Universidad de las Ciencias Informáticas

Facultad 2

Título: Sistema web de control y monitoreo de flotas.

Trabajo de Diploma para optar por el título de

Ingeniero Informático

Autores:

Luis López Fillad

Jonathan Vega González

Tutores:

Ing. Joan Martínez Herrera

Ing. Erick Pérez Castillo

Junio 2013

"Seamos realistas y hagamos lo imposible"

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Telemática de la Facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Autores:

_____ _____

Luis López Fillad Jonathan Vega González

Tutores:

_____ _____

Ing. Joan Martínez Herrera Ing. Erick Pérez Castillo

DATOS DE CONTACTO

AGRADECIMIENTOS

Queremos agradecer de forma especial a aquellas personas que formaron parte del desarrollo de este trabajo de diploma, personas que siempre estuvieron dispuestas a ayudarnos en el momento que lo necesitamos.

A los tutores, Joan Martínez Herrera y Erick Pérez Castillo, por su apoyo y guía durante todo el desarrollo del trabajo de diploma.

Al profesor Deivis Ricardo Álvarez Mendoza por su ayuda con el diseño de la página y el trabajo con jQuery.

Al profesor Yosel Hernández Escalona por dedicarnos el tiempo necesario para establecer la conexión con el servidor Nébula.

Al tribunal por las recomendaciones realizadas durante la predefensa.

DEDICATORIA

Luis

Dedico esta tesis a mis padres Gisela y Luis, a mi hermanita Gabriela, a mi novia Ibelise, y a mis abuelos y tíos, gracias a todos por su apoyo y confianza.

Jonathan

Dedico esta tesis a mis padres Manuela González y Omar Vega, por su cariño y guía en todo momento. Además, por su apoyo incondicional que me ayudó a alcanzar esta meta en mi vida y ser parte indispensable del hombre que soy. A mi novia Suany por haber llegado a mi vida en el momento indicado.

A mi familia y amigos por el apoyo y la confianza que siempre tuvieron en mí.

RESUMEN

Los sistemas de control y gestión de flotas son de gran utilidad para muchas empresas ya que permiten controlar y visualizar en un mapa el recorrido de sus flotas con el empleo de dispositivos GPS (del inglés Global Position System). Estos sistemas proporcionan un enorme beneficio tangible en cuanto a utilidades y el ahorro de capital. El empleo de estos sistemas ha ayudado a las empresas a eliminar problemas tales como, entregas fuera de tiempo, desvíos en los recorridos, reportes de mantenimiento y ahorro de combustible. En Cuba se cuenta con un sistema de control de flota, pero este no realiza un monitoreo durante la trayectoria del vehículo, por lo que con la ocurrencia de algún desperfecto técnico en la vía, se perderían valiosos minutos u horas entre la ocurrencia del suceso y la respuesta al mismo. Para dar solución a este problema se creó un sistema web capaz de monitorear y analizar el comportamiento de los vehículos durante su recorrido con el empleo de información GPS en formato KML o GPRMC. Además permitirá a los supervisores la creación de rutas y áreas, las que serán asignadas a los vehículos para definir su comportamiento. El sistema mantiene un control de las flotas a través del empleo de alarmas que son capaces de detectar las irregularidades en el comportamiento de los vehículos, tales como exceso de velocidad y desvíos de las rutas asignadas. Estas irregularidades quedan registradas en una base de datos para su posterior consulta.

PALABRAS CLAVE

GPS, Flota, Posicionamiento, Monitoreo, Control.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Los numerosos avances de la última década en la rama de la informática y las telecomunicaciones ha traído consigo la creación de diversos Sistemas de Información Geográfica (SIG), que combinados con tecnología GPS (del inglés Global Position System) han facilitado el desarrollo de aplicaciones informáticas capaces de monitorear cualquier tipo de navegación ya sea terrestre o marítima. Estos programas son conocidos como Sistemas de Control de Flotas, los cuales son usados para visualizar en un mapa vehículos que cuentan con dispositivos GPS. La gestión de flotas puede incluir una variedad de objetivos y funciones como el mantenimiento, el seguimiento y control, la detención remota de vehículos, el diagnóstico mecánico, la administración de conductores, la gestión de combustible, la gestión de la seguridad y, en general, todo lo referido al análisis de los datos e información disponible y a la toma de decisiones vinculados a la flota de vehículos. (1)

Cada vez es mayor la tendencia a instalar sistemas de control de flotas ya que proporcionan un enorme beneficio tangible a corto y mediano plazo en cuanto a utilidades y el ahorro de capital (2). Las empresas que más hacen uso de este tipo de sistemas son las relacionadas con el transporte de bienes por carreteras, equipos de ventas, ambulancias, bomberos, transportación de pasajeros o cualquier mercado que tenga la necesidad de mantener un control riguroso de su flota.

Una necesidad de las empresas cubanas es administrar los recursos con los que cuenta, por lo cual se ven obligadas a crear estrategias que ayuden a mejorar la administración de los medios que tienen a su disposición, trayendo consigo una mejora sustancial en la prestación de servicios tanto a la población como al sector empresarial.

La administración de los medios de transporte dentro de las empresas cubanas resulta ser una de las áreas más perjudicadas debido al gran número de indisciplinas que son cometidas por algunos trabajadores en este sector. Habitualmente ocurre que numerosas entidades tienen pérdidas económicas debido a las ineficiencias en los servicios que se prestan a través de medios de transporte. Entre los principales fallos se encuentran las entregas fuera de tiempo, las rutas ineficaces, los desvíos en los recorridos con intereses no correspondientes con la empresa, aumentando de esta forma el kilometraje, el descontento de los clientes, el gasto de combustible y una disminución de la vida útil de los equipos.

En Cuba se cuenta con un sistema de control de flota que posibilita obtener diversos datos de los vehículos una vez concluido su recorrido. Este sistema no permite estar al tanto de todo lo que ocurre durante el trayecto de las flotas hacia su destino. Por ejemplo, con la ocurrencia de algún desperfecto

técnico en la vía, se perderían valiosos minutos u horas entre la ocurrencia del suceso y la respuesta al mismo.

Luego de descrita la situación por la que atraviesan hoy las empresas cubanas que cuentan con diversos medios de transporte, surge el siguiente **problema a resolver**: ¿Cómo controlar el recorrido de los vehículos que pertenecen a las empresas cubanas durante su trayectoria?

A partir del problema a resolver se puede definir como **objeto de estudio**: Los procesos para el control de flotas, derivándose del mismo como **objetivo general**: Obtener un sistema informático para las empresas cubanas que permita la supervisión de los vehículos activos durante su recorrido. En el mismo se encuentra enmarcado el **campo de acción**: el control de flotas en empresas cubanas.

Para profundizar el objetivo general, se persiguen como **objetivos específicos**:

- Crear un mecanismo de obtención y decodificación de tramas con información GPS ya sea en formato KML (del inglés Keyhole Markup Language) o GPRMC (del inglés Recommended Minimum).
- Crear un sistema capaz de interactuar con los supervisores de flotas de vehículos mediante la visualización, inserción y modificación de datos en el mismo.

Con la finalidad de un desarrollo óptimo de la aplicación surgen las siguientes **preguntas científicas:**

- ¿Es posible visualizar el recorrido de los vehículos mientras se trasladan en sus funciones cotidianas?
- ¿La supervisión del recorrido de vehículos mediante medios informáticos, aumentará el control de las empresas cubanas sobre estos?

Para el cumplimiento de los objetivos se definen como **tareas de investigación**:

Selección de la metodología a seguir para el desarrollo del sistema.

Análisis de los principales lenguajes de programación para la implementación del sistema.

Análisis de servidores de base de datos para el almacenamiento de la información del sistema.

Definición de los patrones de diseño y arquitectura a utilizar para la posible solución.

Análisis de los diferentes sistemas de control de flota.

Análisis de las potencialidades de los diferentes servidores de mapa.

Análisis de la información contenida en las tramas GPS en los formatos KML y GPRMC.

Para la realización de las tareas de la investigación se utilizaron los siguientes métodos de investigación:

Métodos Teóricos:

Analítico sintético: Se aplica para analizar la documentación referente a servidores de mapa, las diferentes tecnologías y herramientas con las que se va a trabajar, para luego definir una síntesis específica de estas y lograr una mejor comprensión del objeto de estudio. Este método se evidencia en el capítulo uno en las definición de Tecnologías a Utilizar, Metodología de Desarrollo y Herramientas.

Histórico lógico: Este método ha sido usado con el fin de estudiar la evolución de los sistemas de control de flotas, tanto en Cuba como a nivel internacional. Este método se aplica en el capítulo uno en el análisis de las soluciones existentes

El presente trabajo de diploma está compuesto por cuatro capítulos, de los cuales se muestra un resumen a continuación:

Capítulo I "Fundamentación Teórica": Se exponen los aspectos teóricos relacionados con el desarrollo de este trabajo. Se hace un estudio de las tecnologías que se usan a nivel mundial y nacional en el ámbito de la gestión de flotas. Además se les da fundamento a las metodologías, lenguajes, conceptos y herramientas utilizadas.

Capítulo II "Características del Sistema": Se da una propuesta del sistema, se describe el modelo de dominio, se especifican los requisitos que debe cumplir el sistema y se define el diagrama de funcionalidades del sistema.

Capítulo III "Diseño del Sistema": Se elaboran los diagramas del diseño que propone la metodología empleada. Muestra la descripción de la arquitectura y patrones de diseño empleados. Incluye el diseño de la base de datos.

Capítulo IV "Implementación y Prueba": Contiene el diagrama de despliegue, a través del cual se puede ver la distribución del sistema de forma física. Contiene además la realización y descripción de los casos de prueba.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se hace un estudio de los principales sistemas de control de flotas en Cuba y el resto del mundo. Además se definen las herramientas, lenguajes y metodologías que serán utilizadas para el desarrollo de la aplicación.

1.2 Estudio del arte

Para conocer las características de las principales aplicaciones informáticas de gestión y control de flotas a nivel mundial se tomaron en cuenta los resultados de las búsquedas realizadas en Internet con diferentes buscadores como por ejemplo Google, Bing y Yahoo. A raíz de los resultados obtenidos se decide tomar como posibles soluciones las siguientes empresas.

Mobilefleet: Es una empresa especializada en servicios de gestión y control de flotas a través de una solución informática denominada PSPFleet, la cual es capaz de controlar activos móviles (vehículos, personas, maquinaria, etc.) donde se encuentren y además permite hacerlo desde cualquier sitio gracias a las nuevas tecnologías. PSPFleet es una aplicación web se encuentra disponible en Internet. Cuenta con acceso restringido mediante usuario y contraseña para que cada cliente acceda a la información de los activos móviles que quiere controlar, como por ejemplo rutas realizadas, informes, alertas de actividad, etc. (3)

Otra solución que se analizó, fue la desarrollada por la empresa Micronav. Empresa que surge en el año 2004 como respuesta a la creciente demanda por parte de las empresas de transporte de disponer de una solución tecnológica estándar que permita gestionar y controlar eficientemente los recursos de los que disponen. Entre las principales características con las que cuenta este sistema se encuentran el uso de la cartografía de GoogleMaps, seguimiento en tiempo real de las flotas y la opción de exportar los reportes. (4)

En Cuba se cuenta con una solución para la localización de vehículos denominada MovilWeb, esta es una aplicación web para el seguimiento de móviles sobre cartografía vectorial y raster, diseñada para controlar flotas dentro de una arquitectura cliente – servidor. Esta herramienta permite el monitoreo de móviles de manera remota sobre una red de comunicaciones, posibilitando reconstruir el comportamiento del vehículo en un determinado periodo de tiempo, reelaborando su trayectoria, analizando su velocidad y las detenciones no autorizadas. Esta aplicación se nutre de los servicios geoespaciales disponibles en la

Infraestructura de Datos Espaciales de la República de Cuba (IDERC), como los servicios de imágenes satelitales y cartografía vectorial. (5)

Luego de analizar las soluciones que ofrecen Mobilefleet y Micronav, se llega a la conclusión de que estas empresas disponen de software capaces de solucionar el problema, pero requieren de una conexión a Internet rápida y accesible en todo momento, ya que las aplicaciones se encuentran publicadas en los servidores de las respectivas empresas. Además utilizan servicios como Google Maps y Google Earth por lo que la utilización de estos sistemas en Cuba depende en gran medida de una infraestructura tecnológica con la cual no cuentan las empresas hoy en día. Existe una solución cubana denominada MovilWeb que permite gestionar las flotas y reelaborar su trayectoria, pero no realiza un monitoreo durante el recorrido del vehículo, imposibilitando que se tenga la mayor cantidad de información que permita controlar las flotas activas. Por lo que resulta necesaria la creación de un nuevo sistema de control de flotas que se adecue a las necesidades y características de las empresas cubanas.

1.3 Tecnologías utilizadas

Las tecnologías de la informática y las comunicaciones han tenido un gran avance en estos últimos años lo que ha propiciado la creación de muchas herramientas que facilitan la creación de sitios web. Para el desarrollo de la aplicación se hace necesario que las tecnologías que se usen ayuden a mejorar los resultados de la misma en cuanto a estabilidad y seguridad.

1.3.1 GPS (Global Positioning System)

El GPS es un sistema de radionavegación, basado en el espacio, que proporciona servicios fiables de posicionamiento, navegación, y cronometría gratuita e ininterrumpidamente a usuarios en todo el mundo. A todo el que cuente con un receptor del GPS, el sistema le proporcionará su localización y la hora exacta en cualesquiera que sean las condiciones atmosféricas, de día o de noche, en cualquier lugar del mundo y sin límite al número de usuarios simultáneos.

El GPS se compone de tres elementos: los satélites en órbita alrededor de la Tierra, las estaciones terrestres de seguimiento y control, y los receptores GPS propiedad de los usuarios. Desde el espacio, los satélites GPS transmiten señales que reciben e identifican los receptores del GPS; ellos, a su vez, proporcionan por separado sus coordenadas tridimensionales de latitud, longitud y altitud, así como la hora local. (6)

1.3.2 Socket

Un socket (enchufe), es un método para la comunicación entre un programa del cliente y un programa del servidor en una red. Un socket se define como el punto final en una conexión. Los sockets se crean y se utilizan con un sistema de peticiones o de llamadas de función a veces llamados interfaz de programación de aplicación de sockets. Los sockets permiten implementar una arquitectura cliente-servidor. La comunicación debe ser iniciada por uno de los programas que se denomina programa cliente. El segundo programa espera a que otro inicie la comunicación, por motivo se denomina programa servidor. La principal ventaja de los sockets radica en que son muy eficientes a la hora de enviar un número elevado de mensajes y datos. (7)

1.3.3 Lenguaje de programación

Un lenguaje de programación es un conjunto de símbolos, que combinados con reglas sintácticas y semánticas definen la estructura y el significado de sus elementos y expresiones. Los lenguajes de programación son utilizados para controlar el comportamiento físico y lógico de una máquina. *(8)*

1.3.3.1 Java

Java fue creado por un grupo de ingenieros de la empresa informática Sun Microsystems en 1991, en sus inicios se utilizó en Sun para uso interno. La idea original de Java era crear un lenguaje orientado a objetos independiente de la plataforma. Java ha tenido una gran evolución y hoy en día es uno de los lenguajes de programación más usados en el mundo. (9)

Java cuenta con las siguientes características:

- ✓ **Multiplataforma**: Java puede ser utilizado en cualquiera de los principales sistemas operativos del mercado, incluyendo Linux, Unix, Microsoft Windows, Mac OS X. Solo es necesario tener instalado la máquina virtual de Java (Java Virtual Machine).

- ✓ **Orientado a objetos**: Los objetos son estructuras encapsuladas que agrupan tanto su propia información como los métodos que manipulan esos datos permitiendo dividir programas largos en unidades semi-autónomas fáciles de identificar.

- ✓ **Compatibilidad con Base de Datos**: Amplia capacidad de conexión con la mayoría de los motores de base de datos como por ejemplo MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server y SQLite.

- ✓ **Software Libre**: El lenguaje Java está bajo licencia GPL2 (General Public License) por lo que se presenta como una alternativa de fácil acceso para todos.

1.3.4 Lenguaje de Modelado. UML (Unified Modeling Language)

El UML es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Captura decisiones y conocimiento sobre los sistemas que se deben construir y tiene como objetivo entender, diseñar, configurar, mantener, y controlar la información sobre tales sistemas. Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida, dominios de aplicación y medios. El lenguaje de modelado pretende unificar la experiencia pasada sobre técnicas de modelado e incorporar las mejores prácticas actuales en un acercamiento estándar. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos. (10)

1.3.5 Framework utilizados

Un framework es una estructura de archivos y utilidades que aceleran la programación de una aplicación informática, que provee una metodología de trabajo que sistematiza y facilita la generación de formularios, funciones y módulos de uso común, permitiendo al desarrollador dedicar su atención hacia los aspectos específicos de cada aplicación.

1.3.5.1 Spring Framework

Spring es un framework de código libre, creado por Rod Johnson[1] para hacer frente a la complejidad del desarrollo de aplicaciones empresariales. Cuenta con varios años en el mercado, lo que facilitado su evolución de forma constante y gracias a esto actualmente provee soluciones muy bien documentadas y fáciles de usar. Brinda además la posibilidad de integrarse con otras herramientas, así como con diversos frameworks como son: Hibernate y Spring Security. (11)

Las principales características por las que se decidió usar este framework se exponen a continuación (11):

- ✓ **Ligero**: Es un framework ligero en términos de tamaño y costes. La mayor parte de Spring Framework se puede distribuir en un único archivo .jar de aproximadamente 3MB (Megabytes).

[1] **Rod Johnson** es un especialista en computación australiano que creó el Spring Framework y es co-fundador de SpringSource.

- ✓ **Inyección de dependencia** (Dependency Injection) – Provee bajo acoplamiento a través de técnicas conocidas como inyecciones de dependencia. Cuando las inyecciones de dependencias son aplicadas, los objetos pasan en tiempo de ejecución sus dependencias en vez de crear la propia clase en el objeto.
- ✓ **Configurable**: Permite configurar y componer complejas solicitudes de los componentes más simples. Los objetos de aplicación son compuestos de forma declarativa, por lo general en un archivo XML (del inglés Extensible Markup Language) o en Anotaciones.
- ✓ **Flexible**: Diseñado como una serie de módulos que pueden trabajar independientemente uno de otro. Además mantiene un mínimo acoplamiento entre la aplicación y el propio Framework de forma que podría ser desvinculada de él sin demasiada dificultad.

1.3.5.2 Spring Security Framework

Spring Security es un framework escrito en Java que hace uso de las ventajas que brinda Spring Framework, como por ejemplo las inyecciones de dependencia. Spring Security utiliza filtros, que son los encargados de interceptar las peticiones que se realizan a la aplicación web, estos son conocidos como servlets. De esta forma se provee autenticación y autorización avanzada, permitiendo verificar la identidad de los usuarios y controlar el acceso de los mismos a los recursos de la aplicación. (11)

Spring Security está conformado por cinco componentes claves:

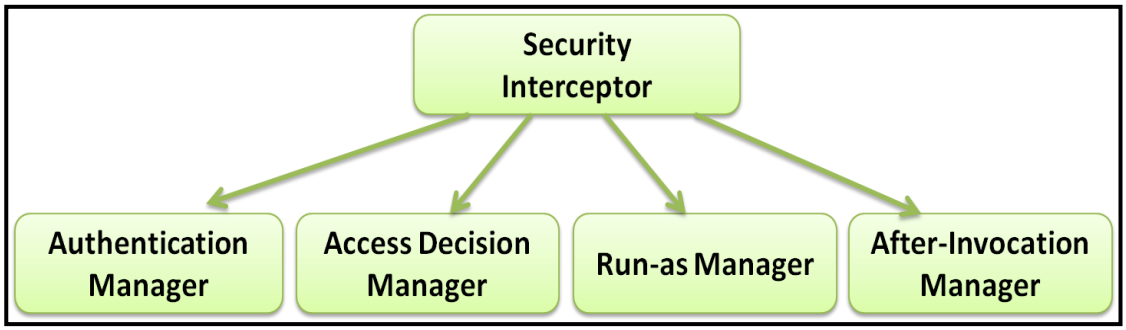

Figura 1.1. Elementos fundamentales de Spring Security.

Security Interceptor: Este componente se encarga de interceptar el acceso a los recursos para implementar la seguridad y delegar la autoridad de administrarla a los demás componentes o managers.

Authentication Manager: Es el responsable de identificar si el usuario que intenta acceder al sistema tiene acceso al mismo.

Access Decision Manager: Este es el que realiza el proceso de autorización; decide si se puede acceder a un recurso, en dependencia de la información de la autenticación y de los atributos de seguridad asociados a cada recurso.

Run-as Manager: Reemplaza la autenticación existente por una que da completo acceso a los recursos del sistema.

After-Invocation Manager: Asegura que el usuario que ha realizado una petición al sistema, tenga permitido ver los datos que están siendo retornados de un recurso con seguridad.
Se selecciona este framework para mantener el control de acceso a la aplicación ya que resulta fácil de integrar con el framework spring.

1.3.5.3 Hibernate Framework

Hibernate es un framework específicamente diseñado para los entornos que se desarrollan en Java. Utiliza una técnica conocida como Mapeo Relacional de Objetos, que es la encargada de establecer una conexión entre la representación de un objeto en Java y una base de datos relacional, a través del lenguaje HQL (del inglés Hibernate Query Language). Esta técnica permite en gran medida el ahorro de tiempo en proyectos productivos, sobre todo porque abstrae al programador del lenguaje SQL (del inglés System Query Language) específico de cada motor de base de datos. (12)

Se decide emplear Hibernate para el acceso a la base de datos ya que el mismo emplea la técnica de Mapeo Relacional de Objetos lo que facilita y agiliza la labor de los desarrolladores con la base de datos.

1.3.5.4 OpenLayers Framework

OpenLayers es una herramienta de código libre que es ejecutada del lado del cliente. Está diseñada para trabajar específicamente con mapas y es compatible con la mayoría de los navegadores web. El framework mencionado funciona íntegramente del lado del cliente, no es necesario ninguna configuración específica o software del lado del servidor para ser ejecutado. OpenLayers ha alcanzado un alto nivel de madurez debido a la gran cantidad de desarrolladores que colaboran entre sí, formando parte activa de la comunidad de este framework. (13)
Se empleará OpenLayers ya que es una herramienta de código libre para la visualización de mapas del lado del cliente y por la compatibilidad que presenta con la mayoría de los navegadores web.

1.3.5.5 Jquery Framework

Jquery es una librería JavaScript que se encuentra bajo licencia GPL, y se encarga de simplificar la interacción entre un documento HTML (del inglés HyperText Markup Language) más específicamente el DOM (del inglés Document Object Model) y un fichero JavaScript. Además facilita la manipulación de eventos en el navegador, animaciones e interacciones con AJAX (del inglés Asynchrony JavaScript and XML).

Las principales características por las que se decidió usar este framework se exponen a continuación:

- ✓ **Ligero:** Consta de un solo fichero, de aproximadamente 114Kb en la versión mínima.
- ✓ **Compatible con navegadores:** Abstrae al programador de las particularidades de cada navegador web.
- ✓ **Comunidad**: Existe un gran número de usuarios y programadores que son parte de la comunidad y que contribuyen de forma activa al desarrollo de este framework.
- ✓ **Documentado**: Su API (del inglés Application Programming Interface) está completamente documentada, incluyendo además ejemplos de código. (14)

1.4 Metodología de desarrollo

FDD (del inglés Feature Driven Development) es una metodología ágil para el desarrollo de sistemas, no hace énfasis en la obtención de los requerimientos sino en cómo se realizan las fases de diseño y construcción. FDD se preocupa por la calidad, por lo que incluye un monitoreo constante del proyecto.

Propone tener etapas de cierre cada dos semanas máximo, por lo que posibilita obtener resultados periódicos y tangibles.

Un proyecto basado en FDD se divide en cinco fases:

Figura 1.2. Proceso FDD.

- ✓ **Desarrollo de un modelo global**: Cuando comienza el desarrollo, los expertos del dominio están al tanto de la visión, el contexto y los requerimientos del sistema a construir. Se divide el dominio global en áreas que son analizadas detalladamente. Los desarrolladores construyen un diagrama de clases o de objetos por cada área y se construye un modelo global del sistema.
- ✓ **Construcción de una lista de funcionalidades**: Se elabora una lista de funcionalidades que resuma la funcionalidad general del sistema; la lista es elaborada por los desarrolladores y es evaluada por el cliente y se divide en subconjuntos según la afinidad y la dependencia de las funcionalidades, la lista es finalmente revisada por los usuarios y los responsables para su validación y aprobación.
- ✓ **Planeación por funcionalidad**: Se procede a ordenar los conjuntos de funcionalidades conforme a su prioridad y dependencia, y se asigna a los programadores jefes.
- ✓ **Diseño por funcionalidad y Construcción por funcionalidad**: Se selecciona un conjunto de funcionalidades de la lista, se procede a diseñar y construir la funcionalidad mediante un proceso iterativo. Una iteración puede tomar de unos pocos días a un máximo de dos semanas. El proceso iterativo incluye inspección de diseño, codificación, integración e inspección de código.

1.5 Herramientas

1.5.1 Entorno Integrado de Desarrollo. Spring Tool Suit 2.8

Spring Tool Suite es un IDE (del inglés Environment Development Interface) basado en Eclipse, que provee el mejor entorno de desarrollo para soluciones empresariales basadas en Spring Framework. Este IDE trae incluido las últimas actualizaciones de Java, Spring, y de Eclipse. Se encuentra disponible de forma gratuita para desarrolladores, y no establece un tiempo límite de uso. (15)

Principales características por las que se decidió usar Entorno de desarrollo son:

- ✓ Asistente para creación de XML.
- ✓ Soporte para anotaciones Spring.
- ✓ Ofrece avanzado completamiento de código, validaciones y correcciones rápidas (quick-fix support) para aplicaciones en Spring.
- ✓ Soporte para los servidores más comunes de Java EE (del inglés Enterprise Edition).
- ✓ Trae por defecto instalado los componentes más usados de Java. (15)

1.5.2 Servidor Web. Apache Tomcat 6.0.26

Apache Tomcat es un servidor web de código libre. Es desarrollado en un entorno abierto y participativo, y se encuentra publicado bajo la licencia Apache versión 2. (16)

El servidor web se acopla de forma sencilla al IDE Spring Tool Suit y desde este resulta sencillo manipular las opciones de configuración las cuales se muestran a través de una interfaz. Otra de las facilidades que nos brinda la combinación del este servidor con el IDE a usar, es la posibilidad de depurar el código de la aplicación, permitiendo de esta forma poder identificar y corregir errores de forma más fácil. Apache Tomcat es capaz de funcionar en todos los sistemas operativos que dispongan de la máquina virtual de Java.

1.5.3 Sistema Gestor de Base de Datos. PostgreSQL 8.4.0

PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado. (17)

Algunas de las principales características de postgres son:

Límite	Valor
Máximo tamaño base de dato	Ilimitado (Depende del sistema de almacenamiento)
Máximo tamaño de tabla	32 TB
Máximo tamaño de fila	1.6 TB
Máximo tamaño de campo	1 GB
Máximo número de filas por tabla	Ilimitado
Máximo número de columnas por tabla	250 - 1600 (dependiendo del tipo)
Máximo número de índices por tabla	Ilimitado

Tabla 1.1. Principales características de PostgresSQL.

1.5.4 GeoServer 2.1.3

GeoServer es un servidor de software de código abierto escrito en Java que permite a los usuarios compartir y editar datos geoespaciales. Al ser un proyecto impulsado por una comunidad, GeoServer es desarrollado, probado y apoyado por un grupo diverso de desarrolladores y organizaciones de todo el mundo.

GeoServer es la implementación de referencia del Open Geospatial Consortium (OGC), cuenta también con un alto rendimiento certificado de cumplimiento Web Map Service (WMS).

1.5.5 Front-End AVL 1.0

El Front-End AVL se conecta al servidor AVL a través del puerto socket y mediante comandos de comunicación obtiene la información geográfica de cada uno de los dispositivos que cuentan con tecnología GPS. Además se debe configurar el formato en que se recibirá la información geográfica como puede ser, en formato NMEA. Además permite la creación de tareas TIG para compartir la información GPS con los Sistemas de Información Geográfica que consuman de este servicio (18).

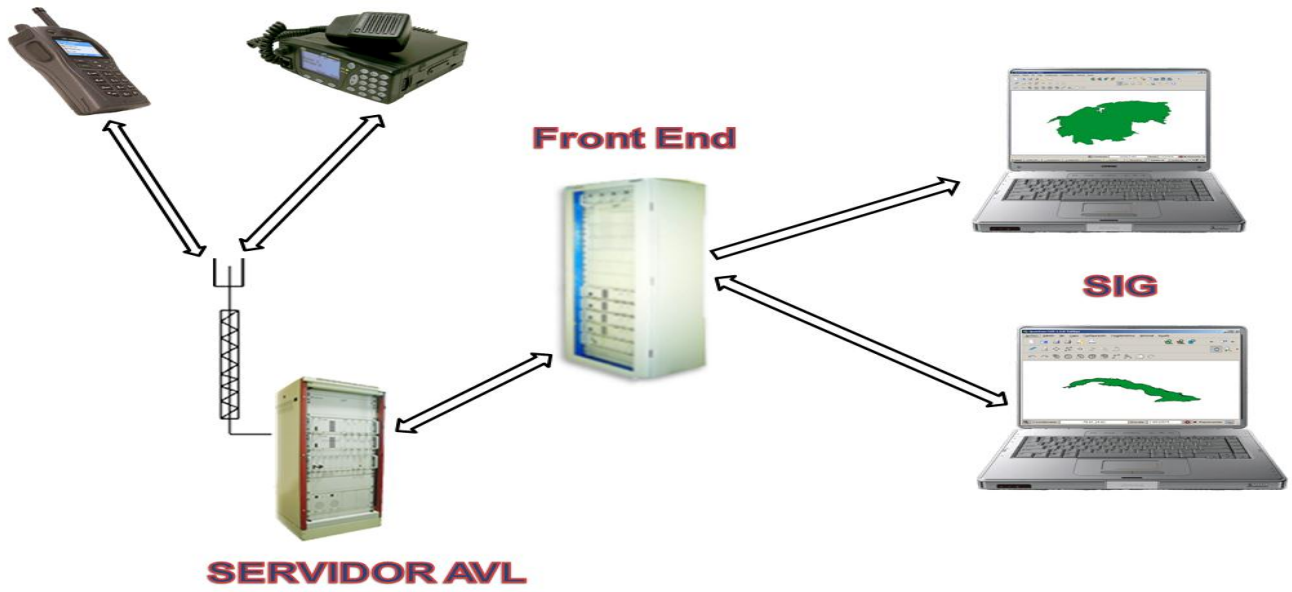

Figura 1.3. Infraestructura AVL.

1.6 Herramienta de Modelado. Visual Paradigm 8.0

Visual Paradigm es una herramienta diseñada para ayudar al desarrollo de software, esta cuenta con soporte para estándares de la industria como: UML (del inglés Unified Modeling Languaje). Ofrece una

amplia gama de herramientas muy útiles para equipos de desarrollado de software facilitando la captura de requisitos, planificación de proyecto, modelado de clases, modelado de base de datos. (19)

1.7 Conclusiones Parciales

En el estudio realizado en el presente capítulo sobre las características y funcionalidades de las soluciones existentes, se llega a la conclusión de que estos sistemas no satisfacen la problemática actual. Debido a esto surge la necesidad de desarrollar un sistema que realice un control y monitoreo de flotas durante su recorrido. Para la confección de la solución se empleará como lenguaje de programación Java y como frameworks, Spring 3.0.1, Spring Security 3.0.1, Hibernate 3.0 para el acceso a datos, OpenLayers 2.10 para la visualización de mapas y jQuery 1.8 para la interacción con el usuario. Además se empleará como entorno de desarrollo Spring Tool Suit, como lenguaje de modelado UML haciendo uso de la herramienta Visual Paradigm 8 y como metodología de desarrollo se empleará FDD.

CAPÍTULO 2: CARACTERÍSTICAS DEL SISTEMA

2.1 Introducción

En este capítulo se abordan los aspectos relacionados con el proceso del negocio, se describe la propuesta del sistema a automatizar y se define el modelo de dominio para representar la visión del proceso de control de flotas. Además se establecen los requisitos funcionales y no funcionales con los que debe cumplir el sistema, así como el diagrama de funcionalidades del sistema.

2.2 Problemática

En Cuba se emplea un sistema de control de flota el cual verifica el comportamiento de las flotas una vez finalizado el recorrido de los vehículos, lo que impide estar al tanto de todo lo que ocurre durante el trayecto de las flotas hacia su destino. Si ocurriera, por ejemplo, algún desperfecto técnico en la vía, se perderían valiosos minutos u horas entre la ocurrencia del suceso y la respuesta al mismo.

En la Universidad de las Ciencias Informáticas existe un centro de desarrollo especializado en el trabajo con mapas, este centro cuenta con un proyecto para la creación de un sistema de control de flotas el cual aún se encuentra en fase de desarrollo y no se cuenta con información de las características y funcionalidades del mismo.

Por lo antes expuesto resulta necesario crear un sistema que permita visualizar en un mapa la posición y comportamiento de los vehículos durante su trayecto.

2.3 Objeto de automatización

Por lo descrito en el epígrafe anterior surge la necesidad de automatizar el proceso de control de flotas durante su recorrido, así como la gestión de rutas y áreas, que luego podrían ser asignadas a las flotas. De igual forma resulta necesario mantener un registro donde se almacene la información de los vehículos que han incurrido en alguna infracción durante su trayecto.

2.4 Propuesta del sistema

Se propone crear un sistema web que permita monitorear y analizar el comportamiento de los vehículos durante su recorrido. El mismo estará desplegado en un servidor web, específicamente Apache Tomcat v6.0.26, el cual podrá ser accedido por los supervisores desde una computadora cliente mediante el uso de un navegador web. Para visualizar la posición de los vehículos, el sistema hará uso de la aplicación externa Front-End AVL 1.0 (del inglés Automatic Vehicle Location), la cual proporcionará información GPS de las flotas en formato KML o GPRMC; esta información está compuesta por la longitud, latitud, altitud y

la hora y fecha en que se tomó la posición del vehículo, para su representación en un mapa, que será obtenido a través de un servicio WMS (del inglés Web Map Service) publicado por el servidor de mapas GeoServer en su versión 2.1.3. Además permitirá a los supervisores la creación de áreas y rutas, las que definirán el comportamiento de las flotas.

El sistema será capaz de detectar irregularidades en el comportamiento de los vehículos, tales como exceso de velocidad y desvíos de las rutas asignadas. Estas irregularidades quedarán registradas en una base de datos, almacenándose el nombre del vehículo, el tipo de infracción y la hora en la que incurrió en la falta, para su posterior consulta.

En la siguiente imagen se muestra una descripción visual de la propuesta del sistema:

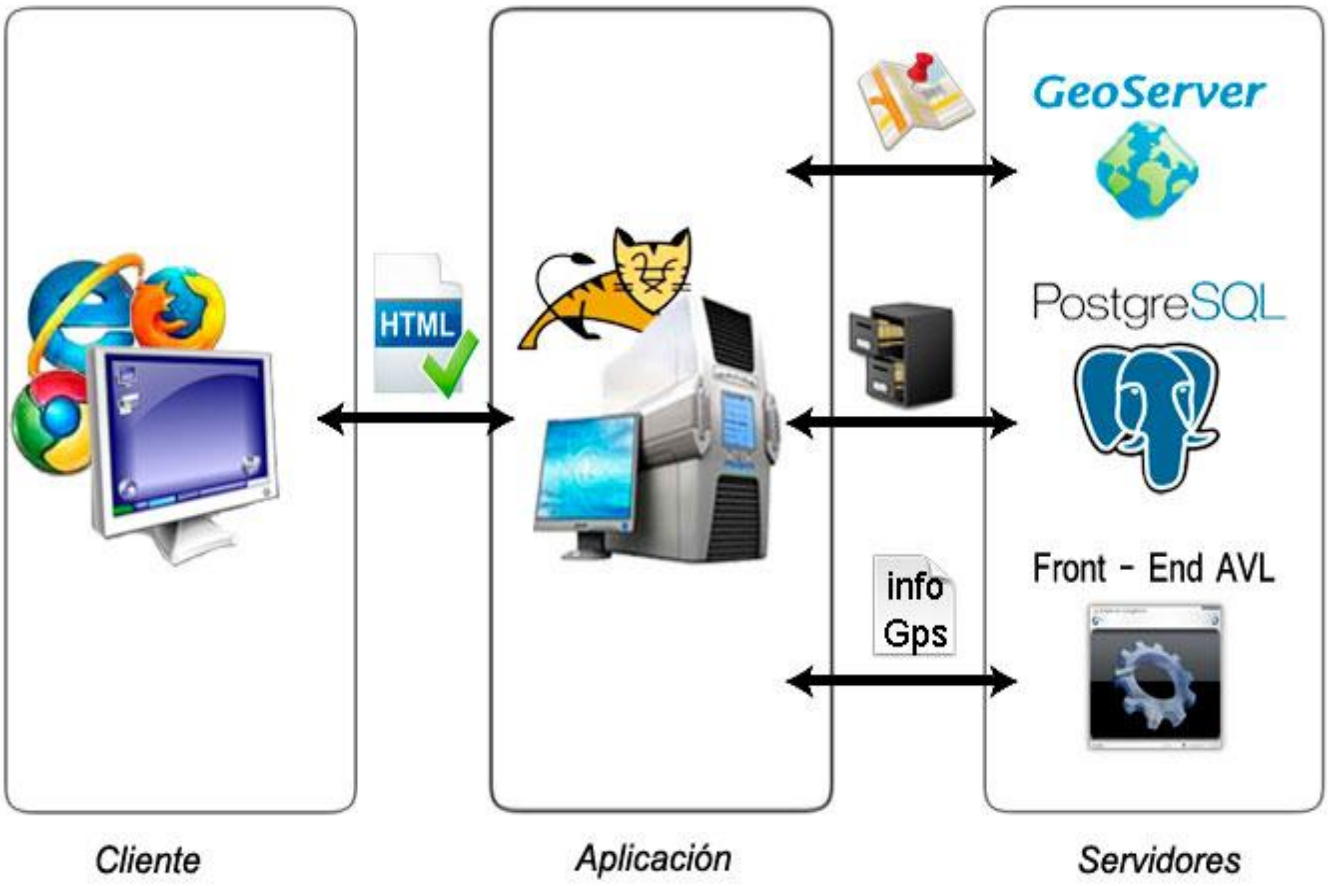

Figura 2.1. Propuesta del sistema.

