EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Colima

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

PLATAFORMA IoT PARA EL RASTREO Y MONITOREO REMOTO DE PARÁMETROS DE VEHÍCULOS

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:
ING. ISMAEL VILLAVICENCIO JACOBO

DIRECTOR DE TESIS:
DR. JESÚS ALBERTO VERDUZCO RAMÍREZ

CO-DIRECTOR:
DR. NOEL GARCÍA DÍAZ

VILLA DE ALVAREZ, COLIMA, AGOSTO 2020

Villa de Álvarez, Colima, **4/Septiembre/2020**
Oficio No. **DEPI 1.2.110/120/2020**

ALUMNO VILLAVICENCIO JACOBO ISMAEL
PASANTE DE LA MAESTRÍA EN SISTEMAS COMPUTACIONALES
PRESENTE

La División de Estudios de Posgrado e Investigación de acuerdo al procedimiento para la obtención del Título de Maestría de los Institutos Tecnológicos y habiendo cumplido con todas las indicaciones que la comisión revisora hizo a su trabajo profesional denominado **"PLATAFORMA IOT PARA EL RASTREO Y MONITOREO REMOTO DE PARÁMETROS DE VEHÍCULOS"**, por la opción de tesis, que para obtener el grado de Maestro en Sistemas Computacionales será presentado por Usted, tiene a bien concederle la **AUTORIZACIÓN** de impresión de la tesis citada.

Sin otro particular por el momento, aprovecho la ocasión para enviarle un cordial y afectuoso saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®

RAMONA EVELIA CHÁVEZ VALDEZ
JEFA DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

SEP - TecNM
INSTITUTO TECNOLÓGICO
De Colima
DIVISIÓN DE ESTUDIOS DE
POSGRADO E INVESTIGACIÓN

RECV/cas

C.p. Archivo.

Epígrafe

“Si no estás dispuesto a aprender nadie te puede ayudar. Si estás dispuesto a aprender nadie te puede parar”

Zig Ziglar

Resumen

El rastreo y monitoreo de vehículos es un área que cobra cada vez más interés para organizaciones que requieren administrar en tiempo real y de forma eficiente su parque vehicular. Sin embargo, para pequeñas y medianas empresas el adquirir una solución de este tipo implica una inversión importante que no todas pueden solventar. Este trabajo describe el desarrollo de una plataforma Web automatizada para gestionar, rastrear y monitorizar el seguimiento y los parámetros relacionados con la funcionalidad y eficiencia del vehículo. Para este fin se utiliza un dispositivo IoT especializado, el cual consiste en un módulo con interfaz On Board Diagnostic II y tecnologías GPS/GSM/GPRS, que recopila y transmite los datos al servidor central para su publicación y análisis en tiempo real. El sistema Web ha sido desarrollado siguiendo las pautas de la metodología de desarrollo del Proceso Unificado Ágil y utiliza tecnologías Web de código abierto como HTML, CSS y JavaScript, el framework Spring MVC basado en java como entorno de la aplicación, Mapbox y OpenStreetMap para la interpretación de datos geográficos y PostgreSQL como gestor de Base de datos. Este sistema contribuye a eficientar la gestión, aumentar la productividad y favorecer a la mejora continua de la administración del parque vehicular, aportando soluciones de bajo costo.

Abstract

Vehicle tracking and monitoring is an area that is gaining more and more interest for organizations that need to manage their vehicle fleet in real time and efficiently. However, for small and medium-sized companies, acquiring a solution of this type implies a significant investment that not all can afford. This work describes the development of an automated Web platform to manage, track and monitor the tracking and parameters related to the functionality and efficiency of the vehicle. For this purpose, a specialized IoT device is used, which consists of a module with On Board Diagnostic II interface and GPS / GSM / GPRS technologies, which collects and transmits the data to the central server for publication and analysis in real time. The Web system has been developed following the guidelines of the Agile Unified Process development methodology and uses open source Web technologies such as HTML, CSS and JavaScript, the Java-based Spring MVC framework as the application environment, Mapbox and OpenStreetMap for the interpretation of geographic data and PostgreSQL as a Database manager. This system contributes to making management more efficient, increasing productivity and favoring the continuous improvement of vehicle fleet management, providing low-cost solutions.

Índice general

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I INTRODUCCIÓN

1.1 Introducción

La administración eficiente de las flotas vehiculares pertenecientes a cualquier organización representa un reto complejo y difícil de lograr. Actividades tales como, definir el momento ideal para aplicar mantenimiento preventivo y correctivo, auditar el buen uso de las unidades, evitar los robos de combustible y componentes, dar seguimiento puntual a los recorridos, entre otras, requiere de recursos y esfuerzos constantes de la parte administrativa. Existen herramientas que se han aplicado para este objetivo, de las que podemos mencionar, bitácoras de seguimiento, software de mantenimiento, controles de todo tipo, sin embargo, todas ellas presentan ciertas carencias que reducen la efectividad de las medidas de control.

Con el advenimiento de las tecnologías relacionadas al IoT, actualmente, es posible monitorizar las actividades y los parámetros técnicos de funcionamiento y operación de vehículos. Estas herramientas son de gran interés para muchas organizaciones que cuentan con parques vehiculares, porque posibilitan el conocimiento en tiempo real de ciertos datos, tales como la posición, el desplazamiento, el estado general de los vehículos, la temperatura del motor, el nivel de combustible, la velocidad, las revoluciones por minuto, etc. Los beneficios que se pueden obtener con la aplicación de esta tecnología, son la mejora de la seguridad, la reducción de los gastos de operación, el control del consumo de combustible, incremento en la productividad y en la prevención de percances; en términos generales mejorar la eficiencia de los procesos administrativos involucrados.

1.2 Naturaleza del problema

Una problemática común en las organizaciones públicas o privadas es el gasto excesivo de los recursos asignados al control y mantenimiento de sus flotas vehiculares. Entre otros factores, ocasionados por el incumplimiento de rutas e itinerario definidos, infracciones al reglamento de tránsito, inseguridad, robo de combustible, falta de mantenimiento preventivo, el uso incorrecto de las unidades y el seguimiento puntual, todo esto con la finalidad de asegurar el rendimiento óptimo de la unidad, así como mejorar el control de los gastos de operación.

Recientemente, numerosas empresas adquieren y utilizan soluciones basadas en sistemas de monitoreo y rastreo satelital, sin embargo; por su alto costo no son accesibles principalmente para pequeñas y medianas empresas. Otro factor que desmotiva la adquisición de estas soluciones, es el grado de generalidad con el que han sido desarrolladas, que ocasiona problemas de adaptación a las necesidades específicas de estas pequeñas organizaciones. Este proyecto de tesis está encaminado a satisfacer esa área de oportunidad.

1.3 Revisión de la literatura

Aljaafreh et al., (2011) describen un sistema de adquisición de datos vehiculares para automatizar la gestión de flotas. La solución expuesta propone un sistema implementado en dos partes; el sistema embebido, el cual se instala en el interior del vehículo y un sistema Web de registro y publicación de datos. El sistema embebido obtiene la ubicación del vehículo desde el receptor GPS (Global Positioning System), el estado del vehículo desde la interfaz OBD (On-Board Diagnostics) y la identificación del conductor desde una etiqueta RFID (Radio-Frequency Identification), esta parte del sistema recopila los datos durante la operación del vehículo y envía la información al servidor Web a través de la red WIFI, una vez que el vehículo regresa al lugar de estacionamiento. Esta investigación tiene varias aplicaciones prácticas como optimizar la distribución de

vehículos y conductores, informar sobre parámetros del motor y supervisar el comportamiento del conductor a través de eventos de seguimiento como aceleración rápida y frenado brusco, además de ser un sistema de bajo costo. Sin embargo, una debilidad importante es el método de transmisión, este no es muy eficiente, ya que dicha operación se realiza cuando el vehículo regresa a su lugar de origen imposibilitando la gestión de flotillas en tiempo real.

Fuad & Drieberg (2013) desarrollaron un prototipo de un sistema de seguimiento remoto de vehículos, utilizando el sistema global para comunicaciones móviles (GSM) y la API de Google Maps. Esta solución consiste en enviar la ubicación del vehículo a través del servicio SMS de un teléfono móvil, el modem GSM (U-blox EVK-G26H) instalado en el centro de control recibe ese mensaje y actualiza la información de ubicación en una base de datos MySQL. La información de ubicación se muestra mediante la API de Google Maps incorporado en la aplicación Web. Además, proporciona tres funciones; la ubicación más reciente del vehículo rastreado, el historial de rutas y el planificador de rutas. El hecho de que esta solución esté básado en un teléfono móvil, permite que se adapte a cualquier objeto que requiera rastreo sin la necesidad de una inversión costosa.

En el proyecto de tesis de Conza Berrocal (2013) desarrolló una aplicación georreferencial, cuyo propósito es rastrear y monitorear una flota vehicular para una compañía de taxis. La solución que propone, es asignar a cada vehículo un equipo móvil Android con una aplicación que obtiene las coordenadas de ubicación, hora y velocidad; dichos parámetros se envían mediante la red celular GSM/GPRS de manera automática a una estación central. En la estación central se procesan los parámetros y son almacenados en la base de datos del servidor Web; al mismo tiempo se envían esos datos a las máquinas cliente mediante el uso de websockets, que en ese momento están rastreando los vehículos en tiempo real, la ubicación de los vehículos se visualiza en un mapa que provee Google Maps. La solución que presenta es bastante útil, sencilla y fácil de implementar con pocos recursos, no obstante, una limitación destacable es que la información la provee directamente el teléfono móvil y no propiamente el vehículo,

si en algún momento el dispositivo se extrajera del vehículo este no trasmitirá información correcta.

En la tesis de Montero Revelo (2016) se implementó un sistema Web para monitorear remotamente el estado del motor de un automóvil. Como solución realizó dos subsistemas, el primero tiene el propósito de recolectar y transmitir los datos, el segundo gestiona las funcionalidades y visualiza el monitoreo de los parámetros del automóvil, en el que además permite la configuración dinámica de los parámetros que se desean monitorear del dispositivo OBD. Esta solución es una alternativa que permite cumplir con el objetivo planteado, no obstante, su principal desventaja es el sistema de recolección de datos, ya que requiere un conjunto de dispositivos para cumplir con su propósito.

Desai & Phadke (2017) desarrollaron un sistema para monitorear la ubicación y parámetros de vehículos de prueba para el equipo de investigación y desarrollo de una compañía automotriz. El trabajo que implementaron consiste en un pequeño sistema embebido integrado por un Arduino Mega 2560, un módulo GPS y una tarjeta SIM808 GSM/GPRS, este obtiene los datos del vehículo y su posicionamiento los cuales son transferidos al servidor a través de la tecnología GSM/GPRS de una operadora móvil, la información enviada al servidor se almacena en una base de datos y se visualiza en un sistema Web, los datos de ubicación se visualizan con los mapas de Google Maps. El artículo menciona la lectura de parámetros del vehículo como el nivel de combustible y la temperatura del motor que son enviados al servidor de la misma manera que los datos de ubicación, sin embargo, no hace mención de cómo estos datos son obtenidos, limitándose solamente al monitoreo de la ubicación del vehículo.

Othmane et al., (2018) desarrollaron un prototipo de un sistema de bajo costo para el monitoreo y la automatización de la gestión de flotas. Los autores proponen un sistema que consiste en cuatro componentes el recopilador de datos de flota, el servicio de gestión de flota, la base de datos de gestión y la ampliación Web. El recolector de datos obtiene la información del vehículo mediante un conector OBD II y extrae las coordenadas GPS con el dispositivo Adafruit Ultimate GPS, ambos

conectados a una tarjeta PiCAN 2 Raspberry Pi 3, para la trasmisión de información se utiliza un dispositivo Holograma Nova el cual permite la transferencia de los datos a través de la red celular al servicio de gestión de flota, este los almacena en la base de datos de gestión y son visualizados en una aplicación Web, desarrollada para este fin en tiempo real. El sistema Web gestiona funcionalidades como el registro de los datos del vehículo, personalización de parámetros a visualizar, información en tiempo real de ubicación, velocidad, y temperatura. Una de las ventajas principales de este sistema es la adquisición de los dispositivos, que como mencionan los autores son de bajo costo. A diferencia, el sistema propuesto implementa un único dispositivo que integra tecnología IoT (OBD, GPS, GSM/GPRS) para la adquisición y transferencia de datos.

En resumen, en el análisis de la revisión de la literatura, los trabajos relacionados principalmente se caracterizan por construir una arquitectura compuesta de dispositivos independientes para la recolección, almacenamiento y envío de datos, aunque son alternativas que se pueden implementar, no aseguran un funcionamiento adecuado para cumplir con el objetivo planteado, debido a que no se ha comprobado su factibilidad y confiabilidad al no ser dispositivos industriales que garanticen precisión. Además, carecen de funcionalidades de gestión que en este proyecto se tienen considerados implementar. Ante ese escenario, se propone el desarrollo de un sistema de información integral, que mediante un dispositivo con tecnología especializada permita monitorizar en tiempo real los parámetros funcionales, de operación y de desplazamiento de vehículos, contribuyendo así a mejorar la eficiencia de la administración de cualquier flotilla vehicular.

1.4 Descripción de la solución

Para abordar la solución del problema descrito en párrafos anteriores, en este proyecto de tesis se pretende desarrollar la arquitectura mostrada en la Figura 1.1, desarrollando un sistema que monitorice en tiempo real los parámetros relacionados con los sensores de la ECU (Engine Control Unit) del motor de un

vehículo, el rastreo y seguimientos de rutas y geocercas, así como el cumplimiento de las reglas de tránsito, lo cual permitirá mejorar la administración y control de los parques vehiculares en conjunto con los recursos operativos de la organización.

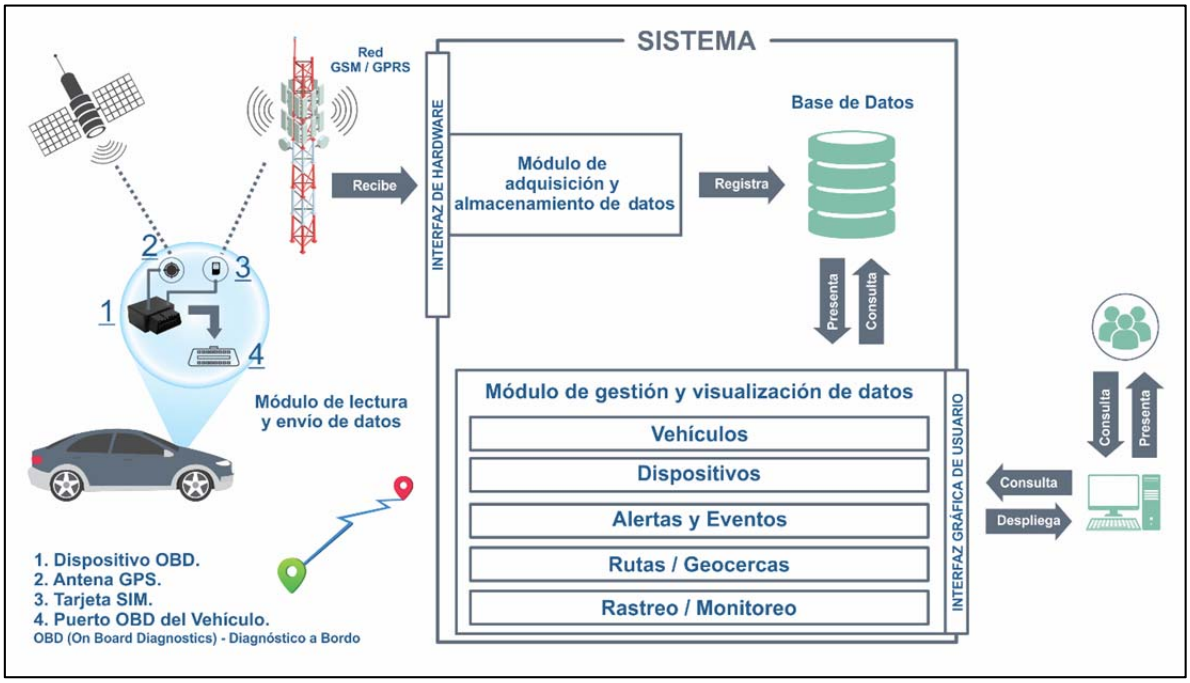

Figura 1.1 Esquema general de la propuesta de solución (Fuente: Elaboración propia).

La solución consiste de tres módulos.

- Módulo de lectura y envío de datos. Este módulo consiste en un dispositivo IoT industrializado que permite adquirir los parámetros técnicos y de posicionamiento del vehículo, dicho dispositivo es un OBD IDD-213GD, con tecnología integrada (GPS y GSM/GPRM), el cual se interconecta en el puerto OBDII del vehículo. Este es el encargado de recolectar los datos y enviarlos hacia la plataforma Web utilizando la red de telefonía celular.

- Módulo de adquisición y almacenamiento de datos. Este módulo recibe los datos recolectados por los dispositivos OBD residentes en cada uno de los

vehículos, la información es recibida en formatos XML y JSON y es transformada para identificar a la unidad, posteriormente prepara dichos datos para ser almacenados en la Base de Datos.

- Módulo de gestión y visualización de datos. Este módulo gestiona la información relacionada con el rastreo y monitoreo de vehículos. Está integrado por los siguientes componentes:

    - Gestión de vehículos. Componente que permite el registro de las unidades vehiculares.
    - Gestión de dispositivos. Componente que permite el registro de los dispositivos OBD y de relacionarlos a dicha unidad.
    - Configuración de alertas y eventos. Componente que permite habilitar/deshabilitar las alertas y eventos por cada dispositivo.
    - Gestión de rutas, sub-rutas y geocercas. Componente que permite la creación de rutas, sub-rutas y geocercas mediante punto georreferénciales en el mapa.
    - Despacho de vehículos. Componente que permite generar el itinerario del conductor mediante la asignación de rutas y geocercas.
    - Visualización del rastreo y monitoreo. Componente que permite visualizar la información en tiempo real de la unidad vehicular.

1.5 Motivación

La automatización de los procesos se presenta como un reto tecnológico atractivo que permite aplicar las nuevas tecnologías, para la mejora continua de las tareas que se hacen de manera tradicional en las organizaciones, que impiden elevar la competitividad y rentabilidad de los negocios.

En el caso del sistema a desarrollar, la organización que cuente con tal desarrollo tecnológico podrá tomar mejores decisiones con respecto a la administración de su flota vehicular, tomando como base los datos arrojados por el sistema.

Además del costo y adaptabilidad del sistema para pequeñas y medianas organizaciones, los principales beneficios que se obtendrían son:

- Optimizar los procesos administrativos de la organización.
- Incrementar la productividad.
- Mejorar la seguridad de los vehículos y conductores.
- Conocer en tiempo real la ubicación y el estado general del vehículo.
- Conocer el recorrido de cada unidad que compone la flota vehicular.
- Reducir los gastos de operación, combustibles y mantenimiento.
- Reducir y detectar percances.

1.6 Justificación

El desarrollo de este proyecto busca beneficiar la administración de flotas vehiculares para pequeñas y medianas empresas, se justifica por las siguientes razones:

- No existe un sistema centralizado que permita monitorizar los parámetros de los automóviles de manera remota en tiempo real y de bajo costo.
- Con la información obtenida se podrán tomar mejores decisiones con respecto a la seguridad, al mantenimiento del parque vehicular definiendo las condiciones y limitaciones con las que trabajará el vehículo.
- Contribuirá a la eficiencia en el uso de los recursos de operación vehicular.

1.7 Objetivos

A continuación, se describe el objetivo general y los objetivos específicos de este trabajo.

1.7.1 Objetivo general

Implementar un sistema de información basado en tecnologías de Internet de las Cosas, que permita monitorizar los parámetros funcionales, de operación y de desplazamiento de vehículos.

1.7.2 Objetivos específicos

- Estudiar el campo de conocimiento de la temática abordada.
- Analizar y definir los requerimientos del proyecto.
- Modelar los módulos de la solución.
  - Módulo de adquisición de parámetros funcionales y del posicionamiento GPS del automóvil por medio de la interfaz OBD.
  - Desarrollar el módulo de comunicación de datos.
  - Integración de la API de recepción y elaboración de parámetros técnicos de diagnóstico y posicionamiento.
  - Módulo de administración, visualización del monitoreo y rastreo del parque vehicular.
- Implementar los módulos del sistema siguiendo las pautas obtenidas en la fase del modelado.
- Poner en operación el sistema y realizar pruebas de funcionamiento.
- Medir el impacto de este desarrollo tecnológico.
- Publicar los resultados de la investigación.
- Redactar la documentación del proyecto.

1.8 Hipótesis

El desarrollo de una plataforma IoT para el rastreo y monitoreo remoto de parámetros en vehículos hará posible disponer en tiempo real de la información relacionada con los parámetros funcionales, de operación y de desplazamiento de vehículos contribuyendo a mejorar la eficiencia la administración de una flotilla vehicular.

