

UNIVERSIDAD TÉCNICA DEL NORTE



FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CARRERA DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

TRABAJO DE GRADO, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS COMPUTACIONALES

TEMA: Diseño e implementación de un sistema web geo-referenciado para la localización y análisis de información en tiempo real de vehículos, utilizando software libre y cartografía editable.

AUTOR: Galo Javier Pule Revelo

DIRECTORA: Ing. Nancy Cervantes

Ibarra - Ecuador

2013

CERTIFICACIÓN

En calidad de Directora del Trabajo de Grado, titulado: **“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA WEB GEO-REFERENCIADO PARA LA LOCALIZACIÓN Y ANÁLISIS DE INFORMACIÓN EN TIEMPO REAL DE VEHÍCULOS, UTILIZANDO SOFTWARE LIBRE Y CARTOGRAFÍA EDITABLE”**, del señor egresado **Galo Javier Pule Revelo**, portador de la cédula de identidad 1003051693, doy fe que el mencionado Trabajo reúne los requisitos y méritos suficientes para ser sometido a la presentación y evaluación por parte del Jurado Calificador, previo a la obtención del Título de Ingeniero en Sistemas Computacionales, realizándolo con interés profesional y responsabilidad, lo cual certifico en honor a la verdad.

Ing. Nancy Cervantes

DIRECTORA DEL TRABAJO DE GRADO

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR

Ibarra, a los 10 días del mes de abril del 2013.

Yo, Galo Javier Pule Revelo, con cédula de identidad Nro. 100305169-3, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículos 4, 5, 6, en calidad de autor del Trabajo de grado denominado: **“Diseño e implementación de un sistema web geo-referenciado para la localización y análisis de información en tiempo real de vehículos, utilizando software libre y cartografía editable”**, que ha sido desarrollado para optar por el Título de Ingeniero en Sistemas Computacionales, en la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer, plenamente, los derechos cedidos, anteriormente.

En mi condición de autor, me reservo los derechos morales de la obra antes citada, aclarando que el trabajo de Grado, es de mi autoría y que no ha sido presentado para ningún grado o calificación profesional.

En concordancia, suscribo este documento en el momento que hago entrega del Trabajo Final en formato impreso y digital a la Biblioteca de la Universidad Técnica del Norte.

Galo Javier Pule Revelo

C.I. 100305169-3

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

1. IDENTIFICACIÓN DE LA OBRA

La Universidad Técnica del Norte, dentro del proyecto de Repositorio Digital Institucional, determina la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento, dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	100305169-3		
NOMBRES:	Galo Javier Pule Revelo		
DIRECCIÓN:	Río Chimbo 2-95 y Pastaza, Cda. "Los Ceibos"		
EMAIL:	gpulerevelo@gmail.com		
TELÉFONO FIJO:	06 2 954 650	TELÉFONO MÓVIL:	099 3 14 35 66

DATOS DE LA OBRA	
TÍTULO:	Diseño e implementación de un sistema web geo-referenciado para la localización y análisis de información en tiempo real de vehículos, utilizando software libre y cartografía editable
AUTOR:	Galo Javier Pule Revelo
FECHA:	2013-01-28
<i>SOLO PARA TRABAJOS DE GRADO</i>	
PROGRAMA:	<input checked="" type="checkbox"/> PREGRADO <input type="checkbox"/> POSTGRADO
TÍTULO POR EL QUE OPTA:	INGENIERO EN SISTEMAS COMPUTACIONALES
ASESORA/DIRECTORA:	ING. NANCY CERVANTES

2. AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, **Galo Javier Pule Revelo**, con cédula de identidad Nro. **100305169-3**, en calidad de autor y titular de los derechos patrimoniales del Trabajo de Grado, descrito anteriormente; hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación del trabajo en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad, con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior en su Artículo 143.

Firma:

Nombres y apellidos: GALO JAVIER PULE REVELO

Cédula: 100305169-3

Ibarra, a los 10 días del mes de abril del 2013.

DEDICATORIA

A mi madre, Mariana.

Por haber sido la persona que me dio la vida, por haber luchado incansablemente día a día, sin excepción para que yo sea un buen hombre, aquella persona que me ha dado infinito amor.

A mi padre, Galo.

Por haber sido un modelo de hombre, inculcándome siempre los buenos valores a través de su ejemplo, de hombre responsable y de carácter fuerte que me ha permitido ser un hombre de bien, y por su amor incondicional.

A mi hijo, Rommelito, quien también me ha motivado para ser mejor y llegar a culminar mi sueño de ser un profesional capaz y eficiente; y darle el ejemplo para que en el futuro sea una persona adornado con valores morales y éticos al servicio de la sociedad.

AGRADECIMIENTOS

Agradezco a la Ing. Nancy Cervantes que ha sabido confiar en mi persona, por la paciencia y por la aceptación de la dirección de este Trabajo de Grado; al Ing. Enver Jarrín que ha sido el generador de la idea y asesor externo del desarrollo del sistema.

A mis padres, por haberme apoyado durante todo el desarrollo del Trabajo de Grado y por haberme presionado a cada momento.

A mi hermano Fabricio por haberme apoyado, dado ánimos y por haberme hecho escoger esta hermosa Carrera.

A mis familiares por creer en mí que han sido, también, fuente de inspiración para lograr este Trabajo.

Gracias, a todas las personas que directa o indirectamente fueron partícipes de la realización de este Trabajo de Grado.

¡Muchas gracias!

ÍNDICE DE CONTENIDOS

CERTIFICACIÓN	III
CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR	V
AUTORIZACIÓN DE USO Y PUBLICACIÓN	VII
DEDICATORIA.....	IX
AGRADECIMIENTOS.....	X
ÍNDICE DE CONTENIDOS	XI
ÍNDICE DE ILUSTRACIONES	XIII
ÍNDICE DE TABLAS.....	XV
RESUMEN.....	XVI
ABSTRACT	XVIII
INTRODUCCIÓN	1
El Problema.....	3
Antecedentes.....	3
Situación actual	4
Prospectiva	6
Resumen	7
Objetivos.....	7
Objetivo principal	7
Objetivos específicos	7
Alcance del Proyecto	8
Esquema general del sistema	9
Arquitectura funcional del sistema	10
Limitaciones.....	11
Justificación del Proyecto	12
Justificación de las herramientas de desarrollo de software.....	13
Justificación de la metodología de desarrollo de software.....	13
CAPÍTULO I.....	15
1. Marco Teórico	15
1.1. Conceptos previos	15
1.1.1. ¿Qué es un sistema web geo-referenciado?.....	15
1.1.2. ¿Cómo se representa la información en un sistema web geo-referenciado?.....	16
1.2. Servidor web y contenedor de servlets Apache Tomcat.....	19
1.2.1. Estructura de directorios	19
1.2.2. Implementación de una aplicación web.....	20
1.3. Sistema de gestión de base de datos PostgreSQL + PostGIS	21
1.3.1. Características y cumplimiento de normas	22
1.3.2. PostGIS.....	24
1.4. Plataforma de programación Java Enterprise Edition.....	29
1.4.1. Arquitectura de tres capas	30
1.4.2. Arquitectura Modelo Vista Controlador.....	32
1.5. Plataforma de programación Java Standard Edition.....	45
1.5.1. Programación de sockets	47
1.6. JavaScript	55
1.6.1. ECMAScript	57
1.6.2. Ediciones de ECMAScript.....	58

1.6.3.	Compatibilidad de ECMAScript en navegadores web	58
1.6.4.	El Modelo de Objetos de Documento (DOM)	59
1.7.	AJAX	60
1.7.1.	¿Qué es AJAX?	60
1.7.2.	Las tecnologías AJAX.....	60
1.7.3.	Aplicaciones AJAX multinavegador	62
1.7.4.	Fases en la ejecución de una aplicación AJAX	63
1.8.	Visor de mapas	65
1.8.1.	Clases básicas	66
1.8.2.	Controles	71
1.8.3.	Capas.....	75
1.9.	OpenStreetMap	79
1.9.1.	Tecnología.....	79
CAPÍTULO II		81
2.	DESARROLLO DEL SISTEMA	81
2.1.	Ápice arquitectónico.....	81
2.1.1.	Interacción con el cliente	81
2.1.2.	Estudio de las herramientas, tecnologías y prácticas de desarrollo	84
2.1.3.	Construcción del prototipo de estructura y diseño	86
2.2.	Plan de entregas	88
2.2.1.	Priorización de las historias de usuario	88
2.2.2.	Elaboración del plan de entregas	89
2.3.	Construcción	92
2.3.1.	Iteraciones	92
2.4.	Entrega.....	110
2.4.1.	Implementación de la entrega final	110
2.4.2.	Manuales	127
CAPÍTULO III.....		146
3.	CONCLUSIONES Y RECOMENDACIONES.....	147
3.1.	Conclusiones.....	147
3.2.	Recomendaciones.....	148
GLOSARIO		150
REFERENCIAS BIBLIOGRÁFICAS Y DE INTERNET		156
ANEXOS.....		158
ANEXO I: Diagrama de Clases UML del Módulo de Comunicaciones.....		159

ÍNDICE DE ILUSTRACIONES

Ilustración 0.1: Esquema general de sistema	9
Ilustración 0.2: Arquitectura funcional de sistema	11
Ilustración 0.3: Ciclo de vida XP	14
Ilustración 1.1: Información espacial representada en capas.....	15
Ilustración 1.2: Modelos de Datos Raster y Vectorial	17
Ilustración 1.3: Modelos de Datos Raster y Vectorial	17
Ilustración 1.4: Representación de datos en formato Raster	18
Ilustración 1.5: Arquitectura de tres capas	30
Ilustración 1.6: Esquema de una aplicación MVC.	33
Ilustración 1.7: Tecnologías componentes de Java	45
Ilustración 1.8: Comunicación Cliente/Servidor	48
Ilustración 1.9: Pila de software TCP/IP	49
Ilustración 1.10: Asignación TCP/UDP de paquetes entrantes al puerto/proceso adecuado	50
Ilustración 1.11: Establecimiento de ruta de acceso para la comunicación de dos vías entre un cliente y un servidor	51
Ilustración 1.12: Componentes de una implementación JavaScript.....	57
Ilustración 1.13: Lenguajes que implementan ECMAScript	58
Ilustración 1.14: Jerarquía de nodos del DOM.....	59
Ilustración 2.1: Diagrama Entidad-Relación de la base de datos	87
Ilustración 2.2: Prototipo del módulo visor de mapas	88
Ilustración 2.3: Diagrama de Gantt del plan de entregas	91
Ilustración 2.4: Administración de usuarios	95
Ilustración 2.5: Administración de AVL's	95
Ilustración 2.6: Visualización del vehículo en la aplicación	98
Ilustración 2.7: Panel de control.....	101
Ilustración 2.8: Detalles de la última trama recibida	102
Ilustración 2.9: Histórico de velocidades.....	105
Ilustración 2.10: Modificación del plan de entregas en la iteración quinta.....	107
Ilustración 2.11: Creación del archivo compile.sh.....	111
Ilustración 2.12: Resultado de ejecutar el archivo compile.sh	112
Ilustración 2.13: Listado que se genera al ejecutar "make"	112
Ilustración 2.14: Listado que se genera al ejecutar "make install"	113
Ilustración 2.15: Listado que se genera al inicializar el motor de base de datos.....	114
Ilustración 2.16: Listado que se genera al iniciar el motor de base de datos	115
Ilustración 2.17: Listado que se genera al crear una base de datos	115
Ilustración 2.18: Listado que se genera al ejecutar el comando para manejar una base de datos	115
Ilustración 2.19: Comandos para agregar como servicio al motor de base de datos..	117
Ilustración 2.20: Listado que se genera al ejecutar el script "configure" de proj.....	118
Ilustración 2.21: Listado que se genera al ejecutar "make" de proj	118
Ilustración 2.22: Listado que se genera al ejecutar "make install" de proj.....	119
Ilustración 2.23: Listado que se genera al ejecutar el script "configure" de geos	120
Ilustración 2.24: Listado que se genera al ejecutar "make" de geos	120
Ilustración 2.25: Listado que se genera al ejecutar "make install" de geos.....	121

Ilustración 2.26: Listado que se genera al ejecutar el script “configure” de postgres ...	122
Ilustración 2.27: Listado que se genera al ejecutar “make” de postgres	122
Ilustración 2.28: Listado que se genera al ejecutar “make install” de postgres.....	123
Ilustración 2.29: Listado que se genera al ejecutar el script “lwpostgres.sql”	124
Ilustración 2.30: Escritorio de Windows 7	128
Ilustración 2.31: Página de login del sistema.....	128
Ilustración 2.32: Botón “ingresar” de la página de login del sistema	129
Ilustración 2.33: Página “index.jsp” del sistema	129
Ilustración 2.34: Botón para “administrar” el sistema	130
Ilustración 2.35: Página de administración de usuarios del sistema	131
Ilustración 2.36: Formulario para el ingreso de usuarios.....	131
Ilustración 2.37: Botones de edición en la página de administración de usuarios	132
Ilustración 2.38: Formulario para la edición de usuarios	132
Ilustración 2.39: Botones de eliminación en la página de administración de usuarios	133
Ilustración 2.40: Cudro de diálogo de eliminación de usuario.....	133
Ilustración 2.41: Botón de administración de AVLS	134
Ilustración 2.42: Botón para agregar AVLS	134
Ilustración 2.43: Formulario de creación de AVLS.....	135
Ilustración 2.44: Botones de edición en la página de administración de AVLS.....	136
Ilustración 2.45: Formulario de edición de AVLS.....	136
Ilustración 2.46: Botones de eliminación en la página de administración de AVLS.....	137
Ilustración 2.47: Cuadro de diálogo para la eliminación de AVLS.....	137
Ilustración 2.48: Botón de administración de usuarios.....	138
Ilustración 2.49: Página inicial del sistema.....	139
Ilustración 2.50: Ubicación del vehículo.....	139
Ilustración 2.51: Pop-up de los detalles del vehículo	140
Ilustración 2.52: Opción de modo de persecución.....	140
Ilustración 2.53: Botón "Historial" del vehículo	141
Ilustración 2.54: Página de "Historial" del vehículo	142
Ilustración 2.55: Calendario que se despliega para ingresar la fecha inicial del "Historial"	142
Ilustración 2.56: Fecha inicial seleccionada desde el calendario	143
Ilustración 2.57: Fecha final seleccionada desde el calendario	143
Ilustración 2.58: Vista extendida del historial del vehículo.....	144
Ilustración 2.59: Detalles de cada posición del vehículo en el historial.....	145
Ilustración 3.1. Diagrama de Clases UML del Módulo de Comunicaciones.....	161

ÍNDICE DE TABLAS

Tabla 1.1: Límites del RDBMS PostgreSQL	21
Tabla 1.2: Esquema de la tabla CLIENTE.....	37
Tabla 1.3: Compatibilidad de ECMAScript en navegadores web.	58
Tabla 2.1. Historia de usuario Nro. 1	81
Tabla 2.2. Historia de usuario Nro. 2.....	82
Tabla 2.3. Historia de usuario Nro. 3	82
Tabla 2.4. Historia de usuario Nro. 4.....	83
Tabla 2.5. Historia de usuario Nro. 5.....	83
Tabla 2.6: Priorización de las historias de usuario	88
Tabla 2.7. Tarea Nro. 1 de la historia Nro. 1.....	92
Tabla 2.8. Tarea Nro. 2 de la historia Nro. 1.....	92
Tabla 2.9. Tarea Nro. 3 de la historia Nro. 1.....	93
Tabla 2.10. Tarea Nro. 4 de la historia Nro. 1.....	93
Tabla 2.11. Tarea Nro. 5 de la historia Nro. 1.....	94
Tabla 2.12. Caso de prueba Nro. 1 de la historia Nro. 1	95
Tabla 2.13. Caso de prueba Nro. 2 de la historia Nro. 1	96
Tabla 2.14. Tarea Nro. 1 de la historia Nro. 2.....	97
Tabla 2.15. Tarea Nro. 2 de la historia Nro. 2.....	98
Tabla 2.16. Caso de prueba Nro. 1 de la historia Nro. 2	99
Tabla 2.17. Tarea Nro. 1 de la historia Nro. 3.....	100
Tabla 2.18. Tarea Nro. 2 de la historia Nro. 3.....	100
Tabla 2.19. Caso prueba Nro. 1 de la historia Nro. 3	102
Tabla 2.20. Tarea Nro. 1 de la historia Nro. 4.....	103
Tabla 2.21. Tarea Nro. 2 de la historia Nro. 4.....	104
Tabla 2.22. Tarea Nro. 3 de la historia Nro. 4.....	104
Tabla 2.23. Caso de prueba Nro. 1 de la historia Nro. 4	105
Tabla 2.24. Tarea Nro. 1 de la historia Nro. 5.....	106
Tabla 2.25. Caso de prueba Nro. 1 de la historia Nro. 5	107
Tabla 2.26. Caso de prueba Nro. 2 de la historia Nro. 5	108
Tabla 2.27. Tarea Nro. 1 de la historia Nro. 6.....	109
Tabla 2.28. Caso de prueba Nro. 1 de la historia Nro. 6	109

RESUMEN

El presente Trabajo de Grado, trata acerca del diseño e implementación de un sistema web geo-referenciado para la localización y análisis de información en tiempo real de vehículos, utilizando software libre y cartografía editable; en busca de un mecanismo que permita la disminución del robo de vehículos y “secuestros express” en el Ecuador; como también para la mejoría en la rentabilidad de los negocios, dedicados al transporte de personas y/o valores y para que a futuro se siga investigando en tecnologías acordes al tratamiento de datos espaciales. La aplicación web geo-referenciada se encuentra realizada, enteramente, en base a software libre y código abierto; basada en Java, PostgreSQL, Apache Tomcat, JPA, AJAX, JavaScript y OpenLayers, herramientas que han sido utilizadas de la mejor manera, gracias a una metodología ágil de desarrollo y las prácticas más adecuadas, tales como un patrón de diseño MVC, estándares de programación y como también la optimización de recursos tecnológicos, mediante subprocesamiento múltiple, persistencia de datos y la utilización de UDP como el protocolo más óptimo de transmisión de datos para el envío de tramas desde el localizador del vehículo al servidor. Se logró una óptima interacción con el dispositivo localizador importado desde China; también, se obtuvo un buen servicio a lo que se refiere al Servidor Privado Virtual (VPS) ubicado en Madrid, España. Por lo tanto, es necesario desarrollar estos dos tipos de tecnología para un efectivo desarrollo económico y tecnológico que el país requiere.

ABSTRACT

This degree work deals with the design and implementation of a geo-referenced web system for locating and analyzing real-time information of vehicles, using free software and mapping editable; in search of a mechanism for reducing theft vehicles and "express kidnappings" in Ecuador, as well as for the improvement in the profitability of the businesses engaged in the carriage of persons and/or securities and futures, so that in the future people keep doing further research in technologies accord to the treatment of spatial data. The geo-referenced Web application is based entirely on free software and open source, it is based on Java, PostgreSQL, Apache Tomcat, JPA, AJAX, JavaScript and OpenLayers, these tools have been used in the best way through a methodology agile development and best practices, such as a MVC design pattern, programming standards, and also the optimization of technological resources using multithreading, data persistence and the using of UDP as the most optimal protocol data transmission regarding the transmission of data for sending wefts from the vehicle locator towards the server. Was achieved an optimal interaction with the tracking device imported from China; also, was got a good service regarding the Virtual Private Server (VPS) based in Madrid, Spain. Therefore, developing these types of technology is necessary for an effective economic and technological development that the country needs.

INTRODUCCIÓN

El presente Trabajo de Grado, tiene como objetivo principal diseñar e implementar un sistema web geo-referenciado para la localización y análisis de información en tiempo real de vehículos, utilizando software libre y cartografía editable; dicho sistema fue desarrollado en función de que no se ha evidenciado un progreso significativo de la investigación en el campo de la localización de vehículos, a través de un sistema web geo-referenciado, con las características especiales que éste posee (las cuales se detallarán a lo largo de todo el documento), como también por la repercusión que éste tiene en la disminución del índice delincriminal de robo de vehículos y “secuestros express”.

La aplicación hizo uso de software libre para que su utilización sea más común entre los desarrolladores de software, mostrando las ventajas que éste conlleva; como también el uso de cartografía editable, en donde cualquier usuario puede enrolarse en acciones como: modificar, corregir o aumentar información geográfica de uso común, que por consiguiente sea de gran utilidad a nivel de una comunidad internacional.

Además, se ha hecho uso de software libre, debido a que el Gobierno ecuatoriano ha decretado el uso del mismo en toda institución pública, para fortalecer el desarrollo tecnológico del país, evitando utilizar tecnologías importadas.

Con este tipo de aplicaciones se pretende solucionar problemas empresariales, relacionados con el transporte de personas o valores, cuyas soluciones sean capaces de controlar a los conductores de los vehículos o monitorear los valores de las empresas. De esta manera, se puede incrementar la rentabilidad de negocios de este tipo y proyectarse hacia el futuro con más soluciones empresariales.

El Ecuador se encuentra dando sus primeros pasos hacia el desarrollo tecnológico, ya que se ha evidenciado que el país tiene un gran potencial en lo que se refiere a software financiero-contable, exportando productos de este tipo hacia el extranjero, donde existe gran demanda de software ecuatoriano; pero por otra parte, se está iniciando el proceso de desarrollo tecnológico en el área de software de tipo

geográfico como el del presente trabajo, que tiene el propósito de incentivar el desarrollo de proyectos como éste; puesto que un problema que sufre el Ecuador es la falta de tecnologías propias, debido a la escasa inversión en investigación, que conlleva a un gasto de tiempo y dinero por encima de las posibilidades de un emprendedor; por ende, las empresas optan por utilizar tecnologías importadas.

Siendo así, se ha procurado utilizar las mejores tecnologías existentes en el ámbito del desarrollo de software para que se puedan obtener y explotar al máximo todas las funcionalidades que un sistema web geo-referenciado debería poseer; de igual manera, se ha procurado optimizar al máximo los recursos tecnológicos, aplicando técnicas como la utilización de subprocesamiento múltiple y otras.

Se ha utilizado el mejor DBMS¹ del mundo: PostgreSQL más su extensión PostGIS para el procesamiento de datos espaciales, conjuntamente, con la plataforma de programación Java, cuyo núcleo es lo mejor en lo que refiere a seguridad informática y versatilidad; también, la utilización de AJAX y JavaScript, como herramientas para el desarrollo de una interfaz de cliente rica, y la utilización de OpenLayers como librería utilitaria para el manejo del visor del mapa del sistema web geo-referenciado. Todo esto es software libre y de código abierto.

Como prácticas de desarrollo, se han definido varias estrategias, tales como: una metodología ágil de desarrollo XP, un patrón de diseño MVC y algunos estándares de programación; todo esto, ayuda a que las herramientas, anteriormente mencionadas, se las explote de mejor manera; ya que, no hace falta una extensa documentación para este tipo de proyectos; el patrón de diseño se ajusta, perfectamente, a la plataforma de programación que se está utilizando y los estándares de programación que permiten que el código fuente y las tablas de base de datos se encuentren organizadas; logrando así, que el trabajo del desarrollador se torne cómodo y placentero.

A continuación se presenta, brevemente, cómo se encuentran constituidos los capítulos del presente trabajo de grado:

¹ Data Base Management System: Sistema de Gestión de Base de Datos (SGBD)

La presente Introducción, comprende una explicación del problema, los objetivos, el alcance y la justificación del trabajo de grado. Se explica el porqué del presente trabajo y las causas que influyeron para su realización; por ende, las soluciones y los objetivos que se pretenden lograr para que el problema no persista, como también sus limitaciones y su alcance.

En el capítulo I, se detalla el fundamento teórico de las herramientas utilizadas, lo cual ha sido una pequeña parte de todo el conocimiento adquirido, durante el transcurso del pregrado; en sí, es la teoría a la que el desarrollo del trabajo se refirió durante todo su proceso. Sin lugar a duda, haciendo uso, enteramente, de software libre y las mejores prácticas y herramientas disponibles, en aquel momento.

El capítulo II trata acerca del proceso de desarrollo del sistema; aquí se explica a detalle cómo se consiguió realizar la solución informática, acorde a la metodología propuesta y todos los pasos que se siguió; al igual que los cambios que se realizó en el transcurso de la construcción del sistema. Además, en este capítulo se muestran los manuales del administrador y del usuario final y también el cómo se hizo la entrega final.

El capítulo III, contiene las conclusiones a las que el presente trabajo llegó, después de una ardua labor; también se encuentran las recomendaciones que se plantea, luego de experimentar con este tipo de proyecto.

El Problema

Antecedentes

A medida que las tecnologías de la información y la comunicación se desarrollaron a pasos agigantados en las últimas décadas, las necesidades tecnológicas del ser humano aumentaron de forma geométrica, tales como: la necesidad trivial del procesamiento rápido y eficaz de datos numéricos y alfabéticos; y, últimamente, la de localizar una persona o cosa, por medio de un sistema computacional geográfico.

Esta última, que surgió en los últimos años se ha desarrollado, de tal manera, que hoy en día se ha convertido en un servicio, indispensable, para algunos. Sin embargo, a

causa de que la tecnología se mantiene en constante evolución de manera fugaz, todavía no se ha logrado un progreso ideal en la investigación de este ámbito.

A causa de este fallido progreso de la investigación en el ámbito de la geo-localización, en el Ecuador no se han desarrollado soluciones eficientes, relativo a la optimización de costos y de seguridad informática que se ajusten a las necesidades reales de clientes potenciales. Adyacente a esto, surge la necesidad de adquirir soluciones extranjeras; por lo cual, se detiene la utopía de que la educación sea garantía de la creación de fuentes de trabajo.

Por otra parte, la situación delincuencial en el Ecuador es dramática, durante décadas ha existido un alto índice, respecto al robo de vehículos; así como también, los denominados “Secuestros Express” que tienen sumida en la desesperación a la sociedad ecuatoriana; y los esfuerzos que hace el Gobierno por controlar y prevenir no son lo suficientemente efectivos.

Por otro lado, las empresas que lucran del transporte de personas o valores, no visualizan la información real de la ubicación de sus vehículos de forma eficaz, por lo que se someten a la pérdida de rentabilidad de su negocio. A todo lo descrito se añade, la falta de seguridad y control de personas de la tercera edad, niños, diplomáticos, discapacitados; así como también, el monitoreo de vehículos de transporte terrestre, aéreo y marítimo; lo cual hace imprescindible la utilización de AVLS² para la localización de un objeto en particular.

Situación actual

Una de las políticas establecidas en el actual Gobierno³, acerca de la implementación de software en las instituciones públicas, es la de adquirir, únicamente, Software Libre; por lo que en el transcurso del último año y del actual, todas estas instituciones se encuentran en el proceso de migración del software; por ejemplo, han tenido que reemplazar todas las herramientas desarrolladas por *Microsoft Corporation*[®] por herramientas libres.

² Automatic Vehicle Locator: Localizador Automático de Vehículos

³ Gobierno presidido por el Presidente de la República Rafael Correa Delgado.

En efecto, se abre una puerta en el mercado para el software geográfico libre. Software Libre no quiere decir gratuito; se trata de software libre de licencias y con estándares abiertos, que permitan la futura manipulación del código fuente para que el software solucione necesidades más específicas; con esta política de Estado se le ha quitado las cadenas al software en general.

Las instituciones públicas del país tienen la necesidad de controlar y monitorear los vehículos que están a su responsabilidad y tienen que estar debidamente, monitoreados, tanto los conductores como los vehículos; por lo que se hace imprescindible la adquisición de un sistema web geográfico, para la localización, seguimiento y análisis de información de los mismos. El software libre para la localización y análisis de información en tiempo real de vehículos, hasta el día de hoy, no ha logrado convertirse en “Producto Estrella” de las grandes compañías de software de Latino América, por la falta de una investigación adecuada, en este ámbito tecnológico. Los requerimientos de las instituciones públicas para este tipo de soluciones informáticas se pueden encontrar en el portal web de compras públicas <http://www.compraspublicas.gob.ec>.

Existe gran demanda de soluciones informáticas para la seguridad y el rastreo de vehículos que sean personalizables, en tal medida que se puedan ajustar a los requerimientos más específicos del cliente; es decir, que la solución se ajuste a las necesidades reales de éstas. En lo que se refiere a software en general, las instituciones públicas se encuentran en la obligación de contratar profesionales capacitados para la ejecución de proyectos que conlleven el rediseño del software para que éste se ajuste a sus requerimientos.

Por ende, se requiere que estas soluciones sean de código abierto, ya que, la mayoría de las aplicaciones comerciales tienen estándares cerrados o están desarrolladas con código propietario, como es el caso de las aplicaciones desarrolladas con herramientas de *Microsoft Corporation*[®], barrera que a larga suprime el tan anhelado progreso tecnológico del país, porque la sociedad ecuatoriana se hace dependiente de tecnología foránea al no poseer tecnología propia, haciendo que la porción del Presupuesto General del Estado destinado a las instituciones públicas para la adquisición de tecnología finalice en el extranjero y no en el Ecuador; toda vez que las

necesidades reales de estas instituciones no pueden ser cubiertas de manera ideal a causa de que los profesionales empleados no pueden modificar el software para su óptima aplicación, privando así a las instituciones públicas de la total propiedad intelectual de estas soluciones tecnológicas.

Prospectiva

De continuar con la falta de aplicaciones informáticas para el control y monitoreo de vehículos y/o personas, en las instituciones del Estado; el presupuesto asignado para la adquisición de éstas, tendrá un fin en el extranjero, debido a que software es de este tipo desarrollado en el Ecuador no existirá, por falta de investigación en el campo del desarrollo de software geográfico; o por lo contrario, existirán, pero serán soluciones informáticas que conlleven compra de licenciamiento, que implica que el código fuente sea privativo o a su vez sean desarrolladas con estándares cerrados que impidan la manipulación del código, para ajustar la solución a los requerimientos reales de estas instituciones.

En las empresas que lucran del transporte de personas o de valores, no habrá una estrategia eficaz que promueva la rentabilidad y estrategia que permita una constante visualización de la posición y un efectivo control de los vehículos, con el fin de minimizar tiempo y costos a la empresa; estrategia que se acoplaría también a las instituciones públicas, así como para las personas particulares que deseen garantizar el posicionamiento de su vehículo en caso de que este sea robado, o peor aún, esta persona haya sido víctima de un secuestro, como ha sido habitual en el transcurso de las dos últimas décadas.

En conclusión, se privaría a la población de esta estrategia tecnológica, a la vez que se impediría el desarrollo científico en el país en esta área del conocimiento. Cabe mencionar, que para que una sociedad se desarrolle, positivamente; y, los tradicionales y rutinarios dogmas de atraso tecnológico que han gobernado a la sociedad ecuatoriana, durante siglos, sufran un cambio radical, ésta debe llevar consigo la ciencia como un ideal para alcanzar el éxito.

Resumen

En el Ecuador existe un alto índice, respecto al robo de vehículos y por ende una terrible inseguridad ciudadana; al igual que la falta de control y monitoreo de vehículos, pertenecientes a las instituciones del Estado que se movilizan por vía terrestre, aérea o marítima; como también, la nula seguridad para las personas vulnerables como: personas de la tercera edad, niños, discapacitados, diplomáticos u otros; lo que afecta a la calidad de vida de todo ciudadano, privándoles de estrategias alternativas para controlar la inseguridad.

Por otro lado, no existen herramientas tecnológicas que permitan monitorear y controlar la actividad productiva de empresas que lucran del transporte de personas o valores. Por lo tanto, es evidente la falta de aplicaciones informáticas destinadas al control y monitoreo de vehículos de una manera interactiva y amigable entre el usuario y el sistema para la seguridad o la rentabilidad de negocios, que a su vez sean de bajo coste e independientes de tecnología foránea.

Objetivos

Objetivo principal

Diseñar, desarrollar e implementar un sistema web geo-referenciado para la localización y análisis de información en tiempo real de vehículos, utilizando software libre y cartografía editable; para solucionar parte de la problemática de seguridad que afecta al país y mejorar la rentabilidad de negocios de transporte de valores y/o personas; en un tiempo límite de seis meses.

Objetivos específicos

- Diseñar el sistema de acuerdo a una arquitectura tecnológica robusta, estable y comprobada, enfocándose a la solución del problema, utilizado de la manera más eficaz la infraestructura de hardware con la que se interactúa.
- Desarrollar el sistema utilizando software libre en su totalidad, en base a las mejores prácticas de desarrollo y a los estándares establecidos en la planificación

del mismo; de manera que el sistema sea extensible y fácilmente mantenible en el transcurso del tiempo.

- Implementar el sistema en base a los resultados obtenidos en las pruebas de funcionalidad realizadas.

Alcance del Proyecto

Respecto a los dispositivos empleados para la localización por GPS⁴, el sistema será desarrollado para la compatibilidad con los AVLs VT-310 de origen chino por su bajo costo y gran demanda en el país, a causa de sus características versátiles y de última tecnología. En lo que refiere a la cartografía web se utilizará el servidor de mapas de terceros: OpenStreetMap, por ser editable y software libre.

La aplicación será multiusuario, donde cada usuario podrá disponer de un solo AVL, sea éste: vehículo o persona, para su respectivo monitoreo. Se podrá rastrear el vehículo o la persona en tiempo real, permitiéndole al usuario observar un histórico de posiciones y velocidades que han transcurrido en un mes. Cabe señalar que la aplicación soportará un número ilimitado de usuarios.

Dado que se trata de un AVL básico por su costo, la aplicación medirá únicamente datos como: distancia recorrida, velocidad promedio, dirección de traslado, altitud geográfica y posición actual en un intervalo de tiempo determinado.

Se configurará un servidor Linux con las aplicaciones, frameworks, librerías Open Source compatibles con sistemas GIS; para la ejecución de pruebas locales. Posteriormente, se realizará la misma configuración en un VPS en línea con IP pública, para que la aplicación reciba los datos reales que genere el AVL, que por consiguiente sean procesados por el sistema. La arquitectura será cliente-servidor de manera local como parte de las pruebas de desarrollo y distribuida de manera on-line para el monitoreo real del vehículo.

La aplicación contendrá los siguientes módulos:

⁴ Global Position System: Sistema de Posición Global

- Módulo de comunicaciones, para la recepción y descifrado de la información enviada por el AVL, para el posterior almacenamiento en la Base de Datos.
- Módulo de seguridad de la aplicación, para la autenticación de los usuarios.
- Módulo de administración de AVLs, para seleccionar y obtener la información de un AVL en particular, misma que se actualizará cada tres minutos.
- Módulo visor de mapas y cartografía, para la localización visual del AVL en un mapa actualizado y detallado.
- Módulo de reportes, para consultas de históricos referentes a distancias, velocidades y posiciones.

Esquema general del sistema

A continuación se describen los módulos de la aplicación mediante el Esquema General del Sistema, véase la Ilustración 0.1 de a continuación.

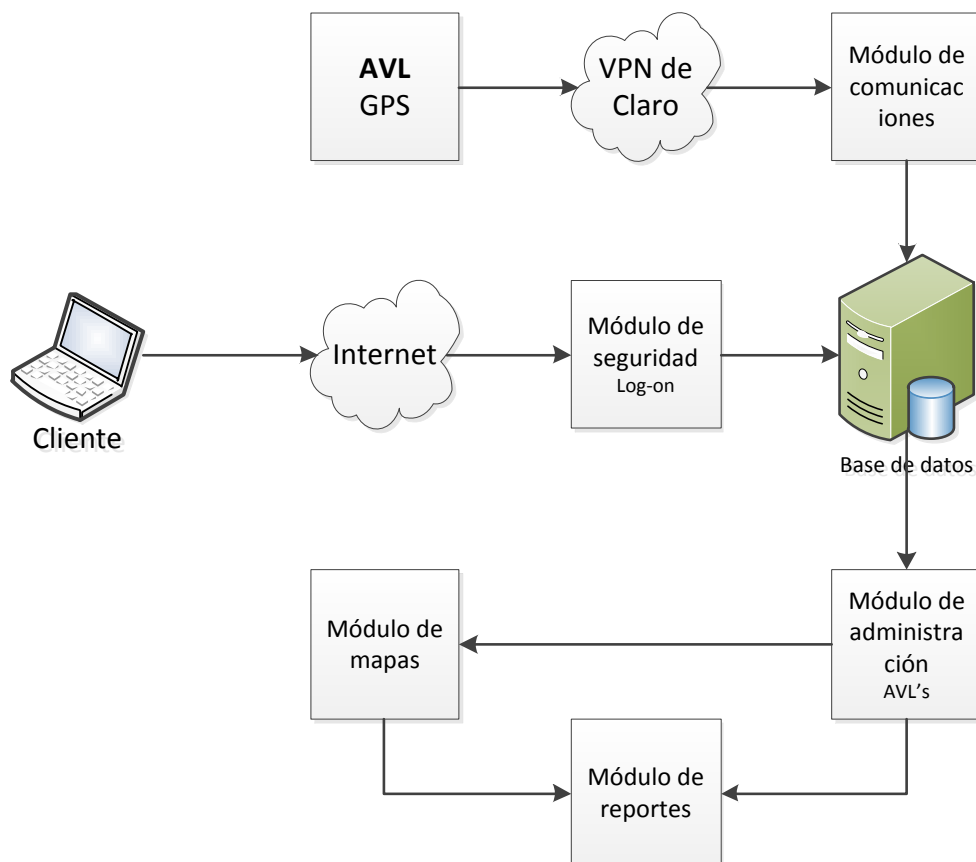


Ilustración 0.1: Esquema general de sistema. Fuente propia.

Arquitectura funcional del sistema

La aplicación depende del Sistema de Posicionamiento Global (GPS) liberado por la Armada de los Estados Unidos en 1999 para uso civil; cuya importancia es trascendental para la construcción de sistemas web geo-referenciados para la localización de vehículos, como también es indispensable para los desarrolladores de este tipo de software. Este sistema se encuentra formado por ocho satélites artificiales que circundan la Tierra. El GPS es el sistema que envía los geo-datos necesarios para el rastreo del objeto como la posición (latitud y longitud), velocidad promedio, distancia recorrida, dirección de traslado, altitud geográfica en un intervalo de tiempo determinado por el desarrollador del software, que serán receptados y posteriormente almacenados en la memoria del AVL instalado en el vehículo. Siendo ésta la parte más compleja y fundamental para que los datos almacenados en el AVL sean procesados por el sistema web geo-referenciado.

Obtenidos y almacenados los geo-datos en la memoria del AVL, éste será configurado de manera que envíe los geo-datos al VPS mediante la VPN de alguna operadora telefónica, para luego ser almacenados en la base de datos de la aplicación web. En consecuencia, con anterioridad, al VPS se le instalará y configurará los siguientes componentes de software:

- Sistema Operativo: Linux CentOS 5.5.
- Base de datos: PostgreSQL 8.2 + PostGIS 1.3.5.
- Servidor web y contenedor de servlets: Apache Tomcat 7.0.2.
- Máquina virtual de Java: Java Runtime Environment (JRE) 7.

Así, el VPS brindará todos los servicios necesarios para que el cliente disponga de toda la funcionalidad del sistema web geo-referenciado para la localización y análisis de información de su vehículo. A continuación se detalla gráficamente la arquitectura funcional del sistema, véase la Ilustración 0.2 a continuación.

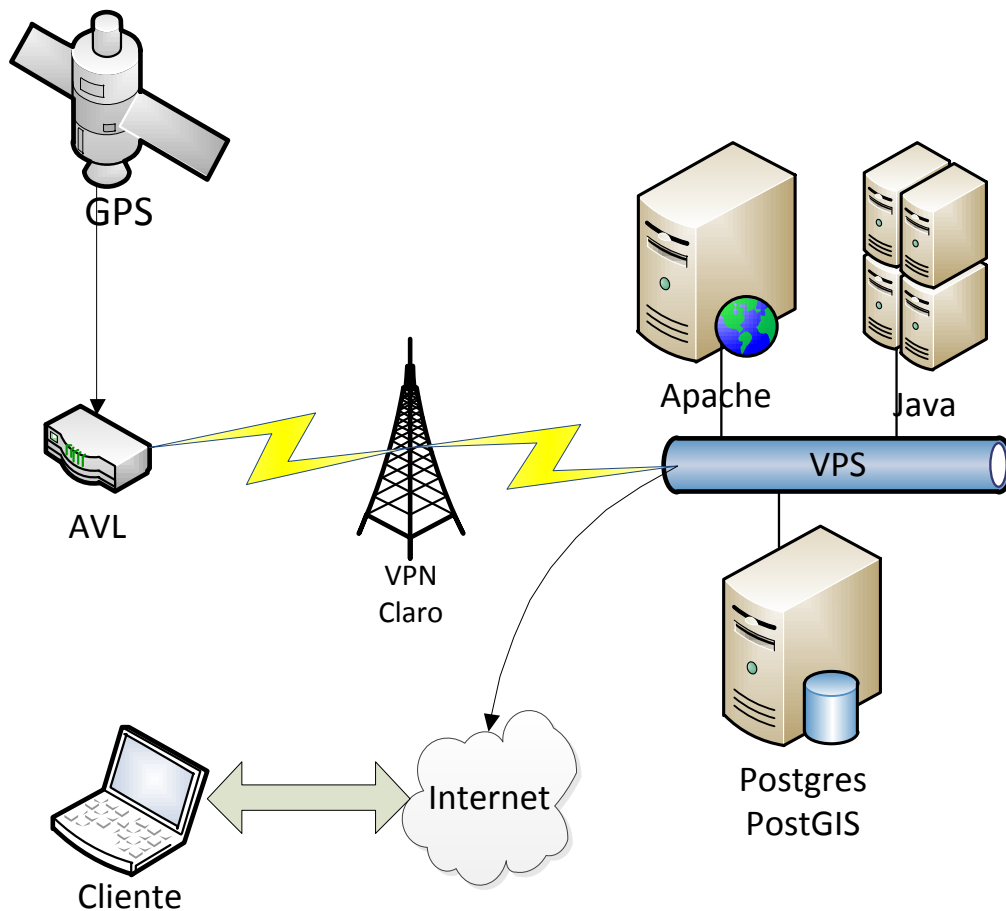


Ilustración 0.2: Arquitectura funcional de sistema. Fuente propia.

Limitaciones

La aplicación no se diseñó para el soporte de flotas de vehículos, esto quiere decir que la aplicación únicamente controla a un solo vehículo por cada usuario; por ende no se puede crear geo-cercas, ni geo-rutas para el análisis de la posición de varios vehículos en conjunto. Tampoco se lo diseñó para acoplarse a cartografía suministrada por el usuario, por ejemplo cartografía suministrada por el usuario como: el mapa de Pimampiro, que en la actualidad todavía no está geo-referenciado en la cartografía de OpenStreetMap.

Dadas las capacidades del AVL que se utilizó, no se permite la administración del vehículo por comandos, es decir, la interacción física con el vehículo, tales como la apertura de puertas, apagado de motor, el botón de pánico y otros. Tampoco se maneja un sistema de alertas que reporte al usuario, por ejemplo, cuando el vehículo exceda una velocidad determinada o cuando la aplicación no reciba a tiempo

información del AVL, sea esto a causa de una falla en el hardware, una inhabilitación causada del dispositivo, o como también la falta de señal en la red de la operadora telefónica, o si el vehículo estuviese fuera del alcance del GPS (más conocido como zona blindada de GPS).

Se excluyó la interacción de la aplicación con servicios celulares como: SMS y aplicaciones móviles. Se excluyó también el desarrollo de un módulo de usuarios y permisos. Los costos de los chips de telefonía celular están definidos por la operadora celular, al igual que el costo del megabyte. Cabe recalcar que, el AVL no guarda en su memoria el histórico de posiciones en el sector que carezca de señal celular y mucho menos de señal satelital.

