ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA WEB PARA EL MONITOREO DEL ESTADO DE UN MOTOR

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

ELIANA LORENA MONTERO REVELO

elianamonterolr@gmail.com

DIRECTOR: Ing. JULIO CÉSAR CAIZA ÑAMO, MSc.

julio.caiza@epn.edu.ec

Quito, Marzo 2016

DECLARACIÓN

Yo, Eliana Lorena Montero Revelo, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Eliana Lorena Montero Revelo

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Eliana Lorena Montero Revelo, bajo mi supervisión.

_____

Ing. Julio César Caiza, MSc.

DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

Al mi director del proyecto de titulación, MSc. Julio Cesar Caiza, por la guía durante el desarrollo de este proyecto, por su confianza y paciencia.

A mis padre, Luis y Lilia por el apoyo incondicional en los buenos y malos momentos de mi vida.

A mis hermanos, Evelyn, Paúl y Fernanda por los ánimos a seguir para la culminación de este proyecto.

A quienes conforman New Access por su compresión y permisos brindados para la culminación de este proyecto, especialmente a Erika Mera.

A mis amigos con quienes se formó una gran amistad, especialmente a David y Fernando por su ayuda y paciencia.

Ely

DEDICATORIA

A Dios por ser mi fuerza para todo momento, a mis padres a las personas que más quiero y admiro y a Fernando por sus palabras de aliento y compresión para culminar está etapa.

Ely

CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

PRESENTACIÓN

En la actualidad los automóviles cuentan con una computadora que se encarga de monitorear un conjunto de sensores que ayudan a verificar el correcto funcionamiento del automóvil. Para acceder a toda la información se utiliza el sistema OBD-ll, el cual procesa la información recibida desde el motor. El propósito de este proyecto es ofrecer un sistema web que permita visualizar la información de los sensores del sistema OBD-II de un automóvil, para detectar e informar cuando suceda una incidencia inesperada.

Los teléfonos inteligentes, tabletas y otros dispositivos móviles menos populares, han revolucionado el mundo del internet y se han vuelto indispensables para la vida diaria, por lo que se ha visto la necesidad de desarrollar un sistema web que se adapte de manera automática y flexible al tamaño de los diferentes dispositivos móviles.

El sistema web está compuesto por módulos de administración que abarcan los procesos de ingreso, modificación y eliminación de datos. Está información es guardada en la base de datos y posteriormente es visualizada por el usuario.

El sistema ofrece una interfaz para realizar el proceso de configuración remota que permita la selección de un conjunto de medidas que se desea obtener del motor del automóvil y posteriormente ser visualizadas por el usuario. Además el sistema notificará al usuario sobre el rendimiento del motor en caso de que el usuario lo desee, es decir, se establecen criterios de advertencia en función de las medidas límites de los sensores, este proceso se realiza en el módulo de administración de incidencias.

El sistema permite visualizar las medidas capturadas por el sistema OBD-ll a través de reportes históricos y gráficos estadísticos, esto permite al usuario verificar, en función de su conocimiento, el rendimiento del motor.

Finalmente el sistema ofrece al usuario una interfaz para poder visualizar la posición actual y el historial de las rutas recorridas por el automóvil a través de Google Maps.

Para el desarrollo del sistema web se ha utilizado el framework Laravel conjuntamente con el patrón MVC como mecanismo de organización de código y la metodología XP para el desarrollo del mismo.

RESUMEN

El presente documento ha sido dividido en 5 capítulos.

En el capítulo 1 se presenta el estudio de soluciones similares al proyecto a desarrollar. Adicionalmente, se describe los principales conceptos de las tecnologías utilizadas en el desarrollo del proyecto.

En el capítulo 2 se presenta un resumen de dos trabajos realizados previamente y que son complementarios al presente proyecto. El primero obtiene, procesa y trasmite información de los sensores de un motor de un automóvil [6]; el segundo realiza una configuración remota para el proceso de obtención de datos del sistema OBD-ll [7]. Además, se realiza el análisis de requerimientos respectivo, y se plantea la arquitectura a implementar.

El capítulo 3 se divide en dos secciones, diseño e implementación. En la parte de diseño se describe la arquitectura de software del sistema, definiendo las funcionalidades de cada módulo establecido en el capítulo 2. Asimismo, se define el diagrama de clases y el diagrama de base de datos cada una con sus respectivas relaciones. Se define cada una de las tablas que forman parte del sistema. Se detallan los procesos que se llevan a cabo dentro del sistema a través de diagramas de secuencia y de flujo. Finalmente para terminar esta sección se realizan bocetos de las interfaces que ofrece el sistema web. En la sección implementación, se detalla los criterios de selección de las tecnologías a usar en el sistema. Además, se presenta piezas de código relevantes para el proyecto las cuales están descritas en base al patrón MVC. Finalmente, se detalla las modificaciones que se realizaron en [7].

En el capítulo 4 se presentan las pruebas realizadas al sistema web en dos modos. El primer modo verifica las funcionalidades que ofrece el sistema web a los diferentes usuarios cuando el automóvil no está siendo monitoreado. El segundo modo es en tiempo real (con el automóvil en movimiento) para verificar el funcionamiento de la

ubicación del automóvil, la toma de medidas de los sensores, y la recepción del mail al producirse una incidencia. Las pruebas en los diferentes modos se realizan por un conjunto de dispositivos del cliente que incluyen: computadora personal, una tableta y un teléfono inteligente.

En el capítulo 5 se presenta las conclusiones obtenidas conforme a las experiencias y conocimientos adquiridos durante el desarrollo del proyecto. Adicionalmente, se presenta un conjunto de recomendaciones que serán de gran ayuda para trabajos futuros.

CAPÍTULO 1

1 FUNDAMENTOS TEÓRICOS

1.1 SOLUCIONES SIMILARES

En la actualidad existe un interés general sobre la localización de vehículos, debido a que se los considera como uno de los bienes más preciados y necesarios para la vida diaria. En los últimos años se han desarrollado proyectos orientados a monitorear la trayectoria de los vehículos que van a la par con las innovaciones tecnológicas que han tenido una gran aceptación por parte de los usuarios [1].

Se ha revisado artículos científicos y proyectos desde el año 2009 hasta el 2015 con el propósito de recolectar información necesaria para el diseño del sistema a desarrollar, entre ellas se tiene: el tipo de lenguajes de programación, motor de la base de datos y API. Asimismo, se analizó las características principales de los trabajos y las arquitecturas de las soluciones. En el estudio realizado se encontró varios sistemas de localización de vehículos, recolección de medidas del motor de forma estática y dinámica; la funcionalidad de estos trabajos son descritos a continuación.

1.1.1 COST EFFECTIVE GPS-GPRS BASED OBJECT TRACKING SYSTEM [2]

Un grupo de investigadores, realizó un estudio en el año 2009, con el objetivo de ofrecer un sistema de seguimiento de vehículos a través de la red GPRS y del sistema de posicionamiento global GPS.

El sistema permite al usuario observar la posición actual y las últimas posiciones registradas del automóvil mediante Google Maps a través de Internet. En la Figura 1.1 se presenta el funcionamiento de la aplicación. El sistema obtiene la posición del automóvil a través del GPS, posteriormente los datos son enviados hacia un servidor

donde está alojado la base de datos través de la red GPRS. Finalmente el usuario accede al sitio web a través de una laptop o una computadora en el cual se visualiza en un navegador la posición del vehículo en el mapa de Google y las posiciones anteriormente registradas.

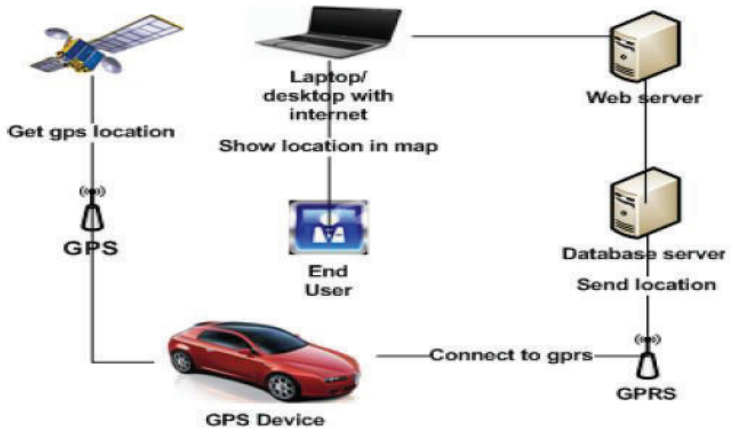

Figura 1.1 Arquitectura del sistema

En la Tabla 1.1 se presenta las especificaciones de hardware que se utilizó para la implementación del sistema.

ESPECIFICACIONES DEL HARDWARE	
Telit GM862-GPS GPS / GPRS	Módulo compatible con Frecuencias: 850MHz /900MHz / 1800MHz / 190MHz de redes celulares.
	Compatible con comandos AT[1]
	Requiere de una antena
	Adaptación de una tarjeta SIM

Tabla 1.1 Especificaciones del hardware

En la Tabla 1.2, se presenta las especificaciones del software.

ESPECIFICACIONES DEL SOFTWARE	
Lenguaje de programación	PHP5, JavaScript
Técnica de envío de datos	Ajax
Almacenamiento de datos	Base de datos MySQL
API	Google Maps

Tabla 1.2 Especificaciones del software

[1] Los comandos AT son instrucciones codificadas que conforman un lenguaje de comunicación entre el hombre y un terminal modem.

1.1.2 A REVIEW OF LOW COST OBJECT TRACKING SYSTEM [3]

Un grupo de investigadores, realizaron un estudio en el año 2013, cuyo objetivo fue ofrecer un sistema de seguimiento de vehículos a través de GPS, GPRS y GIS[2], que sustituya a los sistemas de seguimiento basados en SMS ya que son tradicionales y costosos. El sistema consta de tres partes: el dispositivo de rastreo, el servidor de base de datos y un servidor de mapas.

El sistema utilizó GPRS como el método principal de comunicación entre los vehículos y el servidor. El dispositivo está conectado al vehículo en movimiento y obtiene la posición del GPS en tiempo real. Posteriormente, envía la información de la posición con el número de International Mobile Equipment Identity (IMEI) como su propia identidad en el servidor. Cuando un usuario desea realizar un seguimiento del vehículo debe iniciar la sesión en el sitio web y obtiene la posición exacta del dispositivo en el mapa de Google, como se indica en la Figura 1.2.

En la Tabla 1.3 se detalla las especificaciones de hardware usado para la captura de datos de la posición del vehículo y para el envío de los mismos hacia el servidor donde esta aloja el sitio web.

ESPECIFICACIONES DEL HARDWARE	
Telit GM862-GPS GPS / GPRS	Módulo compatible con Frecuencias: 850MHz /900MHz / 1800MHz / 190MHz de redes celulares.
	Compatible con comandos AT
	Requiere de una antena
	Adaptación de una tarjeta SIM

Tabla 1.3 Especificaciones del hardware

Para el desarrollo del software del sistema se utilizó las siguientes tecnologías que se detallan en la Tabla 1.4.

[2] GIS: Geographic Information System, es el sistema de información capaz de integrar, almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada.

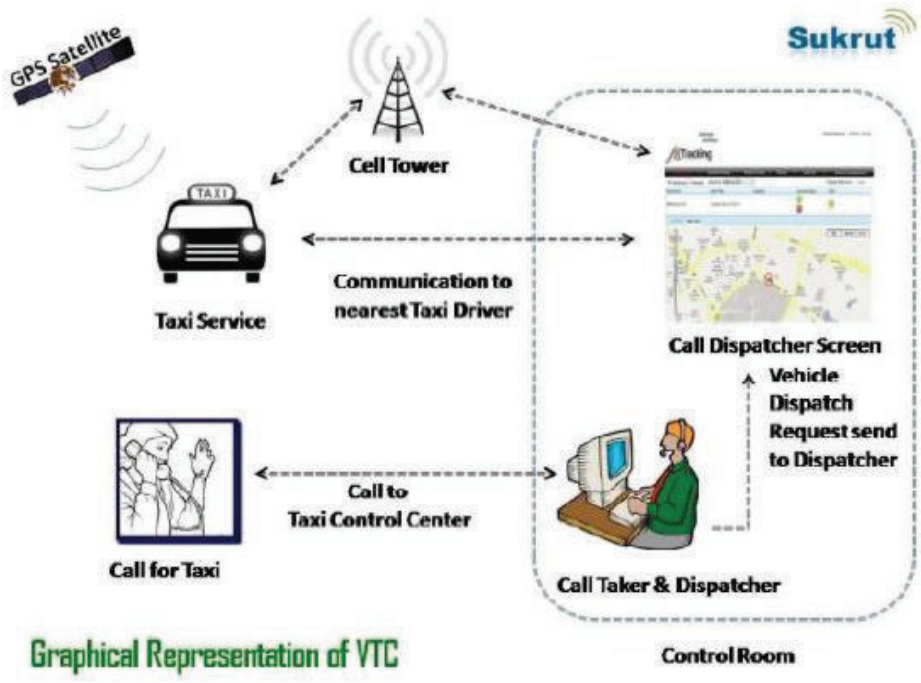

Figura 1.2 Arquitectura del sistema tracking system

ESPECIFICACIONES DEL SOFTWARE	
Lenguaje de programación	PHP5, JavaScript
Se utiliza para buscar la nueva posición del vehículo desde el servidor. Esto lo hace en intervalos fijos para actualizar el mapa sin volver a cargar todo el sitio Web.	Ajax
Almacenamiento de datos	Base de datos MySQL
API	Google Maps y GIS

Tabla 1.4 Especificaciones del Software

1.1.3 MYCARTRACKS [4]

Es una aplicación desarrollada para dispositivos con sistema Android para el seguimiento de vehículos. Una vez instalada la aplicación en los dispositivos móviles se convierten en unidades de seguimiento GPS.

Además envía datos de kilometraje y posición del vehículo. Los datos de seguimiento se envían como paquetes de datos (Wi-Fi, GSM, 3G, 4G Y LTE) desde el dispositivo móvil a los servidores mycartracks donde se almacenan de forma segura en la nube.

Además ofrece una página web para gestionar la ubicación de los vehículos y los datos almacenados.

La aplicación ofrece diferentes paquetes de servicio los cuales se adaptan a las necesidades de los usuarios o empresas. En la Figura 1.3 se detalla el funcionamiento de la aplicación. En la Tabla 1.5 se detalla las especificaciones de hardware para la utilización de la aplicación.

ESPECIFICACIONES DEL HARDWARE	
Dispositivo móvil	Sistema operativo Android
	Compatible con las tecnologías: Wi-Fi, GSM, 3G, 4G y LTE

Tabla 1.5 Especificaciones de Hardware

A continuación en la Tabla 1.6 se detalla las especificaciones de software de la aplicación.

ESPECIFICACIONES DEL SOFTWARE	
Lenguaje de programación	JavaScript
	El código está bajo la licencia Apache versión 2.0[3]
	El contenido está disponible bajo CC BY-AS 3.0
API	Google Maps

Tabla 1.6 Especificaciones de Software

Figura 1.3 Arquitectura de la aplicación

[3] Licencia: http://www.mycartracks.com/terms-and-conditions

1.1.4 DESIGN, DEVELOPMENT, AND VALIDATION OF A REMOTELY RECONFIGURABLE VEHICLE TELEMETRY SYSTEM FOR CONSUMER AND GOVERNMENT APPLICATIONS [5]

Este estudio propone el diseño y desarrollo de un sistema fácil de utilizar para monitorear remotamente el rendimiento y ubicación del vehículo.

El sistema ofrece una interfaz sencilla pero completa en la cual se visualiza las medidas como la velocidad del vehículo, la velocidad del motor, carga del motor, caudales de aire, la aceleración, posición del acelerador, medidor de gas y un parabrisas virtual.

La aplicación propuesta está conformada por dos módulos, el primer módulo es el dispositivo colocado en el vehículo el cual realiza un escáner de la ECU[4] del vehículo y captura las coordenadas geográficas donde está el vehículo en ese instante, este dispositivo se lo conoce como "GPRS and CAN Bus Fault Monitoring". En la Tabla 1.7 se detalla las especificaciones de hardware.

ESPECIFICACIONES DEL HARDWARE	
GPRS and CAN Bus Fault Monitoring	Combina la tecnología de bus CAN y la tecnología de comunicación de GPRS. Los datos se recogen a través del bus CAN y se transmite al servidor por el módem GPRS.

Tabla 1.7 Especificaciones del hardware

El segundo módulo es la aplicación web, la cual propone una interfaz donde se puede visualizar los datos capturados del vehículo. En la Tabla 1.8 se detalla las especificaciones de software.

ESPECIFICACIONES DEL SOFTWARE	
Lenguaje de programación	Python
Entorno de desarrollo para la creación de aplicaciones web	Framework Zope
Base de datos	MySql
Para la interfaz de usuario	CSS, JavaScript y AJAX.

Tabla 1.8 Especificaciones de software

[4] ECU (Engine Control Unit) conocida como la computadora del automóvil.

1.1.5 DISEÑO E IMPLEMENTACIÓN DE UNA SOLUCIÓN TELEMÁTICA BASADA EN OBD-II (ON-BOARD DIAGNOSTIC) QUE PERMITA OBTENER Y PROCESAR LA INFORMACIÓN DE LOS SENSORES DEL MOTOR DE UN AUTOMÓVIL [6]

El objetivo de este proyecto fue implementar una solución telemática[5] la cual permita monitorear los sensores del motor de un automóvil. El dispositivo OBD-II[6] tomará las medidas de los sensores las cuales son procesadas por el microcontrolador Arduino y una vez lista la información, se envía al servidor mediante la red GPRS. A través de un sistema web básico se visualiza la información enviada.

La solución propuesta integra dos subsistemas, cada uno con una función específica permitiendo obtener un sistema funcional. Los subsistemas son:

- En el subsistema de transmisión es el encargado de recolectar, procesar y enviar los datos obtenidos de la ECU del automóvil.
- El subsistema de recepción es el encargado de recibir, leer, decodificar y almacenar los datos enviados desde el subsistema de transmisión, además en este subsistema se aloja una página web básica que permite visualizar la información recolectada.

En la Tabla 1.9 se detalla las especificaciones de hardware que forman parte del subsistema de transmisión.

ESPECIFICACIONES DEL HARDWARE	
Plataforma Arduino	Arduino Mega 2560
Lector de códigos OBD	ELM327
Módulo GPRS	Permite el envío hacia el servidor remoto los mensaje creados en el dispositivo Arduino

Tabla 1.9 Especificaciones del Hardware

[5] Telemática: Es una disciplina científica y tecnológica, originada por la convergencia entre las tecnologías de las telecomunicaciones y de la informática.
[6] OBD (On Board Diagnostics): Es un sistema informático diseñado originalmente para reducir las emisiones mediante la supervisión del rendimiento de los principales componentes del motor.

En la Tabla 1.10 se detalla las especificaciones de software que se ha utilizado en la solución telemática.

ESPECIFICACIONES DEL SOFTWARE	
Lenguaje de programación	PHP, Java, C/C++, HTML, Arduino
IDE Integrated Development	Netbeans
Entorno de desarrollo	Arduino
Base de datos	MySql
Sistema Operativo del Servidor	Centos

Tabla 1.10 Especificaciones del Software

1.1.6 IMPLEMENTACIÓN DE UN SISTEMA DE ADMINISTRACIÓN REMOTA PARA EL PROCESO DE OBTENCIÓN DE DATOS DEL SISTEMA OBD-II DE UN AUTOMÓVIL [7]

Este proyecto propone un sistema telemático que permite realizar una configuración remota para definir dinámicamente los parámetros que se desean monitorear del sistema OBD-II de un automóvil, estos datos son visualizados a través de una interfaz web. La configuración remota la realiza en el subsistema de transmisión a través de una interfaz HMI Web, esta información es guardada y enviada a través de la red GSM en forma de SMS hacia el subsistema de recepción.

En el subsistema de recepción el SMS de configuración es procesado para obtener el tiempo de muestreo, el identificador del automóvil, y los identificadores (PIDs) de los parámetros cuyo valor se quiere obtener. Con esta información se obtiene de la ECU las medidas necesarias. Finalmente se crea un mensaje que es enviado a un servidor remoto a través de la red GPRS.

El subsistema de transmisión es el encargado de recibir el mensaje, lo procesa y lo guarda en la base de datos. Esta información es visualizada por el usuario a través de la interfaz HMI.

Para la implementación de la aplicación web se usó el lengua de programación PHP y se ha desplegado sobre el servidor Apache, en distribución CentOS. Para almacenar los datos se seleccionó MySQL.

1.2 TECNOLOGÍAS USADAS EN EL PRESENTE PROYECTO

A continuación se detallan las tecnologías usadas en el desarrollo de este proyecto. Se incluyeron las tecnologías usadas en los proyectos [6] [7] debido a que fue necesario usar la infraestructura ahí reportada, para comprobar el funcionamiento del sistema.

1.2.1 SISTEMA DE POSICIONAMIENTO GLOBAL (GPS) [8]

GPS son las siglas de Global Positioning System; es decir “Sistema de Posicionamiento Global”, aunque su nombre técnico es NAVSTAR-GPS1. Fue desarrollado por el Departamento de Defensa de los Estados Unidos para fines militares. Permite la determinación de coordenadas espaciales de objetos, personas, vehículos o naves ubicados en cualquier lugar del planeta. Pueden permanecer estáticos o en movimiento y las observaciones pueden realizarse las 24 horas del día sin importar las condiciones del tiempo.

1.2.1.1 Componentes del sistema GPS [8], [9]

El sistema GPS está constituido por tres componentes fundamentales:

Segmento espacial, constituido por 24 satélites artificiales con trayectorias sincronizadas para cubrir toda la superficie del planeta.

Segmento de control, denominado internacionalmente con las siglas OCS (Operational Control Segment) incluye una estación de control principal, una estación de control maestro suplente, 12 de mando y control de antenas, y 16 sitios de monitoreo.

La ubicación de estas instalaciones se encuentra en todo el mundo con el fin de mantener los satélites en órbita y ajustar los relojes satelitales.

Segmento del usuario, se refiere a un equipo integrado por una antena y un receptor, el cual recepta las señales emitidas por los satélites y las procesa para calcular la posición, velocidad y tiempo.

1.2.1.2 Funcionamiento del sistema GPS [9] [10]

La red de satélites se encuentra repartida por la órbita terrestre a una altura aproximadamente de 20.200 Km de altura, girando alrededor de la tierra sin descanso. El principio básico del receptor GPS es calcular la posición de un punto en un espacio de coordenadas (x, y, z).

El receptor captura la señal de un primer satélite calculando el tiempo que tarda en llegar la señal, y conociendo la velocidad de propagación de la misma, este calcula la distancia entre ambos y se determina una esfera, esto indica que el receptor puede estar ubicado en cualquier punto dentro de la superficie de la esfera. Como se indica en la Figura 1.4.

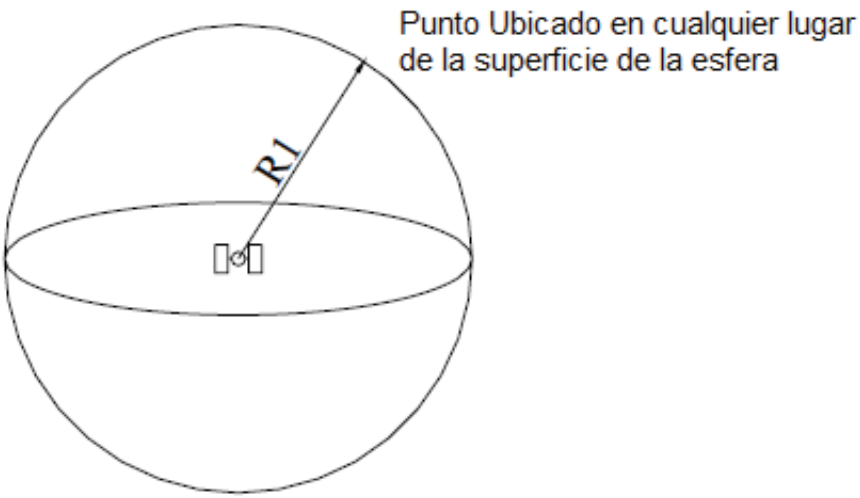

Figura 1.4 Punto ubicado en cualquier lugar de la superficie terrestre [10]

Al realizar los mismos cálculos para un segundo satélite, se genera una segunda superficie esférica, que al intersecarse con la primera esfera, formarán un círculo en cuyo perímetro pudiera estar ubicado el punto a medir. Se puede determinar la ubicación del punto a medir, pero no es precisa. Como se indica en la Figura 1.5.

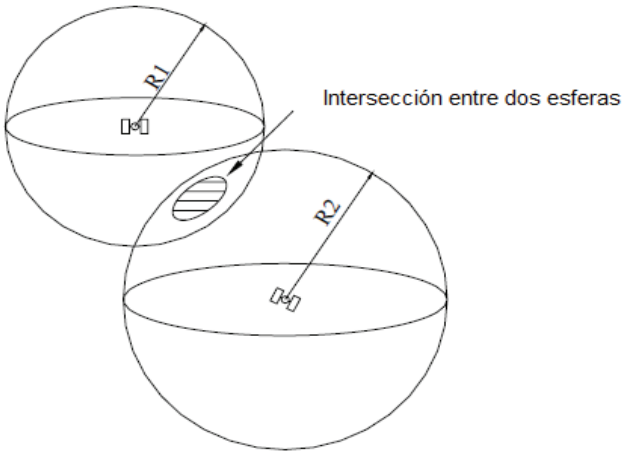

Figura 1.5 Intersección entre dos esferas [10]

Para ello se realiza una tercera superficie esférica, la intersección de la nueva esfera con las dos anteriores se reduce a dos puntos, uno de los dos puntos puede ser descartado por una respuesta incorrecta. Como se indica en la Figura 1.6.

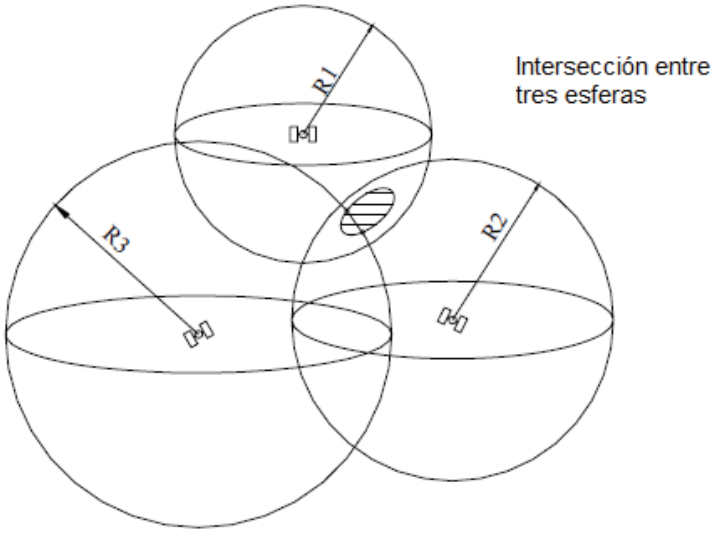

Figura 1.6 Intersección entre tres esferas [10]

Matemáticamente se necesita determinar una cuarta superficie esférica, la intersección de todas ellas determina la posición real del receptor. Este proceso se lo conoce como triangulación, se requiere al menos 4 satélites diferentes para determinar la posición exacta del objeto. En la Figura 1.7 se presenta lo antes mencionado.

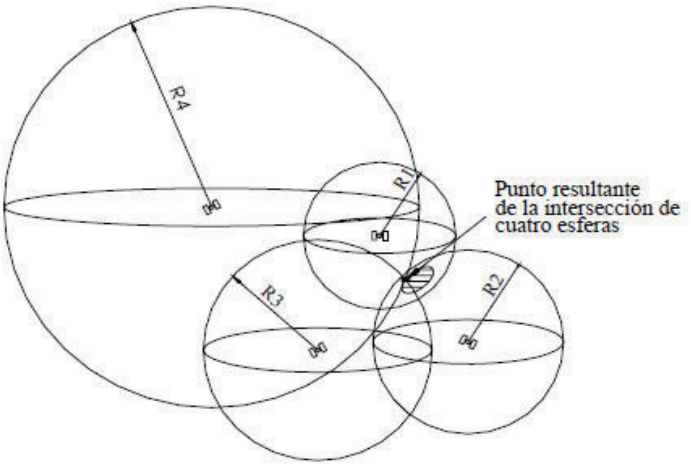

Figura 1.7 Triangulación de satélites [10]

1.2.2 MODEM GPS

1.2.2.1 GPS Shield Dexter [11]

El GPS Shield Dexter no requiere de una antena exterior, debido a que posee una antena integrada inteligente. El GPS Shield es compatible con Arduino UNO, Mega, Duemilanove y Leonardo. Además viene con una batería, SRAM y puentes instalados.

El shield GPS fue desarrollado por industrias Dexter. En la Figura 1.8 se presenta el shield mencionado.

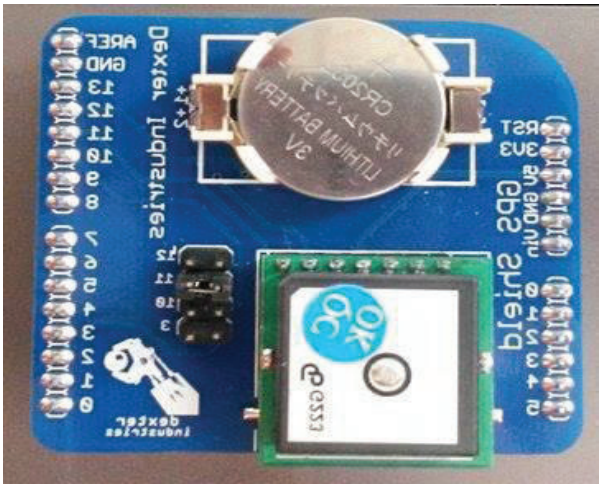

Figura 1.8 GPS Shield

En la Tabla 1.11 se muestran las siguientes especificaciones del shield GPS.

Especificaciones Técnicas	
Exactitud	Posición: 2.5m CEP
	Velocidad : 0,1 m / sec
	Tiempo: 60ns
Tiempo de Inicio	1 segundo arranque bajo el cielo abierto soleado
	29 segundos de arranque bajo el cielo abierto nublado
Sensibilidad	-161 dBm
Velocidad de actualización	Soporta 1/2/4/5/8/10 Hz frecuencia de actualización (por defecto 1 Hz)
Dinámica	4G (39.2m / s2)

Tabla 1.11 (1) Especificaciones técnicas del GPS Dexter

Especificaciones Técnicas	
Límites operacionales	Altitud <18.000m o velocidad <515m / s
Protocolo:	NMEA-0183 V3.01 GPGGA, GPGLL, GPGSA, GPGSV, GPRMC, GPVTG
Datos por defecto	WGS-84
Voltaje de entrada	3.0V ~ 5.5V DC
Corriente de entrada	33Ma
Temperatura de funcionamiento	-40oC ~ + 85Oc
Temperatura de almacenamiento	-55 ~ + 100 ° C
Humedad	5% ~ 95%

Tabla 1.12 (2) Especificaciones técnicas del GPS Dexter

1.3 METODOLOGÍA DE DESARROLLO SELECCIONADA

1.3.1 METODOLOGÍA PROGRAMACIÓN EXTREMA (XP – EXTREME PROGRAMMING) [12], [13]

Es una metodología ágil de desarrollo, la cual está basada en la simplicidad, comunicación, reutilización del código desarrollado, con el objetivo de satisfacer al usuario final.

La metodología XP está orientada hacia los desarrolladores que demandan cambios continuos en el transcurso del proyecto.

XP ofrece a los programadores reestructurar, simplificar, añadir flexibilidad al sistema, corregir los fallos solicitados por los usuarios al realizar pruebas de aceptación, sin cambiar su comportamiento, gracias a la característica de refactorización que posee XP.

Las fases que ofrece la metodología XP se muestran en la Figura 1.9.

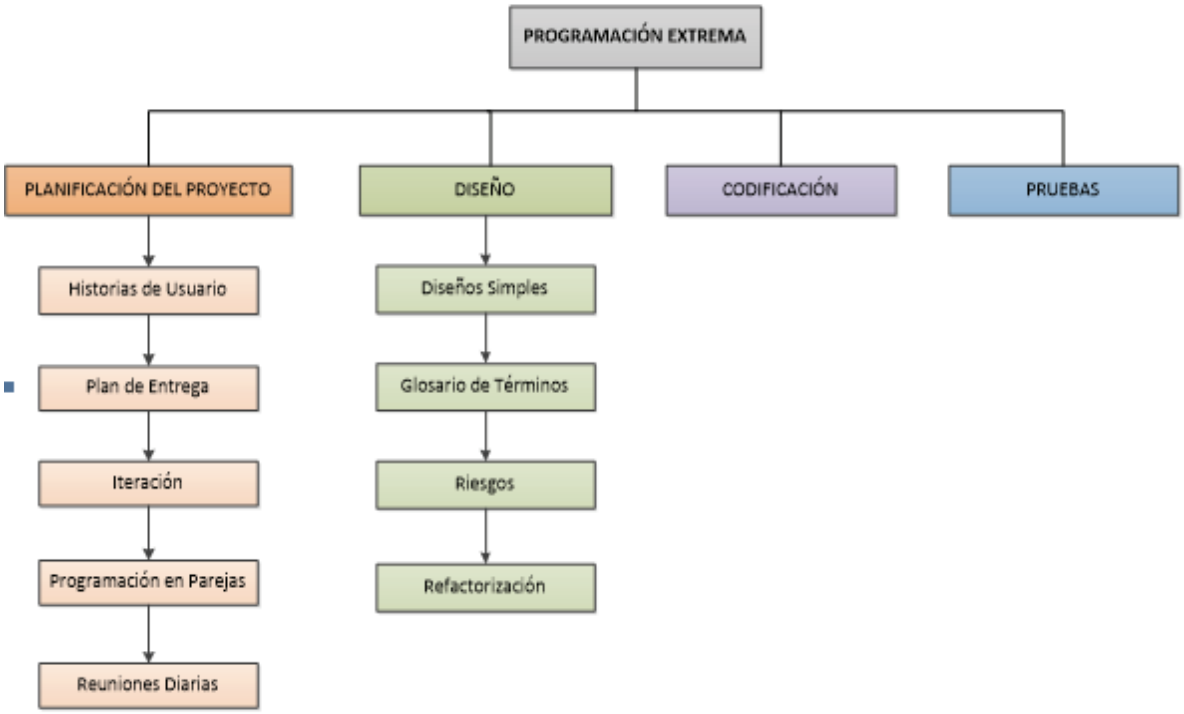

Figura 1.9 Fases de la metodología de desarrollo XP [12]

1.3.1.1 Planificación

- Historias de Usuario

Las historias de usuario son utilizadas como una herramienta para dar a conocer una descripción breve de los requerimientos y necesidades del sistema. Detallan una estimación del tiempo en que se demorará la implementación de cada historia de usuario.

Este proceso es desarrollado por el cliente, por lo tanto debe ser clara y concisa sin profundizar en detalles, debido a que la redacción de la misma se realiza bajo la terminología del cliente.

- Plan de entrega

Una vez terminado la documentación de las historias de usuario se procederá a definir un plan de publicaciones, donde los desarrolladores y clientes establecerán los tiempos ideales para el desarrollo de cada historia. En esta fase se establece a cada historia de usuario una prioridad de elaboración.

- Iteración

Una vez determinado el orden en el cual se implementarán las historias de usuario, estas son asignadas a diferentes iteraciones según el orden de relevancia para el desarrollo del proyecto.

- Programación en parejas

XP recomienda trabajar en pareja pues incrementa el rendimiento del desarrollo del proyecto. El trabajo es dividido en dos partes, en la codificación y en el análisis del método o función la cual se está implementando.

- Reuniones diarias

Para que el proyecto se desarrolle de forma eficiente y sin ningún inconveniente es necesario establecer reuniones para exponer los problemas y soluciones de manera conjunta.

1.3.1.2 Diseño

- Diseños Simples

XP recomienda realizar un diseño simple y sencillo, que sea entendible ya que será más fácil para la implementación y menor tiempo de desarrollo del proyecto.

- Glosario de Términos

XP sugiere usar glosarios de términos y una clara especificación de los nombres que se les otorga a las clases, métodos y funciones al momento de realizar el diseño. Con esto ayudará al desarrollador evitar confusiones o errores al momento de implementar y reutilizar el código.

- Riesgos

En esta fase XP menciona la importancia de trabajar en equipo, debido a que si existen problemas potenciales durante el diseño, se apoyará en los diferentes miembros del equipo para la investigación y solución del problema que se presente.

- Refactorización

Con XP se parte de un diseño sencillo y general, y a medida que el proyecto avanza se realizan las correcciones y se hacen adiciones al mismo, pero siempre conservando su simplicidad.

La refactorización del código tiene como propósito el mejorar y modificar el código sin alterar la funcionalidad.

Este proceso se realiza en cualquier punto del ciclo de vida del proyecto ya que los requerimientos tienen la posibilidad de cambiar.

1.3.1.3 Codificación

La codificación se realiza en forma paralela con el diseño. Al momento de codificar las historias de usuario es necesario que el cliente este presente para certificar que lo implementado cubra las necesidades planteadas en las historias de usuario.

1.3.1.4 Pruebas

Las pruebas son parte fundamental de la metodología XP, ya que de esto depende el éxito del proyecto. Existen una gran variedad de pruebas que propone XP; una de las más usadas durante el proceso de codificación son las pruebas unitarias. Estas pruebas son ejecutadas constantemente ante cada modificación del sistema.

Otras pruebas son las que se realizan conjuntamente con el cliente llamadas pruebas de aceptación las cuales validan las funcionalidades de cada historia de usuario.

1.4 TECNOLOGÍAS WEB

1.4.1 PHP [14], [15]

Hypertext Preprocessor (PHP), es un lenguaje de programación de código abierto del lado del servidor especialmente adecuado para el desarrollo web, es decir, el código fuente escrito en PHP es transparente para el navegador y para el cliente ya que el servidor se encarga de ejecutar el código y enviar el resultado en HTML al navegador.

PHP presenta múltiples características y ventajas, las cuales se listan a continuación:

- Es un lenguaje multiplataforma, es decir, que tiene la capacidad de funcionar en más de un sistema operativo. Por ejemplo un script PHP puede ejecutarse sin cambiar una sola línea de código en cualquier servidor que interprete PHP.
- Es gratuito, porque es una alternativa de fácil acceso para todos.
- Disponibilidad de bibliografía en su página oficial.
- Facilita la conexión con los diferentes motores de base de datos, destacando su conectividad con MySQL.
- Otorga protección del código, es decir, al tener código almacenado en el servidor, este será inalcanzable e inalterable por parte del cliente.
- Fácil de aprender, cualquier persona que tenga conocimiento de algún lenguaje podrá aprender los fundamentos de PHP.
- Dispone de una amplia gama de librerías.

1.4.2 JAVASCRIPT [15]

Es un lenguaje de programación utilizado principalmente del lado del cliente y que es interpretado línea a línea por el navegador.

Es un lenguaje orientado a objetos, debido a que la mayoría de las instrucciones son llamadas a propiedades y métodos de objetos del navegador, y en algunos casos del propio lenguaje.

A continuación se detalla algunas características que ofrece JavaScript.

- Se puede crear efectos atractivos y dinámicos en las páginas web.
- Es un lenguaje independiente de plataformas.
- Es compatible con la mayoría de los navegadores

1.4.3 HTML5 [13]

HTML5 es acrónimo de HyperText Markup Language, versión 5. Es la última versión del lenguaje de marcado de la World Wide Web.

La versión actual ofrece animación de gráficos, soporte de video y audio sin necesidad de instalar plugins[7] adicionales.

HTML5 es adaptable para trabajar sobre diferentes dispositivos móviles para brindar un alto impacto visual conjuntamente con CSS3.

1.4.4 CSS (CASCADING STYLE SHEETS) [13]

Es un lenguaje de hojas de estilos creado para organizar la presentación y aspecto de una página web. CSS3 trabaja conjuntamente con HTML5, es el lenguaje que se encarga de que hacer visualmente atractivo el sitio web.