1.8.1 Variable independiente

El desarrollo de una plataforma IoT para el rastreo y monitoreo remoto de parámetros en vehículos

1.8.2 Variable dependiente

Hará posible disponer en tiempo real de la información relacionada con los parámetros funcionales, de operación y de desplazamiento de vehículos contribuyendo a mejorar la eficiencia la administración de una flotilla vehicular.

1.9 Métodos y herramientas empleados

1.9.1 Método de investigación

El presente trabajo de tesis conlleva implementar la investigación aplicada, ya que su finalidad según (Vargas Cordero, 2009) es resolver o mejorar una situación específica o particular de una institución o empresa, mediante una aplicación o propuesta, a fin de poner en práctica los conocimientos que se han adquirido. Además, el desarrollo del proyecto de tesis se sitúa en la metodología de investigación cuantitativa porque el enfoque de este método consiste en la recolección de datos para probar hipótesis o teorías con base en la medición numérica y análisis estadístico. En conclusión, este trabajo se puede considerar como una investigación mixta (Hernández Sampierí , Fernández Collado, & Baptísta Lucio, 2006).

1.9.2 Técnicas e instrumentos de recolección de datos

Como principal técnica de recolección de datos se utilizó la entrevista, ya que es el principal medio para la obtención de requisitos en la ingeniería de requerimientos. La entrevista permite la comunicación interpersonal, mediante una conversación estructurada con diferentes personas, con el fin de conocer información cualitativa y subjetiva de diferentes actividades.

1.9.3 Metodología de desarrollo de la aplicación Web

Para el desarrollo del sistema, se utilizó la metodología Proceso Unificado Ágil (PUA) basada en una versión simplificada de la metodología Rational Unified

Process (RUP), (Pressman, 2010). El objetivo principal de esta metodología es asegurar la producción de software de alta calidad con base a las necesidades de los usuarios finales, proporciona un método sistemático de diseño, desarrollo e implementación de artefactos mediante iteraciones que son representados a través de diagramas estandarizados del Lenguaje Unificado de Modelado (UML). (Object Management Group, Inc. , 2020).

1.9.4 Herramientas de desarrollo

Con base a la investigación realizada del entorno de desarrollo, se determinó usar las siguientes herramientas: Spring MVC v4 basado en Java para el desarrollo y maquetado del sistema Web en conjunto con librerías de JavaScript, hojas de estilos CSS y Bootstrap, este último para agilizar el desarrollo de front-end. PostgreSQL como gestor de base de datos. Las tecnologías mencionadas se seleccionaron por su confiabilidad en cuanto a seguridad, diversa documentación y por ser tecnologías de código abierto.

1.10 Aportaciones de esta tesis

El desarrollo de este proyecto de tesis aportará un producto tecnológico que puede ser utilizado por cualquier organización interesada en administrar de manera efectiva, incrementar la seguridad, y hacer un uso eficiente de los recursos de su parque vehicular.

1.11 Organización de este documento

Este documento de tesis se organiza en los siguientes apartados:

Capítulo I "Introducción", describe el contexto general del proyecto de tesis, así como se establecen los objetivos, hipótesis del trabajo y la justificación.

Capítulo II "Estado del campo del conocimiento", se presenta el marco histórico, contextual y teórico de los proyectos similares existentes.

Capítulo III "Métodos empleados", se describe la metodología utilizada en el desarrollo de este proyecto de tesis.

Capítulo IV "Desarrollo del proyecto", se presenta detalladamente la implementación de la arquitectura y la configuración del software.

Capítulo V "Resultados", se analiza la información de los resultados obtenidos.

Capítulo VI "Conclusiones y trabajos a futuro", se describen las conclusiones finales sobre el proyecto y se proporcionan algunas pistas para su continuación.

CAPÍTULO II ESTADO DEL CAMPO DEL CONOCIMIENTO

El campo del estudio de conocimiento permite conocer el contexto que aborda el tema de investigación, desde los posibles orígenes hasta la situación actual. En este capítulo se aborda el marco histórico, el marco contextual y el marco teórico.

2.1 Marco histórico

Los sistemas de rastreo de vehículos inician en 1996, cuando el presidente de los Estados Unidos Bill Clinton firmó una directiva para que el sistema GPS se convirtiera en un servicio público (U.S. Department of Transportation, 2015), dando origen a los primeros sistemas de rastreo de vehículos, estos utilizaban hardware en el interior del vehículo, tal el caso de (Valencia Ruiz, 1997) que implementó un sistema localizador de vehículos basado en esa tecnología, utilizando la red de telefonía celular para el envío de datos, en ese entonces el Sistema Telefónico Móvil Avanzado o *Advanced Mobile Phone System (AMPS).*

Por otro lado, el sistema de diagnóstico a bordo OBD I se implementó en 1988, como medida para la reducción de la contaminación del aire por la California Air Resources Board, de Estados Unidos, determinando que todos los automóviles deben contar con ese sistema (Camarillo Alfaro) .El OBD I fue considerado de gran ayuda para la regulación de gases contaminantes.

Debido a nuevas medidas de contaminación en 1996 se adoptó al OBD II que a diferencia del OBD I detecta fallos eléctricos, químicos y mecánicos que pueden afectar al nivel de emisiones del vehículo. Desde ese entonces es utilizado para detectar fallas principalmente en talleres mecánicos.

Para el año 2000 surge el modelo de distribución de software SaaS (Software como servicio), lo cual permitió brindar servicio a través de la nube computacional, a partir de entonces, las aplicaciones de rastreo satelital comenzaron a tomar forma (Pérez Villegas, 2018).

Principalmente gracias al gran avance tecnológico, actualmente existen plataformas que utilizan este tipo de tecnologías y que ofrecen sistemas de gestión de flotas, por ejemplo: WEBFLET (Webfleet Solutions B.V., 2019), GEOTAB (Geotab Inc, 2019) , INDACAR (Indacar, 2019), entre muchas otras.

2.2 Marco contextual

En el estado de Colima son cada vez más las organizaciones que operan con unidades vehiculares por lo que se ha incrementado el parque vehicular un 4.72%, con respecto al 2017 (INEGI, 2018). Dichas organizaciones han requerido monitorizar y rastrear sus parques vehiculares, la gran mayoría lo realiza de manera no automática impidiendo una buena administración y control de la flotilla vehicular. Ante la carencia del control, la administración de recursos, los gastos de operación, la seguridad y productividad se ven afectados.

En la Figura 2.1 se observa la zona conurbada Colima- Villa de Álvarez el cual es el escenario principal, en donde las unidades de una pequeña organización fueron sujetas a ser monitoreadas durante el desplazamiento rutinario de cada una de ellas.

Figura 2.1 Zona conurbada Colima-Villa de Álvarez (INEGI Mapa digital, 2020).

2.3 Marco teórico

El objetivo de este apartado es conceptualizar todos los aspectos que forman parte global del desarrollo del proyecto, aquellas tecnologías y herramientas utilizadas que permiten comprender el problema y su solución.

El Internet de las Cosas o *Internet of Things* (IoT) por sus siglas en inglés, es un término que surgió en el Instituto de Tecnología de Massachusetts y hace referencia (Barrio Andrés, 2018) a la conexión de objetos cotidianos a Internet, que intercambian, agregan y procesan información sobre su entorno físico con el fin de proporcionar servicios de valor agregado a los usuarios finales, a partir de esto surgen las plataformas IoT como base para que diversos dispositivos estén interconectados y así permita automatizar el entorno de los usuario finales. Las plataformas IoT son capaces de generar un ecosistema propio, según (McClelland, 2017) consisten en: Hardware, Conectividad, Software e Interfaz de usuario. Por lo tanto, esta tecnología satisface las necesidades de este proyecto de tesis.

2.3.1 Tecnologías de hardware

Los componentes de hardware que impactan en el desarrollo de este proyecto consisten en dispositivos con tecnología OBD, GPS y GSM/GPRS. El término OBD corresponde a las siglas en inglés de *On Board Diagnostic* o “Diagnostico a bordo”, es un sistema de monitoreo que permite diagnosticar las condiciones de operación de un automóvil, accediendo a una serie de señales que la computadora central del vehículo registra y envía a un puerto en forma de códigos de error (Staff Editorial de Electrónica y Servicio, 2014). De esta forma, es posible conocer el estado general del vehículo.

El sistema de Posicionamiento Global GPS (por sus siglas en inglés *Global Position System*), es un sistema de radionavegación basado en el espacio, que proporciona servicios fiables de posicionamiento, navegación y cronometría, localizando cualquier objeto en la superficie terrestre, por consiguiente, en el marco de este proyecto, permite obtener las coordenadas geográficas del

vehículo (National Coordination Office for Space-Based Positioning, Navigation, and Timing, s.f.)

GSM / GPRS (*Sistema Global para las Comunicaciones Móviles o en inglés Global System for Mobile Communications*), es un estándar de comunicaciones móviles digital orientada a circuitos, constituye la segunda generación (2G) de sistemas móviles, diseñado originalmente para la transmisión de voz, pero también es capaz de transmitir datos. Con el objetivo de aumentar la velocidad de transmisión en la red GSM original, se introdujo el Servicio General de Paquetes vía Radio o por sus siglas en inglés *General Packet Radio Service (GPRS)*, lo que dio origen al Internet en los teléfonos móviles. De acuerdo a lo anterior, este sistema permite la transferencia de los datos recolectados a un servidor central en tiempo real. (Becvar, Mach, & Pravda).

2.3.2 Tecnologías de software

Las tecnologías de desarrollo Web han estado evolucionando constantemente y gracias a esto han surgido una gran cantidad de tecnologías que facilitan el desarrollo de aplicaciones en ambiente Web, tales como: frameworks, librerías, arquitecturas de diseño e incluso nuevos lenguajes que al mismo tiempo se han estado mejorando.

Los patrones de diseño permiten a los desarrolladores diseñar partes específicas de subsistema (Deitel & Deitel, 2003), con el objetivo de resolver un problema particular, de tal forma que pueda ser reutilizado. Uno de los patrones arquitectónicos que se ha vuelto muy popular en los últimos años es el patrón MVC (Modelo Vista Controlador) (Jaramillo Valbuena, Augusto Cardona, & Villa Zapata, 2008), el cual permite separar la interfaz de usuario de los datos y de la parte lógica, este se divide en tres capas:

- Modelo: representación de los datos y reglas de negocio, encargado de manejar un registro de las vistas y controladores.
- Vista: Muestra la información del modelo en un formato adecuado para la interpretación del usuario.

- Controlador: Corresponde a la respuesta de los eventos provocador por el usuario, dando correcta gestión a las entradas del usuario.

Spring MVC (Pivotal Software, Inc, 2018), es un framework Web de código abierto utilizado para la construcción de este proyecto basado en java, el entorno de trabajo facilita la integración directamente con tecnologías de representación basadas en plantillas, como JSP, HTML, XML, JSON entre otros, y ofrece funcionalidades que facilita el desarrollo de aplicaciones Web. Se optó utilizar este framework porque adquiere un nivel de seguridad alto y es compatible con diversos sistemas por ser un lenguaje multiplataforma.

Al ser un desarrollo Web implica la integración de diversas tecnologías tales como:

- HTML (*Hypertext Markup Language*) que sirve para estructurar el contenido de la página mediante etiquetas.
- CSS *(Cascading Style Sheets)* lenguaje de estilos para el diseño y presentación de la página.
- JavaScript permite que el contenido de las páginas sea dinámico, mejorando la interactividad, velocidad y el uso de aplicaciones.
- Ajax (*Asynchronous JavaScript And XML*) consiste en la comunicación asíncrona con el servidor.
- WebSockets establece la comunicación continua entre el cliente y servidor con el fin de que se puedan implementar funciones en tiempo real.

Por otro lado, Bootstrap (Bootstrap, 2011), es un framework de front-end de código abierto, este contiene un conjunto de herramientas para agilizar el desarrollo de aplicaciones o prototipos.

PostgreSQL(PostgreSQL Global Development Group, 2019), es un Sistema Gestor de Base de Datos (SGBD) relacional de código abierto que utiliza y amplía el Lenguaje SQL permitiendo almacenar y escalar las cargas de trabajo complicadas de forma segura.

MapBox (Mapbox, 2019), es una plataforma de georreferenciación para aplicaciones móviles y Web, permite la integración de funciones como mapas,

búsqueda y navegación considerando las necesidades del usuario. Esta herramienta se integra a las tecnologías Web usadas en el proyecto para la interpretación de datos georreferenciados.

Servicios Web

Para el intercambio de información en la plataforma, entre el cliente y el servidor se utiliza los servicios Web (Gisbert Vercher, 2015). Es una tecnología que usa un conjunto o agrupación de protocolos que sirven para cambiar o intercambiar datos entre diferentes aplicaciones, están disponibles para los usuarios que hacen peticiones hacia el servidor Web mediante una Interfaz de Programación de Aplicaciones (API).

CAPÍTULO III MÉTODOS EMPLEADOS

En este capítulo se describe la metodología utilizada en el desarrollo de este proyecto de tesis, partiendo de la revisión de la literatura, el diseño, modelado e implementación hasta la documentación del sistema, contemplando la metodología PUA.

El desarrollo del sistema y modelado se llevó a cabo a través de la definición del modelo conceptual mostrado en la Figura 1.1, se definieron siete iteraciones de tres semanas de duración cada una, las actividades se ordenaron de manera secuencial priorizando los requerimientos más significativos, incorporando de manera iterativa y transversal los siguientes puntos:

3.1 Revisión bibliográfica relacionada con el problema

En este punto se realizó una investigación documental del problema a investigar con la finalidad de comprender, conocer y detectar áreas de oportunidad a través de medios científicos, tecnológicos e institucionales, como son los artículos científicos, revistas tecnológicas y tesis de grado.

3.2 Naturaleza de la metodología aplicada

La naturaleza de la metodología PUA consta de cuatro fases y siete disciplinas, en la Figura 3.1 se muestra la distribución del flujo de trabajo.

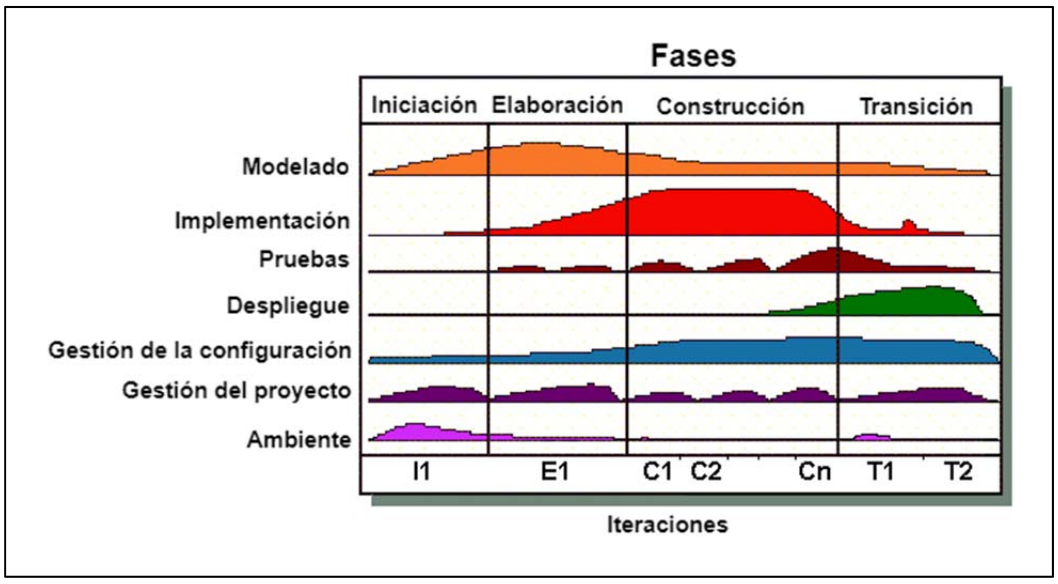

Figura 3.1 Ciclo de Vida del Proceso Unificado Ágil (PUA) (Ambler, 2005).

Las fases representan un ciclo de desarrollo en el ciclo de vida de un producto de software, las fases son las siguientes (Ambler, 2005) :

- Concepción: identificar el alcance del proyecto.
- Elaboración: identificar y probar la arquitectura del sistema.
- Construcción: crear software de mayor prioridad para los interesados de forma regular e incremental.
- Transición: implementar y validar el sistema en su ambiente de producción.

Las disciplinas se realizan de manera iterativa en cada una de las fases y de manera transversal en las fases de Configuración y Administración de cambios, gestión del proyecto y entorno de producción, las disciplinas son las siguientes (Ambler, 2005) :

- Modelado. Comprender el contexto del negocio de la organización, el dominio del problema e identificar una solución viable a través de representaciones UML.
- Implementación. El modelado UML se transforma a código ejecutable.
- Pruebas. Ejecutar pruebas objetivas para garantizar la calidad del proyecto.

- Despliegue. Se planifica la entrega de los incrementos del sistema, para que estos estén disponibles y se puedan ejecutar planes de acción para obtener retroalimentación.
- Gestión de la configuración. Administra el seguimiento y el acceso a las versiones de los artefactos, así como el control y gestión de los cambios.
- Gestión de proyectos. Dirige las actividades en el proyecto, incluye la gestión de riesgos, la dirección y coordinación de personas, esto para asegurar que el proyecto esté dentro del presupuesto y se entregue en tiempo.
- Ambiente. Coordina y apoya el esfuerzo para asegurar los procesos, incorpora estándares, herramientas y tecnologías necesarias para el equipo de desarrollo.

En la etapa de modelado se desarrollaron los principales diagramas que permiten definir la arquitectura del sistema, los dominios del negocio y el problema:

- Modelo de requisitos define las necesidades y el alcance del sistema, tareas asociadas con el descubrimiento, la evaluación, el registro, la documentación y la validación para un proyecto en particular (Sparx Systems Pty Ltd., 2020).

- Modelo de casos de uso describe la funcionalidad propuesta del sistema y un caso de uso representa la interacción entre el usuario (humano maquina) y el sistema (Sparx Systems Pty Ltd (2020).

- Modelo de clases representa la estructura básica del sistema, según (Pressman,2010) el diagrama de clases aporta una visión estática de un sistema sin mostrar la naturaleza dinámica de las comunicaciones entre los objetos y clases (Pressman, 2010).

- Modelo de datos representa la estructura y relación detallada de los datos que contienen el sistema, se conforma por entidades que son objetos exclusivos del mundo real que se controla, dichas entidades contienen atributos que son simplemente características de la entidad. (IBM, 2020).

- Diagrama de navegación representa de forma visual la estructura lógica de la aplicación para plasmar la organización de la información. (Medina Martínez, Alonso Guzmán, Hernández Alarcón, & Móndragon Gómez, 2013).

- Modelo de interfaces representa el prototipo de diseño en el que los usuarios pueden interactuar con el sistema; (Pressman, 2010) menciona que el diseño de la interfaz es un medio eficaz de comunicación entre los seres humanos y la computadora. A nivel prototipado se crean escenarios de usuarios y formatos de pantalla que en cada iteración van modificando.

- Modelo de despliegue tiene como objetivo representar la distribución física del sistema a través de nodos interconectados, (Alonso F, Martínez Normand, & Segovia Pérez, 2005).

3.3 Levantamiento de requerimientos y modelado

Con base a la metodología para el desarrollo de software mencionada en el punto anterior 3.2, se determinaron los requerimientos funcionales y no funcionales, a partir de estos, se identificaron los usuarios del sistema, se diseñaron prototipos de modelado como: el diagramas de casos de uso, el modelo de interfaces, diagrama de clases y diagrama de datos, utilizando la herramienta Enterprise Architect (Sparxsystems, s.f), representados en el lenguaje UML, que en cada iteración de la fase de análisis fueron mejorados.

3.4 Diseño de los módulos que integran la solución

Partiendo de los requerimientos y diagramas del modelado, se determinó el hardware y software necesario para el diseño de los siguientes módulos:

- Módulo de lectura de parámetros del vehículo.
- Módulo de transmisión y recepción de datos.
- Sistema Web de gestión de monitoreo y vehículos.

3.5 Desarrollo del sistema

De acuerdo a los módulos establecidos en el diseño, se comenzó a desarrollar partiendo de aquellos que son de mayor relevancia, los cuales son:

- Instalación de los módulos de lectura y transmisión de datos en los vehículos.
- Recuperación y procesamiento de datos en formato XML/JSON mediante un servidor Web.
- Visualización de datos a nivel de consola.
- Desarrollo del sistema Web de gestión y monitoreo, permitiendo el registrar, consultar, modificar y eliminar:
    - Usuarios
    - Vehículos
    - Dispositivos OBD
    - Rutas
    - Geocercas

3.6 Realización de pruebas funcionales

Se realizaron pruebas unitarias y de integración a cada función y módulo del sistema para garantizar el funcionamiento del software, lo que contribuye a reforzar la calidad y el cumplimiento de los requerimientos del usuario. Se utilizó el proceso de pruebas de tres etapas mostrado en la Figura 6.3 donde se ponen a

prueba primero los componentes del sistema, en seguida con el sistema integrado y posteriormente con los datos del cliente. El proceso puede ser iterativo ya que requiere la repetición de otras etapas en el proceso al detectar errores(Somerville,2011).