Justificación del Proyecto

El presente trabajo se justifica porque es necesario el desarrollo de un sistema web geo-referenciado para la localización y análisis de información en tiempo real de vehículos que sea nacional e independiente, porque los sistemas que existen de este tipo en el país son adquiridos en el extranjero o a su vez desarrollados en el país pero con dependencia tecnológica del mismo; tal es el caso de la utilización de la cartografía de *Google Maps*® para su desarrollo, o peor aún software desarrollado con herramientas privativas como las de *Microsoft Corporation*®, lo que conlleva a la construcción de software con estándares cerrados, impidiendo así, su manipulación. Esta aplicación pretende independizar a la sociedad ecuatoriana de tecnología foránea, aislándola de soporte técnico del extranjero, como también de la compra de licenciamiento de software de corporaciones extranjeras.

De igual forma, existen empresas en el país que brindan el servicio de geo-localización utilizando aplicaciones foráneas, empresas como SmartCargo, Hunter, Trail y otras; que dependen del contrato de licenciamiento y del soporte técnico extranjero. En consecuencia, estas deben incrementar sus inversiones, haciendo que estos costos se los transfiera al consumidor final, dando como resultado el encarecimiento de este servicio. Lo que se quiere con este trabajo es abaratar los costos de este tipo de software, al igual que hacer a las empresas que lo adquieran, propietarias del código fuente para que a futuro estén en la capacidad de manipularlo para que el software se

ajuste a las necesidades críticas de la empresa, mas no a las necesidades esporádicas de la misma.

- Beneficiarios

Propietarios de vehículos privados, instituciones propietarias de vehículos públicos o privados, personas de la tercera edad, niños, diplomáticos, discapacitados, personas que quieran monitorear sus bienes, desarrolladores que quieran aportar con código fuente a la aplicación, ya que se trata de software de código abierto.

Justificación de las herramientas de desarrollo de software

Se utilizó el IDE (Interface Development Enviroment) Eclipse® Indigo 3.7.1 con su plugin para Java EE⁵ versión 1.4.1, para el desarrollo de toda la aplicación, por ser software libre y una potente herramienta para la escritura y organización del código fuente; como también la para construcción y el despliegue de la aplicación y sus pruebas unitarias.

Justificación de la metodología de desarrollo de software

Se empleó la metodología ágil de desarrollo de software “Programación Extrema”, más conocida como “XP” (Extreme Programming). Se utilizó ésta por ser una metodología ágil, basada en una serie de buenos valores y mejores prácticas que persiguen el objetivo de aumentar la productividad.

Este modelo de programación se basa en la recopilación y síntesis de metodologías tradicionales, en la que se da prioridad a los trabajos con resultados directos, disminuyendo notablemente el protocolo de la documentación. Menos documentación y más software funcionando.

A continuación, la Ilustración 0.3 explica el ciclo de vida de la metodología XP.

⁵ Java Enterprise Edition: Java Edición Empresarial

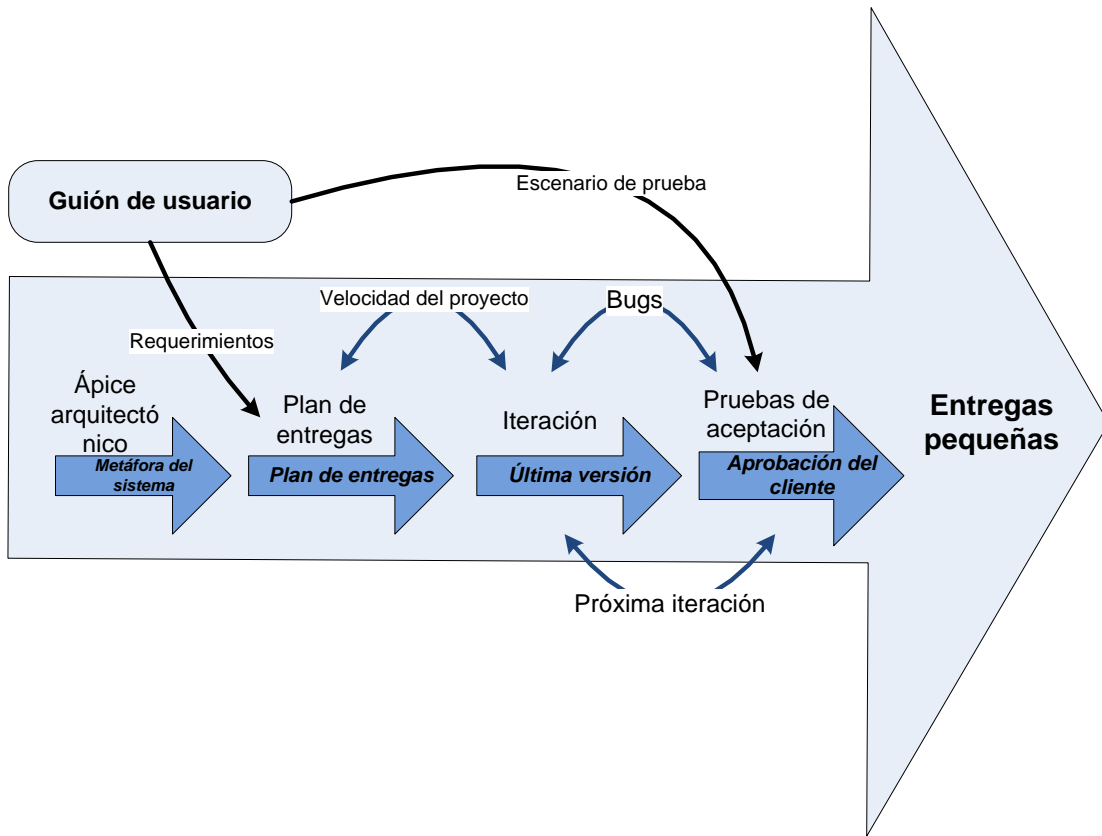


Ilustración 0.3: Ciclo de vida XP. Fuente: www.extremeprogramming.org.

CAPÍTULO I

Marco Teórico

1.1. Conceptos previos

1.1.1. ¿Qué es un sistema web geo-referenciado?

Un sistema web geo-referenciado es un Sistema de Información Geográfica (GIS) que opera en Internet. El término GIS o SIG se aplica actualmente a los sistemas computarizados para el almacenamiento y el análisis de datos, mediante equipos y programas especializados en el manejo de datos espaciales de referencia geográfica. Existen diversas definiciones para caracterizar un GIS. “Un GIS es una ‘Herramienta computacional’ compuesta por equipos, programas, datos geo-referenciados y usuarios que requieren organizar, analizar, automatizar procesos y producir información” (Solivelles, 2012).

En general la información espacial se representa en forma de “capas”, en los que se describen por ejemplo la topografía, la disponibilidad de agua, los suelos, los bosques y praderas, el clima, la geología, la población, la propiedad de la tierra, los límites administrativos, la infraestructura (carreteras, vías férreas, sistemas de electricidad o de comunicaciones), como se representa en la siguiente figura (Solivelles, 2012):

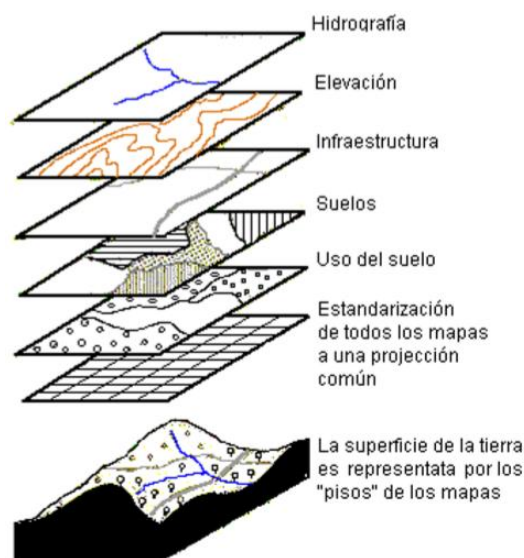


Ilustración 1.1: Información espacial representada en capas. Fuente: (Solivelles, 2012).

Así, se puede enumerar algunas de las principales características de un GIS:

- Manejo de grandes volúmenes de información.
- Posibilidad de información de distintas fuentes y escalas.
- Rapidez en el procesamiento de la información y obtención de productos cartográficos.
- Capacidad de modelar información.
- Manejo de información geo-referenciada.

1.1.2. ¿Cómo se representa la información en un sistema web geo-referenciado?

Existen dos Modelos de datos espaciales que permiten representar la información contenida en un GIS (Solivelles, 2012)⁶:

- **Modelo Vectorial:** En él, los objetos y sus atributos (condiciones) son representados por puntos y líneas que definen sus límites. La posición de cada objeto es definida por su localización en un “mapa” que es organizado a través de un sistema de coordenadas de referencia. Cada posición en el mapa tiene un único valor de coordenada.
- **Modelo Raster:** El espacio es dividido regularmente en “celdas” (usualmente formadas por cuadrados). La localización de los objetos geográficos y sus atributos, está definida por la posición que las celdas ocupan en las columnas y las filas. El área que cada celda representa define la resolución de la información.

Los estudios realizados en el presente trabajo se aplican sobre un GIS basado en el modelo Vectorial de datos espaciales. A continuación, de acuerdo con la referencia bibliográfica de (Solivelles, 2012) se explica mediante un ejemplo los dos modelos antes mencionados.

La Ilustración 1.2 muestra las características principales de estos dos modelos, al representar una porción de terreno del mundo real. En ella existen dos bosques, uno de pino y otro de eucaliptus que se encuentran espacialmente separados por un río que cruza la porción de terreno. Junto al río existe una casa. En el caso del modelo

⁶ Solivelles, U. (2012). *Sistemas de Información Geográfica (SIG) y percepción remota*.

Raster cada uno de estos elementos son representados por conjuntos de celdas vecinas con una posición geográfica, dada por la posición fila - columna de cada una que forman parte de un arreglo de celdas que representa toda el área de interés.

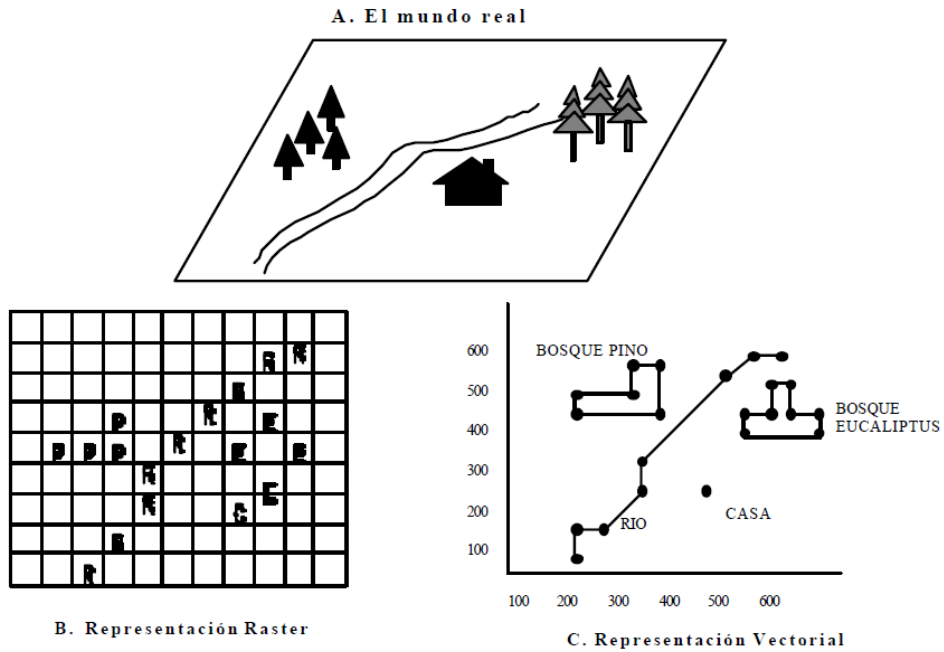


Ilustración 1.2: Modelos de Datos Raster y Vectorial. Fuente: (Solivellas, 2012).

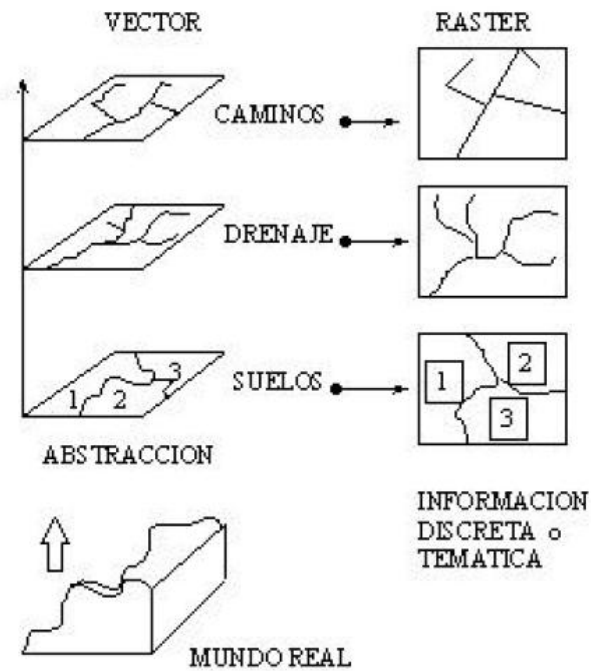


Ilustración 1.3: Modelos de Datos Raster y Vectorial. Fuente: (Solivellas, 2012).

Estas celdas contienen un valor que indicará qué se está representando en ella. Este valor o atributo de la celda puede representar una porción de río (r), bosque de Eucaliptus (e) o de Pinos (p).

La fidelidad con que se representa una situación del mundo real, depende de la resolución utilizada con este arreglo de celdas. Puntualmente, la resolución está dada por el área real representada por la celda. Además, se asume que el valor de cada atributo asociado a ésta es homogéneo en toda el área que representa. En este sentido, este modelo de datos espaciales una discretización del mundo real, lo que permite hacer representaciones temáticas de la realidad, como se muestra en la Ilustración 1.3 de arriba.

En la Ilustración 1.4, además se puede mostrar el hecho que cada celda de cada capa tiene una celda correspondiente en cada una de las otras capas, producto de la geo-referenciación.

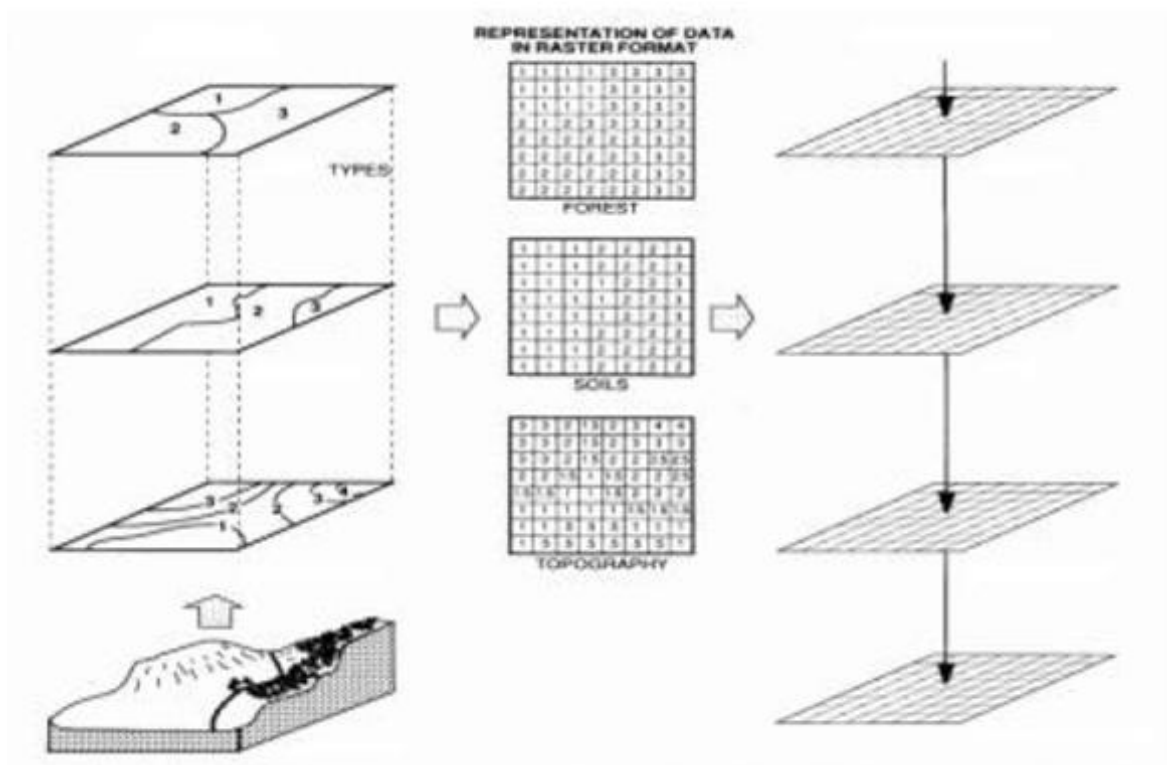


Ilustración 1.4: Representación de datos en formato Raster. Fuente: (Solivelles, 2012).

1.2. Servidor web y contenedor de servlets Apache Tomcat

El servidor web y contenedor de servlets cabe en esta sección ya que es el software que permite que un sistema geo-referenciado funcione en la web, es aquel que se encarga de hacer el despliegue de la aplicación web desarrollada en lenguaje Java y bajo una plataforma Java Enterprise Edition (J2EE) multicapa. Para el caso del presente trabajo se hizo uso del servidor web y contenedor de servlets Apache Tomcat, ya que hacer uso de un servidor de aplicaciones era innecesario porque éste posee características adicionales que no se ajustan a este tipo de aplicaciones. El contenedor de servlets se encarga de interpretar “servlets” (clase Java que implementa interfaces que permiten la recepción de datos desde el cliente y envío de datos hacia el mismo, entre otros beneficios), procesándolos mediante la Máquina Virtual de Java instalada en el servidor, por lo que el procesamiento de datos lo lleva a cabo el servidor, en lugar del cliente. Por otra parte, el servidor web se encarga del alojamiento y procesamiento de las páginas de servidor Java (JSP⁷) y otros componentes web de la aplicación en general, tales como páginas HTML, XHTML, DHTML, XML, si es que fuese el caso.

Además, el proyecto Apache Tomcat es una implementación de software libre de las tecnologías Java Servlet y JavaServer Pages (The Apache Software Foundation, 2012). Se utilizó la versión 7.0.34, siendo el último lanzamiento hasta el momento, el cual implementa las especificaciones para Servlet 3.0 y JavaServer Pages 2.2 según la Java Community Process⁸, e incluye muchas características adicionales que la hacen una plataforma útil para desarrollar y desplegar aplicaciones web y servicios web (web services).

1.2.1. Estructura de directorios

Apache Tomcat despliega en la ruta raíz los siguientes directorios en su instalación(The Apache Software Foundation, 2012):

- **bin** – arranque, cierre, y otros scripts y ejecutables.
- **common** – clases comunes que pueden utilizar Catalina y las aplicaciones web.

⁷ Java Server Page: Página de Servidor Java

⁸ El “Proceso de la Comunidad Java” es el mecanismo para desarrollar especificaciones técnicas estándares para tecnología Java.

- **conf** – ficheros XML y los correspondientes DTD para la configuración de Tomcat.
- **logs** – logs de Catalina y de las aplicaciones.
- **server** – clases utilizadas solamente por Catalina.
- **shared** – clases compartidas por todas las aplicaciones web.
- **webapps** – directorio que contiene las aplicaciones web.
- **work** – almacenamiento temporal de ficheros y directorios.

1.2.2. Implementación de una aplicación web

El objetivo principal de instalar un servidor web y contenedor de servlets como Apache Tomcat es la de implementar aplicaciones web en él.

La implementación es el término que se utiliza para el proceso de instalación de una aplicación web (ya sea un WAR de terceros o una propia aplicación web personalizada) en el servidor Tomcat.

El despliegue de aplicaciones web puede llevarse a cabo en un número de maneras dentro del servidor Tomcat(The Apache Software Foundation, 2012)⁹.

- Estáticamente, la aplicación web se configura antes de que Tomcat se haya iniciado.
- Dinámicamente, en conjunto con la aplicación web Tomcat Manager o la manipulación de aplicaciones web ya implementadas.

El Tomcat Manager es una herramienta que permite a las aplicaciones web basadas en URL características de implementación. También hay una herramienta llamada “el cliente de implementación”, que es un shell de comandos basado en script que interactúa con el Tomcat Manager, que ofrece funcionalidades adicionales, como la recopilación y validación de las aplicaciones web así como el empaquetado de la aplicación en el archivo de aplicación web (WAR) (The Apache Software Foundation, 2012). Aunque para el presente trabajo se empaquetó la aplicación en un “Web Archive” (WAR) y posteriormente se implementó en el servidor Tomcat de manera estática.

⁹ The Apache Software Foundation. (2012). *Apache Tomcat*. Recuperado el 10 de Julio de 2012, de <http://tomcat.apache.org/>.

1.3. Sistema de gestión de base de datos PostgreSQL + PostGIS

Como sistema de gestión de base de datos relacionales (RDBMS) se escogió PostgreSQL, debido a que hoy en día es el RDBMS más utilizado en el desarrollo de software, ya que tiene gran rentabilidad en su transaccionabilidad y una gran infinidad de características que lo hacen aún más potente; superando así a marcas de gran renombre como Oracle© y Microsoft©.

Incluye la mayor parte de los tipos de datos SQL:2008, incluyendo INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL y TIMESTAMP. También, soporta almacenamiento de objetos binarios grandes, incluyendo imágenes, sonidos o vídeo. Cuenta con interfaces nativas de programación C/C++, Java, .NET, Perl, Python, Ruby, Tcl, ODBC, entre otros, y una documentación excepcional (PostgreSQL Global Development Group, 2011)¹⁰.

Una base de datos de clase empresarial, PostgreSQL cuenta con características avanzadas tales como control de concurrencia multi-versión (MVCC), recuperación de punto en el tiempo, espacios de tabla (tablespaces), replicación asincrónica, transacciones anidadas (puntos de retorno), copias de seguridad en caliente o en línea, un sofisticado planificador/optimizador de consulta, y escritura por delante de registro para la tolerancia de fallos. Es compatible con varios conjuntos de caracteres internacionales, codificaciones de caracteres multibyte, Unicode. Es altamente escalable, tanto en la gran cantidad de datos que puede manejar y en el número de usuarios concurrentes que puede acomodar. Hay sistemas activos con PostgreSQL en entornos de producción que manejan más de 4 terabytes de datos. Algunos de los límites de PostgreSQL generales están incluidas en la **Tabla 1.1** siguiente.

Tabla 1.1: Límites del RDBMS PostgreSQL. Fuente:(PostgreSQL Global Development Group, 2011).

Límite	Valor
Tamaño máximo de base de datos	Ilimitada
Tamaño máximo de tabla	32 TB

¹⁰ PostgreSQL Global Development Group. (2011). About. Recuperado en Julio de 2012, de <http://www.postgresql.org/about/>.

Tamaño máximo de fila	1.6 TB
Tamaño máximo de campo	1GB
Número máximo de filas por tabla	Ilimitado
Número máximo de columnas por tabla	250 – 1600 dependiendo de los tipos de columna
Número máximo de índices por tabla	Ilimitado

1.3.1. Características y cumplimiento de normas

En el sitio web oficial de PostgreSQL se exponen algunas de las características más relevantes que convierten a éste en un poderoso sistema de gestión de base datos, como también se explica el cumplimiento de las normas del lenguaje de consulta estructurado (SQL) (PostgreSQL Global Development Group, 2011).

PostgreSQL se enorgullece del cumplimiento de las normas. Su puesta en práctica de SQL muy conforme con la norma ANSI-SQL: 2008 estándar. Cuenta con soporte completo para subconsultas (incluyendo subconsultas en la cláusula FROM), lectura cometida (read-commited) y los niveles de aislamiento de transacciones serializables. Además, PostgreSQL tiene un catálogo de sistema completamente relacional por lo que soporta varios esquemas de base de datos y su catálogo también es accesible a través del esquema de información tal como se define en el estándar SQL.

Las características de integridad de datos incluyen claves primarias, claves foráneas con restricción y actualizaciones/eliminaciones en cascada, restricciones de chequeo (check constraints), restricciones de únicos (unique constraints) y restricciones de no nulos (not null constraints).

También, cuenta con una gran cantidad de extensiones y características avanzadas. Entre las conveniencias están las columnas de incremento automático a través de secuencias, y LIMIT / OFFSET permitiendo el retorno de conjuntos de resultados parciales. PostgreSQL soporta índices compuestos, únicos, parciales y funcionales los cuales pueden utilizar cualquiera de sus métodos de almacenamiento B-tree, R-tree, hash, o GiST.

La indexación GIST (Generalized Search Tree) es un sistema avanzado que reúne a una amplia gama de diferentes algoritmos de ordenación y búsqueda, incluido el B-tree, B+-tree, R-tree, los árboles de suma parcial, B+-trees clasificados y muchos otros. También proporciona una interfaz que permite tanto la creación de tipos de datos personalizados, así como métodos de consulta extensibles para buscarlos. Por lo tanto, GIST ofrece la flexibilidad para especificar lo que le permite almacenar, cómo almacenarlo, y la posibilidad de definir nuevas formas de buscar a través de él -formas que superan con creces los ofrecidos por la norma B-tree, R-tree y otros algoritmos de búsqueda generalizados.

GIST sirve como base para muchos proyectos públicos que utilizan PostgreSQL como OpenFTS y PostGIS. OpenFTS (motor de búsqueda de texto completo Open Source) ofrece la indexación en línea de los datos y la clasificación de relevancia para la búsqueda de base de datos. PostGIS es un proyecto que añade soporte para objetos geográficos en PostgreSQL, lo que le permite ser utilizado como una base de datos espacial para los sistemas de información geográfica (SIG), muy parecido a la SDE de ESRI o la extensión espacial de Oracle.

Además, PostgreSQL ejecuta procedimientos almacenados en más de una docena de lenguajes de programación, incluyendo Java, Perl, Python, Ruby, Tcl, C/C++, y su propio PL/pgSQL, el cual es muy similar al PL/SQL de Oracle. Incluido en su biblioteca de funciones estándares están cientos de funciones integradas que van desde las matemáticas básicas y operaciones con cadenas a la criptografía y la compatibilidad de Oracle. Los disparadores y procedimientos almacenados pueden ser escritos en C y se cargan en la base de datos como una biblioteca, lo que permite una gran flexibilidad en la ampliación de sus capacidades. Del mismo modo, PostgreSQL incluye un marco de trabajo que permite a los desarrolladores definir y crear sus propios tipos de datos personalizados junto con funciones de apoyo y operadores que definen su comportamiento. Como resultado, una gran cantidad de tipos de datos avanzados se han creado que van desde primitivas geométricas y espaciales hasta direcciones de red e incluso tipos de datos ISBN / ISSN (Número de libro estándar internacional / Número de serie estándar internacional), todos estos se pueden añadir opcionalmente al sistema.

En el caso del presente trabajo, de acuerdo al estudio realizado se utilizaron procedimientos almacenados escritos en PL/pgSQL y también en SQL, dichos procedimientos almacenados utilizan algunas funciones integradas de PostgreSQL. Como resultado, se obtiene un mayor rendimiento en las consultas de base de datos ya que dichas consultas se almacenan en la memoria del servidor, mas no se tiene que enviar la consulta una y otra vez desde el cliente.

Así como hay muchos lenguajes de procedimiento soportados por PostgreSQL, también hay muchas interfaces de la biblioteca como tal, permitiendo que varios lenguajes sean compilados e interpretados a la vez para interactuar con PostgreSQL. Hay interfaces para Java (JDBC), ODBC, Perl, Python, Ruby, C, Esquema C + +, PHP, Lisp, y Qt sólo para nombrar unos pocos.

Lo mejor de todo, el código fuente de PostgreSQL está disponible bajo una licencia de código abierto liberal: la Licencia PostgreSQL. Esta licencia otorga la libertad de usar, modificar y distribuir PostgreSQL en cualquier forma, abierto o cerrado el código. Todas las modificaciones, mejoras o cambios que se realice son propias del autor para hacer lo que le plazca. Como tal, PostgreSQL no es sólo un sistema de base de datos potente capaz de poner en marcha una empresa, es una plataforma de desarrollo en donde es posible desarrollar para la empresa, para la web o productos comerciales de software que requieren una capacidad RDBMS (Sistema de Gestión de Base de Datos Relacionales).

1.3.2. PostGIS

El motivo más relevante para la selección del sistema de gestión de base de datos fue la necesidad de disponer de un sistema que posea características que permitieran el análisis de datos geo-espaciales para que sea evidente la solución a los requerimientos técnicos y académicos del presente trabajo. Por esta razón se escogió al sistema de gestión de base de datos PostgreSQL más su extensión geo-espacial PostGIS. Si se tiene entendido que un GIS o SIG (Sistema de Información Geográfico) es una integración organizada de hardware, software y datos geográficos diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de

planificación y gestión, es fácil entender que PostGIS es una de las herramientas que permiten realizar lo anteriormente dicho, calculando las relaciones entre los objetos geográficos que son muy difíciles de modelar sin usar objetos espaciales. Algunas de las relaciones típicas entre objetos espaciales son: proximidad, adyacencia y contención, donde PostGIS conlleva funciones de base de datos que permiten el manejo de estas relaciones.

PostGIS adiciona soporte para objetos geográficos a la base de datos objeto-relacional PostgreSQL. De hecho PostGIS “habilita espacialmente” al servidor PostgreSQL, para que pueda ser utilizada como una base de datos espacial de “back-end” para sistemas de información geográficos (SIG). PostGIS sigue las especificaciones del OGC (OpenGIS Consortium). Con PostGIS se pueden usar todos los objetos que aparecen en la especificación como puntos, líneas, polígonos, multi-líneas, multipuntos, y colecciones geométricas.

- Objetos GIS

Los objetos GIS son parte fundamental para el manejo de datos y consultas en una base de datos espacial; los objetos GIS soportados por PostGIS son un superconjunto de las “figuras simples” definidos por el Consorcio OpenGIS (OGC).

La especificación OpenGIS define dos formas estándar de expresión de los objetos espaciales: la forma Well-Known Text (WKT) y la forma Well-Known Binary (WKB). Tanto WKT y WKB incluyen información sobre el tipo de objeto y las coordenadas que forman el objeto.

Ejemplos de las representaciones de texto (WKT) de los objetos espaciales de las figuras son las siguientes:

```
- POINT(0 0)
- LINESTRING(0 0,1 1,1 2)
- POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
- MULTIPOINT(0 0,1 2)
- MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
- MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)),((-1
-1,-1 -2,-2 -2,-2 -1,-1 -1)))
- GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))
```

La especificación OpenGIS también requiere que el formato interno de almacenamiento de objetos espaciales incluyan un sistema de referencia espacial identificador (SRID). El SRID es necesario al crear objetos espaciales para la inserción en la base de datos.

Las entradas/salidas de estos formatos están disponibles usando las siguientes interfaces:

```
bytea WKB = ST_AsBinary(geometry);  
text WKT = ST_AsText(geometry);  
geometry = ST_GeomFromWKB(bytea WKB, SRID);  
geometry = ST_GeometryFromText(text WKT, SRID);
```

Por ejemplo, una instrucción de inserción válida para crear e insertar un objeto espacial OGC sería:

```
INSERT INTO geotable ( the_geom, the_name )  
VALUES ( ST_GeomFromText('POINT(-126.4 45.32)', 312), 'A Place');
```

- Creación de una tabla espacial

Crear una tabla con datos espaciales, se puede hacer en un solo paso. Como se muestra en el siguiente ejemplo que crea una tabla de carreteras con una columna geométrica de cadenas de líneas 2D en latitud y longitud WGS84.

```
CREATE TABLE ROADS ( ID int4  
, ROAD_NAME varchar(25), geom geometry(LINESTRING,4326) );
```

Se puede añadir columnas adicionales utilizando el comando ALTER TABLE estándar como se hace en el siguiente ejemplo en donde se añade una cadena de líneas 3D.

```
ALTER TABLE roads ADD COLUMN geom2 geometry(LINESTRINGZ,4326);
```

Por razones de compatibilidad hacia atrás, se puede crear una tabla espacial en dos etapas utilizando las funciones de gestión.

- Crear una tabla no espacial normal.

Por ejemplo:

```
CREATE TABLE ROADS ( ID int4, ROAD_NAME varchar(25) )
```

- Añadir una columna espacial a la table utilizando la función “AddGeometryColumn” de OpenGIS.

La sintaxis es:

```
AddGeometryColumn (
    <schema_name>,
    <table_name>,
    <column_name>,
    <srid>,
    <type>,
    <dimension>
)
```

Ó, utilizando el esquema actual:

```
AddGeometryColumn (
    <table_name>,
    <column_name>,
    <srid>,
    <type>,
    <dimension>
)
```

Ejemplo 1: `SELECT AddGeometryColumn('public', 'roads', 'geom', 423, 'LINESTRING', 2)`

Ejemplo 2: `SELECT AddGeometryColumn('roads', 'geom', 423, 'LINESTRING', 2)`

A continuación un ejemplo de SQL utilizado para crear una tabla y para añadir una columna espacial (asumiendo que un SRID de 128 ya existe):

```
CREATE TABLE parks (
    park_id INTEGER,
    park_name VARCHAR,
    park_date DATE,
    park_type VARCHAR
);
SELECT AddGeometryColumn('parks', 'park_geom', 128, 'MULTIPOLYGON', 2
);
```

Seguidamente, otro ejemplo utilizando el tipo genérico “geometry” y el valor SRID indefinido de 0:

```
CREATE TABLE roads (
    road_id INTEGER,
    road_name VARCHAR
);
SELECT AddGeometryColumn('roads', 'roads_geom', 0, 'GEOMETRY', 3 );
```

- Las relaciones y medidas espaciales

Para el presente trabajo, de todas las funciones espaciales que posee PostGIS, se hizo uso de una sola función; la cual se describe a continuación.

ST_Distance - Calcula la distancia mínima cartesiana bi-dimensional entre dos geometrías en unidades proyectadas.

Sinopsis

```
float ST_Distance(geometry g1, geometry g2);
float ST_Distance(geography gg1, geography gg2);
float ST_Distance(geography gg1, geography gg2, boolean use_spheroid);
```

Ejemplos

```
--Geometry example - units in planar degrees 4326 is WGS 84 long lat
unit=degrees
SELECT ST_Distance(
ST_GeomFromText('POINT(-72.1235 42.3521)',4326),
ST_GeomFromText('LINESTRING(-72.1260 42.45, -72.123 42.1546)', 4326)
);
st_distance
-----
0.00150567726382282

-- Geometry example - units in meters (SRID: 26986 Massachusetts state
plane meters) (most -
accurate for Massachusetts)
SELECT ST_Distance(
ST_Transform(ST_GeomFromText('POINT(-72.1235 42.3521)',4326),26986),
ST_Transform(ST_GeomFromText('LINESTRING(-72.1260 42.45, -72.123
42.1546)', 4326) -
,26986)
);
st_distance
-----
123.797937878454

-- Geometry example - units in meters (SRID: 2163 US National Atlas
Equal area) (least -
accurate)
SELECT ST_Distance(
ST_Transform(ST_GeomFromText('POINT(-72.1235 42.3521)',4326),2163),
ST_Transform(ST_GeomFromText('LINESTRING(-72.1260 42.45, -72.123
42.1546)', 4326) -
,2163)
);
st_distance
-----
126.664256056812
```

```

-- Geography example -- same but note units in meters - use sphere for
slightly faster less -
accurate
SELECT ST_Distance(gg1, gg2) As spheroid_dist, ST_Distance(gg1, gg2,
false) As sphere_dist
FROM (SELECT
ST_GeographyFromText('SRID=4326;POINT(-72.1235 42.3521)') As gg1,
ST_GeographyFromText('SRID=4326;LINESTRING(-72.1260 42.45, -72.123
42.1546)') As gg2
) As foo ;
spheroid_dist | sphere_dist
-----+-----
123.802076746848 | 123.475736916397

```

1.4. Plataforma de programación Java Enterprise Edition

En lo que se refiere a la plataforma de programación Java para la construcción del presente trabajo, el sistema web geo-referenciado se divide en dos partes: el módulo de comunicaciones, que se construyó con la plataforma de programación Java Standard Edition; y la otra parte, que son los módulos de seguridad, administración, mapas y reportes que se construyó con la plataforma de programación Java Enterprise Edition y con una arquitectura multicapa; considerando estas condiciones, es conveniente hacer referencia a la arquitectura.

Toda vez, que se planteó construir una aplicación multicapa se tuvo que elegir la mejor tecnología que satisfaga las necesidades de mejora en la operatividad y aseguramiento de la calidad. La plataforma Java EE (Java Enterprise Edition) proporciona el marco de trabajo necesario para la creación de aplicaciones web en Java, basadas en una arquitectura multicapa (Martín Sierra, 2011). Planteado en ese contexto se procedió a construir una aplicación web en este sentido, debido a que los requerimientos del cliente se prestan para utilizar procedimientos de última tecnología y que además, se ajusten perfectamente a las necesidades de operatividad y seguridad del cliente.

Existen varias características que posee Java Enterprise Edition, una de ellas es que: “Utilizando las librerías incluidas en esta plataforma, es posible construir robustas aplicaciones que pueden ser alojadas en una amplia gama de servidores y que son capaces de ser ejecutadas desde diferentes tipos de clientes” (Martín Sierra, 2011). Entonces, considerando las futuras necesidades que se presentarían para este tipo de

aplicaciones web, se ha planificado elaborar un sistema con estas prestaciones y servicios que a futuro pueden escalar.

1.4.1. Arquitectura de tres capas

Una aplicación web es un programa informático que puede dar servicio simultáneamente a múltiples usuarios que lo ejecutan a través de Internet. Este tipo de aplicaciones se basan en lo que se conoce como una arquitectura de tres capas, donde los diferentes actores y elementos implicados en la misma se encuentran distribuidos en tres bloques o capas (Martín Sierra, 2011), por lo que dadas las necesidades actuales respecto a las soluciones informáticas en Internet, se planificó la construcción de una aplicación web con la arquitectura mencionada.

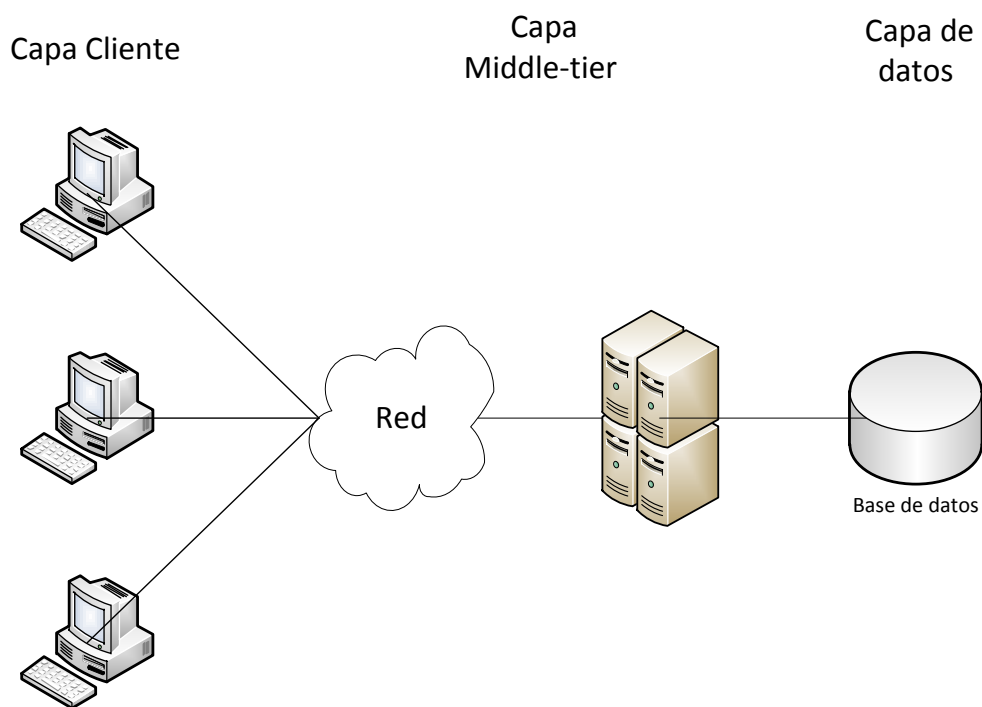


Ilustración 1.5: Arquitectura de tres capas. Fuente:(Martín Sierra, 2011).

Estas capas son:

- Capa cliente
- Capa intermedia
- Capa de datos

- Capa cliente

Se trata de la capa con la que interactúa el usuario de la aplicación, normalmente a través de un navegador web. Realiza principalmente dos funciones: por un lado se encarga de capturar los datos de usuario con los que opera la capa intermedia y enviárselos a ésta, y por otro presentar al usuario los resultados generados por la aplicación. Las páginas web, construidas mediante XHTML/CSS, son las encargadas de implementar esta funcionalidad, ayudándose como se ha visto de código JavaScript/AJAX para mejorar la experiencia del usuario con la aplicación (Martín Sierra, 2011)¹¹.

- Capa intermedia

Esta capa está constituida por la aplicación en sí, se encuentra la lógica de negocio del sistema, es donde se encuentra el cerebro del sistema; en esta capa se encuentra el servidor web y contenedor de servlets que aloja el código fuente de la aplicación, también el módulo de comunicaciones, ambos desarrollados en la plataforma Java.

La aplicación de la capa intermedia es ejecutada por un motor de aplicación especial capaz de permitir que una misma instancia de ella pueda dar servicio a múltiples clientes. Además de este motor, los servidores necesitan otro software, conocido como servidor Web, que sirva de interfaz entre la aplicación y el cliente, realizando el diálogo HTTP con éste. De forma resumida se podría decir que las funciones de la capa intermedia consisten en (Martín Sierra, 2011).

- Recoger los datos enviados desde la capa cliente.
- Procesar la información y, en general, implementar la lógica de la aplicación, incluyendo el acceso a los datos.
- Generar las respuestas para el cliente.

Es precisamente en esta capa intermedia donde el contenedor de servlets y servidor Web actúa como intermediario entre el cliente y la aplicación, dando lugar a que la plataforma de desarrollo JEE gestione la información con todas sus funcionalidades.

¹¹Martín Sierra, A. J. (2011). AJAX en JAVA EE. Madrid: RA-MA Editorial.

- Capa de datos

La capa de datos tiene como misión el almacenamiento permanente de la información manejada por la aplicación y la gestión de la seguridad de los mismos (Martín Sierra, 2011).

La capa de datos del presente trabajo consiste en la base datos relacional que funciona bajo el RDBMS PostgreSQL con su extensión para procesamiento de datos espaciales PostGIS; donde se almacena de forma permanente los datos necesarios para el funcionamiento adecuado del sistema web geo-referenciado.

Para la interoperabilidad de esta capa con la intermedia se utilizó la API de Persistencia de Java (Java Persistence API), la cual permite el manejo de datos relacionales en aplicaciones que usan la plataforma JEE aprovechando las ventajas de la orientación a objetos al interactuar con una base de datos relacional (siguiendo el patrón de mapeo objeto-relacional).

1.4.2. Arquitectura Modelo Vista Controlador

La capa intermedia se desarrolló en función de una arquitectura Modelo Vista Controlador (MVC), ya que es necesario un modelo que permita estructurar esta capa en una serie de bloques o componentes, de modo que cada uno de éstos tengan funciones definidas dentro de la aplicación y puedan desarrollarse de manera independiente. Además, la arquitectura MVC se ajusta al desarrollo de una aplicación con la plataforma Java Enterprise Edition; arquitectura que proporciona una clara separación entre las distintas responsabilidades de los componentes de la aplicación.

