UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

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

TÍTULO: ¨Desarrollo de un Sistema de Información Geográfica utilizando Google Web Toolkit y OpenLayers.¨

AUTORES: Yanisley González González
Midia Elena Sierra Dumitrescu

TUTOR: Ing. Betsy Cuza García
CO-TUTOR: Ing. Yordanys Piñeiro Gómez

La Habana, 20 de Junio de 2011.
¨Año 53 de la Revolución.¨

"El hombre debe transformarse al mismo tiempo que la producción progresa; no realizaríamos una tarea adecuada si fuéramos tan sólo productores de artículos, de materias primas y no fuéramos al mismo tiempo productores de hombres."

Ernesto Che Guevara

DECLARACIÓN DE AUTORÍA

Declaramos que somos las únicas autoras de este trabajo y autorizamos al Departamento Geoinformática de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autora: Yanisley González González

_____

Autora: Midia Elena Sierra Dumitrescu

_____

Tutora: Ing. Betsy Cuza García

_____

DEDICATORIA

De Yanisley:

Dedico mi trabajo de diploma a todos los que han hecho realidad el sueño de graduarme hoy en esta universidad. Es dedicado con mucho orgullo a mis padres por su infinita confianza y su cariño constante, a mis hermanos por todo su amor y apoyo, a mi novio por su eterno amor, por su confianza y dedicación, por su ayuda incondicional durante toda mi carrera, por dedicarme todo el tiempo del mundo cuando lo he necesitado. También es dedicado a nuestro Comandante Fidel Castro por ser nuestro guía en cada momento.

De Midia:

Dedico este trabajo a todos los que me han formado como persona y profesional.

AGRADECIMIENTOS

De Yanisley:

Quiero agradecer infinitamente a todos los que han hecho posible que hoy pueda sentir la dicha inmensa de graduarme en la Universidad de las Ciencias Informáticas:

A nuestro querido Comandante Fidel Castro y a nuestra invicta Revolución por toda su confianza en los estudiantes de la UCI.

A mis queridísimos padres: a mi mamá Yaquelín y mi papá William Gricel por su infinito amor, por toda su confianza y comprensión, por ser los mejores padres del mundo, por permitirme ser niña a pesar de haber crecido, por sus preocupaciones en todo momento, por ser excelentes padres ante todo, mil gracias por darme todo en la vida.

A mi hermana Yuri por ser única y especial, por ser la mejor del mundo, por su cariño incomparable, por su comprensión, por ser la mejor hermana y amiga que he conocido, porque sin ella mi vida no fuera nada.

A mi hermano Yanier por quererme tanto, por pensar en sus hermanas por encima de todo, por ser ya otro hombre de la casa.

A mi queridísimo novio por su eterno amor, por permitirme ser parte de su vida durante estos cinco años de la carrera, por toda su confianza y constancia, por su comprensión cuando más la he necesitado, por su ayuda incondicional en todo momento, por ser un tesoro muy especial de mi vida, gracias por existir.

A mi prima Nelda por toda su ayuda y preocupación durante mi vida en la universidad.

A mi tío Onel por ser guía de mis pasos revolucionarios, por ser un tío ejemplar.

A mis abuelitos por quererme tanto, por esperar con paciencia que me gradúe de ingeniera, gracias por su confianza constante.

A mi suegra bella por ser tan cariñosa, por quererme mucho y disfrutar de mi felicidad con su hijo.

A mi suegro Raúl y a Frank por todo su cariño, por permitirme ser parte de su familia.

A mi amiga Dilenis por ser como una hermana para mí, por soportarme cada día, por demostrarme el valor de la verdadera amistad, te quiero muchísimo.

A mi amigas Yessy y Yordanys por compartir conmigo momentos especiales, por ser también para mí un pedacito importarte de mi corazón, por ser de las personas que nunca olvidaré.

A mi amiga Ivelin por también estar presente en cada momento, por ser tan especial, por formar parte de mí.

A mi amiga Yarelis por ser tan preocupada y ser una amiga incondicional.

A mi amiga Karem por llegar a mi vida y compartir alegrías en días difíciles para mí.
A mis amigas de toda la vida por estar siempre presentes: Anaidy, Keylan y Linet.
A mis cuñados Rayner y Ray por ser parte de mi familia, por ser tan especiales.
A toda mi familia por ser tan grande y unida, por ser tan maravillosa, por confiar tanto en mí.
A todas mis amistades que son muchas: Maité, Ledier, Manuel Alejandro, Mara, Yeisy, Elisa, Viviana, Krysna, Lisandra, Anita, Lisbet, Eddy, a todos los que han compartido conmigo el grupo y el apartamento.
A mi tutora Betsy por todo su apoyo, por ser además una amiga.
A nuestro tribunal por todo su compromiso, por cada acertado consejo.
A mi compañera de tesis por compartir conmigo todo este tiempo de estrés, compromiso y dedicación al trabajo de diploma, gracias por estar presente porque te quiero como una amiga.
A todos, muchas gracias.

De Midia:

Quiero agradecer a todas las personas que con su presencia y acciones han contribuido a hoy yo me gradúe en esta universidad:

Ante todo a nuestro querido Comandante Fidel Castro y a nuestra invicta Revolución por toda su confianza en los estudiantes de la UCI.
A la mejor madre del mundo, que nunca se ha rendido conmigo para hacerme quien soy hoy, que se convirtió en mi mejor amiga, por su infinita comprensión, por su infinito apoyo, por su incansable lucha en hacerme una mejor persona, pero sobre todo por su eterno amor. Gran parte de lo que soy se lo debo a ella.
A mi papá, por ser una muralla protectora, por ser mi mejor guía y ejemplo entre todos, por ser el gran hombre que admiro, por estar presente para mí a toda hora cuando lo necesito. Por su consejo sabio, por su paciencia, por sus enseñanzas incontables.
A mi hermano querido Jean, por ser el mejor hermano que yo puedo desear.
A Mi cuñada Yanet, por ser como una hermana para mí, por darme su apoyo siempre.
A Elena por su ayuda en todos estos años, por sus concejos, por su amistad.
A Sandra, mi hermanita querida, por formar parte de mi vida.
A mis mejores amigas en el planeta, Zenia, Jany y Roselí, mis hermanas escogidas por mí. Por conocerme en el momento más oportuno, cuando más las necesitaba y desde entonces brindarme su

apoyo incondicional, su paciencia, su comprensión, su compañía. Por estar a mi lado en cada momento, en cada problema, en cada alegría, en cada tristeza.

A David, por ser mi tutor personal, por su guía, por su apoyo, por su preocupación.

A Nancy, por ser tan preocupada y atenta como otra mama.

A mi amiga Yeisy por ser mi confidente.

A todas mis amistades por estar presentes en diferentes momentos a lo largo del camino: Lisandra Hernández, Yassel, Olaidis, Yadira, Ada, Osvaldo, Milenis, Lisbet.

A Yanisley, mi compañera de tesis. Me acuerdo cuando te conocí, una personita fuerte pero a la ves asustada por todo. Hicimos química inmediata y desde entonces me regalaste tu amistad. Gracias por tu compromiso, por tu resistencia, por tu entrega, por compartir este trabajo conmigo.

A Pupo por su preocupación y ayuda en todo momento.

A mi tutora Betsy por su apoyo y preocupación, por ser también una amiga.

A mi tribunal, por su excelente consejo y guía.

RESUMEN

Los Sistemas de Información Geográfica (SIG) se han convertido en una herramienta importante para la toma de decisiones en cualquier rama de la sociedad. En el Centro de Desarrollo GEySED de la Facultad 6 de la Universidad de las Ciencias Informáticas se encuentra el Departamento Geoinformática, donde se implementan sistemas de este tipo, dentro de ellos está el desarrollo de SIG genéricos que integren diferentes tecnologías y que implementan funcionalidades básicas.

El presente trabajo tiene como premisa desarrollar un SIG integrando las tecnologías Google Web Toolkit (GWT) y OpenLayers. Para ello se utilizó como metodología de desarrollo de software a RUP, como lenguaje de modelado a UML, como herramienta CASE Visual Paradigm, lo que permitió un correcto entendimiento de todo el proceso ingenieril que sentó las bases para la implementación de la aplicación.

El desarrollo de esta investigación tiene un aporte importante para el Departamento Geoinformática, puesto que cuando un cliente solicite un producto SIG implementado con estas tecnologías sólo se tendrá que adaptar la solución existente a las condiciones del cliente. Esta estrategia de realizar SIG genéricos, hace que el departamento agilice su producción y que el cliente reciba el software en un tiempo corto.

PALABRAS CLAVES: Departamento Geoinformática, SIG, GWT, OpenLayers**.**

ÍNDICE DE TABLAS Y FIGURAS

ÍNDICE

INTRODUCCIÓN

Con el desarrollo del hombre se han ampliado sus fronteras en todas las ramas de la sociedad. En aras de delimitar de cierto modo estas fronteras surgió la geografía como una ciencia del conocimiento, en dicha rama el volumen de información en el mundo es cada vez mayor, por lo que esto conduce a utilizar Sistemas de Información, y si se trata de ubicar estos datos geoespacialmente entonces se pueden utilizar Sistemasde Información Geográfica (SIG).

Los Sistemas de Información Geográfica (SIG) son el"conjunto de métodos, herramientas y datos que están diseñados para actuar coordinada y lógicamente para capturar, almacenar, analizar, transformar y presentar toda la información geográfica y de sus atributos con el fin de satisfacer múltiples propósitos. Esta tecnología permite gestionar y analizar la información espacial, y surgió como resultado de la necesidad de disponer rápidamente de información para resolver problemas y contestar a preguntas de modo inmediato".(MCDONNELL, 1997)

"Nuestro país, se encuentra inmerso en un proceso de transformaciones desde el punto de vista tecnológico, dirigidas a lograr lugares cimeros a nivel mundial en el desarrollo de este campo, y dentro de muchas de estas tecnologías que hoy en día se estudian, se encuentran las relacionadas con el desarrollo de los Sistemas de Información Geográfica, su vinculación y adaptación con las alternativas libres (software libre) existentes en el mundo. Se han creado diversas instituciones a nivel nacional que contribuyen al crecimiento informático del país, y como parte de los aportes que cada día sustentan ese desarrollo se encuentra el estudio de los SIG". (Rodríguez, 2008)

La Universidad de las Ciencias Informáticas (UCI) es uno de los centros de altos estudios más importantes del país a pesar de ser muy joven; la misma combina el estudio, la investigación y la producción de forma organizada. Es una de las instituciones que existe en el país para el desarrollo de su economía y el proceso de informatización. La universidad cuenta con diversos Centros de Desarrollo que se encargan de potenciar el trabajo en diferentes líneas. Uno de ellos es el centro de desarrollo GEySED de la Facultad 6; el mismo está compuesto por dos áreas productivas: el Departamento Señales Digitales y el Departamento Geoinformática, este último se destaca por desarrollar Sistemas de Información Geográfica. Tiene una fuerza productiva heterogénea de profesores y estudiantes desde 2do a 5to año. Desarrolla sus aplicaciones bajo tecnologías OpenSource y cumpliendo con los estándares OpenGIS. Incluye la

implementación de soluciones informáticas para la toma de decisiones empresariales basado en SIG, desarrollo de soluciones sobre bases de datos espaciales, implementación de servicios de mapas, administración de metadatos geográficos, soluciones para dispositivos móviles, desarrollo de SIG genéricos, entre otros.

Este Departamento tiene como misión desarrollar productos, servicios y soluciones informáticas permitiendo un posicionamiento en el mercado nacional e internacional, por lo que trata cada día de redoblar los esfuerzos para cumplir con el compromiso de cada cliente, puesto que son muchos los que se acercan de diversas empresas con la necesidad inmediata de soluciones SIG. Estas peticiones son cada vez más numerosas y resulta imposible atenderlas todas al mismo tiempo,puesto que cuando se conceptualiza un proyecto nuevo, la elección de marcos de trabajo, entornos de desarrollo y otras herramientas consumen tiempo y recursos tanto humanos como materiales.

Como es mucho el trabajo del Departamento Geoinformática, el mismo ha tomado la estrategia de realizar SIG genéricos que integren diferentes tecnologías y herramientas posibles. Estos se implementan con funcionalidades básicas de todo SIG para demostrar al final el correcto funcionamiento de los mismos. Esta idea llevada a cabo resulta un gran avance para los proyectos del departamento, pues cuando un cliente solicite un sistema de información geográfica, especificando alguna de estas tecnologías integradas, sólo hay que adaptar la solución existente a las necesidades del cliente.

Partiendo de la situación expuesta con anterioridad se ha identificado como **problema científico:** ¿Cómo agilizar la respuesta del Departamento Geoinformática a clientes interesados en soluciones SIG en la web que utilizan Google Web Toolkit (GWT) y OpenLayers?

A partir de la problemática planteada anteriormente se puede establecer el **Objeto de Estudio** de la presente investigación, el cual se centra en el proceso de desarrollo de Sistemas de Información Geográfica, siendo el **Campo de Acción** el proceso de desarrollo de Sistemas de Información Geográfica utilizando GWT y OpenLayers. De aquí surge como **Objetivo General** lo siguiente: Desarrollar un Sistema de Información Geográfica genérico utilizando GWT y OpenLayers.

Por lo tanto la **Idea a defender** sería: El desarrollo de un sistema de Información Geográfica en la web utilizando GWT y OpenLayers, con la correspondiente documentación técnica del proceso ingenieril,

agilizará la respuesta por parte del Departamento Geoinformática a clientes interesados en el desarrollo de aplicaciones de este tipo.

Teniendo en cuenta lo antes mencionado se presentan las siguientes **tareas en la investigación**:

1. Caracterizar los Sistemas de información Geográfica.
2. Identificar los logros y limitaciones en los enfoques existentes sobre el desarrollo de Sistemas de Información Geográfica.
3. Caracterizar las tecnologíasde GWT y OpenLayers.
4. Seleccionar, argumentar y fundamentar la Metodología a utilizar en el proceso de desarrollo.
5. Seleccionar, argumentar y fundamentar las herramientas a utilizar en el proceso de desarrollo.
6. Identificar y describir las funcionalidades que debe brindar el sistema.
7. Elaborar la documentación técnica correspondiente al Modelo de Casos de Uso del Sistema del SIG.
8. Elaborar la documentación técnica correspondiente al Modelo de diseño del SIG.
9. Implementar el sistema.
10. Validar el resultado del sistema.

Con el objetivo de llevar a cabo la investigación se hizo necesario recurrir a los siguientes métodos científicos:

Métodos Teóricos

El método **Histórico-Lógico** se aplica para investigar la existencia de alguna plataforma que implemente un SIG con características similares a la solución que se propone en el presente trabajo.

El método de **Análisis y Síntesis**, se define con el objetivo de analizar los diversos documentos relacionados con el proceso de desarrollo de los Sistemas de Información Geográfica y las metodologías existentes para la construcción de los artefactos necesarios así como para realizar una síntesis de los mismos.

El método de **Modelación** se emplea para mostrar los diferentes diagramas y componentes que se construyen como resultado del proceso de Ingeniería de Software.

Métodos Empíricos

Las **Entrevistas** se realizan a líderes de proyectos y desarrolladores de SIG para recopilar información referente al proceso de desarrollo de los mismos en el mundo, en Cuba y fundamentalmente en el seno de la UCI, así como para identificar las herramientas óptimas destinadas a la construcción de dichos sistemas.

La **Observación** se usa para entender cómo se realiza el proceso de desarrollo de los Sistemas de Información Geográfica.

El trabajo de diploma cuyo **posible resultado** está dado en un Sistema de Información Geográfica genérico desarrollado con GWT y OpenLayers, incluyendo toda la documentación del mismo, y para que se garantice una implementación robusta de la aplicación, se estructura en cuatro capítulos, que son descritos a continuación:

El **Capítulo 1** contempla la fundamentación teórica de esta investigación, en la cual son expuestos los principales conceptos que contribuyen al mejor entendimiento del problema en cuestión, se especifican detalladamente todos los argumentos que esclarecen el objeto de estudio caracterizando los Sistemas de Información Geográfica y se identifican los logros y limitaciones en los enfoques existentes sobre el desarrollo de Sistemas de Información Geográfica.

Por su parte el **Capítulo 2** trata acerca de las tendencias y las tecnologías que se utilizan en la actualidad para el desarrollo de Sistemas de Información Geográfica. Se caracterizan las tecnologías GWT y OpenLayers. Luego se estudian un conjunto de metodologías que se manejan en el mundo para el desarrollo de software de forma general, con el objetivo de seleccionar de ellas, la más adecuada para guiar el proceso de desarrollo así como todas las herramientas que se utilizarán en el proceso evolutivo de este SIG.

En el **Capítulo 3** se realiza el modelo de dominio, así como la identificación de requisitos funcionales y no funcionales del sistema a construir, los funcionales se agrupan en casos de uso para conformar el Modelo de Casos de Uso del Sistema, también se realiza una descripción textual de los mismos.

Por último el **Capítulo 4** trata acerca de la construcción de la solución propuesta en términos de la realización de los artefactos que forman parte del flujo de trabajo de análisis y diseño, específicamente del diseño, y de las disciplinas de Implementación y Prueba.

Para finalizar se presentan las conclusiones, las recomendaciones, las referencias bibliográficas, la bibliografía, un glosario de términos y el conjunto de anexos para un mejor entendimiento de lo expuesto a lo largo de la investigación.

CAPÍTULO1: Fundamentación Teórica

Introducción

El manejo y control de los grandes volúmenes de datos adquiridos en la actualidad se hace imposible para el hombre, al menos, utilizando los métodos convencionales; es por eso que nacen las tecnologías de la información, precisamente éstas se basan en la realización de actividades humanas comunes pero asistidas por ordenadores. Estos datos constituyen el principal activo de cualquier sistema de información, una de las aplicaciones actuales y de gran auge a nivel mundial son los Sistemas de Información Geográfica (SIG), los cuales están destinados específicamente al trabajo con datos geoespaciales.

El enfoque de este capítulo es precisamente abordar en el diseño teórico de la investigación, iniciando con algunos conceptos asociados al tema en cuestión. Además se realiza un análisis de soluciones existentes sobre el desarrollo de SIG, donde se identifican los logros y limitaciones de cada sistema estudiado.

1.1 Conceptos asociados al dominio del problema

Con el objetivo de que el lector pueda tener una comprensión mayor de los temas que se relacionan en el capítulo, se describen detalladamente a continuación un grupo de conceptos asociados al dominio del problema, entre los que se destacan: Sistema de Información, Información Geográfica, Sistema de Información Geográfica, entre otros.

1.1.1 Sistema de Información

Un sistema de información es un conjunto de procedimientos ordenados que, al ser ejecutados, proporcionan información para apoyar la toma de decisiones y el control de la Institución. La información se define como una entidad tangible o intangible que permite reducir la incertidumbre acerca de algún estado o suceso.

1.1.2 Información geográfica

Se denomina Información Geográfica a aquellos datos espaciales georreferenciados, requeridos como parte de las operaciones científicas, administrativas o legales. Dichos datos espaciales suelen llevar una información alfanumérica asociada. (HUXHOLD, y otros, 1995)

1.1.3 Sistema de información Geográfica (SIG)

El término SIG procede del acrónimo de Sistema de Información Geográfica (en inglés GIS, Geographic Information System).Técnicamente se puede definir un SIG como una tecnología de manejo de información geográfica formada por equipos electrónicos (hardware) programados adecuadamente (software) que permiten manejar una serie de datos espaciales (información geográfica) y realizar análisis complejos con estos siguiendo los criterios impuestos por el equipo científico (personal).(HUXHOLD, y otros, 1995)

Los Sistemas de Información Geográfica son, en primer término, sistemas de información, es decir, programas diseñados para representar y gestionar grandes volúmenes de datos sobre ciertos aspectos del mundo. Utilizan datos georreferenciados mediante coordenadas espaciales asistidos por computador para la captura, almacenamiento, recuperación, análisis y despliegue de la información espacial, permitiendo procesar y generar nueva información derivada de la ya existente sobre la base de conceptos como localización, relación, descripción y base de datos relacional.(YAGÜEZ, y otros, 2002)

Se puede definir a un SIG como aquel método o técnica de tratamiento de la información geográfica que nos permite combinar eficazmente información gráfica (imágenes de mapas) y alfanumérica para obtener una información derivada sobre el espacio. Para ello, se cuenta tanto con las fuentes de información como con un conjunto de herramientas informáticas (hardware y software) que facilitan esta tarea.(Maguire, y otros, 1997)

1.1.4 Datos espaciales

Los datos espaciales son informaciones sobre la localización y las formas de un objeto geográfico y las relaciones entre ellos, normalmente con coordenadas y topología.

Los datos geográficos son entidades espacio-temporales que cuantifican la distribución, el estado y los vínculos de los distintos fenómenos u objetos naturales o sociales. Se refieren a entidades o fenómenos que cumplen los siguientes principios básicos:

- Tienen posición absoluta: sobre un sistema de coordenadas (x, y, z).
- Tienen una posición relativa: frente a otros elementos del paisaje (topología: incluido, adyacente, cruzado y otros).

- Tienen una figura geométrica que las representan (punto, línea, polígono).
- Tienen atributos que lo describen (características del elemento o fenómeno).

Estos datos en su conjunto son almacenados en bases de datos geográficos.(COMAS, y otros, 1993)

1.1.5 Cartografía

La cartografía es la ciencia que se encarga del estudio y de la elaboración de los mapas geográficos, territoriales y de diferentes dimensiones lineales. Además se puede definir como el conjunto de mapas producidos por una institución, relativos a un determinado territorio. Es la técnica geográfica que estudia la secuencia de etapas y procesos ejecutados para la visualización de un espacio geográfico mediante la producción de mapas, cartas, planos o croquis. (Ibero, 2009)

1.2 Proceso de Desarrollo de los Sistemas de Información Geográfica

El objeto de estudio de la presente investigación se centra en el proceso de desarrollo de los Sistemas de Información Geográfica. A continuación se hace una descripción general de estos sistemas donde se explica el funcionamiento de ellos mediante algunos aspectos que son relevantes en su desarrollo.

1.2.1 Descripción General

Los SIG funcionan con dos tipos diferentes de información geográfica: el modelo vector y el modelo raster, esto se puede ver en la Figura 1.

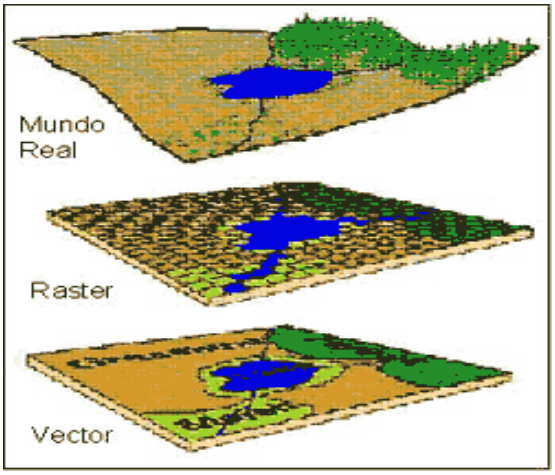


Figura 1. Modelos para el funcionamiento de un SIG

El modelo raster ha evolucionado para modelar características continuas. Una imagen raster comprende una colección de celdas (píxel) de una grilla como un mapa o una figura escaneada.

En el modelo vector, la información sobre puntos, líneas y polígonos se almacena como una colección de coordenadas (x, y). La ubicación de una característica puntual, puede describirse con un solo punto (x, y). Las características lineales, pueden almacenarse como un conjunto de puntos de coordenadas(x, y) y las poligonales, pueden almacenarse como un circuito cerrado de coordenadas. (Echeverría, X. V., 2007)

Procesos de un SIG

Los SIG básicamente realizan seis procesos o tareas:

- **Ingreso:** Antes que los datos geográficos puedan utilizarse en un SIG, deben ser convertidos a un formato digital adecuado.
- **Manipulación:** Es probable que los tipos de datos requeridos para un proyecto SIG necesitarán ser transformados o manipulados de alguna forma para hacerlos compatibles al sistema. Por ejemplo, si la información geográfica está disponible en diferentes escalas debe ser transformada a la misma escala previamente para que puedan superponerse e integrarse. La manipulación incluye cambios de proyección, agregación de datos y generalización (limpiar datos innecesarios).
- **Manejo/Administración:** Para proyectos menores de SIG, puede ser suficiente almacenar información geográfica como archivos. Sin embargo, cuando los volúmenes de datos son grandes y el número de usuarios de los datos crece cada día más, es mejor usar un sistema de manejo de bases de datos (SMBD)[1] para ayudar a almacenar, organizar y manejar datos.
- **Consulta:** Una vez que se tiene un SIG en funcionamiento, conteniendo información geográfica se puede comenzar a realizar consultas, donde se obtienen datos teniendo en cuenta determinados filtros o parámetros.
- **Análisis:** Los SIG funcionan realmente en su terreno cuando se utilizan para analizar datos geográficos. Los procesos de análisis geográficos utilizan propiedades geográficas de características para buscar patrones y tendencias, y para elaborar escenarios potenciales.
- **Visualización:** Para muchos tipos de operaciones geográficas, el resultado final se visualiza como un mapa o gráfico. Los mapas son muy eficientes para almacenar y comunicar información

[1]Un Sistema de Manejo de Bases de Datos (SMBD) es un software para manejar una base de datos, o sea, una colección integrada de datos.

geográfica. Mientras que los cartógrafos han creado mapas por milenios, los SIG proveen herramientas nuevas y emocionantes para extender el arte y la ciencia de la cartografía.(Echeverría, X. V., 2007)

Componentes de un SIG

Son cinco los elementos constitutivos de un sistema de estas características:

- **Hardware:** Los SIG corren en un amplio rango de tipos de computadores desde equipos centralizados hasta configuraciones individuales o de red, una organización requiere de hardware suficientemente específico para cumplir con las necesidades de aplicación.
- **Software:**

  Los componentes principales del software SIG son:

  - Sistema de manejo de base de datos.
  - Una interfaz gráfica de usuarios (IGU) para el fácil acceso a las herramientas.
  - Herramientas para captura y manejo de información geográfica.
  - Herramientas para soporte de consultas, análisis y visualización de datos geográficos.
- **Información:** El componente más importante para un SIG es la información. La recolección de los datos es un proceso largo que frecuentemente demora el desarrollo de productos que son de utilidad. Una información incorrecta o insuficiente introducida en el SIG produciría respuestas erróneas, por muy perfeccionada o adaptada al usuario que pueda ser la tecnología. Así, el condicionante principal a la hora de afrontar cualquier proyecto basado en SIG lo constituye la disponibilidad de datos geográficos del territorio a estudiar, convirtiéndose en los que consumen hoy día la mayor parte de las inversiones en términos económicos y de tiempo.
- **Personal:** Las tecnologías SIG son de valor limitado sin los especialistas en manejar el sistema y desarrollar planes de implementación del mismo. Sin el personal experto en su desarrollo, la información se desactualiza y se maneja erróneamente, el hardware y el software no se manipula en todo su potencial. Cuando se define un SIG se tiende a limitar a equipos y programas como el sistema completo, relegando tal vez el elemento más primordial: El talento humano que hace funcionar eficazmente todo el sistema.
- **Métodos:** Para que un SIG tenga una implementación exitosa debe basarse en un buen diseño y reglas de actividad definidas, que son los modelos y prácticas operativas exclusivas en cada organización.(Rodríguez, 2008)

Importancia de los SIG

Actualmente, debido a la disminución en el costo de los Sistemas Informáticos debido a su proliferación, están materializándose importantes beneficios económicos en las empresas y entidades que implementan la tecnología SIG. Entre estos beneficios se destacan:(SANABRIA., et al., 1998)

- Realizar un gran número de manipulaciones, sobresaliendo las superposiciones de mapas en corto tiempo, transformaciones de escala, la representación gráfica y la gestión de bases de datos, así como su administración y mantenimiento.
- Consultar rápidamente las bases de datos, tanto espacial como alfanumérica, almacenadas en el sistema, con información exacta, actualizada y centralizada.
- Realizar pruebas analíticas complejas rápidas y repetir modelos conceptuales en despliegue espacial, sin la necesidad de repetir actividades redundantes o tediosas.
- Comparar eficazmente los datos espaciales a través del tiempo (análisis temporal).

1.3 Estudio de soluciones existentes

Los Sistemas de Información Geográfica están cada vez más presentes en la sociedad. En el dominio de estos, también se ha generalizado el uso de tecnologías de software libre. En la actualidad, los SIG han logrado ampliar su utilidad práctica incluyendo usuarios con diferentes niveles de preparación debido al auge de las aplicaciones web, cada vez más enfocadas al usuario final y con interfaces gráficas más enriquecidas.

A continuación se realiza una descripción de algunos proyectos SIG de software libre. Se exponen parámetros como son: su licencia, los idiomas que manejan, los lenguajes de programación que permiten, su independencia con respecto a programas servidores de mapas, entre otros. El objetivo de este epígrafe es presentar las potencialidades de estas aplicaciones y sus principales deficiencias.

1.3.1 Aplicaciones SIG a nivel internacional

1.3.1.1 GRASS

GRASS (acrónimo inglés de *Geographic Resources Analysis Support System*) es una aplicación SIG bajo licencia GPL. Fue desarrollado por el Cuerpo de Ingenieros del Laboratorio de Investigación de Ingeniería de la Construcción del Ejército de los Estados Unidos (USA-CERL) como herramienta para la supervisión

y gestión medioambiental de los territorios bajo administración del Departamento de Defensa. Puede soportar información tanto raster como vectorial y posee herramientas de procesamiento digital de imágenes. Una de las más novedosas funcionalidades de este software es su motor de topologías para información vectorial en 2D y 3D, al igual que el análisis de redes vectoriales. GRASS está disponible principalmente para plataformas UNI (GNU/Linux).(U.S. Army Construction Engineering Research Laboratories, 2010)

1.3.1.2 MapGuide Open Source

Es una plataforma basada en web que permite a los usuarios desarrollar y desplegar aplicaciones web de mapas y servicios geoespaciales. Dispone de un visor interactivo que incluye soporte para la función de selección, inspección de la propiedad, consejo de mapas, entre otras. MapGuide incluye una base de datos XML para la gestión de contenidos, y soporta los formatos geoespaciales más populares de archivos, bases de datos y normas. Es licenciado bajo la LGPL[2].(MapGuide Open Source. OSGeo, 2007)

A continuación se detallan sus potencialidades:

Visualización Interactiva de Mapas, Calidad de Cartografía, Bases de Datos con alta capacidad de gestión, Acceso uniforme a los datos, Desarrollo flexible de aplicaciones, Extensiones APIs del lado del Servidor, Rápido, Escalable, Plataforma de Servidores Segura, Compatibilidad con múltiples plataformas, Estándares Abiertos Consorcio Geoespacial.(MapGuide Open Source. OSGeo, 2007)

1.3.1.3 QUANTUM GIS

Quantum GIS (o QGIS) es un Sistema de Información Geográfica de código libre para plataformas GNU/Linux, Unix, Mac OS y Microsoft Windows. El proyecto Quantum GIS nació oficialmente en mayo de 2002, cuando comenzó la codificación.

Era uno de los primeros ocho proyectos de la Fundación OSGeo y en 2008 oficialmente graduó de la fase de incubación. Permite manejar formatos raster y vectoriales, así como bases de datos. Está liberado bajo la GNU Public License. Está desarrollado en C++, usando la biblioteca Qt para su Interfaz gráfica de usuario. Dado esta característica, se deriva el nombre de Quantum GIS.

[2]LGPL es una licencia de software creada por la Free Software Foundation. Los contratos de licencia de la mayor parte del software están diseñados para jugar con su libertad de compartir y modificar dicho software.

Algunas de sus características son:

- Soporte para un gran número de tipos de archivos raster (GRASS GIS, GeoTIFF, TIFF, JPG)
- Superposición de datos vectoriales y de tramas en diferentes formatos y proyecciones, sin conversión a un formato interno o común.
- Soporte para la extensión espacial.
- Creación de mapas y exploración de forma interactiva de los datos espaciales con una interfaz gráfica de usuario amigable.
- Publicación de mapas en Internet con la exploración a la capacidad de archivo de asignaciones (requiere un servidor web con UMNMapServer instalado).
- Puede adaptarse a sus necesidades especiales a través de la arquitectura de pluggin extensible.(OSGeo, 2010)

1.3.2 Aplicaciones SIG a nivel nacional.

Cuba no ha quedado exento del desarrollo de los SIG, en 1987 surge en el Instituto de Geografía de la Academia de Ciencias de Cuba con el objetivo fundamental de actualizar el atlas nacional de Cuba. (Silva, 2009)

El Departamento de Computación y Matemática aplicada del Instituto cubano de Hidrografía (hoy GEOCUBA) desarrolla a partir de 1990 el producto TeleMap que en su versión actual constituye una herramienta muy poderosa para el diseño de un SIG y ha sido ampliamente generalizado en todo el país.

Otra rama que utiliza los SIG es la salud, donde especialistas han creado el Sistema de Información Geográfica para la Gestión de la Estadística de Salud de Cuba (SIGESAC). El mismo permite la representación cartográfica de las estadísticas de salud de Cuba de una forma muy sencilla y de fácil aprendizaje. Este permite cartografiar y realizar diferentes tipos de análisis con respecto a diversos aspectos de la salud como: morbilidad, mortalidad, población, recursos y servicios; el diseño de la base de datos, creada para este sistema, puede emplearse en otros sistemas de información geográfica, utilizados para desarrollar diversos análisis epidemiológicos como el Sistema de Información Geográfica de Epidemiología (SIGEPI).(Silva, 2009)

1.3.3 Aplicaciones SIG en la Universidad de las Ciencias Informáticas.

La Universidad de Ciencias Informáticas (UCI), las Fuerzas Armadas y los Especialistas de GEOCUBA desarrollaron la plataforma GENESIG la cual constituye una herramienta informática, que surge como necesidad de contar con un producto soberano que sirva como soporte al desarrollo de aplicaciones de Sistemas de Información Geográfica en entornos Web con tecnologías libres.

El objetivo fundamental de GENESIG es realizar la representación geoespacial de la información asociada a negocios específicos, además debe permitir realizar análisis sobre dicha información. Cuenta con la particularidad de poseer información que otros SIG existentes no brindan, por ejemplo le ofrece al usuario la posibilidad de configurar la representación del mapa en cuanto a estilos y simbologías mediante el módulo de catálogo.

Esta es una herramienta poderosa para ampliar las personalizaciones de sistemas de información geográfica bajo los estándares internacionales y un resultado que apoya al componente investigativo aportado para nuevas extensiones y colaboraciones de entidades de la rama de la Geomática[3] que utilizan los Sistemas de Información Geográfica para su funcionamiento y toma de decisiones.

1.3.3.1 SIGUCI

El SIGUCI es un producto resultante del primer proyecto de personalización sobre la Plataforma de Desarrollo de Sistemas de Información Geográfica GENESIG desarrollado por el proyecto de igual nombre del antiguo Polo Productivo Geoinformática de la antigua Facultad 9, actualmente Departamento de la Facultad 6, integrado además por especialistas del grupo empresarial GEOCUBA y el Centro UCID.(UCI, 2009)

El Sistema de Información Geográfica de la Universidad (SIG-UCI v.1.0 Beta), aplicación Web que georreferencia los objetivos socioeconómicos del centro, brinda servicios de localización de personas, instalaciones, lugares de interés, edificios de residencia, entre otros, así como soporte para la toma de decisiones en varios niveles. (Informáticas, 2010)

[3]Geomática es el término científico moderno que hace referencia a un conjunto de ciencias en las cuales se integran los medios para la captura, tratamiento, análisis, interpretación, difusión y almacenamiento de información geográfica. También llamada información espacial o geoespacial.

1.3.4 Conclusiones de las soluciones existentes

En la actualidad existe una infraestructura fuertemente respaldada para el desarrollo de aplicaciones SIG usando tecnologías libres. La Open Source Geospatial Foundation (OSGeo) es la principal organización a nivel mundial que se encarga de organizar y soportar los desarrollos de aplicaciones relacionadas con los sistemas de información geográfica del software libre. La Open Geospatial Consortium (OGC) es la encargada de dirigir a nivel mundial las disposiciones referentes a los SIG. Los principales estándares abiertos para los SIG son Web Feature Service (WFS) y Web Coverage Service (WCS), Web Map Service (WMS), Web Catalogue Service (CSW), Lenguaje de Marcado Geográfico (GML) y Keyhole Markup Language KML. Varias de las aplicaciones analizadas pudieran ser utilizadas o aplicadas en diferentes ramas en nuestro país. Según la opinión de los autores de este trabajo, los SIG más potentes desarrollados bajo el software libre son: MapGuide Open Source y QGIS. De los Sistemas de Información Geográfica de código abierto analizados, ninguno de ellos combina en su desarrollo GWT y OpenLayers, por lo que se avanza a desarrollar la investigación propuesta.

1.4 Conclusiones Parciales

Luego de haber abordado los conceptos asociados al problema y profundizado en el objeto de estudio se concluye que para representar y gestionar grandes volúmenes de datos sobre ciertos aspectos del mundo se necesita de un SIG. Se identifican los principales procesos que llevan a cabo estos sistemas, sus componentes, e importancia así como sus aplicaciones, lo cual ha posibilitado obtener una concepción general de los elementos esenciales para concebir la presente investigación. También se realiza un estudio del estado del arte de los sistemas nacionales e internacionales similares al que se propone en este trabajo, donde se concluye que no existe un SIG que integre las tecnologías GWT y OpenLayers. Estos elementos proporcionan la base de conocimientos necesaria para llevar a cabo la implementación del sistema.

CAPÍTULO 2: Tendencias y tecnologías a utilizar

Introducción

Los múltiples procesos que conforman el desarrollo de un buen software son en su generalidad procedimientos difíciles de tratar, por lo que se recomienda el empleo de una metodología especializada en estas actividades, la cual proporciona una guía completa de trabajo donde no solo se implementa un producto determinado sino que se puede observar la evolución que va teniendo desde sus inicios hasta que culmina el sistema.

Aunque el uso de metodologías facilita el trabajo de los desarrolladores, también ha de tenerse en cuenta qué herramientas y de qué forma se puede obtener un buen modelado del problema.

En el actual capítulo se aborda el estudio de diversas metodologías de desarrollo de software, donde se selecciona la más indicada para realizar la aplicación. De igual forma se puede obtener información en cuanto a Herramientas CASE, el lenguaje de programación a utilizar, IDEs de programación, servidor web, donde se elige de cada una de estas herramientas la más adecuada para un desarrollo eficiente del sistema. También se hace una caracterización de las dos tecnologías propuestas para el desarrollo del SIG.

2.1 Caracterización de las tecnologías GWT y OpenLayers

2.1.1 GWT

GWT es un framework de desarrollo en Java de código abierto. Es creado por Google bajo la Licencia Apache v2.0. Con GWT, se puede desarrollar y depurar aplicaciones AJAX utilizando el lenguaje de programación Java en cualquier entorno de desarrollo integrado. Cuando se ha acabado de implementar la aplicación escrita en Java, GWT compila y traduce dicho programa a JavaScript y HTML, el cual será compatible con cualquier navegador web. (Toolkit, 2010)

2.1.1.1 Principales características deGWT

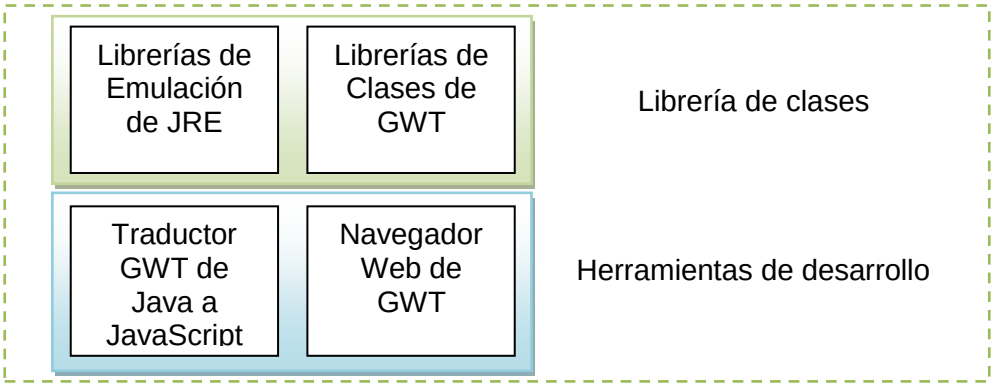
GWT posee una gran cantidad de características muy ventajosas para desarrolladores de aplicaciones web:

- Es compatible con cualquier navegador. Esto es muy importante ya que cada navegador necesita de un código distinto y esto hace que los desarrolladores pasen una gran parte del tiempo estudiando las peculiaridades de cada explorador. Con GWT, el compilador convierte el código escrito en Java al código JavaScript más adecuado para cada navegador.
- Es posible realizar el desarrollo en cualquier plataforma y también en cualquier entorno, según las preferencias del desarrollador. El desarrollo es más rápido como consecuencia del uso de IDEs y de herramientas propias de Java.
- Edita código Java y visualiza los cambios inmediatamente sin tener que volver a compilarlo. Durante el desarrollo de una aplicación, se puede ver inmediatamente los cambios que se hayan realizado en el código mediante el navegador en el modo alojado de GWT.
- El compilador GWT puede optimizar el código JavaScript generado ya que realiza un análisis completo de toda la base de código de GWT de manera que es capaz de sustituir código introducido de manera manual por uno que se carga y ejecuta de manera más rápida. También puede eliminar código que no se utiliza para que el código generado sea el mínimo posible.
- Se puede depurar el código Java en tiempo real. Mientras se está produciendo, el código corre sobre una Máquina Virtual de Java (JVM). Esto lleva consigo la posibilidad de depurar la aplicación con las herramientas incluidas en los entornos de desarrollo para estas funciones. Estarán disponibles funciones como la depuración paso a paso o con puntos de interrupción. De esta manera, los errores son detectados mientras se desarrolla la aplicación.
- Se pueden ver los cambios realizados durante el desarrollo de la aplicación en el navegador de modo alojado de GWT sin necesidad de tener que volver a compilar el código ni implementarlo en un servidor, lo cual aligera bastante el proceso de desarrollo al solo tener que hacer una actualización en el navegador para ver los resultados.
- Es de código abierto, así que todo el código de GWT está disponible bajo la licencia de Apache 2.0.
- Existe una gran cantidad de bibliotecas para GWT que aumentan su funcionalidad, desarrolladas por Google o terceros.
- Se puede mezclar código JavaScript con el código fuente de Java que se está desarrollando usando la Interfaz Nativa JavaScript o JSNI. Esto puede ser útil cuando las clases propias de GWT no satisfacen las necesidades a implementar para el desarrollo de la aplicación.

- Es posible reutilizar clases prediseñadas de componentes gráficos para implementar comportamientos que son iguales en distintos proyectos. Ejemplos de esto pueden ser arrastrar y soltar, un menú, entre otros.
- Soporte para las API[4] de Google**.**
- Reutilización del código a través del uso de patrones de diseño y la orientación a objetos.
- GWT permite la comunicación entre el navegador y el servidor web mediante RPC[5]de una manera muy sencilla.
- Permite integración con JUnit, con lo cual se puede probar y depurar las aplicaciones por unidad tanto en un depurador como en un navegador mientras se construye. También se puede comprobar el funcionamiento de las llamadas asíncronas a procedimientos remotos.
- Internacionalización fácil y rápida de sus aplicaciones y librerías. La internacionalización es el proceso de diseñar software de forma que, sin necesidad de cambios de ingeniería ni código, se puede adaptar a otros idiomas y regiones.

Arquitectura de GWT

GWT tiene cuatro componentes principales: un traductor Java aJavaScript, un navegador web "hosted", y dos librerías de clases:



[4]Una 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.
[5]El RPC (del inglés RemoteProcedureCall, Llamada a Procedimiento Remoto) es un protocolo que permite a un programa de ordenador ejecutar código en otra máquina remota sin tener que preocuparse por las comunicaciones entre ambos.

Librerías de Emulación JRE[6]: GWT contiene implementaciones de las librerías de clases más usadas en Java y las compila a JavaScript, incluyendo la mayoría de las clases del paquete java.lang y un subconjunto de clases del paquete java.util. El resto del estándar de librerías de Java no es soportado nativamente con GWT. Por ejemplo, las clases de los paquetes como java.io no se utilizan en aplicaciones web ya que estas acceden a recursos en la red y al sistema de archivos local.

Librería de clases de interfaz de usuario de GWT: Las librerías de clases de interfaz de usuario de GWT son un conjunto de interfaces y clases personalizadas que permiten crear componentes gráficos para el navegador, como botones, cajas de texto, imágenes, y texto. Este es el núcleo de las librerías de interfaz de usuario para crear aplicaciones GWT.

Traductor GWT Java a JavaScript: El traductor GWT de Java a JavaScript traduce del lenguaje de programación Java a JavaScript. Se utiliza cuando se necesita correr la aplicación en modo web. (Asociación Java Hispano, 2009)

Modos de ejecución

Las aplicaciones GWT pueden ser ejecutadas de los modos siguientes:

- **Modo hosted (hosted mode):** En modo hosted la aplicación se ejecuta como bytecodes de Java sobre una máquina virtual. Por lo general, se consume menos tiempo desarrollando en modo hosted, ya que se cuenta con todas las ventajas que proporciona Java para depurar usando un IDE como Eclipse.
- **Modo Web (Web mode):** En modo web la aplicación se ejecuta como HTML + JavaScript sobre un navegador, traducido desde el código fuente Java original con el compilador de GWT. Cuando la aplicación finaliza, lo único que se debe hacer es subirla a un servidor web, y los usuarios finales accederán a ella a través de un navegador en modo web. (Toolkit, 2009)

[6] Java Runtime Environment (JRE) es un conjunto de utilidades que permite la ejecución de programas Java.

2.1.2 OpenLayers

2.1.2.1 Principales características de OpenLayers

OpenLayers es una librería JavaScript que permite mostrar un mapa dinámico y marcadores cargados desde cualquier fuente de datos, utilizando una variante de la licencia BSD[7], destinada a mostrar mapas interactivos en los navegadores web. OpenLayers ofrece una API para acceder a múltiples tipos de fuentes de información cartográfica como Servicios de Mapas Web, Mapas comerciales, Servicios de Características Web, distintos formatos vectoriales, entre otros.(Sitio Oficial de OpenLayers, 2010)

OpenLayers se ha desarrollado para promover el uso de la información geográfica de todo tipo. Es totalmente gratuito, de código abierto JavaScript.

Entre las ventajas que posee se encuentran:

- No requiere instalación.
- Menor procesamiento en el servidor.
- Puede ampliar fácilmente el código para su aplicación en particular.
- Puede utilizar múltiples servidores de datos.

Un requisito que puede ser de desventaja en otro ambiente es que el desarrollador necesita de conocimientos de JavaScript, CSS y HTML. Para contrarrestar esto el uso de GWT abstrae al desarrollador de dichos lenguajes encapsulándolos mediante el plugin GWT-OpenLayers en el lenguaje de desarrollo Java creando así una homogeneidad en el proceso de desarrollo.

2.1.2.2 Funciones de OpenLayers

Cambiador de Capas: Es el control para la gestión de la visibilidad de capas.

Barra de zum y navegación: Crea una barra de zum y un panel de navegación que contienen los botones de *ZoomIn* y *ZoomOut* a los extremos. Hay tres maneras para utilizar esta barra:

[7] La licencia BSD es la licencia original de una distribución de Software: *Berkeley Software Distribution*, que acabó convirtiéndose en un derivativo de UNIX realizado por la conocida Universidad de California, Berkeley. Permite el uso del código fuente en software no libre.

- Clic sobre los botones de + ó - puestos en los extremos de la barra. El resultado será el aumento o disminución de un nivel de zum.
- Clic en cualquier punto de la barra, el desplazador (*slider*) se mueve al nivel de zum más cercano. El mapa se actualiza al nivel de zum correspondiente.
- Desplazamiento del cursor a un nivel de zum concreto cuando se deja el botón del ratón, el desplazador se posiciona en el nivel de zum más cercano y el mapa se actualiza al nivel de zum correspondiente.

Mapa de navegación: Crea un pequeño mapa de navegación. