UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

ESCUELA DE INGENIERÍA EN MECATRÓNICA

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

TEMA:

"INTERFAZ ABIERTA PARA SERVIDOR DE DATOS DE LOCALIZACIÓN"

AUTOR: PABLO GEOVANNY FLORES MORILLO

DIRECTOR: CARLOS XAVIER ROSERO CHANDI

IBARRA-ECUADOR
FEBRERO 2020

UNIVERSIDAD TÉCNICA DEL NORTE

BIBLIOTECA UNIVERSITARIA

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

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD	1003561287		
APELLIDOS Y NOMBRES	FLORES MORILLO PABLO GEOVANNY		
DIRECCIÓN	ECUADOR, IMBABURA, IBARRA, LOS GIRASOLES		
EMAIL	pgfloresm@utn.edu.ec		
TELÉFONO FIJO	062-512-768	**TELÉFONO MÓVIL**	0967050759
DATOS DE LA OBRA			
TÍTULO	"INTERFAZ ABIERTA PARA SERVIDOR DE DATOS DE LOCALZACIÓN"		
AUTOR	PABLO GEOVANNY FLORES MORILLO		
FECHA	ABRIL DE 2020		
PROGRAMA	PREGRADO		
TÍTULO POR EL QUE OPTA	INGENIERO EN MECATRÓNICA		
DIRECTOR	CARLOS XAVIER ROSERO CHANDI		

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

2. CONSTANCIAS

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

Ibarra, a los 27 días del mes abril de 2020

Pablo Geovanny Flores Morillo
C.I.: 1003561287

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN

En calidad de director del trabajo de grado "INTERFAZ ABIERTA PARA SERVIDOR DE DATOS DE LOCALZACIÓN", presentado por el egresado PABLO GEOVANNY FLORES MORILLO, para optar por el título de Ingeniero en Mecatrónica, certifico que el mencionado proyecto fue realizado bajo mi dirección.

Ibarra, a los 27 días del mes abril de 2020

Carlos Xavier Rosero Chandi
DIRECTOR DE TESIS

Agradecimiento

A Dios, por darme la vida y las fuerzas para seguir adelante. A Geovanny, Hilda y Santiago por todo el amor, consejo y esfuerzo para darme la oportunidad de cumplir este objetivo. A Evelyn, por su gran cariño y apoyo incondicional. A Xavier, por la ayuda, tiempo, guía y por compartir su conocimiento para el desarrollo de este proyecto.

Pablo Flores

Dedicatoria

El presente trabajo está dedicado a Dios, mi padre, mi madre, mi hermano y mi novia.

Pablo Flores

Resumen

El uso de un dispositivo GPS para monitoreo satelital o *tracking* es cada vez mas amplio y para el usuario final existen varias soluciones disponibles, por ejemplo, en Ecuador, casi la totalidad de estas son ofrecidas por empresas privadas que trabajan con herramientas de desarrollo cerradas, al igual que la plataforma que ofrecen. Esto representa un obstáculo para la investigación y elaboración de nuevas plataformas, ya que, los desarrolladores deben empezar un proyecto desde cero e invertir dinero en adquirir licencias del software para el desarrollo. Esto se traduce en un mayor costo para el usuario final o las entidades investigadoras. Es por eso que el uso de software libre y gratuito se convierte en una alternativa viable para reducir costos y facilitar el desarrollo y la investigación de nuevos proyectos de posicionamiento. Este trabajo detalla el desarrollo de una interfaz para servidor de datos de localización de código abierto que se realiza usando herramientas de software libre y gratuito como: el framework de desarrollo Django, el servidor de mapas OpenStreetMap y la librería de visualización de mapas *Leaflet*, entre otros. La interfaz se implementa sobre servidor virtual privado VPS y se encarga de gestionar la información satelital almacenada en una base de datos y plasmarla en un mapa, permitiendo ofrecer funciones como: mostrar los dispositivos que le han sido asignados a un usuario, visualizar las rutas recorridas de los mismos y mostrar información detallada de cada punto de toma de muestra.

Palabras clave: monitoreo satelital, código abierto, interfaz.

Abstract

The use of a GPS device for satellite monitoring or *tracking* is becoming increasingly widespread and for the end user there are several solutions available, for example, in Ecuador, almost all of these are offered by private companies that work with closed source development tools, as well as the platform they offer. This represents an obstacle for researching and elaborating of new platforms, since the developers must start a project from scratch and invest money in acquiring licenses for development software. Which translates into a higher cost for the end user and research institutions. This work details the development of an open source localization data server interface to be developed using free and open source software tools: the Django development framework, the OpenStreetMap map server and the Leaflet map visualization library, and other; the interface is implemented on a virtual private VPS server and it managing the satellite information stored in a database and translating it into a map, allowing to offer functions such as: displaying the devices that have been assigned to a user, visualizing the routes traveled by them and providing detailed information of each sampling point.

Keywords: tracking, opensource, inferface.

Índice general

Lista de figuras

Lista de tablas

Introducción

Este trabajo de grado se realizará con el *Grupo de Investigación en Sistemas Inteligentes de la Universidad Técnica del Norte (GISI-UTN)*.

Motivación

Actualmente el ser humano posee bienes que son importantes para su vida cotidiana por tener un valor significativo en su desarrollo socioeconómico y emocional. Sin embargo, no puede llevarlos consigo todo el tiempo, por lo que están propensos a extraviarse. Por esta razón, el propietario debe conocer la localización de estas posesiones para facilitar su recuperación en caso de extravío [10].

Existen sistemas de rastreo como el GPS (Sistema de Posicionamiento Global), que permiten obtener la posición en cualquier momento, lugar y condición climatológica [11]. Usando esta información se ha desarrollado software usando sistemas GPS para usos como: alternativas de rutas para ir de un punto a otro, localización y recorrido de vehículos, personas, animales u objetos [12], [1].

Las opciones de software existentes ofrecen servicios a personas y empresas, sin embargo, mayoritariamente son de pago y se han desarrollado en código cerrado [11]. Hasta el momento en Ecuador se han llevado a cabo varias investigaciones sobre *tracking* o rastreo de rutas utilizando software privado y alquiler de satélites, por lo tanto no existen interfaces abiertas y gratuitas disponibles para este fin [11].

Es posible ofrecer soluciones de localización y rastreo satelital basado en sistemas GPS, gratuitas y de código abierto, de manera que las personas interesadas en desarrollar software con aplicaciones similares puedan utilizar el mismo código libremente y sin pago [11].

Al estar disponibles aplicaciones gratuitas y abiertas, pueden ser útiles para todas las personas que las requieran, de lo contrario, si son cerradas y de alto costo, su uso es limitado. Las aplicaciones serán asequibles para personas que tengan la posibilidad de cubrir los altos costos para su uso. Además, el código cerrado representa una gran dificultad para quien desea desarrollar aplicaciones similares, ya que, su uso está restringido [13].

El presente proyecto desarrolla de una interfaz abierta de gestión de datos de localización con el fin administrar datos de posicionamiento guardados en una base de datos ubicada en la nube.

Objetivos

Objetivo general

- Desarrollar una interfaz abierta para servidor de datos de localización.

Objetivos específicos

- Implementar un servidor que alojará la interfaz.
- Determinar las funciones necesarias para la operatividad de la interfaz.
- Desarrollar la interfaz con todas las funciones necesarias para su funcionamiento.
- Validar el sistema completo bajo condiciones reales.

Alcance

La interfaz consiste en una aplicación Web cuyo código fuente estará almacenado en un servidor web denominado web host. Esta usará un software abierto que permita visualizar los mapas de la zona. La aplicación mostrará el mapa y permitirá conocer en todo momento la localización y trayectorias realizadas por el objetivo seleccionado previamente por el usuario. La interfaz logrará esto mediante herramientas gráficas como botones, pestañas, listas, entre otros; usará los datos obtenidos desde dispositivos GPS, fruto de los resultados de otros proyectos finales de ingeniería realizados en la carrera de Ingeniería en Mecatrónica.

Justificación

Actualmente, para la población existen herramientas disponibles que usan GPS para realizar tracking, que pueden ser aprovechadas por una gran cantidad de aplicaciones para diferentes usos. Sin embargo, estas herramientas representan un alto costo para acceder a ellas [14]. La importancia de este trabajo radica en la iniciativa de código abierto y software libre, con el fin de que cualquier persona resulte beneficiada por el uso sin costo y basado en que el software libre le pertenece a cada persona en el mundo, para usar, modificar y copiar [14]. Es un legado de la humanidad que no tiene propietario, así como las leyes de la física o las matemáticas. No existe un monopolio y no es necesario pagar por su uso [15].

Capítulo 1

Revisión literaria

Este capítulo muestra una revisión de varios sistemas que realizan una función similar a la del presente trabajo. Las partes que generalmente componen un sistema de posicionamiento satelital se muestran en la Figura 1.1, la cual se usará como referencia en el análisis, enfocándose principalmente en las partes que componen la interfaz.

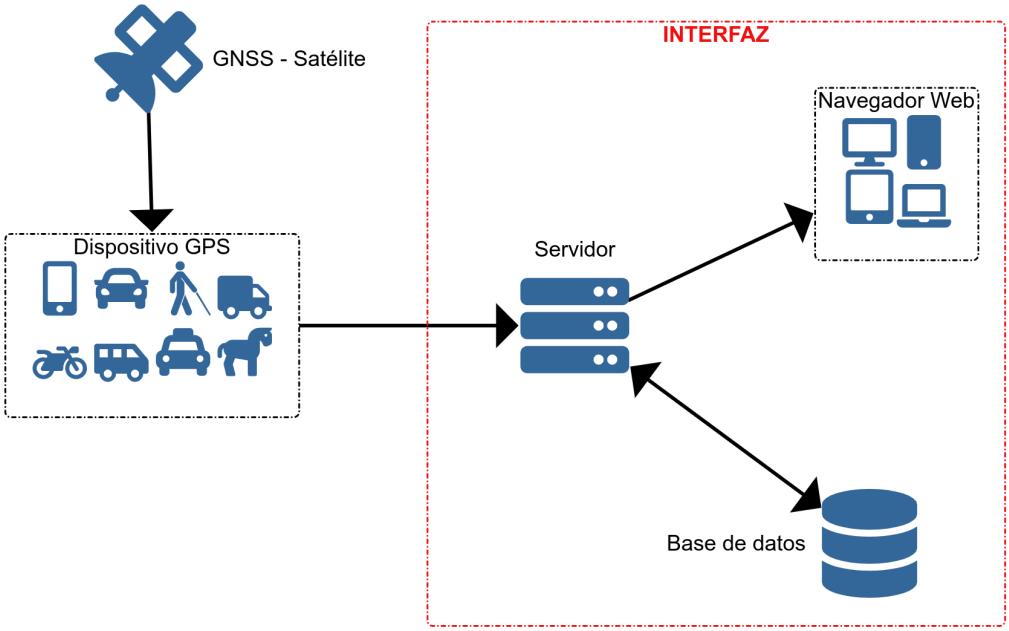


Figura 1.1: Diagrama del sistema de posicionamiento satelital

Uno de los componentes del sistema, es el GNSS[1], del cual, el mas usado es el sistema GPS Navstar, comúnmente llamado GPS[2]. Debido a su superioridad, la gran mayoría de proyectos desarrollados conocidos usan este sistema de posicionamiento satelital, incluidos los analizados en este apartado. Sin embargo, esto podría cambiar en un futuro debido a que GLONASS[3] ha mejorado su precisión, incluso llegando a compararse con la del GPS y también porque los sistemas Galileo[4] y BeiDou[5] están próximos a entrar en pleno funcionamiento [16].

Otro componente usado en los sitemas de posicionamiento y que no se integra en el presente proyecto es el módulo GPS/GSM que utiliza tecnología GPS o localización por GSM[6] para obtener su posición y graficarla en el mismo dispositivo (si este lo soporta). También pueden enviar la información para distintos propósitos como en el caso de los proyectos que a continuación se describen.

1.1 Desarrollo de una aplicación web orientada a servicios para el monitoreo de una flota de vehículos haciendo uso de la tecnología GPS

Esta investigación se desarrolló en la Universidad Nacional de San Antonio de Abad del Cusco, por Mary Conza, describe el desarrollo de una aplicación georeferencial, que tiene como propósito rastrear y monitorear una flota vehicular de taxis. La arquitectura del sistema se muestra en la Figura 1.2 [1].

El proyecto se ha desarrollado como una aplicación Web, ya que, se puede ejecutar en diferentes dispositivos sin necesidad de instalar algún software específico. El autor resalta algunas ventajas de su uso como: el aporte de interoperabilidad entre plataformas independientemente de sus propiedades o del sistema sobre las que se instalen; el fomento de estándares y protocolos basados en texto, los cuales hacen más fácil acceder a su contenido y entender su funcionamiento y la posibilidad de que diferentes compañías sen la misma aplicación web para integrarla parcialmente a su sistema y proveer servicios unificados.

[1] GNSS: Sistema global de navegación por satélite, Global Navigation Satellite System [16].
[2] GPS: Sistema de posicionamiento global, Global Positioning System [17].
[3] GLONASS: GNSS desarrollado por Rusia, Global Navigation Satellite System [16].
[4] Galileo: GNSS desarrollado por la Unión Europea [16].
[5] BeiDou: GNSS desarrollado por China [16].
[6] GSM: Sistema global para las comunicaciones móviles, Global System for Mobile communications.

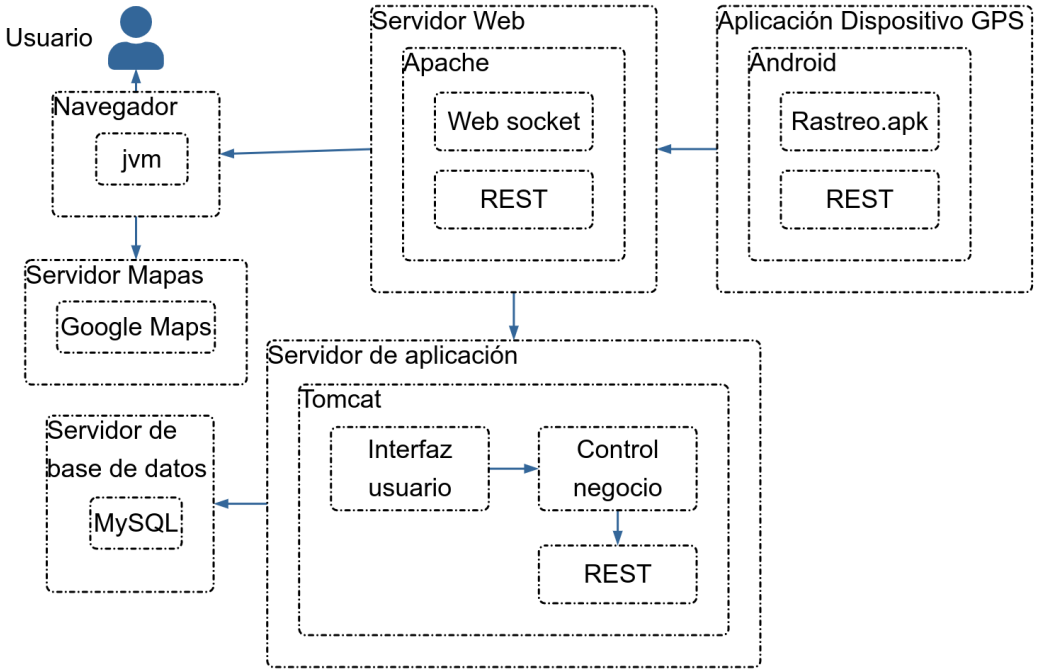


Figura 1.2: Diagrama del sistema de la flota de taxis [1]

1.1.1 Aplicación para el dispositivo GPS

Los taxis están equipados con receptores GPS que incluyen un módem inalámbrico, los cuales están en el dispositivo móvil Android, para el cual se desarrolló una APK[7] encargada de calcular su respectiva posición, hora y velocidad; dichos parámetros son enviados mediante la red celular GSM/GPRS[8] de manera automática a una estación central usando los servicios web REST[9]. Este es un estilo de arquitectura para sistemas hipermedia distribuidos como el internet. Para el desarrollo de esta aplicación se utilizó el software multiplataforma y de código abierto[10] Eclipse.

[7] APK: Paquete de Aplicación Android, Android Application Package.

[8] GPRS: Servicio general de paquetes vía radio, General Packet Radio Service.

[9] REST: Transferencia de Estado Representacional, Representational State Transfer.

[10] Código abierto: Software que ofrece acceso al código fuente, gratuidad, la posibilidad de evitar monopolios de software propietario y un modelo de avance, Open Source.

1.1.2 Servidor Web

Funciona bajo un servidor Apache que se comunica con la aplicación del dispositivo GPS mediante el formato JSON[11] usando los servicios Web REST y mediante Web sockets[12] permite el envío de datos directamente a las máquinas cliente para el rastreo de vehículos en tiempo real. El servidor es mediador entre el dispositivo GPS y el servidor de aplicación, proporcionando también las coordenadas que serán gestionadas por la interfaz.

El servidor está alojado en una VPC[13] que ofrece la empresa Amazon, en esta VPC se requiere fundamentalmente el software: Apache Tomcat 7.2, Mysql5.5 server y client, Jdk 7 (1.7) y PhpMyAdmin.

1.1.3 Servidor de aplicación

Este servidor es el encargado de gestionar la interfaz como tal, recibe los datos directamente del servidor web, proporcionados por la aplicación del GPS, se encarga de almacenar la información en la base de datos.