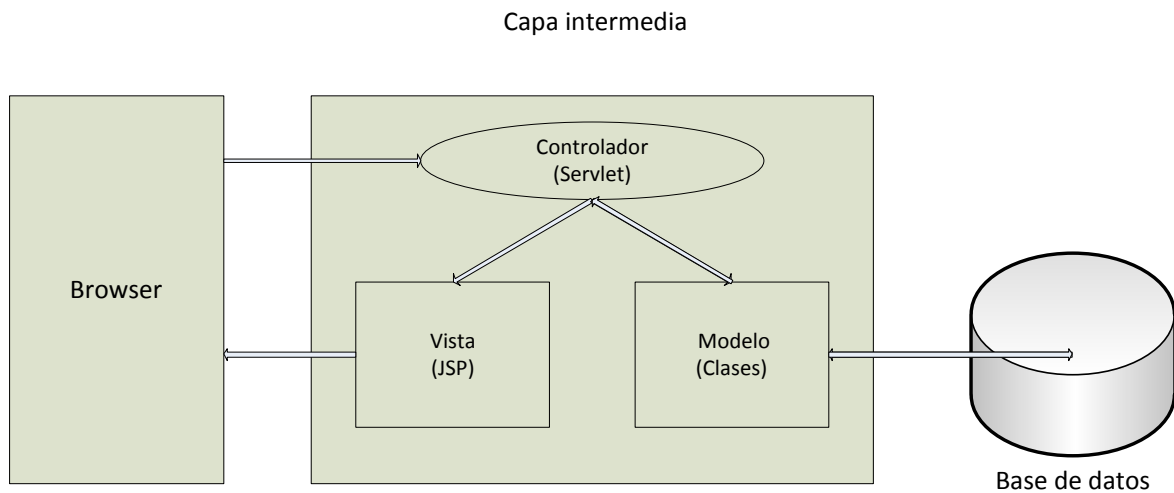


Ilustración 1.6: Esquema de una aplicación MVC. Fuente: (Martín Sierra, 2011).

Según la arquitectura MVC, la capa intermedia de una aplicación web puede dividirse en tres grandes bloques funcionales:

- Controlador
- Vista
- Modelo

- El Controlador

El controlador de la capa intermedia se desarrolló creando varios “servlets” que hacen la función de controlador, los cuales hacen la petición al Modelo para obtener los datos necesarios para presentárselos a la Vista, o viceversa.

Se puede decir que el controlador es el “cerebro” de la aplicación. Todas las peticiones a la capa intermedia que se realicen desde el cliente son dirigidas al controlador, cuya misión es determinar las acciones a realizar para cada una e invocar el resto de los componentes de la aplicación (Modelo y Vista) para que realicen las acciones requeridas en cada caso, encargándose también de todo el proceso. Así, en el caso de que una petición requiera enviar como respuesta al cliente determinada información existente en una base de datos, el controlador solicitará los datos necesarios al modelo y, una vez recibidos, se los proporcionará a la vista para que ésta les aplique el formato de presentación correspondiente y le envíe la respuesta al cliente (Martín Sierra, 2011).

- La Vista

La Vista de la aplicación está basada en páginas JSP las cuales obtienen la información de la capa de datos en su mayoría a través de AJAX asíncrono utilizando archivos JavaScript (más adelante se detallará la utilización de AJAX asíncrono y JavaScript) de igual forma que para el envío de información a la capa de datos. También se encuentra conformada de archivos CSS que se encargan del diseño gráfico de las páginas JSP.

Tal y como deduce (Martín Sierra, 2011), la Vista es la encargada de generar las respuestas que deben ser enviadas al cliente. Cuando esta respuesta tiene que incluir datos proporcionados por el controlador, el código XHTML de la página no será fijo, sino que deberá ser generado de forma dinámica, por lo que su implementación dependerá de una página JSP. Cuando la información que se va a enviar es estática, es decir, no depende de datos extraídos de un almacenamiento externo, podrá ser implementada por una página o documento XHTML.

Es precisamente en la Vista donde AJAX juega un papel importante. Tanto las páginas JSP como las XHTML pueden incluir código script del cliente que se comunique en segundo plano con el servidor para obtener datos de él.

En este caso, una vez que la Vista ha generado la respuesta y la ha enviado al cliente, las peticiones realizadas al servidor desde el código AJAX serán dirigidas al controlador, de este modo, cuando el servlet recibe la petición desde una aplicación AJAX cliente, recuperará los datos necesarios a través del modelo y, en vez de encaminar la petición a la Vista para que vuelva a generar la respuesta, lo que supondría una recarga de la página, aplicará a los datos el formato correspondiente (texto plano, XML, JSON) y los enviará directamente a la página cliente que hizo la petición (Martín Sierra, 2011).

- El Modelo

El Modelo se implementó creando varios “paquetes de clases Java” convirtiéndose en capas del modelo, estas son:

- **DAO** (Data Access Object): La capa de Objetos de Acceso a Datos, es aquella que contiene a los objetos que suministran una interfaz común entre la capa intermedia y la de datos.

- **DTO** (Data Transfer Object): La capa de Objetos de Transferencia de Datos, es aquella que contiene los objetos que transfieren datos entre la aplicación y la base de datos conjuntamente con los DAOs y el objeto JPA. Estos objetos no poseen comportamiento alguno, excepto por almacenar y devolver su propia información mediante accesores y mutadores (getters and setters). Cabe señalar que mediante el uso de JPA cada objeto DTO maneja su tabla de base de datos correspondiente.
- **JPA** (Java Persistence API): La capa de la API de Persistencia de Java posee el objeto que hace uso de la JPA y optimiza la creación de instancias “EntityManagerFactory”.
- o La API de persistencia de Java

Para la elaboración de la capa de persistencia de datos la cual permite una programación orientada a objetos haciendo que las tablas de base de datos y sus relaciones se manejen como tales y logrando que estos objetos persistan, es decir que se mantengan en memoria con todas sus propiedades y métodos en lugar de efectuar consultas SQL en todo momento, es necesario hacer uso de la API de persistencia de Java (Java Persistence API - JPA). Tal API (Interfaz de programación de aplicación) consiente en hacer que la escritura de código para el manejo de datos se convierta en una pieza natural de la arquitectura orientada a objetos.

Es imprescindible admitir que los datos son una parte integral de cualquier aplicación. Cada elemento se detiene al encontrarse con la capa de persistencia, donde los desarrolladores Java tradicionalmente han tenido que escribir consultas SQL complejas, las cuales pueden llegar a ser difíciles de manejar a medida que la aplicación crece. Por esta razón es necesario implementar tal API que permita tratar estas consultas como objetos, y aplicar los conceptos de la orientación a objetos como la encapsulación, abstracción, herencia y polimorfismo a estos.

De hecho, la comunidad Java ha producido numerosos enfoques orientados a objetos para la persistencia de datos: EJB, JDO, Hibernate, y Toplink son soluciones valiosas que han abordado este problema. La API de Persistencia de Java, o JPA, es una API de persistencia estándar presentada como parte de la plataforma Java EE 5. La especificación JPA fue primero presentada como parte de la JSR 220: EJB 3.0, con el

objetivo de simplificar el modelo de programación de los “EJB entity beans”. Aunque todo comenzó con los “entity beans” y fue empaquetada con Java EE 5.0, JPA se puede utilizar fuera del contenedor en un entorno Java SE (Das, 2008)¹².

▪ *Justificación del uso de JPA*

Una pregunta fundamental que se hacen los desarrolladores Java es: ¿Por qué utilizar JPA? ¿Por qué necesito saber cómo utilizar esta API cuando herramientas de mapeo objeto-relacional como Hibernate y Toplink se encuentran ya disponibles? La respuesta es que JPA no es una nueva tecnología; al contrario, ha recolectado las mejores ideas de tecnologías de persistencia existentes como Hibernate, Toplink y JDO. El resultado es una especificación estandarizada que ayuda a construir una capa de persistencia que es independiente de cualquier proveedor en particular (Das, 2008).

El presente trabajo hace uso de la implementación de JPA de código abierto OpenJPA, una implementación desarrollada por Apache Software Foundation. Se ha elegido esta implementación por ser una implementación diferente a las convencionales como Hibernate o EclipseLink, también porque usa del contenedor de servlets Apache Tomcat el cual se acopla idealmente a OpenJPA, ya que ambos pertenecen a Apache. Se usa la versión 1.0.1 por ser la versión más estable publicada de este software.

▪ *Uso de JPA*

En esta sección se hace referencia al artículo publicado por (Das, 2008), ya que se explica mediante un ejemplo el uso de JPA para el entorno al cual se enfoca este trabajo.

Se necesitan tres elementos para implementar un programa funcional JPA:

- Una clase entidad
- Un archivo persistence.xml
- Una clase a través de la cual se insertará, actualizará o encontrará una entidad.

¹² Das, A. (17 de Enero de 2008). *JAVAWORLD*. Recuperado el 24 de Diciembre de 2012, de <http://www.javaworld.com/javaworld/jw-01-2008/jw-01-jpa1.html>.

JPA trata todo acerca de persistencia de datos, por lo tanto es necesario explicar su funcionamiento con un diseño de almacenamiento de datos. Se asume que se tiene una tabla CLIENTE, como se muestra en la Tabla 1.2.

Tabla 1.2: Esquema de la tabla CLIENTE. Fuente: (Das, 2008).

NOMBRE	PK?	TIPO	NULL?
CLI_ID	Y	INTEGER	NOT NULL
NOMBRE		VARCHAR(50)	NOT NULL
APELLIDO		VARCHAR(50)	
DIRECCION		VARCHAR(50)	
PROFESION		VARCHAR(20)	NOT NULL
CIUDAD		VARCHAR(25)	
COD_POSTAL		VARCHAR(10)	NOT NULL
CLI_TIPO		VARCHAR(10)	NOT NULL
ULTIMA_FECHA_UPDATED		TIMESTAMP	NOT NULL

➤ *El objeto de persistencia: La Entidad*

En vista de que JPA trata todo acerca de mapeo entidad-relación, es necesario mirar el diseño del objeto entidad `Cliente` dado como ejemplo. El objeto entidad no es más que una clase POJO¹³ como una entidad marcada con la anotación `@Entity`, tal y como se muestra en el siguiente listado:

```
import javax.persistence.*;
import java.io.Serializable;
import java.util.Date;

@Entity(name = "CLIENTE") //Nombre de la entidad
public class Cliente implements Serializable{
    private long cliId;
    private String nombre;
    private String apellido;
```

¹³ Plane Old Java Object: Objeto Java Simple Plano

```

private String direccion;
private String profesion;
private String ciudad;
private String codPostal;
private String cliTipo;
private Date updatedTime;

// Aquí van los accesores y mutadores
...
}

```

Listado 1.1: La entidad Cliente. Fuente: (Das, 2008).

La entidad Cliente necesita saber cómo mapear los atributos (o propiedades) a la tabla CLIENTE. Se puede hacer esto ya sea a través de un archivo de configuración orm.xml (similar al archivo .hbm en Hibernate) o, como se muestra en el `import javax.persistence.*;`

```

import java.io.Serializable;
import java.util.Date;

@Entity(name = "CLIENTE") //Nombre de la entidad
public class Cliente implements Serializable {

    @Id //significa la clave primaria
    @Column(name = "CLI_ID", nullable = false)
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long cliId;

    @Column(name = "NOMBRE", nullable = false, length = 50)
    private String nombre;

    @Column(name = "APELLIDO", length = 50)
    private String apellido;

    // Por defecto el nombre de columna es el mismo que el nombre de
    // atributo
    private String direccion;

    @Column(name = "PROFESION", nullable = false)
    private String profesion;

    // Por defecto el nombre de columna es el mismo que el nombre de
    // atributo
    private String ciudad;

    @Column(name = "COD_POSTAL", nullable = false)
    // Nombre de la correspondiente columna de base de datos
    private String codPostal;

    @Column(name = "CLI_TIPO", length = 10)
    private String cliTipo;

    @Version

```



```

@Column(name = "ULTIMA_FECHA_UPDATED")
private Date updateTime;

// Getters and setters go here
.....
}

```

Listado 1.2, a través de anotaciones JPA.

```

import javax.persistence.*;
import java.io.Serializable;
import java.util.Date;

@Entity(name = "CLIENTE") //Nombre de la entidad
public class Cliente implements Serializable {

    @Id //significa la clave primaria
    @Column(name = "CLI_ID", nullable = false)
    @GeneratedValue(strategy = GenerationType.AUTO)
    private long cliId;

    @Column(name = "NOMBRE", nullable = false,length = 50)
    private String nombre;

    @Column(name = "APELLIDO", length = 50)
    private String apellido;

    // Por defecto el nombre de columna es el mismo que el nombre de
    atributo
    private String direccion;

    @Column(name = "PROFESION",nullable = false)
    private String profesion;

    // Por defecto el nombre de columna es el mismo que el nombre de
    atributo
    private String ciudad;

    @Column(name = "COD_POSTAL",nullable = false)
    // Nombre de la correspondiente columna de base de datos
    private String codPostal;

    @Column(name = "CLI_TIPO", length = 10)
    private String cliTipo;

    @Version
    @Column(name = "ULTIMA_FECHA_UPDATED")
    private Date updateTime;

    // Getters and setters go here
    .....
}

```

Listado 1.2: La entidad Cliente con anotaciones. Fuente: (Das, 2008).

A continuación se hace una breve explicación de cada anotación y algo más:

- Las anotaciones se definen en `javax.persistence`, por lo que es necesario importar dicho paquete.
- `@Entity` significa que una clase particular es una clase de entidad. Si el nombre de la entidad es diferente del nombre de la tabla, a continuación, la anotación `@Table` se utiliza, de lo contrario, no se requiere.
- `@Column` proporciona el nombre de la columna en una tabla, si es diferente del nombre de atributo (de manera predeterminada, los dos nombres se supone que son el mismo).
- `@Id` representa la clave principal.
- `@Version` significa un campo de versión en una entidad. JPA utiliza un campo de versión para detectar modificaciones simultáneas a un registro de almacenamiento de datos. Cuando el tiempo de ejecución de la JPA detecta varios intentos de modificar el mismo registro al mismo tiempo, se produce una excepción a la transacción al intentar confirmar el último registro. Esto le impide sobrescribir la confirmación anterior con datos obsoletos.

Por defecto, todos los campos son de tipo `@Basic`, que persisten tal y como se encuentran en la base de datos.

`@GeneratedValue` significa una estrategia para asignar un valor único a los campos de identificación automática. Los tipos de estrategias disponibles son `IDENTITY`, `SEQUENCE`, `TABLE` y `AUTO`. La estrategia por defecto es `AUTO`, cuya aplicación se deja implementar al proveedor JPA (OpenJPA se implementa a través de una secuencia de tabla).

Hay algunos puntos a tener en cuenta a la hora de crear una clase de entidad:

- JPA permite que clases persistentes hereden de clases no persistentes, clases persistentes hereden de otras clases persistentes y clases no persistentes hereden de clases persistentes.
- La clase de entidad debe tener un constructor sin argumentos por defecto.
- La clase de entidad no debe ser final.

- Clases persistentes no puede heredar de ciertas clases de sistema implementadas de forma nativa como `java.net.Socket` y `java.lang.Thread`.
- Si una clase persistente hereda de una clase no persistente, los campos de la superclase no persistente no puede ser persistente.

➤ *La unidad de persistencia*

Estando la clase entidad ya completa, se puede proseguir con el archivo `persistence.xml`, el cual se muestra en el Listado 1.3. Este es un archivo XML ubicado en la carpeta `META-INF`; es usado para especificar el nombre del proveedor de persistencia, los nombres de las clases entidad, las propiedades como la URL de conexión de la base de datos, el manejador, el usuario, la contraseña y otros.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  version="1.0">
  <persistence-unit name="tesisjpa" transaction-
  type="RESOURCE_LOCAL">
    <provider>

    org.apache.openjpa.persistence.PersistenceProviderImpl
    </provider>
    <class>ec.edu.utn.gpr.model.dto.Urusuario</class>
    <class>ec.edu.utn.gpr.model.dto.Uravl</class>
    <class>ec.edu.utn.gpr.model.dto.Guardato</class>
    <properties>
      <property name="openjpa.ConnectionURL"
value="jdbc:postgresql:db_tesis_ras"/>
      <property name="openjpa.ConnectionDriverName"
value="org.postgresql.Driver"/>
      <property name="openjpa.ConnectionUserName"
value="postgres"/>
      <property name="openjpa.ConnectionPassword"
value="sasa"/>
      <property name="openjpa.Log" value="SQL=TRACE"/>
    </properties>
  </persistence-unit>
</persistence>
```

Listado 1.3: Archivo `persistence.xml`. Fuente propia.

Algunos puntos importantes a tener en cuenta sobre el Listado 1.3 y el archivo `persistence.xml`:

- El archivo persistence.xml puede tener varias unidades de persistencia. Cada unidad puede ser utilizada por diferentes proveedores JPA o se puede utilizar para persistir a diferentes bases de datos.
 - El nombre del proveedor de persistencia específico del vendedor se especifica en la etiqueta <provider>. El proveedor de persistencia para OpenJPA es org.apache.openjpa.persistence.PersistenceProviderImpl.
 - Los nombres de las clases de entidad se especifican en la etiqueta <clase>.
 - Las propiedades de conexión de base de datos se puede especificar en la etiqueta <properties>. Se debe tener en cuenta que el nombre de la propiedad serán diferentes para cada proveedor.
 - OpenJPA tiene su propia facilidad de logging por defecto, el nivel por defecto es INFO.
- Clase utilitaria de JPA

Cada vez que se utilice una arquitectura MVC conjuntamente con la utilización de JPA es necesario crear una clase utilitaria que optimice la creación de objetos de tipo JPA como su recolección (eliminación de objetos de la memoria). Por lo tanto, se hace uso de la clase nativa de Java, ThreadLocal, la cual se la utiliza para obtener el objeto actual asignado al hilo del proceso, por lo que permite reducir el número de objetos creados en la memoria del servidor, en el presente caso.

JPA interactúa con la capa de datos a nivel de sesión, a nivel de transacción y a nivel de entidad, por lo que la clase utilitaria se divide en estas tres partes tal y como se puede observar en el Listado 1.4.

```
package ec.edu.utn.gpr.model.jpa;

import java.io.Serializable;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class JPAUtil {
    private static EntityManagerFactory sf;
    @SuppressWarnings("rawtypes")
    private static final ThreadLocal tls = new ThreadLocal();
    @SuppressWarnings("rawtypes")
```

```

private static final ThreadLocal tltx = new ThreadLocal();

//Runtime configuration
protected static synchronized EntityManagerFactory
getSessionFactory(
    String name) {
    if (sf == null)
        sf = Persistence.createEntityManagerFactory(name);
    return sf;
}

//Session Management
@SuppressWarnings("unchecked")
public static void openSession() {
    EntityManager s = (EntityManager) tls.get();
    if (s == null) {
        s =
getSessionFactory("tesisjpa").createEntityManager();
        tls.set(s);
    }
}

public static EntityManager getCurrentSession() {
    return (EntityManager) tls.get();
}

@SuppressWarnings("unchecked")
public static void closeSession() {
    EntityManager s = (EntityManager) tls.get();
    tls.set(null);
    if (s != null && s.isOpen())
        s.close();
}

//Transaction Management
@SuppressWarnings("unchecked")
public static void beginTransaction() {
    EntityTransaction tx = (EntityTransaction) tltx.get();
    if (tx == null) {
        tx = getCurrentSession().getTransaction();
        tx.begin();
        tltx.set(tx);
    }
}

@SuppressWarnings("unchecked")
public static void commitTransaction() {
    EntityTransaction tx = (EntityTransaction) tltx.get();
    if (tx != null && tx.isActive())
        tx.commit();
    tltx.set(null);
}

```

```

@SuppressWarnings("unchecked")
publicstaticvoid rollbackTransaction() {
    EntityTransaction tx = (EntityTransaction) tltx.get();
    tltx.set(null);
    if (tx != null&& tx.isActive())
        tx.rollback();
}

//Entity Management
publicstaticvoid create(Serializable obj) {
    getCurrentSession().persist((Object) obj);
}

@SuppressWarnings("unchecked")
publicstatic Object retrieve(@SuppressWarnings("rawtypes") Class
clz, Serializable key) {
    returngetCurrentSession().find(clz, (Object) key);
}

publicstatic Object update(Serializable obj) {
    returngetCurrentSession().merge((Object) obj);
}

publicstaticvoid delete(Serializable obj) {
    getCurrentSession().remove((Object) obj);
}
}

```

Listado 1.4: Clase utilitaria para JPA. Fuente propia.

Primeramente, se crea tres instancias de objetos, la primera de tipo `EntityManagerFactory`, la cual permite manejar de manera óptima los objetos entidades; la segunda y la tercera de tipo `ThreadLocal`, la cual permite obtener el hilo de proceso actual al cual está referenciado un objeto en este caso objetos de tipo `EntityManager` y `EntityTransaction`. Seguidamente el método `getSessionFactory()` el que se encarga de la configuración en tiempo de ejecución, el que se encarga de construir por única y primera vez al objeto `EntityManager`; a continuación la sección del manejo a nivel de sesión cuya finalidad es abrir, cerrar y obtener la sesión actual, es decir el objeto `EntityManager`; luego se encuentra la sección del manejo de la transacción, esta contiene los métodos encargados de iniciar, confirmar y cancelar la transacción; y finalmente, se encuentra la sección del manejo de la entidad cuya finalidad es crear, obtener, actualizar y eliminar el objeto entidad en cuestión, es decir el objeto que contiene la información correspondiente a una fila de una tabla de la base de datos.

En la arquitectura MVC, la lógica de negocio de la aplicación, incluyendo el acceso a los datos y su manipulación, está encapsulada dentro del modelo. En una aplicación JAVA EE, el modelo puede ser implementado mediante clases estándar Java o a través de Enterprise Java Beans (Martín Sierra, 2011).

1.5. Plataforma de programación Java Standard Edition

La otra parte de la aplicación que consiste en el módulo de comunicaciones se desarrolló con la plataforma de programación Java Standard Edition, la cual implementa una aplicación para servidor. Es una plataforma que ofrece todas las funcionalidades necesarias para desarrollar una aplicación que permita la interacción adecuada del servidor con el AVL. En la Ilustración 1.7 se muestran en forma de diagrama conceptual las tecnologías componentes de Java.

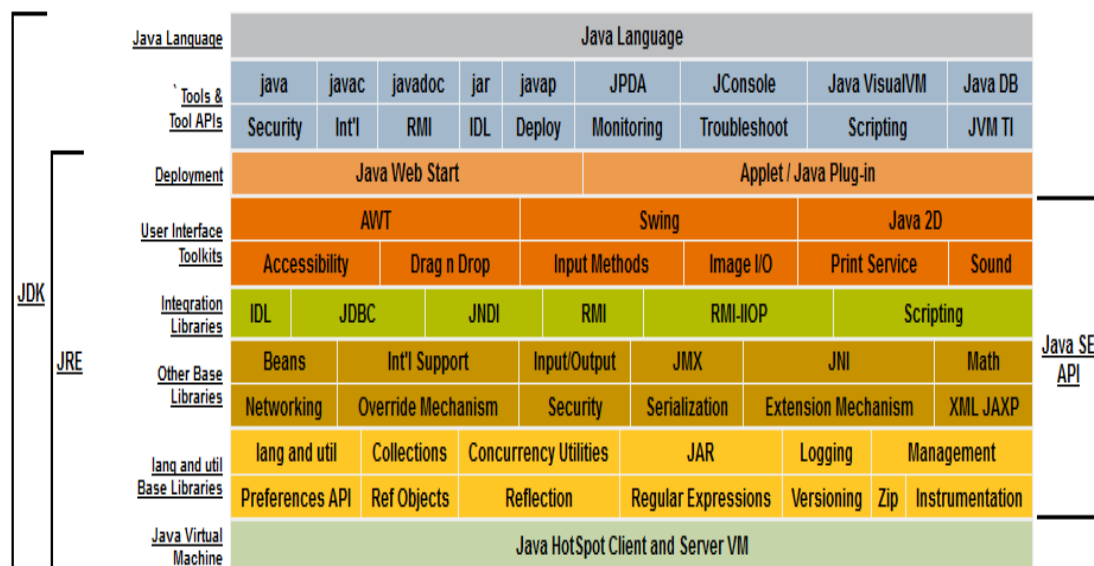


Ilustración 1.7: Tecnologías componentes de Java. Fuente: www.oracle.com.

Los paquetes o librerías de Java utilizados para el desarrollo de la aplicación del módulo de comunicaciones se detallan a continuación:

- `java.io`: Proporciona las entradas y salidas del sistema a través de flujos de datos, la serialización y el sistema de archivos.
 - o `java.io.BufferedReader`: `UnBufferedReader` añade funcionalidad a otro flujo de entrada, es decir, la capacidad para amortiguar la entrada y para apoyar la marca y restablecer métodos.

- `java.io.BufferedOutputStream`: La clase implementa un flujo de salida de buffer.
 - `java.io.BufferedReader`: Lee texto desde un flujo de caracteres de entrada, almacenando en un búfer, a fin de proporcionarlos para la lectura eficiente de caracteres, matrices y líneas.
 - `java.io.BufferedWriter`: Escribe texto en un flujo de caracteres de salida, almacenando en búfer, a fin de proporcionarlos para la escritura eficiente de caracteres individuales, matrices y cadenas.
 - `java.io.File`: Una representación abstracta de archivos y rutas de directorios.
 - `java.io.FileInputStream`: `UnFileInputStream` obtiene bytes de entrada desde un archivo en un sistema de archivos.
 - `java.io.FileOutputStream`: Un flujo de salida del archivo es una secuencia de salida para escribir datos en un archivo o un `FileDescriptor`.
 - `java.io.FileReader`: Clase conveniente para leer archivos de caracteres.
 - `java.io.IOException`: Señales de que una excepción de E/S de algún tipo se ha producido.
- **java.net**: Proporciona las clases para la implementación de aplicaciones de red.
 - `java.net.ServerSocket`: Esta clase implementa los sockets de servidor.
 - `java.net.Socket`: Esta clase implementa los sockets del cliente (también llamados simplemente "sockets").
 - **java.util**: Contiene el marco de las colecciones, el legado de clases heredadas, el modelo de evento, las facilidades de fecha y hora, la internacionalización, y la miscelánea de clases utilitarias (tokenizer de cadena, generador de números aleatorios, matriz de bits).
 - `java.util.ArrayList`: Implementación de una matriz que puede cambiar su tamaño de la interfaz `List`.
 - `java.util.Collection`: La interfaz raíz en la jerarquía de colecciones.
 - `java.util.List`: Una colección ordenada (también conocido como una secuencia).

- `java.sql`: Proporciona la API para acceder y procesar los datos almacenados en un origen de datos (por lo general una base de datos relacional) usando el lenguaje de programación Java™.
 - o `java.sql.Connection`: Una conexión (sesión) con una base de datos específica.
 - o `java.sql.DriverManager`: El servicio básico para la gestión de un conjunto de drivers JDBC. Debe tenerse en cuenta que, la interfaz `DataSource`, nueva en la API de JDBC 2.0, proporciona otra manera de conectarse a un origen de datos.
 - o `java.sql.ResultSet`: Una tabla de datos que representan un conjunto de resultados de base de datos, que normalmente se genera mediante la ejecución de una instrucción que consulta la base de datos.
 - o `java.sql.ResultSetMetaData`: Un objeto que puede ser utilizado para obtener información sobre los tipos y las propiedades de las columnas de un objeto `ResultSet`.
 - o `java.sql.SQLException`: Una excepción que proporciona información sobre un error de base de datos de acceso u otros errores.
 - o `java.sql.Statement`: El objeto utilizado para la ejecución de una sentencia SQL estática y la devolución de los resultados que produce.

En el Anexo I se puede encontrar el diagrama de clases UML del módulo de comunicaciones desarrollado para el presente trabajo.

1.5.1. Programación de sockets

Para que sea posible la interacción del servidor con el AVL y así rastrear el vehículo en cuestión, es necesario realizar una programación de sockets en el servidor como tal. Así, el AVL tendrá la capacidad de poder comunicarse con el servidor donde se encuentra alojada la aplicación eb del presente trabajo; por lo que es necesario implementar una plataforma de comunicación cliente/servidor. A continuación se hace referencia a la programación de sockets explicada y publicada por: (Buyya, Selvi, & Chu, 2009).

- Comunicación cliente/servidor

A un nivel básico, los sistemas basados en red consisten en un servidor, en un cliente y en un medio de comunicación como se muestra en la Ilustración 1.8. Un computador ejecutando un programa que hace una petición de servicios es llamado máquina cliente. Un computador ejecutando un programa que oferta servicios requeridos por uno o más clientes es llamado máquina servidor. El medio de comunicación puede ser una red alámbrica o inalámbrica.

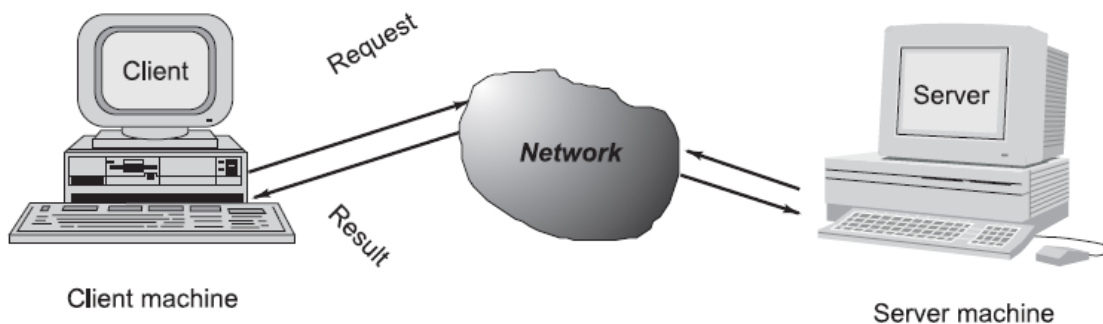


Ilustración 1.8: Comunicación Cliente/Servidor. Fuente (Buyya, Selvi, & Chu, 2009)¹⁴.

Generalmente, los programas que se ejecutan en máquinas cliente hacen peticiones a un programa (a menudo conocido como programa de servidor) que se ejecuta en una máquina servidor. Estos implican servicios de red proporcionados por la capa de transporte, que es parte de la pila de software de Internet, a menudo llamado TCP / IP (Transport Control Protocol / Internet Protocol) de pila, que se muestra en la Ilustración 1.9. La capa de transporte comprende dos tipos de protocolos, TCP (Transport Control Protocol) y UDP (User Datagram Protocol). Las interfaces de programación más utilizadas para estos protocolos son los conectores o sockets.

¹⁴ Buyya, R., Selvi, S. T., & Chu, X. (2009). *Object-Oriented Programming with Java*. Melbourne: McGraw Hill.

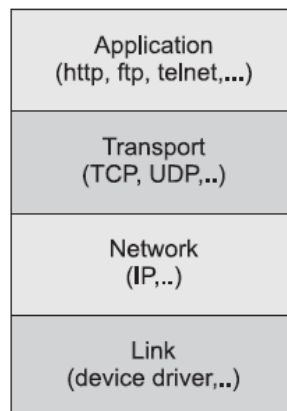


Ilustración 1.9: Pila de software TCP/IP. Fuente (Buyya, Selvi, & Chu, 2009).

TCP es un protocolo orientado a la conexión que proporciona un fiable flujo de datos entre dos computadores. Algunos ejemplos de aplicaciones que utilizan dichos servicios son HTTP, FTP y Telnet.

UDP es un protocolo que envía paquetes de datos independientes, llamados datagramas, de un computador a otro sin garantías sobre su llegada y su secuenciación. Algunos ejemplos de aplicaciones que utilizan estos servicios incluyen el servidor de reloj y Ping. Los protocolos TCP y UDP utilizan puertos para asignar los datos de entrada a un proceso en particular que se ejecuta en un computador. Un puerto está representado por un valor (16-bit) entero positivo. Algunos puertos han sido reservados para soportar servicios comunes y bien conocidos:

- ftp 21/tcp
- telnet 23/tcp
- smtp 25/tcp
- login 513/tcp
- http 80/tcp,udp
- https 443/tcp,udp

Los procesos/servicios a nivel de usuario por lo general utilizan un número de puerto ≥ 1024 .

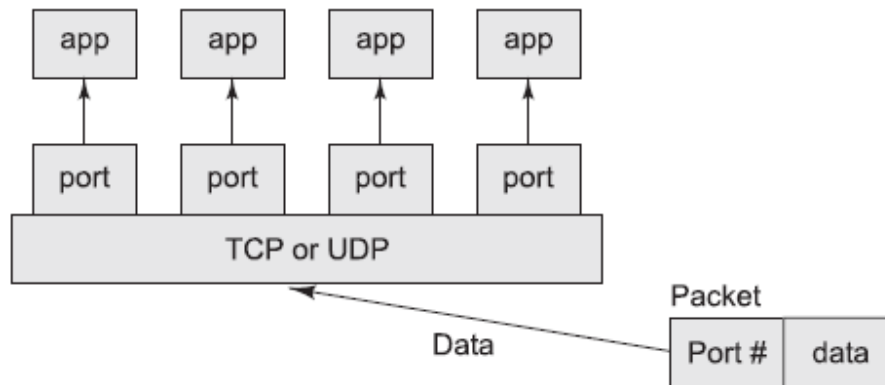


Ilustración 1.10: Asignación TCP/UDP de paquetes entrantes al puerto/proceso adecuado. Fuente (Buyya, Selvi, & Chu, 2009).

Las tecnologías Java orientadas a objetos tales como: Sockets (Conectores), Threads (Hilos), RMI, clustering, web services (servicios web); se han convertido en las principales soluciones para la creación de extensas y complejas aplicaciones de Internet portables, eficientes y sostenibles.

- Los sockets y comunicación basada en sockets

Los sockets proporcionan una interfaz para la programación de redes en la capa de transporte. La comunicación en red mediante sockets es muy similar a la realización de E/S de archivos. De hecho, el identificador de socket es tratado como identificador de archivo.

Los flujos (streams) utilizados en la operación de E/S de archivos también son aplicables a la E/S basada en socket. La comunicación basada en socket es independiente de un lenguaje de programación utilizado para su implementación. Esto significa que un programa de sockets escrito en lenguaje Java puede comunicarse con un programa de socket no escrito en Java (por ejemplo C o C++).

Un servidor (programa) se ejecuta en un equipo específico y tiene un socket que está enlazado a un puerto específico. El servidor escucha al socket por un cliente que haga una solicitud de conexión (véase la Ilustración 1.11a). Si todo va bien, el servidor acepta la conexión (véase la Ilustración 1.11b). Una vez aceptado, el servidor obtiene un nuevo socket enlazado a un puerto diferente. Se necesita un nuevo socket (por consiguiente, un número de puerto diferente) para que pueda continuar escuchando al

socket original por solicitudes de conexión mientras está sirviendo al cliente conectado.

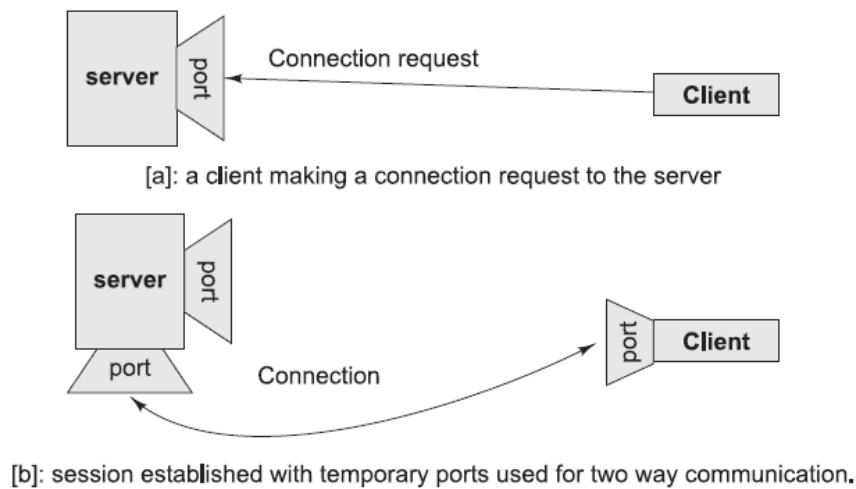


Ilustración 1.11: Establecimiento de ruta de acceso para la comunicación de dos vías entre un cliente y un servidor. Fuente (Buyya, Selvi, & Chu, 2009).

- Programación de sockets y la clase java.net

Un socket es un punto final de un enlace de comunicación bidireccional entre dos programas que se ejecutan en la red. Un socket está enlazado a un número de puerto de manera que la capa TCP puede identificar la aplicación en donde los datos están siendo enviados. Java proporciona un conjunto de clases, que se definen en un paquete denominado `java.net`, para permitir el rápido desarrollo de aplicaciones de red. Las principales clases, interfaces y excepciones en el paquete `java.net` simplifican la complejidad involucrada en la creación de programas cliente y servidor, estos son:

Clases

- `ContentHandler`
- `DatagramPacket`
- `DatagramSocket`
- `DatagramSocketImpl`
- `URLConnection`
- `InetAddress`
- `MulticastSocket`
- `ServerSocket`
- `Socket`
- `SocketImpl`
- `URL`

- URLConnection
- URLEncoder
- URLStreamHandler

Interfaces

- ContentHandlerFactory
- FileNameMap
- SocketImplFactory
- URLStreamHandlerFactory

Excepciones

- BindException
- ConnectException
- MalformedURLException
- NoRouteToHostException
- ProtocolException
- SocketException
- UnknownHostException
- UnknownServiceException

Ya que, de acuerdo al estudio realizado en el presente trabajo acerca de los protocolos TCP y UDP, y su fiabilidad de utilización y optimización de los recursos tecnológicos, se ha definido que se use el protocolo UDP por su bajo coste en lugar del protocolo TCP, aunque no sea tan fiable la transmisión de los datos basados en el protocolo UDP. Por lo tanto, se ha hecho el estudio de la programación de sockets basada en UDP.

- Programación de sockets basada en UDP

Como ya se ha dicho, TCP garantiza la entrega de paquetes y conserva su orden de destino. A veces, estas características no son necesarias y, ya que no vienen sin costos de rendimiento, sería mejor utilizar un protocolo de transporte ligero. Este tipo de servicio se realiza mediante el protocolo UDP que transporta paquetes de datagramas.

Los paquetes de datagramas se utilizan para implementar un servicio de entrega de paquetes sin conexión soportada por el protocolo UDP. Cada mensaje se transfiere desde la máquina de origen hacia el destino basándose en la información contenida dentro de ese paquete. Eso significa que, cada paquete tiene que tener dirección de destino y cada paquete podría ser direccionado de manera distinta, y podría llegar en cualquier orden. La entrega de paquetes no está garantizada. El formato de un paquete de datagramas es:

```
| Msg | length | Host | serverPort |
```

Java soporta la comunicación de datagramas mediante las siguientes clases:

- DatagramPacket
- DatagramSocket

La clase `DatagramPacket` contiene varios constructores que pueden ser usados para crear objetos de paquete. Uno de ellos es:

```
DatagramPacket(byte[] buf, int length, InetAddress address, int port);
```

Este constructor se utiliza para crear un paquete de datagrama para el envío de paquetes de longitud `length` al número de puerto específico en el host especificado. El mensaje a transmitir se indica en el primer argumento. Los principales métodos de la clase `DatagramPacket` son:

```
byte[] getData()
```

Devuelve el buffer de datos.

```
int getLength()
```

Devuelve la longitud de los datos a enviarse o la longitud de los datos recibidos.

```
void setData(byte[] buf)
```

Modifica el buffer de datos para este paquete.

```
void setLength(int length)
```

Modifica la longitud para este paquete.

La clase `DatagramSocket` admite varios métodos que pueden ser usados para transmitir o recibir un datagrama de datos por la red. Los dos métodos principales son:

```
void send(DatagramPacket p)
```

Envía un paquete de datagrama desde este socket.

```
void receive(DatagramPacket p)
```

Recibe un paquete de datagrama desde este socket.

Un programa simple de servidor UDP que espera peticiones del cliente y luego acepta el mensaje (datagrama) y envía de vuelta el mismo mensaje, está dado en el Listado

1.5 de abajo. Por supuesto, un programa de servidor extendido puede manipular los mensajes/solicitudes del cliente y enviar un nuevo mensaje como respuesta.

```
// UDPServer.java: A simple UDP server program.
import java.net.*;
import java.io.*;

publicclass UDPServer {
    publicstaticvoid main(String args[]) {
        DatagramSocket aSocket = null;
        if (args.length< 1) {
            System.out.println("Usage: java UDPServer <Port
Number>");
            System.exit(1);
        }
        try {
            int socket_no = Integer.valueOf(args[0]).intValue();
            aSocket = new DatagramSocket(socket_no);
            byte[] buffer = newbyte[1000];
            while (true) {
                DatagramPacket request =
newDatagramPacket(buffer,
                    buffer.length);
                aSocket.receive(request);
                DatagramPacket reply =
newDatagramPacket(request.getData(),
                    request.getLength(),
request.getAddress(),
                    request.getPort());
                aSocket.send(reply);
            }
        } catch (SocketException e) {
            System.out.println("Socket:" + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO: " + e.getMessage());
        } finally {
            if (aSocket != null)
                aSocket.close();
        }
    }
}
```

Listado 1.5: Programa de servidor UDP. Fuente (Buyya, Selvi, & Chu, 2009).

```
//UDPClient.java: A simple UDP client program.
import java.net.*;
import java.io.*;

publicclassUDPClient {
    publicstaticvoid main(String args[]) {
        // args give message contents and server hostname
        DatagramSocket aSocket = null;
```



```

        if (args.length < 3) {
            System.out
                .println("Usage: java UDPClient
<message><Host name><Port number>");
            System.exit(1);
        }
        try {
            aSocket = new DatagramSocket();
            byte[] m = args[0].getBytes();
            InetAddress aHost = InetAddress.getByName(args[1]);
            int serverPort = Integer.valueOf(args[2]).intValue();
            DatagramPacket request = new DatagramPacket(m,
args[0].length(),
                aHost, serverPort);
            aSocket.send(request);
            byte[] buffer = new byte[1000];
            DatagramPacket reply = new DatagramPacket(buffer,
buffer.length);
            aSocket.receive(reply);
            System.out.println("Reply: " + new
String(reply.getData()));
        } catch (SocketException e) {
            System.out.println("Socket: " + e.getMessage());
        } catch (IOException e) {
            System.out.println("IO: " + e.getMessage());
        } finally {
            if (aSocket != null)
                aSocket.close();
        }
    }
}

```

Listado 1.6: Programa de cliente UDP. Fuente (Buyya, Selvi, & Chu, 2009).

Un programa cliente correspondiente para la creación de un datagrama para luego enviarlo al programa de servidor del Listado 1.5 para luego aceptar una respuesta, se muestra en el Listado 1.6.

1.6. JavaScript

JavaScript más que un lenguaje de programación se ha convertido en una herramienta fundamental en el desarrollo de la interfaz de usuario de una aplicación web, gracias a que se puede manejar todos los componentes funcionales de un navegador web y manipular la experiencia del usuario de acuerdo a las necesidades del mismo y las capacidades del desarrollador del software. Por lo tanto, en esta sección se hace una

referencia bibliográfica a (Zakas, 2006)¹⁵, dando lugar a un breve resumen de este lenguaje de programación el cual se detalla a continuación.

Desde su aparición en 1995, el principal objetivo de JavaScript fue encargarse de la validación de entradas que, previamente, se había dejado en manos de lenguajes del lado del servidor como Perl. Antes se necesitaba un viaje de ida y vuelta al servidor para determinar si un campo obligatorio se había dejado en blanco o si el valor introducido era válido. Netscape Navigator intentó cambiarlo con la introducción de JavaScript. La posibilidad de realizar validaciones básicas en el cliente era muy atractiva en una época donde predominaban los módem telefónicos de 28.8 kbps. Esta reducida velocidad convertía a los viajes al servidor en un alarde de paciencia.

Desde entonces, JavaScript ha evolucionado hasta convertirse en una característica básica de los principales navegadores web del mercado. Además de sus vinculaciones con la validación de datos, ahora interactúa con todos los aspectos de la ventana del navegador y sus contenidos. Incluso Microsoft, con su propio lenguaje de creación de secuencias de comandos del lado del servidor (VBScript), ha acabado por incluir su propia implementación de JavaScript desde las primeras versiones de Internet Explorer.

Debido a la aparición de Microsoft en el mercado y debido a la rápida evolución de este nuevo lenguaje de secuencia de comandos, fue necesaria una estandarización para que el lenguaje fuera compatible con todos los navegadores web del mercado; por lo que surge ECMAScript como estándar para definir el nuevo lenguaje de secuencia de comandos.

Aunque ECMAScript es un estándar importante, no es la única parte de JavaScript y, ciertamente, tampoco la única parte que se ha estandarizado. De hecho, una implementación completa de JavaScript se compone de tres partes, como se indica en la ilustración 1.12:

- El núcleo (ECMAScript).

¹⁵ Zakas, N. C. (2006). Profesional JavaScript para desarrolladores Web. Madrid: Ediciones Anaya Multimedia.

- El Modelo de objetos de documento (DOM).
- El Modelo de objetos de navegador (BOM).

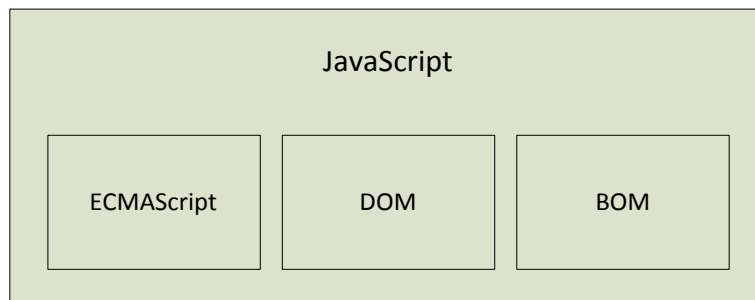


Ilustración 1.12: Componentes de una implementación JavaScript. Fuente (Zakas, 2006).

1.6.1. ECMAScript

ECMAScript no está vinculado a ningún navegador concreto y, en realidad, carece de métodos para entradas y salidas del usuario (no es como en C, que depende de bibliotecas externas para realizar estas tareas).

En lo que respecta a ECMAScript, un navegador web es un entorno anfitrión pero no el único. En realidad, muchos otros entornos (como ScriptEase de Nombas y ActionScript de Macromedia, utilizados en Flash y Director MX) pueden alojar implementaciones de ECMAScript. ECMAScript fuera del navegador especifica lo siguiente (Zakas, 2006):

- Sintaxis
- Tipos
- Instrucciones
- Palabras clave
- Palabras reservadas
- Operadores
- Objetos

ECMAScript es simplemente una descripción que define todas las propiedades, métodos y objetos de un lenguaje de secuencias de comandos. Otros lenguajes, como JavaScript (véase la Ilustración 1.13), implementan ECMAScript como base de sus funciones (Zakas, 2006).

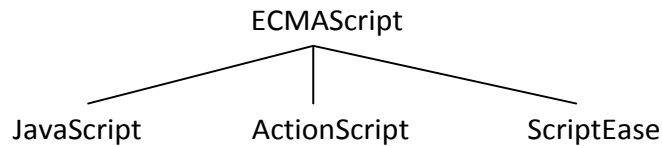


Ilustración 1.13: Lenguajes que implementan ECMAScript. Fuente (Zakas, 2006).

1.6.2. Ediciones de ECMAScript

ECMAScript se separa en ediciones y no en versiones ya que se define el estándar ECMA-262. Al igual que otros estándares, ECMA-262 se puede modificar y actualizar. Cuando se produce una actualización importante, se publica una nueva edición del estándar. La última edición de ECMA-262 es la 3, publicada en diciembre de 1999.

1.6.3. Compatibilidad de ECMAScript en navegadores web

En la siguiente tabla se muestra la compatibilidad de cada navegador existente en el mercado con la edición del estándar ECMAScript.

Tabla 1.3: Compatibilidad de ECMAScript en navegadores web. Fuente (Zakas, 2006).

Navegador	Compatibilidad con ECMAScript
Netscape Navigator 2.0	-
Netscape Navigator 3.0	-
Netscape Navigator 4.0-4.05	-
Netscape Navigator 4.06-4.079	Edition 1
Netscape 6.0 (Mozilla 0.6.0+)	Edition 3
Internet Explorer 3.0	-
Internet Explorer 4.0	-
Internet Explorer 5.0	Edition 1
Internet Explorer 5.5+	Edition 3
Opera 6.0-7.1	Edition 2
Opera 7.2+	Edition 3

1.6.4. El Modelo de Objetos de Documento (DOM)

El DOM es utilizado para modificar mediante código JavaScript en la capa cliente cualquier elemento de una página web sin tener que volver a cargarla, se lo puede definir de la siguiente forma: El Modelo de objetos de documento (DOM) es una interfaz de programación de aplicaciones (API) para HTML y XML. El DOM confecciona una página completa en forma de documento compuesto por una serie de nodos jerárquicos. Cada parte de una página HTML o XML parte de un nodo.

```
<html>
  <head>
    <title>Sample page</title>
  </head>
  <body>
    <p>Hello World!</p>
  </body>
</html>
```

La anterior es una página HTML (Zakas, 2006); este código puede representarse en una jerarquía de nodos con ayuda del DOM, como se muestra en la Ilustración 1.14.

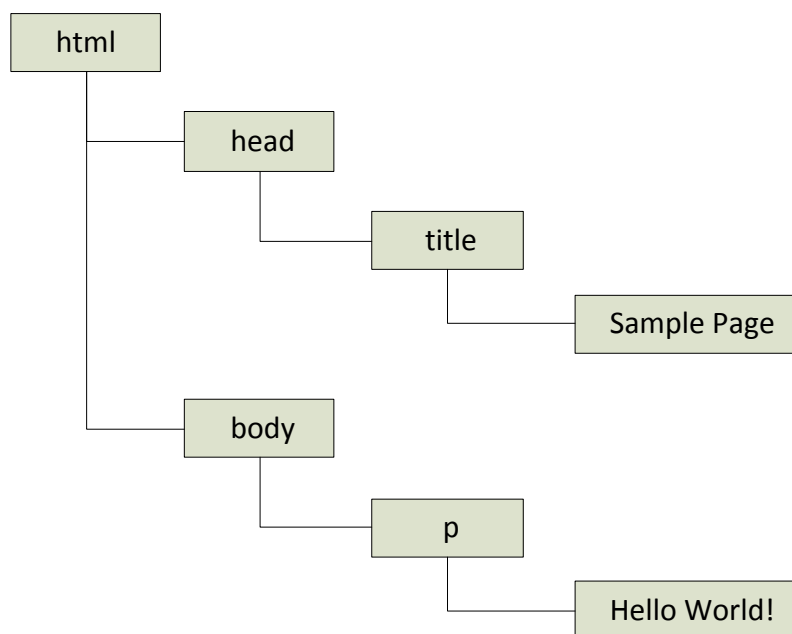


Ilustración 1.14: Jerarquía de nodos del DOM. Fuente (Zakas, 2006).

Al crear un árbol para representar un documento, el DOM permite a los programadores obtener control total sobre el contenido y la estructura. Con ayuda del API DOM resulta muy sencillo eliminar, añadir y sustituir nodos.

1.7. AJAX

En esta sección se detalla una tecnología que para el presente trabajo tiene una gran importancia en lo que refiere al conocimiento de herramientas para el desarrollo de aplicaciones web, ya que es una tecnología con la que se dio un paso gigante en la construcción de una experiencia de usuario inteligente y amigable. A continuación se explica a detalle esta tecnología haciendo una referencia bibliográfica casi completa acorde a (Martín Sierra, 2011).

1.7.1. ¿Qué es AJAX?

El término AJAX hace referencia a un mecanismo de combinación de tecnologías y estándares de cliente, consistente en la solicitud asíncrona de datos al servidor desde una página web y la utilización de éstos para actualizar una parte de la misma, sin obligar al navegador a realizar una recarga completa de toda la página.

En la actualidad, es la tecnología más demandada por los usuarios de aplicaciones web y la más utilizada por los desarrolladores web por el ahorro en el envío y recepción de datos al servidor, enviando y recibiendo únicamente los datos que hay que actualizar en lugar de enviar toda la página web completa y recargarla en su totalidad.

1.7.2. Las tecnologías AJAX

En síntesis, el proceso de funcionamiento de AJAX consiste en realizar peticiones HTTP de forma asíncrona al servidor desde el cliente, recibiendo los datos para actualizar determinadas partes de la página web.

Para poder realizar estas operaciones, las aplicaciones AJAX se apoyan en las siguientes tecnologías:

- **XHTML Y CSS:** La interfaz gráfica de una aplicación AJAX es una página web cargada en un navegador. XHTML y CSS son los datos estándares definidos por el W3C para

la construcción de páginas web, mientras que XHTML se basa en la utilización de un conjunto de marcas o etiquetas para la construcción de la página, el estándar CSS define una serie de propiedades de estilo que pueden aplicarse sobre las etiquetas XHTML, a fin de mejorar sus capacidades de presentación.

- **JavaScript:** Las peticiones HTTP que lanza la página web en modo asíncrono al servidor son realizadas por un bloque de script implementado con cualquier lenguaje capaz de ser interpretado por el navegador; este lenguaje es, en la gran mayoría de los casos, JavaScript.

Los bloques de código JavaScript que forman la aplicación AJAX se encuentran embebidos dentro de la propia página web, o bien en archivos independientes con extensión .js que son referenciados desde ésta. Estos bloques de código son traducidos y ejecutados por el navegador, bien cuando se produce la carga de la página en él o bien cuando tiene lugar alguna acción del usuario sobre la misma. Mediante código JavaScript las aplicaciones AJAX pueden realizar solicitudes al servidor desde la página web, recuperar los datos enviados en la respuesta y modificar el aspecto de la interfaz.

- **XML:** Aunque podría utilizarse cualquier otro formato basado en texto, cuando la aplicación del servidor tiene que enviar una serie de datos de forma estructurada al cliente, XML resulta la solución más práctica y sencilla, ya que es posible manipular fácilmente un documento de estas características y extraer sus datos desde el código JavaScript cliente.
- **DOM:** El Modelo de Objeto de Documento (DOM) proporciona un mecanismo estándar para acceder desde código a la información contenida en un documento de texto basado en etiquetas.

Mediante el DOM, tanto páginas web como documentos XML pueden ser tratados como un conjunto de objetos organizados de forma jerárquica, cuyas propiedades y métodos pueden ser utilizados desde código JavaScript para modificar el contenido de la página en un caso, o para leer los datos de la respuesta en otro.

- **El objeto XMLHttpRequest:** Se trata del componente fundamental de una aplicación AJAX. A través de sus propiedades y métodos es posible lanzar peticiones en modo asíncrono al servidor y acceder a la cadena de texto enviada en

la respuesta. Para poderlo utilizar, deberá ser previamente instanciado desde la aplicación.

Las anteriores tecnologías y componentes constituyen el núcleo fundamental de AJAX, sin embargo, estas aplicaciones se apoyan también para su funcionamiento en los siguientes estándares y tecnologías:

- **HTTP:** Al igual que el propio navegador, el objeto XMLHttpRequest utiliza el protocolo HTTP para realizar las solicitudes al servidor, manejando también las respuestas recibidas mediante este protocolo.
- **Tecnologías de servidor:** Una aplicación AJAX no tendrá sentido sin la existencia de un programa en el servidor que atendiera las peticiones enviadas desde la aplicación y devolviera resultados al cliente.

Las mismas tecnologías que se utilizan para crear procesos en el servidor en una aplicación web convencional, como Java EE, ASP.NET o PHP, pueden utilizarse igualmente en el caso de que la capa cliente esté basada en AJAX.

1.7.3. Aplicaciones AJAX multinavegador

Para la creación del objeto XMLHttpRequest que es el punto de partida para la construcción de una aplicación AJAX, no todos los navegadores que existen en el mercado actualmente, referencian a un solo tipo de objeto XMLHttpRequest para el funcionamiento de AJAX.

El objeto ActiveXObject es utilizado por Internet Explorer para crear instancias de componentes COM registrados en la máquina cliente, a partir de la cadena de registro de los mismos. En el caso de XMLHttpRequest, el componente COM que lo implementa es Microsoft.XMLHttp, aunque es muy posible que el cliente disponga además de versiones modernas de éste, como la MSXML2.XMLHttp, la MSXML2.XMLHttp.3.0 o incluso la MSXML2.XMLHttp.5.0.

El resto de los navegadores más comúnmente utilizados por los usuarios de Internet, como Opera, Firefox, Safari e incluso Internet Explorer a partir de la versión 7,

implementan XMLHttpRequest como un objeto nativo, lo que significa que su creación se debe llevar a cabo instanciando directamente la clase mediante el operador new:

```
Xhr = new XMLHttpRequest();
```

Ésta es la manera en que el W3C, organismo encargado de regular las especificaciones para la web, recomienda que debe crearse este objeto. Así pues, es de suponer que todas las futuras versiones de navegadores que se desarrollen en adelante soporten este mecanismo de creación.

Mientras sigan utilizándose navegadores Internet Explorer versión 6 y anteriores, las aplicaciones AJAX están obligadas a incluir cierta lógica que garantice la compatibilidad del código en todos los tipos de navegadores, lo que implica combinar en una función las dos formas analizadas de crear el objeto XMLHttpRequest.

Para ello, bastará con comprobar qué tipo de objeto nativo soporta el navegador donde se está ejecutando el código, ActiveXObject o XMLHttpRequest. Esto puede realizarse consultando las propiedades del mismo nombre del objeto window, ya que cuando un navegador soporta uno de estos objetos lo expone como una propiedad de window, tal como en el siguiente ejemplo:

```
function crearObjetoAjax() {
    if (window.ActiveXObject) {
        //navegador IE
        xhr = new ActiveXObject("Microsoft.XMLHttp")
    }
    else if ((window.XMLHttpRequest) ||
        (typeof XMLHttpRequest != undefined)) {
        //navegadores Firefox, Opera y Safari
        Xhr = new XMLHttpRequest();
    }
    else {
        //navegadores sin soporte AJAX
        alert("Su navegador no tiene soporte para AJAX");
        return;
    }
}
```

1.7.4. Fases en la ejecución de una aplicación AJAX

Es importante comprender el mecanismo de funcionamiento de una aplicación AJAX desde el momento que el usuario hace clic en un botón o despliega una lista o ingresa

texto por teclado, hasta la visualización de los datos procesados en la pantalla de la máquina cliente.

Durante este proceso de ejecución puede distinguirse las siguientes fases o etapas:

1. **Creación y configuración del objeto XMLHttpRequest:** El primer paso a realizar cuando se produce el evento que desencadena la ejecución de una aplicación AJAX consiste en obtener el objeto XMLHttpRequest.

Una vez creado el objeto, deben configurarse una serie de parámetros del mismo, como la URL del recurso a solicitar o la función que va a procesar la respuesta.

2. **Realización de la petición:** Tras configurar los parámetros adecuados del objeto XMLHttpRequest, se procede a lanzar la petición al servidor, operación ésta que puede realizarse en modo síncrono o asíncrono, siendo este último el modo de funcionamiento mayoritariamente utilizado por las aplicaciones AJAX.
3. **Procesamiento de la petición en el servidor:** El servidor recibe la petición y ejecuta el componente correspondiente que, a partir de los datos recibidos, deberá realizar algún tipo de procesamiento, incluyendo consultas a bases de datos, y generar una respuesta con los resultados obtenidos.

En el caso del presente trabajo, este componente del servidor es un servlet.

Cuando la petición se realiza en modo asíncrono, mientras el componente del servidor se está ejecutando para realizar su función, el usuario puede seguir interactuando con la página sin necesidad de mantenerse bloqueado a la espera de recibir la respuesta.

4. **Recepción de los datos de respuesta:** Una vez completada la ejecución del código del servidor, se envía una respuesta HTTP al cliente con los resultados obtenidos en el formato adecuado para su manipulación. En ese momento, el navegador invoca a la función de retrollamada definida por el objeto XMLHttpRequest.
5. **Manipulación de la página cliente:** A partir de los datos recibidos en la respuesta y mediante código JavaScript de cliente, se modifican las distintas zonas de la página XHTML que sea necesario actualizar.

1.8. Visor de mapas

Para lograr la visualización del mapa y la construcción de características dinámicas relacionadas con la posición del vehículo en el sistema web georreferenciado del presente trabajo se utilizó la librería JavaScript de uso libre: OpenLayers, la cual ha permitido desarrollar las características que hacen de este sistema una aplicación web para finalidades de uso geográfico. A continuación se hace un desglose de esta librería utilitaria haciendo una referencia bibliográfica a (Higuera, 2010)¹⁶.

OpenLayers es una biblioteca pura de JavaScript para la visualización de los datos del mapa en la mayoría de los navegadores web modernos, sin dependencias del lado del servidor. OpenLayers implementa una API de JavaScript para la construcción de aplicaciones web geográficas ricas, de forma similar a la de Google Maps y MSN Virtual Earth API, con una diferencia importante - OpenLayers es software libre, desarrollado por y para la comunidad de software Open Source.

OpenLayers permite incorporar mapas dinámicos en las páginas web. Los mapas se pueden dotar de diversos controles con capacidades de zoom, panning, medida de distancias y muchas otras herramientas.

OpenLayers proporciona herramientas para acceder a todo tipo de información geográfica proveniente de muy variadas fuentes, por ejemplo Web Map Services, Web Feature Services, Mapas comerciales, información genérica vectorial, y otros.

OpenLayers es un proyecto del Open Source Geospatial Foundation (OSGeo). Es una librería en puro Javascript, de uso totalmente libre bajo licencia BSD.

Para poder utilizar las clases proporcionadas por OpenLayers habrá que incorporar una referencia al script de la librería en la cabecera del documento HTML de esta aplicación:

```
<script src="http://www.openlayers.org/api/OpenLayers.js"></script>
```

¹⁶ Higuera, S. (7 de Junio de 2010). *Manual OpenLayers*. Obtenido de <http://openlayers.ingemoral.es/manualOpenLayers.html>.

A continuación se detallarán algunos de los componentes de OpenLayers utilizados en la construcción del visor de mapas para la culminación del presente trabajo.

1.8.1. Clases básicas

- Tipos básicos de datos

OpenLayers está construida sobre Javascript y por tanto dispone de los mismos tipos de datos. En Javascript hay cinco tipos de datos primitivos: *Undefined*, *Null*, *Boolean*, *Number* y *String*.

Además se dispone de una colección de objetos nativos: *Object*, *Boolean*, *Error*, *SyntaxError*, *Function*, *Number*, *EvalError*, *TypeError*, *Array*, *Date*, *RangeError*, *URIError*, *String*, *RegExp*, y *ReferenceError*.

OpenLayers ofrece clases con métodos útiles para operar con los tipos de datos básicos:

- **OpenLayers.String**

Esta clase proporciona los siguientes métodos:

- **startsWith(string, substr):** Comprueba si la cadena 'string' comienza con la cadena 'substr'. Devuelve un Boolean.
- **contains(str, substr):** Comprueba si la cadena 'substr' está contenida dentro de la cadena 'str'. Devuelve 'TRUE' en caso afirmativo y 'FALSE' en caso negativo.
- **trim(str):** Devuelve una cadena en la que se han eliminado todos los espacios que pudiera haber al principio o al final de la cadena 'str'.
- **camelize(str):** Camel-Case. Convierte una cadena a base de guines en el convenio Camelize. Por ejemplo la cadena 'lat-punto' la convierte en latPunto y la cadena -lat-punto en 'LatPunto'. Devuelve una cadena con las modificaciones, sin alterar el original.
- **isNumeric(str):** Devuelve un Boolean si la cadena corresponde a un número. Reconoce el formato exponencial para numeros reales. (p. ej *isNumeric("2.5e3")* devuelve 'TRUE').
- **numericIf(str):** Si la cadena 'str' es un valor numérico devuelve un Number con ese valor. En otro caso devuelve un String con la misma cadena recibida.

- **OpenLayers.Number**

La clase *OpenLayers.Number* tiene dos propiedades que se utilizan para dar formato a los números: 'decimalSeparator' y 'thousandsSeparator'.

Además contiene las siguientes funciones para manipular datos tipo Number:

- **limSigDigs(float, sig):** Redondea el valor del Float 'float' al número de decimales indicado por el Integer 'sig'. Devuelve un Float con el número redondeado.
- **format(num, dec, tsep, decsep):** Devuelve un String con el Float 'num' redondeado al número de decimales indicados por el Integer 'dec'. Como separadores de millares y decimales se utilizarán los String 'tsep' y 'decsep' respectivamente.

- **OpenLayers.Function**

- **OpenLayers.Array**

- **OpenLayers.Date**

Además hay algunas clases utilitarias:

- **OpenLayers.Class:** Se usa como clase base para todas las demás.
- **OpenLayers.Element:** Representa a un elemento DOM.

Hay otras clases de elementos simples:

- **OpenLayers.Pixel**

Representa una posición de pantalla. Tiene dos propiedades: 'x' e 'y'.

El constructor admite como parámetros los valores de las coordenadas x e y:

```
OpenLayers.Pixel( x: Number, y: Number );
```

La clase *OpenLayers.Pixel* tiene los siguientes métodos de utilidad:

- **toString():** Devuelve una cadena de la forma 'x=200.4,y=242.2', supuestas éstas como las coordenadas del objeto.
- **clone():** Devuelve un 'clon', una instancia de *OpenLayers.Pixel* con idénticas x e y que el original.
- **equals(px: OpenLayers.Pixel):** Devuelve un Boolean indicando si el Pixel pasado como argumento es equivalente al objeto propietario del método.

- **add(dx: Number, dy: Number):** Devuelve un `OpenLayers.Pixel` cuyas componentes son la suma de las originales más los incrementos 'dx' y 'dy'.
- **offset(px: OpenLayers.Pixel):** Devuelve un `OpenLayers.Pixel` cuyas coordenadas son la suma del original más las coordenadas del punto offset.

- **OpenLayers.LonLat**

Representa una posición geográfica identificada por sus propiedades longitud, 'lon', y latitud 'lat'. Las dos únicas propiedades específicas de la clase `LonLat` son 'lon' y 'lat' que son del tipo `Number` y que se corresponderán con las coordenadas de un punto expresadas en las unidades de la proyección del mapa. Así si el mapa está trabajando en una proyección con coordenadas geográficas, el par representará una longitud y una latitud expresadas en grados sexagesimales. En el caso de que el mapa esté en otro tipo de proyección, la pareja 'lon', 'lat' representará un punto en las coordenadas y unidades de la proyección del mapa. Si nuestro mapa utiliza la proyección `Spherical Mercator`, la pareja 'lon', 'lat' serán las coordenadas de un punto en dicha proyección y expresadas en metros.

Para crear un objeto `LonLat` se le pasan al constructor la pareja de datos 'lon' 'lat':

```
var pto = new OpenLayers.LonLat( -3.54, 42.37 );
```

Una vez creado un objeto de la clase `OpenLayers.LonLat` se puede acceder a sus propiedades individuales de forma sencilla:

```
var longitud = pto.lon;
var latitud = pto.lat;
```

La clase `LonLat` expone los siguientes métodos:

- **toString():** Devuelve un `String` formateado con las coordenadas del punto.
- **toShortString():** Devuelve un `String` formateado con las coordenadas del punto en formato compacto.
- **clone():** Devuelve un objeto `LonLat` con los mismos valores de 'lon' y 'lat'.
- **add(inclon, inclat):** Devuelve un objeto `LonLat` cuyas coordenadas son el resultado de sumar ordenadamente a las coordenadas del objeto original los valores pasados en los parámetros 'inclon' y 'inclat'.

- **equal(otherLonLat):** Este método devuelve un Boolean que indica si el objeto 'otherLonLat' pasado como parámetro tiene los mismos valores de las propiedades 'lon' y 'lat' que el objeto original.
- **transform(sourceProy, destProy):** En este método los dos parámetros que se pasan como argumentos son dos objetos OpenLayers.Projection. El método modifica las coordenadas del LonLat original mediante la aplicación del cambio de coordenadas entre la proyección 'sourceProy' y la proyección 'destProy'. El método devuelve una referencia al objeto original, con las coordenadas transformadas. (Este método modifica las coordenadas del objeto LonLat original).
- **fromString (str):** Esta función crea un objeto LonLat a partir de un String que tenga las coordenadas separadas por una coma. Por ejemplo: 'var ll = OpenLayers.LonLat.fromString("-3,43")'.

- **OpenLayers.Size**

Representa un tamaño en dos dimensiones. Tiene dos propiedades 'w', anchura y 'h', altura.

El constructor admite dos parámetros Number que se corresponden con los valores de la anchura 'w' y la altura 'h' del objeto 'Size' que se desea construir:

```
OpenLayers.Size(w: Number, h: Number)
```

La clase OpenLayers.Size expone los siguientes métodos públicos:

- **toString():** Devuelve una Cadena (String) del tipo : 'w=55,h=66'.
- **clone():** Devuelve un objeto OpenLayers.Size idéntico al propietario del método.
- **equals(px: OpenLayers.Pixel):** Devuelve un Boolean que indica si los objetos son equivalentes (misma anchura, misma altura).

- **OpenLayers.Bounds**

Representa una región rectangular, identificada por sus propiedades 'left', 'bottom', 'right' y 'top'.

- **OpenLayers.Icon**

Encapsula un icono. Tiene propiedades 'url', 'size', 'px' y 'offset'. También ofrece una propiedad 'calculateOffset' que permite calcular el offset en función del tamaño.

- **OpenLayers.Projection**

Representa un cambio de coordenadas.

- La clase OpenLayers.Map

En esta sección se explica la funcionalidad de la clase principal de OpenLayers, la clase OpenLayers.Map.

OpenLayers.Map es la clase fundamental de la librería OpenLayers. Todo programa de OpenLayers tiene como objeto crear un mapa que se visualizará en pantalla. Los objetos de la clase OpenLayers.Map son una colección de capas, OpenLayers.Layer, que contienen la información que se quiere mostrar, y controles, OpenLayers.Control, que permiten interactuar con el usuario. El objeto Map también es el responsable de gestionar la visualización del mapa en cuanto a Zoom y Panning se refiere.

El constructor de la clase Map tiene la siguiente forma:

```
OpenLayers.Map( div : [DOMElement/ String], options: Object)
```

El primer parámetro es una referencia al elemento 'div' del documento html destinado a contener el mapa. En lugar de una referencia se puede pasar una cadena de texto con el 'id' del elemento.

El segundo parámetro es un Array de opciones en la forma 'key:value'. Las opciones son valores de las propiedades del objeto 'Map' que se quieren fijar con un valor determinado en el constructor.

Un ejemplo de mapa sin opciones adicionales podría ser:

```
var map = new OpenLayers.Map("divMapa");
```

Es posible añadir opciones adicionales directamente en el constructor:

```
var map = new OpenLayers.Map("divMapa", {  
  projection: 'EPSG:4326',  
  units: 'm'  
  });
```

Pero también se puede preparar primero el objeto de opciones y añadirlo luego al constructor por referencia:

```
var opciones = {projection: 'EPSG:4326', units: 'm'};
```



```
var map = new OpenLayers.Map("divMapa", opciones);
```

La clase `OpenLayers.Map` expone una buena colección de propiedades y métodos.

○ Propiedades del objeto Map

Básicamente un objeto `Map` es una colección de capas (`Layer`) y controles (`Control`). A continuación se describen algunas de las propiedades, que son colecciones de objetos pertenecientes al mapa:

- **layers:** La propiedad 'layers' es un Array de objetos `OpenLayers.Layer`, que contiene la colección de capas del mapa. Para gestionar la colección de capas se dispone de los métodos: `addLayer()`, `addLayers()`, `removeLayer()`, `getNumLayers()`, `getLayerIndex()`, `setLayerIndex()`, `raiseLayer()`, `setBaseLayer()`, `getLayer()`, `getLayersBy()`, `getLayersByClass()`, `getLayersByName()`, `setLayerZIndex()`, `resetLayersZIndex()`.
- **controls:** Colección de controles asociados al mapa. Para manejar la colección se utilizan los métodos: `addControl()`, `addControls()`, `addControlToMap()`, `getControl()`, `getControlsBy()`, `getControlsByClass()` y `removeControl()`.

1.8.2. Controles

Esta sección explica la utilización de los controles en `OpenLayers` y sus diferentes tipos.

Los controles se utilizan para interactuar con el mapa. Permiten hacer zoom, mover el mapa, conocer las coordenadas del cursor, dibujar features, etc. En `OpenLayers V2.10` hay 40 clases de controles para utilizar con los mapas. De ellos, dos se van a suprimir en la próxima versión de `OpenLayers`, la versión 3.0. Los controles descatalogados son '`MouseDefaults`' y '`MouseToolBar`'. De los 38 controles restantes, dos son nuevos de esta versión, '`SLDSelect`' y '`WMTSGetFeatureInfo`'. El resto vienen de versiones anteriores, si bien el control '`OpenLayers.Control.Panel`' ha sufrido modificaciones en su funcionamiento respecto de anteriores versiones.

Todos los controles derivan de la clase '`OpenLayers.Control`', y todos los controles se añaden al objeto '`OpenLayers.Map`', directa o indirectamente. La clase '`OpenLayers.Map`' tiene una propiedad '`controls`' que guarda la lista de controles del mapa. Para añadir un control al mapa se puede hacer mediante el método

'*OpenLayers.Map.addControl()*' o bien directamente en el constructor de '*OpenLayers.Map*'.

El constructor de la clase '*OpenLayers.Map*' permite añadir controles en el momento de la creación del mapa. Si no se indica nada, el mapa añade los siguientes controles por defecto:

- *OpenLayers.Control.Navigation*
- *OpenLayers.Control.PanZoom*
- *OpenLayers.Control.ArgParser*
- *OpenLayers.Control.Attribution*

Esto es, si se crea el mapa con el siguiente constructor:

```
var map = new OpenLayers.Map("divMapa");
```

Automáticamente quedan añadidos al mapa los controles por defecto indicados. Es posible crear un mapa sin ningún control, de la siguiente manera:

```
var map = new OpenLayers.Map("divMapa", {controls: [] } );
```

Si lo que se quiere es añadir una serie de controles elegidos por el desarrollador habría que invocar el constructor del mapa de la siguiente manera:

```
var map = new OpenLayers.Map("divMapa", {  
controls: [  
new OpenLayers.Control.PanZoom(),  
new OpenLayers.Control.Attribution()  
] }  
);
```

También se puede crear un mapa sin controles y añadirselos después:

```
var map = new OpenLayers.Map("divMapa", { controls: []});  
var ctrlPanZoom = new OpenLayers.Control.PanZoom();  
map.addControl(ctrlPanZoom);  
var ctrlAttribution = new OpenLayers.Control.Attribution();  
map.addControl(ctrlAttribution);
```

En ambos casos se puede crear un mapa con los controles '*PanZoom*' y '*Attribution*', cuidando siempre de añadir el control '*Attribution*' para mostrar correctamente los *Attribution* de los mapas que se utilizan.

En OpenLayers las definiciones de estilo por defecto de controles y otros elementos se declaran y definen en la hoja de estilo:

<http://www.openlayers.org/api/theme/default/style.css>.

Siendo posible añadir un enlace a dicha hoja de estilo en el código:

```
<link type="text/css" rel="stylesheet"
href="http://www.openlayers.org/api/theme/default/style.css"/>
```

Como puede observarse se definen propiedades de estilo para todo tipo de elementos, que son fáciles de identificar por su nombre de clase. También es posible personalizar la posición o el aspecto de los controles.

La lista de controles que se pueden utilizar es la siguiente:

- Controles generales
 - o ArgParser
 - o Attribution
 - o Button
 - o Graticule
 - o KeyboardDefaults
 - o LayerSwitcher
 - o OverviewMap
 - o Panel
 - o PanPanel
 - o PermaLink
 - o SLDSelect (nuevo versión 2.10)
 - o Snapping
 - o Split

- Zoom, Panning, Position
 - o DragPan
 - o MousePosition
 - o Navigation

- NavigationHistory
 - NavToolBar
 - Pan
 - PanZoom
 - PanZoomBar
 - ZoomBox
 - ZoomIn
 - ZoomPanel
 - ZoomOut
 - ZoomToMaxExtent
- Features
- EditingToolbar
 - DragFeature
 - DrawFeature
 - GetFeature
 - ModifyFeature
 - SelectFeature
 - TransformFeature
- Medición
- Measure
 - Scale
 - ScaleLine
- Servicios de Mapas
- WMSGetFeatureInfo
 - WMTSGetFeatureInfo (nuevo versión 2.10)

1.8.3. Capas

- La clase OpenLayers.Layer

Clase importante para el manejo de los controles del mapa, cuyas propiedades se explican en esta sección.

La clase OpenLayers.Layer es la clase base para todos los tipos de capas especializadas, que son las que realmente se añaden a los mapas.

El constructor de la clase OpenLayers.Layer tiene la siguiente signatura:

```
OpenLayers.Layer( name: String, options: Object)
```

El primer parámetro es una cadena que permitirá identificar a la capa, por ejemplo en el control LayerSwitcher. El segundo parámetro es un Hashtag de opciones adicionales que se añadirán a la capa. La forma de este parámetro es un Array asociativo de parejas 'key:value' separadas por comas. El constructor no se utiliza directamente, pues se utilizan las clases derivadas, pero un ejemplo ilustrativo de la forma de pasar el segundo parámetro podría ser el siguiente:

```
var capa = new OpenLayers.Layer(  
  "L1",  
  {opacity: 0.5, isBaseLayer: false }  
);
```

La clase OpenLayers.Layer expone las siguientes propiedades:

- **id:** String.- El valor del atributo 'id' asignado al elemento 'div' de la capa.
- **name:** String.- El nombre de la capa es una cadena que permite identificar a la capa en algunas situaciones, por ejemplo en el Control LayerSwitcher.
- **div:** DOMELEMENT.- Es una referencia al elemento 'div' que alberga la capa.
- **opacity:** Float.- Es un número entre 0 (= transparente) y 1 (= opaco) que indica el grado de transparencia de la capa.
- **alwaysInRange:** Boolean.- Se debe establecer en 'true' cuando la visualización de la capa no se debe basar en zoom.
- **events:** OpenLayers.Events.- La propiedad 'events' es la colección de eventos de la capa. Es una referencia a un objeto de la clase OpenLayers.Events.

- **map**: Es una referencia al mapa que contiene a la capa. Se establece en la función `addLayer()` del mapa o en la función `setMap()` de la capa. El objeto apuntado es un `OpenLayers.Map`.
- **isBaseLayer**: Es un Boolean que indica si se trata de una capa base. Por defecto es falso. Las clases derivadas especializadas establecen un valor por defecto para cada tipo de capa.
- **alpha**: Es un Boolean que indica si las imágenes de la capa tienen canal alfa. Por defecto es `false`.
- **displayInLayerSwitcher**: Boolean que indica si el nombre de la capa debe de aparecer o no en el control `LayerSwitcher`. El valor por defecto es `'true'`.
- **visibility**: Es un Boolean que indica si la capa es visible o no. El valor por defecto es `'true'`.
- **attribution**: Se trata de la cadena, `String`, que se mostrará en el control `Attribution`.
- **inRange**: Es un Boolean que indica si el valor de la resolución actual está entre el mínimo y el máximo de la capa (`minResolution`, `maxResolution`). Se establece cada vez que cambia el zoom.
- **imageOffset**: Desplazamientos x, y debidos al borde, `'gutter'`, en las imágenes que tienen `'gutter'`. Es un objeto `OpenLayers.Pixel`.
- **options**: Se trata de un objeto mediante el cual se pueden pasar al constructor de la capa valores iniciales para cualquiera de las propiedades de la capa.
- **eventListeners**: (`Object`)
- **gutter**: (`Integer`) El valor del ancho del borde, si lo tiene. Por defecto es cero.
- **projection**: (`Object`) Objeto `OpenLayers.Projection` con la proyección de la capa. Si se pasa en el objeto `'options'` del constructor, se puede pasar como una cadena del tipo `'EPSG:4326'`, pero durante la creación de la capa se construirá un objeto `Projection`. Cuando se utiliza esta opción suele ser necesario fijar también `'maxExtent'`, `'maxResolution'` y `'units'`.
- **units**: (`String`). Las unidades de medida de la capa. Por defecto son grados, y la variable `'units'` tiene el valor `'degrees'`. Los valores posibles son: `'degrees'` (o `'dd'`), `'m'`, `'ft'`, `'km'`, `'mi'`, `'inches'`.
- **scales**: (`Array`). Un array con las escalas del mapa para cada zoom en orden descendente. Este parámetro sólo tiene sentido si está bien calibrado con la

resolución concreta del monitor en el que se está trabajando. Además debe estar bien definida la propiedad 'units' de la capa. En general es preferible utilizar la propiedad 'resolutions'.

- **resolutions:** (Array). Un array con las resoluciones del mapa para cada zoom en orden descendente. La resolución es el número de unidades de mapa por pixel. Si no se especifica al construir la capa, se calcula en base a otras propiedades de la capa (maxExtent, maxResolution, maxScale, etc).
- **maxExtent:** (OpenLayers.Bounds). El centro de estos límites no se apartará fuera del alcance de la ventana durante la panoramización. Además, sidisplayOutsideMaxExtent se establece en falso, los datos no serán requeridos completamente fuera de estos límites.

- Tipos de capas

En esta sección se muestra los dos tipos de capas más utilizados en la construcción de este sistema web geo-referenciado.

- **OpenLayers.Layer.Markers**

OpenLayers ofrece una capa especial para alojar los marcadores: OpenLayers.Layer.Markers. Un marcador es una instancia de la clase OpenLayers.Marker, que es una combinación de un icono y una posición. El constructor de la clase OpenLayers.Markers tiene la siguiente forma:

```
OpenLayers.Markers (name: String, options: Object)
```

El primer parámetro es una cadena que permitirá identificar a la capa, por ejemplo en el control LayerSwitcher. El segundo parámetro es una Hashtable de opciones extra que se quieran asignar a la capa. Las propiedades públicas de la capa Markers son:

- **isBaseLayer:** (Boolean = false) Las capas de marcadores no son capas base. Se sobrescribe la propiedad de la clase base OpenLayers.Layer con el valor false.
- **markers:** (Array(OpenLayers.Marker)) Lista de marcadores de la capa. Esta propiedad es un Array que guarda las referencias a los marcadores que se han ido añadiendo a la capa.

- **drawn:** (Boolean) Indica si se ha dibujado la capa. En algunas situaciones al inicializar la capa o hacer zoom es necesario comprobar esta variable y si es necesario llamar al método 'draw()'.

La capa Markers proporciona los siguientes métodos:

- **addMarker(marker: OpenLayers.Marker):** Añade a la capa el marcador referenciado por la variable 'marker'.
 - **removeMarker(marker: OpenLayers.Marker):** Elimina de la capa el marcador 'marker'.
 - **drawMarker(marker: OpenLayers.Marker):** Calcula la posición pixel del marcador, lo crea y añade al 'div' de la capa.
 - **clearMarkers():** Esta función deja vacía la lista de marcadores de la capa. Los marcadores en sí mismos no se eliminan, pero si se retiran de la lista de marcadores de la capa.
 - **setOpacity(opacity: float):** Establece la opacidad de todos los marcadores de la capa (0 = transparente; 1= opaco).
 - **getDataExtent():** Devuelve un OpenLayers.Bounds con la extensión que abarca todos los marcadores de la capa.
- **OpenLayers.Layer.Vector**

Las capas vectoriales están pensadas para alojar 'features' vectoriales, que serán instancias de las clases derivadas de OpenLayers.Feature.Vector. El constructor de la clase OpenLayers.Layer.Vector admite dos parámetros:

```
OpenLayers.Layer.Vector ( name : String, options: Object );
```

El parámetro 'nombre' es una cadena de texto que sirva para identificar la capa y el parámetro 'options', que es opcional, es un objeto con propiedades de la capa que se quiera establecer en un valor distinto del que se asigna por defecto. Una vez creada la capa se le pueden ir añadiendo las "features" que se quieran visualizar mediante el método 'addFeatures()' que admite como parámetro un Array de features. Dichas 'features' habrán sido creadas previamente y serán instancias de la clase OpenLayers.Feature.Vector. La geometría y el estilo de visualización asignados a la "feature" serán las que marquen la forma en que se visualizarán las "features" en la

capa. La capa, una vez creada, se añade al mapa mediante el método `OpenLayers.Map.addLayer()`. Un ejemplo de una capa vectorial con un punto podría ser:

```
var ptgeom = new OpenLayers.Geometry.Point(-3.7856, 42.2540);
var ptstyle = OpenLayers.Util.extend({},
OpenLayers.Feature.Vector.style['default']);
var ptfeat = new OpenLayers.Feature.Vector(
ptgeom, null, ptstyle);
var layerVector = new OpenLayers.Layer.Vector("Capa Vectorial");
layerVector.addFeatures([feat]);
map.addLayer(layerVector);
```

1.9. OpenStreetMap

OpenStreetMap, también conocido como OSM, es un servidor de mapas creado como proyecto colaborativo para crear mapas libres y editables.

Los mapas son creados a través de colaboraciones de usuarios que a nivel mundial suben información geo-referenciada, a través de dispositivos GPS móviles, ortofotografías y otras fuentes libres. Esta cartografía se distribuye de forma totalmente libre bajo una licencia abierta Open Database License (ODbL).

Los usuarios pueden registrarse de manera gratuita y subir sus trazas GPS; también pueden crear y corregir errores en la cartografía mediante herramientas de edición elaboradas por la comunidad OpenStreetMap.

En la actualidad Yahoo© y Bing© liberaron sus fotos aéreas para que los usuarios puedan dibujar la cartografía de manera más eficiente, y así, incrementar el volumen de datos cartográficos a nivel mundial.

1.9.1. Tecnología

OpenStreetMap utiliza una estructura de datos topológica. Los datos se almacenan en el datum WGS84 lat/lon (EPSG:4326) de proyección de Mercator.

Los servidores principales se encuentran alojados en la University College de Londres. La infraestructura de servidores asociada al proyecto OSM se encuentra conformada por un servidor de base de datos de gran rendimiento, un servidor de aplicaciones

para el sitio web, tres servidores de aplicaciones para la API y un servidor destinado al renderizado del mapa.

El servidor de base de datos utiliza como sistema de gestión de base de datos PostgreSQL más su extensión espacial PostGIS para el almacenamiento de geometrías de objetos espaciales en formato vectorial; además se utiliza Mapnik© como renderizador de estos datos espaciales que se almacenan en la misma. El sitio web y la API están programados en gran medida en Ruby on Rails.

CAPÍTULO II

DESARROLLO DEL SISTEMA

2.1. Ápice arquitectónico

Esta sección tiene como objetivo explicar a detalle todo el proceso realizado en la fase de planificación del desarrollo del sistema web geo-referenciado. Como se señaló con anterioridad, el presente trabajo se rige a la metodología ágil de desarrollo de software “XP” (eXtreme Programming).

2.1.1. Interacción con el cliente

Respecto a la interacción con el cliente, se identificaron todos los posibles escenarios para la definición de requerimientos del sistema, en metodologías tradicionales conocidos como casos de uso, en el caso de XP las “historias de usuario”. Como primer preámbulo se crearon todas las historias de usuario posibles, las cuales constituyen el primer boceto de la arquitectura del sistema, definiendo las principales necesidades del cliente respecto al uso del sistema.

A continuación se detallan las historias de usuario que se crearon para planificar el desarrollo del sistema:

Tabla 2.1. Historia de usuario Nro. 1. Fuente propia.

HISTORIA DE USUARIO	
Número: 1	Usuario: Administrador
Nombre: Gestión administrativa del sistema	
Modificación de historia número: 1	Iteración asignada: 1
Prioridad en negocio: Alta	Riesgo en desarrollo: Bajo
Puntos estimados: 4	Puntos reales: 3
Descripción: La gestión administrativa del sistema como tal, conlleva la gestión de usuarios del	

sistema y la gestión de los dispositivos GPS administrados; por ende el usuario administrador del sistema debe estar en capacidad de crear, consultar, actualizar y borrar, sea éstos: usuarios o dispositivos GPS administrados por el sistema.

Observaciones:

Tabla 2.2. Historia de usuario Nro. 2. Fuente propia.

HISTORIA DE USUARIO	
Número: 2	Usuario: Cliente
Nombre: Visualización del vehículo en tiempo ficticio.	
Modificación de historia número: 1	Iteración asignada: 2
Prioridad en negocio: Baja	Riesgo en desarrollo: Alto
Puntos estimados: 4	Puntos reales: 6
Descripción: Visualizar el vehículo en la aplicación web en tiempo ficticio, es decir simulando los datos reales del dispositivo.	
Observaciones: Realizar el respectivo proceso para que se generen datos ficticios en la base de datos.	

Tabla 2.3. Historia de usuario Nro. 3. Fuente propia.

HISTORIA DE USUARIO	
Número: 3	Usuario: Cliente
Nombre: Visualización organizada de los datos del vehículo en tiempo ficticio.	
Modificación de historia número: 1	Iteración asignada: 3
Prioridad en negocio: Media	Riesgo en desarrollo: Medio
Puntos estimados: 4	Puntos reales: 4

<p>Descripción:</p> <p>Visualizar los datos del vehículo de forma organizada en la aplicación web: ID, nombre, fecha y hora de envío de los geo-datos, latitud, longitud, altitud, velocidad, orientación, dirección y otros; de manera que se pueda diferenciar los datos de cada vehículo y además se muestren y oculten en el mapa.</p>
<p>Observaciones:</p> <p>Realizar el respectivo proceso para que se generen los datos ficticios en la base de datos.</p>

Tabla 2.4. Historia de usuario Nro. 4. Fuente propia.

HISTORIA DE USUARIO	
Número: 4	Usuario: Cliente
Nombre: Histórico de un mes de posiciones y velocidades.	
Modificación de historia número: 1	Iteración asignada: 4
Prioridad en negocio: Media	Riesgo en desarrollo: Medio
Puntos estimados: 4	Puntos reales: 5
<p>Descripción:</p> <p>Generar un reporte de posiciones y velocidades del vehículo que han transcurrido en un mes.</p>	
<p>Observaciones:</p>	

Tabla 2.5. Historia de usuario Nro. 5. Fuente propia.

HISTORIA DE USUARIO	
Número: 5	Usuario: Cliente
Nombre: Visualización del vehículo en tiempo real.	

Modificación de historia número: 1	Iteración asignada: 5
Prioridad en negocio: Alta	Riesgo en desarrollo: Alto
Puntos estimados: 6	Puntos reales: 6
Descripción: Visualizar el vehículo en la aplicación web en tiempo real, es decir el gráfico del vehículo en la aplicación debe aparecer exactamente dónde se encuentre el vehículo en la realidad.	
Observaciones: La actualización de la posición se realizará cada 3 minutos.	

2.1.2. Estudio de las herramientas, tecnologías y prácticas de desarrollo

Luego de la interacción con el cliente con su consecuente elaboración de las historias de usuario y acorde con el marco teórico, se definen las herramientas, tecnologías y prácticas de desarrollo, las cuales se detallan en el siguiente literal.

- Herramientas de desarrollo

La experimentación de otras herramientas como Netbeans 7.1.2, permitió definir la herramienta de desarrollo: el IDE (Interface Development Environment) Eclipse® Indigo 3.7.1 con su plug-in para Java EE versión 1.4.1. La razón por la cual se optó por Eclipse® Indigo 3.7.1 en lugar de Netbeans 7.1.2 fue que este primero posee una extensión por defecto que permite ejecutar un entorno de contenedor de servlets y servidor web Tomcat, por lo que es más sencillo hacer las pruebas unitarias y globales, y también adaptar la publicación de la aplicación web en un entorno de tipo Tomcat.

- Tecnologías de desarrollo

Respecto a las tecnologías de desarrollo, se utilizará la plataforma de programación Java Enterprise Edition para la construcción de los módulos de administración, reportes y seguridad, y la plataforma de programación Java Standard Edition para la construcción del módulo de comunicaciones, ya que previamente se realizaron

pruebas de concepto con la plataforma de programación PHP en el diseño de los prototipos para los módulos de administración, reportes y seguridad.

En lo que refiere al módulo de mapas, no hubo otra opción que definir la utilización del lenguaje de programación del lado del cliente (navegador web) a JavaScript, como también la librería utilitaria OpenLayers versión 2.11 basada en JavaScript, para el manejo de las características del visor del mapa; en consecuencia, se definió la utilización de AJAX como tecnología de programación para clientes ricos, es decir, para que la aplicación sea dinámica y asíncrona.

- Prácticas de desarrollo

Como prácticas de desarrollo se ha considerado ciertos estándares de programación y el patrón de diseño MVC en los cuales se ha basado el desarrollo del presente trabajo, sin tomar en cuenta la metodología de desarrollo. A continuación se enumeran algunos estándares de programación que se manejaron en el transcurso de construcción del sistema:

- Nombrar a las tablas de base de datos manejando el siguiente esquema:

Anteponer un prefijo que se rija al siguiente patrón: <g>&<u> |<l> | <p>&<u>&<r>.

<g>: Anteponer la letra “g” si se trata de una tabla que contenga al menos un campo geométrico.

<u> | <l> | <p>: Poner la letra “u” si se trata de un punto, la letra “l” si es una línea o la letra “p” si es un polígono en el espacio.

<u>: Esta letra es obligatoria, quiere decir UTN (Universidad Técnica del Norte) entidad a la que pertenece el proyecto.

<r>: Esta letra quiere decir que se trata de un proyecto de “rastreo”.

- No poner código CSS en las páginas JSP.
- Nombrar a los métodos o funciones con letras minúsculas y a las clases la letra inicial con mayúscula, sea en Java, JavaScript o PHP.

2.1.3. Construcción del prototipo de estructura y diseño

La construcción del primer prototipo se realiza en partes; la primera consiste en diseñar la base de datos, la que será el almacén de la aplicación y de dónde se iniciará el proceso de desarrollo del sistema web geo-referenciado completo. Posteriormente se desarrolla la prueba de concepto del módulo visor de mapas, donde se muestran las funcionalidades básicas que este módulo ofrecerá al ser finalizado.

- Modelo entidad-relación de la base de datos

En esta sección de la construcción del prototipo, se diseñó el modelo de la base de datos relacional que interactuaría con todos los componentes funcionales de la aplicación. Definiendo tres tablas de base de datos, las cuales se detallan a continuación:

- **URUSUARIO:** Tabla de base de datos que almacena la información referente a los usuarios de la aplicación.
- **URAVL:** Tabla de base de datos que almacena la información referente a los AVL's manejados por la aplicación.
- **GUURDATO:** Tabla de base de datos espacial que almacena las tramas enviadas por el AVL mediante la conexión GPRS.

A continuación se muestra el diagrama del modelo entidad-relación de la base de datos, con sus respectivos campos de tabla y las relaciones entre las tablas.

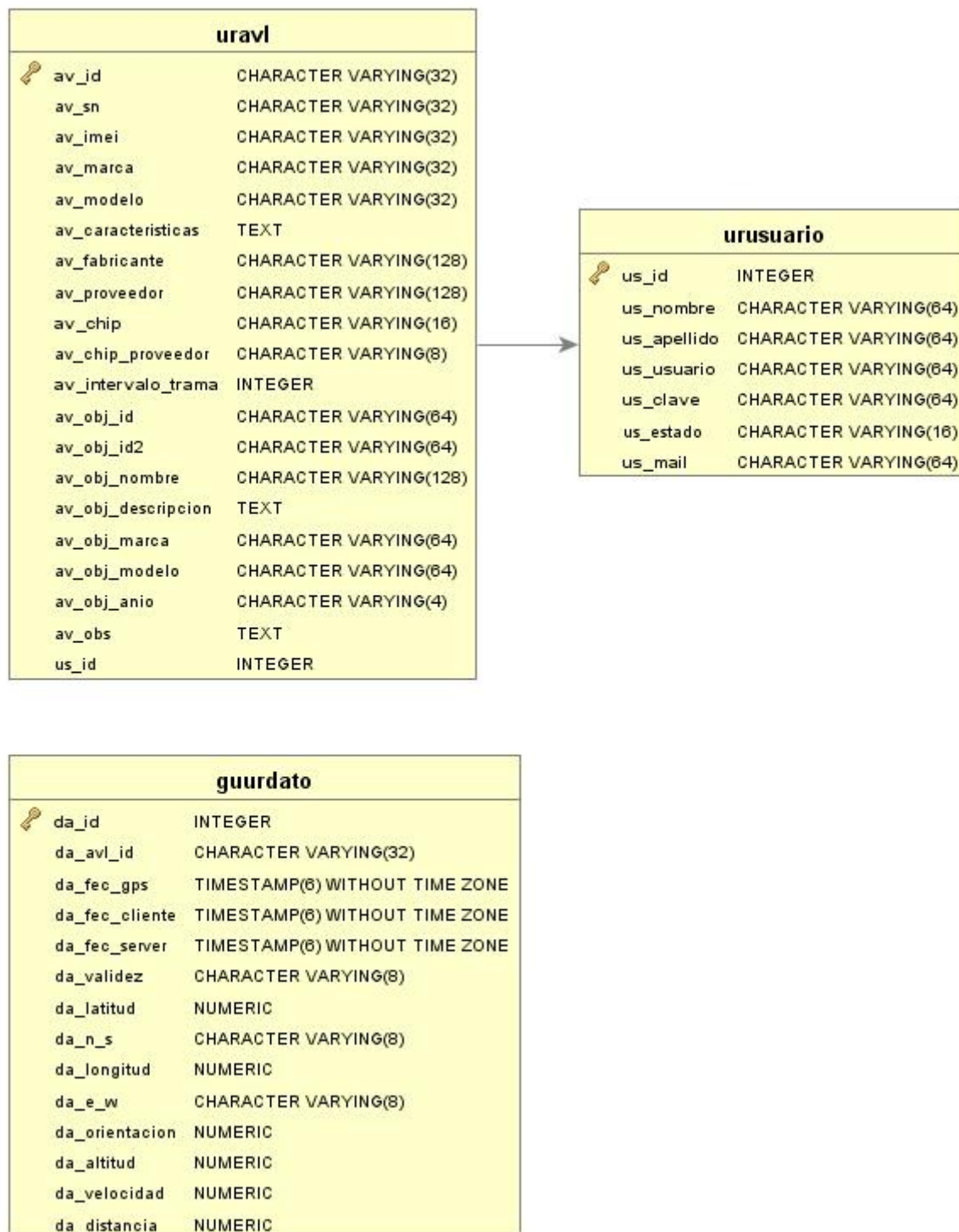


Ilustración 2.1: Diagrama Entidad-Relación de la base de datos. Fuente propia.

- Prototipo del módulo visor de mapas

Cabe recalcar que el prototipo del módulo visor de mapas fue desarrollado bajo el lenguaje de programación PHP en el lado del servidor, y JavaScript en el lado del cliente. A continuación se muestra una imagen del prototipo que se desarrolló.

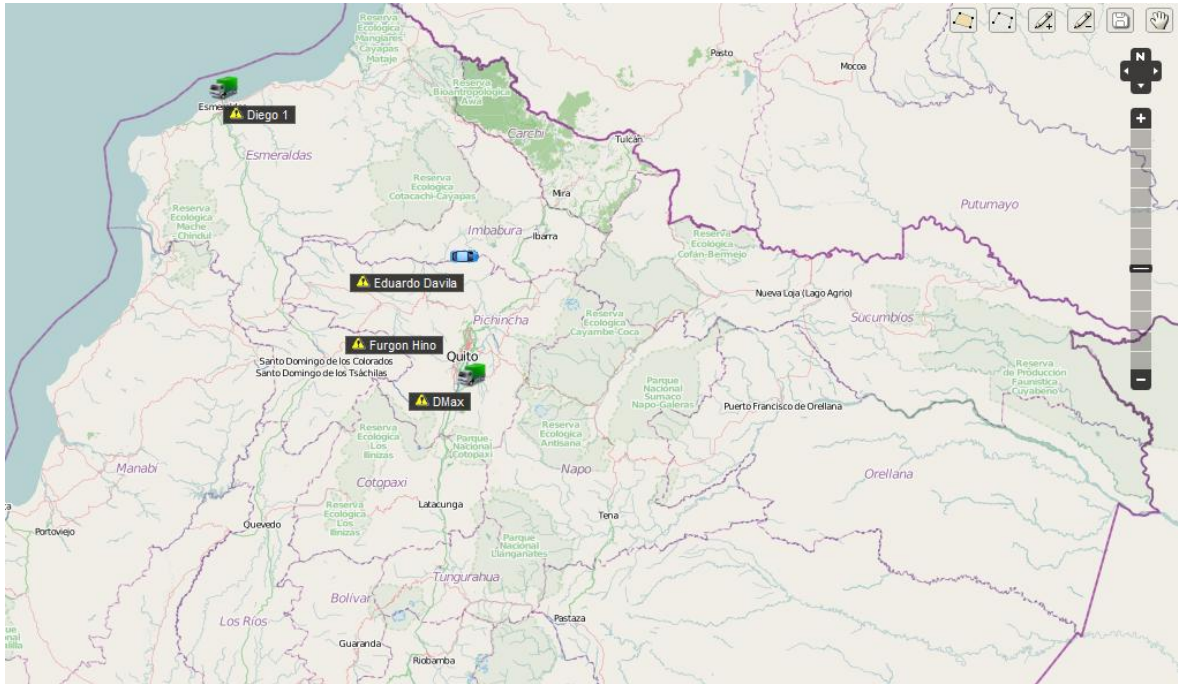


Ilustración 2.2: Prototipo del módulo visor de mapas. Fuente propia.

2.2. Plan de entregas

2.2.1. Priorización de las historias de usuario

Dadas las historias de usuario detalladas anteriormente, se procedió a la priorización de las mismas; cuyas prioridades, riesgos, esfuerzos e iteraciones asignadas se organizaron para su posterior ejecución. En la siguiente tabla se muestran las historias de usuario con sus respectivos valores.

Tabla 2.6: Priorización de las historias de usuario. Fuente propia.

Nro.	Nombre	Prioridad	Riesgo	Esfuerzo	Iteración
1	Gestión administrativa del sistema	Alta	Bajo	3.5	1
2	Visualización del vehículo en	Baja	Alto	5	2

	tiempo ficticio				
3	Visualización organizada de los datos del vehículo en tiempo ficticio	Media	Medio	4	3
4	Histórico de un mes de posiciones y velocidades	Media	Medio	4.5	4
5	Visualización del vehículo en tiempo real	Alta	Alto	6	5

2.2.2. Elaboración del plan de entregas

En el caso de la planificación del presente trabajo se asignó cada historia de usuario a una sola iteración, en consecuencia se obtuvo una planificación de cinco iteraciones; esto, por tratarse de software hecho a la medida, lo cual rompe las convencionalidades respecto al desarrollo del software tradicional. De manera que las iteraciones se definieron así:

- **Iteración primera:** En esta iteración se prepararán las funcionalidades básicas relacionadas con la administración del sistema web geo-referenciado en forma global.
- **Iteración segunda:** En esta iteración se pretende entregar el software con sus funcionalidades completas en tiempo ficticio, es decir la parte administrativa y la parte geográfica del sistema.
- **Iteración tercera:** En esta iteración se entregará una aplicación más enriquecida con características que permitan una mejor organización de la información mostrada.
- **Iteración cuarta:** En esta iteración se realizará la entrega de la aplicación con una característica agregada que permita hacer un reporte gráfico del histórico de posiciones de máximo un mes.

- **Iteración quinta:** Siendo esta la última iteración se acometerá en entregar la aplicación finalizada, es decir con todas sus funcionalidades, permitiendo al usuario visualizar su vehículo en tiempo real en el mapa web.

El plan de entregas queda definido de la manera como se muestra en el diagrama de Gantt de la Ilustración 2.3.

Iteración	Historia de usuario	Comienzo	Fin	Duración	dic 2011	ene 2012					feb 2012					mar 2012					abr 2012					may 2012				
						1/1	8/1	15/1	22/1	29/1	5/2	12/2	19/2	26/2	4/3	11/3	18/3	25/3	1/4	8/4	15/4	22/4	29/4	6/5	13/5	20/5				
1	Gestión administrativa del sistema	19/12/2011	12/01/2012	19d																										
2	Visualización del vehículo en tiempo ficticio	13/01/2012	09/02/2012	20d																										
3	Visualización organizada de los datos del vehículo en tiempo ficticio	10/02/2012	29/02/2012	14d																										
4	Histórico de un mes de posiciones y velocidades	01/03/2012	05/04/2012	26d																										
5	Visualización del vehículo en tiempo real	06/04/2012	31/05/2012	40d																										

Ilustración 2.3: Diagrama de Gantt del plan de entregas. Fuente propia.

2.3. Construcción

2.3.1. Iteraciones

En esta sección se detallarán cada una de las iteraciones, describiendo las tareas que se llevaron a cabo, así como también las pruebas de aceptación y las incidencias.

- Iteración primera

Esta iteración consta de una sola historia de usuario:

1. Gestión administrativa del sistema.

- Asignación de tareas

El primer paso de la iteración, fue asignar las tareas respectivas para cada historia de usuario de la iteración. Por lo tanto, a continuación se muestran las tareas asignadas a la única historia de usuario para esta iteración.

Tabla 2.7. Tarea Nro. 1 de la historia Nro. 1. Fuente propia.

TAREA	
Número: 1	Número de historia: 1
Nombre: Construcción y comprobación de la base de datos.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 19/12/2011	Fecha de fin: 21/12/2011
Descripción: Escritura del script de base de datos en Lenguaje de Consulta Estructurada (SQL) para su posterior ejecución y comprobación de su funcionamiento.	

Tabla 2.8. Tarea Nro. 2 de la historia Nro. 1. Fuente propia.

TAREA	
Número: 2	Número de historia: 1
Nombre: Inserción de datos ficticios en la base de datos.	

Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 22/12/2011	Fecha de fin: 10/01/2012
Descripción: Elaboración de los procedimientos almacenados (store procedures) de base de datos para la inserción de datos ficticios que permitan simular la base de datos real para la administración del sistema.	

Tabla 2.9. Tarea Nro. 3 de la historia Nro. 1. Fuente propia.

TAREA	
Número: 3	Número de historia: 1
Nombre: Elaboración de una prueba de concepto de reporte.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 11/01/2012	Fecha de fin: 16/01/2012
Descripción: Elaboración de una prueba de concepto que contemple la ejecución de un reporte de una tabla de base de datos específica, se desarrollará un reporte de velocidades de un vehículo en particular, considerando como parámetros la fecha de inicial y la fecha final.	

Tabla 2.10. Tarea Nro. 4 de la historia Nro. 1. Fuente propia.

TAREA	
Número: 4	Número de historia: 1
Nombre: Elaboración del CRUD de administración de AVL's.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 17/01/2012	Fecha de fin: 22/01/2012
Descripción:	

Elaboración de una prueba de concepto, realizando una página web CRUD (Create, Read, Update, Delete) de una tabla de base de datos, se desarrollará un CRUD de la administración de AVL's (dispositivos GPS que administra el sistema).

Tabla 2.11. Tarea Nro. 5 de la historia Nro. 1. Fuente propia.

TAREA	
Número: 5	Número de historia: 1
Nombre: Elaboración del CRUD de administración de usuarios	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 23/01/2012	Fecha de fin: 26/01/2012
<p>Descripción:</p> <p>Elaboración del CRUD definitivo para la administración de los datos accesibles por el administrador del sistema y en consecuencia la página web CRUD para su administración.</p>	

○ Modificación del plan de entregas

Una vez establecidas las tareas se procedió a su realización de acuerdo al cronograma, sin existencia de cambios en el plan de entregas propuesto inicialmente. Cabe señalar que no se han hecho modificaciones en el plan de entregas, a causa de que la solución informática del presente trabajo no está sujeta a cambios de requerimientos ya que no existe una interacción con un cliente real.

○ Demo de la primera iteración

A continuación se muestran las imágenes de la primera versión de la administración de usuarios y administración de AVL's cubriendo así la historia de usuario de esta iteración: la gestión administrativa del sistema.

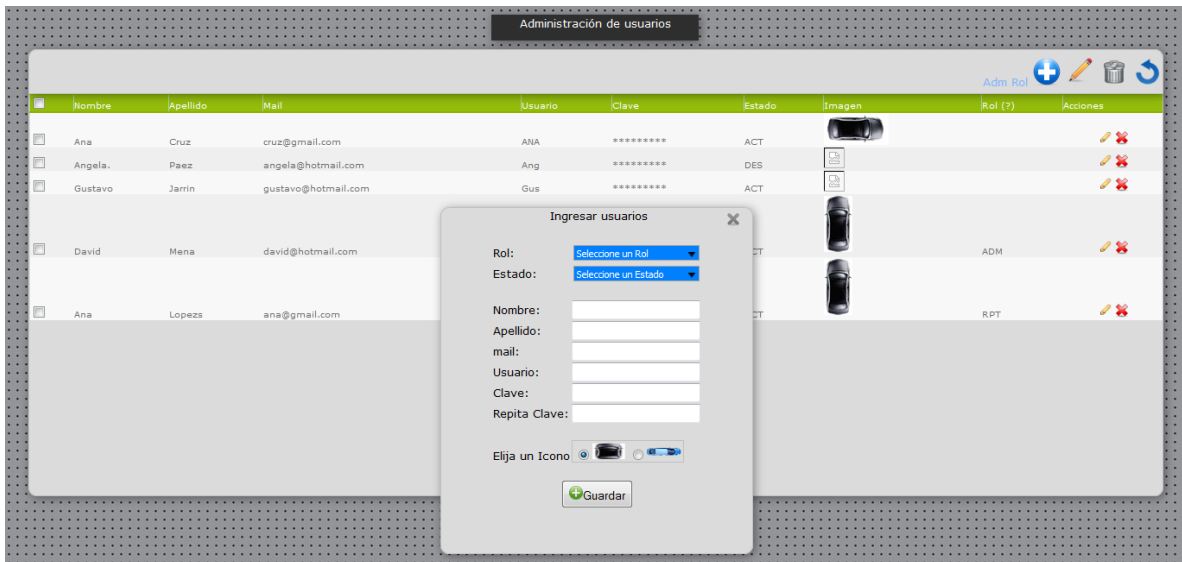


Ilustración 2.4: Administración de usuarios. Fuente propia.

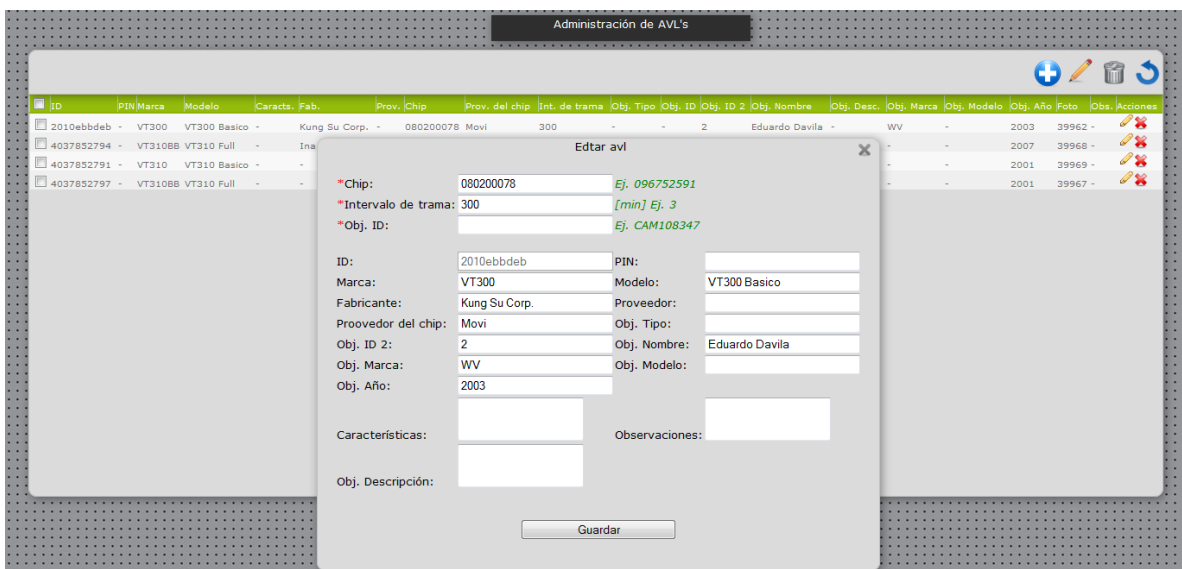


Ilustración 2.5: Administración de AVL's. Fuente propia.

○ Pruebas de aceptación

Culminada la iteración se procedió a realizar las pruebas de aceptación, las cuales se muestran en las tablas siguientes.

Tabla 2.12. Caso de prueba Nro. 1 de la historia Nro. 1. Fuente propia.

CASO DE PRUEBA	
Número: 1	Número de historia: 1

Nombre: CRUD correcto de administración de usuarios.
Descripción: <p>La aplicación debe ejecutar el CRUD (Create, Read, Update, Delete) correctamente sobre la base de datos respecto a la administración de usuarios; el sistema deberá notificar cada una las acciones realizadas.</p>
Condiciones de ejecución: <p>El sistema debe recibir los datos que son obligatorios correctamente; es decir, escritos de manera lógica.</p>
Entradas: <p>Todos los datos respectivos para el CRUD de usuarios del sistema.</p>
Resultado esperado: <p>Resultado satisfactorio, pero se evidenciaron leves errores de diseño de interfaz.</p>
Evaluación: 8/10

Tabla 2.13. Caso de prueba Nro. 2 de la historia Nro. 1. Fuente propia.

CASO DE PRUEBA	
Número: 2	Número de historia: 1
Nombre: CRUD correcto de administración de AVL's.	
Descripción: <p>La aplicación debe ejecutar el CRUD (Create, Read, Update, Delete) correctamente sobre la base de datos respecto a la administración de los AVL's; el sistema deberá notificar cada una las acciones realizadas.</p>	
Condiciones de ejecución: <p>El sistema debe recibir todos los datos que son obligatorios correctamente; es decir escritos de manera lógica.</p>	
Entradas:	

Todos los datos respectivos para el CRUD de AVL's del sistema.
Resultado esperado: Resultado satisfactorio, pero se evidenciaron detalles faltantes en el diseño de interfaz.
Evaluación: 9/10

○ Incidencias

- La creación de la base de datos de acuerdo con las especificaciones ha requerido gran parte de la iteración, la cual ha sufrido varios cambios.
- No se experimentó cambios en los requerimientos.
- La migración de este módulo de la plataforma de programación de PHP a Java se realizará posteriormente en una nueva iteración.

● Iteración segunda

Esta iteración consta de la historia de usuario: Visualización del vehículo en tiempo ficticio.

○ Asignación de tareas

Al igual que en la primera iteración se procede a planificar y asignar las tareas respectivas para la historia de usuario de la iteración. En consecuencia, se muestran las tareas asignadas en las siguientes tablas.

Tabla 2.14. Tarea Nro. 1 de la historia Nro. 2. Fuente propia.

TAREA	
Número: 1	Número de historia: 2
Nombre: Inserción de datos ficticios en la base de datos.	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 27/01/2012	Fecha de fin: 31/01/2012
Descripción: Elaboración del procedimiento almacenado (store procedure) de base de datos que	

permita la inserción de datos ficticios en la tabla GUURDATO para simular la inserción de tramas enviadas por los AVL's.

Tabla 2.15. Tarea Nro. 2 de la historia Nro. 2. Fuente propia.

TAREA	
Número: 2	Número de historia: 2
Nombre: Graficación del vehículo en el mapa de la aplicación	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 01/02/2012	Fecha de fin: 10/02/2012
Descripción: Graficación del vehículo en el mapa de la aplicación web en tiempo ficticio, es decir, con los datos que se simularon en la base de datos.	

○ Modificación del plan de entregas

No existió la necesidad de hacer cambios en el plan de entregas inicial.

○ Demo de la segunda iteración

Tras la ejecución de las tareas para la historia de usuario de esta iteración, se obtiene otra entrega de la aplicación. A continuación se muestra una imagen de la versión.

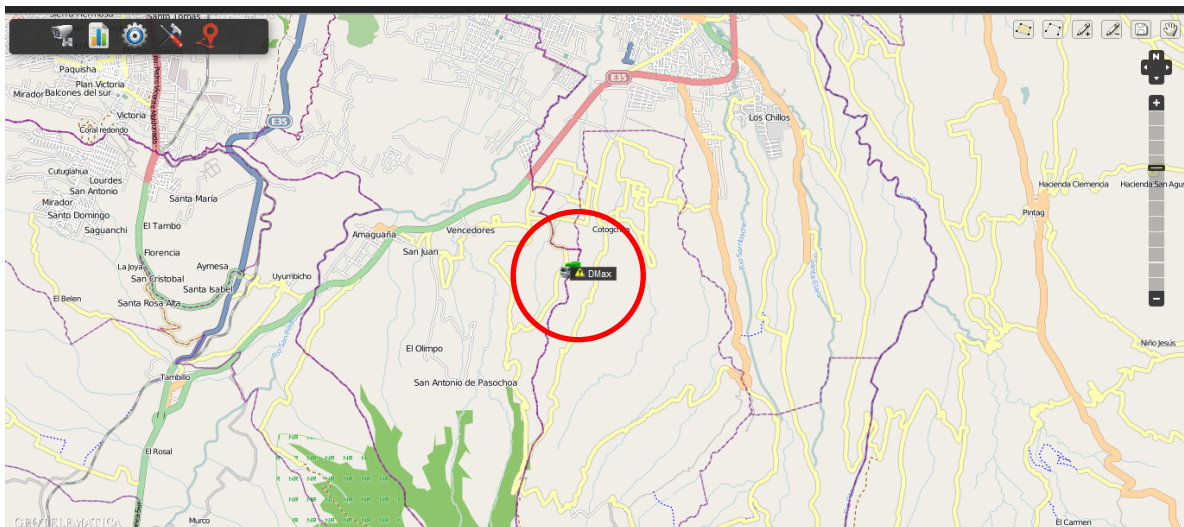


Ilustración 2.6: Visualización del vehículo en la aplicación. Fuente propia.

○ Pruebas de aceptación

Las pruebas realizadas se muestran en las siguientes tablas.

Tabla 2.16. Caso de prueba Nro. 1 de la historia Nro. 2. Fuente propia.

CASO DE PRUEBA	
Número: 1	Número de historia: 2
Nombre: Graficación correcta del vehículo en la aplicación.	
Descripción: El gráfico del vehículo debe mostrarse correctamente de acuerdo a la última posición generada ficticiamente en la aplicación.	
Condiciones de ejecución: La simulación de la base de datos debe ser correcta, para que genere los datos de las posiciones aproximadas a la realidad, es decir, las ubicaciones deben de coincidir en el territorio ecuatoriano.	
Entradas: Los datos de latitud y longitud generados ficticiamente por los procedimientos almacenados de base de datos (“store procedures”).	
Resultado esperado: Resultado satisfactorio.	
Evaluación: 10/10	

○ Incidencias

- Demandó un tiempo considerable la construcción del procedimiento almacenado que genera los datos ficticios en la tabla GUURDATO, la cual contiene las tramas enviadas por los AVL’s administrados por el sistema.
- Algunas variaciones en la base de datos, agregación del campo espacial en la tabla GUURDATO para un futuro geo-análisis.
- No se experimentó cambios en el plan de entregas inicial.

- La migración de este módulo de la plataforma de programación de PHP a Java se realizará posteriormente en una nueva iteración.

- Iteración tercera

Esta iteración consta de la historia de usuario: Visualización organizada de los datos del vehículo en tiempo ficticio.

- Asignación de tareas

Se realizó de acuerdo a la planificación, para la historia de usuario de esta iteración. A continuación, se muestran las tareas asignadas en las siguientes tablas.

Tabla 2.17. Tarea Nro. 1 de la historia Nro. 3. Fuente propia.

TAREA	
Número: 1	Número de historia: 3
Nombre: Elaboración del panel de control	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 13/02/2012	Fecha de fin: 25/02/2012
Descripción: Desarrollo del panel de control donde se mostrará la velocidad promedio del vehículo y las opciones del histórico de posiciones y los detalles de la trama enviada por el AVL.	

Tabla 2.18. Tarea Nro. 2 de la historia Nro. 3. Fuente propia.

TAREA	
Número: 2	Número de historia: 3
Nombre: Detalles de la trama al hacer clic en el vehículo	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 27/02/2012	Fecha de fin: 05/03/2012
Descripción: (El contenido de la descripción de esta tarea no es legible en la imagen)	

Desarrollo de la funcionalidad “clic en vehículo” en el mapa de la aplicación, que muestre los detalles de la última trama recibida de ese vehículo. Manejar una estructura de programación que permita ver solamente la última posición del vehículo.

- Modificación del plan de entregas

No se realizaron cambios en el plan de entregas inicial.

- Demo de la tercera iteración

Tras la ejecución de las tareas para la historia de usuario de esta iteración, se obtiene otra entrega de la aplicación. A continuación se muestran las imágenes de la entrega para cada tarea respectivamente.

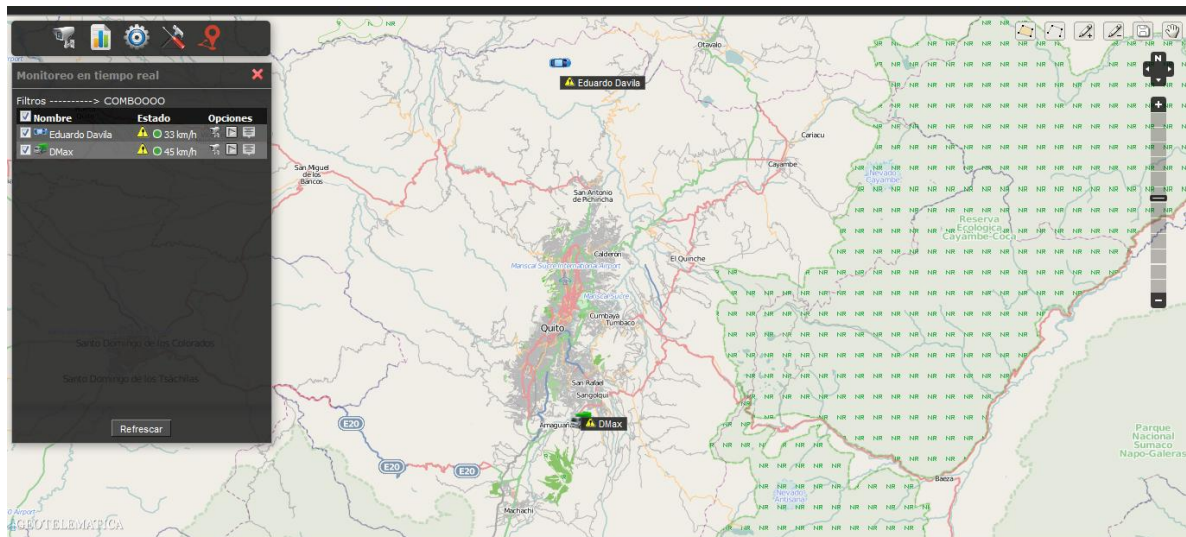


Ilustración 2.7: Panel de control. Fuente propia.

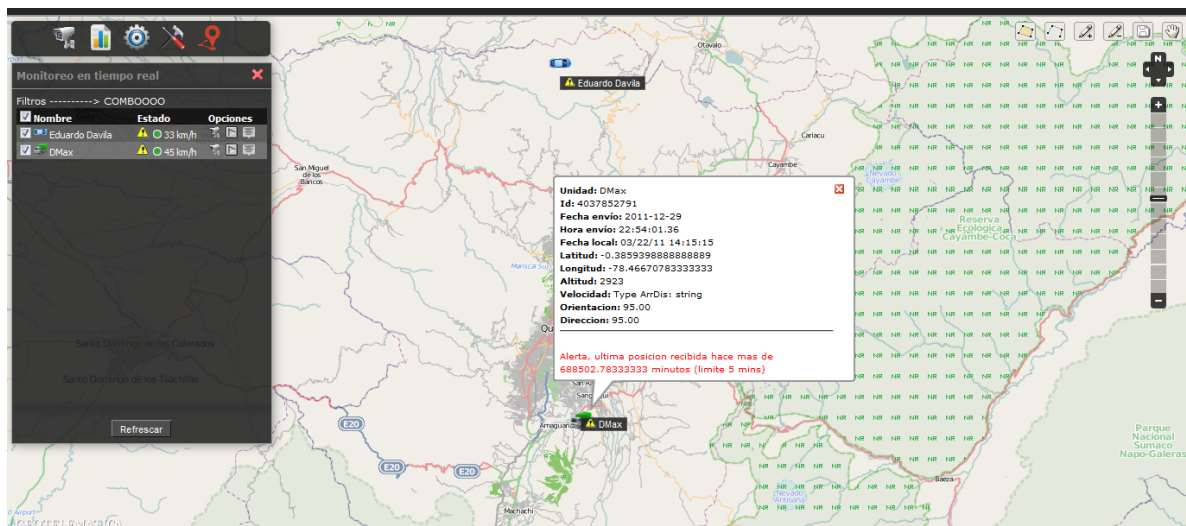


Ilustración 2.8: Detalles de la última trama recibida. Fuente propia.

○ Pruebas de aceptación

Acabada la iteración se procedió a realizar las pruebas de aceptación correspondientes, las cuales se muestran en las siguientes tablas.

Tabla 2.19. Caso prueba Nro. 1 de la historia Nro. 3. Fuente propia.

CASO DE PRUEBA	
Número: 1	Número de historia: 3
Nombre: Visualización correcta de los datos del vehículo.	
Descripción: El gráfico del vehículo debe mostrarse correctamente, de acuerdo a la última posición generada ficticiamente en la aplicación.	
Condiciones de ejecución: La simulación de la base de datos debe ser correcta, para que genere los datos de las posiciones aproximadas a la realidad, es decir, las ubicaciones deben de coincidir en el territorio ecuatoriano.	
Entradas: Los datos de latitud y longitud generados ficticiamente por los procedimientos	

almacenados de base de datos (“store procedures”).
Resultado esperado: Resultado satisfactorio.
Evaluación: 10/10

○ Incidencias

- No se experimentó cambios considerables en el plan de entregas inicial.
- Al igual que las anteriores iteraciones, se hace imprescindible postergar hacia una nueva iteración la migración de la plataforma de programación de servidor, de PHP a Java.

● Iteración cuarta

Esta iteración consta de la historia de usuario: Histórico de un mes de posiciones y velocidades.

○ Asignación de tareas

Al igual que las anteriores iteraciones se procedió a planificar y asignar las tareas respectivas para la historia de usuario de esta iteración. Las siguientes tablas muestran las tareas asignadas a la historia de usuario para esta iteración:

Tabla 2.20. Tarea Nro. 1 de la historia Nro. 4. Fuente propia.

TAREA	
Número: 1	Número de historia: 4
Nombre: Construcción del formulario de parametrización	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 06/03/2012	Fecha de fin: 10/03/2012
Descripción: Construcción del formulario para el ingreso de la fecha de inicio y fecha de finalización del reporte gráfico de histórico de posiciones, que restrinja la parametrización de manera que las fechas no excedan un tiempo mayor a un mes.	

Tabla 2.21. Tarea Nro. 2 de la historia Nro. 4. Fuente propia.

TAREA	
Número: 2	Número de historia: 4
Nombre: Animación del histórico (playback)	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 11/03/2012	Fecha de fin: 25/03/2012
<p>Descripción:</p> <p>Elaboración de la animación del histórico de posiciones recorridas hace un mes sobre el mapa de la aplicación, según los datos de entrada del formulario de parametrización.</p>	

Tabla 2.22. Tarea Nro. 3 de la historia Nro. 4. Fuente propia.

TAREA	
Número: 3	Número de historia: 4
Nombre: Elaboración del reporte de velocidades	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 31/03/2012	Fecha de fin: 06/04/2012
<p>Descripción:</p> <p>Elaboración del reporte de velocidades para cada vehículo/usuario de acuerdo a los parámetros: fecha de inicio y fecha de fin, cuyo intervalo no debe ser mayor a un mes.</p>	

- Modificación del plan de entregas

No se realizaron cambios en el plan de entregas inicial.

○ Demo de la cuarta iteración

Luego de la ejecución de las tareas para la historia de usuario de esta iteración, se obtiene otra entrega de la aplicación. A continuación se muestra una imagen de la entrega.

The screenshot shows a web application titled 'Reporte de velocidad'. At the top, there is a search bar with 'Dispositivo: Seleccione un dispositivo', a date range 'Desde 1998-07-23 16:50:00 hasta 2012-07-13 16:50:00', and a speed filter 'Vel: 70 km/h'. Below this is a table with the following columns: Dispositivo, Fecha Envío, Fecha Local, Hora, Latitud, Longitud, and Velocidad. The table contains 30 rows of data, all showing a speed of 80.00 km/h for a device named 'Diego 1'.

Dispositivo	Fecha Envío	Fecha Local	Hora	Latitud	Longitud	Velocidad
Diego 1	03/22/11 14:15:15	2011-12-29	11:45:02	1.0509295555555556	-80.32507283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	15:30:02	1.1184295555555556	-81.00007283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	09:54:02	1.0176295555555556	-79.99207283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	15:33:02	1.1193295555555556	-81.00907283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	11:48:02	1.0518295555555556	-80.33407283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	15:36:02	1.1202295555555556	-81.01807283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	08:30:02	0.9924295555555556	-79.74007283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	15:39:02	1.1211295555555556	-81.02707283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	11:51:02	1.0527295555555556	-80.34307283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	15:42:02	1.1220295555555556	-81.03607283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	09:57:02	1.0185295555555556	-80.00107283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	15:45:02	1.1229295555555556	-81.04507283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	11:54:02	1.0536295555555556	-80.35207283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	15:48:02	1.1238295555555556	-81.05407283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	09:00:02	1.0014295555555556	-79.83007283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	15:51:02	1.1247295555555556	-81.06307283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	11:57:02	1.0545295555555556	-80.36107283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	15:54:02	1.1256295555555556	-81.07207283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	10:00:02	1.0194295555555556	-80.01007283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	15:57:02	1.1265295555555556	-81.08107283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	12:00:02	1.0554295555555556	-80.37007283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	16:00:02	1.1274295555555556	-81.09007283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	08:09:02	0.9861295555555556	-79.67707283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	16:03:02	1.1283295555555556	-81.09907283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	12:03:02	1.0563295555555556	-80.37907283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	16:06:02	1.1292295555555556	-81.10807283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	10:03:02	1.0203295555555556	-80.01907283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	16:09:02	1.1301295555555556	-81.11707283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	12:06:02	1.0572295555555556	-80.38807283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	16:12:02	1.1310295555555556	-81.12607283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	09:03:02	1.0023295555555556	-79.83907283333334	80.00
Diego 1	03/22/11 14:15:15	2011-12-29	16:15:02	1.1319295555555556	-81.13507283333334	80.00

Ilustración 2.9: Histórico de velocidades. Fuente propia.

○ Pruebas de aceptación

Finalizada la iteración se procedió a realizar las pruebas de aceptación correspondientes, las cuales se muestran en las siguientes tablas.

Tabla 2.23. Caso de prueba Nro. 1 de la historia Nro. 4. Fuente propia.

CASO DE PRUEBA	
Número: 1	Número de historia: 4
Nombre: Generación correcta del reporte de velocidades.	
<p>Descripción:</p> <p>Todos los datos correspondientes a las velocidades promedio que el vehículo ha registrado en el intervalo de tiempo determinado (menor a un mes) deben mostrarse correctamente en la aplicación.</p>	
Condiciones de ejecución:	

La simulación de la base de datos debe ser correcta, para que genere los datos de las posiciones aproximados a la realidad, es decir, las velocidades promedio deben coincidir con la realidad.

Entradas:

Los datos de velocidad promedio generados ficticiamente por los procedimientos almacenados de base de datos (“store procedures”).

Resultado esperado: Resultado satisfactorio.

Evaluación: 10/10

○ Incidencias

- No se experimentó cambios considerables en el plan de entregas inicial.
- Al igual que las anteriores iteraciones, se hace imprescindible postergar hacia una nueva iteración la migración de la plataforma de programación de servidor, de PHP a Java.

● Iteración quinta

Esta iteración consta de la historia de usuario: Visualización del vehículo en tiempo real.

○ Asignación de tareas

Al igual que las anteriores iteraciones se procedió a planificar y asignar las tareas respectivas para la historia de usuario correspondiente. Las siguientes tablas muestran las tareas asignadas a la historia de usuario para esta iteración:

Tabla 2.24. Tarea Nro. 1 de la historia Nro. 5. Fuente propia.

TAREA	
Número: 1	Número de historia: 5
Nombre: Construcción del módulo de comunicaciones	
Tipo de tarea: Desarrollo	Puntos estimados: 1

Fecha de inicio: 06/03/2012	Fecha de fin: 10/03/2012
Descripción:	
Construcción del módulo de comunicaciones que permitirá el ingreso de tramas recibidas desde un AVL a la base de datos.	

○ Modificación del plan de entregas

En este punto se procede a agregar una historia de usuario que permita hacer la migración del lenguaje de programación de PHP a Java. Por lo tanto, el plan de entregas quedaría estipulado de la siguiente manera:

Iteración	Historia de usuario	Comienzo	Fin	Duración	dic 2011	ene 2012					feb 2012					mar 2012					abr 2012					may 2012				
						1/1	8/1	15/1	22/1	29/1	5/2	12/2	19/2	26/2	4/3	11/3	18/3	25/3	1/4	8/4	15/4	22/4	29/4	6/5	13/5	20/5				
1	Gestión administrativa del sistema	19/12/2011	12/01/2012	19d	[Barra azul]																									
2	Visualización del vehículo en tiempo ficticio	13/01/2012	09/02/2012	20d	[Barra azul]																									
3	Visualización organizada de los datos del vehículo en tiempo ficticio	10/02/2012	29/02/2012	14d	[Barra azul]																									
4	Histórico de un mes de posiciones y velocidades	01/03/2012	05/04/2012	26d	[Barra azul]																									
5	Visualización del vehículo en tiempo real	06/04/2012	25/04/2012	14d	[Barra azul]																									
6	Migración de PHP a Java	26/04/2012	31/05/2012	26d	[Barra azul]																									

Ilustración 2.10: Modificación del plan de entregas en la iteración quinta. Fuente propia.

○ Pruebas de aceptación

Finalizada la iteración se procedió a realizar las pruebas de aceptación correspondientes, las cuales se muestran en las siguientes tablas.

Tabla 2.25. Caso de prueba Nro. 1 de la historia Nro. 5. Fuente propia.

CASO DE PRUEBA	
Número: 1	Número de historia: 5
Nombre: Recepción correcta de la trama.	
Descripción:	
La trama debe ser recibida correctamente por el módulo de comunicaciones.	
Condiciones de ejecución:	

El AVL, la red celular (GPRS), el servidor deben de estar en excelentes condiciones.
Entradas: Tramas enviadas por el AVL.
Resultado esperado: Resultado satisfactorio.
Evaluación: 10/10

Tabla 2.26. Caso de prueba Nro. 2 de la historia Nro. 5. Fuente propia.

CASO DE PRUEBA	
Número: 2	Número de historia: 5
Nombre: Inserción correcta de la trama.	
Descripción: La trama debe ser insertada correctamente en la base de datos por el módulo de comunicaciones.	
Condiciones de ejecución: El AVL, la red celular (GPRS), el servidor deben de estar en excelentes condiciones.	
Entradas: Tramas enviadas por el AVL.	
Resultado esperado: Resultado satisfactorio.	
Evaluación: 10/10	

○ Incidencias

- No se experimentó cambios considerables en el plan de entregas inicial.
- Al igual que las anteriores iteraciones, se hace imprescindible postergar hacia una nueva iteración la migración de la plataforma de programación de servidor, de PHP a Java.

- Iteración sexta

Esta iteración consta de la historia de usuario: Migración de PHP a Java.

- Asignación de tareas

Al igual que las anteriores iteraciones se procedió a planificar y asignar las tareas respectivas para la historia de usuario correspondiente. Las siguientes tablas muestran las tareas asignadas a la historia de usuario para esta iteración:

Tabla 2.27. Tarea Nro. 1 de la historia Nro. 6. Fuente propia.

TAREA	
Número: 1	Número de historia: 6
Nombre: Modificación del código fuente de PHP a Java	
Tipo de tarea: Desarrollo	Puntos estimados: 1
Fecha de inicio: 26/04/2012	Fecha de fin: 31/05/2012
Descripción: Migración de todo el código fuente escrito en PHP a código fuente escrito en Java pero con las mismas funcionalidades.	

- Modificación del plan de entregas

No se registraron cambios en el plan de entregas inicial.

- Pruebas de aceptación

Finalizada la iteración se procedió a realizar las pruebas de aceptación correspondientes, las cuales se muestran en las siguientes tablas.

Tabla 2.28. Caso de prueba Nro. 1 de la historia Nro. 6. Fuente propia.

CASO DE PRUEBA	
Número: 1	Número de historia: 6
Nombre: Evidenciar el mismo funcionamiento con las dos.	

<p>Descripción:</p> <p>Evidenciar el mismo funcionamiento que se obtuvo con la plataforma PHP que actualmente con Java.</p>
<p>Condiciones de ejecución:</p> <p>Los recursos tecnológicos no deben experimentar cambios en su funcionalidad.</p>
<p>Entradas:</p> <p>Los mismos datos que se ingresaría con la plataforma PHP.</p>
<p>Resultado esperado: Resultado satisfactorio.</p>
<p>Evaluación: 10/10</p>

- Incidencias

- No se experimentó cambios considerables en el plan de entregas inicial.

2.4. Entrega

2.4.1. Implementación de la entrega final

En esta sección se detalla el proceso realizado para la implementación de la aplicación en el VPS (Virtual Private Server); para lo cual se arrendó este servicio a un proveedor adicionando la pre-instalación del sistema operativo CentOS 5.8.

- Compilación de postgresql

- Previa a la compilación

Previa la selección del motor de base de datos PostgreSQL, se procede a instalarlo; para lo cual es necesario descargar el código fuente para la compilación desde su sitio web: <http://www.postgresql.org> se realizan los siguientes pasos:

- Bajar y descomprimir el archivo postgresql-8.2.1.tar.gz en el directorio /usr/local con el siguiente comando:

```
[root@localhost ~]# tar -xzf postgresql-8.2.1.tar.gz -C /usr/local/
```

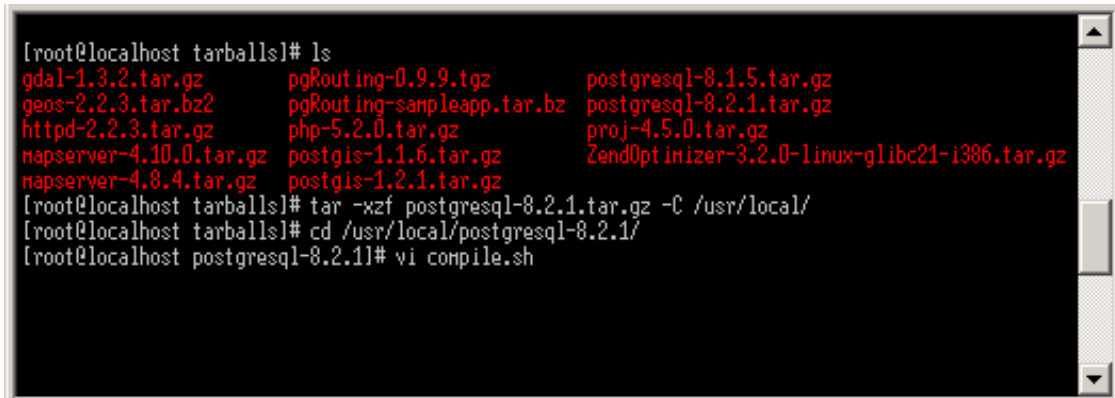
- Cambiar al directorio /usr/local/postgresql-8.2.1:


```
[root@localhost postgresql-8.2.1]# cd /usr/local/postgresql-8.2.1
```

- Crear el archivo `compile.sh`:

```
[root@localhost postgresql-8.2.1]# vi compile.sh
```

Estos pasos se muestran en la Ilustración 2.11.



```
[root@localhost tarballs]# ls
gdal-1.3.2.tar.gz      pgRouting-0.9.9.tgz      postgresql-8.1.5.tar.gz
geos-2.2.3.tar.bz2    pgRouting-sampleapp.tar.bz2 postgresql-8.2.1.tar.gz
httpd-2.2.3.tar.gz     php-5.2.0.tar.gz         proj-4.5.0.tar.gz
mapserver-4.10.0.tar.gz postgis-1.1.6.tar.gz     ZendOptimizer-3.2.0-linux-glibc21-i386.tar.gz
mapserver-4.8.4.tar.gz postgis-1.2.1.tar.gz
[root@localhost tarballs]# tar -xzf postgresql-8.2.1.tar.gz -C /usr/local/
[root@localhost tarballs]# cd /usr/local/postgresql-8.2.1/
[root@localhost postgresql-8.2.1]# vi compile.sh
```

Ilustración 2.11: Creación del archivo `compile.sh`. Fuente propia.

- Copiar y pegar la secuencia de instrucciones que se muestran en el siguiente script:

```
LDFLAGS=-lstdc++ ./configure \
--prefix=/usr/local/pgsql \
--with-perl \
--with-python \
--with-krb5 \
--with-openssl
```

- Hacer ejecutable el script de compilación:

```
[root@localhost postgresql-8.2.1]# chmod 755 compile.sh
```

- Para configurar la compilación de PostgreSQL se ejecuta el script de compilación:

```
[root@localhost postgresql-8.2.1]# ./compile.sh
```

Aparece una secuencia de texto, la cual se muestra en la Ilustración 2.12.

- o Compilación

- Ejecutar `make`:

```
[root@localhost postgresql-8.2.1]# make
```

El proceso entrega una secuencia de texto que se muestra en la Ilustración 2.13.

```

[root@localhost postgresql-8.2.11# ls
aclocal.m4  config.log  configure.in  doc          HISTORY  README
compile.sh  config.status  contrib      GNUmakefile  INSTALL  src
config      configure     COPYRIGHT    GNUmakefile.in  Makefile
[root@localhost postgresql-8.2.11# ./compile.sh
checking build system type... i686-pc-linux-gnu
checking host system type... i686-pc-linux-gnu
checking which template to use... linux
checking whether to build with 64-bit integer date/time support... no
checking whether MLS is wanted... no
checking for default port number... 5432
checking for gcc... gcc

●
●
●

config.status: creating GNUmakefile
config.status: creating src/Makefile.global
config.status: creating src/include/pg_config.h
config.status: src/include/pg_config.h is unchanged
config.status: creating src/interfaces/ecpg/include/ecpg_config.h
config.status: src/interfaces/ecpg/include/ecpg_config.h is unchanged
config.status: linking ./src/backend/port/tas/dummy.s to src/backend/port/tas.s
config.status: linking ./src/backend/port/dynloader/linux.c to src/backend/port/dynloader.c
config.status: linking ./src/backend/port/sysv_sena.c to src/backend/port/pg_sena.c
config.status: linking ./src/backend/port/sysv_shmen.c to src/backend/port/pg_shmen.c
config.status: linking ./src/backend/port/dynloader/linux.h to src/include/dynloader.h
config.status: linking ./src/include/port/linux.h to src/include/pg_config_os.h
config.status: linking ./src/Makefiles/Makefile.linux to src/Makefile.port

```

Ilustración 2.12: Resultado de ejecutar el archivo compile.sh. Fuente propia.

```

[root@localhost postgresql-8.2.11# make
make -C doc all
make[1]: Entering directory '/usr/local/postgresql-8.2.1/doc'
gzip -d -c man.tar.gz | /bin/tar xf -
for file in man1/*.*; do \
  mv $file $file.bak && \
  sed -e 's/\\fr(1)/\\fr(7)/' $file.bak >$file && \
  rm -f $file.bak || exit; \
done
/bin/sh ../config/mkinstalldirs man7
mkdir -p -- man7
for file in man1/*.*; do \
  sed -e '/^\\.TH/s/"1"/"7"/' \
    -e 's/\\fr(1)/\\fr(7)/' \
  $file >man7/$(basename $file) | sed 's/\\.1$/7/' || exit; \
done

gcc -shared -o autoinc.so autoinc.o
rm refint.o autoinc.o
make[3]: Leaving directory '/usr/local/postgresql-8.2.1/contrib/spi'
make[2]: Leaving directory '/usr/local/postgresql-8.2.1/src/test/regress'
make[1]: Leaving directory '/usr/local/postgresql-8.2.1/src'
make -C config all
make[1]: Entering directory '/usr/local/postgresql-8.2.1/config'
make[1]: No se hace nada para 'all'.
make[1]: Leaving directory '/usr/local/postgresql-8.2.1/config'
All of PostgreSQL successfully made. Ready to install.
[root@localhost postgresql-8.2.11#

```

Ilustración 2.13: Listado que se genera al ejecutar “make”. Fuente propia.

- Ejecutar el comando `make install`:

```
[root@localhost postgresql-8.2.1]# make install
```

Esta sentencia entrega la secuencia ilustrada en la Ilustración 2.14.

```
[root@localhost postgresql-8.2.1]# make install
make -C doc install
make[1]: Entering directory '/usr/local/postgresql-8.2.1/doc'
mkdir -p -- /usr/local/pgsql/doc/html
mkdir -p -- /usr/local/pgsql/man/man1 /usr/local/pgsql/man/man7
gzip -d -c ./postgres.tar.gz | ( cd /usr/local/pgsql/doc/html && /bin/tar xf - )
for file in man1/*.* man7/*.* ; do \
  /bin/sh ../config/install-sh -c -m 644 $file /usr/local/pgsql/man/$file || exit; \
done
make[1]: Leaving directory '/usr/local/postgresql-8.2.1/doc'
make -C src install
make[1]: Entering directory '/usr/local/postgresql-8.2.1/src'
/bin/sh ../config/mkinstalldirs '/usr/local/pgsql/lib/pgxs/src'

make -C port install
make[2]: Leaving directory '/usr/local/postgresql-8.2.1/src/test/regress'
make[1]: Leaving directory '/usr/local/postgresql-8.2.1/src'
make -C config install
make[1]: Entering directory '/usr/local/postgresql-8.2.1/config'
mkdir -p -- /usr/local/pgsql/lib/pgxs/config
/bin/sh ../config/install-sh -c -m 755 ./install-sh '/usr/local/pgsql/lib/pgxs/config/install-sh'
/bin/sh ../config/install-sh -c -m 755 ./mkinstalldirs '/usr/local/pgsql/lib/pgxs/config/mkinstalldirs'
make[1]: Leaving directory '/usr/local/postgresql-8.2.1/config'
PostgreSQL installation complete.
[root@localhost postgresql-8.2.1]#
```

Ilustración 2.14: Listado que se genera al ejecutar “make install”. Fuente propia.

- o Post-compilación y configuración

Una vez que los pasos señalados en los acápites anteriores se ejecutan sin problemas, la base de datos debe ser configurada con un usuario propietario, en este caso se crea el usuario *postgres*.

- Para cumplir con esta acción debe ejecutarse la siguiente secuencia de instrucciones:

```
[root@localhost ~]# /usr/sbin/adduser postgres
```

```
[root@localhost ~]# mkdir /usr/local/pgsql/data
```

```
[root@localhost ~]# chown postgres /usr/local/pgsql/data/
```

```
[root@localhost ~]# su - postgres
[postgres@localhost ~]# /usr/local/pgsql/bin/initdb -D
/usr/local/pgsql/data/
```

La secuencia de instrucciones y su resultado satisfactorio se muestran en la Ilustración 2.15.

```
[root@localhost postgresql-8.2.1]# /usr/sbin/adduser postgres
[root@localhost postgresql-8.2.1]# mkdir /usr/local/pgsql/data
[root@localhost postgresql-8.2.1]# chown postgres /usr/local/pgsql/data/
[root@localhost postgresql-8.2.1]# su - postgres
[postgres@localhost ~]# /usr/local/pgsql/bin/initdb -D /usr/local/pgsql/data/
The files belonging to this database system will be owned by user "postgres".
This user must also own the server process.

The database cluster will be initialized with locale es_CL.UTF-8.
The default database encoding has accordingly been set to UTF8.

fixing permissions on existing directory /usr/local/pgsql/data ... ok
creating subdirectories ... ok
selecting default max_connections ... 100
selecting default shared_buffers/max_fsm_pages ... 24MB/153600
creating configuration files ... ok
creating template1 database in /usr/local/pgsql/data/base/1 ... ok
initializing pg_authid ... ok
initializing dependencies ... ok
creating system views ... ok
loading system objects' descriptions ... ok
creating conversions ... ok
setting privileges on built-in objects ... ok
creating information schema ... ok
vacuuming database template1 ... ok
copying template1 to template0 ... ok
copying template1 to postgres ... ok

WARNING: enabling "trust" authentication for local connections
You can change this by editing pg_hba.conf or using the -A option the
next time you run initdb.

Success. You can now start the database server using:

    /usr/local/pgsql/bin/postgres -D /usr/local/pgsql/data
or
    /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data -l logfile start

[postgres@localhost ~]#
```

Ilustración 2.15: Listado que se genera al inicializar el motor de base de datos. Fuente propia.

○ Pruebas de inicio y funcionamiento

- Las pruebas se llevan a cabo con la siguiente secuencia de comandos:

```
[postgres@localhost ~]# /usr/local/pgsql/bin/pg_ctl -D
/usr/local/pgsql/data/ -l /usr/local/pgsql/data/logfile start
```

El resultado se muestra en la Ilustración 2.16.

```
[postgres@localhost ~]$ /usr/local/pgsql/bin/pg_ctl -D /usr/local/pgsql/data/ -l /usr/local/pgsql/data/logfile start
server starting
[postgres@localhost ~]$
```

Ilustración 2.16: Listado que se genera al iniciar el motor de base de datos. Fuente propia.

```
[postgres@localhost ~]# /usr/local/pgsql/bin/createdb test
```

Su resultado se muestra en la Ilustración 2.17.

```
[postgres@localhost ~]$ /usr/local/pgsql/bin/createdb test
CREATE DATABASE
[postgres@localhost ~]$
```

Ilustración 2.17: Listado que se genera al crear una base de datos. Fuente propia.

```
[postgres@localhost ~]# /usr/local/pgsql/bin/psql test
```

Su resultado se muestra en la Ilustración 2.18.

```
[postgres@localhost ~]$ /usr/local/pgsql/bin/psql test
Welcome to psql 8.2.1, the PostgreSQL interactive terminal.

Type: \copyright for distribution terms
      \h for help with SQL commands
      \? for help with psql commands
      \g or terminate with semicolon to execute query
      \q to quit

test=# \q
[postgres@localhost ~]$
```

Ilustración 2.18: Listado que se genera al ejecutar el comando para manejar una base de datos. Fuente propia.

- Configuración del arranque automático

Para hacer que el servicio de PostgreSQL se ejecute automáticamente al iniciar el Sistema Operativo, es necesario crear un script de inicio y control, como se muestra a continuación.

- Como *root*, se crea el archivo `/etc/init.d/postgresql` para lo cual se ejecuta el siguiente comando:

```
[root@localhost postgresql-8.2.1]# vi /etc/init.d/postgresql
```

- En este punto se copia y pega el texto del siguiente script:

```

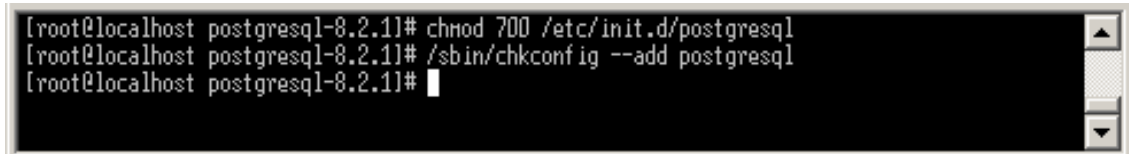
#!/bin/sh
# postgresql This is the init script for starting up the
# PostgreSQL server
# chkconfig: - 85 15
# description: Starts and stops the PostgreSQL backend daemon that
handles all database requests.
# processname: postmaster
# pidfile: /usr/local/pgsql/data/postmaster.pid
#
# Source function library.
. /etc/rc.d/init.d/functions
# Get config.
. /etc/sysconfig/network
# Check that networking is up.
# Pretty much need it for postmaster.
[ ${NETWORKING} = "no" ] && exit 0
[ -f /usr/local/pgsql/bin/postmaster ] || exit 0
# See how we were called.
case "$1" in
start)
pid=`pidof postmaster`
if [ $pid ]
then
echo "Postmaster already running."
else
echo -n "Starting postgresql service: "
su -l postgres -c '/usr/local/pgsql/bin/pg_ctl -D
/usr/local/pgsql/data/ -l /usr/local/pgsql/data/logfile start'
sleep 1
echo
exit
fi
;;
stop)
echo -n "Stopping postgresql service: "
killproc postmaster
sleep 2
rm -f /usr/local/pgsql/data/postmaster.pid
echo
;;
restart)
$0 stop
$0 start
;;
*)
echo "Usage: postgresql {start|stop|restart}"
exit 1
esac
exit 0

```

- Luego se hace el script ejecutable y se lo agrega como servicio del sistema con las siguientes instrucciones:

```
[root@localhost postgresql-8.2.1]# chmod 700 /etc/init.d/postgresql
[root@localhost postgresql-8.2.1]# /sbin/chkconfig --add postgresql
```

Tal y como se muestra en la Ilustración 2.19.



```
[root@localhost postgresql-8.2.1]# chmod 700 /etc/init.d/postgresql
[root@localhost postgresql-8.2.1]# /sbin/chkconfig --add postgresql
[root@localhost postgresql-8.2.1]#
```

Ilustración 2.19: Comandos para agregar como servicio al motor de base de datos. Fuente propia.

- Compilación de PostGIS
- Previa a la compilación
 - *Instalación de Proj4*

Para la instalación de Proj4, es necesario descarga su código fuente de Proj4 desde la dirección web <http://www.remotesensing.org/proj/>, si aún no está disponible.

- Descomprimir el archivo proj-4.5.0.tar.gz.

```
[root@localhost ~]# tar -xzf proj-4.5.0.tar.gz -C /usr/local
```

- Cambiar de directorio a /usr/local/proj-4.5.0.

```
[root@localhost ~]# cd /usr/local/proj-4.5.0
```

- Ejecutar el script de configuración de la compilación.

```
[root@localhost proj-4.5.0]# ./configure
```

Las secuencias de texto de este proceso se muestran en la Ilustración 2.20.

- Ejecutar `make` y `make install`.

```
[root@localhost proj-4.5.0]# make
```

```
[root@localhost proj-4.5.0]# make install
```

Las secuencias de texto se muestran en la Ilustración 2.21 y la Ilustración 2.22.

```

aclocal.m4  config.sub  depcomp  ltconfig  man  NEWS
AUTHORS    configure  INSTALL  ltmain.sh  missing  README
ChangeLog  configure.in  install-sh  Makefile.am  mksinstalldirs  src
config.guess  COPYING  jniurap  Makefile.in  nad

[root@localhost proj-4.5.0]# ./configure
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gauk... gauk
.
.
.

config.status: creating man/Makefile
config.status: creating man/man1/Makefile
config.status: creating man/man3/Makefile
config.status: creating nad/Makefile
config.status: creating jniurap/Makefile
config.status: creating jniurap/org/Makefile
config.status: creating jniurap/org/proj4/Makefile
config.status: creating src/proj_config.h
config.status: executing depfiles commands

```

Ilustración 2.20: Listado que se genera al ejecutar el script “configure” de proj. Fuente propia.

```

[root@localhost proj-4.5.0]# make
Making all in src
make[1]: Entering directory '/usr/local/proj-4.5.0/src'
make all-an
make[2]: Entering directory '/usr/local/proj-4.5.0/src'
if /bin/sh ../libtool --tag=CC --mode=compile gcc -DHAVE_CONFIG_H -I. -I. -I. -DPROJ_LIB=\"/usr/local/share/proj\" -g -O2 -MT PJ_aeqd.lo -MD -MP -MF ".deps/PJ_aeqd.Tpo" -c -o PJ_aeqd.lo PJ_aeqd.c; \
then mv -f ".deps/PJ_aeqd.Tpo" ".deps/PJ_aeqd.Plo"; else rm -f ".deps/PJ_aeqd.Tpo"; exit 1;
.
.
.

make[3]: No se hace nada para 'all-an'.
make[3]: Leaving directory '/usr/local/proj-4.5.0/jniurap/org'
make[2]: Leaving directory '/usr/local/proj-4.5.0/jniurap/org'
make[2]: Entering directory '/usr/local/proj-4.5.0/jniurap'
make[2]: No se hace nada para 'all-an'.
make[2]: Leaving directory '/usr/local/proj-4.5.0/jniurap'
make[1]: Leaving directory '/usr/local/proj-4.5.0/jniurap'
make[1]: Entering directory '/usr/local/proj-4.5.0'
make[1]: No se hace nada para 'all-an'.
make[1]: Leaving directory '/usr/local/proj-4.5.0'

```

Ilustración 2.21: Listado que se genera al ejecutar “make” de proj. Fuente propia.


```
[root@localhost proj-4.5.0]# make install
Making install in src
make[1]: Entering directory '/usr/local/proj-4.5.0/src'
make[2]: Entering directory '/usr/local/proj-4.5.0/src'
test -z "/usr/local/lib" || mkdir -p -- "/usr/local/lib"
/bin/sh ../libtool --mode=install /usr/bin/install -c 'libproj.la' '/usr/local/lib/libproj.la'
/usr/bin/install -c .libs/libproj.so.0.5.2 /usr/local/lib/libproj.so.0.5.2
*
*
*
In jniwrap
make[3]: Leaving directory '/usr/local/proj-4.5.0/jniwrap'
make[2]: Leaving directory '/usr/local/proj-4.5.0/jniwrap'
make[1]: Leaving directory '/usr/local/proj-4.5.0/jniwrap'
make[1]: Entering directory '/usr/local/proj-4.5.0'
make[2]: Entering directory '/usr/local/proj-4.5.0'
make[2]: No se hace nada para 'install-exec-am'.
make[2]: No se hace nada para 'install-data-am'.
make[2]: Leaving directory '/usr/local/proj-4.5.0'
make[1]: Leaving directory '/usr/local/proj-4.5.0'
[root@localhost proj-4.5.0]#
```

Ilustración 2.22: Listado que se genera al ejecutar “make install” de proj. Fuente propia.

- *Instalación de geos*

Este proceso requiere realizar la descarga del código fuente de GEOS desde la dirección web <http://geos.refractions.net/>.

- Descomprimir el archivo geos-2.2.3.tar.bz2.

```
[root@localhost ~]# tar -xjf geos-2.2.3.tar.bz2 -C /usr/local
```

- Cambiar de directorio a /usr/local/geos-2.2.3.

```
[root@localhost ~]# cd /usr/local/ geos-2.2.3
```

- Ejecutar el script de configuración de la compilación.

```
[root@localhost geos-2.2.3]# ./configure
```

Las secuencias de texto de este proceso se muestran en la Ilustración 2.23.

- Luego se ejecuta las instrucciones make y make install.

```
[root@localhost proj-4.5.0]# make
```

```
[root@localhost proj-4.5.0]# make install
```

Las secuencias de texto se muestran en la Ilustración 2.24 y la Ilustración 2.25.

```
[root@localhost local1]# cd geos-2.2.3
[root@localhost geos-2.2.3]# ./configure
checking build system type... i686-redhat-linux-gnu
checking host system type... i686-redhat-linux-gnu
checking target system type... i686-redhat-linux-gnu
checking for a BSD-compatible install... /usr/bin/install -c
checking whether build environment is sane... yes
checking for gauk... gauk
checking whether make sets $(MAKE)... yes

*
*
*

config.status: creating swig/python/tests/cases/Makefile
config.status: creating swig/ruby/Makefile
config.status: creating swig/ruby/test/Makefile
config.status: creating VisualStudio/Makefile
config.status: creating source/capi/geos_c.h
config.status: creating source/headers/config.h
config.status: creating source/headers/geos/platform.h
config.status: executing depfiles commands
[root@localhost geos-2.2.3]#
```

Ilustración 2.23: Listado que se genera al ejecutar el script “configure” de geos. Fuente propia.

```
[root@localhost proj-4.5.0]# make
Making all in src
make[1]: Entering directory '/usr/local/proj-4.5.0/src'
make all-am
make[2]: Entering directory '/usr/local/proj-4.5.0/src'
if /bin/sh ../libtool --tag=CC --mode=compile gcc -DHAVE_CONFIG_H -I. -I. -I. -DPROJ_LIB=\"/usr/local/share/proj\" -g -O2 -MT PJ_aeqd.lo -MD -MP -MF ".deps/PJ_aeqd.Tpo" -c -o PJ_aeqd.lo PJ_aeqd.c; \
then mv -f ".deps/PJ_aeqd.Tpo" ".deps/PJ_aeqd.Plo"; else rm -f ".deps/PJ_aeqd.Tpo"; exit 1;
*
*
*

make[1]: Leaving directory '/usr/local/geos-2.2.3/doc'
Making all in VisualStudio
make[1]: Entering directory '/usr/local/geos-2.2.3/VisualStudio'
make[1]: No se hace nada para 'all'.
make[1]: Leaving directory '/usr/local/geos-2.2.3/VisualStudio'
make[1]: Entering directory '/usr/local/geos-2.2.3'
make[1]: No se hace nada para 'all-am'.
make[1]: Leaving directory '/usr/local/geos-2.2.3'
[root@localhost geos-2.2.3]#
```

Ilustración 2.24: Listado que se genera al ejecutar “make” de geos. Fuente propia.

```
[root@localhost geos-2.2.3]# make install
Making install in source
make[1]: Entering directory '/usr/local/geos-2.2.3/source'
Making install in geon
make[2]: Entering directory '/usr/local/geos-2.2.3/source/geon'
make[3]: Entering directory '/usr/local/geos-2.2.3/source/geon'
test -z "/usr/local/lib" || mkdir -p -- "/usr/local/lib"
/bin/sh ../../libtool --mode=install /usr/bin/install -c 'libgeos.la' '/usr/local/lib/libgeos.la'
*
*
*
make[2]: No se hace nada para 'install-data-am'.
make[2]: Leaving directory '/usr/local/geos-2.2.3/VisualStudio'
make[1]: Leaving directory '/usr/local/geos-2.2.3/VisualStudio'
make[1]: Entering directory '/usr/local/geos-2.2.3'
make[2]: Entering directory '/usr/local/geos-2.2.3'
make[2]: No se hace nada para 'install-exec-am'.
make[2]: No se hace nada para 'install-data-am'.
make[2]: Leaving directory '/usr/local/geos-2.2.3'
make[1]: Leaving directory '/usr/local/geos-2.2.3'
[root@localhost geos-2.2.3]#
```

Ilustración 2.25: Listado que se genera al ejecutar “make install” de geos. Fuente propia.

○ Compilación

Para realizar la compilación, se procede a descargar el código fuente de PostGIS desde el sitio web <http://www.postgis.org>.

- Descomprimir el archivo `postgis-1.2.1.tar.gz` dentro del directorio `contrib` de `postgres`.

```
[root@localhost ~]# tar -xzf postgis-1.2.1.tar.gz -C postgresql-8.2.1/contrib/
```

- Cambiar de directorio a `/usr/local/postgresql-8.2.1/contrib/postgis-1.2.1/`.

```
[root@localhost ~]# cd /usr/local/postgresql-8.2.1/contrib/postgis-1.2.1/
```

- Ejecutar el script de configuración de la compilación con parámetros.

```
[root@localhost postgis-1.2.1]# ./configure --withpgsql=/usr/local/pgsql/bin/pg_config
```

Las secuencias de texto en la consola se aprecian en la Ilustración 2.26.

```

[root@localhost postgres-1.2.1]# ./configure --with-pgsql=/usr/local/pgsql/bin/pg_config
checking for gcc... gcc
checking for C compiler default output file name... a.out
checking whether the C compiler works... yes
checking whether we are cross compiling... no
checking for suffix of executables...
checking for suffix of object files... o
checking whether we are using the GNU C compiler... yes
*
*
*
PORTNAME: linux
PREFIX: /usr/local/pgsql
EPREFIX: ${prefix}
DOC: /usr/local/pgsql/doc/contrib
DATA: ${datarootdir}
MAN: ${datarootdir}/man
BIN: /usr/local/pgsql/bin
EXT: /usr/local/pgsql/lib (\${libdir})
-----
[root@localhost postgres-1.2.1]#

```

Ilustración 2.26: Listado que se genera al ejecutar el script “configure” de postgres. Fuente propia.

- Ejecutarmake y make install.

```

[root@localhost postgres-1.2.1]# make
[root@localhost postgres-1.2.1]# make install

```

Las correspondientes secuencias de texto se muestran en la Ilustración 2.27 y la Ilustración 2.28.

```

[root@localhost postgres-1.2.1]# make
make -C lugeon
make[1]: Entering directory '/usr/local/postgresql-8.2.1/contrib/postgis-1.2.1/lugeon'
cpp -P -traditional-cpp -DUSE_VERSION=82 lpogstgis.sql.in | sed -e 's:@MODULE_FILENAME@:\$libdir/liblugeon.so.1.2:g;s:@POSTGIS_VERSION@:1.2 USE_GEOS=1 USE_PROJ=1 USE_STATS=1:g;s:@POSTGIS_SCRIPTS_VERSION@:1.2.1:g;s:@POSTGIS_BUILD_DATE@/2007-02-13 07:38:23/g' | grep -v '^#' > ./lpogstgis.sql
*
*
*
h.in | sed -e s:@bindir@:/usr/local/pgsql/bin:g -e s:@datadir@:/usr/local/pgsql/share:g -e s:@prefix@:/usr/local/pgsql:g -e s:@SONAME@:1:g > postgres_lib.sh; cat postgres_lib.sh.in | sed -e s:@bindir@:/usr/local/pgsql/bin:g -e s:@datadir@:/usr/local/pgsql/share:g -e s:@prefix@:/usr/local/pgsql:g -e s:@SONAME@:1:g > postgres_lib.sh; cat rntemplate_gis.in | sed -e s:@bindir@:/usr/local/pgsql/bin:g -e s:@datadir@:/usr/local/pgsql/share:g -e s:@prefix@:/usr/local/pgsql:g -e s:@SONAME@:1:g > rntemplate_gis; cat rntemplate_gis.sh.in | sed -e s:@bindir@:/usr/local/pgsql/bin:g -e s:@datadir@:/usr/local/pgsql/share:g -e s:@prefix@:/usr/local/pgsql:g -e s:@SONAME@:1:g > rntemplate_gis.sh;
make[1]: Leaving directory '/usr/local/postgresql-8.2.1/contrib/postgis-1.2.1/extras/template_gis'
[root@localhost postgres-1.2.1]#

```

Ilustración 2.27: Listado que se genera al ejecutar “make” de postgres. Fuente propia.

```

[root@localhost postgis-1.2.1]# make
make -C lugeon
make[1]: Entering directory '/usr/local/postgresql-8.2.1/contrib/postgis-1.2.1/lugeon'
cpp -P -traditional-cpp -DUSE_VERSION=82 lwpostgis.sql.in | sed -e 's:@MODULE_FILENAME@:\$1|
bdir/liblugeon.so.1.2:g;s:@POSTGIS_VERSION@:1.2 USE GEOS=1 USE PROJ=1 USE STATS=1:g;s:@POSTG
IS_SCRIPTS_VERSION@:1.2.1:g;s:@POSTGIS_BUILD_DATE@/2007-02-13 07:38:23/g' | grep -v '^#' > .
./lwpostgis.sql
.
.
.
h.in | sed -e s:@bindir@:/usr/local/pgsql/bin:g -e s:@datadir@:/usr/local/pgsql/share:g -e s
:@prefix@:/usr/local/pgsql:g -e s:@SONAME@:1:g > postgis_env.sh; cat postgres_lib.sh.in | s
ed -e s:@bindir@:/usr/local/pgsql/bin:g -e s:@datadir@:/usr/local/pgsql/share:g -e s:@prefix
@:/usr/local/pgsql:g -e s:@SONAME@:1:g > postgres_lib.sh; cat rntemplate_gis.in | sed -e s:
@bindir@:/usr/local/pgsql/bin:g -e s:@datadir@:/usr/local/pgsql/share:g -e s:@prefix@:/usr/l
ocal/pgsql:g -e s:@SONAME@:1:g > rntemplate_gis; cat rntemplate_gis.sh.in | sed -e s:@bindi
r@:/usr/local/pgsql/bin:g -e s:@datadir@:/usr/local/pgsql/share:g -e s:@prefix@:/usr/local/p
pgsql:g -e s:@SONAME@:1:g > rntemplate_gis.sh;
make[1]: Leaving directory '/usr/local/postgresql-8.2.1/contrib/postgis-1.2.1/extras/templat
e_gis'
[root@localhost postgis-1.2.1]#

```

Ilustración 2.28: Listado que se genera al ejecutar “make install” de postgis. Fuente propia.

o Configuración post-compilación

Una vez completados todos los pasos especificados en los puntos anteriores, es necesario crear una Base de Datos habilitada espacialmente, para lo cual se emplea la base de datos “test” creada en el proceso de configuración de PostgreSQL.

El procedimiento es el siguiente:

- Realizar la autenticación como usuario postgres, propietario del Motor de Base Datos.

```
[root@localhost postgis-1.2.1]# su - postgres
```

- Cargar el lenguaje de procedimientos almacenados plpgsql en la base de datos “test”.

```
[postgres@localhost postgis-1.2.1]# /usr/local/pgsql/bin/createlang
plpgsql test
```

- Cargar las funciones de PostGIS en la base de datos “test”.

```
[postgres@localhost postgis-1.2.1]# /usr/local/pgsql/bin/psql -d test
-f /usr/local/pgsql/share/lwpostgis.sql
```

Las secuencias de texto correspondientes se aprecian en la Ilustración 2.29.

```
[postgres@localhost ~]$ /usr/local/pgsql/bin/psql -d test -f /usr/local/pgsql/share/lwpostgis.sql
BEGIN
psql:/usr/local/pgsql/share/lwpostgis.sql:52: NOTICE: type "histogran2d" is not yet defined
DETAIL: Creating a shell type definition.
CREATE FUNCTION
psql:/usr/local/pgsql/share/lwpostgis.sql:57: NOTICE: argument type histogran2d is only a shell
hell
.
.
.

CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
COMMIT
[postgres@localhost ~]$
```

Ilustración 2.29: Listado que se genera al ejecutar el script "lwpostgis.sql". Fuente propia.

- Finalmente se ejecuta `exit` para volver al usuario `root`.

```
[postgres@localhost postgres-1.2.1]# exit
```

- Instalación de Apache Tomcat

En esta sección se detallan los pasos que se llevan a cabo para la instalación de Tomcat 7.0.23.

- Descarga y desempaquetado

Para este proceso se descarga el instalador de Apache Tomcat 7.0.23 llamado "apache-tomcat-7.0.23.tar.gz" desde el sitio web <http://tomcat.apache.org/download-70.cgi>.

- Se guarda el archivo en el directorio `/root/apache-tomcat-7.0.23.tar.gz`, luego se mueve el archivo al directorio `/usr/share`.

```
[root@srv6 ~]# mv apache-tomcat-7.0.23.tar.gz /usr/share/apache-tomcat-7.0.23.tar.gz
```

- Se hace el cambio de directorio a `/usr/share` y se desempaqueta el archivo usando la instrucción `tar -xzf`.

```
[root@srv6 ~]# cd /usr/share
```

```
[root@srv6 share]# tar -xzf apache-tomcat-7.0.23.tar.gz
```

Esto creará el directorio /usr/share/apache-tomcat-7.0.23.

○ Configuración para la ejecución como servicio

- Se hace el cambio de directorio a /etc/init.d y se crea un script llamado "tomcat".

```
[root@srv6 share]# cd /etc/init.d
```

```
[root@srv6 init.d]# vi tomcat
```

- Aquí se muestra el script que se utilizó.

```
#!/bin/bash
# description: Tomcat Start Stop Restart
# processname: tomcat
# chkconfig: 234 20 80
JAVA_HOME=/usr/java/jdk1.7.0_
export JAVA_HOME
PATH=$JAVA_HOME/bin:$
export
CATALINA_HOME=/usr/share/apache-tomcat-7.0.
case $1
start
sh $CATALINA_HOME/bin/startup.
stop
sh $CATALINA_HOME/bin/shutdown.
restart
sh $CATALINA_HOME/bin/shutdown.
sh $CATALINA_HOME/bin/startup.
exit 0
```

- Se cambian los permisos del script para hacerlo ejecutable.

```
[root@srv6 init.d]# chmod 755 tomcat
```

- Luego, se utiliza chkconfig para tener el inicio de Tomcat al tiempo de arranque.

```
[root@srv6 init.d]# chkconfig --add tomcat
```

```
[root@srv6 init.d]# chkconfig --level 234 tomcat on
```

- Se prueba el correcto funcionamiento del script.
 - Iniciando Tomcat

```
[root@srv6 ~]# service tomcat start
```

```
Using CATALINA_BASE:   /usr/share/apache-tomcat-7.0.23
```

```
Using CATALINA_HOME:   /usr/share/apache-tomcat-7.0.23
```

```
Using CATALINA_TMPDIR: /usr/share/apache-tomcat-7.0.23/temp
```

```
Using JRE_HOME:          /usr/java/jdk1.7.0_02
Using CLASSPATH:         /usr/share/apache-tomcat-
7.0.23/bin/bootstrap.jar:/usr/share/apache-tomcat-7.0.23/bin/tomcat-
juli.jar
```

- Deteniendo Tomcat

```
[root@srv6 ~]#service tomcat stop
Using CATALINA_BASE:     /usr/share/apache-tomcat-7.0.23
Using CATALINA_HOME:     /usr/share/apache-tomcat-7.0.23
Using CATALINA_TMPDIR:   /usr/share/apache-tomcat-7.0.23/temp
Using JRE_HOME:          /usr/java/jdk1.7.0_02
Using CLASSPATH:         /usr/share/apache-tomcat-
7.0.23/bin/bootstrap.jar: /usr/share/apache-tomcat-
7.0.23/bin/tomcat-juli.jar
```

- Reiniciando Tomcat (debe estar iniciado primero)