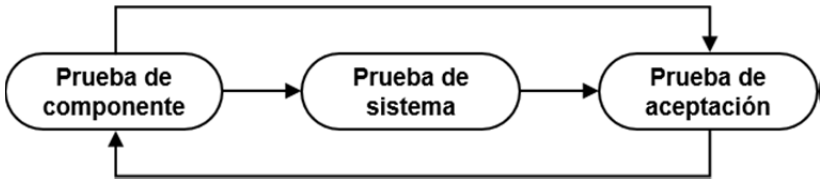

Figura 3.2 Etapas de pruebas (Somerville, 2011).

3.7 Puesta en operación del sistema

Como primera fase, la implementación del sistema se realizó en una plataforma que ofrece servicios de computación en la nube llamada Heroku a modo de prueba.

3.8 Estudio del impacto de proyecto

En esta fase se realizó la reflexión acerca de los beneficios que se obtuvieron con la implementación y puesta en marcha del sistema.

3.9 Documentación del proyecto

Finalmente se redactó el presente documento de tesis en el que se especifica el proceso de desarrollo del proyecto.

CAPÍTULO IV DESARROLLO DEL PROYECTO

4.1 Introducción

En este capítulo se describe el proceso de desarrollo del proyecto de tesis apegándose a la metodología PUA, la cual está basado en el Lenguaje de Modelado Unificado UML.

4.2 Modelado

En esta etapa, el objetivo es determinar detalladamente las especificaciones del sistema, esto mediante catálogos clasificados de requerimientos y modelos para la interpretación y seguimiento del desarrollo del proyecto, con el fin de cubrir las necesidades de los usuarios finales.

Partiendo del modelo conceptual, se obtiene información detallada de los requisitos funcionales y no funcionales, se elaboró el modelo de caso de uso para describir las acciones de cada uno de los actores que interactúan con el sistema, como se muestra en los siguientes puntos.

4.2.1 Modelo de requisitos

Los requerimientos se dividen en funcionales y no funcionales; (Sommerville, 2011) define que los requerimientos funcionales son enunciados de los servicios que el sistema debe de proveer y de cómo debería de comportarse y de reaccionar a situaciones específicas, los requerimientos no funcionales son limitaciones sobre servicios o funciones que ofrece el sistema. Estos requerimientos se obtuvieron a partir de las reuniones periódicas con el usuario final.

4.2.1.1 Requisitos funcionales

Los requerimientos se pueden observar en representación gráfica, estos contienen un identificador único y su relación mediante la dependencia derivada. El

identificador del requerimiento se representa con RF (Requerimiento Funcional) agregando el número consecutivo del requerimiento.

En la Figura 4.1 se observa el diagrama de requisitos para empresas y posteriormente en la Tabla 4.1 se muestran los detalles de cada uno de ellos.

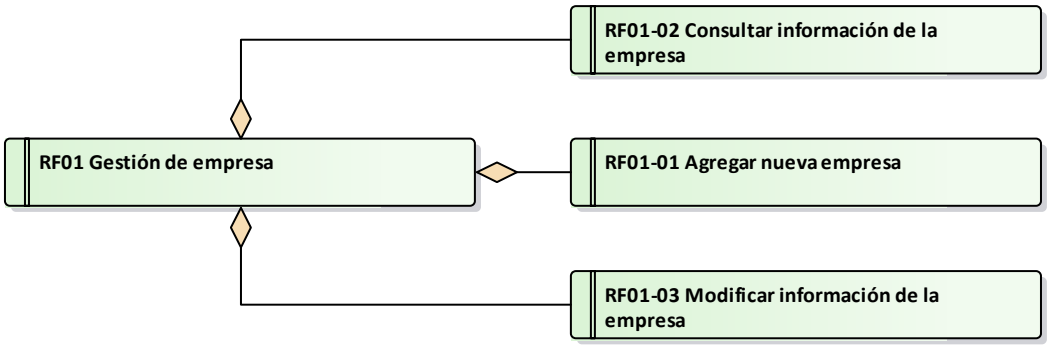

Figura 4.1 RF01 – Empresas (Fuente: Elaboración propia).

Tabla 4.1 Especificaciones del RF01 (Fuente: Elaboración propia).

01	**Empresa**		
Id	Descripción	Prioridad	Condición
RFN01	El sistema deberá permitir registrar nuevas empresas, así como consultar y modificar su información.	Alta	Requerido
RF01-01	El sistema deberá permitir dar de alta una nueva empresa con los siguientes atributos: • *Nombre* • *Giro* • *Domicilio* • *Teléfono*	Alta	Requerido
RF01-02	El sistema deberá permitir consultar la información de la empresa	Alta	Requerido
RF01-03	El sistema deberá permitir modificar la información de la empresa	Alta	Requerido

En la Figura 4.2 se observa el diagrama de requisitos para gestionar usuarios y posteriormente en la Tabla 4.2 se muestran los detalles de cada uno de ellos, dicho diagrama y tabla representa las funcionalidades necesarias para dar de alta un usuario al sistema con los privilegios requeridos.

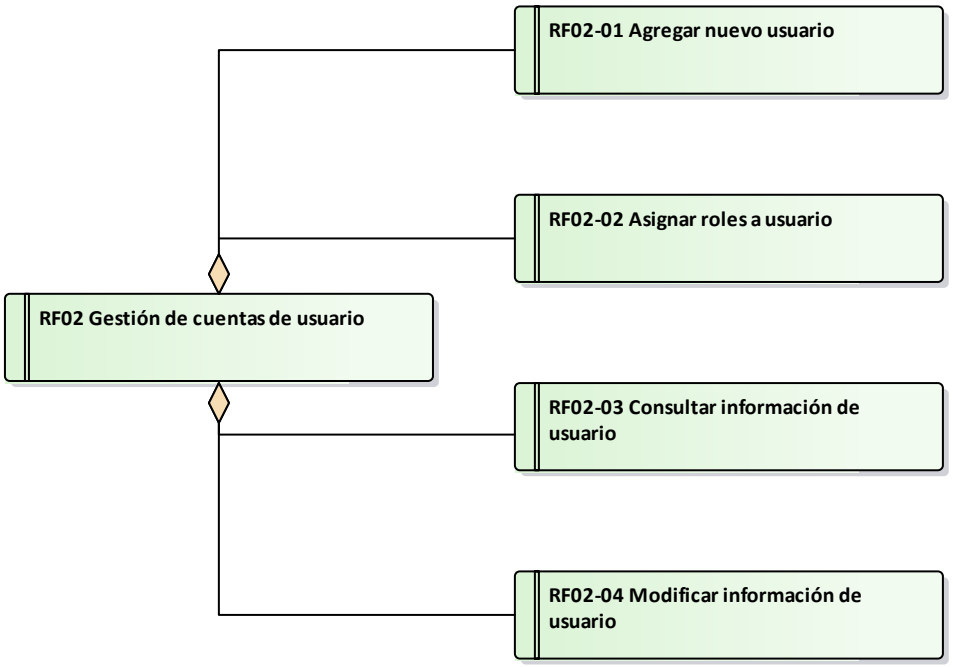

Figura 4.2 RF02 Gestión de usuarios (Fuente: Elaboración propia).

Tabla 4.2 Especificaciones del RF02 (Fuente: Elaboración propia).

02	**Gestión de usuarios**		
Id	Descripción	Prioridad	Condición
RF02	El sistema deberá permitir registrar nuevos usuarios, asignar roles, así como consultar y modificar su información.	Alta	Requerido
RF02-01	El sistema deberá permitir dar de alta nuevos usuarios, considerando los tipos de roles, “Administrador” y “Operador”, estos registrados por el “SuperAdministrador”, con los siguientes atributos: • *Nombre completo* • *Usuario* • *Contraseña*	Alta	Requerido

	• *Correo* • *Teléfono* • *Empresa* • *Tipo de usuario* • *Última sesión*		
RF02-02	El sistema deberá permitir asignar roles a cada usuario que se dé de alta en la plataforma.	Alta	Requerido
RF02-03	El sistema deberá permitir a los usuarios "Administrador" consultar la información de los usuarios registrados bajo su dominio.	Alta	Requerido
RF02-04	El sistema deberá permitir modificar la información del usuario. Solo el "SuperAdministrador" tendrá privilegios de modificar, activar/desactivar y eliminar (lógicamente) el usuario.	Alta	Requerido

En la Figura 4.3 se observa el diagrama de requisitos para el control de acceso al sistema y en la Tabla 4.2 se muestran los detalles de cada uno de ellos. El diagrama se asocia con la Figura 4.1 gestión de usuarios ya que su principal función es acceder al sistema.

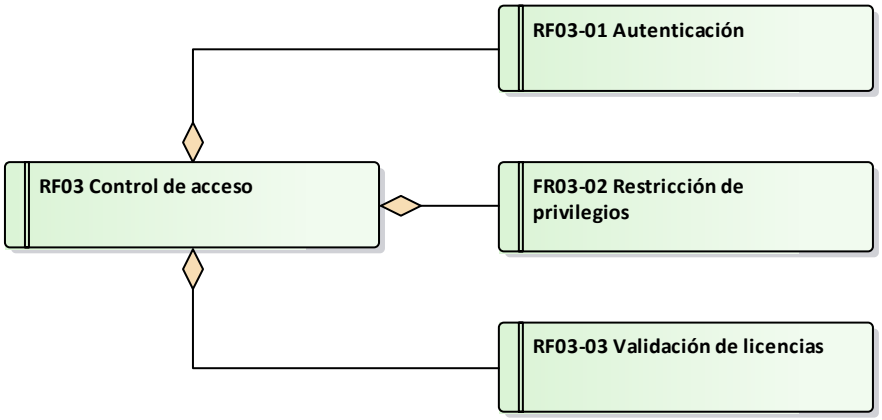

Figura 4.3 RF03 - Control de acceso (Fuente: Elaboración propia).

Tabla 4.3 Especificaciones del RF03 (Fuente: Elaboración propia).

03	**Control de acceso**		
Id	Descripción	Prioridad	Condición
RF03	El sistema deberá permitir la autenticación de usuario, restringir privilegios y validar si se encuentra con licencia activa.	Alta	Requerido
RF03-01	El sistema deberá permitir la autenticación del usuario mediante un nombre de usuario y una contraseña.	Alta	Requerido
RF03-02	El sistema deberá permitir visualizar y modificar la información de acuerdo a los privilegios de cada tipo de usuario	Alta	Requerido
RF03-03	El sistema deberá permitir la validación de la licencia adquirida. Si la licencia se encuentra inactiva solo debe mostrar la información histórica bajo la siguiente leyenda: “Su licencia de uso de software ha expirado comuníquese con su proveedor de servicios”.	Media	Deseable

En la Figura 4.4 se observa el diagrama de requisitos para la gestión de unidades vehiculares y en la Tabla 4.4 se muestran las funcionalidades pertenecientes a cada uno de ellos. Cabe destacar que este diagrama representa la parte principal del sistema pues es el objeto a monitorear.

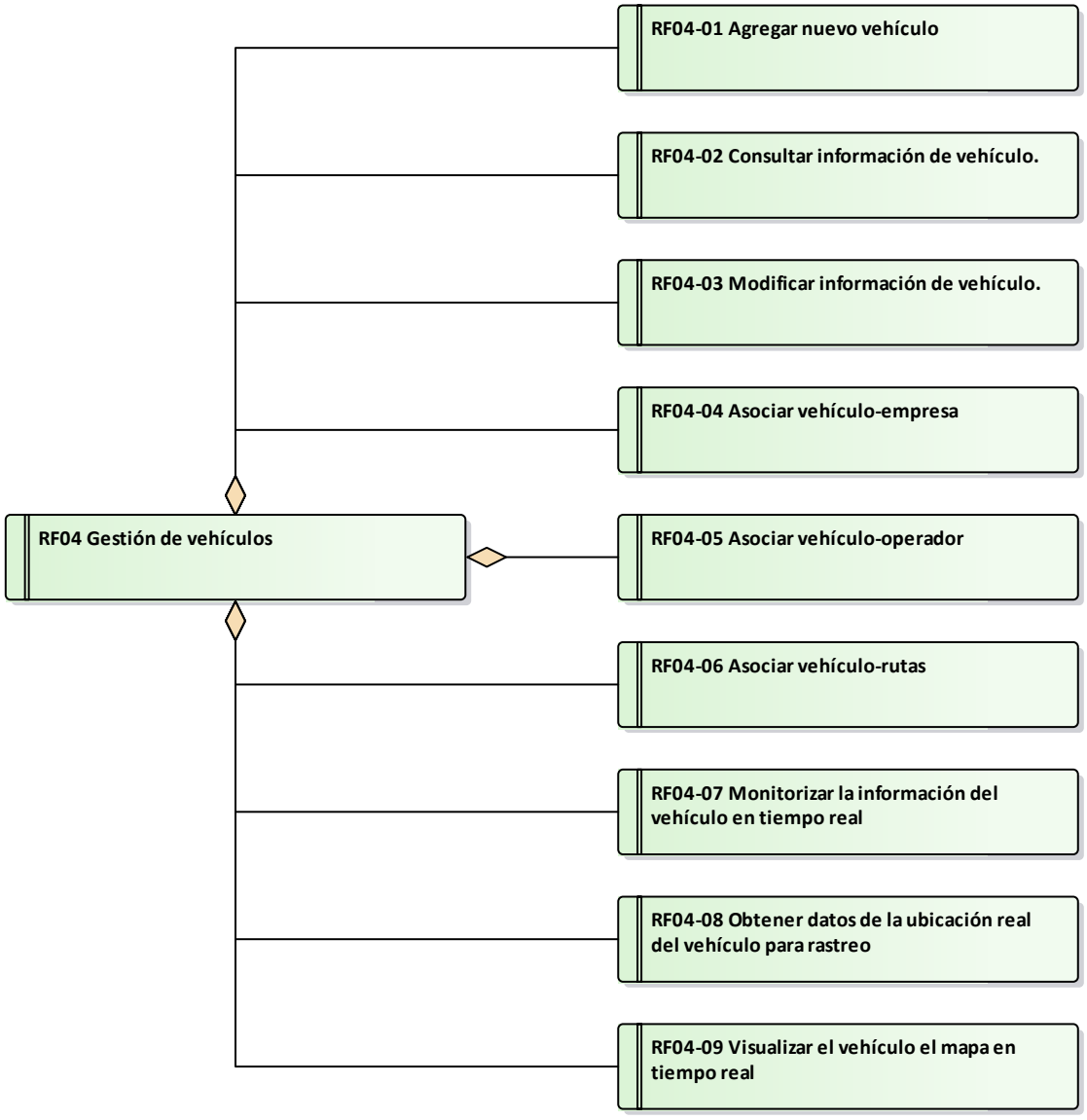

Figura 4.4 RF04 - Gestión de vehículos (Fuente: Elaboración propia).

Tabla 4.4 Especificaciones del RF04 (Fuente: Elaboración propia).

04	**Gestión de vehículos**		
Id	Descripción	Prioridad	Condición
RF04	El sistema deberá permitir registrar, consultar, modificar y asociar el vehículo. Además de monitorizarlo y rastrearlo en tiempo real.	Alta	Requerido

RF04-01	El sistema deberá permitir registrar un nuevo vehículo con los parámetros siguientes: • *Nombre* • *Placa* • *Modelo* • *Marca* • *NIV (Número de identificación vehicular)* • *Color* • *Imagen* • *Tipo de vehículo* o *Vehículos de transporte público de cuota fija y/o por recorrido.* o *Vehículos de atención a emergencias.* o *Vehículos de transporte de carga.* o *Vehículos de transporte público de cuota variable* • *Empresa* • *Operador* • *Cilindraje*	Alta	Requerido
RF04-02	El sistema deberá permitir consultar cualquier información referente al vehículo.	Alta	Requerido
RF04-03	El sistema deberá permitir modificar la información del vehículo y habilitar/deshabilitar si está o no en operación.	Alta	Requerido
RF04-04	El sistema deberá permitir asociar el vehículo a una empresa, con el fin de conocer qué vehículos pertenecen a la empresa.	Alta	Requerido
RF04-05	El sistema deberá permitir asociar el vehículo con el operador perteneciente a la empresa, con el fin de conocer que usuario es el que opera la unidad.	Alta	Requerido
RF04-06	El sistema deberá permitir asociar una o varias rutas a un vehículo,	Alta	Requerido
RF04-07	El sistema debe permitir mostrar la información de cada vehículo que se	Alta	Requerido

	seleccione. La información que mostrará a manera de mensaje y la siguiente: • *Operador* • *Nombre* • *Placa* • *Velocidad* Además, deberá contar con una liga “Ver más” la que llevará a visualizar toda la información del vehículo.		
RF04-08	El sistema deberá proveer una interfaz para poder recibir información de un vehículo a rastrear.	Alta	Requerido
RF04-09	El sistema debe mostrar los vehículos en el mapa por medio de la interfaz de MapBox con la ubicación real de cada uno de ellos de acuerdo a lo siguiente para cada uno de los usuarios: • *Para “Administrador” podrá visualizar todos los vehículos con los que cuente.* • *Operador solo podrá visualizar el vehículo al que se encuentra asignado.*	Alta	Requerido

En la Figura 4.5 se observa el diagrama de requisitos para la gestión de dispositivos OBD el cual se relaciona principalmente con la Figura 4. pues es parte fundamental para la recolección y envío de información. En la Tabla 4.5 se detalla las funcionalidades que debe considerar.

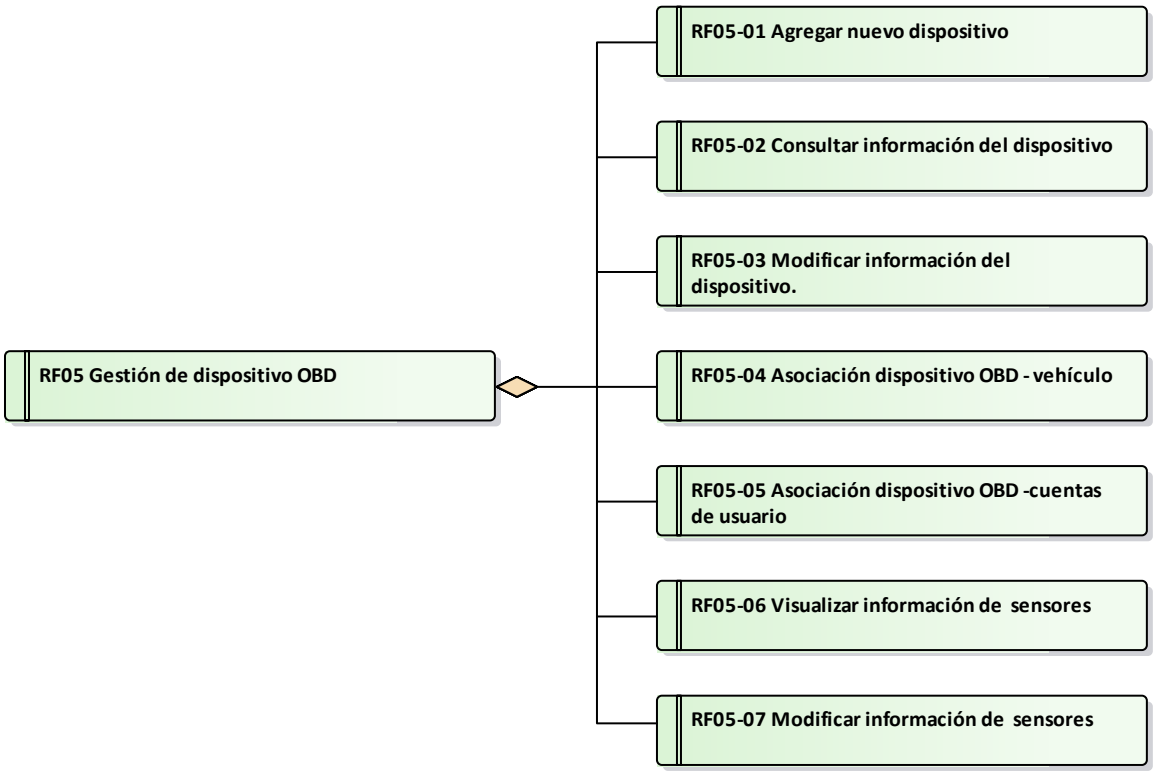

Figura 4.5 RF05 - Dispositivo OBD (Fuente: Elaboración propia).

Tabla 4.5 Especificaciones del RF05 (Fuente: Elaboración propia).