CSS3 permite colocar al sitio web en vanguardia frente a otros sitios, debido que permite lograr estilo y efectos visuales que antes era solo posible con tecnologías adicionales.

[7] Plugins: Añade una funcionalidad adicional o una nueva característica al software.

1.4.5 AJAX [13]

AJAX es el acrónimo de Asynchronous JavaScript + XML, es una técnica de desarrollo web para elaborar aplicaciones interactivas. Esta técnica se aplica al lado del cliente y adicionalmente solicita al servidor que se carguen los datos. AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano, mejorando la respuesta de la aplicación.

Una característica fundamental de AJAX es la independencia del tipo de servidor y lenguaje de programación web, asimismo es compatible con cualquier plataforma y navegador. AJAX ayuda a crear páginas dinámicas y agiles que actúan como potentes aplicaciones web.

1.4.6 SUBLIME TEXT [16]

Sublime Text es un editor de texto de código multiplataforma, ligero, rápido y fácil de utilizar.

Una de las características de Sublime text es poseer la herramienta “go to anything”, la función es buscar archivos, métodos, variables, funciones e ir a cierta línea de código, esto hace que sea más rápido encontrar algo, sin la necesidad de saber de memoria la estructura de la carpeta. Una de las ventajas que posee Sublime Text es el multi cursor. Permite hacer múltiples cambios al mismo tiempo.

1.4.7 FRAMEWORK LARAVEL [17], [18]

Laravel fue creado en 2011 por Taylor Otwell, es un framework de código abierto para desarrollar aplicaciones y servicios web con PHP de una manera más ágil. Además Laravel facilita la conexión con bases de datos, y consultas de ejecución y extremadamente simples.

1.4.7.1 Principales características de Laravel

Las principales características que ofrece Laravel son:

- Sistema de ruteo

Laravel proporciona flexibilidad en la definición de las rutas de la aplicación, además es posible trabajar con filtros que se ejecuta en rutas específicas. El sistema de ruteo sirve para gestionar las peticiones que son realizadas a través de HTTP. Los filtros ayudan a generar control de acceso sobre las rutas.

- Plantillas Blade

Es un motor de plantillas, son archivos de extensión blade.php que contienen los segmentos de código que se repiten en más de una vista, como por ejemplo la barra de navegación, un menú de opciones, etc. Las plantillas Blade ayudan a reducir el código para la creación de las vistas y evita el código redundante. Blade permite al desarrollador un mejor manejo de las vistas ya que se reutilizará el código ya implementado.

- ORM ELOQUENT

Object Relational Mapper, es una técnica de programación, que permite mapear las tablas en clases, las columnas de una tabla en métodos de una clase, y los nombres de cada columna de una tabla en atributos de una clase, es decir empareja las entidades establecidas en el sistema. ORM permite independizarse de un motor de base de datos porque al momento de tratar con objetos y no como base de datos directamente se puede cambiar el motor de base de datos en cualquier momento y el código continuará funcionando sin ningún problema.

Ventajas que ofrece el uso de ORM se listan a continuación.

- ✓ ORM, facilita las tareas de CRUD (Create, Read, Update y Delete).
- ✓ Ofrece velocidad de consulta.
- ✓ Seguridad de la capa de acceso a datos contra ataques.
- Interfaz de línea de comandos Artisan

Artisan permite ejecutar instrucciones preestablecidas a través de la interfaz de línea de comando CLI, sobre un programa en este caso Laravel. Es una herramienta que

permite de forma rápida y simple crear y realizar tareas como creación de los controladores, verificación de rutas, realizar migraciones entre otras.

- Soporte para MVC

Laravel, presenta una forma de desarrollar aplicaciones web de un modo mucho más ágil. En Laravel se puede usar el patrón de diseño MVC (Modelo-Vista-Controlador).

En la siguiente Figura 1.10 se observa el proceso que se lleva a cabo cuando se realiza una petición al servidor.

1. El navegador envía una petición GET al servidor.
2. El routing direcciona la petición al controlador.
3. El controlador interactúa con el modelo.
4. El controlador envía los resultados a la vista correspondiente.
5. La vista entrega los resultados al navegador.

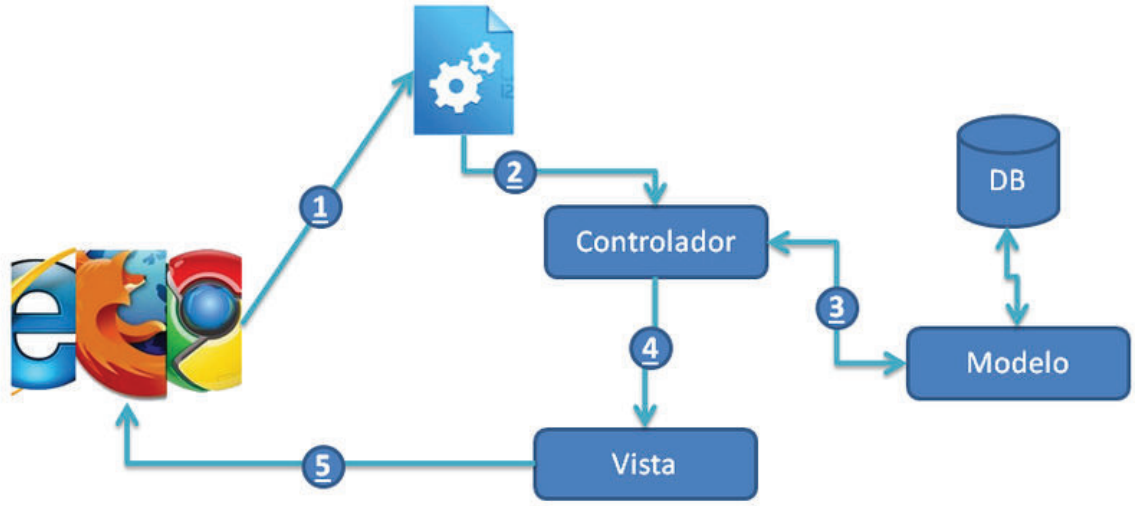

Figura 1.10 Funcionamiento de laravel [19]

1.4.8 FRAMEWORK BOOTSTRAP [20]

Bootstrap es conjunto de herramientas de software libre, el cual simplifica el proceso de maquetación web. Bootstrap realiza un diseño capaz de adaptarse a distintos navegadores y dispositivos móviles sin que el usuario tenga que hacer nada, esto se denomina Reponsive Desing. Este framework ofrece una amplia colección de

elementos web para crear diseños web como: plantillas, menús, botones, etc, que son combinaciones de HTML, CSS y Javascript.

1.4.9 SERVIDOR HTTP APACHE [21]

Es el servidor web de código abierto más utilizado para la creación de páginas y servicios web, que implementan el protocolo HTTP.

Es un servidor web multiplataforma, es decir, puede trabajar con diferentes sistemas operativos y mantener su excelente rendimiento.

1.5 SISTEMA GESTOR DE BASE DE DATOS

1.5.1 MYSQL [22]

MySql es un gestor de base de datos de código abierto. Se encuentra bajo la licencia GNU GPL, por lo que es usada ampliamente en el mundo debido a la compatibilidad que ofrece.

MySQL es una base de datos de alto rendimiento ya que realiza las operaciones a gran velocidad y soporta diferentes sistemas operativos. Además es compatible una con el lenguaje de programación PHP.

CAPÍTULO 2

2 ANÁLISIS DE REQUERIMIENTOS

En el presente capítulo se establecen los requerimientos para el sistema a desarrollar. Se definen los módulos, los actores que interactúan en el sistema y se describe cada uno de los servicios que ofrece el mismo.

La metodología a utilizar durante todo el proceso de desarrollo es XP, esto se debe a la capacidad de respuesta ante imprevistos y cambios en el transcurso del desarrollo; además XP no tiene que adaptarse por completo.

Puede adaptarse solo algunas características que se crean que son valiosas para el sistema.

2.1 ANÁLISIS DEL SISTEMA ACTUAL

Para realizar un adecuado establecimiento de requerimientos se estudió los trabajos previos reportados en [6] y [7].

En el primer trabajo se realizó una solución telemática enfocándose en la lectura de los datos desde el Sistema OBD-II a través del lector ELM327, formando luego un mensaje, con los datos, transmitido sobre UDP hacia un servidor remoto.

En el segundo trabajo se hizo hincapié en configurar de manera remota el proceso de lectura, y se dio soporte para capturar un número variable de medidas.

2.1.1 ARQUITECTURA DEL SISTEMA TELEMÁTICO PARA LA LECTURA DE DATOS DEL SISTEMA OBD-II Y SU ALMACENAMIENTO

El sistema plantea dos subsistemas: de transmisión y de recepción, cada uno de ellos contiene bloques, los cuales serán utilizados y modificados de ser necesario. En la Figura 2.1 se observa la arquitectura del sistema telemático.

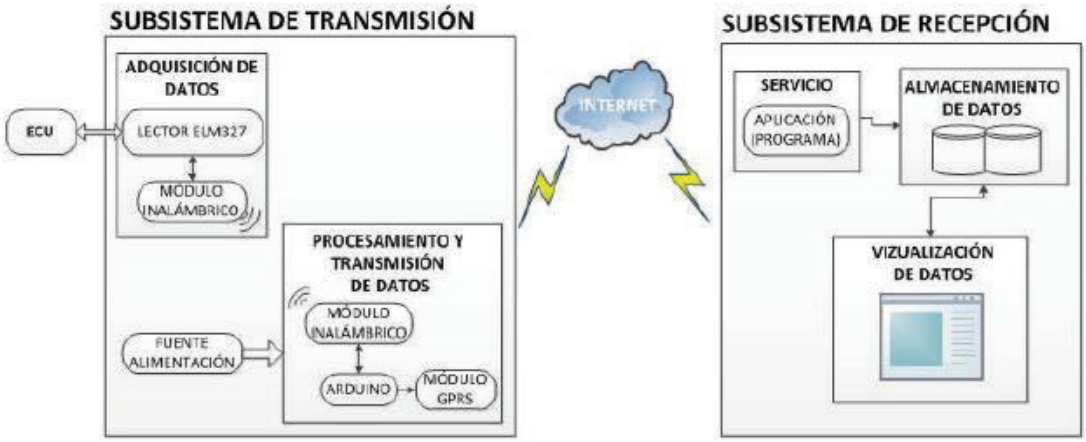

Figura 2.1 Arquitectura del sistema telemático [6]

2.1.1.1 Subsistema de transmisión

Este subsistema es el encargado de adquirir los datos a través del lector ELM327, envía los datos recolectados a través de la red Zigbee al módulo de procesamiento y transmisión de datos. Este módulo se encarga de procesar los datos, colocarlos en un mensaje UDP y enviarlos al subsistema de recepción. Esto se observa en la Figura 2.2.

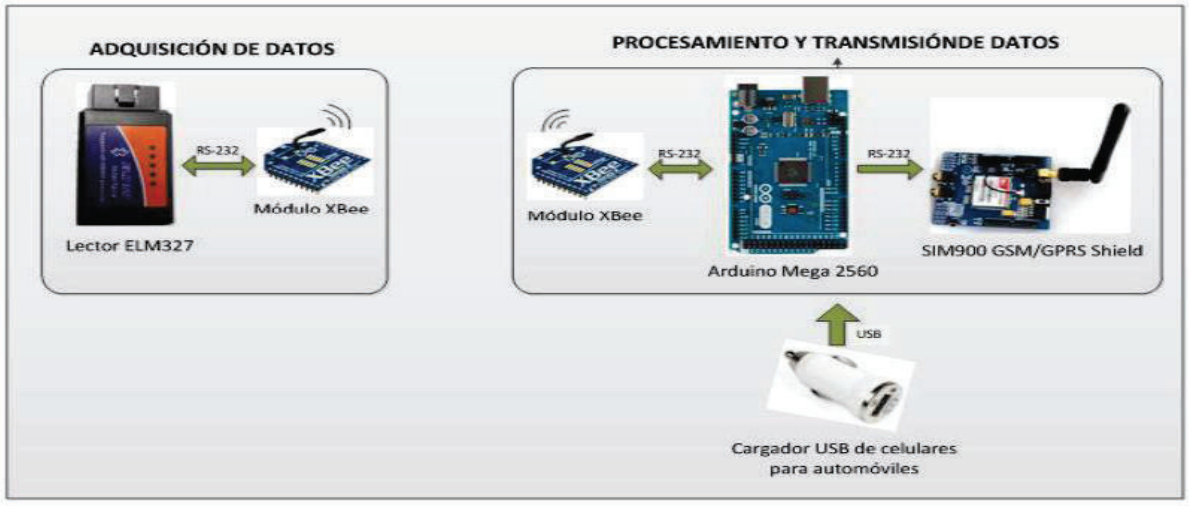

Figura 2.2 Subsistema de transmisión [6]

El subsistema de transmisión está conformado por dos bloques:

- Adquisición de datos

Este bloque utiliza el lector de tipo hardware ELM327, este se conecta al ECU[8] del automóvil a través del conector DCL para la lectura de los códigos OBD-II, adicionalmente este bloque se conecta al bloque de procesamiento y transmisión a través de una conexión inalámbrica usando el módulo Xbee.

- Procesamiento y transmisión de datos

Este bloque está compuesto por: un dispositivo Arduino Mega 2560, un Shield GSM/GPRS SIM900 y un módulo Xbee. Este bloque implementa comandos AT tanto para iniciar la comunicación y para leer los datos de los sensores. Una vez leído los datos, se forma un mensaje UDP, y finalmente se transmite al servidor remoto sobre la infraestructura GPRS.

2.1.1.2 Subsistema de recepción

El subsistema de recepción se encarga de recoger los datos enviados por el módulo de procesamiento y transmisión de datos, los mismos son guardados en la base de datos y visualizados por el cliente. En la Figura 2.3 se visualiza los bloques que forman parte del subsistema.

Figura 2.3 Bloques del subsistema de recepción [6]

- Bloque de servicio

El bloque servicio es un programa que se ejecuta en segundo plano, las funciones principales son:

[8] ECU (Engine Control Unit) conocida como la computadora del automóvil

1. Escuchar el puerto de conexión procedente del subsistema de transmisión.
2. Leer y codificar el mensaje enviado desde el Arduino.
3. Guardar la información en el bloque de almacenamiento.

- Bloque de almacenamiento de la información

Es el bloque encargado de almacenar la información enviada desde el subsistema de transmisión. Para esto se usó una base de datos relacional.

- Bloque de visualización de la información

Este bloque proporciona una interfaz web básica, permitiendo al usuario consultar los valores obtenidos del subsistema de transmisión.

2.1.2 ARQUITECTURA DEL SISTEMA PARA EL PROCESO DE CONFIGURACIÓN REMOTA

El sistema propone dos subsistemas: de transmisión y recepción, cada uno con sus respectivos módulos, los cuales serán utilizado y modificados en caso de ser necesarios. En la Figura 2.4 se observa el sistema de configuración remota.

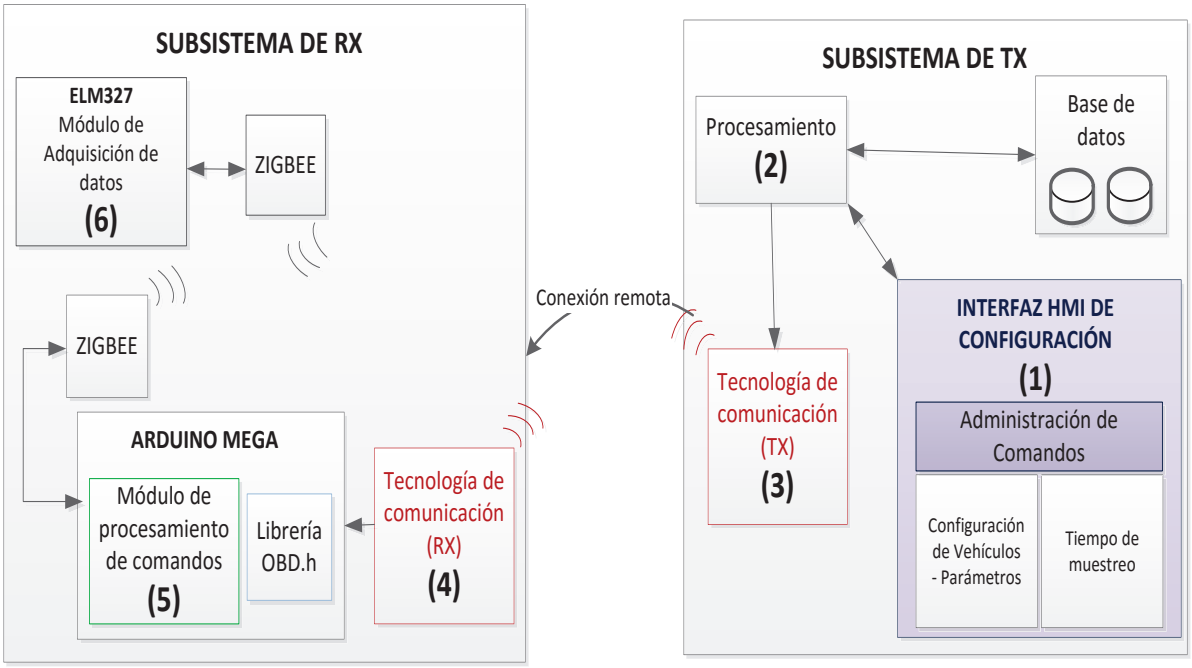

Figura 2.4 Arquitectura del sistema de configuración remota [7]

2.1.2.1 Subsistema de transmisión

El subsistema de transmisión se encarga de realizar: la configuración remota, de almacenar los datos recibidos desde el subsistema de recepción y los configurados en la HMI. Además envía los datos al subsistema de recepción.

Dentro de este subsistema se tiene los siguientes módulos que se describen a continuación:

- Interfaz HMI de configuración

El sistema provee una interfaz de usuario para realizar la administración y configuración de parámetros que se desea obtener del sistema ODB-II. Además permite escoger el tiempo de muestreo para capturar los parámetros seleccionados.

- Procesamiento

Este módulo se encarga de que exista comunicación entre la base de datos, la información de administración y configuración remota.

Además permite interactuar con la tecnología de comunicación inalámbrica para la transmisión de datos.

En este módulo se forma el mensaje para transmitir información al subsistema de recepción. En la Figura 2.5 se observa la trama a enviar en el mensaje. El mensaje tiene el siguiente formato.

Figura 2.5 Formato de la trama [7]

- Tecnología de comunicación

Este módulo utiliza la tecnología GSM con el servicio mensajería corta. Para la implementación de este módulo se usó una la librería Gnokii para plataformas Linux, que da soporte a una gran cantidad de modelos de teléfonos móviles.

2.1.2.2 Subsistema de recepción.

Este sistema se encarga de recibir el mensaje de configuración remota, lo procesa y envía un nuevo mensaje al subsistema de transmisión con la información solicitada.

Dentro del subsistema de transmisión se tiene los siguientes módulos:

- Procesamiento de comandos

Es el encargado de la recepción, lectura y ejecución de comandos, este módulo fue desarrollado en el trabajo previo [6]. En este módulo se forma el mensaje UDP para ser enviado al subsistema de transmisión. El mensaje tiene la siguiente trama como se indica en la Figura 2.6.

Figura 2.6 Forma del Mensaje UDP [7]

A continuación se especifica cada uno de los parámetros de la trama UDP:

- Id: es el identificador del automóvil, seleccionado para la configuración remota.
- PID: es el identificador del sensor.
- Valor: es el valor de la medición del sensor.

Los valores del mensaje UDP son separador por un limitador para identificar y capturar los datos en el módulo de procesamiento. Para separar el identificador del automóvil con los PIDs se coloca ";", para separar el PID del valor se coloca el signo @, finalmente para separar el conjunto de PID y valor se coloca en signo "/". De manera que el mensaje que recibe el módulo procesamiento es el siguiente:

id;*PID*1@*Valor*1/*PID*2@*Valor*2/*PID*3@*Valor*3/…*PID*n@*Valor*n

- Adquisición de datos

El módulo adquisición de datos consiente en la lectura de las medidas de los sensores del sistema OBD-II, este módulo fue diseñado en el trabajo previo [6].

Para poder realizar las pruebas de configuración remota, se adecuo el primer trabajo en torno al envío de datos hacia el servidor remoto, con todas las mejoras indispensables para el correcto funcionamiento del sistema completo. Se modificó el algoritmo en el Arduino para que se almacene la configuración del proceso de lectura, se hizo que el proceso de lectura soporte n medidas, se formó un mensaje a transmitirse con un nuevo formato; en el servidor, se modificó el servicio, y la base de datos en la cual se almacenaban los datos. Puede referirse a los trabajos citados para mayor información.

2.1.3 TECNOLOGÍAS USADAS

A continuación se especifica en la Tabla 2.1 las tecnologías utilizadas tanto en hardware y software que se usaron en los respectivos módulos estudiados.

Procesamiento de comandos . 	Librería software serial.h	Permite la comunicación en serie utilizando los pines digitales del dispositivo Arduino.
	Arduino Mega 2560 R3	Es una plataforma electrónica de hardware. Es programada mediante el lenguaje Arduino. Posee una memoria flash con mayor capacidad a otros dispositivos Arduinos, esto permite alojar gran cantidad de código. Es compatible con cualquier tipo de librerías.
	Librería obd.h	Es de código abierto.Permite el acceso a los valores medidos de los sensores del motor de un automóvil.
	SIM 900 GSM/GPRS	El shield es compatible con el arduino Mega 2560 R3. Permite utilizar la red GSM y GPRS.
	Librería EEPROM.h	Permite leer y escribir información en la memoria EEPROM del Arduino
Adquisición de datos	Lector ELM327	Es un hardware que permite escanear los datos de la ECU
	Módulo XBee	Permite la comunicación de los bloques del subsistema de recepción.

Tabla 2.1Tecnologías usadas [6], [7]

2.1.4 PROPUESTA DEL SISTEMA ACTUAL

El presente proyecto ofrece una interfaz amigable para el usuario final como para el administrador. Las funcionalidades que ofrece el sistema incluyen:

- Interfaces de administración para acciones CRUD sobre usuarios, dispositivos, parámetros de medición, automóviles e incidencias.
- Configuración remota, que permite la selección de medidas que se desea obtener de un automóvil.
- Configuración de incidencias, es decir, se establecen criterios en función de medidas límites de los sensores; en caso de que ocurra una incidencia se enviará una notificación al correo del cliente.
- Visualización de historiales de las medidas obtenidas, permitiendo monitorear el estado del motor de un automóvil, a través de la información recolectada por un conjunto de sensores.
- Visualización la ubicación actual del automóvil a través de un mapa y el estado del mismo.
- Visualización del recorrido del automóvil a través del mapa.
- Visualización de un conjunto de medidas a través de gráficos estadísticos.

A continuación se presenta en la Figura 2.7 los subsistemas, conjuntamente con los módulos utilizados. El SISTEMA WEB (1), permite al administrador y helpdesk realizar la administración CRUD de: usuarios, dispositivos, automóviles, parámetros de medición e incidencias. Además permitirá realizar la configuración remota de parámetros y de incidencias. Las incidencias ocurridas son enviadas al cliente a través de un mail.

Asimismo, se diseña una interfaz para el usuario pueda visualizar un conjunto de medidas, incidencias, gráficos estadísticos, reportes históricos y las marcas de posicionamiento del automóvil.En el módulo de ALMACENAMIENTO (2), se ha diseñado la base de datos partiendo del diagrama de clases. En dicha base se

guarda la información generada por cada uno de los formularios del sistema Web. Dicho sistema está estructurado en capas y se trabaja con el patrón MVC.

El módulo de SERVICIO (7), permite la recepción de los datos enviados por el subsistema de recepción, este módulo fue diseñado en el trabajo previo [6]. Este módulo será utilizado con las modificaciones necesarias.

Tanto el módulo de COMUNICACIÓN DE TRANSMISIÓN (3) y RECEPCIÓN (4), se diseñaron para el envío y recepción del mensaje a través de la red GSM. Este módulo fue diseñado en el trabajo previo [7]. Este módulo será utilizado con las modificaciones necesarias.

El módulo DISPOSITIVO (5), se ha diseñado para la recepción, lectura y ejecución de comandos, este módulo fue desarrollado en los trabajos previos [6], [7]. A Este módulo será utilizado con las modificaciones necesarias. A este módulo se le agregará el módulo GPS para la recolección de las marcas de posición.

El módulo ADQUISICIÓN DE DATOS (6), consiente en la lectura de las medidas de los sensores del sistema OBD-II, este módulo fue diseñado en el trabajo previo [6].

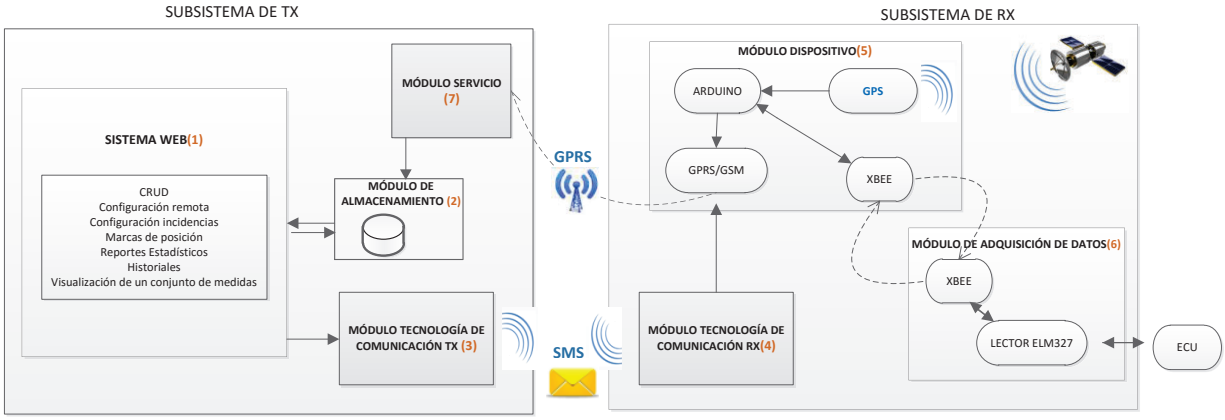

Figura 2.7 Subsistemas del sistema

2.1.4.1 ELEMENTOS DEL SISTEMA

El sistema está compuesto por 7 módulos:

1. Sistema Web

2. Módulo de almacenamiento
3. Módulo de tecnología de comunicación (TX)
4. Módulo de tecnología de comunicación (RX)
5. Módulo dispositivo
6. Módulo adquisición de datos
7. Módulo de servicio

2.2 USUARIOS Y PERMISOS DEL SISTEMA WEB

Se han definido tres actores que interactuarán con el sistema, los mismos que se detallan a continuación:

- Administrador

Como su nombre lo indica es el encargado de administrar el sistema Web, realiza las acciones CRUD de usuarios, automóviles, parámetros, incidencias, modos del sistema OBD-II, y tarjetas SIMs.

Además realiza las tareas de configuración remota y de incidencias. Adicionalmente podrá visualizar la ubicación actual y el historial del recorrido del automóvil.

- Helpdesk

Este actor realiza las tareas CRUD de usuarios, automóviles, parámetros indecencias, modos del sistema OBD-II y tarjetas SIM, realiza las configuraciones de los diferentes procesos de configuración remota e incidencia.

- Cliente final

Utiliza y accede al sistema web a través de cualquier dispositivo para monitorear las medidas realizadas a los sensores del automóvil.

Además podrá visualizar el estado del automóvil, las historias de mediciones, historial de incidencias y la posición actual del automóvil como también la posición en una determinada fecha.

2.2.1 HISTORIAS DE USUARIO

2.2.1.1 Formato de las historias de usuario [23]

En la Tabla 2.2 Formato de historias de usuarios se observa la información que contiene la historia de usuario.

<table>
<tr><td>Número de Historias de Usuario</td><td colspan="3"></td><td>Fecha</td><td colspan="2"></td></tr>
<tr><td>Nombre de Historia de Usuario</td><td colspan="6"></td></tr>
<tr><td rowspan="2">Tipo de Actividad</td><td>Nueva</td><td>Cambio</td><td>Afinamiento</td><td rowspan="2">Estimación</td><td rowspan="2"></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td rowspan="2">Prioridad</td><td>Alta</td><td>Media</td><td>Baja</td><td rowspan="2">Iteración</td><td rowspan="2"></td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>Descripción</td><td colspan="5"></td></tr>
<tr><td>Responsable</td><td colspan="5"></td></tr>
</table>

Tabla 2.2 Formato de historias de usuarios

- N° Historia de Usuario

Identificador de la historia de usuario. Está conformado por las siglas de “HU” y el número de la historia de usuario.

- Fecha

Fecha en la cual se escribe la historia de usuario.

- Nombre Historia de Usuario

Título descriptivo de la historia de usuario.

- Estimación

Número de horas estimadas para la implementación de la historia de usuario.

- Prioridad

Indica el grado de rapidez con el que la historia de usuario debe ser desarrollada dentro del sistema. La prioridad está valorada de la siguiente manera: alta =1, media= 2 y baja=3.

- Descripción

Descripción de los requerimientos establecidos por el cliente.

- Responsable

Nombre del programador responsable de desarrollar la historia de usuario.

- Iteración

Prioridad de la historia de usuario.

2.2.2 DESCRIPCIÓN DE LAS HISTORIAS DE USUARIO

A continuación se describe las historias de usuario planteadas para este proyecto.

2.2.2.1 Módulo de administración de automóviles

En la Tabla 2.3 se presenta la historia de usuario administración de las marcas de los automóviles.

<table>
<tr><td>Número de Historias de Usuario</td><td colspan="2">HU1</td><td>Fecha</td><td colspan="2">30/07/2015</td></tr>
<tr><td>Nombre de Historia de Usuario</td><td colspan="5">Administración de marcas de automóviles</td></tr>
<tr><td rowspan="2">Tipo de Actividad</td><td>Nueva</td><td>Cambio</td><td>Afinamiento</td><td rowspan="2">Estimación</td><td rowspan="2">8</td></tr>
<tr><td>X</td><td></td><td></td></tr>
<tr><td rowspan="2">Prioridad</td><td>Alta</td><td>Media</td><td>Baja</td><td rowspan="2">Iteración</td><td rowspan="2">1</td></tr>
<tr><td>X</td><td></td><td></td></tr>
<tr><td>Descripción</td><td colspan="5">El helpdesk y el administrador realizan el CRUD de las marcas de los vehículos a registrar.</td></tr>
<tr><td>Responsable</td><td colspan="5">Eliana Montero</td></tr>
</table>

Tabla 2.3 Historia de usuario para la administración de marcas

En la Tabla 2.4 se presenta la historia de usuario administración de los modelos de los automóviles.

Número de Historias de Usuario	HU2		Fecha	31/07/2015	
Nombre de Historia de Usuario	Administración de los modelos de los automóviles				
Tipo de Actividad	Nueva	Cambio	Afinamiento	Estimación	4
	X				
Prioridad	Alta	Media	Baja	Iteración	1
	X				
Descripción	El helpdesk y el administrador realizan el CRUD de los modelos de los automóviles.				
Responsable	Eliana Montero				

Tabla 2.4 Historia de usuario para la administración de modelos

En la Tabla 2.5 se presenta la historia de usuario administración de automóviles.

Número de Historias de Usuario	HU3		Fecha	02/08/2015	
Nombre de Historia de Usuario	Administración de Automóviles				
Tipo de Actividad	Nueva	Cambio	Afinamiento	Estimación	10
	X				
Prioridad	Alta	Media	Baja	Iteración	1
	X				
Descripción	El helpdesk y administrador realizan las acciones CRUD sobre los automóviles.				
Responsable	Eliana Montero				

Tabla 2.5 Historia de usuario para la administración de automóviles

2.2.2.2 Módulo de administración de usuarios

En la Tabla 2.6 se presenta la historia de usuario administración de clientes.

Número de Historias de Usuario	HU4		Fecha	4/08/2015	
Nombre de Historia de Usuario	Administración de clientes				
Tipo de Actividad	Nueva	Cambio	Afinamiento	Estimación	4
	X				
Prioridad	Alta	Media	Baja	Iteración	1
	X				
Descripción	El helpdesk y administrador realizan las acciones CRUD de los clientes.				
Responsable	Eliana Montero				

Tabla 2.6 Historia de usuario para la administración de clientes

En la Tabla 2.7 se presenta la historia de usuario administración de helpdesk.

Número de Historias de Usuario	HU5		**Fecha**	05/08/2015	
Nombre de Historia de Usuario	Administración de helpdesk				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	4
	X				
Prioridad	**Alta**	**Media**	**Baja**	**Iteración**	1
	X				
Descripción	El administrador y el helpdesk realizan el CRUD de los helpdesk.				
Responsable	Eliana Montero				

Tabla 2.7 Historia de usuario para la administración de helpdesk

En la Tabla 2.8 se presenta la historia de usuario administración de administradores.

Número de Historias de Usuario	HU6		**Fecha**	06/08/2015	
Nombre de Historia de Usuario	Administración de administradores				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	4
	X				
Prioridad	**Alta**	**Media**	**Baja**	**Iteración**	1
	X				
Descripción	El administrador y el helpdesk realizan el CRUD de los administradores.				
Responsable	Eliana Montero				

Tabla 2.8 Historia de usuario para la administración de administradores

2.2.2.3 Módulo de administración de dispositivos

En la Tabla 2.9 se presenta la historia de usuario administración de Arduinos.

Número de Historias de Usuario	HU7		**Fecha**	16/08/2015	
Nombre de Historia de Usuario	Administración de Arduinos				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	4
	X				
Prioridad	**Alta**	**Media**	**Baja**	**Iteración**	1
	X				
Descripción	El helpdesk y administrador realizan el CRUD del Arduino.				
Responsable	Eliana Montero				

Tabla 2.9 Historia de usuario para la administración de Arduinos

En la Tabla 2.10 se presenta la historia de usuario administrador de SIMs

Número de Historias de Usuario	HU8		**Fecha**	16/08/2015	
Nombre de Historia de Usuario	Administración de SIMs				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	4
	X				
Prioridad	**Alta**	**Media**	**Baja**	**Iteración**	1
	X				
Descripción	El helpdesk y el administrador realizan el CRUD de tarjetas SIM.				
Responsable	Eliana Montero				

Tabla 2.10 Historia de usuario para la administración de SIMS

2.2.2.4 Módulo de administración OBD-II

En la Tabla 2.11 se presenta la historia de usuario administración de modos del sistema OBD-ll.

Número de Historias de Usuario	HU9		**Fecha**	17/08/2015	
Nombre de Historia de Usuario	Administración de modos				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	4
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	1
	X				
Descripción	El administrador y helpdesk realizan el CRUD de los modos de medición, cada modo permite examinar ciertos datos de la ECU.				
Responsable	Eliana Montero				

Tabla 2.11 Historia de usuario para la administración de modos

En la Tabla 2.12 se presenta la historia de usuario administración de parámetros.

Número de Historias de Usuario	HU10		**Fecha**	18/07/2015	
Nombre de Historia de Usuario	Administración de parámetros				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	6
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	1
	X				
Descripción	El administrador y helpdesk realizan el CRUD de los parámetros que se ofrecen en el sistema.				
Responsable	Eliana Montero				

Tabla 2.12 Historia de usuario para la administración de parámetros

2.2.2.5 Módulo de configuración remota para medidas de sensores

En la Tabla 2.13 se presenta la historia de usuario de la configuración remota.

Número de Historias de Usuario	HU11		**Fecha**	29/08/2015	
Nombre de Historia de Usuario	Configuración Remota				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	40
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	2
	X				
Descripción	El helpdesk y administrador seleccionan los parámetros que el cliente desea visualizar en el sistema.				
Responsable	Eliana Montero				

Tabla 2.13 Historia de usuario para la configuración remota

2.2.2.6 Módulo de administración de incidencias

En la Tabla 2.14 se presenta la historia de usuario administración de incidencias.

Número de Historias de Usuario	HU12		Fecha	15/08/2015	
Nombre de Historia de Usuario	Administración de incidencias				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	12
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	1
	X				
Descripción	El helpdesk y el administrador realizan el CRUD de las incidencias				
Responsable	Eliana Montero				

Tabla 2.14 Historia de usuario para la administración de indecencias

2.2.2.7 Módulo de configuración para la detección de incidencias

La Tabla 2.15 se presenta la historia de usuario para la configuración de incidencia.

Número de Historias de Usuario	HU13		Fecha	22/08/2015	
Nombre de Historia de Usuario	Configuración de Incidencias				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	28
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	2
	X				
Descripción	El administrador y helpdesk seleccionan las incidencias de un cliente				
Responsable	Eliana Montero				

Tabla 2.15 Historia de usuario para la configuración de incidencias

2.2.2.8 Módulo de autenticación

En la Tabla 2.16 se presenta la historia de usuario para el proceso de autenticación.

Número de Historias de Usuario	HU14		**Fecha**	09/08/2015	
Nombre de Historia de Usuario	Autenticación				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	6
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	2
		X			
Descripción	Se valida al cliente, administrador y helpdesk, antes de acceder al sistema.				
Responsable	Eliana Montero				

Tabla 2.16 Historia de usuario para la autenticación

2.2.2.9 Módulo de detección de incidencias

En la Tabla 2.17 se presenta la historia de usuario para detección de incidencias.

Número de Historias de Usuario	HU15		**Fecha**	20/10/2015	
Nombre de Historia de Usuario	Detección de Incidencias				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	30
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	2
	X				
Descripción	El sistema detecta la incidencia, guarda e informa al cliente.				
Responsable	Eliana Montero				

Tabla 2.17 Historia de usuario para la detección de incidencias

2.2.2.10 Módulo de reportes

En la Tabla 2.18 se presenta la historia de usuario para la visualización de gráficos estadísticos de los parámetros.

Número de Historias de Usuario	HU16		**Fecha**	01/09/2015	
Nombre de Historia de Usuario	Visualización de gráficos estadísticos				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	30
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	4
			X		
Descripción	El cliente, administrador y helpdesk visualizan las medidas de cada parámetro del automóvil, a través de un gráfico estadístico.				
Responsable	Eliana Montero				

Tabla 2.18 Historias de usuario para los gráficos estadísticos de parámetros

En la Tabla 2.19 se presenta la historia de usuario para la visualización de gráficos estadísticos de incidencias.

Número de Historias de Usuario	HU17		**Fecha**	06/09/2015	
Nombre de Historia de Usuario	Visualización de gráficos estadísticos de incidencias				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	30
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	4
			X		
Descripción	El cliente, administrador y helpdesk visualizan las medidas de cada incidencia que haya ocurrido a través de un gráfico estadístico.				
Responsable	Eliana Montero				

Tabla 2.19 Historia de usuario para gráficos estadísticos de incidencias

2.2.2.11 Módulo de rastreo GPS

En la Tabla 2.20 se presenta la historia de usuario de la ruta del automóvil.

Número de Historias de Usuario	HU18		**Fecha**	12/10/2015	
Nombre de Historia de Usuario	Histórico de posiciones del automóvil				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	24
	X				
Prioridad	**Alta**	**Media**	**Baja**	**Iteración**	3
			X		
Descripción	El cliente, helpdesk y administrador observan la ruta sobre el mapa				
Responsable	Eliana Montero				

Tabla 2.20 Historia de usuario para observar la ruta del automóvil

En la Tabla 2.21 se presenta la historia de usuario para la visualización de la posición actual del automóvil.

Número de Historias de Usuario	HU19		**Fecha**	10/10/2015	
Nombre de Historia de Usuario	Visualización de la posición actual del automóvil				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	30
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	3
		X			
Descripción	El cliente, el administrador, y el helpdesk visualizan la posición actual				
Responsable	Eliana Montero				

Tabla 2.21 Historia de usuario para visualizar la posición actual

2.2.2.12 Módulo de visualización de históricos

En la Tabla 2.22 se presenta la historia de usuario para la visualización de las medidas del motor.

Número de Historias de Usuario	HU20		**Fecha**	01/09/2015	
Nombre de Historia de Usuario	Historial de las medidas del motor				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	8
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	4
		X			
Descripción	El cliente, el administrador y el helpdesk visualizan las mediadas del motor previamente seleccionadas por el helpdesk o el administrador.				
Responsable	Eliana Montero				

Tabla 2.22 Historias de usuario para observar las medidas capturadas

En la Tabla 2.23 se presenta la historia de usuario para la visualización de incidencias.

