitud de Permiso de Ubicación del Dispositivo. Fuente: Autor.

Luego de aceptar el permiso anteriormente mencionado, se presenta una siguiente solicitud de permiso, la misma que tiene como objeto permitir que el aplicativo móvil EMMAIPC-EP se ejecute en segundo plano, solo al momento de activar la funcionalidad de notificaciones en la Pantalla de Geolocalización del Vehículo Recolector. Es un permiso sumamente importante para el óptimo funcionamiento del apartado de las notificaciones del aplicativo.

Ilustración 14. Solicitud de Permiso de Ejecución en Segundo Plano Aplicativo. Fuente: Autor.

Una vez, proporcionado todos los permisos al aplicativo, el mismo solicita que se active la localización o ubicación del teléfono.

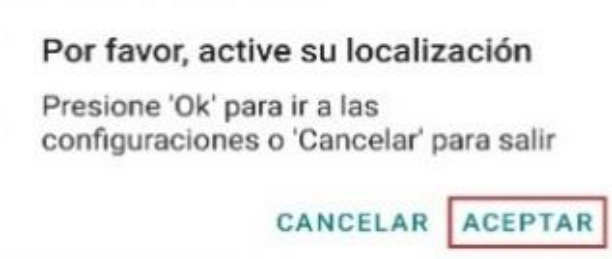


Ilustración 15. Solicitud de Activación de la Ubicación del Dispositivo. Fuente: Autor.

Al proporcionar el permiso de ubicación y activar la misma se observa la pantalla principal, la cual cuenta con una interfaz amigable. El aplicativo identifica automáticamente la ubicación del usuario en tiempo real. En esta pantalla se debe seleccionar la Zona, Subzona y Ruta correspondiente del usuario, para poder darle al botón de siguiente.

Ilustración 16. Pantalla Principal del Aplicativo. Fuente: Autor.

Para seleccionar la ruta, el usuario debe identificar a través de la visualización grafica de la misma y ver si es que su ubicación actual está dentro de ella.

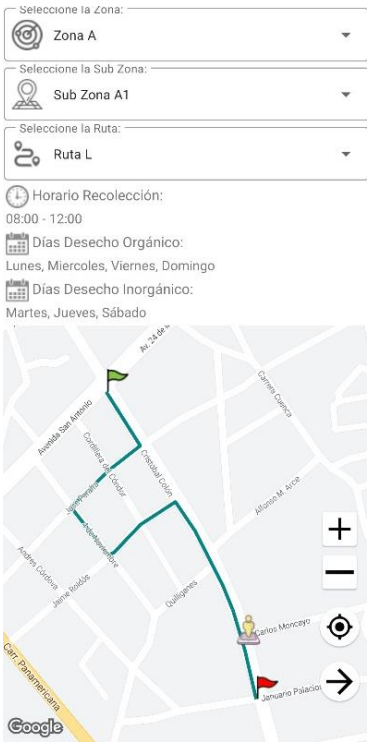


Ilustración 17. Selección de la Ruta de Recorrido. Fuente: Autor.

Una vez, identificada la ruta se presiona el siguiente botón representado con una flecha en la parte derecha inferior para continuar a la pantalla de geolocalización del vehículo recolector.

Ilustración 18. Visualización del Botón Siguiente en la Pantalla Principal. Fuente: Autor.

Pantalla de Geolocalización del Vehículo Recolector

Para el acceso a la pantalla de geolocalización del vehículo recolector primero se visualiza la siguiente pantalla, la cual, tiene la funcionalidad de una pantalla de espera de 20 segundos aproximadamente, para la obtención correcta de la ubicación y estado del vehículo recolector.

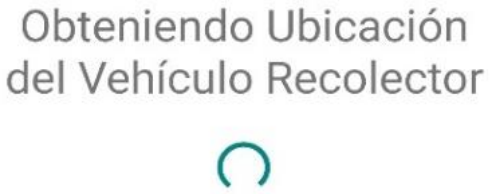


Ilustración 19. Pantalla de espera para la Obtención de la Ubicación del Vehículo Recolector. Fuente: Autor.

Una vez, que se identifica la ubicación y estado del vehículo recolector, el aplicativo permite el acceso a la pantalla de geolocalización, donde se puede observar el estado, la distancia y el tiempo del vehículo recolector en tiempo real.

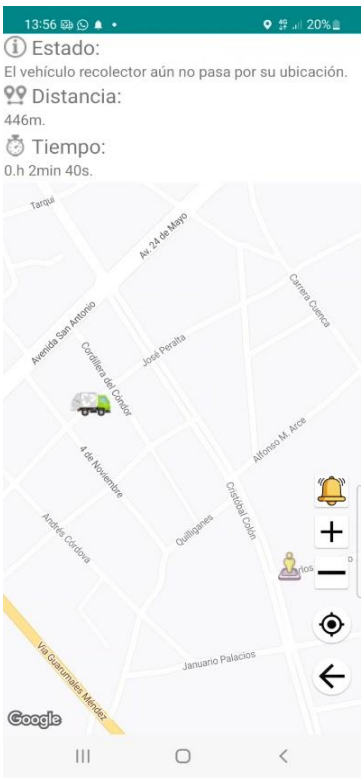


Ilustración 20. Pantalla de Geolocalización del Vehículo Recolector. Fuente: Autor.

- **Estado:** Permite visualizar el estado del vehículo recolector y otros, así como:
  - Si el vehículo recolector se encuentra activo o no.
  - Si el vehículo recolector ya paso o aun no pasa por la ubicación actual del usuario.
  - Si el usuario se encuentra dentro de la ruta seleccionada.
  - Si el usuario se encuentra dentro del horario de recolección correspondiente a la ruta seleccionada.
- **Distancia:** Permite visualizar la distancia en tiempo real entre el vehículo recolector y la ubicación actual del usuario en base a la ruta. La distancia es medida en metros.

- **Tiempo:** Permite visualizar el tiempo estimado de llegada del vehículo recolector a la ubicación actual del usuario. El tiempo se encuentra en horas, minutos y segundos.

A través del mapa se visualiza la ubicación en tiempo real del vehículo recolector, el mismo que se encuentra representado con el siguiente icono:

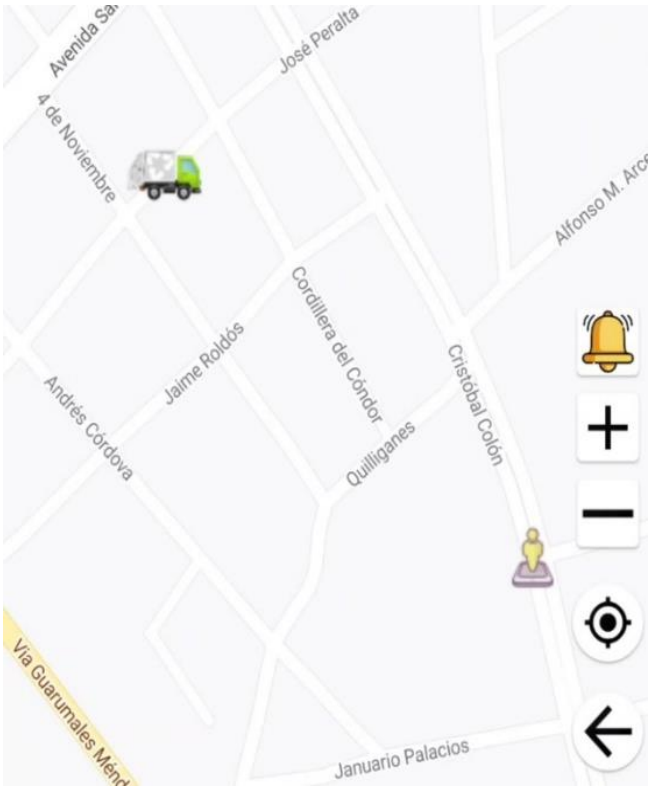


Ilustración 21. Visualización del Icono del Vehículo Recolector dentro del Mapa. Fuente: Autor.

La visualización del vehículo recolector, solo se encuentra activa cuando el vehículo se localiza a una distancia cercana del usuario, es decir, una distancia menor a 500 m, así como también, cuando el usuario que utiliza la pantalla de geolocalización se encuentra dentro del horario de recolección asignado a la ruta seleccionada.

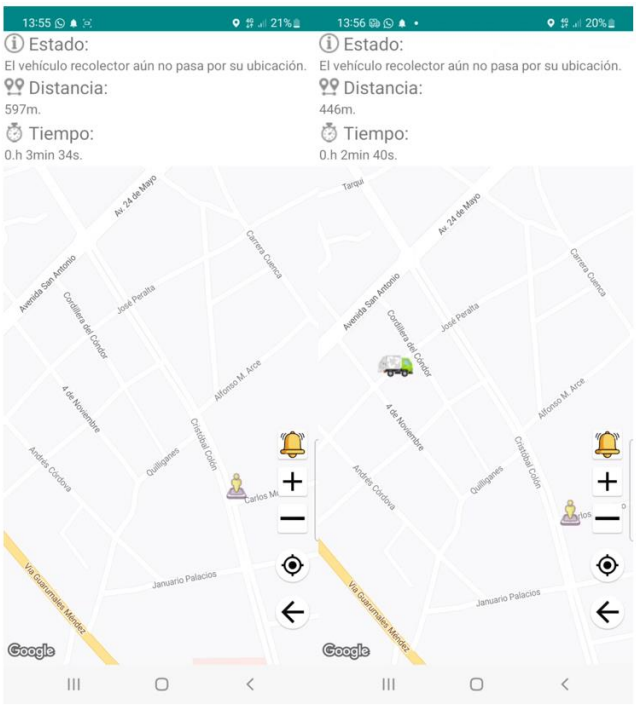


Ilustración 22. Demostración de visualización del Vehículo Recolector a una Distancia menor a 500 m. Fuente: Autor.

Para activar la funcionalidad de notificación se debe presionar el ícono de campana ubicado en la parte inferior derecha de la ilustración 28.

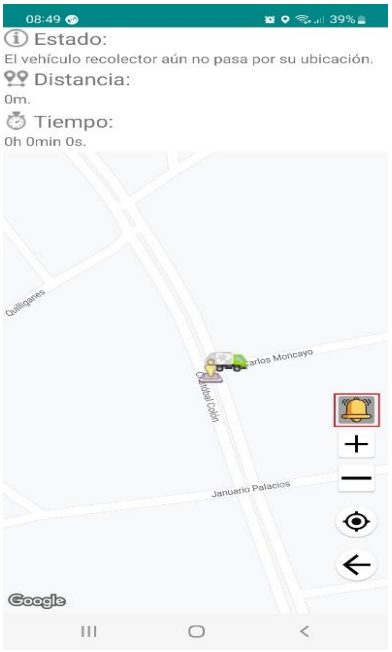


Ilustración 23. Visualización del Icono de Notificaciones. Fuente: Autor.

Al activar esta funcionalidad, el aplicativo permite notificar al usuario si el vehículo recolector se encuentra a una distancia cercana al mismo, visualizando la notificación de la distancia en metros.

Ilustración 24. Notificación de la Distancia a la que se encuentra el Vehículo Recolector con respecto al Usuario. Fuente: Autor.

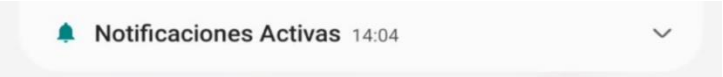


Ilustración 25. Notificación que indica que se encuentra activa la funcionalidad de las Notificaciones del Aplicativo. Fuente: Autor.

Se puede cerrar el aplicativo completamente y las notificaciones siguen activas en segundo plano, hasta que el usuario decida desactivarlas o se desactivan automáticamente en caso de que el vehículo recolector ya haya pasado por la ubicación actual del usuario, mostrando el siguiente mensaje de notificación.

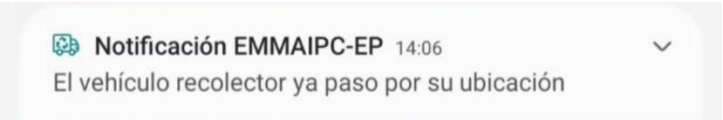


Ilustración 26. Notificación de que el Vehículo Recolector ya pasó por la Ubicación del Usuario. Fuente: Autor.

El aplicativo también muestra otras notificaciones con respecto al estado del vehículo recolector durante la activación de las notificaciones en segundo plano.

Ilustración 27. Notificación de que el horario de recolección ha terminado. Fuente: Autor.

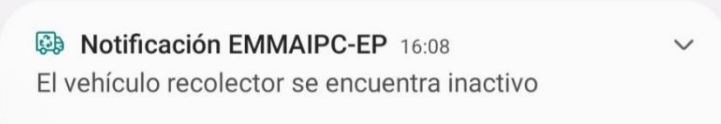


Ilustración 28. Notificación de que el Vehículo Recolector se encuentra inactivo. Fuente: Autor.

Modo Oscuro

El aplicativo cuenta con la funcionalidad de modo oscuro en cada una de las pantallas, para una mejor visualización en horarios nocturnos. El modo oscuro cambia automáticamente al modo claro y viceversa en base a la configuración del dispositivo móvil de este apartado.

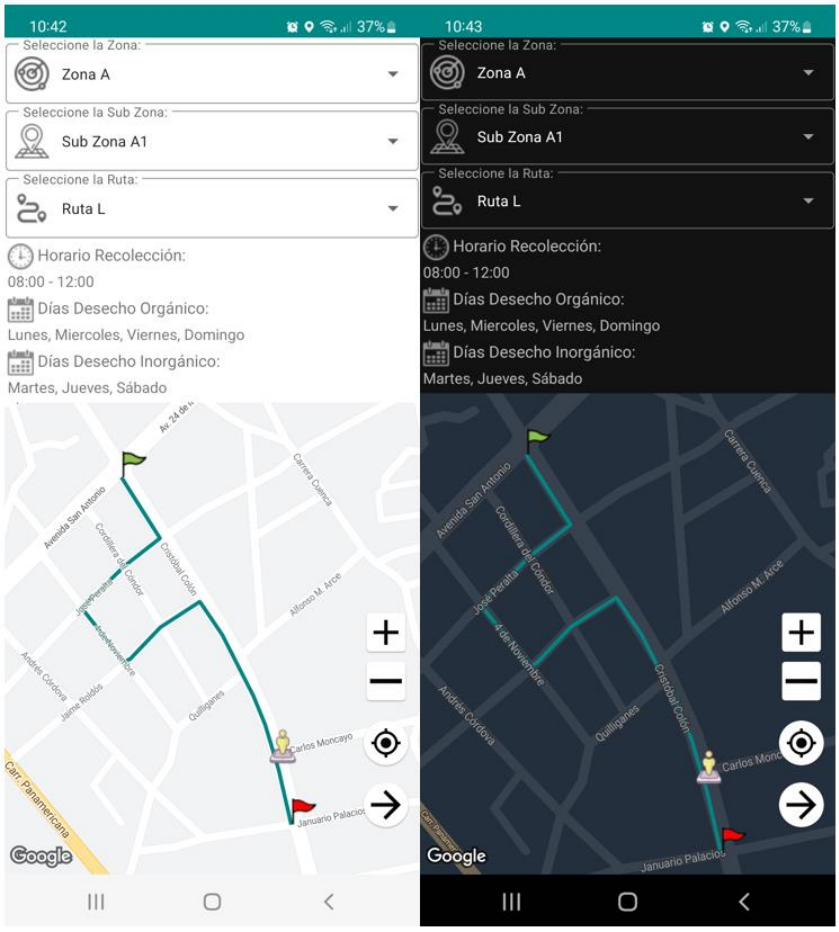


Ilustración 29. Modo Oscuro Pantalla Selección de Ruta. Fuente: Autor.

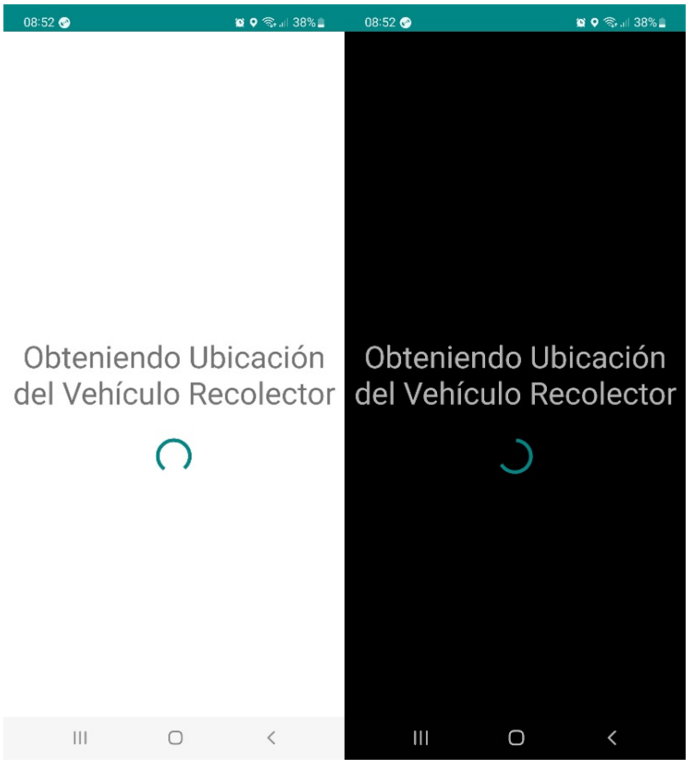


Ilustración 30. Modo Oscuro Pantalla Obtención Ubicación Vehículo Recolector. Fuente: Autor.

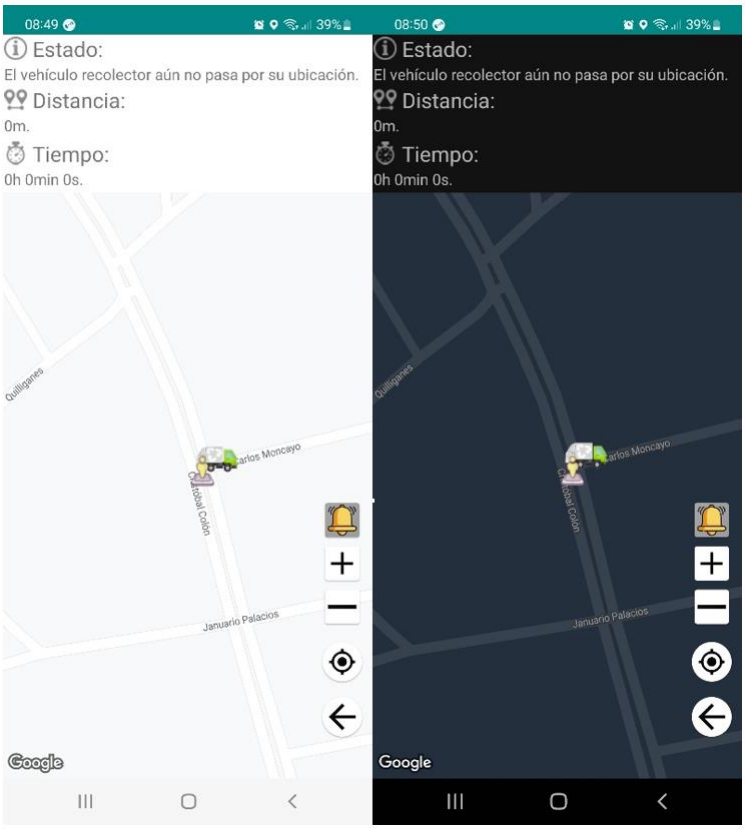


Ilustración 31. Modo Oscuro Pantalla de Geolocalización del Vehículo Recolector. Fuente: Autor.

Manejo del Sistema de Información de Escritorio para la Gestión del Aplicativo

Inicio de Sesión.

El sistema cuenta con un inicio de sesión para mayor seguridad al momento de manipular la información y registros que son gestionados por el sistema informático. Para ello se debe ingresar el usuario y contraseña correspondiente.

Ilustración 32. Pantalla de Inicio de Sesión del Sistema de Información. Fuente: Autor.

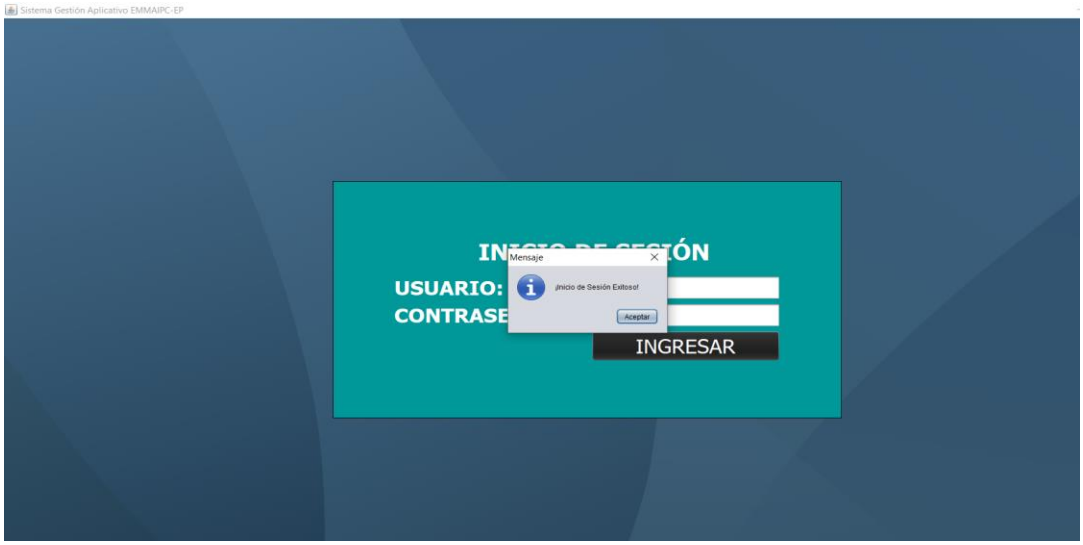


Ilustración 33. Inicio de Sesión Exitoso del Sistema de Información. Fuente: Autor.

Pantalla Principal

Una vez que se tiene un inicio de sesión exitoso, el sistema da acceso a la pantalla principal, en la cual se aprecia todos los accesos a las pantallas principales y secundarias, que permiten gestionar los registros e información de cada entidad.

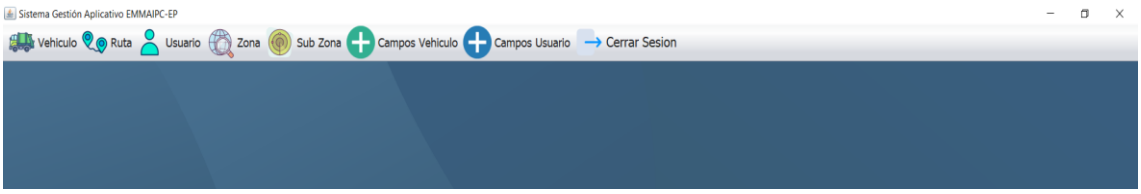


Ilustración 34. Pantalla Principal del Sistema de Información. Fuente: Autor.

Campos Usuario

En este apartado, se puede crear, eliminar, actualizar, buscar y visualizar los registros de Ciudad, Tipo Usuario y Sexo.

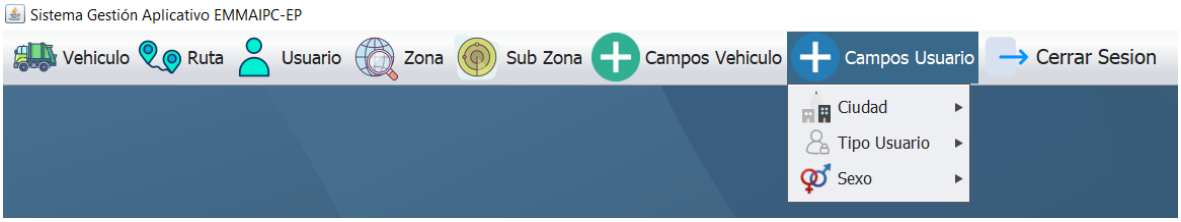


Ilustración 35. Ítems del Menú Campos de Usuario. Fuente: Autor.

Pantalla Ciudad

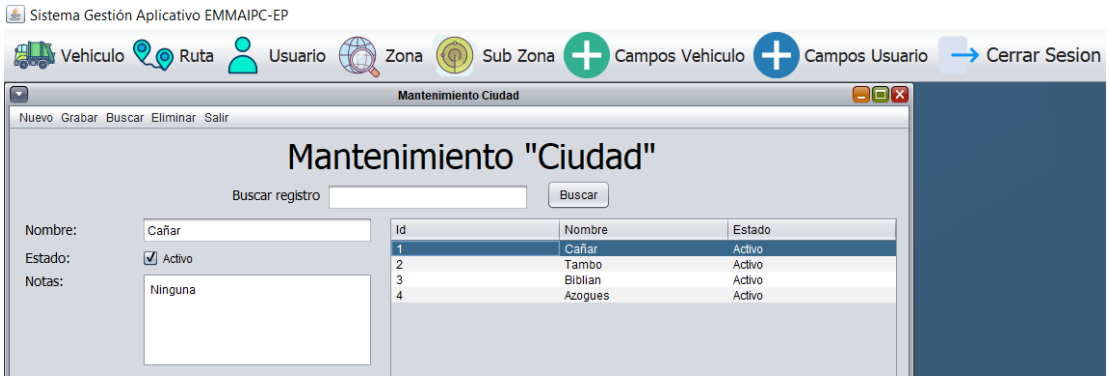


Ilustración 36. Pantalla de Gestión Ciudad. Fuente: Autor.

Pantalla Tipo Usuario

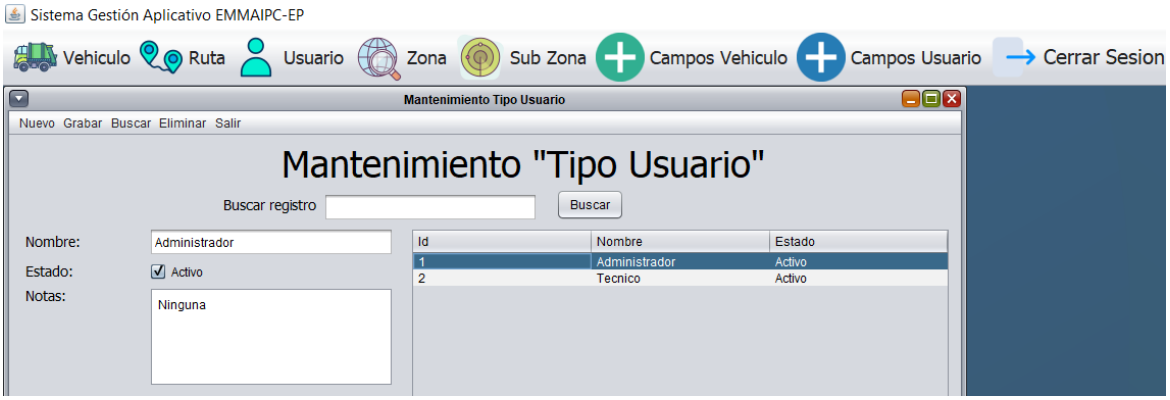


Ilustración 37. Pantalla de Gestión Tipo Usuario. Fuente: Autor.

Pantalla Sexo

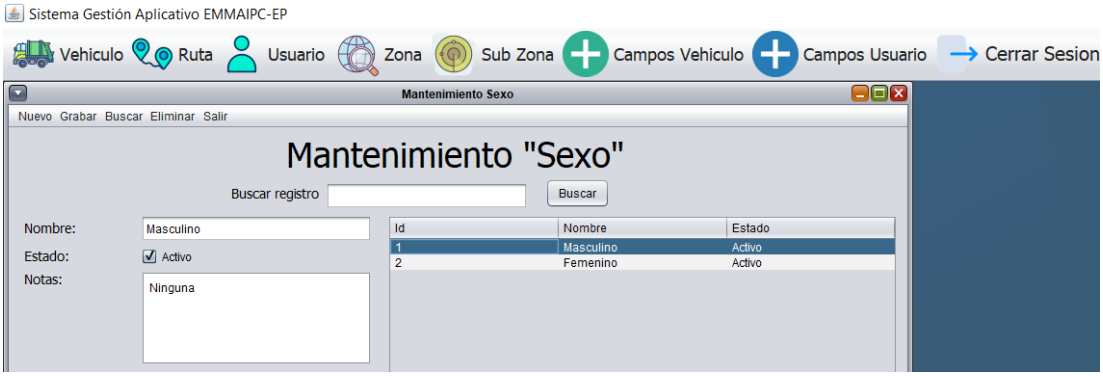


Ilustración 38. Pantalla de Gestión Sexo. Fuente: Autor.

Campos Vehículo

En este apartado, se puede crear, eliminar, actualizar, buscar y visualizar los registros de Marca, Color, Tipo Vehículo y Unidad Medida.

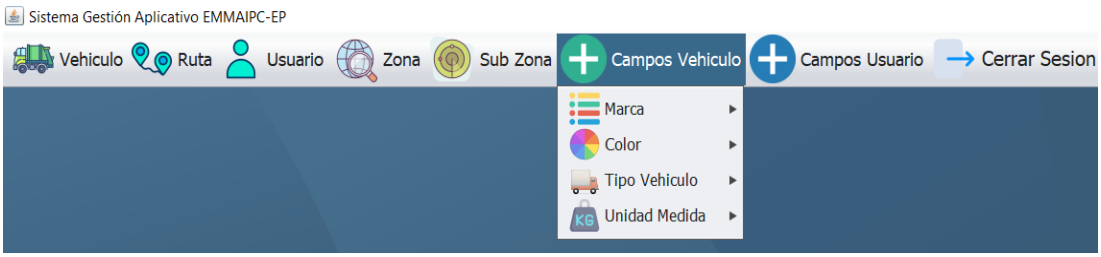


Ilustración 39. Ítems del Menú Campos Vehículo. Fuente: Autor.

Pantalla Marca

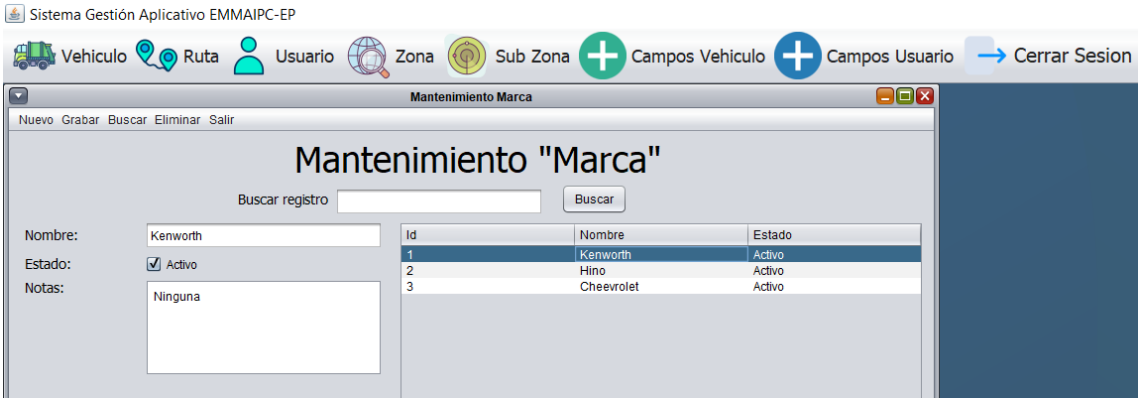


Ilustración 40. Pantalla de Gestión Marca. Fuente: Autor.

Pantalla Color

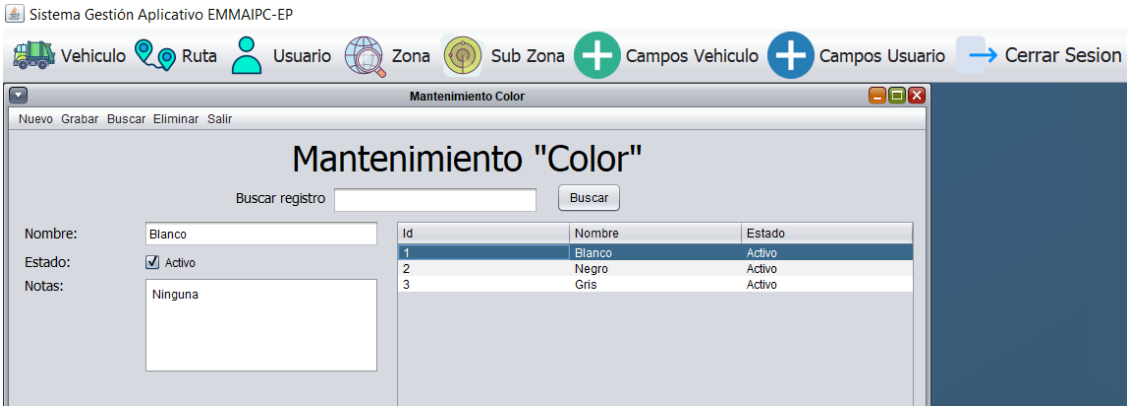


Ilustración 41. Pantalla de Gestión Color. Fuente: Autor.

Pantalla Tipo Vehículo

Ilustración 42. Pantalla de Gestión Tipo Vehículo. Fuente: Autor.

Pantalla Unidad Medida

Ilustración 43. Pantalla de Gestión Unidad Medida. Fuente: Autor.

Pantalla Sub Zona

En este apartado, se puede crear, actualizar, eliminar, buscar y visualizar los registros de las sub zonas que se encuentran relacionadas con las zonas que son manejadas por la empresa EMMAIPC-EP.

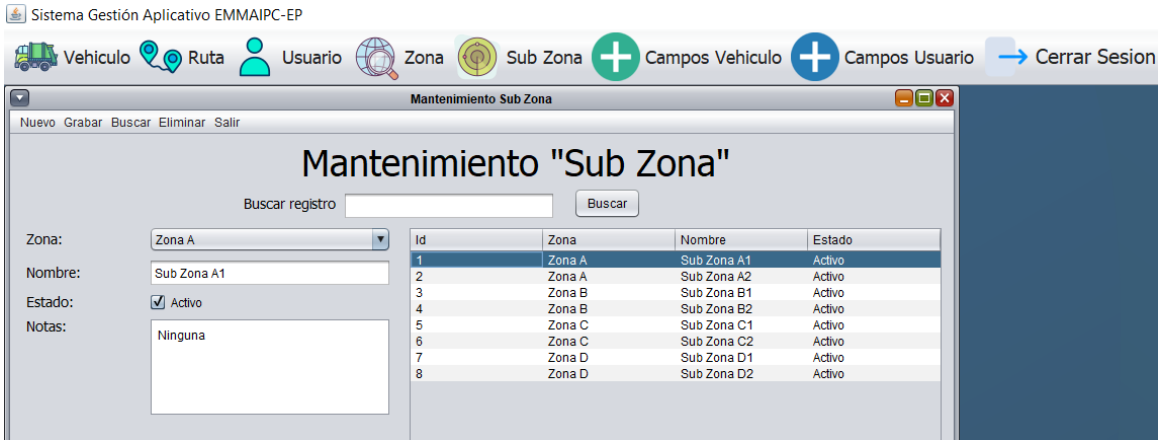


Ilustración 44. Pantalla de Gestión Sub Zona. Fuente: Autor.

Pantalla Zona

En este apartado, permite crear, actualizar, eliminar, buscar y visualizar los registros de las zonas que son manejadas por la empresa EMMAIPC-EP.

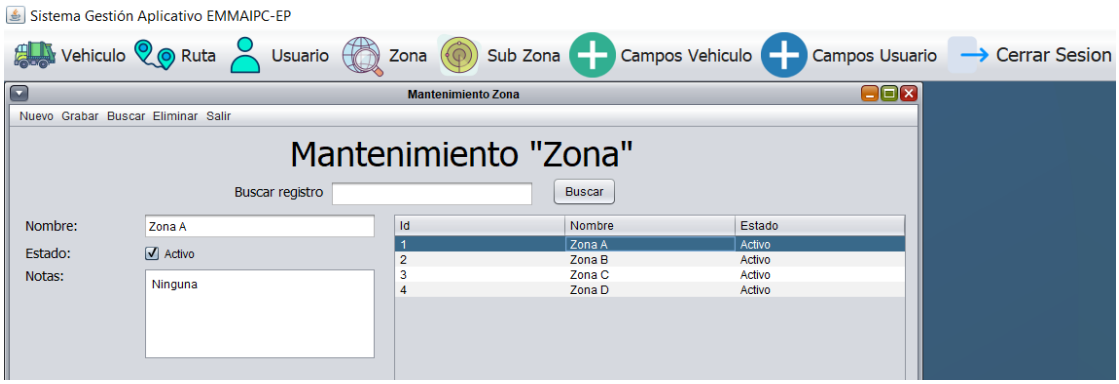


Ilustración 45. Pantalla de Gestión Zona. Fuente: Autor.

Pantalla Usuario

En este apartado, se puede crear, actualizar, eliminar, buscar y visualizar los usuarios que están inmersos en el manejo del sistema, en donde, se registra los datos personales de los mismos, así como también los respectivos datos para el manejo del sistema como el username, password y los privilegios basados en el tipo de usuario.

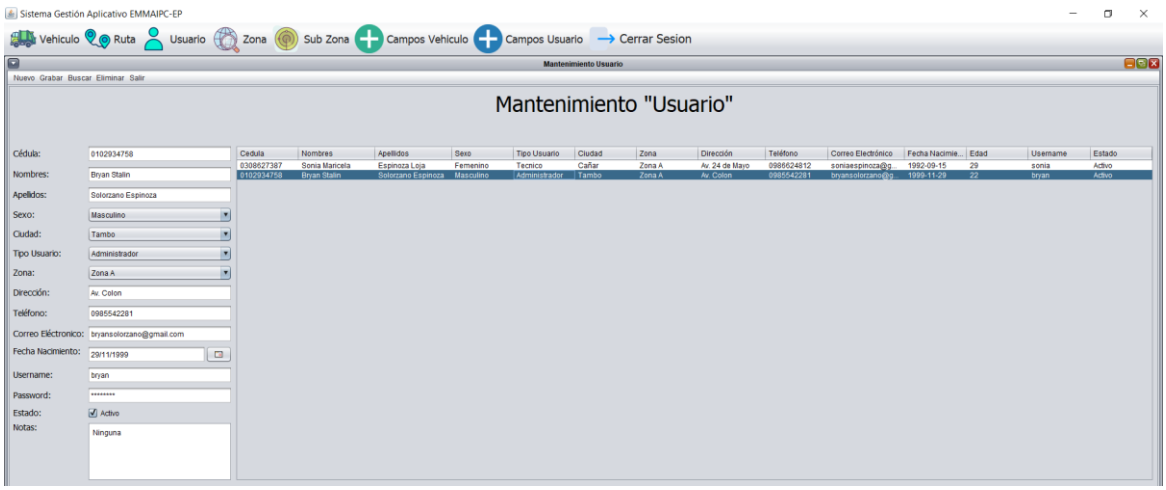


Ilustración 46. Pantalla de Gestión Usuario. Fuente: Autor.

Pantalla Ruta

En este apartado, permite crear, actualizar, eliminar, buscar y visualizar los registros de las rutas de recolección que son visualizadas en el aplicativo móvil. En esta pantalla se registra información de la ruta como el código, nombre, zona, subzona, vehículo recolector, horarios

de recolección, horarios de visualización del vehículo recolector en el aplicativo (Para mayor seguridad, la información que se ingrese en este campo deberá tener relación con la información ingresada en el horario de recolección asignado a la ruta), una descripción, la carga del archivo en formato KML (Hace referencia al conjunto de puntos que conforman la ruta de recorrido, generada o creada a partir de otro software, se recomienda el uso de Google My Maps para óptimo funcionamiento), registro de los días correspondientes a cada uno de los tipos de desechos, orgánico e inorgánico de la ruta.