El servidor usado es Tomcat, que como tal, no es un servidor de aplicaciones. Sin embargo, trabajando junto a Apache, puede funcionar como un servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

1.1.4 Base de datos

La base de datos o DBMS[14] seleccionada es MySQL, la cual en realidad consiste en un sistema de gestión de bases de datos relacionales, multi-hilo y multi-usuario que se caracteriza por su facilidad de uso. Se adapta muy bien a la administración de datos en un entorno de red, especialmente en arquitecturas cliente-servidor [1]. Las mejores virtudes de MySQL son su fácil implementación con el lenguaje PHP y su buena compatibilidad con el servidor Apache, que en este caso resulta muy conveniente.

La herramienta de gestión de bases de datos usada es MySQL Workbench, que brinda la posibilidad de manipular las con una interfaz gráfica. En algunos casos esto resulta una una ventaja para la gestión realizada el desarrollador.

1.1.5 Mapa

Para poder visualizar los datos proporcionados por el dispositivo GPS es necesario un mapa, y el servicio de mapas seleccionado en este proyecto es Google Maps que ofrece también el

[11] JSON: Formato de texto sencillo para el intercambio de datos. Objeto de JavaScript, JavaScript Object Notation.

[12] Web sockets: Protocolo para la web bajo TCP que proporciona un canal de comunicación bidireccional y full-dúplex sobre un único socket TCP.

[13] VPC: Servidor virtual privado alojado en la nube, Virtual private cloud.

[14] DBMS: Sistema de administración de base de datos, es un software que se utiliza para crear y manipular bases de datos, Database management system.

servicio de imágenes vía satélite de todo el planeta, es el mas conocido mundialmente y el mas usado en proyectos de geolocalización.

Google maps trabaja implementando lenguajes como HTML[15], CSS[16] y JavaScript, los mapas son imágenes que se cargan en el fondo a través de peticiones ejecutadas por la tecnología de AJAX[17], y se insertan en un <*div*> en la página HTML de la interfaz.

1.2 Desarrollo de un sistema de localización de vehículos por GPS y GSM/GPRS

El desarrollo de este sistema se llevó a cabo en la Universidad Politécnica de Madrid por Ignacio García, el trabajo pretende que los usuarios puedan localizar a sus vehículos mediante GPS, visualizar el historial mediante un mapa en una página web [2].

El sistema Keep Track, como el autor lo ha nombrado, se divide en dos partes, el dispositivo GPS y la aplicación del servidor que realiza las gestiones en la base de datos y muestra la página web a través de la cual el usuario interactúa con el sistema.

1.2.1 Hardware

El microcontrolador elegido es la placa Arduino Uno debido a la facilidad de uso que esta posee para crear un prototipo, además se ha decidido usar la placa Adafruit FONA 808, la razón principal por la que esta se ha escogido es por el nivel de desarrollo de las librerías.

1.2.2 Aplicación del servidor

Para el desarrollo del servidor se ha escogido Symfony como framework[18], este trabaja con el lenguaje PHP. Tras el desarrollo de la aplicación del servidor se ha montado en un servidor Amazon Web Services donde se instaló el servidor LAMP[19] en el que se realizaron las pruebas.

El gestor de plantillas para las vistas seleccionado es Twig que es el más recomendado para utilizar en conjunto con Symfony. Ofrece la posibilidad de crear plantillas que después se modifican y rellenan en el servidor con datos para un usuario antes de mostrar la vista.

[15]HTML: Lenguaje de marcado para la elaboración de páginas web, Hypertext Markup Language.

[16]CSS: Usado para establecer el diseño visual de los documentos web, Cascading Style Sheets.

[17]AJAX: JavaScript asíncrono y XML, técnica de desarrollo cuya principal utilidad es la de poder realizar cambios sobre las páginas o archivos AJAX sin necesidad de recargar la página.

[18]Entorno de trabajo: Estructura conceptual y tecnológica de asistencia definida, normalmente, con artefactos o módulos concretos de software, que puede servir de base para la organización y desarrollo de software.

[19]LAMP: Servidor Linux Apache MySQL PHP

1.2.3 Base de datos

El DBMS que se ha elegido es MySQL, por su facilidad de instalación e integración con el software Symfony, donde es posible utilizar un ORM[20] llamado Doctrine que facilita la creación de nuevas entradas en las tablas acelerando el tiempo de desarrollo de la interfaz.

Los datos almacenados en la MySQL se manejaron de acuerdo al modelo entidad relación. El formato tanto para el usuario, el historial y el tracker se pueden apreciar en la Figura 1.3.

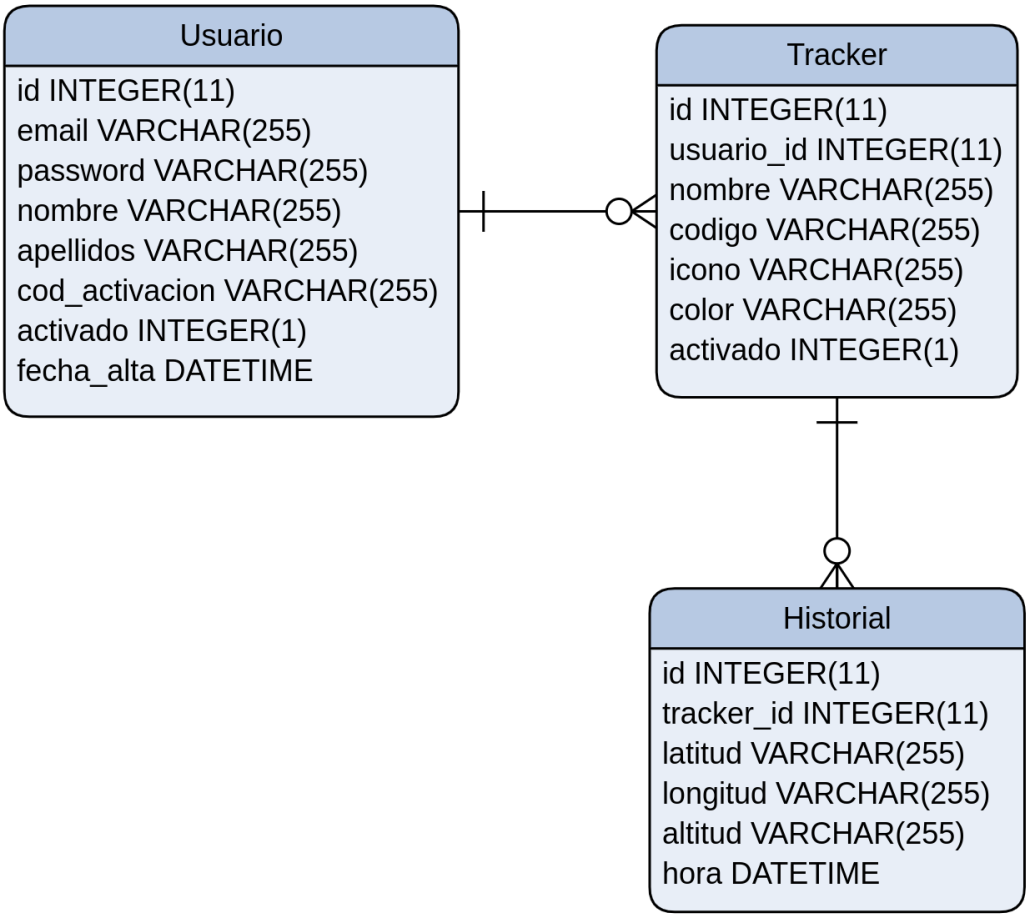


Figura 1.3: Modelo entidad relación del sistema [2]

[20]ORM: Técnica de programación para convertir datos entre el sistema de tipos utilizado en un lenguaje de programación orientado a objetos y la base de datos relacional como motor de persistencia, Object-relational mapping.

1.2.4 Mapa

Para poder mostrar el historial de un localizador se implementó un mapa, en este caso se ha usado la API[21] de Google Maps y las marcas de los localizadores se han creado en JavaScript implementando una interfaz proporcionada por Google. El autor comenta que el desarrollo fue lento debido a la poca documentación disponible y la falta de ejemplos tan complejos como el que se quería desarrollar.

Además el autor pone a conocimiento su afinidad con el software libre y pone a disposición su proyecto en un repositorio alojado en GitHub para que cualquier persona pueda hacer uso del mismo, implementando cambios para que el proyecto siga creciendo y siendo útil para todos los usuarios.

1.3 Desarrollo de un sistema para monitoreo y control satelital de vehículos mediante el uso del dispositivo GPS TK 303G para la comercializadora de dispositivos satelitales GENIUS EC

Este proyecto, realizado por Cristian Parra, se desarrolló en la Universidad de las Fuerzas Armadas del Ecuador, trata de un sistema de monitoreo satelital para obtener la ubicación geográfica de dispositivos GPS en tiempo real, se toma como base una plataforma de distribución libre que se ejecuta bajo un sistema operativo Linux, plataforma que se adaptó a las necesidades del proyecto [8]. En la Tabla 1.1, el autor realiza una comparación de las tres mejores opciones de software libre disponibles. En base en esto selecciona *Traccar*, misma que se adapta para su sistema.

[21] API: Conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción, Application programming interface.

Tabla 1.1: Herramientas de software libre [8]

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*

1.3.1 Hardware

El hardware seleccionado para proporcionar las coordenadas es el dispositivo GPS TK 303G que trabaja bajo dos posibles modalidades GSM o GPRS, para ello se ha debido realizar diferentes configuraciones después de la instalación, en adición a este dispositivo se trabajó con un teléfono Android con un módulo GPS integrado.

1.3.2 Traccar

Traccar es una plataforma de localización por GPS de código abierto (Free and Open Source) compatible con varias marcas que fabrican equipos de localización GPS; Traccar ofrece Traccar Server, es una plataforma completa de rastreo GPS, la arquitectura del sistema se muestra en la Figura 1.4 [3].

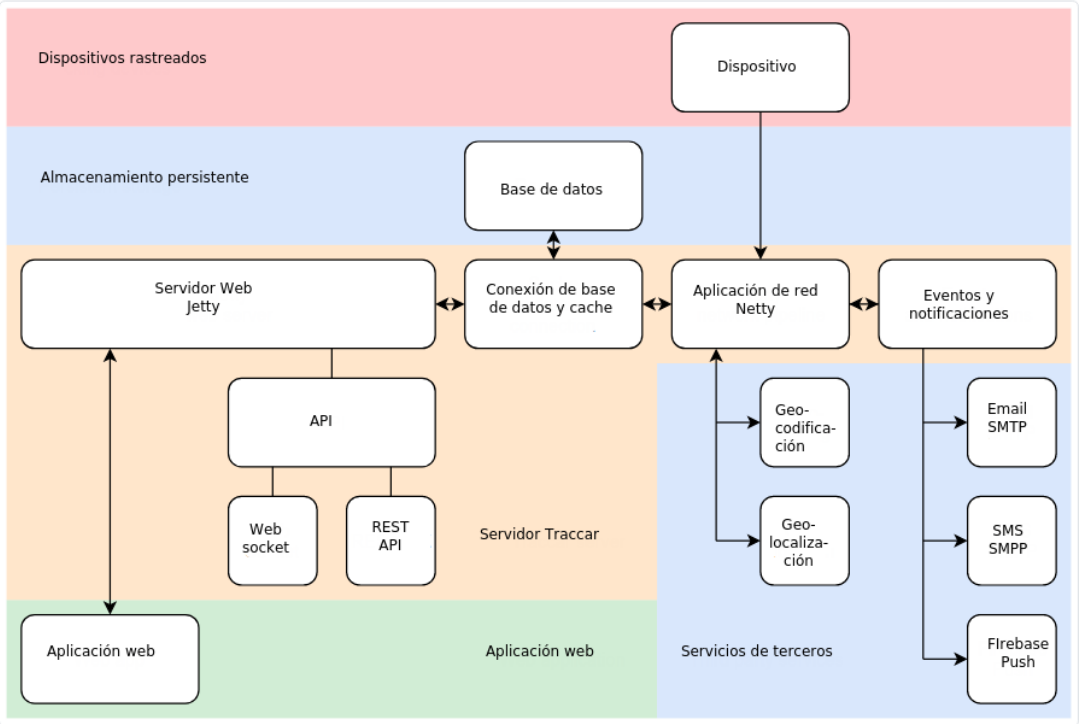


Figura 1.4: Arquitectura del sistema Traccar [3]

Las aplicaciones que Traccar pone a disposición son las siguientes:

- Traccar Client
- Traccar Manager

Todas las aplicaciones de Traccar son de código abierto. El código fuente está disponible en GitHub bajo la licencia Apache License 2.0.

Traccar Client

Esta solución incluye una aplicación denominada Traccar Client que permite usar como un módulo GPS a cualquier teléfono móvil con Android o iOS y se puede integrar en el sistema completo, adicionalmente Traccar permite usar módulos GPS comerciales, soportando mas de 1000 modelos. De forma predeterminada, las aplicaciones de Traccar Client están configuradas para usar el servidor gratis de demostración de Traccar (dirección - demo.traccar.org, puerto - 5055). Traccar Client es totalmente compatible con el servidor Traccar y puede utilizarlo con su propia servidor Traccar. Además otras plataformas de rastreo GPS populares también son compatibles con las aplicaciones Traccar Client [3].

Traccar Manager

Traccar manager es otro producto ofrecido que sirve para usar cualquier dispositivo móvil con Android o iOS para administrar los dispositivos de rastreo GPS conectados al sistema. Requiere un servidor traccar para funcionar. Por defecto, la aplicación está configurada para utilizar el servicio gratuito de demostración de Traccar.Traccar Manager muestra la pantalla de selección del servidor sólo en el primer inicio. Adicionalmente estas funciones también se pueden directamente desde el software de la aplicación web para un computador [3].

1.3.3 Servidor Web

El sistema Traccar incluye un servidor web Jetty incorporado. Jetty es una implementación estable y madura del servidor HTTP de Java y el contenedor de Servlet[22]. Se utiliza en muchos productos y marcos populares. El servidor web posee dos componentes principales [3].

- Web API
- Aplicación Web

Web API

La interfaz consta de las dos partes principales:

- REST API
- WebSocket

REST API se implementa usando el estándar Java EE API para los servicios web de RESTful. Cada modelo del sistema tiene su propio punto final dedicado.

Aplicación Web

El sistema Traccar incluye la aplicación web para la gestión de usuarios, dispositivos y otros elementos. La aplicación depende directamente de la Web API descrita anteriormente. Está basada en Sencha ExtJS framework y usa OpenLayers para la visualización de mapas.

1.3.4 Base de datos

El servidor se puede configurar para funcionar con casi cualquier motor de base de datos SQL, incluidas opciones populares como MySQL, PostgreSQL, Microsoft SQL Server y Oracle. En este caso reemplazan la base de datos H2, integrada por defecto en Traccar, y se ha configurado a MySQL en su lugar [8].

[22]Servlet: Clase en el lenguaje de programación Java, utilizados comúnmente para extender las aplicaciones alojadas por servidores web.

1.4 Propuesta

Los proyectos analizados en el apartado anterior son una muestra de la gran variedad de sistemas de localización que existen. Estos muestran sus características más relevantes. Varios proyectos han sido realizados para empresas privadas, por lo tanto la información sobre su metodología de desarrollo y código fuente podría no estar disponible para el público en general, como es el caso de [1], que no menciona si su software está disponible ni si es de código abierto. En cuanto a la visualización de mapas, todos los proyectos usan la herramienta Google Maps, la cual soporta, entre muchas otras acciones, la de graficar polilíneas, polígonos o círculos que permiten realizar rutas o geocercas útiles para trabajos de localización. El proyecto [2] ha usado la herramienta de desarrollo Symfony debido a que al igual que otros frameworks posee la ventaja de ahorrar recursos y tiempo de desarrollo.

El presente proyecto plantea una solución diferente que consiste en el desarrollo de una interfaz para servidor de datos de localización que se alojará en un VPS[23], será de código abierto y estará disponible en un repositorio usando herramientas de desarrollo como Django, que no ha sido usada en ningún proyecto. Trabajará con componentes como OpenStreetMap que tampoco ha sido usado a pesar de que en integración con Leaflet puede llegar a tener algunas características iguales o mejores que Google Maps.

[23] VPS: Servidor virtual privado, Virtual private server.

Capítulo 2

Metodología

2.1 Descripción del sistema

El sistema en general brinda al usuario la posibilidad de visualizar sus dispositivos registrados, el historial de días recorridos y el mapa con la ruta realizada de los mismos. El usuario puede acceder al sistema mediante una aplicación para dispositivos móviles o una aplicación web por medio de un navegador, las aplicaciones se encargarán de mostrar la información gestionada por el servidor y almacenada en la base de datos accediendo a estas por medio de una conexión a internet. El servidor y la base de datos están ubicados en una nube virtual o VPS; estos reciben la información posicional desde un dispositivo GPS que envía información como latitud, longitud, altitud, entre otros datos que son obtenidos directamente de los satélites. La topología descrita se muestra a continuación en la Figura 2.1.