05	**Dispositivo OBD**		
Id	Descripción	Prioridad	Condición
RF05	El sistema deberá permitir registrar, consultar, modificar y asociar dispositivos OBD a los vehículos y cuentas de usuarios. Además de visualizar y habilitar/deshabilitar los sensores del mismo.	Alta	Requerido
RF05-01	El sistema deberá permitir registrar un nuevo dispositivo con los siguientes parámetros: • *Número de serie* • Número de tarjeta SIM • Parámetros OBD	Alta	Requerido
RF05-02	El sistema deberá permitir consultar la información del dispositivo OBD.	Alta	Requerido
RF05-03	El sistema deberá permitir modificar la información del dispositivo OBD.	Alta	Requerido
RF05-04	El sistema deberá permitir asociar un	Alta	Requerido

	dispositivo OBD con un vehículo, cuando este se requiera cambiar de asociación la siguiente información se perdería: *Información del perfil asociado (recorridos, alertas y eventos).*		
RF05-05	El sistema deberá permitir asociar un dispositivo a tres cuentas de usuario las cuales son: • Administrador • Operador	Alta	Requerido
RF05-06	El sistema permitirá visualizar los sensores que el dispositivo GPS sea capaz de detectar además de su comportamiento en el tiempo.	Alta	Requerido
RF05-07	El sistema permitirá habilitar y deshabilitar la visualización de los sensores que el dispositivo GPS se capaz de detectar	Alta	Requerido

En la Figura 4.6 se observa el diagrama de requisitos para la creación de rutas y sub rutas, En la Tabla 4.6 se detalla las funcionalidades que debe considerar cada requisito.

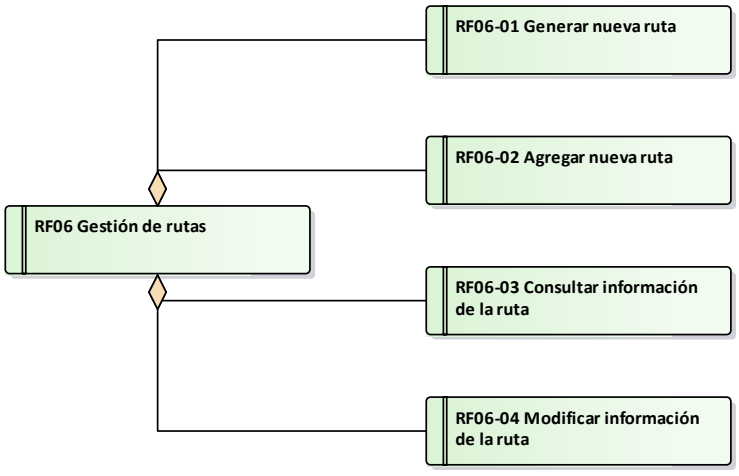

Figura 4.6 RF06 Rutas / Sub-rutas (Fuente: Elaboración propia).

Tabla 4.6 Especificaciones del RF06 (Fuente: Elaboración propia).

06	**Rutas / Sub-rutas**		
Id	Descripción	Prioridad	Condición
RF06	El sistema deberá permitir generar, registrar, consultar, modificar rutas/sub-rutas.	Alta	Requerido
RF06-01	El sistema deberá permitir generar rutas en el mapa, trazando origen y destino.	Alta	Requerido
RF06-02	El sistema deberá permitir registrar nueva ruta con los siguientes parámetros: • *Nombre* • *Descripción* • *Datos de geolocalización origen / destino* • *Estado*	Alta	Requerido
RF06-03	El sistema deberá permitir consultar la información de la ruta.	Alta	Requerido
RF06-04	El sistema deberá permitir modificar la información de la ruta, Así como habilitar/deshabilitar el estado de la misma.	Alta	Requerido

En la Figura 4.7 se observa el diagrama de requisitos para la creación de geocercas y en la Tabla 4.7 se detalla las funcionalidades que debe considerar cada requisito. Dicho diagrama se relaciona estrechamente con la Figura 4.4 gestión de vehículos pues cada unidad se asigna a una geocerca, respectivamente.

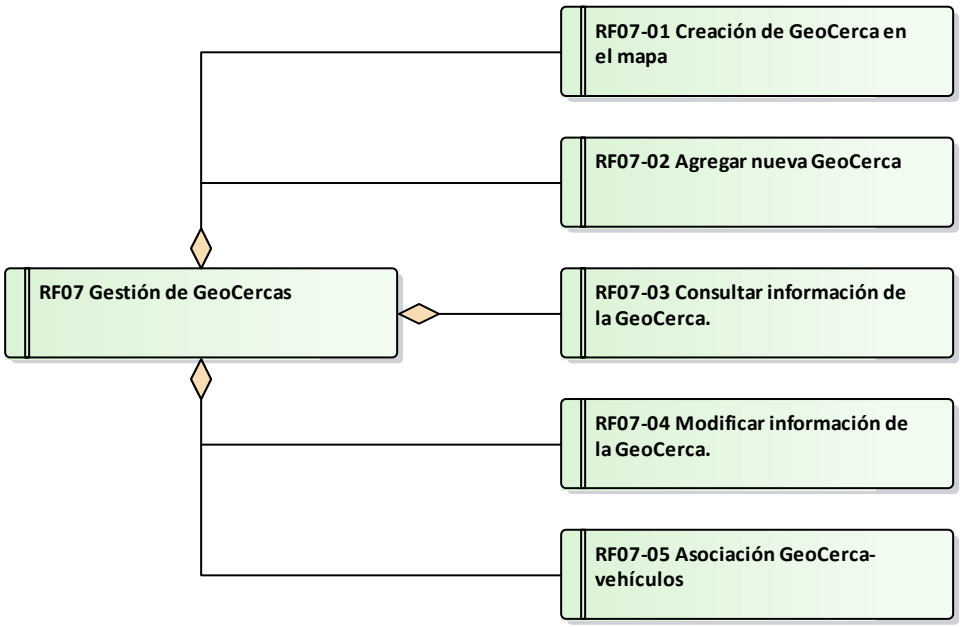

Figura 4.7 RF07 Geocercas (Fuente: Elaboración propia).

Tabla 4.7 Especificaciones del RF07 (Fuente: Elaboración propia).

07	**Geo cercas**		
Id	Descripción	Prioridad	Condición
RF07	El sistema deberá permitir crear, registrar, consultar y modificar geocercas en el mapa así mismo asignar vehículos a estas.	Alta	Requerido
RF07-01	El sistema deberá permitir crear una geocerca en el mapa de Google Maps	Alta	Requerido
RF07-02	El sistema deberá permitir registrar una nueva geocerca con los siguientes atributos: • Nombre • Estado • Datos de geolocalización	Alta	Requerido
RF07-03	El sistema deberá permitir consultar la información de la geocerca.	Alta	Requerido
RF07-04	El sistema deberá permitir modificar la información de la geocerca. Así como activar/desactivar el estado de esta.	Alta	Requerido
RF07-05	El sistema deberá permitir asociar uno o varios vehículos a la geocerca.	Alta	Requerido

En la Figura 4.8 se observa el diagrama de requisitos para el despacho de vehículos, cabe destacar que este diagrama se relaciona con la Figura 4.4 gestión de vehículos, 4.6 rutas y sub rutas y 4.7 geocercas, ya que es un componente principal, es decir, al despachar la unidad se le asigna las rutas que debe de seguir y la geocerca en la que tiene que permanecer. En la Tabla 4.8 se detalla las funcionalidades de la Figura 4.8

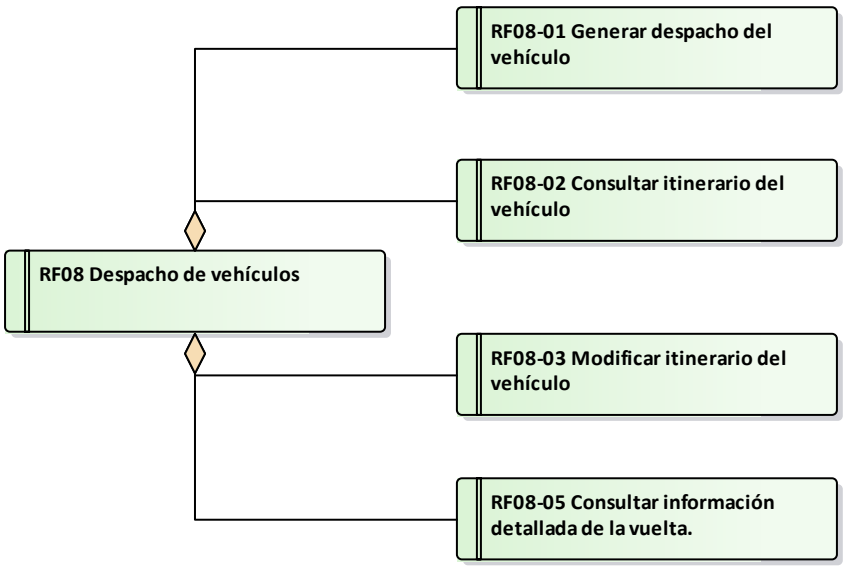

Figura 4.8 RF08 Despacho e Itinerario (Fuente: Elaboración propia).

Tabla 4.8 Especificaciones del RF08 (Fuente: Elaboración propia).

08	**Despacho de vehículos**		
Id	Descripción	Prioridad	Condición
RF08	El sistema deberá permitir el despacho de vehículos y generar, consultar y modificar el itinerario que debe seguir cada uno de ellos. Además de registrar los recorridos fuera de servicio.	Alta	Requerido
RF08-01	El sistema deberá permitir realizar el despacho de vehículos con los siguientes parámetros: • Vehículo • Conductor/Operador	Alta	Requerido

	• Rutas y Sub-rutas • Geocerca • Fecha/hora inicio		
RF08-02	El sistema deberá permitir mostrar la información del itinerario que debe seguir cada uno de los vehículos despachados.	Alta	Requerido
RF08-03	El sistema deberá permitir modificar la información del itinerario.	Alta	Requerido
RF08-04	El sistema deberá permitir consultar la información detallada de cada una de las vueltas que realiza la unidad.	Alta	Requerido

En la Figura 4.9 se observa el diagrama de requisitos para gestionar las alertas y eventos, este diagrama se relaciona estrechamente con la Figura 4.4 gestión de vehículos y 4.5 dispositivos pues la principal función es notificar de cualquier percance mediante alertas en el sistema. En la Tabla 4.9 se detalla las funcionalidades que debe considerar cada requisito.

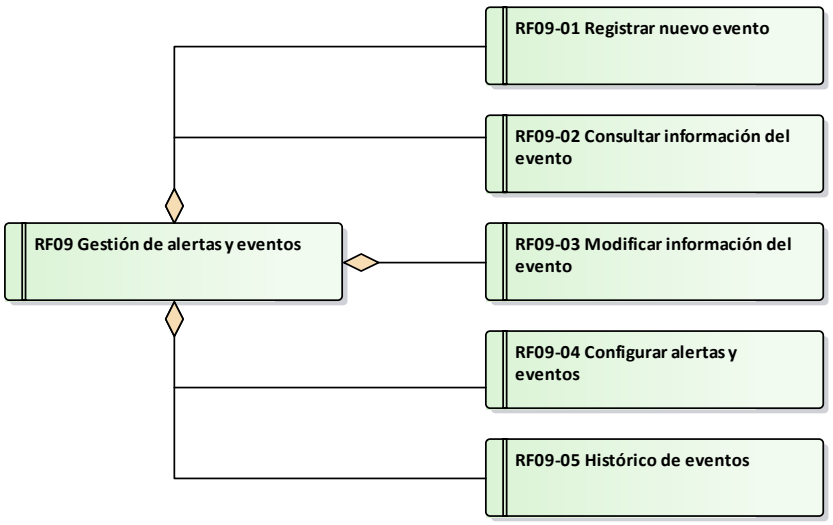

Figura 4.9 RF09 Alertas y eventos (Fuente: Elaboración propia).

Tabla 4.9 Especificaciones del RF09 (Fuente: Elaboración propia).

09	**Alertas y eventos**		
Id	Descripción	Prioridad	Condición
RF09	El sistema deberá permitir registrar, consultar, modificar y configurar eventos, a partir de estos generar alertas. Deberá indicar en el mapa en que parte de la ruta se generó dicho evento mostrando la información dando clic sobre el evento sucedido en el mapa.	Alta	Requerido
RF09-01	El sistema deberá permitir registrar un nuevo evento, con los siguientes parámetros: • *Nombre* • *Código* • *Descripción*	Alta	Requerido
RF09-02	El sistema deberá permitir consultar la información del evento.	Alta	Requerido
RF09-03	El sistema deberá permitir modificar la información del evento.	Alta	Requerido
RF09-04	El sistema deberá permitir configurar los eventos que el dispositivo puede detectar y definir en cada una de ellas en qué momento se debe generar una alerta, así mismo, activarlas o desactivarlas, esto debe ser cuando sucedan cualquiera de los siguientes eventos: • Motor encendido • Motor apagado • Temperatura del refrigerante del motor • Exceso de velocidad • Alto nivel de RPM (revoluciones por minuto) • Aceleración difícil • Desaceleración difícil • Frenado duro • Giro a la izquierda (vuelta cerrada) • Cambio de carril rápido • Choque • Geocercas • Salida de la Geocerca • Entrada a la Geocerca	Alta	Requerido

	• Bajo voltaje de la batería • Tiempo excesivo de inactividad del motor • Remolcado • Vibración • Fatiga de conducción • MIL (Luz indicadora de mal funcionamiento) encendida/apagada. o Código de diagnóstico o Fecha y hora o Dónde ocurrió • Borrado de código de diagnóstico • Combustible bajo • Dispositivo conectado Dispositivo desconectado • Señal perdida • Señal recuperada • Dispositivo conectado en otro vehículo • Robo de combustible • Retraso de itinerario • Vehículo fuera de ruta		
RF09-04	El sistema deberá contar con un histórico de las alertas y eventos que hayan sucedido en el vehículo clasificándolas de la siguiente manera: • Manejo • Vehículo • Seguridad	Alta	Requerido

4.2.1.2 Requisitos no funcionales

El identificador del requerimiento no funcional se representa con RNF (Requerimiento No Funcional) agregando el número consecutivo del requerimiento.

En la Figura 4.10 se observa el diagrama de requisitos no funcionales y posteriormente en la Tabla 4.10 se muestra las especificaciones de cada uno de ellos.

Figura 4.10 Requisitos no funcionales (Fuente: Elaboración propia).

Tabla 4.10 Listado de requisitos no funcionales (Fuente: Elaboración propia).

Id	Descripción	Prioridad	Condición
RNF01	Se deberá registrar el usuario que inserte y/o actualice los registros de la base de datos, además registrar la fecha en que se crea y modifica.	Media	Requerido
RNF02	Deberá evitarse utilizar nombres, abreviaciones de empresas para la asignación de nombres o uso en documentación que será entregado al usuario final.	Media	Requerido
RNF03	Cada excepción propia, el nombre de los log's deberá ser: <Nombre corto de la entidad de software>ddMMyy.reg	Media	Requerido
RNF04	El sistema deberá ser desarrollado en el siguiente entorno: • Java 8 con Spring MVC • HIBERNATE • PostgreSQL para base de datos	Media	Requerido
RNF05	Se deberá encriptar los datos que contengan: • Información de configuraciones. • Información personal. • Contraseñas de usuarios. • Información de cuentas o de identificación de pertenencias del usuario final.	Media	Requerido
RNF06	Las aplicaciones Web y de escritorio deberán ser responsivas.	Media	Requerido
RNF07	Los mensajes del sistema como alertas o información deberán ser congruentes con la acción y descritos en lenguaje natural para que sean comprensibles para el lector.	Media	Requerido
RNF08	La solución deberá brindar un performance óptimo.	Media	Requerido
RNF09	El tiempo de respuesta máximo que pueda ser percibido por el usuario final deberá ser de 10 segundos	Media	Requerido
RNF10	El esfuerzo para el mantenimiento o actualización de las funcionalidades deberá ser bajo, por lo que se requiere que la organización del código fuente sea modular y que cumpla con los lineamientos de codificación de la	Media	Requerido

	empresa.		
RNF11	Todas las creaciones, lecturas, actualizaciones y eliminaciones de datos que involucren más de una tabla deberán de hacerse por medio de procedimientos almacenados o funciones de la base de datos.	Media	Requerido
RNF12	El sistema debe almacenar la información de los vehículos solamente por 12 meses, pasado este tiempo, se eliminará el registro.	Media	Requerido

4.2.2 Modelo de caso de usos

En la Figura 4.12 se puede observar el modelo de caso de usos general, el cual representan la forma en que los actores interactúan con el sistema, incluye las asociaciones principales de dispositivos a vehículos, la configuración de alertas y eventos, la asociación de rutas, geocercas, itinerarios con los usuarios, en con junto con las principales operaciones de cada uno de ellos. En la Figura 4.11 se muestran los actores que intervienen en el sistema.

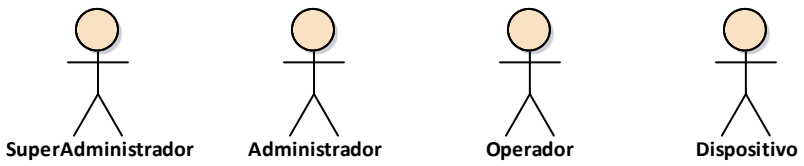

Figura 4.11 Actores del sistema (Fuente propia).

SuperAdministrador: es el encargado de administrar todo el sistema, registrar empresas, usuarios y dispositivos.

Administrador: es el encargado de administrar los todos los movimientos que estén bajo su dominio, por ejemplo: registrar vehículos, rutas, geocercas, asignar despacho etc.

Operador: Es el encargado de operar el vehículo y de cumplir con el itinerario asignado.

Dispositivo: Es el encargado de recolectar los datos del vehículo, como los datos de posicionamiento y parámetros del estado general del motor.

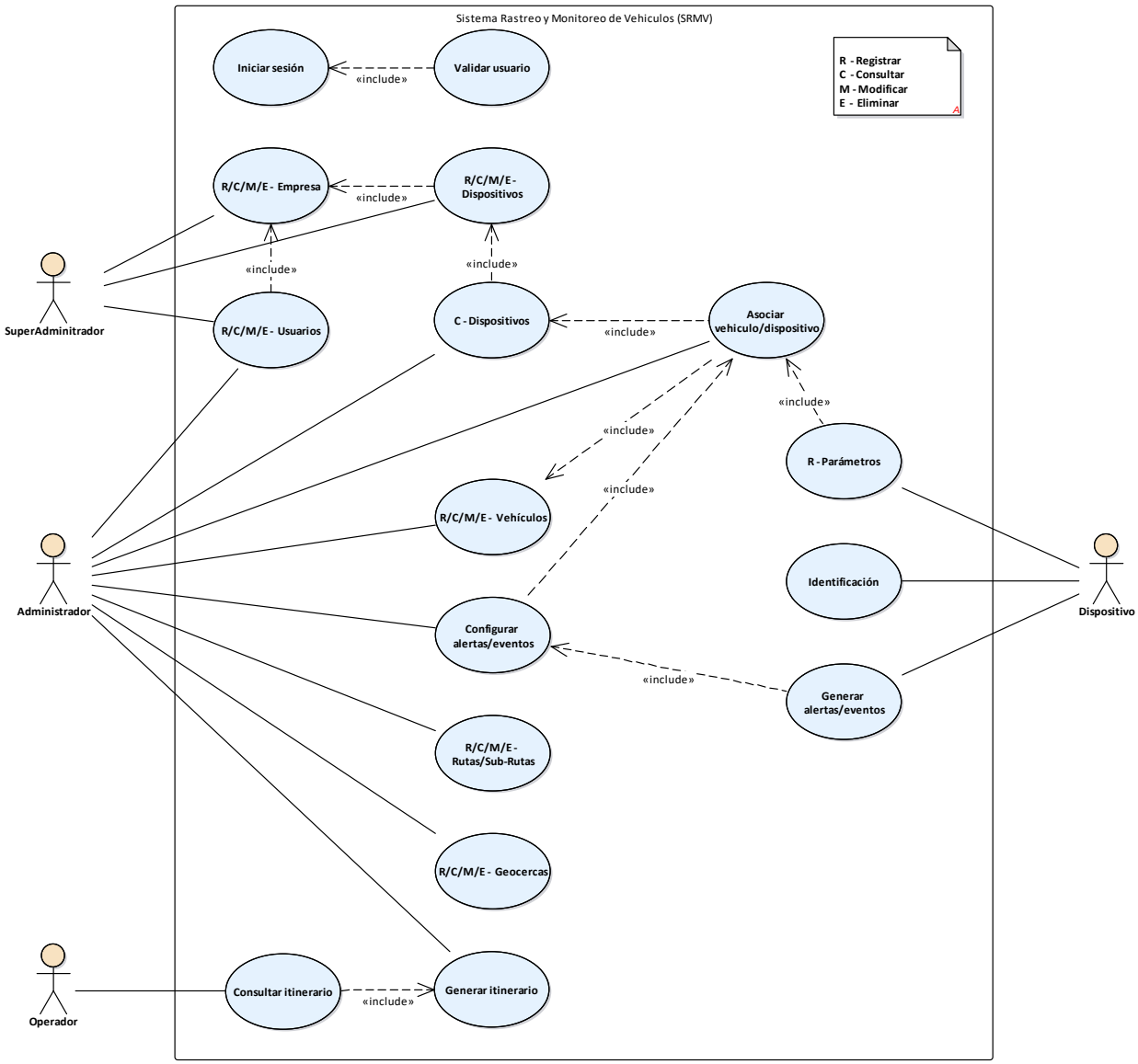

Figura 4.12 Modelo general de casos de uso (Fuente: Elaboración propia).