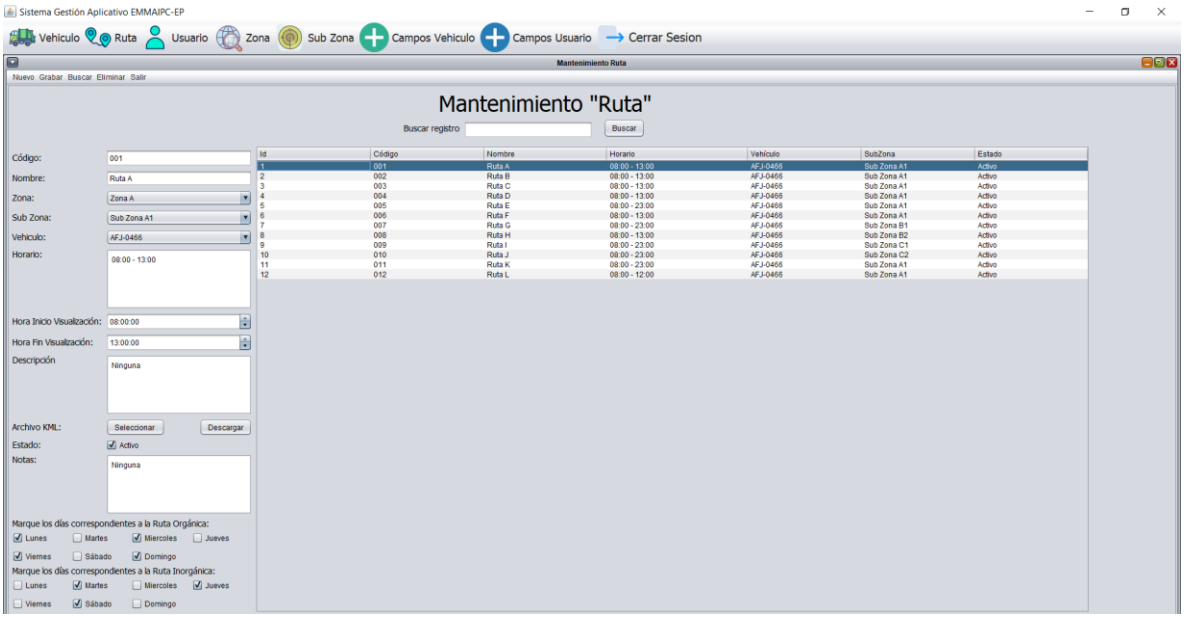


Ilustración 47. Pantalla de Gestión Ruta. Fuente: Autor.

Pantalla Vehículo

En este apartado, se puede crear, actualizar, eliminar, buscar y visualizar los registros de los vehículos recolectores. Registrando información como la placa, marca, tipo vehículo, color, capacidad, unidad medida, zona, código dispositivo (Hace referencia al código único del dispositivo GPS, el mismo que permite identificar dicho dispositivo para la geolocalización), el nombre del dispositivo (Hace referencia al nombre técnico del dispositivo GPS).

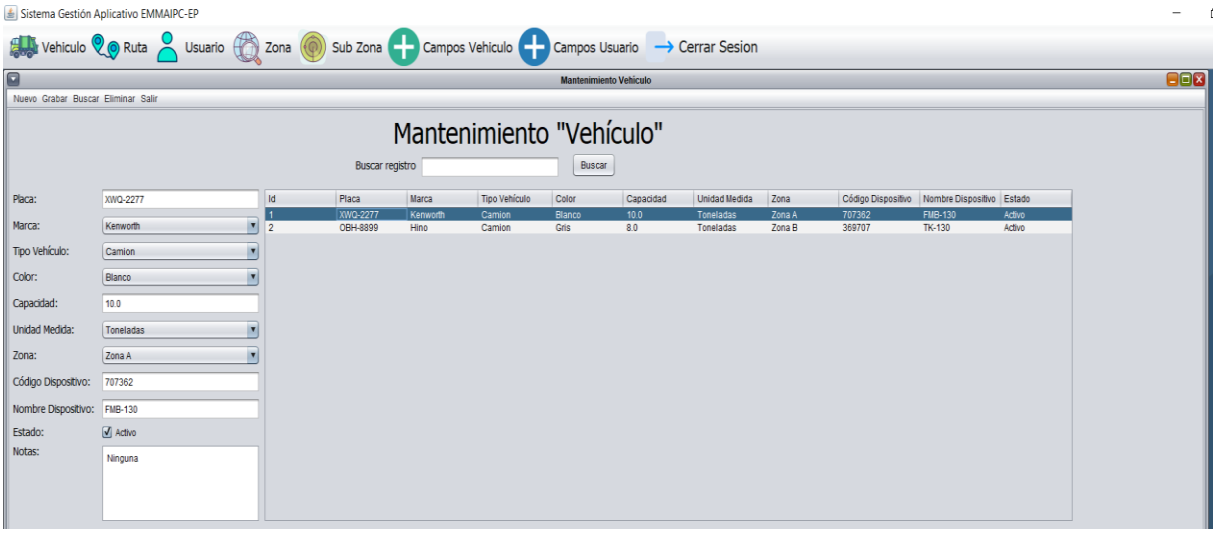


Ilustración 48. Pantalla de Gestión Vehículo. Fuente: Autor.

Esta pantalla y las acciones de crear, actualizar y eliminar, se encuentran sincronizadas con la plataforma Traccar, es decir, cualquiera de las acciones anteriormente mencionadas al ser ejecutadas, también generaran el mismo resultado o efecto dentro de dicha plataforma.

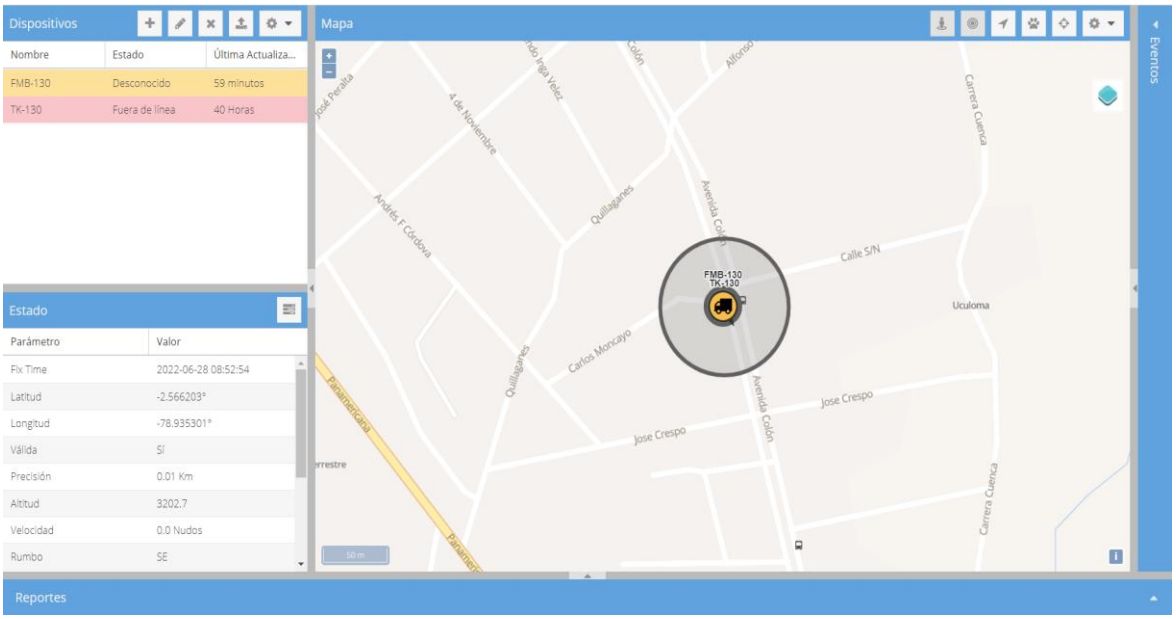


Ilustración 49. Interfaz de la Plataforma Traccar sincronizada con el Sistema de Información. Fuente: Autor.

2022

MANUAL DEL PROGRAMADOR

PROTOTIPO DE UN APLICATIVO MÓVIL DE GEOLOCALIZACIÓN DEL RECOLECTOR DE DESECHOS EN BASE A SU RECORRIDO PARA LOS USUARIOS DE LA EMMAIPC-EP

BRYAN S.

UNIVERSIDAD CATÓLICA DE CUENCA EXTENSIÓN CAÑAR

Anexo 2: Manual del Programador

El manual del programador tiene la finalidad de dar a conocer como está estructurado y desarrollado el prototipo, identificando que herramientas se utilizan para el desarrollo del mismo. Todo esto a través de información técnica, la misma que se encuentra escrita de una manera adecuada para que cualquier persona que tenga el conocimiento suficiente sobre este apartado pueda entender e interpretar lo descrito.

Diseño e Implementación de la Base de Datos

Diseño e Implementación de la Base de Datos de la EMMAIPC-EP

El modelo conceptual, lógico y físico de la base de datos están desarrollados en Power Designer, para posteriormente ser ejecutada en un motor de base de datos MySQL, utilizando la plataforma XAMPP junto con la herramienta phpMyAdmin para la administración de la base de datos.

Modelo Conceptual de Datos

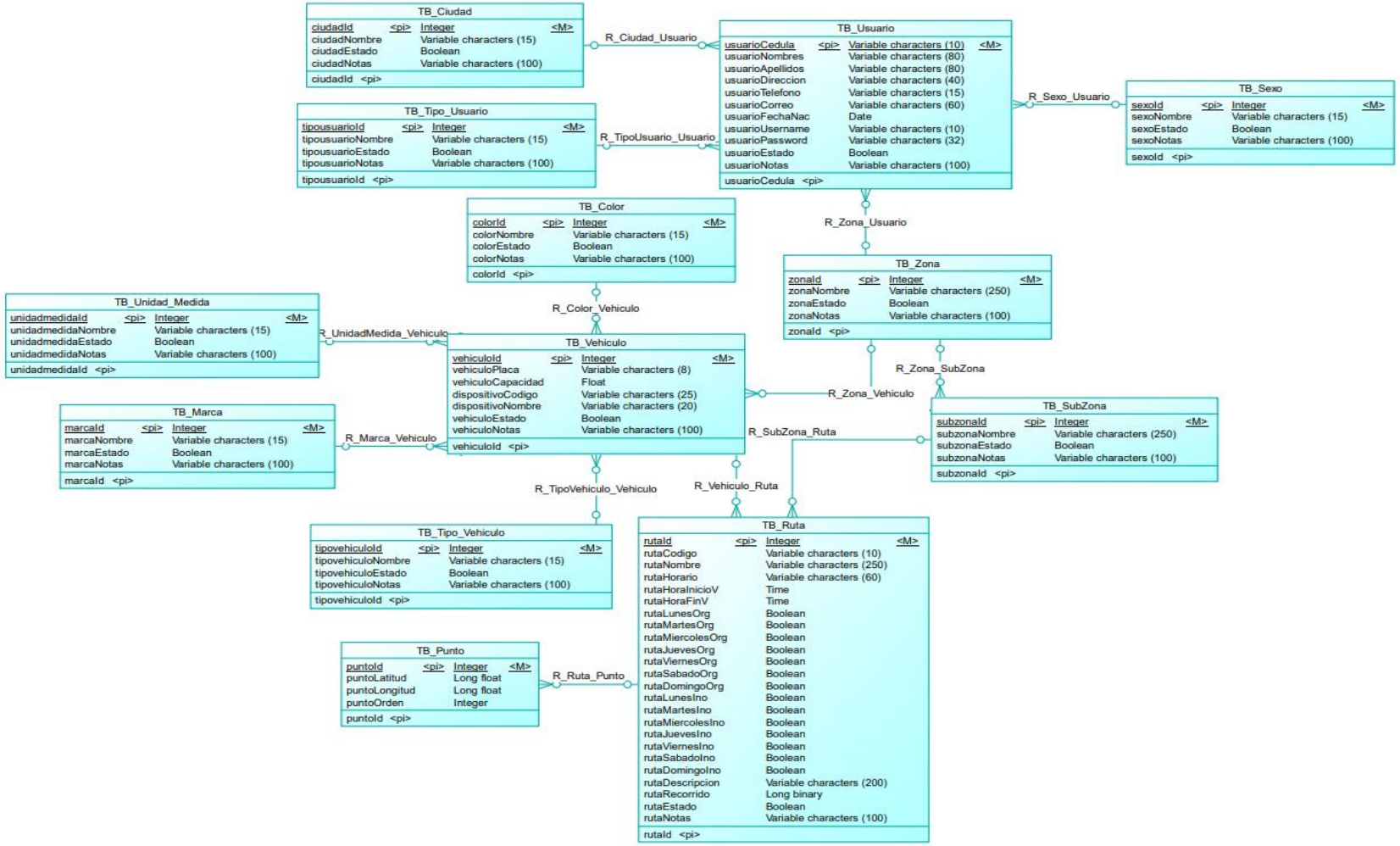


Ilustración 50. Modelo Conceptual de Datos en Power Designer. Fuente: Autor.

Modelo Lógico de Datos

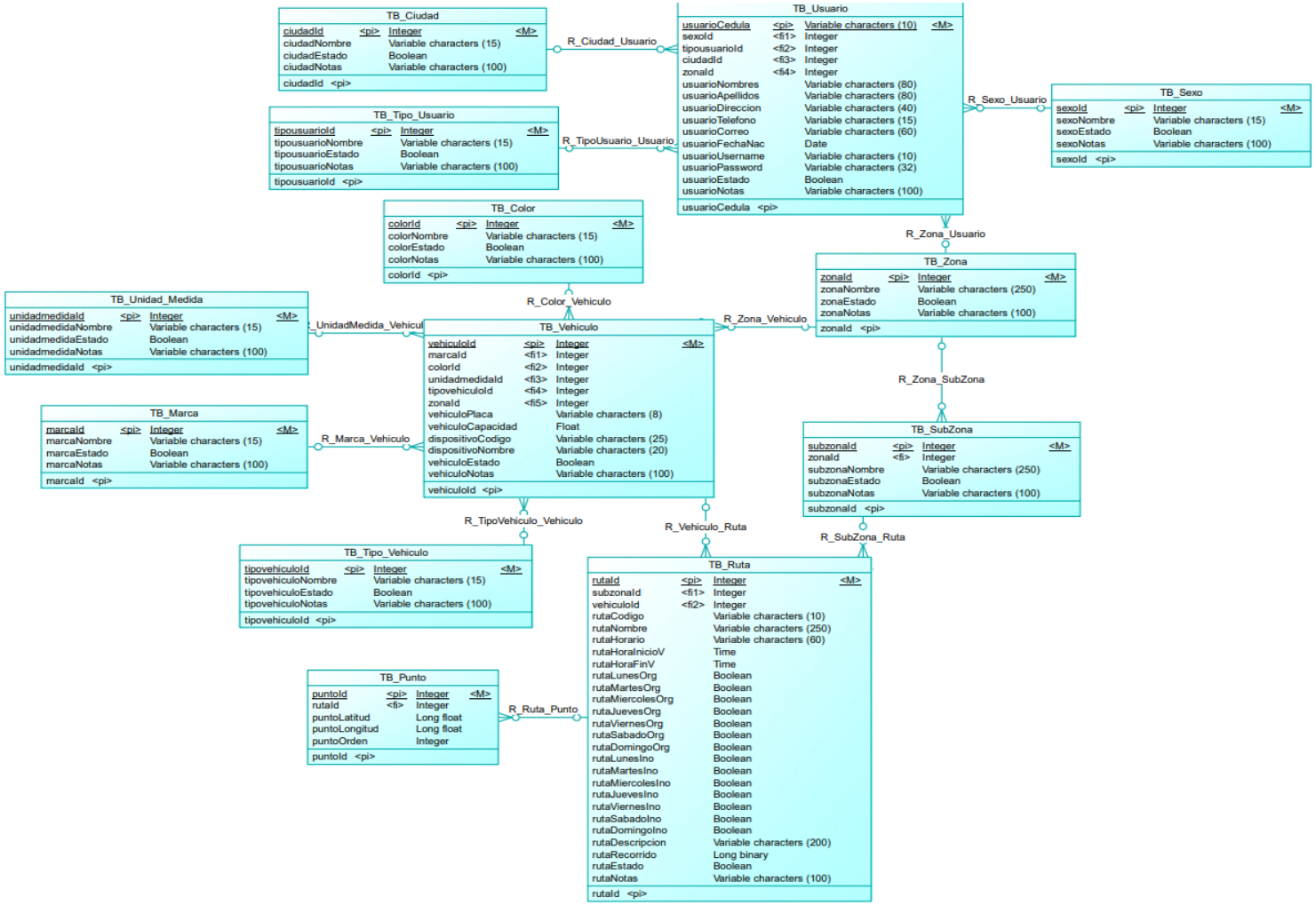


Ilustración 51. Modelo Lógico de Datos en Power Designer. Fuente: Autor.

Modelo Físico de Datos

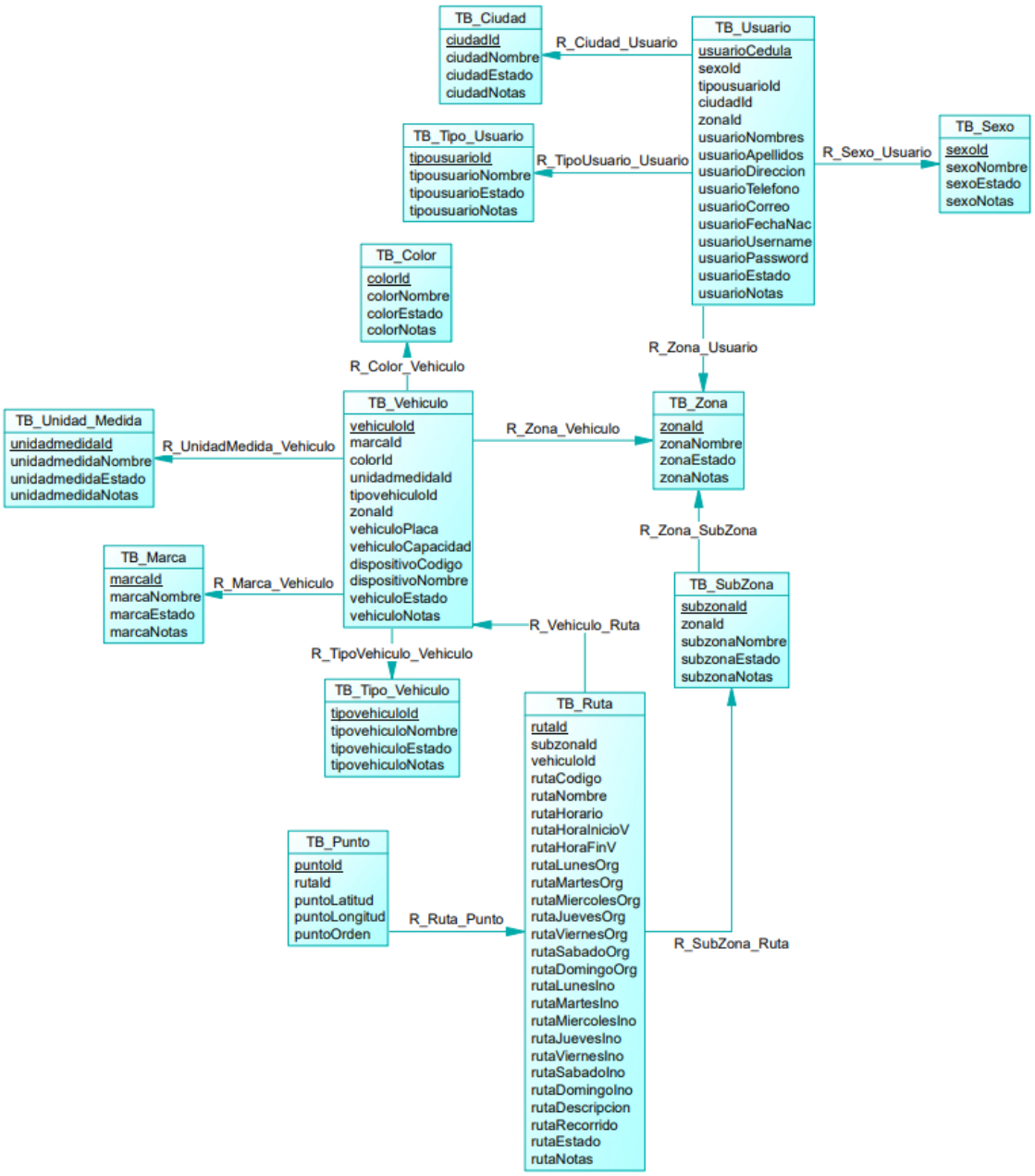


Ilustración 52. Modelo Físico de Datos en Power Designer. Fuente: Autor.

Con el modelo físico de datos, se procede a generar el respectivo script de la base de datos para ser ejecutado dentro de la plataforma phpMyAdmin en una base de datos MySQL.

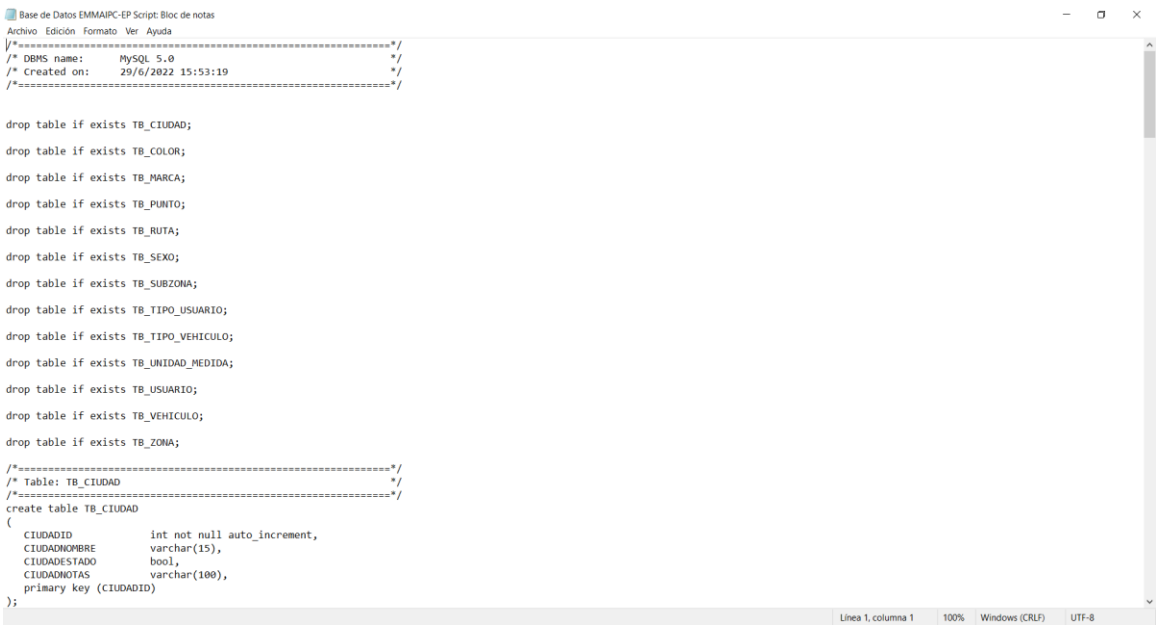


Ilustración 53. Script generado del Modelo Físico de Datos en Power Designer. Fuente: Autor.

Se procede a iniciar los módulos MySQL y Apache desde el panel de control de XAMPP.

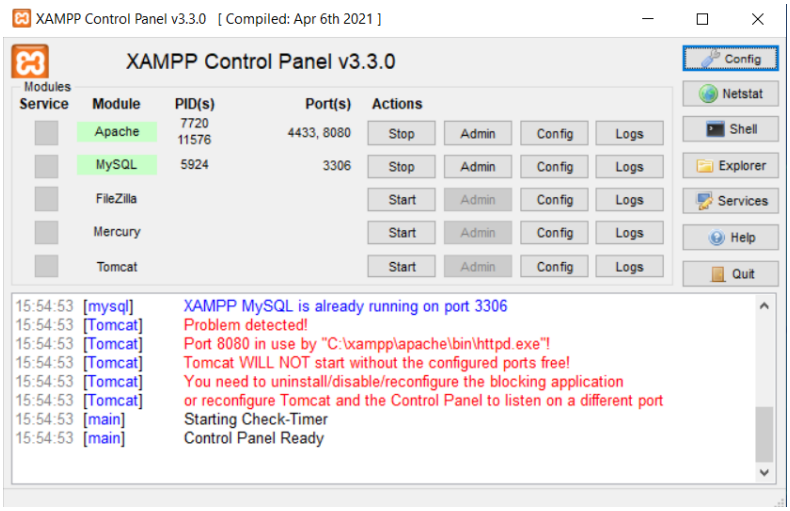


Ilustración 54. Ejecución de los Módulos Apache y MySQL en el Panel de Control de XAMPP. Fuente: Autor.

Se crea una base de datos de MySQL con el nombre db_emmaipc_app en phpMyAdmin, ingresando por http://localhost:8080/phpmyadmin/index.php.

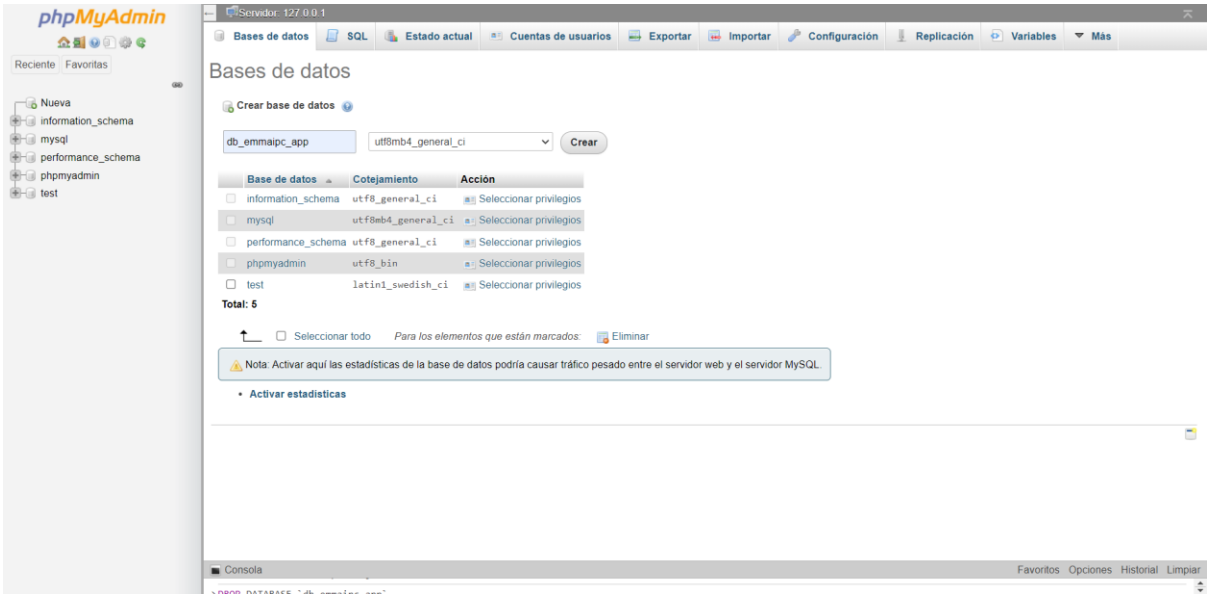


Ilustración 55. Creación de la Base de Datos EMMAIPC-EP en phpMyAdmin. Fuente: Autor.

Se procede a ingresar a la base de datos creada y ejecutar una consulta SQL con el script del modelo físico de datos generado en Power Designer.

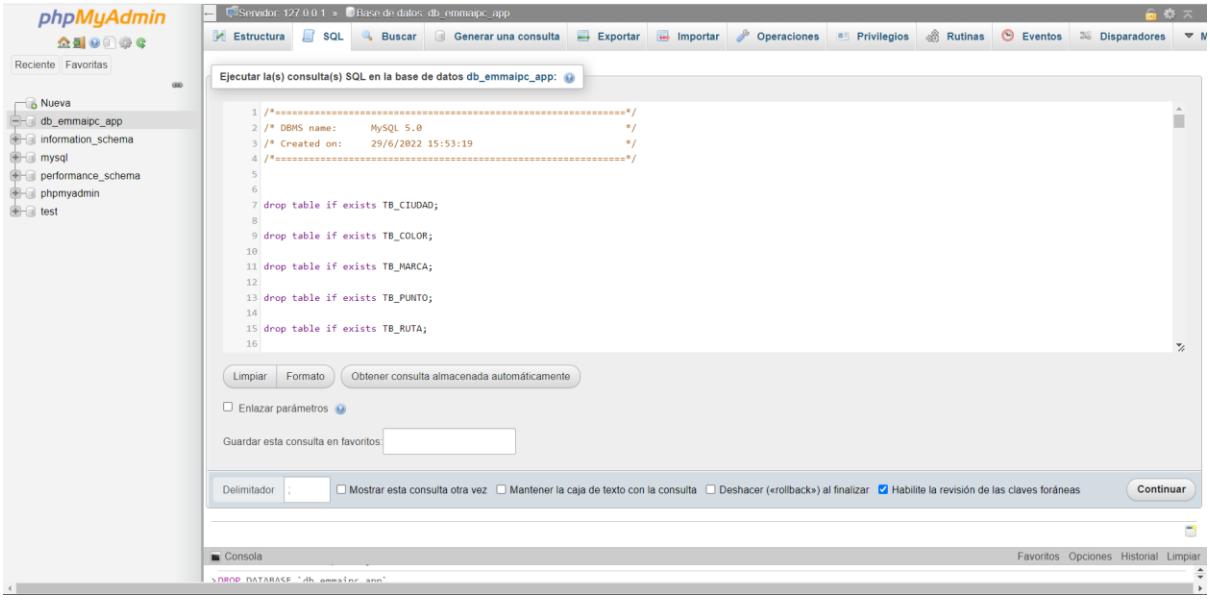


Ilustración 56. Ejecución del Script generado del Modelo Físico de Datos en phpMyAdmin. Fuente: Autor.

Se verifica la creación de las tablas a través del script en la base de datos db_emmaipc_app en phpMyAdmin

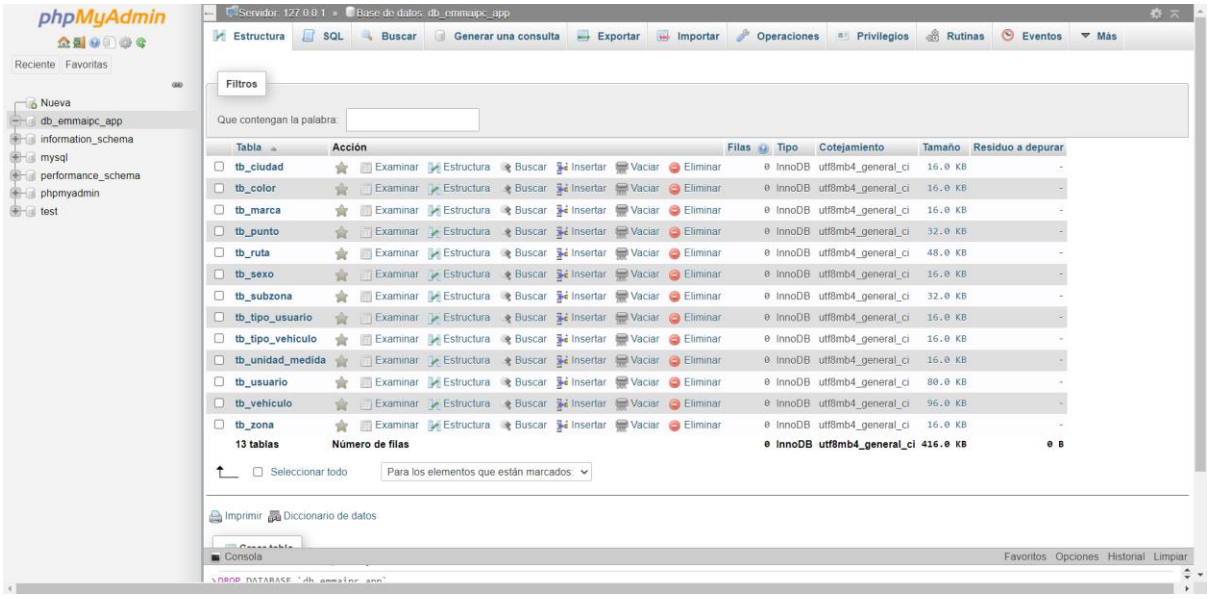


Ilustración 57. Visualización de la creación de las tablas al ejecutar el Script del Modelo Físico de Datos. Fuente: Autor.

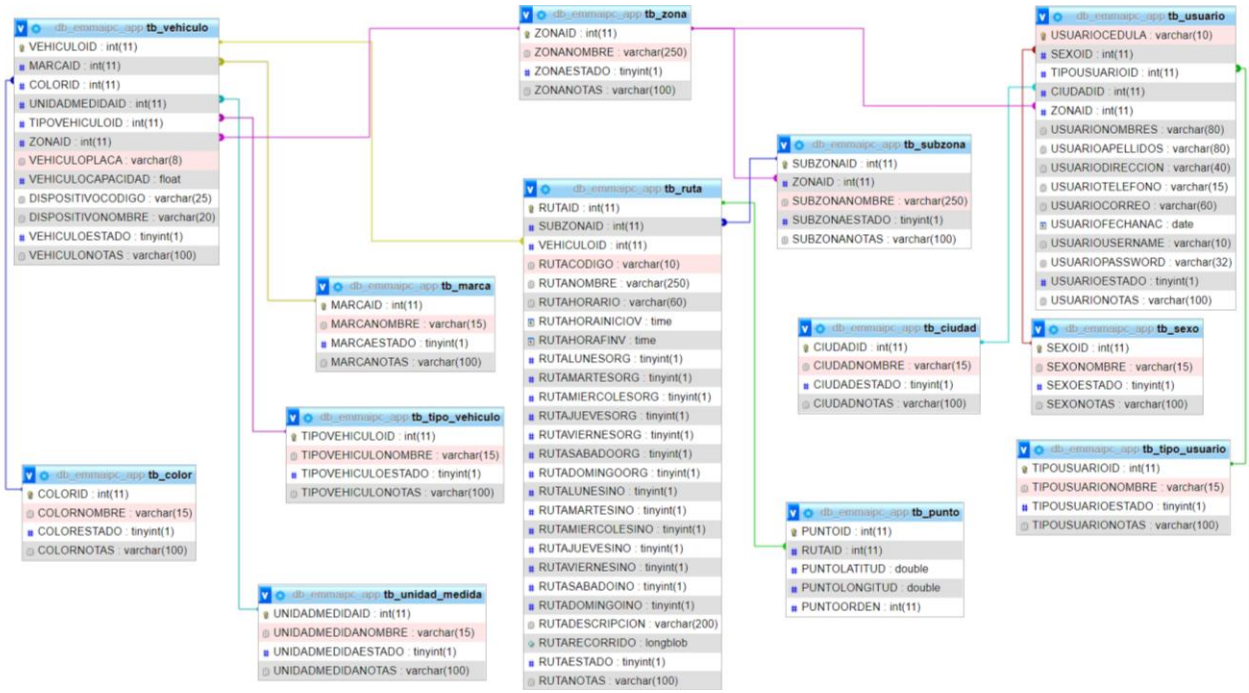


Ilustración 58. Modelo Relacional de la Base de Datos EMMAIPC-EP en phpMyAdmin. Fuente: Autor.

Interpretación de las Tablas de la Base de Datos de la EMMAIPC-EP

La mayoría de las tablas presentadas en el modelo relacional son tablas auxiliares, las cuales son similares unas con otras y cumplen la funcionalidad de realizar el registro de los diferentes campos de las tablas principales de una manera ágil y segura.

Tablas Auxiliares

- TB_CIUDAD
- TB_TIPO_USUARIO
- TB_SEXO
- TB_MARCA
- TB_COLOR
- TB_TIPO_VEHICULO
- TB_UNIDAD_MEDIDA

Tablas Principales

- **TB_ZONA:** Esta tabla es considerada debido a que la EMMAIPC-EP tiene categorizadas las rutas de recolección en sub zonas y a su vez en zonas. Además, se la considero para que el usuario pueda identificar con mayor facilidad la ruta que le corresponde.
- **TB_SUBZONA:** Esta tabla es considerada para tener una categorización más específica de las zonas, con el objeto de que el usuario pueda identificar con mayor facilidad la ruta que le corresponde.
- **TB_USUARIO:** Esta tabla es considerada ya que el sistema de información que permite gestionar el aplicativo cuenta con un inicio de sesión y con usuarios que tienen diferentes privilegios dentro del sistema.

- **TB_VEHICULO:** Esta tabla es considerada ya que permite registrar información del vehículo recolector, así como también información referente al dispositivo GPS, con el fin de poder geolocalizar dicho vehículo.
- **TB_RUTA:** Esta tabla es considerada ya que permite registrar todas las rutas de recorrido, horarios de recolección, días correspondientes al tipo de desecho y asignación del vehículo recolector correspondiente. Ayuda a visualizar en el aplicativo móvil la ruta de manera gráfica e información mencionada anteriormente que corresponde a la misma.

Instalación del Servidor Traccar e Incorporación a la Base de Datos EMMAIPC-EP

Para almacenar los datos de geolocalización que son enviados por los vehículos recolectores o dispositivos GPS se utiliza un servidor de la plataforma de software libre Traccar, el mismo que se incorpora a la base de datos EMMAIPC-EP.

Proceso de Instalación Servidor Traccar

Para la instalación del servidor Traccar solamente se descarga el instalador desde la página oficial, ya sea este para sistemas operativos Linux o Windows. En este caso, se utiliza la versión de Windows.

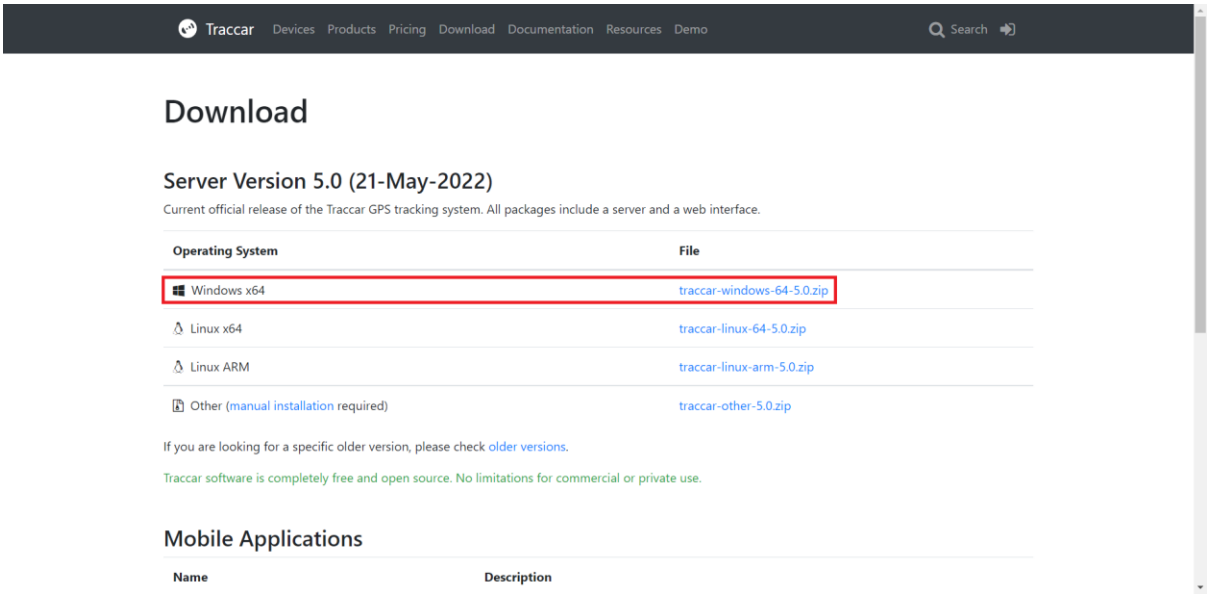


Ilustración 59. Página de descarga del Servidor Traccar. Fuente: https://www.traccar.org/download/

Una vez descargado el archivo se descomprime y se ejecuta el instalador del servidor Traccar.