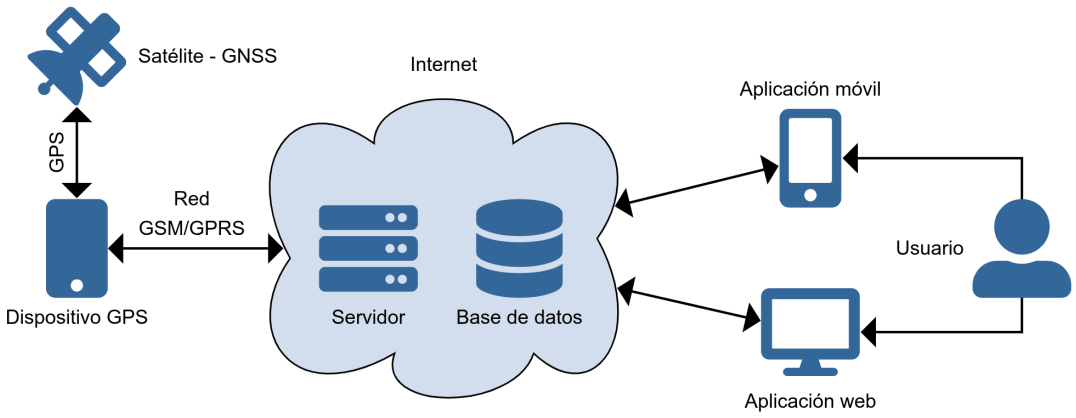


Figura 2.1: Topología general del sistema

A diferencia de los usuarios registrados, los usuarios no registrados no pueden acceder a esta información. Por otra parte, el usuario administrador tiene la capacidad de agregar, modificar

y eliminar usuarios, dispositivos y días de recorrido, las funciones de los diferente tipos de usuarios se detallan en la Figura 2.2.

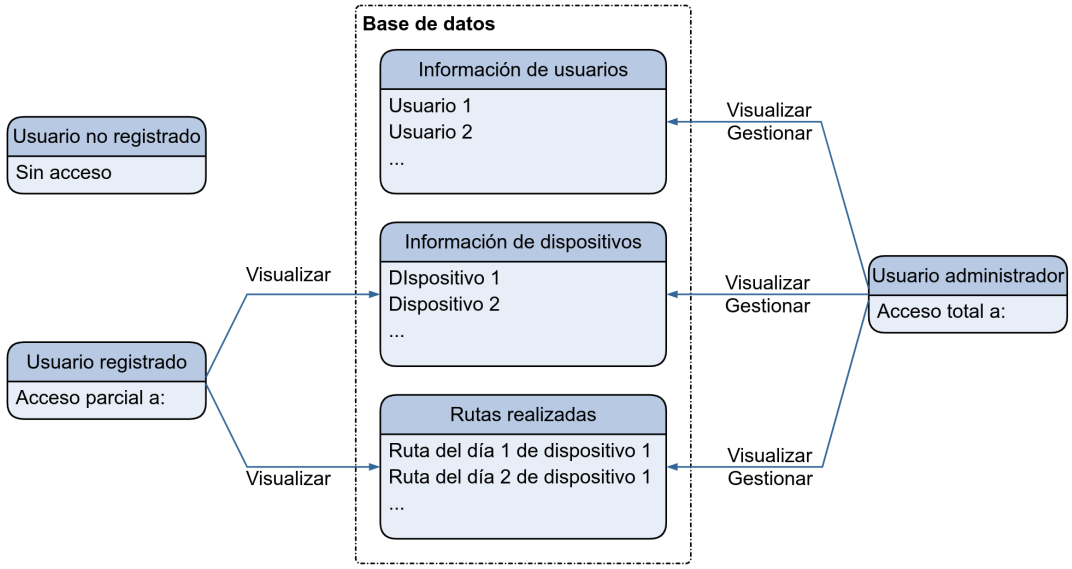


Figura 2.2: Acceso de los usuarios en el sistema

La interfaz para el servidor de localización está integrada por tres componentes que se muestran en la Figura 2.3 y son los siguientes:

- **Entorno web:** Contiene toda la estructura principal del sistema.
- **Base de datos:** Gestiona la información de usuarios, dispositivos y datos de posicionamiento.
- **Geovisualización:** Todas las funciones y características pertinentes a la visualización del mapa y las rutas pertenecen a esta sección.

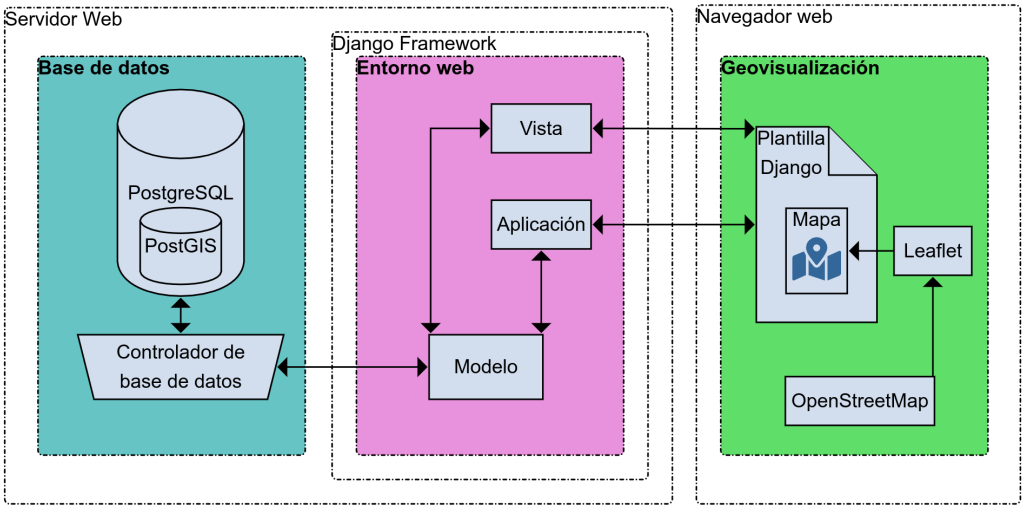


Figura 2.3: Diagrama general del sistema

2.1.1 Entorno web

Django es un framework web de alto nivel hecho con Python, promueve un rápido desarrollo y un diseño limpio y pragmático. Su principal ventaja es que ahorra tiempo, ya que, se encarga de gran parte del desarrollo web y el programador se puede enfocar en escribir la aplicación. Puede funcionar en cualquier sistema operativo que soporte Python, es gratuito y de código abierto [9]. Django posee una arquitectura MVT[1], como se aprecia en la Figura 2.4 [4]

Django tiene un módulo integrado que lo convierte en un web framework geográfico, lleva por nombre GeoDjango y se esfuerza por hacer lo más simple posible la creación de aplicaciones Web geográficas, como los servicios basados en la localización [9].

La plataforma de implementación principal de Django es *WSGI*[2], que es el estándar *python* para servidores web y apliaciones, puede trabajar con varios compiladores *WSGI*, entre ellos Apache, Gunicornio o uWSGI [9]. Estos servidores se ocupan de manejar todas las solicitudes entrantes como por ejemplo la de los usuarios al querer ingresar a la página.

Una instalación Django con algunos ajustes, se considera un proyecto. Django permite agregar aplicaciones que son un grupo de modelos, vistas, plantillas y URLs. Las aplicaciones interactúan con el framework para proporcionar algunas funcionalidades específicas y puede ser reutilizado en diversos proyectos. Se puede definir un proyecto como un sitio web, que contiene varias aplicaciones como blog, wiki o foro, que puede ser usado por otros proyectos también [18]. En este caso todo el sistema se ha implementado en un solo proyecto de Django que a su vez está contenido en la aplicación llamada *track*.

[1]MVT: Modelo-Vista-Plantilla, Model-View-Template.

[2]WSGI: Interfaz de la puerta de enlace del servidor web, Web Server Gateway Interface.

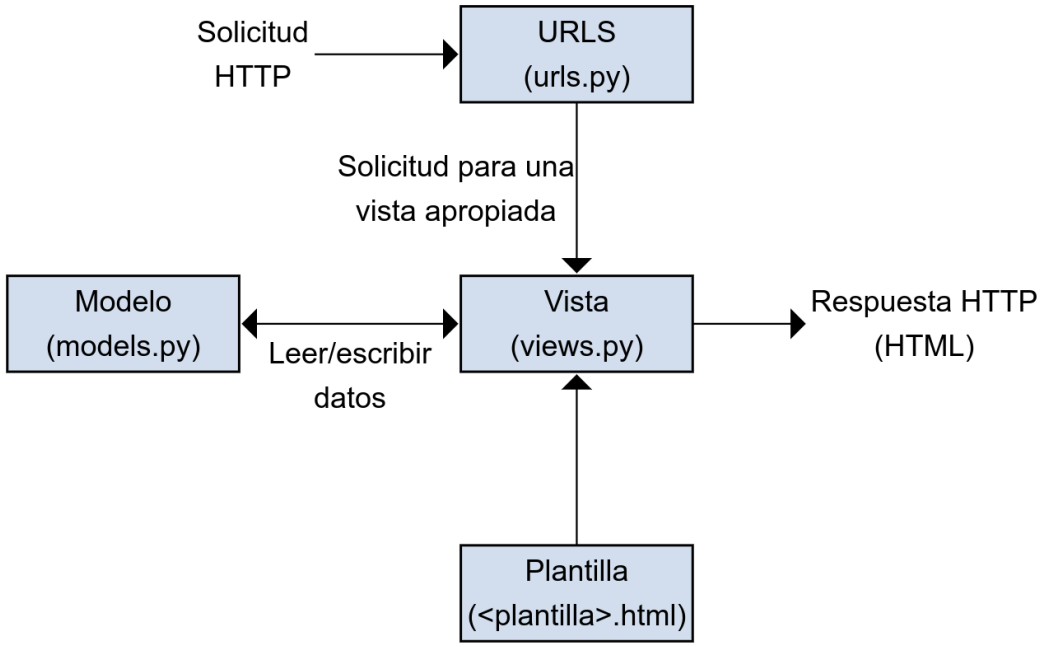


Figura 2.4: Arquitectura del framework Django [4]

Modelos

Un modelo es una clase Python que subclasifica *django.db.models.Model*, en el cual cada atributo representa un campo de la base de datos. Django crea una tabla por cada modelo definido en el archivo *models.py*. Al crear un modelo, Django proporciona una práctica API para consultar fácilmente los objetos en la base de datos [18].

En el sistema se agregaron varios modelos como:

- *Device*, que posee los atributos para ingresar el nombre de usuario, descripción e IMEI[3].
- *Reg*, registra el día al que se le asignó un historial, este está ligado a un usuario y dispositivo específicos.
- La información de posicionamiento de cada día de los dispositivos se almacenó en la base de datos por medio de un modelo para cada día de recorrido.

Vistas

Una vista es una función de gestión que recibe peticiones HTTP y devuelve respuestas HTTP. Las vistas acceden a los datos necesarios para satisfacer las peticiones por medio de modelos, y entregan el formato de la respuesta a las plantillas HTML [4].

[3]IMEI: Identidad internacional de equipo móvil, Inglés International Mobile Equipment Identity.

En las vistas es posible elegir que mostrar en las plantillas, entonces se tomaron varias características como:

- Aplicar filtros para discriminar los datos de la base que no se requieran en la plantilla.
- Se gestionó los permisos para que ciertas vistas no estén disponibles a determinados usuarios, por ejemplo, un usuario que no podrá acceder al historial de un dispositivo que no le pertenece.
- Se restringió el acceso a visualizar dispositivos e historiales a usuarios que no estén registrados.

Plantillas

Una plantilla es un archivo de texto que define la estructura común de una página HTML, con marcadores de posición que se utilizan para representar el contenido real. Una vista puede dinámicamente crear una página usando solo una plantilla, rellenándola con datos del modelo. Se puede usar una plantilla para definir la estructura de cualquier tipo de archivo, sin necesidad de que sea HTML [4].

Se usaron plantillas para mostrar contenido web como:

- La portada principal de la página web.
- El esqueleto completo de la página.
- La pestaña de visualizar dispositivos, días disponibles
- La pantalla de inicio de sesión.
- El mapa integrado que muestra el historial.

2.1.2 Base de datos

Para seleccionar la base de datos del presente proyecto se tomaron en cuenta las que son soportadas por el framework GeoDjango, estas se pueden ver en la Tabla 2.1, las mismas que son las mas usadas en el desarrollo de aplicaciones Web incluyendo los proyectos analizados en el capítulo anterior.

Tabla 2.1: Bases de datos soportadas por GeoDjango [9]

Base de datos	**Librerías requeridas**	**Versiones soportadas**	**Notas**
PostgreSQL	GEOS, GDAL, PROJ.4, PostGIS	9.4+	Requiere PostGIS.
MySQL	GEOS, GDAL	5.6+	No compila OGC; funcionalidad limitada.
Oracle	GEOS, GDAL	12.1+	XE no es soportado.
SQLite	GEOS, GDAL, PROJ.4, SpatiaLite	3.8.3+	Requiere SpatiaLite 4.1+

Django recomienda la base de datos PostGIS con PostgreSQL, porque soporta datos espaciales, es de código abierto, más madura y con más características en comparación con las otras, por lo tanto esta fue la que se utilizó en este proyecto.

En la base de datos se almacenan tanto los datos espaciales con los campos que se muestran en la Tabla 2.2 como los datos que se envían desde el modelo correspondiente a los datos del dispositivo registrado, día registrado y información de registro de usuarios.

La capacidad de almacenamiento de dispositivos, días o usuarios está limitada directamente por la base de datos, que según su página oficial [19], no tiene límite, por lo tanto se ve limitada por el desempeño del servidor web *WSGI*. De todas maneras estan cantidades no son bajas ya que en sitios que usan Django como *Instagram* o *Bitbucket* tienen alrededor de 300 y 6 millones de usuarios activos diarios respectivamente según la agencia [20].

Tabla 2.2: Registro de datos PostgreSQL

id	**Timestamp**	**Latitude**	**Longitude**	**Altitude**	**Distance**
1	2019-07-17T17:58:04Z	0.32857683	-78.1109563	2307	0.20
2	2019-07-17T17:58:02Z	0.32857888	-78.11097054	2307	1.60
3	2019-07-17T17:58:01Z	0.32857641	-78.11098746	2306	1.90
4	2019-07-17T17:57:58Z	0.32857103	-78.11105763	2306	7.83
5	2019-07-17T17:57:57Z	0.32856588	-78.11108161	2304	2.73

Se pueden almacenar datos en PostgreSQL de las siguientes formas:

- Por medio de la pagina web enviando los datos al módulo y luego a la interfaz, por ejemplo al registrar un dispositivo.
- Con el ORM de Django, este permite interactuar con la base de datos por medio de la shell de Django para crear, editar, eliminar datos, entre otras funciones.

- A través de la shell de Postgres(Si la base de datos es PostgreSQL), al igual que la opción anterior, se puede gestionar los datos mediante esta consola por medio del lenguaje SQL.
- Si existe un software de gestión visual para la base de datos, por ejemplo pgAdmin para PostgreSQL, MySQL Workbench para MySQL.

2.1.3 Geovisualización

Para la geovisualización se usa Leaflet, una biblioteca JavaScript de código abierto para mapas interactivos aptos para dispositivos móviles [5], entre sus principales características destacan su simplicidad, rendimiento y usabilidad. OpenStreetMap es el software que se encarga de mostrar el mapa de la Figura [21], integrado con Leaflet puede soportar muchas características importantes como:

- Soporte de capas como crear líneas, polilíneas, polígonos y otros.
- Funciones de personalización con CSS, marcadores HTML e imagen.
- Soporta casi todos los navegadores web.
- Características interactivas como zoom, arrastre con inercia entre otros.
- Efectos visuales como zoom animado, capas y ventanas emergentes animadas.
- Control de mapas con botones, casilleros para zoom, habilitar capas, entre otros.

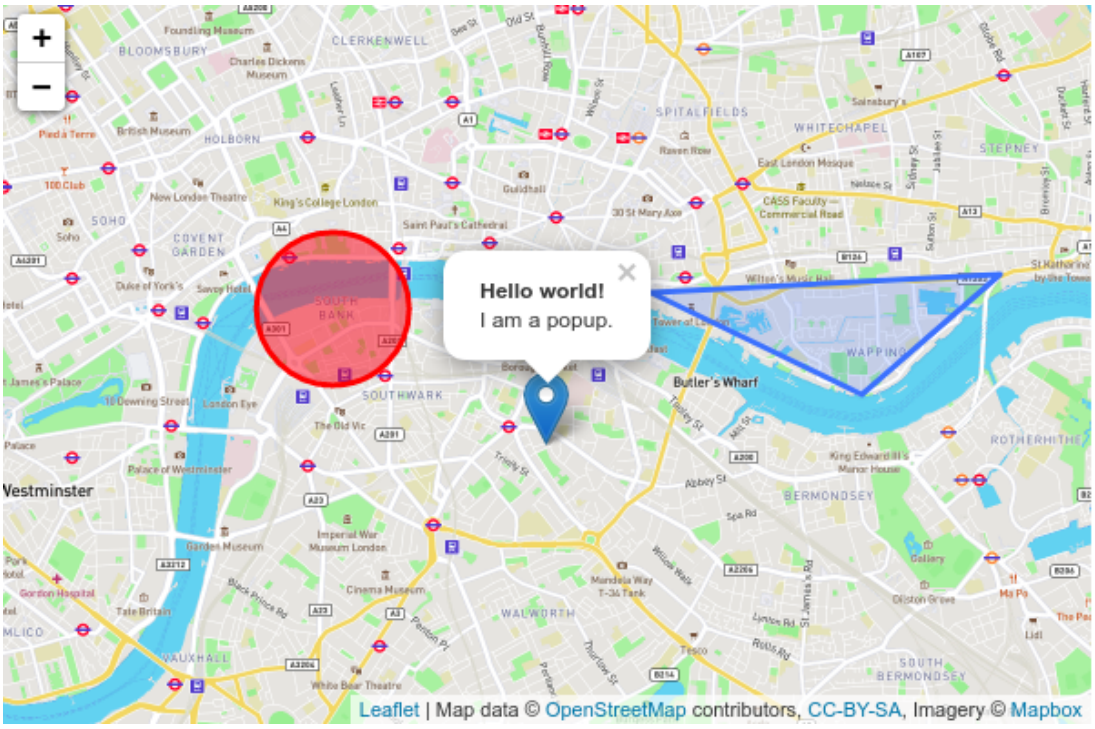


Figura 2.5: Mapa de Leaflet con OpenStreetMap [5]

Para el presente sistema, OpenStreetMap y Leaflet se encargan de varias funciones en la plantilla:

- Mostrar el mapa centrado con la ruta recorrida total del día seleccionado.
- Botones para modificar el zoom a la comodidad del usuario.
- Implementar marcadores para cada punto de datos de coordenadas.
- Casillero para habilitar la capa de marcadores para cada punto de datos almacenado.
- Ventana emergente al hacer clic para mostrar información detallada de cada marcador.