2.5 Modelo de Dominio

Para realizar un correcto diseño del sistema es preciso realizar un Modelo de Negocio o Modelo de Dominio para capturar y enunciar la visión del proceso. (20)

Para la realización de un Modelo de Negocio es necesario haber llevado a cabo una identificación de los procesos existentes, que estos tengan un alto nivel de estructuración y cuenten con la existencia de flujos de información bien definidos. En cambio, el Modelo de Dominio posibilita la comprensión de los conceptos con los que trabajan los usuarios y con los que deberá trabajar la aplicación, cuando no son claramente visibles los procesos del negocio.

Para la construcción del sistema se considera apropiado la realización de un modelo de dominio, ya que no existe un cliente o socio que pueda explicar claramente los procesos del negocio, por lo que estos no quedan bien identificados. Además no se cuenta con experiencia en el trabajo con sistemas de control de flotas.

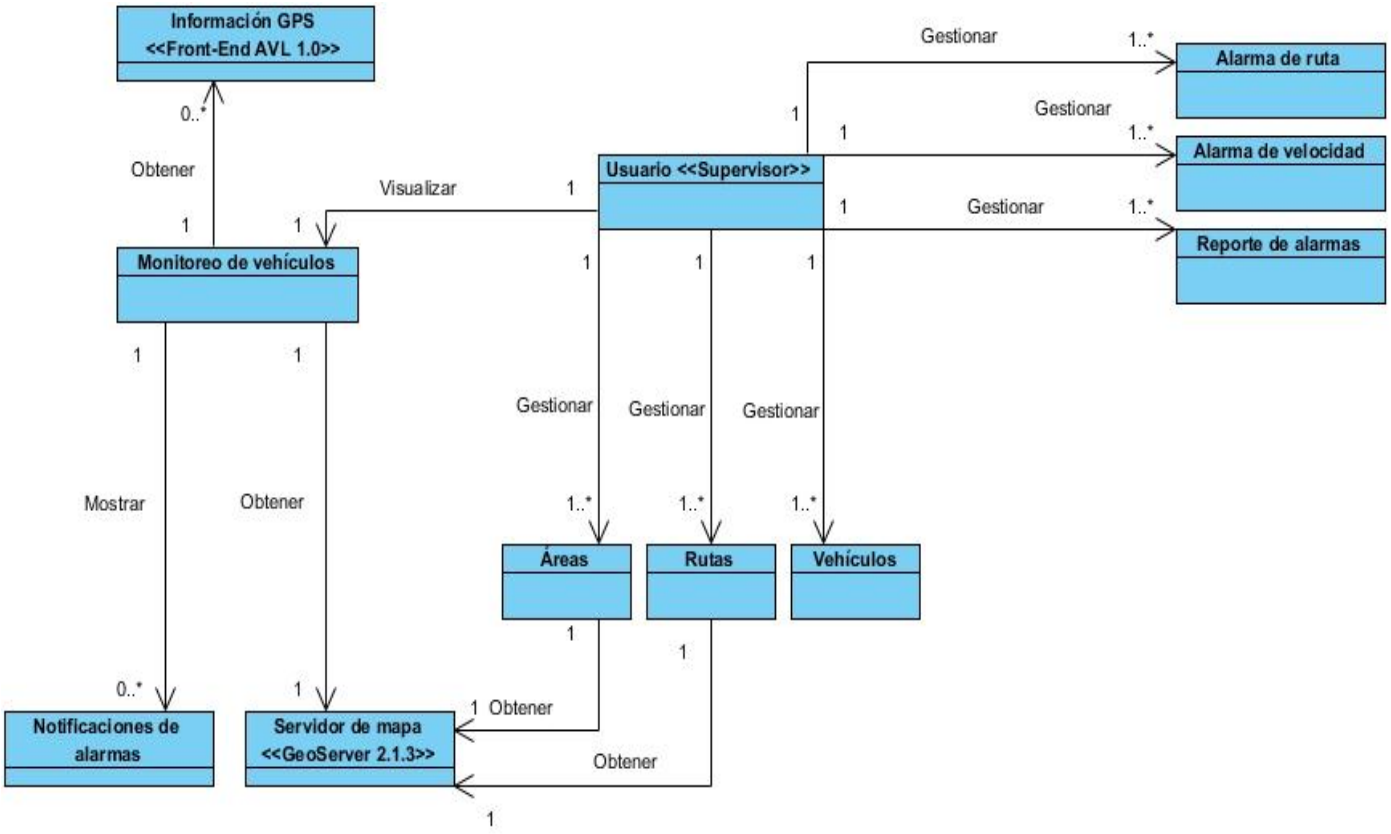

Figura 2.2. Modelo de dominio.

- **Supervisor:** Encargado de realizar el control sobre los vehículos.
- **Alarma de ruta:** Representa los datos de la ruta asignada a los vehículos.
- **Alarma de velocidad:** Representa los datos de velocidad establecidos para cada vehículo.
- **Monitoreo de vehículos:** Representa la información y posición de los vehículos activos.
- **Notificaciones de alarmas:** Representa la información que será mostrada al supervisor en caso detectar algún comportamiento inadecuado por parte de algún vehículo.
- **Área:** Representa la información de las áreas almacenadas en el sistema.
- **Rutas:** Representa la información de las rutas almacenadas en el sistema.
- **Vehículos:** Representa la información de los vehículos almacenadas en el sistema.
- **Historial:** Representa los datos almacenados de los vehículos durante un periodo de tiempo.
- **Servidor de mapa:** Representa el servidor GeoServer 2.1.3, el cual proporciona los mapas para visualizar la ubicación de las flotas.
- **Información GPS:** Representa la información GPS de las flotas a través de la aplicación externa Front-End AVL 1.0.

2.6 Especificación de las funcionalidades

2.6.1 Requerimientos funcionales (RF)

Los requerimientos funcionales son condiciones o capacidades que el sistema debe cumplir. A continuación se enumeran los que han sido identificados.

RF 1: Autenticar usuario.

RF 2: Visualizar y monitorear vehículos en un mapa.

RF 3: Gestionar ruta.

3.1 Adicionar ruta.

3.2 Eliminar ruta.

3.3 Listar rutas.

RF 4: Gestionar área.

4.1 Adicionar área.

4.2 Eliminar área.

4.3 Listar áreas.

RF 5: Configurar alarma de ruta para vehículos.

5.1 Activar alarma de ruta.

5.2 Desactivar la alarma de ruta.

5.3 Modificar la alarma de ruta.

RF 6: Configurar alarma de velocidad para vehículos.

6.1 Activar alarma de velocidad.

6.2 Desactivar la alarma de velocidad.

6.3 Modificar la alarma de velocidad.

RF 7: Visualizar rutas en un mapa.

RF 8: Visualizar áreas en un mapa.

RF 9: Gestionar reporte de alarmas.

9.1 Adicionar reportes.

9.2 Buscar reportes.

9.3 Listar reportes.

RF 10: Mostrar notificaciones de alarma.

RF 11: Gestionar Vehículos.

11.1 Adicionar vehículo.

11.2 Buscar vehículo.

11.3 Eliminar vehículo.

11.4 Listar vehículos.

2.6.2 Requerimientos no funcionales (RNF)

Los requisitos no funcionales son aquellos, que están relacionados directamente con el funcionamiento del producto a desarrollar, permite que el resultado del software sea atractivo, usable, rápido, es decir, determina cómo ha de comportarse y qué cualidades debe tener el mismo. Para el desarrollo del sistema se han determinado los siguientes requisitos no funcionales.

RNF 1: Software

- ✓ **Servidor:** Se debe contar con la Máquina Virtual de Java versión 7.3 instalada, Apache Tomcat versión 6.0.26 y PostgresSQL en su versión 8.4.
- ✓ **Sistema Operativo:** Podrá instalarse en cualquier sistema operativo que cuenta con la máquina virtual de Java 7.3.

- ✓ **Cliente:** Debe disponer en la computadora del cliente con un navegador con soporte para Javascript, se recomiendan lo navegadores (Mozilla Firefox 18 o superior, Internet Explorer 8, Google Chrome 25 o superior).

RNF 2: Hardware

Debe existir una conexión de red basada en TCP/IP para establecer comunicación entre el sistema web y la aplicación proveedora de información GPS, así como con los servidores de mapa y base de datos. Además resulta necesaria esta conexión para comunicar la computadora del cliente con el servidor donde se encuentra ubicada la aplicación.

RNF 3: Seguridad.

El sistema debe proteger la confidencialidad de la información referente a las flotas, a esta información solo tendrá acceso el personal autorizado a través de un proceso de autentificación del usuario mediante el uso de contraseñas.

RNF 4: Portabilidad

El sistema deberá ser multiplataforma.

RNF 5: Diseño

Se empleará como lenguaje de programación Java; para el desarrollo del sistema se empleará el framework Spring 3.0.1 y para el acceso a la información almacenada en la base de datos se empleará Hibernate 3.0.

Como metodología de desarrollo se empleará FDD, haciendo uso del lenguaje de modelación UML y como herramienta para la elaboración de los diferentes diagramas Visual Paradigm for UML en su versión 8.0. Para almacenar la información del sistema se utilizará como servidor de base de datos PostgresSQL versión 8.4.

2.7 Definición de funcionalidades del sistema

2.7.1 Definición de los actores del sistema

Actores	Justificación

Supervisor	Es el encargado del monitoreo de los vehículos, de la creación de rutas, áreas y de la asignación de alarmas a las flotas. Además también puede consultar los reportes de vehículos que han cometido alguna infracción.

Tabla 2.1 Definición de los actores.

2.7.2 Diagrama de funcionalidades

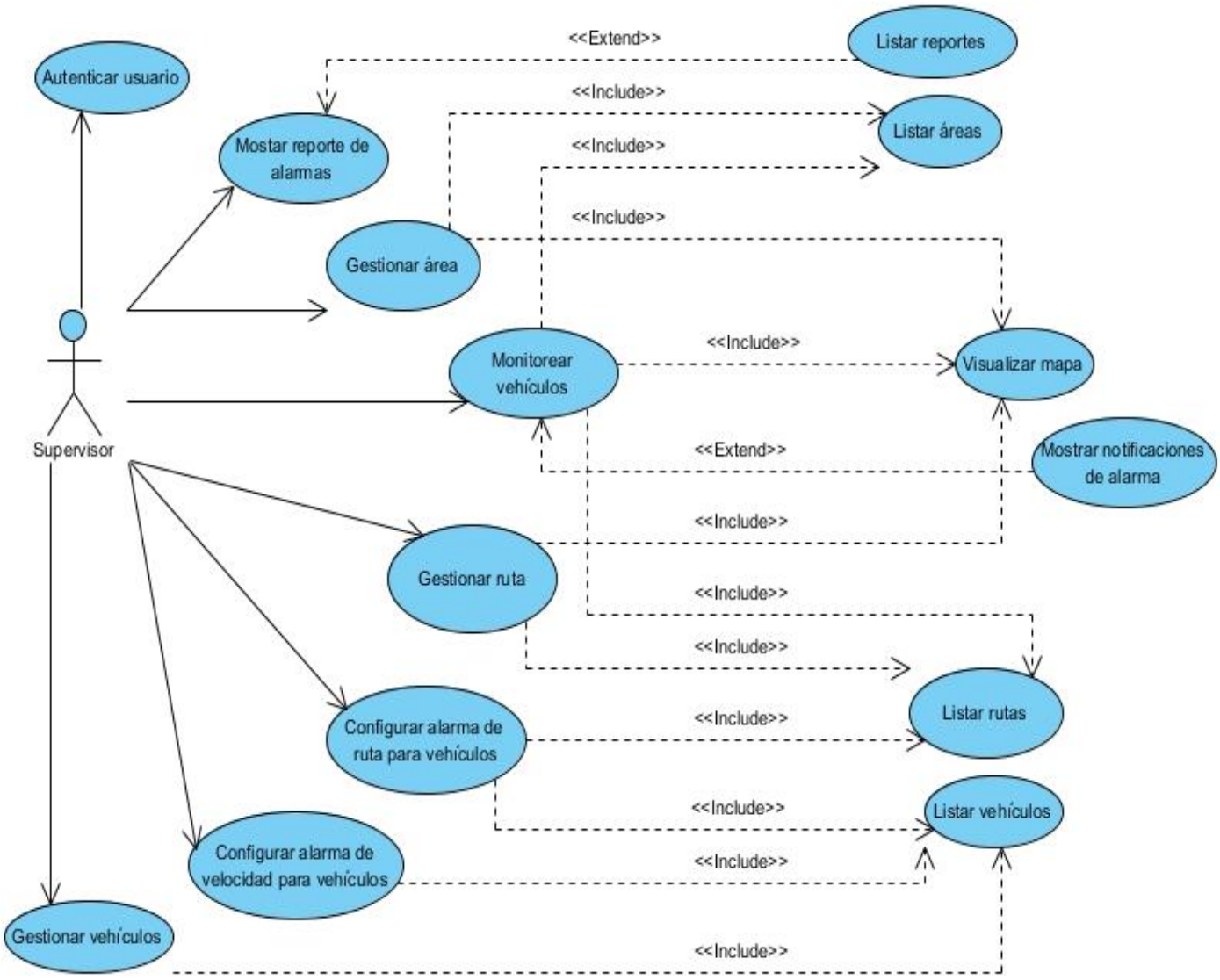

Figura 2.3. Diagrama de funcionalidades.

2.8 Conclusiones Parciales

Durante este capítulo se presentó la propuesta del sistema la cual constará de tres partes lógicas siendo la más importante la que contiene el servidor de aplicación Apache Tomcat en el que se encontrará desplegado el sistema web. Se definieron los requerimientos funcionales y los no funcionales con los que debe cumplir la solución a desarrollar. Se describió el modelo de dominio para facilitar la comprensión de los conceptos con los que trabajará la aplicación. Además se elaboró el diagrama de funcionalidades y quedó definido el actor que interviene en el sistema.

CAPÍTULO 3: DISEÑO DEL SISTEMA

3.1 Introducción

Basándose en el diagrama de funcionalidades, en este capítulo se describirán los procesos a través de los cuales se llevará a cabo el diseño y la implementación del sistema. También se mostrará el modelo de datos, los patrones de diseño que fueron utilizados en el desarrollo de la aplicación, la arquitectura del sistema, y el diagrama de secuencia para el diseño de las funcionalidades.

3.2 Arquitectura

La arquitectura es la estructura jerárquica de los componentes del programa (módulos), la manera en que los componentes interactúan y la estructura de datos que van a utilizar los componentes. Se pueden generalizar para representar los elementos principales del sistema y sus interacciones. (20)

3.2.1 Patrones

Los patrones de diseño son modelos de trabajo enfocados a dividir un problema en partes, de modo que sea posible abordar cada una de ellas por separado para simplificar su resolución. (21)

3.2.1.1 Patrón Arquitectónico MVC (del inglés Model View Controller)

El patrón MVC, alude a la división en tres partes de una aplicación, las cuales son:

- ✓ **Modelo**: Contiene la información que se transfiere entre la controladora y la vista. También almacena los datos del modelo de negocio, como operaciones, transformaciones y reglas para la manipulación de los datos.
- ✓ **Vista**: Las vistas son usadas para mostrar la información del sistema a través de interfaces de usuario.
- ✓ **Controlador**: Es el encargado de procesar las peticiones, realizar las operaciones con las clases modelo y devolver la vista que será mostrada al usuario. (22)

Cada parte de la arquitectura MVC está definida solamente con un propósito, por ejemplo la lógica encargada de manipular los datos está contenida solamente en el modelo, la lógica para mostrar los datos es solamente manejada por la vista, y el código que se encarga de manejar las peticiones de los usuarios y devolver las vistas especificas está contenido en el controlador. Esta división entre los componentes de la aplicación permite organizar todos sus elementos, facilitando las tareas de mantenimiento y de extensión de funcionalidades en cualquier momento del ciclo de vida del software. (22)

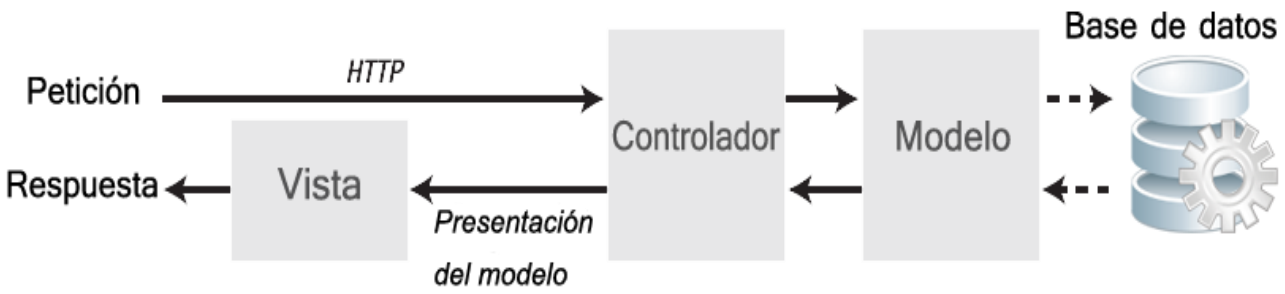

Figura 3.1. Arquitectura Modelo Vista Controlador.

A continuación se muestra un conjunto de imágenes, las cuales reflejan el conjunto de clases del sistema que pertenecen a cada capa de la arquitectura MVC.