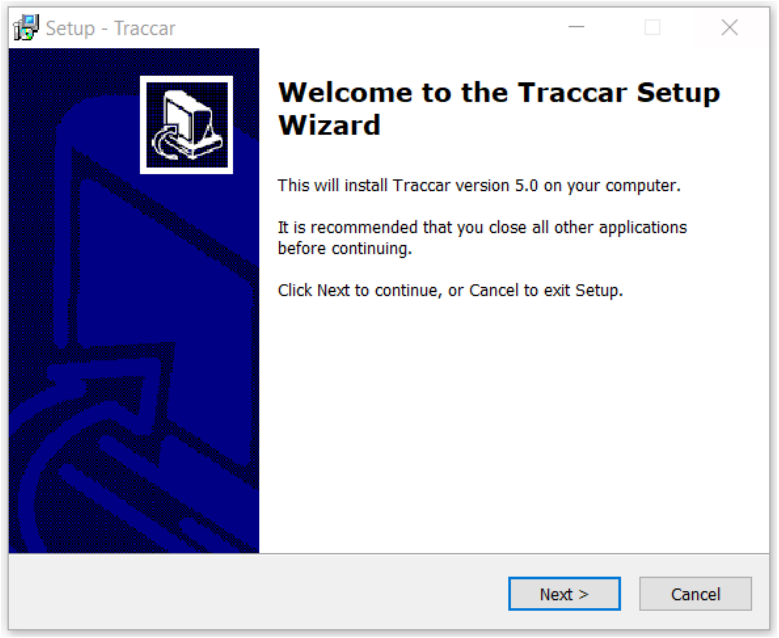


Ilustración 60. Pantalla de instalación del Servidor Traccar. Fuente: Autor.

Finalizada la instalación del servidor Traccar en el ordenador, se accede a la carpeta de instalación de Traccar y se modifica ciertos parámetros del archivo traccar.xml que se encuentra en la siguiente ruta: C:\Program Files\Traccar\conf\traccar.xml.

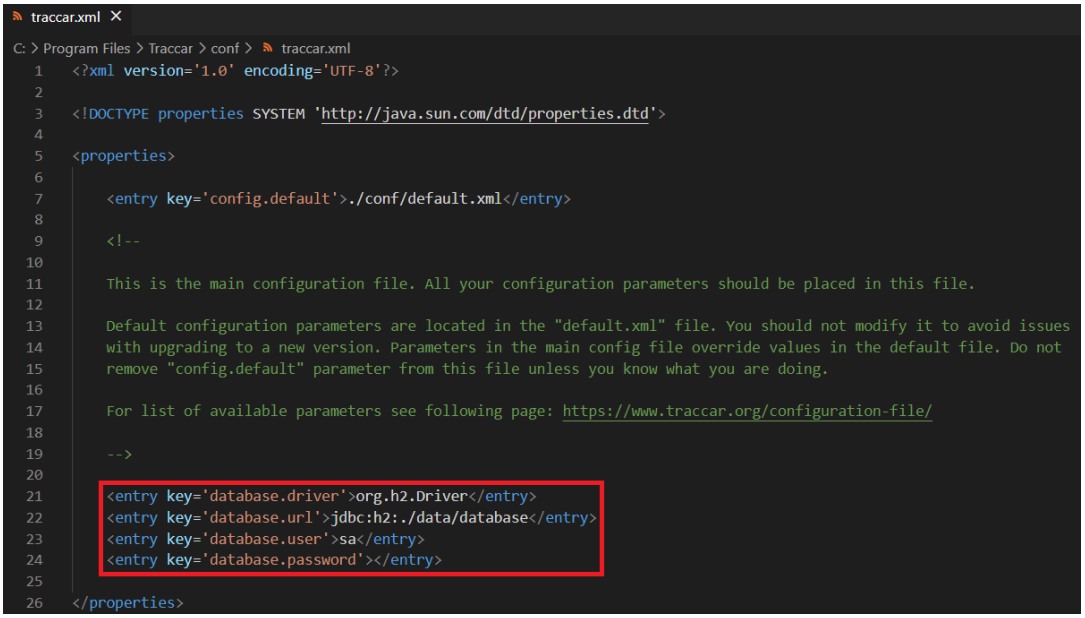


Ilustración 61. Visualización de los parámetros a modificar en el archivo traccar.xml. Fuente: Autor.

Se modifica las líneas de código señaladas anteriormente por las siguientes:

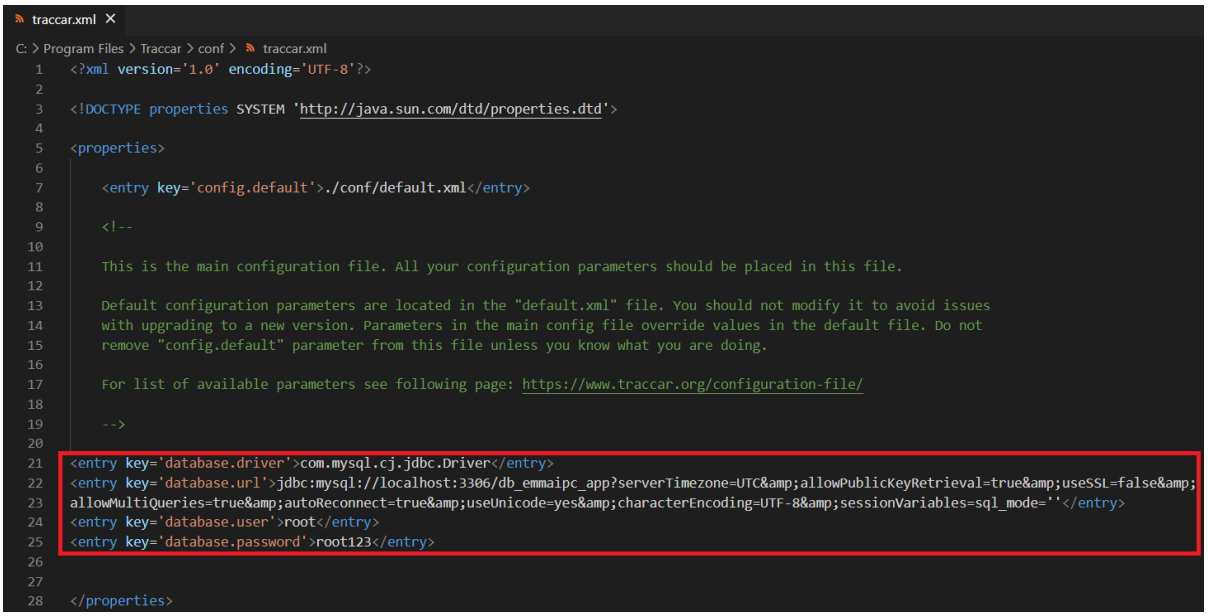


Ilustración 62. Visualización de los parámetros una vez modificaos para la Base de Datos MySQL en el archivo traccar.xml. Fuente: Autor.

Donde se coloca, a más del driver de MySQL y su dirección URL, los datos respectivos de la base de datos, como el nombre de la misma, su usuario y contraseña de acceso.

Una vez realizada dichas modificaciones se guarda el archivo traccar.xml y se procede a ejecutar el servicio de Traccar, ingresando a la pantalla de Servicios de Windows, para ello, antes de iniciar este servicio se tiene que tener iniciado ya los módulos MySQL y Apache en XAMPP.

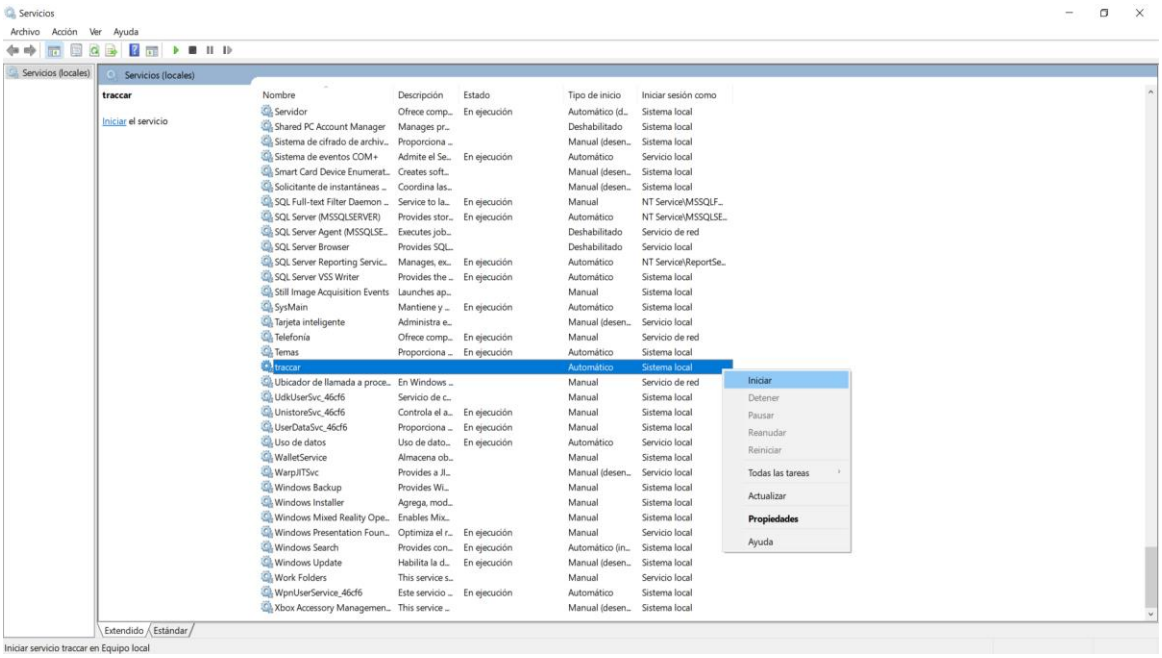


Ilustración 63. Iniciación del Servicio de Traccar en Windows. Fuente: Autor.

Al momento de iniciar el servicio, el servidor Traccar crea las tablas de su base de datos necesarias para su funcionamiento dentro de la base de datos “db_emmaipc_app” creada en MySQL. Esta acción se verifica ingresando a phpMyAdmin y visualizando las tablas existentes en la base de datos “db_emmaipc_app”.

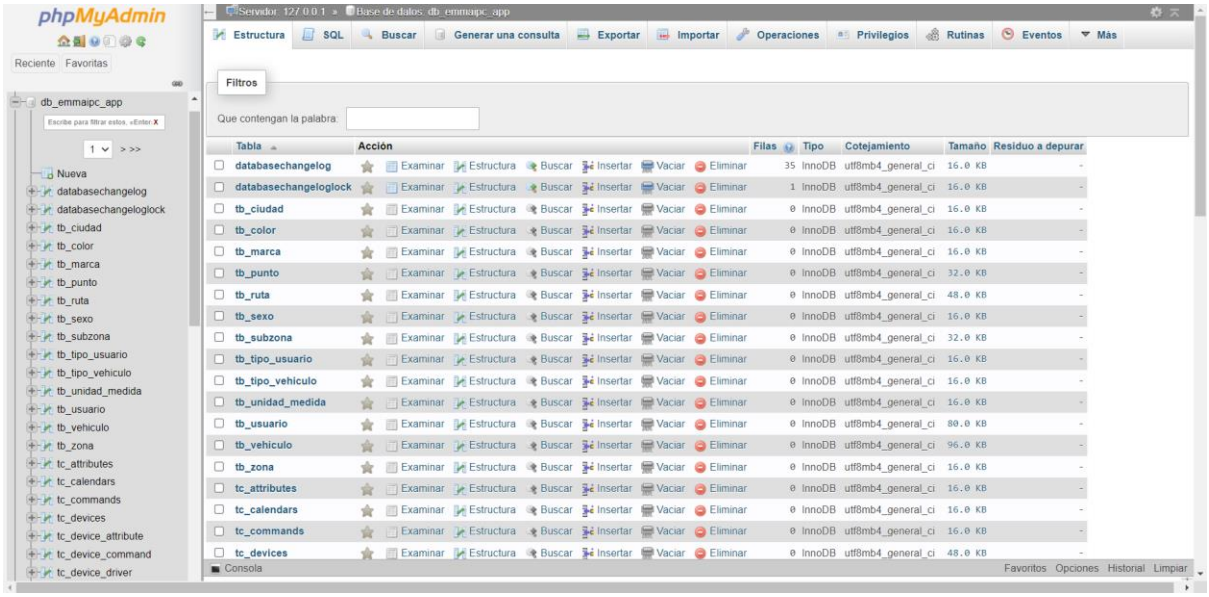


Ilustración 64. Visualización de la creación de las Tablas de la Base de Datos del Servidor Traccar dentro de la Base de DATOS EMMAIPC-EP en phpMyAdmin. Fuente: Autor.

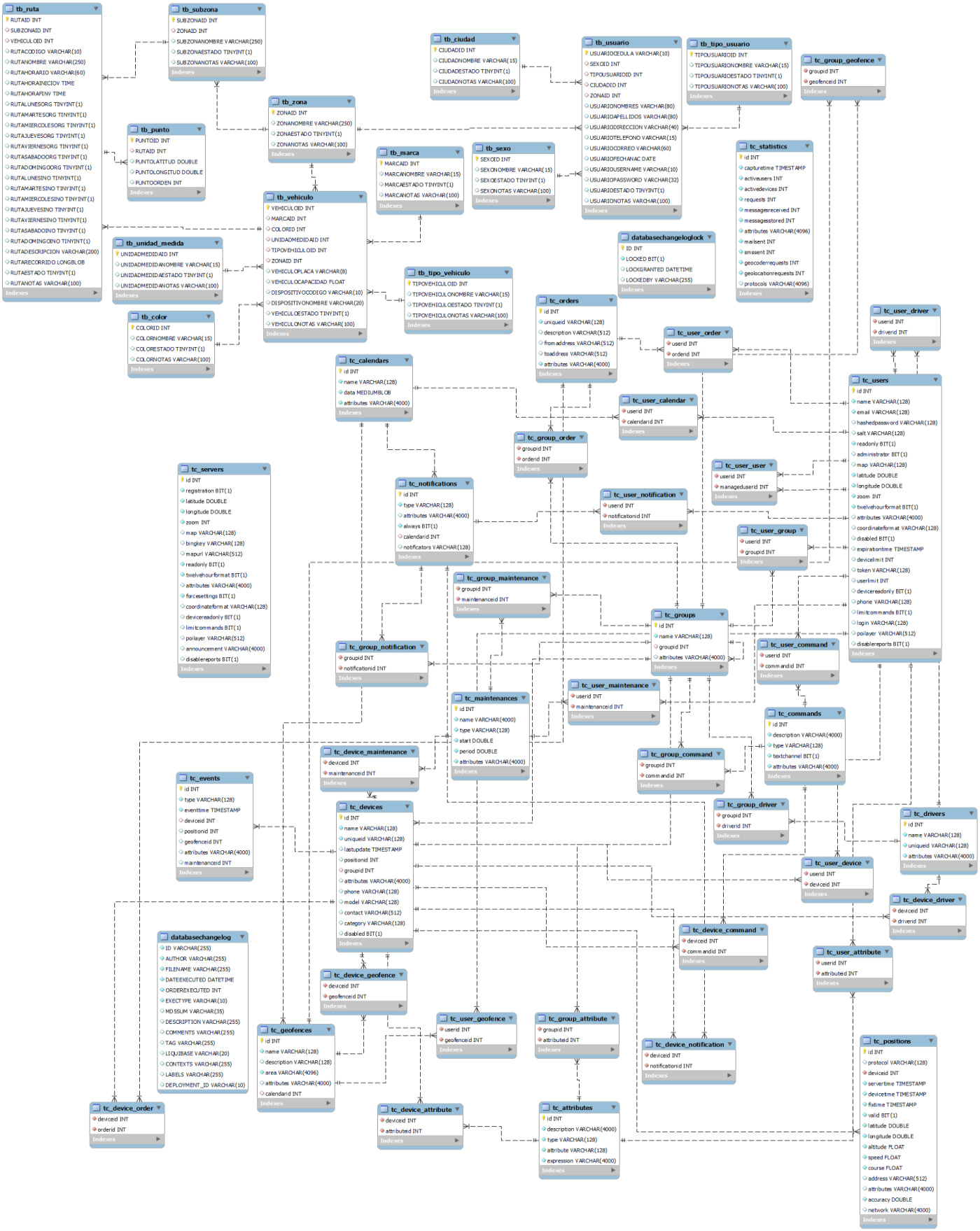


Ilustración 65. Modelo Relacional en phpMyAdmin de la Base de Datos del Servidor Traccar junto con la Base de Datos EMMAIPC-EP. Fuente: Autor.

Programación e Implementación de API REST

API REST

La programación del API REST es realizada en el lenguaje PHP a través del editor de código fuente Visual Studio Code, funcionando en el servidor APACHE de manera local mediante XAMPP.

El API REST permite la interacción entre el sistema de información y el aplicativo móvil, junto con la base de datos de la EMMAIPC-EP.

Estructuración del API REST en Visual Studio Code

El API REST se encuentra estructurado de manera general de la siguiente manera dentro de Visual Studio Code.

Ilustración 66. Estructuración del Proyecto API REST en Visual Studio Code. Fuente: Autor.

Directorio Config.

En el apartado “config”, se encuentran creados dos archivos .php que permiten configurar ciertos parámetros del API REST.

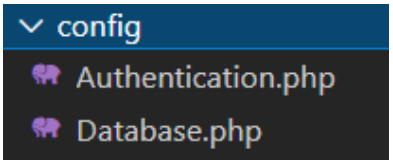


Ilustración 67. Estructuración Directorio Config. Fuente: Autor.

Authentication.php

Permite crear una Autenticación Básica para el acceso a los registros del API REST, configurando el usuario y contraseña del mismo.

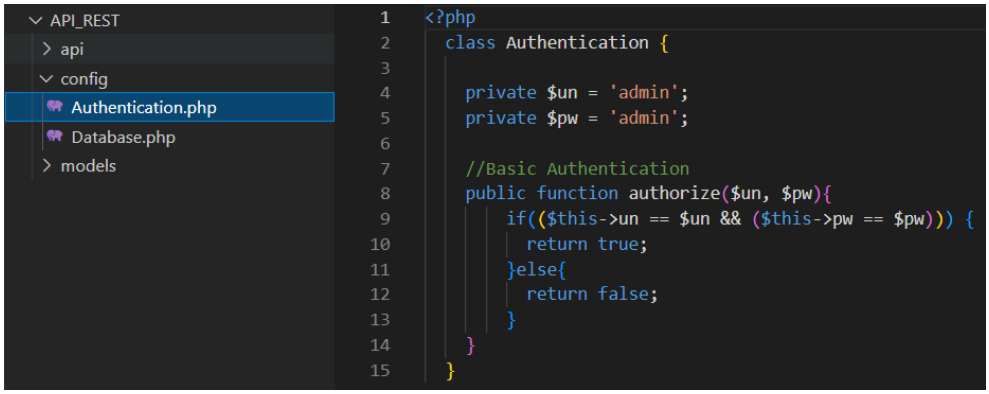


Ilustración 68. Código Fuente de Authentication.php. Fuente: Autor.

Database.php

Permite configurar los parámetros necesarios para la conexión con la base de datos "db_emmaipc_app".

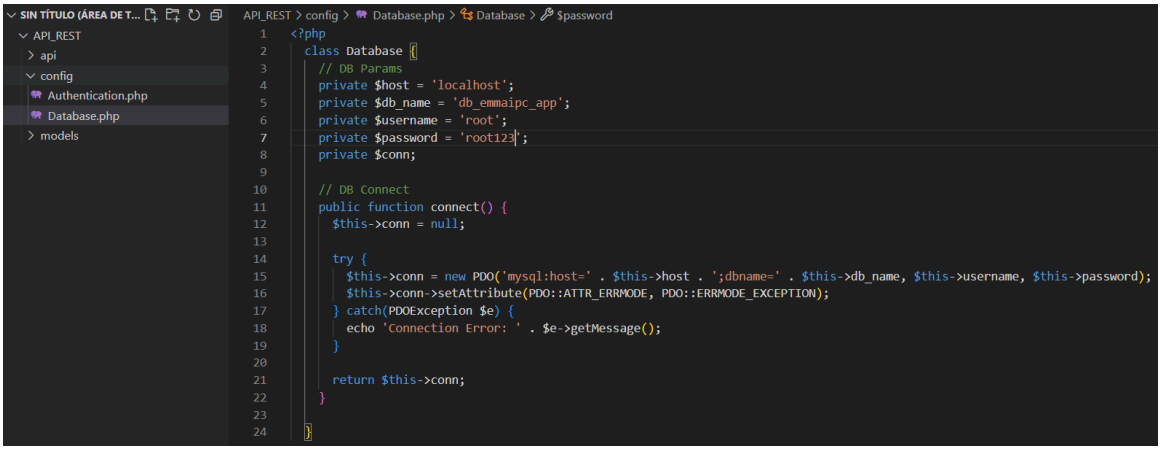


Ilustración 69. Código Fuente de Database.php. Fuente: Autor.

Directorio API.

En el apartado de "api", involucra sub carpetas que contienen el nombre de los modelos y dentro de ellas todos los archivos .php respectivos para realizar el proceso CRUD que son accedidos o ejecutados a través de la URL del API REST.

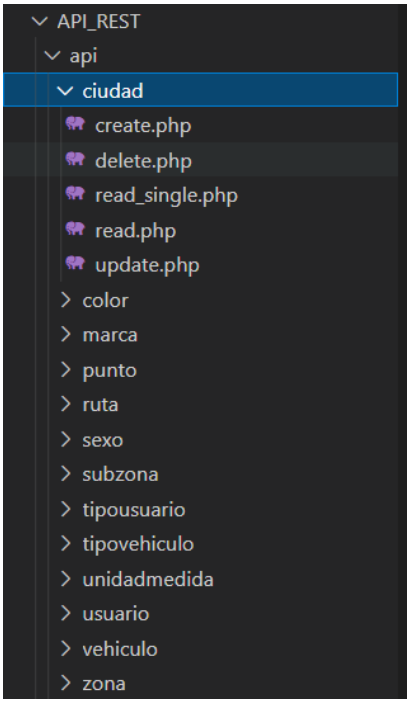


Ilustración 70. Estructuración Directorio API. Fuente: Autor.

Contenido Archivos PHP Directorio API

Primeramente, se realiza una comprobación de las credenciales ingresadas en API REST a través de la URL.

```
    include_once
'../../config/Authentication.php';

    // Instantiate DB & authorize
    $authentication = new Authentication();
    $auth = $authentication->authorize($_SERVER[
'PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW']);
```

Ilustración 71. Código Fuente Comprobación Credenciales API REST. Fuente: Autor.

Una vez que se verifica que las credenciales son exitosas se procede a obtener los datos respectivos de la entidad, que se encuentran en el archivo tipo JSON enviado a través del API REST, para ser procesado y enviado a la ejecución del método CRUD respectivo que se encuentra en la clase de la entidad dentro del paquete models.

```php
if($auth) {
  include_once '../../config/Database.php';
  include_once '../../models/Ciudad.php';

  // Instantiate DB & connect
  $database = new Database();
  $db = $database->connect();

  // Instantiate blog post object
  $ciudad = new Ciudad($db);

  // Get raw posted data
  $data = json_decode(file_get_contents("php://input"));

  $ciudad->CIUDADNOMBRE = $data->ciudadNombre;
  $ciudad->CIUDADESTADO = $data->ciudadEstado;
  $ciudad->CIUDADNOTAS = $data->ciudadNotas;

  // Create post
  if($ciudad->create()) {
    echo json_encode(
      array('message' => 'Ciudad Creada')
    );
  } else {
    echo json_encode(
      array('message' => 'Ciudad No Creada')
    );
  }
}else{
  header ("WWW-Authenticate: Basic realm=\"Private Area\"");
  header ("HTTP/1.0 401 Unauthorized");
  print "Lo sentimos, necesita las credenciales adecuadas";
}
```

Ilustración 72. Código Fuente de la Obtención de Datos tipo JSON de la Entidad Ciudad enviados al API REST. Fuente: Autor.

Directorio Models

En el apartado “models”, se encuentran todas las entidades de la base de datos “db_emmaipc_app” representados en archivos .php. Cada uno de los archivos contiene sus respectivos atributos y métodos CRUD para la interacción directa de los mismos con la base de datos.

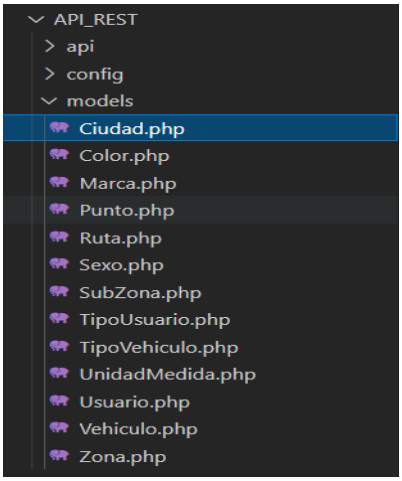


Ilustración 73. Estructuración del Directorio Models. Fuente: Autor.

Contenido Archivos PHP Directorio Models

En la clase de la entidad se declaran los atributos correspondientes a la misma.

```php
class Ciudad {
  // DB stuff
  private $conn;
  private $table = 'tb_ciudad';

  // Post Properties
  public $CIUDADID ;
  public $CIUDADNOMBRE;
  public $CIUDADESTADO;
  public $CIUDADNOTAS;
```

Ilustración 74. Código Fuente Declaración Atributos de la Entidad Ciudad. Fuente: Autor.

En este apartado se podrá visualizar cada uno de los métodos CRUD que contiene la clase, donde se utiliza sentencias SQL para ser ejecutadas a través del conector de base de datos y realizar la respectiva operación en la misma.

```php
public function create() {
    // Create query
    $query = 'INSERT INTO ' . $this->table . ' SET CIUDADNOMBRE = :CIUDADNOMBRE, CIUDADESTADO = :CIUDADESTADO,
    CIUDADNOTAS = :CIUDADNOTAS';

    // Prepare statement
    $stmt = $this->conn->prepare($query);

    // Clean data
    $this->CIUDADNOMBRE = htmlspecialchars(strip_tags($this->CIUDADNOMBRE));
    $this->CIUDADESTADO = htmlspecialchars(strip_tags($this->CIUDADESTADO));
    $this->CIUDADNOTAS = htmlspecialchars(strip_tags($this->CIUDADNOTAS));

    // Bind data
    $stmt->bindParam(':CIUDADNOMBRE', $this->CIUDADNOMBRE);
    $stmt->bindParam(':CIUDADESTADO', $this->CIUDADESTADO);
    $stmt->bindParam(':CIUDADNOTAS', $this->CIUDADNOTAS);

    // Execute query
    if($stmt->execute()) {
      return true;
  }

  // Print error if something goes wrong
  printf("Error: %s.\n", $stmt->error);

  return false;
}
```

Ilustración 75. Código Fuente del Método Crear de la Entidad Ciudad. Fuente: Autor.

```php
public function update() {
    // Create query
    $query = 'UPDATE ' . $this->table . '
                SET CIUDADNOMBRE = :CIUDADNOMBRE, CIUDADESTADO = :CIUDADESTADO, CIUDADNOTAS = :CIUDADNOTAS
                WHERE CIUDADID = :CIUDADID';

    // Prepare statement
    $stmt = $this->conn->prepare($query);

    // Clean data
    $this->CIUDADID = htmlspecialchars(strip_tags($this->CIUDADID));
    $this->CIUDADNOMBRE = htmlspecialchars(strip_tags($this->CIUDADNOMBRE));
    $this->CIUDADESTADO = htmlspecialchars(strip_tags($this->CIUDADESTADO));
    $this->CIUDADNOTAS = htmlspecialchars(strip_tags($this->CIUDADNOTAS));

    // Bind data
    $stmt->bindParam(':CIUDADID', $this->CIUDADID);
    $stmt->bindParam(':CIUDADNOMBRE', $this->CIUDADNOMBRE);
    $stmt->bindParam(':CIUDADESTADO', $this->CIUDADESTADO);
    $stmt->bindParam(':CIUDADNOTAS', $this->CIUDADNOTAS);

    // Execute query
    if($stmt->execute()) {
      return true;
    }

    // Print error if something goes wrong
    printf("Error: %s.\n", $stmt->error);

    return false;
}
```

Ilustración 76. Código Fuente del Método Actualizar de la Entidad Ciudad. Fuente: Autor.

```php
public function delete() {
    // Create query
    $query = 'DELETE FROM ' . $this->table . ' WHERE CIUDADID = :CIUDADID';

    // Prepare statement
    $stmt = $this->conn->prepare($query);

    // Clean data
    $this->CIUDADID = htmlspecialchars(strip_tags($this->CIUDADID));

    // Bind data
    $stmt->bindParam(':CIUDADID', $this->CIUDADID);

    // Execute query
    if($stmt->execute()) {
    return true;
    }

    // Print error if something goes wrong
    printf("Error: %s.\n", $stmt->error);

    return false;
}
```

Ilustración 77. Código Fuente del Método Eliminar de la Entidad Ciudad. Fuente: Autor.

```php
public function read() {
    // Create query
    $query = 'SELECT * FROM ' . $this->table . ' ORDER BY CIUDADID ASC';

    // Prepare statement
    $stmt = $this->conn->prepare($query);

    // Execute query
    $stmt->execute();

    return $stmt;
}

// Get Single Post
public function read_single() {
    // Create query
    $query = 'SELECT * FROM ' . $this->table . ' WHERE CIUDADID=:CIUDADID';

    // Prepare statement
    $stmt = $this->conn->prepare($query);

    // Bind data
    $stmt->bindParam(':CIUDADID', $this->CIUDADID);

    // Execute query
    $stmt->execute();

    $row = $stmt->fetch(PDO::FETCH_ASSOC);

    // Set properties
    $this->CIUDADID = $row['CIUDADID'];
    $this->CIUDADNOMBRE = $row['CIUDADNOMBRE'];
    $this->CIUDADESTADO = $row['CIUDADESTADO'];
    $this->CIUDADNOTAS = $row['CIUDADNOTAS'];
}
```

Ilustración 78. Código Fuente del Método Obtener y Obtener Todo de la Entidad Ciudad. Fuente: Autor.

Implementación del API REST en XAMPP

Para la implementación del API REST se procede a colocar la carpeta “API_REST”, que contiene todos las sub carpetas y archivos php. Indicados anteriormente dentro del siguiente directorio de instalación de XAMPP: C:\xampp\htdocs.

Se inicia los módulos MySQL y Apache de XAMPP, luego se accede al API REST mediante la dirección URL: http://localhost:8080/API_REST/api/...

Se ingresa el username y password configurado en el archivo Authentication.php para acceder a los registros.

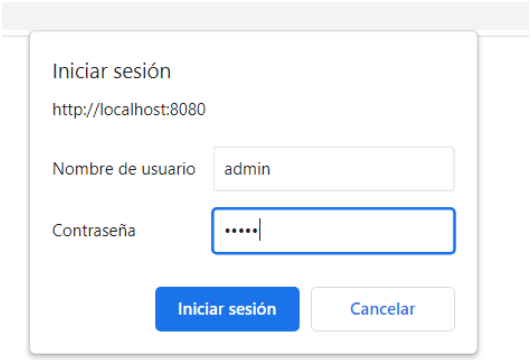


Ilustración 79. Acceso a los Datos del API REST mediante Autentificación. Fuente: Autor.

API REST del Servidor Traccar

Traccar ofrece una API para interactuar y manipular los datos de cada una de sus entidades, la misma que viene incorporada dentro del paquete de instalación de Traccar, es decir, que al instalar Traccar en el computador esta API ya está inmersa.

Para hacer uso de ella se tiene ingresar a la interfaz de Traccar a través de la URL: http://localhost:8082/, e iniciar sesión con email y contraseña por defecto: admin, admin.

Ilustración 80. Interfaz de Acceso a la Plataforma Traccar. Fuente: Autor.

Posteriormente se debe crear una cuenta para cambiar los datos de inicio de sesión inicial que se muestran en la ilustración 85. Para ello se debe colocar un nombre de usuario, el email y una nueva contraseña.

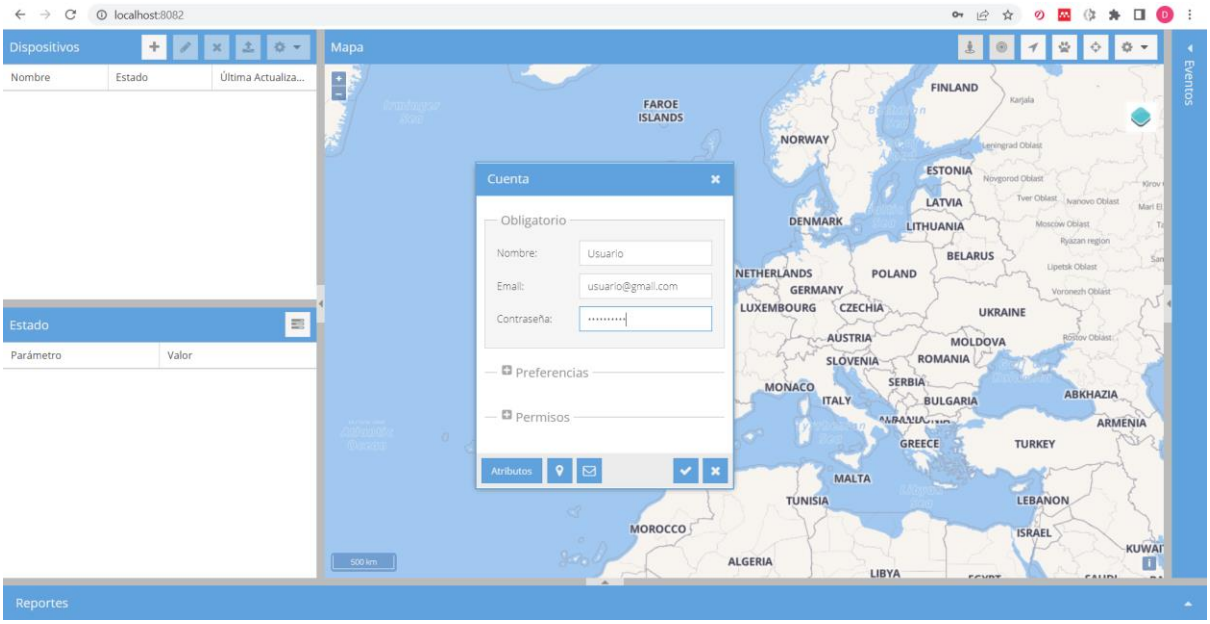


Ilustración 81. Creación de una Cuenta Administrador en la Plataforma Traccar. Fuente: Autor.

Una vez creada la cuenta con email y contraseña, se obtiene las credenciales necesarias para acceder a los registros que ofrece la API, mediante su URL: http://localhost:8082/api/...

Los métodos que se utilizan de esta API para el funcionamiento del prototipo desarrollado son los siguientes, con respecto a cada una de las clases de Traccar utilizadas.

Clase Devices

Esta clase de Traccar se utiliza con el objetivo de crear, actualizar, eliminar y obtener los datos e información de los dispositivos GPS, que se encuentran almacenados en el servidor Traccar. El API de esta clase es utilizada mayormente por el sistema de información para la gestión del aplicativo móvil, específicamente en la Pantalla Vehículo, ya que la creación, actualización, eliminación y obtención de estos registros se encuentra sincronizada con esta clase, a través de su API.

A continuación, se presenta el Request Body en formato JSON, de cada uno de los métodos utilizados de la API de la clase Devices.

Método POST

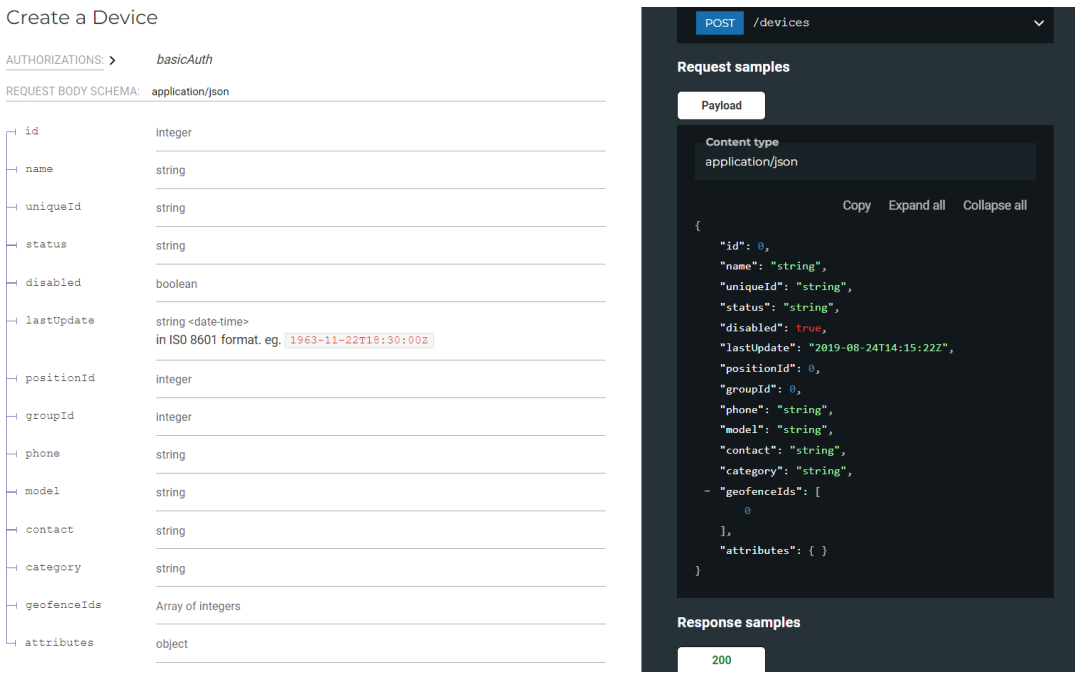


Ilustración 82. Método POST Devices. Fuente: https://www.traccar.org/api-reference/#tag/Devices/paths/~1devices/post.