4.3 Modelado

En esta fase se definió la arquitectura básica del sistema a través del establecimiento de estructuras de datos representados en el modelo de clases y el modelo de datos. El modelo de interfaces muestra un diseño previo a las interfaces elegidas.

4.3.1 Modelo de clases

En la Figura 4.13 se muestra el modelo de clases relacionado entre objetos que componen la estructura del sistema. Las clases principales que constituyen el pilar del sistema de información son <vehiculos> ya que son el objeto a monitorear, <dispositivos> estos son asociados a los vehículos y recopilan datos durante el <seguimiento> en tiempo real de la unidad, las <alertas> y <eventos> se generan al ocurrir una incidencia ya establecida, de la misma manera o seguir una <ruta> o permanecer en un <geocerca>.

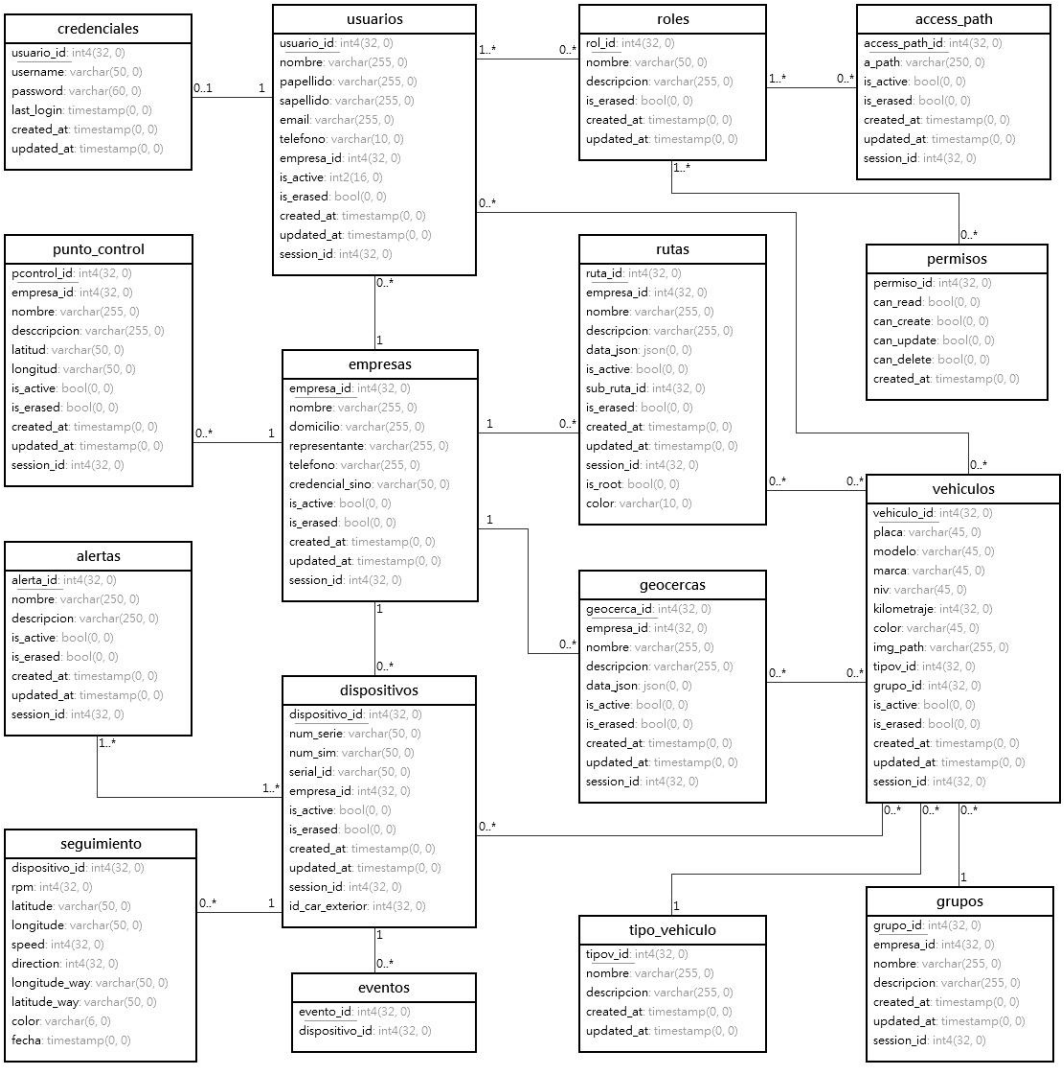

Figura 4.13 Modelo de Clases del Sistema (Fuente: Elaboración propia).

4.3.2 Modelo de datos

En la Figura 4.14 se observa el modelo de datos del sistema en el que se pueden apreciar las entidades relacionadas. Las entidades están compuestas por atributos. De la figura mencionada se observan las principales tablas que conforman el sistema, puesto que son esenciales para cumplir con el objetivo, de este conjunto de entidades podemos mencionar <Vehículos>, <Usuarios>, <Dispositivos>, <Seguimiento>, <Alertas>,<Geocercas> y <Rutas>.

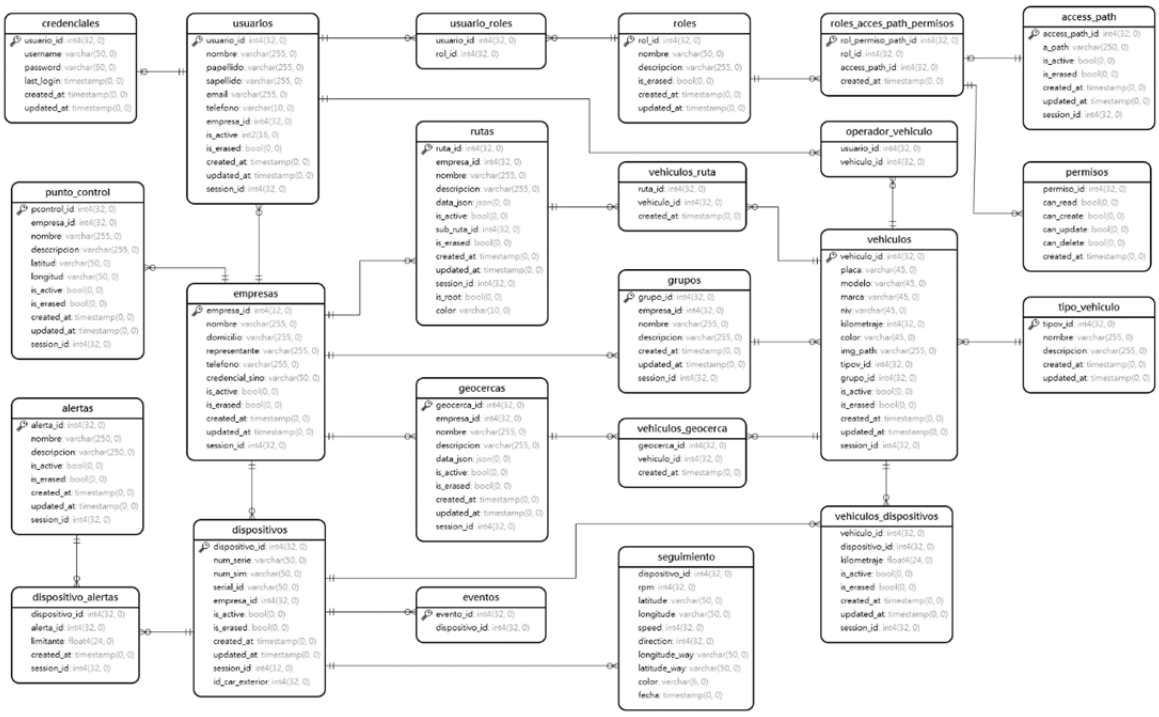

Figura 4.14 Modelo de Datos Entidad-Relación (Fuente: Elaboración propia).

4.3.3 Diagrama de Navegación

En la Figura 4.15 se puede observar el diagrama o modelo de navegación, muestra el orden jerárquico de páginas que se ejecutan al acceder a la aplicación, estas están relacionadas mediante enlaces.

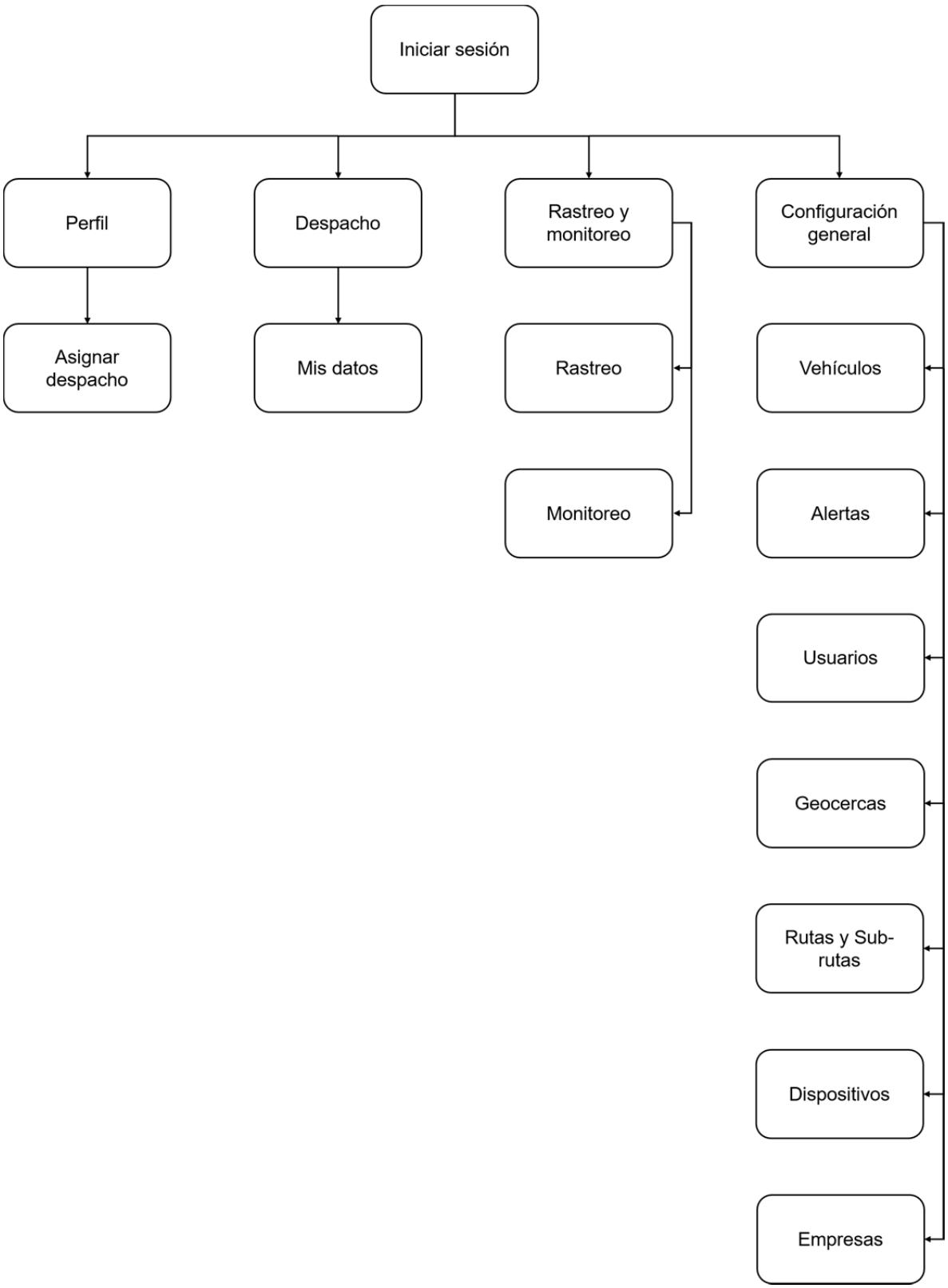

Figura 4.15 Modelo de navegación (Fuente: Elaboración propia).

4.3.4 Modelo de interfaces

Al ejecutar el sistema se muestra la vista de inicio de sesión el cual solicita un usuario y contraseña, como se muestra en la Figura 4.16 Interfaz inicio de sesión.

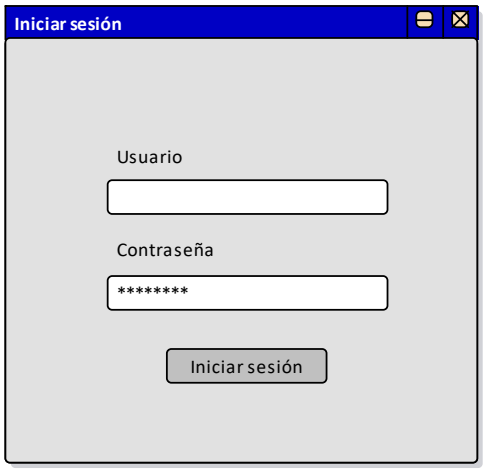

Figura 4.16 Interfaz inicio de sesión (Fuente: Elaboración propia).

Una vez que el usuario haya iniciado sesión, el sistema valida el tipo de usuario y solo muestra la información que está bajo su dominio. En la Figura 4.17 Interfaz para el registro de vehículos se muestra la interfaz para el registro de vehículo, en donde se solicitan algunos campos como el número de placa, el modelo, la marca, el NIV (Número de Identificación Vehicular), la foto del vehículo, entre otros datos; con el fin de identificar la unidad. En la parte lateral izquierda de la interfaz se observa el menú, el cual le corresponden los apartados vistos en el modelo de navegación Figura 4.15.

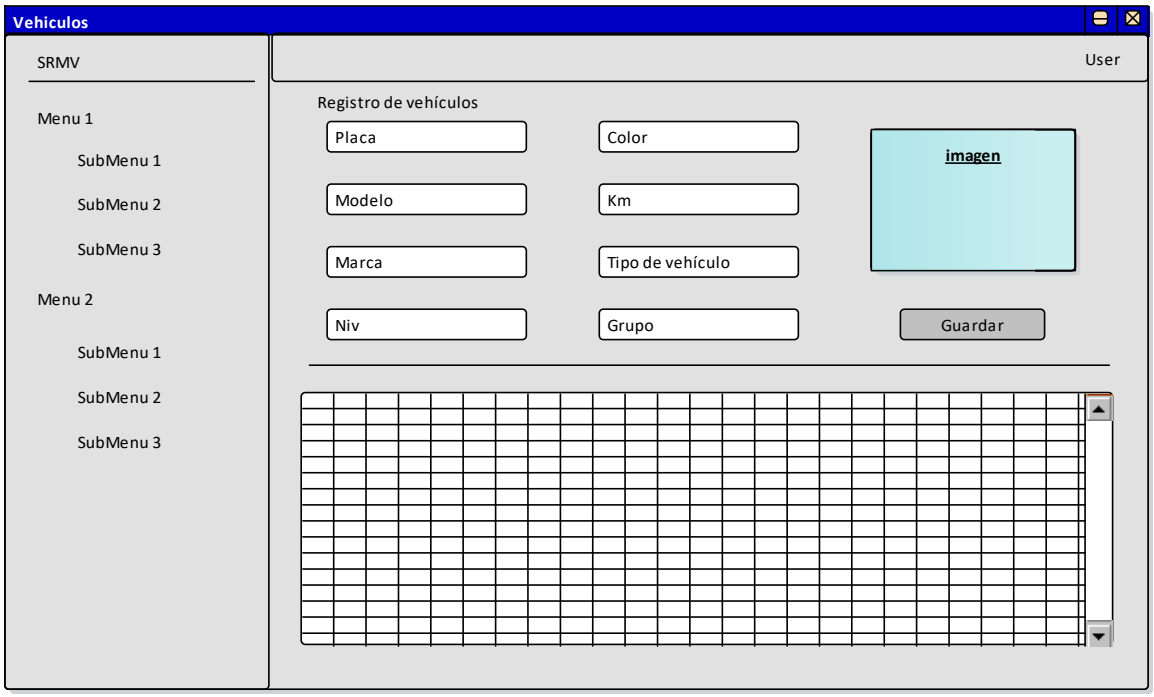

Figura 4.17 Interfaz para el registro de vehículos (Fuente: Elaboración propia).

En la Figura 4.18 Interfaz para el registro de Rutas y Sub-Rutas se observa la interfaz para registrar las rutas y sub-rutas, se muestra los campos como el nombre, el color para identificarla en el mapa y la descripción. Si la ruta a registrar es una ruta alternativa a otra, se marca como sub-ruta, en el mapa se traza la ruta obteniendo las coordenadas y posteriormente se pulsa el botón guardar. En la parte inferior se enlistarán las rutas registradas.

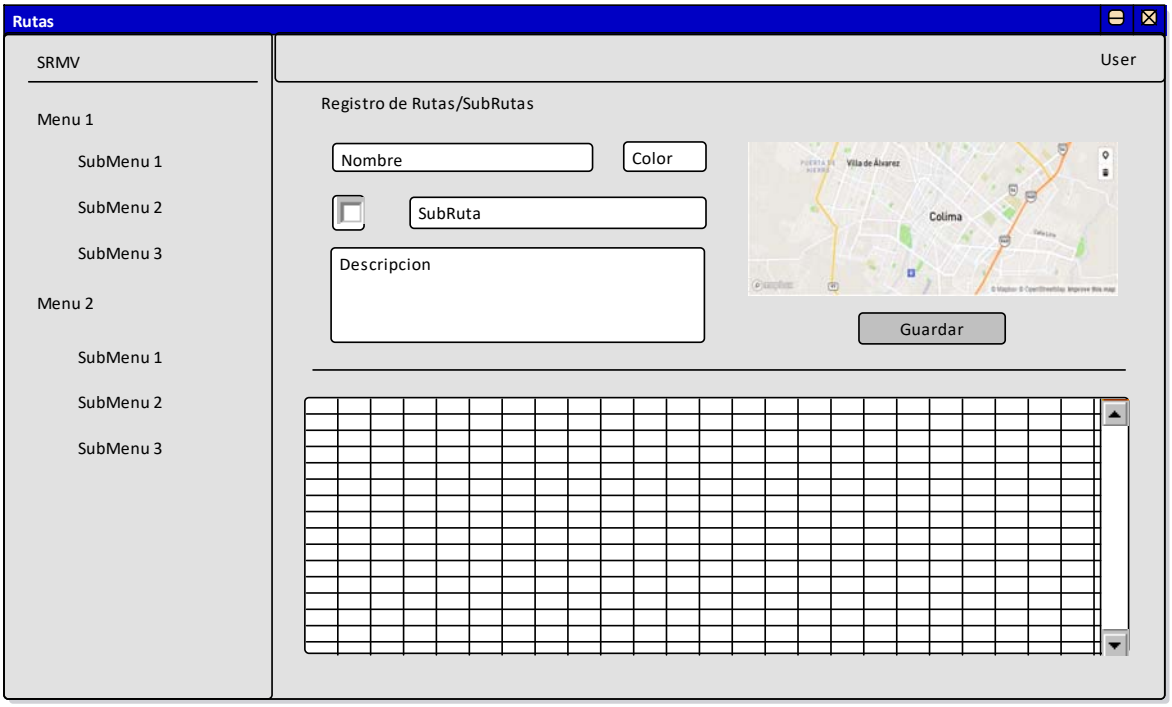

Figura 4.18 Interfaz para el registro de Rutas y Sub-Rutas (Fuente: Elaboración propia).

La Figura 4.19 muestra el formulario para el registro de la geocerca, el cual solicita como campos un nombre, color, observación, el trazo de la geocerca mediante punteros el mapa y el botón guardar, en la parte inferior se observa la tabla con la lista de la geocercas registradas.

