Universidad de las Ciencias Informáticas

Facultad 6

Desarrollo de una solución arquitectónica base e ingenieril para la concesión de un SIG con la herramienta Open Jump

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE INGENIERO EN CIENCIAS INFORMÁTICAS

Autores: *Isaura Tasé Figueredo.*

Liuver Romel Sañudo Ortiz.

Tutor: *Ing. Alejandro Orgelio Hernández Cebrián.*

Ciudad de La Habana, Cuba.
Año del 53 Aniversario del Triunfo de la Revolución.

La magnitud de lo que logramos

No depende de lo que tenemos,

Sino de lo que seamos capaces de hacer.

Dedicatoria

A mis padres por su consagración y su amor inmoderado, a mi padre por nunca haber perdido la confianza en mí y ser el ejemplo que siempre seguí, a mí madre por mostrarme el camino correcto y por haber impregnado en mi corazón sentimientos tan nobles...

A mis hermanos para que les sirva de ejemplo y vean que no hay cosas imposibles, sino hombres incapaces.

A mi abuelita mami Aleida a la cual las palabras más dulces de mi alma no podrían describir la ternura y delicadeza que ella guarda en sí y a mi abuelito papi Quiko por su experiencia y ejemplo.

A mi novio por estar siempre a mi lado y ser guía en medio de este hermoso pero arduo sendero, para que le sirva de muestra no solo de sacrificio sino de alegría por el deber cumplido.

A todos mis familiares y Amigos.

Isaura.

Dedicatoria

Dedicatoria Liuver

A las dos personas que sin dudar un segundo dieran todo a su alcance para hacer de mí, un mejor hombre:

….. *Mis padres*….

Agradecimientos

A manera de homenaje y no de agradecimiento: a mis padres, por depositar todos sus esfuerzos en pos de hacerme una persona pura y segura de lo que quiero. Por estar conmigo a lo largo de todos estos años de estudio, apoyándome, corrigiéndome pero por encima de todo respetando mis decisiones en cada paso que di. Por darme su amor incondicional y sacrificio inmedido. A mi papá por confiar siempre en mí, en todas las cosas que he hecho. A mi mamá por su perseverancia y fidelidad, aunque nunca se conformó con verme en algún momento tropezar. A mis hermanitos lindos: Erich, por ser tan dulce conmigo aunque muchas veces no nos podíamos ni ver, José Roberto a pesar de ser muy pequeñito siempre ese amor que solo él sabe dar me hace sentir cada día un poquito más conforme de ser no solo su hermana mayor, sino un ejemplo para cuando crezca y a Ernestico que aunque no sabe casi quien soy, se que también estará feliz de que este acontecimiento este pasando. A mi súper abuela Aleida por todo el cariño, confianza y malcrío que siempre me dio. A mi abuelo Quiko por esa disposición siempre de ir a verme en cada beca, por estar y ser mi mayor orgullo en la inolvidable graduación de bachiller, por apoyarme, comprenderme, por tener ese carácter tan fuerte y ser el ejemplo que siempre envidié. A mis abuelos Francisca y Wilber por confiar en que si se puede. A mis tíos lindos que siempre me ayudaron y apoyaron Carmen, Camilo, Jorge, Isabel, Ernesto, Marisleidis, Ato y Mirta, Xiomara y Nani. A mis queridos suegros que los adoro muchísimo por todo lo dulce, comprensivos, atentos y cariñosos, por su apoyo y confianza, por convencerme con hechos de que las cosas se luchan y que nada es imposible. Por ser los progenitores de mi niño lindo (Roger), quien ha sido más que novio, amigo y compañero, presente en cada uno de los contratiempos que tuve a lo largo de la realización del presente trabajo, así como en cada alegría. Muchas gracias mi amor por ser siempre tan dispuesto. A mis amigos Yamile (mi hermanita), Lisandra, Danae, Gerardo, Ángel, Aksana y Thais por su amistad sin condiciones ni intereses, por sacarme de apuros tantas veces. A mi tribunal… Más bueno no puede ser, Nilberto por no cansarse de atenderme, Yoandri por su disposición, Héctor que casi no estuvo vinculado al proceso completo, pero nunca nos dejo de tener bien en cuenta a la hora de una crítica. A mi tutor querido Alejandro, “mejor ni mandado a hacer”, por toda la dedicación, paciencia, innovación, pero por sobre todo por no dejarnos solos ni un instante, de todo corazón: GRACIAS.

Agradecimientos

Agradecimientos Liuver

Primero, que todo a mis maravillosos padres que siempre me han apoyado y guiado, con su amor y dedicación en todos los momentos de mi vida.

A mi hermanito, por estar siempre ahí y darme ese cariño tan preciado.

Agradezco a toda mi familia en general, por también seguir mis pasos desde pequeño guiándome a través de los años.

Agradezco al amor por haber llegado a mi vida en esa forma tan hermosa, con esa ternura y ese dedicación que me has dado en todo este tiempo.. Lilibeth

Agradezco a mis amigos, hermanos, por compartir conmigo en las buenas y en las malas, siempre estarán conmigo.

Agradezco a nuestro tutor, Alejandro por habernos apoyado cuando de verdad lo necesitábamos.

Agradezco a nuestro tribunal, Nilberto, Yoandris, Amador y Yoandri por siempre darnos esas valiosas críticas y recomendaciones que lograron el desarrollo exitoso de nuestro trabajo.

Agradezco a nuestro oponente Hector René que a pesar de llegar tarde supo aconsejarnos cuando lo necesitábamos.

Agradezco a todos los demás que han hecho posible la culminación de nuestra Tesis.

Declaración de Autoría

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del presente trabajo y autorizamos a la Universidad de las Ciencias Informáticas (UCI) a que haga uso del mismo de la manera que mejor estime.

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

Isaura Tasé Figueredo

_____

Firma de la autora

Liuver Romel Sañudo Ortiz

_____

Firma del autor

Ing. Alejandro Orgelio Hernández

_____

Firma del Tutor

Resumen

El software SIG hace que sea imprescindible contar con herramientas adecuadas y necesarias para su desarrollo. El presente trabajo de diploma está enfocado en el análisis y diseño de una plataforma de desarrollo libre y de escritorio que pueda ser aplicable a diferentes necesidades de negocio, así como el desarrollo de una Arquitectura flexible y robusta teniendo en cuenta el nivel de importancia de las funcionalidades de representación geográfica, basado principalmente en la solución existente: OpenJump.

Para el desarrollo del sistema propuesto se analizaron las necesidades generales y actuales principalmente de las empresas e instituciones involucradas en acciones referentes a recursos minerales, hidráulicos, de orden social así como muchos otros. Se tuvieron en cuenta las metodologías y tecnologías actuales, utilizando como Lenguaje Unificado de Modelamiento (UML) para construir modelos por ingeniería inversa a partir de sistemas existentes**.** Y Visual Paradigm como herramienta CASE para el trabajo con UML, entre otras características soporta el ciclo de vida del desarrollo del software. Al concluir se presentan los principales artefactos generados durante el proceso de desarrollo, siguiendo la metodología seleccionada, que le dan solución al problema existente y posibilitan un mejor entendimiento de este para futuras versiones, además se da una propuesta de funcionalidad nueva, que posibilita nuevas estrategias de trabajo, así como una mejor manipulación de la herramienta.

Palabras Claves: SIG, OpenJump.

Índice

Índice

Introducción

Un Sistema de Información Geográfica(SIG) 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. En el sentido más estricto, es cualquier sistema de información capaz de integrar, almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada (Berry, 1993.). Los SIG son herramientas que permiten a los usuarios crear consultas interactivas, analizar la información espacial, editar datos, mapas y presentar los resultados de todas estas operaciones. Los SIG han llegando a ser un nuevo medio de los humanos para entender, comunicar y colaborar. En la última década el desarrollo de los Sistemas de Información Geográfica ha sido acelerado, puede ser utilizada para investigaciones científicas, la gestión de los recursos, gestión de activos, la arqueología, la evaluación del impacto ambiental, la planificación urbana, la cartografía, la sociología, la geografía histórica, el marketing y la logística por nombrar unos pocos.

Por ejemplo, un SIG podría permitir a los grupos de emergencia calcular fácilmente los tiempos de respuesta en caso de un desastre natural, puede ser usado para encontrar los humedales que necesitan protección contra la contaminación, o pueden ser utilizados por una empresa para ubicar un nuevo negocio y aprovechar las ventajas de una zona de mercado con escasa competencia.

Sin embargo los Sistemas de Información Geográfica han sido poco explotados como técnica válida en el proceso educativo. El manejo de estos escenarios para el estudio de distintas ramas de la educación puede ser un gran apoyo para el aprendizaje del estudiante.

Los SIG en fueron utilizados por primera vez en Ottawa (Ontario, Canadá) a cargo del Departamento Federal de Silvicultura y Desarrollo Rural, empleándolo para almacenar, manipular y analizar datos recogidos para el inventario de tierras de Canadá. Este SIG era como los que se conocen hoy en día puesto que permitía superponer capas de información, realizar mediciones y llevar a cabo digitalizaciones y escaneos de datos. En las décadas posteriores, organizaciones como ESRI (*Environmental Systems Research Institute)* y CARIS (*Computer Aided Resource Information System*), emergieron como

proveedores comerciales de softwares Sistemas de Informaciones Geográficos, abriendo paso a una etapa comercial para profesionales donde los SIG. Más reciente se ha visto una expansión en el número de desarrollos de software SIG de código libre, los cuales, a diferencia del software comercial, suelen abarcar una gama más amplia de sistemas operativos, permitiendo ser modificados para llevar a cabo tareas específicas.

Las expectativas erigidas sobre SIG están también presentes en Cuba con sus correspondientes limitaciones y paradojas. El primer SIG utilizado en Cuba fue el MAP ANALYSIS PACKAGE, Versión 1.0, 31 de Agosto de 1987 (OSU-MAP), sistema elaborado para Computadoras Personales IBM, desarrollado por la Universidad de Ohio a partir de la versión PC-MAP de la Universidad de Harvard, basado en los trabajos de Dana Tomlin. Toda la información que se introducía en los softwares posteriores utilizados era mediante el programa "Digicapt", módulo elaborado por los cibernéticos cubanos para el Sistema de Información Geográfica de Cuba (SIGC), el cual tenía como objetivo principal actualizar el nuevo Atlas Nacional, tomándolo como fuente inicial de datos y receptor final. Este SIGC era llamado el "SIG Frankestein" el cual estaba compuesto por 15 módulos, 10 de ellos creados por los especialistas cubanos. Otros más lo siguieron como el IDRISI para Windows y el ILWIS donado por la universidad de Holanda, hasta que se creó TELEMAP, el primer SIG desarrollado completamente en Cuba, entre los muchos que se utilizaron a partir de ese momento (Yosleny Galvez Leal, 2007).

Dadas estas condiciones y características que posee el universo de los SIG, en la Universidad de las Ciencias Informáticas se desarrollan una serie de proyectos que si bien no todos están enmarcados directamente con este tema de la geografía, sí muchos desarrollan sus productos basándose en características muy propias de este tema y fundamentándose en que este es un mundo muy complejo y en el cual se necesita un trabajo integrado y conjunto.

Para el logro de estos objetivos es necesario que exista una biblioteca que cuente con los mapas, herramientas técnicas, óptimas y necesarias con las cuales se pueda desarrollar un entorno, que basándose y contando con dichas técnicas y herramientas sea capaz de mantener la información de los planos mostrados.

Introducción

En estos días existe un número creciente de herramientas para el análisis y la realización de operaciones de incorporación topológica. Pero la mayoría de ellas son privadas, muy caras o no cumplen de forma general con todos los aspectos y funcionalidades que desea la Universidad de las Ciencias Informáticas para poder insertarse con fuerza en este creciente mercado. Como estrategia comercial, se dio a la tarea de comenzar el análisis y diseño de una plataforma de desarrollo libre y de escritorio que pueda ser aplicable a diferentes necesidades de negocio, así como el desarrollo de una arquitectura flexible y robusta teniendo en cuenta el nivel de importancia de las funcionalidades de representación geográfica, basado principalmente en la solución existente: OpenJump.

Por lo planteado anteriormente se tiene como **problema a resolver** la siguiente interrogante**:**

¿Cómo dar respuesta a clientes interesados en Sistemas de Información Geográfica (SIG) de escritorio sobre la herramienta libre OpenJump, en el Departamento Geoinformática de la Universidad de las Ciencias Informáticas?

El **objeto de estudio** de la investigación se encuentra delimitado en el proceso de desarrollo del análisis, diseño y arquitectura de un Sistema de Información Geográfica (SIG) desarrollado a partir de la herramienta OpenJump.

Para darle solución a la problemática se propone como **objetivo general** el desarrollo de una solución arquitectónica base e ingenieril para la concesión de un SIG con la herramienta Open Jump.

El **campo de acción** se enmarca en el análisis, diseño y arquitectura de Sistemas de Información Geográfica de escritorio con la herramienta OpenJump. A partir de aquí se propone la siguiente **idea a defender:**

El desarrollo de una solución arquitectónica base e ingenieril para la concesión de un SIG con la herramienta OpenJump, ayudará a dar respuesta a clientes interesados en SIG de escritorios sobre dicha herramienta.

Para dar cumplimiento a los objetivos antes mencionados se desarrollaron las **tareas de investigación** que se describen a continuación:

- Realizar el diseño teórico y metodológico de la investigación.
- Describir los conceptos asociados al dominio del problema.
- Caracterizar las tendencias actuales en el desarrollo de los SIG.
- Seleccionar y argumentar las tendencias y tecnologías actuales a utilizar en el proceso.
- Realizar la documentación técnica correspondiente al modelo de dominio del Sistema de Información Geográfica: OpenJump.
- Capturar los requisitos funcionales y no funcionales.
- Realizar la documentación técnica correspondiente al modelo de casos de uso del Sistema de Información Geográfica: OpenJump.
- Diseñar el sistema.
- Caracterizar la Arquitectura de Software de la solución OpenJump.
- Desarrollar la Arquitectura de Software ajustada a cada uno de los niveles identificados.

En el desarrollo de la presente investigación se tendrán en cuenta una serie de métodos científicos los cuales se exponen a continuación:

Entre los **Métodos Teóricos** que se emplearán, se encuentra el **Histórico Lógico**, este se utilizó para el análisis histórico de los procesos de gestión de información y las características de los principales servicios geológicos del mundo.

Dentro de los **Métodos Empíricos** se utilizó el método de **Modelación** para el modelado de todos los diagramas presente en la metodología de desarrollo de software que se seleccionó y el **Análisis de documentos** el cual será utilizado fundamentalmente para analizar y estudiar documentos necesarios en el proceso de investigación.

El trabajo se divide fundamentalmente en 4 capítulos:

Introducción

En el capítulo 1 se realizará una caracterización general del objeto de estudio de la investigación, y se abordará sobre algunas soluciones existentes. Se deben especificar las herramientas y/o plataformas tecnológicas (hardware, software, equipamiento, etc.) necesarios para instalar la solución tomando en cuenta los requerimientos macros del SIG.

En el capítulo 2 se realizará un levantamiento de los requisitos funcionales y no funcionales del sistema con el fin de lograr los objetivos planteados. A partir de la recopilación de los requisitos funcionales construir el Modelo de Casos de Uso del Sistema.

En el capítulo 3 se realizarán los diagramas de colaboración y diagramas de secuencia del diseño correspondientes a los casos de uso identificados. Construcción del Diagrama de Despliegue y Modelo de implementación.

En el capítulo 4 y final, se realizarán todos los artefactos relacionados con la creación de una solución arquitectónica propuesta.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

En este capítulo se aborda en detalle todo lo relacionado con la fundamentación teórica, con el objetivo de definir los elementos teóricos para la realización de la investigación. Se especifican algunos conceptos asociados a la investigación, al igual que se profundiza en el Objeto de Estudio y la Situación Problemática así como la realización de su descripción detallada. A partir de todos estos elementos se desarrolla la investigación, haciéndose más sencilla la comprensión de los temas que serán abordados en lo adelante.

1.2 Conceptos asociados al dominio del problema

A continuación se reflejan y explican algunos conceptos relacionados al dominio del problema para un mayor entendimiento.

Código Libre es el código en el cual se respeta la libertad de los usuarios sobre un producto adquirido, una vez obtenido dicho código puede ser cambiado a gusto del usuario y posteriormente redistribuido libremente. El software libre puede estar disponible gratuitamente, pero no tiene que ser necesariamente así, en ocasiones solo incluye el código libre. La revolución que ha supuesto la filosofía del software libre ha llegado también a los sistemas de información geográfica y hoy en día existen ya varios proyectos serios que pretenden ser una alternativa viable a programas comerciales como ArcView, Geomedia o IDRISI (Miguel Montesinos Lajara, 2005.).Cuando se habla de este tema es mejor evitar los términos regalar o gratuito ya que estos implican que la cuestión pase por el precio del producto y es todo lo contrario, es la palabra libertad en todos los sentidos.