```
[root@srv6 ~]#service tomcat restart
Using CATALINA_BASE:     /usr/share/apache-tomcat-7.0.23
Using CATALINA_HOME:     /usr/share/apache-tomcat-7.0.23
Using CATALINA_TMPDIR:   /usr/share/apache-tomcat-7.0.23/temp
Using JRE_HOME:          /usr/java/jdk1.7.0_02
Using CLASSPATH:         /usr/share/apache-tomcat-
7.0.23/bin/bootstrap.jar:/usr/share/apache-tomcat-7.0.23/bin/tomcat-juli.jar
Using CATALINA_BASE:     /usr/share/apache-tomcat-7.0.23
Using CATALINA_HOME:     /usr/share/apache-tomcat-7.0.23
Using CATALINA_TMPDIR:   /usr/share/apache-tomcat-7.0.23/temp
Using JRE_HOME:          /usr/java/jdk1.7.0_02
Using CLASSPATH:         /usr/share/apache-tomcat-
7.0.23/bin/bootstrap.jar:/usr/share/apache-tomcat-7.0.23/bin/tomcat-juli.jar
```


- Creación de la base de datos

Una vez instalado el motor de base de datos PostgreSQL con su extensión PostGIS, se procede a crear la base de datos de la aplicación mediante el script de base de datos que se encuentra en el Anexo II.

2.4.2. Manuales

En esta sección del documento se describirán los manuales que explican el cómo los usuarios deben usar el sistema de manera correcta. El sistema maneja dos tipos de usuarios que son los siguientes:

- **ADMINISTRADOR:** Es el que se encarga de la administración del sistema, es decir se encarga de crear AVLS en el sistema, como también crear usuarios finales, mas no usuarios de tipo Administrador.
- **USUARIO FINAL:** Es el que obtiene todas los beneficios funcionales del sistema, el sistema fue creado pensando en este usuario.

Por lo tanto, estos manuales servirán de guía para los dos tipos de usuarios, para que ambos obtengan las capacidades necesarias para manejar el sistema completamente.

- Manual del administrador

En este manual se detallarán los pasos que debe seguir un usuario Administrador para manejar adecuadamente el sistema. A continuación se explicará paso a paso las acciones que se tiene que realizar para explotar a fondo todas sus funcionalidades.

- Ingreso al sistema

Para ingresar al sistema se deben tener en cuenta algunos aspectos que a continuación se detallan.

- *Cómo acceder al sistema webgeo-referenciado*

- 1) Después de encender el computador, ubíquese en el ícono de Mozilla Firefox



y haga doble click (ver Ilustración 2.30).

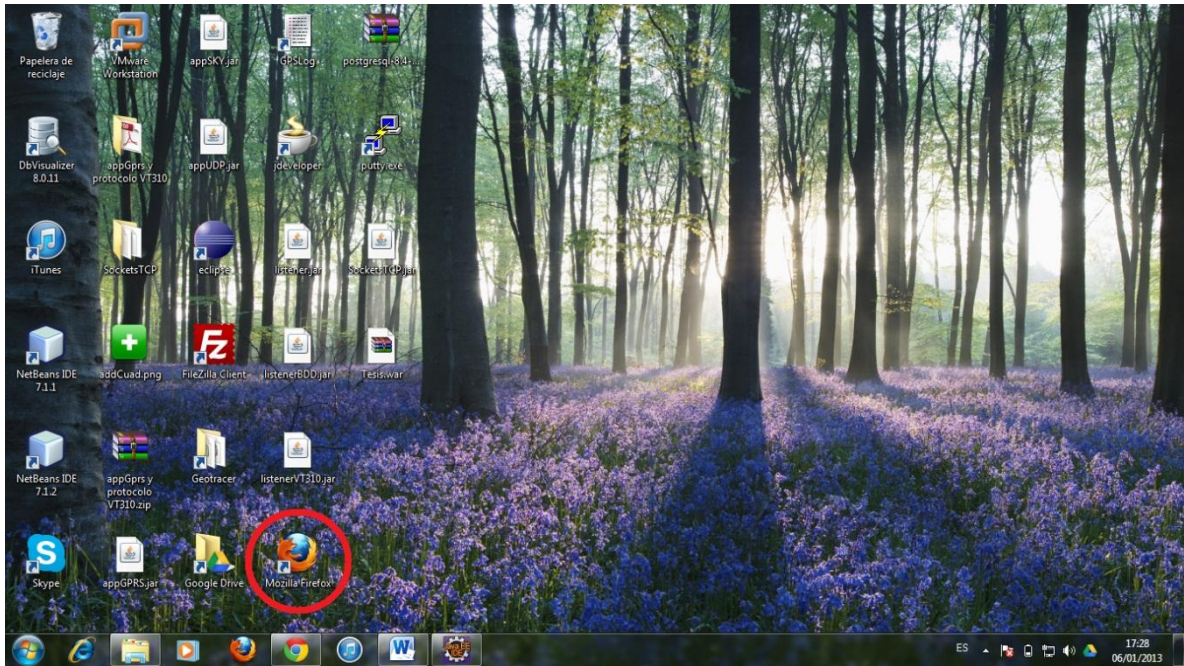


Ilustración 2.30: Escritorio de Windows 7. Fuente propia.

- 2) A continuación aparecerá la pantalla de Mozilla Firefox, ubíquese en la barra de direcciones, escriba la dirección del web Site: <http://146.255.98.137:8080/Tesis/> y pulse “Enter”; aparecerá la página de Login del sistema (ver Ilustración 2.31).

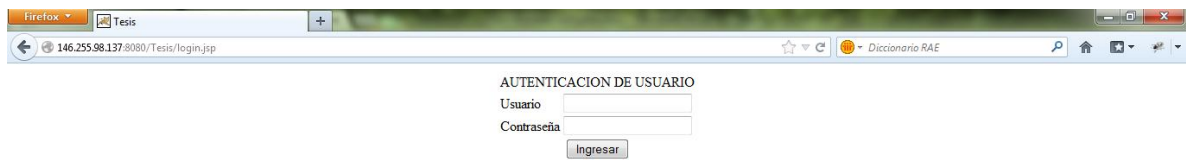


Ilustración 2.31: Página de login del sistema. Fuente propia.

- 3) Seguidamente se tendrá que autenticar el usuario Administrador con las credenciales otorgadas por el autor del presente trabajo, escribiendo el nombre del usuario y su contraseña, y luego pulse el botón “Ingresar” (ver Ilustración 2.32).

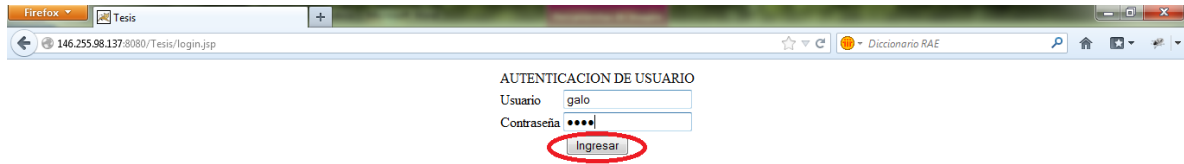


Ilustración 2.32: Botón “ingresar” de la página de login del sistema. Fuente propia.

- 4) Tras esto se podrá observar la página inicial (index.jsp) del sistema web geo-referenciado (ver Ilustración 2.33).

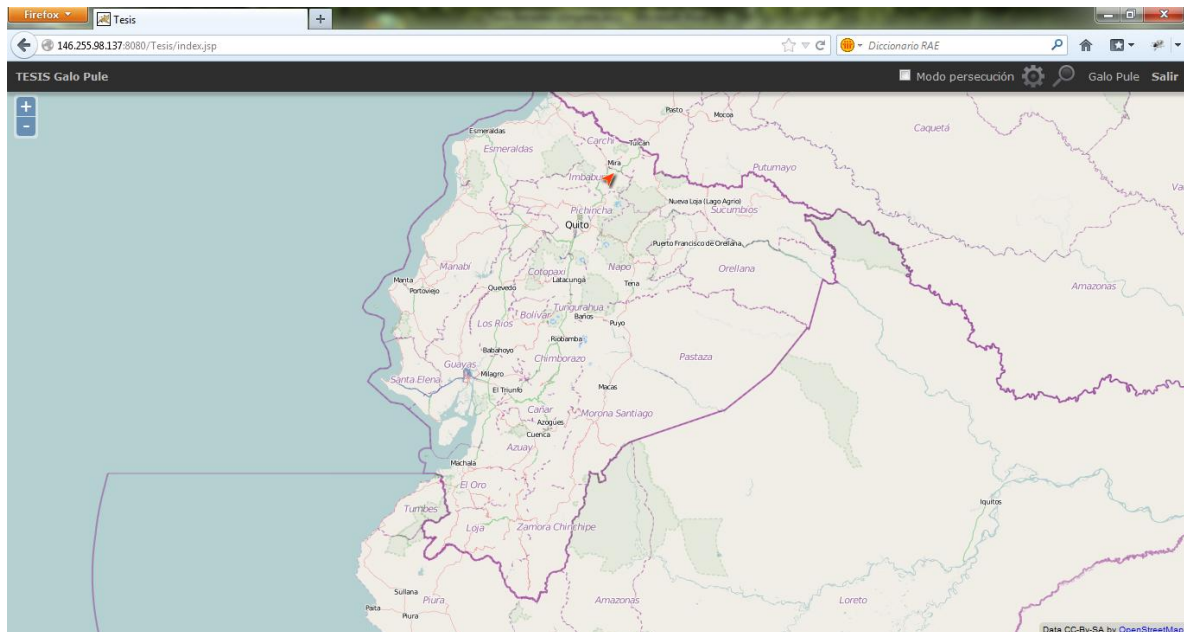



Ilustración 2.33: Página “index.jsp” del sistema. Fuente propia.

5) Para administrar el sistema como tal, se deberá acceder al Módulo de Administración, el cual se encuentra enlazado al botón de Administración  de la página inicial del sistema. Haga “Click” en este botón para acceder a la administración (ver Ilustración 2.34). Al acceder a la administración se podrá observar la pantalla de administración de usuarios primeramente, la que se explica en la siguiente sección.

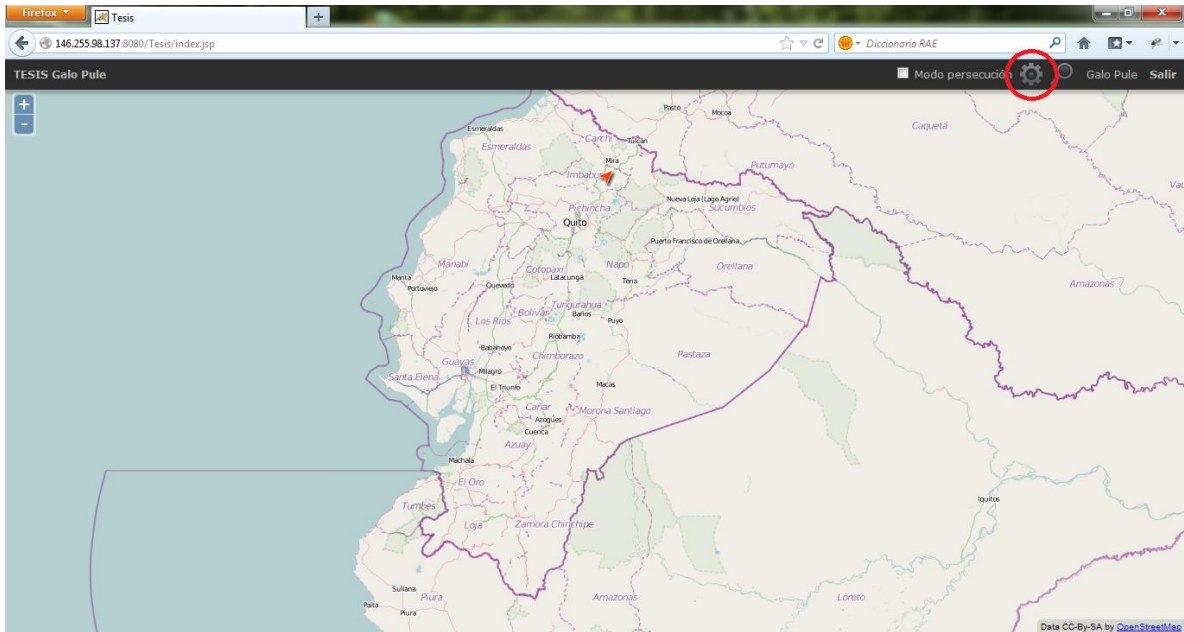


Ilustración 2.34: Botón para “administrar” el sistema. Fuente propia.

○ Administración de usuarios

- 1) En la pantalla de administración de usuarios se podrá crear, consultar, actualizar y eliminarlos usuarios del sistema (ver Ilustración 2.35); donde primeramente se pueden ver todos los usuarios ya creados.
- 2) Para **CREAR** un usuario usted deberá hacer “Click” en el botón “Agregar usuario”



(ver Ilustración 2.35), seguidamente aparecerá un formulario donde deberá llenar los datos del usuario como el estado, nombre, apellido, e-mail, usuario, y contraseña (ver Ilustración 2.36) y luego pulsar el botón “Guardar” o caso contrario el botón “Cancelar”.



Ilustración 2.35: Página de administración de usuarios del sistema. Fuente propia.

Ilustración 2.36: Formulario para el ingreso de usuarios. Fuente propia.

- 3) Para **EDITAR** un usuario usted deberá hacer “Click” en la acción “Editar usuario” que se encuentra ubicado a la derecha de cada registro (ver imagen 3.4.2.1.2.3) o también puede hacer “Click” en el botón “Editar usuario desde el menú” (ver Ilustración 2.37) seleccionando un usuario de la lista desde el “checkbox” de cada registro ubicado a la izquierda. Seguidamente aparecerá un formulario donde deberá modificar los datos del usuario y luego pulsar el botón “Guardar” o caso contrario el botón “Cancelar” (ver Ilustración 2.38).

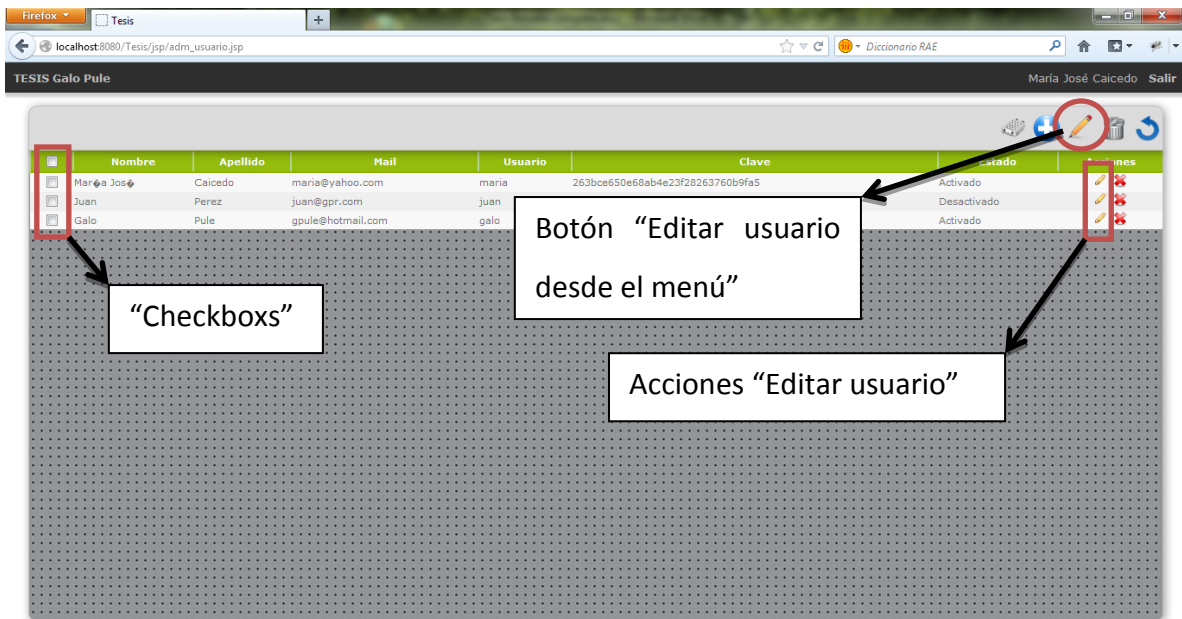




Ilustración 2.37: Botones de edición en la página de administración de usuarios. Fuente propia.

Ilustración 2.38: Formulario para la edición de usuarios. Fuente propia.

- 4) Para **ELIMINAR** un usuario usted deberá hacer “Click” en la acción “Eliminar usuario”  que se encuentra ubicado en la columna “Acciones” de cada registro (ver Ilustración 2.39) o también puede hacer “Click” en el botón “Eliminar usuario desde el menú”  (ver Ilustración 2.39) seleccionando un usuario de la lista desde el “checkbox” de cada registro ubicado en la columna de la izquierda.

Seguidamente aparecerá un cuadro de diálogo donde deberá aceptar o cancelar la transacción de eliminar el usuario seleccionado (ver Ilustración 2.40).

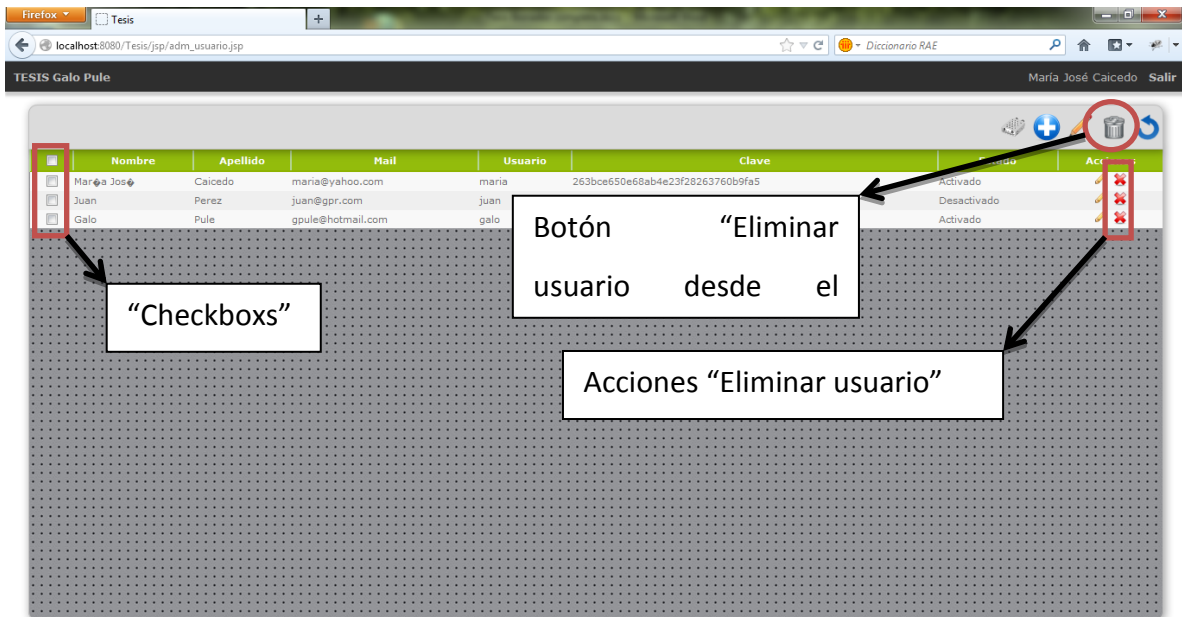


Ilustración 2.39: Botones de eliminación en la página de administración de usuarios. Fuente propia.

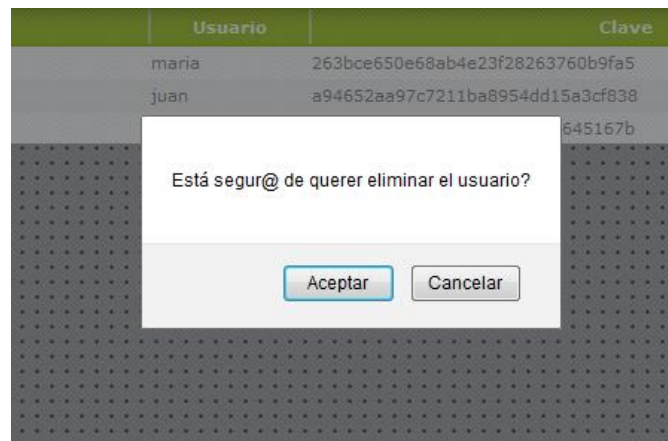




Ilustración 2.40: Cuadro de diálogo de eliminación de usuario. Fuente propia.

IMPORTANTE: Al eliminar un usuario debe tener en cuenta que también se eliminarán los AVLs que pertenezcan a este.

- 5) Además, se encuentra el botón “Refrescar”  que le permite actualizar los datos de la página si esto fuese necesario (ver Ilustración 2.41); también se encuentra el botón “Administración de AVLs”  el cual le llevará a la página

donde podrá manejar los datos de los AVLS del sistema (ver Ilustración 2.41); en la sección a continuación se explicará el funcionamiento de esta administración.

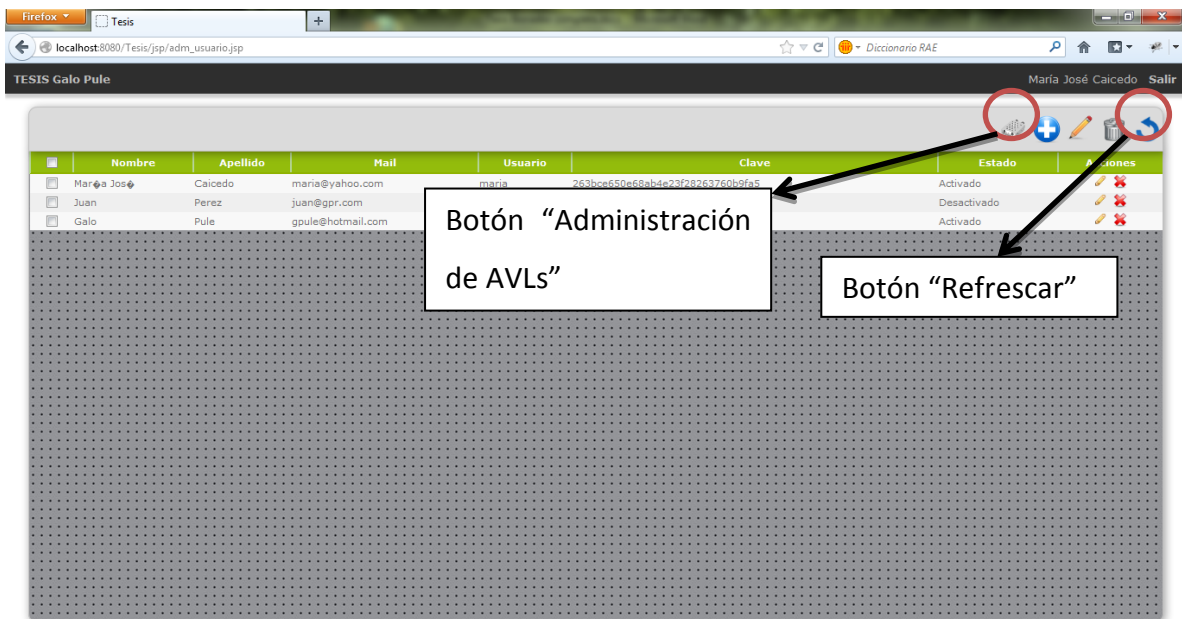



Ilustración 2.41: Botón de administración de AVLS. Fuente propia.

o Administración de avls

- 1) En la pantalla de administración de AVLS se podrá crear, consultar, actualizar y eliminar los AVLS del sistema (ver Ilustración 2.42); donde primeramente se pueden ver todos los AVLS ya creados.





Ilustración 2.42: Botón para agregar AVLS. Fuente propia.

- 2) Para **CREAR** un AVL debe hacer “Click” en el botón “Agregar AVL”  (ver Ilustración 2.42), seguidamente aparecerá un formulario donde deberá llenar todos los datos referentes al AVL (ver Ilustración 2.43) y luego pulsar el botón “Guardar” o caso contrario el botón “Cancelar”. Cabe señalar que los campos ID, chip, intervalo de trama e ID del objeto son campos obligatorios que tendrá que llenar.



The screenshot shows a web application window titled "Ingreso del AVL". The form is organized into two main sections. The first section, "Datos del objeto", contains the following fields: ID (text input), S/N (text input), Marca (text input), Características (text input), Proveedor (text input), Proveedor del chip (text input), Usuario (dropdown menu with "Seleccione un usuario" selected), IMEI (text input), Modelo (text input), Fabricante (text input), Chip (text input), and Intervalo de trama (text input). The second section, "Datos del objeto", contains: ID (text input), ID #2 (text input), Nombre (text input), Descripción (text input), Marca (text input), Modelo (text input), Año (text input), and Observaciones (text input). At the bottom of the form are two buttons: "Guardar" (with a green plus icon) and "Cancelar" (with a red minus icon).

Ilustración 2.43: Formulario de creación de AVLs. Fuente propia.

- 3) Para **EDITAR** un usuario usted deberá hacer “Click” en la acción “Editar AVL”  que se encuentra ubicada a la derecha de cada registro (ver Ilustración 2.44) o también puede hacer “Click” en el botón “Editar AVL desde el menú”  (ver Ilustración 2.44) seleccionando un AVL de la lista desde el “checkbox” de cada registro ubicado a la izquierda. Seguidamente aparecerá un formulario donde deberá modificar los datos del AVL y luego pulsar el botón “Guardar” o caso contrario el botón “Cancelar” (ver Ilustración 2.45).

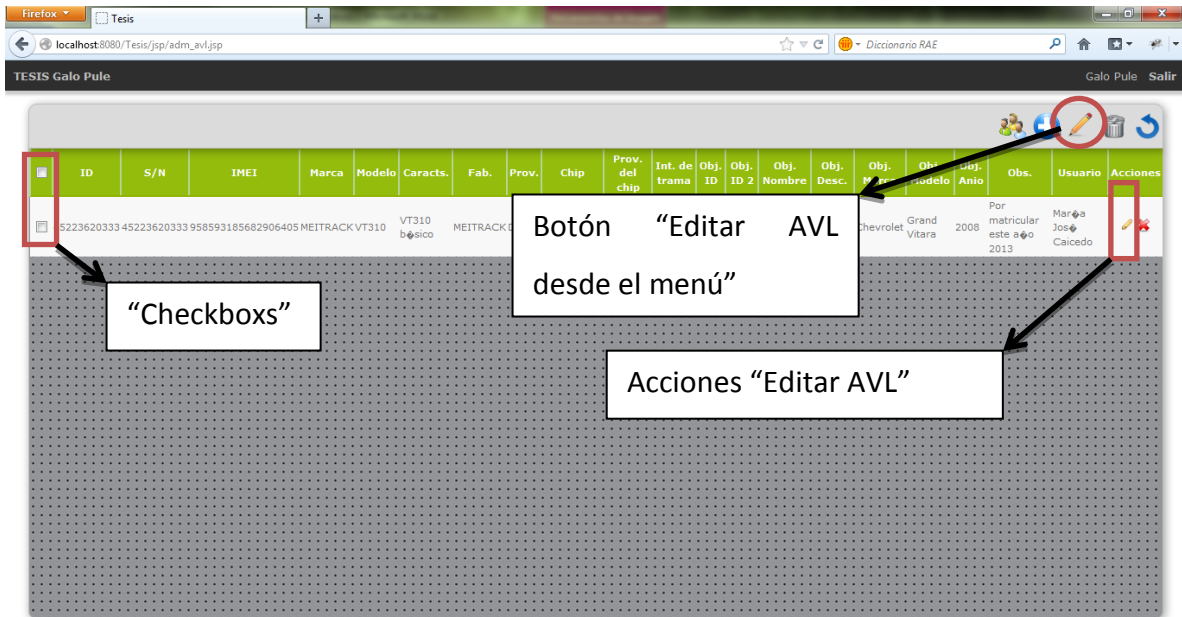



Ilustración 2.44: Botones de edición en la página de administración de AVLs. Fuente propia.

Ilustración 2.45: Formulario de edición de AVLs. Fuente propia.

- 4) Para **ELIMINAR** un AVL usted deberá hacer "Click" en la acción "Eliminar AVL"  que se encuentra ubicada en la columna "Acciones" de cada registro (ver Ilustración 2.46) o también puede hacer "Click" en el botón "Eliminar AVL desde el menú"



(ver Ilustración 2.46) seleccionando un usuario de la lista desde el "checkbox"

de cada registro ubicado en la columna de la izquierda. Seguidamente aparecerá un cuadro de diálogo donde deberá aceptar o cancelar la transacción de eliminar el AVL seleccionado (ver Ilustración 2.47).

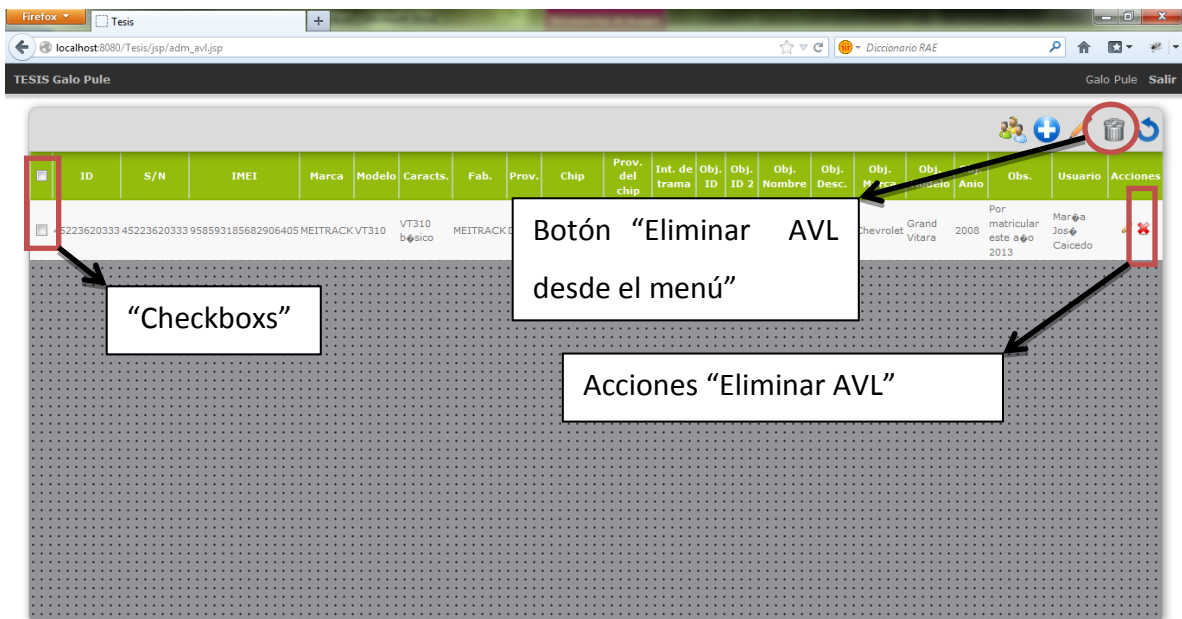


Ilustración 2.46: Botones de eliminación en la página de administración de AVLS. Fuente propia.

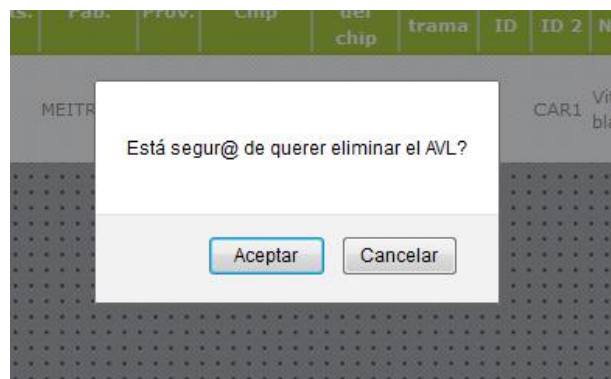




Ilustración 2.47: Cuadro de diálogo para la eliminación de AVLS. Fuente propia.

IMPORTANTE: Al eliminar un usuario o un AVL debe tener en cuenta que no se podrán recuperar los estos datos en la posterioridad.

- 5) Además, se encuentra el botón “Refrescar”  que le permite actualizar los datos de la página (ver Ilustración 2.48); también se encuentra el botón “Administración de usuarios”  el cual le llevará a la página donde podrá

manejar los datos de los usuarios del sistema explicado en la sección anterior (ver Ilustración 2.48).

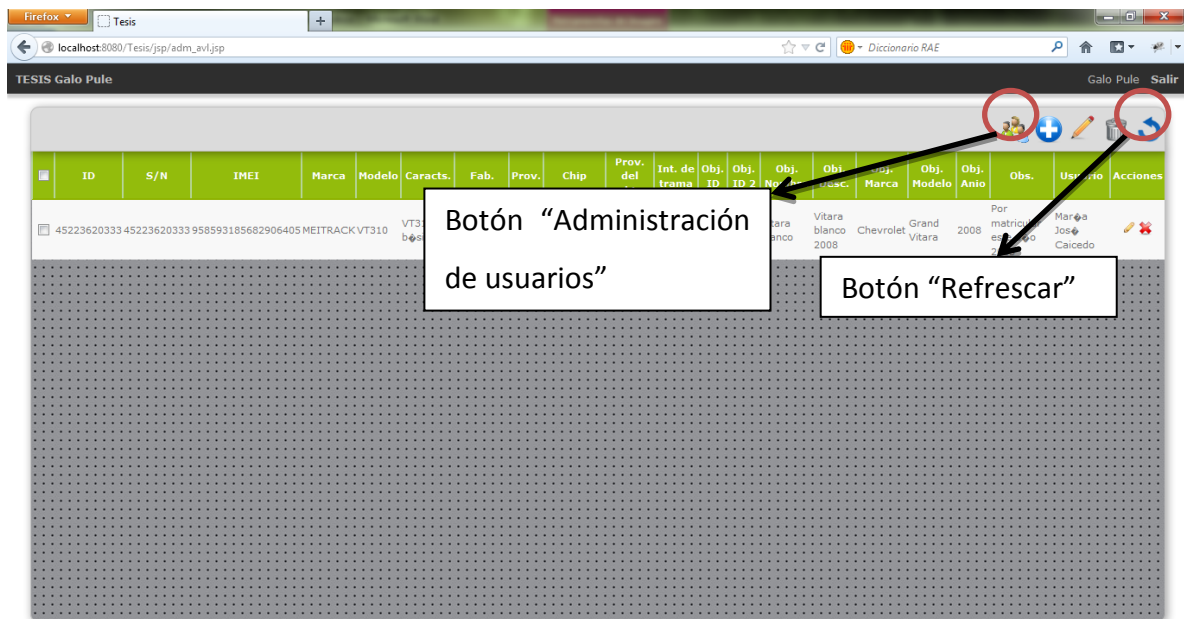





Ilustración 2.48: Botón de administración de usuarios. Fuente propia.

- Manual del usuario final

En esta sección se explicará la funcionalidad del sistema web geo-referenciado orientado al usuario final. Se explicará paso a paso cómo debe el usuario final utilizar el sistema de manera que obtenga las ventajas de este recurso tecnológico.

1) Se tendrá que ingresar al sistema, tal y como se explicó anteriormente, con las credenciales otorgadas por el autor del presente trabajo, de manera que se encuentre con la página inicial del sistema web geo-referenciado (ver Ilustración 2.49).

2) Para localizar su vehículo haga “Click” en el botón “Localizar”  (ver Ilustración 2.49), esta acción le permite hacer un zoom (acercar) a su vehículo y centrarlo en el mapa (ver Ilustración 2.49). Podrá observar la posición actual de su vehículo, estará representado mediante una saeta o flecha que apunta hacia la dirección a donde su vehículo se dirige (ver Ilustración 2.50). El color de la saeta expresa el estado del vehículo, estos estados son:

- Color verde : Este estado significa que el vehículo se encuentra en movimiento en ese momento determinado.
- Color rojo : Este estado quiere decir que el vehículo se encuentra parado (estacionado) en ese momento determinado.

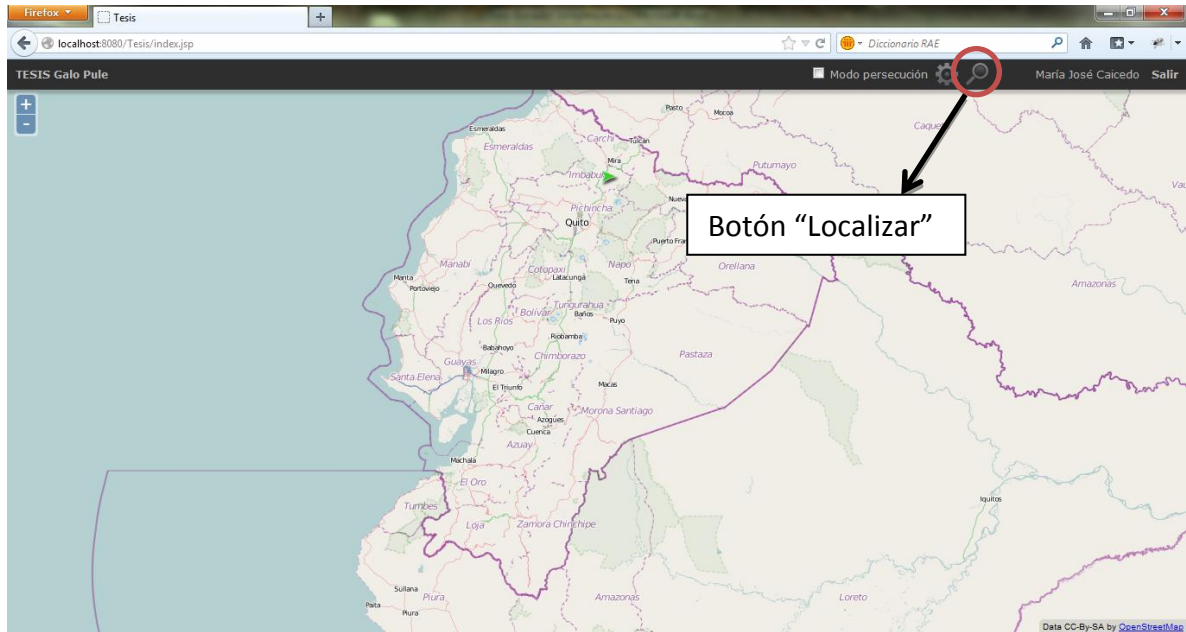


Ilustración 2.49: Página inicial del sistema. Fuente propia.

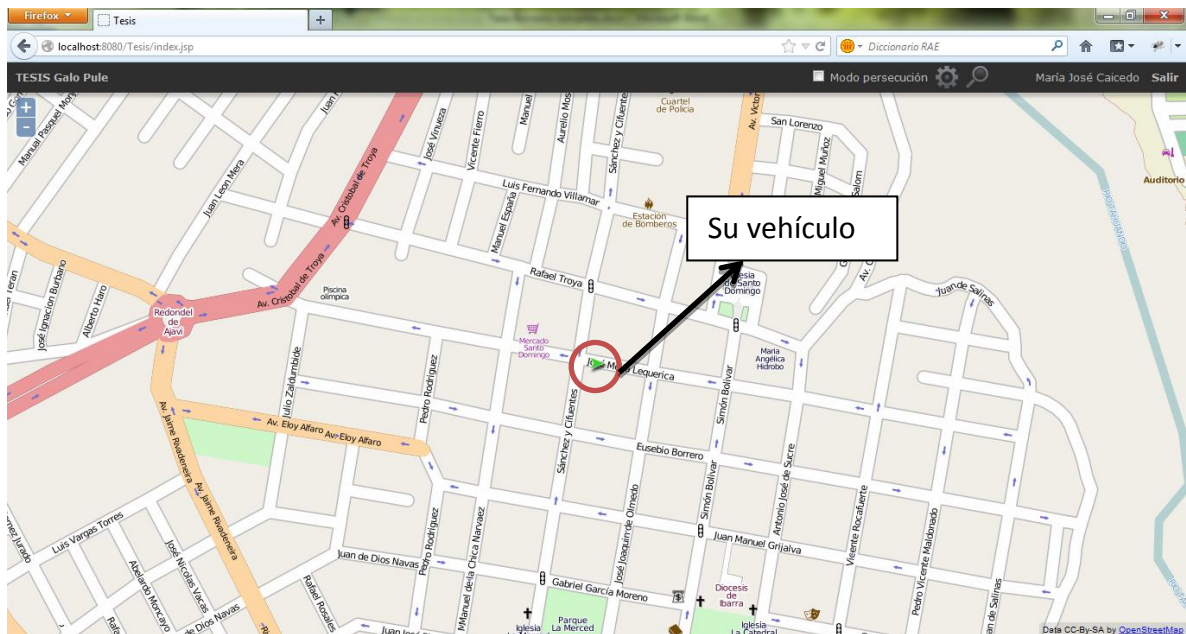



Ilustración 2.50: Ubicación del vehículo. Fuente propia.

- 3) Para observar los detalles de la posición actual haga “click” en la saeta , de inmediato aparecerá un cuadro donde se observan los datos de la localización actual del vehículo (ver Ilustración 2.51). En este cuadro podrá observar datos como el ID del AVL, la fecha y hora del GPS, la local y del servidor en la cual se ha registrado la posición del vehículo; también podrá observar la latitud y longitud, altitud, velocidad y la orientación del vehículo.

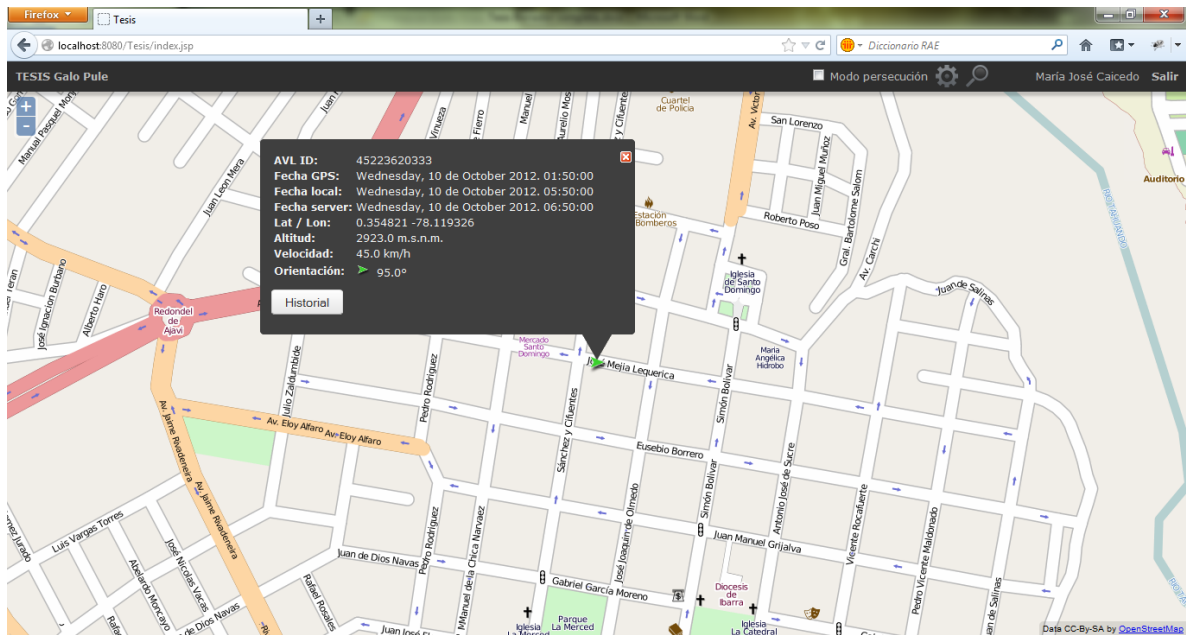


Ilustración 2.51: Pop-up de los detalles del vehículo. Fuente propia.

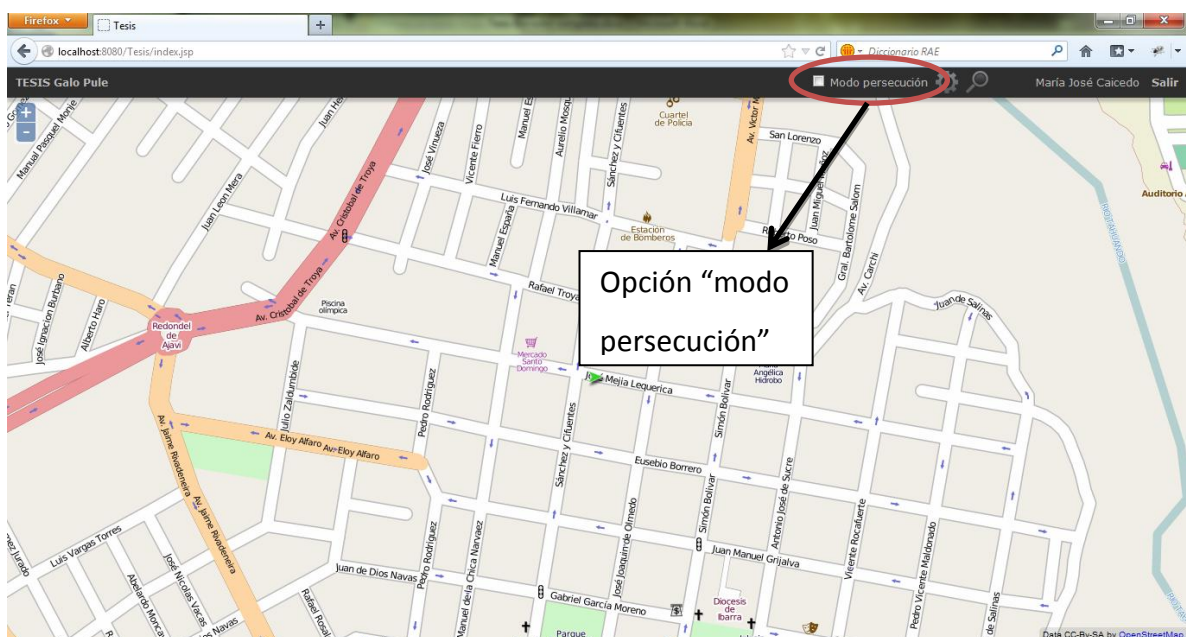
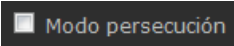


Ilustración 2.52: Opción de modo de persecución. Fuente propia.

- Modo persecución

El modo persecución le permite observar el movimiento del vehículo sin perder la perspectiva del mapa, esto quiere decir que el vehículo va a estar en el centro del mapa si la opción de “modo de persecución”  se encuentra activada. Esta opción se encuentra en la barra superior de la página de inicio del sistema (ver Ilustración 2.52).

- Historial

Esta funcionalidad le permite hacer un recuento de las posiciones en las que su vehículo estuvo en cierto lapso de tiempo, usted puede determinar este lapso de tiempo ingresando la fecha y hora inicial como también la final en las que usted quiere que se procese el historial. A continuación se detallarán los pasos que debe seguir para efectuar el “historial”.

1. Debe acceder a la página inicial del sistema tal y como se explicó anteriormente, luego hacer “click” en la saeta que determina la posición actual de su vehículo. Aparecerá el cuadro de detalles de la posición actual del vehículo, el cual contiene el botón “Historial” (ver Ilustración 2.53), haga “click” en este y le llevará a la página de “historial” (ver Ilustración 2.54).

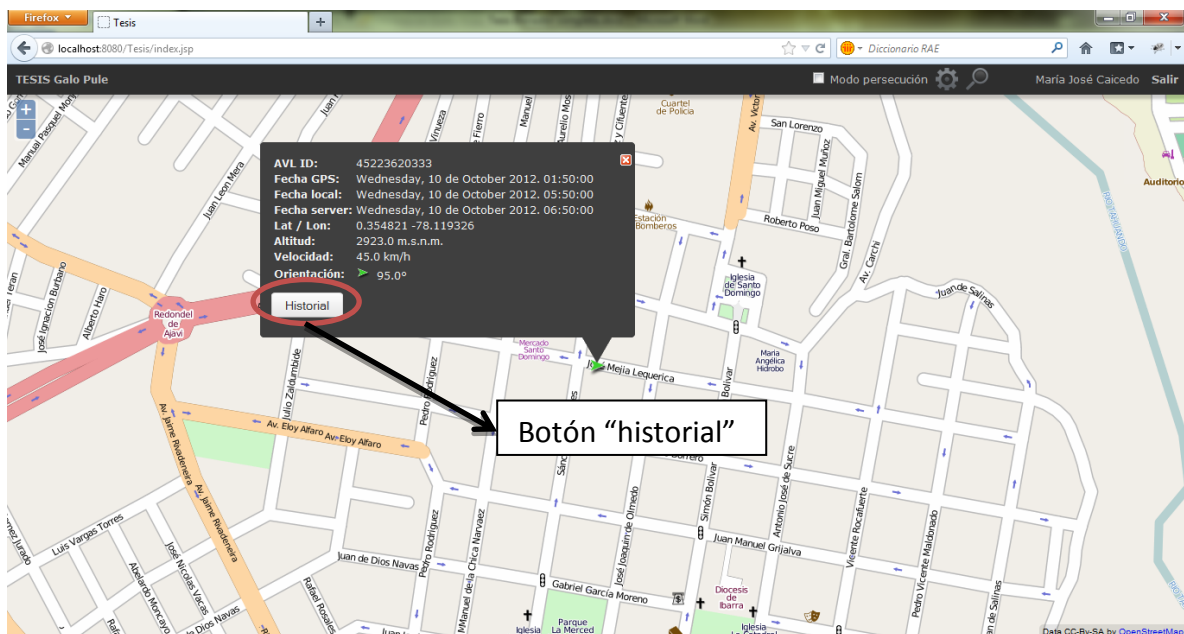


Ilustración 2.53: Botón "Historial" del vehículo. Fuente propia.

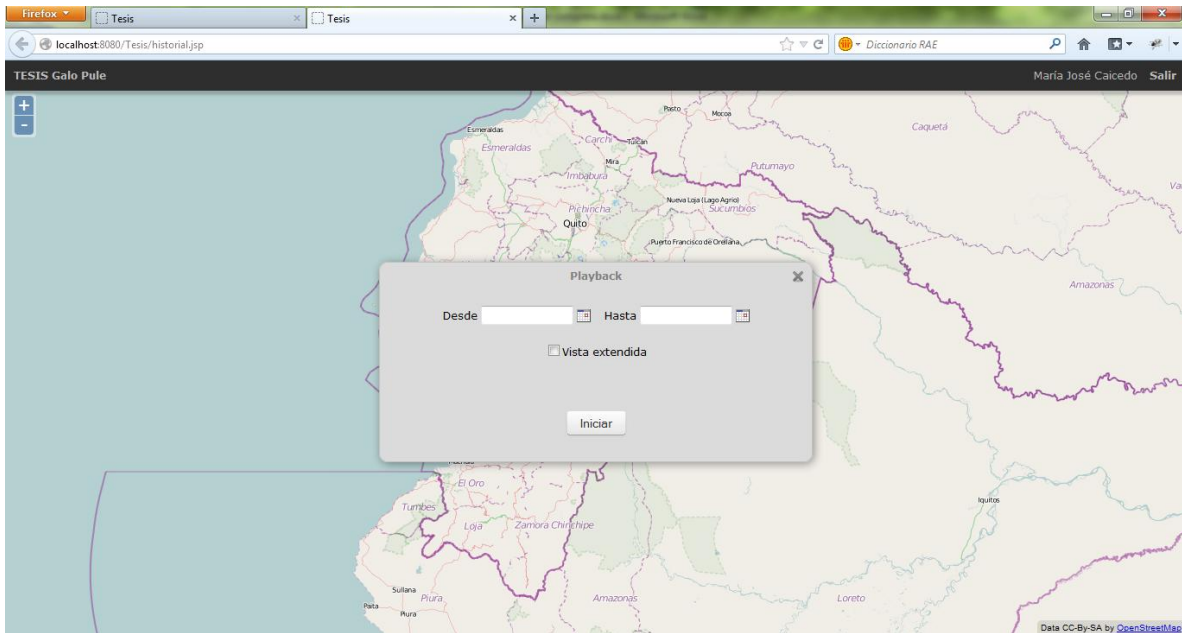



Ilustración 2.54: Página de "Historial" del vehículo. Fuente propia.

2. Ingrese la fecha y hora inicial haciendo "click" en el calendario  del campo "Desde" del formulario (ver Ilustración 2.55); luego seleccione la fecha y hora que desee la cual aparecerá en el campo "Desde" del formulario (ver Ilustración 2.56).

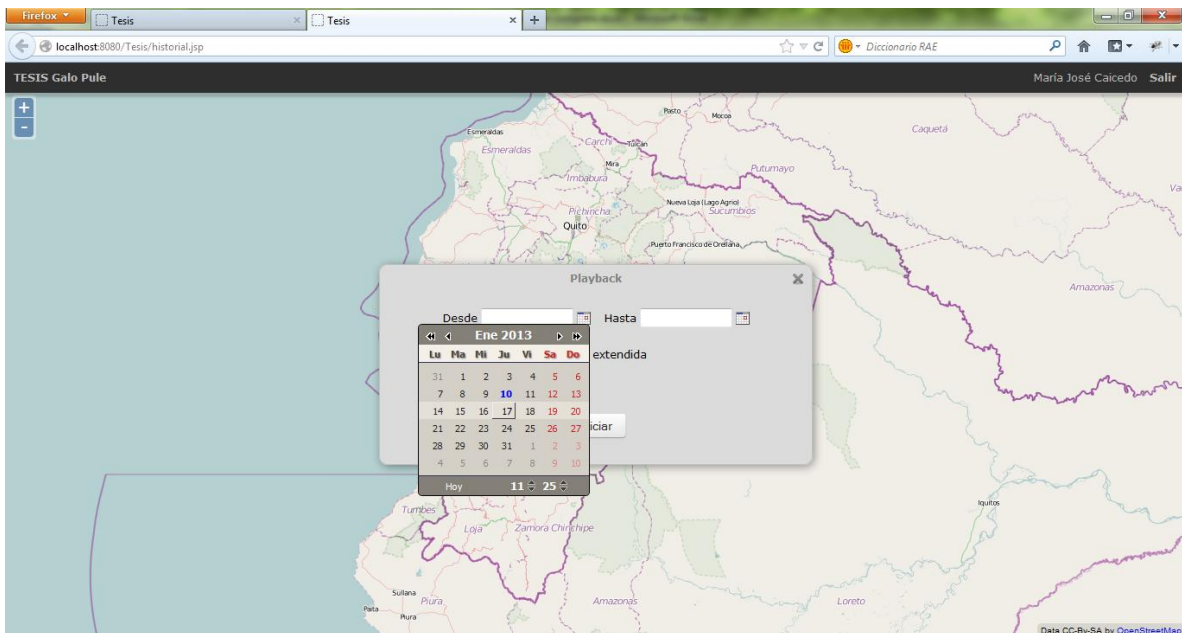


Ilustración 2.55: Calendario que se despliega para ingresar la fecha inicial del "Historial". Fuente propia.

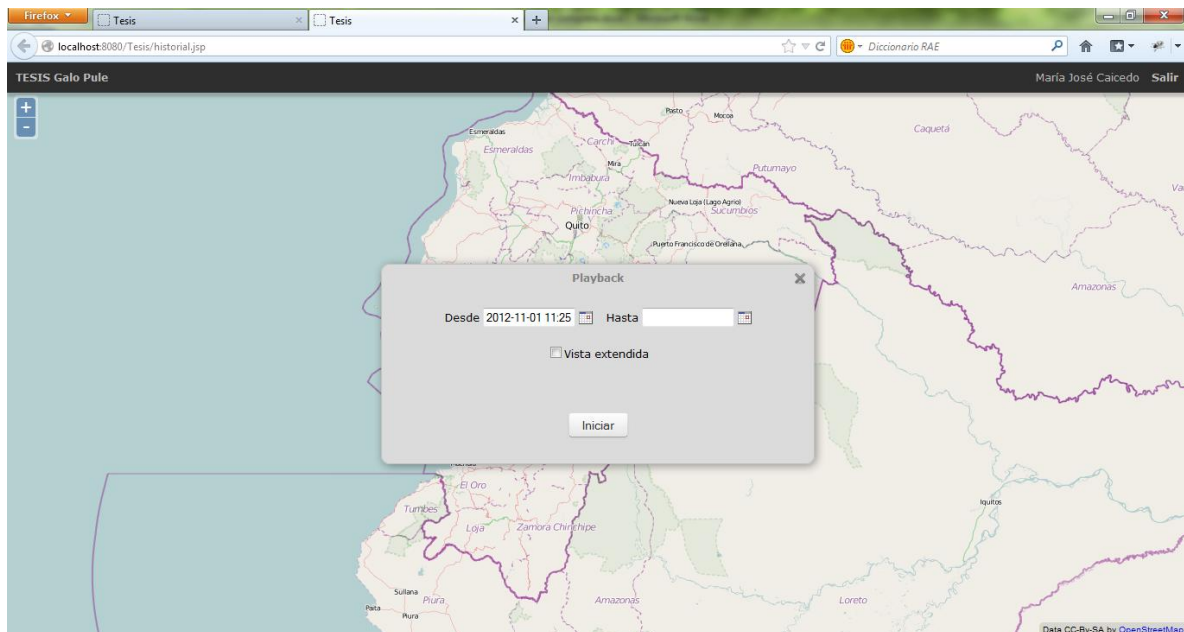


Ilustración 2.56: Fecha inicial seleccionada desde el calendario. Fuente propia.

3. Ingrese la fecha y hora final en el campo “Hasta” tal y como se explicó en el paso 2 (ver Ilustración 2.57).

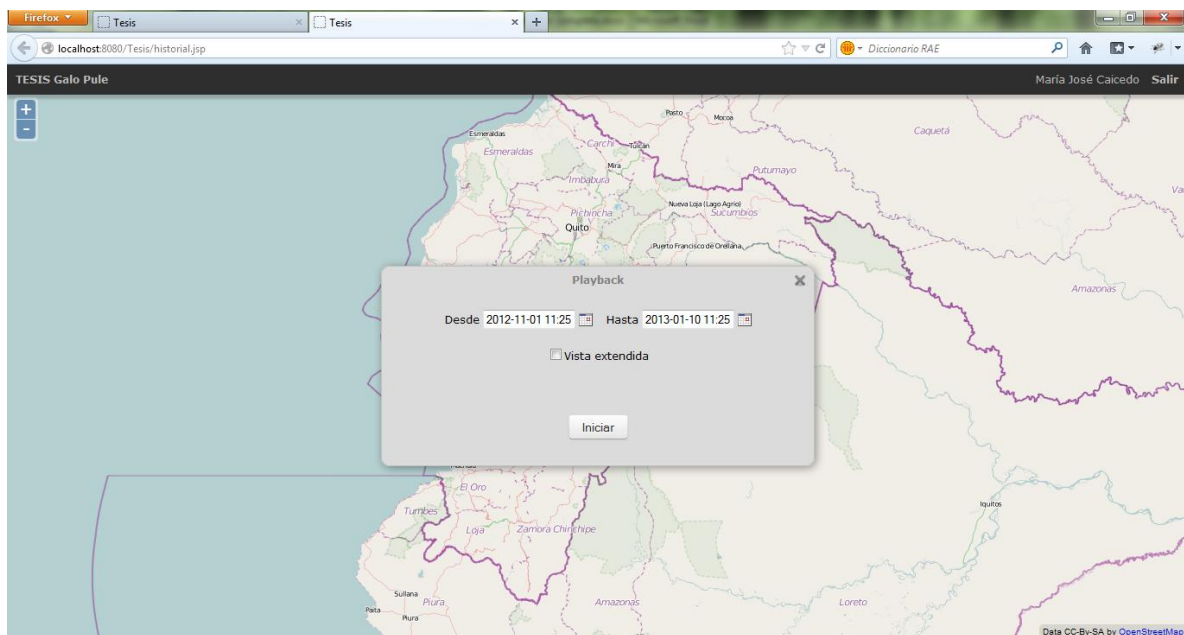
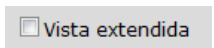


Ilustración 2.57: Fecha final seleccionada desde el calendario. Fuente propia.

4. Si está de acuerdo con los datos que ha ingresado, puede optar por hacer una vista extendida del historial haciendo “click” en el cuadro de selección “Vista extendida”



La vista extendida le permite ver un historial ampliado y rápido a diferencia de que si no seleccionara esta opción, vería un historial lento y en modo persecución que se detalló anteriormente.

Veamos como es un historial con vista extendida, para lo cual debe hacer “click” en el botón “Iniciar” y seleccionando la opción “Vista extendida” (ver Ilustración 2.58).

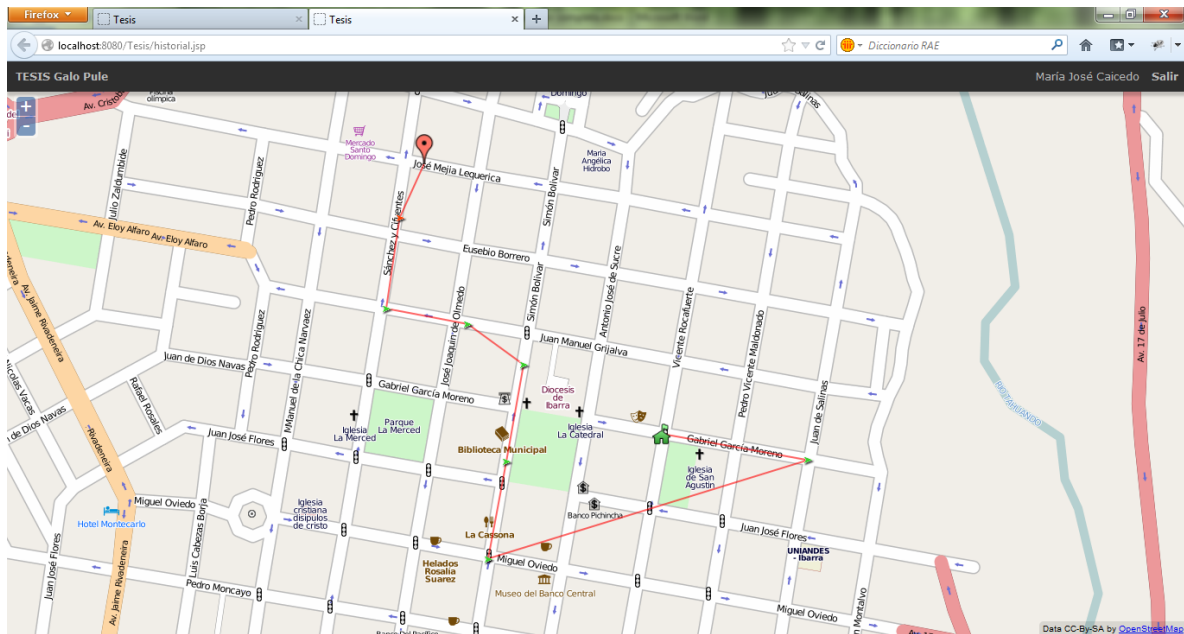



Ilustración 2.58: Vista extendida del historial del vehículo. Fuente propia.

Se puede observar la ruta que trazó el vehículo en cierto intervalo de tiempo a través

de una línea de color rojo. También se puede observar el punto de partida  y el

de llegada . En cada uno de estos íconos incluyendo las saetas que muestran la posición de cada transmisión del AVL, se puede hacer “click” y observar un cuadro de detalles para cada posición (ver Ilustración 2.59).

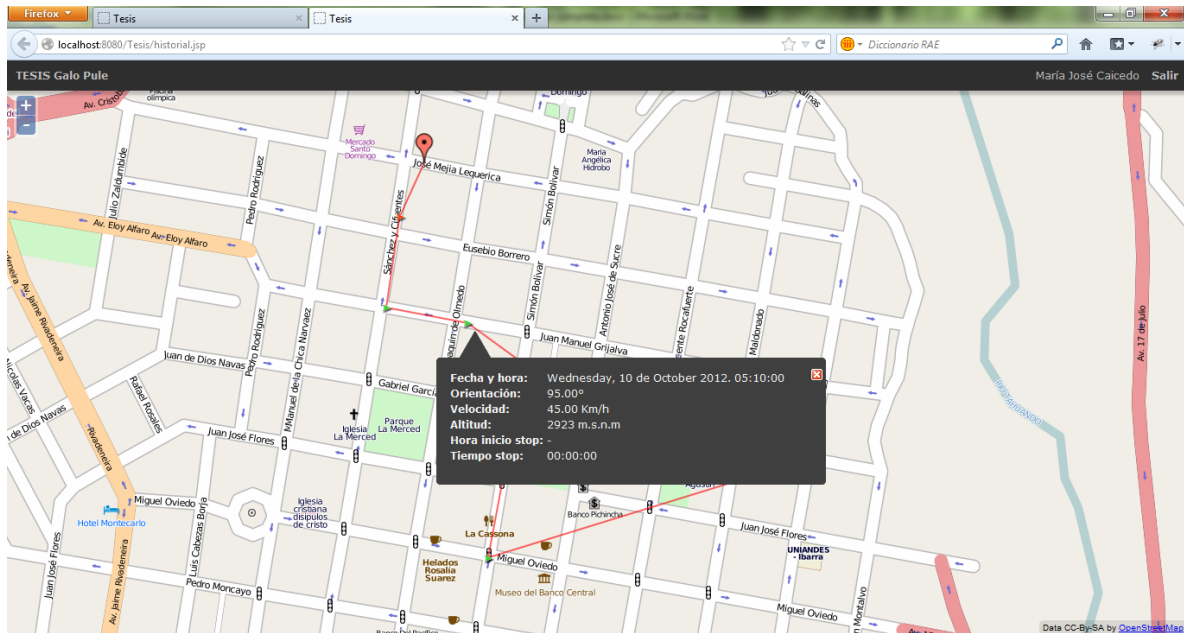


Ilustración 2.59: Detalles de cada posición del vehículo en el historial. Fuente propia.

En este cuadro de detalles podrá observar datos como la fecha y hora en la que estuvo el vehículo en esta posición, su orientación, velocidad, altitud y, si es que el vehículo ha hecho una parada en este punto se generará la hora de inicio de esta parada y el tiempo que se mantuvo en este estado de “stop o parado”.

CAPÍTULO III

CONCLUSIONES Y RECOMENDACIONES

3.1. Conclusiones

Para finalizar, de acuerdo a la información presentada en cada capítulo del presente trabajo es conveniente hacer un resumen de conclusiones a las cuales se llegó acorde al avance en el desarrollo del trabajo y del documento en sí. Es imprescindible afirmar que este ha sido un trabajo en el que el autor ha puesto mucho esfuerzo en investigación y en la aplicación de las mejores técnicas para que este obtenga el peso justo en lo que refiere a la aplicación de las ciencias de la computación. El siguiente resumen de conclusiones se encuentra orientado al cumplimiento de los objetivos específicos y general que se plantearon al iniciar el trabajo, estos son:

- Con respecto a la implementación del VPS con el software base necesario como el sistema operativo, el arrendamiento y la puesta en marcha del mismo, fue muy fácil conseguirlo; lo que refiere a la instalación de software complementario como la base de datos geo-espacial igualmente no demandó una investigación exhaustiva.
- Se logró adquirir el AVL necesario para el rastreo del vehículo, con demora en el envío desde China aunque sin mayor inconveniente. Se tuvo que optar por otra operadora celular debido a que Claro, la que se propuso utilizar en el anteproyecto manejaba costos mayores con relación a Movistar, la que finalmente se decidió utilizar. Por otra parte, la configuración del AVL para que opere con Movistar no tuvo mayor inconveniente debido a la documentación concisa y detallada que contenía este.
- El desarrollo de la aplicación Java para la comunicación, descryptado y almacenamiento de las tramas de los AVLS se logró satisfactoriamente gracias a la aplicación de la programación de Sockets explicada en el capítulo II; fue la parte más emocionante del desarrollo ya que se creía que iba a ser lo más complicado del trabajo.
- La modificación de la cartografía de OpenStreetMap es una ventaja esencial para lo que es el desarrollo de soluciones de este tipo, ya que es posible corregir errores

de cartografía y adicionar información que haga falta en la misma, y así colaborar con la comunidad internacional de OSM para que así todo el mundo esté en la capacidad de usar esta cartografía.

- El desarrollo de la aplicación Web Java para la visualización de los datos en tiempo real y su historial, al igual que para la administración de los mismos; tuvo grandes modificaciones de principio a fin ya que se la desarrolló en primera instancia con PHP lo cual es muy recomendable para comparar los dos tipos de lenguaje de programación: PHP y Java. De igual manera la utilización de JPA y la utilización de hilos de procesos son herramientas muy versátiles que permiten la optimización de los recursos tecnológicos, al igual que la utilización de un patrón de diseño MVC.
- La incorporación y personalización de herramientas Open Source tales como el visor de mapas, la cartografía, los frameworks (librerías JavaScript) de lado cliente y de lado de servidor ha sido un ámbito muy enriquecedor debido a que se ha podido explotar estas herramientas al máximo de su capacidad gracias a que existe una extensa documentación y se ha podido también manipular el código fuente en algunas de ellas; por lo tanto se considera a las herramientas Open Source como una fuente rica de conocimiento y ciencia.
- En general, el desarrollo del presente trabajo ha tenido gran repercusión en el autor ya que este ha sido su primer trabajo de esta magnitud en el ámbito del conocimiento de las ciencias de la computación y las comunicaciones; por lo que este trabajo va a ser el ápice de una gran labor investigativa en lo que refiere a estas ciencias y el propulsor de grandes sueños.

3.2. Recomendaciones

A continuación se describirán algunas recomendaciones que se han tomado en cuenta a lo largo del desarrollo del presente trabajo.

- Se sugiere que se investigue y aplique una nueva metodología de desarrollo de software que se ajuste a los constantes cambios que sufre esta rama de los sistemas computacionales, ya que la metodología utilizada no se ajustó completamente a las necesidades que tenía el presente desarrollo de software porque no se trataba de software convencional que se ha venido desarrollando

durante años sino que es un tipo de software moderno que no tiene muchos años en el mercado; por lo tanto es necesario crear una nueva metodología de desarrollo de software que se ajuste a las nuevas necesidades.

- Un ideal cercano sería la independencia tecnológica del extranjero en lo que refiere a la producción de equipos localizadores mediante GPS; en el Ecuador se está incursionando en este campo pero todavía no es suficiente ya que la falta de industrialización en nuestro medio se encuentra retrasado. De igual manera lo que se refiere a la adquisición de servidores ecuatorianos que ofrezcan sus servicios, se debería poner más énfasis en estos temas.
- La cartografía de OpenStreetMap se encuentra liberada para uso público por lo que cada quien debería corregir o aumentar información en esta plataforma para el beneficio común.
- Respecto al desarrollo de una aplicación Web como la del presente trabajo desarrollada en Java, se recomienda que como primer preámbulo se desarrolle ésta en PHP haciendo una prueba de concepto evidenciando las funcionalidades más importantes en una aplicación de este tipo, luego se migre el código fuente a Java para así comparar estas dos plataformas de programación Web para disponer de un amplio espectro en este ámbito. De igual forma que se investigue acerca del procesamiento múltiple en los servidores y el procesamiento distribuido para la optimización de recursos tecnológicos como la transferencia de datos por la red y el procesamiento.
- También, se recomienda hacer énfasis en la utilización de software libre y código abierto debido ya que con ellos se puede ampliar el espectro investigativo en el área de las ciencias de la computación y los sistemas computacionales lo cual desembocaría en el desarrollo tecnológico y por ende el desarrollo económico del país.
- Finalmente, se recomienda que se sigan investigando nuevas tecnologías relacionadas al software geográfico porque es un ámbito nuevo que se encuentra tomando la posta del software tradicional como el software financiero-contable al igual que el área de la inteligencia artificial; por ende que se haga uso del presente trabajo para aquellas finalidades.

GLOSARIO

A

AJAX.- Mecanismo de combinación de tecnologías y estándares de cliente, consistente en la solicitud asíncrona de datos al servidor desde una página web y la utilización de éstos para actualizar una parte de la misma. · 2, 31, 34, 60, 61, 62, 63, 64, 85

Apache Tomcat.- Servidor web y contenedor de servlets que implementa tecnologías Java Servlet y JavaServer Pages. · 19, 20, 36, 124

Aplicación.- Programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajos. · 1, 6, 8, 9, 10, 11, 12, 13, 19, 20, 29, 30, 31, 32, 33, 34, 35, 40, 45, 47, 51, 55, 60, 61, 62, 63, 64, 65, 69, 82, 83, 84, 85, 86, 89, 90, 96, 98, 99, 100, 101, 102, 104, 105, 110, 127, 147, 148, 149

AVL.- Localizador Automático de Vehículos, sistema de localización remota en tiempo real, basados generalmente en el uso de un GPS y un sistema de transmisión que es frecuentemente un módem inalámbrico. · 8, 9, 10, 11, 12, 45, 47, 86, 93, 94, 95, 96, 97, 99, 100, 107, 108, 135, 136, 137, 140, 144, 147

B

Back-end.- Parte del software que procesa la entrada desde front-end (parte del software que interactúa y obtiene los datos de entrada del usuario). · 25, 116

Base de datos.- Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. · 2, 10, 21, 22, 23, 24, 25, 26, 32, 33, 35, 38, 39, 40, 41, 42, 44, 47, 79, 80, 82, 83, 85, 86, 87, 92, 93, 96, 97, 98, 99, 102, 105, 106, 107, 108, 110, 113, 114, 115, 117, 123, 127, 147

C

Cartografía.- Es la ciencia que se encarga del estudio y de la elaboración de los mapas geográficos, territoriales y de diferentes dimensiones lineales y demás. · 7, 8, 9, 11, 12, 79, 147, 148, 149

CSS.- Las hojas de estilo en cascada (Cascading Style Sheets) hacen referencia a un lenguaje de hojas de estilos usado para describir la presentación semántica (el aspecto y formato) de un documento escrito en lenguaje de marcas. · 31, 34, 60, 85

D

Datos espaciales.- Datos que representas figuras simples en forma de geometrías con coordenadas de latitud y longitud. · XVII, 2, 15, 16, 18, 26, 32, 80

DBMS.- Sistema Manejador de Base de Datos, es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos. · 2

E

Entidad.- En bases de datos, una entidad es la representación de un objeto o concepto del mundo real que se describe en una base de datos. · 36, 37, 38, 39, 40, 41, 42, 44, 86

F

Frameworks.- Marco de trabajo. es una estructura conceptual y tecnológica de soporte definido, normalmente con artefactos o módulos de software concretos, que puede servir de base para la organización y desarrollo de software. · 8, 148

G

Geo-datos.- Dato que lleva consigo información geográfica. · 10, 83

Geo-espacial.- La información geo-espacial proporciona una perspectiva única para analizar eventos y procesos que tienen lugar sobre el territorio, pues permite localizar cada evento en su posición geográfica. · 24, 147

Geo-localización.- Forma de localizar geográficamente. · 4, 12

Geo-referenciación.- Neologismo que refiere al posicionamiento con el que se define la localización de un objeto espacial (representado mediante punto, vector, área, volumen) en un sistema de coordenadas y datum determinado. · 18

GIS.- Sistema de Información Geográfica. Es una integración organizada de hardware, software y datos geográficos diseñada para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión geográfica. · 15, 16, 24

GPRS.- General Packet Radio Service (GPRS) o servicio general de paquetes vía radio creado en la década de los 80 es una extensión del Sistema Global para Comunicaciones Móviles (Global System for Mobile Communications o GSM) para la transmisión de datos mediante conmutación de paquetes. · 86, 107, 108

GPS.- El SPG o GPS (Global Positioning System: sistema de posicionamiento global) o NAVSTAR-GPS1 es un sistema global de navegación por satélite (GNSS) que permite determinar en todo el mundo la posición de un objeto. · 8, 10, 12, 79, 81, 93, 140, 149

H

HTTP.- Hypertext Transfer Protocol o HTTP (en español protocolo de transferencia de hipertexto) es el protocolo usado en cada transacción de la World Wide Web. · 31, 49, 60, 61, 62, 64

I

Internet.- Es un conjunto descentralizado de redes de comunicación interconectadas que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial. · 15, 30, 48, 50, 56, 58, 62, 63

J

Java.- Java es un lenguaje de programación de propósito general, concurrente, basado en clases, y orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. · 2, 10, 13, 19, 21, 23, 24, 29, 31, 32, 34, 35, 36, 37, 42, 45, 47, 48, 50, 51, 53, 62, 85, 97, 100, 103, 106, 107, 108, 109, 110, 147, 148, 149

Java Enterprise Edition.- es una plataforma de programación para desarrollar y ejecutar software de aplicaciones en el lenguaje de programación Java. Permite utilizar arquitecturas de N capas distribuidas y se apoya ampliamente en componentes de software modulares ejecutándose sobre un servidor de aplicaciones. · 29, 32, 84

Java Standard Edition.- Es una colección de APIs del lenguaje de programación Java útiles para muchos programas de la Plataforma Java. · 29, 45, 84

JavaScript.- JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,³ basado en prototipos, imperativo, débilmente tipado y dinámico. · 2, 31, 34, 55, 56, 57, 59, 61, 64, 65, 85, 88, 148

JPA.- Es un framework del lenguaje de programación Java que maneja datos relacionales en aplicaciones usando la Plataforma Java en sus ediciones Standard (Java SE) y Enterprise (Java EE). · 35, 36, 37, 39, 40, 42, 44, 148

JSON.- Acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. · 34

JSP.- JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo. · 19, 34, 85

L

Latitud.- La latitud es la distancia angular entre la línea ecuatorial (el ecuador), y un punto determinado de la Tierra, medida a lo largo del meridiano en el que se encuentra dicho punto. · 10, 26, 68, 83, 99, 102, 140

Linux.- Es un núcleo libre de sistema operativo basado en Unix.³ Es uno de los principales ejemplos de software libre. Linux está licenciado bajo la GPL v2 y está desarrollado por colaboradores de todo el mundo. · 8, 10

Longitud.- expresa la distancia angular entre un punto dado de la superficie terrestre y el meridiano que se tome como 0° (es decir el meridiano base) medida a lo largo del paralelo en el que se encuentra dicho punto. · 10, 26, 68, 83, 99, 102, 140

M

MVC.- Modelo Vista Controlador (MVC) es un patrón o modelo de abstracción de desarrollo de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos (modelo, vista y controlador). · 2, 32, 33, 42, 45, 85, 148

O

Open Source.- Código abierto (o fuente abierta) es el término con el que se conoce al software distribuido y desarrollado libremente. · 8, 23, 65, 148

OpenLayers.- OpenLayers es una biblioteca de JavaScript de código abierto bajo una derivación de la licencia BSD para mostrar mapas interactivos en los navegadores web. · 2, 65, 66, 67, 68, 69, 70, 71, 72, 73, 75, 76, 77, 78, 79, 85

P

Persistencia.- Se entiende por persistencia (en programación) como la acción de preservar la información de un objeto de forma permanente (guardar), pero a su vez también se refiere a poder recuperar la información del mismo (leer) para que pueda ser nuevamente utilizada. · 35, 36, 37, 41, 42

PostGIS.- PostGIS es un módulo que añade soporte de objetos geográficos a la base de datos objeto-relacional PostgreSQL, convirtiéndola en una base de datos espacial para su utilización en Sistema de Información Geográfica. · 2, 10, 21, 23, 24, 25, 28, 32, 80, 117, 121, 123, 127

PostgreSQL.- Es un Sistema de Gestión de Base de Datos r elacional orientado a objetos y libre, publicado bajo la licencia BSD. · 10, 21, 22, 23, 24, 25, 32, 80, 110, 111, 115, 116, 123, 127

R

RDBMS.- Sistema de Gestión de Base de Datos Relacional. · 21, 24, 32

S

Servidor.- Es un nodo que forma parte de una red, provee servicios a otros nodos denominados clientes. · 8, 19, 20, 24, 25, 31, 34, 42, 45, 46, 47, 48, 49, 50, 51, 53, 54, 55, 56, 60, 61, 62, 64, 65, 79, 80, 84, 88, 103, 106, 107, 108, 140, 148

Servlets.- Los servlets son objetos que corren dentro y fuera del contexto de un contenedor de servlets (ej: Tomcat) y extienden su funcionalidad. · 10, 19, 20, 31, 33, 36, 84

Shell.- Se emplea para referirse a aquellos programas que proveen una interfaz de usuario para acceder a los servicios del sistema operativo. · 20

SMS.- Servicio de mensajes cortos o SMS (Short Message Service) es un servicio disponible en los teléfonos móviles que permite el envío de mensajes cortos (también conocidos como mensajes de texto, o más coloquialmente, textos) entre teléfonos móviles · 12

Software.- Equipamiento lógico o soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware. · 1, 2, 3, 4, 5, 6, 7, 8, 10, 12, 13, 19, 21, 24, 31, 36, 48, 49, 55, 65, 81, 89, 147, 148, 149

Software Libre.- es la denominación del software que respeta la libertad de todos los usuarios que adquirieron el producto y, por tanto, una vez obtenido el mismo puede ser usado, copiado, estudiado, modificado, y redistribuido libremente de varias formas. · 4, 5

SQL.- El lenguaje de consulta estructurado o SQL (por sus siglas en inglés structured query language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. · 21, 22, 23, 24, 27, 35, 41, 92

SRID.- El SRID corresponde a un sistema de referencia espacial basado en el elipsoide concreto usado para la creación de mapas de tierra plana o de tierra redonda. · 26, 27, 28, 29

T

TCP.- Transmission Control Protocol (en español Protocolo de Control de Transmisión) o TCP, es uno de los protocolos fundamentales en Internet. · 48, 49, 50, 51, 52

Tomcat.- Servidor web y contenedor de servlets. · 10, 20, 84, 124, 125, 126

U

UDP.- User Datagram Protocol (UDP) es un protocolo del nivel de transporte basado en el intercambio de datagramas (Encapsulado de capa 4 Modelo OSI). · 48, 49, 50, 52, 53, 54, 55

URL.- Un localizador de recursos uniforme, más comúnmente denominado URL (sigla en inglés de uniform resource locator), es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación. · 20, 41, 51, 64

V

VPN.- Una red privada virtual, RPV, o VPN de las siglas en inglés de Virtual Private Network, es una tecnología de red que permite una extensión segura de la red local sobre una red pública o no controlada como Internet. · 10

VPS.- Una partición dentro de un servidor que habilita varias máquinas virtuales dentro de dicha máquina por medio de varias tecnologías. · 8, 10, 110, 147

W

WAR.- Archivo Web (sigla en inglés de Web Archive) que se utiliza para desplegar una aplicación web basada en la plataforma Java. · 20

X

XHTML.- Siglas del inglés eXtensible HyperText Markup Language. XHTML es básicamente HTML expresado como XML válido. Es más estricto a nivel técnico, pero esto permite que posteriormente sea más fácil al hacer cambios o buscar errores entre otros. · 19, 31, 34, 60, 64

XML.- Siglas en inglés de eXtensible Markup Language ('lenguaje de marcas extensible'), es un lenguaje de marcas desarrollado por el World Wide Web Consortium (W3C). Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes. · 19, 20, 34, 41, 59, 61

XP.- La programación extrema o eXtreme Programming (XP) es una metodología de desarrollo de la ingeniería de software. · 2, 13, 14, 81

REFERENCIAS BIBLIOGRÁFICAS Y DE INTERNET

- Buyya, R., Selvi, S. T., & Chu, X. (2009). *Object-Oriented Programming with Java*. Melbourne: McGraw Hill.
- Das, A. (17 de Enero de 2008). *JAVAWORLD*. Recuperado el 24 de Diciembre de 2012, de <http://www.javaworld.com/javaworld/jw-01-2008/jw-01-jpa1.html>
- Higuera, S. (7 de Junio de 2010). *Manual OpenLayers*. Obtenido de <http://openlayers.ingemoral.es/manualOpenLayers.html>
- Martín Sierra, A. J. (2011). *AJAX en JAVA EE*. Madrid: RA-MA Editorial.
- PostgreSQL Global Development Group. (2011). *About*. Recuperado el Julio de 2012, de <http://www.postgresql.org/about/>
- Solivelles, U. (2012). *Sistemas de Información Geográfica (SIG) y percepción remota*.
- The Apache Software Foundation. (2012). *Apache Tomcat*. Recuperado el 10 de Julio de 2012, de <http://tomcat.apache.org/>
- Zakas, N. C. (2006). *Profesional JavaScript para desarrolladores Web*. Madrid: Ediciones Anaya Multimedia.

ANEXOS

ANEXO I: Diagrama de Clases UML del Módulo de Comunicaciones

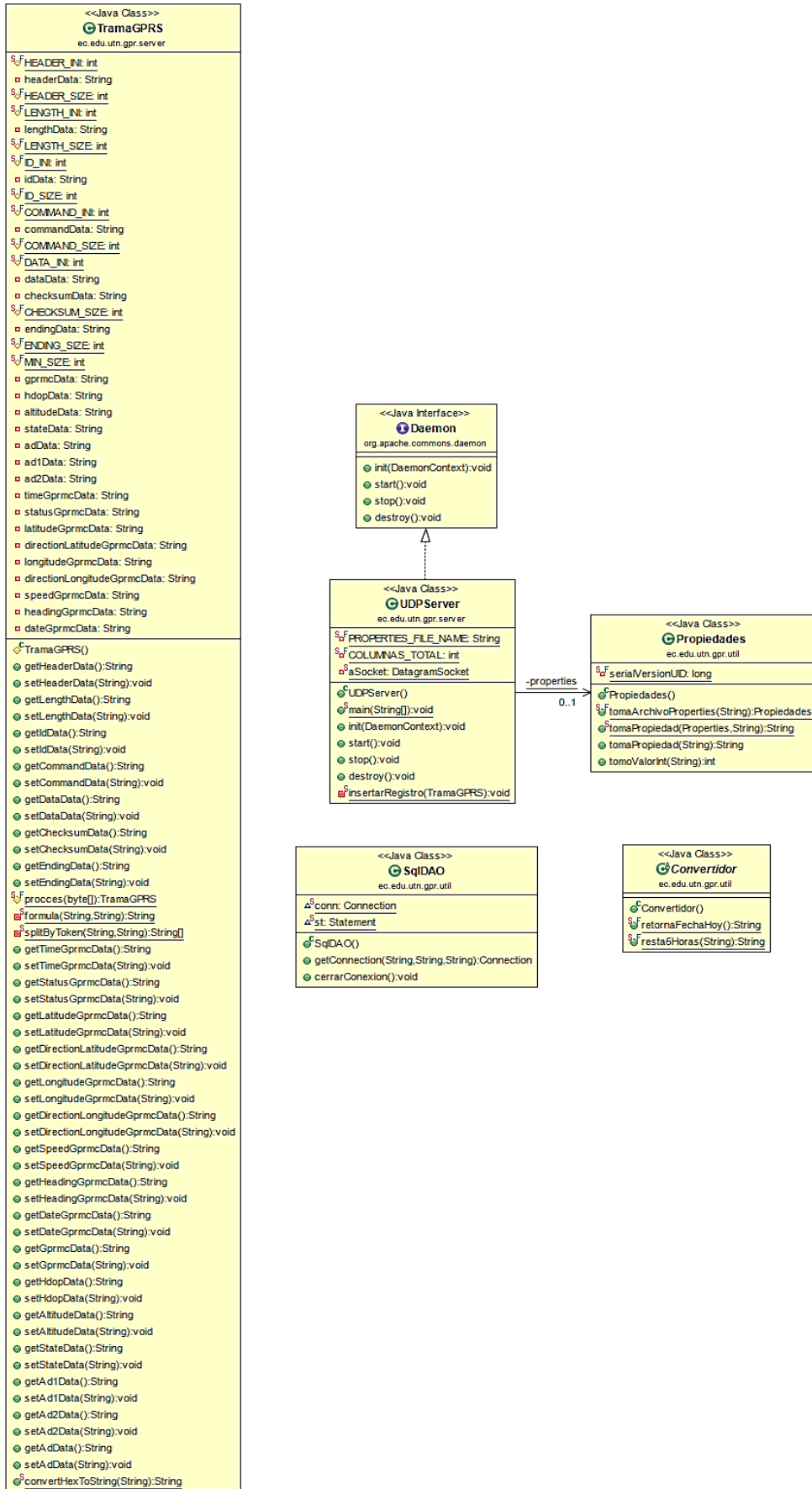


Ilustración 3.1. Diagrama de Clases UML del Módulo de Comunicaciones. Fuente propia.