Número de Historias de Usuario	HU21		Fecha	15/10/2015	
Nombre de Historia de Usuario	Historial de Incidencias				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	17
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	4
			X		
Descripción	El cliente, el administrador y el helpdesk observan el historial de las incidencias ocurridas para un determinado vehículo.				
Responsable	Eliana Montero				

Tabla 2.23 Historias de usuario para observar las medidas de una incidencia

En la Tabla 2.3 se presenta la historia de usuario para el envió de reporte de incidencia al correo electrónico del cliente.

Número de Historias de Usuario	HU22		Fecha	18/10/2015	
Nombre de Historia de Usuario	Envió de reporte de incidencia al correo electrónico de cliente				
Tipo de Actividad	**Nueva**	**Cambio**	**Afinamiento**	**Estimación**	9
	X				
Prioridad	**Alta**	**Media**	**Baja**	**N° Iteración**	4
			X		
Descripción	Al cliente se le envía al correo electrónico la incidencia.				
Responsable	Eliana Montero				

Tabla 2.24 Historia de usuario para el envío del correo electrónico

2.2.3 ANÁLISIS DE REQUERIMIENTO DE USUARIO

En base a las historias de usuario expuestas se definen los siguientes módulos.

- Módulo de administración de automóviles
- Módulo de administración de usuarios.
- Módulo de administración dispositivos
- Módulo de administración OBD-II
- Módulo de configuración remota para medidas de sensores
- Módulo de administración de incidencias
- Módulo de configuración para detección de incidencias
- Módulo de autenticación
- Módulo de detección de incidencias
- Módulo de reportes
- Módulo de rastreo GPS
- Módulo de visualización de históricos

2.2.4 SERVICIOS QUE OFRECE EL SISTEMA WEB

Los requerimientos se han dividido en no funcionales y funcionales.

- Requerimientos no funcionales

Se definen requerimientos no funcionales aquellos que no describen información a guardar, ni funciones a realizar. En la Tabla 2.25 se detallan los requerimientos no funcionales.

#	REQUERIMIENTOS NO FUNCIONALES
1	Cada usuario accederá al sistema web desde cualquier lugar del mundo. La única condición necesaria es una conexión a Internet.
2	El sistema web deberá estar disponible los 365 días del año, 24 horas del día.
3	El sistema web será adaptable a cualquier dispositivo.
4	El sistema debe usar un dispositivo Arduino Mega para procesar y gestionar la lectura de las marcas de posición.

Tabla 2.25 Requerimientos no funcionales

- Requerimientos funcionales

Son las funcionalidades que provee el sistema a cada uno de los usuarios, en la Tabla 2.26 se describe cada uno de estos.

REQUERIMIENTOS FUNCIONALES	
1	El sistema web permitirá crear, modificar y eliminar usuarios.
2	El sistema web permitirá crear usuarios de cada rol definido (administrador, helpdesk y cliente).
3	El sistema permitirá crear credenciales a cada usuario.
4	El sistema permitirá al administrador y al helpdesk crear, actualizar, y eliminar marcas de automóviles.
5	El sistema permitirá al administrador y al helpdesk crear, actualizar, y eliminar modelos de automóviles.
6	El sistema permitirá al administrador y al helpdesk crear, actualizar y eliminar automóviles.
7	El sistema permitirá al administrador y al helpdesk crear, actualizar y eliminar arduinos.
8	El sistema permitirá al administrador y al helpdesk crear, actualizar y eliminar SIMs.
9	El sistema permitirá al administrador y al helpdesk crear, actualizar y eliminar los parámetros a medir del sistema ODB-II.
10	El sistema permitirá al administrador y al helpdesk crear, actualizar y eliminar los modos de medición donde cada uno de estos permite el acceso a los datos de la ECU del automóvil.
11	El sistema permitirá al administrador y al helpdesk realizar la selección de los parámetros que se desea obtener remotamente del automóvil y la selección del tiempo de muestreo para la toma de las mismas.
12	El sistema permitirá al administrador y al helpdesk crear, actualizar y eliminar los criterios en función de las medidas límites de los sensores.
13	El sistema permitirá al administrador y al helpdesk realizar la selección de las incidencias para cada automóvil.
14	El sistema enviará un mail al cliente con la incidencia ocurrida.
15	El sistema permitirá al administrador, al helpdesk y al cliente observar en un mapa la posición actual de un automóvil.
16	El sistema permitirá al administrador, al helpdesk y al cliente escoger por fechas el recorrido del automóvil a través de un mapa.
17	El sistema permitirá al administrador, al helpdesk y al cliente observar en un mismo gráfico estadístico uno o más parámetros de medición, además se podrá filtrar por fechas.
18	El sistema permitirá al administrador, helpdesk y cliente observar en un mismo gráfico estadístico una o más incidencias ocurridas, además se podrá filtrar por fechas.

Tabla 2.26 (1) Requerimientos Funcionales

REQUERIMIENTOS FUNCIONALES	
19	El sistema permitirá al administrador y helpdesk revisar el historial de todas las configuraciones de las incidencias, están ordenadas por fecha.
20	El sistema permitirá al administrador y al helpdesk revisar la última configuración de incidencia realizada previamente de cada automóvil, en donde se podrá observar la fecha en la que se realizó y algunos datos del propietario del vehículo.
21	El sistema permitirá al administrador, helpdesk y cliente observar el historial de las incidencias con la fecha y la hora en que ocurrió cada una de las incidencias
22	El sistema permitirá al administrador, helpdesk y cliente observar el historial de los parámetros con fecha y la hora en que se tomaron las medidas de cada sensor de automóvil.
23	El sistema permitirá al administrador y helpdesk revisar el historial de todas las configuraciones remotas, las configuraciones están ordenadas por fecha.
24	El sistema permitirá al administrador y al helpdesk revisar la última configuración remota realizada previamente de cada automóvil.

Tabla 2.27 (2) Requerimientos Funcionales

2.2.5 HISTORIAS DE USUARIO POR ITERACIÓN

2.2.5.1 Primera iteración

Se realizan cada uno de los módulos de administración del sistema. En la Tabla 2.28 se observa los módulos que forman parte de la primera iteración.

Módulo	Historia de Usuario	Prioridad	Estimación (horas)
Administración de automóviles	Administración de marcas de automóviles	1	8
	Administración de modelos de automóviles	1	4
	Administración de automóviles	1	10
Administración de dispositivos	Administración de SIMs	1	4
	Administración de Arduinos	1	4
Administración OBD-II	Administración de modos	1	4
	Administración de parámetros	1	6
Administración Usuario	Administración de clientes	1	4
	Administración de helpdesk	1	4
	Administración de administradores	1	4
Administración de incidencias	Administración de incidencias	1	12
		Total	64

Tabla 2.28 Historias de usuario de la primera iteración

2.2.5.2 Segunda Iteración

En esta iteración se realiza la configuración remota e incidencias con sus respectivos formularios. Además se realiza el proceso de autenticación para identificar a los diferentes usuarios que forman parte del sistema. En la Tabla 2.29 se observa los módulos que forman parte de la segunda iteración.

Módulo	Historia de Usuario	Prioridad	Estimación (horas)
Configuración remota para medidas de sensores	Configuración Remota	1	40
Detección de incidencias	Detección de incidencias	1	30
Configuración para la detección de incidencias	Configuración incidencias	1	28
Autenticación	Autenticación	2	6
		Total	104

Tabla 2.29 Historias de usuario de la segunda iteración

2.2.5.3 Tercera Iteración

En esta iteración se realiza el proceso de recolectar las marcas de posición que envía el GPS hacia el Arduino y este es enviado a la base de datos.

Una vez que se tiene estas marcas de posición se realiza el módulo GPS. En la Tabla 2.30 se observa los módulos que forman parte de la tercera iteración.

Módulo	Historia de Usuario	Prioridad	Estimación (horas)
Rastreo GPS	Historias de la posición del automóvil	3	24
	Visualización de la posición actual del automóviles	2	30
		Total	54

Tabla 2.30 Historias de usuario tercera iteración

2.2.5.4 Cuarta iteración

En la cuarta iteración se realiza los distintos reportes estadísticos que provee el sistema y la visualización de los parámetros seleccionados. También suministra un

historial de incidencias, por cada incidencia que se genere se envía un mail al usuario informando la incidencia. En la Tabla 2.31 se observa los módulos que forman parte de la cuarta iteración.

Módulo	Historia de Usuario	Prioridad	Estimación (horas)
Visualización de históricos	Historial de medidas del motor	2	8
	Historial de las incidencias	3	17
	Envío de reporte de incidencia al correo electrónico del cliente	3	9
Reportes	Visualización de gráficos estadísticos de parámetros	3	30
	Visualización de gráficos estadísticos de incidencias	3	30
		Total	94

Tabla 2.31 Historias de usuario cuarta iteración

CAPÍTULO 3

3 DISEÑO E IMPLEMENTACIÓN DEL SISTEMA WEB

3.1 DISEÑO

En este capítulo se determina el funcionamiento del sistema, y se describen los diferentes componentes a través de diferentes artefactos diseñados.

3.1.1 ARQUITECTURA DE LA SOLUCIÓN PROPUESTA

En base al análisis previo se definió en la sección 2.2.3 doce módulos que forman el sistema. En la Figura 3.1 se observa los módulos que forman parte del sistema.

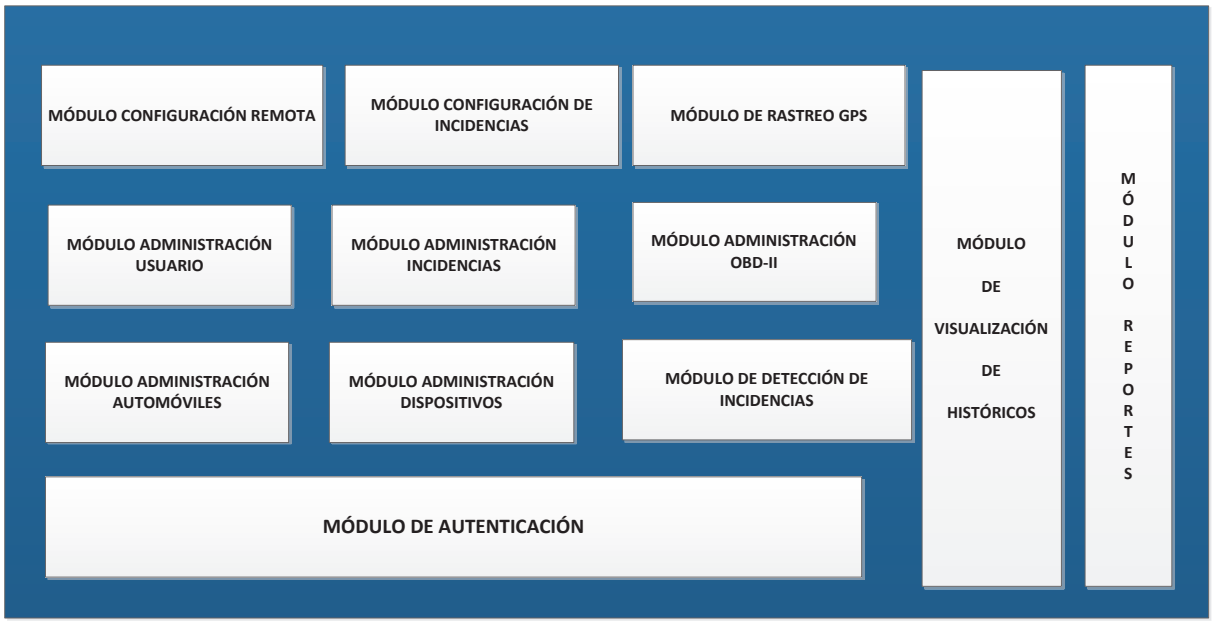

Figura 3.1 Arquitectura de la solución

3.1.1.1 Módulo autenticación

Este módulo es el encargado de validar que el usuario es quien dice ser y dar acceso a una serie de recursos que ofrece el sistema. Para la autenticación se usa una combinación de identificador de usuario y contraseña.

3.1.1.2 Módulo administración de automóviles

Este módulo es el encargado de realizar las siguientes actividades administrativas:

- Administración de marcas de automóviles
- Administración de modelos de automóviles
- Administración de automóviles

En la Figura 3.2 se indica cada una de las funcionalidades dentro del módulo de administración de automóviles.

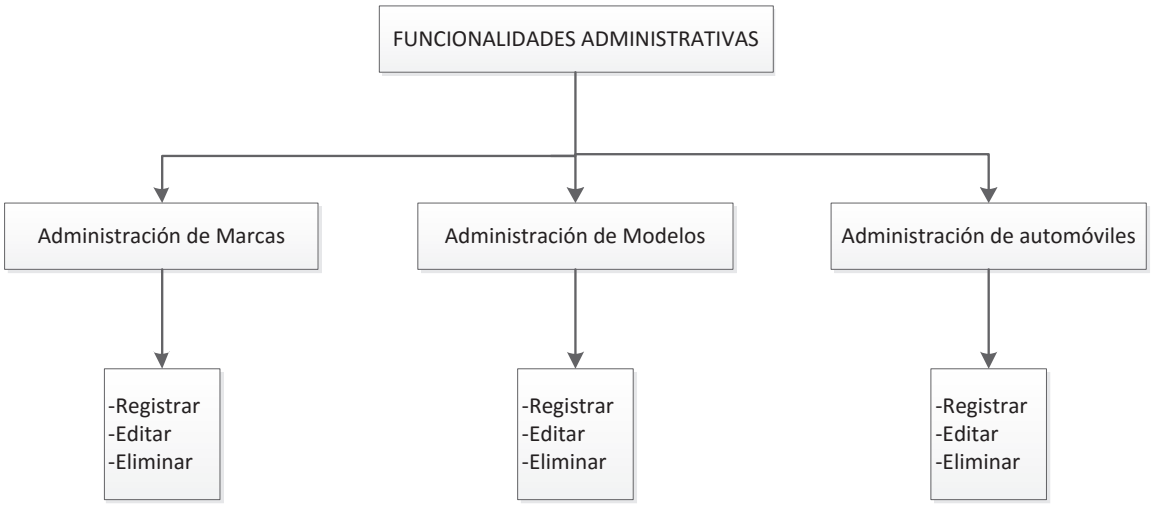

Figura 3.2 Funcionalidades del módulo administración de automóviles

3.1.1.3 Módulo administración de dispositivos

En este módulo se realiza la administración de la tarjeta SIM y del dispositivo Arduino. La tarjeta SIM (Subscriber Identity Module) almacena la clave de servicio del suscriptor usada para identificar a la red. La tarjeta SIM es introducida en el módulo GSM/GPRS con esto se podrá recibir los mensajes que son enviados desde el SISTEMA WEB (1) usando la red GSM y enviar la información generada en el módulo DISPOSITIVO (5) al módulo de SERVICIO (7) a través de la red GPRS, por lo tanto una tarjeta SIM pertenecerá a un módulo GSM/GPRS.

El dispositivo Arduino está constituido por la placa Arduino, el módulo GSM/GPRS y el módulo GPS, por lo tanto un dispositivo Arduino tendrá una tarjeta SIM.

En la Figura 3.3 se presenta las funcionalidades administrativas pertenecientes a este módulo.

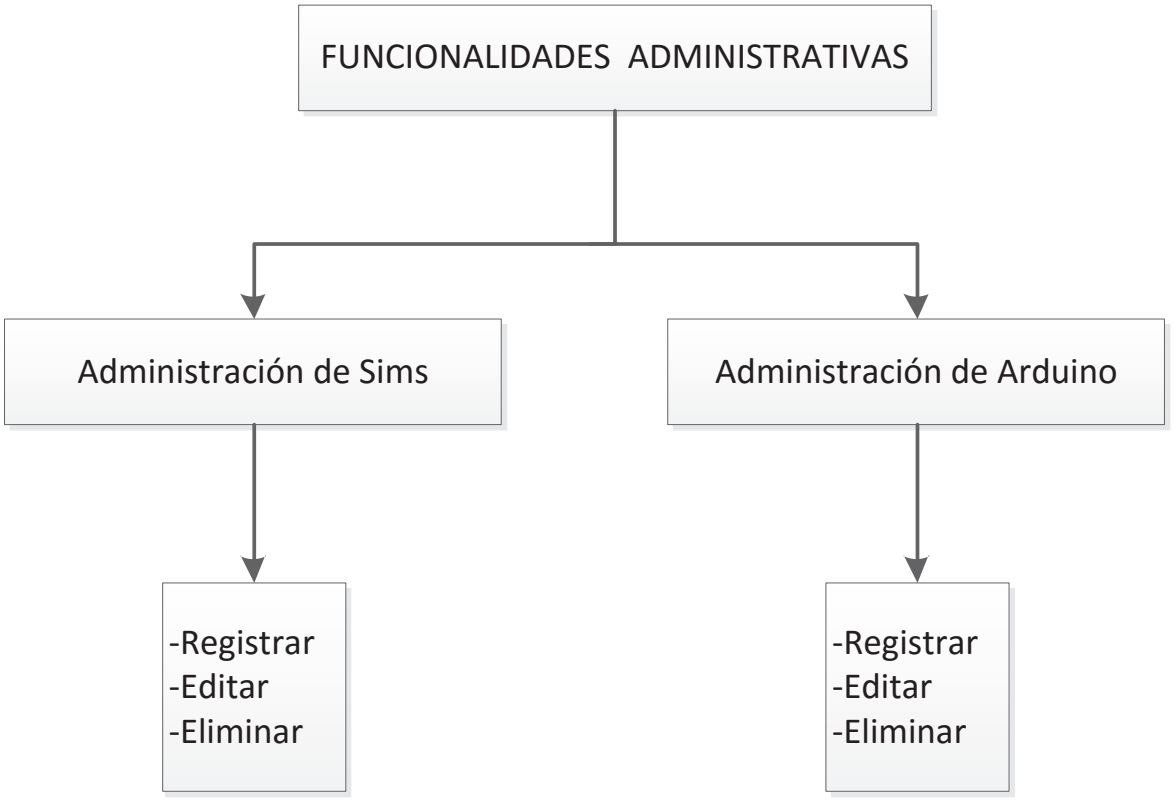

Figura 3.3 Funcionalidades del módulo de administración de dispositivos

3.1.1.4 Módulo administración de OBD-II

El sistema OBD-ll que se está monitoreando está compuesto por un conjunto de parámetros organizados dentro de diferentes modos, pueden ser revisados en detalle en [7].

El poder realizar acciones CRUD sobre ellos permite que en caso de aparecer nuevos parámetros dentro del estándar OBD-II, estos se pudieran registrar en el sistema, y dependiendo del soporte del hardware, el sistema podría realizar procesos de configuración que los incluyan.

En la Figura 3.4 se presenta cada una de las funcionalidades que realiza el módulo.

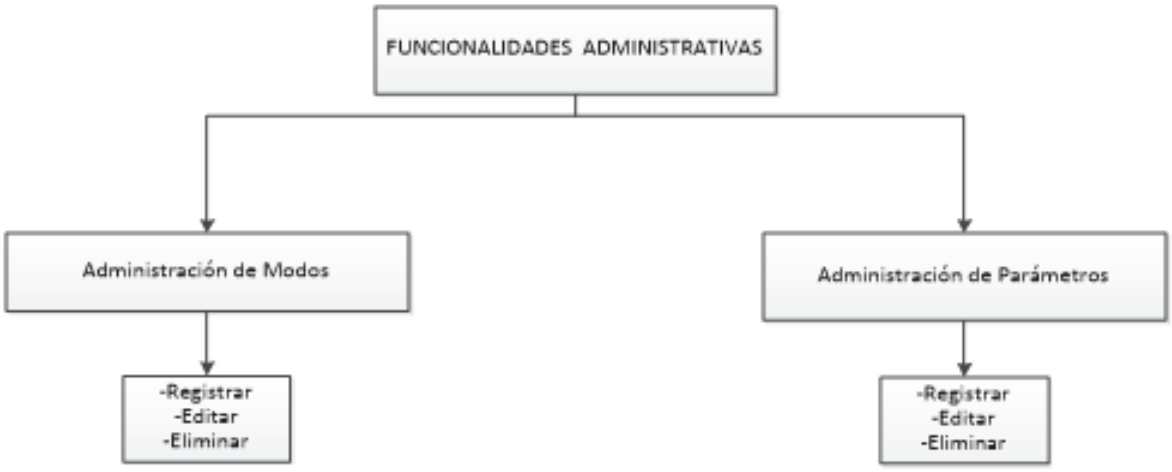

Figura 3.4 Funcionalidades del módulo de administración de OBD-II

3.1.1.5 Módulo administración de usuarios

Este módulo es el encargado de administrar los diferentes usuarios del sistema. En la Figura 3.5 se presenta las funcionalidades administrativas que se realizan en este módulo.

Figura 3.5 Funcionalidades del módulo administración de usuarios

3.1.1.6 Módulo administración de incidencias

Este módulo es el encargado de registrar, actualizar y eliminar incidencias. Las incidencias son criterios en función de los valores que obtengan los parámetros monitoreados. La incidencia tendrá las opciones: igual (=), mayor (>), menor (<). En la Tabla 3.1 se detalla los siguientes ejemplos:

Descripción de la Incidencia	Parámetro	Operación	Valor
Calentamiento del motor	Temperatura del refrigerante del motor	>	95 °C
Reduzca la velocidad	Velocidad del vehículo	>	90

Tabla 3.1 Ejemplos de incidencias

3.1.1.7 Módulo configuración remota

En la Figura 3.6 se presenta cada una de las actividades que ofrece el módulo. Este módulo permite la selección de los parámetros que se desea obtener desde el sistema OBD-II del automóvil. Estos parámetros deben haberse registrado en su administración respectiva.

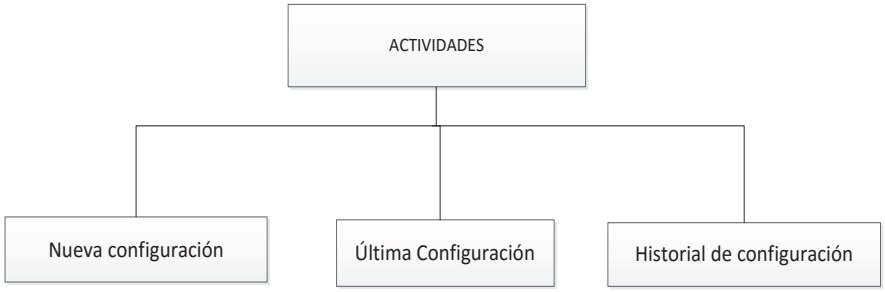

Figura 3.6 Actividades del módulo de configuración remota

- Nueva Configuración: en este ítem se realiza el proceso de la configuración remota con los siguientes pasos detallados a continuación:
    1. El sistema solicita al usuario seleccionar un dispositivo y una tarjeta SIM.
    2. El sistema solicita seleccionar: el modo de operación del sistema OBD-II, los parámetros vinculados a dicho módulo y el tiempo de muestreo.
    3. Antes de guardar y enviar la configuración se presenta un resumen con: datos del propietario, datos del automóvil, datos del dispositivo, los parámetros conjuntamente con el modo al cual pertenecen y el tiempo de muestreo.
- Última configuración: en este ítem se puede revisar los detalles de la última configuración realizada por el administrador o helpdesk.
- Historial de configuración: en este ítem se presenta por fechas ordenadas de forma ascendente todas las configuraciones realizadas. Al escoger una de ellas se visualiza un resumen de dicha configuración.

3.1.1.8 Módulo configuración de incidencias

En este módulo se permite seleccionar las incidencias previamente registradas, en caso de que se cumplan las incidencias se enviará un mail informando al usuario sobre la incidencia producida. En la Figura 3.7 se presenta las actividades que se realizan en este módulo.

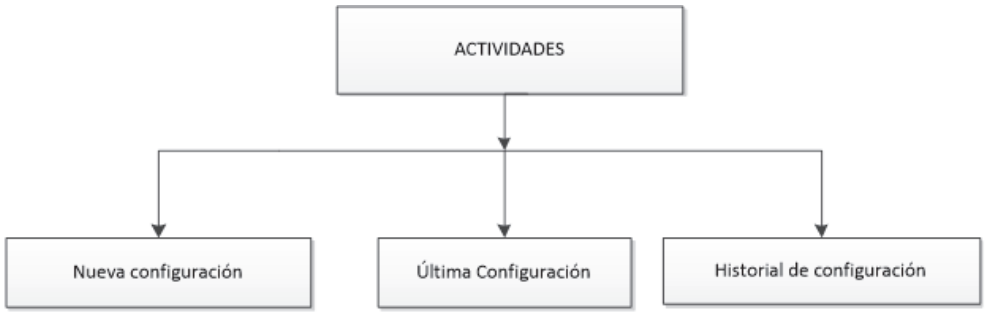

Figura 3.7 Actividades del módulo de configuración de incidencias

- Nueva configuración: en este ítem se realiza el proceso de configuración de incidencias se debe seguir los siguientes pasos.

  1. El sistema ofrece las incidencias previamente registradas, pero únicamente las que estén asociadas a los parámetros previamente seleccionados en el proceso de la configuración remota.
  2. El sistema presenta un resumen antes de que la configuración sea guardada.

- Última configuración: en este ítem se presenta un resumen con los siguientes datos: del propietario, del automóvil, el nombre de la incidencia, la hora y fecha de la configuración de la incidencia.
- Historial de configuración: en este ítem se presenta todas las configuraciones de incidencias realizadas, ordenadas de forma ascendente.

3.1.1.9 Módulo de detección de incidencias

Este módulo permite detectar la incidencia, cuando se cumpla la condición establecida en el módulo administración de incidencias, la incidencia debe estar asociada a un automóvil previamente.

3.1.1.10 Módulo de rastreo GPS

Este módulo ofrece al usuario la verificación de la posición actual del automóvil conjuntamente con los parámetros seleccionados indicando el estado del mismo. Además permite obtener un historial de ubicación del automóvil, la cual puede ser filtrar por fechas.

3.1.1.11 Módulo visualización de históricos

Este módulo tiene la finalidad de ofrecer a los usuarios una interfaz de visualización de la información de las medidas obtenidas después de realizar cada uno de los procesos de configuración, es decir, se presenta cada parámetro con su respectivo valor capturado conjuntamente con la hora y fecha en la que se produjo dicha medición.

Asimismo, se presenta el nombre de la incidencia, las medidas, la fecha y hora de las incidencias ocurridas pero con la diferencia que solo se visualizan aquellas que cumplan con la condición establecida por la incidencia, en caso de que no exista medidas capturadas se presenta un cuadro de dialogo indicando que no existe ningún registro de incidencia.

3.1.1.12 Módulo de reportes

Este módulo ofrece al usuario la opción de observar los parámetros con sus respectivas medidas capturadas tanto de la configuración remota y de incidencias a través de un gráfico estadístico, se puede observar en un mismo grafico una o más parámetros. Además presenta la opción de filtrar por hora y fecha.

3.1.2 MODELO DE CLASE DEL SISTEMA

En la Figura 3.8 se muestra el diagrama de clases planteado para el sistema.

3.1.3 DISEÑO DE LA BASE DE DATOS

En la Figura 3.9 se presenta la estructura de la base de datos del sistema web.

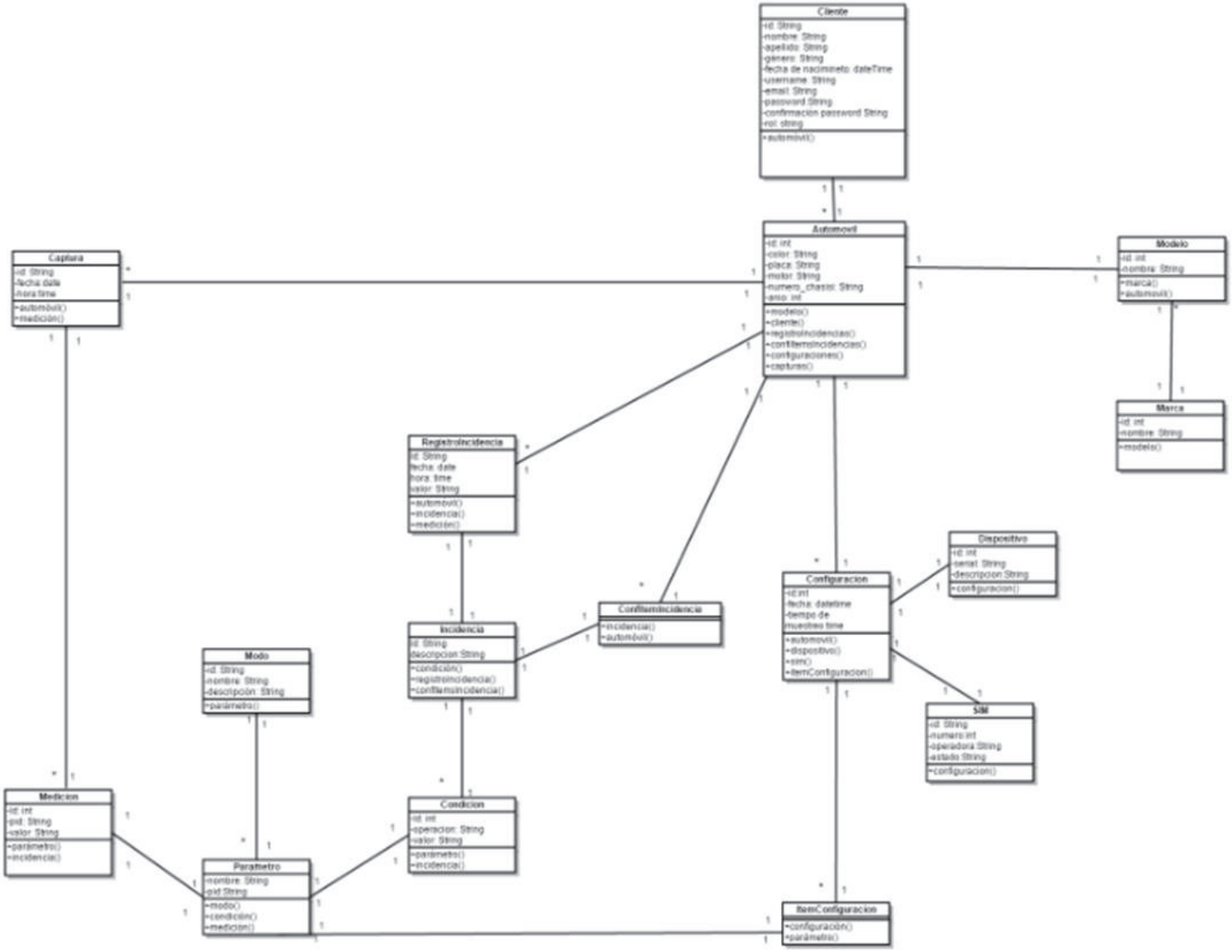

Figura 3.8 Diagrama de clases

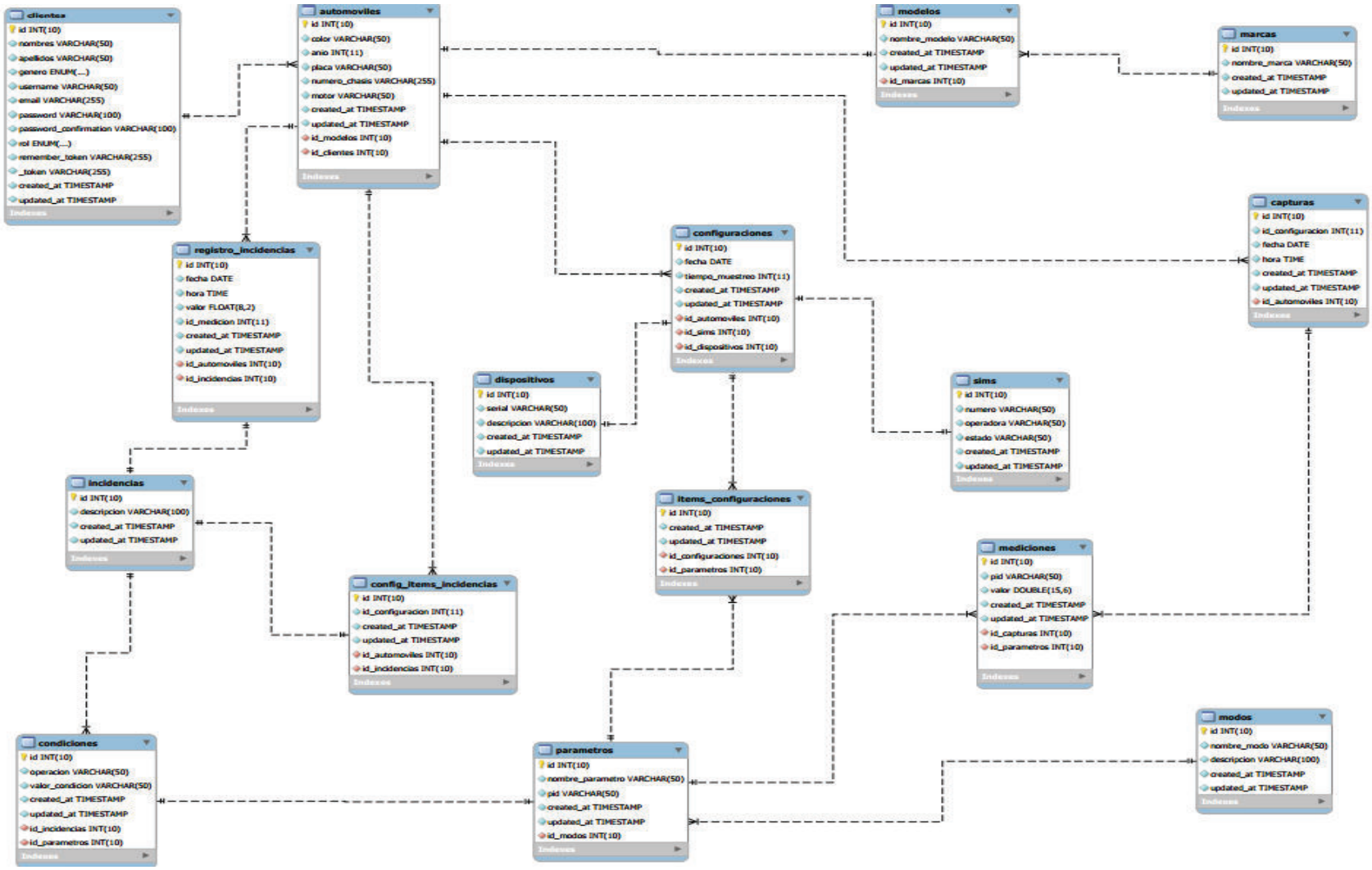

Figura 3.9 Diagrama de base de datos

3.1.4 PROCESOS EN EL SISTEMA

El sistema puede ser descrito a través de un conjunto de procesos que se llevan a cabo dentro del mismo. A continuación se definen los procesos que sirven como patrones dentro del funcionamiento del sistema, es decir, aquellos procesos que tengan un detalle similar no serán descritos.

En la Figura 3.10 se observa el diagrama de actividades para el ingreso del sistema. El usuario debe introducir las respectivas credenciales, si en la base de datos existe el registro, el ingreso será exitoso, caso contrario, se indicará una mensaje denegación.

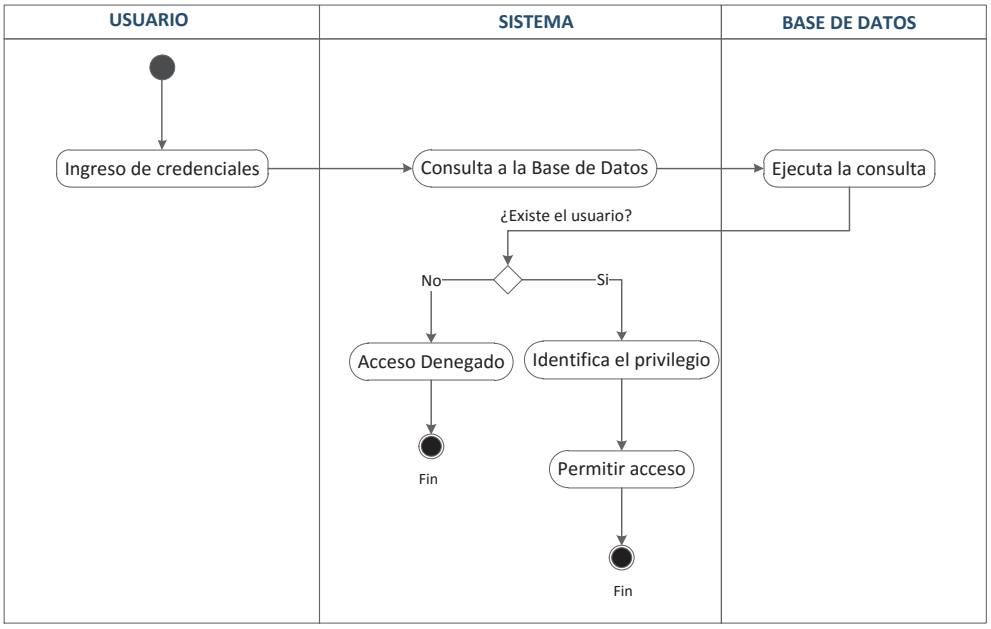

Figura 3.10 Diagrama de actividades para el ingreso del sistema

En la Figura 3.11 se presenta el diagrama de actividades que describe el proceso de registro de usuarios. La interfaz se divide en dos secciones los datos para establecer la sesión de usuario y los datos personales. En la primera sección se coloca el nombre de usuario, el cual debe ser único, el mail con su respectivo formato y la contraseña la cual debe ser confirmanda.

En la segunda sección se introduce el nombre y el apellido, además seleccionará el género a cual pertenece. Todos los datos son requeridos para el registro del usuario, en caso de que los datos sean incorrectos o algún campo este incompleto el sistema

pedirá corregir cada campo, una vez hecho las correcciones se ejecutará el proceso de guardar e indicará al usuario que los datos han sido registrados con éxito. Este proceso es similar para los módulos que realizan esta actividad.

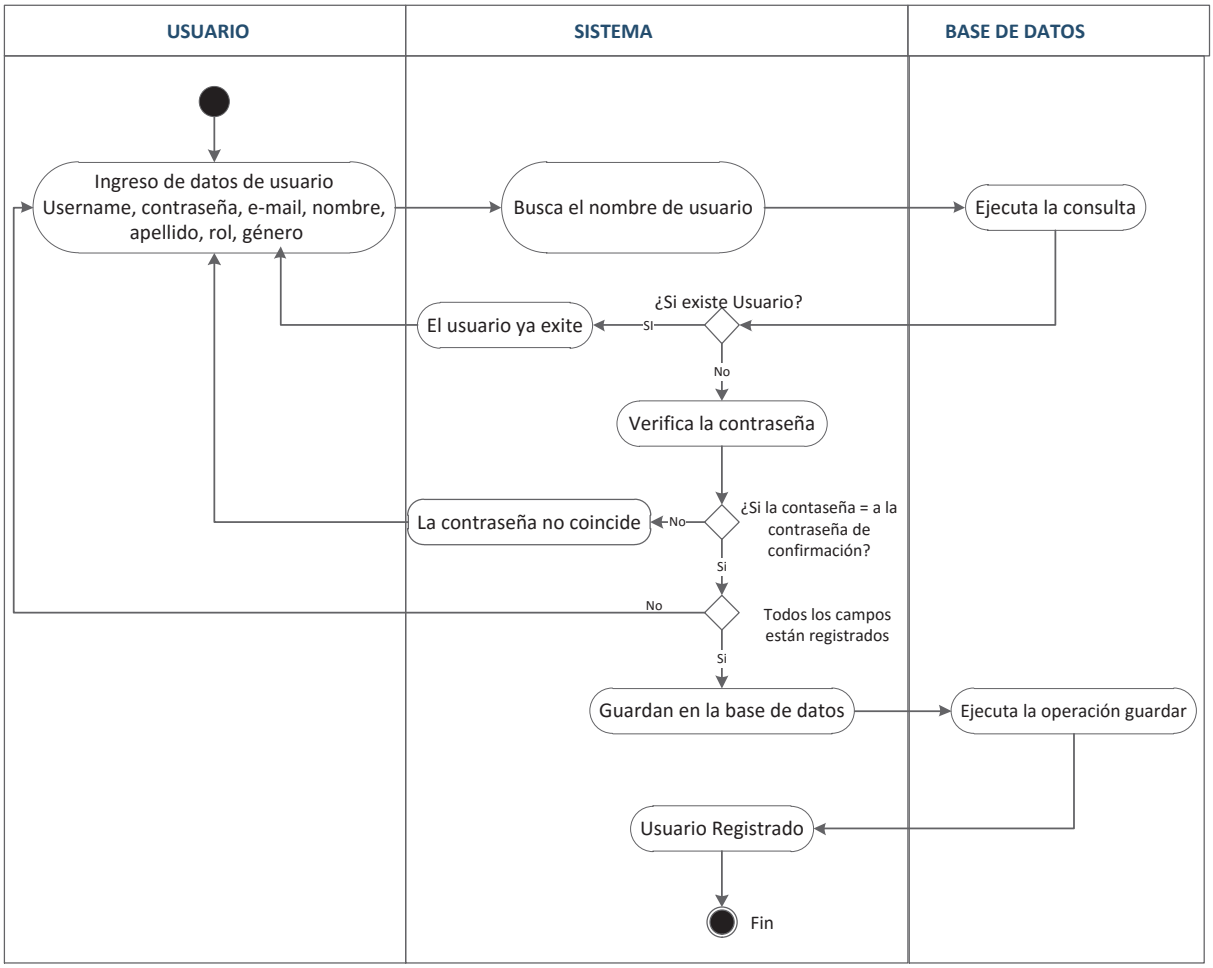

Figura 3.11 Diagrama de actividades para el registro de usuario

En la Figura 3.12 se muestra el proceso de editar un usuario, para lo cual si se edita el username la base de datos validará si el nuevo username existe, en caso de que exista se negará la actualización del campo, de lo contrario se actualizará el registro. Si el cambio es en la contraseña, se validará si es la misma contraseña en el campo confirmación.