Multiplataforma es un término usado para referirse a los programas, lenguajes de programación, u otra clase de software, que puedan funcionar en diversas plataformas. Una plataforma es una combinación de hardware y software usada para ejecutar aplicaciones; en su forma más simple consiste únicamente de un sistema operativo, una arquitectura, o una combinación de ambos. El software en general está escrito de modo que dependa de las características de una plataforma particular; bien sea el hardware, sistema

operativo, o máquina virtual en que se ejecuta. La plataforma java es una máquina virtual multiplataforma, tal vez la más conocida de este tipo, así como una plataforma popular para hacer software (que, por supuesto, se considera multiplataforma) (Christl, 2004.).

Metadatos: El término metadatos no tiene una definición única. Según la definición más difundida metadatos constituyen “datos sobre datos”. Debido a que muchas veces no se tiene en cuenta la diferencia entre dato e información también hay muchas declaraciones como “informaciones sobre datos”, “datos sobre informaciones” e “informaciones sobre informaciones”(Oliva, 2006). Al final estos pueden incluir información descriptiva sobre el contexto, calidad y condición o características del dato.

El servicio Web Map Service (WMS): Produce mapas de datos referenciados espacialmente, de forma dinámica a partir de información geográfica. Este estándar internacional define un "mapa" como una representación de la información geográfica en forma de un archivo de imagen digital conveniente para la exhibición en una pantalla de ordenador. Los mapas producidos por WMS se generan normalmente en un formato de imagen como PNG, GIF o JPEG, y opcionalmente como gráficos vectoriales en formato SVG (Gráficos de Vectores Escalables) o WebCGM (Web Computer Graphics Metafile).

Formatos Ráster: Son imágenes rasterizadas, también llamados mapa de bits, imagen matricial o bitmap, son una estructura o fichero de datos que representa una rejilla rectangular de píxeles o puntos de color, denominada ráster, que se puede visualizar en un monitor de ordenador, papel u otro dispositivo de representación. A las imágenes rasterizadas se las suele caracterizar por su altura y anchura (en píxels) y por su profundidad de color (en bits por píxel), que determina el número de colores distintos que se pueden almacenar en cada píxel, y por lo tanto, en gran medida, la calidad del color de la imagen.

Plugin (complemento): Es una aplicación que se relaciona con otra para aportarle una función nueva y generalmente muy especifica. Esta aplicación adicional es ejecutada por la aplicación principal e interactúan por medio de la API. También se lo conoce como plug-in (del inglés "enchufable"), add-on (agregado), complemento, conector o extensión.

Interfaz de programación de aplicaciones o API (del inglés Application Programming Interface): Es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Son usados generalmente en las librerías.

1.3 Objeto de Estudio

El objeto de estudio del presente trabajo se encuentra delimitado en el proceso de desarrollo del análisis, diseño y arquitectura de un Sistema de Información Geográfica (SIG) desarrollado a partir de la herramienta OpenJump.

1.3.1 Descripción General

Hoy en día el desarrollo de Sistemas de Información Geográfica libres en el mundo y en el país está presentando un vertiginoso avance, en el último caso teniendo gran protagonismo la Universidad de las Ciencias Informáticas y sus proyectos en el centro de Geoinformática, en sus dos soluciones, aplicaciones web y desktop, en cuanto a estas últimas surge un proyecto propio de este centro derivado de un FOSS (Free and Open Source Software), llamado OpenJump.

El OpenJump es demandado hoy en día por clientes alrededor del globo debido a que es un software de código libre, cuya licencia concede el derecho de los usuarios de usar, estudiar, cambiar y mejorar su diseño a través de la disponibilidad de su código fuente (Jorge Gaspar Sanz),dicho software, desarrollado por la empresa Vivid Solutions, a partir de su creación ha tenido una serie de actualizaciones o nuevas versiones llegando hasta la 1.4.0.2, la cual, mediante un estudio profundo entre sus otras versiones llegó a ser, la más adecuada para poder crear un nuevo Sistema de Información Geográfica basado en estas características pero con nuevas funcionalidades.

Recopilando todas las características y funcionalidades principales de las versiones anteriores e incluso de los otros SIG libres de escritorio y comprobando su utilidad, se logra crear un punto de partida que junto a las verdaderas necesidades que se reflejan en la investigación, conformaron una guía a seguir para la implementación del SIG. Comenzando por lo básico se tiene que este software brinda la

posibilidad de cargar y visualizar datos, trabajar con datos alfanuméricos, crear mapas temáticos, exportar vistas, imprimir cualquier vista en que se trabaja, consultar y analizar datos, herramientas para crear, dibujar y editar datos geométricos, así como avanzadas herramientas de selección. Posee un extractor de segmentos el cual elimina los segmentos que aparecen varias veces, exportación de imágenes en formato de gráficos vectoriales escalables (SVG), la opción para mostrar la escala del mapa real, la copia de objetos incluyendo los atributos, el análisis y unión de capas pudiendo unirlas individualmente o una colección como en el software similar Sky Jump.

En cuanto a las capacidades de edición este software posee herramientas para autocompletar polígonos así como la marcación de puntos de interés, auto asignar atributos, conectar y extender geometrías de línea. Por otra parte, si de análisis espacial se trata, se pueden obtener estadísticas básicas de atributos (max, min, media, etc), obtener mapas con diferentes métodos de clasificación y temas de colores (clasificación jensen, método de maximal), además una herramienta más robusta de búsqueda como la "*Search All Atrtibutes"*(Miguel Montesinos Lajara, 2005), ejecutando análisis ráster y cargando directamente las imágenes con OpenJump.

Utiliza el gestor de base de datos PostgreSQL con extensión PostGIS, que alcanza la capacidad de almacenar, consultar y acceder a información geoespacial, de realizar operaciones de análisis geográfico, además utiliza la extensión pgRouting que puede emplear como un potente motor de cálculo de rutas y otras operaciones geoespaciales. Teniendo también gran cantidad de plugins o sea nuevos módulos con funcionalidades que robustecen el sistema, facilitando y diversificando el trabajo. Sobre esta base se va a trabajar y agregar las funcionalidades que según un estudio realizado necesita dicha versión. Al hacer el estudio de este sistema tanto de análisis como arquitectónico se llegó a la conclusión de la utilidad para una nueva versión de funcionalidades que debe poseer, como son, una potente herramienta de selección totalmente nueva, como lo es **tematizar**, la cual dibuja mapas de distintos colores dependiendo del criterio del usuario mediante una consulta que realiza. Una interfaz de usuario mucho más amigable, que cuente con colores y de ayuda para los usuarios pioneros en el tratado de la herramienta, así como un módulo de seguridad donde se gestione la seguridad del sistema y la posibilidad de exportar las vistas en las que se trabaja en formato JPG

1.4 Análisis de otras soluciones existentes

Como resultado de una ardua investigación tanto en materiales bibliográficos como en consultas a profesionales que incursionan en este tipo de trabajos, han sido identificados una serie de Sistemas de Información Geográfica, FOSS (*Free and Open Source Software*), que a la vez son clientes pesados de escritorio, cuyas funcionalidades están basadas en obtener, guardar, manipular, analizar y desplegar la información geográfica con el fin de resolver problemas complejos. Para llegar a OpenJump hay que seguir un camino largo de SIGs libres a través de la historia, lo cual se ve explícitamente en el Anexo1.

Como uno de los SIG de escritorio más fuertes y difundidos por todo el mundo se encuentra **GRASS**, un SIG con capacidad ráster y vectorial. Además cuenta con sistemas integrados de visualización de datos y de procesado de imágenes. GRASS incluye más de 350 módulos para la gestión, procesado, análisis y visualización de datos georreferenciados. GRASS está amparado por GNU GPL desde 1999. Esta licencia protege a los contribuidores de GRASS frente al uso de sus contribuciones en proyectos propietarios que no permiten el libre acceso a su código fuente. La GPL asegura que todo el código publicado basado en código GPL, solo pueda ser publicado a su vez bajo licencia GPL.

GRASS es un SIG modular con datos organizados como coberturas ráster, vectoriales y de puntos. Proporciona una gran variedad de herramientas que permiten clasificarlo como un SIG de altas prestaciones (2006). A continuación se resumen las principales funcionalidades de **GRASS:**

- Integración de datos espaciales
- Procesado de datos ráster
- Procesado de datos vectoriales
- Modelado y simulaciones
- Soporte para datos temporales
- Procesado de datos 3D

- Enlaces con otras herramientas
- Visualización
- Procesado de datos puntuales
- Procesado de imágenes

La versión 5.0 de GRASS tiene un soporte limitado para datos temporales y 3D, sin embargo, la versión experimental 5.7 (la versión 5.7.0 ha sido liberada recientemente en Junio de 2004) está siendo diseñada como un SIG 3D completo con soporte para datos ráster 3D, datos vectoriales 3D y datos puntuales 3D. GRASS soporta la importación y exportación de los formatos más conocidos tanto ráster como vectoriales.

GRASS (La Interfaz de usuario)

Aunque GRASS puede funcionar bajo sistemas operativos Windows NT, 2000 o XP (utilizando cygwin, que es un entorno Linux para Windows), para asegurar el correcto funcionamiento es aconsejable utilizar la versión para GNU/Linux.

Cada capacidad o funcionalidad de GRASS se materializa con el correspondiente comando en modo consola, lo que representa una desventaja para el usuario final y más aún si no tiene conocimientos de LINUX. Aunque es cierto que se ha implementado un entorno de ventanas con TCL/TK o lenguaje de herramientas de comando no deja de ser una solución intermedia ya que no ofrece la potencialidad de utilizar un GUI (Graphical User Interface) como GNOME o KDE (proyecto de software libre). (Ourense, 2003)

El **Quantum GIS** (**QGIS**) es un Sistema de Información Geográfica (SIG) de código libre para plataformas GNU/Linux, Unix, Mac OS y Microsoft Windows. Permite manejar formatos ráster y vectoriales, así como bases de datos (2009).

Las principales características del **Quantum GIS** son:

- Soporte para la extensión espacial de PostgreSQL, PostGIS.

- Manejo de archivos vectoriales Shapefile, ArcInfo coverages, Mapinfo, GRASS GIS, etc.
- Soporte para un importante número de tipos de archivos ráster (GRASS GIS, GeoTIFF, TIFF, JPG, etc.).

Una de sus mayores ventajas es la posibilidad de usar Quantum GIS como GUI del SIG GRASS, utilizando toda la potencia de análisis de este último en un entorno de trabajo más amigable. QGIS está desarrollado en C++, usando la biblioteca Qt para su Interfaz gráfica de usuario, aunque no posee un modulo de seguridad (2010).

GvSIG Desktop es un programa informático para el manejo de información geográfica con precisión cartográfica que se distribuye bajo licencia GNU GPL v2 (2003). Permite acceder a información vectorial y rasterizada así como a servidores de mapas que cumplan las especificaciones del OGC (*Open Geospatial Consortium*). Ésta última es una de las principales características de gvSIG respecto a otros Sistema de Información Geográfica, la importante implementación de servicios OGC: WMS (Web Map Service), WFS (Web Feature Service), WCS (Web Coverage Service), Servicio de Catálogo y Servicio de Nomenclador.

Está desarrollado en lenguaje de programación Java, funcionando con los sistemas operativos Microsoft Windows, Linux y Mac OS X, y utiliza librerías estándares de GIS reconocidas, como Geotools o Java Topology Suite (JTS). Por otra parte, gvSIG posee un lenguaje de scripting basado en Jython y también se pueden crear extensiones en Java utilizando las clases de gvSIG.

Funcionalidades gvSIG Desktop

En gvSIG Desktop se encuentran las herramientas propias de un completo cliente SIG de escritorio, entre otras:

- **Acceso a formatos vectoriales**: SHP, GML, KML, DXF, DWG DGN
- **Acceso a formatos ráster**: BMP, GIF, TIF, TIFF, JPG, JPEG, PNG, VRT, DAT de ENVI, ERDAS (LAN, GIS, IMG), PCI Geomatics (PIX, AUX), ADF de ESRI, ILWIS (MPR, MPL), MAP de PC Ráster, ASC, PGM, PPM, RST de IDRISI, RMF, NOS, KAP, HDR, RAW.

- **Acceso a servicios remotos**: OGC (WMS, WFS, WCS, WFS-T, WPS), ArcIMS, Ecwp.
- **Acceso a bases de datos y tablas**: PostGIS, MySQL, ArcSDE, Oracle, JDBC, CSV.
- **Navegación**: zooms, desplazamiento, gestión de encuadres, localizador.
- **Consulta**: información, medir distancias, medir áreas, hiperenlace.
- **Selección**: por punto, por rectángulo, por polígono, por capa, por atributos, invertir selección, borrar selección.
- **Búsqueda**: por atributo, por coordenadas.
- **Geoprocesos**: área de influencia, recortar, disolver, juntar, envolvente convexa, intersección, diferencia, unión, enlace espacial, translación 2D, reproyección (sólo vectorial), geoprocesos Sextante.
- **Edición gráfica**: añadir capa de eventos, snapping, rejilla, flatness, pila de comandos, deshacer/rehacer, copiar, simetría, rotar, escalar, desplazar, editar vértice, polígono interno, matriz, explotar, unir, partir, autocompletar polígono, insertar punto, multipunto, línea, arco, polilínea, polígono, rectángulo, cuadrado, círculo, elipse.
- **Edición alfanumérica**: modificar estructura tabla, editar registros, calculadora de campos.
- **Representación vectorial**: símbolo único, cantidades (densidad de puntos, intervalos, símbolos graduados, símbolos proporcionales), categorías (expresiones, valores únicos), múltiples atributos, guardar/recuperar leyenda, editor de símbolos, niveles de simbología, bibliotecas de símbolos.
- **Representación ráster**: brillo, contraste, realce, transparencia por píxel, opacidad, tablas de color, gradientes.
- **Etiquetado**: etiquetado estático, etiquetado avanzado, etiquetado individual.

- **Tablas**: estadísticas, filtros, orden ascendente/descendente, enlazar, unir, mover selección, exportar, importar campos, codificación, normalización.
- **Constructor de mapas**: composición de página, inserción de elementos cartográficos (Vista, leyenda, escala, símbolo de norte, cajetín, imagen, texto, gráfico), herramientas de maquetación (alinear, agrupar/desagrupar, ordenar, enmarcar, tamaño y posición), grid, plantillas.
- **Impresión**: impresión, exportación a PDF, a Postcript, a formato de imagen (Miguel Montesinos Lajara, 2005).

OpenJump es el resultado del *Jump Pilot Project (JPP), d*esarrollado por Vivid Solutions así como él, surgieron varias versiones más, las cuales se integraban y actualizaban entre ellas constantemente. Por ejemplo **deeJUMP** es mucho más antiguo que OpenJump siendo un completo paquete de software geoespaciales con las implementaciones de OGC Web Services como WMS y WFS, un geoportal, una aplicación de escritorio, mecanismos de seguridad y varias herramientas para el procesamiento de datos geoespaciales y de gestión, pero al final no se completó como un solo programa, en su lugar las funcionalidades se incorporaron a OpenJump a través de dos plugins.

PirolJUMP, esta es la versión final de OpenJUMP 1.3, donde se añadieron varias características nuevas, tales como métodos de clasificación de atributos de color para la tematización, función que genera la capa de un atributo de las estadísticas de tabla (integrado a partir de PirolJUMP), Jython / support script Python como segunda posibilidad, herramientas de análisis de imagen (integrado a partir de PirolJUMP).

Otro proyecto abierto para crear un GIS lo que desarrollado en España es el **Kosmo,** cliente SIG de escritorio de funcionalidades avanzadas, con gestión de capas, múltiples vistas de información, múltiples plantillas de impresión, herramientas de selección cartográfica, medición de longitudes, gestión de escalas, múltiples formatos de datos, educación vectorial avanzada y mucho más (2009). También existen varias versiones pero no de tanta importancia debido a que son más antiguas que la 1.3 como por ejemplo la SIGLEJUMP y AgilesJUMP.

Actualmente no existe un software de este tipo que no sea libre, que permita la gestión de datos. En Cuba no existe un SIG específico que complete todas las funciones que este tipo de herramienta requiere para ser útil en una empresa o institución, como las generadas entre la herramienta base y las nuevas que propone esta investigación.

1.5 Selección de la metodología a utilizar

Cada día la producción del software busca ajustarse más a las necesidades de los usuarios, esto trae como consecuencia que aumente en tamaño y complejidad. Para lograr la productividad del software se necesita un proceso que integre las múltiples facetas del desarrollo del mismo.

En el desarrollo de software la necesidad de organizar o estructurar de forma correcta y disciplinada, es uno de los factores más importantes pues sin esta propiedad los mismos podrían convertirse en verdaderas catástrofes, obteniendo como resultados grandes pérdidas de tiempo y recursos. Para evitar tales errores es preciso definir una estrategia para darle un orden a las tareas posibles a desarrollar, y llevar a cabo una guía de cómo efectuar las actividades, constituyendo esto una metodología de desarrollo de software.

La metodología escogida para este proyecto fue la *Agile Unified Process (AUP),* la cual es una versión simplificada del Proceso Unificado de Rational (RUP) descrito en una forma simple, fácil de entender y brinda un enfoque de desarrollo de software utilizando técnicas ágiles y conceptos del RUP. El ciclo de vida de esta metodología AUP abarca siete flujos de trabajos, cuatro ingenieriles y tres de apoyo: Modelado, Implementación, Prueba, Despliegue, Gestión de configuración, Gestión de proyectos y Ambiente como se muestra en el Anexo2.

- ✓ **Modelo.** El objetivo de esta disciplina es entender el negocio de la organización, el dominio del problema que ocupa el proyecto, y determinar una solución viable para hacer frente al dominio del problema.

- ✓ **Implementación.** El objetivo de esta disciplina es transformar el modelo de dominio en el código ejecutable y llevar a cabo un nivel básico de las pruebas, en las pruebas de unidad en particular.
- ✓ **Prueba.** El objetivo de esta disciplina consiste en realizar una evaluación objetiva para asegurar la calidad. Esto incluye encontrar defectos, validar que el sistema funcione como está previsto, y verificar que se cumplan los requisitos.
- ✓ **Despliegue.** El objetivo de esta disciplina es realizar el plan para la entrega del sistema, así como para ejecutarlo a disposición del usuario.
- ✓ **Gestión de la Configuración.** El objetivo de esta disciplina es administrar el acceso a los artefactos del proyecto. Esto incluye no sólo el seguimiento de versiones de los artefactos a través del tiempo, sino también el control y la gestión de los cambios a los mismos.
- ✓ **Gestión de Proyectos.** El objetivo de esta disciplina es dirigir las actividades se llevan a cabo en el proyecto. Esto incluye la gestión de riesgos, la dirección de personas (la asignación de tareas, el seguimiento de los progresos, etc.), y coordinar con las personas y los sistemas fuera del alcance del proyecto para asegurarse de que se entregue a tiempo y dentro del presupuesto.
- ✓ **Medio Ambiente.** El objetivo de esta disciplina es apoyar el resto de los esfuerzos por garantizar que la orientación adecuada del proceso, (normas y directrices), y herramientas (hardware, software, etc.) están disponibles para el equipo según sea necesario. (Ambler, 2009)

Además dispone de cuatros fases igual que RUP:

- ✓ **Creación.** El objetivo es identificar el alcance inicial del proyecto, una arquitectura potencial de su sistema, obtener la financiación inicial del proyecto y la aceptación de las partes interesadas.

- ✓ **Elaboración.** El objetivo es probar la arquitectura del sistema.

- ✓ **Construcción.** El objetivo es la construcción de software que trabajan en forma regular y gradual que responda a las necesidades de mayor prioridad de las partes interesadas su proyecto.

- ✓ **Transición.** El objetivo es validar y desplegar el sistema en su entorno de producción. (Ambler, 2009).

Este proceso se basa en diferentes filosofías o principios:

- ✓ El personal debe saber lo que están haciendo. Las personas no van a leer la documentación detallada del proceso, pero querrán cierta orientación de alto nivel y / o entrenamiento de vez en cuando. El producto AUP ofrece enlaces a muchos de los detalles, si los usuarios están interesados.
- ✓ Simplicidad. Todo lo que se describe de manera concisa con un puñado de páginas, no miles de ellos.
- ✓ Agilidad. El Agile UP se ajusta a los valores y principios de la Alianza Ágil.
- ✓ Centrarse en actividades de alto valor. La atención se centra en las actividades que en realidad cuenta, no todos los sucesos que podrían presentarse en un proyecto.
- ✓ La independencia de herramientas. Se puede utilizar cualquier conjunto de herramientas que desee con el Agile UP. Pero se sugiere utilizar las herramientas que son las más adecuadas para el trabajo, que a menudo son herramientas simples o incluso herramientas de código abierto.
- ✓ Adaptar siempre el producto para satisfacer las propias necesidades. (Ambler, 2009)

Agile Unified Process (AUP) es calificado como un proceso altamente ceremonioso porque especifica muchas actividades y artefactos involucrados en el desarrollo de un proyecto de software, pero dado que es un marco de procesos, puede ser adaptado ya que es muy semejante a RUP. Se escogió AUP porque propone los mismos roles y artefactos de RUP, pero en una versión simplificada; además solo utiliza los artefactos que son imprescindibles y realmente necesarios para la realización del producto, proporciona un ambiente de desarrollo de software iterativo e incremental. También, AUP adopta muchas de las técnicas ágiles de XP y otros procesos ágiles que aun mantienen la formalidad de RUP.

1.6 Selección de la herramienta CASE (Computer Assisted Software Engineering) a utilizar

Las herramientas de Ingeniería de Software Asistida por Ordenador (CASE, por sus siglas en Inglés, Computer Aided Software Engineering) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software reduciendo el coste de las mismas en términos de tiempo y de

dinero. Estas herramientas pueden ayudar en todos los aspectos del ciclo de vida de desarrollo del software en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores entre otras. (slideshare.net)

Sin lugar a dudas las herramientas CASE han venido a revolucionar la forma de automatizar los aspectos que son imprescindibles en el desarrollo de los sistemas de información, además han sido creadas con una gran exactitud en torno a las necesidades de los desarrolladores de sistemas para la automatización de las fases de desarrollo incluyendo el análisis, diseño e implantación. (csi.map.es)

Visual Paradigm.

Visual Paradigm es una herramienta CASE, para el trabajo con lenguaje de modelado unificado (UML), profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite realizar ingeniería tanto directa como inversa. Además, la herramienta es colaborativa, es decir, soporta múltiples usuarios trabajando sobre el mismo proyecto; genera la documentación del proyecto automáticamente en varios formatos como Web o PDF, y permite control de versiones. Visual Paradigm ofrece: (freedownloadmanager.org)

- Diseño centrado en casos de uso y enfocado al negocio que generan un software de mayor calidad.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo
- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales IDEs.
- Entorno de creación de diagramas para UML 2.0.

Para maximizar la interoperabilidad de los productos de Visual Paradigm con otras aplicaciones, se han introducido opciones de exportación e importación de sus modelos a otras aplicaciones, lo que posibilita que los usuarios y proveedores de tecnología puedan integrar modelos de Visual Paradigm en sus soluciones con un mínimo esfuerzo.

Rational Rose

Rational Rose es una herramienta para modelado visual, que forma parte de un conjunto más amplio de herramientas que juntas cubren todo el ciclo de vida del desarrollo de software. Rational permite completar una gran parte de las disciplinas (flujos fundamentales) del proceso unificado de Rational (RUP), en concreto: (sistemaronald.blogspot.com)

- Captura de requisitos (parcial).
- Análisis y diseño (completo).
- Implementación (como ayuda).
- Control de cambios y gestión de configuración (parte).
- Modelado del negocio.

Características principales de Rational Rose: (sistemaronald.blogspot.com)

- Realiza chequeo semántico de los modelos.
- Admite como notaciones: UML, COM, OMT y Booch.
- Ingeniería "de ida y vuela": Rose permite generar código a partir de modelos y viceversa.
- Desarrollo multiusuario.
- Integración con modelado de dato.

- Generación de documentación.
- Tiene un lenguaje de script para poder ampliar su funcionalidad.
- Soporta OLE.
- Disponible en múltiples plataformas.

Versiones de Rational Rose: (sistemaronald.blogspot.com)

- Professional: incluye soporte sólo para un lenguaje.
- Modeler: no tiene soporte para lenguajes de programación
- Enterprise: múltiples lenguajes, incluyendo C++, Visual C++ (6.0), Visual Basic, Java, CORBA, etc.).

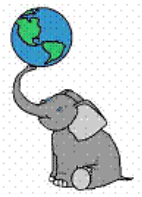
Selección de la Herramienta CASE a utilizar.

La herramienta CASE a utilizar en la investigación es Visual Paradigm, debido a que esta herramienta emplea las últimas notaciones del Lenguaje Unificado de Modelado, ingeniería inversa, generación de código, importación de Rational Rose, exportación/importación XML, generador de impresos e integración con el Visio. Además de lo anterior, soporta aplicaciones web, genera código para el lenguaje Java, y exporta en formato HTML, es una tecnología libre y está disponible en varios idiomas, en conjunto con esto es fácil de instalar y fácil de actualizar. Visual Paradigm admite compatibilidad con las demás versiones, es multiplataforma; además facilita la organización de los diagramas, generando la documentación del proyecto automáticamente en varios formatos como Web o PDF y permite el control de versiones.

1.7 Selección del Gestor de Base de Datos a utilizar

Un *Sistema Gestor de Bases de Datos (SGBD)* permite: definir, crear y mantener las bases de datos, además permite la autonomía entre los datos y los programas de aplicación, minimiza las redundancias, garantiza la integridad, la seguridad y protección de los datos, proporciona la manipulación de la

información, permite un control centralizado, también proporciona un acceso controlado a estas y un lenguaje de manejo de datos para la inserción, actualización, eliminación y consulta de información en estas bases de datos.



El caso de la investigación realizada, es un Sistema de Información Geográfica (SIG) y los datos con los que trabaja y salva información, no importa su extensión, son datos espaciales, por lo que utiliza PostGIS, que no es más que una extensión al sistema de base de datos objeto-relacional PostgreSQL el cual permite el uso de objetos SIG (Sistemas de Información Geográfica). El cual presenta las siguientes características que lo hacen ideal para esta aplicación:

- Alto rendimiento, con una base de datos espacial robusta construida en PostgreSQL
- Permite usar todos los objetos que aparecen en las especificaciones OpenGIS (puntos, líneas, polígonos, multilíneas, multipuntos y colecciones geométricas).
- Puede funcionar con Geometry Engine Open Source (GEOS) como motor de comprobación topológica.
- Permite realizar operaciones de análisis geográfico como lo son la creación y conversión geométrica, generalización, unión, reproyección entre otras.
- Tiene una gran cantidad de plugins (que va en aumento) para todo tipo de operaciones, todas compatibles con todas las versiones del OpenJump.
- Posee herramientas gráficas y líneas de comando para hacer más flexible la gestión.

También incluye:

- Predicados espaciales para determinar las interacciones de geometrías usando la matriz de 3x3 Egenhofer.
- Los operadores espaciales para determinar medidas geo-espaciales tienen gusto de área, de distancia, de longitud y de perímetro.
- Operadores espaciales para determinar operaciones geo-espaciales del sistema, como la unión, la diferencia, la diferencia simétrica y almacenadores intermediarios.
- Ayuda con la selectividad del índice para proporcionar alto rendimiento a los *planes de la pregunta*, para las preguntas espaciales/no-espaciales mezcladas. (WorldLingo, 2011).

Además tiene sus desventajas:

- ✓ Alto coste de implementación.
- ✓ Poca flexibilidad.
- ✓ Incompatibilidad de algunos programas SIG.
- ✓ Más lento que pequeños ficheros en local.

PostGIS se basa en geometrías ligeras y los índices optimizados para reducir la utilización de memoria. La primera versión fue lanzada en el 2001 por Refractions Research bajo la Licencia pública del GNU, y el desarrollo ha continuado desde entonces activamente. Actualmente PostGIS ha sido certificado con las características simples y obedientes para la base de datos del SQL por el consorcio abierto de Geospatial (Briones, 2007).

1.8 Selección del lenguaje de programación a utilizar

Lenguaje de programación no es más que un código artificial que puede ser usado para controlar el comportamiento de una máquina, especialmente una computadora. Estos se componen de un conjunto de reglas sintácticas y semánticas que permiten expresar instrucciones que luego serán interpretadas. (Mitecnológico)

En el caso de este trabajo de diploma solo se puede utilizar un único lenguaje de programación pues tiene como base un software de código libre (OpenJump), y este fue programado en Java, por lo que el lenguaje de la aplicación que se va a desarrollar es Java el cual ha sido concebido desde sus orígenes orientado a objetos. Es un lenguaje totalmente transportable, o sea, que las herramientas que lo utilizan pueden ser transportados de sistema operativo, computadora o entorno, sin necesidad de cambiar el código. Java es sencillo, fácil de aprender, dinámico y se adapta con suma facilidad a los cambios en los componentes fundamentales del lenguaje

Posee un conjunto grande de clases, ordenadas en paquetes, que se pueden usar en los programas. La programación en Java permite el desarrollo de programas de rendimiento seguro y alto en las plataformas

múltiples, esto hace que sea robusto, puesto que no permite el manejo directo de memoria. Java tiene una característica adicional: es multihilos, lo que le permite atender eficientemente varios procesos a la vez.
Los programas escritos en Java pueden ejecutarse en cualquier arquitectura, pues no se genera un programa ejecutable para cada una de éstas, sino un archivo de arquitectura neutral con códigos que no dependen de algún tipo de procesador específico.

1.9 Librería a utilizar

Se utilizará la librería Java Topology Suite (JTS) la cual es una API que proporciona un modelo de objetos espaciales y funciones fundamentales geométricas 2D. JTS ha sido desarrollada por la empresa Vivid Solutions y esta implementada íntegramente en el lenguaje de programación Java, distribuyéndose bajo licencia LGPL.

Cumple con la especificación Simple Features Specification for SQL publicada por el Open Geospatial Consortium la cual define un esquema SQL estándar que soporta almacenamiento, recuperación, consulta y actualización de colecciones simples elementos geoespaciales a través de la API de ODBC(Open DataBase Connectivity, estándar de acceso a bases de datos). Proporciona una implementación completa, consistente y robusta de algoritmos espaciales bidimensionales. Es una librería que resuelve problemas complejos y en la que sus arquitectos han recurrido al uso exhaustivo de patrones de diseño de software como medio de gestionar eficientemente dicha complejidad. El hecho de que JTS haga uso de patrones de diseño es una muestra más de su fiabilidad. (Departamento de proyectos e ingeniería. Los Sistemas de información Geográfica.)

Esta biblioteca es ampliamente utilizada en el software SIG de código libre con funciones de análisis espacial, consultas avanzadas y creación de topología. Existe una versión de esta biblioteca en C++, llamada GEOS.

1.10 Selección del ambiente de desarrollo a utilizar

Un entorno de desarrollo integrado (Integrated Development Environment o IDE) es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar códigos. Ésta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. (Petra.euitio)

Los entornos de desarrollo (IDEs) han incorporado a medida que evolucionan características de integración multi-código y multi-plataforma. Existen muchos entornos de desarrollo, sin embargo, para este trabajo de diploma se seleccionó el NetBeans 6.9 luego de tener en consideración solo dos de ellos: NetBeans 6.9 y Eclipse, que gozan de reconocimiento en el mundo.

Eclipse

Eclipse es un entorno de desarrollo integrado multiplataforma (utilizado para los lenguajes C, C++, Python y Java entre otros) de código abierto, utilizado en su mayoría para desarrollar otros entornos de desarrollo (como el JDT) aunque también puede ser utilizado para desarrollar aplicaciones cliente como Azureus un cliente de BitTorrent (Un programa de ordenador que implementa el protocolo P2P que permite a distintos ordenadores compartir ficheros a través de una red). Fue originalmente desarrollado por IBM para pasar a la familia de herramientas VisualAge que la marca poseía. Sin embargo en la actualidad esta herramienta está siendo desarrollada por la Fundación Eclipse, una organización independiente sin ánimo de lucro que intenta fomentar una comunidad de código abierto así como un conjunto de productos complementarios, capacidades y servicios.

La versión existente en la actualidad de esta herramienta ofrece las siguientes características: editor de texto, resaltado de sintaxis, compilación en tiempo real, pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant y asistentes para el inicio en algunos de los elementos soportados (como proyectos, clases y test). Además, a través del uso de plug-ins se pueden utilizar tanto Subversion para

realizar el control de versiones como Hibernate para desarrollar las funciones de un motor de persistencia. (petra.euitio.uniovi.es).

NetBeans

El NetBeans, es un reconocido IDE, disponible para Windows, Mac, Linux y Solaris. El proyecto de NetBeans está formado por un IDE de código abierto y una plataforma de aplicación que permite a los desarrolladores crear con rapidez aplicaciones web, empresariales, de escritorio y móviles utilizando la plataforma Java, así como JavaFX, PHP, Java Script y Ajax, Ruby y C/C++.

El proyecto de NetBeans está apoyado por una comunidad de desarrolladores dinámica y ofrece documentación y recursos de formación exhaustivos, así como una selección diversa de complementos de terceros.

NetBeans es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest) que lo identifica como módulo.

La plataforma NetBeans es una base modular y extensible usada como una estructura de integración para crear grandes aplicaciones de escritorio y web.

La plataforma ofrece servicios comunes a las aplicaciones de escritorio, permitiéndole al desarrollador enfocarse en la lógica específica de su aplicación. Entre las características de la plataforma están la administración de las interfaces de usuario, administración de las configuraciones del usuario, administración del almacenamiento, administración de ventanas, frameworks basado en asistentes, entre otros. (Wiccsi) Por lo que es imprescindible para el desarrollo de esta aplicación. (Departamento de proyectos e ingeniería. Los Sistemas de información Geográfica.)

1.11 Conclusiones

- Con la culminación del presente capítulo se logró definir los aspectos teóricos y conceptuales, imponentes en la realización del problema propuesto, o sea, se ha explicado de forma clara el objeto de estudio, la descripción general de este, se analizaron algunas soluciones existentes y se especificaron algunas palabras claves importantes para el entendimiento de la situación problemática.

- También se han analizado los conceptos fundamentales referentes al desarrollo del problema y los beneficios que estos presentan. Partiendo del resultado de este estudio también en este capítulo, se seleccionan las tecnologías necesarias así como las herramientas y/o plataforma para instalar la solución tomando en cuenta los requerimientos del SIG.

- Se profundizó en el análisis de las características fundamentales de las tecnologías, lenguajes de programación y sistemas gestores de bases de datos candidatos para la implementación de la propuesta de este trabajo y se tuvo muy en cuenta que cada herramienta seleccionada fuera libre y de código abierto.

Capítulo 2: Análisis y Presentación de la solución propuesta.

2.1 Introducción

Para crear un software con calidad se hace necesario percibir la estructura y dinámica de la empresa u organización en cuestión y los procesos que en ella tienen lugar, con el objetivo de alcanzar una visión de las dificultades existentes, además de obtener un entendimiento común entre clientes y desarrolladores.

En este capítulo se efectúa el modelo de dominio, el cual posibilita obtener una vista más clara de los procesos que tienen lugar en la actualidad. Se hace una descripción los trabajadores del negocio. También se realiza una descripción de la solución propuesta, planteándose los requisitos funcionales y no funcionales del sistema a desarrollar, el diagrama de casos de uso del sistema, la descripción de los casos de uso críticos, se realizara los diagramas de clases del análisis y los diagramas de colaboración del análisis.

2.2 Descripción de procesos

Los procesos que posee esta aplicación SIG se dividen en diferentes clasificaciones. Entre las básicas tenemos la carga y visualización de datos alfanuméricos, el análisis y consulta avanzada de datos, estos datos geográficos pueden estar almacenados bajo diferentes formatos como ESRI, DXF, PNG GML entre otros. En cuanto a capacidades de edición se trabaja con una herramienta para autocompletar polígonos, el marcando puntos de interés con la nueva herramienta de notas, la posibilidad mediante una herramienta de auto asignar atributos y conectar o extender geometrías de línea con la herramienta de edición de línea.

Por otra parte, en el análisis espacial, se pueden obtener estadísticas básicas de atributos (max, min, media, etc), obtener mapas con diferentes métodos de clasificación (clasificación jensen, método de maximal), además una herramienta más robusta de búsqueda como la "*Search All Atrtibutes,* ejecutando análisis ráster y cargando directamente las imágenes con OpenJump. Se utilizan herramientas para crear, dibujar y editar datos geométricos, así como herramientas de selección posee un extractor de

segmentos el cual elimina los segmentos que aparecen varias veces, exportación de imágenes en formato de gráficos vectoriales escalables (SVG), la opción para mostrar la escala del mapa real (Medina, 2000).

Para la salva de los datos con los que se trabaja utiliza el gestor de base de datos PostgreSQL con PostGIS, el módulo que adquiere la capacidad de almacenar consultar y acceder a información geoespacial, de realizar operaciones de análisis geográfico, además utilizando la extensión pgRouting que puede emplear como un potente motor de cálculo de rutas y otras operaciones geoespaciales.

2.2 Modelo de Dominio

Haciendo un estudio profundo del SIG OpenJump se llega a la conclusión que en la presente investigación no se definen concretamente todos los procesos y existen múltiples responsabilidades, por lo que se decide dar un nuevo enfoque a todo el proceso. Para ello se realiza un Modelo de Dominio. Este modelo contribuye a identificar las clases que se utilizarán en el sistema, así como comprender mejor la estructura y los conceptos necesarios en el sistema.

Un modelo de dominio permite de manera visual mostrar al usuario los principales conceptos que se manejan en el dominio del problema. Esto ayuda a los usuarios, clientes, desarrolladores e interesados a utilizar un vocabulario común para poder entender el contexto en que se enmarca el sistema. Además es necesario tener un vasto conocimiento de cómo debe funcionar el proceso en cuestión, para poder capturar correctamente los requisitos y así poder construir una aplicación con las características que el cliente desee.

Un modelo de dominio captura los tipos más importantes de objetos que existen o los eventos que suceden en el entorno donde estará el sistema. A continuación se muestra el diagrama de clases del modelo de dominio:

2.2.1 Conceptos Fundamentales

Institución Geográfica

Es un grupo empresarial que se dedica a la elaboración, producción y venta de planos, mapas y cartas náuticas con diversos fines, así como a la realización de estudios geográficos, de impacto ambiental, e investigaciones científicas en ramas del campo de las geociencias, entregando a sus clientes, productos informativos terminados con una alta calidad y fiabilidad.

Cartógrafo

Persona que trabaja en una empresa, organismo u organización que necesite cambiar o consultar algún tipo de información incluida en un mapa.

Entidad (UCI)

Empresa, organismo u organización que solicita un servicio determinado utilizando un mapa y que proporciona la información socioeconómica referente al mismo.

Mapa

Un mapa o plano cartográfico es una representación gráfica y métrica de una porción de territorio generalmente sobre una superficie bidimensional, generalmente plana, pero que puede ser también esférica como ocurre en los globos terráqueos. Si el mapa tenga propiedades métricas significa que ha de ser posible tomar medidas de distancia, ángulos o superficies sobre él y obtener un resultado aproximadamente exacto en la realidad. (Báez, 2010).

Escala

Relación entre la distancia que separa dos puntos en un mapa y la distancia real de esos dos puntos en la superficie terrestre. En los mapas, la escala puede expresarse de tres modos distintos: en forma de proporción o fracción, con una escala gráfica o con una expresión en palabras y cifras. Cuanto mayor es la escala, más se aproxima al tamaño real de los elementos de la superficie terrestre. Los mapas a pequeña escala generalmente representan grandes porciones de la Tierra y, por tanto, son menos detallados que los mapas realizados con escalas más grandes. Por extensión puede referirse a la mayor o menor profundidad del enfoque en un tema geográfico. (Báez, 2010)

Leyenda

Explicación de los símbolos, los colores, las tramas y los sombreados empleados en un mapa; suele encontrarse a pie de página o en un recuadro, situado en sus márgenes o bien en su dorso. (Báez, 2010)Los símbolos empleados en los mapas pueden llegar a contener un gran volumen de información, que por su facilidad de lectura permiten una rápida interpretación.

Tipo de Mapa

Clasificación que se le da a los mapas de acuerdo con su especificación.

Información Socioeconómica

Es un conjunto organizado de datos procesados referentes al aspecto social y económico de cualquier lugar de interés del país.

2.2.2 Diagrama de clases de dominio

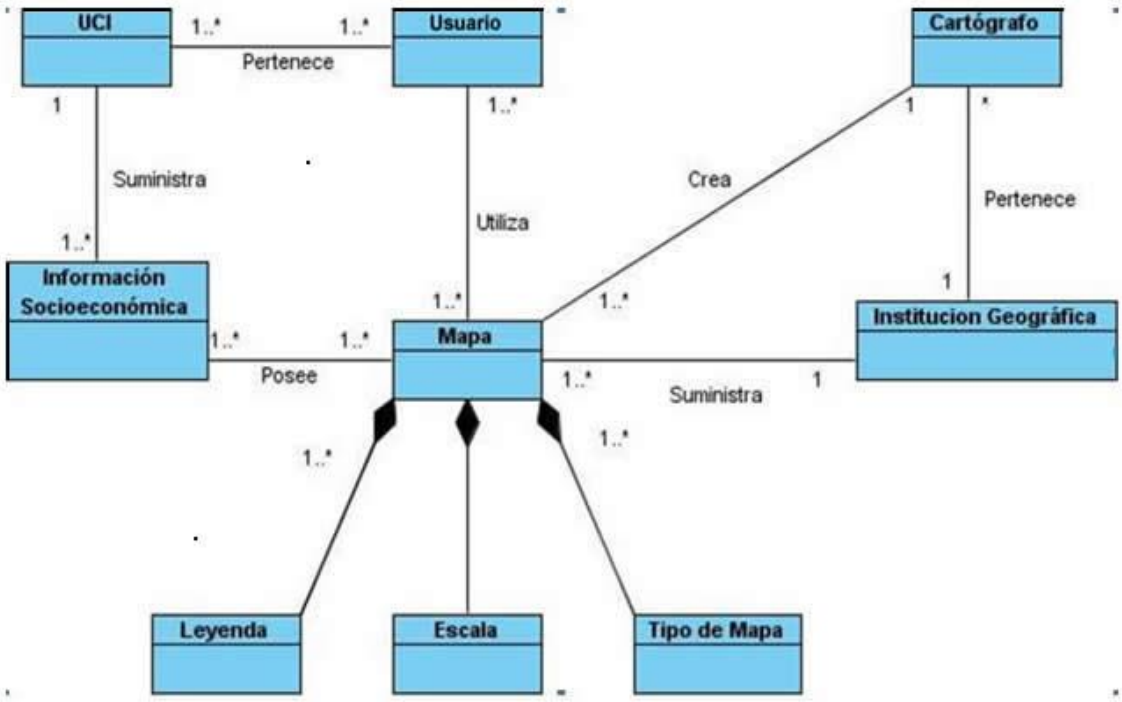


Figura 1 Diagrama de Dominio.

2.2.2.1 Breve descripción del diagrama de clases del dominio

En el proceso intervienen cinco elementos fundamentales: Operador, Entidad (UCI), Mapa, Información socioeconómica y el Proveedor de Mapas. La interacción entre estas cinco entidades del dominio confluye en la clase mapa, la cual es considerada el objeto fundamental. Las clases Leyenda, Escala y Tipo de mapa, son clases que forman parte de la clase mapa, teniendo una relación fuerte con esta. Por último la

clase entidad es el medio entre el cliente y la información desde el punto de vista estructural u organizacional, no así Mapa la cual media entre cliente y la información de manera virtual o gráfica.

2.3 Requisitos Funcionales

Los requisitos funcionales son capacidades o condiciones que el sistema debe cumplir, para que las peticiones del cliente queden satisfechas, y responden a: ¿Qué debe hacer el sistema? (Estrugo Lahera, 2010.) Los cuales se muestran a continuación:

***RF1*.** El sistema debe permitir gestionar imágenes Vectoriales.

RF 1.1. **Acercar.** Esta funcionalidad permite aumentar el tamaño del mapa y disminuye la escala, ubicando en el centro del mapa el punto en el que el usuario realizó la operación de Acercar.

***RF 1.2*. Alejar.** Esta funcionalidad disminuye el tamaño del mapa y aumenta la escala.

RF 1.3. **Ver todo.** Esta funcionalidad permite visualizar el mapa según la escala inicial de la aplicación.

***RF 1.4*. Anterior.** Esta funcionalidad permite visualizar el mapa anterior al que se visualiza en la aplicación.

***RF 1.5*. Siguiente.** Esta funcionalidad permite una vez que haya seleccionado la opción “Anterior” vuelva al estado en el que se encontraba.

***RF 1.6*. Recentrar mapa.** Con este requerimiento se quiere que el usuario pueda recentrar una región previamente seleccionada al dar clic en el mapa, sin modificar su escala.

***RF 1.7*. Mover el mapa.** Con este requerimiento se quiere que el usuario pueda mover el mapa variando con el puntero del ratón la posición de la vista que se presenta.

RF2. El sistema debe permitir cargar capas de BD.

RF3. El sistema debe permitir importar proyectos.

RF4. El sistema debe permitir exportar vistas como imágenes JPG.

RF5. El sistema debe permitir guardar proyecto en formato (*.jmp).

RF6. El sistema debe permitir crear Mapa Temático: Con este requerimiento se quiere que el usuario pueda crear un mapa temático entrando valores asociados a criterios de búsquedas seleccionados.

RF7. El sistema debe permitir autenticar usuarios.

RF8. El sistema debe permitir cerrar la sesión del Usuario***.***

RF9. El sistema debe permitir medir ángulo.

RF10. El sistema debe permitir añadir punto.

RF11. El sistema debe permitir añadir líneas.

RF12. El sistema debe permitir añadir polígono.

RF13. El sistema debe permitir medir áreas.

2.4 Requisitos no Funcionales

Los requisitos no funcionales son las cualidades o propiedades que el producto debe tener. En muchos casos los requisitos no funcionales son fundamentales en el éxito del producto. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o confiable. Están vinculados normalmente a requisitos funcionales, es decir una vez se conozca lo que el sistema debe hacer podemos determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser. (Estrugo Lahera, 2010.)

Usabilidad: El sistema podrá ser usado por personas con conocimientos básicos en el manejo de computadoras, la herramienta tendrá siempre visible la opción de ayuda posibilitando un mejor aprovechamiento por parte de los usuarios de sus funcionalidades.

Eficiencia o Rendimiento: El tiempo de respuesta estará condicionado por la cantidad de datos a procesar y la complejidad de la operación, este oscilará entre 3 segundos, para las operaciones simples como calculo de áreas y 5 segundos para trabajos complicados con datos geográficos, como es el caso de tematizar.

Restricciones de diseño: Diseño sencillo, con pocas entradas, donde no sea necesario mucho entrenamiento para utilizar el sistema.

Requisitos para la documentación de usuarios en línea y ayuda del sistema: La herramienta tendrá siempre la posibilidad de ayuda disponible para cualquier tipo de usuario.

Interfaz de software: Sistemas Operativos: Windows 2000 o Superior.

Hardware: El sistema estará instalado en una PC cliente, donde se encuentra el Gestor de Base de Datos.
Requisitos Mínimo: Procesador: 800 MHz, Memoria: 256 MB, Disco Duro: 60 GB, Tarjeta de Red: Ethernet 10/100 MB.
Requisitos Recomendados: Procesador: 1.8 GHz, Memoria: 1GB, Disco Duro: 80 GB o más, Tarjeta de Red: Ethernet 10/100 MB.

2.5 Modelo del Sistema

Con la comprensión de los objetivos que debe cumplir el SIG OpenJump se analizan las características que debe presentar el mismo, para ello se identifican los requisitos funcionales y no funcionales del sistema.

Los requisitos son las condiciones o capacidades que tienen que ser logradas o poseídas por un sistema o componente de un sistema para integrar un contrato, estándar u otro documento impuesto formal (Medina, 2000).

2.3.1 Diagrama de Casos de Uso del Sistema

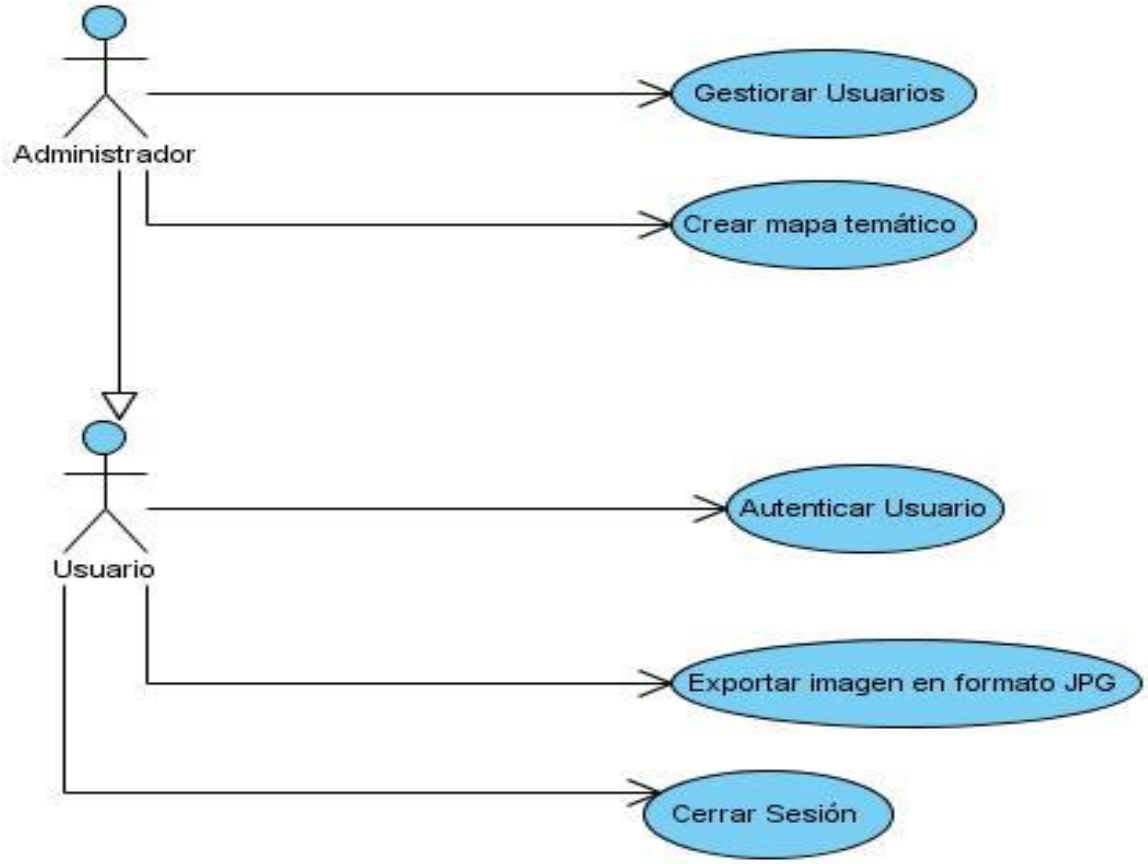


Figura 2 Diagrama de Casos de Uso del Sistema.

2.6 Descripción Textual de los Casos de Uso del Sistema

2.6.1: Descripción textual del CU: Autenticar Usuario

Caso de uso	Autenticar Usuario.
Actores	Usuario
Propósito	Este caso de uso tiene como objetivo lograr la seguridad completa del SIG al autenticar cada usuario que trate de entrar al software.
Resumen	El caso de uso se inicia cuando un usuario accede a la aplicación y para ello introduce su usuario y contraseña, si los datos del mismo son correctos, tendrá acceso a las funcionalidades que brinda la aplicación, finalizando así el caso de uso, en caso de no estar correctos los datos el sistema muestra un mensaje advirtiendo al usuario.
Prioridad	**Critico**

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El caso de uso se inicia cuando el usuario accede a la aplicación. 2. El usuario introduce su nombre de usuario y contraseña y presiona el botón para entrar.	1.1 El sistema muestra una interfaz para acceder a la aplicación. 2.2 El sistema verifica que los datos sean correctos. 2.3 El sistema le da acceso a la aplicación dependiendo del permiso

	del usuario y finaliza el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del sistema**
	3.1 El sistema muestra un mensaje en caso que existan campos vacíos o datos incorrectos y retorna a la página de inicio, finalizando el caso de uso.
Poscondiciones	Se ha autenticado un usuario en el sistema.

2.6.2: Descripción Textual del CU: Permitir exportar vistas como imágenes en formato JPG

Caso de uso	Permitir exportar vistas como imágenes JPG.
Actores	Usuario
Propósito	Este caso de uso tiene como objetivo expedir y generar las imágenes gestionadas anteriormente, convirtiéndolas en formato JPG.
Resumen	El caso de uso es iniciado cuando el operador desea obtener la información asociada a una zona geográfica analizada, permitiendo generar la información que el sistema muestra en formato JPG.
Prioridad	**Critico**

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El caso de uso se inicia cuando el actor selecciona la opción Guardar Como. 2. El usuario de clic derecho encima de la opción Guardar Como, se activa la ventana de exportar, escoge el lugar donde se quiere guardar así como el formato JPG.	2.2 El caso de uso concluye cuando la se guarda la vista deseada en la computadora con el formato JPG.

2.7 Análisis

Durante el análisis, se examinan los requisitos que se describieron en la captura de requisitos, refinándolos y estructurándolos. El objetivo de hacerlo es conseguir una comprensión más precisa de los requisitos y una descripción de los mismos que sea fácil de mantener y ayude a estructurar el sistema completo (Paul, 2001).

2.8 Diagrama de Clases del Análisis

Uno de los principales artefactos del análisis es el Diagrama de Clases del Análisis. Estos representan los conceptos fundamentales del dominio del problema y las relaciones entre las clases.

Clasificación de las clases del análisis:

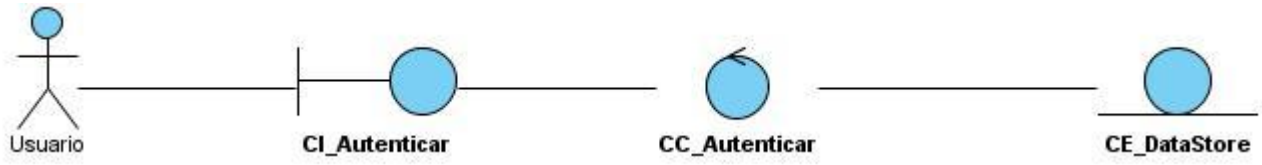
Clase Interfaz: Modela la interacción entre el sistema y sus actores.

Clase Controladora: Representa coordinación, secuencia, transacciones y control de otros objetos.

Clase Entidad: Modela información que posee una larga vida y es a menudo persistente.

A continuación se muestra el diagrama de clases del análisis correspondiente a dos de los casos de usos descritos:

2. 8.1 Diagrama de Clase del Análisis del CU Autenticar Usuario



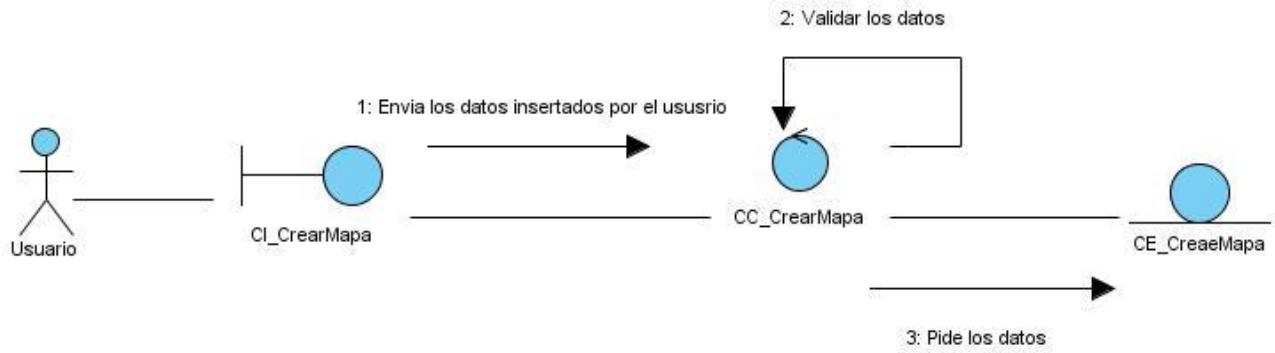
2. 8.2 Diagrama de Clase del Análisis del CU Exportar Imágenes en Formato JPG

2.9 Diagrama de Colaboración del Análisis

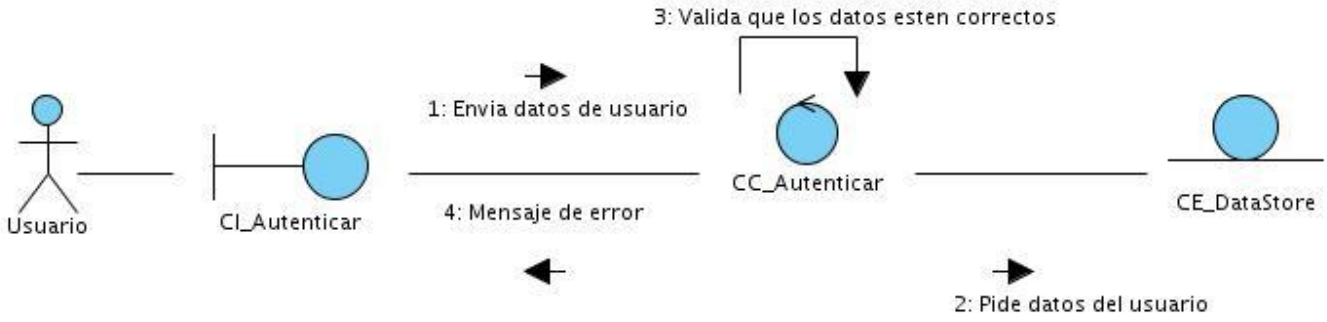
Los diagramas de colaboración exponen cómo las instancias específicas de las clases trabajan juntas para conseguir un objetivo común. Ellos destacan el contexto y organización general de los objetos que interactúan. Se organizan de acuerdo al espacio y muestran a los objetos junto con los mensajes que se envían entre ellos.

A continuación se muestra el diagrama de colaboración del análisis correspondiente a dos de los diagramas de clases del análisis:

2.9.1 Diagrama de Colaboración del Análisis del CU Crear Mapa Temático



2.9.2 Diagrama de Colaboración del Análisis del CU Autenticar Usuario



2.10 Conclusiones

- Se describe el dominio del problema a través de su diagrama de clase conceptual, lo que permite un mayor entendimiento de los elementos que giran alrededor de la situación problemática.
- Se definen los requisitos funcionales que el sistema debe cumplir lo cual sirve para identificar las funcionalidades con las que contará el SIG y los no funcionales que patentizarán las cualidades o propiedades que el SIG debe tener.

- Se establecen los requisitos funcionales que contribuyen a identificar los actores del sistema y los casos de usos del sistema.
- Se efectuaron las descripciones textuales de los Casos de Usos.
- Se modelaron los diagramas de clases del análisis de cada caso de uso con el fin de una mejor visualización y entendimiento de las relaciones entre clases.
- Se realizó los diagramas de colaboración del análisis exponiendo las clases y relaciones entre estas.

Capítulo 3: Diseño de la Solución propuesta

3.1 Introducción

El propósito fundamental del diseño es indicar cómo se llevará a cabo el sistema. El mismo constituye una parte fundamental del software y durante él se toman decisiones estratégicas y tácticas para cumplir los requerimientos funcionales y de calidad de un sistema, así como también se refina la visión obtenida en el análisis.

En el flujo de trabajo Diseño se define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán, la ubicación en ellos de los componentes. El objetivo de este flujo de trabajo del Proceso Unificado de Desarrollo, es desarrollar la arquitectura y el sistema como un todo (Jacobson Ivar, 2000). En el presente capítulo se abunda sobre el diseño del sistema propuesto, además se expone el diagrama de paquetes, Diagramas de clases del diseño para cada caso de uso y se describen los patrones de diseño empleados en el sistema.

3.2 Modelo de Diseño

En el modelo de diseño es un modelo de objetos que se centra en cómo los requerimientos funcionales y no funcionales, junto con otras restricciones del entorno de implementación, tienen impacto en el sistema que se desarrolla. El diseño debe ser suficiente para que el sistema pueda ser implementado sin ambigüedades.

3.3 Diagrama de Paquetes

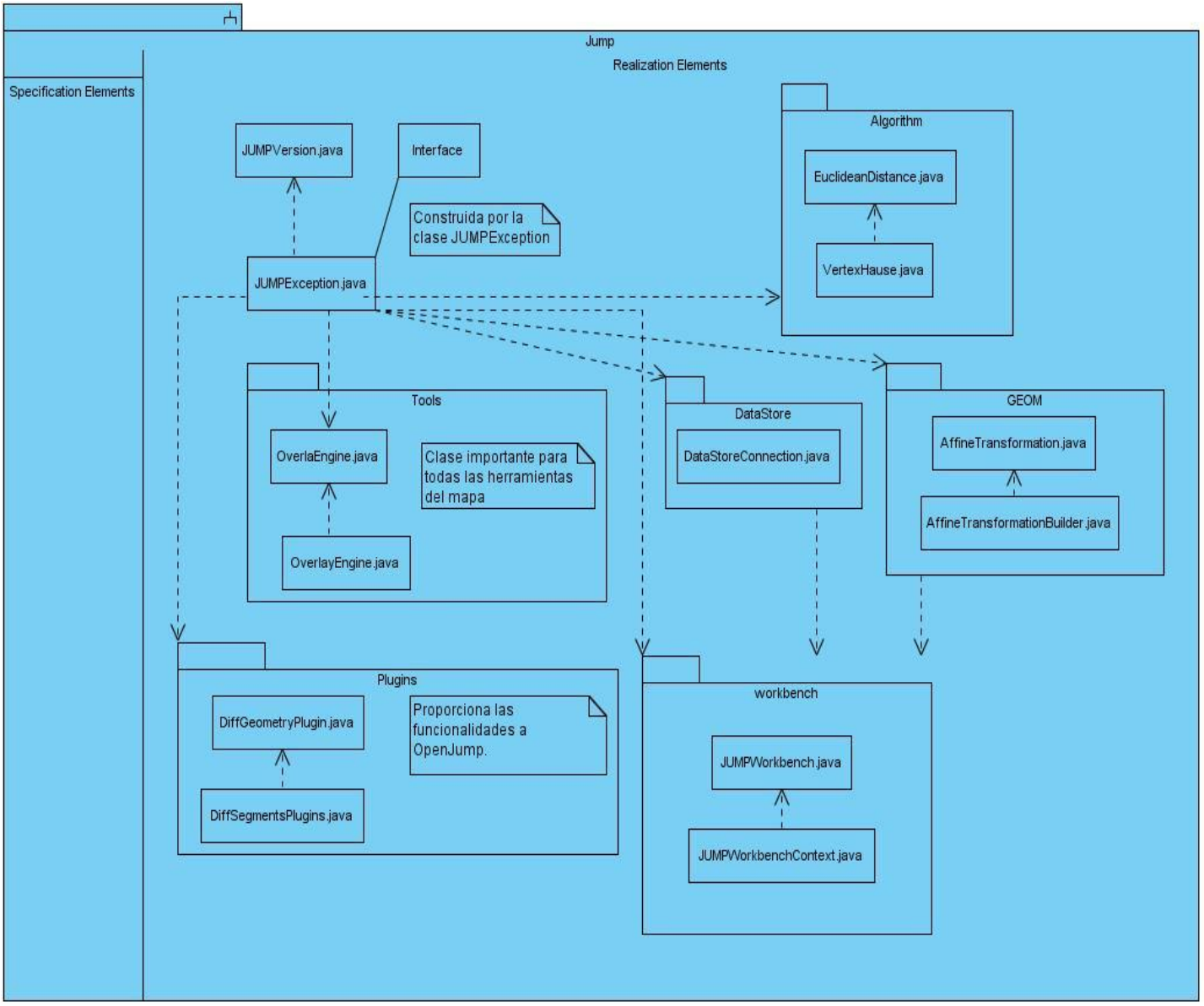


Figura 3 Diagrama de Paquetes.

3.4 Diagrama de clases del diseño del CU Autenticar Usuario

3.5 Patrones de diseño empleados

Para el diseño de la solución propuesta se escogieron los siguientes patrones: Orientado a Objeto y Orientado a Componentes.

Como expresan los autores de Pattern-Oriented Software Architecture (POSA), los patrones de diseño son los patrones de mediana escala, siendo un poco más específicos que los patrones arquitectónicos, pero tienden a ser independientes de un lenguaje de programación o paradigma de programación. La aplicación de un modelo de diseño no tiene ningún efecto sobre los aspectos fundamentales de la estructura de un sistema de software, pero puede tener una fuerte influencia en la arquitectura de un subsistema (Marca Huallpara).

Capítulo 3: Diseño de la Solución Propuesta.

Los patrones de diseño además de ser soluciones a problemas comunes de diseño en un determinado contexto y estar presentes en el desarrollo de software; permiten formalizar un vocabulario común entre los diseñadores del sistema y estandarizar el modo en que se realiza el diseño. La utilización de los patrones de diseño posibilita entender, mantener y ampliar el sistema de manera fácil, así como contribuir a la reutilización y diseño de componentes de software, a la organización del código, a la flexibilidad y extensibilidad, y la facilidad de realizar cambios en el sistema.

Los mismos cumplen iguales principios que los estilos pero a un nivel más determinante en la arquitectura, puesto que estarán asociados a las formas más concretas, a la especialización que adoptan los objetos y clases de acuerdo al tipo de aplicación o entorno tecnológico.

Como se describe anteriormente los patrones de diseño se han categorizado en dos grupos. Se detallan entonces los patrones de diseños a utilizar:

GRAPS (Patrones Vencimiento General de Responsabilidad de Software)

- ✓ Experto (Expert): La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados (atributos). Una clase que contiene toda la información necesaria para realizar la labor que tiene encomendada.
- ✓ Creador (Creator): Es el que crea y guía la asignación de responsabilidades relacionadas con la creación de objetos. Se asigna la responsabilidad de que una clase B cree un objeto de la clase A. Esta tarea es muy frecuente en los sistemas orientados a objetos.
- ✓ Controlador (Controller): Es un evento generado por actores externos. Se asocian con operaciones del sistema como respuestas a los eventos del sistema, tal como se relacionan los mensajes y los métodos.

Un controlador delega en otros objetos el trabajo que se necesita hacer y coordina o controla la actividad. No realiza mucho trabajo por sí mismo. Los tipos de controladores a usar son:

- ✓ Controlador de Fachada: Representa al sistema global, dispositivo o subsistema.

- ✓ Controlador de casos de uso: Constitución artificial para dar soporte al sistema, se utilizan cuando los controladores de fachada conduce a diseños con baja cohesión o alto acoplamiento.

- ✓ No hables con extraños (Don't Talk to Strangers): Asignar la responsabilidad al objeto directo de un cliente para colaborar con un objeto indirecto, para que dicho cliente no necesite saber sobre el objeto indirecto.
- ✓ Indirección (Indirection): Asignar la responsabilidad a un objeto intermedio para mediar entre otros componentes o servicios, y así evitar que se acoplen directamente.

GoF (Gans of Four)

- Patrones Creacionales (Creational Patterns)

✓ Factoría Abstracta (Abstract Factory): Abastecer una interfaz con el fin de crear familias de objetos, que dependen entre sí o estén relacionados, sin especificar sus clases concretas. Abstraer algunas clases que se instancian y encapsular la funcionalidad que se desea. Es la vía de comunicación hacia las clases concretas instanciadas.

✓ Constructor (Builder): Aislar la construcción de un objeto complejo de su representación. El mismo proceso de construcción puede crear diferentes representaciones. Una clase “Constructor” se aplica en la creación de otras clases cuando es necesario crear y agregar instancias de clases, contener múltiples instancias de clases y cuando la clase “constructor” posee los datos necesarios para instanciar la nueva clase.

✓ Método Factoría (Factory Method): Utilizar la abstracción de clases para crear y relacionar objetos sin tener en cuenta de qué clase proviene. Permite definir una interfaz para crear un objeto y posibilita que sean las subclases quienes decidan que clases instanciar. Permite que una clase delegue en sus subclases la creación de objetos (Constructor Virtual).

- Patrones Estructurales (Structural Patterns)

✓ Adaptador (Adapter): Permite que cooperen clases a su través por la complejidad que resultaría hacerlo de otro modo por poseer interfaces incompatibles. Es decir, dicho patrón adecúa los componentes de software a las necesidades de un determinado sistema.

✓ Bridge - Handle/Body: Separar una abstracción de su implementación de forma que ambas pueden evolucionar por separado y así la abstracción pueda tener varias implementaciones posibles. Es decir, una clase abstracta define la interfaz y clases concretas la implementa de diferentes formas. El camino habitual para acomodarlas es a través del uso de la herencia.

✓ Decorador (Decorator): Asignar nuevas responsabilidades a un objeto dinámicamente y añadir nuevas responsabilidades a objetos individuales y no a toda la clase. Es una alternativa a la creación de subclases por herencia.

✓ Fachada (Facade): Proporcionar una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar. Reduce la dependencia entre clases, ya que ofrece un punto de acceso al resto de las clases, si estas cambian o se sustituyen por otras solo hay que actualizar la clase Fachada sin que el cambio afecte a las aplicaciones cliente.

- Patrones de Comportamiento (Behavioral Patterns)

✓ Chain of Responsability: Delegar responsabilidades a una clase especializada. Permite distribuir la ejecución repartiendo las responsabilidades y constituye la forma básica al realizar un Diseño Orientado a Objetos, por lo que su uso es de forma intuitiva en este tipo de diseño.

✓ Memento: Guarda el estado de un objeto para restaurarlo posteriormente.

✓ Observador (Observer): Notificar automáticamente los cambios de estado de un objeto a todos sus dependientes. Ello permite implementar dinámicamente dependencias entre objetos, de forma que los objetos dependientes sean notificados de los cambios que se producen en los objetos de los que dependen. Posibilita encapsular la gestión de las dependencias dinámicas, realizar cambios futuros en una sola clase y reutilizar el mecanismo de interdependencia.

✓ Template Method: Encapsula un algoritmo en una clase abstracta de forma que puede ser reutilizada. Muchos algoritmos pueden ser reutilizados aportando cierta información específica propia de los elementos que tenga que tratar en cada momento.

Singlenton (Instancia única): En el diseño de clases es necesario aplicar la solución del patrón Singlenton que no es más que garantizar el acceso único a una clase mediante una única instancia. De esta forma se controla el acceso a las clases.

El diseño es el centro de atención al final de la fase de elaboración y el comienzo de las iteraciones de construcción. Esto contribuye a una arquitectura estable y sólida, y crear un plano del modelo de implementación.

3.6 Conclusiones

- ✓ Se realizó el diagrama de paquetes para el SIG en general mostrando los objetos presentes en el sistema.
- ✓ Se modelaron los diagramas de clases del diseño por cada caso de uso para un mejor entendimiento
- ✓ Se definieron los patrones de diseño que se emplearon, permitiendo determinar un entendimiento entre los diseñadores del sistema y estandarizar el modo en que se realiza el diseño.

Capítulo 4: Descripción de la Arquitectura de la Solución propuesta

4.1Introducción

La arquitectura de software consiste en un conjunto de patrones y meditaciones relacionados que proporcionan el marco de referencia necesario para guiar la construcción de sistemas de softwares.

Un estilo arquitectónico define las reglas generales de organización en términos de un patrón, así como las restricciones en la forma y la estructura de un grupo numeroso de sistemas de software.

Los objetivos del presente capítulo son la definición del objetivo de los patrones arquitectónicos para el software, importante para construir sistemas complejos con el fin de reutilizarlos en aplicaciones futuras para proporcionar los servicios que éstas necesitan. Selección de los atributos de calidad presentes en el impacto de la arquitectura. La presentación de la Vista Lógica, Vista de Implementación, Vista de Despliegue y Vista de Componentes.

4.2 Arquitectura de software

Existen numerosos conceptos de la Arquitectura de Software, pero todos se articulan alrededor de algunos principios: guía todo el proceso de desarrollo de los sistemas informáticos de software y debe garantizar que sean cumplidos los requisitos funcionales y no funcionales descritos por el cliente. La misma se refina durante todo el ciclo de vida de los proyectos y contribuye a que no excedan los límites de costo y tiempo establecidos en ella. Se puede manejar además como un conjunto de decisiones a tener en cuenta en el diseño de un proyecto, las cuales deben estar correctamente planteadas, de no ser así se podría anular todo un producto. Uno de los conceptos más reconocidos o usados son los mencionados con anterioridad además de existir otros de gran relevancia como:

… “Arquitectura es un nivel de diseño que hace foco en aspectos más allá de los algoritmos y estructuras de datos de la computación; el diseño y especificación de la estructura global del sistema es un nuevo tipo de problema" (Bosch, 2000)

Capítulo 4: Arquitectura de la Solución Propuesta.

La presente investigación se ajusta al concepto emitido por la IEEE la cual expone que es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento del software; es decir, la aplicación de la ingeniería al software (IEEE, 1993).

La misma debe apoyar las configuraciones, tanto estáticas como dinámicas, de componentes a través de un conjunto de interfaces bien documentados o contratos conociendo a los clientes de un sistema de software determinado. La arquitectura debe ser descrita en un conjunto de patrones para servir de claridad y como un simple medio de comunicación a las partes interesadas del sistema. La arquitectura de software debe apoyar variabilidad, la portabilidad, escalabilidad, reutilización para mencionar algunos. (sei.cmu.edu)

Se puede decir entonces que la arquitectura de software es una vista estructural de alto nivel, ocurre muy tempranamente en el ciclo de vida y define los estilos o grupos de estilos adecuados para cumplir con los requerimientos no funcionales.

Estilos Arquitectónicos:

Una de las definiciones por autores dedicados al tema es:

"... una familia de sistemas de software en términos de un patrón de organización estructural, que define un vocabulario de componentes y tipos de conectores y un conjunto de restricciones de cómo pueden ser combinadas. Para muchos estilos puede existir uno o más modelos semánticos que especifiquen cómo determinar las propiedades generales del sistema partiendo de las propiedades de sus partes." es la definición que le dan Shaw y Garlan (Bosch, 2000).

Por ende se puede decir que un estilo arquitectónico es una transformación impuesta al diseño de todo un sistema con el objetivo de establecer una estructura para todos los componentes de dicho sistema.

Estos estilos se ordenan en seis clases fundamentales, a continuación cada una de ellas:

Estilos de Flujo de Datos

- ✓ Tubería y filtros

Estilos Centrados en Datos

- ✓ Arquitecturas de Pizarra o Repositorio

Estilos de Llamada y Retorno

- Model-View-Controller (MVC)
- Arquitecturas en Capas
- Arquitecturas Orientadas a Objetos
- Arquitecturas Basadas en Componentes

Estilos de Código Móvil

- Arquitectura de Máquinas Virtuales

Estilos heterogéneos

- Sistemas de control de procesos
- Arquitecturas Basadas en Atributos

Estilos Peer-to-Peer

- Arquitecturas Basadas en Eventos
- Arquitecturas Orientadas a Servicios
- Arquitecturas Basadas en Recursos

El paquete de Arquitectura de Software que se propone para el Sistema de Información Geográfica de escritorio OpenJump está fundamentado por el estilo arquitectónico orientado a objetos (AOO), aunque también hace uso de la arquitectura basada en componentes (ABC) ya que se requiere una arquitectura

de sistema flexible y fácil de personalizar. Por lo que permite que se le integren plugins que permitan más funcionalidades al sistema según sean necesarias. A continuación la descripción de estos dos estilos:

Arquitecturas Orientadas a Objetos: También reconocido por algunos autores como Arquitecturas Basadas en Objetos u Organización Orientada a Objetos, en este estilo los componentes son los objetos o más bien instancias de los tipos de dato abstractos.

Los componentes del sistema encapsulan datos y operaciones que deben usarse para manipular dichos datos. La comunicación y coordinación entre esos componentes se realiza mediante envío de mensajes. Este es un sistema donde se enfatiza el empaquetamiento entre datos y operaciones que permiten manipular y acceder a dichos datos (Pressman, 2003)

Como se explica, esta arquitectura estructura el sistema en un conjunto de objetos débilmente acoplados y con interfaces bien definidas donde estos objetos hacen llamadas a las funciones de otros objetos. Una organización orientada a objetos está relacionada con las clases de objetos, sus atributos y sus operaciones. Cuando se implementa, los objetos se crean a partir de estas clases y se usan algunos modelos de control para coordinar las operaciones de dichos objetos (Sommerville, 2005).

Ello se traduce en la descomposición del diseño en componentes prácticos o lógicos que exponen interfaces de comunicación bien definidas. La misma no se enfoca en asuntos específicos de los objetos. Describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema.

Las ventajas de este estilo radican en que como los objetos están débilmente acoplados, la modificación de ellos puede efectuarse sin comprometer a los otros. Los mismos son representaciones de entidades del mundo real lo que posibilita que la estructura del sistema sea fácilmente comprensible; y ello a la vez permite la reutilización de dichos objetos dado que las entidades en el mundo real se usan en sistemas diferentes.

Arquitectura Basada en Componentes: Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee

un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado (Application Architecture Guide, 2000). Es decir, esta descomposición posibilita la transformación de los componentes pre-existentes en piezas más grandes de un software. Es necesario conocer qué es un componente de software para su mejor comprensión, y el mismo se puede definir como algo que puede ser utilizado como una caja negra, de manera que se tiene una especificación externa general, y ésta es completamente independiente de la especificación interna.

Un componente es en sí un objeto de software diseñado para cumplir con cierto propósito. Cuando se diseña un componente estos deben poseer principios fundamentales, tales como:

- ✓ **Reusable:** Los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas.
- ✓ **Sin contexto específico:** Los componentes son diseñados para operar en diferentes ambientes y contextos. Información específica como el estado de los datos deben ser pasadas al componente en vez de incluirlos o permitir al componente acceder a ellos.
- ✓ **Extensible:** Un componente puede ser extendido desde un componente existente para crear un nuevo comportamiento.
- ✓ **Encapsulado:** Los componentes exponen interfaces que permiten al programa usar su funcionalidad. Sin revelar detalles internos, detalles del proceso o estado.
- ✓ **Independiente:** Los Componentes están diseñados para tener una dependencia mínima de otros componentes. Por lo tanto los componentes pueden ser instalados en el ambiente adecuado sin afectar otros componentes o sistemas.

Los beneficios de este estilo, es que permite reemplazar una nueva versión del sistema por la existente sin impacto en el mismo o en sus componentes. La utilización de componentes de otros permite distribuir el costo del desarrollo y del mantenimiento, reduciendo el mismo considerablemente. Facilita el desarrollo ya que al igual que Arquitectura Orientada a Objetos los componentes implementan una interfaz bien

definida lo que permite el desarrollo sin impactar otras partes del sistema. Otro de los más significativos es la reusabilidad, que significa que los componentes pueden ser usados para distribuir el desarrollo y el mantenimiento entre múltiples aplicaciones y sistemas. La mitigación de complejidad técnica consiste en la moderación de las múltiples relaciones entre clases y objetos así como la dependencia entre ellos, por medio del uso de contenedores de componentes y sus servicios.

Es importante señalar que aunque este estilo tiene sus similitudes al estilo Orientado a Objetos, el primero proporciona un mayor nivel de abstracción que los principios de diseño del segundo, ya que se basa en la relación de los componentes.

4.3 Atributos de calidad y requisitos que tienen impacto sobre la arquitectura

La calidad puede definirse como la conformidad relativa con las especificaciones, el grado en que un producto cumple con los requisitos establecidos, o como comúnmente encontrar la satisfacción en un producto cumpliendo todas las expectativas que busca algún cliente.

Pressman la define como: “Concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados y con los requerimientos implícitos no establecidos formalmente, que desea el usuario.”

La clasificación de calidad que el usuario percibe, es separada en los sistemas informáticos como “Atributos Externos de la Calidad”, es decir aquello que se percibe desde el exterior. La presente investigación utilizará los atributos de calidad del modelo ISO/EC 9126 adaptado para arquitecturas de software para la evaluación del diseño arquitectónico debido a la correspondencia de dichos atributos con las necesidades del sistema. El modelo de calidad ISO 9126-1, identifica varias características básicas de calidad que pueden estar presentes en cualquier producto de software. El estándar provee una descomposición de las características en subcaracterísticas.

Característica	**Subcaracterística**	**Elementos de tipo arquitectónico**

Funcionalidad	Adecuación	Refinamiento de los diagramas de secuencia.
	Exactitud	Identificación de los componentes con las funciones responsables de los cálculos.
	Interoperabilidad	Identificación de conectores de comunicación con sistemas externos.
	Seguridad	Mecanismos o dispositivos que realizan explícitamente la tarea.
Confiabilidad	Tolerancia a fallas	Existencia de mecanismos o dispositivos de software para manejar excepciones.
	Recuperabilidad	Existencia de mecanismos o dispositivos de software para restablecer el nivel de desempeño y recuperar datos.
Eficiencia	Desempeño	Componentes involucrados en un flujo de ejecución para una funcionalidad.
	Utilización de recursos	Relación de los componentes en términos de espacio y tiempo.
Mantenibilidad	Acoplamiento	Interacciones entre componentes.
	Modularidad	Número de componentes que dependen de un componente.
Portabilidad	Adaptabilidad	Presencia de mecanismos de adaptación.
	Instalabilidad	Presencia de mecanismos de instalación.
	Coexistencia	Presencia de mecanismos que faciliten la coexistencia.
	Reemplazabilidad	Lista de componentes reemplazables para cada componente.

Tabla 1. Atributos de calidad adaptados para arquitecturas de software

Se tienen entonces identificados los siguientes atributos que componen la calidad de OpenJump:

Funcionalidad: Se adecúan cada uno de los diagramas de secuencia en cada iteración de refinamiento de los diagrama de clases y se propone realizar diagramas de secuencia independientes a los métodos que están sobrecargados. Se asegura la conectividad con subsistemas y sistemas externos a la aplicación a través de protocolos de integración como Internet Communications Engine (ICE) y estándares que propicien dicha comunicación (Security Assertion Markup Language SAML). La seguridad informática, tema de gran importancia ya que es fundamental mantener la protección del sistema y evitar que sean vulnerable a robos de información o a accesos no autorizados, se garantiza haciendo uso del RBAC (Role Based Access Control) como una propuesta que reúne las principales bondades de los tipos de control de acceso existentes (Discretionary Access Control DAC y Mandatory Access Control MAC). Dicha seguridad se garantiza a través de un subsistema de gestión de la seguridad.

Confiabilidad: Se previenen los fallos para protegerlo de interferencias o fallas esperadas, tanto de hardware mediante la utilización de componentes fiables, técnicas rigurosas de montaje de subsistemas y el aislamiento de hardware; como de software realizando la especificación de requisitos rigurosamente, los comprobando los métodos de diseño, utilizando lenguajes con abstracción de datos, modularidad, y utilización herramientas CASE adecuadas para gestionar los componentes. La separación de los servidores del cliente proporciona también un alto grado de confiabilidad puesto que las fallas en uno no repercuten directamente en el otro, además la existencia de un servidor de respaldo. Deben realizarse pruebas que garanticen la veracidad y robustez del código. Con la correcta aplicación del método de caja blanca se satisfacen principalmente tres necesidades: que el sistema cumpla con las especificaciones, es decir, que sea consistente con éstas; que tenga la capacidad de reaccionar bien ante situaciones excepcionales (tolerancia a fallas) y recuperarse de las mismas fallas en el menor tiempo posible; y la capacidad de estar operativo el mayor tiempo posible. Todo proporciona al sistema una mayor confiabilidad.

Eficiencia: En este sentido a la aplicación le corresponde permitir que la información almacenada pueda ser consultada y actualizada permanentemente, sin que se afecte el tiempo de respuesta. El sistema debe

poseer la capacidad de dar respuesta al acceso de cada usuario, con un tiempo de respuesta aceptable y uniforme, en la medida de las posibilidades tecnológicas posea el grupo de desarrollo. Aunque una práctica muy común en los desarrolladores es la optimización excesiva, lo importante es que el SIG sea implementado manteniendo un balance adecuado entre eficiencia y corrección. Obteniendo una buena eficiencia se logra que el software tenga la capacidad de hacer buen uso de los recursos que manipula, teniendo en cuenta principalmente los recursos de hardware, tiempo de procesador y ancho de banda.

Mantenibilidad: Es la capacidad del producto de software para ser modificado. Las modificaciones pueden incluir correcciones, mejoras o adaptaciones del software a los cambios del entorno y en los requerimientos y especificaciones funcionales. Dado el uso de la arquitectura orientada a objeto la cual permite que los objetos estén débilmente acoplados, la modificación de ellos puede efectuarse sin comprometer a los otros, lo que, al igual que la arquitectura basada en componentes, como se implementan interfaces bien definidas y con bajo acoplamiento, permite el desarrollo de nuevos módulos en el sistema sin impactar otras partes del mismo.

Potabilidad: El sistema se implementará sobre tecnología libre, y sobre un entorno de desarrollo con el que se pueda generar aplicaciones que permitan ser instaladas y ejecutadas en diferentes sistemas operativos. Se puede dividir además los servidores proveedores de datos del sistema que los consume o de la aplicación cliente, para facilitar la movilidad de este último y posibilitar un balance de carga adecuado para el Sistema, es decir que el SIG pueda ser instalado en cualquier ordenador personal. Como se define el uso de una arquitectura basada en componentes, se hace posible que la facilidad de instalación permita actualizar el sistema con nuevos subsistemas sin impacto en el mismo o en sus componentes. Estos elementos posibilitan la instalabilidad, reemplazabilidad, adaptabilidad y coexistencia del sistema.

4.5 Vista Lógica

Esta vista soporta el análisis y la especificación de los requisitos funcionales que son los que el sistema suministra en términos de prestaciones a los usuarios finales. La notación a usar es UML, y dentro de esta diagramas de clases y paquetes. El estilo es el Orientado a Objetos. Todo ello se traduce en que la vista

lógica es la responsable de describir las clases más importantes que formarán parte del ciclo de desarrollo además de los paquetes del sistema y las relaciones existentes entre ellos.

4.5.1 Diagrama de clases por paquetes y subsistemas de la Vista Lógica

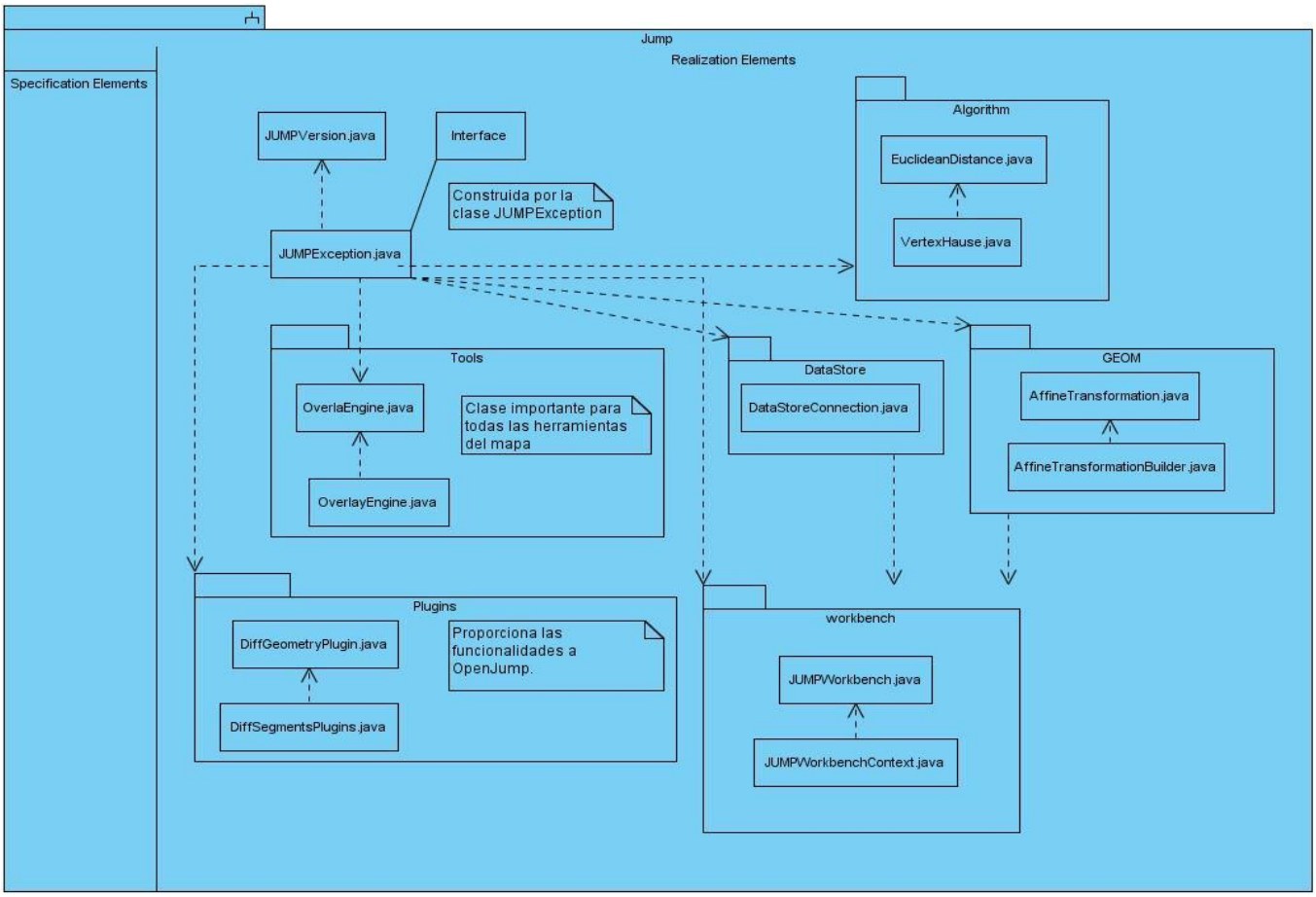


Figura 4 Diagrama de Vista Lógica.

4.6 Vista de Implementación

El Modelo de Implementación es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la

relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos. (MeRinde 2010)

Uno de los artefactos que se generan en este modelo es el Diagrama de Componentes, el cual muestra un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones. Se utiliza para modelar la vista estática de un sistema y muestra la organización y las dependencias lógicas entre un conjunto de componentes software.

En el Diagrama de componentes que aparece en el epígrafe 4.7.1 muestra la relación de dependencia que existe entre los paquetes JFreeComponents y Librerías. El primero contiene todos los componentes a desarrollar (en este caso solo se muestran dos componentes, para que el diagrama no se haga engorroso), los cuales se agrupan en el mismo paquete debido a sus similares características. El segundo agrupa las librerías JExcelAPI y JFreeChart, la cual depende a su vez de la librería Jcommon.

Paquete JUMP

En JUMP estarán las clases que organizan a OpenJump, de ahí que se encuentre en el mismo la clase JUMConeccion.Java la cual tiene la responsabilidad de construir la interfaz principal del sistema tomando las funcionalidades de los demás componentes de implementación.

Paquete Tools

Contará con las clases que den vida a OpenJump como las herramientas de MapTool. Conjunto de herramientas para manipular el mapcanvas (dibujo que se puede realizar sobre el mapa en cualquier capa, pudiéndose realizar polígonos).

Paquete Plugin

Posee la implementación de los plugins, importantes para darle funcionalidades al SIG, contiene la clase DiffSegmentsPlugin.JAVA de la cual heredan los demás plugins a desarrollar para OpenJump.

Paquete Workbench

Contiene las clases principales de OpenJump, como por ejemplo Workbenchcontex.JAVA, la cual proporciona las constantes globales.

Paquete DataStore

Se encuentran las clases que brindan información para elaborar la vista de los mapas.

Paquete Algorithm

Estarán las clases para las configuraciones de los dibujos.

4.7 Vista de Componentes

Un componente es una parte física y reemplazable de un sistema.

Las clases representan abstracciones lógicas. Los componentes son elementos físicos del mundo real. Un componente es la implementación física de un conjunto de otros elementos lógicos, como clases y colaboraciones.

La relación que se establece entre componentes en un diagrama de componentes es de dependencia. Esto quiere decir que un componente necesita del otro para completar su definición. (Molina, 2004/2005)

4.7.1 Diagrama de componentes de OpenJump

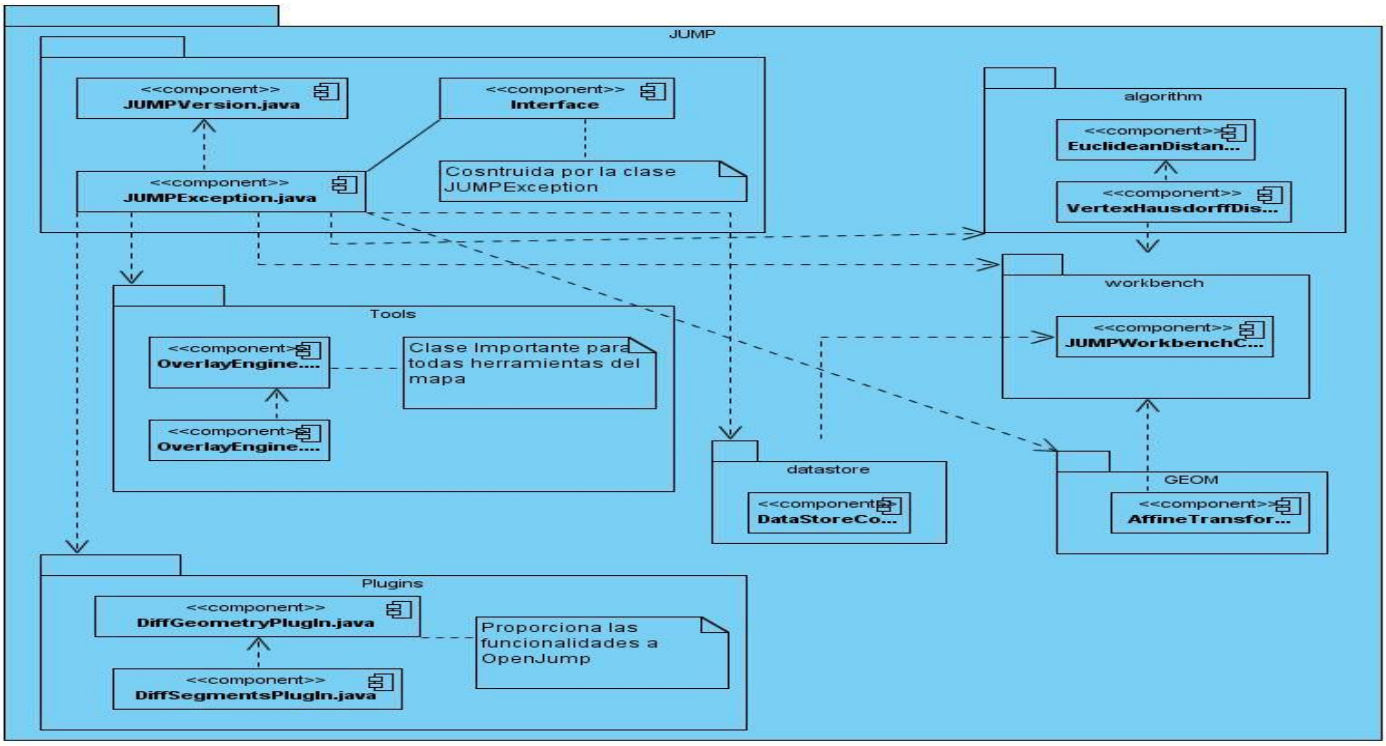


Figura 5 Diagrama de Componentes.

4.8 Vista de Despliegue

En un diagrama de despliegue se muestra la distribución física de los objetos, es decir, la configuración de los elementos de procesamiento en tiempo de ejecución y los componentes software (procesos y objetos que se ejecutan en ellos). A continuación se muestra el diagrama de despliegue de la solución propuesta:

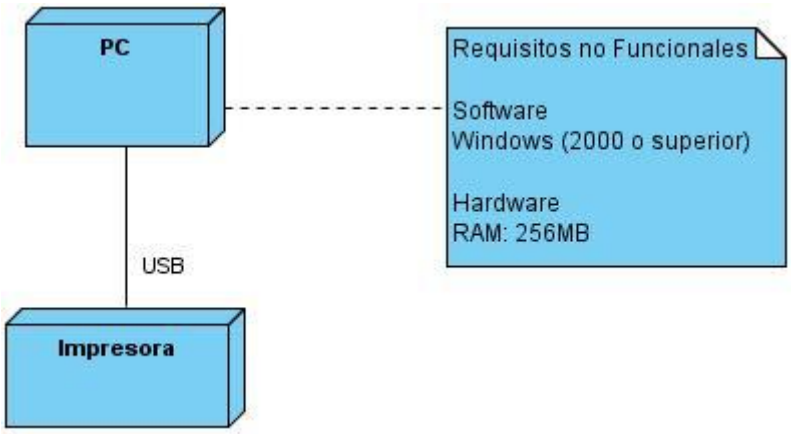


Figura 6 Diagrama de Despliegue.

2.2 Conclusiones

- Como resultado de este capítulo se explicó el uso de los distintos patrones arquitectónicos que dieron una estructura a la solución propuesta.
- Se identificaron los atributos que componen la calidad de la arquitectura propuesta para OpenJump.
- Se realizó el Diagrama de clases por paquetes y subsistemas de la Vista Lógica.
- Se hizo el diagrama de componentes de OpenJump.
- Se mostró el modelo de despliegue permitiendo un entendimiento de la distribución física y lógica del sistema.