2.2 Implementación

En esta sección se detalla paso a paso como se fue desarrollado el sistema, desde acceder al servidor virtual privado, implementar el software hasta la visualización del mapa. El proyecto está disponible en el repositorio *https://github.com/pablogeovanny/OpenGPS*.

2.2.1 Acceso al servidor VPS

El sistema se ha implementado en un servidor VPS con las características siguientes:

Tabla 2.3: Características del VPS

Sistema operativo	Ubuntu 16.04 LTS
Almacenamiento RAID-10	20 GB
Núcleos	1
Transferencia	2 TB
Memoria RAM	512 MB

Para acceder remotamente a la VPS contratada, en este caso bajo el servicio de A2Hosting, se ingresa mediante SSH[4] que permite el acceso remoto por medio de un canal seguro en el que toda la información está cifrada. Se crea un usuario y se habilita al mismo los accesos de superusuario.

```
adduser nombredeusuario
apt-get install sudo
usermod -a -G sudo nombredeusuario
```

Para acceder al servidor por medio del usuario creado y con la dirección IP o el dominio adquirido, se ejecuta el siguiente comando en la terminal del sistema y se escribe la contraseña.

```
ssh -p 7822 nombredeusuario@direcciónIPodominnio
```

Este proceso varia dependiendo del sistema operativo del servidor VPS o los puertos que el proveedor del servicio proporcione.

2.2.2 Instalación del software

Para evitar conflictos entre las versiones del software que dependen de Python en el sistema y ejecutar Django en un entorno aislado, se instala el software *virtualenv* mediante *pip* y se crea el entorno virtual.

```
sudo pip3 install virtualenv
virtualenv env
source env/bin/activate
```

[4]SSH: Acceso seguro por terminal, Secure Shell.

Ahora ya la terminal se encuentra en el entorno virtual y se instala el software Django.

```
sudo pip3 install django
```

PostgreSQL se instala desde los repositorios para los sistemas que esté disponible o desde la página oficial.

```
sudo apt-get install libpq-dev python-dev
sudo apt-get install postgresql postgresql-contrib
sudo pip3 install psycopg2==2.7.4
```

Se instala el complemento de la base de datos PostGIS y otras librerías requeridas para un pleno funcionamiento. Las librerías requeridas son GEOS, GDAL, PROJ4, gcc, gmake, LibXML2, JSON-C y LLVM. Las librerías opcionales son GTK, SFCGAL, CUnit, DocBook, DBLatex y ImageMagick. Para instalar PostGIS desde el archivo fuente se ejecuta:

```
tar xvfz postgis-3.0.1dev.tar.gz
cd postgis-3.0.1dev
./configure
make
make install
```

Una vez instaladas todas las librerías de dependencia, el entorno está listo para la configuración y el desarrollo web.

2.2.3 Configuración de PostgreSQL

Se puede configurar la base de datos fácilmente desde *pgAdmin* en el caso de trabajar con un entorno gráfico, pero al estar ejecutando el sistema en SSH, se realiza el proceso por medio de la terminal. Accediendo a *PostgreSQL* con el nombre se superusuario *postgres* de la siguiente forma:

```
sudo su postgres
psql
```

Ya dentro de *psql* se debe crear la base de datos asignada al usuario *postgres* con el siguiente comando:

```
CREATE DATABASE nombredelabasededatos
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'en_US.UTF-8'
    LC_CTYPE = 'en_US.UTF-8'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;
```

Opcionalmente se procede a cambiar la contraseña al superusuario denominado *postgres*.

```
ALTER USER postgres WITH PASSWORD 'test123';
```

Se habilita la extension PostGIS en la base de datos con el comando:

```
CREATE EXTENSION postgis;
```

2.2.4 Proyecto web

Django proporciona un comando que permite crear la estructura inicial del proyecto, este debe ser ejecutado desde la shell en la ubicación en la que se quiera crear el proyecto.

```
django-admin startproject nombredelproyecto
```

La estructura del proyecto se establece de la siguiente forma:

```
nombredelproyecto/
    manage.py
    nombredelproyecto/
        __init__.py
        settings.py
        urls.py
        wsgi.py
```

Se accede a la carpeta contenedora del proyecto y se crea un usuario para usar todas las funciones de Django de la siguiente forma:

```
cd nombredelproyecto
python3 manage.py createsuperuser
```

El proyecto debe conectarse con la base de datos, para esto se debe modificar el código del archivo *settings.py* colocando la información respectiva:

settings.py

```
DATABASES = {
    'default': {
        'ENGINE': 'django.contrib.gis.db.backends.postgis',
        'NAME': 'nombredelabasededatos',
        'USER': 'postgres',
        'PASSWORD': 'test123',
        'HOST': 'localhost',
        'PORT': '5432',
     }
     }
```

Se debe también incluir a la extensión PostGIS en las aplicaciones en el mismo archivo de la siguiente manera:

settings.py

```
INSTALLED_APPS = [
    'django.contrib.gis',
    '...'
    ]
```

Se configura el host, dominio o dirección IP adquiridos desde donde se podrá acceder vía web al proyecto, en este caso se ha colocado la dirección IP modificando el archivo *settings.py*.

settings.py

```
ALLOWED_HOSTS = [68.66.235.160:8000']
```

Se usa CSS para modificar el aspecto de la interfaz por lo tanto se debe agregar también el soporte para archivos estáticos.

settings.py

```
STATIC_URL = '/static/'
```

Al crear el proyecto de crearon varios modelos que se deben agregar a la base de datos esto se realiza ejecutando el siguiente comando desde el directorio que contiene *manage.py*.

```
python3 manage.py makemigrations
python3 manage.py migrate
```

Cada vez que se modifiquen los modelos es necesario ejecutar estos comandos para sincronizar los cambios a la base de datos. Ahora el proyecto puede ejecutarse aunque solo tenga una estructura básica, se ejecuta el servidor web de desarrollo llamando al comando runserver (en el mismo directorio donde está *manage.py*).

```
python3 manage.py runserver 68.66.235.160:8000
```

Una vez ejecutado el comando, en la terminal se visualiza lo siguiente:

```
System check identified no issues (0 silenced).
January 31, 2020 - 20:41:46
Django version 2.2.6, using settings 'opengps6.settings'
Startin development server at http://68.66.235.160:8000/
Quit the server with CONTROL-C.
```

Al acceder desde un navegador a la dirección IP configurada *68.66.235.160:8000* se podrá confirmar que el proyecto se ha implementado correctamente al visualizar un mensaje de django informando de que la instalación se realizó correctamente.

Posteriormente se detiene el servidor y dentro del proyecto se crea la aplicación que contendrá la interfaz que en este caso se denomina *track*.

```
python3 manage.py startapp track
```

La aplicación creada debe ser incluida en la sección de aplicaciones en *settings.py*.

settings.py

```
INSTALLED_APPS = [
    'django.contrib.gis',
    'track.apps.TrackConfig',
]
```

Esquema de datos del sistema

Los primeros datos que son almacenados en la base de datos son los de la información de los usuarios que se crean iniciando con la cuenta administrador en página de administración de Django. En la Tabla 2.4 se muestra cierta información de los usuarios registrados.

Tabla 2.4: Información de los usuarios registrados

username char varyng(150)	**first_tname** char varyng(30)	**last_name** char varyng(150)	**email** char varyng(254)
Katy	Katy	Paz	katy@utn.edu.ec
Ana	Ana	Cano	ana@utn.edu.ec
Superlimpio	Super	Limpio	info@superlimpio.com
Ventasexpres	Ventas	Ventas	info@ventasexpres.com
Juan	Juan	Alba	juan@utn.edu.ec
Pablo			pgfloresm@utn.edu.ec

Para agregar modelos se modifica el archivo *models.py*, el modelo *Device* se crea agregando las siguientes líneas:

models.py

```
from django.db import models
from django.contrib.gis.db import models as geomodels
from django.contrib.auth.models import User
from django.urls import reverse

class Device(models.Model):
    IMEI = models.CharField(max_length=100, blank=False)
    Description = models.CharField(max_length=100,
     blank=False)
    User = models.ForeignKey(User, on_delete=models.CASCADE)
    def __str__(self):
        return self.Description
    class Meta:
        ordering = ('id',)
        verbose_name_plural = 'Devices'
        pass
```

Varios dispositivos añadidos se pueden observar en la Tabla 2.5 y los campos definidos son los siguientes:

- *IMEI*: Número de identificación único del dispositivo. Este es un campo *Charfield*, el cual se traduce como un *VARCHAR* dentro de la base de datos SQL.
- *Description*: En este campo se puede agregar información relevante para distinguir el dispositivo.
- *User*: Este es un campo *Foreign key*. Define una relación *many-to-one*, que le dice a Django que cada dispositivo le pertenece a un usuario, y a un usuario le pueden pertenecer cualquier cantidad de dispositivos. El parámetro *on_delete* especifica el comportamiento a adaptar cuando el objeto referenciado es eliminado. Esto es específico de SQL y no de Django. Se usa *CASCADE* para especificar que cuando un usuario asociado es eliminado, también se elimina el dispositivo [18]. La sección *return self.Description* le dice a Django que debe mostrar cuando se registre otro campo que quiera relacionarse con *Device*. La parte *ordering = ('id',)* solamente le dice a Django en que orden quiere que se muestre los dspositivos cuando estos se muestren al usuario. La última parte *verbose_name_plural = 'Devices'* define como se mostrarán el nombre en plural del modelo, en este caso *Devices*.

Tabla 2.5: Información de los dispositivos registrados

id PK integer	**IMEI** char varyng(100)	**Description** char varyng(100)	**User_id** integer
1	8356	Taxi	2
2	9012	Perro	2
3	0684	Gato	4
4	1847	Vendedor	3
5	9656	Cobrador	3
6	6468	Auto de ventas	5

El siguiente modelo se ha denominado *Reg* y registra los días de los que exista una ruta para visualizar, se agrega las siguientes líneas:

models.py

```
class Reg(models.Model):
    Day = models.CharField(max_length=100, blank=False)
    Description = models.ForeignKey(Device,
     on_delete=models.CASCADE)
    User = models.ForeignKey(User, on_delete=models.CASCADE)
    def get_absolute_url(self):
        return reverse('test-detail', args=[str(self.id)])
        pass
```

Ciertos campos agregados con este modelo se aprecian en la Tabla 2.6. Los campos definidos son los siguientes:

- *Day:* Registra el día del que se dispone una ruta.
- *Description:* Es un *ForeignKey* que está ligado al dispositivo al cual se le asignarán los días registrados.
- *User*: Este campo permite asignarle el día registrado a un usuario mediante la relación *ForeignKey*.

Tabla 2.6: Información de los registros de días

id PK integer	**Day** char varyng(100)	**Description_id** integer	**User_id** integer
2	2019_07_05	1	2
3	2019_07_06	1	2
4	2019_07_09	1	2
5	2019_07_13	2	2
6	2019_07_16	2	2
7	2019_07_17	3	4

A continuación se muestra el código para agregar el modelo que contendrá el día de recorrido, asignada a un dispositivo y un usuario.

models.py

```
class juan_8356_2019_07_03(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4,
    decimal_places=0)
    Distance = models.DecimalField(max_digits=5,
    decimal_places=2)
```

La altitud y distancia desde el punto anterior son campos secundarios que para la posición pueden no ser relevantes, sin embargo, se han tomado en cuenta en el modelo. Toda la información de posición de un día de recorrido de seis puntos se muestra en la Tabla 2.7.

- *Timestamp:* Almacena la información del momento exacto que se tomó la muestra.

- *Latitude:* La latitud del punto en grados es almacenada en este campo.
- *Longitude:* La longitud es almacenada en este campo.

Tabla 2.7: Información de posición de un día de recorrido

id	Timestamp	Latitude	Longitude	Altitude	Distance
1	2019-07-03T17:58:04Z	0.32857683	-78.1109563	2304	0.51
2	2019-07-03T17:58:02Z	0.32857888	-78.11097054	2304	1.63
3	2019-07-03T17:58:01Z	0.32857641	-78.11098746	2305	1.20
4	2019-07-03T17:57:58Z	0.32857103	-78.11105763	2304	0.37
5	2019-07-03T17:57:57Z	0.32856588	-78.11108161	2303	0.27

Vistas y permisos

Las vistas gestionan que es lo que se le permitirá mostrar a la plantilla HTML, a continuación se indica como se crea la vista para mostrar los dispositivos mostrados en la Tabla 2.5, se edita el archivo *views.py*.

views.py

```
from django.shortcuts import render
from .models import Device, Reg
from django.views.generic import DetailView, TemplateView,
ListView
from django.views import generic
from django.contrib.auth.mixins import LoginRequiredMixin,
PermissionRequiredMixin
from django.contrib.auth.models import User

class DeviceListView(LoginRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = Device
```

La línea *model = Device* llama al modelo *device* y los datos que se almacenan ahí, sin embargo la sección *LoginRequiredMixin* establece que si el usuario no ha ingresado con su cuenta válida, no tendrá acceso a esta información y *login_url = ”* indica que será redireccionado a la página de inicio de sesión.

Si el acceso a la vista de dispositivos ha sido exitoso, las siguientes vistas determinan que muestre la información de los días de recorrido disponibles para este. Se muestra una vista de

ejemplo denominada *TaxiListView*.

views.py

```
class TaxiListView(LoginRequiredMixin,
 PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = Reg
  permission_required = "track.view_juan_8356_2019_07_03"
  queryset = Reg.objects.filter(Description="1")#Taxi
```

Para el caso del dispositivo *taxi*, se ha usado un *QuerySet* que en este caso se usa como un filtro para mostrar solamente los días que tengan en su descripción el número uno, que son los relacionados con el dispositivo *taxi* como se muestra en las Tablas 2.5 y 2.6. Además de la restricción *LoginRequiredMixin* se agregó *PermissionRequiredMixin* para evitar que otros usuarios puedan visualizar un dispositivo que no le corresponda.

La siguiente vista es la que establece que se muestren los datos de geolocalización como en la Tabla 2.7, se muestra un ejemplo de como agregar la vista.

views.py

```
class superlimpio_6468_2019_07_31DetailView
(LoginRequiredMixin, PermissionRequiredMixin,
 generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = superlimpio_6468_2019_07_31
  permission_required =
   "track.view_superlimpio_6468_2019_07_29"
```

El permiso *PermissionRequiredMixin* se asocia con la última línea que le otorga el permiso a la vista para acceder al modelo *superlimpio_6468_2019_07_29* y así acceder a toda la información en la base de datos.