Los demás campos, si son editados, se guardarán en la base e indicará al usuario que la actualización se ha realizado. Este proceso es similar para los módulos que realizan esta actividad.

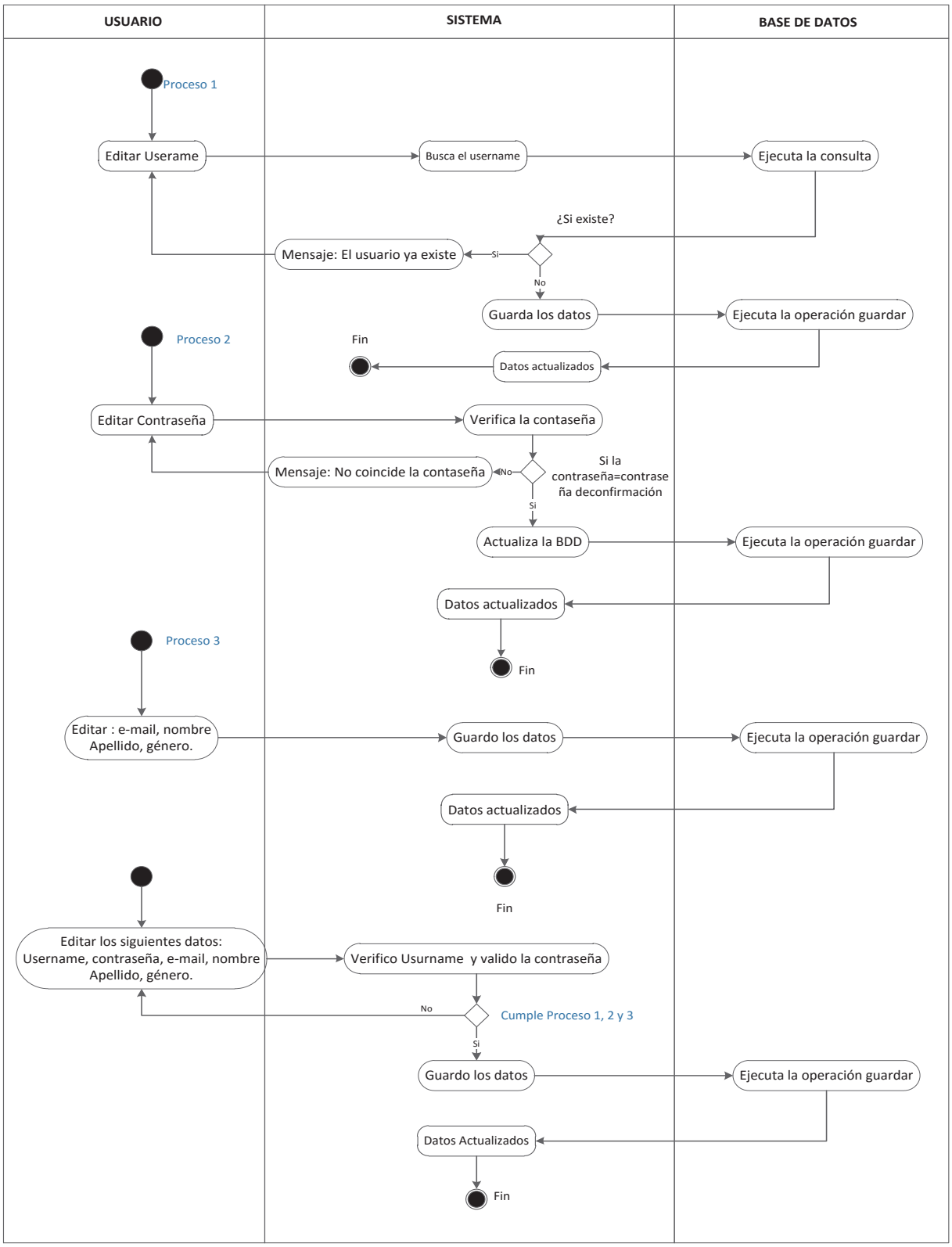

Figura 3.12 Diagrama de actividades para editar un usuario

En la Figura 3.13 se presenta el proceso para eliminar un usuario. Este diagrama se aplica para todos los módulos que realizan la actividad de eliminar algún registro. Se

realizará un solo diagrama debido a que este proceso es similar; solo se diferencia en las consultas que se realiza a la base de datos.

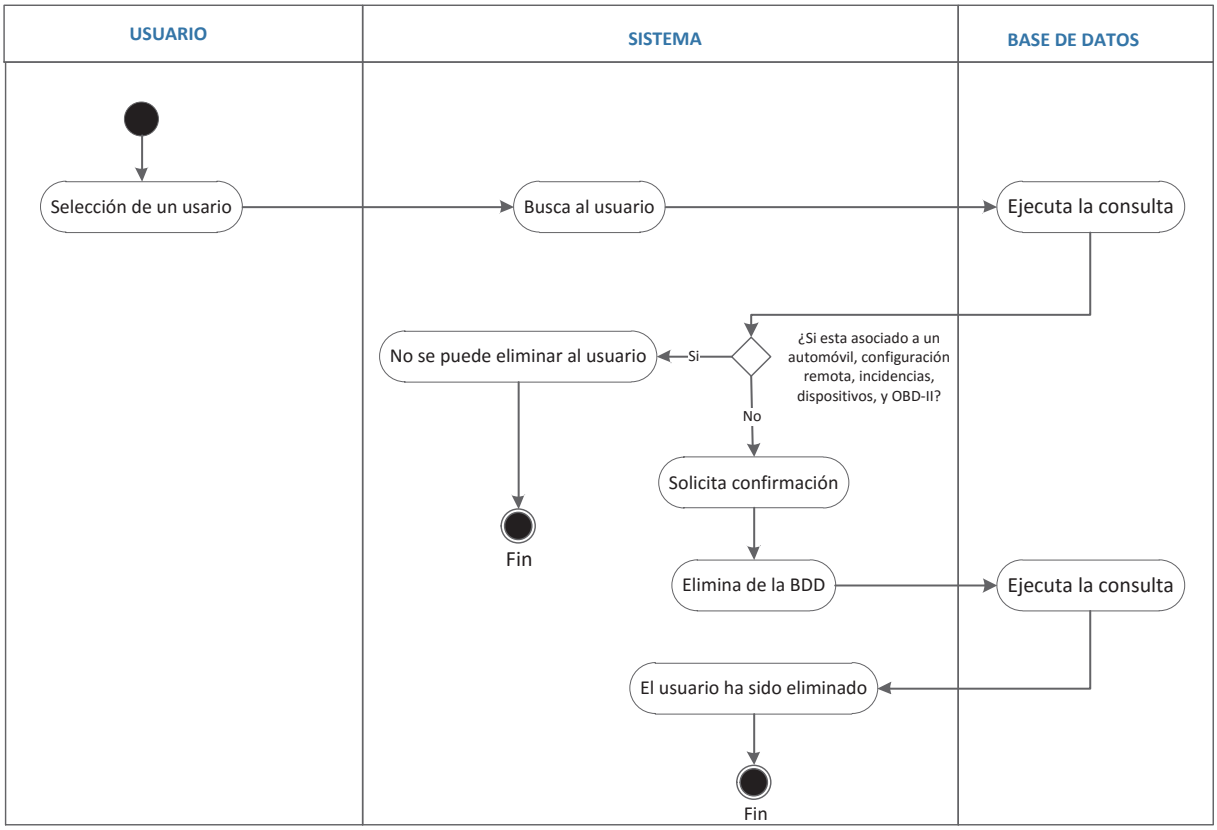

Figura 3.13 Diagrama de actividades para eliminar un usuario

En la Figura 3.14 se presenta el proceso para obtener los reportes estadísticos. Este proceso es similar para el reporte de incidencias y de las mediadas obtenidas al realizar la configuración remota.

En la Figura 3.15 se presenta el diagrama del registro de incidencias, el cliente podrá escoger el parámetro y la operación (>, < o =). El usuario colocará el valor límite para cada parámetro. En caso de que la incidencia necesite ser asociada a mas parámetros se escoge la opción agregar y se realiza el mismo procedimiento indicado anteriormente.

En la Figura 3.16 se presenta el diagrama de configuración remota, el usuario seleccionará una tarjeta SIM y un dispositivo Arduino, en caso de que no haber un Arduino y una SIM disponible, el sistema pedirá registrar los elementos antes mencionados.

Una vez terminado el proceso se procede a la selección del modo del sistema OBD-II y se desplegarán los parámetros asociados a dicho modo, luego se seleccionará el tiempo de muestreo.

Una vez que el cliente acepte la configuración, se guardará esta información en la base de datos y luego se enviará el mensaje de configuración a través de un SMS.

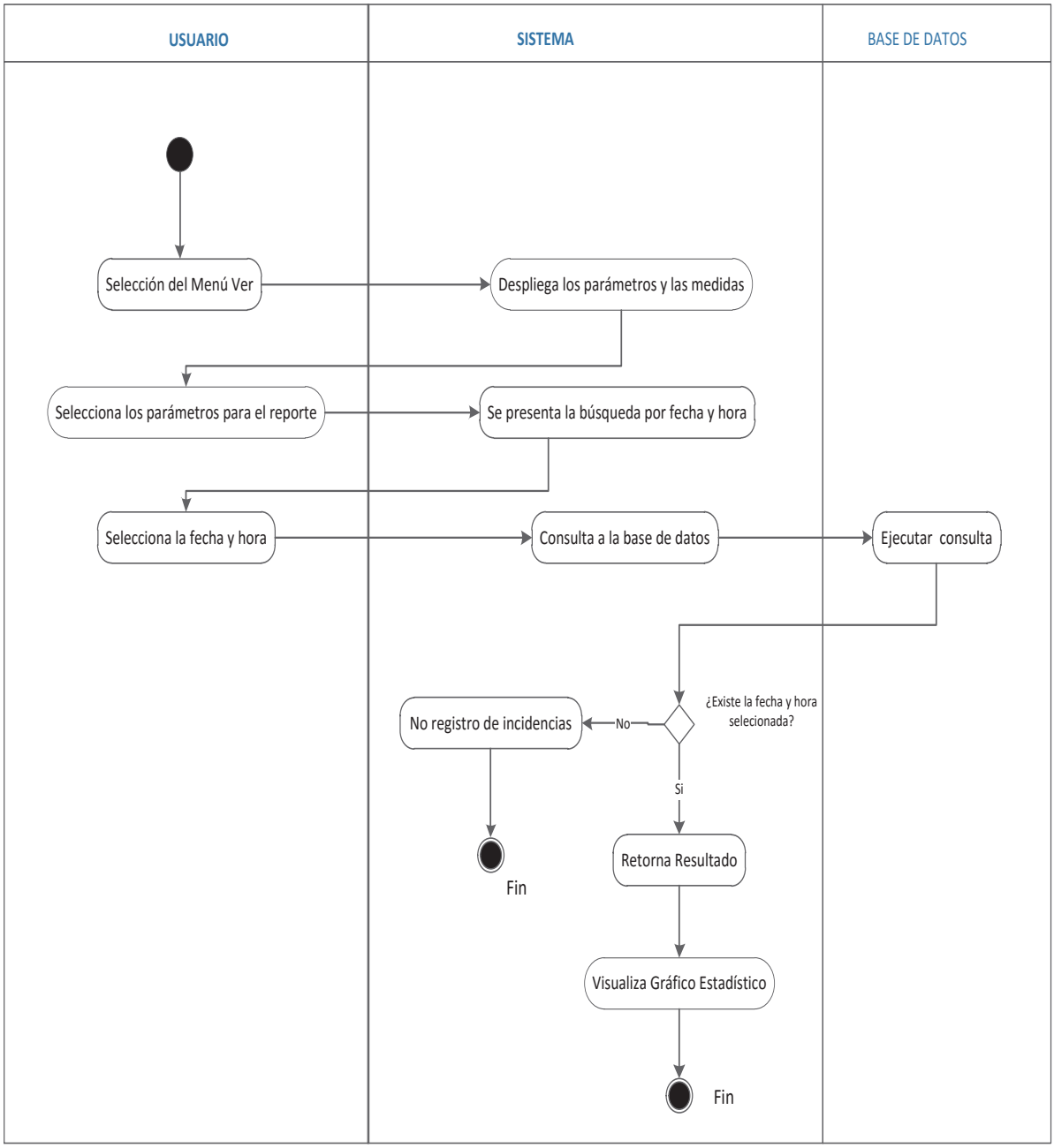

Figura 3.14 Diagrama de actividades para obtener los reportes estadísticos

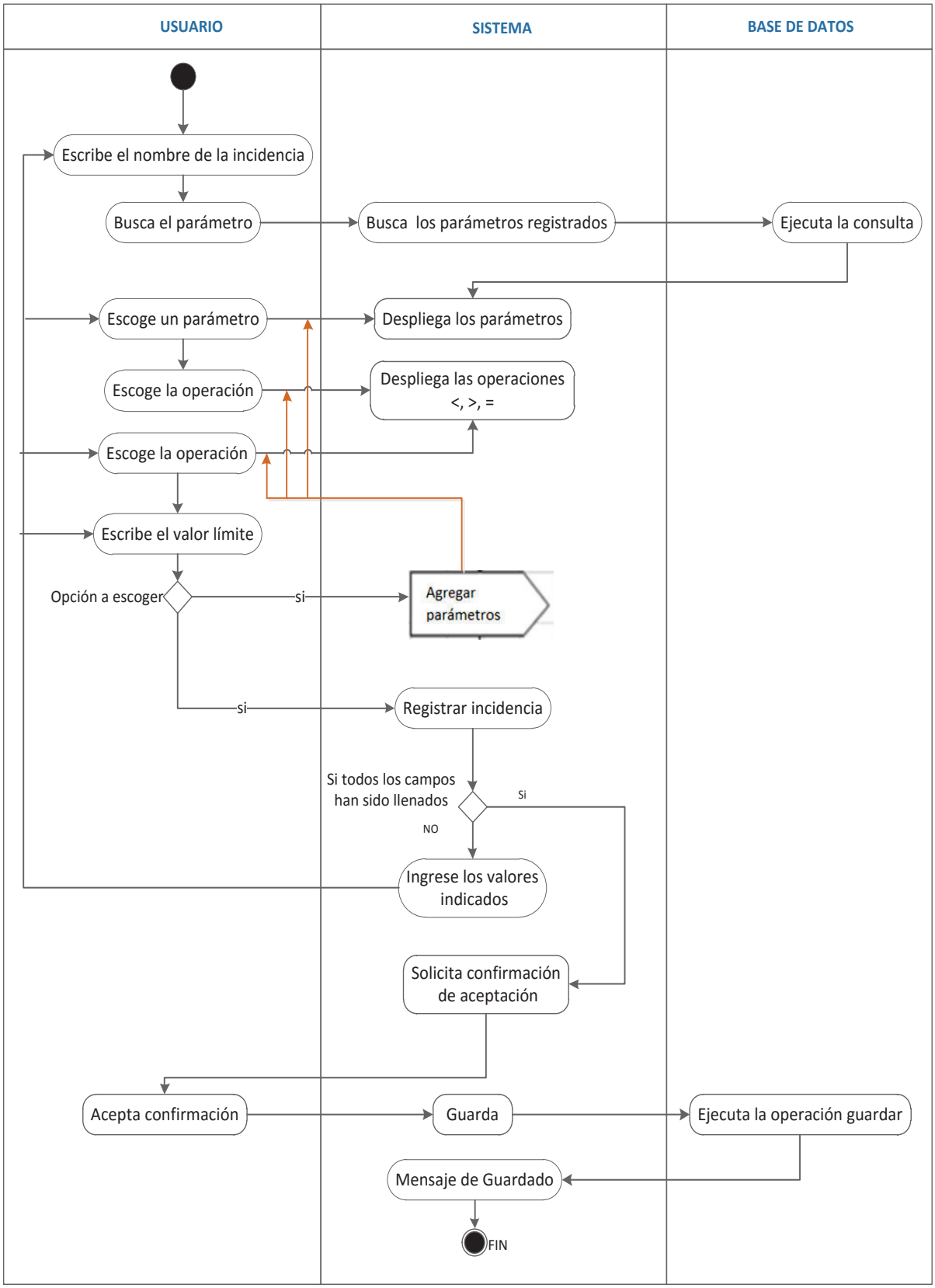

Figura 3.15 Diagrama de actividades para el registro de incidencias

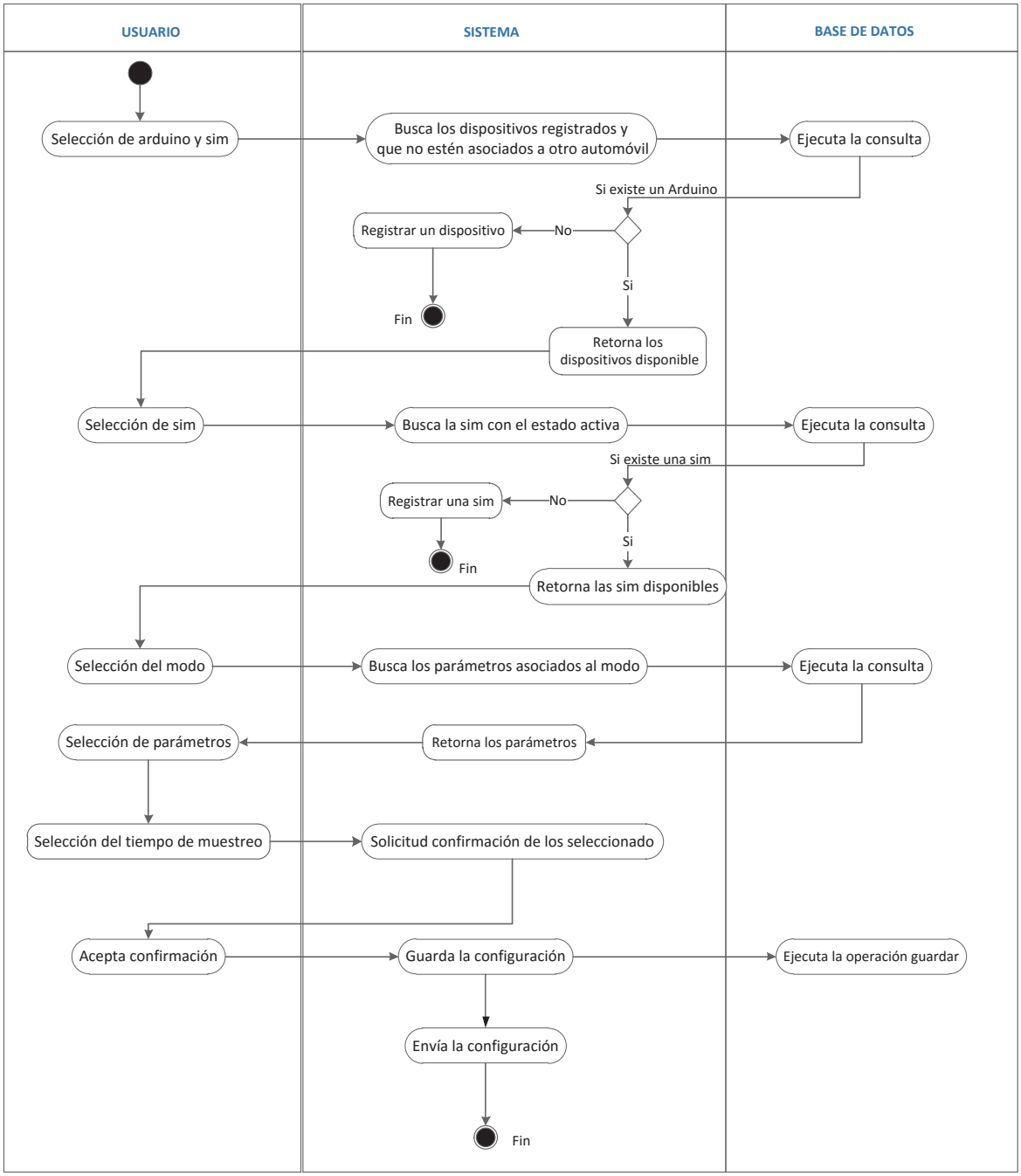

Figura 3.16 Diagrama de actividades de la configuración remota

En la Figura 3.17 se presenta el proceso de la configuración de incidencias, el usuario seleccionará el automóvil al cual se le asignara una o más incidencias, el sistema consulta cuales son los parámetros asociados al automóvil y busca si existe una incidencia asignada para dichos parámetros, una vez terminado el proceso de consulta a la base de datos, el sistema presentará las incidencias al usuario, quien seleccionará cuál de ellas requiere.

Antes de ejecutar el proceso de guardar en la base de datos el sistema presentará un resumen con los datos del propietario, del automóvil y las incidencias seleccionadas, al aceptar dicha configuración el sistema informará que se ha registrado exitosamente la incidencia.

En caso de que no exista ningún parámetro asociado al automóvil el sistema solicitará que se realice el proceso de configuración remota, de igual manera el sistema informará cuando el automóvil ya está asociado a la configuración de incidencias, por lo que el sistema pedirá realizar una nueva configuración remota o editar en caso de ser necesario.

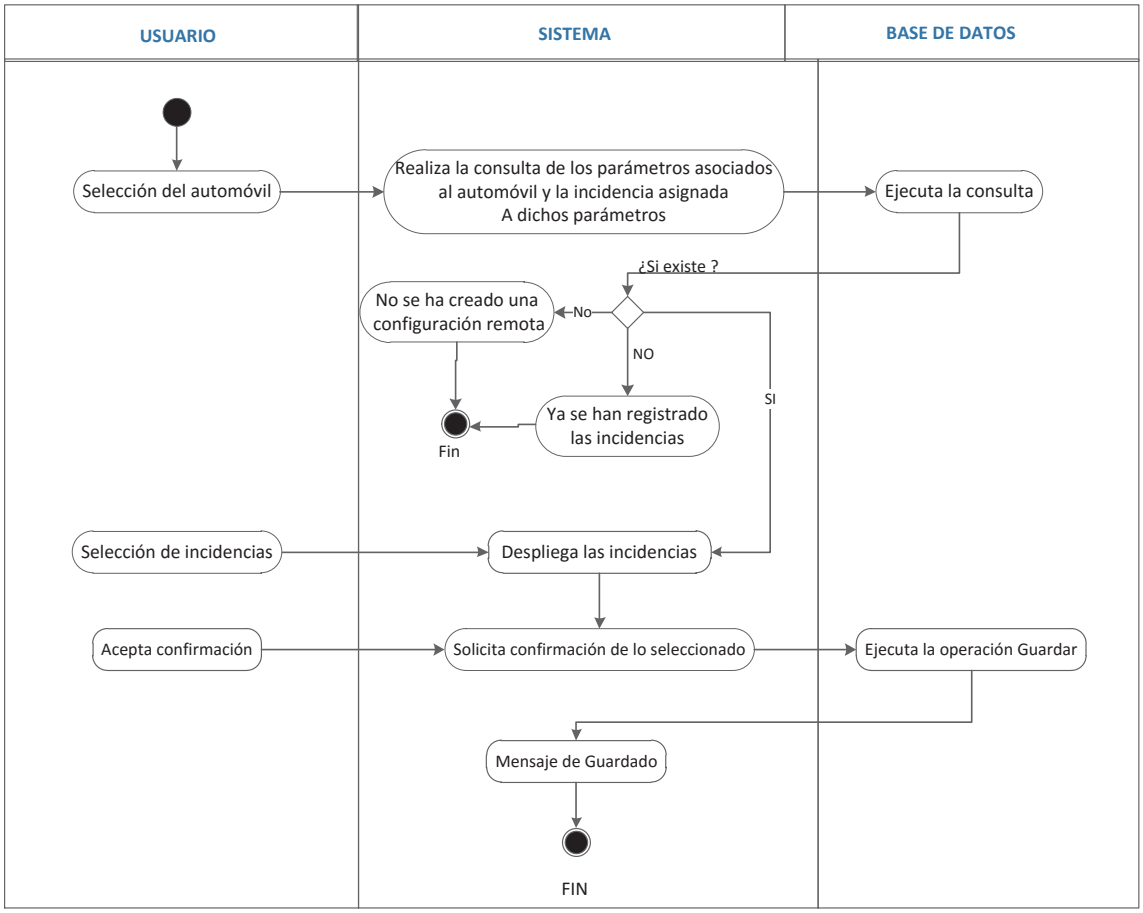

Figura 3.17 Diagrama de actividades de la configuración de incidencias

En la Figura 3.18 se presenta el proceso de detección de incidencias, el sistema realiza una comparación entre las medidas de los parámetros obtenidos y las condiciones que se establecieron al momento de registrar la incidencia. Si es exitosa

la comparación se guarda la incidencia, se estructura el mensaje a enviar y finalmente se envía el mail a la cuenta de correo del usuario.

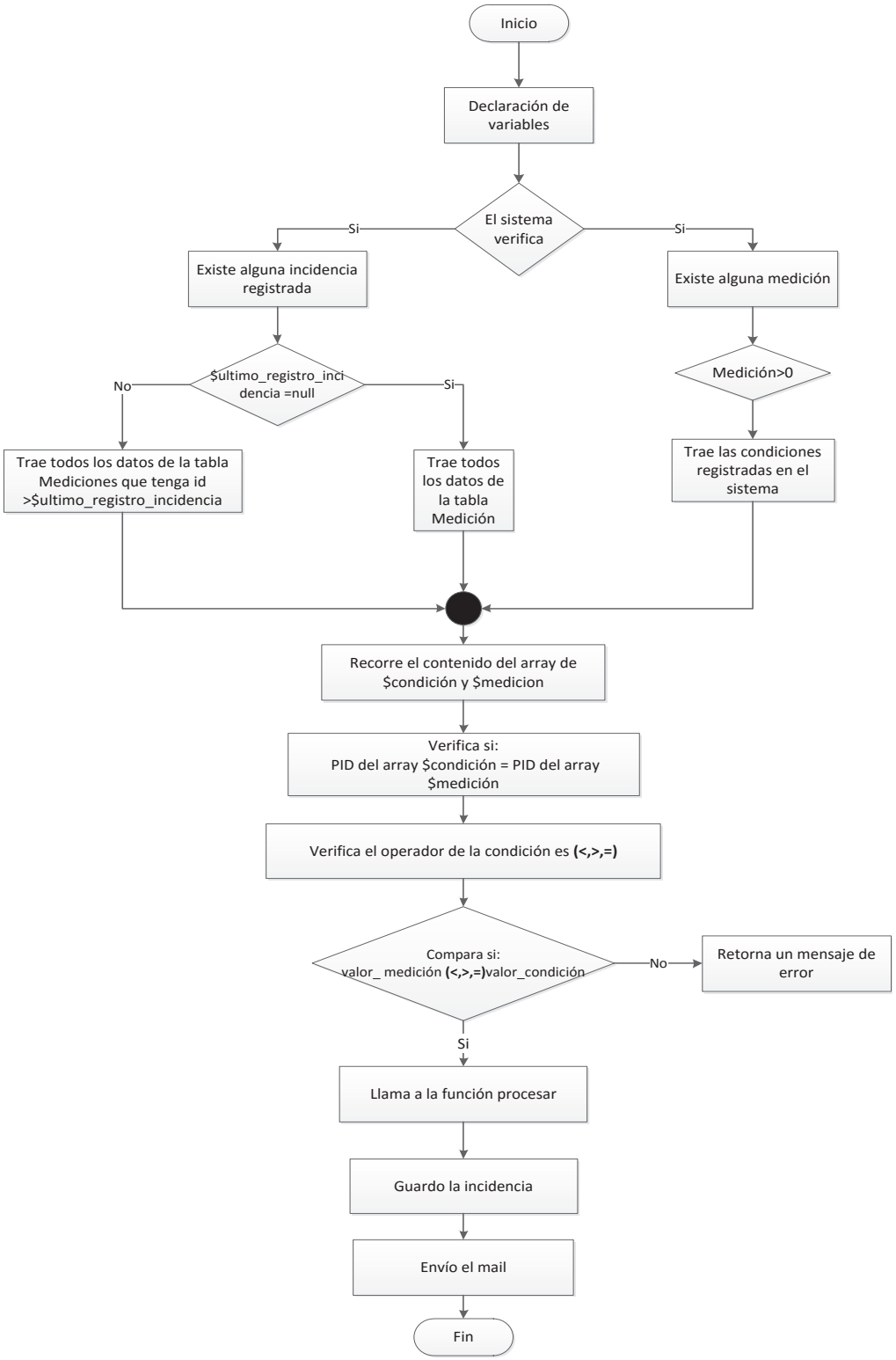

Figura 3.18 Diagrama de flujo para el proceso de generación de incidencias

3.1.5 INTERFACES DEL SISTEMA

A continuación se presentan las interfaces del sistema.

3.1.5.1 Autenticación

Para el ingreso del sistema se diseñó la interfaz de login, donde cada usuario introducirá el nombre de usuario y contraseña. En la Figura 3.19 se presenta un boceto de dicha interfaz.

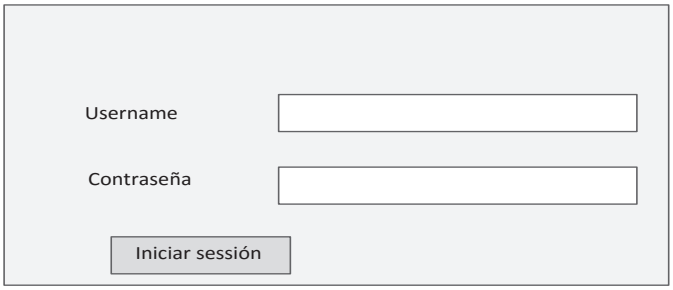

Figura 3.19 Interfaz de login

3.1.5.2 Interfaz general

La Figura 3.20 se presenta la interfaz general para el administrador y helpdesk. En la parte izquierda de la interfaz se presenta un panel vertical dinámico, en el cual están los siguientes módulos de administración: automóviles, dispositivos, usuarios, OBD-II, incidencias, asimismo, se tiene los módulos de configuraciones y el módulo de rastreo GPS.

Figura 3.20 Interfaz general del sistema

3.1.5.3 Interfaz de Administración

Se ha diseñado interfaces web de administración, donde se pueda realizan las funciones de crear, obtener, actualizar y borrar. La interfaz es similar para cada módulo que realiza las actividades administrativas. En la Figura 3.21 se presenta el diseño de la interfaz de administración.

Figura 3.21 Interfaz de administración

Al seleccionar las opciones: registrar y editar se desplegará interfaces secundarias, en la Figura 3.22 se presenta la interfaz de registro de incidencias, todos los módulos que cumplen con esta opción presentan una interfaz similar ya que dependen de los campos que tenga cada interfaz.

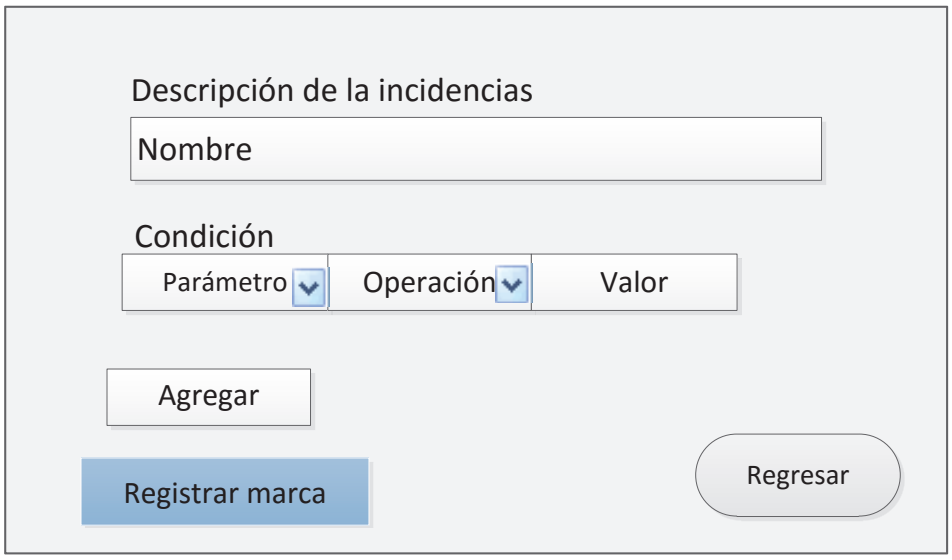

Figura 3.22 Interfaz para el registro de incidencias

En la Figura 3.23 se presenta la interfaz para el proceso de editar una incidencia, el usuario podrá editar todos los campos que se presentan en la interfaz, además al realizar cualquiera de las dos opciones, como registrar y editar, se presentará una alerta de confirmación, para asegurar si es correcta al momento de registrar o editar.

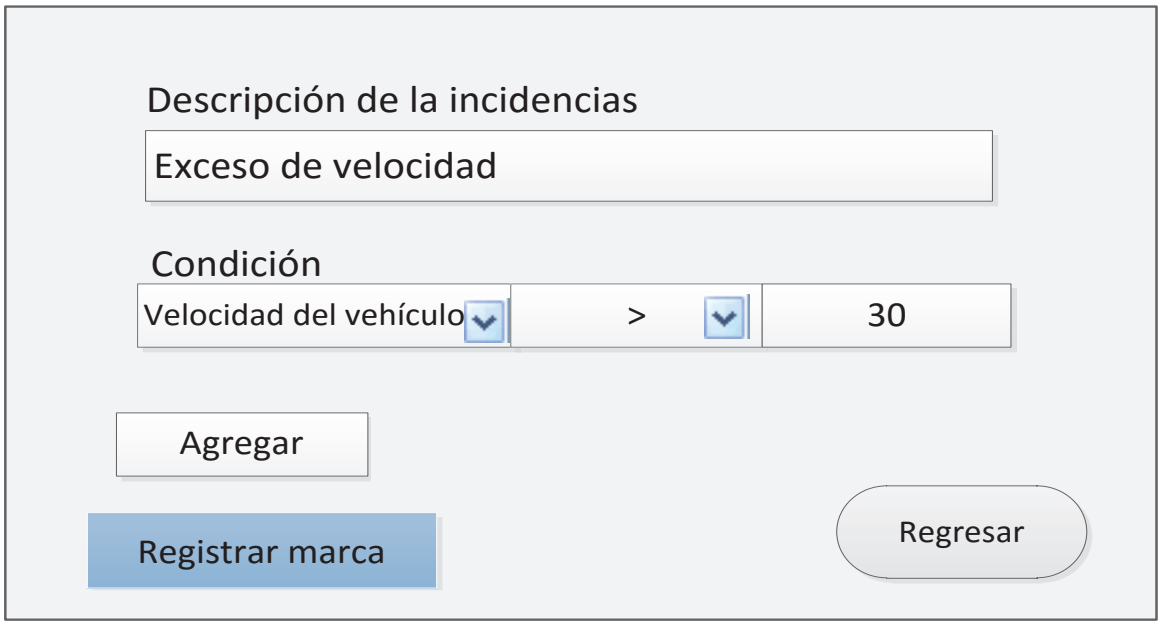

Figura 3.23 Interfaz para editar una incidencia

Para el proceso de eliminar únicamente se despliega un cuadro de dialogo informando si está seguro de eliminar dicho registro.

3.1.5.4 Interfaz de configuración de incidencias

Se ha diseñado una interfaz principal donde se presenta un listado de los nombres de los clientes conjuntamente con los automóviles y sus respectivas características. Asimismo presenta cuatro opciones, donde cada una de estas cumplen ciertas funciones tales como: Registrar una nueva configuración, visualizar todos los parámetros con sus respectivas medidas que han provocado una incidencia, y la opción de revisar la última configuración y un historial de las configuraciones realizadas.

En la Figura 3.24 se presenta la interfaz general donde se presentan dichas opciones.

Figura 3.24 Interfaz general de la configuración de incidencias

- Interfaz nueva configuración:

Es la interfaz principal, debido a que se realiza el proceso de configuración de incidencias, para ello se ha diseñado 2 interfaces secundarias que realizan lo siguiente:

1. La primera interfaz permitirá al usuario seleccionar las incidencias, además se podrá observar la información del propietario y características del automóvil. En la Figura 3.25 se presenta lo antes mencionado.

Figura 3.25 Interfaz para el registro de incidencias

2. La segunda interfaz presentará un resumen, y permitirá guardar la incidencia seleccionada, como se indica en la Figura 3.26.

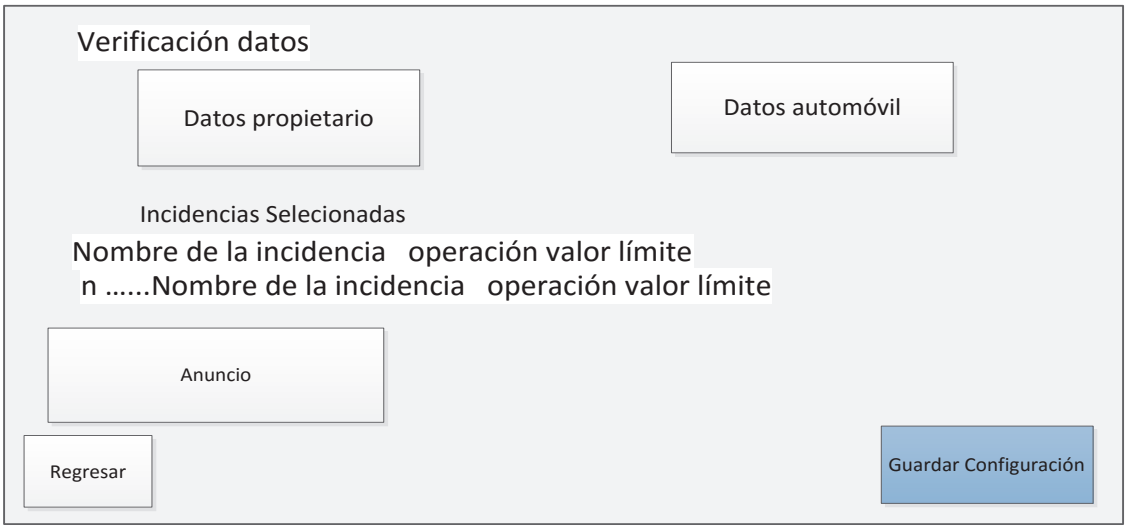

Figura 3.26 Interfaz resumen de la configuración de incidencia

- Interfaz última configuración:

La opción última configuración, presenta un resumen de la última configuración realizada como se observa en la Figura 3.27.

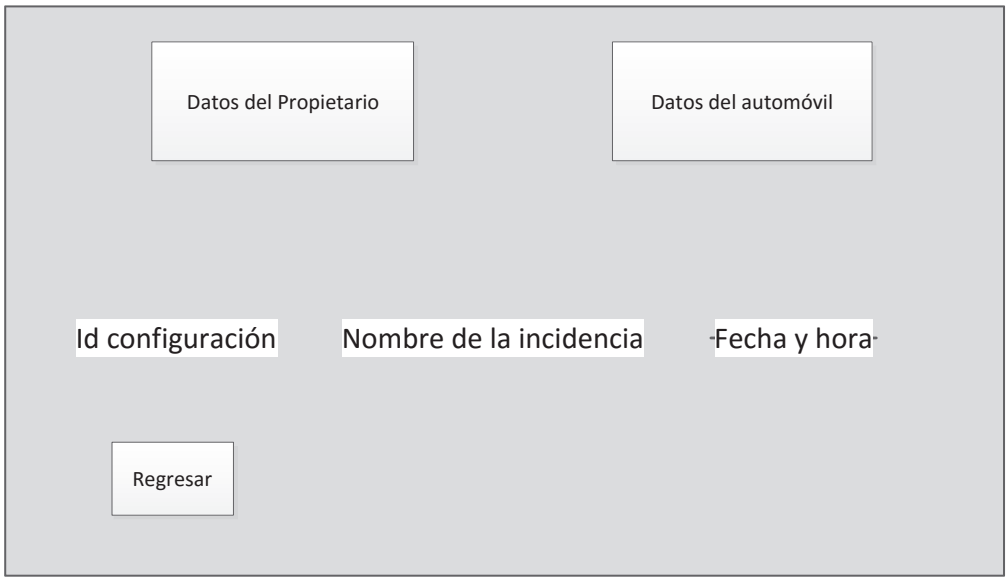

Figura 3.27 Interfaz de la última configuración

Al seleccionar la fecha y la hora se desplegará un resumen con el nombre de la incidencia y parámetro conjuntamente con el valor límite registrado como se indica en la Figura 3.28.

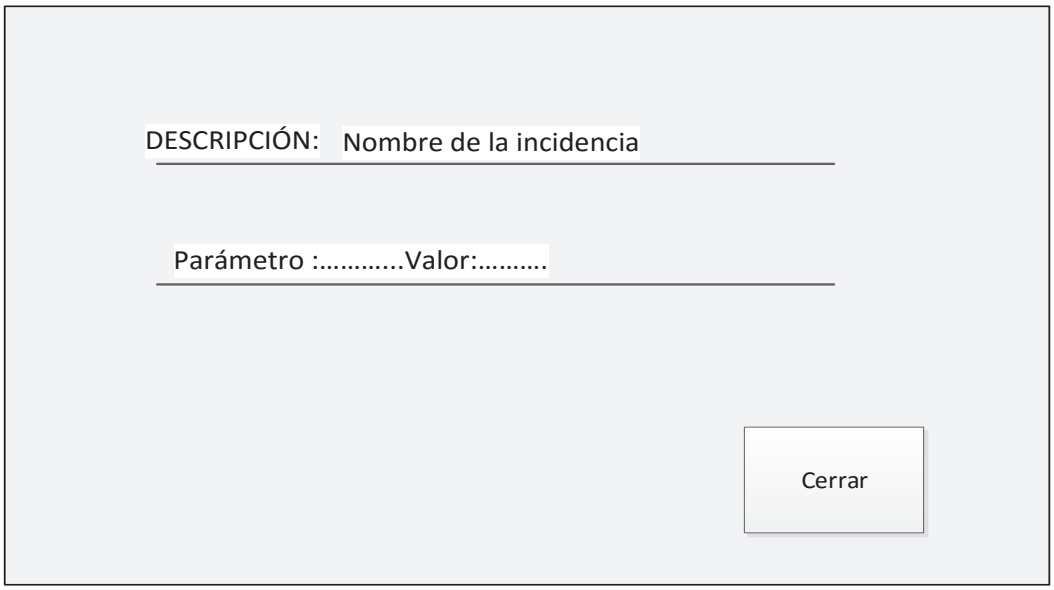

Figura 3.28 Interfaz resumen de la última configuración de incidencias

- Interfaz historial de configuración

La opción historial de configuración, presenta una lista de las fecha de las configuraciones realizadas de forma descendente como se indica en la Figura 3.29.

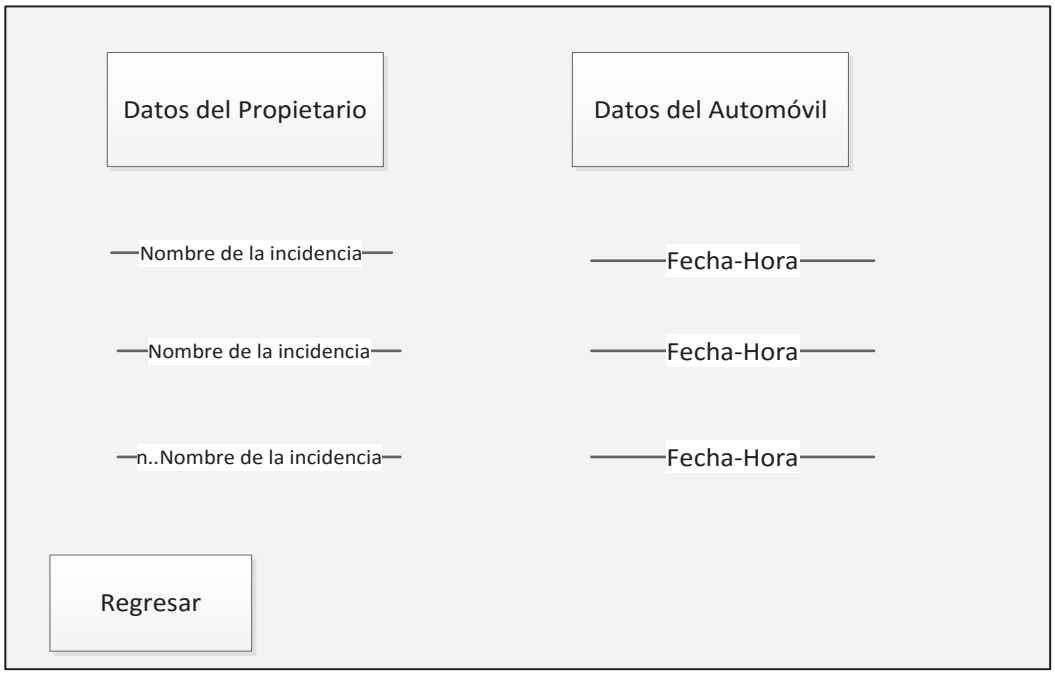

Figura 3.29 Interfaz del historial de las configuraciones de incidencia

Al seleccionar una de ellas, se despliega una interfaz informativa de la configuración realizada en dicha fecha, como se indica en la Figura 3.30.

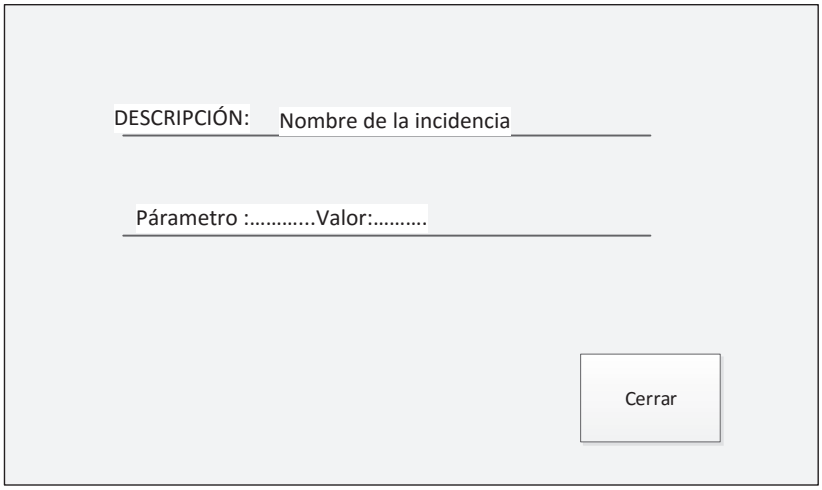

Figura 3.30 Resumen de la configuración seleccionada

- Interfaz ver historial

En esta interfaz se podrá visualizar "n" nombres de las incidencias con sus respectivas medidas conjuntamente con la hora y fecha suscitadas, esto únicamente se podrá visualizar una vez que se genere el proceso de detección de incidencia. Adicionalmente se presentará datos del propietario y automóvil como se indica en Figura 3.31.

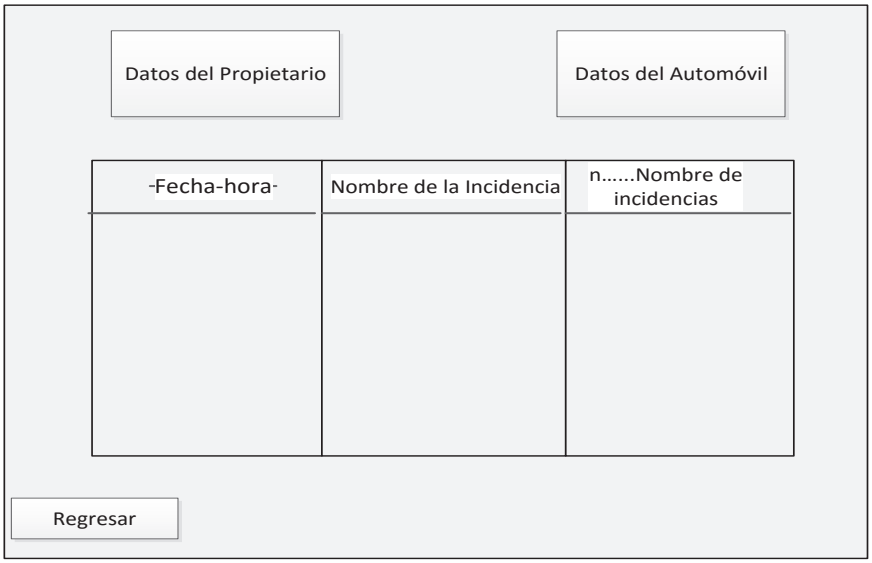

Figura 3.31 Interfaz ver historial de incidencias

- La interfaz reportes estadísticos.

El usuario al seleccionar el nombre de la incidencia se le desplegará los checkbox en la parte superior de cada incidencia, el usuario podrá elegir una o más incidencias

como se indica en la Figura 3.32. Además estos gráficos se podrán observar dependiendo de la fecha que se seleccione.

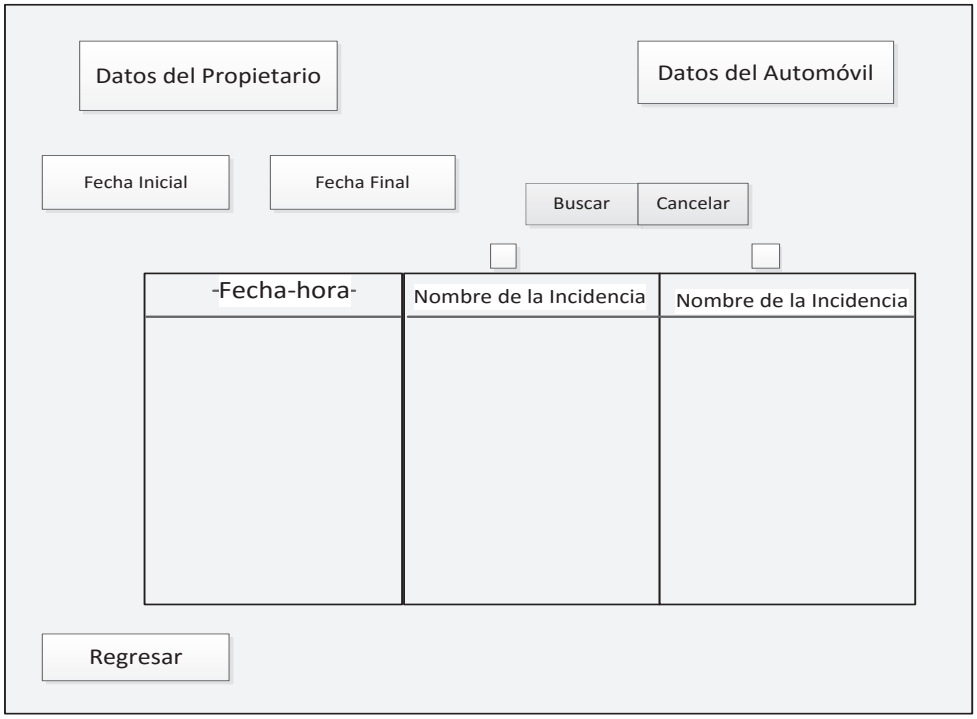

Figura 3.32 Interfaz de selección de incidencias para generar el grafico estadístico

Luego se deberá seleccionar el evento buscar para visualizar el gráfico estadístico, dependiendo del número de incidencias seleccionadas se dibujarán los valores y se distinguirán de las demás por un color que se indicará en la parte inferior a que incidencia pertenece, como se observa en la Figura 3.33.

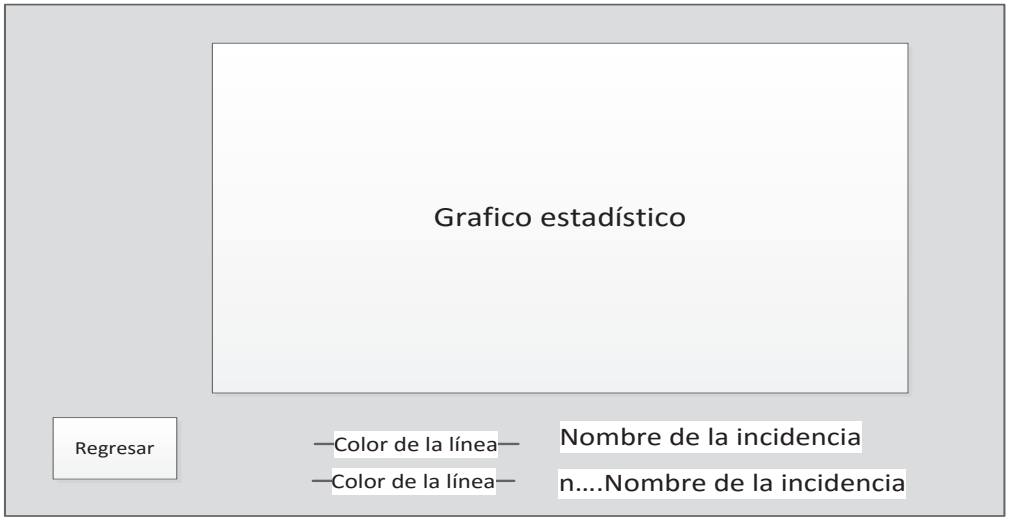

Figura 3.33 Interfaz del gráfico estadístico

3.1.5.5 Interfaz de configuración remota

Se ha diseñado la interfaz general donde se presenta cuatro opciones relacionadas con el proceso de configuración remota, en la Figura 3.34 se presenta cada una de las opciones.

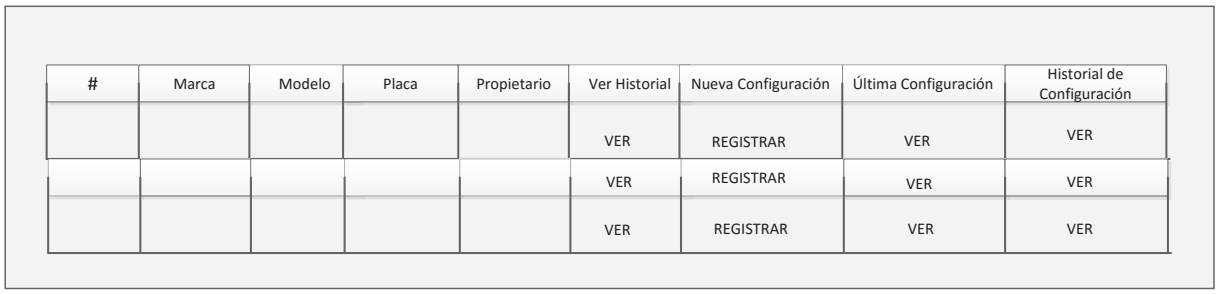

Figura 3.34 Interfaz general

- Interfaz nueva configuración

En esta interfaz se realiza el proceso de la configuración remota, se detallará a continuación los pasos a seguir.

1. Se presentará una interfaz donde el usuario seleccionará la tarjeta SIM y el dispositivo Arduino, como se indica en la Figura 3.35.

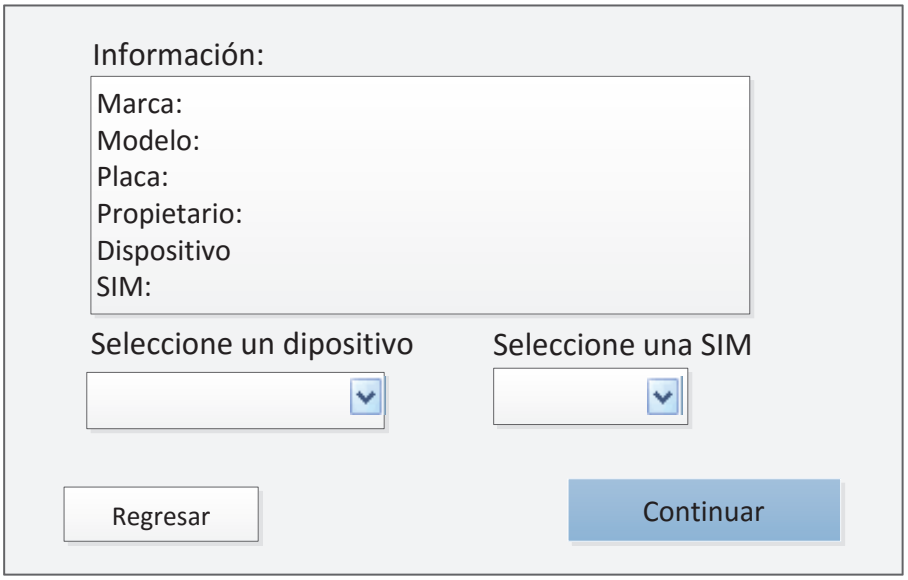

Figura 3.35 Interfaz para la selección de la tarjeta SIM y el Arduino

2. Luego se presentará una segunda interfaz donde se deberá seleccionar el modo de operación del sistema OBD-II, luego se desplegará los parámetros pertenecientes a dicho modo, asimismo, se debe seleccionar el tiempo de

muestreo. Adicionalmente se presentará información del propietario, el automóvil y del dispositivo como se presenta en la Figura 3.36.

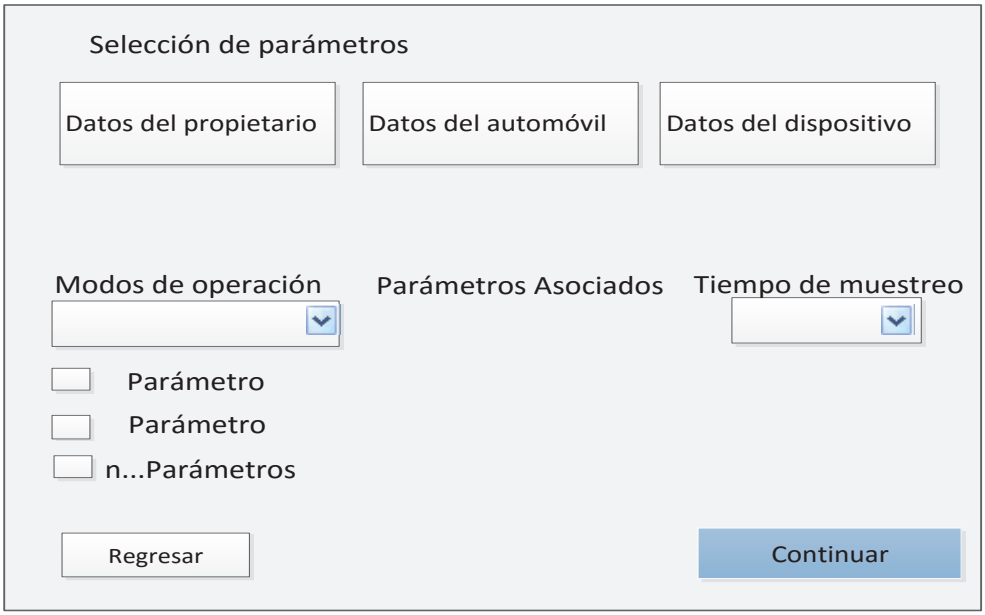

Figura 3.36 Interfaz para seleccionar los parámetros y el tiempo de muestreo

3. Finalmente se presentará una tercera interfaz la cual contiene un resumen de la información seleccionada como se indica en la Figura 3.37.

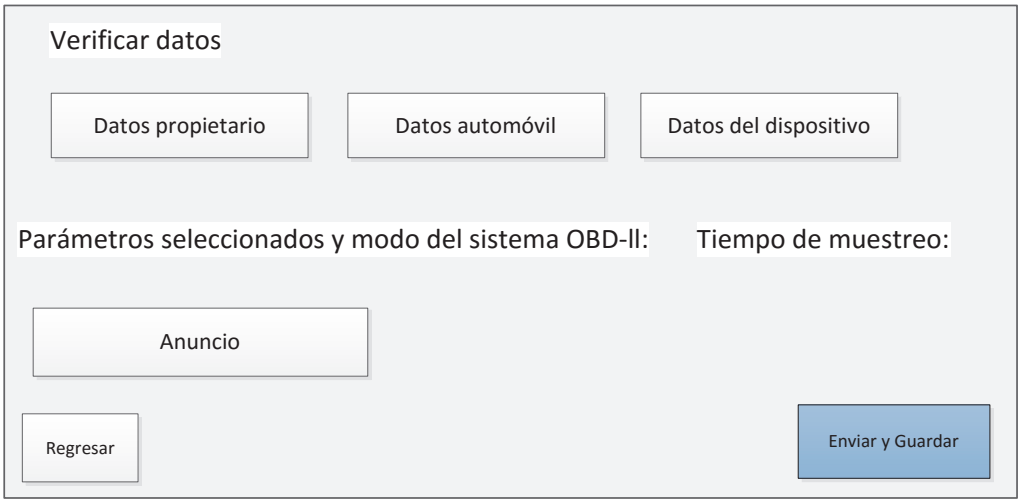

Figura 3.37 Interfaz de resumen

- Interfaz última configuración

La interfaz presentará un resumen de la última configuración realizada, esta contendrá la siguiente información: datos del propietario, datos del automóvil, la fecha y la hora en que se realizado la configuración, datos de la tarjeta SIM, datos

de modo de operación OBD-II conjuntamente con los parámetros asociados y el tiempo de muestreo como se indica en la Figura 3.38.

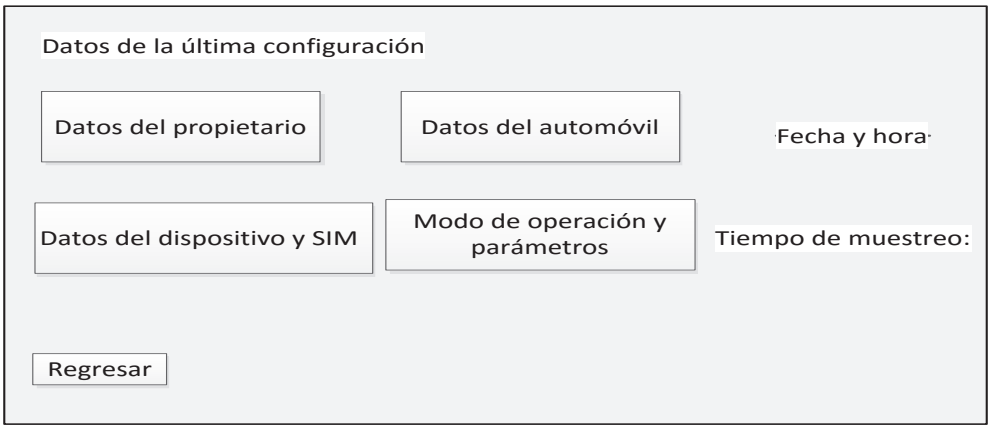

Figura 3.38 Interfaz de la última configuración remota

- Interfaz historial de configuración

La interfaz presentará un listado de la misma placa conjuntamente con la marca, el modelo y el color del automóvil, la hora y fecha en la que se realizó la configuración. Las fechas se presentan de forma descendente como se indica en la Figura 3.39.

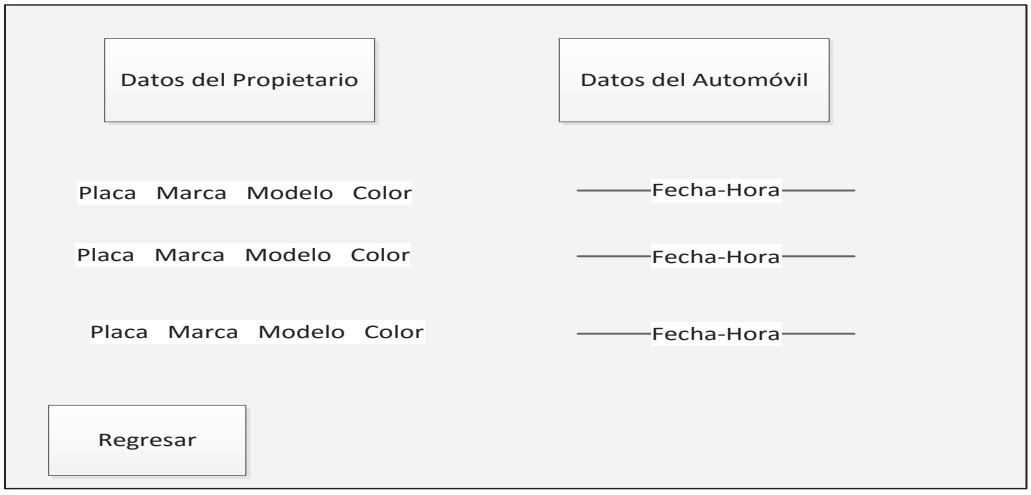

Figura 3.39 Interfaz del historial de las configuraciones remotas

El usuario al seleccionar la fecha y hora se desplegará un resumen con la siguiente información: información del automóvil, información del dispositivo, información de la tarjeta SIM el tiempo de muestreo y los parámetros seleccionados como se presenta en la Figura 3.40.

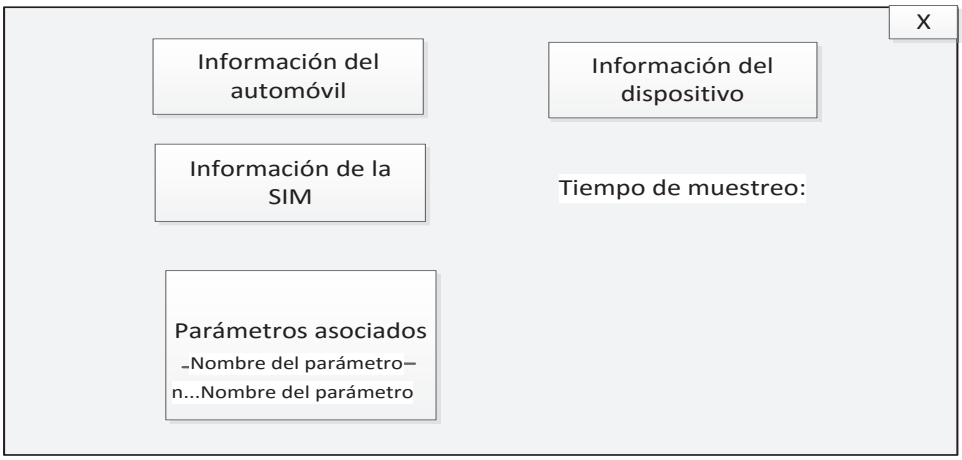

Figura 3.40 Interfaz de resumen dela configuración remota

- Interfaz ver mediciones

En esta interfaz se presentará “n” medidas enviadas por el módulo DISPOSITIVO (5), conjuntamente con la hora y fecha en las que se realizó la captura de las mismas, adicionalmente se presenta información del propietario y del automóvil como se indica en la Figura 3.41.

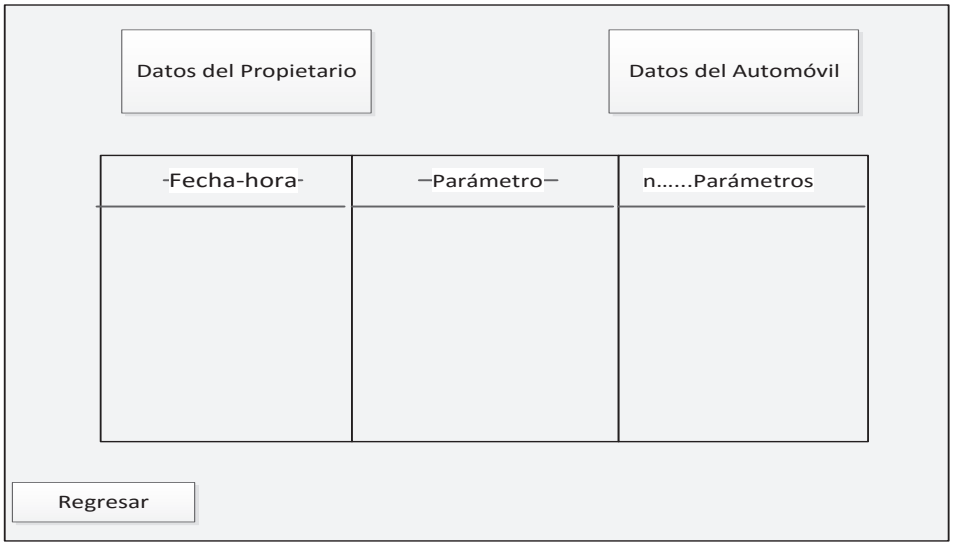

Figura 3.41 Interfaz de visualización de históricos de las medidas

- Interfaz reportes estadísticos

En la misma interfaz ver mediciones el usuario al seleccionar uno de los parámetros y se desplegará los checkbox en la parte superior de cada parámetro, el usuario podrá escoger uno o más parámetros para ser presentados en una gráfico estadístico como se indica en la Figura 3.42.

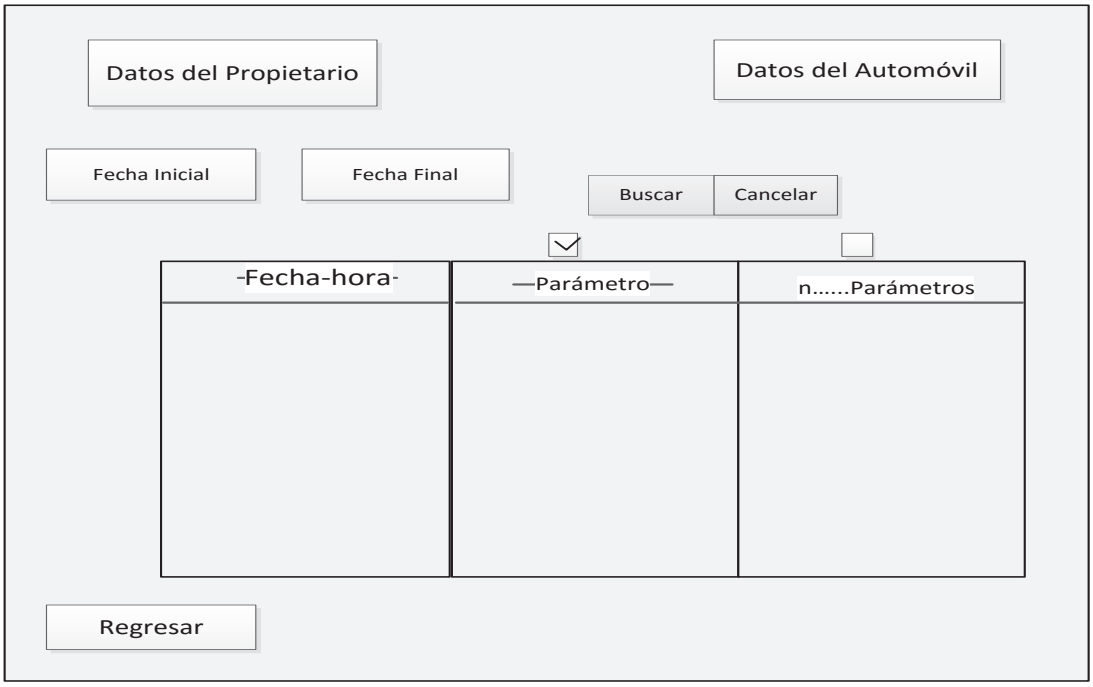

Figura 3.42 Interfaz de selección de parámetros para generar el gráfico estadístico

Los parámetros serán diferenciados por colores como se indica en la Figura 3.43.

Figura 3.43 Interfaz del grafico estadístico

3.1.5.6 Interfaz de rastreo GPS

En esta interfaz se podrá visualizar el historial de las marcas de posición del automóvil a través de un mapa como se indica en la Figura 3.44.

En la interfaz de la posición actual se podrá observar la posición del automóvil a través en un mapa y el estado del mismo en ese instante, como se indica en la Figura 3.45.

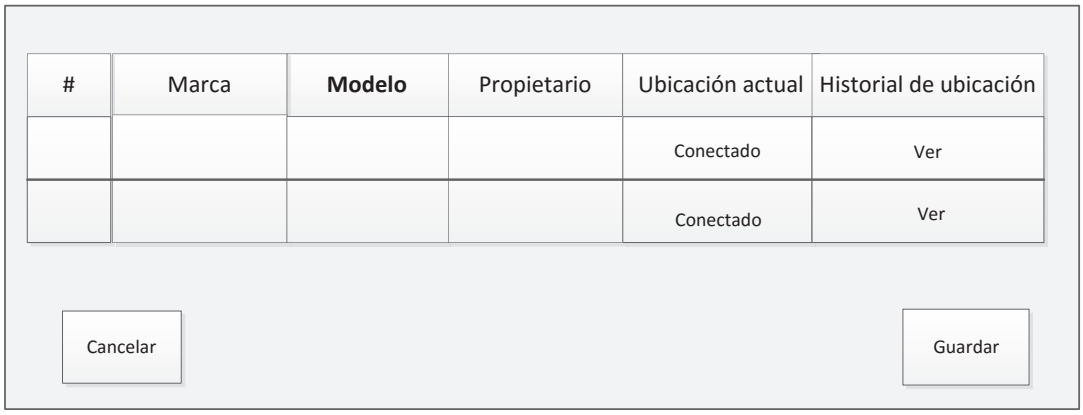

Figura 3.44 Interfaz general del rastreo GPS

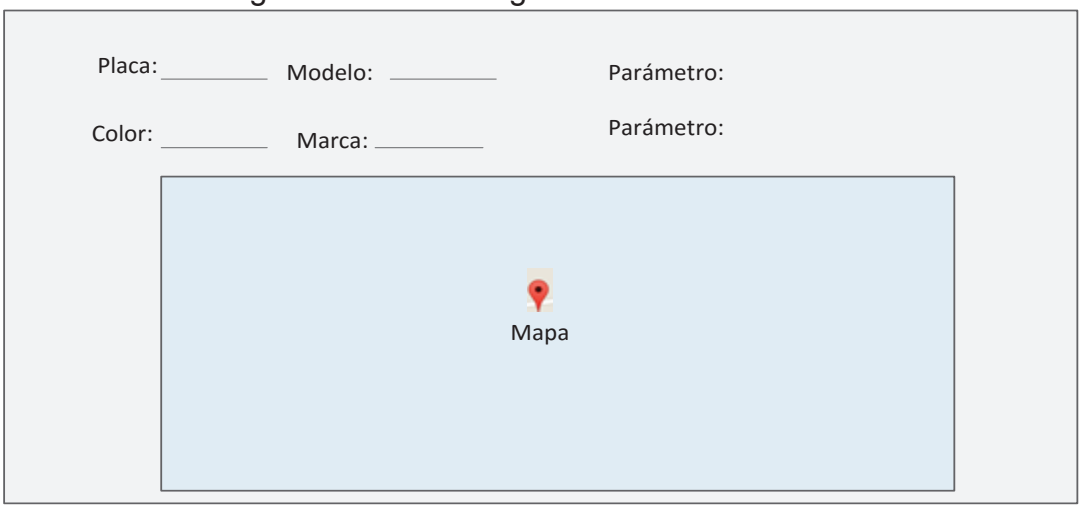

Figura 3.45 Interfaz de la ubicación actual del automóvil

En la interfaz del historial de la posición se presentará el recorrido realizado por el automóvil, se podrá verificar el recorrido por fechas, como se indica en la Figura 3.46.

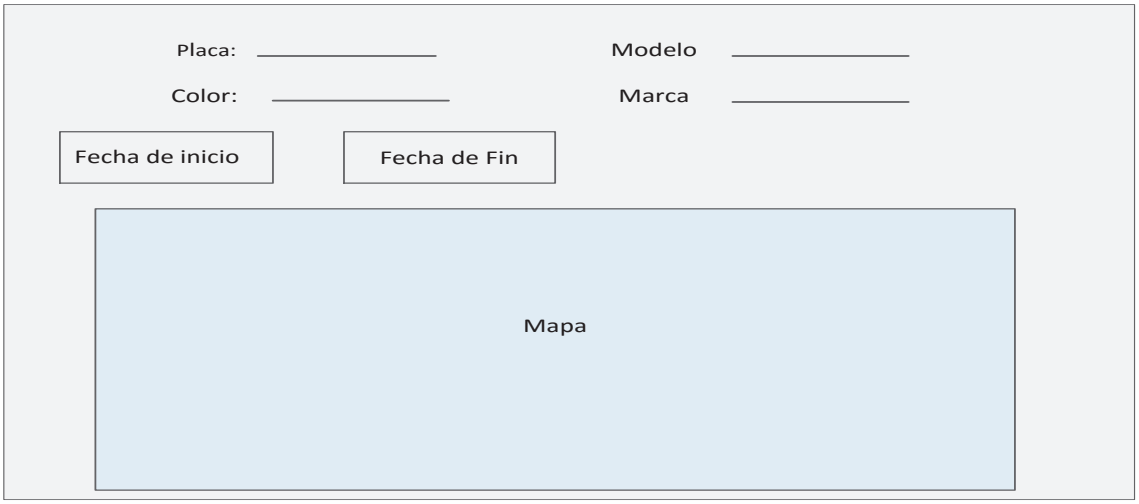

Figura 3.46 Interfaz historial del automóvil

3.1.5.7 Interfaz de cliente

La interfaz del cliente es sencilla y amigable. Las opciones que se presentan al cliente son: Incidencias registradas, medidas capturadas y Geo-posicionamiento. En la Figura 3.47 se observa la interfaz del cliente.

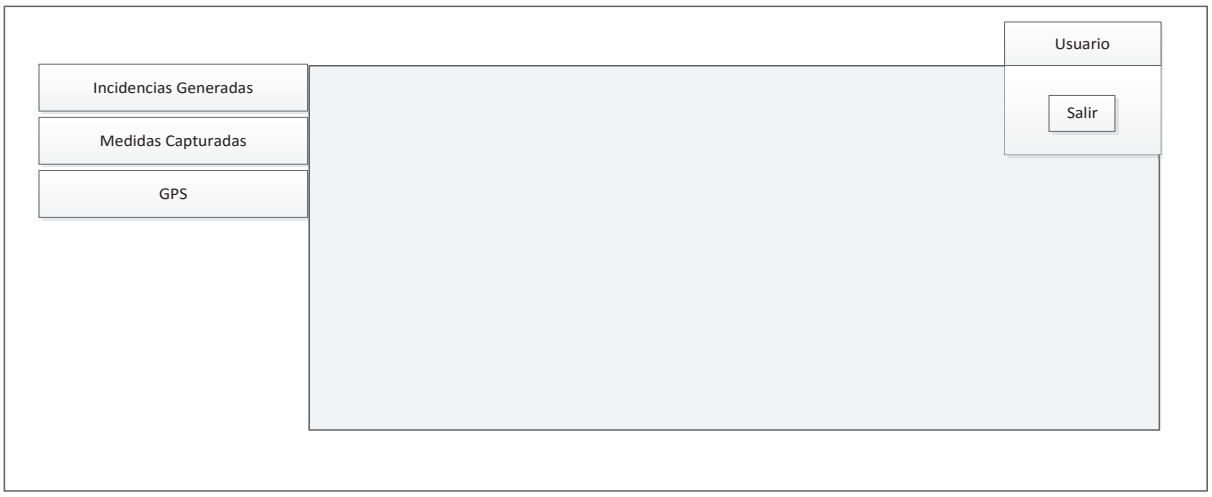

Figura 3.47 Interfaz general del cliente

3.2 IMPLEMENTACIÓN

En esta sección se presenta el proceso de implementación del sistema propuesto.