Conclusión General

- Se realizó un estudio del arte sobre los diferentes sistemas similares y se seleccionó las tecnologías para modelar el SIG OpenJump.
- Se realizó el análisis del SIG OpenJump para la Universidad de las Ciencias Informáticas abarcando:
    - Modelo de dominio.
    - Requisitos funcionales y no funcionales.
    - Actores y casos de usos del sistema.
    - Modelos de casos de uso del sistema.
    - Descripción textual de los casos de uso del sistema.
    - Se efectúo los diagramas de clases del análisis.
    - Se modeló los diagramas de colaboración del análisis.
- Se diseñó el sistema obteniendo así los patrones de diseño necesario y conveniente para el sistema propuesto así como la realización del diagrama de paquetes para el SIG OpenJump.
- Se definió los estilos arquitectónicos importantes para modelar el sistema.
- Se realizó el diagrama de clases por paquetes y subsistemas de la vista lógica.
- En la vista de componentes se diseño dicho diagrama.
- En la vista de implementación se modeló el despliegue.

Con el cumplimiento de todas las tareas asignadas en la investigación así como la creación de todos los artefactos necesarios e imprescindibles se dejan las bases sentadas para el desarrollo de una aplicación SIG de escritorio cubano siguiendo con todas las propuestas vistas en este trabajo, tanto de análisis, diseño y arquitectura.

RECOMENDACIONES

Luego de haber concluido el análisis, diseño y arquitectura del sistema propuesto y cumplido los objetivos trazados, se plantean las siguientes recomendaciones:

Utilizando toda la documentación generada en este trabajo construir un Sistema de Información Geográfica basado en la herramienta OpenJump.

El refinamiento constante de la arquitectura propuesta, durante el ciclo de desarrollo.

Incorporación de plugins capaces de generar nuevas funcionalidades definidas en el análisis.

▪ Bibliografía Referenciada

Berry, J.K. 1993. *Beyond Mapping: Concepts, Algorithms and Issues in GIS. Fort Collins, CO: GIS World Books.* 1993.

Christl, Arnulf. 2004. *Adopting OS GIS technology in heterogeneous environments, Providing decision makers with arguments beyond cost. Procedings of the 2nd MapServer Users Meeting.* 2004.

Jorge Gaspar Sanz, Miguel Montesinos. *Panorama Actual del ecosistema de Software libre para SIG.* Valencia : s.n.Z

2009. KOSMO. *KOSMO.* [En línea] 2009. [Citado el: 1 de 12 de 2010.] http://www.tecnomaps.com.

2003. *Licencia Pública General de GNU.* 2003.

Miguel Montesinos Lajara, Jorge Gaspar Sanz Salinas. 2005. *Panorama Actual del Ecosistema de Software libre para SIG v4.0.* 2005.

P, Ramsey. 2003. User Friendly Desktop Internet GIS (uDyg) forOpenGIS Spatial Data Infraestructure. [En línea] 2003. [Citado el: 10 de 11 de 2010.] http://udig.refractions.net.

2006. PortalGrass. *PortalGrass.* [En línea] 2006. [Citado el: 1 de 12 de 2010.] http://www.scribd.com.

2009. Quantum GIS Guia de usuario e instalacion v0.9.1. *Quantum GIS Guia de usuario e instalacion v0.9.1.* [En línea] 2009. [Citado el: 22 de 11 de 2010.] http://www.qgis.org.

(Abad, 2008.)

P.Letelier. 2006. Introducción Proceso de Desarrollo de SW. *Portal Desarrollo de Software.* [En línea] 15 de Febrero de 2006. [Citado el: 21 de Enero de 2010.] https://pid.dsic.upv.es/C1/Material/default.aspx.

Wiccsi. Nuevos Entornos y Plataformas para el Desarrollo de Software. *Wiccsi.* [En línea] [Citado el: 12 de Enero de 2010.] http://www.wiccsi.com.ar/proyectos/14.1.pdf.

Ambler, Scott W. 2009. Ambysoft.com. *Ambysoft.com.* [En línea] 2009. [Citado el: 17 de 2 de 2011.] http://www.ambysoft.com/unifiedprocess/agileUP.html.

Mitecnologico. Definición de Lenguaje de programación. *Mitecnologico.* [En línea] [Citado el: 15 de Enero de 2010.] http://www.mitecnologico.com/Main/DefinicionDeLenguajeDeProgramacion.

Referencias Bibliográficas

NetBeans. Información de la versión de NetBeans IDE 6.8. *NetBeans.* [En línea] [Citado el: 12 de Enero de 2010.] http://netbeans.org/community/releases/68/index_es.html.

Petra.euitio. Entornos de Desarrollo Integrado. *Sitio Web de la E.U de Ingeniería Técnica informática de Oviedo.* [En línea] [Citado el: 25 de Enero de 2011.] http://petra.euitio.uniovi.es/~i1667065/HD/documentos/Entornos%20de%20Desarrollo%20Integrado.pdf.

2011. Slideshare. [En línea] SlideShare Inc, 2011. [Citado el: 17 de 2 de 2011.] http://www.slideshare.net/vanquishdarkenigma/visual-paradigm-for-uml.

WorldLingo. 2011. WorldLingo. *WorldLingo.* [En línea] WorldLingo Translations LLC, 2011. [Citado el: 18 de 2 de 2011.] http://www.worldlingo.com/ma/enwiki/es/PostGIS.

Cibertec. 2007. *UML-Analisis-del-negocio.* 2007.

Oliva, Rafael. *Los metadatos Geograficos: Actualidad y Estandares.* 2006.

Oriente, Geominera. *Datos Geologicos. Santiago de Cuba.* 2007.

Ourense, Grupo de Usuarios de Linux de. *Introducción a Linux y al Software Libre.* 2003.

slideshare.net. slideshare.net. *Herramienta Case.* [En línea] [Citado el: 25 de septiembre de 2010.] www.slideshare.net/guest15d257/herramientas-case-508428.

Departamento Docente Central de Ingeniería de Software. *Conferencia 4.Flujo de trabajo de requerimientos.* Ciudad de La Habana : s.n.

WordReference.com. Diccionario de la lengua española. *WordReference.com.* [En línea] WordReference.com. http://www.wordreference.com/definicion/informatizar.

eva.uci.cu. eva.uci.cu. *Estimación de Esfuerzo.Proyecto de Software.* [En línea] [Citado el: 5 de mayo de 2011.] http://eva.uci.cu/file.php/659/Documentos/Materiales_complementarios/UD_1_Procesos/Sobre_Estimacion/Estimacion_de_esfuerzo_proyectos_de_software.pdf.

Departamento de proyectos e ingeniería. Los Sitemas de información Geográfica.

Sitio Web oficial del Ministerio de la Informática y las Comunicaciones. *Informatización de la Sociedad.* [En línea] http://www.mic.gov.cu/.

Referencias Bibliográficas

Piattini, M.G. y Calvo, J.A. Manzano. *Análisis y diseño detallado de aplicaciones informáticas de gestión.* 1996.

Pressman, Roger. *Ingeniería de Software.*

Pressman, Roger S. *ngeniería de Software. Un enfoque práctico.* s.l. : Quinta Edición, 2005.

Reynoso, Carlos Billy. *Introducción a la Arquitectura de Software.* s.l. : UNIVERSIDAD DE BUENOS AIRES, Marzo de 2004.

Oliva, Rafael. *Los metadatos Geograficos: Actualidad y Estandares.* 2006.

Oriente, Geominera. *Datos Geologicos. Santiago de Cuba.* 2007.

Ourense, Grupo de Usuarios de Linux de. *Introducción a Linux y al Software Libre.* 2003.

slideshare.net. slideshare.net. *Herramienta Case.* [En línea] [Citado el: 25 de septiembre de 2010.] www.slideshare.net/guest15d257/herramientas-case-508428.

csi.map.es. csi.map.es. *CASE. Ventajas e Inconvenientes. Consejo Superior de Administracion Electronica.* [En línea] [Citado el: 28 de enero de 2010.] www.csi.map.es/csi/silice/Global75.html..

freedownloadmanager. freedownloadmanager. *Visual Paradigm Internacional.* [En línea] [Citado el: 10 de febrero de 2011.] www.freedownloadmanager.org/es/downloads/Paradigma_Visual_para_UML_%28M%C3%8D%.

sistemaronald.blogspo. sistemaronald.blogspo. *Introduccion al Rational Rose.* [En línea] [Citado el: 12 de marzo de 2010.] http://sistemaronald.blogspot.com/2008/08/introduccion-rational-rose.html..

Briones, Pedro. *Curso de PostGIS.* 2007.

Cristina Lazalde Miranda ,Mayra Yanin Miranda Valles ,Jesús Robles Domínguez ,Esthela Gomez Hinojosa ,Karla Domínguez Hernandez. www.mitecnologico.com. *Herramientas Cases.* [En línea] [Citado el: 5 de mayo de 2011.] http://www.mitecnologico.com/Main/HerramientasCase.

Departamento de proyectos e ingeniería. Los Sitemas de información Geográfica.

petra.euitio.uniovi.es. petra.euitio.uniovi.es. *petra.euitio.uniovi.es.* [En línea] [Citado el: 2011 de 2 de 15.]

Medina, Carlos Gustavo Mejía. *Apuntes de un investigador.* 2000.

Referencias Bibliográficas

Estrugo Lahera, Evelyn y Cuellar Rodríguez, Dany. *Diseño e implementación del portal seb para la dirección general de prevención al delito.* Ciudad de la Habana : s.n., Junio 2010.

Paul, Pacheco Toscano Santiago. *"Análisis, diseño e implementación de un sistema para el envío de publicidad dirigida a través del servicio de telefonía móvil SMS para la empresa One Shot Marketing Cia. Ltda.".* 2001.

Ivar, Jacobson. *El Proceso Unificado de Desarrollo de Software. Pearson Addisson-Wesley.* 2004.

Case, Instituto Nacional de Estadísticas e Informática. Herramientas. *Talleres de la Oficina de Impresiones de la Oficina Técnica de Difusión.* Noviembre, 2009.

Bibliografía

1. **Departamento Docente Central de Ingeniería de Software.** *Conferencia 4.Flujo de trabajo de requerimientos.* Ciudad de La Habana : s.n.

2. **WordReference.com.** Diccionario de la lengua española. *WordReference.com.* [En línea] WordReference.com. http://www.wordreference.com/definicion/informatizar.

3. **eva.uci.cu.** eva.uci.cu. *Estimación de Esfuerzo.Proyecto de Software.* [En línea] [Citado el: 5 de mayo de 2011.] http://eva.uci.cu/file.php/659/Documentos/Materiales_complementarios/UD_1_Procesos/Sobre_Estimacion/Estimacion_de_esfuerzo_proyectos_de_software.pdf.

4. **Departamento de proyectos e ingeniería.** Los Sitemas de información Geográfica.

5. Sitio Web oficial del Ministerio de la Informática y las Comunicaciones. *Informatización de la Sociedad.* [En línea] http://www.mic.gov.cu/.

6. **Piattini, M.G. y Calvo, J.A. Manzano.** *Análisis y diseño detallado de aplicaciones informáticas de gestión.* 1996.

7. **Pressman, Roger.** *Ingeniería de Software.*

8. **Pressman, Roger S.** *ngeniería de Software. Un enfoque práctico.* s.l. : Quinta Edición, 2005.

9. **Reynoso, Carlos Billy.** *Introducción a la Arquitectura de Software.* s.l. : UNIVERSIDAD DE BUENOS AIRES, Marzo de 2004.

10. **Mora, Roberto Canales.** Patrones de GRASP. *Patrones de GRASP.* [En línea] 2003-2004. . [Citado el: 20 de 3 de 2011.]

11. **Ochoa, Sergio.** Introducción a los patrones(Diseño y arquitectura). *Introducción a los patrones(Diseño y arquitectura).* [En línea] 2005. [Citado el: 2 de 4 de 2011.]

12. **Mestras, Juan Pavón.** *Estructura de las Aplicaciones Orientadas a Objetos.* Universidad Complutense Madrid : s.n.

13. **Castro, Jon.** *JAVA 2 ENTERPRISE.* 2000.

ANEXO 3: **Descripción del CU: Crear mapas temáticos**

Caso de uso: Crear Mapas Temáticos.

Caso de uso	Crear Mapas Temáticos
Actores	Administrador
Propósito	Este caso de uso tiene como objetivo darle la posibilidad al usuario de hacer una búsqueda en el mapa según los parámetros o características que el desee.
Resumen	El caso de uso es iniciado cuando el usuario desea obtener la información asociada a una zona geográfica, según los parámetros que el este usuario decida insertarle.

Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El caso de uso se inicia cuando el actor selecciona la Crear Mapa Temático. 2. El usuario llena los campos de la ventana desplegada para crear el mapa temático. 3. El usuario da clic en crear mapa.	1.1 El sistema despliega una ventana con la opción de darle nombre y color a los parámetros a insertar en el mapa. 3.1 El sistema muestra el mapa temático**.**

ANEXO 4: **Descripción del CU: Cerrar Cesión.**

Caso de uso: Cerrar Cesión.

Caso de uso	Cerrar Cesión
Actores	Usuario

Propósito	Este caso de uso tiene como objetivo que el usuario una vez terminado el trabajo con el SIG cierre su cesión para mantener la seguridad del sistema.
Resumen	El caso de uso es iniciado cuando el usuario desea salir de la aplicación, este se puede hacer desde cualquier vista.
Flujo normal de eventos	
Acción del actor	**Respuesta del sistema**
1. El caso de uso se inicia cuando el actor selecciona el botón cerrar cesión.	1.1 El sistema cierra la cesión por la que entró el usuario guardando sus cambios.

ANEXO 5: **Descripción del CU: Gestionar Usuarios.**

Caso de Uso:	**Gestionar usuarios.**
Actores:	Administrador
Resumen:	El caso de uso se inicia cuando el administrador del sistema decide gestionar usuarios y permite insertar, eliminar, modificar los datos de un usuario existente, como son el nombre, el usuario, la contraseña y seleccionar el grupo al

	cual va a pertenecer y finaliza el caso de uso.
Precondiciones:	El administrador debe haberse autenticado previamente y debe existir un grupo en el sistema.
Referencias	
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso comienza cuando el administrador escoge la opción Gestionar usuarios.	1.1. El sistema muestra la interfaz donde el administrador puede insertar, eliminar o modificar un usuario. 1. Para insertar un usuario ir a la sección **"Insertar usuario"**. 2. Para modificar los datos de un usuario ir a la sección **"Modificar usuario"**. 3. Para eliminar un usuario ir a la sección **"Eliminar usuario"**.
Sección "Insertar usuario"	
Acción del Actor	**Respuesta del Sistema**

2. El usuario escoge la opción Insertar usuario.	2.2. El sistema muestra la interfaz para insertar los datos.
3.El administrador del sistema entra los datos del usuario para luego solicitar su adición al sistema.	3.1. Verifica que no existan campos en blanco y que los datos son correctos. 3.2.Verifica que no exista el usuario. 3.3.Guarda los datos del nuevo usuario. 3.4.Muestra mensaje indicando que el usuario se ha creado y guardado correctamente y de esta forma finaliza el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1. Muestra un mensaje de error en caso que existan campos en blanco, o que los datos sean incorrectos. 3.2. Muestra en mensaje de error advirtiendo que el usuario ya existe.
Poscondiciones	Se ha creado un nuevo usuario en el sistema.
Sección "Modificar Usuario"	
Acción del Actor	**Respuesta del Sistema**

2-El usuario escoge la opción Modificar usuario.	2.1-El sistema muestra una interfaz introducir usuario que se desea modificar.
3-El administrador introduce el usuario que quiere modificar.	3.1-El sistema verifica la existencia del usuario. 3.2-El sistema muestra los datos del usuario seleccionado.
4-Modifica los parámetros deseados y solicita modificar al usuario.	4.1-Verifica que los datos sean correctos. 4.2-Se actualizan los datos del usuario en la Base de Datos y finaliza el caso de uso.
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1-Se emite un mensaje de error porque no existe el usuario.
	4.1-El sistema muestra un mensaje de error cuando los datos no son correctos.
Poscondiciones	Se ha modificado un usuario existente en el sistema.
Sección "Eliminar usuario".	
Acción del Actor	**Respuesta del Sistema**

2-El usuario selecciona la opción eliminar usuario.	2.1-El sistema muestra una interfaz para introducir el usuario que desea eliminar.
3-El administrador selecciona introduce el usuario que desea eliminar.	3.1-El sistema verifica que el usuario existe. 3.2-Se elimina el usuario, se actualiza la Base de Datos y finaliza el CU.
Flujos alternos	
Acción del Actor	**Respuesta del Sistema**
	3.1-Se emite mensaje de error porque no existe el usuario.
Poscondiciones	Se ha eliminado un usuario existente en el sistema.