2.2.5 Geovisualiación

El Leaflet debe ser agregado al proyecto Django como una aplicación agregando la siguiente línea al archivo *settings.py*.

settings.py

```
INSTALLED_APPS = [
'leaflet',
]
```

La visualización del mapa se realiza en la plantilla HTML en este caso se usa como ejemplo *ana_0684_2019_07_17_list.html*, para la integración se debe colocar las siguientes líneas:

ana_0684_2019_07_17_list.html

```
{% load leaflet_tags %}
{% leaflet_css %}
{% leaflet_js %}
<!DOCTYPE html>
<div width="100%" height="100%">
{% leaflet_map "main" callback="map_init" %}
</div>
```

Leaflet ahora se puede usar mediante un script de java, sin embargo, se debe importar los datos de posición disponibles como el tiempo de muestra, latitud, longitud, altitud y distancia; cada conjunto de datos se almacenan en un <*li*> por medio de un *object_list*:

ana_0684_2019_07_17_list.html

```
<div class = "container" hidden>
    {% for ana_0684_2019_07_17 in object_list %}
    <li1>{{ ana_0684_2019_07_17.Timestamp }}</li1>
    <li2>{{ ana_0684_2019_07_17.Latitude }}</li2>
    <li3>{{ ana_0684_2019_07_17.Longitude }}</li3>
    <li4>{{ ana_0684_2019_07_17.Altitude }}</li4>
    <li5>{{ ana_0684_2019_07_17.Distance }}</li5>
    {% endfor %}
</div>
```

A continuación se crea el *script* y los datos se importan y almacenan en variables individuales.

ana_0684_2019_07_17_list.html

```
<script type="text/javascript">
    function map_init(map, options) {
    var time = document.querySelectorAll("li1");
    var lat = document.querySelectorAll("li2");
    var lon = document.querySelectorAll("li3");
    var alt = document.querySelectorAll("li4");
    var dist = document.querySelectorAll("li5");
    }
</script>
```

Los datos de longitud y latitud se almacenan en un *array* para obtener las coordenadas, la variable *lim* determina la cantidad máxima de puntos que tomará de las variables anteriores. Dentro del *Script* se agrega:

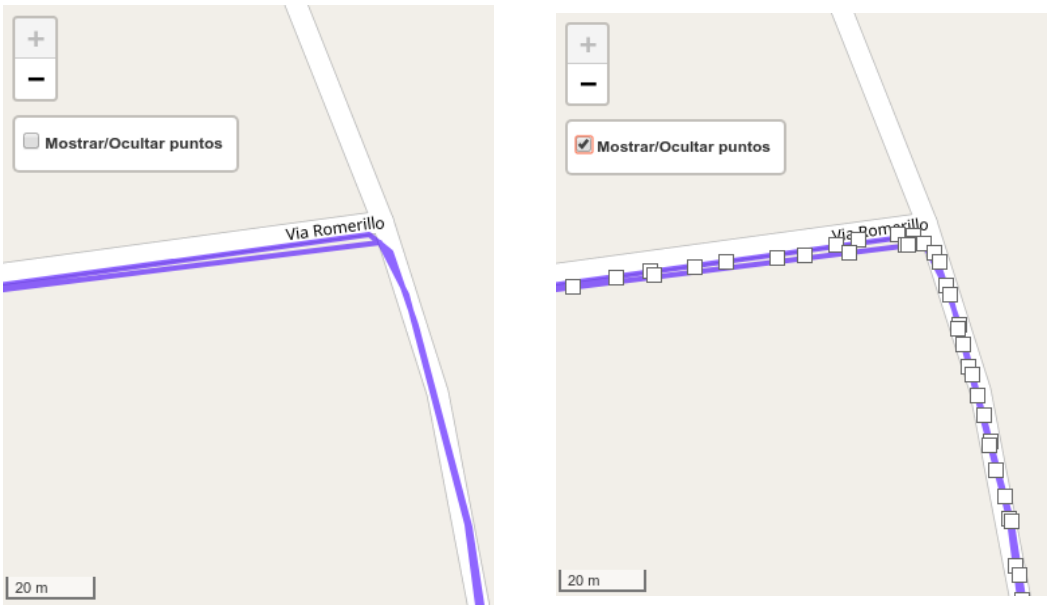
ana_0684_2019_07_17_list.html

```
var lim = 2000;
    var coor = [];
    for (i = 0; i < lim; i++) {
        coor.push([lat[i].innerHTML, lon[i].innerHTML]);
        var latlng = L.latLng([lat[i].innerHTML,
         lon[i].innerHTML]);
        }
```

Se crea un marcador para cada punto que mostrará toda la información disponible de ese punto. Para disminuir una ralentización se crea un ícono de marcador simple que usa menos recursos. Estos se pueden observar en la Figura 2.6b.

ana_0684_2019_07_17_list.html

```
    var myIcon = L.divIcon({html: ''});
    var a = L.marker([lat[i].innerHTML, lon[i].innerHTML],
     {icon: myIcon}).bindPopup("Hora: " +
     time[i].innerHTML + "<br />" + "Latitud: " +
     lat[i].innerHTML + "<br />" + "Longitud: " +
     lon[i].innerHTML + "<br />" + "Altitud: " +
     alt[i].innerHTML + " m<br />" + "Distancia: " +
     dist[i].innerHTML + " m").openPopup();
    points.addLayer(a);
        }
```



(a) Marcadores desactivados

(b) Marcadores activados

Figura 2.6: Capa de activación y desactivación de marcadores

Se crea un botón para activar y desactivar la capa que muestra todos los marcadores como se muestra en la Figura 2.6.

ana_0684_2019_07_17_list.html

```
var points = L.layerGroup([]);
    }
var overlayMaps = {
  "Mostrar/Ocultar puntos": points
  };
  var baseMaps = {};
 L.control.layers(baseMaps, overlayMaps,
  {collapsed: false, position: 'topleft'}).addTo(map);
```

Para mostrar el la ruta realizada se usa la variable *latlng* que almacenó todos los datos anteriormente, se crea una polilínea que une todos los puntos de coordenadas.

ana_0684_2019_07_17_list.html

```
var polylineOptions = {
          color: 'blue',
          weight: 4,
          opacity: 0.6,
          smoothFactor: 5
        };
var polyline = new L.Polyline(coor, polylineOptions);
map.addLayer(polyline);
```

Capítulo 3

Pruebas de funcionamiento

3.1 Funcionamiento

3.1.1 Inicio de sesión

Para acceder a la página principal el usuario debe acceder por medio del navegador de cualquier dispositivo a la dirección *http://68.66.235.160:8000/*, la página tiene el siguiente aspecto:

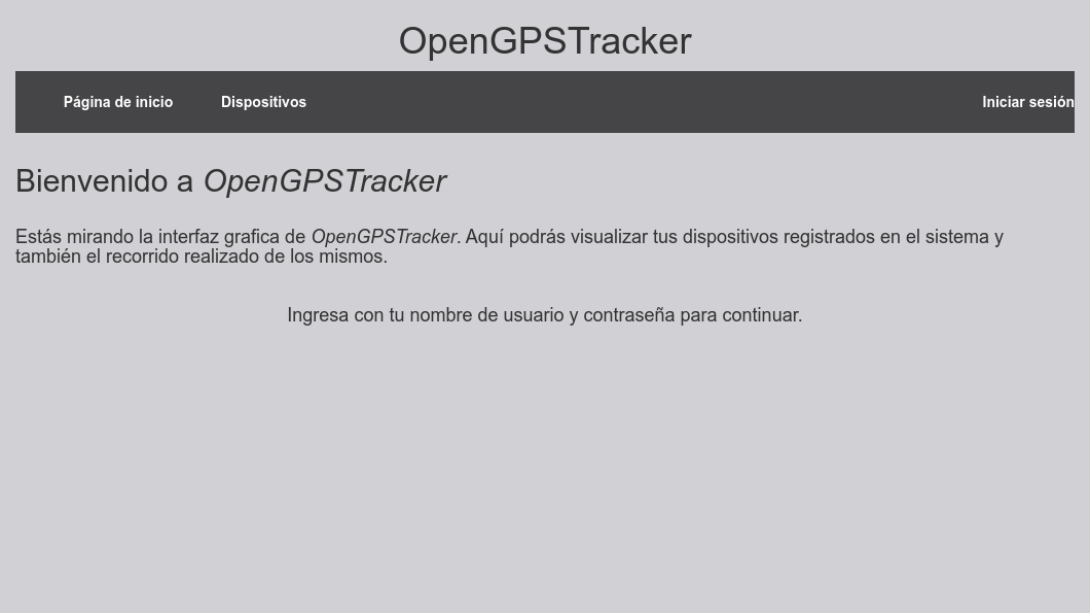


Figura 3.1: Página principal del sistema

Se accede con un nombre de usuario y contraseña, en este caso se colocan los datos de *Juan* en los casilleros correspondientes como se muestra en la siguiente Figura.

Figura 3.2: Página de inicio de sesión

3.1.2 Selección de dispositivo

Si la información de inicio de sesión es correcta, el usuario puede visualizar sus dispositivos registrados presionando el texto *Dispositivos*.

Figura 3.3: Página de visualización de dispositivos

Se debe seleccionar un dispositivo para poder visualizar los días de recorrido disponibles.

Figura 3.4: Página de selección del día de registro

3.1.3 Vista del mapa

A continuación se muestra el mapa y en este se muestra la ruta recorrida del dispositivo en el día especificado.

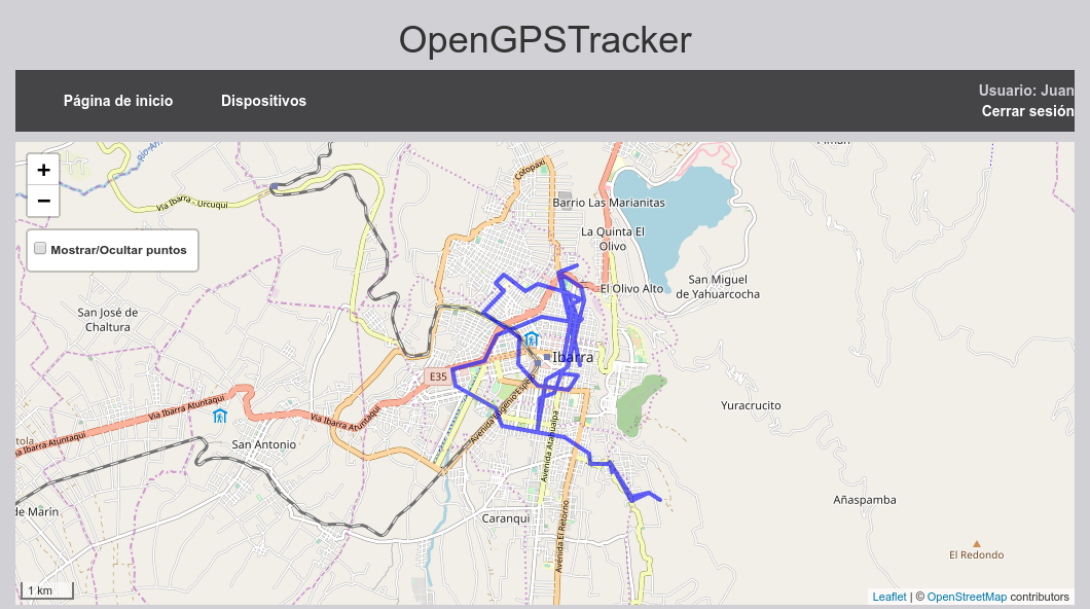


Figura 3.5: Página del mapa con la ruta recorrida

Con los botones se puede acercar y alejar, la casilla activa y desactiva los puntos de la ruta(marcadores) y al hacer click en uno de ellos se muestra toda la información de posición de ese punto mediante una ventana emergente como se muestra en la figura siguiente.

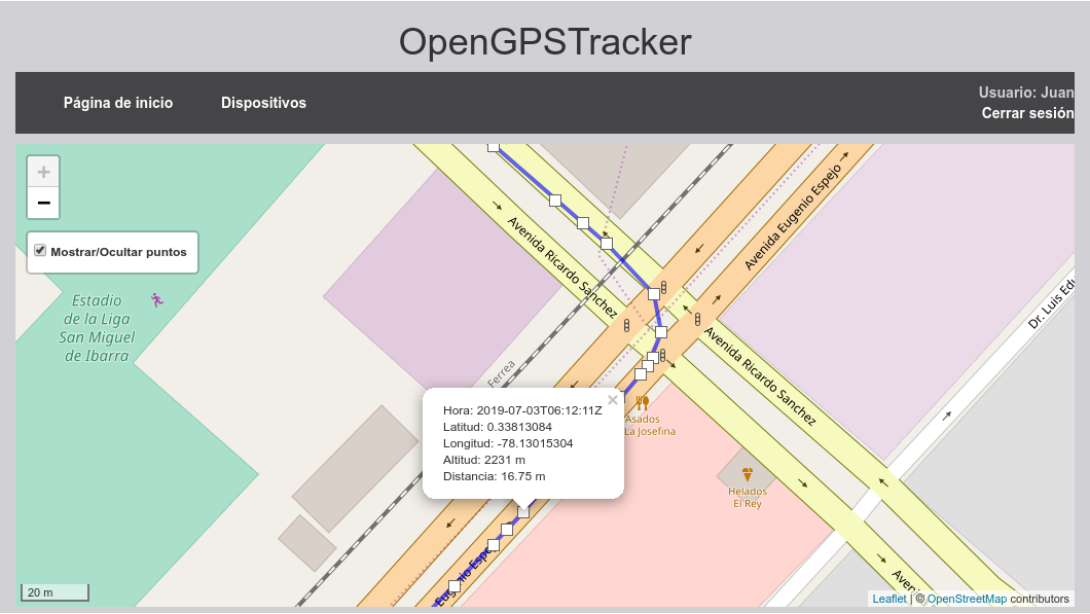


Figura 3.6: Mapa con puntos de ruta y *popup* activado

3.1.4 Acceso del usuario administrador

El administrador puede ingresar con sus datos de inicio de sesión y tiene acceso a los dispositivos de todos los usuarios como se muestra en la Figura 3.7 y además las rutas de los mismos.

Figura 3.7: Página de visualización de dispositivos del administrador

3.1.5 Gestión de usuarios y dispositivos

Por medio del sitio de administrador de Django, accesible desde *http://68.66.235.160:8000/admin* El administrador tiene la capacidad de agregar, editar y eliminar dispositivos como se muestra en la Figura 3.8, también puede hacer esto con los usuarios y la información de posicionamiento de los mismos.

(a) Agregar dispositivo (b) Editar o eliminar dispositivo

Figura 3.8: Página de gestión de dispositivos

3.2 Pruebas

El software desarrollado se someterá a pruebas bajo condiciones reales de funcionamiento para determinar su desempeño.

3.2.1 Evaluación de usabilidad

Para evaluar la usabilidad del software se toma como referencia la evaluación *SUS*[1] [22]. El sistema *SUS* consiste en un cuestionario o encuesta de diez preguntas acerca de la usabilidad con cinco opciones de respuesta para cada una; van desde *muy en desacuerdo* hasta *totalmente de acuerdo* [22], una pregunta de muestra se presenta a continuación en la Figura 3.9.

[1] SUS: System Usability Scale, Escala de uso del sistema.

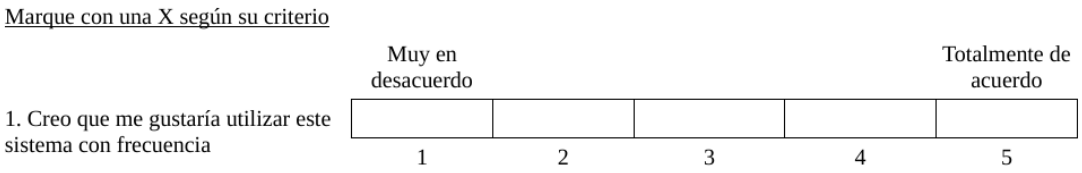


Figura 3.9: Pregunta de muestra de *SUS*

Esta evaluación se realiza luego de que la persona haya culminado una tarea en la que use el sistema a ser analizado, que en este caso es la interfaz. *SUS* es parte de la norma *ISO 9241-11* [23] y se ha popularizado en la industria porque proporciona una rápida y confiable herramienta para medir la usabilidad de una gran variedad de productos y servicios, incluyendo hardware, software, dispositivos móviles, aplicaciones web, entre otros. En el estudio [24] se demuestra que se puede obtener un resultado confiable con solamente una muestra de ocho a doce personas, por lo tanto se resolvió realizar la prueba a diez personas aleatorias en de biblioteca de la Universidad Técnica del Norte.

A las personas se les entregó una hoja impresa con la tarea encomendada como se muestra en la Figura 3.10. Se les proporcionó un dispositivo con un navegador ya abierto, y en él, la página de inicio del proyecto ya cargada. No se les ofreció más tutoría hasta culminar el tiempo límite que considere el evaluador, que en este caso se estableció en diez minutos.

Suposiciones:

- Tu nombre es *Juan*.
- Posees un taxi y perteneces a una cooperativa de taxis.
- Tienes un chofer a cargo de tu taxi.
- Contrataste una empresa que registra diariamente el recorrido de tu taxi.
- Puedes acceder a toda la información almacenada del recorrido de tu taxi mediante la página web de la empresa.

Pregunta:

- ¿A que hora ingresó o salió tu taxi de la laguna de yahuarcocha el día *viernes 05 de julio de 2019*?

Nota: Tus datos para ingresar a la página son:

Nombre de usuario:	*Juan*
Contraseña:	*juan2019*

Figura 3.10: Tarea encomendada a las personas encuestadas

Cuando la tarea haya sido terminada o hayan transcurridos los diez minutos, lo que ocurra primero, se procede a entregar una hoja con las preguntas propuestas para la evaluación, las cuales están basadas en [22] y son las siguientes:

1. Creo que me gustaría utilizar este sistema con frecuencia
2. Encontré el sistema innecesariamente complejo
3. Creo que el sistema era fácil de usar
4. Creo que necesitaría el apoyo de otra persona para ser capaz de usar este sistema
5. Encontré que las diversas funciones en este sistema estaban bien integradas
6. Creo que había demasiada inconsistencia en este sistema
7. Creo que la mayoría de la gente aprendería a usar este sistema muy rápidamente
8. Encontré el sistema muy difícil de usar
9. Me sentí muy confiado usando el sistema

10. Necesito aprender muchas cosas antes de poder trabajar con este sistema

Todos los involucrados culminaron la tarea exitosamente, los resultados fueron procesados y se muestran en la Tabla 3.1, en la que se realiza una media del valor de la puntuación para cada pregunta,

Tabla 3.1: Puntuación de cada pregunta y valor medio

Pregunta	**Participantes**										**Valor medio**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
1	5	4	4	3	5	5	5	5	5	5	4.6
2	4	3	2	3	1	3	4	1	5	1	2.7
3	4	3	4	4	5	4	5	5	4	5	4.3
4	1	2	4	4	3	5	1	4	1	1	2.6
5	5	4	3	3	5	3	5	5	4	5	4.2
6	1	2	3	2	1	1	1	1	1	1	1.4
7	4	4	2	5	5	5	5	5	5	5	4.5
8	1	2	3	2	1	3	1	1	1	1	1.6
9	3	3	3	3	5	4	5	5	4	1	3.6
10	5	2	4	2	2	5	3	4	1	1	2.9

A continuación en la Tabla 3.2 se toma el valor medio de las preguntas impares y se resta uno a cada valor, los valores medios de las preguntas pares son restados por cinco. Se suma la puntuación obtenida de las diez preguntas y este valor se multiplica por 2.5, obteniendo así el valor final de la evaluación SUS [22]. En este caso la evaluación de usabilidad de la interfaz, presentó un puntaje de 75.

Tabla 3.2: Resultados del cuestionario de la escala de usabilidad

Pregunta	**Valor medio**	**Cálculo**	**Puntuación**
1	4.6	4.6-1	3.6
2	2.7	5-2.7	2.3
3	4.3	4.3-1	3.3
4	2.6	5-2.6	2.4
5	4.2	4.2-1	3.2
6	1.4	5-1.4	3.6
7	4.5	4.5-1	3.5
8	1.6	5-1.6	3.4
9	3.6	3.6-1	2.6
10	2.9	5-2.9	2.1
		Total	30
		x 2.5	**75**

La Figura 3.11 indica el *ranking* general de percentiles de puntuaciones *SUS*. Como lo muestra [6], el promedio de este *ranking* es 68, esto significa que el valor final de la evaluación de la interfaz (75), está sobre el promedio.

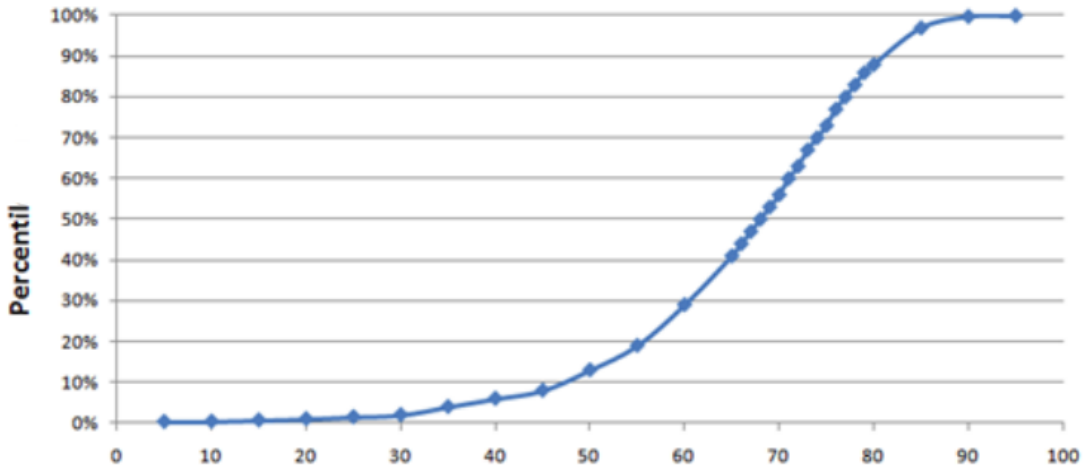


Figura 3.11: *Ranking* de percentiles de las puntuaciones de SUS [6]

En [7] se establece que adjetivo se coloca a cada puntuación de *SUS* de acuerdo a su ubicación en el *ranking*, como se muestra en la Figura 3.12, que va desde el puntaje mas bajo como *lo peor imaginable*, *malo*, *ok*, *bueno*, *excelente* y *lo mejor imaginable*. Además establece un

sistema de ponderación que va desde el más bajo con la letra *F* hasta el más alto ponderado con la letra *A* [7]. A un puntaje de 75, le corresponde la ponderación *C* y el adjetivo *Bueno*.

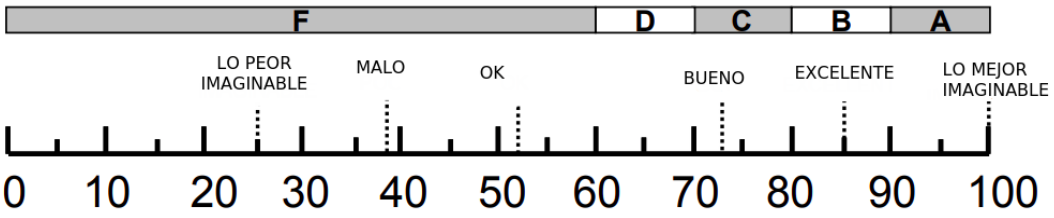


Figura 3.12: *Ranking* de adjetivos de SUS [7]

3.2.2 Evaluación de velocidad de carga

Un aspecto importante en una aplicación web es el tiempo de carga y este es relevante si posee complementos adicionales, como en el presente trabajo que carga mapas usando librerías e importa mucha información desde una base de datos. Según un estudio realizado por el sitio Akamai [25], un usuario promedio espera que una página web cargue en tres segundos o menos antes de abandonar la página, sin embargo, se evidencia que varios de los principales sitios web de comercio pueden duplicar o hasta triplicar este tiempo de carga [26]. Esto puede generar inconformidad para el usuario final y representar pérdidas de miles de dólares para la empresa según [26]. Independientemente del fin de la página web se puede usar la información presentada en el estudio para recalcar la importancia del tiempo de carga y usarla como referencia para comparar los tiempos de carga de la presente aplicación web.

El tiempo de carga en la interfaz depende principalmente de la calidad y ubicación del *Web Hosting* contratado, la conexión a internet de la máquina cliente así como el hardware de la misma y la estructura interna de la página web que en este caso depende en gran parte del framework Django. Para no afectar negativamente en el tiempo de carga se realizaron las pruebas en con una conexión a internet de velocidad moderada y un hardware estándar como se muestra en la Tabla 3.3; cabe recalcar que se usaron estas mismas condiciones y especificaciones para todas las pruebas realizadas a continuación y cada una de las cinco repeticiones correspondientes que se decidieron realizar.

Tabla 3.3: Detalle de condiciones para evaluación

Velocidad de bajada	56 Mbps
Velocidad de subida	56 Mbps
Ping	8 ms
Navegador web	Vivaldi 2.11
Sistema operativo	Ubuntu 18.04 LTS
Procesador	2.1 Ghz
Núcleos	4
Memoria RAM	8 GB

Se obtuvieron los tiempos de carga de la página principal y los subdominios de la página destinados al usuario final, los datos obtenidos se muestran a continuación.

Tabla 3.4: Tiempos de carga de los subdominios de la aplicación web

Página	**Tiempo de carga (s)**
Bienvenida	0.55
Inicio de sesión	0.44
Dispositivos	0.51
Historial	0.49
Vista del mapa	6.07

Se puede visualizar en la Tabla 3.4 que los tiempos de carga oscilan en el medio segundo, excepto en la página donde se visualiza el mapa, esto se debe a la carga del mismo y a la importación de toda la información posicional desde la base de datos.

Carga de marcadores

Otro punto que influye en el desempeño de la aplicación web, específicamente en la navegación del mapa, es la carga de los miles de marcadores, que suponen un alto consumo de recursos, lo cual se traduce en un mayor tiempo de carga de los mismos. Para ilustrar esto se han realizado pruebas alterando la cantidad de marcadores y registrando la variación del tiempo de carga, en la Figura 3.13 se muestra una gráfica que representa estas variaciones en función del tiempo.

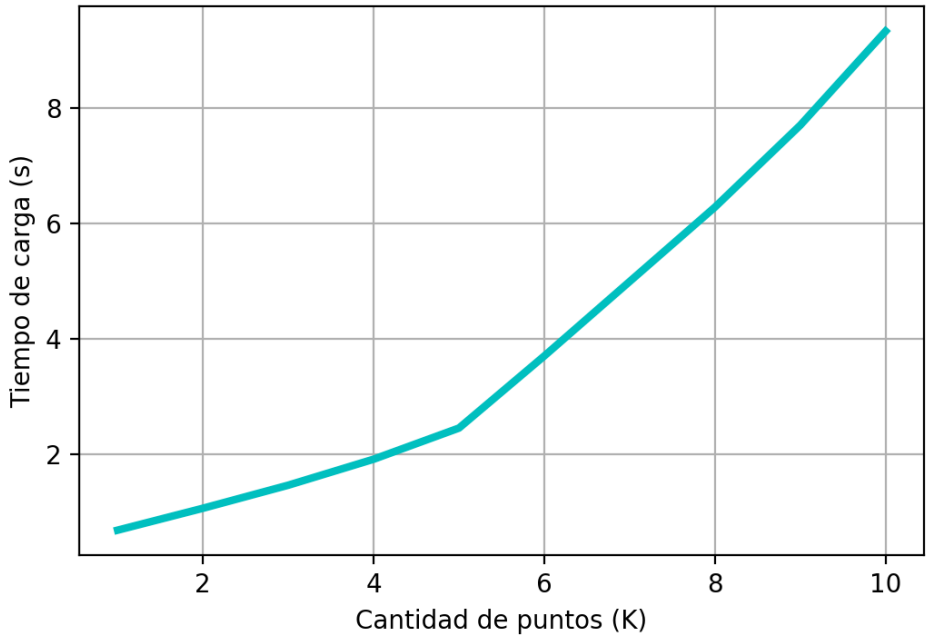
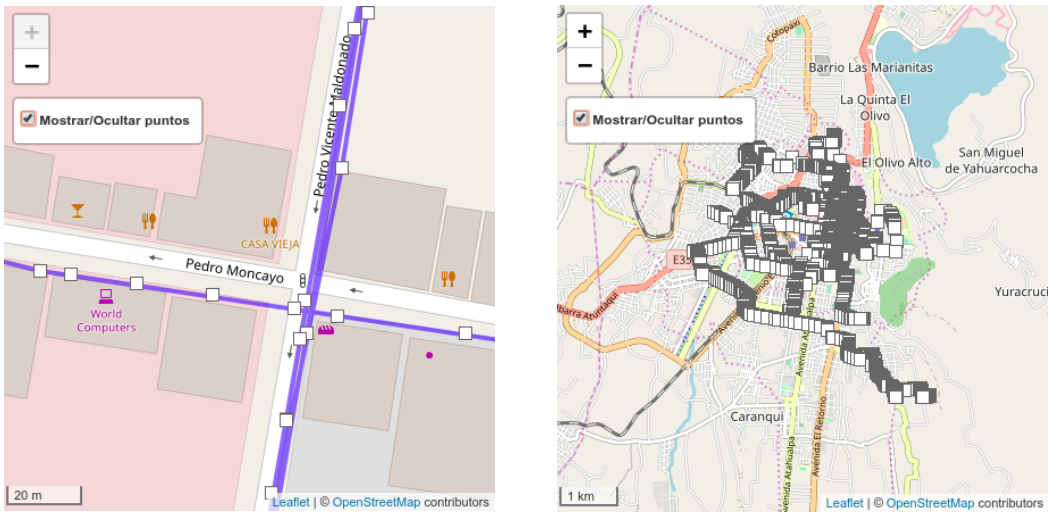


Figura 3.13: Tiempo de carga de marcadores

Se puede apreciar que al cargar 2000 marcadores el tiempo de carga es de aproximadamente un segundo, muy bajo en comparación de la carga de los mas de 9 segundos que le toma cargar 10.000 marcadores

Además este tiempo también es influenciado por el nivel de acercamiento (zoom) del mapa en el momento de cargar los marcadores, en la Figura 3.15 se puede apreciar que cuando el nivel de acercamiento es menor como en la Figura 3.14b, el tiempo de carga es alto, casi 6 segundos según la gráfica; pero si se acerca la vista del mapa como en la Figura 3.14a, el tiempo disminuye hasta mas de la mitad, en este caso aproximadamente 2.6 segundos.



(a) Nivel de acercamiento 19

(b) Nivel de acercamiento 13

Figura 3.14: Marcadores a diferente nivel de acercamiento

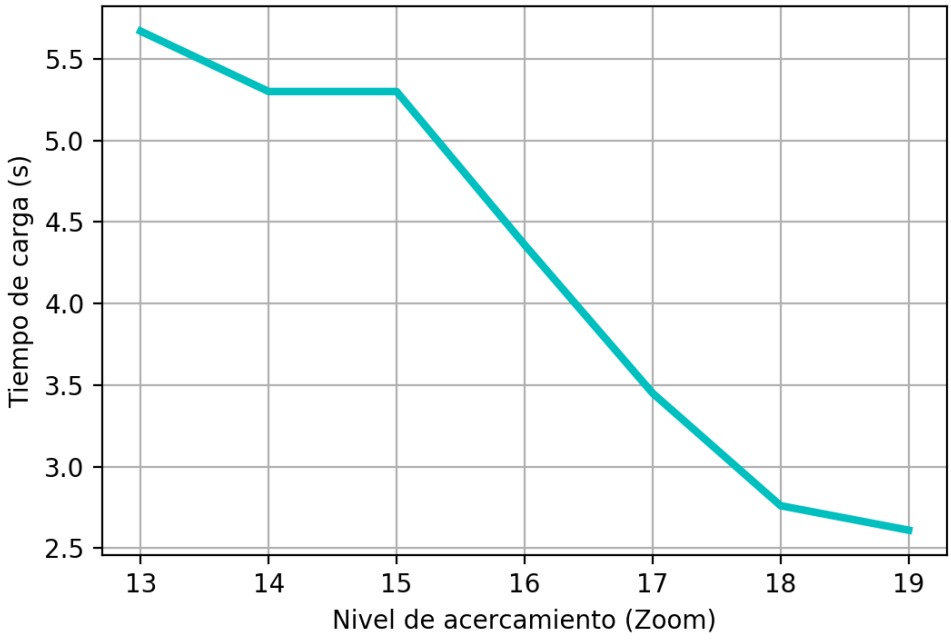


Figura 3.15: Tiempo de carga de marcadores en función del nivel de acercamiento

3.2.3 Discusión

La interfaz desarrollada es un proyecto *Open Source* o de código abierto, disponible en *GitHub* bajo la licencia *GNU General Public Licence v3.0*. El uso de herramientas de código abierto como Python, Django, OpenStreetMap o Leaflet; permite que este proyecto sea usado, copiado, y modificado por desarrolladores independientes, instituciones o empresas, sin restricciones y para diferentes fines. Puntualmente, OpenStreetMap posee la ventaja de permitir modificaciones, adiciones o actualizaciones ágiles y rápidas de lugares, puntos relevantes o vías según sus necesidades, directamente en la base de datos. Además en conjunto con la librería leaflet se alcanzan funcionalidades de capas, permitiendo agregar figuras básicas y avanzadas. Y trabajando con software de terceros, puede ampliar sus capacidades gratuitamente o bajo un coste. Todas estas ventajas no serían aprovechables si se usara software de código cerrado como por ejemplo, la solución de mapas utilizada en la mayoría de los proyectos existentes en el medio.

El desarrollo de la interfaz utilizando el framework Django permitió un desarrollo ágil, limpio y dinámico; debido entre a otras razones, a la detallada información que proporciona sobre el desarrollo en general y su módulo para la geovisualización. Frente a otras opciones, estas características destacables posicionan a Django como uno de los mas usados, contando con millones de usuarios según [27], por lo que posibilita la fusión o integración de este proyecto con otros similares, para distintos fines.

En las pruebas de funcionamiento, casi todos los subdominios del sistema presentan un tiempo de carga bajo, alrededor de medio segundo. Sin embargo, el subdominio encargado de la visualización del mapa presenta un tiempo mucho mayor (6 segundos). Esto se debe principalmente a que importa alrededor 20.000 datos desde PostgreSQL. La Figura 3.13 muestra que esta cantidad de puntos es elevada y por esta razón aumenta el tiempo de carga. Afectando directamente la fluidez y navegabilidad del sistema específicamente en la sección del mapa.

La navegación del mapa se ve afectada por la gran cantidad de marcadores mostrados, graficar un número excesivo de ellos puede ser perjudicial, ya que, aumenta el tiempo de carga y ralentiza el desempeño; además, demasiada información del recorrido que puede ser innecesaria dependiendo de las necesidades del proyecto. Por el contrario, disminuir demasiado la cantidad de marcadores que se grafican podría ocasionar una pérdida de información relevante del recorrido. Lo correcto para establecer un tiempo de carga óptimo, sería determinar una cantidad idónea de puntos a graficar. Esta cantidad depende del intervalo de tiempo y distancia en los cuales se desea saber la ubicación del objetivo, además del tiempo de toma de muestra y la cantidad de horas de monitoreo.

Visualizar el mapa con el nivel de acercamiento máximo resulta ligero y fluido para una determinada cantidad de marcadores, ya que se grafica una cantidad menor que al visualizar el mapa en el nivel mínimo, donde la cantidad de marcadores mostrados es mucho mayor, y por ende ralentiza la navegabilidad.

Al igual que algunos proyectos similares, esta interfaz muestra en el mapa la información estática almacenada en la base de datos [2], sin embargo, tiene la capacidad de manejar datos dinámicos como en la popular base de datos *Firebase*, soportando así el seguimiento en tiempo real o *tracking*.