Método PUT

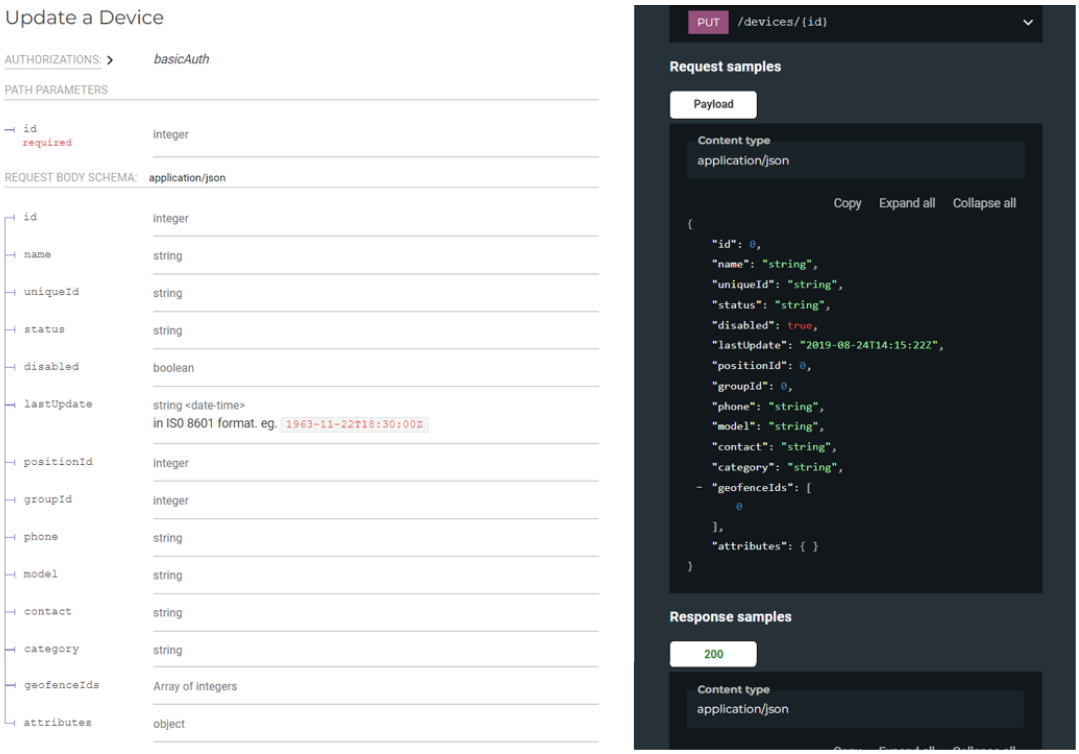


Ilustración 83. Método PUT Devices. Fuente: https://www.traccar.org/api-reference/#tag/Devices/paths/~1devices~1%7Bid%7D/put.

Método DELETE

Ilustración 84. Método DELETE Devices. Fuente: https://www.traccar.org/api-reference/#tag/Devices/paths/~1devices~1%7Bid%7D/delete.

Método GET

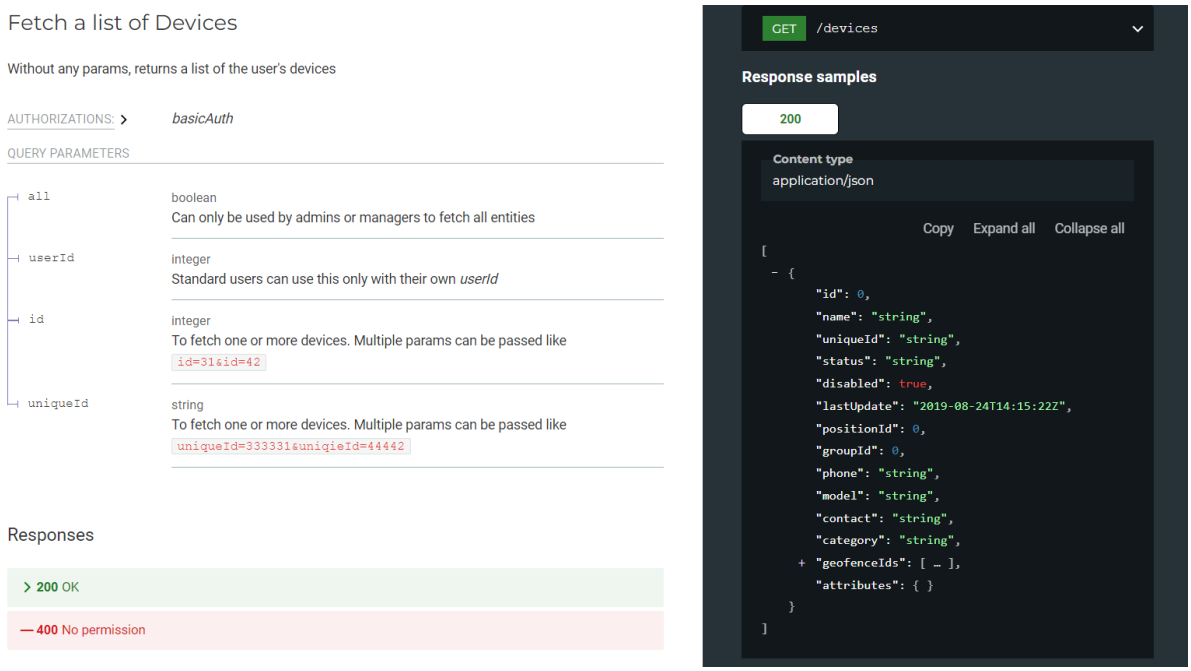


Ilustración 85. Método GET Devices. Fuente: https://www.traccar.org/api-reference/#tag/Devices/paths/~1devices/get.

Clase Positions

Esta clase de Traccar se utiliza con el objeto de poder obtener los datos de las posiciones de geolocalización de cada uno de los dispositivos GPS (Devices), es decir, Latitud y Longitud de la ubicación del dispositivo. La API de esta clase fue mayormente utilizada por el aplicativo móvil, específicamente al momento de identificar la ubicación del vehículo recolector correspondiente a la ruta en el mapa, permitiendo su visualización en tiempo real, ya que constantemente el aplicativo está obteniendo los datos de geolocalización a través de está API.

A continuación, se presenta el Request Body en formato JSON, del método utilizado de la API de la clase Positions, ya que solamente cuenta con un método.

Método GET

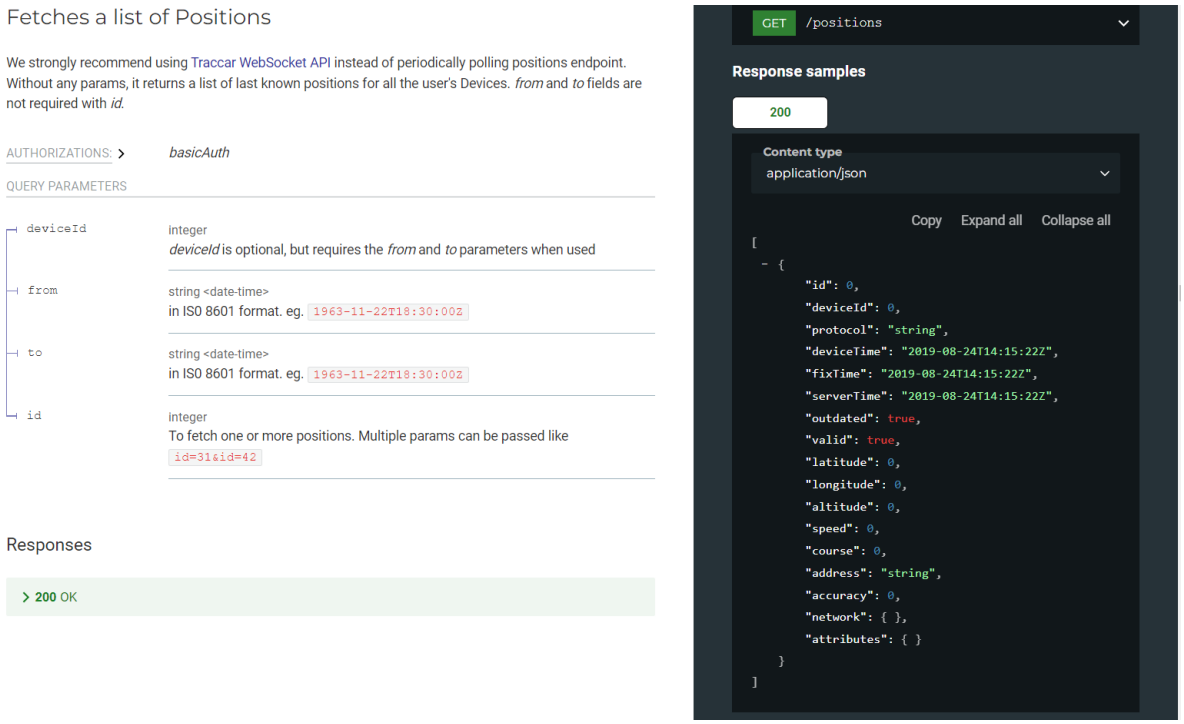


Ilustración 86. Método GET Positions. Fuente: https://www.traccar.org/api-reference/#tag/Positions/paths/~1positions/get.

Configuración y Envió de datos de Geolocalización del Dispositivo GPS al Servidor Traccar

El Servidor Traccar es compatible con una gran cantidad de dispositivos GPS que existen en el mercado. Traccar identifica a cada uno de estos mediante un puerto configurado en su servidor, para que al momento de que se apunte a una URL desde la configuración del dispositivo GPS, se adjunte el número de puerto correspondiente al dispositivo.

La información para identificar el puerto que corresponde a cada uno de los dispositivos GPS que son compatibles con la plataforma Traccar, se encuentran en su página oficial: https://www.traccar.org/devices/.

Para el desarrollo del prototipo se utiliza el aplicativo Traccar Client para simular el funcionamiento de un dispositivo GPS real, a través de la instalación del APK en un

Smartphone y configurando ciertos parámetros para que envié datos de geolocalización al servidor Traccar.

Para que el servidor Traccar obtenga datos de geolocalización del Traccar Client, se debe de ingresar el identificador del dispositivo (IMEI en Dispositivos GPS reales) en la plataforma Traccar a través de la Pantalla Vehículos del Sistema de Información desarrollado.

Traccar Client por defecto al instalar el APK en el dispositivo genera un identificador del dispositivo, que se puede apreciar en la ilustración 93.

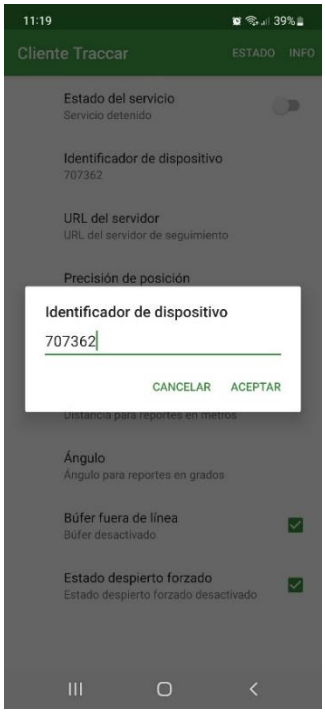


Ilustración 87. Identificador de dispositivo generado por Traccar Client. Fuente: Autor.

Configuración del Aplicativo Traccar Client

El aplicativo Traccar Client se puede descargar desde la Play Store y es compatible con dispositivos móviles iOS y Android.

Una vez instalado se procede a abrir el aplicativo, y configurar la dirección URL del servidor, añadiendo su puerto, que por defecto para Traccar Client es 5055.

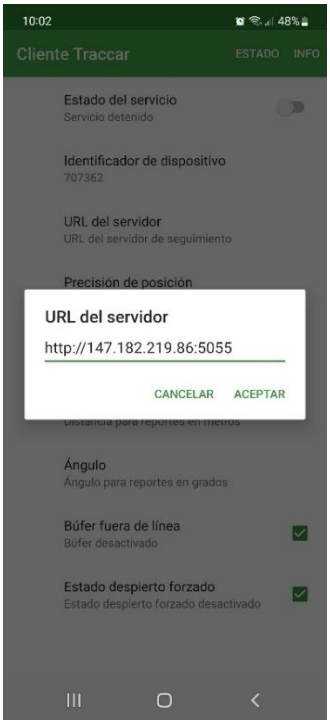


Ilustración 88. Configuración de la URL del servidor y puerto desde Traccar Client. Fuente: Autor.

También se debe de configurar la frecuencia de rastreo, es decir, la frecuencia con la que se enviaran los datos de geolocalización del Smartphone al servidor Traccar, en este caso se debe de configurar con una frecuencia de 3 segundos, para el funcionamiento óptimo del prototipo.

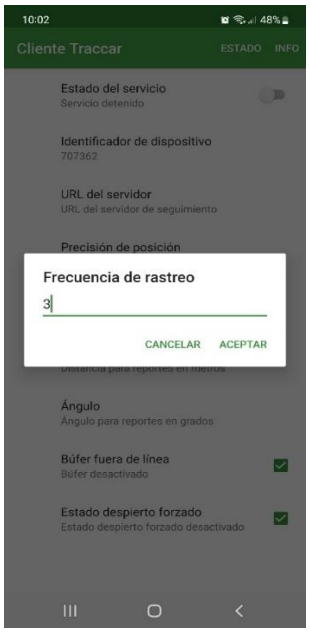


Ilustración 89. Configuración de la Frecuencia de Rastreo desde Traccar Client. Fuente: Autor.

Una vez configurado todos los parámetros necesarios, se procede a iniciar el servicio para enviar los datos de geolocalización al servidor Traccar.

Ilustración 90. Activación del servicio de Traccar Client. Fuente: Autor.

Una vez iniciado el servicio desde Traccar Client y registrado el código del dispositivo o identificador desde la Pantalla Vehículo del Sistema de Información desarrollado, se puede observar desde la plataforma Traccar el envío de datos de geolocalización del dispositivo GPS. Ingresando a la dirección URL: http://localhost:8082/.

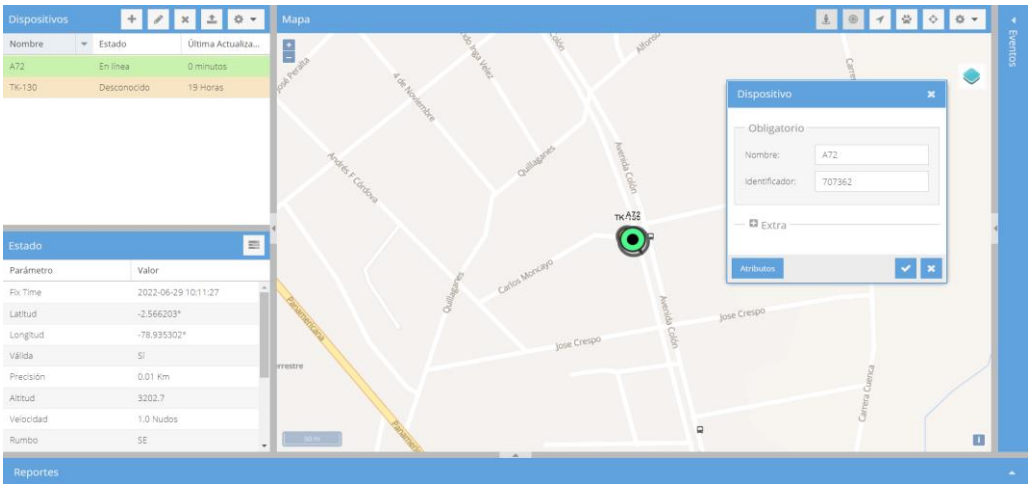


Ilustración 91. Obtención de los datos de Geolocalización del Traccar Client en la Plataforma Traccar. Fuente: Autor.

Programación Aplicativo Móvil EMMAIPC-EP

El aplicativo móvil es desarrollado en lenguaje de programación Java para dispositivos móviles con sistema operativo Android, a través del IDE Android Studio.

Dependencias

- **Retrofit:** Se considera el uso de esta dependencia con el objeto de poder realizar peticiones al servidor. Los métodos que mayormente se utilizaron de esta dependencia dentro del desarrollo fueron los métodos GET y POST, con la finalidad de obtener tanto los datos del servidor y mostrarlos al usuario mediante la interfaz del aplicativo.
- **GSON:** Se consideró el uso de esta dependencia con el objeto de poder manipular datos de tipo JSON, los mismos que se obtienen a través de los métodos HTTP, al momento de interactuar con el servidor.
- **Material Design:** Se utilizó algunas dependencias de material design, con el objetivo de presentar una interfaz más amigable al usuario a través de sus componentes que incorpora.
- **Maps SDK for Android:** Se utilizó esta dependencia con el objeto de poder hacer uso y agregar los mapas de Google dentro del aplicativo móvil, a su vez permitiendo y facilitando la integración de líneas, polígonos, y demás elementos dentro del mapa.
- **Google Play Services Location API:** Se utilizó esta dependencia con el objeto de poder identificar y actualizar la ubicación actual del usuario dentro del aplicativo móvil, haciendo uso de los servicios de google play.

Consideraciones del Proyecto

Especificación de los permisos del aplicativo en Android Manifest

Se debe colocar las líneas de código xml tal y como se muestra en la ilustración 97. Con la finalidad de que el aplicativo pueda funcionar eficientemente y obtener los accesos a dichos permisos para su ejecución.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.hardware.location.gps" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<uses-permission android:name="android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS" />
```

Ilustración 92. Especificación de Permisos del Aplicativo dentro de Android Manifest. Fuente: Autor.

Especificación del Google Maps Key dentro del proyecto generado en la plataforma Google Cloud

Primeramente, se generan dos activity denominados: SeleccionRutaActivity y GeolocalizacionActivity.

Una vez que se crean dichos activity se genera un archivo google_maps_api.xml. Donde se debe acceder a una dirección URL que proporciona dicho archivo para registrar el API y generar un API KEY para el proyecto.

```
<resources>
    <!--
    TODO: Before you run your application, you need a Google Maps API key.

    To get one, follow this link, follow the directions and press "Create" at the end:

    https://console.developers.google.com/flows/enableapi?apiid=maps_android_backend&keyType=CLIENT_SIDE_ANDROID&r=24:CE:FC:B6:4A:D8:A2:74:4F:B1:91:A1:43:DE:47:54:EF:74:15:8D%3B

    You can also add your credentials to an existing key, using these values:

    Package name:
    com.example.myapplication

    SHA-1 certificate fingerprint:
    24:CE:FC:B6:4A:D8:A2:74:4F:B1:91:A1:43:DE:47:54:EF:74:15:8D

    Alternatively, follow the directions here:
    https://developers.google.com/maps/documentation/android/start#get-key

    Once you have your key (it starts with "AIza"), replace the "google_maps_key"
    string in this file.
    -->
    <string name="google_maps_key" templateMergeStrategy="preserve" translatable="false">AIzaSyBTanXiTGE-rjuMz8XMKxMEHe578kUSGT8</string>
</resources>
```

Ilustración 93. Identificación del link de acceso para la creación y habilitación del API Key de Google Maps en google_maps_api.xml. Fuente: Autor.

Una vez que se accede al link que genera este archivo, se debe crear un proyecto en Google Cloud para registrar el API KEY de Google Maps. Una vez creado el proyecto se selecciona y se procede con el proceso de habilitar el acceso a la API de acuerdo a los siguientes pasos:

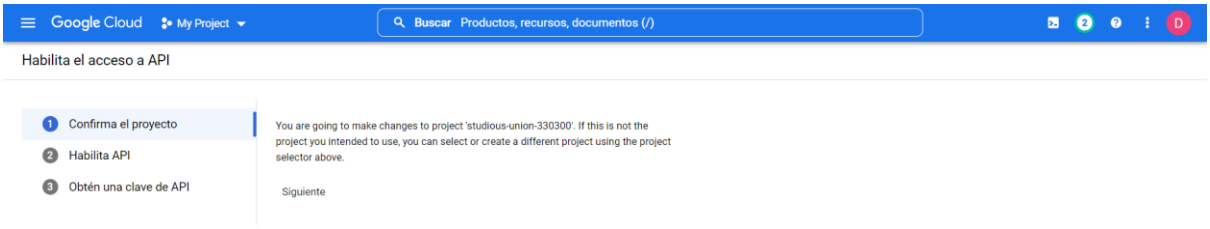


Ilustración 94. Google Cloud Console confirmación del proyecto para el API Key. Fuente: https://console.cloud.google.com/.

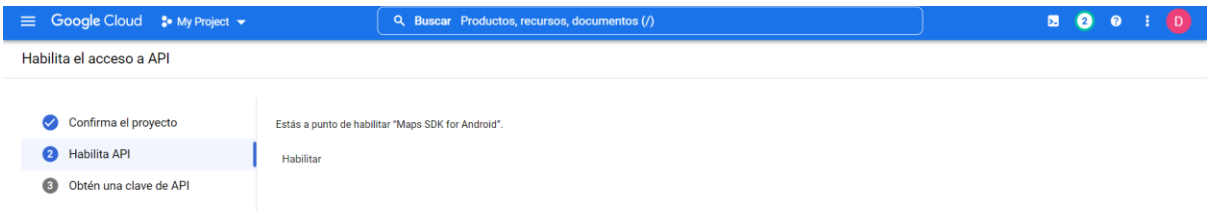


Ilustración 95. Google Cloud Console habilitación del API Key. Fuente: https://console.cloud.google.com/.

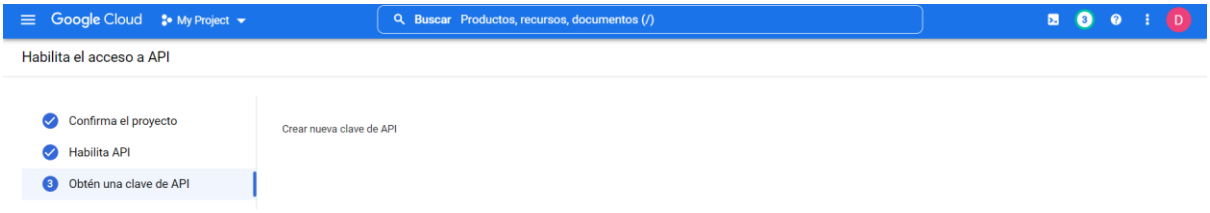


Ilustración 96. Google Cloud Console obtención de la clave API. Fuente: https://console.cloud.google.com/.

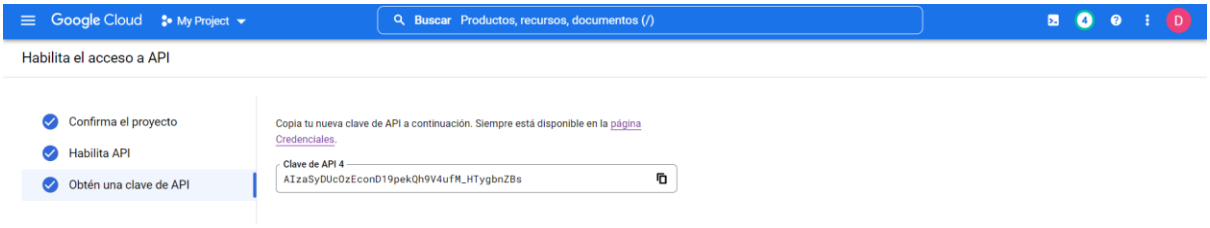


Ilustración 97. Google Cloud Console visualización de la clave API. Fuente: https://console.cloud.google.com/.

Una vez generada el API KEY se copia y se pega en el proyecto en Android Studio, específicamente en el archivo google_maps_api.xml. Una vez realizado esto el proyecto del aplicativo podrá hacer uso de los mapas de google.

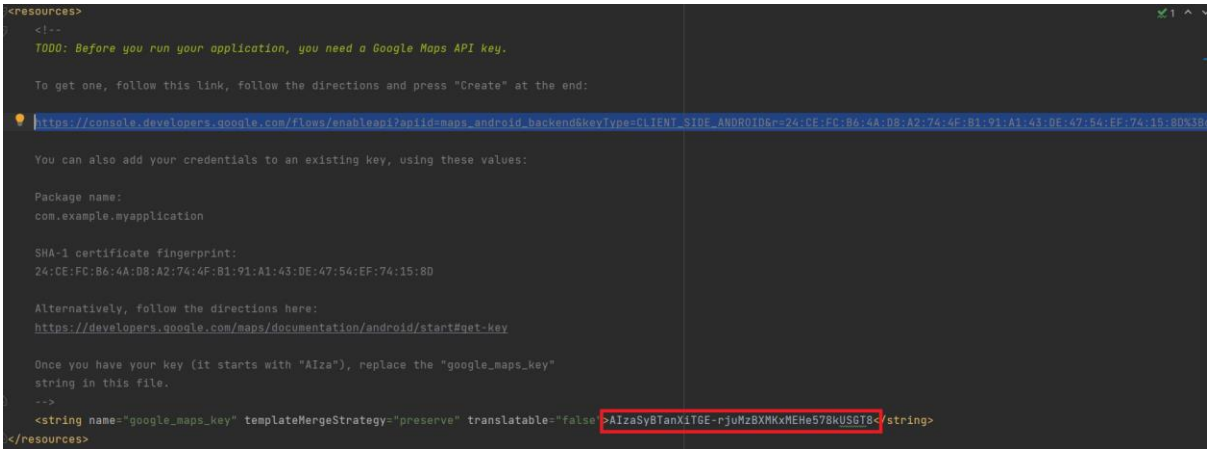


Ilustración 98. Colocación del API Key dentro de google_maps_api.xml. Fuente: Autor.

Especificación de las dependencias dentro del Proyecto

Las dependencias que se mencionaron para este proyecto se las colocan dentro del archivo build.gradle.

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'com.google.android.gms:play-services-maps:17.0.1'
    implementation 'com.google.android.gms:play-services-location:19.0.0'
    implementation 'com.google.android.libraries.places:places:2.4.0'

    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'

    implementation 'com.google.android.material:material:1.3.0'
    implementation 'com.google.code.gson:gson:2.8.7'
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
}
```

Ilustración 99. Visualización de las Dependencias del Proyecto dentro de build.gradle. Fuente: Autor.

Estructuración del Proyecto y Programación

Capa de Datos

Dentro del proyecto del aplicativo movil se considera la creación del paquete de Capa de Datos, misma que contiene algunas de las entidades de la base de datos EMMAIPC-EP, como las siguientes:

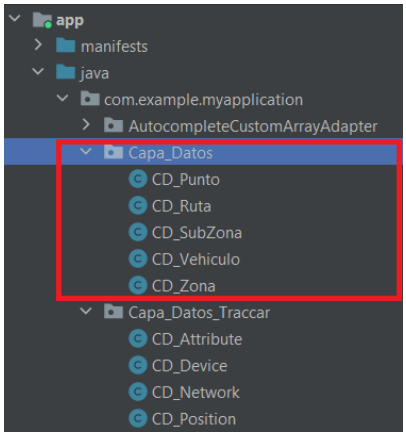


Ilustración 100. Paquete Capa de Datos Aplicativo. Fuente: Autor.

Las mismas que fueron consideradas con el objeto de poder consumir los datos que proveen cada una de estas entidades a través del API REST desarrollado.

```
public class CD_Zona {
    private int zonaId;
    private String zonaNombre;
    private int zonaEstado;
    private String zonaNotas;

    public CD_Zona(String zonaNombre, int zonaEstado, String zonaNotas) {
        this.zonaNombre = zonaNombre;
        this.zonaEstado = zonaEstado;
        this.zonaNotas = zonaNotas;
    }

    public CD_Zona(int zonaId) { this.zonaId = zonaId; }

    public CD_Zona() {}

    public CD_Zona(int zonaId, String zonaNombre, int zonaEstado, String zonaNotas) {
        this.zonaId = zonaId;
        this.zonaNombre = zonaNombre;
        this.zonaEstado = zonaEstado;
        this.zonaNotas = zonaNotas;
    }
```

Ilustración 101. Atributos y Métodos Constructores Entidad Ciudad Aplicativo. Fuente: Autor.

```
public int getZonaId() { return zonaId; }

public String getZonaNombre() { return zonaNombre; }

public int getZonaEstado() { return zonaEstado; }

@Override
public String toString() { return zonaNombre; }
```

Ilustración 102. Métodos Getters y Setters Entidad Ciudad Aplicativo. Fuente: Autor.

Capa de Datos Traccar

Otro paquete de Capa de Datos que se considera es el paquete de Traccar, donde se encuentran algunas entidades de la base de datos de Traccar, siendo las siguientes:

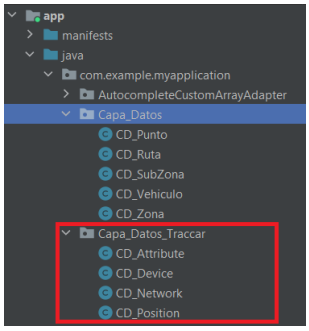


Ilustración 103. Paquete Capa de Datos Traccar Aplicativo Móvil. Fuente: Autor.

Cada una de las clases dentro de estos paquetes, están conformadas por sus atributos, métodos constructores y métodos getter and setter.

```
public class CD_Device {
    private String id;
    private CD_Attribute attributes;
    private Long groupId;
    private String name;
    private String uniqueId;
    private String status;
    private Date lastUpdate;
    private Long positionId;
    private List<Long> geofenceIds;
    private String phone;
    private String model;
    private String contact;
    private String category;
    private Boolean disabled;
```

Ilustración 104. Atributos de la Entidad Device Aplicativo Móvil. Fuente: Autor.

```
public String getId() { return id; }

public void setId(String id) { this.id = id; }

public Long getPositionId() { return positionId; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getUniqueId() { return uniqueId; }

public String getStatus() { return status; }
```

Ilustración 105. Métodos Getters y Setters Entidad Device Aplicativo Móvil. Fuente: Autor.

Interfaces Retrofit

Dentro del proyecto del aplicativo movil se considera la creación del paquete de Interfaces Retrofit, el mismo que contiene todas las interfaces que tienen relación con las clases creadas en el paquete Capa de Datos, las mismas que contienen los métodos HTTP necesarios para la obtención de los datos de las entidades de la base de datos EMMAIPC-EP a través del API REST.

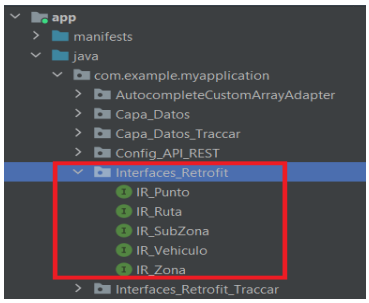


Ilustración 106. Paquete Interfaces Retrofit Aplicativo Móvil. Fuente: Autor.

```
public interface IR_Zona {

    @GET("/REST_API/api/zona/read.php")
    Call<List<CD_Zona>> findAll(@Header("Authorization") String authHeader);
}
```

Ilustración 107. Métodos HTTP Interfaz Retrofit Zona Aplicativo Móvil. Fuente: Autor.

Interfaces Retrofit Traccar

Se crea otro paquete para las interfaces Retrofit de Traccar mismas que se encuentran relacionadas con algunas clases del paquete “Capa de Datos Traccar”, con la finalidad de interactuar con los datos de entidades de la base de datos mediante el API REST Traccar.

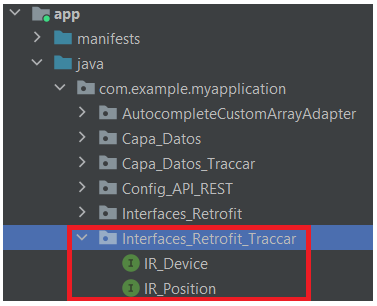


Ilustración 108. Paquete Interfaces Retrofit Traccar Aplicativo Móvil. Fuente: Autor.

```
public interface IR_Device {

    //Obtener un dispositivo en especifico de Traccar
    @GET("/api/devices")
    Call<List<CD_Device>> findDevice(@Header("Authorization") String authHeader, @Query("uniqueId") String id);
}
```

Ilustración 109. Métodos HTTP Interfaz Retrofit Device Traccar Aplicativo Móvil. Fuente: Autor.

Autocomplete Custom Array Adapter

Se crea este paquete con el objetivo de crear clases de un Adaptador de Array Personalizado para poder visualizar los datos de las zonas, sub zonas y rutas a través de un contenedor desplegable con el uso del componente Autocomplete.

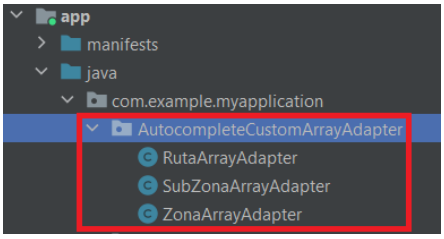


Ilustración 110. Paquete AutoCompleteCustomArrayAdapter. Fuente: Autor.

```java
public class RutaArrayAdapter extends ArrayAdapter<CD_Ruta> {
    final String TAG = "RutaArrayAdapter.java";

    Context mContext;
    int layoutResourceId;
    List<CD_Ruta> rutaList2 = new ArrayList<>();

    public RutaArrayAdapter(Context mContext, int layoutResourceId, List<CD_Ruta> rutaList2) {
        super(mContext, layoutResourceId, rutaList2);
        this.layoutResourceId = layoutResourceId;
        this.mContext = mContext;
        this.rutaList2 = rutaList2;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        try{
            if(convertView==null){
                LayoutInflater inflater = ((SeleccionRutaActivity) mContext).getLayoutInflater();
                convertView = inflater.inflate(layoutResourceId, parent,  attachToRoot: false);
            }
            CD_Ruta objectItem = rutaList2.get(position);
            TextView textViewItem = (TextView) convertView.findViewById(R.id.textViewItem);
            textViewItem.setText(objectItem.getRutaNombre());
        } catch (NullPointerException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return convertView;
    }
}
```

Ilustración 111. Código Fuente RutaArrayAdapter. Fuente: Autor.

Configuración de API REST

Se creó este paquete con el fin de almacenar una clase que permita configurar algunos parámetros del API REST desarrollada y de la API REST de Traccar.

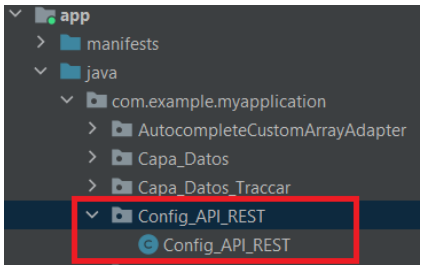


Ilustración 112. Paquete Config API REST Aplicativo Móvil. Fuente: Autor.

Dichos parámetros son:

- El username y password para crear la Basic Authentication y acceder a los datos de las API's REST

- La URL de los servidores, donde se asigna la dirección IP y el Puerto de acceso.

```
public class Config_API_REST {
    public static String username = "admin";
    public static String password = "admin";
    public static String authString = username + ":" + password;
    public static String authHeader = "Basic " + Base64.encodeToString(authString.getBytes(), Base64.NO_WRAP);
    public static Retrofit retrofit = new Retrofit.Builder().baseUrl("http://147.182.219.86:80")
            .addConverterFactory(GsonConverterFactory.create()).build();

    //Traccar
    public static String usernameTraccar = "dremko.cs@gmail.com";
    public static String passwordTraccar = "RSKdremko2017";
    public static String authStringTraccar = usernameTraccar + ":" + passwordTraccar;
    public static String authHeaderTraccar = "Basic " + Base64.encodeToString(authStringTraccar.getBytes(), Base64.NO_WRAP);
    public static Retrofit retrofitTraccar = new Retrofit.Builder().baseUrl("http://147.182.219.86:8082")
            .addConverterFactory(GsonConverterFactory.create()).build();

}
```

Ilustración 113. Código Fuente Config_API_REST Aplicativo Móvil. Fuente: Autor.

FullscreenActivity

Esta clase es la que se muestra en un inicio durante la ejecución del aplicativo. Esta clase fue creada con el objeto de proporcionar una pantalla completa de inicio con imagen de la empresa, con una duración de 2 segundos.

```
public class FullscreenActivity extends AppCompatActivity {

    Timer timer;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_fullscreen);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN, WindowManager.LayoutParams.FLAG_FULLSCREEN);
        getSupportActionBar().hide();
        timer = new Timer();
        timer.schedule(new TimerTask(){
            @Override
            public void run(){
                Intent i = new Intent( packageContext: FullscreenActivity.this, SeleccionRutaActivity.class);
                startActivity(i);
                finish();
            }
        }, delay: 2000);
    }
}
```

Ilustración 114. Código Fuente FullscreenActivity. Fuente: Autor.

SeleccionRutaActivity

Esta clase se ejecuta a continuación de la anterior, donde representa la pantalla principal del aplicativo móvil. Aquí se cargan todos los datos de las Zonas, Sub Zonas, Rutas, Horario y Días de desecho orgánico e inorgánico.

Cuando se inicia por primera vez el aplicativo solicitara permiso de ubicación a través del siguiente método:

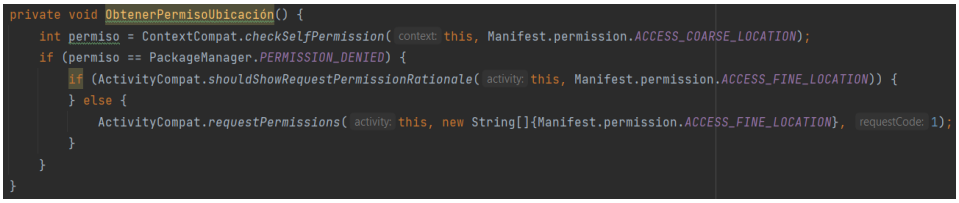


Ilustración 115. Código Fuente Método ObtenerPermisoUbicacion(). Fuente: Autor.

Otro permiso que solicitara el aplicativo es permitirle ejecutarse en segundo plano. Utilizando el siguiente método.

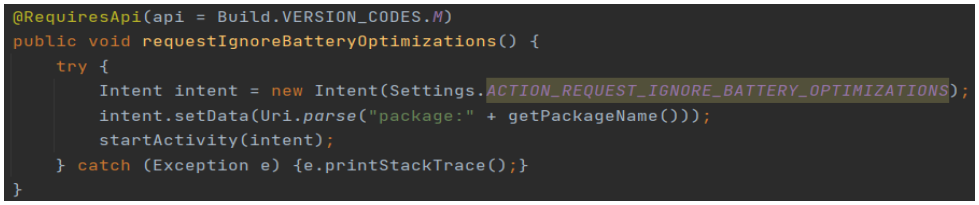


Ilustración 116. Código Fuente Método requestIgnoreBatteryOptimizations(). Fuente: Autor.

Luego de concebir los permisos, el aplicativo verifica la activación de la ubicación del dispositivo y solicita activarla en caso de que se encuentre desactivada. Utilizando el siguiente método:

```
private void VerificarEstadoGPS() {
    LocationManager locationManager = null;
    boolean gps_enabled = false;
    boolean network_enabled = false;
    if (locationManager == null) {locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);}
    try {gps_enabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);} catch (Exception ex) {}
    try {network_enabled = locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);} catch (Exception ex) {}
    if (!gps_enabled && !network_enabled) {
        new AlertDialog.Builder( context: this)
                .setTitle("Por favor, active su localización")
                .setMessage("Presione 'Ok' para ir a las configuraciones o 'Cancelar' para salir")
                .setPositiveButton(android.R.string.yes, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {
                        Intent intent = new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                        startActivity(intent);
                    }
                }).setNegativeButton(android.R.string.no, new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int which) {System.exit( status: 0);}
                }).show();
    }
}
```

Ilustración 117. Código Fuente Método VerificarEstadoGPS(). Fuente: Autor.