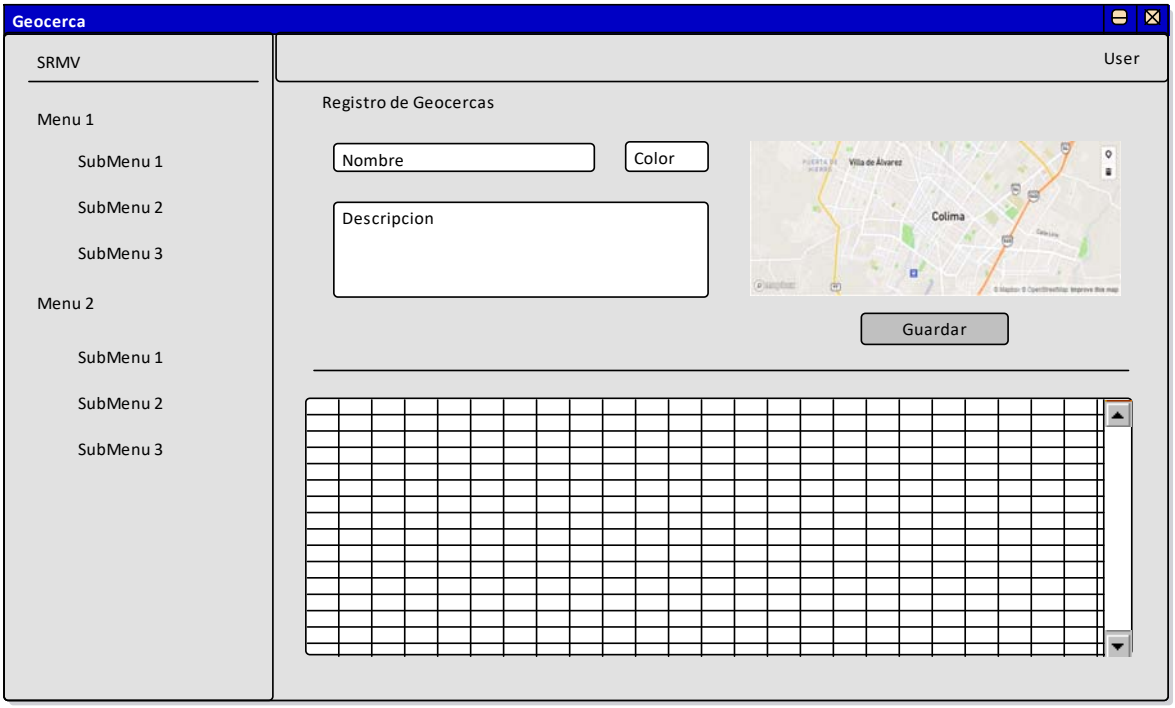

Figura 4.19 Interfaz para el registro de Geocercas (Fuente: Elaboración propia).

La Figura 4.20 muestra la configuración de las alertas y eventos que el usuario administrador requiera conocer en tiempo real, se muestra un input de tipo select en donde despliega la lista de vehículos, al seleccionar la unidad se puede modificar el checkbox de cada alerta y/o evento.

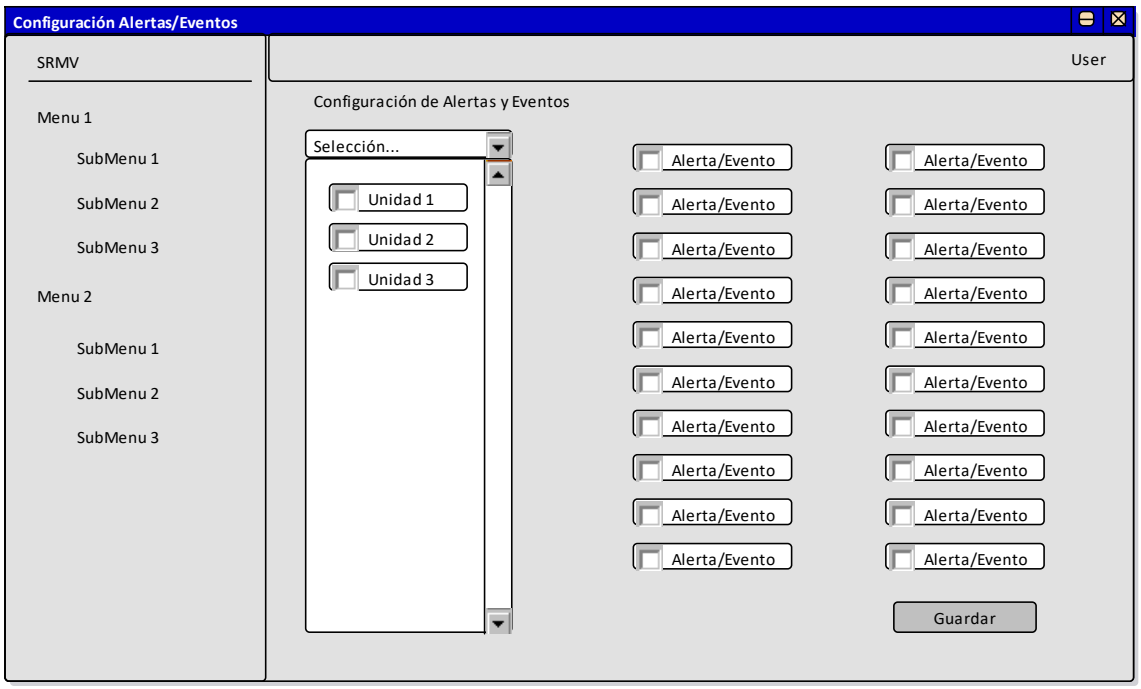

Figura 4.20 Interfaz para la configuración de alertas y eventos (Fuente: Elaboración propia).

En la Figura 4.21 se muestra la vista para generar el despacho de la unidad, donde se le asigna el conductor operador, los recorridos o rutas que tiene que realizar con fecha y hora, además, si la unidad pertenece a una geocerca de igual manera se le asigna. En la parte inferior se muestra la tabla con las rutas registradas.

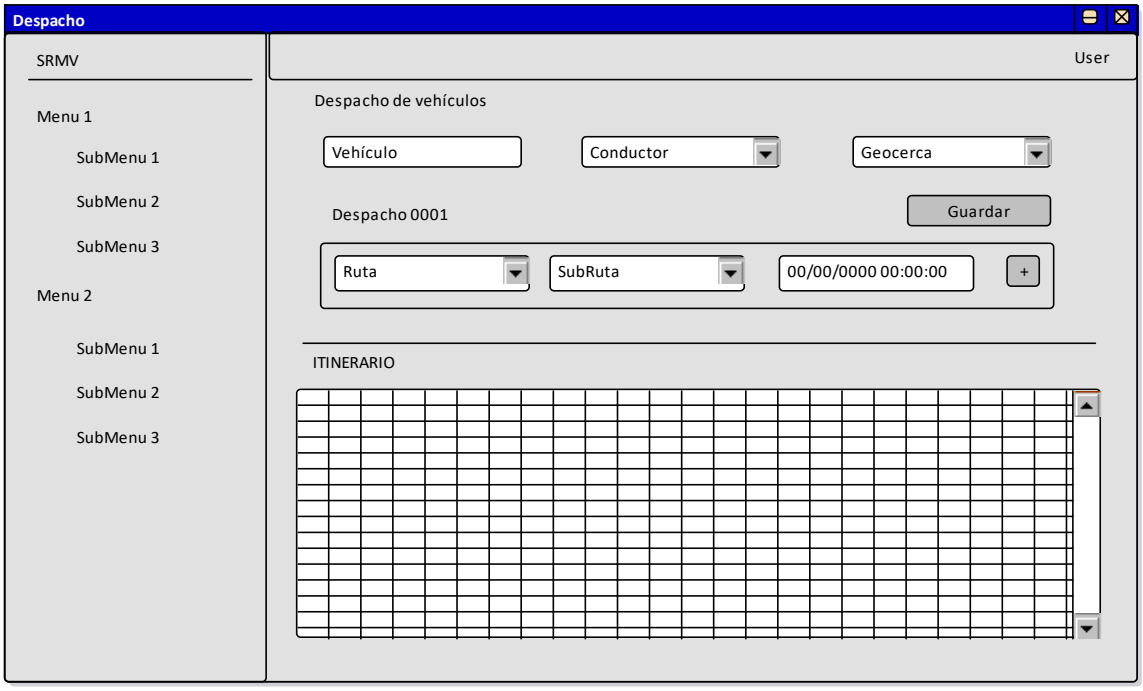

Figura 4.21 Interfaz de Despacho (Fuente: Elaboración propia).

La Figura 4.22 muestra el seguimiento de la unidad en un determinado periodo de fechas, se despliega la lista de las unidades vehiculares y posteriormente se selecciona el rango de fechas y en el mapa se mostrará el recorrido que realizó el vehículo, en la parte inferior del mapa se muestra una tabla con el histórico de alertas y eventos.

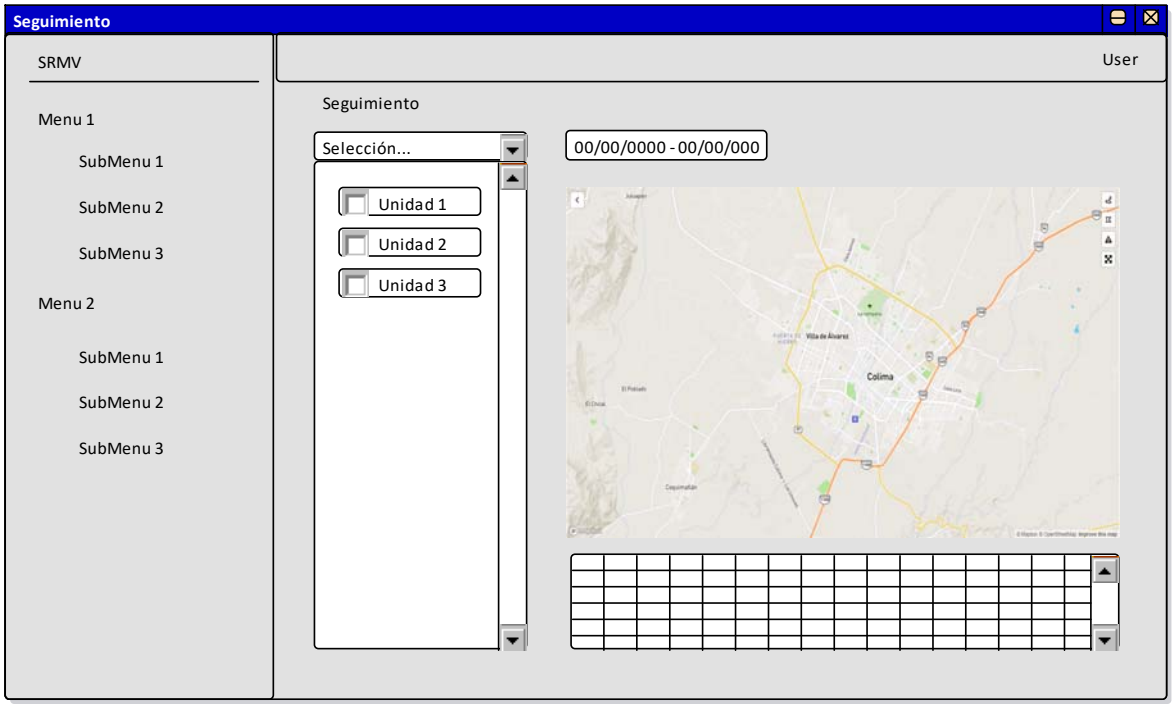

Figura 4.22 Interfaz para el rastreo del vehículo (Fuente: Elaboración propia).

En la Figura 4.23 muestra la información de las unidades que están operando, al seleccionar el vehículo de una lista desplegable es posible conocer la ubicación exacta, las alertas y eventos que se generan en tiempo real, además se muestra la ruta o la geocerca, si la unidad tiene asignada.

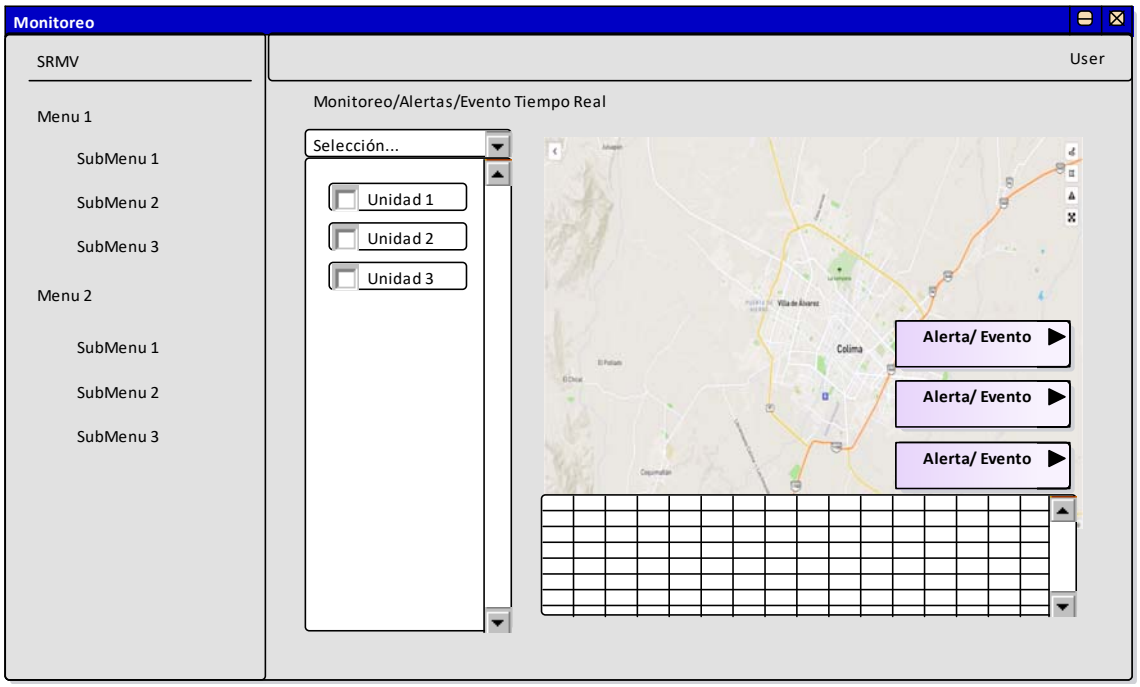

Figura 4.23 Interfaz para el monitoreo del vehículo (Fuente: Elaboración propia).

4.3.5 Modelo de despliegue

En la Figura 4.24, se observa el modelo de despliegue, el cual se compone de tres nodos físicos, el nodo que representa el equipo de cómputo o móvil en donde despliega a través del navegador el sistema, este se enlaza al dispositivo host mediante el protocolo HTTP. En el nodo Host se ejecuta el servidor Web Apache Tomcat y el Sistema Gestor de Base de Datos PostgreSQL enlazados mediante el protocolo TCP/IP. El tercer nodo representa el dispositivo conectado en la unidad vehicular, el cual transmite los datos recolectados hacia el nodo Host mediante el protocolo TCP/IP. En el servidor Web se ejecuta la aplicación.

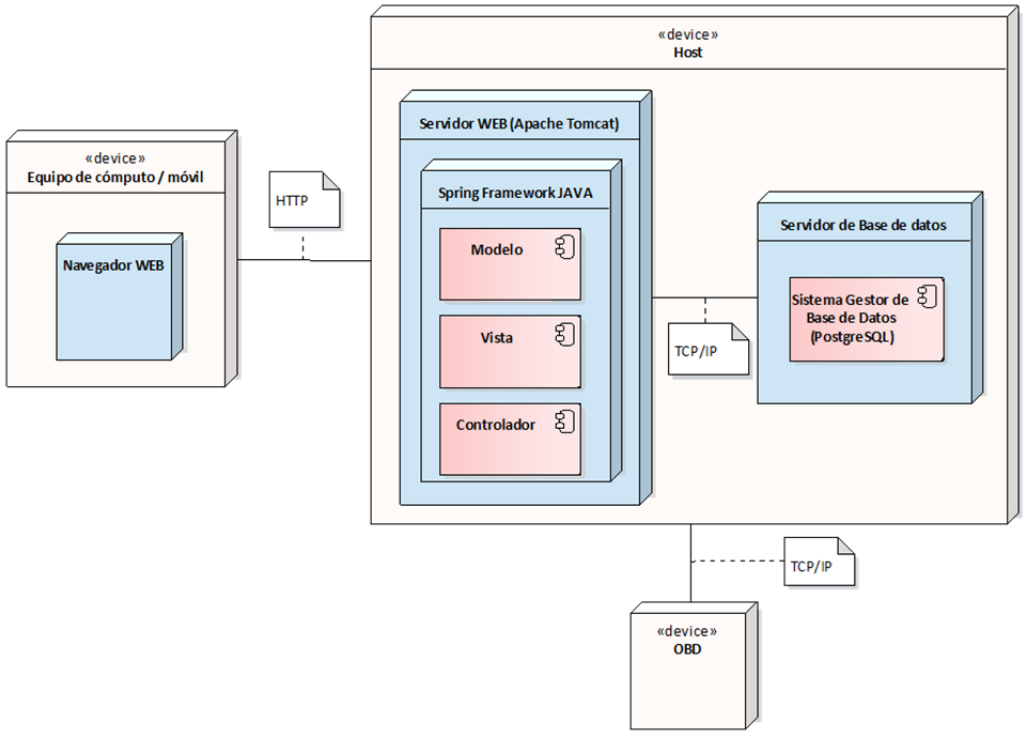

Figura 4.24 Modelo de despliegue (Fuente: Elaboración propia).

4.4 Implementación

Los modelos desarrollados fueron traducidos a códigos ejecutables que se fueron integrando hasta formar los módulos requeridos. Spring MVC es el framework que se utilizó para el desarrollo del sistema y como IDE de desarrollo NetBeans 8.2. Para la implementación del front-end se utilizó la plantilla Gentelella en conjunto con Bootstrap, para la base de datos PostgreSQL 10.

En las siguientes figuras se pueden observar pequeñas fracciones de código fuente, en la Figura 4.25 se muestra la clase Vehículo en la que se declaran los atributos requeridos. En las Figuras 4.26 y 4.27 se observa una fracción de código fuente de la clase controlador vehículo el cual sirve para responder a las acciones solicitadas en la aplicación y la Figura 4.28 se muestra una fracción del modelo de la clase vehículos, el que se encarga de acceder a la información de la base de

datos y la Figura 4.29 muestra una pequeña fracción de código fuente de la base de datos.

```
1. package com.pojos;
2. import org.springframework.web.multipart.MultipartFile;
3.
4. /**
5.  * @author Ismael villavicencio
6.  */
7. public class Vehiculo {
8.     private int vehiculo_id;
9.     private String placa;
10.    private String modelo;
11.    private String marca;
12.    private String niv;
13.    private float kilometraje;
14.    private String color;
15.    private MultipartFile img;
16.    private String img_path;
17.    private boolean is_active;
18.    private int grupo_id;
19.    private String nombre_grupo;
20.    private int tipo_vehiculo;
21.    private String nombre_tipo_vehiculo;
22.    private int dispositivo_id;
23.    private String nombre_dispositivo_id;
24.    private int id_car_exterior;
25.
26.    /**
27.    * Getters y Setters
28.    */
29. }
```

Figura 4.25 Fracción de código fuente de la Clase Vehículo (Fuente: Elaboración propia).

```
1.  package com.controller;
2.  /**
3.   * @author Ismael Villavicencio
4.   */
5.  @Controller
6.  public class VehiculoController {
7.
8.     @Autowired VehiculoDao vDao;
9.     @Autowired GrupoDao gDao;
10. @Autowired ObdDao oDao;
11.
12. @RequestMapping(value = {"/vehiculos"},  method = RequestMethod.GET)
13.  public String index(Model m){
14.     Vehiculo newV = new Vehiculo();
15.     m.addAttribute("grupos", gDao.GetALL());
16.     m.addAttribute("dispositivos",
17.     m.addAttribute("_title", "SRMV | Vehiculos");
18.     m.addAttribute("TipoV", vDao.TipoDeVehiculos());
19.     m.addAttribute("vehiculo", newV);
20.     return "vehiculos/vehiculos";
21.  }
22.
23.  @RequestMapping(value = "/allVehiculos", method = RequestMethod.GET)
24.  public @ResponseBody
25.     JsonResponse getAll(HttpServletRequest request) {
26.     JsonResponse response = new JsonResponse();
27.     response.setStatus("success");
28.     response.setResult(vDao.GetALLById(c.getEmpresa_id()));
29.     return response;
30.  }
31.
32.  @RequestMapping(value = "/editVehiculo",method = RequestMethod.GET,
    produces = {"application/json", "application/xml"})
33.  public @ResponseBody
34.     JsonResponse
    VehiculoById(@RequestParam("vehiculo_id") String vehiculo_id) {
35.     JsonResponse response = new JsonResponse();
36.     if (vehiculo_id.isEmpty()) {
37.        response.setStatus("error");
38.        response.setResult("El parámetro no debe estar vacío");
39.     } else {
40.        response.setStatus("success");
41.        response.setResult(vDao.FindById(vehiculo_id));
42.     }
43.     return response;
44.  }
45.
46. }
```