Con respecto a la usabilidad del sistema desarrollado, según el *ranking* de la prueba *SUS* elaborada por [6], un puntaje de 75 es mejor que el 73% de los sistemas evaluados y un puntaje de 52 está por debajo del 85% de ellos, la prueba *SUS* de la aplicación web del presente trabajo, presento un resultado de 75, el cual la coloca sobre el 73% de los sistemas, le asigna una ponderación *C* y le califica como un sistema *Bueno*.

Conclusiones y trabajo futuro

Conclusiones

- La interfaz abierta para servidor de datos de localización es un sistema que integra herramientas de software libre como el framework *Django* y la librería *Leaflet* para implementar el servidor de mapas *OpenStreetMap*. Estas herramientas hacen posible un desarrollo ágil debido a la rapidez y facilidad que estos proveen, a pesar de su uso en este tipo de sistemas es poco común.

- Para mejorar la disponibilidad del sistema al usuario y permitir al desarrollador trabajar con mayor seguridad, la interfaz se implementó en un servidor virtual privado (VPS) que brinda mayor estabilidad por medio de una conexión a internet estable y continua, un sistema de alimentación ininterrumpida (UPS) y un sistema operativo robusto y ligero como *Ubuntu 16.04 server*.

- Para conocer la ubicación detallada de un dispositivo es necesario que un sistema posea funciones específicas que lo permitan, la interfaz desarrollada en este trabajo satisface estas funciones, ya que, permite visualizar los dispositivos que le han sido asignados a un usuario, las rutas recorridas de los mismos, los puntos de toma de muestra e información adicional sobre cada uno de ellos como: tiempo de toma de la muestra, latitud, longitud, altitud y distancia hasta el punto anterior.

- El sistema fue sometido a pruebas bajo condiciones reales de funcionamiento, según la prueba *SUS* el sistema se cataloga como *Bueno* y con una ponderación *C*, por lo tanto podría ser usado para diferentes fines, sin embargo, todavía existen áreas donde realizar mejoras y ampliaciones para un trabajo a futuro como se muestra en la siguiente sección.

Trabajo futuro

- Agregar la característica de *tracking* en vivo o monitoreo en tiempo real.

- Implementar un motor de agrupamiento de marcadores automático para gestionar el número de marcadores que se muestran en la pantalla de acuerdo al zoom y la proximidad de la vista del mapa para disminuir el tiempo de carga de la página.

- Utilizar un método que importe solamente la cantidad de datos que serán usados en el mapa para optimizar tiempo y recursos.
- Complementar el sistema con otros enfocados en recolectar datos directamente desde un módulo GPS.
- Mejorar la navegabilidad de la página web en dispositivos móviles o desarrollar una aplicación específica para los mismos.

Bibliografía

[1] M. Conza, "Desarrollo de una aplicación web orientada a servicios para el monitoreo de una flota de vehículos haciendo uso de la tecnología GPS," 2013. [En línea]. Disponible en: http://repositorio.unsaac.edu.pe/handle/UNSAAC/947

[2] I. García, "Desarrollo de un sistema de localización de vehículos por GPS y GSM/GPRS," 2018. [En línea]. Disponible en: http://oa.upm.es/51311/

[3] "Traccar - GPS Tracking Software - Free and Open Source System - Traccar." [En línea]. Disponible en: https://www.traccar.org/

[4] "Framework Web Django (Python)." [En línea]. Disponible en: https://developer.mozilla.org/es/docs/Learn/Server-side/Django

[5] "Leaflet — an open-source JavaScript library for interactive maps." [En línea]. Disponible en: https://leafletjs.com/

[6] J. Sauro, *A Practical Guide to the System Usability Scale: Background, Benchmarks & Best Practices*. Denver, Colo: CreateSpace Independent Publishing Platform, abr. 2011.

[7] A. Bangor, P. Kortum, y J. Miller, "Determining What Individual SUS Scores Mean: Adding an Adjective Rating Scale," 2009. [En línea]. Disponible en: http://uxpajournal.org/determining-what-individual-sus-scores-mean-adding-an-adjective-rating-scale/

[8] C. Parra, "Desarrollo de un sistema para monitoreo y control satelital de vehículos mediante el uso del dispositivo GPS TK 303G para la comercializadora de dispositivos satelitales GENIUS EC," 2018. [En línea]. Disponible en: http://repositorio.espe.edu.ec/jspui/handle/21000/14674

[9] "Django | The Web framework for perfectionists with deadlines." [En línea]. Disponible en: https://www.djangoproject.com/

[10] J. Astudillo y E. Delgado, "Sistema de localización monitoreo y control vehicular basado en los protocolos GPS/GSM/GPRS," abr. 2012. [En línea]. Disponible en: http://dspace.ups.edu.ec/handle/123456789/1927

[11] M. Chuquitarco y R. Naranjo, "Diseño e instalación de un sistema de rastreo satelital mediante GPS y GPRS al vehículo Chevrolet-Aveo de la Escuela de Conducción de ESPE -Latacunga." sep. 2012. [En línea]. Disponible en: http://repositorio.espe.edu.ec/jspui/handle/21000/5879

[12] C. Vázquez, E. Palacios, L. Córdova, y M. Paz, "Dispositivo de adquisición y transmisión de la posición de un vehículo mediante GPS y Wi-Fi," p. 13, 2016.

[13] P. Wayner, *La ofensiva del software libre*, primera edición ed. España: ESIC Editorial, 2017. [En línea]. Disponible en: https://books.google.com/books/about/La_ofensiva_del_software_libre.html?id=eFuBwP6apJMC

[14] V. Quintero, A. Medina, C. Martínez, F. Ibáñez, y M. Muñoz, "Rastreo satelital de código abierto en sistemas ciberfísicos, retos y oportunidades." vol. 38, dic. 2016.

[15] J. Hernández, *Software libre: ténicamente viable, económicamente sostenible y socialmente justo.*, España, 2005.

[16] I. G. Petrovski, *GPS, GLONASS, Galileo, and BeiDou for Mobile Devices: From Instant to Precise Positioning*. Cambridge University Press, mayo 2014, google-Books-ID: PqpkAwAAQBAJ.

[17] W. Tomasi, *Sistemas de comunicaciones electrónicas*. Pearson Educación, 2003, google-Books-ID: _2HCio8aZiQC.

[18] A. Melé, *Django 2 by Example*, 2nd ed. Birmingham Mumbai: Ingram short title, 2018.

[19] "PostgreSQL: The world's most advanced open source database." [En línea]. Disponible en: https://www.postgresql.org/

[20] "Mr. Milú - Agencia de Desarrollo Digital y Diseño en Madrid — Django," library Catalog: mrmilu.com. [En línea]. Disponible en: https://mrmilu.com/p6751350/

[21] "OpenStreetMap." [En línea]. Disponible en: https://www.openstreetmap.org/

[22] J. Brooke, "SUS - A quick and dirty usability scale," p. 7.

[23] T. Dwi Susanto, A. Ingesti Prasetyo, y H. M. Astuti, "Web usability evaluation on BloobIS website by using hallway usability testing method and ISO 9241:11," vol. 974, p. 012043, mar. 2018. [En línea]. Disponible en: http://adsabs.harvard.edu/abs/2018JPhCS.974a2043D

[24] T. Tullis, "A Comparison of Questionnaires for Assessing Website Usability," *ResearchGate*, 2006. [En línea]. Disponible en: https://www.researchgate.net/publication/228609327_A_Comparison_of_Questionnaires_for_Assessing_Website_Usability

[25] “Seguridad, distribución en la nube, rendimiento | Akamai.” [En línea]. Disponible en: https://www.akamai.com/es/es/

[26] “Why Faster Websites Make More Money [Infographic],” jul. 2014. [En línea]. Disponible en: https://www.webfx.com/blog/internet/website-page-load-time-conversions/

[27] W. Vincent, *Django for beginners*. [En línea]. Disponible en: https://djangoforbeginners.com/

Apéndice

Apéndice A: Códigos

OpenGPS/opengps6/settings.py

```
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

SECRET_KEY = 'uf(_9=r9^gf1^23i-trz6nv1c-vn0smyclz7a%$vlvj)jhvadj'

DEBUG = True

ALLOWED_HOSTS = ['68.66.235.160',
                 'www.opengpstracker.xyz',
                 'server.opengpstracker.xyz',
                 '68.66.235.160:8000',
                 '127.0.0.1:8000',
                 '127.0.0.1'
]

INSTALLED_APPS = [
    'leaflet',
    'django.contrib.gis',
    'track.apps.TrackConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
```

```
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'opengps6.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        #'DIRS': [os.path.join(BASE_DIR, 'templates'),],
        'DIRS': ['./templates',],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'opengps6.wsgi.application'

DATABASES = {
    'default': {
        'ENGINE': 'django.contrib.gis.db.backends.postgis',
        'NAME': 'my_database_6',
        'USER': 'postgres',
        'PASSWORD': 'test123',
        'HOST': 'localhost',
        'PORT': '5432',
    }
}

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidator',
```

```
    },
    {
        'NAME':
'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]

LEAFLET_CONFIG = {
'RESET_VIEW': False,
'MIN_ZOOM': 3,
'MAX_ZOOM': 19
}

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True

STATIC_URL = '/static/'
LOGIN_REDIRECT_URL = '/'
```

OpenGPS/opengps6/urls.py

```
from django.contrib import admin
from django.urls import path, include
from django.views.generic import RedirectView
from django.conf import settings
from django.conf.urls.static import static
from django.conf.urls import url, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('opengps/', include('track.urls')),
    path('', RedirectView.as_view(url='/opengps/', permanent=True)),
    url(r'^accounts/', include('django.contrib.auth.urls')),
] + static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)
```

OpenGPS/templates/registration/login.html

```
{% extends "base_generic.html" %}

{% block content %}

{% if form.errors %}
<p>Tu nombre de usuario y contraseÃśa no coindiden.
 Porfavor intenta de nuevo.</p>
{% endif %}

{% if next %}
    {% if user.is_authenticated %}
    <p>Tu cuenta no tiene acceso a esta pÃągina. Para continuar,
    porfavor ingrese con una cuenta que tenga acceso.</p>
    {% else %}
    <p>Porfavor inicie sesiÃşn para ver la pÃągina.</p>
    {% endif %}
{% endif %}

<form method="post" action="{% url 'login' %}">
{% csrf_token %}
<center>
<div>
  <td>{{ form.username.label_tag }}</td>
  <td>{{ form.username }}</td>
</div>
<div>
  <td>{{ form.password.label_tag }}</td>
  <td>{{ form.password }}</td>
</div>

<div>
  <input type="submit" value="Ingresar" />
  <input type="hidden" name="next" value="{{ next }}" />
</div>
</center>
</form>
{% endblock %}
```

OpenGPS/templates/registration/logged_out.html

```
{% extends "base_generic.html" %}
```

```
{% block content %}
<p>Haz herrado la sesiÃşn</p>

<a href="{% url 'login'%}">Haz click aquÃ■ para iniciar sesiÃşn de nuevo.</a>
{% endblock %}
```

OpenGPS/track/admin.py

```
from django.contrib import admin
from .models import Device, Reg
from .models import juan_8356_2019_07_03, juan_8356_2019_07_05,
juan_8356_2019_07_06, juan_8356_2019_07_09, juan_9012_2019_07_13,
juan_9012_2019_07_16, ana_0684_2019_07_17, ana_0684_2019_07_18,
ana_0684_2019_07_19, ventasexpres_1847_2019_07_20,
ventasexpres_1847_2019_07_22, ventasexpres_1847_2019_07_23,
ventasexpres_9656_2019_07_24, ventasexpres_9656_2019_07_25,
superlimpio_6468_2019_07_29, superlimpio_6468_2019_07_30,
superlimpio_6468_2019_07_31
from leaflet.admin import LeafletGeoAdmin

class DeviceAdmin(LeafletGeoAdmin):
    list_display = ('IMEI', 'Description', 'User')
class RegAdmin(LeafletGeoAdmin):
    list_display = ('id', 'Day', 'Description', 'User')
class juan_8356_2019_07_03Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class juan_8356_2019_07_05Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class juan_8356_2019_07_06Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class juan_8356_2019_07_09Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class juan_9012_2019_07_13Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class juan_9012_2019_07_16Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class ana_0684_2019_07_17Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class ana_0684_2019_07_18Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class ana_0684_2019_07_19Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class ventasexpres_1847_2019_07_20Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
```

```
class ventasexpres_1847_2019_07_22Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class ventasexpres_1847_2019_07_23Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class ventasexpres_9656_2019_07_24Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class ventasexpres_9656_2019_07_25Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class superlimpio_6468_2019_07_29Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class superlimpio_6468_2019_07_30Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')
class superlimpio_6468_2019_07_31Admin(LeafletGeoAdmin):
list_display = ('Timestamp', 'Latitude', 'Longitude', 'Altitude', 'Distance')

admin.site.register(Device, DeviceAdmin)
admin.site.register(Reg, RegAdmin)
admin.site.register(juan_8356_2019_07_03, juan_8356_2019_07_03Admin)
admin.site.register(juan_8356_2019_07_05, juan_8356_2019_07_05Admin)
admin.site.register(juan_8356_2019_07_06, juan_8356_2019_07_06Admin)
admin.site.register(juan_8356_2019_07_09, juan_8356_2019_07_09Admin)
admin.site.register(juan_9012_2019_07_13, juan_9012_2019_07_13Admin)
admin.site.register(juan_9012_2019_07_16, juan_9012_2019_07_16Admin)
admin.site.register(ana_0684_2019_07_17, ana_0684_2019_07_17Admin)
admin.site.register(ana_0684_2019_07_18, ana_0684_2019_07_18Admin)
admin.site.register(ana_0684_2019_07_19, ana_0684_2019_07_19Admin)
admin.site.register(ventasexpres_1847_2019_07_20,
    ventasexpres_1847_2019_07_20Admin)
admin.site.register(ventasexpres_1847_2019_07_22,
    ventasexpres_1847_2019_07_22Admin)
admin.site.register(ventasexpres_1847_2019_07_23,
    ventasexpres_1847_2019_07_23Admin)
admin.site.register(ventasexpres_9656_2019_07_24,
    ventasexpres_9656_2019_07_24Admin)
admin.site.register(ventasexpres_9656_2019_07_25,
    ventasexpres_9656_2019_07_25Admin)
admin.site.register(superlimpio_6468_2019_07_29,
    superlimpio_6468_2019_07_29Admin)
admin.site.register(superlimpio_6468_2019_07_30,
    superlimpio_6468_2019_07_30Admin)
admin.site.register(superlimpio_6468_2019_07_31,
    superlimpio_6468_2019_07_31Admin)
```

OpenGPS/track/models.py

```
from django.db import models
from django.contrib.gis.db import models as geomodels
from django.contrib.auth.models import User
from django.urls import reverse

class Device(models.Model):
    IMEI = models.CharField(max_length=100, blank=False)
    Description = models.CharField(max_length=100, blank=False)
    User = models.ForeignKey(User, on_delete=models.CASCADE)
    def __str__(self):
        return self.Description
    class Meta:
        ordering = ('id',)
        verbose_name_plural = 'Devices'
        pass
class Reg(models.Model):
    Day = models.CharField(max_length=100, blank=False)
    Description = models.ForeignKey(Device, on_delete=models.CASCADE)
    User = models.ForeignKey(User, on_delete=models.CASCADE)
    def get_absolute_url(self):
        return reverse('test-detail', args=[str(self.id)])
class juan_8356_2019_07_03(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class juan_8356_2019_07_05(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class juan_8356_2019_07_06(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class juan_8356_2019_07_09(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class juan_9012_2019_07_13(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
```

```
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class juan_9012_2019_07_16(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class ana_0684_2019_07_17(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class ana_0684_2019_07_18(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class ana_0684_2019_07_19(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class ventasexpres_1847_2019_07_20(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class ventasexpres_1847_2019_07_22(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class ventasexpres_1847_2019_07_23(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class ventasexpres_9656_2019_07_24(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class ventasexpres_9656_2019_07_25(models.Model):
```

```
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class superlimpio_6468_2019_07_29(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class superlimpio_6468_2019_07_30(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
class superlimpio_6468_2019_07_31(models.Model):
    Timestamp = models.CharField(max_length=100, blank=False)
    Latitude = models.FloatField()
    Longitude = models.FloatField()
    Altitude = models.DecimalField(max_digits=4, decimal_places=0)
    Distance = models.DecimalField(max_digits=5, decimal_places=2)
```