3.2.1 TECNOLOGÍAS SELECCIONADAS

Para el desarrollo del sistema se optó por las siguientes tecnologías.

Lenguajes de programación

El lenguaje seleccionado para el sistema es PHP, debido a su independencia de plataformas y la gran variedad de información sobre su uso ya que es un lenguaje usado para realizar aplicaciones Web. Además el framework seleccionado trabaja con PHP.

Java script es un lenguaje del lado del cliente que permite desarrollar interfaces dinámicas, con diferentes efectos y que facilitan la interacción con el usuario. CSS, son hojas de estilo que ayudan a definir un formato para las interfaces web, en este proyecto no se manejó de forma directa las hojas de estilo debido a que CSS está implícito en el framework Bootstrap.

Ajax

Ayudo a realizar hacer peticiones de datos al servidor y recibirlas sin necesidad de cargar nuevamente toda la interfaz.

Framework Larevel

Para el desarrollo del proyecto se usó el framework Laravel debido a su simplicidad, flexibilidad y robustez[9]. Además, tiene las siguientes características:

- Ayuda a reducir el tiempo en el desarrollo y mantenimiento del proyecto.
- Provee una variedad de documentación y esto ayuda aprender de una forma rápida las utilidades y funcionalidades de Laravel.
- Trabaja con el patrón MVC, que ayuda a ordenar el código, separando las tareas de la aplicación en modelo, vista y controlador, esto hace que el sistema sea fácil de entender comprender y administrar.
- Laravel realiza un manejo fácil de los datos mediante Eloquent, debido a que la interacción con las bases de datos es totalmente orientada a objetos, siendo compatible con la gran mayoría de las bases de datos del mercado actual y facilitando la migración de los datos de una forma fácil y segura. Por otra parte permite la creación de consultas robustas y complejas.
- Ofrece un sistema de plantillas Blade de Laravel, que provee mejoras en la parte de presentación de la aplicación como la generación de plantillas más simples y limpias en el código.
- También cuenta con una herramienta de interfaces de líneas de comando llamada Artisan que permite programar tareas programadas como por ejemplo ejecutar migraciones, creación de controladores, verificación de ruta, etc.

La instalación de Laravel se presenta en el ANEXO 1.

Bootstrap

Para este proyecto se utilizará el framework Bootstrap, para darle un estilo personalizado al sistema web de manera sencilla, debido a que el desarrollador no

[9] https://platzi.com/blog/laravel-framework-php/

tiene que agregar líneas de código. Existe una gran variedad de plantillas Bootstrap, se los puede encontrar en [24]. Bootstrap está formado por varios archivos, los cuales se colocarán en las siguientes carpetas public/css y public/js respectivamente. Además este framework fue seleccionado ya que posee la característica de adaptarse a cualquier dispositivo móvil sin perder el diseño y su integridad en pantallas pequeñas.

Gestor de base de datos

Para el almacenamiento de la información se ha seleccionado el sistema de gestión de base de datos MySQL, debido a que tiene una fácil integración con el lenguaje de programación PHP [25]. La creación de las tablas está en el ANEXO 2.

Librería

Para conseguir las marcas de posicionamiento se trabajó con la librería dGPS.h, creada por Dexter industrias bajo código abierto. Dicha librería contiene funciones que permiten obtener: longitud, latitud, fecha y hora enviadas por el satélite. La librería se presenta en el ANEXO 3.

3.2.2 IMPLEMENTACIÓN DE LARAVEL CON MVC

Laravel propone una estructura de carpetas, para la organización del código. En la Figura 3.48 se presentan las carpetas principales del framework, las cuales se detallan a continuación.

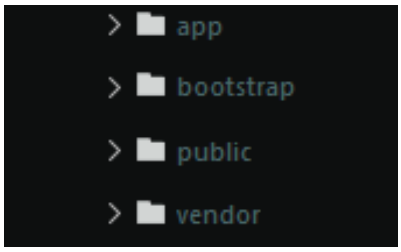

Figura 3.48 Estructura Laravel

/app

App es el directorio raíz, contiene el patrón Model-View-Controllers y los archivos de configuración de la aplicación. Además en esta carpeta esta la mayor parte del código de la aplicación. Cabe mencionar que dentro de esta carpeta se encuentra el

archivo “router.php”, el cual contiene las rutas de la aplicación[10]. En Figura 3.49 se observa las carpetas que tiene el directorio app.

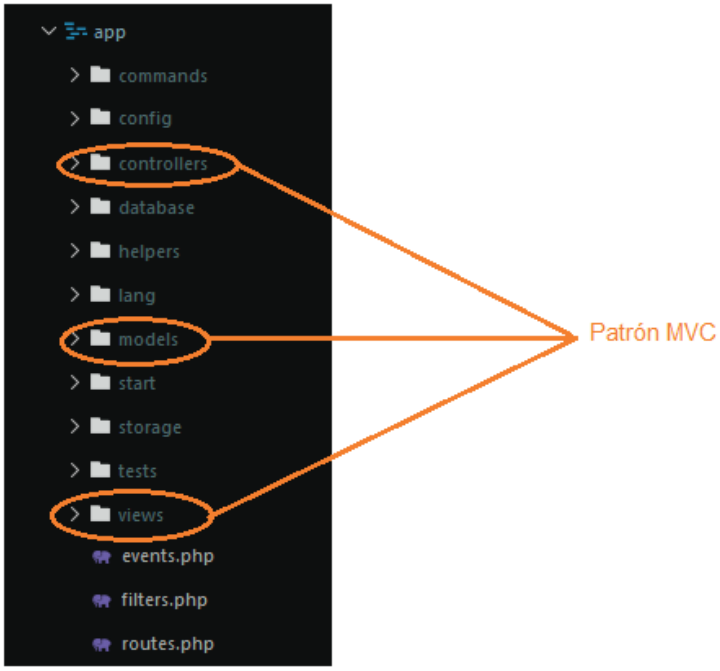

Figura 3.49 Directorio app

Dentro del directorio app, se tiene otro directorio de importancia llamado config.

/app/config

En este directorio existe los archivos de configuración de la aplicación, los archivos que se usaron para el proyecto son:

Database.php, se configura el motor de la base de datos.

Mail.php, donde se configura el servidor SMTP[11] para el envío de correos.

/public

Es la única carpeta que puede acceder el usuario. Además se alojan los archivos CSS, Javascript, imágenes y todos los archivos que se harán públicos. En la Figura 3.50 se observa lo que contiene la carpeta public.

[10] El sistema de rutas implementado por Laravel, define los links de acceso para la aplicación, hasta los métodos a los cuales se puede utilizar. si no se declaran las rutas, muy difícilmente laravel podrá acceder a ella.
[11] Simple Mail Transfer Protocol (SMTP): Es el protocolo estándar de Internet para el intercambio de correo electrónico

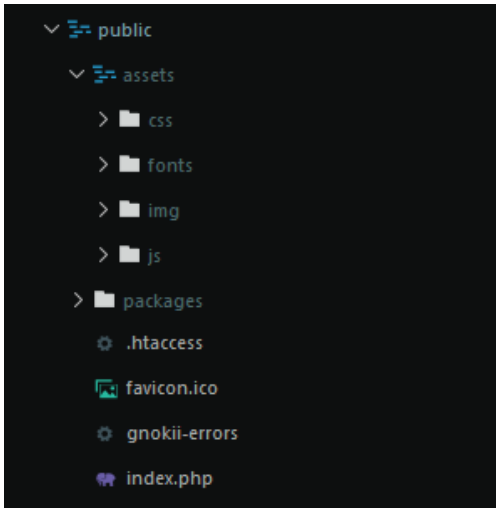

Figura 3.50 Carpeta public

/vendor

En este directorio se alojan las librerías y las dependencias del framework.

3.2.3 IMPLEMENTACIÓN DE MÓDULOS RELEVANTES

3.2.3.1 Conexión con la base de datos

Para la conexión con la base de datos únicamente se necesita editar el archivo de configuración de Laravel “/app/config/database.php”. Como se indica en la Figura 3.51 únicamente se coloca en el campo de la conexión ‘mysql’ el nombre de la base, el usuario y la contraseña.

```
'mysql' => array(
    'driver'    => 'mysql',
    'host'      => 'localhost',
    'database'  => 'tesis_eli',
    'username'  => 'root',
    'password'  => 'admin2014',
    'charset'   => 'utf8',
    'collation' => 'utf8_unicode_ci',
    'prefix'    => '',
),
```

Figura 3.51 Conexión base de datos

3.2.3.2 Implementación del módulo administración de automóviles

El sistema presenta varios módulos de administración que fueron detallados en la sección 3.1.1, estos módulos cumplen con las funciones CRUD para los diferentes

datos pertenecientes a cada módulo. El proceso de administración es similar para los demás módulos.

A continuación se presenta la implementación del módulo administración de automóviles. Este módulo ofrece al usuario una interfaz para que realice las siguientes acciones:

- Ver el listado de automóviles registrados en el sistema.
- Agregar un nuevo automóvil al sistema.
- Modificar los datos de automóvil.
- Eliminar el automóvil del sistema.

La tabla en la cual se guardan toda la información que corresponde a los automóviles y que se va alterar durante el proceso de administración es la tabla automóvil.La clase que permite representar la entidad automóvil es clase Automóvil.

Al implementar MVC para brindar las funcionalidades CRUD se requirió de las siguientes vistas: créate, edit y show

De igual manera se requirió del siguiente controlador: AutomovilController.

A continuación se presenta en la Figura 3.52 el funcionamiento de MVC aplicado al módulo de administración de automóviles.

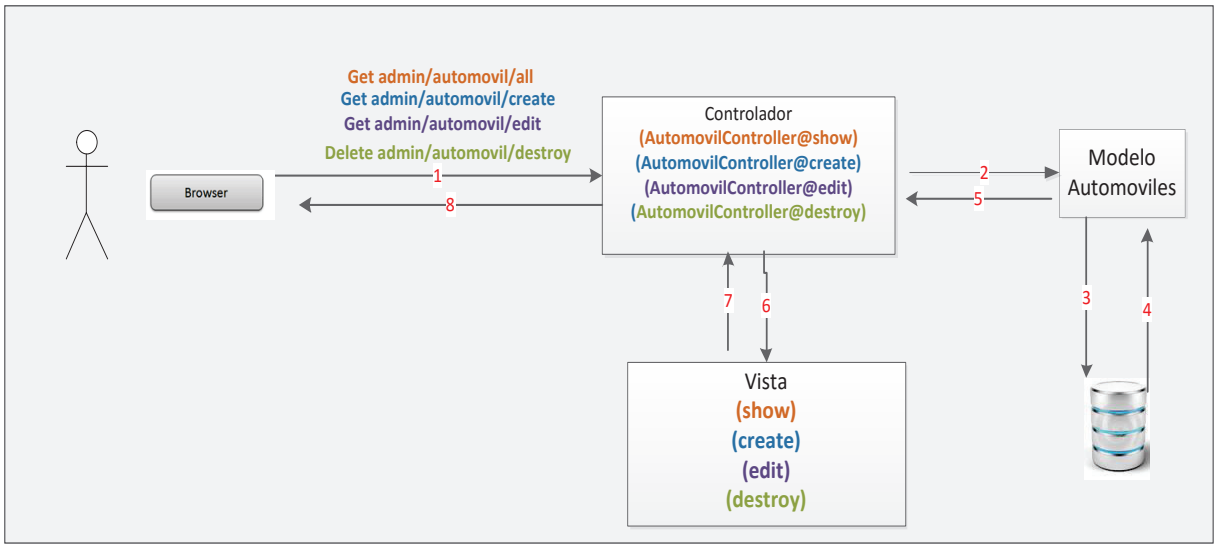

Figura 3.52 MCV para administración de automóviles

Modelo

Se encarga de gestionar, procesar y validar los datos. Además se encarga de las relaciones existes con otras clases establecidas en la Figura 3.8. A continuación se detallan los métodos utilizados para este módulo.

- **validarDatos:** sirve para validar los datos que ingresa el usuario, además especifica si son datos requeridos, si son únicos y el tipo de dato.

Dentro del modelo se representa las relaciones de las tablas, esto ayuda a realizar las operaciones de consulta a la base de datos como se presenta en la Figura 3.53.

```
class Automovil extends \Eloquent {

    protected $table = 'automoviles';
    protected $fillable = array('id_cliente', 'id_modelo',
                                'color', 'placa', 'motor',
                                'anio', 'numero_chasis',);

    /*Validación*/
    public function validarDatos($datos){

        $validar = Validator::make($datos, array(
            'id_cliente'=>'required',
            'id_modelo'=>'required',
            'color'=>'required|alpha',
            'placa'=>'required|alpha_num|unique:automoviles',
            'motor'=>'required|alpha_num|unique:automoviles',
            'anio'=>'required|numeric',
            'numero_chasis'=>'required|alpha_num|unique:automoviles'
            )
        );

        if($validar->passes()){
            return true;
        }

        $this->errors = $validar->errors();

        return false;
    }
    /*Relaciones*/
    public function modelo(){
        return $this->hasOne('Modelo');
    // un automóvil tiene muchos modelos
    }
    public function cliente(){
        return $this->belongsTo('Cliente');
    // un automóvil le pertenece a un cliente
    }
    public function registroIncidencia(){
        return $this->hasMany('RegistroIncidencia', 'id_automovil');
    // un automóvil tiene muchos registros de incidencias
    }
    public function configuraciones(){
        return $this->hasMany('Configuracion', 'id_automovil');
    // un automóvil tiene muchas configuraciones
    }
    public function configItemsIncidencias(){
        return $this->hasMany('ConfigItemIncidencia', 'id_automovil');
    // un automóvil tiene muchas item de configuraciones
    }
    public function capturas(){
        return $this->hasMany('Captura', 'id_automovil');
    // un automóvil tiene muchas capturas
```

Figura 3.53 Clase automóvil

Vista

Se ha diseñado 3 vistas que permiten al usuario visualizar la información requerida para cada proceso CRUD.

La vista **show** se ha desarrollado para presentar un listado de automóviles conjuntamente con las acciones registrar, editar y eliminar. La vista se ha diseño como se presenta en la Figura 3.21.

La vista **create** considera la Figura 3.22, está orientada a presentar un listado de campos vacíos para que el usuario ingrese los datos que requiere un nuevo automóvil.

Dentro de esta vista se implementa la siguiente función.

- `consultarModelos`, al seleccionar un marca se desplegará todos los modelos pertenecientes a dicha marca como se indica en la Figura 3.54.

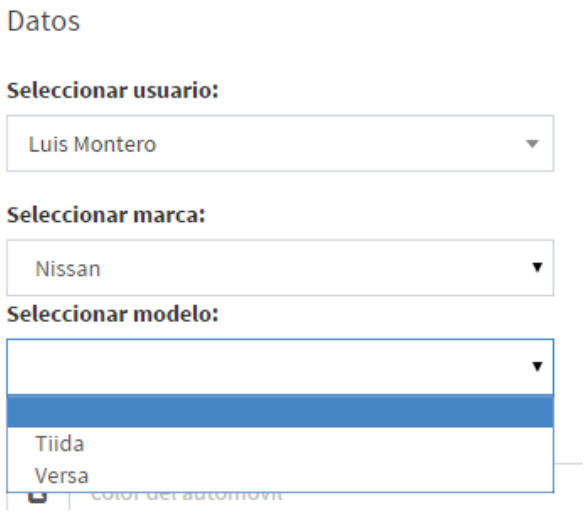

Figura 3.54 Visualización de los modelos pertenecientes a una marca

La vista **edit** considera la Figura 3.23, está orientada para que el usuario realice las modificaciones de los datos del automóvil en caso de ser necesario.

Dentro de esta vista se implementa la siguiente función:

- `consultarMarcas`, al seleccionar un modelo se visualizará de manera instantánea la marca a la cual pertenece dicho modelo como se indica en la Figura 3.55.

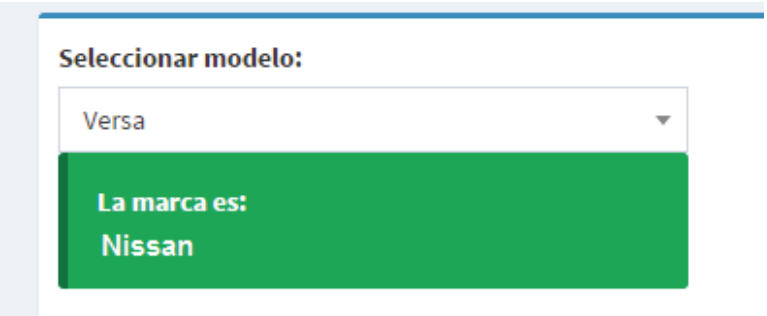

Figura 3.55 Visualización de la marca del modelo seleccionado

Si el usuario realiza el proceso de eliminar se presentará un cuadro de dialogo informando si está seguro de eliminar el registro.

Controlador

El controlador es el intermediario entre la vista y el modelo. Controla las peticiones de los usuarios, es decir, espera las peticiones, comprueba su validez de acuerdo a las normas de autenticación o autorización, y delega la búsqueda de datos al modelo. Finalmente, delega el proceso de presentación a la capa de la vista.

Dentro del controlador se presenta los siguientes métodos:

- **create**, despliega al usuario la vista para el registro de un automóvil, como se indica en la Figura 3.56.

```
public function create()
{
    //
    $clientes = Cliente::where('rol', 'LIKE', 'Cliente')->get();
    $marcas = Marca::all();

    if(count($clientes) == 0){
        return Redirect::to('/admin/tablero')->with('error',
            'Aún no hay clientes registrados. Antes de continuar con el registro
             de automoviles, por favor registre clientes');
    }else if(count($marcas) == 0){
        return Redirect::to('/admin/tablero')->with('error',
            'Aún no hay marcas registradas. Antes de continuar con el registro
            de automoviles, por favor registre marcas');
    }else{
        return View::make('automovil/create')->with('clientes', $clientes)->with('marcas', $marcas);
    }
}
```

Figura 3.56 Método create

- **store**, recupera todos los datos enviados desde la vista `create` y los guarda en la base datos como se indica en la Figura 3.57.

```php
public function store()
{
    //
    //Recuperando todos los datos enviados desde el formulario
    $datosAutomovil = Input::all();

    $automovil = new Automovil();

    //Validando los datos del formulario
    if($automovil->validarDatos($datosAutomovil)){

        //Asignando de forma masiva los datos
        $automovil->fill($datosAutomovil);

        //Guardando los datos
        if($automovil->save()){

            return Redirect::to('admin/automovil/all')->with('mensaje',
                'El automovil ha sido registrada exitosamente');

        }else{
            return Redirect::back()->with('error',
                'No se ha podido guardar los datos, por favor intentelo nuevamente');
        }
    }else{

        return Redirect::back()->withInput()->withErrors($automovil->errors);
    }
}
```

Figura 3.57 Método store

- **show,** trae todos los datos de la tabla automóvil y los presenta en la vista **`show`** como se presente en la Figura 3.58.

```php
public function show($id)
{
    //Trea todos los datos de las tablas automovil,
    // clientes, modelos una vez terminada la consulta join
    if($id == 'all'){
        $automoviles = DB::table('automoviles')
        ->join('clientes', 'clientes.id', '=', 'automoviles.id_cliente')
        ->join('modelos', 'modelos.id', '=', 'automoviles.id_modelo')
        ->join('marcas', 'marcas.id', '=', 'modelos.id_marcas')
        ->get(array('automoviles.id','color', 'anio', 'placa',
            'numero_chasis', 'motor', 'id_cliente', 'id_modelo',
            'nombres', 'apellidos','username', 'email', 'nombre_modelo',
            'id_marcas', 'nombre_marca'));

        return View::make('automovil/show')->with('automoviles', $automoviles);
    }else{
        $id_cliente = explode("=", $id);
        $datos = DB::table('automoviles')
        ->join('clientes', 'clientes.id', '=', 'automoviles.id_cliente')
        ->join('modelos', 'modelos.id', '=', 'automoviles.id_modelo')
        ->join('marcas', 'marcas.id', '=', 'modelos.id_marcas')
        ->where('id_cliente', '=', $id_cliente[1])
        ->get(array('automoviles.id','color', 'anio', 'placa',
            'numero_chasis', 'motor', 'id_cliente', 'id_modelo',
            'nombres', 'apellidos','username', 'email', 'nombre_modelo',
            'id_marcas', 'nombre_marca'));

        return View::make('automovil/show')->with('datos', $datos);
    }
}
```

Figura 3.58 Método show

- **edit,** entrega al usuario la vista para editar un automóvil como se presenta en la Figura 3.59.

```
public function edit($id)
{
    //llama a la vsita edit conn los datos del automovil a editar
    $automovil = Automovil::find($id);
    $modelo = Modelo::find($automovil->id_modelo);
    $modelos = Modelo::all();

    return View::make('automovil/edit')->with('automovil', $automovil)
                                       ->with('modelos', $modelos)
                                       ->with('modelo', $modelo);
}
```

Figura 3.59 Método edit

- **update**, se utiliza para actualizar los datos al momento de presionar el botón actualizar registro como se presenta en la Figura 3.60.

```
public function update($id)
{
    //Guardo los datos actualizados en la base de datos
    $automovil = Automovil::find($id);
    $datos = Input::all();

    $automovil->fill($datos);

    if($automovil->save())
    {
        return Redirect::to('admin/automovil/all')->with('mensaje', 'Automovil actualizado con éxito.');
    }else{
        return Redirect::back()->with('error', 'Algo salio mal, por favor intentelo nuevamente.');
    }
}
```

Figura 3.60 Método update

- **destroy**, elimina el registro de la base de datos, en la Figura 3.61 se presenta el código del método.

Un proceso similar se realizó para la implementación de los siguientes módulos:

- Módulo administración de Usuarios
- Módulo administración de dispositivos
- Módulo administración del sistema OBD-II
- Módulo administración de incidencias

El código implementado se puede observar en el ANEXO 4.

```php
public function destroy($id)
{
    //busca en la tabla el configuraciones el id del automovil y si existe ya una configuración
    //el sistema notificará qeu no se podrá borrar dicho auto, en caso contrario se borrara el
    // automovil de la tabla automovil.
    $parametros = explode("=", $id);

    if($parametros[0] == 'confirmar'){

        $configuracion = DB::table('configuraciones')->where('id_automovil', '=', $parametros[1])->get();

        if(count($configuracion) > 0){
            return Redirect::back()
            ->with('url', '/admin/automovil/'.$parametros[1])
            ->with('advertencia', 'El automóvil ya tiene registrada configuraciones, no se puede eliminar');
        }else{
            Automovil::find($parametros[1])->delete();
            return Redirect::back()->with('mensaje', 'El registro del automóvil se ha borrado exitosamente!');
        }
    }else{

        //Borrando las datos asociadas al automóvil
        Captura::where('id_automoviles', '=', $id)->delete();
        $id_configuracion = Configuracion::where('id_automovil', '=', $id)->get(array('id'));

        foreach($id_configuracion as $id_config){
            ItemConfiguracion::where('id_configuracion', '=', $id_config['id'])->delete();
        }
        Configuracion::where('id_automovil', '=', $id)->delete();
        ConfigItemIncidencia::where('id_automovil', '=', $id)->delete();
        RegistroIncidencia::where('id_automovil', '=', $id)->delete();
        Automovil::find($id)->delete();
        return Redirect::back()->with('mensaje', 'El registro del automóvil se ha borrado exitosamente!');
    }
}
```

Figura 3.61 Método destroy

3.2.3.3 Implementación del módulo configuración remota

Este módulo permite al usuario seleccionar un conjunto de parámetros del sistema ODB-II que se desea obtener del motor. El módulo ofrece una interfaz para realizar las siguientes funciones.

- Seleccionar una tarjeta SIM y un dispositivo Arduino los cuales deben estar registrados previamente en el sistema Web.
- Seleccionar el modo del sistema OBD-II, y los parámetros asociados a dicho modo y el tiempo de muestreo.

Las tablas en la cual se guardan toda la información que corresponde a la configuración remota son tablas de configuraciones y tabla ItemConfiguracion.

La clase que permite representar la entidad configuraciones es la clase Configuración.

Al implementar MVC para el proceso de configuración remota se requirió de la siguiente vista:

- Show

De igual manera se requirió del siguiente controlador: ConfiguracionController.

A continuación se presenta en la Figura 3.62 el funcionamiento de MVC aplicado al módulo configuración remota.

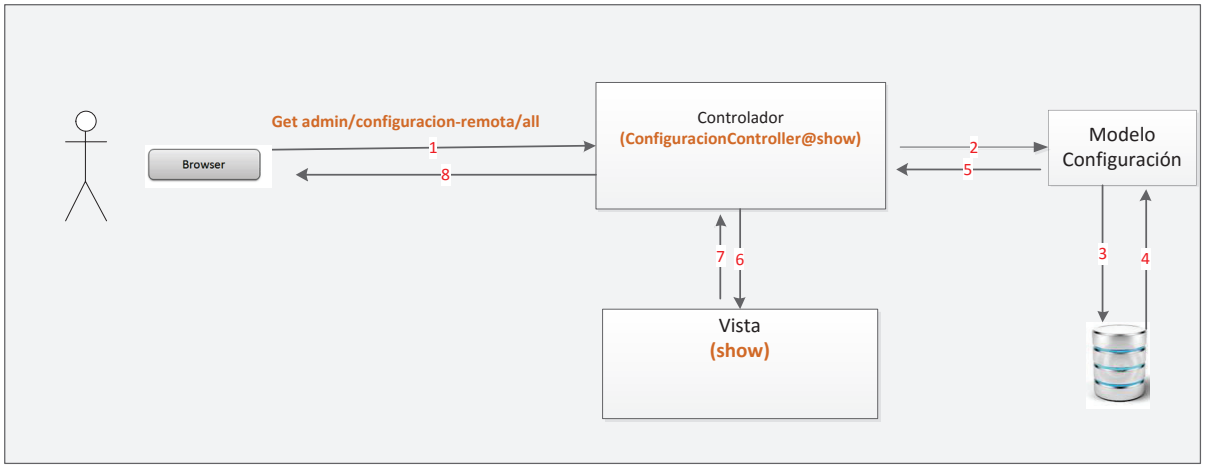

Figura 3.62 MVC para el proceso de configuración remota

Modelo

Se encarga de gestionar, procesar y validar los datos. Además se encarga de las relaciones existes con otras clases establecidas en la Figura 3.8. A continuación se detallan los métodos utilizados para este módulo.

- **validarDatos:** sirve para validar los datos que ingresa el usuario, además especifica si son datos requeridos, si son únicos y el tipo de dato.

Dentro del modelo se representa las relaciones de las tablas, esto ayuda a realizar las operaciones de consulta a la base de datos, lo antes mencionado se presenta en la Figura 3.63.

Vista

Se ha diseñado 1 vista que permiten al usuario realizar las funciones requerida para el proceso de configuración remota.

La vista **show** se ha desarrollado para presentar un listado clientes y automóviles conjuntamente con cada una de las actividades que puede realizar el usuario como:

verificar el historial de mediciones, nueva configuración, última configuración e historial de configuración.

Además en esta vista se realiza el proceso de selección de Dispositivos Arduinos, SIM, modos, parámetros y tiempo de muestreo. La vista se ha diseño como se presenta en la sección 3.1.5.5.

```
class Configuracion extends \Eloquent {

// verifica que los datos que se introcuden sean los solicitados, en caso de que no cumpla
// con lo solicitado se desplegará una aviso de error.
    protected $table = 'configuraciones';
    protected $fillable = array('id_sim', 'id_dispositivo', 'id_automovil', 'fecha', 'tiempo_muestreo');

    public function validarDatos($datos){

        $validar = Validator::make($datos, array(
            'id_sim'=>'required|unique:configuraciones',
            'id_dispositivo'=>'required|unique:configuraciones',
            'id_automovil'=>'required|unique:configuraciones',
            'tiempo_muestreo'=>'required'
            )
        );

        if($validar->passes()){
            return true;
        }

        $this->errors = $validar->errors();

        return false;
    }
// Relaciones
    public function automovil(){
        return $this->belongsTo('Automovil'); // una configuracion remota le pertenece a un automóvil
    }

    public function dispositivo(){
        return $this->hasOne('Dispositivo'); // una configuración remota tiene un dispositivo
    }

    public function sim(){
        return $this->hasOne('Sim');// una configuración remota tiene una tarjeta SIM
    }

    public function itemsConfiguracion(){
        return $this->hasMany('ItemConfiguracion');// una configuración remota tiene muchos item de configuración
    }
```

Figura 3.63 Clase configuración**Controlador**

Dentro del controlar se presentan los siguientes métodos:

- Show; direcciona a la vista donde el usuario selecciona cada uno de los elementos necesarios para realizar el proceso de configuración remota.
- Store; importa todos los datos seleccionados en la vista show y los guarda en la base de datos. Además realiza él envió del mensaje de configuración como se indica en la Figura 3.64.

```php
public function store()
{
    $id_parametros = [];
    $pid_parametros = [];

    $datosConfiguracion = Input::only(array('id_automovil', 'id_dispositivo', 'id_sim', 'tiempo_muestreo'));
    $parametros = Input::except(array('numero_sim','id_automovil', 'id_dispositivo', 'id_sim', 'id_modo',
                                      'tiempo_muestreo', '_token'));
    $numero_sim = Input::only('numero_sim');

    //Separando id y PID de parámetros

    foreach ($parametros as $key => $id_pid) {
        $separacion = explode("-", $id_pid);
        array_push($id_parametros, $separacion[0]);
        array_push($pid_parametros, $separacion[1]);
    }

    // Guardando la configuración
    $configuracion = new Configuracion();
    $configuracion->fill($datosConfiguracion);
    $configuracion->save();
    $id_configuracion = $configuracion->id;

    foreach ($id_parametros as $id_parametro) {
        $items_configuracion = new ItemConfiguracion();
        $items_configuracion->id_configuracion = $id_configuracion;
        $items_configuracion->id_parametro = $id_parametro;
        $items_configuracion->save();
    }

  // envió de la trama
    $comunicacion = new ComunicacionArduino();

    $trama = $comunicacion->procesar($datosConfiguracion['id_automovil'], $datosConfiguracion['tiempo_muestreo'],
     $pid_parametros, $id_configuracion);

    if($comunicacion->enviar($numero_sim['numero_sim'], $trama)){

        return Redirect::to('/admin/configuracion-remota/all')->with('mensaje',
            'La configuracion se ha guardado exitosamente<br> El mensaje se ha enviado correctamente');

    }else{

        return Redirect::to('/admin/configuracion-remota/all')->with('error',
            'El mensaje no se ha podido enviar');
    }


    Session::forget('sim');
    Session::forget('arduino');
    Session::forget('confAutomovil');
```

Figura 3.64 Método store

3.2.3.4 Implementación del módulo visualización de históricos

Este módulo permite al usuario visualizar un conjunto de parámetros del sistema OBD-II que fueron solucionados previamente.

Las tablas que intervienen en el proceso de visualización de históricos son:

- Mediciones.
- Capturas

La clase que permite representar la entidad mediciones es la clase Medición.

Al implementar MVC para realizar el proceso de visualización de medidas se requirió de la vista `show`.

De igual manera se requirió del controlador: MedicionesController.

A continuación se presenta en la Figura 3.65 el funcionamiento de MVC aplicado al módulo visualización de históricos.

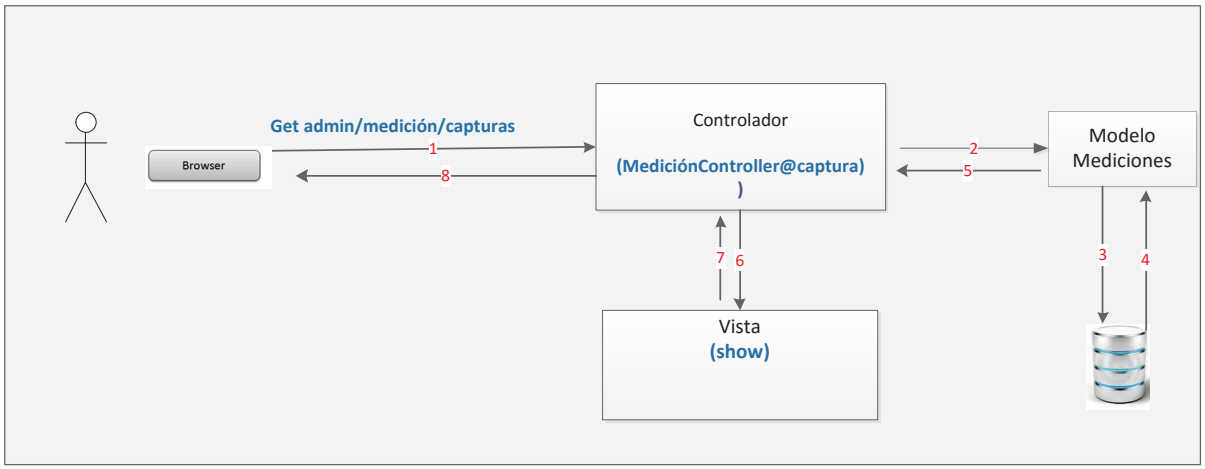

Figura 3.65 MVC de proceso de visualización de históricos

Modelo

Dentro del modelo se describe las relaciones establecidas en el diagrama de clases como se indica en Figura 3.66 .

```php
<?php

class Medicion extends \Eloquent {

    protected $table = 'mediciones';

    public function caputras(){
        return $this->belongsTo('Captura');
    // Una medición le pertence a una captura
    }

    public function parametro(){
        return $this->belongsTo('Parametro');
    // una medición de pertenece a un parámetro
    }
```

Figura 3.66 Clase medición

Vista

Se implementó la vista `show`, en la cual el usuario puede visualizar los parámetros seleccionados en el proceso de configuración remota, en Figura 3.41 se presenta el bosquejo de la vista desarrollada.

Controlador

El contralor es el intermediario en la vista y el modelo, el método creado para el proceso de visualización de históricos es:

- **Capturas**; en este método se direcciona a vista `show`, en la cual se presentan los datos de los parámetros seleccionados, en la Figura 3.67 se presenta del código implantado para el método antes mencionado.

```
public function capturas($id){

    //En esta condición trae los valores de mediciones
    if(is_numeric($id)){

        $arrayConfig = array();

        $configuraciones = DB::table('configuraciones')->where('id_automovil', '=', $id)
                                                       ->orderBy('created_at', 'desc')
                                                       ->get(array('id', 'tiempo_muestreo',
                                                                   'created_at'));

        foreach ($configuraciones as $value) {
            $arrayConfig[] = $value->id;
        }

        $ultimaConfiguracion = array_shift($arrayConfig);

        $resultado = $this->consultarHistorial($ultimaConfiguracion, $id);

        return View::make('mediciones/show')
        ->with('parametros', $resultado['parametros'])
        ->with('medidas', $resultado['medidas'])
        ->with('configuraciones', $configuraciones)
        ->with('datos', $resultado['datos_automovil'])
        ->with('ultimaConfiguracion', $ultimaConfiguracion)
        ->with('capturas', $resultado['capturas'])
        ->with('nombre_parametros', $resultado['nombre_parametros']);
```

Figura 3.67 Método capturas

Un proceso similar se realizó para el módulo configuración de incidencias, el código completo está adjunto en el ANEXO 4.

3.2.3.5 Implementación del módulo gráficos estadísticos

Este módulo permite al usuario visualizar los gráficos estadísticos de las medidas seccionadas en la configuración remota.

Las tablas que intervienen en el proceso de presentar los graficas estadísticos son:

- Mediciones.
- Capturas

La clase que permite representar la entidad medición es la clase Medición.

Al implementar MVC para realizar el proceso de visualización de los gráficos se requirió de la vista `show`.

De igual manera se requirió del controlador: MedicionesController.

A continuación se presenta en la Figura 3.68 el funcionamiento de MVC aplicado al módulo visualización de gráficos estadísticos.

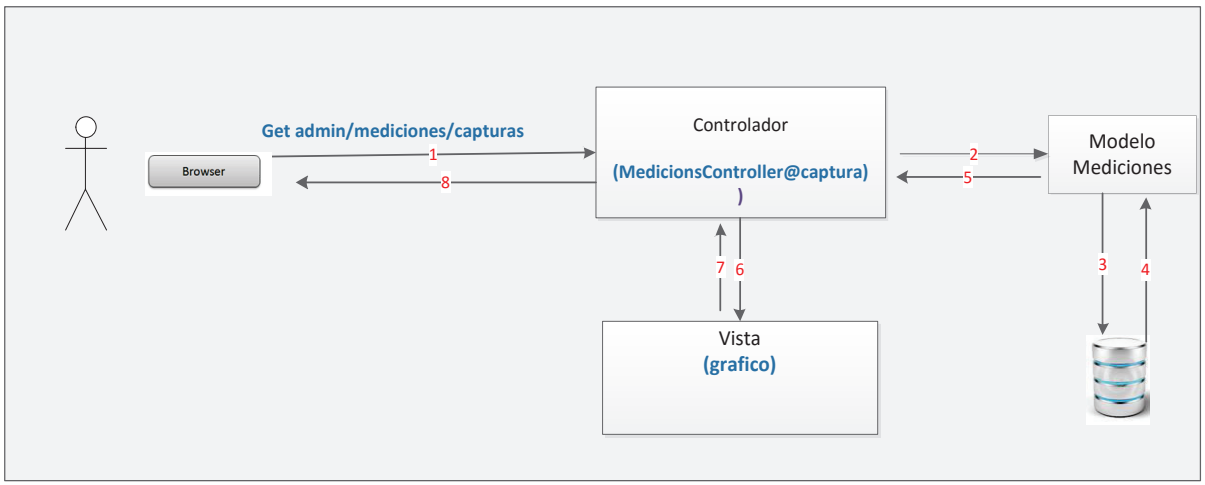

Figura 3.68 MVC para el proceso de visualización de gráficos estadísticos

Modelo

El modelo es el mismo que se usó en el módulo visualización de históricos, debido a que los gráficos se utilizaron para representar dichas medidas de manera gráfica.

Visita

Se usó la misma vista del módulo visualización de históricos con la diferencia que al seleccionar un parámetro se desplegará los checkbox, esta vista se puede visualizar en la Figura 3.42.

Controlador

Se tomó a consideración el mismo método capturas, únicamente se implementa las siguientes condiciones para graficar las medidas como se indica en la Figura 3.69.

- Si se no especifica la hora y fecha
- Si se introduce la hora y la fecha tanto inicial como final

```
}else{//En esta condición trae los valores seleccionados para ser graficados


    $valores = [];
    $nombre_parametros = [];

    $fecha_inicial = Input::get('fecha_init');
    $fecha_final = Input::get('fecha_fin');
    $id_parametros = Input::except(array('ultimaConfiguracion','fecha_init', 'fecha_fin'));

    $identificadores = explode("-", $id);//Primer elemento id del automovil

    if(empty($fecha_inicial))
    {
        //Sirve solo como referencia cuando no se especifica ningúna fecha en el formulario
        $fecha_init = '2010-01-01';
        $hora_init= '00:00:00';
    }else{
        $dataTimeInit = explode("/", $fecha_inicial);
        $fecha_init = $dataTimeInit[0];
        $hora_init = $dataTimeInit[1];
    }
    if(empty($fecha_final))
    {
        //Sirve solo como referencia cuando no se especifica ningúna fecha en el formulario
        $fecha_fin= '2020-01-01';
        $hora_fin = '23:59:59';
    }else{

        $dataTimeEnd  = explode("/", $fecha_final);
        $fecha_fin  = $dataTimeEnd[0];
        $hora_fin = $dataTimeEnd[1];
    }
```

Figura 3.69 Método capturas para visualizar los gráficos estadísticos

Se realiza un procesamiento similar para visualizar los gráficos estadísticos de los valores de las incidencias, el código se adjunta en el ANEXO 4.