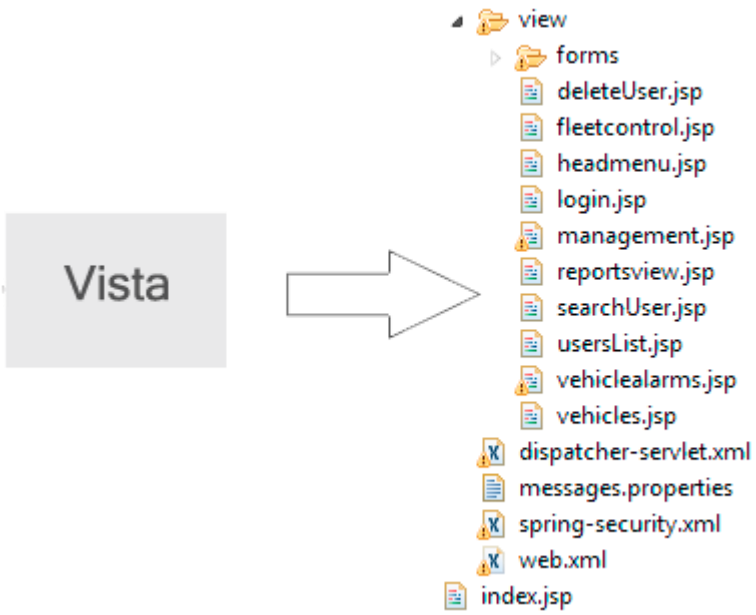

Figura 3.2. Paquete de clases del sistema que pertenecen a la capa Vista en la arquitectura MVC.

Figura 3.3. Paquetes de clases del sistema que pertenecen a la capa Modelo en la arquitectura MVC.

Figura 3.4. Paquete de clases del sistema que pertenecen a la capa Control en la arquitectura MVC.

3.3 Patrones de diseño

Los patrones de diseño benefician en gran medida el desarrollo y la calidad del software. Permite evitar la reiteración en la búsqueda de soluciones a problemas ya conocidos y solucionados anteriormente, permite además formalizar un vocabulario común entre diseñadores y estandarizar el modo en que se realiza el diseño. Durante el desarrollo de la solución se han usado los patrones de diseño GRASP (del inglés General Responsibility Assignment Software Patterns).

3.3.1 Patrones de diseño GRASP

3.3.1.1 Experto

Es un patrón que se usa más que cualquier otro al asignar responsabilidades; es un principio básico que suele utilizarse en el diseño orientado a objetos. Da origen a diseños donde el objeto de software realiza las operaciones que normalmente se aplican a lo que se desea representar, por lo que ofrece una analogía con el mundo real.

Con la utilización de este patrón se conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida, alentando la creación de clases sencillas y más cohesivas que son fáciles de comprender y mantener.

La siguiente imagen muestra la utilización del patrón Experto en el desarrollo del sistema:

```
@Repository("routeDao")
public class RouteDao implements IRouteDao{

    private SessionFactory sessionFactory;

    public void AddRoute(Route route) {

    public Route GetRouteById(String id) {

    public List<Route> GetAllRoutesWithPoints (){

    public List<Route> GetAllRoutes() {

    public void DeleteRouteById(String id) {

    public void DeleteRouteByName(String name) {

    public Route GetRouteByName(String name) {
```

Figura 3.1. Ejemplo de Patrón Experto.

3.3.1.2 Creador

El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. Brinda soporte a un bajo acoplamiento lo que supone menos dependencias respecto al mantenimiento y mejores oportunidades de reutilización.

La siguiente imagen muestra la utilización del patrón Creador en el desarrollo del sistema:

```
@Controller
@RequestMapping (value = "/management")
public class ManagementController
{
    @Autowired
    private IRouteService routeService;

    @Autowired
    private IGrateService grateService;

    @Autowired
    IVehicleService vehicleService;

    @Autowired
    private IPointGpsService pointService;
```

Figura 3.2. Ejemplo de Patrón Creador.

3.3.1.3 Bajo Acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo (o débil) acoplamiento no depende de muchas otras clases. Una clase con alto (o fuerte) acoplamiento recurre a muchas otras. Este tipo de clases no es conveniente, ya que presentan los siguientes problemas:

- ✓ Los cambios de las clases afines ocasionan cambios locales.
- ✓ Son más difíciles de entender cuando están aisladas.
- ✓ Son más difíciles de reutilizar porque se requiere la presencia de otras clases de las que dependen.

Con el uso de este patrón los componentes no se afectan por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar.

3.3.1.4 Alta Cohesión

En el diseño orientado a objetos, la cohesión es una medida de cuan relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Las clases con baja cohesión a menudo representan un alto grado de abstracción o han asumido responsabilidades que deberían haber delegado a otros objetos. No conviene este tipo de clases pues presentan los siguientes problemas:

- ✓ Son difíciles de comprender.
- ✓ Son difíciles de reutilizar.
- ✓ Son difíciles de conservar.
- ✓ Son delicadas: las afectan constantemente los cambios.

Una clase con alta cohesión realiza poco trabajo, colaborando con otros objetos para compartir el esfuerzo si la tarea es grande, generándose a menudo un bajo acoplamiento. La ventaja de una gran funcionalidad es que soporta una mayor capacidad de reutilización. Con el uso de este patrón se mejora la claridad y la facilidad con que se entiende el diseño, además se simplifican las tereas de mantenimiento y de mejoras a las funcionalidades.

3.3.1.5 Controlador

Un controlador es un objeto de interfaz no destinada al usuario. Se asocia a operaciones del sistema generadas por un actor externo o en respuesta a eventos del sistema. El controlador es el encargado de asignar la responsabilidad del manejo de los eventos de un sistema a una clase que represente una de las siguientes opciones:

- ✓ La empresa u organización global.
- ✓ Un manejador artificial de todos los eventos del sistema de un caso de uso
- ✓ Utilice la misma clase de controlador con todos los eventos del sistema en el mismo caso de uso.

La siguiente imagen muestra la utilización del patrón Controlador en el desarrollo del sistema:

```
@Controller
@RequestMapping (value = "/management")
public class ManagementController
{
    public @ResponseBody List<JsonErrorResponse> AddRouteAjax (
    public @ResponseBody List<JsonErrorResponse> AddGrateAjax (
    public @ResponseBody List<PointGps> GetRouteWithPoints (@RequestParam String routeId)
    public @ResponseBody List<PointGps> GetGrateWithPoints (@RequestParam String grateId)
    public @ResponseBody String DeleteRouteById (@RequestParam String routeId)
    public ModelAndView RoutesView (@ModelAttribute ("route")Route route,
    public @ResponseBody String DeleteGrateById (@RequestParam String grateId)
    public String ViewFleetControl ()
```

Figura 3.3. Ejemplo de Patrón Controlador.

3.3.2 Patrones GoF (Gang of Four)

3.3.2.1 Patrón DI (Dependency Injection)

La inyección de dependencia es un patrón de diseño orientado a objetos, en el que se inyectan los objetos a una clase en lugar de ser la propia clase quien cree los objetos. Con este patrón de diseño la entidad que coordina cada objeto en el sistema es la encargada de realizar el trabajo en el momento de instanciar el objeto invirtiéndose la responsabilidad en cuanto a la manera en que un objeto obtiene la referencia a otro objeto. Al aplicar este patrón las clases quedan independientes unas de otras incrementándose la reutilización y extensibilidad de las aplicaciones.

La siguiente imagen muestra la utilización del patrón Inyección de Dependencia en el desarrollo del sistema:

```
@Controller
@RequestMapping(value="/vehiclealarms")
public class VehicleAlarmsController {

    @Autowired
    IVehicleService vehicleService;

    @Autowired
    IAlarmService alarmService;

    @Autowired
    IRouteService routeService;
```

Figura 3.4.Ejemplo de Patrón Inyección de Dependencia.

3.3.2.2 Singleton

El patrón singleton asegura que no exista más de una instancia de una clase en el sistema, proveyendo un punto global de acceso para la creación del objeto. Evitándose de esta forma que si clientes distintos precisan referenciar a un mismo elemento, no se cree una nueva instancia de este, si no que se use la misma que fue creada con anterioridad (23). Spring Framework 3.0 hace uso de este patrón para todos los beans definidos en los archivos de configuración del sistema.

La siguiente imagen muestra la utilización del patrón Singleton en los beans definidos en el archivo de configuración dispatcher-servlet.xml del sistema:

```
<bean id="jspViewResolver"
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="viewClass"
        value="org.springframework.web.servlet.view.JstlView" />
    <property name="prefix" value="/WEB-INF/view/" />
    <property name="suffix" value=".jsp" />
</bean>

<bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <property name="driverClassName" value="${database.driver}" />
    <property name="url" value="${database.url}" />
    <property name="username" value="${database.user}" />
    <property name="password" value="${database.password}" />

</bean>

<bean id="sessionFactory"
    class="org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean">
    <property name="dataSource" ref="dataSource" />
    <property name="annotatedClasses">
        <list>
    </property>
    <property name="hibernateProperties">
        <props>
            <prop key="hibernate.dialect">${hibernate.dialect}</prop>
            <prop key="hibernate.show_sql">${hibernate.show_sql}</prop>
            <prop key="hibernate.hbm2ddl.auto">${hibernate.hbm2ddl.auto}</prop>
        </props>
    </property>
</bean>
```

Figura 3.5. Ejemplo de Patrón Singleton.

3.3.2.3 Template Method

El patrón Template Method define una estructura dentro de un algoritmo de una superclase, permitiendo que los pasos de este algoritmo definido en la clase padre, sean redefinidos en las subclases sin necesidad de tener que sobrescribir la operación entera. En la siguiente figura del sistema se muestra de la clase VehicleDao el método GetAllVehicles, para el cual no fue necesario invocar los métodos de abrir, ejecutar y cerrar conexión con la base de datos.

```java
@Repository("vehicleDao")
public class VehicleDao implements IVehicleDao{

    @Autowired
    SessionFactory sessionFactory;

    @Override
    public List<Vehicle> GetAllVehicle() {
        List<Vehicle> vehicles = sessionFactory.getCurrentSession().createCriteria(Vehicle.class).list();

        return vehicles;
    }
```

Figura 3.6. Ejemplo de Patrón Template Method.

3.4 Diagrama de paquetes

El diagrama de paquetes representa una colección de clases. Es usado para estructurar el modelo de diseño dividiéndolo en partes más pequeñas. Los paquetes de diseño deben usarse fundamentalmente como herramienta organizacional del modelo para agrupar elementos relacionados.

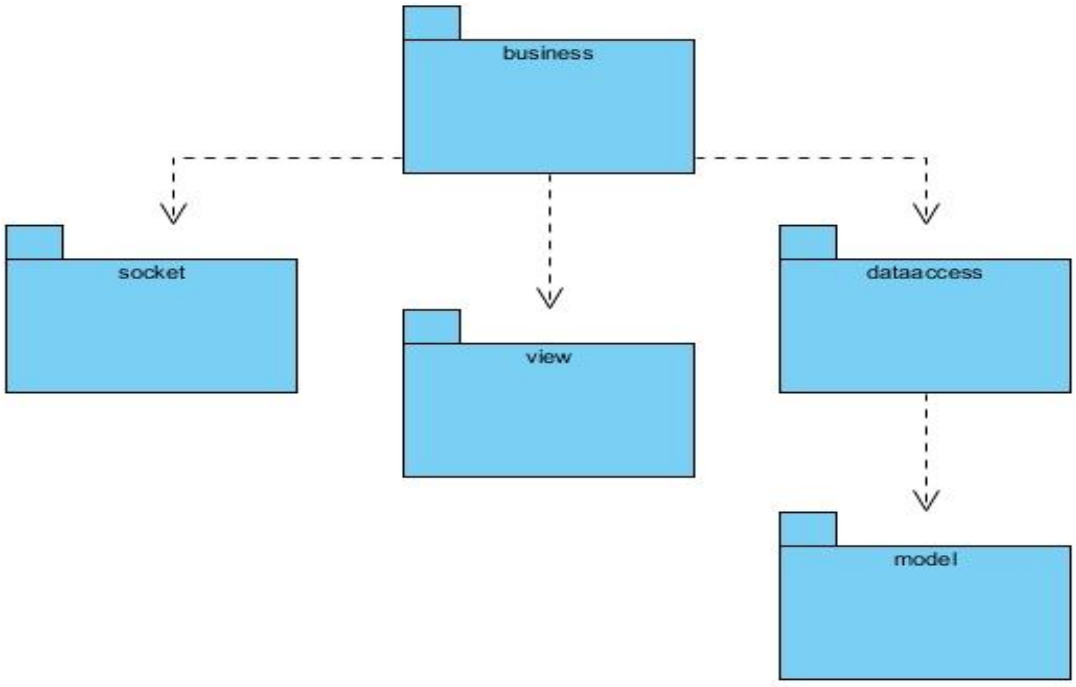

Figura 3.7. Diagrama de paquetes del sistema.

Business: Agrupa las clases controladores, las cuales reciben y dan respuesta a las peticiones que se realizan desde las vistas. En este paquete también se encuentran las clases relacionadas con el control de las flotas las cuales realizan el proceso de chequeo de alarmas. El paquete Business también hace uso de las funcionalidades incluidas en los paquetes Socket, View y DataAccess.

Socket: Engloba todo lo relacionado con la conexión de la aplicación al Front-End AVL a través del protocolo TCP/IP para la obtención de información GPS.

View: Contiene las interfaces que posibilitan la interacción de la aplicación con los usuarios.

DataAccess: Comprende las clases encargadas de realizar las peticiones a la base de datos, para la obtención y almacenamiento de la información. Este paquete hace uso de las funcionalidades incluidas en el paquete Model.

Model: Contiene las entidades que representan las tablas en la base de datos.

3.5 Diagramas de funcionalidades

3.5.1 Descripción de la funcionalidad "Monitorear vehículo"

La funcionalidad inicia cuando el supervisor accede a la página "fleetcontrol". Inicialmente, se envía una petición Ajax a la clase controladora "FleetControlController", para obtener un listado de las rutas y áreas disponibles para visualizarlas en el mapa. Luego, el sistema realiza peticiones Ajax por intervalos de un segundo que permiten actualizar la posición GPS de cada vehículo. La información GPS es obtenida a través de una conexión TCP/IP entre el sistema web y la aplicación Front-End AVL 1.0.

El sistema realiza, por intervalos de tiempo, un chequeo de las alarmas que le han sido asignadas a cada uno de los vehículos, comprobando su complimiento. En caso de ocurrir alguna incidencia se muestra al supervisor una notificación, indicando los datos específicos de la falta cometida, quedando registrada en la base de datos la infracción para su posterior consulta.

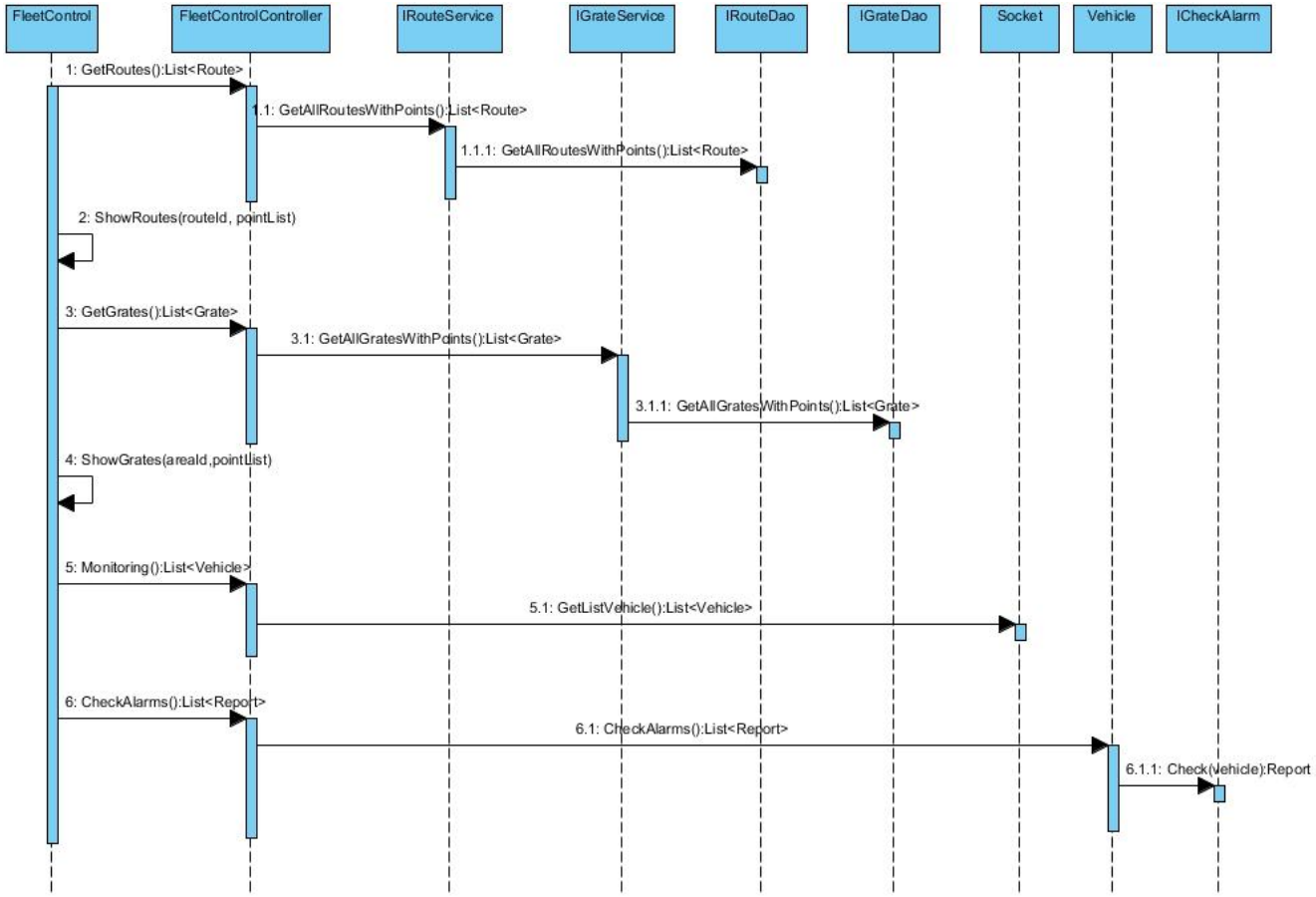

Figura 3.8. Diagrama de secuencia de la funcionalidad Monitorear vehículos.