OpenGPS/track/urls.py

```
from django.conf.urls import url
from django.urls import include, path
from . import views

urlpatterns = [
        url(r'^$', views.index, name='index'),
        url(r'^device/$', views.DeviceListView.as_view(), name='device_list'),
        url(r'^device/taxi/$',
        views.TaxiListView.as_view(), name='taxi_list'),
        url(r'^device/perro/$',
        views.PerroListView.as_view(), name='perro_list'),
        url(r'^device/gato/$',
        views.GatoListView.as_view(), name='gato_list'),
        url(r'^device/vendedor/$',
        views.VendedorListView.as_view(), name='vendedor_list'),
        url(r'^device/cobrador/$',
        views.CobradorListView.as_view(), name='cobrador_list'),
        url(r'^device/autodeventas/$',
        views.AutodeventasListView.as_view(), name='autodeventas_list'),
```

```
        url(r'^device/1$',
        views.juan_8356_2019_07_03DetailView.as_view(), name='test-detail'),
        url(r'^device/2$',
        views.juan_8356_2019_07_05DetailView.as_view(), name='test-detail'),
        url(r'^device/3$',
        views.juan_8356_2019_07_06DetailView.as_view(), name='test-detail'),
        url(r'^device/4$',
        views.juan_8356_2019_07_09DetailView.as_view(), name='test-detail'),
        url(r'^device/5$',
        views.juan_9012_2019_07_13DetailView.as_view(), name='test-detail'),
        url(r'^device/6$',
        views.juan_9012_2019_07_16DetailView.as_view(), name='test-detail'),
        url(r'^device/7$',
        views.ana_0684_2019_07_17DetailView.as_view(), name='test-detail'),
        url(r'^device/8$',
        views.ana_0684_2019_07_18DetailView.as_view(), name='test-detail'),
        url(r'^device/9$',
        views.ana_0684_2019_07_19DetailView.as_view(), name='test-detail'),
        url(r'^device/10$',
views.ventasexpres_1847_2019_07_20DetailView.as_view(), name='test-detail'),
        url(r'^device/11$',
views.ventasexpres_1847_2019_07_22DetailView.as_view(), name='test-detail'),
        url(r'^device/12$',
views.ventasexpres_1847_2019_07_23DetailView.as_view(), name='test-detail'),
        url(r'^device/13$',
views.ventasexpres_9656_2019_07_24DetailView.as_view(), name='test-detail'),
        url(r'^device/14$',
views.ventasexpres_9656_2019_07_25DetailView.as_view(), name='test-detail'),
        url(r'^device/15$',
views.superlimpio_6468_2019_07_29DetailView.as_view(), name='test-detail'),
        url(r'^device/16$',
views.superlimpio_6468_2019_07_30DetailView.as_view(), name='test-detail'),
        url(r'^device/17$',
views.superlimpio_6468_2019_07_31DetailView.as_view(), name='test-detail'),
        url(r'^device/(?P<pk>\d+)$',
                views.DeviceListView.as_view(), name='test-detail'),
]
```

OpenGPS/track/views.py

```
from django.shortcuts import render
from .models import Device, Reg
from .models import juan_8356_2019_07_03, juan_8356_2019_07_05,
juan_8356_2019_07_06, juan_8356_2019_07_09, juan_9012_2019_07_13,
juan_9012_2019_07_16, ana_0684_2019_07_17, ana_0684_2019_07_18,
```

```
ana_0684_2019_07_19, ventasexpres_1847_2019_07_20,
ventasexpres_1847_2019_07_22, ventasexpres_1847_2019_07_23,
ventasexpres_9656_2019_07_24, ventasexpres_9656_2019_07_25,
superlimpio_6468_2019_07_29, superlimpio_6468_2019_07_30,
superlimpio_6468_2019_07_31
from django.views.generic import DetailView, TemplateView, ListView
from django.views import generic
from django.contrib.auth.mixins import LoginRequiredMixin,
PermissionRequiredMixin
from django.contrib.auth.models import User

def index(request):
    num_devices=Device.objects.all().count()
    num_juan_8356_2019_07_03=juan_8356_2019_07_03.objects.all().count()
    return render(
        request,
        'index.html',
        context={'num_devices':num_devices,
        'num_juan_8356_2019_07_03':num_juan_8356_2019_07_03},
    )
class DeviceListView(LoginRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = Device
class TaxiListView(LoginRequiredMixin, PermissionRequiredMixin,
  generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = Reg
  permission_required = "track.view_juan_8356_2019_07_03"
  queryset = Reg.objects.filter(Description="1")#Taxi
class PerroListView(LoginRequiredMixin, PermissionRequiredMixin,
  generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = Reg
  permission_required = "track.view_juan_8356_2019_07_03"
  queryset = Reg.objects.filter(Description="2")#Perro
class GatoListView(LoginRequiredMixin, PermissionRequiredMixin,
  generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = Reg
  permission_required = "track.view_ana_0684_2019_07_17"
  queryset = Reg.objects.filter(Description="3")#Gato
class VendedorListView(LoginRequiredMixin, PermissionRequiredMixin,
  generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = Reg
```

```
    permission_required = "track.view_ventasexpres_1847_2019_07_20"
    queryset = Reg.objects.filter(Description="4")#Vendedor
class CobradorListView(LoginRequiredMixin, PermissionRequiredMixin,
    generic.ListView):
    login_url = ''
    redirect_field_name = 'redirect_to'
    model = Reg
    permission_required = "track.view_ventasexpres_1847_2019_07_20"
    queryset = Reg.objects.filter(Description="5")#Cobrador
class AutodeventasListView(LoginRequiredMixin, PermissionRequiredMixin,
    generic.ListView):
    login_url = ''
    redirect_field_name = 'redirect_to'
    model = Reg
    permission_required = "track.view_superlimpio_6468_2019_07_29"
    queryset = Reg.objects.filter(Description="6")#Autodeventas

class juan_8356_2019_07_03DetailView(LoginRequiredMixin,
    PermissionRequiredMixin, generic.ListView):
    login_url = ''
    redirect_field_name = 'redirect_to'
    model = juan_8356_2019_07_03
    permission_required = "track.view_juan_8356_2019_07_03"
class juan_8356_2019_07_05DetailView(LoginRequiredMixin,
    PermissionRequiredMixin, generic.ListView):
    login_url = ''
    redirect_field_name = 'redirect_to'
    model = juan_8356_2019_07_05
    permission_required = "track.view_juan_8356_2019_07_03"
class juan_8356_2019_07_06DetailView(LoginRequiredMixin,
    PermissionRequiredMixin, generic.ListView):
    login_url = ''
    redirect_field_name = 'redirect_to'
    model = juan_8356_2019_07_06
    permission_required = "track.view_juan_8356_2019_07_03"
class juan_8356_2019_07_09DetailView(LoginRequiredMixin,
    PermissionRequiredMixin, generic.ListView):
    login_url = ''
    redirect_field_name = 'redirect_to'
    model = juan_8356_2019_07_09
    permission_required = "track.view_juan_8356_2019_07_03"
class juan_9012_2019_07_13DetailView(LoginRequiredMixin,
    PermissionRequiredMixin, generic.ListView):
    login_url = ''
    redirect_field_name = 'redirect_to'
    model = juan_9012_2019_07_13
    permission_required = "track.view_juan_8356_2019_07_03"
class juan_9012_2019_07_16DetailView(LoginRequiredMixin,
    PermissionRequiredMixin, generic.ListView):
    login_url = ''
```

```
  redirect_field_name = 'redirect_to'
  model = juan_9012_2019_07_16
  permission_required = "track.view_juan_8356_2019_07_03"
class ana_0684_2019_07_17DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = ana_0684_2019_07_17
  permission_required = "track.view_ana_0684_2019_07_17"
class ana_0684_2019_07_18DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = ana_0684_2019_07_18
  permission_required = "track.view_ana_0684_2019_07_17"
class ana_0684_2019_07_19DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = ana_0684_2019_07_19
  permission_required = "track.view_ana_0684_2019_07_17"
class ventasexpres_1847_2019_07_20DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = ventasexpres_1847_2019_07_20
  permission_required = "track.view_ventasexpres_1847_2019_07_20"
class ventasexpres_1847_2019_07_22DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = ventasexpres_1847_2019_07_22
  permission_required = "track.view_ventasexpres_1847_2019_07_20"
class ventasexpres_1847_2019_07_23DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = ventasexpres_1847_2019_07_23
  permission_required = "track.view_ventasexpres_1847_2019_07_20"
class ventasexpres_9656_2019_07_24DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = ventasexpres_9656_2019_07_24
  permission_required = "track.view_ventasexpres_1847_2019_07_20"
class ventasexpres_9656_2019_07_25DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = ventasexpres_9656_2019_07_25
```

```
  permission_required = "track.view_ventasexpres_1847_2019_07_20"
class superlimpio_6468_2019_07_29DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = superlimpio_6468_2019_07_29
  permission_required = "track.view_superlimpio_6468_2019_07_29"
class superlimpio_6468_2019_07_30DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = superlimpio_6468_2019_07_30
  permission_required = "track.view_superlimpio_6468_2019_07_29"
class superlimpio_6468_2019_07_31DetailView(LoginRequiredMixin,
  PermissionRequiredMixin, generic.ListView):
  login_url = ''
  redirect_field_name = 'redirect_to'
  model = superlimpio_6468_2019_07_31
  permission_required = "track.view_superlimpio_6468_2019_07_29"
```

OpenGPS/track/templates/base_generic.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <head>
  {% block title %}<title>OpenGPSTracker</title>{% endblock %}
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/
  bootstrap/3.3.7/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/
  jquery.min.js"></script>
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/
  bootstrap.min.js"></script>

  {% load static %}
  <link rel="stylesheet" href="{% static 'css/styles.css' %}">
  <style>
body {
  background-color: #D1D0D5;
}
table{
    background-color: #454447;
```

```
    color: white;
    height: 60px;
}
</style>
</head>
<body>
<h1 style="text-align: center">OpenGPSTracker</h1>
  <div class="container-fluid">
    <div class="row">
      <div class="col-sm-12">
      {% block sidebar %}
      <ul class="sidebar-nav">
        <table style="width:100%">
          <th style="text-align:center; width: 200px">
            <a href="{% url 'index' %}" style="color: white">
            PÃągina de inicio</a></th>
          <th><a href="{% url 'device_list' %}" style="color: white">
          Dispositivos</a></th>
          <th style="text-align:right">
          {% if user.is_authenticated %}
          <li style="color: #D1D0D5">Usuario:
          {{ user.get_username }}</li>

          <li><a href="{% url 'logout'%}?next={{request.path}}"
            style="color: white">Cerrar sesiÃşn</a></li>
          {% else %}
          <li><a href="{% url 'login'%}?next={{request.path}}"
            style="color: white">Iniciar sesiÃşn</a></li>
          {% endif %}
          </th>
        </table>
      </ul>
     {% endblock %}
      </div>
      <div class="col-sm-12">
      {% block content %}{% endblock %}
      </div>
    </div>

  </div>
</body>
</html>
```

OpenGPS/track/templates/index.html

```
{% extends "base_generic.html" %}

{% block content %}

  <h2>Bienvenido a <em>OpenGPSTracker</em></h2>

  <h4><br>
        EstÃąs mirando la interfaz grafica de <em>OpenGPSTracker</em>.
        AquÃ■ podrÃąs visualizar tus dispositivos registrados en el sistema
        y tambiÃl'n el recorrido realizado de los mismos.<br><br><br>
        <center>Ingresa con tu nombre de usuario y contraseÃśa para continuar.
                <br></center>
        <br>
  </h4>

{% endblock %}
```

OpenGPS/track/templates/track/reg_list.html

```
{% extends "base_generic.html" %}

{% block content %}
    <h1>Historial</h1>

        {% if reg_list %}
    {% for reg in reg_list %}
<ul>
<a href="{{ reg.get_absolute_url }}" style="color: black">{{ reg.Day }}</a>
</ul>
    {% endfor %}
    {% else %}
      <p>No se han encontrado dispositivos.</p>
    {% endif %}
{% endblock %}
```

OpenGPS/track/templates/track/device_list.html

```
{% extends "base_generic.html" %}
```

```
{% block content %}
    <h1>Lista de dispositivos</h1>

    {% if device_list %}
<div class = "container" hidden="">
    {% for device in device_list %}
    <li1>{{ device.Description }}</li1>
    <li2>{{ device.IMEI }}</li2>
    {% endfor %}
</div>
<div>
    <ul>
  <p><a id="1" href= "{% url 'taxi_list' %}"
    style="color: black"></a></p>
  <p><a id="2" href= "{% url 'perro_list' %}"
    style="color: black"></a></p>
  <p><a id="3" href= "{% url 'gato_list' %}"
    style="color: black"></a></p>
  <p><a id="4" href= "{% url 'vendedor_list' %}"
    style="color: black"></a></p>
  <p><a id="5" href= "{% url 'cobrador_list' %}"
    style="color: black"></a></p>
  <p><a id="6" href= "{% url 'autodeventas_list' %}"
    style="color: black"></a></p>
  <p><a id="7" href= "" style="color: black"></a></p>
  <p><a id="8" href= "" style="color: black"></a></p>
  <p><a id="9" href= "" style="color: black"></a></p>
  <p><a id="10" href= "" style="color: black"></a></p>
    </ul>
</div>
<p id="demo"></p>
<script>
    var desc = document.querySelectorAll("li1");
    var im = document.querySelectorAll("li2");
    if ( "{{user.get_username}}" == "pablo"){
    document.getElementById("1").innerHTML =
    desc[0].innerHTML + " (IMEI: " + im[0].innerHTML + ")";
    document.getElementById("2").innerHTML =
    desc[1].innerHTML + " (IMEI: " + im[1].innerHTML + ")";
    document.getElementById("3").innerHTML =
    desc[2].innerHTML + " (IMEI: " + im[2].innerHTML + ")";
    document.getElementById("4").innerHTML =
    desc[3].innerHTML + " (IMEI: " + im[3].innerHTML + ")";
    document.getElementById("5").innerHTML =
    desc[4].innerHTML + " (IMEI: " + im[4].innerHTML + ")";
    document.getElementById("6").innerHTML =
    desc[5].innerHTML + " (IMEI: " + im[5].innerHTML + ")";}

      if ( "{{user.get_username}}" == "Juan"){
```

```
        document.getElementById("1").innerHTML =
        desc[0].innerHTML + " (IMEI: " + im[0].innerHTML + ")";
        document.getElementById("2").innerHTML =
        desc[1].innerHTML + " (IMEI: " + im[1].innerHTML + ")";}

          if ( "{{user.get_username}}" == "Ana"){
          document.getElementById("3").innerHTML =
          desc[2].innerHTML + " (IMEI: " + im[2].innerHTML + ")";}

              if ( "{{user.get_username}}" == "Ventasexpres"){
              document.getElementById("4").innerHTML =
              desc[3].innerHTML + " (IMEI: " + im[3].innerHTML + ")";
              document.getElementById("5").innerHTML =
              desc[4].innerHTML + " (IMEI: " + im[4].innerHTML + ")";}

                  if ( "{{user.get_username}}" == "Superlimpio"){
                  document.getElementById("6").innerHTML =
                  desc[5].innerHTML + " (IMEI: " + im[5].innerHTML + ")";}

                      if ( "{{user.get_username}}" == "Katy"){
                      document.getElementById("demo").innerHTML =
                       "No tienes dispositivos registrados";}
</script>
    {% else %}
      <p>No se han encontrado dispositivos.</p>
    {% endif %}
{% endblock %}
```

OpenGPS/track/templates/track/juan_8356_2019_07_03_list.html

```
{% extends "base_generic.html" %}
{% block content %}
{% load leaflet_tags %}
{% leaflet_css %}
{% leaflet_js %}
<!DOCTYPE html>
<style>
    .leaflet-container {  /* all maps */
        width:  100%;
        height: 450px;}
    #specialbigmap {
        height: 800px;}
    /* Resize the "display_raw" textbox */
    .django-leaflet-raw-textarea {
        width: 100%;}
```

```
</style>
<div width="100%" height="100%">
{% leaflet_map "main" callback="map_init" %}
</div>
<div class = "container" hidden>
    {% for juan_8356_2019_07_03 in object_list %}
    <li1>{{ juan_8356_2019_07_03.Timestamp }}</li1>
    <li2>{{ juan_8356_2019_07_03.Latitude }}</li2>
    <li3>{{ juan_8356_2019_07_03.Longitude }}</li3>
    <li4>{{ juan_8356_2019_07_03.Altitude }}</li4>
    <li5>{{ juan_8356_2019_07_03.Distance }}</li5>
    {% endfor %}
</div>
<script type="text/javascript">
    function map_init(map, options) {
    var time = document.querySelectorAll("li1"); //Get data from parraf
    var lat = document.querySelectorAll("li2");
    var lon = document.querySelectorAll("li3");
    var alt = document.querySelectorAll("li4");
    var dist = document.querySelectorAll("li5");
    //var max = lat.length;                    //Var to max points
    var lim = 5000;                   //Var to set max number of points

    var coor = [];                            //Create array to save coor
    for (i = 0; i < lim; i++) {
        coor.push([lat[i].innerHTML, lon[i].innerHTML]);
        var latlng = L.latLng([lat[i].innerHTML, lon[i].innerHTML]);
        }

    var myIcon = L.divIcon({html: ''}); //Set a simple icon for marker
    var points = L.layerGroup([]);
    for (i = 0; i < lim; i++) {
    var a = L.marker([lat[i].innerHTML, lon[i].innerHTML],
     {icon: myIcon}).bindPopup("Hora: " + time[i].innerHTML + "<br />" +
      "Latitud: " + lat[i].innerHTML + "<br />" + "Longitud: " +
      lon[i].innerHTML + "<br />" + "Altitud: " + alt[i].innerHTML +
      " m<br />" + "Distancia: " + dist[i].innerHTML + " m").openPopup();
    points.addLayer(a);
        }
      var overlayMaps = {
      "Mostrar/Ocultar puntos": points
      };
      var baseMaps = {};
     L.control.layers(baseMaps, overlayMaps,
      {collapsed: false, position: 'topleft'}).addTo(map);

    map.setView([0.345,-78.122], 13); //Map view
    var polylineOptions = {             //Polilyne
              color: 'blue',
              weight: 4,
```

```
                opacity: 0.6,
                smoothFactor: 5
            };
    var polyline = new L.Polyline(coor, polylineOptions);
    map.addLayer(polyline);
    }
</script>
</html>
{% endblock %}
```