3.2.3.6 Implementación del módulo GPS – posición actual

Este módulo permite al usuario visualizar la posición actual del automóvil a través de un mapa.

Las tablas que intervienen en el proceso de visualización de la posición son:

- Automóviles
- Clientes
- Modelos
- Marcas

Al implementar MVC para realizar el proceso de visualización de la posición actual se implementó la vista `gps`.

De igual manera se requirió del controlador: GpsController.

A continuación se presenta en la Figura 3.70 el funcionamiento de MVC aplicado al módulo visualización de la posición actual.

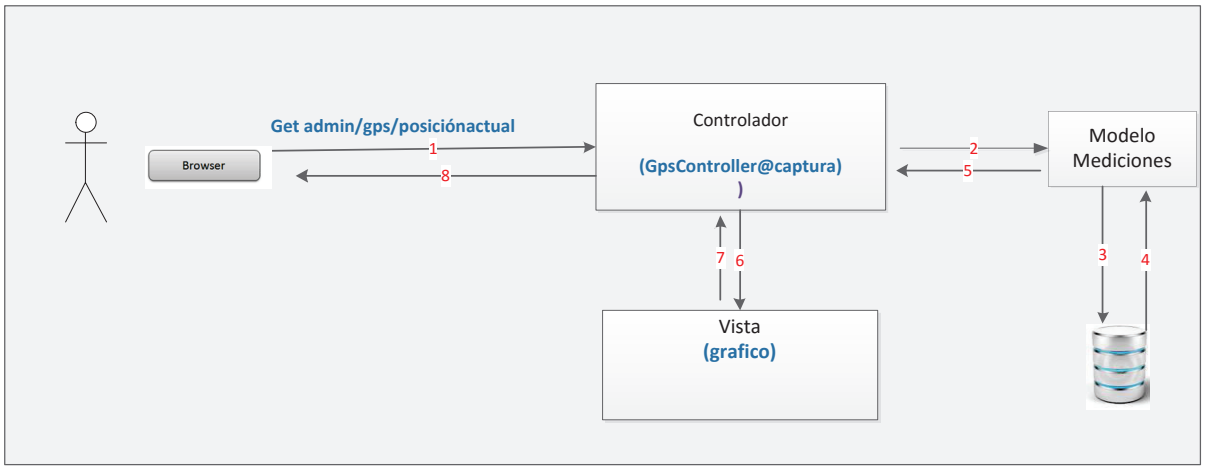

Figura 3.70 MVC para el proceso de visualización de la posición actual

Vista

Se implementó la vista `mapa` para obtener la posición actual del automóvil, dentro de esta vista se utilizó AJAX y funciones propias que define la API de Google Maps[12] como:

- `map = new google.maps.Map(document.getElementById('map')`
- `marker = new google.maps.Marker`

Controlador

Dentro del controlador se desarrolló los siguientes métodos.

- **Index,** sirve para verificar que el automóvil está conectado al sistema de monitoreo.
- **Posición actual**, sirve para traer los datos necesarios para presentar en la vista mapa.

Se desarrolló un proceso similar para visualizar la ruta del automóvil a través de Google Maps. El código completo esta adjunto en el ANEXO 4.

[12] https://developers.google.com/maps/documentation/javascript/markers

3.2.3.7 Implementación del módulo detección de incidencias

Para que se realice la detección de incidencias, se crea un procesos en segundo plano con la herramienta cron, que hace que se ejecute el programa escrito en PHP verificar_incidencias. Como se indica en la Figura 3.71.

```
SHELL=/bin/bash
PATH=/sbin:/bin:/usr/sbin:/usr/bin
MAILTO=root
HOME=/

# For details see man 4 crontabs

# Example of job definition:
# .---------------- minute (0 - 59)
# |  .------------- hour (0 - 23)
# |  |  .---------- day of month (1 - 31)
# |  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
# |  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) OR sun,mon,tue,wed,thu,fri,sat
# |  |  |  |  |
# *  *  *  *  * user-name command to be executed
*/1 * * * *  root GET http://localhost/verificar_incidencias
```

Figura 3.71 Proceso demonio

Modelo

Dentro del modelo se presentan las relaciones que se especificaron en la Figura 3.8 esto ayuda a realizar consultas a la base de datos. En la Figura 3.72 se presenta las relaciones implantadas en este modelo,

```
class RegistroIncidencia extends \Eloquent {

    protected $table = 'registro_incidencias';

    /*Relaciones*/
    public function automovil(){
        return $this->belongsTo('Automovil');
    //Un registro de incidecia le pertenece a un automóvil
    }

    public function incidencia(){
        return $this->hasOne('Incidencia');
    //Un registro de incidencia tiene una incidencia
    }

    public function medicion(){
        return $this->hasMany('Medicion');
    //Un registro de incidencia tiene muchas mediciones
    }
}
```

Figura 3.72 Modelo detección de incidencias

Controlador

Esta clase ayuda a verificar si se ha generado una incidencia y poder enviar al cliente una notificación de la incidencia producida. En la Figura 3.73 se presenta las funciones implementadas para el algoritmo detección de incidencias.

```php
<?php
class RegistroIncidenciaController extends BaseController
{

    public function index()
    {...
    }

    //estructura del mail
    private function procesar($medicion, $condicion)
    {...
    }

    // Envia el mail al usuario
    private function enviarEmail($id_capturas, $medicion, $condicion, $informacionCliente)
    {...
    }
}
```

Figura 3.73 Implementación del controlador del Registro de incidencias

La función index(), contiene el código donde se verifica si existe una incidencia, primero realiza una búsqueda del "último_id_medicón" en la tabla "registro_incidencias", de esta búsqueda se derivan dos opciones: si no existe, traerá todos los datos de la tabla "mediciones", caso contrario, sí existe datos de registro de incidencias traerá los datos de la tabla "mediciones" con la condición que el id_medicion sea mayor al valor de la variable $ultimo_registro_incidencia. Como se indica en la Figura 3.74.

El sistema verifica si se produjo una incidencia realizando comparaciones de los valores de los campos de tablas: condiciones, parámetros, incidencias y mediciones. Si se cumple una de las condiciones se ha generará una incidencia como se indica en Figura 3.75.

```php
<?php
class RegistroIncidenciaController extends BaseController
{

    public function index()
    {


        $mensaje = '';

        $ultimo_registro_incidencia = DB::table('registro_incidencias')->max('ultima_id_medicion');


        if ($ultimo_registro_incidencia === null) {
            $mediciones = Medicion::all(array(
                'id',
                'id_capturas',
                'pid',
                'valor'
            ));


        } else {
            $mediciones = DB::table('mediciones')
            ->where('pid', '!=', 'hora')
            ->where('pid', '!=', 'long')
            ->where('pid', '!=', 'lat')
            ->where('pid', '!=', 'fecha')
            ->where('id', '>', $ultimo_registro_incidencia)->get(array(
                'id',
                'id_capturas',
                'pid',
                'valor'
            ));
        }
```

Figura 3.74 Implementación de la función index

```php
// Verificación si existe  incidencias
//se realiza una consulta entre las tablas "parámetros" e "incidencias"
//los valores de los campos "id_incidencia, operación, valor_condicion,
//pid, nombre_parámetros y descripción".
if (count($mediciones) > 0) {
    $condiciones = DB::table('condiciones')->join('parametros', 'parametros.id', '=', 'condiciones.id_parametro')
                                           ->join('incidencias', 'incidencias.id', '=', 'condiciones.id_incidencia')
                                           ->get(array(
        'incidencias.id',
        'operacion',
        'valor_condicion',
        'pid',
        'nombre_parametro',
        'descripcion'
    ));
// Para que exista la incidencia debe cumplir una de las tres condiciones
//Si el valor de campo "valor" de la tabla mediciones" es <, > o = al valor del campo "valor" de la tabla "condiciones",
//se ha generado una incidencia. Y se llama a la función procesar.
//En caso de que no se cumpla con ninguna condición, no se a producido ninguna incidencia


    foreach ($condiciones as $keyCon => $condicion) {
        foreach ($mediciones as $keyMed => $medicion) {
            if ($condicion->pid === $medicion->pid) {
                if ($condicion->operacion === "<") {
                    if (intval($medicion->valor) < intval($condicion->valor_condicion)) {
                        $mensaje = $this->procesar($medicion, $condicion);
                    }
                } elseif ($condicion->operacion === ">") {
                    if (intval($medicion->valor) > intval($condicion->valor_condicion)) {
                        $mensaje = $this->procesar($medicion, $condicion);
                    }
                } else {
                    if (intval($medicion->valor) == intval($condicion->valor_condicion)) {
                        $mensaje = $this->procesar($medicion, $condicion);
```

Figura 3.75 Implementación del proceso de comparación

Luego se estructura el mail, se realiza una consulta de los datos de la placa, motor, nombre, apellidos, username y email del automóvil que ocasione la incidencia. La

incidencia será guardada en la tabla “registro_incidencias”. Como se indica en la Figura 3.76.

```php
//estructura del mail
private function procesar($medicion, $condicion)
{

    date_default_timezone_set('America/Guayaquil');
    $fecha_actual = date('Y-m-d');
    $hora_actual  = date('H:i:s');

    $informacionCliente = DB::table('automoviles')
    ->join('capturas', 'capturas.id_automoviles', '=', 'automoviles.id')
    ->join('clientes', 'clientes.id', '=', 'automoviles.id_cliente')
    ->where('capturas.id', '=', $medicion->id_capturas)
    ->get(array(
        'automoviles.id',
        'placa',
        'motor',
        'nombres',
        'apellidos',
        'username',
        'email'
    ));
```

Figura 3.76 Implementación para la estructura del mail

Finalmente en la Figura 3.77 se observa el algoritmo para enviar el mail al correo del usuario.

```php
    // Envia el mail al usuario
    private function enviarEmail($id_capturas, $medicion, $condicion, $informacionCliente)
    {
        try {
            Mail::send('emails.notificacion', array(
                'medicion' => $medicion,
                'condicion' => $condicion,
                'informacionCliente' => $informacionCliente
            ), function($message) use ($informacionCliente)
            {
                $message->to($informacionCliente[0]->email, $informacionCliente[0]->nombres)->subject('Incidencias registradas');
            });
            return true;
        }
        catch (Exception $e) {
            return false;
        }
    }
}
```

Figura 3.77 Implementación para él envió del mail al usuario

3.2.3.8 Implementación para el envió del mail

Laravel proporciona una API simple y limpia llamada mandrill[13]. Para utilizar la solución es necesario registrarse en Mandrill y actualizar el archivo mail.php. El archivo mail.php se encuentra en el directorio “app/config/mail.php”.

[13] https://laravel.com/docs/5.2/mail

Primero se debe cambiar el servidor “SMTP”, en este caso se usará mandrillapp.com, luego se debe especificar el puerto en ese caso se usó “587” como puerto por defecto. En la línea ‘from’, se coloca la dirección de correo que aparecerá con remitente conjuntamente con el nombre de la persona que envía el mail.

En la siguiente línea se especifica el tipo de encriptación en este caso se dejará por defecto el “tls”, es necesario colocar el username con la que se ingresa a mandrill y colocar en password, la key que proporciona mandrill. En la Figura 3.78 se observa lo antes descrito.

```php
<?php
return array(
    /* Mail Driver */
    'driver' => 'smtp',
    /*SMTP Host Address */
    'host' => 'smtp.mandrillapp.com',
    /*SMTP Host Port*/
    'port' => 587,
    /*Global "From" Address*/
    'from' => array('address' => 'eliana.monterolr@gmail.com', 'name' => 'Eliana Montero'),
    /* E-Mail Encryption Protocol*/
    'encryption' => 'tls',
    /* SMTP Server Username */
    'username' => 'eliana.monterolr@gmail.com',
    /*SMTP Server Password*/
    'password' => 'EccoH8JfpG5bDk7Byc93Gw',
    /* Sendmail System Path    */
    'sendmail' => '/usr/sbin/sendmail -bs',
    /* Mail "Pretend"*/
    'pretend' => false,
);
```

Figura 3.78 Código del archivo mail.php

3.2.3.9 Implementación para enviar el sms desde la aplicación

El código para él envío del mensaje desde la aplicación hacia el sistema de recepción se lo realizo en el directorio `helpers`. Estas funciones ayudan a realizar tareas o procedimiento específico y se las puede llamar o utilizar en cualquier parte código. En la Figura 3.79 se muestra el directorio `helpers` con las funciones necesarias para la solución del sistema.

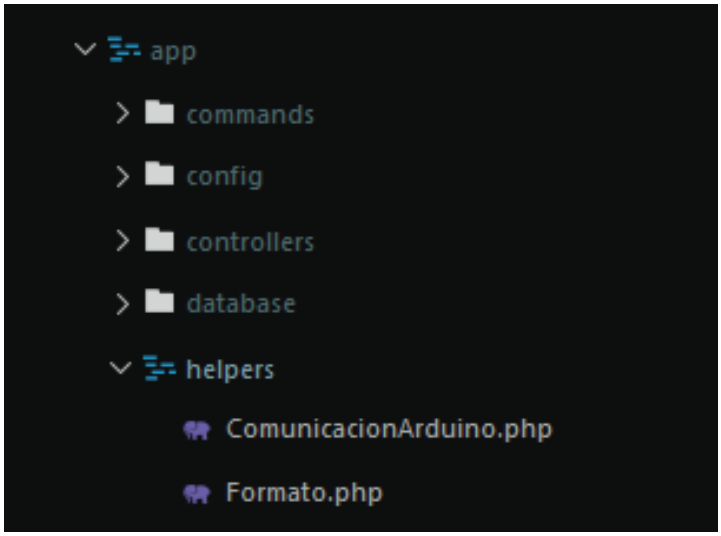

Figura 3.79 Directorio helper

3.2.3.9.1 Formato sms

En la Figura 3.80 se observa el formato de la trama del mensaje que se enviará desde el sistema al subsistema de recepción.

Tiempo de muestreo	Id_automóvil	Id_configuración	Número_total_PIDs	PID1	PIDn

Figura 3.80 Formato trama sms

Los valores id_automovil; id_configuración y el número total de PIDs, se deben transformar a hexadecimal.

El tiempo de muestreo se le da valores con formato hexadecimal en este caso se da los siguientes valores:

4 minutos= 01

8 minutos =02

12 minutos = 03

Luego en el array que trae los PIDs se inserta el separador ";". Finalmente se llama a la función enviar, en la cual se envía dos parámetros el número de la sim y la trama generada. El código fuente para la comunicación del Arduino se indica en la Figura 3.81.

```php
<?php

class ComunicacionArduino
{

    public function procesar($id_automovil, $tiempo_muestreo, $arrayPids, $id_configuracion)
    {
        $pids = array();
        $tiempo = "";
        $id_automovil_hex    = dechex(intval($id_automovil));
        $numero_parametros   = dechex(count($arrayPids));
        $id_config = dechex(intval($id_configuracion));

        if($tiempo_muestreo == "1"){
            $tiempo = "01";
        }elseif($tiempo_muestreo == "2"){
            $tiempo = "02";
        }else{
            $tiempo = "03";
        }

        $pids = implode(";", $arrayPids);

        $trama = $tiempo . ";". $id_automovil_hex . ";" . $id_config . ";" . $numero_parametros . ";" . $pids;

        return strtoupper($trama);

    }

    public function enviar($numero_sim, $trama)
    {

        $outcome = shell_exec("echo '$trama' | gnokii --sendsms '$numero_sim'");

        return true;
    }

}
```

Figura 3.81 Código fuente para el envío del mensaje

El código fuente completa del sistema Web se adjunta en el ANEXO 4.

3.3 MODIFICACIONES

Las modificaciones que se realizarón en el algoritmo del Arduino en la función `readTiemMuestreo()`, la cual se utiliza para leer la EEPROM. Se coloca una condición para que en caso de que no exista ninguna información en la EEPROM, se obliga a que lea el buffer cada 30 segundos; sin esta condición el algoritmo ingresará a un lazo infinito y no existe manera de salir de esta configuración a menos que se asigne esta condición. Como se indica en la Figura 3.82.

Para el proceso de captura de marcas de posicionamiento lo primero que se realiza es instanciar un objeto de la clase `dgps`, la cual se encuentra en la librería `dGPS.h`. En este objeto se encuentra todos los métodos necesarios para obtener la información que es enviada por el satélite a continuación se presenta el código implementado en la Figura 3.83.

```
void readTiemMuestreo()  //lee la EEPROM POSISION 0 EEPROM
{
      int value=0;
      Serial.println("TIEMPO DE MUESTREO ");
      value = EEPROM.read(1);  //lee la posicion 0 de memoria EEPROM
      Serial.print(1);
      Serial.print("\t");
      Serial.print(value);  //valor expresado en int
      if (value==01){
         tiemp_muestreo=60;  //
       }else if (value==02){
         tiemp_muestreo=480; // 480 c/8 min
       }else if (value==03){
         tiemp_muestreo=720; // 720 c/12 min
       }else{
         tiemp_muestreo=60; //  Si no tiene en el buffer ninguna información se
        // obliga al que tome como tiempo de muestreo 30 segundos para que
       // siga leyendo el buffer hasta leer el mensaje.
       }
      Serial.print("tiempo muestreo: "); Serial.println(tiemp_muestreo);
      Serial.println();
}
```

Figura 3.82 Código modificado para evitar un lazo infinito

```
float desLat=0;                    //Almacena valores de punto flotante
float desLon=0;
long hora=0;
double latitud=0;
double longitud=0;
long fecha;

dGPS dgps = dGPS();                // Construcción de la clase dgps

void setup()
{
  Serial.begin(9600); // Inicializa el monitor serial
  SIM900.begin(9600); // para GSM shield
  //SIM900power();  // prende el shield
```

Figura 3.83 Clase dgps

Para iniciar la captura de datos se utiliza el método `init` de la clase `dgps`, en Figura 3.84 se presenta como se implementó este método la cual está ubicado en la función `void setup`.

```
  obd.begin();
  // Inicializa la conexion OBD-II hasta que sea exitosa
  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print("Conectando OBDII");
  Serial.println("Conectando OBD");

  while (!obd.init()){
  sprintf(buf, "INTENTOS # %d", ++errors);
  Serial.print(buf);
  Serial.println("OBD LISTO..");
  }
  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print("OBD LISTO..");
   Serial.println("Preparando el GPS");
  dgps.init();                    // Sentencia para la inicialización del dGPS .
  lcd.clear();
  lcd.setCursor(0,1);
  lcd.print("GPS LISTO..");
   Serial.println("GPS listo para usar");

}
```

Figura 3.84 Método init

Para actualizar la información del GPS se utilizó el método `update` implementado en la función `void loop` como se presenta en la Figura 3.85

```
void loop()
{
  Serial.println("Actualizando el GPS");
  dgps.update(desLat, desLon);    //Actualiza los datos del GPS.
  Serial.println("Actualizado el GPS");
    delay(5000);  //espera 10 seg antes de empezar a leer todo
    // Now we simply display any text that the GSM shield sends out on the serial monitor
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("Espera SMS/configuracion");
        read_parse_serial();  //Lee el SMS-configuracion guardado en el buffer
    //Serial.println("Esperando 30 segundos para leer eeprom");
    //delay(30000);
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("Leyendo EEPROM");
        readEEPROM();//lee eeprom
    //retardo(); //Obtiene y envia los datos cada tiempo de muestreo
    lcd.clear();
    lcd.setCursor(0,1);
    lcd.print("Monitoreando sensores OBDII");
        LeerOBD();
```

Figura 3.85 Método Update

La información que se necesitó obtener del módulo gps fue: longitud, latitud, tiempo y fecha. Para extraer esta información del objeto se utilizó los siguientes métodos: `Time`, `Lat`, `Lon` y `Date`. Lo antes mencionado se puede observar en la Figura 3.86.

```
if(dgps.Time()<50000){
  hora = dgps.Time()+190000;
}else{
  hora = dgps.Time()-50000;      // Devuelve la hora en el formato HHMMSS
                                 // ( H - Hora , Minuto - M , S - Segundos ) en 24 horas
}
        Serial.print("hora");
        SIM900.print("hora");
        SIM900.print(separador_PID);
        Serial.print(separador_PID);
        Serial.print(hora);
        SIM900.print(hora);
        SIM900.print(separador_medidas);
        Serial.println(separador_medidas);
latitud = dgps.Lat();
        Serial.print("lat");
        SIM900.print("lat");
        SIM900.print(separador_PID);
        Serial.print(separador_PID);
        Serial.println(latitud,6);
        SIM900.print(latitud,6);
        SIM900.print(separador_medidas);
        Serial.print(separador_medidas);
longitud = dgps.Lon();
        Serial.print("long");
        SIM900.print("long");
        SIM900.print(separador_PID);
        Serial.print(separador_PID);
        SIM900.print(longitud,6);
        Serial.print(longitud,6);
        SIM900.print(separador_medidas);
        Serial.println(separador_medidas);
fecha = dgps.Date();
        Serial.print("fecha");
        SIM900.print("fecha");
        SIM900.print(separador_PID);
        Serial.print(separador_PID);
        Serial.println(fecha);
        SIM900.print(fecha);
        SIM900.print(separador_medidas);
        Serial.println(separador_medidas);
        SIM900.print((char)26); //fin de linea de la trama
        char bufferSensor[16]={0}; //bufer para guardar la inf del sensor
```

Figura 3.86 Métodos implementados

Las modificaciones que se realizaron en el algoritmo Arduino se presentan el ANEXO 5.

CAPÍTULO 4

4 PRUEBAS

4.1 PRUEBAS UNITARIAS

Estas pruebas se realizaron en el transcurso de la implementación para probar cada una de las clases desarrolladas. Como resultado de esta estrategia se optimizó el tiempo de las pruebas y también se garantiza el buen funcionamiento de cada una de las clases así como sus métodos.

En la Tabla 4.1 se detalla las pruebas que se realizaron sobre el proceso de administración de automóviles. Un conjunto de pruebas similares se realizaron para el resto de módulos.

Clases y vista	Métodos	Resultado	Descripción
Class Automovil	validarDatos	Ok	Primero se realizó el ingreso de todos los datos solicitados por la interfaz. Segundo no se ingresó algunos datos y el sistema indico los campos que faltaban llenar.
Create	consultarModelos	Ok	Se verificó que al momento de seleccionar una marca el sistema despliega la lista de modelos pertenecientes a dicha marca.
Show		Ok	Se verificó que la vista show despliega un conjunto de información de los automóviles.
Edit	consultarMarcas	Ok	Se verificó que al seleccionar un modelo de automóvil el sistema indica al usuario la marca de dicho modelo.
Class AutomovilController	Create	Ok	Se comprobó que el método llama a la vista create donde se visualiza los datos a introducir para realizar el proceso de registro.

Tabla 4.1(1) Pruebas unitarias

Class AutomovilController	Store	Ok	Importa los datos introducidos en los campos vacíos de la vista create y los guarda en la base de datos.
	Show	Ok	Se verificó que el método llama a la vista show donde se visualiza un conjunto de información de los automóviles.
	Edit	Ok	Se comprobó que el método direcciona a la vista edit con los datos que el usuario puede editar.
	Update	Ok	Trae los datos de la vista edit y los actualiza en la base de datos.
	Destroy	Ok	Se verificó que al seleccionar el botón eliminar, el sistema despliega un mensaje solicitando la confirmación del proceso eliminar, y los datos son eliminados de la base de datos.

Tabla 4.1 (1) Pruebas unitarias

4.2 PRUEBAS DE FUNCIONALIDAD

Las pruebas de funcionalidad fueron realizadas por el usuario para asegurar que el sistema cumpla con los requisitos solicitados en las historias de usuario planteados en la etapa de diseño.

Se realizaron las pruebas de funcionamiento del sistema web de dos modos.

4.2.1 MODO 1

Se verificó las funcionalidades que el sistema presentó al usuario cuando no se monitoreaba el automóvil del cliente y se tenía datos previamente registrados.

Para comprobar que el sistema web era responsivo se utilizó 3 dispositivos diferentes: computadora personal, tableta y teléfono inteligente.

El escenario para estas pruebas fue un ambiente donde el usuario tenga acceso a internet para ingresar al sistema web a través de los diferentes dispositivos.

A continuación se detalla en la Tabla 4.2 las características de hardware del primer modo.

Dispositivos	Características
Servidor	Marca: Dell Procesador: Core i7 Memoria RAM: 16 GB
Computador personal	Marca: HP Procesador: AMD Memoria RAM: 4 GB
Teléfono inteligente	Marca: Samsung Modelo: J7 Procesador: Octa-Core Memoria RAM: 1,5 GB
Tableta	Marca: Apple Modelo: iPad Air Procesador: Chip A7 Memoria RAM: 1GB

Tabla 4.2 Características de hardware

A continuación se detalla en la Tabla 4.3 las características de software de los dispositivos utilizados en primer modo.

Dispositivos	Características
Servidor	Sistema operativo: Centos versión 6 Apache versión 2.2.31
Computador personal	Sistema Operativo: Windows 8, 64 bits Navegador: Chrome
Teléfono inteligente	Sistema Operativo: Android 5.1 lollipop Navegador: Chrome
Tableta	Sistema Operativo: iOS 9 Navegador: Safari

Tabla 4.3 Características de software

Se realizaron las pruebas de funcionalidad considerando cada una de las historias de usuario para verificar si dichas historias de usuario satisficieron al usuario. Las pruebas se realizaron sobre los tres dispositivos, a continuación se presenta las pruebas realizas utilizando una tableta.

4.2.1.1 Autenticación en el sistema

Esta prueba se hizo para verificar la historia de usuario HU14. Estas pruebas consistieron en validar el ingreso al sistema, si el usuario ingresa correctamente los datos, la siguiente página a observar es la página de inicio. Cuando el usuario ingresa erróneamente el usuario o contraseña se presenta un mensaje de advertencia como se indica en la Figura 4.1.

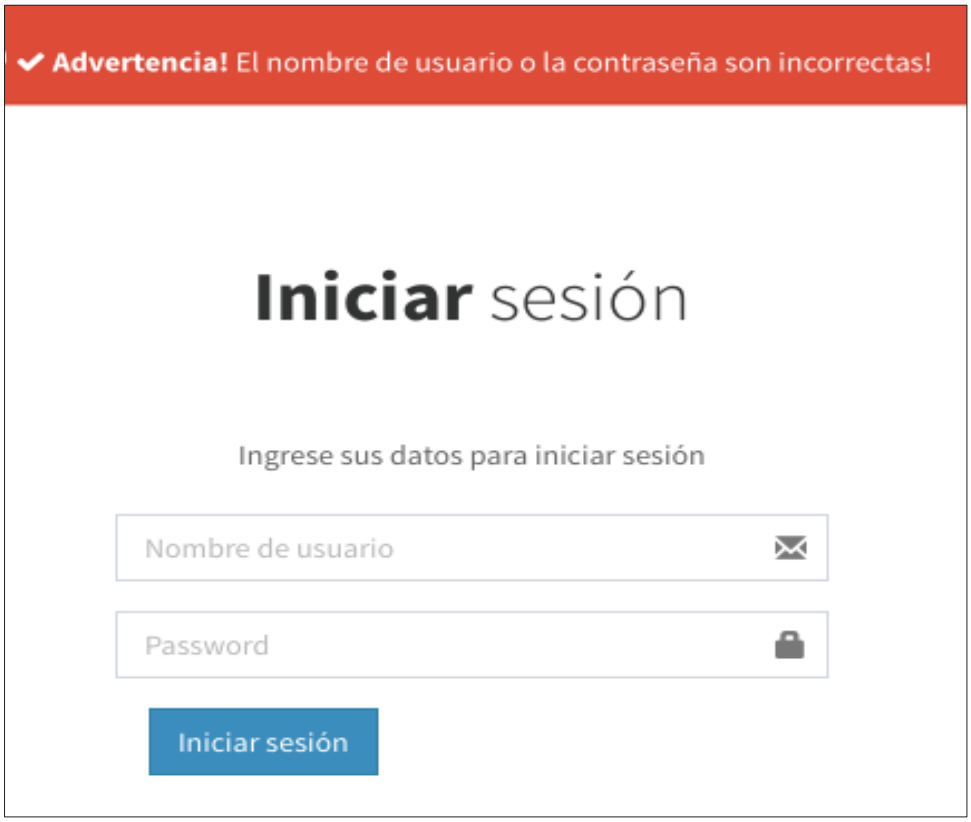

Figura 4.1 Autenticación al sistema

4.2.1.2 Administración de marcas de automóvil

Esta prueba se realizó para comprobar la historia de usuario HU1. Para verificar la historia de usuario se realizó el registro, modificación y eliminar una marca. En la Figura 4.2 se presenta las acciones indicadas, adicionalmente el usuario verificó la lista de modelo asociados a dicha marca.

En la Figura 4.3 se presenta el formulario para el ingreso de datos, al seleccionar el botón guardar el sistema realiza las validaciones respectivas y se presenta un mensaje indicando que los datos se han guardado con éxito.

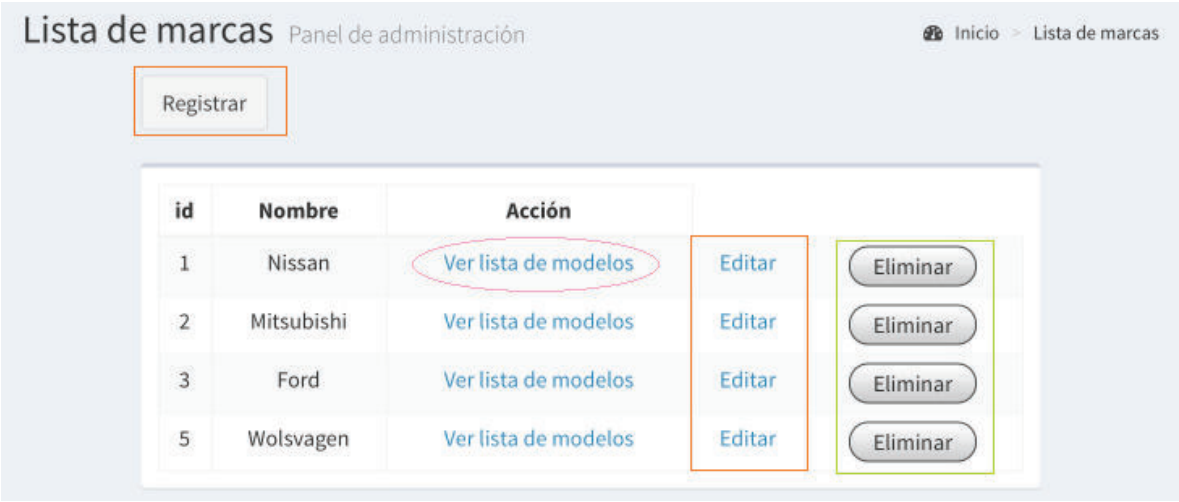

Figura 4.2 Funcionalidades de la administración de marcas

Para el proceso editar se presenta el mismo formulario de registrar con la diferencia que se presentan todos los campos llenos. Para el proceso de eliminar, una vez que se obtiene la lista de marcas de automóvil se selecciona el botón eliminar y se presenta una ventana emergente confirmando la eliminación del registro.

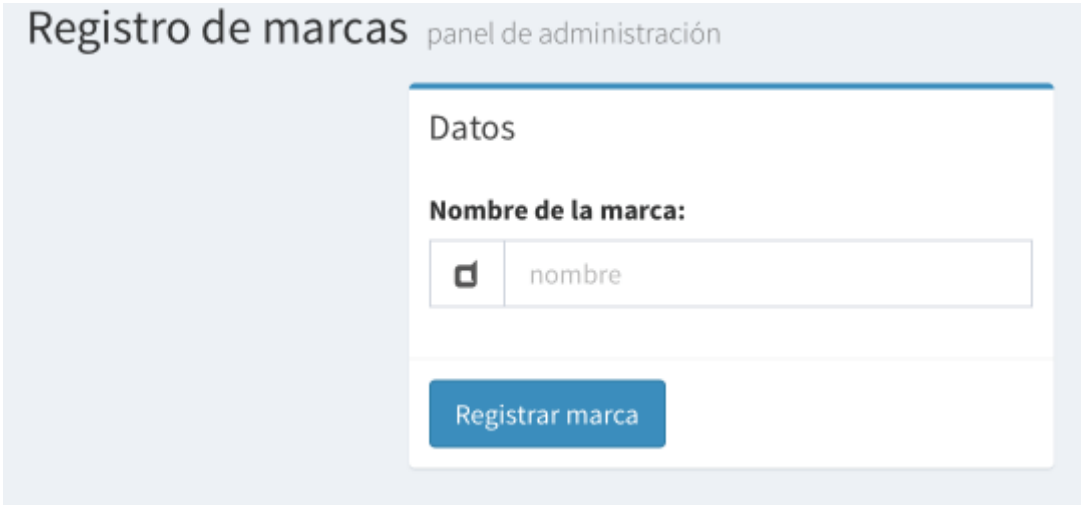

Figura 4.3 Registro de marcas

4.2.1.3 Administración de modelos de automóvil

Esta prueba se hizo para comprobar la historia de usuario HU2. Para verificar la historia de usuario se realizó las acciones de registro, modificación y eliminar de un modelo. El formulario es similar al de administración de marcas.

Para el registro de un modelo se debe seleccionar la marca a la cual pertenece dicho modelo como se indica en la Figura 4.4. Para el proceso editar se presenta el mismo

formulario de registrar con la diferencia que se presentan todos los campos llenos. Para el proceso de eliminar, una vez teniendo la lista de modelos se selecciona el botón eliminar y se presenta una ventana emergente confirmando la eliminación del registro.

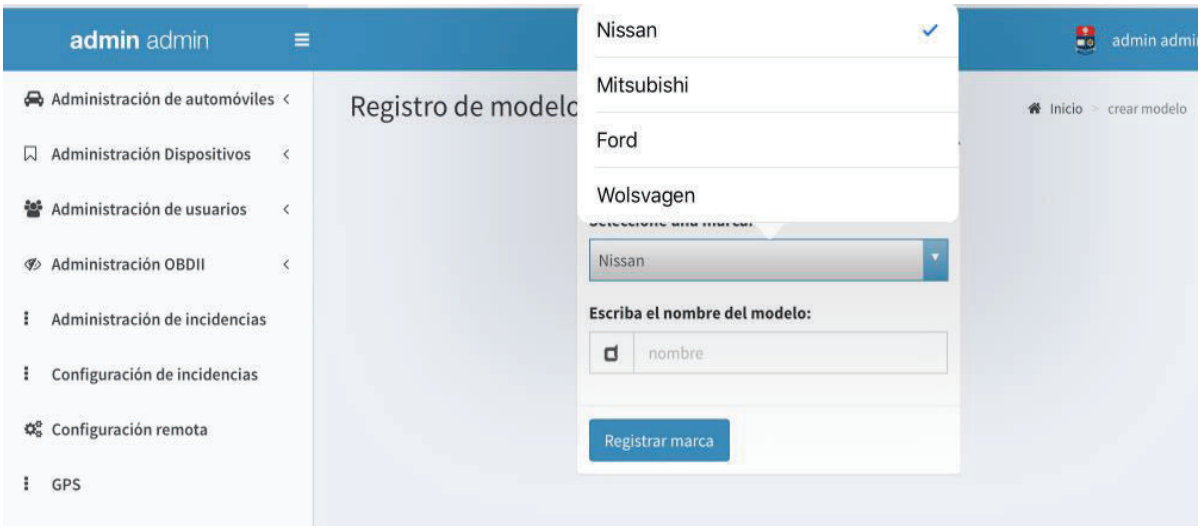

Figura 4.4 Registro de un modelo de automóviles

4.2.1.4 Administración de automóviles.

Esta prueba se realizó para comprobar la historia de usuario HU3. Para verificar la historia de usuario se realizó las acciones de registro, modificación y eliminar de un automóvil. El formulario es similar al de administración de marcas, con la diferencia que el usuario podrá verificar la lista de automóviles asociados a él.

Para el registro de un automóvil se debe seleccionar el nombre del usuario, la marca y el modelo, además se debe ingresar los datos: color, placa, motor, año y número de chasis. Para el proceso editar se presenta el mismo formulario de registrar con la diferencia que se presentan todos los campos llenos. Para el proceso de eliminar, una vez teniendo la lista de los automóviles se selecciona el botón eliminar y se presenta una ventana emergente confirmando la eliminación del automóvil.

4.2.1.5 Administración de clientes

Esta prueba se realizó para comprobar la historia de usuario HU4. Se realizó las acciones de registro, modificación y eliminar clientes. Además este formulario ofrece al usuario verificar la lista de automóviles asociados a él.

En la Figura 4.5 se presenta el formulario de ingreso de datos necesarios para el registro del nuevo cliente. Para el proceso editar se presenta el mismo formulario de registrar con las diferencias que se presentan todos los campos llenos y no podrá editar el rol asignado. Para el proceso de eliminar, una vez teniendo la lista de clientes se selecciona el botón eliminar y se presenta una ventana emergente confirmando la eliminación del automóvil.

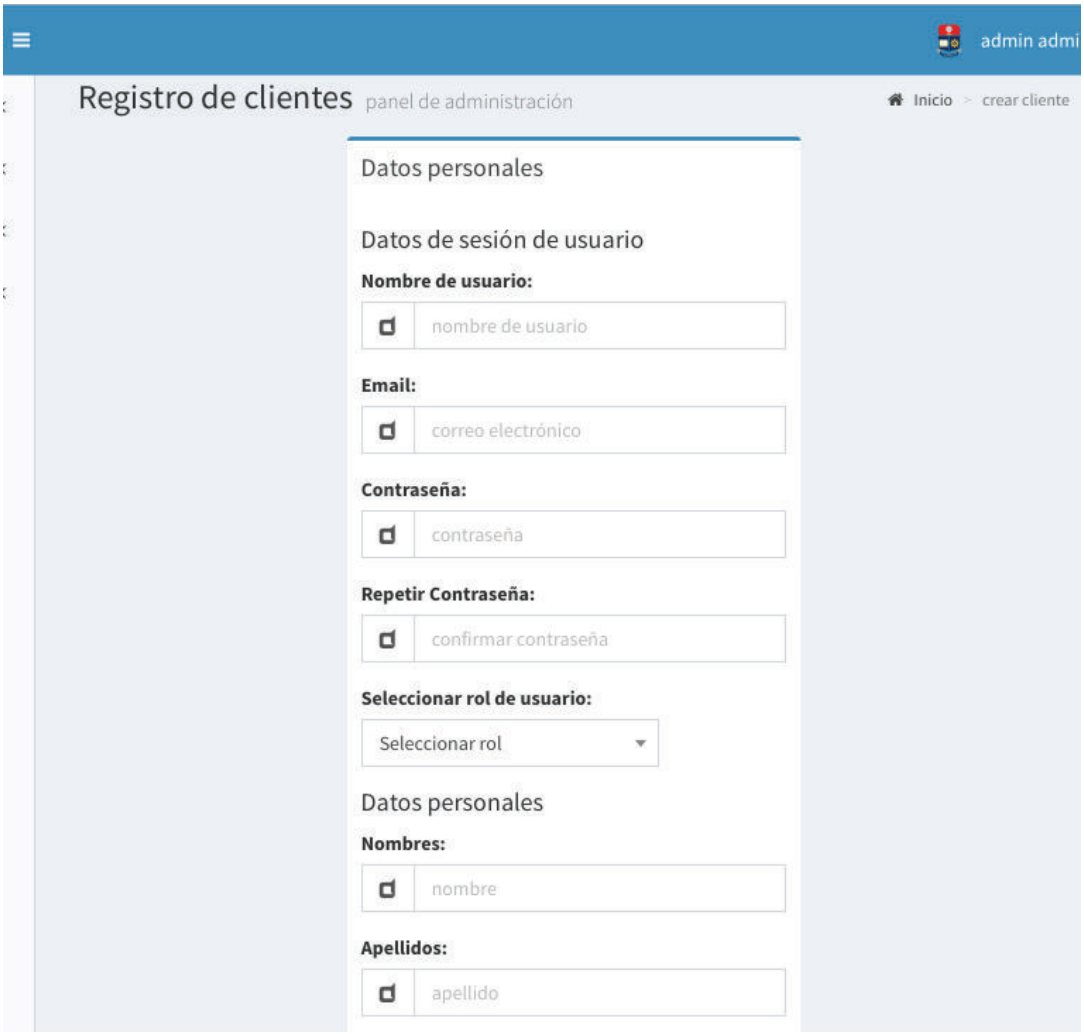

Figura 4.5 Formulario para el registro de clientes

4.2.1.6 Administración de helpdesk

Esta prueba se realizó para comprobar la historia de usuario HU5. Se realizó las acciones de registro, modificación y eliminar helpdesk. El formulario es similar al presentado en la administración del cliente con la única diferencia no tiene la opción de verificar la lista de automóviles asociados. El proceso de ingreso, editar y eliminar es similar al de administración de clientes.