Para cargar las Zonas se utiliza el método findAll(); de las Interfaces Retrofit para obtener los datos del API REST y cargarlos utilizando el siguiente código dentro de un componente AutoCompleteTextView.

```
IR_Zona ir_zona = retrofit.create(IR_Zona.class);
Call<List<CD_Zona>> call = ir_zona.findAll(authHeader);
call.enqueue(new Callback<List<CD_Zona>>() {
    @Override
    public void onResponse(Call<List<CD_Zona>> call, Response<List<CD_Zona>> response) {
        zonaList = response.body();
        if(zonaList != null){
            for (CD_Zona z : zonaList) {
                if(z.getZonaEstado() == 1){
                    zonaList2.add(z);
                }
            }
            zonaArrayAdapter = new ZonaArrayAdapter( mContext: SeleccionRutaActivity.this, R.layout.items_list, zonaList2);
            autoCompleteTextViewZona.setAdapter(zonaArrayAdapter);
        }
    }
    @Override
    public void onFailure(Call<List<CD_Zona>> call, Throwable t) {
        Toast.makeText(getApplicationContext(),  text: "Sin Conexión a Internet", Toast.LENGTH_LONG).show();
    }
});
```

Ilustración 118. Código Fuente Carga de Zonas en AutoCompleteTextView. Fuente: Autor.

Posteriormente se codifica los eventos de los siguientes botones al dar click sobre ellos:

- **buttonNext:** Permite pasar a la siguiente pantalla o activity, enviando como putExtra ciertas variables indispensables para el funcionamiento del siguiente activity.
- **buttonMiUbicación:** Permite centrar y obtener la ubicación actual del usuario dentro del mapa.

```
buttonNext.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        if(vehiculoRutaSelected!=null && idRutaSeleccionada != 0 && nombreRutaSeleccionada!= null && nombreZonaSeleccionada != null &&
                nombreSubZonaSeleccionada != null){
            Intent i = new Intent( packageContext: SeleccionRutaActivity.this, ObtencionUbicacionActivity.class);
            i.putExtra( name: "vehiculoRutaSelected", vehiculoRutaSelected);
            i.putExtra( name: "vehiculoIdSelected", vehiculoIdSelected);
            i.putExtra( name: "idRutaSeleccionada", idRutaSeleccionada);
            i.putExtra( name: "nombreZonaSeleccionada", nombreZonaSeleccionada);
            i.putExtra( name: "nombreSubZonaSeleccionada", nombreSubZonaSeleccionada);
            i.putExtra( name: "nombreRutaSeleccionada", nombreRutaSeleccionada);
            i.putExtra( name: "pathEnvio", pathEnvio);
            i.putExtra( name: "lat", lat);
            i.putExtra( name: "lng", lng);
            finish();
            startActivity(i);
        }
        else{
            Toast.makeText(getApplicationContext(),  text: "Todavía no ha seleccionado una Ruta", Toast.LENGTH_LONG).show();
        }
    }
});

buttonMiUbicacion.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        verificadorRetorno=false;
        verificador=0;
        miUbicacion();
    }
});
```

Ilustración 119. Código Fuente Eventos de Componentes de Botones. Fuente: Autor.

Para obtener la ubicación actual del usuario se utiliza el siguiente método miUbicacion(). Donde se configura el proveedor de GPS y el tiempo de actualización del mismo.

```
public void miUbicacion() {
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        return;
    }
    LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
    Location location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
    actualizarUbicacion(location);
    locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,  minTimeMs: 3000,  minDistanceM: 0, locationListener);
}
```

Ilustración 120. Código Fuente Método miUbicacion(). Fuente: Autor.

Dicho método miUbicación(), va acompañado de otros que permiten actualizar la ubicación en el tiempo especificado en él.

```
LocationListener locationListener = new LocationListener() {
    @Override
    public void onLocationChanged(@NonNull Location location) {actualizarUbicacion(location);}
    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {}
    @Override
    public void onProviderEnabled(@NonNull String provider) {}
    @Override
    public void onProviderDisabled(@NonNull String provider) {}
};
```

Ilustración 121. Código Fuente LocationListener. Fuente: Autor.

El LocationListener permite llamar al método actualizarUbicacion() cada vez que se tenga una actualización de la ubicación del usuario, permitiendo en dicho método agregar el marcador de la ubicación del usuario en el mapa y su respectivo zoom.

```
private void actualizarUbicacion(Location location) {
    if (location != null) {
        lat = location.getLatitude();
        lng = location.getLongitude();
        coordenadaslatlng = new LatLng(lat, lng);
        if (verificador == 0) {
            if(!verificadorRetorno){
                CameraUpdate miUbicacion = CameraUpdateFactory.newLatLngZoom(coordenadaslatlng,  zoom: 18);
                mMap.animateCamera(miUbicacion);
                verificador = verificador + 1;
            }
        }
        if (MarkerMiUbicacion != null) MarkerMiUbicacion.remove();
        MarkerMiUbicacion = mMap.addMarker(new MarkerOptions().position(coordenadaslatlng).title("Mi Posición Actual")
                .icon(BitmapDescriptorFactory.fromResource(R.drawable.user_location_icon)).anchor( u: 0.0f,  v: 1.0f));
    }
}
```

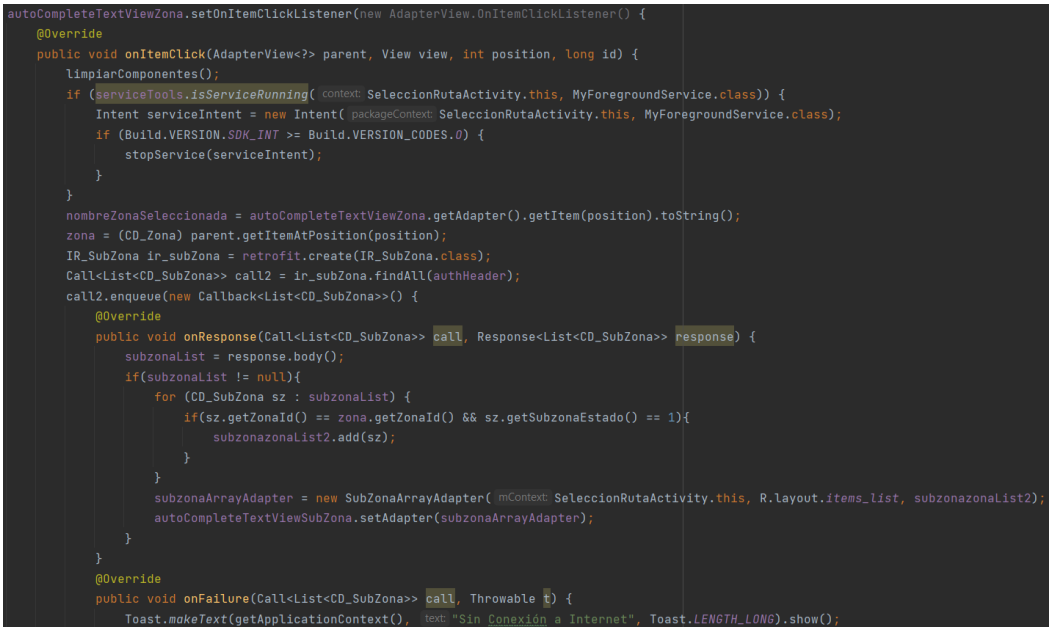
Ilustración 122. Código Fuente Método actualizarUbicacion(). Fuente: Autor.

Para que continuamente se esté verificando la ubicación del usuario, se coloca el método miUbicacion() dentro del método onMapsReady(), ya que este se encuentra en continua ejecución, a su vez acompañada de la codificación para cambiar de modo oscuro a claro.

```
@Override
public void onMapReady(GoogleMap googleMap) {
    boolean success = false;
    int nightModeFlags = getResources().getConfiguration().uiMode & Configuration.UI_MODE_NIGHT_MASK;
    switch (nightModeFlags) {
        case Configuration.UI_MODE_NIGHT_YES: success = googleMap.setMapStyle(new MapStyleOptions("[ { "featureType": "all", "elementType": "geometry", "s..."));
            break;
        case Configuration.UI_MODE_NIGHT_NO: success = googleMap.setMapStyle(new MapStyleOptions("[ { "featureType": "poi", "elementType": "labels", "sty..."));
            break;
    }
    mMap = googleMap;
    miUbicacion();
}
```

Ilustración 123. Código Fuente Método onMapReady() SeleccionRutaActivity. Fuente: Autor.

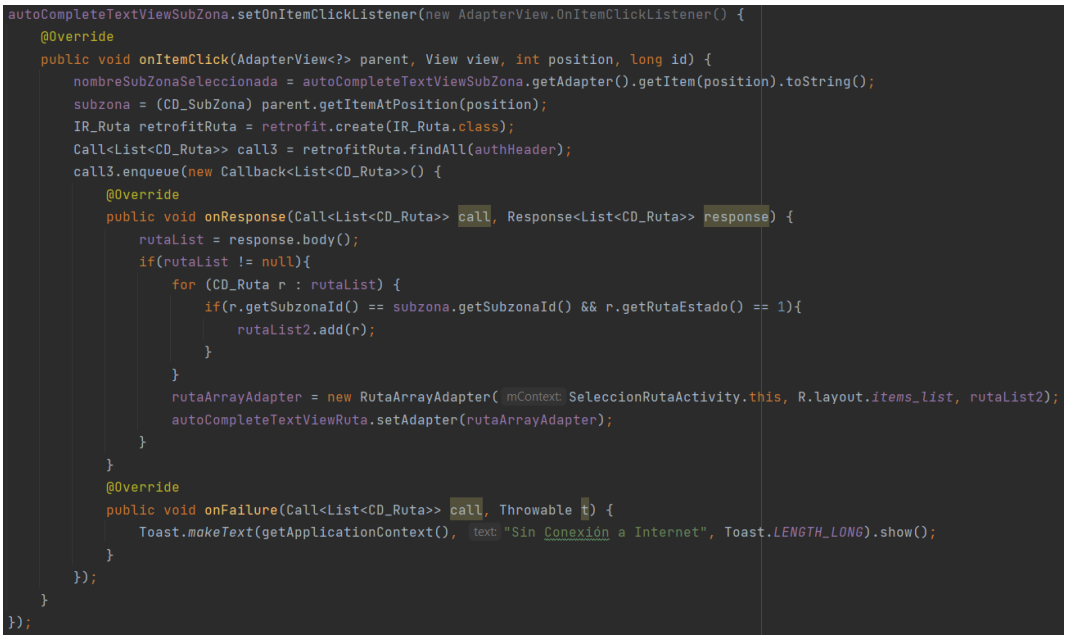
Para cargar los ítems dentro del AutoCompleteTextView de las Sub Zonas se codifico un evento dentro del AutoCompleteTextView Zona. Cuando se selecciona una Zona se ejecuta un evento que permitirá carga en el AutoCompleteTextView Sub Zona los ítems respectivos a la zona seleccionada.



```
autoCompleteTextViewZona.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        limpiarComponentes();
        if (serviceTools.isServiceRunning( context: SeleccionRutaActivity.this, MyForegroundService.class)) {
            Intent serviceIntent = new Intent( packageContext: SeleccionRutaActivity.this, MyForegroundService.class);
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
                stopService(serviceIntent);
            }
        }
        nombreZonaSeleccionada = autoCompleteTextViewZona.getAdapter().getItem(position).toString();
        zona = (CD_Zona) parent.getItemAtPosition(position);
        IR_SubZona ir_subZona = retrofit.create(IR_SubZona.class);
        Call<List<CD_SubZona>> call2 = ir_subZona.findAll(authHeader);
        call2.enqueue(new Callback<List<CD_SubZona>>() {
            @Override
            public void onResponse(Call<List<CD_SubZona>> call, Response<List<CD_SubZona>> response) {
                subzonaList = response.body();
                if(subzonaList != null){
                    for (CD_SubZona sz : subzonaList) {
                        if(sz.getZonaId() == zona.getZonaId() && sz.getSubzonaEstado() == 1){
                            subzonazonaList2.add(sz);
                        }
                    }
                    subzonaArrayAdapter = new SubZonaArrayAdapter( mContext: SeleccionRutaActivity.this, R.layout.items_list, subzonazonaList2);
                    autoCompleteTextViewSubZona.setAdapter(subzonaArrayAdapter);
                }
            }
            @Override
            public void onFailure(Call<List<CD_SubZona>> call, Throwable t) {
                Toast.makeText(getApplicationContext(),  text: "Sin Conexión a Internet", Toast.LENGTH_LONG).show();
```

Ilustración 124. Código Fuente Evento Componente AutoCompleteTextViewZona. Fuente: Autor.

Igualmente, como el evento anterior, se configura un evento para el AutoCompleteTextView de Sub Zona, para que cuando se seleccione una Sub Zona se cargue automáticamente en el AutoCompleteTextView Ruta los ítems respectivos de la sub zona seleccionada.



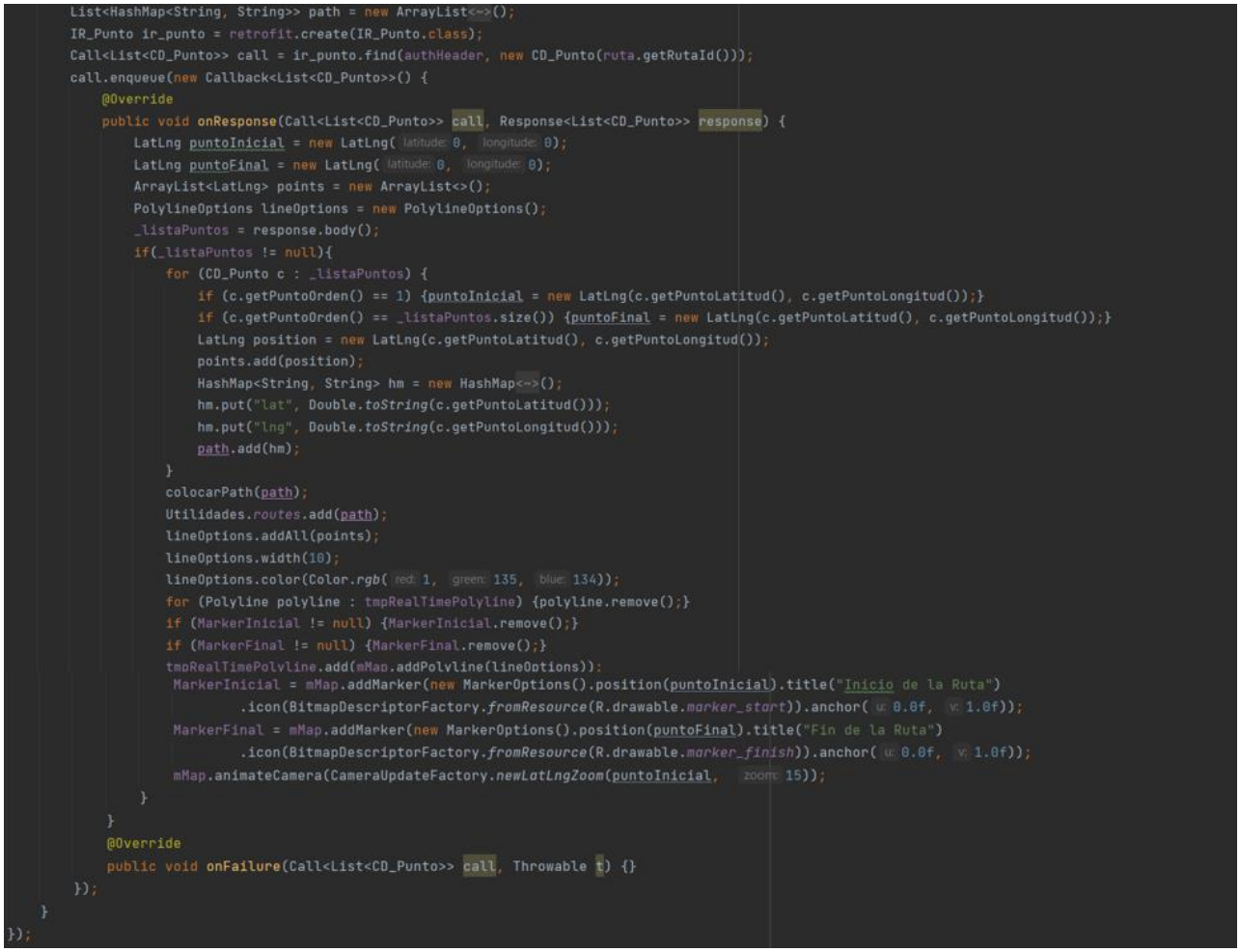
```
autoCompleteTextViewSubZona.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        nombreSubZonaSeleccionada = autoCompleteTextViewSubZona.getAdapter().getItem(position).toString();
        subzona = (CD_SubZona) parent.getItemAtPosition(position);
        IR_Ruta retrofitRuta = retrofit.create(IR_Ruta.class);
        Call<List<CD_Ruta>> call3 = retrofitRuta.findAll(authHeader);
        call3.enqueue(new Callback<List<CD_Ruta>>() {
            @Override
            public void onResponse(Call<List<CD_Ruta>> call, Response<List<CD_Ruta>> response) {
                rutaList = response.body();
                if(rutaList != null){
                    for (CD_Ruta r : rutaList) {
                        if(r.getSubzonaId() == subzona.getSubzonaId() && r.getRutaEstado() == 1){
                            rutaList2.add(r);
                        }
                    }
                    rutaArrayAdapter = new RutaArrayAdapter( mContext: SeleccionRutaActivity.this, R.layout.items_list, rutaList2);
                    autoCompleteTextViewRuta.setAdapter(rutaArrayAdapter);
                }
            }
            @Override
            public void onFailure(Call<List<CD_Ruta>> call, Throwable t) {
                Toast.makeText(getApplicationContext(),  text: "Sin Conexión a Internet", Toast.LENGTH_LONG).show();
            }
        });
    }
});
```

Ilustración 125. Código Fuente Evento Componente AutoCompleteTextViewSubZona. Fuente: Autor.

Al igual que los eventos anteriormente mencionados se creó un evento para el AutoCompleteTextView Ruta, con el objeto de que al seleccionar un ítem del mismo, cargue toda la información referente a la misma dentro de los demás TextView, como el horario, días de desecho orgánico e inorgánico y se grafique dentro del mapa la ruta de recorrido, utilizando los puntos de latitud y longitud de la base de datos.

```
autoCompleteTextViewRuta.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        ruta = (CD_Ruta) parent.getItemAtPosition(position);
        obtenerAtributosVehiculo(ruta.getVehiculoId());
        Toast.makeText(autoCompleteTextViewRuta.getContext(), text: "Ha Seleccionado la Ruta: " + ((CD_Ruta) parent.getItemAtPosition(position)).getRutaNombre(),
                Toast.LENGTH_SHORT).show();
        nombreRutaSeleccionada = autoCompleteTextViewRuta.getAdapter().getItem(position).toString();
        idRutaSeleccionada = ruta.getRutaId();
        String horarioOrganico = "";
        if(ruta.getRutaLunesOrg() == 1){
            horarioOrganico = horarioOrganico + "Lunes, ";
        }
        if(ruta.getRutaMartesOrg() == 1){
            horarioOrganico = horarioOrganico + "Martes, ";
        }
        if(ruta.getRutaMiercolesOrg() == 1){
            horarioOrganico = horarioOrganico + "Miercoles, ";
        }
        if(ruta.getRutaJuevesOrg() == 1){
            horarioOrganico = horarioOrganico + "Jueves, ";
        }
        if(ruta.getRutaViernesOrg() == 1){
            horarioOrganico = horarioOrganico + "Viernes, ";
        }
        if(ruta.getRutaSabadoOrg() == 1){
            horarioOrganico = horarioOrganico + "Sábado, ";
        }
        if(ruta.getRutaDomingoOrg() == 1){
            horarioOrganico = horarioOrganico + "Domingo, ";
        }
        String finalHorarioOrganico = horarioOrganico.substring(0, horarioOrganico.length() - 2);
        String horarioInorganico = "";
        if(ruta.getRutaLunesIno() == 1){
            horarioInorganico = horarioInorganico + "Lunes, ";
        }
        if(ruta.getRutaMartesIno() == 1){
            horarioInorganico = horarioInorganico + "Martes, ";
        }
        if(ruta.getRutaMiercolesIno() == 1){
            horarioInorganico = horarioInorganico + "Miercoles, ";
        }
        if(ruta.getRutaJuevesIno() == 1){
            horarioInorganico = horarioInorganico + "Jueves, ";
        }
        if(ruta.getRutaViernesIno() == 1){
            horarioInorganico = horarioInorganico + "Viernes, ";
        }
        if(ruta.getRutaSabadoIno() == 1){
            horarioInorganico = horarioInorganico + "Sábado, ";
        }
        if(ruta.getRutaDomingoIno() == 1){
            horarioInorganico = horarioInorganico + "Domingo, ";
        }
        String finalHorarioInorganico = horarioInorganico.substring(0, horarioInorganico.length() - 2);
        textViewHorario.setText(ruta.getRutaHorario());
        textViewDiasOrganico.setText(finalHorarioOrganico);
        textViewDiasInorganico.setText(finalHorarioInorganico);
```

Ilustración 126. Código Fuente Evento Componente AutoCompleteTextViewRuta Carga de Horarios y Días de desecho Orgánico e Inorgánico. Fuente: Autor.



```
        List<HashMap<String, String>> path = new ArrayList<>();
        IR_Punto ir_punto = retrofit.create(IR_Punto.class);
        Call<List<CD_Punto>> call = ir_punto.find(authHeader, new CD_Punto(ruta.getRutaId()));
        call.enqueue(new Callback<List<CD_Punto>>() {
            @Override
            public void onResponse(Call<List<CD_Punto>> call, Response<List<CD_Punto>> response) {
                LatLng puntoInicial = new LatLng( latitude: 0,  longitude: 0);
                LatLng puntoFinal = new LatLng( latitude: 0,  longitude: 0);
                ArrayList<LatLng> points = new ArrayList<>();
                PolylineOptions lineOptions = new PolylineOptions();
                _listaPuntos = response.body();
                if(_listaPuntos != null){
                    for (CD_Punto c : _listaPuntos) {
                        if (c.getPuntoOrden() == 1) {puntoInicial = new LatLng(c.getPuntoLatitud(), c.getPuntoLongitud());}
                        if (c.getPuntoOrden() == _listaPuntos.size()) {puntoFinal = new LatLng(c.getPuntoLatitud(), c.getPuntoLongitud());}
                        LatLng position = new LatLng(c.getPuntoLatitud(), c.getPuntoLongitud());
                        points.add(position);
                        HashMap<String, String> hm = new HashMap<>();
                        hm.put("lat", Double.toString(c.getPuntoLatitud()));
                        hm.put("lng", Double.toString(c.getPuntoLongitud()));
                        path.add(hm);
                    }
                    colocarPath(path);
                    Utilidades.routes.add(path);
                    lineOptions.addAll(points);
                    lineOptions.width(10);
                    lineOptions.color(Color.rgb( red: 1,  green: 135,  blue: 134));
                    for (Polyline polyline : tmpRealTimePolyline) {polyline.remove();}
                    if (MarkerInicial != null) {MarkerInicial.remove();}
                    if (MarkerFinal != null) {MarkerFinal.remove();}
                    tmpRealTimePolyline.add(mMap.addPolyline(lineOptions));
                     MarkerInicial = mMap.addMarker(new MarkerOptions().position(puntoInicial).title("Inicio de la Ruta")
                             .icon(BitmapDescriptorFactory.fromResource(R.drawable.marker_start)).anchor( u: 0.0f,  v: 1.0f));
                     MarkerFinal = mMap.addMarker(new MarkerOptions().position(puntoFinal).title("Fin de la Ruta")
                             .icon(BitmapDescriptorFactory.fromResource(R.drawable.marker_finish)).anchor( u: 0.0f,  v: 1.0f));
                     mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(puntoInicial,  zoom: 15));
                }
            }
            @Override
            public void onFailure(Call<List<CD_Punto>> call, Throwable t) {}
        });
    }
});
```

Ilustración 127. Código Fuente Evento Componente AutoCompleteTextViewRuta Carga de Puntos Ruta Recorrido. Fuente: Autor.

ObtencionUbicacionActivity

Esta clase se ejecuta a continuación de la anterior. Esta clase tiene el objeto de obtener la ubicación precisa del vehículo recolector, obteniendo 5 ubicaciones durante 20 segundos de espera y comparándolas para verificar si el vehículo recolector ya pasó o aun no pasa por la ubicación del usuario.

Se tiene como primera codificación dentro de onCreate(), la obtención de los valores de las variables enviadas por la clase SeleccionRutaActivity.

```
Bundle parametros = this.getIntent().getExtras();
if(parametros !=null){
    nombreZonaSeleccionada = parametros.getString( key: "nombreZonaSeleccionada");
    nombreSubZonaSeleccionada = parametros.getString( key: "nombreSubZonaSeleccionada");
    nombreRutaSeleccionada = parametros.getString( key: "nombreRutaSeleccionada");
    vehiculoIdSelected = parametros.getInt( key: "vehiculoIdSelected");
    idRutaSeleccionada = parametros.getInt( key: "idRutaSeleccionada",  defaultValue: 1);
    vehiculoRutaSelected = parametros.getString( key: "vehiculoRutaSelected");
    path = (List<HashMap<String, String>>)parametros.getSerializable( key: "pathEnvio");
    pathEnvio= new ArrayList<>(path.size());
    pathEnvio.addAll(path);
    lat = parametros.getDouble( key: "lat",  defaultValue: 1);
    lng = parametros.getDouble( key: "lng",  defaultValue: 1);
    Utilidades.routes.clear();
    Utilidades.routes.add(path);
}
```

Ilustración 128. Código Fuente Obtención de Parámetros enviados por SeleccionRutaActivity. Fuente: Autor.

Posteriormente se utiliza TimerTask, mismo que se ejecuta cada 4 segundos utilizando el mismo código del método actualizarVehiculo() de la clase GeolocalizacionActivity con algunas excepciones y codificación adicional. Dicha codificación adicional se detalla a continuación:

Dentro del método actualizarVehiculo(), luego de obtener la ubicación del vehículo y usuario dentro de la ruta, se codifica el siguiente código final de dicho método. Que permite verificar a través de contadores cuantas veces se obtiene una ubicación del vehículo que indica que ya paso por la ubicación del usuario y cuantas de que todavía aun no pasa, para que al final de dicha comprobación que va a ser repetitiva 5 veces cada 4 segundos, se verifique a través de una condición y se asigne la ubicación final del vehículo recolector, dependiendo si es la ubicación de que ya paso o aun no pasa por la ubicación del usuario.

```
if (contadorVeritifacion < 5) {
    if(UbicaciónVehiculo != null){
        for (int j = 0; j < path.size(); j++) {
            HashMap<String, String> point = path.get(j);
            double latPath = Double.parseDouble(point.get("lat"));
            double lngPath = Double.parseDouble(point.get("lng"));
            LatLng PuntoPath = new LatLng(latPath, lngPath);
            if (PuntoPath.equals(UbicaciónVehiculo)) {
                veriUbicacionVehiculoNoPaso = UbicaciónVehiculo;
                contadorNoPasoVehiculo = contadorNoPasoVehiculo + 1;
                break;
            }
            if (PuntoPath.equals(UbicaciónUsuario)) {
                veriUbicacionVehiculoPaso = UbicaciónVehiculo;
                contadorPasoVehiculo = contadorPasoVehiculo + 1;
                break;
            }
        }
    }
    contadorVeritifacion = contadorVeritifacion + 1;
}
else{
    veriUbicacionVehiculo = (contadorPasoVehiculo > contadorNoPasoVehiculo) ? veriUbicacionVehiculoPaso : veriUbicacionVehiculoNoPaso;
    t.cancel();
    nextActivity();
}
```

Ilustración 129. Código Fuente Verificación del Estado de la Ubicación del Vehículo Recolector. Fuente: Autor.

Finalmente, el método nextActivity(), que se encuentra dentro del código presentado anteriormente, permite concluir o finalizar la clase ObtencionVehiculoActivity y pasar a la siguiente clase GeolocalizacionActivity enviando los valores de las variables procesadas en dicha clase.

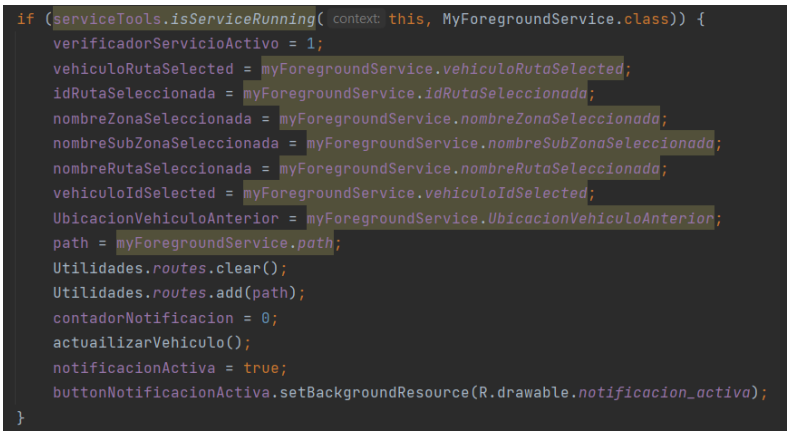
```
public void nextActivity(){
    if(vehiculoRutaSelected!=null && idRutaSeleccionada != 0 && nombreRutaSeleccionada!= null && nombreZonaSeleccionada != null && nombreSubZonaSeleccionada != null){
        Intent i = new Intent( packageContext: ObtencionUbicacionActivity.this, GeolocalizacionActivity.class);
        i.putExtra( name: "vehiculoRutaSelected", vehiculoRutaSelected);
        i.putExtra( name: "vehiculoIdSelected", vehiculoIdSelected);
        i.putExtra( name: "idRutaSeleccionada", idRutaSeleccionada);
        i.putExtra( name: "nombreZonaSeleccionada", nombreZonaSeleccionada);
        i.putExtra( name: "nombreSubZonaSeleccionada", nombreSubZonaSeleccionada);
        i.putExtra( name: "nombreRutaSeleccionada", nombreRutaSeleccionada);
        i.putExtra( name: "pathEnvio", pathEnvio);
        if(veriUbicacionVehiculo!=null){
            i.putExtra( name: "veriUbicacionVehiculoLat", veriUbicacionVehiculo.latitude);
            i.putExtra( name: "veriUbicacionVehiculoLng", veriUbicacionVehiculo.longitude);
        }
        finish();
        startActivity(i);
    }
}
```

Ilustración 130. Código Fuente Método nextActivity(). Fuente: Autor.

GeolocalizacionActivity

Esta clase se ejecuta a continuación de la clase ObtencionUbicacionActivity. Permite geolocalizar el vehículo recolector e informar el estado del mismo, la distancia y el tiempo estimado al que se encuentra con respecto al usuario, a su vez, permite activar las notificaciones en segundo plano del aplicativo.

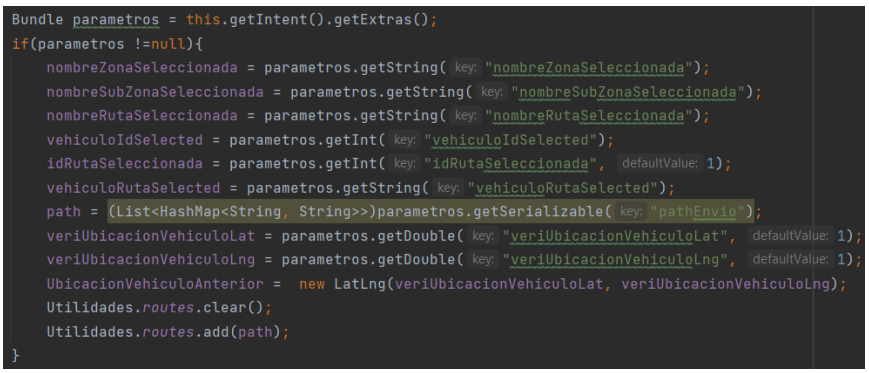
Como primer código dentro del método onCreate() se encuentra la verificación de la ejecución del servicio en segundo plano de las notificaciones, con el objeto de obtener los valores de las variables para sincronizar las actividades realizadas del servicio con la clase.



```
if (serviceTools.isServiceRunning( context: this, MyForegroundService.class)) {
    verificadorServicioActivo = 1;
    vehiculoRutaSelected = myForegroundService.vehiculoRutaSelected;
    idRutaSeleccionada = myForegroundService.idRutaSeleccionada;
    nombreZonaSeleccionada = myForegroundService.nombreZonaSeleccionada;
    nombreSubZonaSeleccionada = myForegroundService.nombreSubZonaSeleccionada;
    nombreRutaSeleccionada = myForegroundService.nombreRutaSeleccionada;
    vehiculoIdSelected = myForegroundService.vehiculoIdSelected;
    UbicacionVehiculoAnterior = myForegroundService.UbicacionVehiculoAnterior;
    path = myForegroundService.path;
    Utilidades.routes.clear();
    Utilidades.routes.add(path);
    contadorNotificacion = 0;
    actuailizarVehiculo();
    notificacionActiva = true;
    buttonNotificacionActiva.setBackgroundResource(R.drawable.notificacion_activa);
}
```

Ilustración 131. Código Fuente Obtención de los Parámetros de MyForegroundService. Fuente: Autor.

El siguiente código es la obtención de las variables enviadas de la clase ObtencionUbicacionActivity.



```
Bundle parametros = this.getIntent().getExtras();
if(parametros !=null){
    nombreZonaSeleccionada = parametros.getString( key: "nombreZonaSeleccionada");
    nombreSubZonaSeleccionada = parametros.getString( key: "nombreSubZonaSeleccionada");
    nombreRutaSeleccionada = parametros.getString( key: "nombreRutaSeleccionada");
    vehiculoIdSelected = parametros.getInt( key: "vehiculoIdSelected");
    idRutaSeleccionada = parametros.getInt( key: "idRutaSeleccionada", defaultValue: 1);
    vehiculoRutaSelected = parametros.getString( key: "vehiculoRutaSelected");
    path = (List<HashMap<String, String>>)parametros.getSerializable( key: "pathEnvio");
    veriUbicacionVehiculoLat = parametros.getDouble( key: "veriUbicacionVehiculoLat", defaultValue: 1);
    veriUbicacionVehiculoLng = parametros.getDouble( key: "veriUbicacionVehiculoLng", defaultValue: 1);
    UbicacionVehiculoAnterior =  new LatLng(veriUbicacionVehiculoLat, veriUbicacionVehiculoLng);
    Utilidades.routes.clear();
    Utilidades.routes.add(path);
}
```

Ilustración 132. Código Fuente Obtención de los Parámetros de ObtencionUbicacionActivity. Fuente: Autor.

Se codifica al evento con el envío de variables para el botón de regreso a la pantalla principal, es decir, a la clase SeleccionRutaActivity, en caso de que el usuario requiera seleccionar otra ruta para la geolocalización.

```
buttonBack.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        Intent i = new Intent( packageContext: GeolocalizacionActivity.this, SeleccionRutaActivity.class);
        i.putExtra( name: "vehiculoRutaSelected", vehiculoRutaSelected);
        i.putExtra( name: "idRutaSeleccionada", idRutaSeleccionada);
        i.putExtra( name: "nombreZonaSeleccionada", nombreZonaSeleccionada);
        i.putExtra( name: "nombreSubZonaSeleccionada", nombreSubZonaSeleccionada);
        i.putExtra( name: "nombreRutaSeleccionada", nombreRutaSeleccionada);
        i.putExtra( name: "vehiculoIdSelected", vehiculoIdSelected);
        finish();  //Kill the activity from which you will go to next activity
        startActivity(i);
    }
});
```

Ilustración 133. Código Fuente Envió de Parámetros a SeleccionRutaActivity. Fuente: Autor.

Como siguiente código fuera del método onCreate(), se tiene el método onMapReady(), donde se configura el modo oscuro y claro del aplicativo, así como también la ejecución continua de los métodos miUbicacion(), actualizarVehiculo(), y del countDownTimer(), este último con el objeto de permitir configurar un tiempo repetitivo de ejecución de 4 segundos del método onMapReady(), para la actualización de la información de geolocalización.

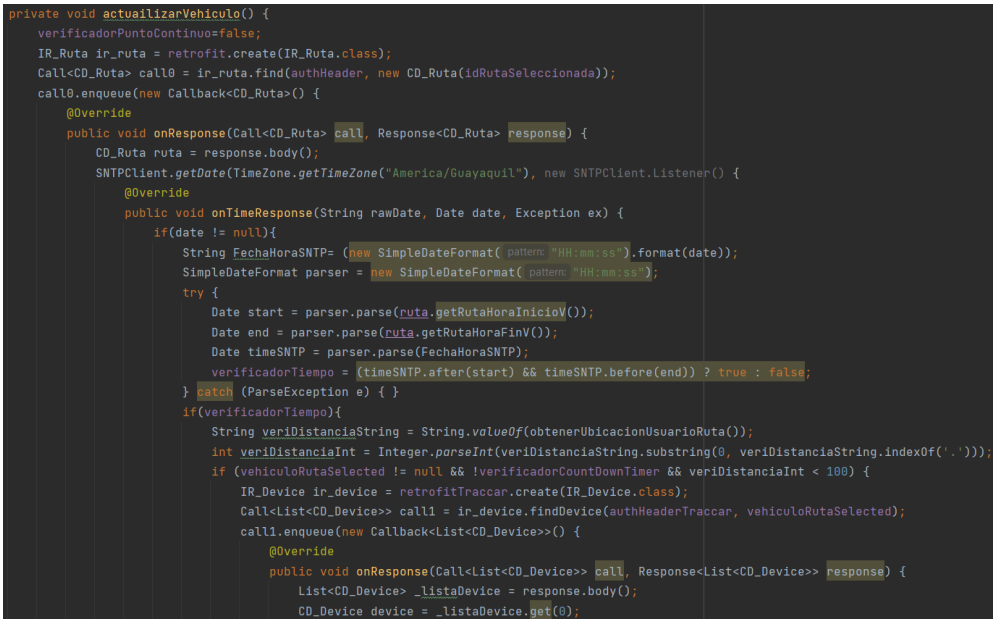
```
@Override
public void onMapReady(GoogleMap googleMap) {
    boolean success = false;
    int nightModeFlags = getResources().getConfiguration().uiMode & Configuration.UI_MODE_NIGHT_MASK;
    switch (nightModeFlags) {
        case Configuration.UI_MODE_NIGHT_YES: success = googleMap.setMapStyle(new MapStyleOptions("[ { "featureType": "all", "elementType": "geometry", "s..."));
            break;
        case Configuration.UI_MODE_NIGHT_NO: success = googleMap.setMapStyle(new MapStyleOptions("[ { "featureType": "poi", "elementType": "labels", "sty..."));
            break;
    }
    mMap = googleMap;
    miUbicacion();
    actuailizarVehiculo();
    countDownTimer(verificadorCountDownTimer);
}
```

Ilustración 134. Código Fuente Método onMapReady GeolocalizacionActivity. Fuente: Autor.

Como el código del método actualizarVehiculo() es extenso se detalla parte por parte:

1. Como primer punto se obtiene los datos de la ruta seleccionada por el usuario utilizando Retrofit.
2. Se verifica si se encuentra dentro del horario de recolección de la ruta, utilizando un servidor NTP con horario de Ecuador.

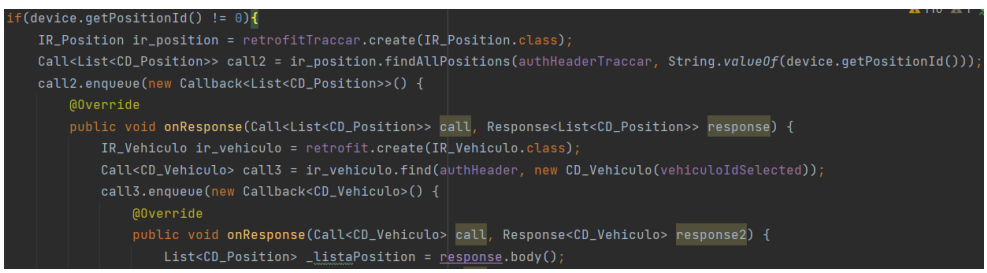
3. Se verifica si el usuario se encuentra a una distancia menor de 100 metros con respecto al punto más cercano de la ruta de recorrido.
4. Se obtiene los datos del registro de la clase Devices correspondiente al vehículo que corresponde a la ruta seleccionada, utilizando el API REST Traccar mediante Retrofit.



```
private void actuailizarVehiculo() {
    verificadorPuntoContinuo=false;
    IR_Ruta ir_ruta = retrofit.create(IR_Ruta.class);
    Call<CD_Ruta> call0 = ir_ruta.find(authHeader, new CD_Ruta(idRutaSeleccionada));
    call0.enqueue(new Callback<CD_Ruta>() {
        @Override
        public void onResponse(Call<CD_Ruta> call, Response<CD_Ruta> response) {
            CD_Ruta ruta = response.body();
            SNTPClient.getDate(TimeZone.getTimeZone("America/Guayaquil"), new SNTPClient.Listener() {
                @Override
                public void onTimeResponse(String rawDate, Date date, Exception ex) {
                    if(date != null){
                        String FechaHoraSNTP= (new SimpleDateFormat( pattern: "HH:mm:ss").format(date));
                        SimpleDateFormat parser = new SimpleDateFormat( pattern: "HH:mm:ss");
                        try {
                            Date start = parser.parse(ruta.getRutaHoraInicioV());
                            Date end = parser.parse(ruta.getRutaHoraFinV());
                            Date timeSNTP = parser.parse(FechaHoraSNTP);
                            verificadorTiempo = (timeSNTP.after(start) && timeSNTP.before(end)) ? true : false;
                        } catch (ParseException e) { }
                        if(verificadorTiempo){
                            String veriDistanciaString = String.valueOf(obtenerUbicacionUsuarioRuta());
                            int veriDistanciaInt = Integer.parseInt(veriDistanciaString.substring(0, veriDistanciaString.indexOf('.')));
                            if (vehiculoRutaSelected != null && !verificadorCountDownTimer && veriDistanciaInt < 100) {
                                IR_Device ir_device = retrofitTraccar.create(IR_Device.class);
                                Call<List<CD_Device>> call1 = ir_device.findDevice(authHeaderTraccar, vehiculoRutaSelected);
                                call1.enqueue(new Callback<List<CD_Device>>() {
                                    @Override
                                    public void onResponse(Call<List<CD_Device>> call, Response<List<CD_Device>> response) {
                                        List<CD_Device> _listaDevice = response.body();
                                        CD_Device device = _listaDevice.get(0);
```

Ilustración 135. Código Fuente Método actualizarVehiculo() I. Fuente: Autor.

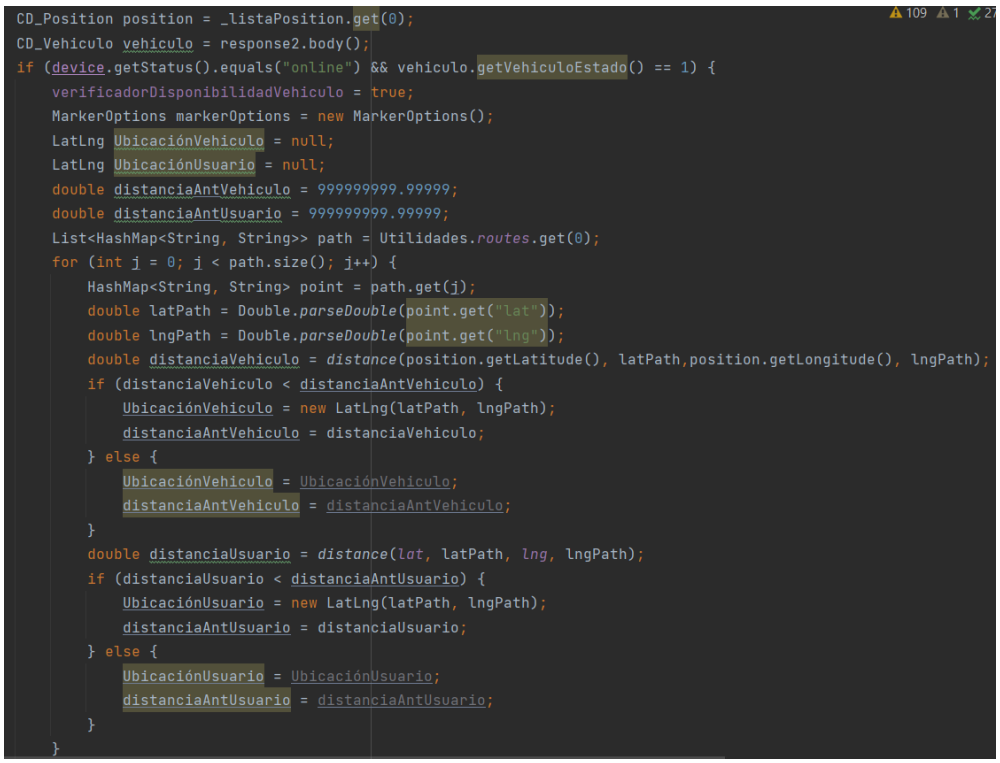
5. Se obtiene los datos de latitud y longitud de la clase Positions correspondiente al Device anteriormente obtenido. Utilizando el API REST Traccar con Retrofit.
6. Posteriormente se obtiene los datos del registro Vehículo que se encuentra dentro de la ruta seleccionada. Utilizando el API REST con Retrofit.



```
if(device.getPositionId() != 0){
    IR_Position ir_position = retrofitTraccar.create(IR_Position.class);
    Call<List<CD_Position>> call2 = ir_position.findAllPositions(authHeaderTraccar, String.valueOf(device.getPositionId()));
    call2.enqueue(new Callback<List<CD_Position>>() {
        @Override
        public void onResponse(Call<List<CD_Position>> call, Response<List<CD_Position>> response) {
            IR_Vehiculo ir_vehiculo = retrofit.create(IR_Vehiculo.class);
            Call<CD_Vehiculo> call3 = ir_vehiculo.find(authHeader, new CD_Vehiculo(vehiculoIdSelected));
            call3.enqueue(new Callback<CD_Vehiculo>() {
                @Override
                public void onResponse(Call<CD_Vehiculo> call, Response<CD_Vehiculo> response2) {
                    List<CD_Position> _listaPosition = response.body();
```

Ilustración 136. Código Fuente Método actualizarVehiculo() II. Fuente: Autor.

7. Se comprueba el estado del devices si está activo o no.
8. Se obtiene y se posiciona la ubicación del vehículo y del usuario dentro de la ruta.



```
CD_Position position = _listaPosition.get(0);
CD_Vehiculo vehiculo = response2.body();
if (device.getStatus().equals("online") && vehiculo.getVehiculoEstado() == 1) {
    verificadorDisponibilidadVehiculo = true;
    MarkerOptions markerOptions = new MarkerOptions();
    LatLng UbicaciónVehiculo = null;
    LatLng UbicaciónUsuario = null;
    double distanciaAntVehiculo = 999999999.99999;
    double distanciaAntUsuario = 999999999.99999;
    List<HashMap<String, String>> path = Utilidades.routes.get(0);
    for (int j = 0; j < path.size(); j++) {
        HashMap<String, String> point = path.get(j);
        double latPath = Double.parseDouble(point.get("lat"));
        double lngPath = Double.parseDouble(point.get("lng"));
        double distanciaVehiculo = distance(position.getLatitude(), latPath,position.getLongitude(), lngPath);
        if (distanciaVehiculo < distanciaAntVehiculo) {
            UbicaciónVehiculo = new LatLng(latPath, lngPath);
            distanciaAntVehiculo = distanciaVehiculo;
        } else {
            UbicaciónVehiculo = UbicaciónVehiculo;
            distanciaAntVehiculo = distanciaAntVehiculo;
        }
        double distanciaUsuario = distance(lat, latPath, lng, lngPath);
        if (distanciaUsuario < distanciaAntUsuario) {
            UbicaciónUsuario = new LatLng(latPath, lngPath);
            distanciaAntUsuario = distanciaUsuario;
        } else {
            UbicaciónUsuario = UbicaciónUsuario;
            distanciaAntUsuario = distanciaAntUsuario;
        }
    }
```

Ilustración 137. Código Fuente Método actualizarVehiculo() III. Fuente: Autor.

9. Se verifica que la ubicación obtenida con anterioridad del vehículo y la ubicación nueva del mismo no tengan una distancia superior a 300 m entre ellas. En caso de que se encuentre una distancia menor a la misma, que se tome en consideración dicha ubicación nueva y se asigne a la ubicación del vehículo anterior, para continuar con el proceso. Esto con la finalidad de evitar las fallas que ocasiona la precisión del dispositivo GPS y proporcione una ubicación nueva del vehículo errónea dentro de la ruta de recorrido.

```
if (UbicaciónVehiculo != null) {
    int sumDistancia = 0;
    veriDentroRutaANP = false;
    double latNueva = UbicaciónVehiculo.latitude;
    double lngNueva = UbicaciónVehiculo.longitude;
    for (int j = 0; j < path.size(); j++) {
        HashMap<String, String> point = path.get(j);
        double latPath = Double.parseDouble(point.get("lat"));
        double lngPath = Double.parseDouble(point.get("lng"));
        LatLng PuntoPath = new LatLng(latPath, lngPath);
        if (PuntoPath.equals(UbicacionVehiculoAnterior) || veriDentroRutaANP) {
            veriDentroRutaANP = true;
            String distanciaString = String.valueOf(distance(latNueva, latPath, lngNueva, lngPath));
            int distanciaInt = Integer.parseInt(distanciaString.substring(0, distanciaString.indexOf('.')));
            sumDistancia = sumDistancia + distanciaInt;
            latNueva = latPath;
            lngNueva = lngPath;
        }
        if (PuntoPath.equals(UbicaciónVehiculo)) {
            if (sumDistancia > 0) {verificadorPuntoContinuo = true;}
            break;
        }
    }
    if (sumDistancia <= 300 && verificadorPuntoContinuo) {UbicacionVehiculoAnterior = UbicaciónVehiculo;}
}
```

Ilustración 138. Código Fuente Método actualizarVehiculo() IV. Fuente: Autor.

10. Una vez validado lo anterior, se procede a obtener la distancia existente entre la ubicación del vehículo recolector y del usuario.

```
if (UbicacionVehiculoAnterior != null) {
    int sumDistancia = 0;
    veriDentroRutaANP = false;
    double latAnterior = UbicacionVehiculoAnterior.latitude;
    double lngAnterior = UbicacionVehiculoAnterior.longitude;
    for (int j = 0; j < path.size(); j++) {
        HashMap<String, String> point = path.get(j);
        double latPath = Double.parseDouble(point.get("lat"));
        double lngPath = Double.parseDouble(point.get("lng"));
        LatLng PuntoPath = new LatLng(latPath, lngPath);
        if (PuntoPath.equals(UbicacionVehiculoAnterior) || veriDentroRutaANP) {
            veriDentroRutaANP = true;
            String distanciaString = String.valueOf(distance(latAnterior, latPath, lngAnterior, lngPath));
            int distanciaInt = Integer.parseInt(distanciaString.substring(0, distanciaString.indexOf('.')));
            sumDistancia = sumDistancia + distanciaInt;
            latAnterior = latPath;
            lngAnterior = lngPath;
        }
        if (PuntoPath.equals(UbicaciónUsuario)) {
            break;
        }
    }
    distancia = sumDistancia;
}
```

Ilustración 139. Código Fuente Método actualizarVehiculo() V. Fuente: Autor.

11. Una vez obtenida la distancia entre el vehículo recolector y el usuario, se procede a realizar el cálculo del tiempo estimado, basándose en un valor estático de velocidad promedio del vehículo recolector.

```
double horas = 0;
double minutos = 0;
double segundos = 0;
double velocidad = 2.77778;
double tiempo = distancia / velocidad;
String tiempoString = String.valueOf(tiempo);
tiempo = (esDecimal(tiempo)) ? Double.parseDouble(tiempoString.substring(0, tiempoString.indexOf('.'))) : Double.parseDouble(tiempoString);
if (tiempo != 0) {
    horas = tiempo / 3600;
    String horaString = String.valueOf(horas);
    horas = (esDecimal(horas)) ? Double.parseDouble(horaString.substring(0, horaString.indexOf('.'))) : Double.parseDouble(horaString);
    minutos = (tiempo % 3600) / 60;
    String minutosString = String.valueOf(minutos);
    minutos = (esDecimal(minutos)) ? Double.parseDouble(minutosString.substring(0, minutosString.indexOf('.'))) : Double.parseDouble(minutosString);
    segundos = (tiempo % 3600) % 60;
    String segundosString = String.valueOf(segundos);
    segundos = (esDecimal(segundos)) ? Double.parseDouble(segundosString.substring(0, segundosString.indexOf('.'))) : Double.parseDouble(segundosString);
}
```

Ilustración 140. Código Fuente Método actualizarVehiculo() VI. Fuente: Autor.

12. Se procede a colocar el estado, distancia y tiempo en los TextView, así como también activar la visualización del vehículo recolector en el mapa, todo esto, verificando si la distancia entre el vehículo y el usuario es menor a 500 m.

```
if (veriDentroRutaANP && distancia <= 500.00) {
    markerOptions.position(new LatLng(UbicaciónVehiculo.latitude, UbicaciónVehiculo.longitude)).title("Vehículo Recolector")
            .icon(BitmapDescriptorFactory.fromResource(R.drawable.camion_basura)).anchor( u: 0.0f,  v: 1.0f);
    for (Marker marker : tmpRealTimeMarkers) {marker.remove();}
    tmpRealTimeMarkers.add(mMap.addMarker(markerOptions));
} else {
    if (tmpRealTimeMarkers.size() > 0) {for (Marker marker : tmpRealTimeMarkers) {marker.remove();}}
}
if (veriDentroRutaANP) {
    textviewEstadoVehiculo.setText("El vehículo recolector aún no pasa por su ubicación.");
    textviewDistancia.setText(distancia + "m.");
    textviewTiempo.setText((int) horas + ".h " + (int) minutos + "min " + (int) segundos + "s.");
}
else {
    textviewEstadoVehiculo.setText("El vehículo recolector ya paso por su ubicación.");
    limpiarPantalla();
}
```

Ilustración 141. Código Fuente Método actualizarVehiculo() VII. Fuente: Autor.

13. Por último, a través de los ELSE de las condiciones IF en un inicio se coloca el estado respectivo como los siguientes:

```
else {
    verificadorDisponibilidadVehiculo = false;
    textviewEstadoVehiculo.setText("El vehículo recolector no se encuentra activo.");
```

Ilustración 142. Código Fuente Método actualizarVehiculo() ELSE Vehículo Recolector Inactivo. Fuente: Autor.

```
else {
    textviewEstadoVehiculo.setText("Actualmente, usted no se encuentra dentro de está ruta");
    limpiarPantalla();
    veriDentroRutaANP = false;
    verificadorFueraRuta = true;
}
```

Ilustración 143. Código Fuente Método actualizarVehiculo() ELSE Fuera de la Ruta. Fuente: Autor.

```
else{
    textviewEstadoVehiculo.setText("Actualmente, usted no se encuentra en el horario de recolección de esta ruta.");
```

Ilustración 144. Código Fuente Método actualizarVehiculo() ELSE Fuera del Horario de Recolección. Fuente: Autor.

Para la ejecución de las notificaciones en segundo plano del aplicativo, se utiliza el método descrito en la ilustración 150 (notificacionActiva) para dar inicio o finalizar las notificaciones mediante un evento del botón de notificaciones.

1. Primero se crea un Intent con el objetivo de poder enviar valores de las variables especificadas a la clase que se ejecutara en segundo plano MyForegroundService, todo esto verificando si el usuario se encuentra dentro de la ruta seleccionada y si el vehículo recolector se encuentra activo.

```
public void notificacionActiva(View view) {
    if (view.getId() == R.id.ActivarNotificaciones) {
        Intent serviceIntent = new Intent( packageContext: GeolocalizacionActivity.this, MyForegroundService.class);
        if (verificadorDisponibilidadVehiculo) {
            if(veriDentroRutaANP){
                serviceIntent.putExtra( name: "nombreZonaSeleccionada", nombreZonaSeleccionada);
                serviceIntent.putExtra( name: "nombreSubZonaSeleccionada", nombreSubZonaSeleccionada);
                serviceIntent.putExtra( name: "nombreRutaSeleccionada", nombreRutaSeleccionada);
                serviceIntent.putExtra( name: "vehiculoIdSelected", vehiculoIdSelected);
                serviceIntent.putExtra( name: "idRutaSeleccionada", idRutaSeleccionada);
                serviceIntent.putExtra( name: "vehiculoRutaSelected", vehiculoRutaSelected);
                serviceIntent.putExtra( name: "UbicacionVehiculoAnteriorLatitud", UbicacionVehiculoAnterior.latitude);
                serviceIntent.putExtra( name: "UbicacionVehiculoAnteriorLongitud", UbicacionVehiculoAnterior.longitude);
                pathEnvio = new ArrayList<>(path.size());
                pathEnvio.addAll(path);
                serviceIntent.putExtra( name: "pathEnvio", pathEnvio);
                serviceIntent.putExtra( name: "lat", lat);
                serviceIntent.putExtra( name: "lng", lng);
                if (notificacionActiva || notificacionActivaNougat) {
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {stopService(serviceIntent);}
                    else {notificacionActivaNougat = false;}
                    notificacionActiva = false;
                    buttonNotificacionActiva.setBackgroundResource(R.drawable.notificacion_inactiva);
                } else {
                    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {startService(serviceIntent);}
                    else {notificacionActivaNougat = true;}
                    notificacionActiva = true;
                    buttonNotificacionActiva.setBackgroundResource(R.drawable.notificacion_activa);
                }
            }
```

Ilustración 145. Código Fuente Método notificacionActiva(). Fuente: Autor.

2. Por último, se codifica el proceso que ocurre si no se cumple las condiciones IF anteriormente presentados. Se verifica si vehículo recolector ya paso por la ubicación del usuario o no para proporcionar su respectivo mensaje informativo.

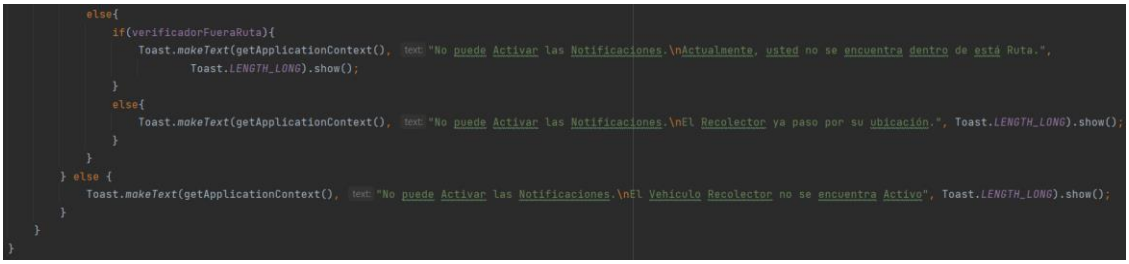


Ilustración 146. Código Fuente Método notificacionActiva() ELSE. Fuente: Autor.

MyForegroundService

Esta clase tiene el objeto de ejecutar el código del método actualizarVehiculo() de la clase GeolocalizacionActivity en segundo plano, con la acción adicional de crear las notificaciones con respecto a cada uno de los estados del vehículo recolector.

En primer lugar, esta clase es una extensión de Service, misma que proporciona el método onStartCommand(), en donde se coloca todo el código de ejecución anteriormente mencionado. Como primer punto se obtiene los valores de las variables enviadas de GeolocalizacionActivity al iniciar el servicio MyforegroundSerivce.

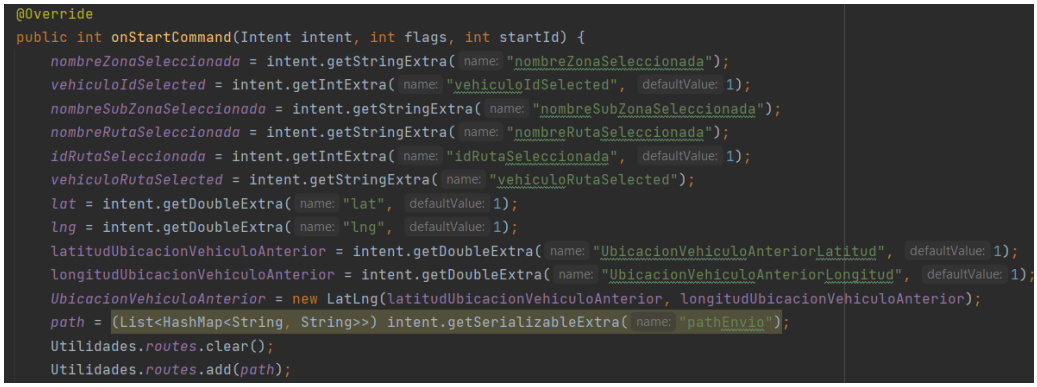
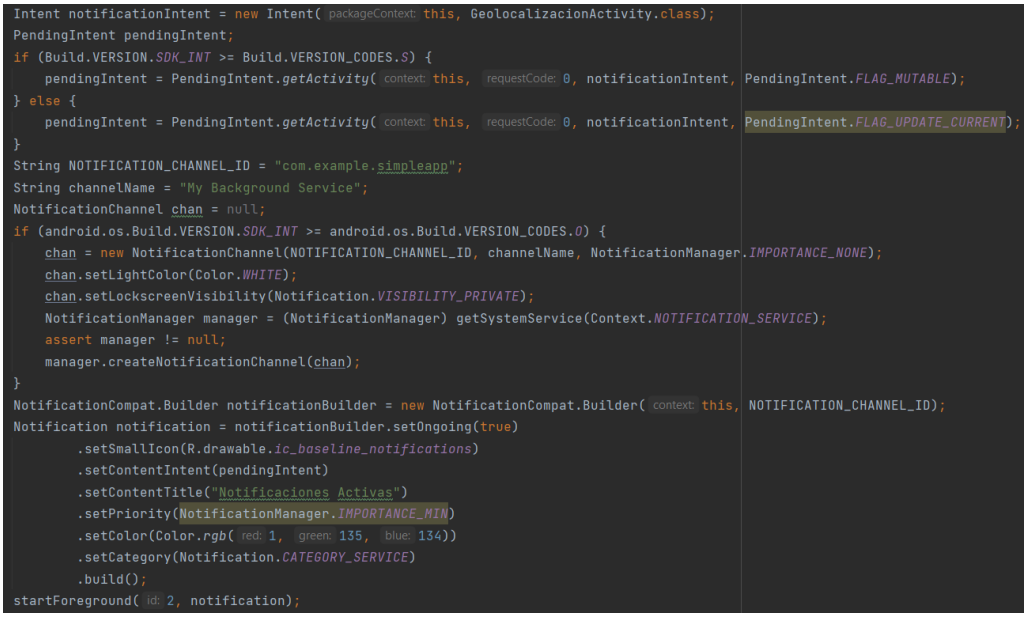


Ilustración 147. Código Fuente Obtención de los Parámetros Enviados de GeolocalizacionActivity. Fuente: Autor.

Posteriormente se procede a crear una notificación para indicar que el servicio de notificaciones en segundo plano se encuentra activo.



```
Intent notificationIntent = new Intent( packageContext: this, GeolocalizacionActivity.class);
PendingIntent pendingIntent;
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
    pendingIntent = PendingIntent.getActivity( context: this,  requestCode: 0, notificationIntent, PendingIntent.FLAG_MUTABLE);
} else {
    pendingIntent = PendingIntent.getActivity( context: this,  requestCode: 0, notificationIntent, PendingIntent.FLAG_UPDATE_CURRENT);
}
String NOTIFICATION_CHANNEL_ID = "com.example.simpleapp";
String channelName = "My Background Service";
NotificationChannel chan = null;
if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {
    chan = new NotificationChannel(NOTIFICATION_CHANNEL_ID, channelName, NotificationManager.IMPORTANCE_NONE);
    chan.setLightColor(Color.WHITE);
    chan.setLockscreenVisibility(Notification.VISIBILITY_PRIVATE);
    NotificationManager manager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
    assert manager != null;
    manager.createNotificationChannel(chan);
}
NotificationCompat.Builder notificationBuilder = new NotificationCompat.Builder( context: this, NOTIFICATION_CHANNEL_ID);
Notification notification = notificationBuilder.setOngoing(true)
        .setSmallIcon(R.drawable.ic_baseline_notifications)
        .setContentIntent(pendingIntent)
        .setContentTitle("Notificaciones Activas")
        .setPriority(NotificationManager.IMPORTANCE_MIN)
        .setColor(Color.rgb( red: 1,  green: 135,  blue: 134))
        .setCategory(Notification.CATEGORY_SERVICE)
        .build();
startForeground( id: 2, notification);
```

Ilustración 148. Código Fuente Creación de la Notificación de Ejecución de Notificaciones en Segundo Plano. Fuente: Autor.

Luego de ello se crea un Timer Task para que ejecute el código de actualizaVehiculo() de la clase GeolocalizacionActivity de forma repetitiva con una frecuencia de 4 segundos.

El Timer Task adiciona las líneas de código que se pueden visualizar en la ilustración 154 para crear la notificación, obteniendo la distancia entre el vehículo recolector y el usuario.

A través de IF y contadores se logra validar la ejecución o no de la notificación, permitiendo su ejecución solamente cuando la distancia entre el vehículo recolector y el usuario es menor a 500 m, y cuando el contadorNotificacion tenga un valor 0 o 7. Este último se notifica cada 21 segundos aproximadamente, debido al uso de Timer Task que ejecuta el código cada 4 segundos y el contadorNotificacion en cada ejecución suma su valor para que cuando llegue al número 7 habrá pasado 21 segundos aproximadamente y ejecute la notificación de manera repetitiva durante el tiempo mencionado hasta que el vehículo

recolector pase ya por la ubicación del usuario y deje su notificación correspondiente, finalizando el servicio.

```
if (veriDentroRutaANP) {
    if (distancia > 500.00 || !veriDentroRutaANP) {
        contadorNotificacion = 0;
    }
    if ((veriDentroRutaANP && distancia <= 500.00) && (contadorNotificacion == 0 || contadorNotificacion == 7)) {
        contadorNotificacion = 0;
        createNotification(distancia);
    }
    if ((veriDentroRutaANP && distancia <= 500.00)) {
        contadorNotificacion = contadorNotificacion + 1;
    }
} else {
    verificadorPasoRecolector = 1;
    createNotification(distancia);
    stopService();
}
```

Ilustración 149. Código Fuente Condiciones para la Ejecución del Método createNotification(). Fuente: Autor.

La siguiente ilustración permite visualizar el código para generar su notificación especifica en el método createNotification(), basándose en la numeración de la variable verificadorPasoRecolector.

```
                                        } else {
                                            verificadorPasoRecolector = 2;
                                            createNotification(sumDistanciaAnt);
                                            stopService();
                                        }
                                    }
                                    @Override
                                    public void onFailure(Call<CD_Vehiculo> call, Throwable t) {}
                                });
                            }
                            @Override
                            public void onFailure(Call<List<CD_Position>> call, Throwable t) {}
                        });
                    }
                    else {
                        verificadorPasoRecolector = 2;
                        createNotification(sumDistanciaAnt);
                        stopService();
                    }
                }
                @Override
                public void onFailure(Call<List<CD_Device>> call, Throwable t) {}
            });
        }
    } else {
        verificadorPasoRecolector = 3;
        createNotification(sumDistanciaAnt);
        stopService();
    }
```

Ilustración 150. Código Fuente ELSE de los IF del Método actualizarVehiculo() en MyForegroundService. Fuente: Autor.

Por último se da a conocer el método createNotificacion(). El mismo que, como su nombre lo indica permite crear una notificación en base a la variable global verificadorPasoRecolector y la distancia existente entre el vehículo recolector y usuario.

Se crea los respectivos datos de personalización de la notificación y de la información que presentara la misma en base a cada condición utilizando la variable verificadorPasoRecolector.

```
public void createNotification(int Distancia) {
    Intent notificationIntent = new Intent( packageContext: this, GeolocalizacionActivity.class);
    PendingIntent pendingIntent;
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.S) {
        pendingIntent = PendingIntent.getActivity( context: this,  requestCode: 0, notificationIntent, PendingIntent.FLAG_MUTABLE);
    } else {
        pendingIntent = PendingIntent.getActivity( context: this,  requestCode: 0, notificationIntent, PendingIntent.FLAG_UPDATE_CURRENT);
    }
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = "Noticación";
        NotificationChannel notificationChannel = new NotificationChannel(CHANNEL_ID, name, NotificationManager.IMPORTANCE_HIGH);
        NotificationManager notificationManager = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        notificationManager.createNotificationChannel(notificationChannel);
    }
    NotificationCompat.Builder builder = new NotificationCompat.Builder(getApplicationContext(), CHANNEL_ID);
    builder.setSmallIcon(R.drawable.ic_garbage_truck);
    builder.setContentTitle("Notificación EMMAIPC-EP");
    if (verificadorPasoRecolector == 1) {builder.setContentText("El vehículo recolector ya paso por su ubicación");
    } else if (verificadorPasoRecolector == 2) {builder.setContentText("El vehículo recolector se encuentra inactivo");
    } else if (verificadorPasoRecolector == 3) {builder.setContentText("El horario de recolección de la ruta ha terminado");
    } else {builder.setContentText("El vehículo recolector esta a: " + Distancia + " m, de su ubicación");}
    if (verificadorPasoRecolector == 0) {builder.setContentIntent(pendingIntent);}
    builder.setColor(Color.rgb( red: 1,  green: 135,  blue: 134));
    builder.setPriority(NotificationCompat.PRIORITY_HIGH);
    builder.setLights(Color.WHITE,  onMs: 1000,  offMs: 1000);
    builder.setVibrate(new long[]{1000, 1000, 1000, 1000, 1000});
    builder.setDefaults(Notification.DEFAULT_SOUND);
    builder.setVisibility(NotificationCompat.VISIBILITY_PUBLIC);
    NotificationManagerCompat notificationManagerCompat = NotificationManagerCompat.from(getApplicationContext());
    notificationManagerCompat.notify(NOTIFICACION_ID, builder.build());
}
```

Ilustración 151. Código Fuente Método createNotification(). Fuente: Autor.

Programación Sistema de Información de Escritorio para la Gestión del Aplicativo Móvil

El sistema de información de escritorio fue desarrollado en lenguaje de programación Java a través de NetBeans IDE 8.2.

Librerías

- **Java API for KML:** Esta librería fue utilizada con el objeto de descomponer los archivos de formato .kml ingresados desde la interfaz del sistema y poder

almacenarlos en la base de datos. Dichos archivos contienen el conjunto de coordenadas o puntos de las rutas de recorrido.

- **Retrofit:** Esta librería fue utilizada con el objeto de realizar peticiones al servidor. Los métodos que mayormente se utilizaron de esta librería dentro del desarrollo fueron los métodos GET y POST, con ellos se logró realizar tanto las acciones de Crear, Actualizar, Eliminar y Listar los registros de cada una de las entidades.
- **GSON:** Se consideró el uso de esta librería con el objeto de manipular datos de tipo JSON, los mismos que se obtienen y se envían a través de los métodos HTTP hacia el servidor para su posterior almacenamiento.
- **OkHttp:** Se utiliza esta librería con el objeto de facilitar la realización de las diferentes operaciones HTTP.
- **Absolute Layout:** Se utiliza esta librería con el objeto de estructurar y diseñar los componentes en las interfaces de las diferentes pantallas del sistema de una manera fácil y eficiente.

Estructuración del Proyecto y Programación

Paquete Librerías

Este paquete contiene todas las librerías anteriormente mencionadas que se encuentran añadidas al proyecto para su óptimo funcionamiento.

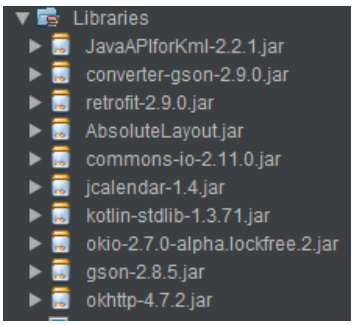


Ilustración 152. Paquete de Librerías Utilizadas en el Proyecto. Fuente: Autor.

Capa de Datos

Se crea este paquete con el objeto de almacenar todas las clases referentes a las entidades de la base de datos EMMAIPC-EP. Dichas clases contienen sus atributos, métodos constructores y métodos getters and setters.

Ilustración 153. Paquete Capa de Datos Sistema de Información. Fuente: Autor.

Capa de Datos Traccar

Se crea este paquete con la finalidad de almacenar las clases referentes a las entidades de Traccar, las mismas, que tienen utilidad dentro del proyecto, específicamente Devices, ya que permite sincronizar el método CRUD de la Pantalla Vehículo. Estas clases contienen así mismo, sus atributos, métodos constructores y métodos getters and setters.

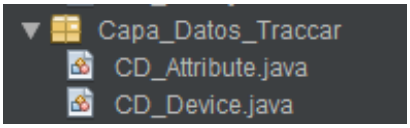


Ilustración 154. Paquete Capa de Datos Traccar Sistema de Información. Fuente: Autor.

Configuración API REST

Los paquetes de configuración se crean con el objeto de modificar ciertos parámetros indispensables para el funcionamiento del sistema de información. Entre ellos se tiene el

paquete de configuraciones del API REST, donde se creaz una clase que contiene las configuraciones necesarias al igual que el aplicativo móvil.

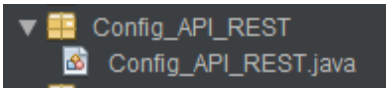


Ilustración 155. Paquete Config_API_REST Sistema de Información. Fuente: Autor.

Dichas configuraciones son el username, password y URL de acceso al API_REST de la base de datos EMMAIPC-EP y al API REST Traccar.

```
public class Config_API_REST {
    public static String username = "admin";
    public static String password = "admin";
    public static String authString = username + ":" + password;
    public static String authStringEnc = new BASE64Encoder().encode(authString.getBytes());
    public static String authHeader = "Basic " + authStringEnc;
    public static Retrofit retrofit = new Retrofit.Builder().baseUrl("http://147.182.219.86:80").addConverterFactory(GsonConverterFactory.create()).build();

    //Traccar
    public static String usernameTraccar = "dremko.cs@gmail.com";
    public static String passwordTraccar = "RSKdremko2017";
    public static String authStringTraccar = usernameTraccar + ":" + passwordTraccar;
    public static String authStringEncTraccar = new BASE64Encoder().encode(authStringTraccar.getBytes());
    public static String authHeaderTraccar = "Basic " + authStringEncTraccar;
    public static Retrofit retrofitTraccar = new Retrofit.Builder().baseUrl("http://147.182.219.86:8082").addConverterFactory(GsonConverterFactory.create()).build();

}
```

Ilustración 156. Código Fuente Config_API_REST Sistema de Información. Fuente: Autor.

Interfaces Retrofit

Se crea este paquete con el objeto de almacenar interfaces de cada una de las entidades de la Capa de Datos. Las mismas que contienen cada uno de los métodos HTTP para el CRUD con el API REST de la base de datos EMMAIPC-EP.

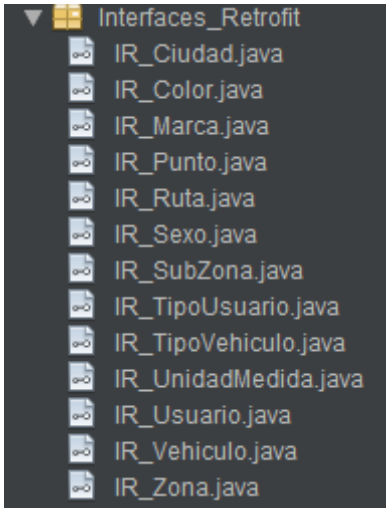


Ilustración 157. Paquete Interfaces Retrofit Sistema de Información. Fuente: Autor.

```java
public interface IR_Ciudad {

    @POST("/REST_API/api/ciudad/create.php")
    Call<CD_Ciudad> create(@Header("Authorization") String authHeader, @Body CD_Ciudad ciudad);

    @POST("/REST_API/api/ciudad/update.php")
    Call<CD_Ciudad> edit(@Header("Authorization") String authHeader, @Body CD_Ciudad ciudad);

    @POST("/REST_API/api/ciudad/delete.php")
    Call<CD_Ciudad> remove(@Header("Authorization") String authHeader, @Body CD_Ciudad ciudad);

    @POST("/REST_API/api/ciudad/read_single.php")
    Call<CD_Ciudad> find(@Header("Authorization") String authHeader, @Body CD_Ciudad ciudad);

    @GET("/REST_API/api/ciudad/read.php")
    Call<List<CD_Ciudad>> findAll(@Header("Authorization") String authHeader);
}
```

Ilustración 158. Métodos HTTP Interfaz Retrofit Ciudad Sistema de Información. Fuente: Autor.

Interfaces Retrofit Traccar

Se crea este paquete para almacenar la interfaz IRT_Devices haciendo referencia a la clase Devices de la Capa de Datos Traccar, la misma que contienen cada uno de los métodos HTTP para interactuar con el API REST Traccar.

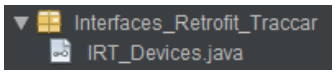


Ilustración 159. Paquete Interfaces Retrofit Traccar Sistema de Información. Fuente: Autor.

```java
public interface IRT_Devices {
    @POST("/api/devices/")
    Call<CD_Device> createDevice(@Header("Authorization") String authHeader, @Body CD_Device devices);

    @PUT("/api/devices/{id}")
    Call<CD_Device> editDevice(@Header("Authorization") String authHeader, @Path("id") String id, @Body CD_Device devices);

    @DELETE("/api/devices/{id}")
    Call<CD_Device> removeDevice(@Header("Authorization") String authHeader, @Path("id") String id);

    @GET("/api/devices")
    Call<List<CD_Device>> findAllDevices(@Header("Authorization") String authHeader);

    @GET("/api/devices")
    Call<List<CD_Device>> findDevice(@Header("Authorization") String authHeader, @Query("uniqueId") String id);
}
```

Ilustración 160. Métodos HTTP Interfaz Retrofit Traccar Devices Sistema de Información. Fuente: Autor.

Capa de Negocios

Se crea este paquete para almacenar clases Java cada una en referencia o relación con las clases de la Capa de Datos. Cada una de las clases contienen sus respectivos métodos con respecto a la entidad, como: crear, actualizar, eliminar, obtener, obtener todo y métodos demás necesarios para su funcionamiento y acciones específicas de cada una.

Ilustración 161. Paquete Capa de Negocios Sistema de Información. Fuente: Autor.

Se procede a mostrar el método CRUD de la clase Vehículo, que contiene algunos métodos CRUD importantes debido a que el efecto de la ejecución de dichos métodos también modifica los datos en el servidor Traccar.

Primero se empieza con el método crear Vehículo, donde a más de crearlo utilizando Retrofit para la API REST de la EMMAIPC-EP, también crea un Device mediante la ejecución del API REST Traccar utilizando Retrofit.

```
public void create(CD_Vehiculo vehiculo){
    IR_Vehiculo r_Vehiculo = retrofit.create(IR_Vehiculo.class);
    Call<CD_Vehiculo> call = r_Vehiculo.create(authHeader, vehiculo);
    call.enqueue(new Callback<CD_Vehiculo>() {
        @Override
        public void onResponse(Call<CD_Vehiculo> call, Response<CD_Vehiculo> response) {
        }
        @Override
        public void onFailure(Call<CD_Vehiculo> call, Throwable t) {
        }
    });
    IRT_Devices irtd = retrofitTraccar.create(IRT_Devices.class);
    CD_Device device = new CD_Device(vehiculo.getDispositivoNombre(), vehiculo.getDispositivoCodigo());
    Call<CD_Device> call2 = irtd.createDevice(authHeaderTraccar, device);
    call2.enqueue(new Callback<CD_Device>() {
        @Override
        public void onResponse(Call<CD_Device> call, Response<CD_Device> response) {
        }
        @Override
        public void onFailure(Call<CD_Device> call, Throwable t) {
        }
    });
}
```

Ilustración 162. Código Fuente Método create() Entidad Vehículo. Fuente: Autor.

El método editar también, se sincroniza con los datos del servidor Traccar mediante su API REST utilizando Retrofit.

```
public void edit(CD_Vehiculo vehiculo, String codigoDispositivo){
    IR_Vehiculo r_Vehiculo = retrofit.create(IR_Vehiculo.class);
    Call<CD_Vehiculo> call = r_Vehiculo.edit(authHeader, vehiculo);
    call.enqueue(new Callback<CD_Vehiculo>() {
        @Override
        public void onResponse(Call<CD_Vehiculo> call, Response<CD_Vehiculo> response) {
        }
        @Override
        public void onFailure(Call<CD_Vehiculo> call, Throwable t) {
        }
    });
    IRT_Devices irtd = retrofitTraccar.create(IRT_Devices.class);
    Call<List<CD_Device>> call2 = irtd.findDevice(authHeaderTraccar, codigoDispositivo);
    call2.enqueue(new Callback<List<CD_Device>>() {
        @Override
        public void onResponse(Call<List<CD_Device>> call, Response<List<CD_Device>> response) {
            List<CD_Device> device = response.body();
            if(device!=null){
                IRT_Devices irtd = retrofitTraccar.create(IRT_Devices.class);
                CD_Device device2 = new CD_Device(device.get(0).getId(), vehiculo.getDispositivoNombre(), vehiculo.getDispositivoCodigo());
                Call<CD_Device> call3 = irtd.editDevice(authHeaderTraccar, device.get(0).getId(), device2);
                call3.enqueue(new Callback<CD_Device>() {
                    @Override
                    public void onResponse(Call<CD_Device> call, Response<CD_Device> response) {
                    }
                    @Override
                    public void onFailure(Call<CD_Device> call, Throwable t) {
                    }
                });
            }
        }
        @Override
        public void onFailure(Call<List<CD_Device>> call, Throwable t) {
        }
    });

}
```

Ilustración 163. Código Fuente Método edit() Entidad Vehículo. Fuente: Autor.

El método eliminar igualmente se encuentra sincronizado con el servidor de Traccar para ejecutar la acción de eliminar devices utilizando la API REST Traccar mediante Retrofit.

```
public void remove(CD_Vehiculo vehiculo){
    IR_Vehiculo r_Vehiculo = retrofit.create(IR_Vehiculo.class);
    Call<CD_Vehiculo> call = r_Vehiculo.remove(authHeader, vehiculo);
    call.enqueue(new Callback<CD_Vehiculo>() {
        @Override
        public void onResponse(Call<CD_Vehiculo> call, Response<CD_Vehiculo> response) {
        }
        @Override
        public void onFailure(Call<CD_Vehiculo> call, Throwable t) {
        }
    });
    IRT_Devices irtd = retrofitTraccar.create(IRT_Devices.class);
    Call<List<CD_Device>> call2 = irtd.findDevice(authHeaderTraccar, vehiculo.getDispositivoCodigo());
    call2.enqueue(new Callback<List<CD_Device>>() {
        @Override
        public void onResponse(Call<List<CD_Device>> call, Response<List<CD_Device>> response) {
            List<CD_Device> device = response.body();
            if(device!=null){
                IRT_Devices irtd = retrofitTraccar.create(IRT_Devices.class);
                Call<CD_Device> call3 = irtd.removeDevice(authHeaderTraccar, device.get(0).getId());
                call3.enqueue(new Callback<CD_Device>() {
                    @Override
                    public void onResponse(Call<CD_Device> call, Response<CD_Device> response) {
                    }
                    @Override
                    public void onFailure(Call<CD_Device> call, Throwable t) {
                    }
                });
            }
        }
        @Override
        public void onFailure(Call<List<CD_Device>> call, Throwable t) {
        }
    });
}
```

Ilustración 164. Código Fuente Método remove() Entidad Vehículo. Fuente: Autor.

El método table (), no se encuentra vinculado con el API REST de Traccar, es por ello que solo hace uso del API REST de la base de datos EMMAIPC-EP.

```
public DefaultTableModel table(JTable table) {
    DefaultTableModel mdl = new DefaultTableModel();
    mdl.addColumn("Id");
    mdl.addColumn("Placa");
    mdl.addColumn("Marca");
    mdl.addColumn("Tipo Vehiculo");
    mdl.addColumn("Color");
    mdl.addColumn("Capacidad");
    mdl.addColumn("Unidad Medida");
    mdl.addColumn("Zona");
    mdl.addColumn("Código Dispositivo");
    mdl.addColumn("Nombre Dispositivo");
    mdl.addColumn("Estado");
    IR_Vehiculo r_Vehiculo = retrofit.create(IR_Vehiculo.class);
    Call<List<CD_Vehiculo>> call = r_Vehiculo.findAll(authHeader);
    call.enqueue(new Callback<List<CD_Vehiculo>>() {
        @Override
        public void onResponse(Call<List<CD_Vehiculo>> call, Response<List<CD_Vehiculo>> response) {
            List<CD_Vehiculo> _lista = response.body();
            for (CD_Vehiculo i: _lista){
                if(userlogin.getTipousuarioNombre().trim().toLowerCase().equals((PrivilegioAlto).trim().toLowerCase())){
                    String Estado = (i.getVehiculoEstado()==1) ? "Activo" : "Inactivo";
                    Object data []={i.getVehiculoId(), i.getVehiculoPlaca(), i.getMarcaNombre(), i.getTipovehiculoNombre(), i.getColorNombre(), i.getVehiculoCapacidad(),
                        i.getUnidadmedidaNombre(),
                        i.getZonaNombre(), i.getDispositivoCodigo(), i.getDispositivoNombre(), Estado};
                    mdl.addRow(data);
                }
                else{
                    if(i.getZonaId() == userlogin.getZonaId()){
                        String Estado = (i.getVehiculoEstado()==1) ? "Activo" : "Inactivo";
                    Object data []={i.getVehiculoId(), i.getVehiculoPlaca(), i.getMarcaNombre(), i.getTipovehiculoNombre(), i.getColorNombre(), i.getVehiculoCapacidad(),
                        i.getUnidadmedidaNombre(),
                        i.getZonaNombre(), i.getDispositivoCodigo(), i.getDispositivoNombre(), Estado};
                    mdl.addRow(data);
                    }
                }
            }
        }
        @Override
        public void onFailure(Call<List<CD_Vehiculo>> call, Throwable t) {
        }
    });
    return mdl;
}
```

Ilustración 165. Código Fuente Método table() Entidad Vehículo. Fuente: Autor.

El método tableid(), no se encuentra vinculado con el API REST de Traccar y solo utiliza el API REST de la base de datos EMMAIPC-EP.

```
public DefaultTableModel tableid(CD_Vehiculo vehiculo) {
    DefaultTableModel mdl = new DefaultTableModel();
    mdl.addColumn("Id");
    mdl.addColumn("Placa");
    mdl.addColumn("Marca");
    mdl.addColumn("Tipo Vehiculo");
    mdl.addColumn("Color");
    mdl.addColumn("Capacidad");
    mdl.addColumn("Unidad Medida");
    mdl.addColumn("Zona");
    mdl.addColumn("Código Dispositivo");
    mdl.addColumn("Nombre Dispositivo");
    mdl.addColumn("Estado");
    IR_Vehiculo r_Vehiculo = retrofit.create(IR_Vehiculo.class);
    Call<CD_Vehiculo> call = r_Vehiculo.find(authHeader, vehiculo);
    call.enqueue(new Callback<CD_Vehiculo>() {
        @Override
        public void onResponse(Call<CD_Vehiculo> call, Response<CD_Vehiculo> response) {
            CD_Vehiculo i = response.body();
            if(userlogin.getTipousuarioNombre().trim().toLowerCase().equals((PrivilegioAlto).trim().toLowerCase())){
                    String Estado = (i.getVehiculoEstado()==1) ? "Activo" : "Inactivo";
                    Object data []={i.getVehiculoId(), i.getVehiculoPlaca(), i.getMarcaNombre(), i.getTipovehiculoNombre(), i.getColorNombre(), i.getVehiculoCapacidad(),
                        i.getUnidadmedidaNombre(),
                        i.getZonaNombre(), i.getDispositivoCodigo(), i.getDispositivoNombre(), Estado};
                    mdl.addRow(data);
            }
            else{
                if(i.getZonaId() == userlogin.getZonaId()){
                    String Estado = (i.getVehiculoEstado()==1) ? "Activo" : "Inactivo";
                Object data []={i.getVehiculoId(), i.getVehiculoPlaca(), i.getMarcaNombre(), i.getTipovehiculoNombre(), i.getColorNombre(), i.getVehiculoCapacidad(),
                    i.getUnidadmedidaNombre(),
                    i.getZonaNombre(), i.getDispositivoCodigo(), i.getDispositivoNombre(), Estado};
                mdl.addRow(data);
                }
            }
        }
        @Override
        public void onFailure(Call<CD_Vehiculo> call, Throwable t) {
            JOptionPane.showMessageDialog(null, "No se encontro ningun registro con el Id: " + vehiculo.getVehiculoId());
        }
    });
    return mdl;
}
```

Ilustración 166. Código Fuente Método tableid() Entidad Vehículo. Fuente: Autor.

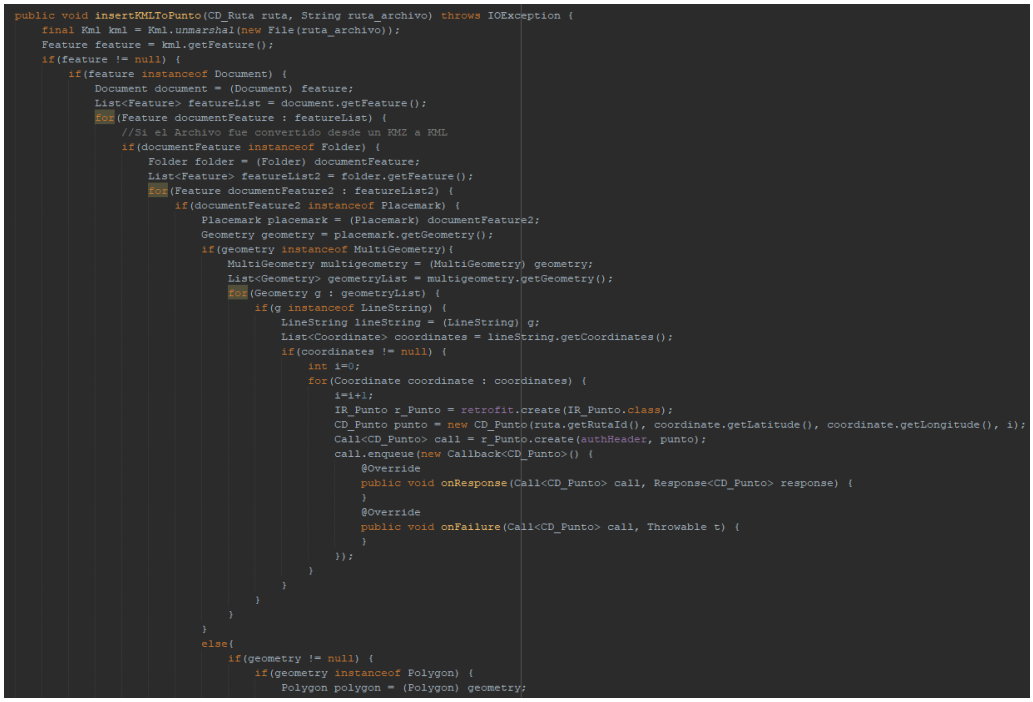
Otro método que se encuentra en cada una de las clases de este paquete, es el método combo, que sirve para poder cargar los datos dentro del componente combobox.

```
public void combo(JComboBox box) {
    DefaultComboBoxModel lista;
    lista = new DefaultComboBoxModel();
    lista.addElement("Seleccione el Vehiculo");
    box.setModel(lista);
    IR_Vehiculo r_Vehiculo = retrofit.create(IR_Vehiculo.class);
    Call<List<CD_Vehiculo>> call = r_Vehiculo.findAll(authHeader);
    call.enqueue(new Callback<List<CD_Vehiculo>>() {
        @Override
        public void onResponse(Call<List<CD_Vehiculo>> call, Response<List<CD_Vehiculo>> response) {
            List<CD_Vehiculo> _lista = response.body();
            for (CD_Vehiculo i: _lista){
                if(i.getVehiculoEstado()== 1){
                    lista.addElement(new CD_Vehiculo(i.getVehiculoId(), i.getMarcaId(), i.getColorId(), i.getUnidadmedidaId(),
                            i.getTipovehiculoId(), i.getZonaId(), i.getVehiculoPlaca(), i.getVehiculoCapacidad(),
                            i.getDispositivoCodigo(), i.getDispositivoNombre(), i.getVehiculoEstado(), i.getVehiculoNotas()));
                }
            }
        }
        @Override
        public void onFailure(Call<List<CD_Vehiculo>> call, Throwable t) {
        }
    });

}
```

Ilustración 167. Código Fuente Método combo() Entidad Vehículo. Fuente: Autor.

También dentro de este paquete hay que tomar en cuenta al método insertKMLToPunto(), que se encuentra dentro de la clase CN_Ruta, ya que el mismo permite convertir el archivo KML que contiene los puntos de la ruta de recorrido, para poder ser almacenados dentro de la tabla Punto de la base de datos EMMAIPC-EP.



```
public void insertKMLToPunto(CD_Ruta ruta, String ruta_archivo) throws IOException {
    final Kml kml = Kml.unmarshal(new File(ruta_archivo));
    Feature feature = kml.getFeature();
    if(feature != null) {
        if(feature instanceof Document) {
            Document document = (Document) feature;
            List<Feature> featureList = document.getFeature();
            for(Feature documentFeature : featureList) {
                //Si el Archivo fue convertido desde un KMZ a KML
                if(documentFeature instanceof Folder) {
                    Folder folder = (Folder) documentFeature;
                    List<Feature> featureList2 = folder.getFeature();
                    for(Feature documentFeature2 : featureList2) {
                        if(documentFeature2 instanceof Placemark) {
                            Placemark placemark = (Placemark) documentFeature2;
                            Geometry geometry = placemark.getGeometry();
                            if(geometry instanceof MultiGeometry){
                                MultiGeometry multigeometry = (MultiGeometry) geometry;
                                List<Geometry> geometryList = multigeometry.getGeometry();
                                for(Geometry g : geometryList) {
                                    if(g instanceof LineString) {
                                        LineString lineString = (LineString) g;
                                        List<Coordinate> coordinates = lineString.getCoordinates();
                                        if(coordinates != null) {
                                            int i=0;
                                            for(Coordinate coordinate : coordinates) {
                                                i=i+1;
                                                IR_Punto r_Punto = retrofit.create(IR_Punto.class);
                                                CD_Punto punto = new CD_Punto(ruta.getRutaId(), coordinate.getLatitude(), coordinate.getLongitude(), i);
                                                Call<CD_Punto> call = r_Punto.create(authHeader, punto);
                                                call.enqueue(new Callback<CD_Punto>() {
                                                    @Override
                                                    public void onResponse(Call<CD_Punto> call, Response<CD_Punto> response) {
                                                    }
                                                    @Override
                                                    public void onFailure(Call<CD_Punto> call, Throwable t) {
                                                    }
                                                });
                                            }
                                        }
                                    }
                                }
                            }
                            else{
                                if(geometry != null) {
                                    if(geometry instanceof Polygon) {
                                        Polygon polygon = (Polygon) geometry;
```

Ilustración 168. Código Fuente Método insertarKMLToPunto() Entidad Ruta I. Fuente: Autor.

```
//Si el Archivo es originalmente KML
else if(documentFeature instanceof Placemark){
    Placemark placemark = (Placemark) documentFeature;
    Geometry geometry = placemark.getGeometry();
    if(geometry != null) {
        if(geometry instanceof Polygon) {
            Polygon polygon = (Polygon) geometry;
            Boundary outerBoundaryIs = polygon.getOuterBoundaryIs();
            if(outerBoundaryIs != null) {
                LinearRing linearRing = outerBoundaryIs.getLinearRing();
                if(linearRing != null) {
                    List<Coordinate> coordinates = linearRing.getCoordinates();
                    if(coordinates != null) {
                        int i=0;
                        for(Coordinate coordinate : coordinates) {
                            i=i+1;
                            IR_Punto r_Punto = retrofit.create(IR_Punto.class);
                            CD_Punto punto = new CD_Punto(ruta.getRutaId(), coordinate.getLatitude(), coordinate.getLongitude(), i);
                            Call<CD_Punto> call = r_Punto.create(authHeader, punto);
                            call.enqueue(new Callback<CD_Punto>() {
                                @Override
                                public void onResponse(Call<CD_Punto> call, Response<CD_Punto> response) {
                                }
                                @Override
                                public void onFailure(Call<CD_Punto> call, Throwable t) {
                                }
                            });
                        }
                    }
                }
            }
        }
        else{
            if (geometry instanceof LineString) {
                LineString lineString = (LineString) geometry;
                List<Coordinate> coordinates = lineString.getCoordinates();
                if(coordinates != null) {
                    int i=0;
                    for(Coordinate coordinate : coordinates) {
                        i=i+1;
                        IR_Punto r_Punto = retrofit.create(IR_Punto.class);
                        CD_Punto punto = new CD_Punto(ruta.getRutaId(), coordinate.getLatitude(), coordinate.getLongitude(), i);
                        Call<CD_Punto> call = r_Punto.create(authHeader, punto);
                        call.enqueue(new Callback<CD_Punto>() {
                            @Override
                            public void onResponse(Call<CD_Punto> call, Response<CD_Punto> response) {
```

Ilustración 169. Código Fuente Método insertarKMLToPunto() Entidad Ruta II. Fuente: Autor.

Capa de Presentación

Se crea este paquete con el objeto de almacenar cada una de las pantallas o interfaces graficas de las entidades, así como también la pantalla principal del sistema de información.

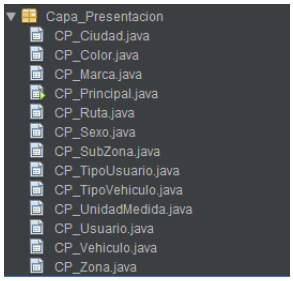


Ilustración 170. Paquete Capa de Presentación Sistema de Información. Fuente: Autor.

Cada una de las interfaces graficas de las entidades se encuentran conformadas por la codificación de eventos al hacer click en cada uno de los componentes de la misma. Dichos eventos permiten crear, actualizar, eliminar, buscar y obtener todos los registros.

La pantalla principal igualmente está conformada por la codificación de eventos que permiten acceder a cada una de las pantallas de las entidades. La pantalla principal contiene el inicio de sesión del sistema y a su vez, es la encargada de permitir o denegar el acceso a cada una de las pantallas internas, basándose en los privilegios del usuario registrado.

```
private void btningresarlMouseClicked(java.awt.event.MouseEvent evt) {
    IR_Usuario r_Usuario = retrofit.create(IR_Usuario.class);
    Call<List<CD_Usuario>> call = r_Usuario.findAll(authHeader);
    call.enqueue(new Callback<List<CD_Usuario>>() {
        @Override
        public void onResponse(Call<List<CD_Usuario>> call, Response<List<CD_Usuario>> response) {
            boolean respuesta = false;
            List<CD_Usuario>_lista = response.body();
            for (CD_Usuario i: _lista){
                if((txtusernamel.getText().equals(i.getUsuarioUsername()) && (txtcontraseñal.getText().equals(i.getUsuarioPassword())))){
                    try {
                        userlogin = i;
                        cedulita = i.getUsuarioCedula();
                        respuesta = true;
                    } catch (Exception e) {
                    }
                }
            }
            if(respuesta){
                JOptionPane.showMessageDialog(null, "¡Inicio de Sesión Exitoso!");
                mostrar();
            }
            else{
                JOptionPane.showMessageDialog(null, "¡Inicio de Sesión Fallido!");
                JOptionPane.showMessageDialog(null, "Por Favor. Verifique que su Usuario y Contraseña sean Correctos");
            }
        }
        @Override
        public void onFailure(Call<List<CD_Usuario>> call, Throwable t) {
        }
    });    // TODO add your handling code here:
}
```

Ilustración 171. *Código Fuente Evento Boton Inicio de Sesión Login Pantalla Principal. Fuente: Autor.*

Anexo 3: Protocolo de Tesis

A. TÍTULO
Prototipo de un aplicativo móvil de geolocalización del recolector de desechos en base a su recorrido para los usuarios de la EMMAIPC-EP

B. DOMINIO, LÍNEA Y ÁMBITOS DE INVESTIGACIÓN			
Energía eléctrica y tecnologías de la información para la innovación y el desarrollo sostenible	**Ciencias de los ordenadores, Analítica de datos y Algoritmos computacionales**	Analítica de datos	
		Ingeniería de software	X
		Algoritmos computacionales	
		Inteligencia de negocios	
		Gobierno de TI	
		Auditoria y seguridad informática	
		Simulación	

C. PLANTEAMIENTO DEL PROBLEMA
Con el continuo avance y uso de las tecnologías dentro de los diferentes ámbitos y entornos de la sociedad, han permitido proporcionar herramientas que ayuden a realizar actividades de una forma eficiente, obteniendo así grandes beneficios para las personas que hagan uso de ellas. Es por ello que hoy en día, se ha visto la necesidad de los usuarios de la EMMAIPC-EP de disponer de una herramienta tecnológica que les permita y facilite conocer los horarios en base a los tipos de desechos de recolección, rutas de recorrido e identificación de la ubicación en tiempo real del recolector al momento de estar a una distancia cercana al usuario, de esta manera, permitirá a la empresa realizar su proceso de recolección de una manera eficiente ya que la ciudadanía estará al tanto de la llegada del recolector a su ubicación, a su vez, teniendo conocimiento del tipo de desecho de recolección y horario correspondiente a su ruta de recorrido.

D. OBJETIVO GENERAL
Desarrollar un prototipo de un aplicativo móvil de geolocalización del recolector de desechos en base a su recorrido, para facilitar a los usuarios de la EMMAIPC-EP la identificación de la llegada del recolector a su ubicación, el tipo de desecho y horario correspondiente a su ruta de recorrido.

E. OBJETIVOS ESPECÍFICOS

1. Identificar un servidor de demostración de software libre para el almacenamiento y obtención de los datos de geolocalización de un dispositivo con GPS.

2. Analizar los requerimientos para desarrollar un prototipo de un aplicativo móvil de geolocalización del recolector de desechos en base a su recorrido para los usuarios de la EMMAIPC-EP.

3. Desarrollar un prototipo de un aplicativo móvil de geolocalización del recolector de desechos en base a su recorrido para los usuarios de la EMMAIPC-EP.

F. JUSTIFICACIÓN

En la actualidad las empresas buscan incorporar las Tecnologías de la Información y Comunicación dentro de sus procesos, tratando de lograr el cumplimento de sus objetivos estratégicos de forma eficiente. Dichas tecnologías son consideradas como pilar fundamental dentro de las organizaciones, a medida que resultan ser consideradas como un elemento estratégico para el crecimiento, desarrollo, transformación y maduración de la empresa (Rocha Velandia & Echavarría Suarez , 2017).

Se optó por el desarrollo de un prototipo de aplicativo móvil debido a la creciente popularidad de los teléfonos inteligentes modernos en la actualidad, a su vez, de la mano con la creciente disponibilidad de los puntos de acceso Wi-Fi y de los datos móviles de alta velocidad, han generado una gran oportunidad para los aplicativos móviles avanzados. Los cuales, son considerados como un factor importante a tener en cuenta dentro de cualquier tipo de negocio o empresa debido a su alta conectividad (el poder tener acceso a Internet) que existe entre las personas hoy en día. Es por ello, que las empresas optan por brindar soluciones informáticas tanto a los clientes como a los empleados, o a su vez, para varias aplicaciones generales de productividad.

Los dispositivos móviles tienen una mayor accesibilidad, por ende, hace que el desarrollo de aplicaciones móviles cobre cada vez más importancia dentro de la sociedad, en especial cuando estas permiten facilitar y ser un soporte para las personas dentro de diferentes aspectos de su vida cotidiana (Meier, 2012).

Con respecto a la temática, se vio la viabilidad del desarrollo del prototipo de aplicativo móvil debido a que día a día dentro del cantón Cañar, se puede evidenciar una gran cantidad de desechos que se encuentran botados en las calles del cantón, generando como resultado evidencias de que los actuales mecanismos o procedimientos de recolección de desechos no son lo suficientemente eficientes, generando contaminación ambiental y posibles riesgos de salud para la ciudadanía, debido al corto tiempo que disponen los vehículos recolectores para realizar la recolección de los desechos dentro de un determinado sector, provocando en algunos casos que la ciudadanía no logre depositar dentro de determinado tiempo sus desechos dentro del vehículo recolector por diferentes motivos, ya sean estos por la ubicación del domicilio del ciudadano, el mismo que dificulta el traslado de los desechos a tiempo hacia el vehículo recolector o por las malas costumbres de la gente de preparar sus desechos a último momento.

Es por todo ello que se ha visto la necesidad del desarrollo de un prototipo de aplicativo móvil que cumpla con las funcionalidades de permitir y facilitar a los usuarios de la EMMAIPC-EP conocer los horarios en base a los tipos de desechos de recolección, rutas de recorrido e identificación de la ubicación en tiempo real del recolector al momento de estar a una distancia cercana al usuario, permitiendo disminuir las incertidumbres de: ¿Qué tipo de desecho se tiene que botar cada día en base a mi ruta de recorrido?, ¿Cuál es el horario de recolección de mí ruta de recorrido?, ¿Cómo puedo identificar que el vehículo recolector está cerca de mi domicilio o punto de recolección?; generando así, una mejor calidad de servicio por parte de la empresa, creando cultura ambiental y satisfaciendo las necesidades de sus usuarios.

G. ALCANCE

- El prototipo tiene como alcance ser realizado en la Zona A (cantón Cañar), la misma que es gestionada por la EMMAIPC-EP.
- Las rutas de recorridos, vehículos, horarios y tipos de desechos que estarán inmersos en el desarrollo del prototipo son los que se ven manejados por la EMMAIPC-EP en el proceso de recolección de desechos dentro de la Zona A (cantón Cañar), a excepción del proceso de recolección de desechos de los puntos conflictivos.
- Para la obtención de los datos de geolocalización de los vehículos recolectores se pretende hacer uso de un Smartphone que simule el funcionamiento de un Dispositivo GPS.

- El almacenamiento de los datos de geolocalización será en servidores de demostración o servidores gratuitos que ofrecen plataformas de software libre.
- El desarrollo del prototipo de aplicativo móvil será para el sistema operativo Android.

H. CONCEPTOS RELACIONADOS

Lenguaje de Programación

Es considerado como un conjunto de instrucciones que son utilizadas para interactuar con las computadoras. La interacción se lo realiza con el uso de instrucciones y algoritmos que se encuentran elaboradas en una sintaxis que le permite a una computadora comprender e interpretar en el lenguaje máquina.

Los lenguajes de programación ayudan a procesar de forma ligera grandes cantidades de información complejas, realizando dicho proceso de una forma eficiente. Entre los lenguajes de programación considerados como los más populares se encuentran JavaScript, Java y Python, C++, C#, Visual Basic, Go, Ruby (López Mendoza, 2020).

Java

Un lenguaje de programación que fue desarrollado en el año 1995 por Sun Microsystems, uno de los puntos robustos de este lenguaje es que es muy versátil. Permitiendo ser utilizado para aplicaciones web hasta aplicativos android, videojuegos, programas computacionales, etc.

Otra de la ventaja de la aplicación de este lenguaje es que es independiente del hardware, gracias al intérprete de Java que permite que una aplicación desarrollada en ella, pueda codificarse solo una vez y ejecutarse en cualquier ordenador o móvil. De esta manera, la única preocupación que se tiene que tener es solamente programar.

Existes mucha documentación acerca de este lenguaje, por ello, la facilidad de su aprendizaje. Es un lenguaje muy seguro, estable y orientado a objetos, lo cual es una forma de programar intuitiva y sencilla, donde hacen presencia las entidades que ejecutan una función y manipulan datos. Se trata de crear relaciones de dichas entidades, según los datos con la finalidad de crear programas (Pascual Estapé, 2020).

NetBeans IDE 8.2

Netbeans es considerando un IDE de desarrollo de uso libre, el mismo que para su funcionamiento hace uso del lenguaje Java. Es una herramienta de desarrollo muy robusto

debido a que cuenta con una gran cantidad de módulos que le permiten hacerlo extensible. Dicha cualidad lo convierte en un entorno de desarrollo poderoso, con la capacidad de desarrollar aplicaciones en lenguaje Java tanto para escritorio como para dispositivos móviles.

También cuenta con una interfaz agradable, con un sinnúmero de componentes y librerías de reutilización. Con respecto al desarrollo de aplicaciones web, NetBeans proporciona el uso de Java Server Page (JSP), aunque también soporta diferentes frameworks, entre ellos se destacan PHP, Swing MVC, C/C++, Groovy, HTML5, etc.

Netbeans ofrece la posibilidad de crear diferentes proyectos basados en la elección o utilización de los frameworks. Por otro lado, el editor de código que incorpora este IDE es multilenguaje y permite disponer del colorado común que se observa en los diferentes lenguajes de programación modernos. Ofrece el control de versiones, plantillas reutilizables, comprobaciones semánticas y sintácticas, accesibilidad a las clases utilizando solamente un clic del mouse, entre otras características comúnmente utilizadas para el desarrollo o creación de software.

El código fuente de este IDE permite ser utilizado para crear aplicaciones tanto no comerciales como comerciales, debido a que es abierto y gratuito (Gómez Jiménez & Moreno Nuñez, 2019).

PHP

PHP es considerado como un lenguaje de desarrollo web, el mismo que está escrito para y por desarrolladores web. PHP son las siglas de Preprocesador de hipertexto.

PHP es el lenguaje de secuencias de comandos dentro del lado del servidor, que permite incorporarse en HTML o usarse como un binario independiente (aunque el uso mencionado anteriormente es con frecuencia más utilizado).

PHP es un módulo oficial de Apache HTTP Server, el servidor web gratuito líder en el mercado que ejecuta aproximadamente el 67 por ciento de la World Wide Web (según la encuesta de servidores web Netcraft, ampliamente citada). Esto significa que el motor de secuencias de comandos PHP puede integrarse en el propio servidor web, lo que permite un procesamiento más rápido, una asignación de memoria más eficiente y un mantenimiento enormemente simplificado. Al igual que Apache Server, PHP es totalmente multiplataforma, lo que significa que se ejecuta de forma nativa en varias versiones de Unix, así como en

Windows y ahora en Mac OS X. Todos los proyectos bajo la égida de Apache Software Foundation, incluido PHP, son software de código abierto (Converse, Park, & Morgan, 2004).

MySQL

Es por excelencia, el sistema gestor de una base de datos relacionales. Hoy en día es un SGBD muy utilizado dentro de las páginas web, ya que es multihilo y multiusuario. Además es muy utilizado en aplicaciones que han sido creadas como software libre.

Dentro de sus principales ventajas, destacan las siguientes:

- Soporte multiplataforma
- Facilidad de usabilidad y gran rendimiento
- Cuenta con el fundamento SSL
- Facilidad para configurar e instalar

Su principal desventaja hace referencia a la escalabilidad, es decir, no trabajo de una manera adecuada con las bases de datos que son consideradas demasiado grandes, es decir, que superan un tamaño determinado (Marín, 2019).

phpMyAdmin

phpMyAdmin es una aplicación web desarrollada en PHP, la cual como la mayoría de las aplicaciones web, contiene código de cliente JavaScript, XHTML y CSS. Proporciona una completa interfaz web para gestionar bases de datos MySQL y es reconocida ampliamente como una de las aplicaciones líderes en este campo. Al ser de open source desde su existencia, ha permitido contar con el apoyo de un gran número de desarrolladores y traductores en todo el mundo (siendo traducido a 54 idiomas en el momento de su publicación). El proyecto está alojado actualmente en SourceForge (Delisle, 2008).

Aplicativo Móvil

Una App o aplicación móvil es un programa diseñado para un objetivo específico o de forma general que permitirá al usuario la interacción con ella y la ejecución de una tarea de cualquier tipo para lo que fue creada.

Estas aplicaciones se encuentran instaladas por defecto en dispositivos de fábrica y muchas de ellas son descargables por medio de conexión a Internet para que el usuario pueda instalarlos en su dispositivo y podrían ser de pago o gratuitas (Pinos Luna, 2017, pág. 25).

Tipos de Aplicativos Móviles.

- **Nativas.** Deberá mantener actitudes nativas típicas de una serie específica de dispositivos por lo que carece de soporte y compatibilidad con múltiples plataformas, están desarrolladas mediante un lenguaje de programación específico. Al ser único de un sistema este tendrá como ventaja un mejor soporte y uso de recursos del hardware además de fluidez y uso independiente de la conexión a redes
- **Web.** La aplicación se desarrolla en un lenguaje compatible con todos los dispositivos que poseen acceso a navegadores de Internet y conexión a la misma, así que será posible su programación independiente del sistema operativo al que desea tener como objetivo. El contenido de la aplicación deberá ser responsive (adaptable a cualquier pantalla en la que se proyecta) y de distribución directa.
- **Híbridas.** Es la combinación de las dos anteriores tomando lo mejor de las dos, esta permitirá ser utilizada en todos los dispositivos y además acceder a una gran parte del hardware por lo que tendrá un mejor control y compatibilidad mediante lenguajes de programación Web por lo que su distribución sería más directa sin pasar por una tienda de aplicaciones como sucede en las aplicaciones nativas (Pinos Luna, 2017, pág. 28).

Android

Android no es más que un SO enfocado a dispositivos móviles, el cual es una versión variante de Linux. Fue desarrollado en el año 2005 por una startup llamada del mismo nombre, Android, Inc. Posteriormente Google compró la empresa, haciéndose cargo tanto de su equipo como trabajo de desarrollo.

Android al ofrecer un enfoque unificado para su desarrollo de aplicaciones, es considerada como una ventaja de la adopción del mismo, por ende, los desarrolladores solamente necesitan codificar para Android en general, las aplicaciones creadas podrán ejecutarse de forma eficiente en un gran número de dispositivos diferentes, siempre y cuando dichos dispositivos posean o se alimenten de Android. Dentro del mundo de los Smartphone, las aplicaciones son consideradas como un fragmento considerablemente importante en la cadena del éxito (DiMarzio & DiMarzio, 2016).

Android Studio

Es un IDE solemne para la creación de app's para el sistema operativo Android, el mismo que se encuentra apoyado en IntelliJ IDEA. A más de las peculiaridades que ofrece IntelliJ, Android Studio ofrece lo siguiente, entre mucho más:

- Construya variantes y generación de múltiples archivos .apk.
- Editor de diseño enriquecido con soporte para la edición de temas de arrastrar y soltar.
- Plantillas de código para ayudarlo a crear funciones comunes de la aplicación.
- Sistema de construcción flexible basado en Gradle.
- Capacidades de firma de aplicaciones y ProGuard.
- Herramientas de pelusa para detectar el rendimiento, la usabilidad, la compatibilidad de las versiones, y otros problemas presentes
- Compatibilidad incorporada para la plataforma Google Cloud, facilitando la integración de la mensajería de App Engine y Google Cloud (Roy, 2016).

Como se mencionó anteriormente, Android Studio se encuentra apoyado en software IntelliJ IDEA de JetBrains, el cual es un IDE gratuito mediante Licencia Apache 2.0. Se encuentra disponible en las plataformas Mac OS X, Microsft Windows y GNU/Linux (Parra Cangas, 2018).

Google Maps API

El servidor de aplicaciones de mapas de Google permite hacer uso de sus funcionalidades a través APIs desarrolladas para Android, las cuales se encuentran disponible en los servicios de Google Play, de tal modo que las aplicaciones puedan y tengan la facilidad de identificar la ubicación, agregar mapas, datos, buscar sitios cercanos relevantes, etc. Dicha API, además permite adicionar mapas apoyados en datos de Google Maps a las aplicaciones. Para ello, la API realiza una gestión de forma automática acerca del acceso a los servidores, visualización de mapas, descargas de datos y replica a gestos de mapas de Google Maps. También ofrece otras funcionalidades para poder añadir polígonos, marcadores y superposiciones a un mapa básico, y para modificar la vista del usuario de tal modo que se presente un área del mapa en particular (Bustillos Maldonado, 2019).

Geolocalización

Se define como geolocalización, a la posibilidad de poder conocer la posición real geográfica a través de diferentes parámetros que permiten identificar a la misma. La localización puede ser realizada en diferentes dimensiones, ya sean estas en un plano con 2 dimensiones (Google

Maps, como ejemplo) o en un plano con tres dimensiones (GPS). Es muy utiliza para identificar la posición de una persona u objeto, los mismos que pueden ser un dispositivo con acceso hacía Internet, un dispositivo móvil o cualquier otro dispositivo que permite ser rastreado. Durante los últimos tiempos, diferentes ejemplares tecnológicos han optado por la Geolocalización, obteniendo un extraordinario apogeo de esta dentro de las tecnologías móviles de última generación (Suárez Codena, 2013).

Sistemas de localización de dispositivos móviles.

Para localizar un dispositivo móvil existen diferentes maneras de realizarlo, pero para su efectividad dependerá de ciertas variables, en este caso, haciendo referencia al medio o a la disponibilidad de dicha medición en el terminal.

Estos sistemas se pueden clasificar en tres grandes tipos:

- **Basados en el terminal:** Los dispositivos que se encuentran dentro de este grupo, son aquellos que tienen a disposición un software cliente para poder precisar la posición del terminal, mediante señales externas, a su vez cuenta con un receptor de señales.
- **Híbridos:** Como su nombre mismo lo dice es un sistema que se basa en la combinación de los sistemas anteriores. Aunque este disponga de las características más fiables de los sistemas anteriores, también presenta los mismos defectos de aquellos.
- **Basados en la red:** Este tipo de sistemas hace uso de los sistemas de proveedores de servicios que permiten especificar la posición del terminal, por lo que no existe la necesidad de disponer de una aplicación específica dentro del dispositivo móvil. El problema principal radica en que es importante estar cerca de dicho proveedor para que el funcionamiento sea eficaz (Suárez Codena, 2013).

A-GPS

Se basa en recibir señal GPS a través de otro tipo de señal, ya sea esta Wireless hasta telefonía, etc. Es un sistema mixto, creado para solventar problemas de geolocalización con GPS dentro de interiores que tienen un déficit de cobertura de señal de satélites o en ciudades grandes, permitiendo aumentar su precisión.

Dicho sistema tiene 2 formas de funcionamiento, la una hace referencia al modo offline, donde el dispositivo no cuenta con una conexión estable hacia Internet, ya sea a través de los

diferentes medios como Wirelees, GPRS o Ethernet, de tal manera que, la tecnología A-GPS para su funcionamiento realiza la descarga de ciertos ficheros que contienen información acerca de las celdas o de las posiciones de los satélites, del entorno, etc., Este proceso de descarga lo realiza mientras se dispone de acceso a Internet, en otras palabras, tener una conexión estable, está tecnología puede hacer uso de dicha información, incluso durante varios días.

La otra forma de funcionamiento es el modo online, en donde el dispositivo se tiene acceso a Internet con una conexión estable ya sea está a través de los diferentes medios mencionados anteriormente o a través de una red telefónica como GSM. En el modo online, la tecnología A-GPS puede obtener el posicionamiento referente a la celda en la que se encuentra, a su vez, recibiendo información acerca del entorno, como las condiciones ionósfericas o la posición de los satélites, con el objeto de aumentar de forma efectiva la precisión de la tecnología GPS, o incluso enviar datos de geolocalización al servidor de asistencia (Suárez Codena, 2013).

Sistema de Posicionamiento Global (GPS)

El GPS es una tecnología de navegación basada en satélites. Los satélites instalados en la órbita terrestre transmiten señales a dispositivos receptores conocidos como rastreadores GPS. Los rastreadores GPS ayudan a decodificar la ubicación exacta del usuario del dispositivo calculando las coordenadas de las señales recibidas del satélite. La tecnología GPS funciona las 24 horas del día, los 7 días de la semana para determinar parámetros como la velocidad, la ubicación, la distancia, el seguimiento, la dirección, etc. de los usuarios / activos que utilizan los dispositivos GPS (Mordy, s.f.).

Componentes necesarios para el funcionamiento GPS

La tecnología GPS consta de tres componentes: satélite, rastreador GPS y aplicación de software de rastreo GPS.

- **Satélite:** es una máquina enviada al espacio (la órbita de la Tierra) para recopilar información importante.
- **Rastreador GPS:** El rastreador GPS es un dispositivo que puede recibir señales de satélites y la red GNSS (Sistema de navegación global por satélite) para determinar la ubicación de:
    - El usuario que lo lleva.

- El activo en el que está instalado.

- **Software de rastreo GPS:** El software de rastreo GPS es una aplicación que puede leer, administrar, organizar y almacenar los datos recibidos de los dispositivos GPS (Mordy, s.f.).

Comparativa de programas de rastreo GPS gratuitos y de código abierto más populares

Name of the Software	Whether Free or not	Whether Open- source or not?	Best Features
Traccar	Free	Open source	Denotes whether the device is connected to the server (online) or not (offline)
OpenGTS	Free	Open source	Supports data collection and storage of GPS Tracking data from remote devices
Trackit	Free	Open source	See the latest status of all device parameters by clicking on the device icon on the map
TrackMyRide	Free	Open source	Displays a track trail of your device location for the past 24 hours
Navit	Free	Open source	Users can tailor map layouts, on-screen display, details of the routing engine, and more
GPS Trace	Free	No	The system sends alerts via email or pop-up notifications
Trackme	Free	No	Create Geo-fences and routes to set boundaries and areas that need specific monitoring
			GoodFirms

Tabla comparativa de programas de rastreo GPS gratuitos y de código abierto más populares; (Mordy, s.f.)

Descripción	Traccar	OpenGTS	OpenGPS
Monitoreo en tiempo real	X	X	X
Encendido y apagado del motor	X	X	X
Ruta recorrida	X	X	
Limitación geográfica	X	X	X
Botón de pánico	X	X	
Conexión por datos móviles	X	X	X
Micrófono	X		X
Reportes	X	X	
Interfaz web	X	X	X
Alertas y notificaciones	X	X	X
Basta información en la red	X		
Autenticación de usuarios	X	X	X
Versatilidad	X	X	
Facilidad de uso	X		X

Comparativa Herramientas de Software Libre; (Flores Morillo, 2020, pág. 10)

Traccar

Traccar es un software de rastreo GPS gratuito y de código abierto con características modernas, alto rendimiento y una plataforma de alta gama. Traccar tiene un soporte sólido para todos los dispositivos modernos. Traccar le permite ver el estado de todos los dispositivos GPS en tiempo real. Traccar viene con múltiples capacidades de mapeo, incluidas imágenes de satélite y navegación por mapas de carreteras. Tiene un sólido sistema de alerta e informes. Traccar también tiene aplicaciones móviles para plataformas Android e iOS (Mordy, s.f.).

Traccar Client

Es un aplicativo móvil multiplataforma perteneciente a Traccar, fue desarrollado tanto para plataformas Android y iOS. Permite ser usado como un módulo GPS, el cual mediante su respectiva configuración, envía los datos de geolocalización a los servidores de demostración de Traccar a través del puerto 5055, exclusivo para Traccar Client. A parte de ofrecer un aplicativo para simular un dispositivo GPS, Traccar es compatible con un sinnúmero de modulos GPS que son comerciales, es capaz, de soportar más de 1000 modelos. Traccar Client no solo es compatible con los servidores de demostración de Traccar, sino que también es compatible con un servidor propio de Traccar o con cualquier otra plataforma que preste servicios de rastreo GPS (Flores Morillo, 2020).

Traccar Manager

Es un aplicativo móvil compatible con plataformas iOS y Android, el cual tiene la funcionalidad de gestionar y visualizar los diferentes dispositivos GPS que se encuentren

inmersos en la base de datos de Traccar. Por defecto, está aplicación está configurada para hacer uso de los servidores de demostración gratuitos de Traccar. Cuando se inicia el aplicativo por primera vez, permite seleccionar el servidor. En adición, a lo mencionado anteriormente, se puede acceder a las diferentes funcionalidades que ofrece dicho aplicativo desde su aplicación web, utilizando la dirección ip o dominio del servidor de demostración pertinente (Flores Morillo, 2020).

Servidor Web Traccar

Traccar dispone de un servidor web que incorpora Jetty, el cual es una implementación madura y estable de un servidor HTTP perteneciente a Java y su contenedor Servlet22. Dicho servidor web de Traccar está compuesto por:

- **Aplicación Web:** Traccar dispone de dicha herramienta para gestionar las diferentes funcionalidades que ofrece la plataforma Traccar para los dispositivos de rastreo GPS que se encuentran inmersos en él. Esta aplicación tiene una dependencia directa de la Web API. Además, se encuentra basada en Sencha ExtJS framework y utiliza OpenLayers para la visualización de mapas.
- **Web API:** Está interfaz está compuesta por dos partes principales WebSocket y REST API. REST API esta implementada usando el estándar de Java EE API para ofrecer los servicios web de REST-ful. Cada uno de los modelos del sistema tienen su propio punto final dedicado (Flores Morillo, 2020).

Base de datos Traccar

El servidor de Traccar permite ser configurado para poder funcionar con cualquier base de datos SQL, entre las más populares se encuentran Microsoft SQL Server, MySQL, PostgreSQL y Oracle. En este caso es remplazada la base de datos H2, la cual se encuentra integrada en Traccar por MySQL (Flores Morillo, 2020).

Servicio Web

Un servicio web permite que dos dispositivos electrónicos se comuniquen a través de Internet. La W3C describe el servicio web como "un sistema de software diseñado con el objeto de admitir la interacción interoperable de aparato a aparato a través de una red". En la práctica, esto significa que un servidor se comunica a través del puerto 80 o el puerto 443 en texto sin formato con el cliente (McWherter & Gowell, 2012).

Notación de objetos JavaScript (JSON)

JSON fue creado en 2001 y Yahoo lo utilizó en 2005. JSON tiene pocas reglas, pocos tipos de base y es legible por humanos. El esquema JSON permite la validación de documentos, pero esto rara vez se usa. JSON es un gran formato para transmitir datos entre sistemas porque es simple, se basa en texto y se describe a sí mismo (McWherter & Gowell, 2012).

REST

REST es un estilo arquitectónico para diseñar aplicaciones distribuidas que pueden intercomunicarse. Esencialmente, es un estilo arquitectónico cliente-servidor donde las conexiones no tienen estado.

REST no es un estándar; más bien, es una alternativa arquitectónica a RPC y Web Services. En el estilo arquitectónico REST, puede comunicarse entre sistemas utilizando el protocolo HTTP (si HTTP es el protocolo en uso). En realidad, la World Wide Web (WWW) puede verse como una arquitectura basada en REST. Una arquitectura RESTful se basa en un protocolo de comunicación sin estado y almacenable en caché.

REST es un estilo arquitectónico que divide el estado y la funcionalidad de una aplicación en recursos. Estos recursos, a su vez, se pueden direccionar mediante URI a través de HTTP. Estos recursos tienen una interfaz común y son direccionables de forma única. Un modelo basado en REST no tiene estado, está basado en cliente-servidor y se puede almacenar en caché.

Los principales objetivos de diseño del estilo arquitectónico REST incluyen:

- Despliegue independiente de los componentes.
- Latencia reducida.
- Alta seguridad de las interacciones del servicio.
- Escalabilidad.
- Alto rendimiento (Kanjilal, 2013).

La arquitectura REST hace uso de algunos métodos HTTP comunes para operaciones CRUD (Crear, Leer, Actualizar y Eliminar). Estos son los siguientes:

- **GET:** se utiliza para solicitar una representación específica de un recurso.
- **HEAD:** se utiliza para recuperar los encabezados de recursos únicamente.
- **PUT:** útil para actualizar un recurso.
- **DELETE:** útil para eliminar el recurso especificado.

- **POST:** se utiliza para enviar datos que serán procesados por el recurso identificado. Idealmente, POST debe usarse solo para crear recursos, mientras que PUT se usa solo para actualizarlos (Kanjilal, 2013).

Métodos HTTP

La simplicidad y la amplia adopción en términos de casi todas las plataformas que lo soportan han hecho de HTTP el transporte superior de Internet. El estilo de arquitectura REST puede beneficiarse de los elementos de HTTP aplicando conceptos REST mientras implementa aplicaciones que se ejecutan en la Web.

La extensibilidad y flexibilidad del protocolo HTTP ha asistido en gran medida al éxito de la Web y hoy en día se considera el protocolo de la Web. El protocolo HTTP se puede utilizar para acceder a recursos, no solo a páginas HTML, sino a todo tipo de recursos, incluidas imágenes, videos y aplicaciones.

Al acceder a los recursos con HTTP, se especifica un identificador de recurso junto con la acción que se realizará en ese recurso. Los URI identifican el recurso. La acción a realizar se define mediante un verbo HTTP. Existe un conjunto de verbos HTTP y cada verbo puede tener una semántica asociada que ayuda a identificar la acción que se realizará en el recurso (Abeysinghe, 2008).

La siguiente tabla resume los verbos HTTP y cómo se aplican al usar REST.

Verb	Description
GET	Retrieves a resource identified by a URI.
POST	Sends a resource to the server. Updates the resource in the location identified by the URI.
PUT	Sends a resource to the server, to be stored in the location identified by the URI.
DELETE	Deletes a resource identified by a URI.
HEAD	Retrieves the metadata of a resource identified by the URI.

Tabla de los Verbos HTTP; (Abeysinghe, 2008, pág. 32)

I. TRABAJOS RELACIONADOS

1. El trabajo nombrado a continuación, es un trabajo de titulación, en donde se pretende realizar el desarrollo de un aplicativo basado en multiplataforma utilizando geolocalización, el mismo que permita localizar establecimientos y sitios cercanos, con el objeto de dar solución dentro del Ecuador a los turistas, nacionales o extranjeros, los cuales se enfrentan al problema de la falta de tecnología para encontrar y localizar sitios o establecimientos cercanos. Es por ello que el autor ve conveniente el desarrollo de un aplicativo móvil que agilice este proceso, en donde para el desarrollo lo realiza a través de un modelo de desarrollo incremental, obteniendo diferentes versiones del aplicativo en cada iteración, a su vez, hace uso de diferentes APIs, como las de google para poder integrar los mapas en el aplicativo. El desarrollo lo hace en multiplataforma como IOS y Android, haciendo uso de framework como Readct Native. También para el correcto funcionamiento del proceso de geolocalización realiza la implementación de un API REST. Uno de los problemas de rendimiento a los que se enfrentó fue el tiempo tardío al momento de que la aplicación consiga la ubicación perteneciente al usuario, en donde el autor para solventar este problema hace uso de código Java para optimizar este problema, detectando si la tecnología GPS que está inmersa en el dispositivo esta o no activada y de cierta manera, adquirir la posición de ubicación perteneciente al usuario de manera iterativa (Morocho Rocha, 2018).

Justificación: Este trabajo proporciona una guía y orientación acerca del uso e integración de Google Maps API dentro de un proyecto de Android, permitiendo facilitar y obtener una mejor presentación de los datos de geolocalización a los usuarios que haga uso del aplicativo a desarrollar. Además el autor se ve envuelto en una problemática acerca del rendimiento del aplicativo móvil al momento de obtener la ubicación del usuario, en donde lo solventa a través del uso de un código de programación java que le permite identificar si el GPS está activo o no dentro del dispositivo y obtener la ubicación periódicamente.

2. El trabajo nombrado a continuación, es un proyecto de investigación y desarrollo de grado, en donde se pretende realizar el desarrollo de un aplicativo móvil que cuente con geolocalización de las diferentes líneas de autobuses y sobre las paradas de los mismos, orientado hacia la municipalidad de Ambato. El autor mediante el desarrollo de este aplicativo pretende proporcionar a los turistas, ciudadanos, como también negociantes tener un una facilidad de identificar de manera más detallada las diferentes rutas, así como también, las paradas de los buses dentro de la ciudad. El autor desarrolla este aplicativo enfocado solamente a los sistemas operativos Android, utilizando para el desarrollo del mismo el IDE

Android Studio, el cual le permite realizar la integración del mapa de google dentro de su aplicativo de una manera eficiente. Realiza el desarrollo de un Web Services en lenguaje de programación Visual C#, que le permita realizar la conexión y comunicación entre la base de datos y la app. El autor además utiliza Google Maps Android API, lo cual le permitió llevar a la práctica funcionalidades de georreferenciación y el diseño de las rutas dentro de la aplicación, lo que género como resultado la posibilidad de determinar la ubicación de los usuarios y con ello mostrar la ruta en tiempo real óptima (Jaramillo Zambrano, 2018).

Justificación: Este proyecto de investigación y desarrollo de grado, permite proporcionarme una visión más cercana, acerca de cómo y que herramientas podría tomar en consideración para la elaboración de mi prototipo propuesto, ya que tiene una estrecha relación a la problemática y objetivos planteados. Me ayuda e incentiva a utilizar Servicios Web para la intercomunicación y conexión entre el aplicativo móvil y la base de datos. A su vez, me ayuda a identificar y observar los diferentes beneficios e importancia que tiene el uso e integración de Google Android API dentro del desarrollo de un aplicativo móvil de geolocalización, ya que proporciona todas las herramientas y funcionalidades para el correcto desempeño y rendimiento del aplicativo.

3. El trabajo nombrado a continuación, es un trabajo de graduación, en donde se pretende realizar el desarrollo de una aplicación móvil con el objeto de identificar la posición de localización ágil de los diferentes medios de transporte terrestre, específicamente los de carga liviana de la ciudad de Latacunga. El autor para el desarrollo del aplicativo se ve envuelto en la siguiente problemática, que es la dificultad de disponer de un servicio de transporte terrestre liviano dentro de la ciudad de Latacunga en horas pico, es por ello, que se ve la necesidad de crear un aplicativo móvil que permita a las personas poder solicitar el servicio de transporte, en donde a su vez, es tomado como pieza fundamental el conductor de dicho transporte, ya que tanto el conductor como la persona que solicita el servicio está inmersa en el desarrollo del aplicativo móvil, ya que serán quienes lo usaran. El autor aclara que el aplicativo a desarrollar tiene una estrecha relación y similitud a los aplicativos populares de solicitud de servicio de transporte como: Uber, Cabify y Easy Taxi. Para el desarrollo del aplicativo móvil, el autor utiliza una Metodología ágil XP o también llamada Programación Extrema. El autor para la codificación de su aplicativo móvil utiliza el lenguaje de programación Java junto con el IDE Android Studio, con respecto a la base de datos, hace uso de la plataforma Firebase y para el funcionamiento de los servicios de google dentro del aplicativo, como los mapas, ubicaciones, localizar sitios cercanos o relevantes, utilizar

polígonos, marcadores, etc., lo hace a través de Google Maps API (Bustillos Maldonado, 2019).

Justificación: Este trabajo de graduación al tener una relación de cierta manera con mi temática, me permite identificar y analizar la forma en la que el autor de este trabajo dio solución a la problemática presentada y que herramientas utilizo para el desarrollo como lo es Android Studio junto con el lenguaje de programación Java. El autor hace uso de la plataforma de google Firebase para almacenar los datos de geolocalización y datos de los usuarios que interactúan dentro del aplicativo, proporcionándome una visión acerca de la utilidad que tiene esta plataforma al momento de crear aplicativos móviles de este tipo. Por otra parte, hace uso de las API de Google para la integración de los mapas, ubicaciones, marcadores, polígonos y graficas de las rutas, lo cual me permite orientarme a hacer uso de dichas API's para el desarrollo de mi prototipo.

4. El trabajo nombrado a continuación, es un trabajo de titulación de una maestría, en donde se pretende realizar el estudio e implementación de un software para el rastreo satelital, el mismo que se encuentra aplicado para las mascotas, haciendo uso de software libres, contando con diferentes tecnologías desde GSM hasta GPS. El autor se envuelto en la siguiente problemática de investigación, que es la deficiencia de medios tecnológicos para geolocalizar de una manera efectiva mascotas dentro del Ecuador, más concretamente en la ciudad de Quevedo, evitando así, el alto índice de mascotas perdidas. El autor se plantea desarrollar un sistema de rastreo satelital que se encuentre conectado a un aplicativo móvil mediante tecnologías GPS y GSM, dicho aplicativo es desarrollado en MIT App Inventor, obteniendo un desarrollo fácil y sencillo de aplicativos para dispositivos Android. Con respecto al apartado de la geolocalización, el autor hace uso del software de código abierto Traccar, donde realiza la implementación del mismo en un servidor Linux junto con la base de datos MySQL a través de VPS. El autor además realiza un análisis de los dispositivos GPS del mercado, para identificar cual es el que mejor se adapta al cumplimiento de sus objetivos, utilizando en este caso el dispositivo GPS "TKstar 909", obteniendo buenos resultados (Morán Cabezas, 2021).

Justificación: Este trabajo de titulación de una maestría se relaciona demasiado en el apartado de la utilización de sistemas de rastreo satelital de código abierto para la geolocalización de determinados objetos. En este caso, el autor hace uso de la plataforma Traccar obteniendo excelentes resultados y brindando una solución a la problemática de una

manera fácil, segura y eficiente. Integrando dicha plataforma, tanto sus servidores como demás funcionalidades dentro de un aplicativo móvil, el cual fue desarrollado a través de MIT App Invetor. Permitiéndome tomar en consideración como una alternativa al desarrollo de este tipo aplicativos móviles sin la utilización de código de programación y el uso de plataformas de rastreo satelital open source para el proceso de almacenamiento y georeferenciación.

J. METODOLOGÍA

Mobile D

Mobile-D es considerado como un enfoque de desarrollo ágil, su metodología tiene una estrecha relación con Extreme Programming (prácticas de desarrollo), a su vez, con Crystal (escalabilidad de métodos) y por último con Rational Unified Process (cobertura del ciclo de vida).

Esta metodología está diseñada para el trabajo en equipo con un grupo de personas menor a 10, las mismas que deben estar trabajando dentro de una misma oficina ubicada en un mismo lugar, con el propósito de generar o crear un aplicativo móvil que se encuentre completamente funcional dentro de un corto periodo de tiempo (menos de 10 semanas). Mobile-D fue desarrollado junto con 3 empresas, las mismas que se encargan de desarrollar productos y servicios en el área de software móvil. El enfoque de esta metodología fue sometido a evaluaciones contra la certificación de nivel 2 de CMMI, obteniendo excelentes resultados (Kyllönen, y otros, 2004).

Cuando un proyecto decide aplicar esta metodología, se debe de tomar en cuenta que se encuentra dividido en cinco iteraciones. Dichas fases son las siguientes:

Fases de Mobile-D; (Molina Ríos, Honores Tapia, Pedreira-Souto, & Pardo León, 2021, pág. 80)

1. **Exploración:** En esta fase se crea un plan y se establece las propiedades del proyecto, a través de tres etapas, las mismas que son: instaurar de actores, la definición del alcance y por último la instauración de proyectos. También existen tareas asociadas a esta fase, como lo es la instauración del cliente (hace referencia a la persona de lado del cliente que participa de forma operante en el proceso de desarrollo), los requisitos de recogida y plan inicial del proyecto y, y por último la implantación de los procesos.
2. **Iniciación:** En este punto se identifica y prepara todos los recursos principales. Se realiza una planificación para las próximas fases y se implanta el entorno técnico, como lo es, los recursos de comunicaciones, tecnológicos y físicos (también se incluye el entrenamiento del equipo de desarrolladores). Esta fase consta de cuatro etapas: el inicio del proyecto, el plan inicial, el día de salida y de prueba.
3. **Producción:** En esta fase se vuelve a aplicar la programación en 3 días (planificar, trabajar y liberar) iterativamente hasta lograr la implementación de todas las funcionalidades del producto final. Es decir, primero se realiza un plan de iteración del trabajo enfocado en términos de tareas y requisitos a realizar. Posteriormente de antemano se procede a preparar pruebas de iteración. Las tareas se ejecutarán durante los días de trabajo, desarrollando y a su vez, incorporando el código junto con los repositorios existentes. Por último, cuando se llegue al día final, se procede a integrar el sistema seguido de las diferentes pruebas de aceptación.
4. **Estabilización:** En esta fase se realiza los últimos procesos u operaciones para la integración, con el fin de asegurar que el producto final funcione correctamente. Dentro de esta fase, si se desarrolló el producto final en multi-equipo se deberá integrar los subsistemas desarrollados en uno solo. Adicionalmente en esta fase se puede realizar la generación de documentación.
5. **Prueba y reparación del sistema:** En esta fase, ya se dispone de una versión del sistema completamente funcional y que es óptimo para la realización de diversas pruebas, donde se toma en consideración los requisitos del cliente y se depuran los errores o defectos encontrados (Amaya Balaguera, 2013).

Cada una de las fases nombradas anteriormente se dividen en tres tipos de días de desarrollo, estos son: día de planificación, día de trabajo y por último día de lanzamiento. En este punto se tiene que tomar en consideración de que si, para el desarrollo del producto final se está dividiendo el equipo de trabajo, en donde cada uno realice una parte del mismo, a la final se tiene que disponer de un día de integración.

En lo referente a las prácticas de las distintas fases de esta metodología, están compuestas por nueve elementos principales. Los mismos, que son los siguientes:

- Métricas
- Enfoque centrado en el usuario
- Desarrollo basado en pruebas móviles
- Mejora ágil de procesos de software
- Programación en pareja
- Fases y ritmo
- Cliente fuera del sitio
- Integración continua
- Línea Arquitectura

Los elementos mencionados, en su mayoría son prácticas agiles especializadas muy populares y han sido consideradas para el desarrollo de software móvil (Kyllönen, y otros, 2004).

Justificación: Esta metodología es de suma importancia para el desarrollo de mi prototipo debido a que:

- Está orientada al desarrollo específico de aplicaciones móviles de forma eficiente.
- Es considerada como una metodología ágil que contiene ciclos de desarrollo cortos para equipos pequeños.
- Permite detectar y solventar fácilmente en etapas tempranas problemas técnicos.
- El desarrollo de esta metodología se encuentra basada en la realización de pruebas, lo cual permite asegurar la calidad del producto final.
- Se obtiene diseños óptimos al basarse en un desarrollo con pruebas.
- Se enfoca en la satisfacción del usuario final, permitiendo optimizar el producto en cada una de las iteraciones cortas.
- Cada una de las tareas de las fases están bien detalladas.

Utilizando esta metodología permitirá a mi prototipo acoplarse cada vez más a solventar la necesidad real de la problemática planteada y satisfaciendo la necesidad de los usuarios finales.

K. CRONOGRAMA DE ACTIVIDADES

N°	ACTIVIDAD	MES I				MES II				MES III				MES IV				MES V				MES VI				MEDIOS DE VERIFICACIÓN
		S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	
1	**Identificar un servidor de demostración de software libre para el almacenamiento y obtención de los datos de geolocalización de un dispositivo con GPS.**																									
1.1	Investigar, analizar y utilizar un servidor de demostración de software libre para el almacenamiento y obtención de los datos de geolocalización de un dispositivo con GPS.	X	X																							Servidor de demostración de software libre para el almacenamiento y obtención de los datos de geolocalización a utilizar en el desarrollo.
2	**Analizar los requerimientos para desarrollar un prototipo de un aplicativo móvil de geolocalización del recolector de desechos en base a su recorrido para los usuarios de la EMMAIPC-EP.**																									
2.1	Levantar Información sobre las rutas de recorridos y horarios de recolección.			X	X																					Entrevista.
2.2	Realizar el modelado de los datos.					X																				Modelo Conceptual, Lógico y Físico.
3	**Desarrollar un prototipo de un aplicativo móvil de geolocalización del recolector de desechos en base a su recorrido para los usuarios de la EMMAIPC-EP.**																									
3.1	Diseñar la base de datos para el correcto funcionamiento del aplicativo móvil.						X																			Base de Datos.
3.2	Codificar el desarrollo del API REST.							X	X																	API REST.
3.3	Codificar el desarrollo del aplicativo móvil.									X	X	X	X	X	X											Aplicativo Móvil.
3.4	Codificar el desarrollo del Sistema de Información de Escritorio.															X	X	X	X	X						Sistema de Información de escritorio.

L. DECLARACIÓN FINAL

Los abajo firmantes declaramos bajo juramento que el proyecto descrito en este documento no ha sido presentado a otra institución nacional o internacional para su financiamiento, no causa perjuicio al ambiente, es de nuestra autoría y no transgrede norma ética alguna.

M. PARTICIPANTES

DIRECTOR:	Luis Fernando Pinos Castillo
ESTUDIANTE:	Bryan Stalin Solórzano Espinoza

N. FIRMAS DE RESPONSABILIDAD

Lugar:	Cañar
Fecha:	28/01/2022

Firmas:

Nombre: Luis Fernando Pinos Castillo
CC: 0301829255
Director del Proyecto

Nombre: Bryan Stalin Solórzano Espinoza
C.C.: 0302146451
Estudiante / Egresado

O. APROBACIÓN

Firmas:

Nombre:
CC:
Primer Par Revisor

Nombre:
C.C.:
Segundo Par Revisor

P. REFERENCIAS

Abeysinghe, S. (2008). *RESTful PHP Web Services.* Packt Publishing, Limited. Recuperado el 4 de Enero de 2022, de https://ebookcentral.proquest.com/lib/ucacue/detail.action?docID=950541

Amaya Balaguera, Y. D. (14 de Noviembre de 2013). Metodologías ágiles en el desarrollo de aplicaciones. *Journal Technology, 12*(2), 111 - 124. Recuperado el 13 de Enero de 2022, de https://dialnet.unirioja.es/descarga/articulo/6041502.pdf

Bustillos Maldonado, L. A. (2019). Aplicación Móvil para localización ágil de transporte terrestre de carga liviana en la ciudad de Latacunga. *(Trabajo de Graduación).* Universidad Técnica de Ambato, Ambato, Tungurahua, Ecuador. Recuperado el 9 de Enero de 2022, de https://repositorio.uta.edu.ec/bitstream/123456789/30100/1/Tesis_t1631si.pdf

Converse, T., Park, J., & Morgan, C. (2004). *PHP5 and MySQL Bible* (1 ed., Vol. 147). John Wiley & Sons, Incorporated. Recuperado el 2 de Enero de 2022, de https://ebookcentral.proquest.com/lib/ucacue/reader.action?docID=243717&ppg=5

Delisle, M. (2008). *Mastering phpMyAdmin 2.11 for Effective MySQL Management.* Packt Publishing, Limited. Recuperado el 2 de Enero de 2022, de https://ebookcentral.proquest.com/lib/ucacue/reader.action?docID=3027380&ppg=26

DiMarzio, J., & DiMarzio, J. F. (2016). *Beginning Android Programming with Android Studio.* John Wiley & Sons, Incorporated. Recuperado el 2 de Enero de 2022, de https://ebookcentral.proquest.com/lib/ucacue/detail.action?docID=4714030

Flores Morillo, P. G. (2020). *Interfaz Abierta para Servidor de Datos de Localización.* Universidad Técnica del Norte, Ibarra. Recuperado el 5 de Enero de 2022, de http://repositorio.utn.edu.ec/bitstream/123456789/10390/2/04%20MEC%20315%20TRABAJO%20GRADO.pdf

Gómez Jiménez, E., & Moreno Nuñez, J. (2019). *Fundamentos de programación Java con NetBeans 8.2* (1 ed.). Mexico: Alfaomega. Recuperado el 12 de Enero de 2022, de https://stream2.docer.com.ar/pdf_dummy/eyJpZCI6IjMxNjU3MTAiLCJuYW1lIjoiRnVuZGFtZW50b3MgZGUgcHJvZ3JhbWFjaVx1MDBmM24gSmF2YSBjb24gTmV0QmVhbnMgOC4yLTEtMSIsImV4dGVuc2lvbiI6InBkZiIsImNoZWNrc3VtX2lkIjoiMTQ3MTMzMTIifQ,,?

Jaramillo Zambrano, E. D. (2018). Desarrollo de Aplicación Móvil, con geolocalización de líneas de autobuses y sus paradas para el Gobierno Autónomo Descentralizado Municipalidad de Ambato. *(Proyecto de investigación y desarrollo de grado).* Pontificia Universidad Católica del Ecuador Ambato, Ambato, Tungurahua, Ecuador. Recuperado el 7 de Enero de 2022, de https://repositorio.pucesa.edu.ec/bitstream/123456789/2229/2/76601.pdf

Kanjilal, J. (2013). *ASP.NET Web API.* Packt Publishing, Limited. Recuperado el 4 de Enero de 2022, de https://ebookcentral.proquest.com/lib/ucacue/detail.action?docID=1532007

Kyllönen, P., Koskela, J., Salo, O., Korkala, M., Jäälinoja, J., Ihme, T., . . . Abrahamsson, P. (2004). Mobile-D: An Agile Approach for Mobile Application Development. *Association for Computing Machinery*, 174 - 175. doi:10.1145/1028664.1028736

López Mendoza, M. (16 de Julio de 2020). *OpenWebinars*. Recuperado el 6 de Enero de 2022, de OpenWebinars: https://openwebinars.net/blog/que-es-un-lenguaje-de-programacion/

Marín, R. (16 de Abril de 2019). *Revista Digital INESEM*. Recuperado el 6 de Enero de 2022, de Revista Digital INESEM: https://revistadigital.inesem.es/informatica-y-tics/los-gestores-de-bases-de-datos-mas-usados/

McWherter, J., & Gowell, S. (2012). *Professional Mobile Application Development* (1 ed.). John Wiley & Sons, Incorporated. Recuperado el 2 de Enero de 2022, de https://ebookcentral.proquest.com/lib/ucacue/detail.action?docID=843643

Meier, R. (2012). *Professional Android 4 Application Development* (3 ed.). John Wiley & Sons, Incorporated. Obtenido de https://ebookcentral.proquest.com/lib/ucacue/detail.action?docID=818033

Molina Ríos, J. R., Honores Tapia, J. A., Pedreira-Souto, N., & Pardo León, H. P. (14 de Junio de 2021). Comparativa de metodologías de desarrollo de aplicaciones móviles. *3C Tecnología, 10*(2), 73-93. Recuperado el 13 de Enero de 2022, de https://doi.org/10.17993/3ctecno/2021.v10n2e38.73-93

Morán Cabezas, A. L. (2021). Análisis e Implementación de un Sistema de Rastreo Satelital aplicado a mascotas mediante software libre con tecnología GPS y GSM. *(Trabajo de Titulación de Maestría).* Universidad Católica de Santiago de Guayaquil, Guayaquil, Guayas, Ecuador. Recuperado el 9 de Enero de 2022, de http://201.159.223.180/bitstream/3317/16332/1/T-UCSG-POS-MTEL-192.pdf

Mordy, J. (s.f.). *GoodFirms*. Recuperado el 6 de Enero de 2022, de GoodFirms: https://www.goodfirms.co/blog/best-free-open-source-gps-tracking-software

Morocho Rocha, D. S. (2018). Desarrollo de una aplicación móvil multiplataforma con Geolocalización para localizar sitios y establecimientos cercanos. *(Trabajo de Titulación).* Universidad Central del Ecuador, Quito, Pichincha, Ecuador. Recuperado el 7 de Enero de 2022, de http://www.dspace.uce.edu.ec/bitstream/25000/16490/1/T-UCE-0011-ICF-035.pdf

Parra Cangas, C. G. (2018). *Desarrollo de un Sistema para Monitoreo y Control Satelital de vehiculos mediante el uso del dispositivo GPS TK 303G para la comercializadora de dispositiovs satelitales Genius EC.* Universidad de las Fuerzas Armadas, Sangolqui. Recuperado el 5 de Enero de 2022, de http://repositorio.espe.edu.ec/bitstream/21000/14674/1/T-ESPE-057783.pdf

Pascual Estapé, J. A. (9 de Mayo de 2020). *ComputerHoy*. Recuperado el 6 de Enero de 2022, de ComputerHoy: https://computerhoy.com/listas/industria/lenguajes-programacion-mas-populares-633547#java

Pinos Luna, K. (2017). Diseño de un videojuego para el rescate cultural identitario en adolescentes ecuatorianos. *(Proyecto de Graduación).* Universidad del Azuay, Cuenca. Recuperado el 6 de Enero de 2022, de https://dspace.uazuay.edu.ec/bitstream/datos/7173/1/13120.pdf

Rocha Velandia , J. T., & Echavarría Suarez , S. (2017). Importancia de las T.I.C.s en el ambiente empresarial. *(Trabajo de Grado).* Universidad de La Salle, Bogota, Colombia. Obtenido de https://ciencia.lasalle.edu.co/administracion_de_empresas/1483?utm_source=ciencia.lasalle.edu.co%2Fadministracion_de_empresas%2F1483&utm_medium=PDF&utm_campaign=PDFCoverPages

Roy, A. (2016). *Android Game Developer's Handbook* (1 ed.). Packt Publishing, Limited. Recuperado el 2 de Enero de 2022, de https://ebookcentral.proquest.com/lib/ucacue/detail.action?docID=4653254

Suárez Codena, S. A. (2013). Aplicación de Realidad Aumentada para ANDROID Look Places. *(Proyecto Fin de Carrera).* Universidad Carlos III de Madrid, Madrid, España. Recuperado el 14 de Enero de 2022, de https://e-

archivo.uc3m.es/bitstream/handle/10016/17888/PFC-SSC-LookPlaces-Memoria.pdf?cv=1&isAllowed=y&sequence=1

Anexo 4: Certificado de Aprobación del Docente Tutor

Universidad Católica de Cuenca

CARRERA DE INGENIERIA EN SISTEMAS DE INFORMACION
EXTENSION CAÑAR

Oficio Nro.: UCACUE-SI-2022-08-OF
Cañar, 2 de agosto de 2022

Asunto: Aprobación trabajo de titulación.

Abg.
Cristian Gavilanes Barahona
SECRETARIO UNIVERSIDAD CATOLICA DE CUENCA, EXTENSION CAÑAR.
Su despacho. –

Por medio de la presente pongo en su conocimiento que he revisado el trabajo de titulación denominado: "Prototipo de un aplicativo móvil de geolocalización del recolector de desechos en base a su recorrido para los usuarios de la EMMAIPC-EP" de la estudiante **BRYAN STALIN SOLÓRZANO ESPINOZA**, con cedula de identidad No 0302146451, el cual cumple con los requisitos establecidos por lo tanto se aprueba y recibe la nota 50/50 en el trabajo escrito de titulación.

Situación que informo para los fines pertinentes.

Atentamente,
DIOS, PATRIA, CULTURA Y DESARROLLO

Ing. Luis Pinos Castillo, Msc.
DOCENTE - TUTOR

Universidad Católica de Cuenca EXTENSIÓN CAÑAR SECRETARIA
02 AGO 2022
RECIBIDO
HORA 20:55 FIRMA

Universidad Católica de Cuenca
EXTENSIÓN CAÑAR
SECRETARÍA

Anexo 5: Certificado de no Adeudar Libros a Biblioteca

Anexo 6: Certificado de Cumplimento del Trabajo de Titulación dentro de la EMMAIPC-EP

EMMAIPC
CAÑAR, BIBLIÁN, EL TAMBO Y SUSCAL

Empresa Pública Municipal Mancomunada de Aseo Integral de Cañar, Biblián, El Tambo y Suscal

Cañar, 27de Julio de 2022

ING. FRANKLIN VINICIO RIVERA MEDINA, GERENTE DE LA EMMAIPC-EP

A petición verbal de parte interesada.

CERTIFICO:

Que, el Sr. BRYAN STALIN SOLÓRZANO ESPINOZA, con número de cédula 0302146451, estudiante de la carrera de Ingeniería en Sistemas de la Información de la Universidad Católica de Cuenca Extensión Cañar, culminó su Trabajo de Titulación denominado "Prototipo de un aplicativo móvil de geolocalización del recolector de desechos en base a su recorrido para los usuarios de la EMMAIPC-EP".

Es cuanto puedo certificar en honor a la verdad, permitiendo al beneficiario hacer uso del presente para los fines que creyere conveniente.

Atentamente,

EMMAIPC-EP GERENCIA

Ing. Franklin Vinicio Rivera Medina
GERENTE DE LA EMMAIPC-EP

Dirección: Av.Ingapirca y Pozo de Chávez
Teléfono: 07 2427001
www.emmaipc-ep.gob.ec

Anexo 7: Certificado de Autorización de Publicación en el Repositorio Institucional

AUTORIZACIÓN DE PUBLICACIÓN EN EL REPOSITORIO INSTITUCIONAL

Yo, **Bryan Stalin Solórzano Espinoza** portador de la cédula de ciudadanía N° **0302146451**. En calidad de autor/a y titular de los derechos patrimoniales del trabajo de titulación **"PROTOTIPO DE UN APLICATIVO MÓVIL DE GEOLOCALIZACIÓN DEL RECOLECTOR DE DESECHOS EN BASE A SU RECORRIDO PARA LOS USUARIOS DE LA EMMAIPC-EP"** de conformidad a lo establecido en el artículo 114 Código Orgánico de la Economía Social de los Conocimientos, Creatividad e Innovación, reconozco a favor de la Universidad Católica de Cuenca una licencia gratuita, intransferible y no exclusiva para el uso no comercial de la obra, con fines estrictamente académicos y no comerciales. Autorizo además a la Universidad Católica de Cuenca, para que realice la publicación de éste trabajo de titulación en el Repositorio Institucional de conformidad a lo dispuesto en el artículo 144 de la Ley Orgánica de Educación Superior.

Cañar, **22 de Septiembre de 2022**

F: .....

Bryan Stalin Solórzano Espinoza

C.I. **0302146451**