Figura 4.26 Fracción de código fuente del controlador Vehículo I (Fuente: Elaboración propia).

```java
package com.dao;
/**
 * @author Ismael Villavicencio
 */
public class VehiculoDao implements GenericDAO<Vehiculo> {

    private final JdbcTemplate jdbcTemplate;
    public VehiculoDao(DataSource dataSource) throws SQLException {
        CustomSQLErrorCodesTranslator
tr = new CustomSQLErrorCodesTranslator();
        jdbcTemplate = new JdbcTemplate(dataSource);
        tr.setDataSource(dataSource);
        this.jdbcTemplate.setExceptionTranslator(tr);
    }

    @Override
    public Vehiculo FindById(int id) {
        String sql = "Select * from vehiculos where vehiculo_id = " + id + "
and is_erased = false";
        return jdbcTemplate.query(sql, (ResultSet rs) -> {
            if (rs.next()) {
                Vehiculo v = new Vehiculo();
                v.setVehiculo_id(rs.getInt(1));
                v.setMarca(rs.getString(2));
                v.setModelo(rs.getString(3));
                v.setPlaca(rs.getString(4));
                v.setNiv(rs.getString(5));
                v.setKilometraje(rs.getInt(6));
                v.setColor(rs.getString(7));
                v.setImg_path(rs.getString(8));
                v.setTipo_vehiculo(rs.getInt(9));
                v.setGrupo_id(rs.getInt(10));
                v.setIs_active(rs.getBoolean(12));
                return v;
            }
            return null;
        });
    }

    @Override
    public List<Vehiculo> GetALL() {
        String SQL = "SELECT * FROM vehiculos WHERE is_erased = false ORDER
BY vehiculo_id DESC";
        List<Vehiculo> listObds = jdbcTemplate.query(SQL, (ResultSet rs, int
 i) -> {
            Vehiculo v = new Vehiculo();
            v.setVehiculo_id(rs.getInt(1));
            v.setPlaca(rs.getString(2));
            v.setModelo(rs.getString(3));
            v.setMarca(rs.getString(4));
            v.setNiv(rs.getString(5));
            v.setKilometraje(rs.getFloat(6));
            v.setColor(rs.getString(7));
            v.setImg_path(rs.getString(8));
            v.setTipo_vehiculo(rs.getInt(9));
            v.setIs_active(rs.getBoolean(12));
            return v;
        });
        return listObds;
    }
```

Figura 4.27 Fracción de código fuente del controlador Vehículo II (Fuente: Elaboración propia).

```
57.
58.     @Override
59.     public int SaveOrUpdate(Vehiculo v, int session_id) {
60.         int vehiculo_id = 0;
61.
62.         if (v.getVehiculo_id() > 0) {
63.             String sql = "UPDATE vehiculos SET placa=?, modelo=?, marca=?,
    niv=?, kilometraje=?, color=?, img_path=?, tipov_id=?,grupo_id=?
    session_id=? WHERE vehiculo_id=?";
64.             jdbcTemplate.update(sql, new Object[]{
65.                 v.getPlaca(),
66.                 v.getModelo(),
67.                 v.getMarca(),
68.                 v.getNiv(),
69.                 v.getKilometraje(),
70.                 v.getColor(),
71.                 v.getImg_path(),
72.                 v.getTipo_vehiculo(),
73.                 v.getGrupo_id(),
74.                 session_id,
75.                 v.getVehiculo_id()
76.             });
77.         } else {
78.             String sql = "INSERT INTO vehiculos (placa, modelo, marca, niv,
    kilometraje, color, img_path, tipov_id ,grupo_id ,session_id)"
79.                     + " VALUES (?,?,?,?,?,?,?,?,?,?)";
80.             jdbcTemplate.update(sql, new Object[]{
81.                 v.getPlaca(),
82.                 v.getModelo(),
83.                 v.getMarca(),
84.                 v.getNiv(),
85.                 v.getKilometraje(),
86.                 v.getColor(),
87.                 v.getImg_path(),
88.                 v.getTipo_vehiculo(),
89.                 v.getGrupo_id(),
90.                 session_id
91.             });
92.         }
93.         return vehiculo_id;
94.     }
95.
96. }
```

Figura 4.28 Fracción de código fuente del objeto de acceso a datos vehículo (Fuente: Elaboración propia).

```
1.    CREATE TABLE vehiculos (
2.      vehiculo_id serial PRIMARY KEY,
3.      placa varchar(45) UNIQUE NOT NULL,
4.      modelo varchar(45) NOT NULL,
5.      marca varchar(45) NOT NULL,
6.      niv varchar(45) UNIQUE NOT NULL,
7.      kilometraje int4 NOT NULL,
8.      color varchar(45) NOT NULL,
9.      img_path varchar(255) DEFAULT NULL,}
10.     tipov_id int4 NOT NULL,
11.     grupo_id int4 NOT NULL,
12.     is_active BOOLEAN NOT NULL DEFAULT TRUE,
13.     is_erased BOOLEAN NOT NULL DEFAULT FALSE,
14.     created_at timestamp(0) WITHOUT TIME ZONE DEFAULT NOT NULL,
15.     updated_at timestamp(0) DEFAULT NULL
16.   );
17.
18.   CREATE TABLE dispositivos (
19.     dispositivo_id serial PRIMARY KEY,
20.     num_serie varchar(50) UNIQUE NOT NULL,
21.     num_sim VARCHAR (50) UNIQUE NOT NULL,
22.     serial_id varchar(50) UNIQUE NOT NULL,
23.     empresa_id int4 NOT NULL,
24.     is_active BOOLEAN NOT NULL DEFAULT TRUE,
25.     is_erased BOOLEAN NOT NULL DEFAULT FALSE,
26.     created_at timestamp(0) WITHOUT TIME ZONE DEFAULT NOT NULL,
27.     updated_at timestamp(0) DEFAULT NULL
28.   );
29.
30.   CREATE TABLE alertas (
31.     alerta_id serial PRIMARY KEY,
32.     nombre varchar(250) NOT NULL,
33.     descripcion varchar(250) NOT NULL,
34.     is_active BOOLEAN NOT NULL DEFAULT TRUE,
35.     is_erased BOOLEAN NOT NULL DEFAULT FALSE,
36.     created_at timestamp(0) WITHOUT TIME ZONE DEFAULT NOT NULL,
37.     updated_at timestamp(0) DEFAULT NULL
38.   );
```

Figura 4.29 Fracción de código fuente de la base de datos (Fuente: Elaboración propia).

4.5 Reutilización de código.

Habitualmente en Java se utiliza la reutilización de código, ya que como en muchos lenguajes ayudan a reducir tiempo y redundancia en las operaciones o en la construcción de otros programas. En este proyecto de tesis se creó una interfaz **GenericDAO** donde cada clase implementa sus métodos lo que facilita las

operaciones de inserción, modificación, selección y eliminación de objetos en la base de datos, En la Figura 4.30 se observa la fracción de reutilización de código fuente de la interfaz. De la misma manera se reutiliza diversas secciones de código JSP perteneciendo al encabezado y pie en cada una de las páginas.

```
1. package com.interfaces;
2. /**
3.  * @author Ismael Villavicencio
4.  * @param <T>
5.  **/
6.    public interface GenericDAO<T> {
7.       //Busca el objeto por el id
8.       T FindById(int id);
9.       //Obtiene una lista de objetos
10.      List<T> GetALL();
11.      //Obtiene una lista de objetos por id
12.      List<T> GetALLById(int id);
13.      //Guarda el objeto
14.      int SaveOrUpdate(T t, int session_id);
15.      //Cambia el estado del objeto
16.      int ChangeStatus(T t, int session_id);
17.      //Elimina el objeto lógicamente
18.      int LogicalErasing(T t, int session_id);
19.   }
```

Figura 4.30 Fracción de reutilización de código fuente (Fuente: Elaboración propia).

4.6 Puesta en operación del sistema

En el desarrollo del proyecto se estuvo trabajando localmente con el IDE mencionado. Una vez terminado el módulo, este se fue integrando e implantado en la plataforma de Heroku con sus respectivas configuraciones para una correcta ejecución. El resultado se desplegó durante una semana para efectos de pruebas de funcionalidad y usabilidad por parte del usuario final.

4.7 Verificación y validación

Durante la transición del desarrollo del proyecto fueron realizadas diferentes pruebas para verificar el correcto funcionamiento y validar los requerimientos de

acuerdo a lo establecido. Las etapas del proceso de pruebas se implementaron de la siguiente manera:

Pruebas de desarrollo: estas pruebas consistieron en examinar detalladamente cada componente individualmente; es decir probar que cada función o método cumpla con lo solicitado.

Pruebas de sistema: una vez pasada la prueba individual de componentes, estos se integran para formar un solo sistema y así validar el requerimiento funcional o no funcional.

Pruebas de aceptación: estas son las últimas pruebas del proceso y consistieron en que el cliente suministre datos al sistema con el fin de determinar si cumple con las necesidades y requerimientos.

CAPÍTULO V RESULTADOS

5.1 Resultados

Como resultado de este proyecto de tesis se obtuvo un sistema de información en ambiente Web que permite gestionar, rastrear y monitorizar en tiempo real las unidades vehiculares.

Durante un lapso de cuatro semanas los vehículos se desplazaron por diferentes partes de la zona conurbada Colima - Villa de Álvarez, logrando obtener información de los desplazamientos, comportamiento de conducción, parámetros del motor y alertas, esto a través de un dispositivo OBD conectado al puerto del vehículo, como se puede observar en la Figura 5.1.

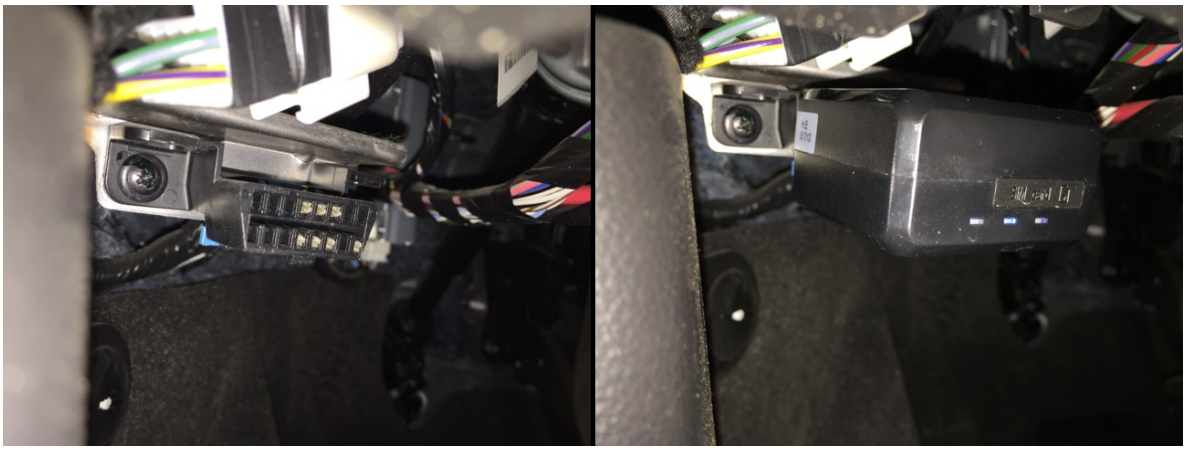

Figura 5.1 Puerto OBD y dispositivo conectado (Fuente: Elaboración propia).

Una vez montada la aplicación en el servidor, la aplicación se ejecuta y carga por defecto la vista de inicio de sesión, solicita al usuario que ingrese las credenciales para acceder al sistema, como se muestra en la Figura 5.2.

Figura 5.2 Interfaz inicio de sesión (Fuente: Elaboración propia).

El sistema requiere de ciertos datos para operar, por lo que es necesario el registro previo de vehículos, dispositivos OBD, rutas, geocercas, entre otros más. En la Figura 5.3 se observa la interfaz para el registro de unidades vehiculares, en la parte superior de la vista se muestra el formulario junto con una imagen y en la parte inferior, la lista de los vehículos registrados, a cada una de estas unidades se les asigna un dispositivo.

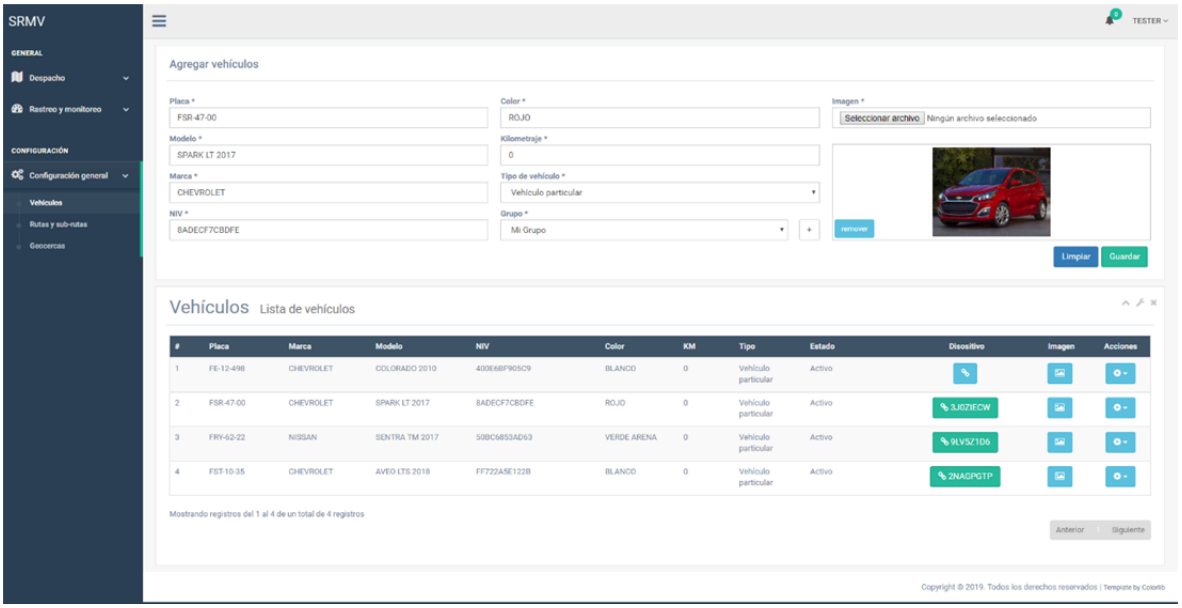

Figura 5.3 Interfaz para el registro de vehículos (Fuente: Elaboración propia).

En la Figura 5.4 se observa la vista para el registro de geocercas, mediante una interfaz de mapas el sistema permite el trazo de puntos georreferenciados que forman un polígono, la interfaz muestra en la parte superior izquierda el formulario y en la parte derecha la geocerca ya trazada, en la parte inferior se observa una lista de las geocercas registradas.

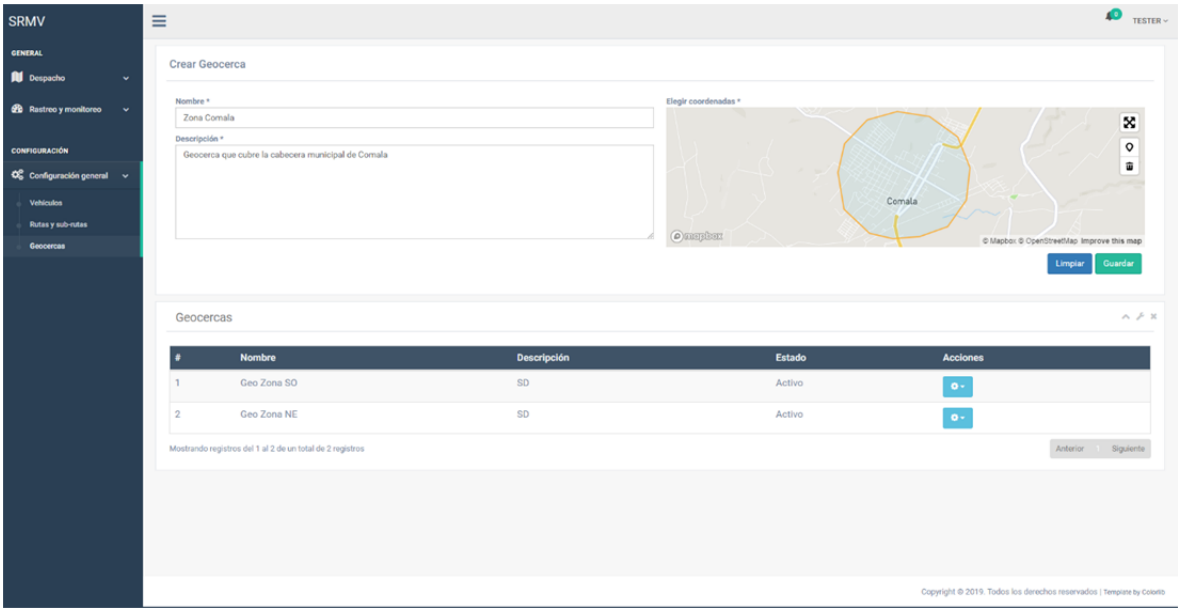

Figura 5.4 Interfaz para el registro de geocercas (Fuente: Elaboración propia).

En la Figura 5.5 se observa el formulario para el registro y mediante la interfaz de mapas, se trazan dos puntos georreferénciales y automáticamente se traza la ruta.

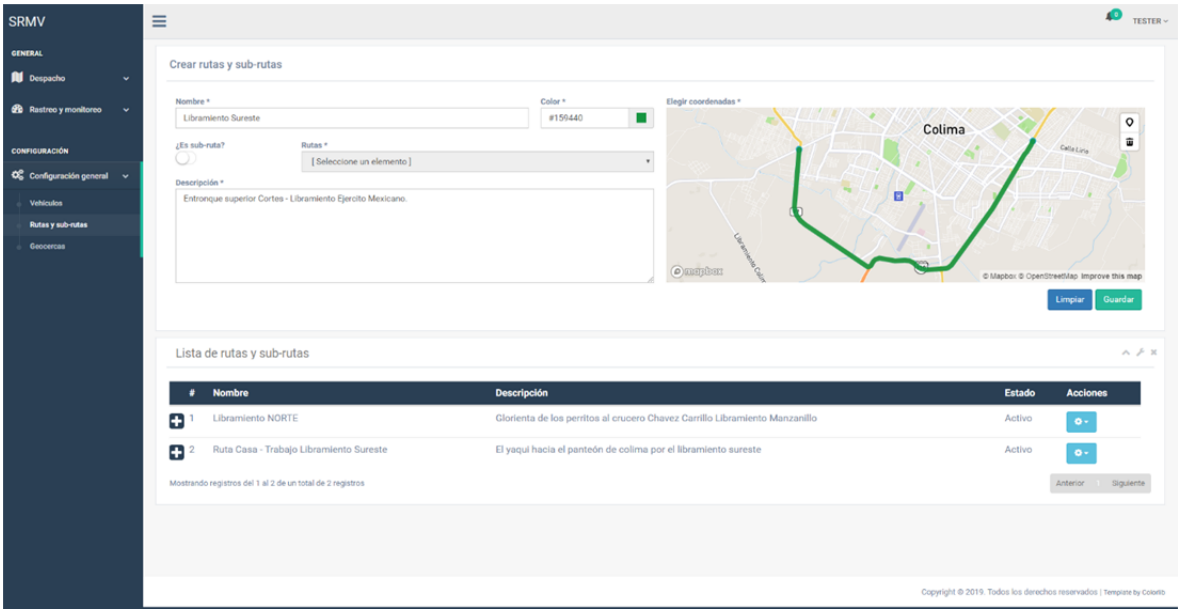

Figura 5.5 Interfaz para el registro de rutas / sub-rutas (Fuente: Elaboración propia).

El sistema permite configurar las alertas y eventos por cada unidad, al seleccionar el vehículo se habilitan las opciones para que el usuario elija que alertas y eventos desee que el sistema le notifique, como se observa en la Figura 5.6.

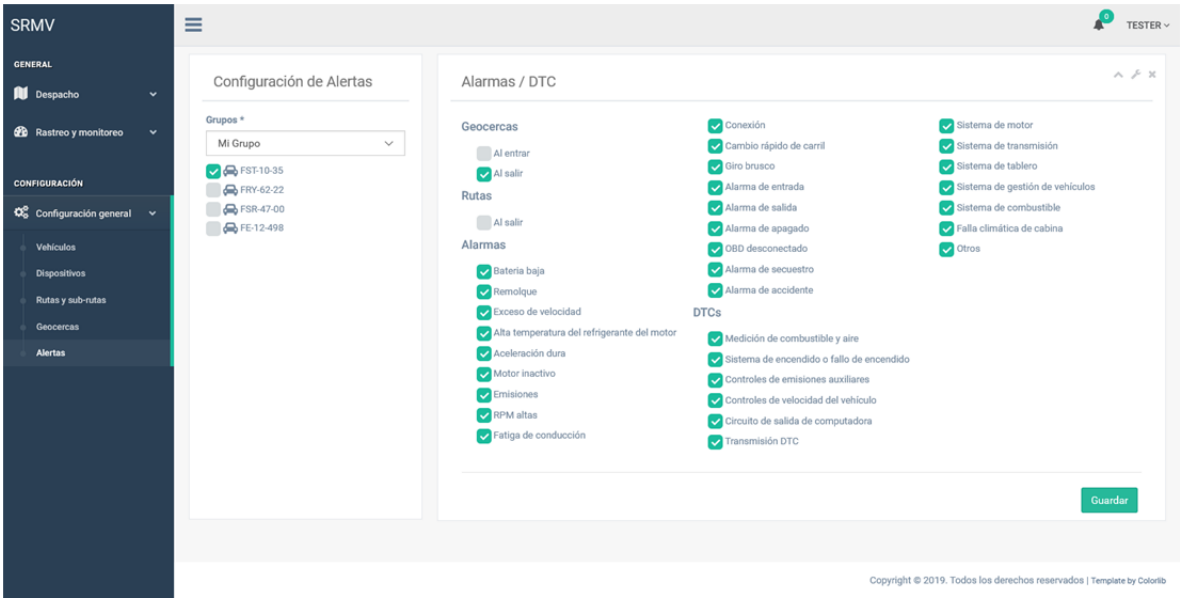

Figura 5.6 Configuración de alertas y eventos (Fuente: Elaboración propia).

En la Figura 5.7 se observan las cuatro unidades que están operando dentro de la zona conurbada Colima- Villa de Álvarez, al seleccionar el vehículo se pueden apreciar las características de este; como la posición geográfica, la velocidad en la que se encuentra, las revoluciones por minuto, y aquellos eventos que pudieran ir surgiendo, como el frenado brusco, la aceleración dura, el exceso de velocidad, entro otros, todo en tiempo real.

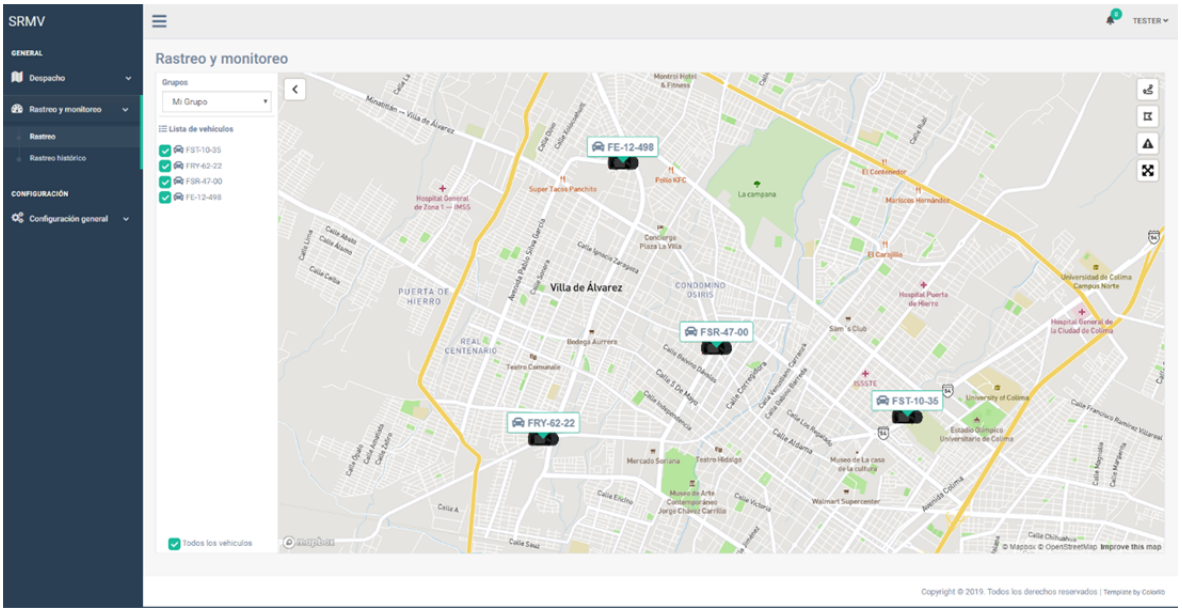

Figura 5.7 Rastreo y monitoreo de unidades vehiculares (Fuente: Elaboración propia).

Las alertas y eventos se muestran en tiempo real por medio de una notificación, en la Figura 5.8, se puede observar la alerta generada al momento en que la unidad vehicular sale de la geocerca notificando al usuario dicho acontecimiento. Una vez mostrada la alerta, esta pasa a un registro histórico para posteriormente ser consultado.

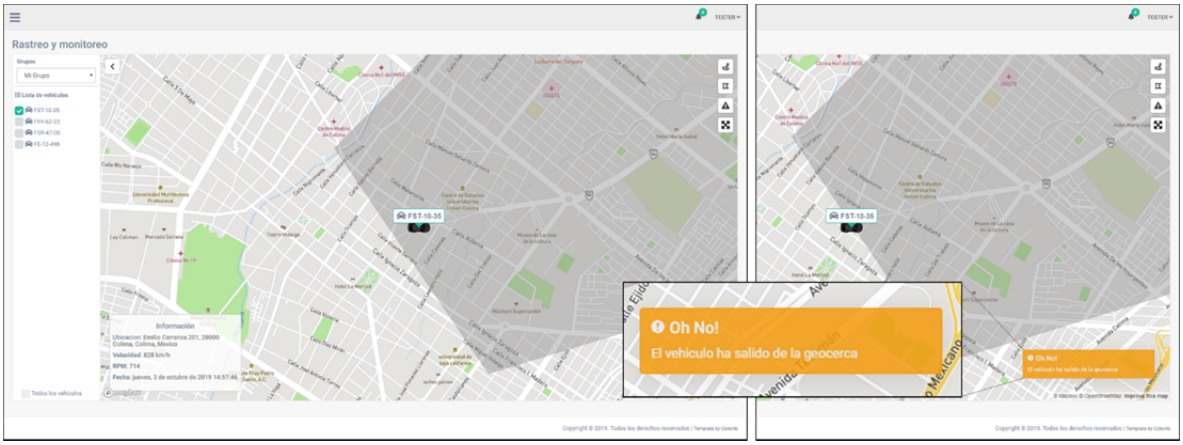

Figura 5.8 Alerta de salida de la geocerca (Fuente: Elaboración propia).

En la Figura 5.9, se observa el histórico de recorridos que ha realizado el vehículo en un determinado periodo de tiempo, al seleccionar la unidad, el sistema solicita al usuario que ingrese el intervalo de fechas, posteriormente traza el recorrido en el mapa y muestra la parte inferior el registro histórico de eventos.

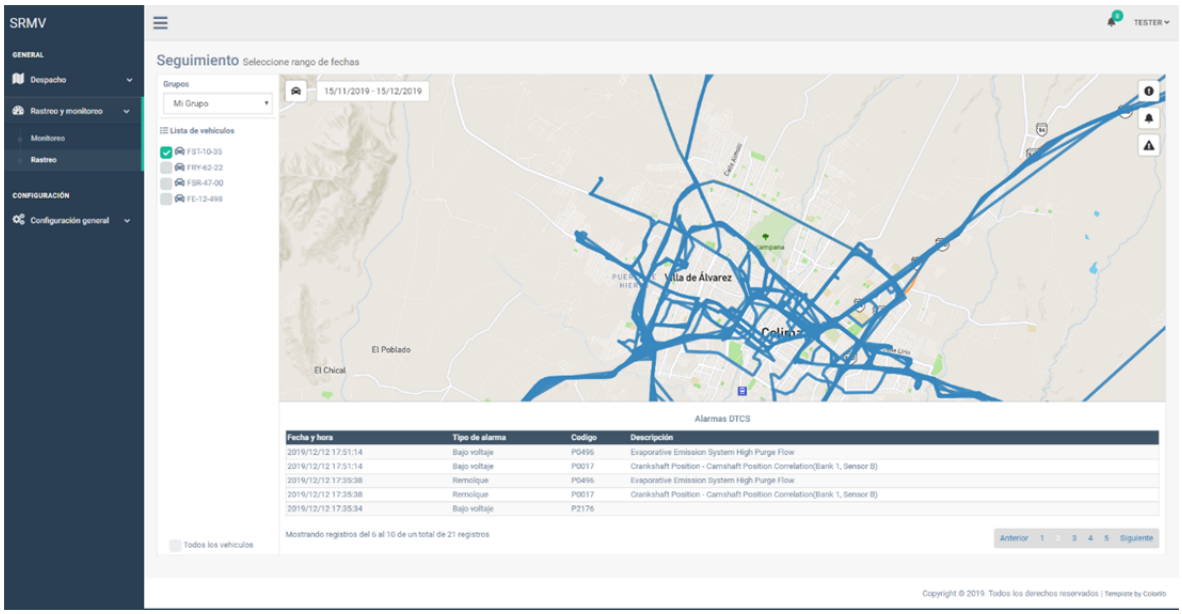

Figura 5.9 Recorrido histórico de la unidad vehicular (Fuente: Elaboración propia).

El registro histórico muestra el historial de eventos que sucedieron durante la operación de la unidad vehicular, se muestra mediante una tabla que incluye la fecha y hora en la que sucedió, el tipo de alarma o evento, si es una falla mecánica se muestra el código de la falla y la descripción respectivamente, como se puede observar en la Figura 5.10.

Figura 5.10 Histórico de alertas y eventos (Fuente: Elaboración propia).

5.2 Discusión

Este trabajo se distingue de otras soluciones que se enfocan en la misma problemática. En contraste con los sistemas propuestos estudiados en el Capítulo I, en la Revisión de la literatura, tal el caso de (Aljaafreh, et al,.2011), (Montero Revelo, 2016) y (L. ben Othmane *et al*., 2018) utilizan sistemas embebidos, la solución presentada utiliza un dispositivo estándar industrializado, específicamente para este tipo de funciones por lo que su factibilidad y confiabilidad es mayor.

En trabajos como el de (Aljaafreh, et al,.2011), el seguimiento en tiempo real no está presente, por lo tanto, al administrador de la flota vehicular no le permite enterarse de percances o mal uso de las unidades.

Soluciones como la de (Fuad & Drieberg, 2013) y (Conza Berrocal, 2013), el seguimiento en tiempo real se da a través de las coordenadas que obtiene un dispositivo móvil, si por alguna razón el dispositivo móvil fuera retirado del vehículo, se podría prestarse a malas interpretaciones, además no incorporan el monitoreo de parámetros del motor.

El enfoque de este desarrollo destaca por generar alertas al ocurrir un determinado evento, por ejemplo, al ocurrir el frenado brusco, entrada/salida de una geocerca, si la computadora central registra alguna falla mecánica del vehículo, entre otras. Asimismo, destaca por la incorporación de geocercas lo que permite mayor control sobre las unidades y conductores.

CAPÍTULO VI CONCLUSIONES Y TRABAJOS A FUTURO

6.1 Conclusiones

El sistema implementado en este trabajo de investigación, permite optimizar la administración de flotas vehiculares. Con el uso de este sistema cualquier organización se ve beneficiada porque resuelve problemas comunes, tales como: conocer en tiempo real la posición de la unidad, la forma de manejo del conductor, los problemas mecánicos que tiene la unidad al encenderse el Check Engine; además, permite conocer el estado general del vehículo, como el nivel de combustible, la velocidad de desplazamiento, las revoluciones por minuto e informa oportunamente de cualquier percance o suceso mediante alertas y eventos en tiempo real. El bajo costo del sistema lo hace accesible a pequeñas organizaciones que normalmente cuentan con presupuestos reducidos. Todo esto nos permitió reflexionar acerca del impacto favorable que tiene al implementar el sistema.

El uso de tecnologías libres en el desarrollo de este proyecto facilitó la construcción del software ya que existe numerosa documentación; además, al ser una framework basado en Java, este no se limita la posibilidad de poder ser ejecutado en cualquier sistema operativo.

6.2 Cumplimiento de los objetivos de la investigación

El objetivo “Implementar un sistema de información basado en tecnologías de Internet de las Cosas que permita monitorizar los parámetros funcionales, de operación y de desplazamiento de vehículos” planteado en este documento de tesis, se cumplió satisfactoriamente, puesto que el sistema es capaz de desplegar los parámetros funcionales de cada uno de los vehículos, proporciona ubicación y desplazamiento en tiempo real, además de las alertas y eventos.

6.3 Aceptación o rechazo de la hipótesis de trabajo

La hipótesis “El desarrollo de una plataforma IoT para el rastreo y monitoreo remoto de parámetros en vehículos hará posible disponer en tiempo real de la

información relacionada con los parámetros funcionales, de operación y de desplazamiento de vehículos contribuyendo a eficientar la administración de una flotilla vehicular" se comprobó con éxito, ya que el sistema logra mostrar en tiempo real la información de la unidad al estar en operación, además; optimiza la gestión de flotas vehiculares significativamente, favorece la toma de decisiones, el incremento de la productividad y la mejora continua en las prácticas de conducción y actividades de mantenimiento.

6.4 Limitaciones de la investigación

La plataforma IoT desarrollada en este documento de tesis, describe un sistema de información para el rastreo y monitoreo de parámetros en vehículos, el cual está compuesto por un sistema en ambiente Web y un módulo recolector de datos referente al dispositivo OBD. Dicho sistema consistió en la creación de un prototipo funcional que permite validar los objetivos propuestos en este documento de tesis y así comprobar la hipótesis.

6.5 Trabajos a futuro

Como etapas futuras de este proyecto se integrarán módulos de reportes y análisis de datos con el fin de obtener indicadores de efectividad, rendimiento y productividad. Se plantea el desarrollo de una aplicación móvil para facilitar la navegación y el cumplimiento del itinerario del operador del vehículo.

BIBLIOGRAFÍA

Aljaafreh, A., Khalel, M., Al-Fraheed, I., Almarahleh, K., Al-Shwaabkeh, R., Al-Etawi, S., & Shaqareen, W. (2011). Vehicular data acquisition system for fleet management automation. *Proceedings of 2011 IEEE International Conference on Vehicular Electronics and Safety*, 130-133. doi:10.1109/ICVES.2011.5983801

Alonso, F., Martínez Normand, L., & Segovia Pérez, F. J. (2005). *Introducción a la Ingeniería del Software* (Primera edición ed.). España: Delta.

Ambler, S. W. (2005). *The Agile Unified Process (AUP)*. Obtenido de Ambysoft: http://www.ambysoft.com/unifiedprocess/agileUP.html

Barrio Andrés, M. (2018). *Internet de las Cosas.* Madrid: Editorial Reus. Obtenido de https://books.google.com.mx/books?id=4zottQEACAAJ

Becvar, Z., Mach, P., & Pravda, I. (s.f.). *Redes móviles .* Obtenido de Improvet: http://techpedia.fel.cvut.cz/download/?fileId=199&objectId=48

Bootstrap. (2011). *Acerca*. Obtenido de getBootstrap: https://getbootstrap.com/docs/4.5/about/overview/

Camarillo Alfaro, D. (s.f.). *Diagnostico a Bordo (OBD).* Recuperado el 05 de 2019, de COMISIÓN NACIONAL PARA EL USO EFICIENTE DE LA ENERGÍA: https://www.gob.mx/cms/uploads/attachment/file/187221/diagnosticoabordo_1_260117.pdf

Conza Berrocal, M. H. (2013). *Desarrollo de una aplicación web orientada a servicios para el monitoreo de una flota de vehículos haciendo uso de la tecnología GPS.* Obtenido de Universidad Nacional de San Antonio Abad del Cusco. Facultad de Ciencias Químicas, Físicas y Matemáticas: http://repositorio.unsaac.edu.pe/handle/UNSAAC/947

Deitel, H. M., & Deitel, P. J. (2003). *Cómo programar en Java.* México: Pearson Educación. Obtenido de https://books.google.com.mx/books?id=is2J44U4DpsC&printsec=frontcover&hl=es&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false

Desai, M., & Phadke, A. (2017). Internet of Things based vehicle monitoring system. *2017 Fourteenth International Conference on Wireless and Optical Communications Networks (WOCN)*, 1-3. doi:10.1109/WOCN.2017.8065840

Fuad, M. R., & Drieberg, M. (2013). Remote vehicle tracking system using GSM Modem and Google map. *2013 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (CSUDET)*, 15-19. doi:10.1109/CSUDET.2013.6670977

Geotab Inc. (2019). *GEOTAB*. Obtenido de GEOTAB INC.: https://www.geotab.com/

Gisbert Vercher, B. (2015). *Administración y auditoría de los servicios web.* ditorial Elearning, S.L.

Hernández Sampierí , R., Fernández Collado, C., & Baptísta Lucio, P. (2006). *Metodología de ka investigación* (Cuarta ed.). México: McGraw-HillI.

IBM. (2020). *Conceptos clave: entidad, atributo y tipo de entidad*. Obtenido de IBM Knowledge Center: https://www.ibm.com/support/knowledgecenter/es

Indacar. (2019). *Indacar*. Obtenido de Indacar: https://indacar.io/

Jaramillo Valbuena, S., Augusto Cardona, S., & Villa Zapata, D. A. (2008). *Programación Avanzada en Java.* (E. S.A.S, Ed.) Armenia. Obtenido de https://books.google.com.mx/books?id=zug36aj0JWIC

Mapbox. (2019). *About Mapbox*. Obtenido de Mapbox: https://www.mapbox.com/about/company/

McClelland, C. (17 de Abril de 2017). *What is an IoT Platform?* Obtenido de iot for all: https://www.iotforall.com/what-is-an-iot-platform/

Medina Martínez, J. C., Alonso Guzmán, L., Hernández Alarcón, V. M., & Móndragon Gómez, C. (2013). Diagramas de navegación en aplicaciones web. *Vínculos, 10*(2), 119-135.

Montero Revelo, E. L. (16 de 03 de 2016). *Diseño e implementación de un sistema web para el monitoreo del estado de un motor.* Obtenido de BIBDIGITAL: http://bibdigital.epn.edu.ec/handle/15000/15057

National Coordination Office for Space-Based Positioning, Navigation, and Timing. (s.f.). *Sistema de Posicionamiento Global*. Obtenido de gps.gov: https://www.gps.gov/spanish.php

Object Management Group, Inc. . (Enero de 2020). *UML*. Obtenido de UML: https://www.uml.org/

Othmane, L. b., Alvarez, V., Berner, K., Fuhrmann, M., Fuhrmann, W., Guss, A., & Hartsock, T. (16 de 09 de 2018). Demo: A Low-Cost Fleet Monitoring System. *2018 IEEE International Smart Cities Conference (ISC2)*, 1-2. doi:10.1109/ISC2.2018.8656826

Pérez Villegas, O. (17 de Mayo de 2018). *Revista Transportes y turismo*. Obtenido de ¿Vehiculos conectados? Así empezó esta historia: https://www.tyt.com.mx/nota/vehiculos-conectados-asi-empezo-esta-historia

Pivotal Software, Inc. (2018). *Spring makes Java*. Obtenido de Spring: https://docs.spring.io/spring/docs/4.2.x/spring-framework-reference/html/mvc.html

PostgreSQL Global Development Group. (2019). *Acerca*. Obtenido de postgresql: https://www.postgresql.org/about/

Pressman, R. S. (2010). *Ingeniería del software un enfoque práctico* (Séptima ed.). México: McGraw-Hill.

Sommerville, I. (2011). *Ingenieria de Software.* México: Pearson.

Sparx Systems Pty Ltd. (2020). *El uso del modelo de caso de uso*. Recuperado el 05 de 2020, de Sparx Systems: https://www.sparxsystems.com/resources/tutorials/uml/use-case-model.html

Sparx Systems Pty Ltd. (2020). *Model de requerimientos*. Recuperado el 04 de 2020, de Sparx Systems: https://sparxsystems.com/enterprise_architect_user_guide/15.1/model_domains/requirements_engineering.html

Staff Editorial de Electrónica y Servicio. (2014). *Electrónica y Servicio: Máquinas de videojuegos tragamonedas.* (2. México Digital Comunicación S.A. de C.V., Ed.) Obtenido de https://books.google.com.mx/books?id=ItOoBQAAQBAJ

U.S. Department of Transportation. (2015). *Satellite Navigation - GPS - Presidential Policy*. Obtenido de Federal Aviation Administration: https://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navservices/gnss/gps/policy/presidential/

Valencia Ruiz, J. C. (07 de 1997). *Diseño y Construcción de un Sistema Localizador de Vehículos basado en GPS y Utilizando la red de Telefonía Celular.* Obtenido de BIBDIGITAL: https://bibdigital.epn.edu.ec/bitstream/15000/5097/1/T295.pdf

Vargas Cordero, Z. R. (2009). La investigación aplicada: Una forma de conocer las realidades con evidencia científica. *Educación*(33), 155-165. Obtenido de https://www.redalyc.org/articulo.oa?id=440/44015082010

Webfleet Solutions B.V. (2019). *Webfleet: Fleet management software*. Obtenido de Webfleet Solutions: https://www.webfleet.com/en_gb/webfleet/products/webfleet/