4.2.1.7 Administración de administradores

Esta prueba se realizó para comprobar la historia de usuario HU6. Se realizó las acciones de registro, modificación y eliminar administradores. El formulario es similar al de administración de helpdesk.

El proceso de ingreso, editar y eliminar es similar al de administración de helpdesk. Con la única diferencia que si el sistema cuanta con un único administrador registrado este no podrá ser eliminado.

4.2.1.8 Administración de Arduinos

Esta prueba se realizó para comprobar la historia de usuario HU7. Se realizó las acciones de registro, modificación y eliminar Arduinos. El formulario indica al usuario el estado del Arduino, es decir si está asociado con un automóvil tendrá el mensaje de ocupado de caso contrario indicara que está disponible como se indica en la Figura 4.6.

Figura 4.6 Formulario del listado de dispositivos Arduinos

El proceso de ingreso, editar y eliminar es similar al de administración de marcas.

4.2.1.9 Administración de Tarjetas SIM

Esta prueba se realizó para comprobar la historia de usuario HU8. Se realizó las acciones de registro, modificación y eliminar tarjetas SIM. El formulario es similar al de administración de Arduinos. El proceso de ingreso, editar y eliminar es similar al de administración de Arduinos

4.2.1.10 Administración de modos del sistema OBD-ll

Esta prueba se realizó para comprobar la historia de usuario HU9. Se realizó las acciones de registro, modificación y eliminar de los modos del sistema OBD-II. Adicionalmente se puede observar que parámetros pertenecen a cada modo. El proceso de ingreso, editar y eliminar es similar al de administración de Arduinos.

4.2.1.11 Administración de parámetros

Esta prueba se realizó para comprobar la historia de usuario HU10. Se realizó las acciones de registro, modificación y eliminar parámetros. El proceso de ingreso y editar es similar al de administración de modelos de automóviles con la diferencia que se debe seleccionar el modo a cual el parámetro estará asociado. Para proceso de eliminar se escoge de la lista el parámetro a eliminar y el sistema pide la confirmación de eliminar dicho parámetro.

4.2.1.12 Administración de incidencias

Esta prueba se realizó para comprobar la historia de usuario HU12. Se realizó las acciones de registro, modificación y eliminar incidencias.

Para el proceso de ingreso de una incidencia se presenta la opción de agregar a dicha incidencia uno o más parámetros con sus respectivas condiciones y valores como se indica en la Figura 4.7. Para el proceso de editar el formulario es similar al de registro. Para proceso de eliminar se elige de la lista la incidencia a eliminar y el sistema pide la confirmación de eliminar dicha incidencia.

Figura 4.7 Formulario para el registro de una incidencia

4.2.1.13 Visualización de históricos de medidas del motor

Esta prueba se realizó para comprobar la historia de usuario HU20. Al seleccionar la opción ver medidas, se visualizó los parámetros obtenidos del módulo dispositivo conjuntamente con la hora y fecha de su captura como se indica en la Figura 4.8.

Datos del propietario.

Nombres: Fernando Becerra
Nombre de usuario: fernando
Correo electrónico:
fellas88@gmail.com

Datos del automóvil.

Color: Plateado
Placa: PCN3329
Motor: hr16832404h
Número de chasis: 3n1cc1ad6fk210280

Tiempo / Hora	RPM del motor	Velocidad del vehiculo
2015-12-19 / 10:09:35	650.000000	0.000000
2015-12-19 / 10:10:09	2250.000000	42.000000
2015-12-19 / 10:10:43	1662.000000	34.000000
2015-12-19 / 10:11:15	1675.000000	48.000000
2015-12-19 / 10:12:21	850.000000	16.000000
2015-12-19 / 10:12:55	650.000000	0.000000
2015-12-19 / 10:13:27	2400.000000	32.000000
2015-12-19 / 10:14:00	1700.000000	36.000000
2015-12-19 / 10:14:33	675.000000	2.000000
2015-12-19 / 10:15:06	662.000000	0.000000

Figura 4.8 Visualización de parámetros

4.2.1.14 Visualización de históricos de incidencias

Esta prueba se realizó para comprobar la historia de usuario HU21. Al seleccionar la opción ver historial, se visualizó los nombres de las incidencias con los valores que activaron la incidencia como se indica en la Figura 4.9.

4.2.1.15 Gráficos estadísticos de parámetros

Esta prueba se realizó para comprobar la historia de usuario HU16. Al seleccionar un parámetro en el formulario presentado en la HU20 se desplegó en la parte superior de cada parámetro un checkbox, al seccionar uno o más checkbox y filtrar por hora y fecha se obtuvo el gráfico estadístico de dichos parámetros como se indica en la Figura 4.10.

Datos del propietario.

Nombres: Fernando Becerra
Nombre de usuario: fer
Correo electrónico: fellas88@gmail.com

Datos del automóvil.

Color: Plateado
Placa: PBD2209
Motor: yh6
Número de chasis: gt65

Tiempo / Hora	RPM alto (Limite: 200)	Reduzca la velocidad (Limite: 30)
2016-01-14 / 20:11:01	862.00	34.00
2016-01-14 / 20:12:01	987.00	44.00
2016-01-14 / 20:12:03	2250.00	42.00
2016-01-14 / 20:13:01	1750.00	
2016-01-14 / 20:14:01	675.00	
2016-01-14 / 20:15:01	912.00	
2016-01-14 / 20:15:03	787.00	
2016-01-14 / 20:16:02	837.00	
2016-01-14 / 20:16:05	662.00	
2016-01-14 / 20:54:01	1937.00	

Figura 4.9 Visualización de históricos de incidencia

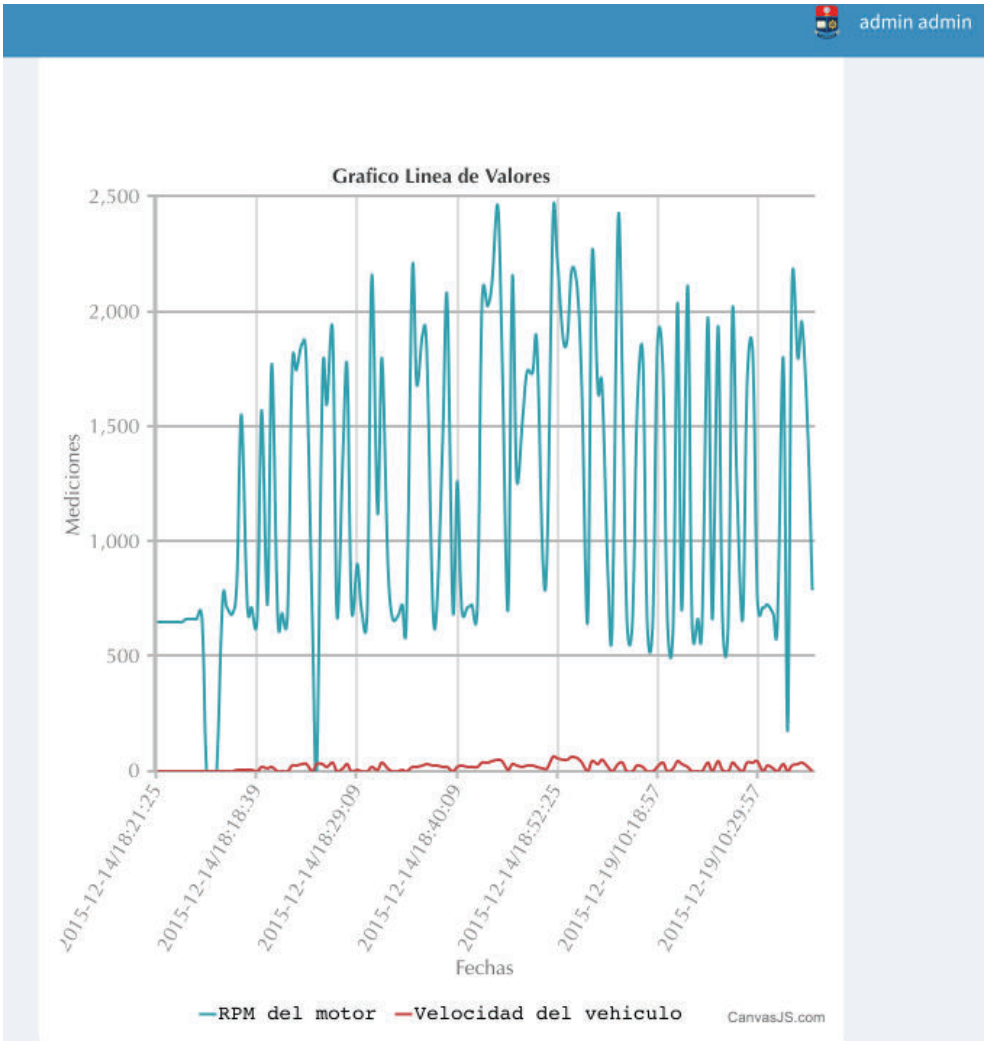

Figura 4.10 Gráfico estadístico de RPM y velocidad del vehículo

4.2.1.16 Gráficos estadísticos de incidencias

Esta prueba se realizó para comprobar la historia de usuario HU17. Se realizó el mismo procedimiento de la HU16, con la diferencia que se debe dirigir al formulario que presenta la HU21.

4.2.1.17 Histórico de la posición del automóvil

Esta prueba se realizó para comprobar la historia de usuario HU18. En la Figura 4.11 se filtró por fecha y hora y se obtuvo el recorrido del automóvil en dichas fechas.

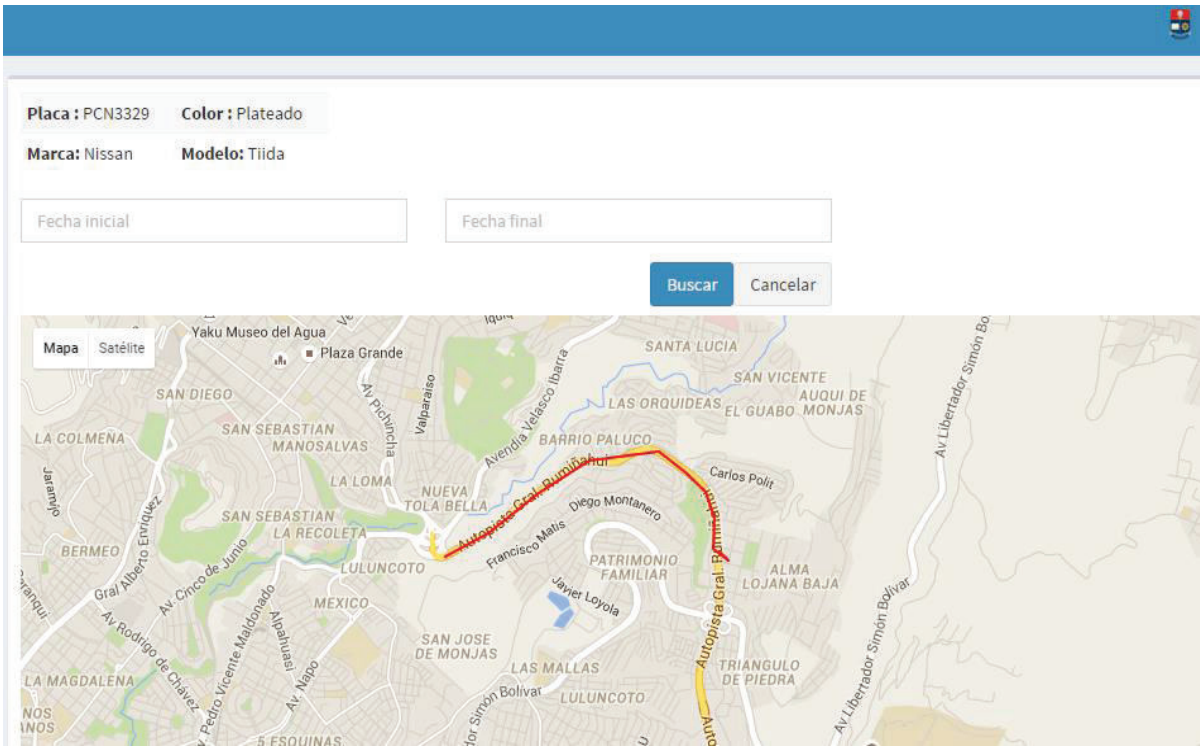

Figura 4.11 Ruta de posición del automóvil

En la Tabla 4.4 se verifica el correcto funcionamiento de cada módulo y además se comprueba que el sistema fue responsivo en todos los dispositivos utilizados; sin embargo hay que recalcar que hubo una distorsión al verificar el módulo del historial de la posición del automóvil en el teléfono inteligente, esto se debe a que la pantalla es muy pequeña, para solucionar este inconveniente se utilizó el modelo de rejilla o grid de 12 columnas de Boostrap, lo que hace es definir un modelo de divisor horizontal (una fila) divisible en 12 columnas [26]. En el código se cambió la etiqueta `<div class="col-sm-8">` por la etiqueta `<div class="col-xs-6">` para que las columnas de la rejilla se muestren horizontalmente en los dispositivos pequeños.

HU	Historia de usuario	Tableta	Computadora personal	Teléfono inteligente	Observación
HU1	Administración de marcas de automóviles	Ok	Ok	Ok	Ninguna
HU2	Administración de marcas de automóvil	Ok	Ok	Ok	Ninguna
HU3	Administración de automóviles	Ok	Ok	Ok	Ninguna
HU4	Administración de clientes	Ok	Ok	Ok	Ninguna
HU5	Administración de helpdesk	Ok	Ok	Ok	Ninguna
HU6	Administración de administradores	Ok	Ok	Ok	Ninguna
HU7	Administración de Arduinos	Ok	Ok	Ok	Ninguna
HU8	Administración de tarjetas SIMs	Ok	Ok	Ok	Ninguna
HU9	Administración de modos del sistema OBD-II	Ok	Ok	Ok	Ninguna
HU10	Administración de parámetros	Ok	Ok	Ok	Ninguna
HU12	Administración de incidencias	Ok	Ok	Ok	Se visualiza de mejor manera la vista de administración de incidencias cuando la pantalla del teléfonos inteligentes esta de forma horizontal.
HU14	Autenticación	Ok	Ok	Ok	Ninguna
HU16	Visualización de gráficos estadísticos de los parámetros	Ok	Ok	Ok	Ninguna
HU17	Visualización de gráficos estadísticos de incidencias	Ok	Ok	Ok	Ninguna
HU18	Histórico de posiciones del automóvil	Ok	Ok	Ok	Ninguna
HU20	Historial de las medidas del motor	Ok	Ok	Ok	Ninguna
HU21	Historial de Incidencias	Ok	Ok	Ok	Ninguna

Tabla 4.4 Resumen de las pruebas de funcionalidad del modo 1

4.3 MODO 2

Se verificó las funcionalidades que ofrece el sistema en tiempo real, es decir, cuando el automóvil estaba siendo monitoreando y con datos previamente registrados.

El escenario para estas pruebas fue un ambiente donde el usuario tenga acceso a internet para acceder al sistema web través de los diferentes dispositivos y un automóvil que tenga conectado el módulo adquisición datos y que este encendió el módulo dispositivo.

A continuación se detalla en la Tabla 4.5 las características de hardware del segundo modo.

Dispositivos	Características
Servidor	Marca: Dell Procesador: Core 7 Memoria RAM: 16 GB
Computador personal	Marca: HP Procesador: AMD Memoria RAM: 4 GB
Teléfono inteligente	Marca: Samsung Modelo: J7 Procesador: Octa-Core Memoria RAM: 1,5 GB
Tableta	Marca: Apple Modelo: iPad Air Procesador: Chip A7 Memoria RAM: 1GB
Automóvil	Marca: Nissan Modelo: Tiidia Año: 2015
Módulo dispositivo	Placa Arduino Mega 2560 Módulo GPS Módulo SIM900 GSM/GPRS
Módulo adquisición de datos	ELM 237

Tabla 4.5 Características de hardware

A continuación se detalla en la Tabla 4.6 las características de software de los dispositivos utilizados en segundo modo.

Dispositivos	Características
Servidor	Sistema operativo: Centos versión 6 Apache versión 2.2.31
Computador personal	Sistema Operativo: Windows 8, 64 bits Navegador: Chrome
Teléfono inteligente	Sistema Operativo: Android 5.1 lollipop Navegador: Chrome
Tableta	Sistema Operativo: iOS 9 Navegador: Safari

Tabla 4.6 Características de software

Las pruebas se realizaron sobre los tres dispositivos, a continuación se presentarán las pruebas realizas utilizando la tableta.

4.3.1.1 Configuración remota

Esta prueba se realizó para comprobar la historia de usuario HU11. Primero se seleccionó un Arduino y una SIM disponible como se indica en la Figura 4.12.

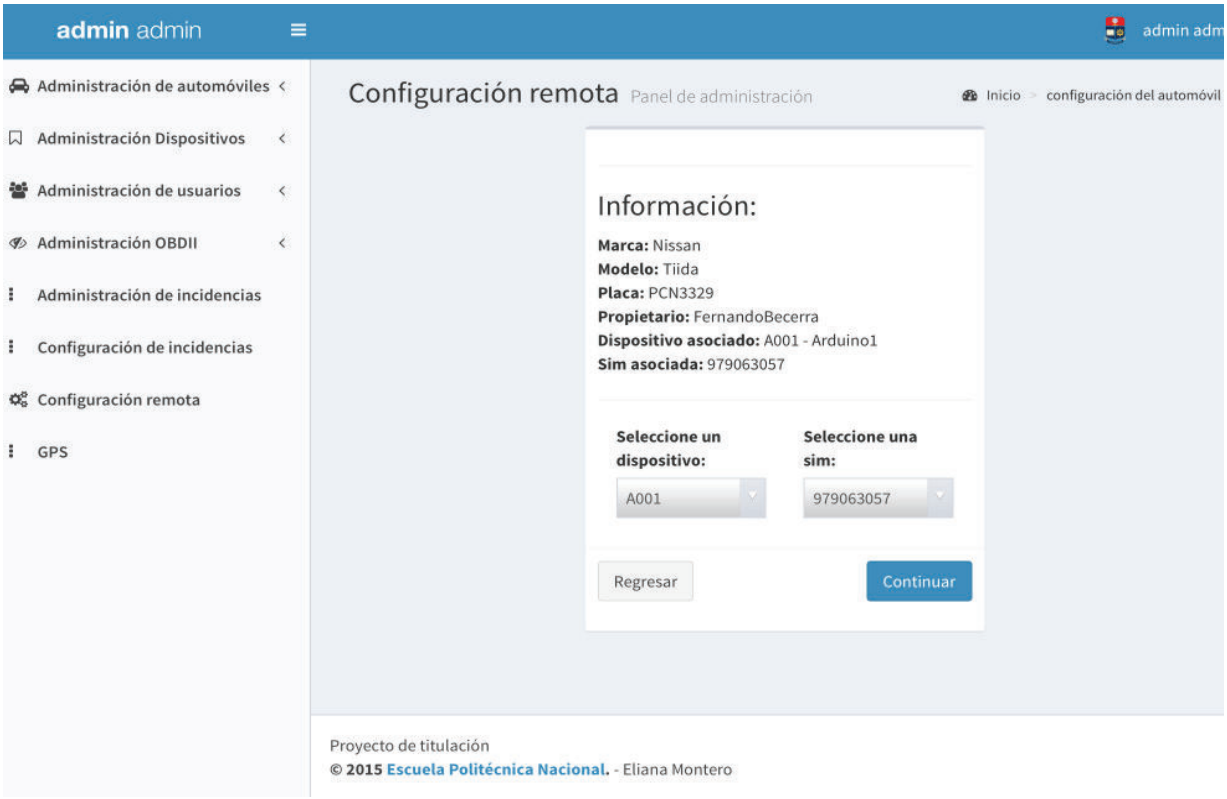

Figura 4.12 Selección de Arduinos y tarjetas SIM

Al presionar el botón “continuar” se despliega una interfaz secundaria, donde se debe seleccionar el modo del sistema OBD-II y se despliegan los parámetros asociados a dicho modo. Asimismo, se debe escoger el tiempo de muestreo.

Al seleccionar en uno de los checkbox se mostraó el parámetro en la columna “parámetros seleccionado” como se indica en la Figura 4.13 .

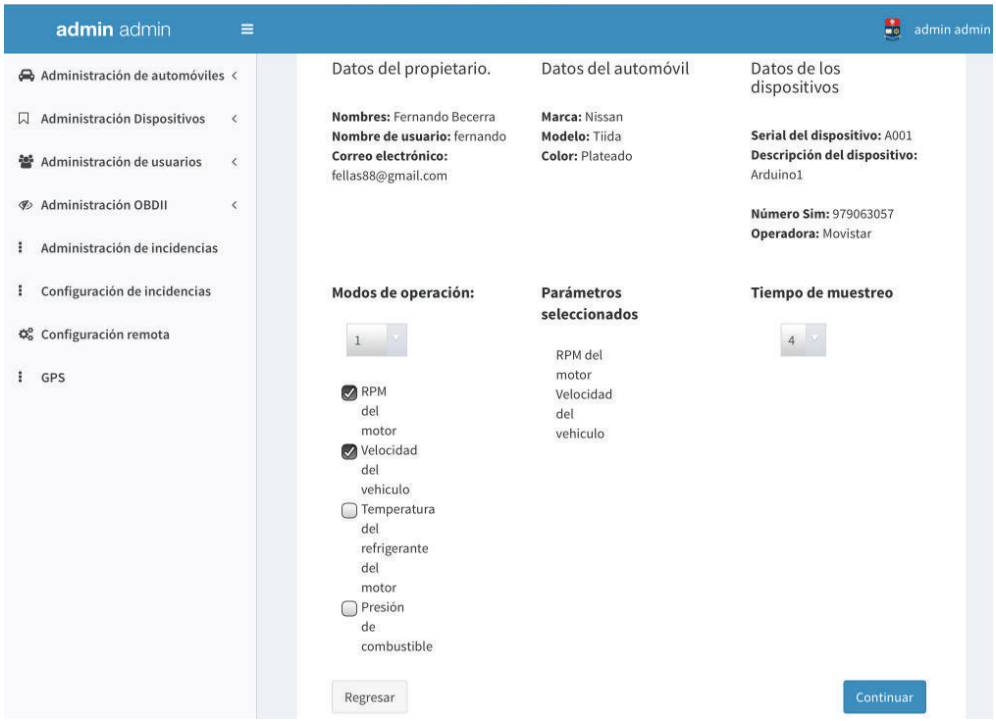

Figura 4.13 Selección de parámetros y tiempo de muestreo

4.3.1.2 Configuración de incidencias

Esta prueba se realizó para comprobar la historia de usuario HU12. Una vez terminado el proceso de configuración remota se procede a realizar la configuración de incidencias. En la interfaz web se despliega los parámetros seleccionados en la configuración remota asociados con una incidencia. Al seleccionar uno de los checkbox las incidencias se visualizaron en la columna “parámetros seleccionados”.

Al seleccionar el botón continuar se despliega un resumen antes de ser guarda en la base de datos.

4.3.1.3 Posición actual

Esta prueba se realizó para comprobar la historia de usuario HU19. Esta prueba consiste en verificar la posición actual del automóvil. En Figura 4.14 se observa que el automóvil está conectado, es decir, el módulo adquisición de datos está conectado al automóvil. Al seleccionar la opción “conectado” se despliega una interfaz secundaria.

Figura 4.14 Listado de automóviles conectados

En la Figura 4.15 se puede observar la posición actual del automóvil y las medidas capturadas en ese instante de los parámetros seleccionados previamente.

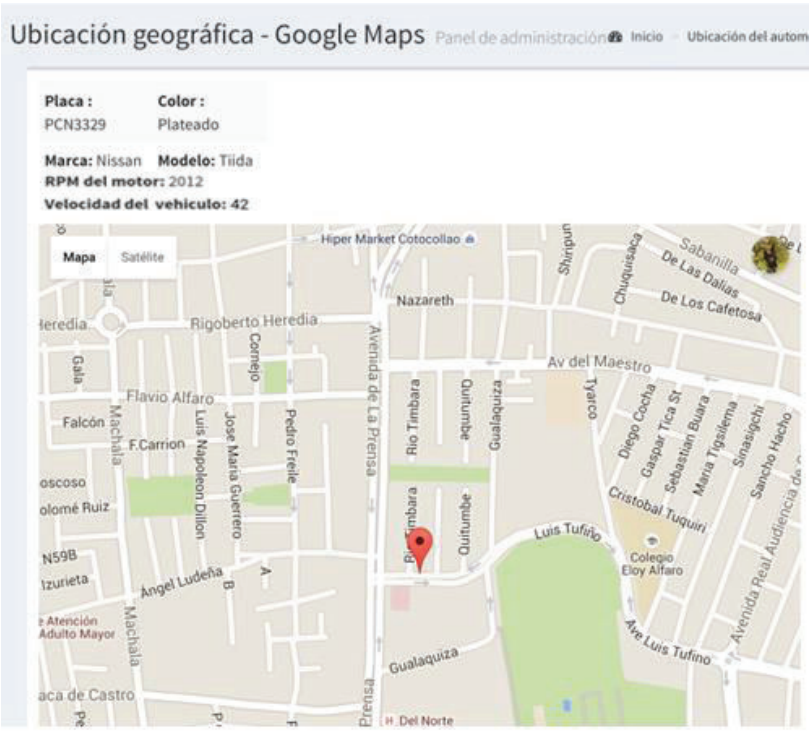

Figura 4.15 Visualización de la posición actual del automóvil

4.3.1.4 Detección de incidencias

Esta prueba se realizó para comprobar la historia de usuario HU15, se verifica que se ha generado la incidencia cuando las medidas cumplieron con la condición establecida en el registro de incidencia, y se puede observar dichas medidas en el módulo visualización de históricos de incidencias, esto se puede observar en la HU21.

4.3.1.5 Envió de la incidencia al usuario

Esta prueba se realizó para comprobar la historia de usuario HU22. Se verifica que se superó el valor de RPM en función de la condición establecida en el registro de incidencia, por lo que se recibe un mail indicando la incidencia como se indica en la Figura 4.16.

Figura 4.16 Mensaje de la incidencia

4.3.1.5.1 Problemas

El problema que se presentó en este escenario fue el envío de mail desde el servidor hacia el cliente, el mismo que sucedía aunque había conectividad a nivel de red y de capa transporte (puertos correctamente configurados). Este problema fue solucionado con el comando “setsebool -P httpd_can_network_connect on”. Este problema es causado por un módulo de seguridad SElinux[14] de Centos [27].

A continuación se presenta en la Tabla 4.7 los resultados obtenidos en cada uno de los dispositivos utilizados. Las pruebas realizadas en los diferentes dispositivos satisficieron las historias de usuario y el sistema cumplió con demostrar que es responsivo.

[14] Security-Enhanced Linux (SELinux) es un módulo de seguridad para el kernel Linux que proporciona el mecanismo para soportar políticas de seguridad para el control de acceso.

HU	Historia de usuario	Tableta	Computadora personal	Teléfono inteligente	Observación
HU11	Configuración remota	Ok	Ok	Ok	Se visualiza de mejor manera el proceso se selección de parámetros cuando la pantalla del teléfonos inteligentes esta forma horizontal.
HU12	Configuración de incidencias	Ok	Ok	Ok	Se visualiza de mejor manera el proceso se selección de incidencias cuando la pantalla del teléfonos inteligentes esta forma horizontal.
HU19	Visualización de la posición actual del automóvil	Ok	Ok	Ok	Ninguna
HU15	Detección de incidencias	Ok	Ok	Ok	Ninguna
HU22	Envió de reporte de incidencia al correo electrónico de cliente	Ok	Ok	Ok	Ninguna

Tabla 4.7 Resumen de las pruebas de funcionalidad del modo 2

CAPÍTULO 5

5 CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presenta un conjunto de conclusiones y recomendaciones conforme a la experiencia y conocimientos adquiridos en el presente trabajo.

5.1 CONCLUSIONES

El presente proyecto, en conjunto con los trabajos presentados en [6] [7], plantea una propuesta para el monitoreo remoto de los parámetros del sistema OBD-II de un vehículo. Para llevarse a cabo se ha utilizado tecnologías de comunicaciones (Arduino, módulos Arduino, módem, etc.), y de desarrollo de software (Framework Laravel, patrón MVC, etc.). Convirtiéndose en una aplicación directa de los conocimientos aprendidos en el transcurso de la carrera.

El haber realizado el proyecto como un sistema Web, se convierte en una gran ventaja, pues no existe la necesidad de desarrollar e instalar un software específico en la diversidad de dispositivos que existe actualmente. El usuario puede usar cualquier equipo como un computador, un teléfono inteligente o una tableta, con acceso a Internet y con un navegador.

Directamente relacionado con lo expresado con el uso del sistema en diferentes dispositivos, está la responsividad en la aplicación, que se refiere a la capacidad para que las interfaces se adapten a la pantalla del dispositivo sobre la cual se muestra. Esto se ha conseguido gracias al uso del fremework Bootstrap.

El sistema es escalable en cuanto a la información que soporta, esto se ha logrado a través de la creación de las siguientes administraciones: automóviles, marcas, modelos, dispositivos Arduinos, tarjetas SIM, modos, y parámetros del sistema OBD-II e incidencias, lo que permite adicionar nuevos datos acorde a las necesidades de los usuarios.

El módulo de configuración permite que controlar completamente el proceso de lectura de datos del sistema OBD-II sin que se necesite tener al auto cerca.

El módulo de incidencias ha sido diseñado de manera tal que el administrador o helpdesk pueda crear una incidencia a través del cumplimiento de una o más condiciones; con ello se pretende que se puedan representar incidencias simples y complejas.

Los gráficos y reportes estadísticos ayudarán a que personal calificado pueda tener interfaces amigables, y con ellas puedas establecer un análisis adecuado. Los mapas ayudarán a que los usuarios finales, puedan tener una visión rápida de la ubicación y el estado de los parámetros de su vehículo.

Respecto a las tecnologías utilizadas, el framework Laravel fue muy adecuado para la realización del sistema Web, pues presenta una curva de aprendizaje rápida, debido a su gran información online y a su abundante documentación en su sitio oficial. Además, ofrece una gran cantidad de funciones utilizadas en el sistema tales como una fácil integración en el proceso de conexión con la base de datos, funciones propias para validar datos y a la integración del servicio de email para notificar al cliente de las incidencias producidas. También se hizo uso del patrón MVC en Laravel, lo que ayudó a mantener el orden y la estructura del proyecto separando la parte de los datos, la lógica del negocio de la interfaz del usuario. El haber utilizado el framework Bootstrap ayudo a reducir tiempo de diseño en la parte gráfica de la aplicación y ayudó a que la aplicación sea responsiva.

Debido a que el sistema Web fue desarrollado con herramientas de hardware y software de código abierto será lo suficiente flexible para soportar el ingreso de nuevas funcionalidades o modificaciones si se desea brindar nuevos servicios a futuro.

Para poder capturar las marcas de posición se modificó el trabajo previo así: en el algoritmo para el Arduino se incluyó el algoritmo del shield GPS; se modificó el servicio Java el cual permite procesar el mensaje para registrar las mediciones en la

tabla respectiva de la base de datos; se modificó el mensaje que se envía desde el sistema Web al módulo de comunicación RX, ya que se incluyó el id de configuración, debido a que se necesitaba verificar las medidas capturadas a que configuración pertenece; y se añadió el mensaje que se envía del módulo dispositivo hacia el servidor, debido a que se incluyó las marcas de posición.

El basarse en una metodología de software para el desarrollo del proyecto ayudo a realizarlo de una forma ordenada y a entender las el ciclo de vida del software además de reducir el tiempo de desarrollo.

Al realizar pruebas de lectura de medidas del sistema OBD-II, se determinó que no todos los automóviles soportan los PIDs de la librería obd.h ya que dependen del protocolo de comunicación que utiliza el fabricante de cada automóvil.

5.2 RECOMENDACIONES

Para la selección de un framework es necesario que este ofrezca gran variedad de documentación tanto en español como en inglés, ya que esto ayudará a entender y aprender de forma rápido el framework y solucionar con brevedad los inconvenientes que se presente.

Al momento de poner en funcionamiento el proyecto se debe tomar en cuenta las limitaciones del GPS por tanto es recomendable ubicarlo en lugares donde no se vea comprometido, la mejor opción es ponerlo en el tablero del automóvil.

Antes de poner en funcionamiento el proyecto se debe verificar en el servidor Centos el módulo de seguridad SElinux, para no retrasar el proceso del funcionamiento del proyecto.

REFERENCIAS BIBLIOGRÁFICAS

[1] COMERCIO, «“Los dispositivos móviles facilitan el control vehicular”,» [En línea]. Available: http://www.elcomercio.com/tendencias/dispositivos-moviles-facilitan-control-vehicular.html. [Último acceso: 4 Noviembre 2015].

[2] M. R. Khondker H, Cost Effective GPS-GPRS Based Object, Conferencia Internacional de Ingenieros y Científicos informáticos, 2009.

[3] D. S. U. S. S. Sindhuja Bharthepudi, A Review of Low Cost Object Tracking System, India: KL University, 2013.

[4] MYCARTRACKS, «“MYCARTRACKS”,»[En línea]. Available:

http://www.mycartracks.com/. [Último acceso: 4 Julio 2015].

[5] J. E. Siegel, Design, Development, and Validation of a Remotely Reconfigurable Vehicle Telemetry System for Consumer and Government Applications, Massachusetts Institute of Technology, 2011.

[6] S. A. WILSON, DISEÑO E IMPLEMENTACIÓN DE UNA SOLUCIÓN TELEMÁTICA BASADA EN OBD-II (ON-BOARD DIAGNOSTIC) QUE PERMITA OBTENER Y PROCESAR LA INFORMACIÓN DE LOS SENSORES DEL MOTOR DE UN AUTOMÓVIL, Quito: Escuela Politecnica Nacional, 2014.

[7] L. Miriam, MPLEMENTACIÓN DE UN SISTEMA DE ADMINISTRACIÓN REMOTA PARA EL PROCESO DE OBTENCIÓN DE DATOS DEL SISTEMA OBD-II DE UN AUTOMÓVIL, Quito: Escuela Politecnica Nacional, 2015.

[8] M. R. Tamara Rodríguez, SISTEMA DE POSICIONAMIENTO GLOBAL,

Gravitación y Astrofísica, 2010.

[9] T. s. gps,[En línea]. Available: http://www.20minutos.es/noticia/2422423/0/galileo-gps/navegacion-satelite/incidentes-rumbo/. [Último acceso: 4 Octubre 2014].

[10] «“Qué es GPS”,» [En línea]. Available: http://www.tecnoprojectltda.com/QUEESGPS.htm. [Último acceso: 8 Octubre 2014].

[11] Dexterindustries, [En línea]. Available: http://www.dexterindustries.com/shop/gps-shield-arduino/.[Último acceso: 4 Julio 2014].

[12] V. M. O. V. Carlos Cortez, Metodologías agiles Metodologia XP, Corporación Universitaria del Caribe, 2013.

[13] L. Web, [En línea]. Available: http://librosweb.es/libro/xhtml/. [Último acceso: 4 Noviembre 2014].

[14] E. Bahit, «POO y MVC en PHP,» 2010.

[15] Apr . [En línea]. Available: http://www.aprenderaprogramar.com/. [Último acceso: 7 Noviembre 2014].

[16] Sublimetext, [En línea]. Available: http://www.sublimetext.com/.[Último acceso: 7 Enero 2015].

[17] Laravel, [En línea]. Available: http://laravel.com/docs/4.2. [Último acceso: 17 Noviembre 2014].

[18] Laravel, [En línea]. Available: http://laraveles.com/docs/4.1/mail. [Último acceso: 15 Diciembre 2014].

[19] Laravel, [En línea]. Available: https://laravelenespanol.wordpress.com/.[Último acceso: 21 Diciembre 2014].

[20] Bootstrap, [En línea]. Available: http://www.oneskyapp.com/es/docs/bootstrap/. [Último acceso: 17 Julio 2014].

[21] «Servidor Apache,» [En línea]. Available: http://httpd.apache.org/docs/2.0/es. [Último acceso: 17 Febrero 2015].

[22] Oracle, [En línea]. Available: http://www.oracle.com/es/products/mysql/overview/index.html. [Último acceso: 1 Septiembre 2014].

[23] A. J. Julio Cesar Caiza, Sistema de solicitudes de órdenes de compra de clientes en un patio de comidas mediante dispositvos móviles en una red inalambrica, Quito, Escuela Politécnica Nacional, 2010.

[24] Bootstrap, [En línea]. Available: http://speckyboy.com/2014/05/16/free-bootstrap-admin-themes/. [Último acceso: 1 Julio 2014].

[25] La Web, [En línea]. Available: http://librosweb.es/libro/javascript/capitulo_1.html. [Último acceso: 1 julio 2014].

[26] Bootstrap, [En línea]. Available: http://proyectosbds.com/blog/bootstrap-capitulo-2-comprender-el-modelo-de-rejilla-de-bootstrap/. [Último acceso: 07 marzo 2016].

[27] Sending mail, [En línea]. Available: http://stackoverflow.com/questions/9973449/sending-email-with-swift-mailer-gmail-and-php-permission-denied-error. [Último acceso: 7 diciembre 2015].

[28] Cerberus, [En línea]. Available: http://www.abc.es/tecnologia/moviles-aplicaciones. [Último acceso: 31 diciembre 2014].

[29] E. ERRERA, DISEÑO SISTÉMICO DE UNA INTERFAZ DE LOCALIZACIÓN AUTOMÁTICA DE VEHÍCULOS, I. P. NACIONAL, Ed., Mexico, 2013.

[30] T. J. Bravo Martín, Desarrollo e implementación del sitio web de la Adepon, Quito: Escuela Pilitecnica Nacional, 2013.

[31] PHP, [En línea]. Available: http://php.net/manual/es/intro-whatis.php. [Último acceso: 13 diciembre 2014].

[32] E. d. d. integrado, [En línea]. Available: https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.[Último acceso: 5 julio 2014].

[33] MySql, [En línea]. Available: http://www.definicionabc.com/tecnologia/mysql.php. [Último acceso: 23 diciembre 2014].

[34] Laravel, [En línea]. Available: http://blog.devacademy.la/post/95503250161/tutorial-laravel-conociendo-la-estructura-de-un. [Último acceso: 12 febrero 2015].

[35] Eloquent,[En línea]. Available: http://laraveles.com/docs/4.1/eloquent. [Último acceso: 21 febrero 2015].

[36] GPS, [En línea]. Available: http://www.tecnoprojectltda.com/QUEESGPS.htm. [Último acceso: 21 diciembre 2014].

[37] MySql,[En línea]. Available: http://www.w3schools.com/php/php_mysql_intro.asp. [Último acceso: 8 febrero 2015].

ANEXOS

ANEXO 1: INSTALACIÓN DE LARAVEL (CD ADJUNTO)

ANEXO 2: CREACIÓN Y POBLACIÓN DE LA BASE DE DATOS (CD ADJUNTO)

ANEXO 3: LIBRERÍA *dGPS.h* (CD ADJUNTO)

ANEXO 4: CÓDIGO FUENTE DEL SISTEMA WEB (CD ADJUNTO)

ANEXO 5: CÓDIGO FUENTE PARA ARDUINO MEGA 2560 (CD ADJUNTO)