3.5.2 Descripción de la funcionalidad "Configurar Alarma de Ruta"

La funcionalidad inicia cuando el supervisor accede a la página "vehiclealarm" y selecciona el vehículo al cual le desea asignar la alarma de ruta; luego elige la opción "Asignar alarma de ruta" y el sistema muestra un campo, el cual que contiene el nombre de las rutas que se encuentran disponibles, el supervisor selecciona la ruta que será asignada al vehículo especificado. Cuando el supervisor pulsa sobre el botón guardar, la vista envía una petición con el nombre "AddUpdateRouteAlarm" a la controladora "VehicleAlarmController" esta invoca el método "GetVehicleByPlate", el cual devuelve el vehículo que fue seleccionado en la vista. Luego se pasa a buscar la ruta con la llamada a los métodos "GetRouteByName" de las interfaces "IRouteService" e "IRouteDao", devolviendo la ruta que se desea adicionar al vehículo. El siguiente paso es agregar la nueva alarma a las que ya tiene asignadas el

vehículo, se logra a través de los métodos “AddAlarm” de las interfaces “IAlarmService” e “IAlarmDao”. El último paso es actualizar las relaciones del vehículo al cual se le asignó la nueva alarma de ruta, para esto se invocan los métodos “AddVehicle” de las interfaces “IVehicleService” e “IVehicleDao”.

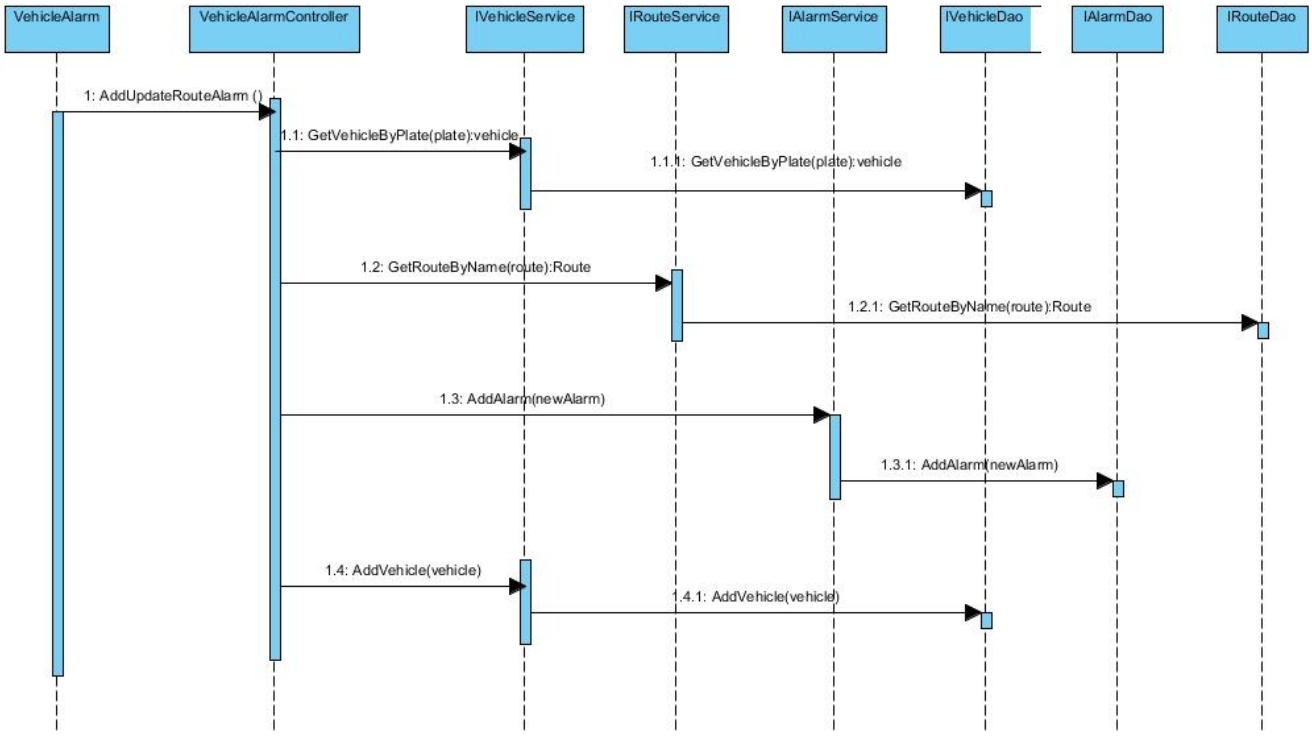

Figura 3.9. Descripción de la funcionalidad Configurar Alarma de Ruta.

3.5.3 Descripción de la funcionalidad “Configurar Alarma de Velocidad”

La funcionalidad inicia cuando el supervisor accede a la página “vehiclealarm” y selecciona el vehículo al cual le desea asignar la alarma de velocidad, luego elige la opción “Asignar alarma de velocidad”. Se muestra un campo de entrada de datos para que el supervisor establezca la velocidad máxima en kilómetros por hora (Km/h) con la que deberá cumplir el vehículo seleccionado. Cuando el supervisor pulsa sobre el botón guardar de la vista, esta realiza una petición con el nombre “AddUpdateSpeedAlarm” a la clase controladora “VehicleAlarmController” esta invoca el método “GetVehicleByPlate”, el cual devuelve el vehículo que fue seleccionado en la vista por el supervisor. El siguiente paso es agregar la nueva alarma a las que ya tiene asignadas el vehículo, se logra a través de los métodos “AddAlarm” de las interfaces “IAlarmService” e “IAlarmDao”. El último paso es actualizar las relaciones del vehículo al cual se

le asignó la nueva alarma de velocidad, para esto se invocan los métodos “AddVehicle” de las interfaces “IVehicleService” e “IVehicleDao”.

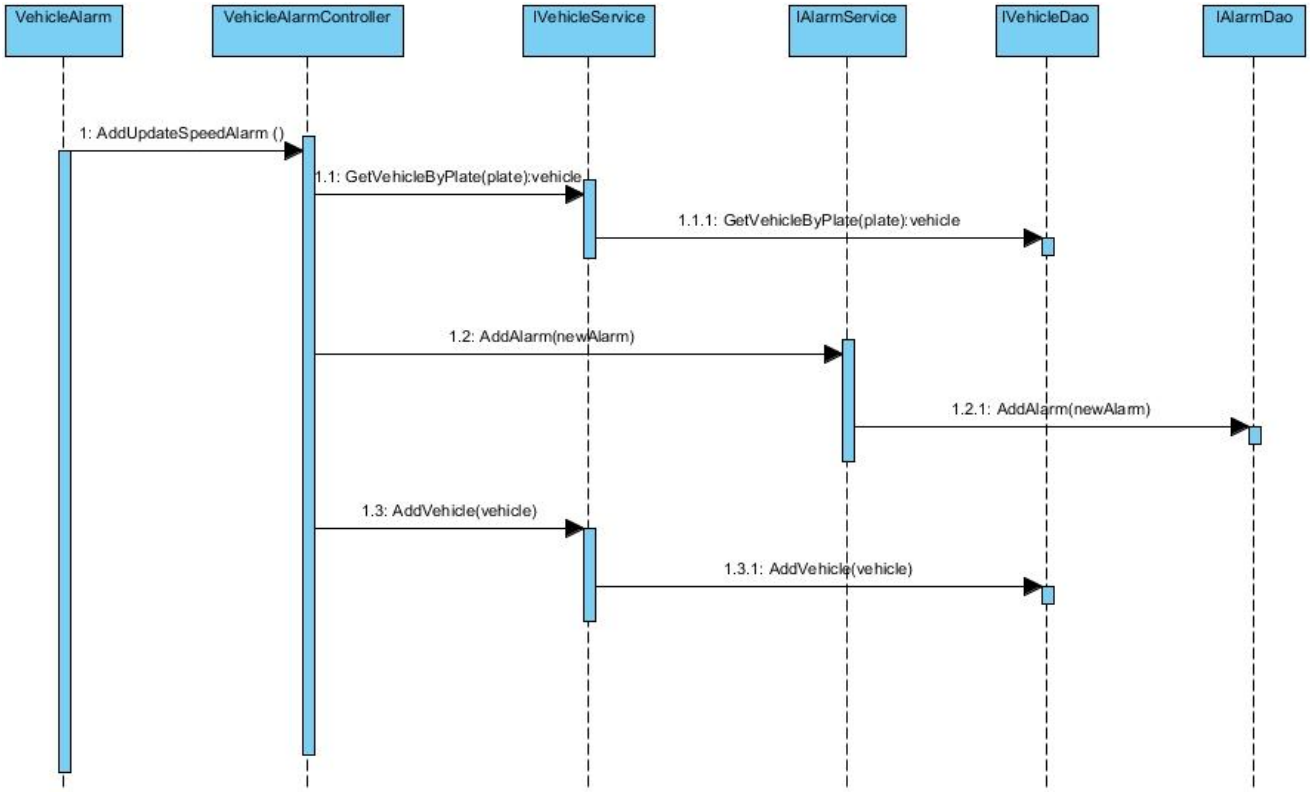

Figura 3.10. Descripción de la funcionalidad Configurar Alarma de Velocidad.

3.6 Diseño de la base de datos

Una de las tareas más importantes en el desarrollo de un sistema es el diseño de la base de datos. Una base de datos correctamente diseñada permite obtener acceso a información exacta y actualizada. Un diseño correcto es esencial para lograr los objetivos fijados ya que posibilita mayor calidad en la implementación del sistema.

3.6.1 Modelo de datos

Describe la representación lógica y física de los datos persistentes usados por el sistema. A continuación se presenta el modelo de datos del sistema:

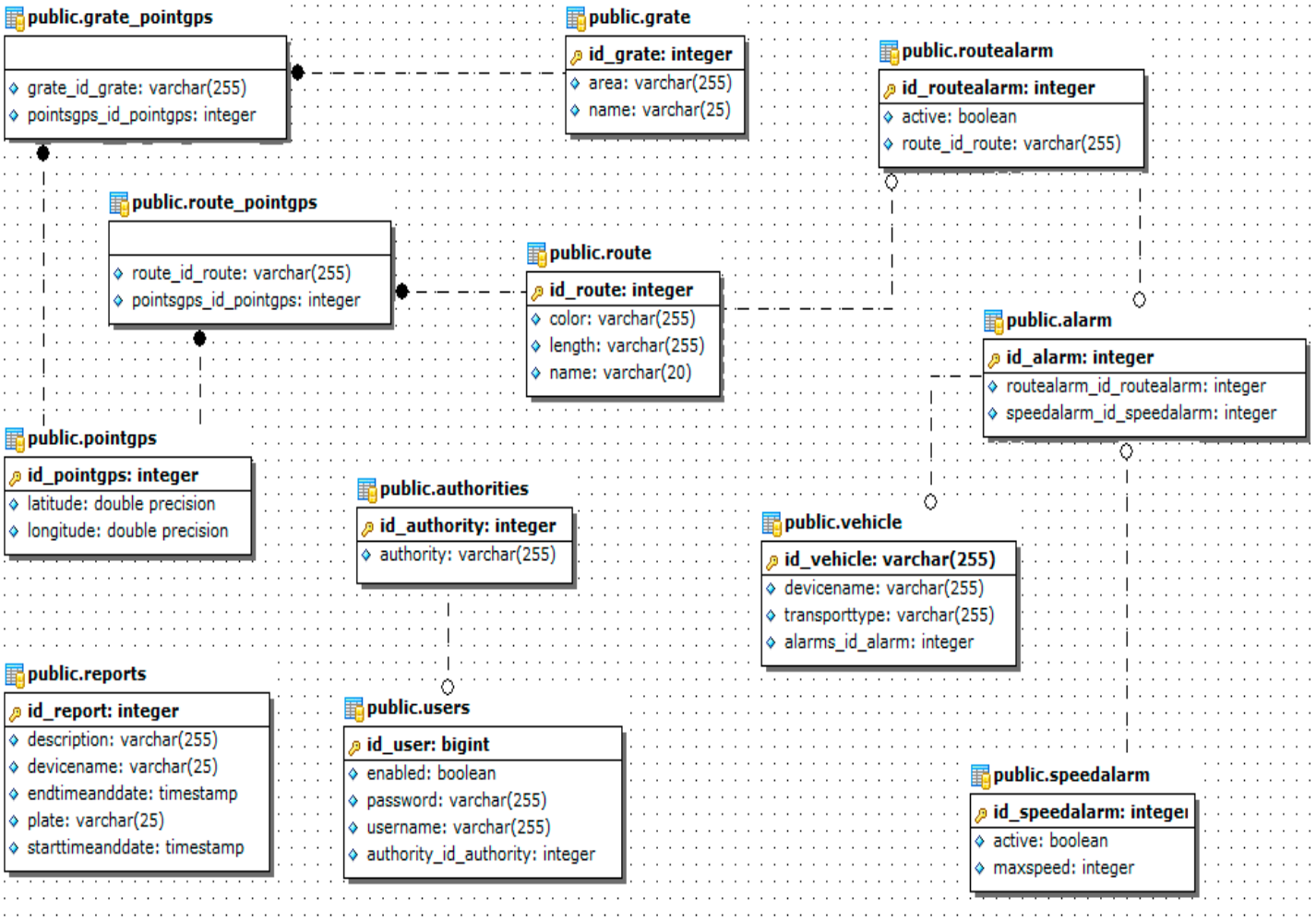

Figura 3.11. Modelo de Datos del sistema.

Para una mejor comprensión del modelo de datos se describen a continuación las tablas que lo componen:

- **users**: Contiene nombre de usuario y contraseña de cada usuario del sistema.
- **authorities**: Contiene información acerca de los permisos de cada usuario registrado en el sistema.
- **route**: Contiene información acerca de las rutas que se crearon.
- **routepointgps**: Representa la descripción existente entre la tabla route y la tabla pointgps.
- **grate**: Contiene información acerca de las áreas creadas por el supervisor.
- **gratepointgps**: Representa la descripción existente entre la tabla grate y la tabla pointgps.

- **routealarm**: Contiene los datos referentes a las alarmas de ruta.
- **speedalarm**: Contiene los datos referentes a las alarmas de velocidad.
- **alarm**: Contiene las referencias a los vehículos y a los tipos de alarmas que les fueron asignadas.
- **report**: Contiene los datos de los reportes de los incumplimientos por parte de los choferes.
- **vehicle**: Contiene los datos de los vehículos.
- **pointgps**: Contiene todos los puntos GPS que componen las rutas y áreas.

3.7 Conclusiones Parciales

En el desarrollo del capítulo se elaboraron los diagramas de funcionalidades más significativos, lo que ha permitido obtener una mejor visión del sistema. Se diseñó además el modelo de datos para mostrar la representación de las entidades persistentes usadas por el sistema, los patrones arquitectónicos presentes en la aplicación y los patrones de diseño a usar durante el desarrollo del sistema.

CAPÍTULO 4: IMPLEMENTACIÓN Y PRUEBA

4.1 Introducción

En el presente capítulo se realizará el diagrama de despliegue, con el objetivo de especificar las conexiones necesarias entre los diferentes elementos de hardware y software que garanticen un correcto funcionamiento del sistema. Además se realizaran pruebas de caja negra con el objetivo de realizar una inspección final de las especificaciones del diseño y la implementación. Con el desarrollo de estas pruebas se garantizará la detección de posibles fallos en el comportamiento del sistema, para ello se especifican los casos de pruebas realizados a las principales funcionalidades.

4.2 Diagrama de despliegue

Los diagramas de despliegue describen la topología del sistema, representando la estructura de los elementos de hardware y el software que se ejecuta en cada uno de ellos. Los diagramas de despliegue representan a los nodos y sus relaciones. Son conectados por asociaciones de comunicación tales como enlaces de red, conexiones TCP/IP. (24)

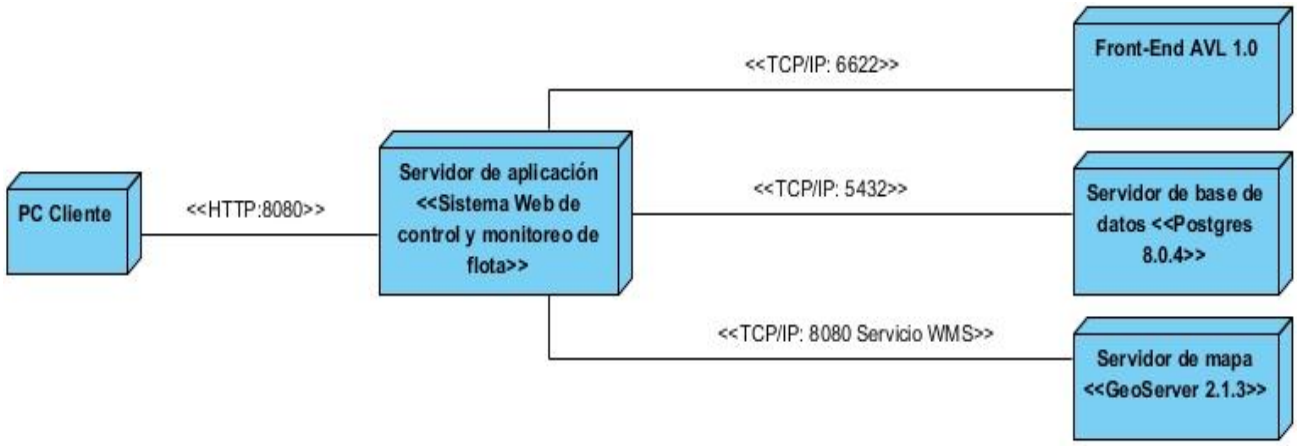

Figura 4.1. Diagrama de despliegue del sistema.

4.3 Pruebas

Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la implementación. Para validar que el sistema cumple con los requerimientos funcionales se realizaron pruebas a las funcionalidades ante diferentes escenarios, los que incluyen la entrada de datos, procesamiento y obtención de resultados. Las pruebas

se realizan a cada una de las funcionalidades del sistema permitiendo conocer si el comportamiento del mismo coincide con los resultados esperados.

4.3.1 Descripción de los casos de prueba

Los casos de prueba especifican una forma de probar el sistema, incluyendo las entradas con las que se probará, los resultados esperados y las condiciones bajo las que se prueba.

Para comprobar el correcto funcionamiento del sistema se definieron un conjunto de casos de pruebas que reúnen la mayor cantidad de escenarios posibles. A continuación se muestran los casos de prueba:

1. **Caso de Prueba Configurar Alarma de Velocidad:**

   Descripción General: Permite al usuario añadir, modificar, habilitar y deshabilitar alarmas de velocidad.

   Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.

Escenario	Descripción	Activar	Velocidad Máxima	Respuesta del sistema	Flujo central
EP 1.1 Se configura la alarma de velocidad para un vehículo.	El supervisor introduce los datos de forma correcta.	V	V	El sistema establece una alarma de velocidad para el vehículo.	1-El supervisor selecciona un vehículo. 2-Selecciona la alarma de velocidad. 3-El sistema muestra la interfaz de configuración "Alarma de velocidad". 4-El supervisor introduce los datos solicitados. 5-Presiona el botón "Guardar".

EP 1.2 Campos vacíos.	El supervisor deja el campo velocidad vacío.	V	NA	Se muestra un mensaje con el texto: "Por favor, rellene este campo".	1-El supervisor selecciona un vehículo. 2-Selecciona la alarma de velocidad. 3-El sistema muestra la interfaz de configuración "Alarma de velocidad". 4-El supervisor deja campos obligatorios vacíos. 5-Presiona el botón "Guardar".
EP 1.3 Datos incorrectos.	El supervisor introduce caracteres extraños o letras.	V	I	Se muestra un mensaje con el texto: "Por favor, ajústese al formato solicitado: Solo se pueden introducir números".	1-El supervisor selecciona un vehículo. 2-Selecciona la alarma de velocidad. 3-El sistema muestra la interfaz de configuración "Alarma de velocidad". 4-El supervisor introduce caracteres extraños o letras. 5-Presiona el botón "Guardar".

Tabla 4.1. Caso de Prueba "Configurar Alarma de Velocidad".

2. **Caso de Prueba Configurar Alarma de Ruta:**

   Descripción General: Permite al usuario añadir, modificar, habilitar y deshabilitar alarmas de ruta.

   Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.

Escenario	Descripción	Activar	Ruta asignada	Respuesta del sistema	Flujo central
EP 2.1 Se configura la alarma de ruta para el vehículo.	El supervisor introduce los datos de forma correcta.	V	V	El sistema establece una alarma de ruta para el vehículo.	1-El supervisor selecciona un vehículo. 2-Selecciona la alarma de ruta. 3-El sistema muestra la interfaz de configuración "Alarma de ruta". 4-El supervisor introduce los datos solicitados. 5-Presiona el botón "Guardar".
EP 2.2 Campos vacíos.	El supervisor no al vehículo una ruta.	V	NA	Se muestra un mensaje con el texto: "Error en la configuración de la alarma de ruta. Seleccione una ruta".	1-El supervisor selecciona un vehículo. 2-Selecciona la alarma de ruta. 3-El sistema muestra la interfaz de configuración "Alarma de ruta". 4-El supervisor deja campos obligatorios vacíos. 5-Presiona el botón "Guardar".

Tabla 4.2. Caso de Prueba "Configurar Alarma de Ruta".

4.4 No conformidades detectadas en el sistema

A continuación se muestra una tabla que contiene los resultados obtenidos al realizar las pruebas a la aplicación, la misma contiene la cantidad de no conformidades que han sido detectadas en el sistema, mostrando cuántas de estas proceden, cuántas no proceden y cuántas fueron resueltas; todas fueron no significativas.

Sistema	Total de no conformidades	Número de no conformidades que proceden	Número de no conformidades que no proceden	Número de no conformidades resueltas
Sistema Web de Control y Monitoreo de Flotas	7	6	1	6

Tabla 4.3. No conformidades detectadas en el sistema.

4.5 Conclusiones parciales

En el presente capítulo se especificaron las conexiones necesarias entre los diferentes elementos de hardware y software a través de la realización del diagrama de despliegue, el cual modela las características necesarias para el funcionamiento del sistema. Se realizaron pruebas para comprobar el correcto funcionamiento del sistema, las cuales fueron realizadas de manera satisfactoria validando que el mismo cumple con las funcionalidades establecidas.

CONCLUSIONES

En el presente trabajo se describieron las herramientas y tecnologías utilizadas, la propuesta del sistema, los diagramas de funcionalidades, la arquitectura del sistema, los patrones utilizados para la elaboraciones de la aplicación y los casos de pruebas identificados para el desarrollo de un sistema web capaz de monitorear y controlar flotas, haciendo uso de dispositivos con tecnología GPS, con el objetivo de facilitar el trabajo de los supervisores de transporte en las empresas y propiciar el ahorro de recursos a las mismas. La solución obtenida está basada en software libre, por tanto se presenta como una alternativa ante la independencia tecnológica ya que evita el pago de licencias a empresas extranjeras. Se definió la arquitectura a utilizar así como los patrones arquitectónicos para el diseño e implementación del sistema. Para comprobar el correcto funcionamiento del sistema, el mismo fue sometido a pruebas de caja negra, arrojando resultados satisfactorios que posibilitaron la validación del cumplimiento de los requisitos funcionales.

Con la realización del sistema de control y monitoreo de flotas se obtiene una posible solución a una serie de problemas existentes en las empresas cubanas que cuentan con diversos medios de transporte como son:

- ✓ Los desvíos en los recorridos.
- ✓ Las entregas fuera de tiempo.
- ✓ La gestión de combustible.

Por lo anteriormente expuesto se concluye que se cumplió con el objetivo general planteado, y con cada una de las tareas propuestas para desarrollar el sistema web de control y posicionamiento de flotas.

RECOMENDACIONES

Después de terminado el desarrollo del sistema se detectaron algunas posibles mejoras que pueden incluirse en el mismo. A continuación se detallan las recomendaciones realizadas al sistema:

1. Incorporar al sistema nuevas alarmas que faciliten el control de las flotas, como pueden ser:
    - ✓ Alarma de área de trabajo.
    - ✓ Alarma de proximidad entre los vehículos.
    - ✓ Alarma de mantenimiento de las flotas.
2. Incorporar una funcionalidad que permita almacenar el historial de los vehículos para su posterior consulta.
3. Permitir que el supervisor pueda definir el intervalo de tiempo para la actualización de la posición de los vehículos y para el chequeo de alarmas.
4. Crear un módulo para la gestión de usuarios, los diferentes roles y el acceso a las funcionalidades acorde al rol que desempeña cada usuario.
5. Diferenciar de manera gráfica los vehículos que han incurrido en alguna violación.
6. Hacer uso de certificados de seguridad para garantizar la integridad de la información de la información con la que se trabaja en el sistema.

REFERENCIAS

1. Gestión de Flotas e Inteligencia de Negocios. *Gestión de Flotas e Inteligencia de Negocios.* [En línea] 2008. [Citado el: 21 de Septiembre de 2012.] http://www.ful-mar.com.ar/es/gestion-de-flotas.php. 1.

2. Introducción respecto a los sistemas para control de flotas de vehículos. [En línea] 11 de Octubre de 2011. [Citado el: 24 de Septiembre de 2012.] http://es.scribd.com/doc/70647884/Control-de-Flotas.

3. MovilFleet. *MovilFleet.* [En línea] 2012. [Citado el: 12 de Octubre de 2012.] http://www.mobilefleet.es/que-es-mobilefleet.php.

4. Micronav. *Micronav.* [En línea] 2012. [Citado el: 14 de Octubre de 2012.] http://www.micronav.net/.

5. *MovilWeb: Aplicación para el control de flotas basada en PostgreSQL.* Suárez, Guillermo González. 1, s.l. : Revista Cubana de Ciencias Informáticas (RCCI), 2011, Vol. 5.

6. Sistema de Posicionamiento Global. *Sistema de Posicionamiento Global.* [En línea] 2012. [Citado el: 21 de Octubre de 2012.] http://www.gps.gov/spanish.php.

7. Calvert, Kenneth L. y Donahoo, Michael J. *TCP/IP Sockets in Java.* s.l. : Morgan Kaufmann Publishers.

8. [En línea] Octubre de 2008. [Citado el: 14 de Noviembre de 2012.] http://guimi.net/descargas/Monograficos/G-Lenguajes_de_programacion.pdf.

9. Holzner, Steven. *La biblia de Java 2.* s.l. : Anaya Multimedia, 2000.

10. Ivar Jacobson, Grady Booch, James Rumbaugh. *The Unified Modeling Language Reference Manual.* Madrid : s.n., 2000.

11. Craig Walls, Ryan Breidenbach. *Spring in Action 2.* 2008.

12. Gavin King, Christian Bauer, Max Rydahl Andersen. *Documentación de referencia de Hibernate.*

13. Hazzard, Erik. *OpenLayers 2.10. Beginner's Guide.* 2011.

14. Experts, jQuery Community. *Jquery Cookbook.* s.l. : O'Reilly Media, 2010.

15. Spring Tool Suite. *Spring Tool Suite.* [En línea] 2012. [Citado el: 30 de Noviembre de 2012.] http://www.springsource.org/sts.

16. Apache Tomcat. *Apache Tomcat.* [En línea] 2011. [Citado el: 1 de Diciembre de 2012.] http://tomcat.apache.org.

17. PostgreSQL. *PostgreSQL.* [En línea] 2011. [Citado el: 4 de Diciembre de 2012.] http://www.postgresql.org.es/sobre_postgresql.

18. Escalona, Yosel Hernandez. *Propuesta de Front End para gestión de posicionamiento vehicular sobre redes TETRA NEBULA.* Habana : s.n., 2011.

19. Visual Paradigm for UML. *Visual Paradigm for UML*. [En línea] 2012. [Citado el: 6 de Diciembre de 2012.] http://www.visual-paradigm.com/product/vpuml/.

20. Pressman, Roger S. *Ingeniería del software, Un enfoque práctico quinta edición.* 2002.

21. Mariñán, Martín Pérez. *Patrones de diseño.*

22. Freeman, Adam y Sanderson, Steven. *Pro asp.net mvc 3 Framework.* s.l. : apress, 2011.

23. Erich Gamma, Richard Helm, Ralph Jhonson, Jhon Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* s.l. : China-Pub.com.

24. Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objeto.*

25. *Fundamentos de la Dirección de Proyectos.* s.l. : Project Management Institute, Inc., 2004.

26. Consortium, Open Geospatial. Open Geospatial Consortium. *Open Geospatial Consortium.* [En línea] 2012. [Citado el: 10 de Diciembre de 2012.] http://www.opengeospatial.org/ogc.

27. Olaya, Víctor. *Sistemas de Información Geográfica.* 2011.

28. Swigart, Scott. Interview With Rod Johnson. [En línea] 17 de Septiembre de 2007. [Citado el: 17 de Enero de 2013.] http://howsoftwareisbuilt.com/2007/09/10/interview-with-rod-johnson-ceo-interface21/.

BIBLIOGRAFÍA

1. Gestión de Flotas e Inteligencia de Negocios. *Gestión de Flotas e Inteligencia de Negocios.* [En línea] 2008. [Citado el: 21 de Septiembre de 2012.] http://www.ful-mar.com.ar/es/gestion-de-flotas.php. 1.

2. Introducción respecto a los sistemas para control de flotas de vehículos. [En línea] 11 de Octubre de 2011. [Citado el: 24 de Septiembre de 2012.] http://es.scribd.com/doc/70647884/Control-de-Flotas.

3. MovilFleet. *MovilFleet.* [En línea] 2012. [Citado el: 12 de Octubre de 2012.] http://www.mobilefleet.es/que-es-mobilefleet.php.

4. Micronav. *Micronav.* [En línea] 2012. [Citado el: 14 de Octubre de 2012.] http://www.micronav.net/.

5. *MovilWeb: Aplicación para el control de flotas basada en PostgreSQL.* Suárez, Guillermo González. 1, s.l. : Revista Cubana de Ciencias Informáticas (RCCI), 2011, Vol. 5.

6. Sistema de Posicionamiento Global. *Sistema de Posicionamiento Global.* [En línea] 2012. [Citado el: 21 de Octubre de 2012.] http://www.gps.gov/spanish.php.

7. Calvert, Kenneth L. y Donahoo, Michael J. *TCP/IP Sockets in Java.* s.l. : Morgan Kaufmann Publishers.

8. [En línea] Octubre de 2008. [Citado el: 14 de Noviembre de 2012.] http://guimi.net/descargas/Monograficos/G-Lenguajes_de_programacion.pdf.

9. Holzner, Steven. *La biblia de Java 2.* s.l. : Anaya Multimedia, 2000.

10. Ivar Jacobson, Grady Booch, James Rumbaugh. *The Unified Modeling Language Reference Manual.* Madrid : s.n., 2000.

11. Craig Walls, Ryan Breidenbach. *Spring in Action 2.* 2008.

12. Gavin King, Christian Bauer, Max Rydahl Andersen. *Documentación de referencia de Hibernate.*

13. Hazzard, Erik. *OpenLayers 2.10. Beginner's Guide.* 2011.

14. Experts, jQuery Community. *Jquery Cookbook.* s.l. : O'Reilly Media, 2010.

15. Spring Tool Suite. *Spring Tool Suite.* [En línea] 2012. [Citado el: 30 de Noviembre de 2012.] http://www.springsource.org/sts.

16. Apache Tomcat. *Apache Tomcat.* [En línea] 2011. [Citado el: 1 de Diciembre de 2012.] http://tomcat.apache.org.

17. PostgreSQL. *PostgreSQL.* [En línea] 2011. [Citado el: 4 de Diciembre de 2012.] http://www.postgresql.org.es/sobre_postgresql.

18. Escalona, Yosel Hernandez. *Propuesta de Front End para gestión de posicionamiento vehicular sobre redes TETRA NEBULA.* Habana : s.n., 2011.

19. Visual Paradigm for UML. *Visual Paradigm for UML*. [En línea] 2012. [Citado el: 6 de Diciembre de 2012.] http://www.visual-paradigm.com/product/vpuml/.

20. Pressman, Roger S. *Ingeniería del software, Un enfoque práctico quinta edición.* 2002.

21. Mariñán, Martín Pérez. *Patrones de diseño.*

22. Freeman, Adam y Sanderson, Steven. *Pro asp.net mvc 3 Framework.* s.l. : apress, 2011.

23. Erich Gamma, Richard Helm, Ralph Jhonson, Jhon Vlissides. *Design Patterns. Elements of Reusable Object-Oriented Software.* s.l. : China-Pub.com.

24. Larman, Craig. *UML y Patrones. Introducción al análisis y diseño orientado a objeto.*

25. *Fundamentos de la Dirección de Proyectos.* s.l. : Project Management Institute, Inc., 2004.

26. Consortium, Open Geospatial. Open Geospatial Consortium. *Open Geospatial Consortium.* [En línea] 2012. [Citado el: 10 de Diciembre de 2012.] http://www.opengeospatial.org/ogc.

27. Olaya, Víctor. *Sistemas de Información Geográfica.* 2011.

28. Swigart, Scott. Interview With Rod Johnson. [En línea] 17 de Septiembre de 2007. [Citado el: 17 de Enero de 2013.] http://howsoftwareisbuilt.com/2007/09/10/interview-with-rod-johnson-ceo-interface21/.

ANEXOS.

GPRMC	,123456.0	,A	,3444.123	,N	,13521.456	,E	,005.60	,123.5	,021100	,01.0	,W
0	1	2	3	4	5	6	7	8	9	10	11

#	Descripción	Rango	Nº Bytes
0	Cabecera del mensaje NMEA	"GPRMC"	5
1	UTC: Hora		
	"12": hh	00 – 23	2
	"34": mm	00 – 59	2
	"56": ss (entero)	00 – 59	2
	"0": ss (fracción)	0 – 9	1
2	Estado	A →Posición fijada V →Posición interrumpida	1
3	Latitud		
	"34": grados	00 – 90	2
	"44": minutos (entero)	00 – 59	2
	"123": minutos (fracción)	000 – 999	3
4	Latitud Norte / Sur N → Norte	N →Norte S →Sur	1
5	Longitud		
	"135": grados	00 – 180	3
	"21": minutos (entero)	00 – 59	2
	"456": minutos (fracción)	000 – 999	3
6	Longitud Este/Oeste E → Este	E →Este W →Oeste	1
7	Velocidad (kts)		
	"005.60"	000.00 – 999.99	6
	Nota : Si no hay información de velocidad disponible se mostrará → 0.00 *1kts =1 milla naútica por hora = 1,852 km por hora*		
8	Rumbo (grados)		
	"123.5"	000.0 – 359.9	5
	Nota: Si no hay información de rumbo disponible se mostrará → 0.00		
9	UTC: Fecha		
	"02": DD	01 – 31	2
	"11": MM	01 – 12	2
	"00": AA	94 – 40 (1994 – 2040)	2
10	Desviación magnética (grados)		
	"01.0"	00.0 – 90.0	4
11	Desviación magnética W→ Oeste	E →Este W →Oeste	1
		Total	=52

Figura 5.1 Descripción del formato GPRMC.

3. **Caso de Prueba Adicionar Ruta:**

   Descripción General: Permite al usuario añadir, listar y eliminar rutas.

   Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.

Escenario	Descripción	Nombre	Respuesta del sistema	Flujo central

EP 3.1 Se adiciona una ruta	El supervisor introduce la ruta de forma correcta	V	El sistema crea la ruta	1-El supervisor selección la opción agregar ruta 2-El sistema muestra la interfaz "Adicionar ruta" 3-El supervisor inserta los datos solicitados y crea la ruta en el mapa 4-Presiona "Enter" o "doble click"
EP 3.2 Campos vacíos	El supervisor deja campos vacíos	NA	Se muestra un mensaje con el texto: "Por favor, rellene este campo".	1-El supervisor selección la opción agregar ruta 2-El sistema muestra la interfaz "Adicionar ruta" 3-El supervisor deja campos obligatorios vacíos 4-Presiona "Enter" o "doble click"
EP 3.3 Campos incorrectos	El supervisor introduce caracteres extraños	I	Se muestra un mensaje con el texto: "Datos introducidos de forma incorrecta".	1-El supervisor selección la opción agregar ruta 2-El sistema muestra la interfaz "Adicionar ruta" 3-El supervisor introduce caracteres extraños 4-Presiona "Enter" o "doble click"

Tabla 5.1.Caso de Prueba Adicionar Ruta.

4. **Caso de Prueba Adicionar Área:**

Descripción General: Permite al usuario añadir, listar y eliminar áreas.

Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.

Escenario	Descripción	Nombre	Respuesta del sistema	Flujo central
EP 4.1 Se adiciona un área	El supervisor introduce el área de forma correcta	V	El sistema crea el área	1-El supervisor selecciona la opción agregar área 2-El sistema muestra la interfaz "Adicionar ruta" 3-El supervisor inserta los datos solicitados y crea el área en el mapa 4-Presiona "Enter" o "doble click"
EP 4.2 Campos vacíos	El supervisor deja campos vacíos	NA	Se muestra un mensaje con el texto: "Por favor, rellene este campo".	1-El supervisor selecciona la opción agregar área 2-El sistema muestra la interfaz "Adicionar área" 3-El supervisor deja campos obligatorios vacíos 4-Presiona "Enter" o "doble click"
EP 4.3 Campos incorrectos	El supervisor introduce caracteres extraños	I	Se muestra un mensaje con el texto: "Datos introducidos de forma incorrecta".	1-El supervisor selecciona la opción agregar área 2-El sistema muestra la interfaz "Adicionar área" 3-El supervisor introduce caracteres extraños 4-Presiona "Enter" o "doble click"

Tabla 5.2. Caso de Prueba Adicionar Área.

5. **Caso de Prueba Buscar Reportes:**

Descripción General: Permite al usuario buscar reportes.

Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.

Escenario	Descripción	Nombre	Fecha inicio	Fecha fin	Respuesta del sistema	Flujo central
EP 5.1 Se buscan reportes	El supervisor introduce los datos de forma correcta	V NA V	V V NA	V V NA	Si listan todos los reportes que coinciden con el criterio de búsqueda especificado	1-El supervisor selecciona la opción buscar reportes 2-El sistema muestra la interfaz "Buscar reportes" 3-El supervisor inserta los datos solicitados 4-El sistema muestra un listado de los reportes encontrados
EP 5.2 Campos vacíos		V V	NA V	V NA	Se muestra un mensaje con el texto: "Por favor, rellene este campo".	1-El supervisor selecciona la opción buscar reportes 2-El sistema muestra la interfaz "Buscar reportes" 3-El supervisor deja campos obligatorios vacíos 4-El sistema muestra un

						listado de los reportes encontrados
EP 5.3 Campos incorrectos	El supervisor introduce caracteres extraños	I V V	V I V	V V I	Se muestra un mensaje con el texto: "Datos introducidos de forma incorrecta".	1-El supervisor selecciona la opción buscar reportes 2-El sistema muestra la interfaz "Buscar reportes" 3-El supervisor introduce caracteres extraños 4-El sistema muestra un listado de los reportes encontrados

Tabla 5.3. Caso de Prueba Buscar Reportes.

6. **Caso de Prueba Adicionar Vehículo:**

Descripción General: Permite al usuario añadir, listar y eliminar vehículos.

Condiciones de Ejecución: El usuario debe estar autenticado en el sistema.

Escenario	Descripción	Matrícula	Dispositivo GPS	Tipo	Respuesta del sistema	Flujo central
EP 6.1 Se adiciona un vehículo	El supervisor introduce los datos de forma correcta	V	V	V	El sistema crea el vehículo	1-El supervisor selecciona la opción "Vehículos" 2-El sistema muestra la interfaz "Vehículos" 3-El supervisor introduce los

						datos de forma correcta 4-Presiona el botón "Adicionar"
EP 6.2 Campos vacíos	El supervisor deja campos vacíos	V V NA	V NA V	NA V V	Se muestra un mensaje con el texto: "Por favor, rellene este campo".	1-El supervisor selecciona la opción "Vehículos" 2-El sistema muestra la interfaz "Vehículos" 3-El supervisor deja campos obligatorios vacíos 4-Presiona el botón "Adicionar"
EP 6.3 Campos incorrectos	El supervisor introduce caracteres extraños	V V I	V I V	I V V	Se muestra un mensaje con el texto: "Datos introducidos de forma incorrecta".	1-El supervisor selecciona la opción "Vehículos" 2-El sistema muestra la interfaz "Vehículos" 3-El supervisor introduce caracteres extraños 4-Presiona el botón "Adicionar"

Tabla 5.4. Caso de Prueba Adicionar Vehículo.

Descripción de la funcionalidad **“Adicionar Área”**.

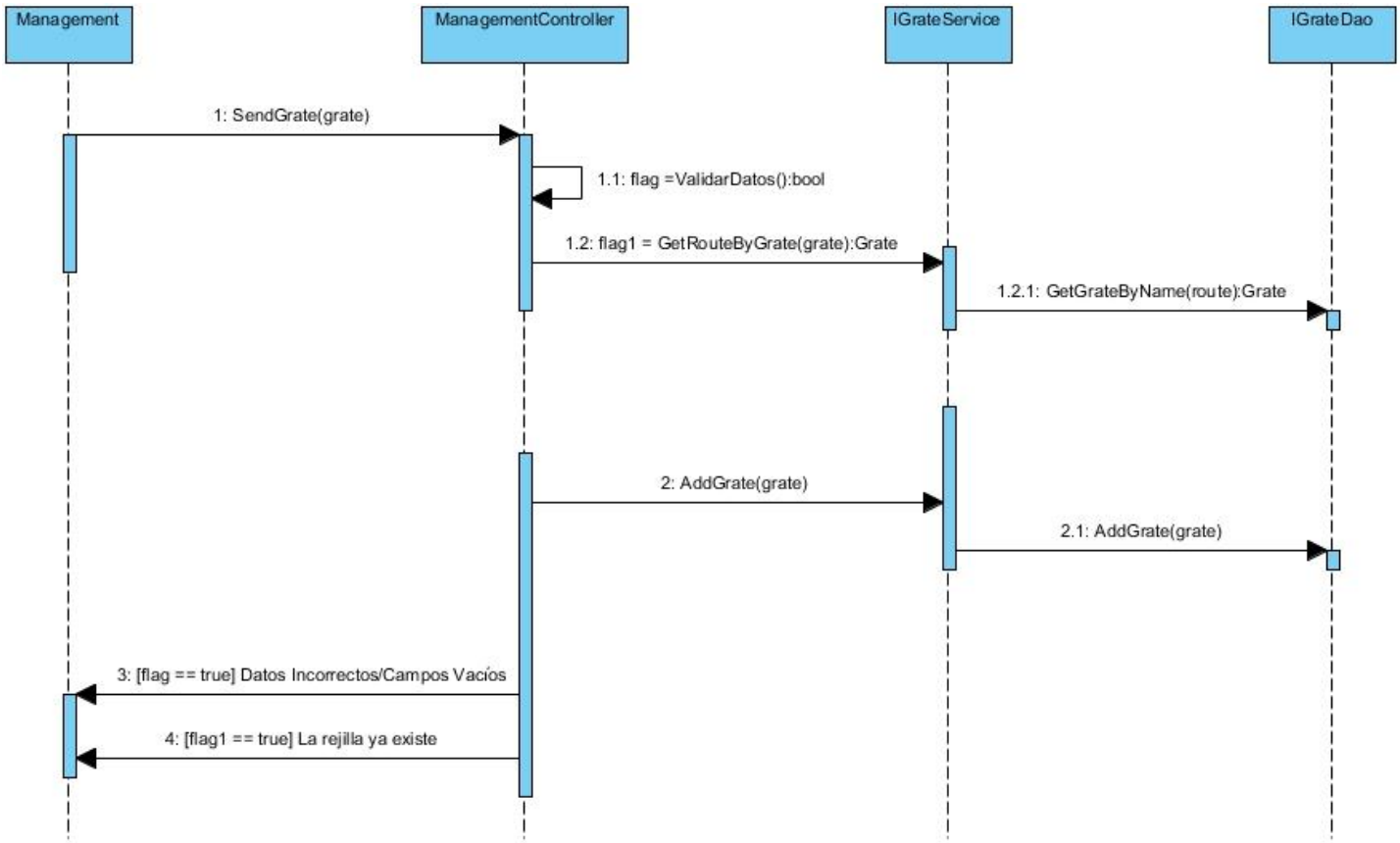

Figura 5.2. Descripción de la funcionalidad Adicionar Área.

Descripción de la funcionalidad **“Eliminar Área”**

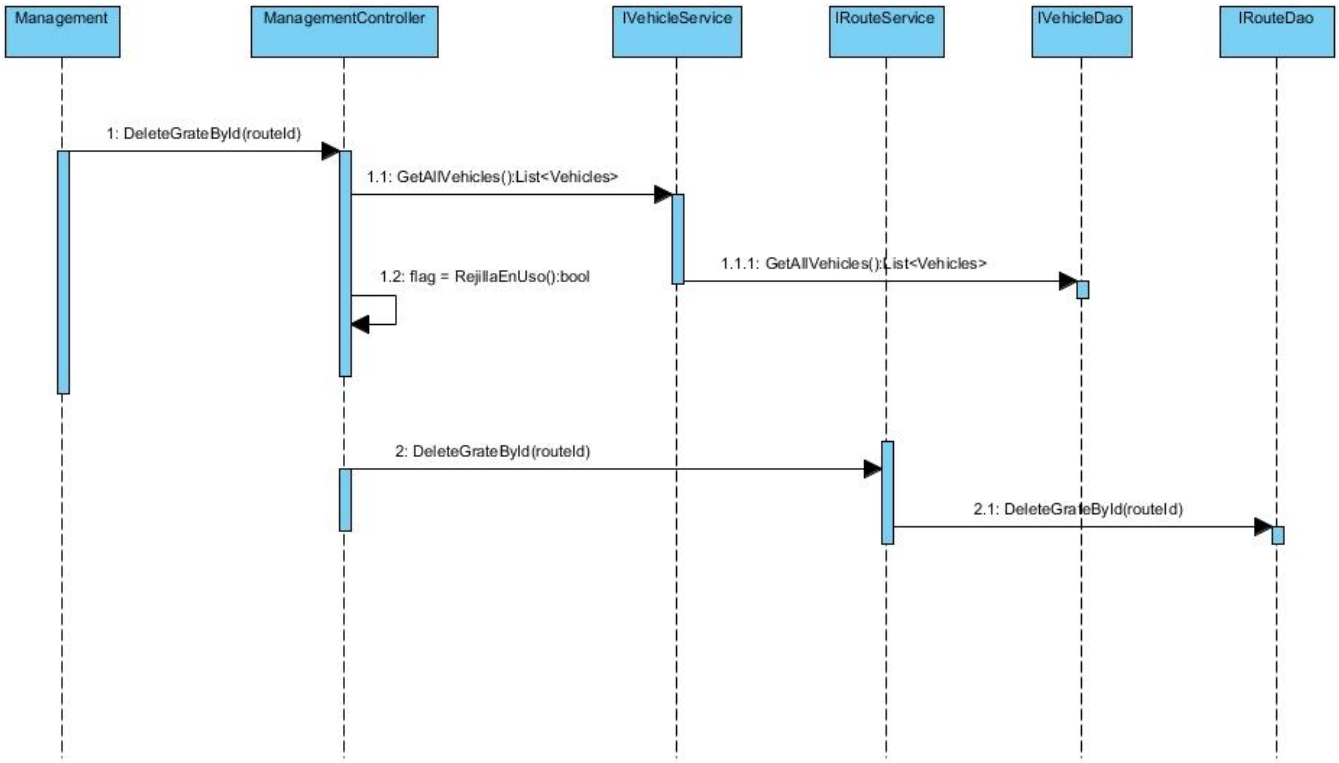

Figura 5.3. Descripción de la funcionalidad Eliminar Área.

Descripción de la funcionalidad **“Adicionar Ruta”.**

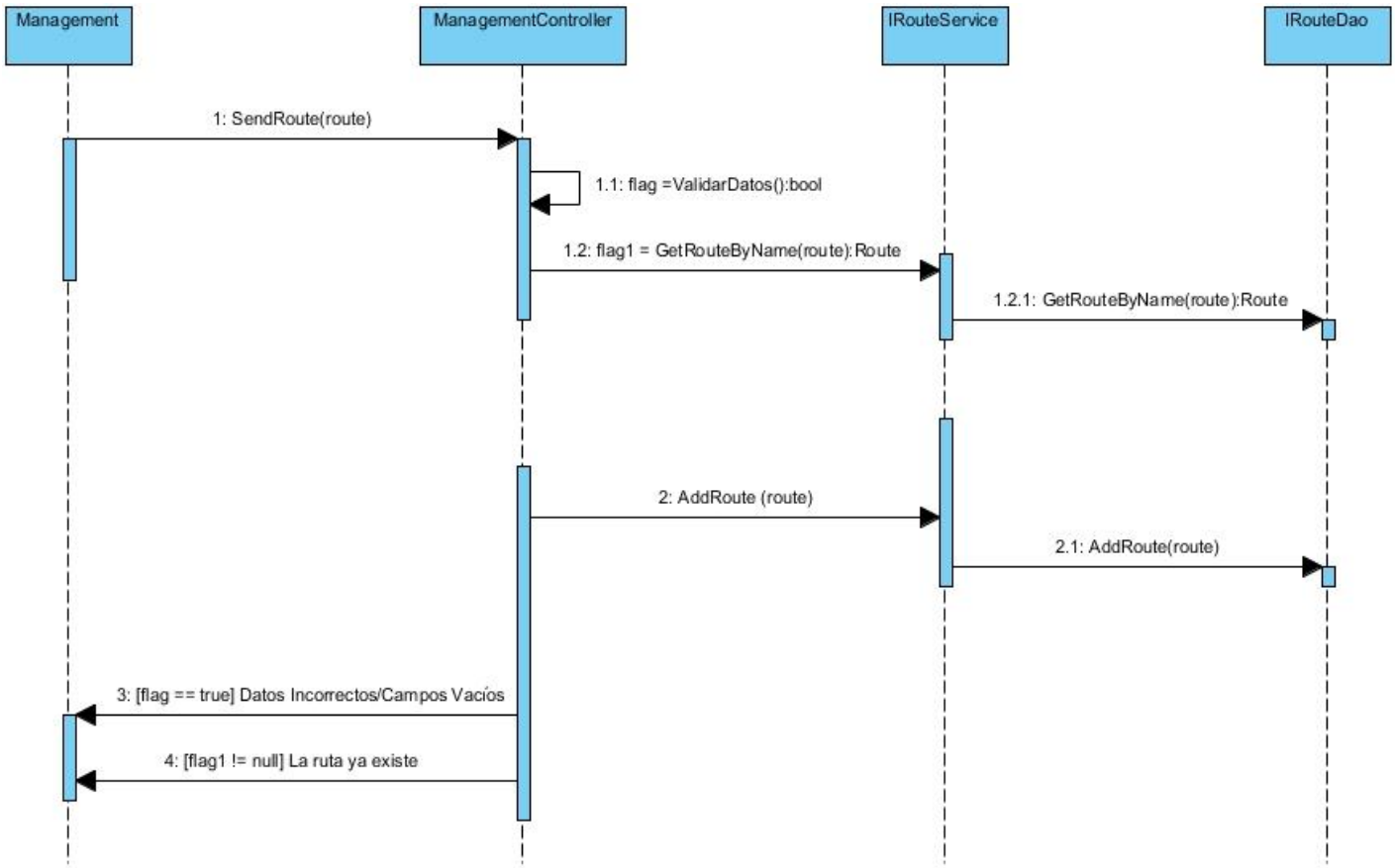

Figura 5.4. Descripción de la funcionalidad Adicionar Ruta.

Descripción de la funcionalidad **“Eliminar Ruta”.**

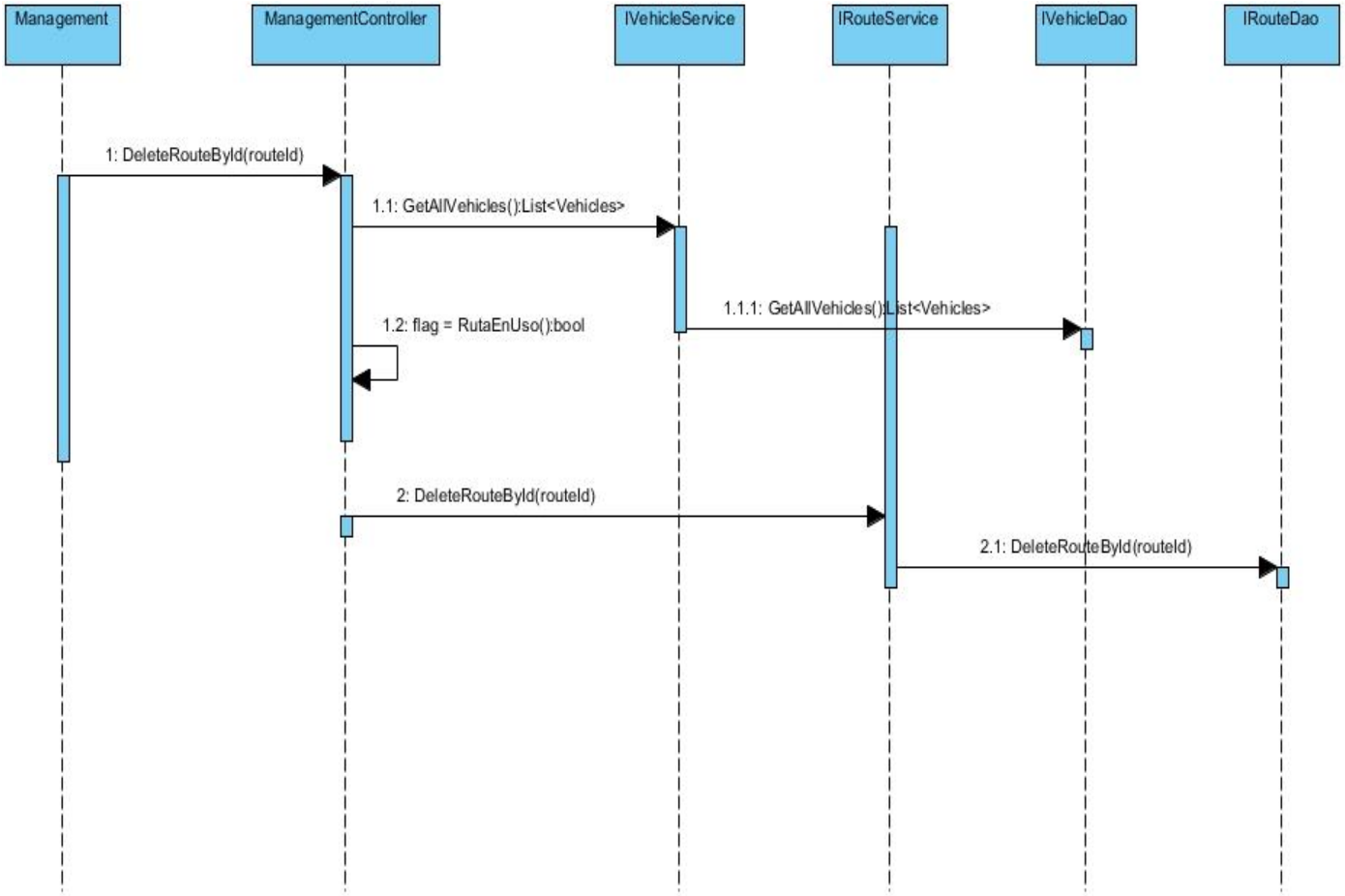

Figura 5.5. Descripción de la funcionalidad Eliminar Ruta.

Descripción de la funcionalidad **“Adicionar Vehículo”.**

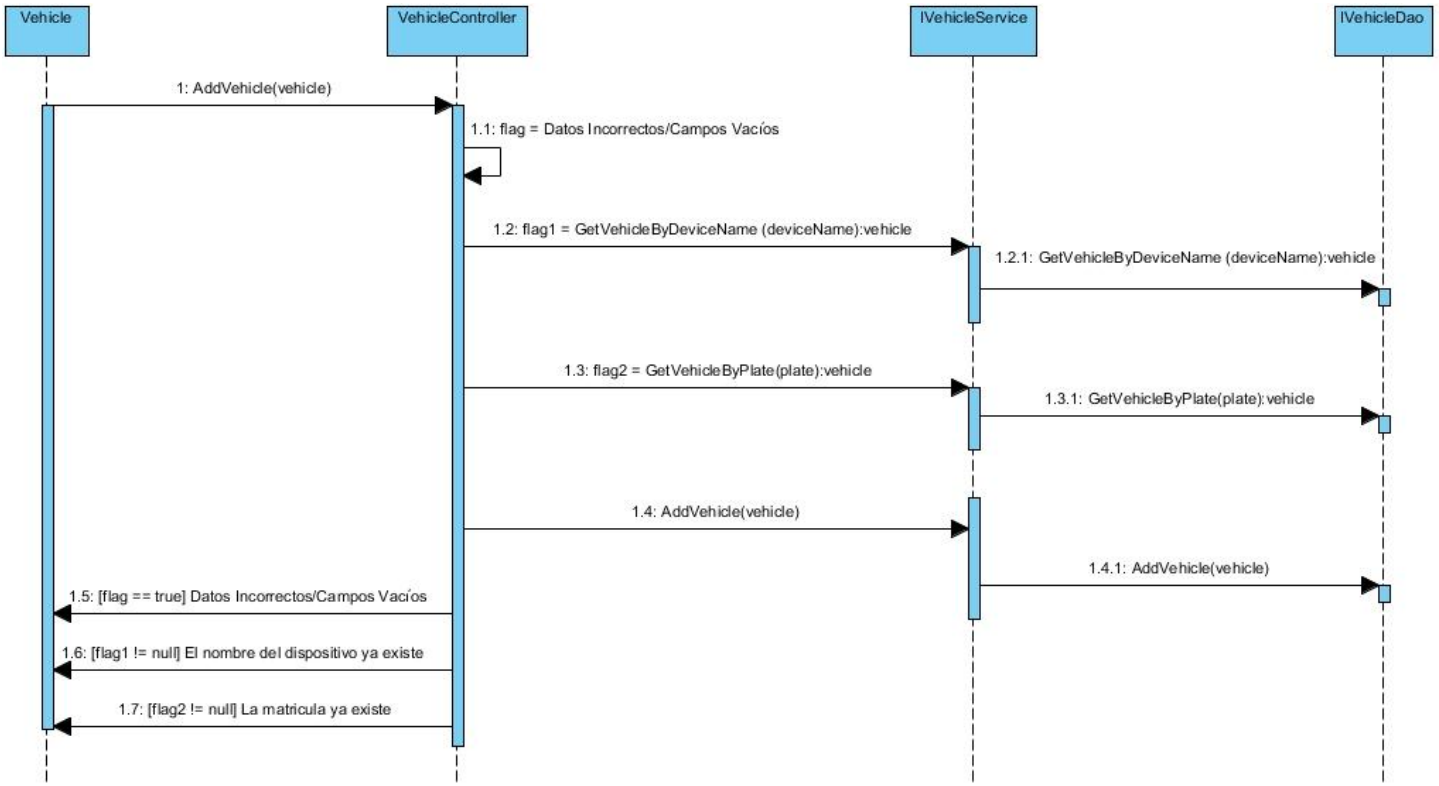

Figura 5.6. Descripción de la funcionalidad Adicionar Vehículo.

Pantallas de las funcionalidades del Sistema Web de Control y Monitoreo de Flotas

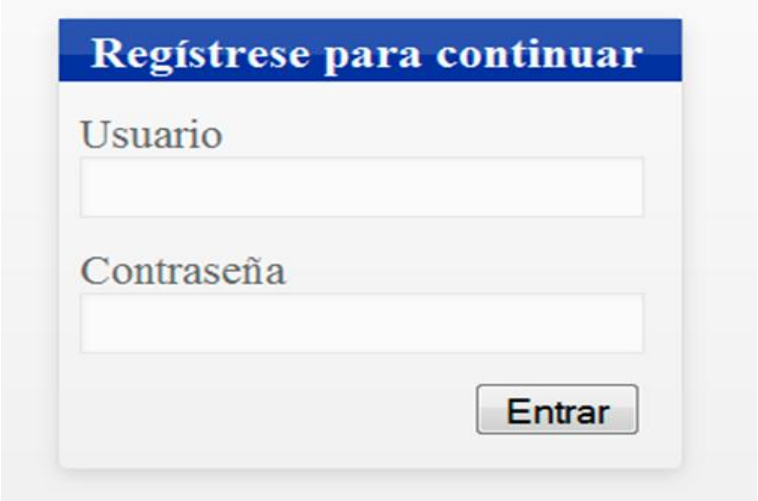

Figura 5.7. Funcionalidad “Autenticar Usuario”.

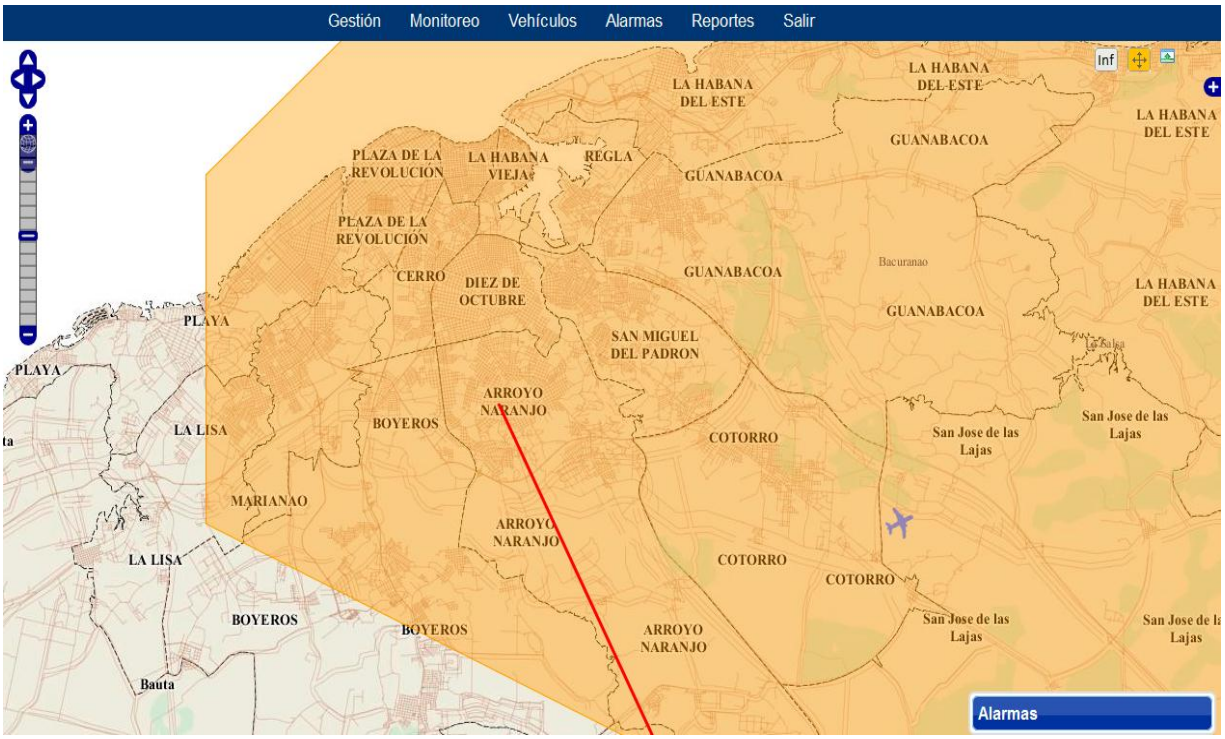

Figura 5.8. Funcionalidad "Monitoreo de vehículos".

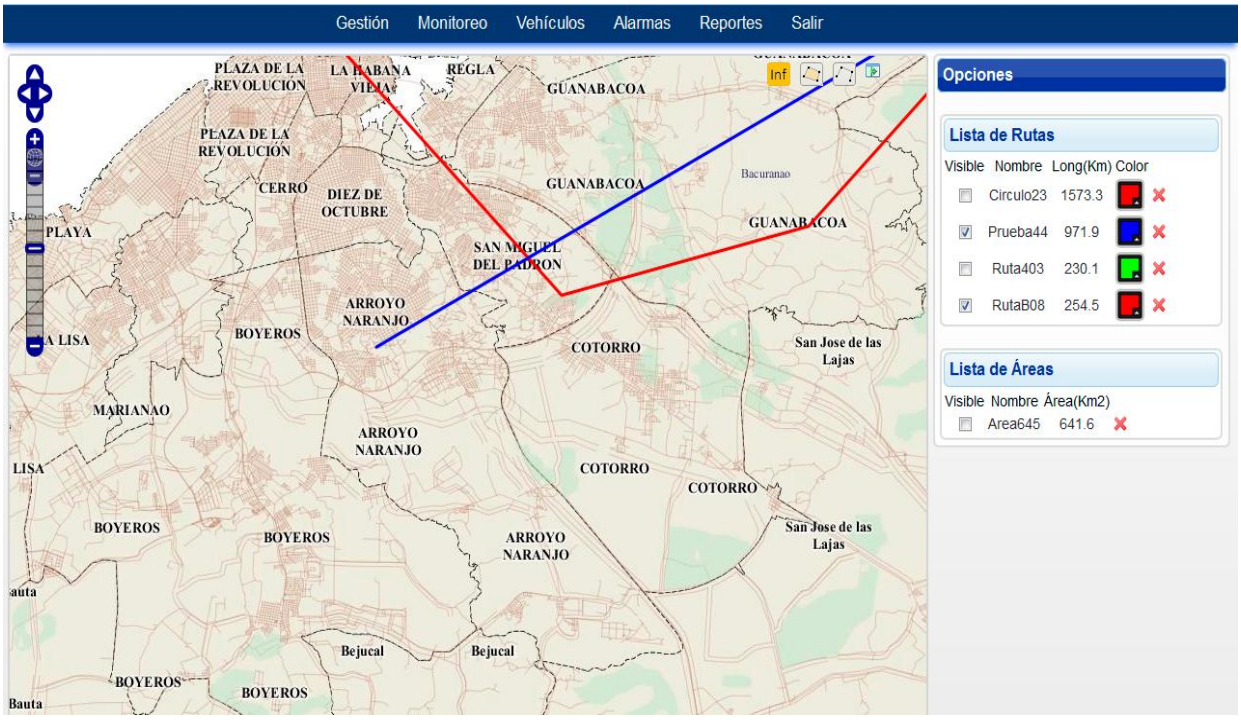

Figura 5.9. Funcionalidad "Gestión de Rutas y Áreas".

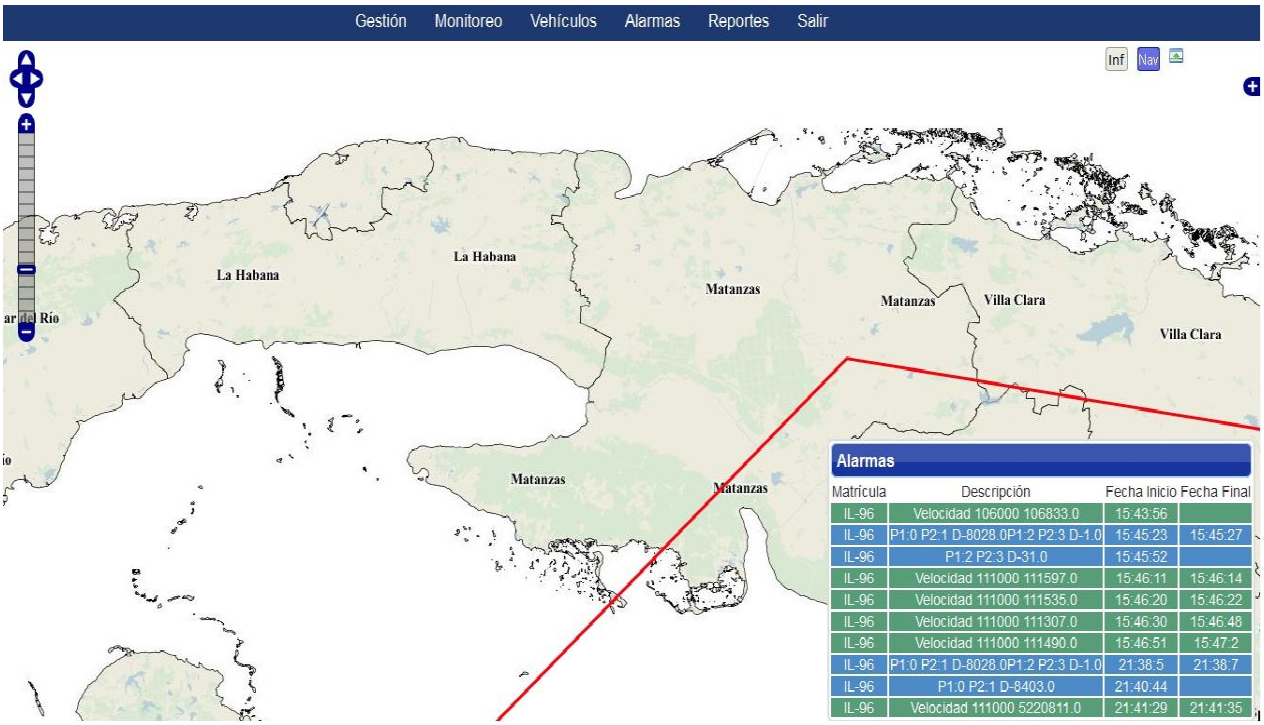

Figura 5.10. Funcionalidad "Notificación de Alarmas".

Figura 5.11. Funcionalidad "Buscar Reportes".

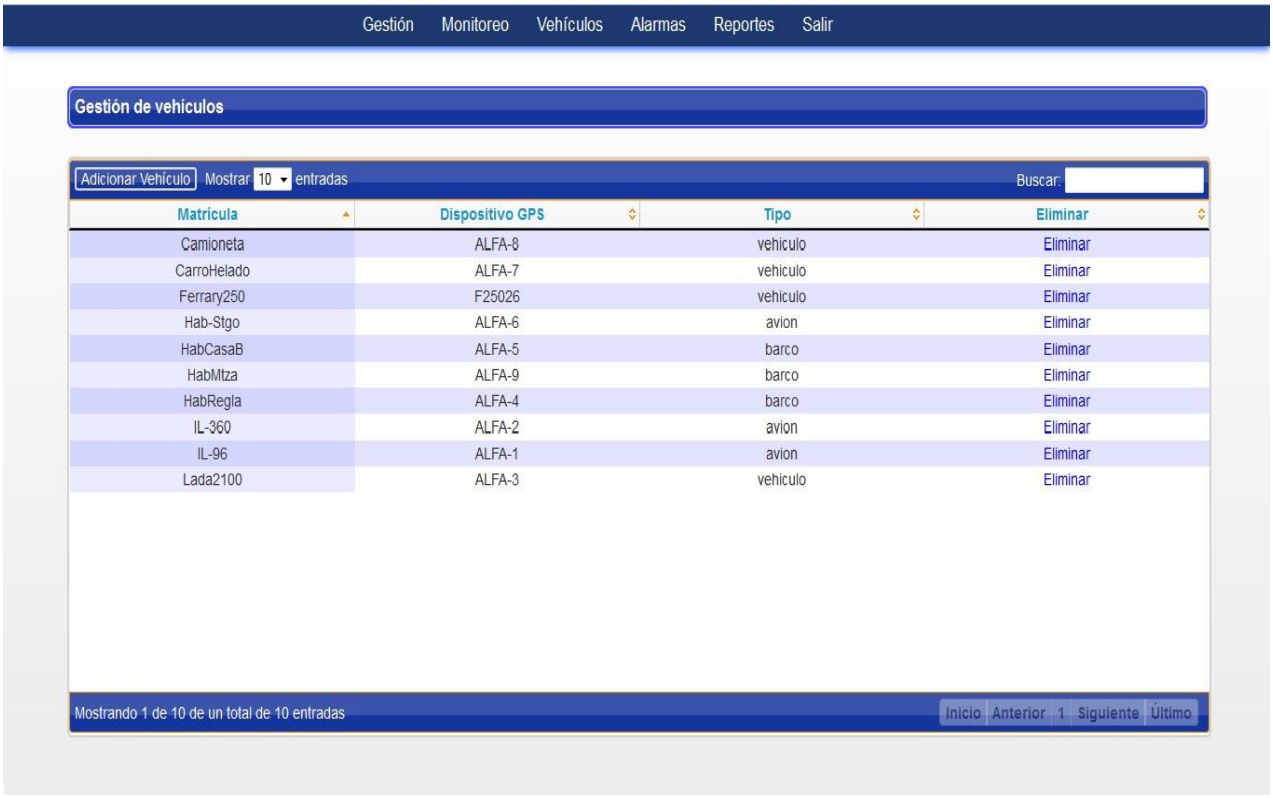

Figura 5.12. Funcionalidad "Gestión de Vehículos".

GLOSARIO DE TÉRMINOS

GPS: Es un sistema de radionavegación, basado en el espacio, que proporciona servicios fiables de posicionamiento, navegación. (6)

GRASP: -*General Responsibility Assignment Software Patterns/Patrones de Principios Generales para Asignar Responsabilidades-.*

IDE: -*Integrated Development Environment / Entrono de Desarrollo Integrado-* es un programa informático compuesto por un conjunto de herramientas de programación.

IP: El número que identifica a cada dispositivo dentro de una red.

KML: -*Keyhole Markup Language/* lenguaje de marcado basado en XML para representar datos geográficos en tres dimensiones. A menudo suelen distribuirse comprimidos como ficheros KMZ.

NMEA*:* Es un formato para codificar la información geográfica.

Open Geospatial Consortium (OGC): Es un consorcio compuesto por 479 empresas, agencias gubernamentales y universidades que participan en un proceso de consenso para desarrollar estándares de interfaz disponibles al público. (26)

Raster: Es una imagen formada por una gran matriz de puntos (pixeles) muy pequeños de diferentes colores. Un mapa raster es una imagen que puede ser una foto de satélite o una foto aérea.

Servidor NEBULA: Es un servidor que posee varios módulos, entre ellos el servidor AVL NEBULA que se encarga de gestionar el posicionamiento de los dispositivos móviles y portátiles y enviarla a través de TCP/IP.

Sistema de información geográfica: Un SIG es un conjunto de software y hardware diseñado específicamente para la adquisición, mantenimiento y uso de datos cartográficos. (27)

XML: -*Extensible Markup Language / Lenguaje de Marcas Extensible-* es un metalenguaje extensible de etiquetas.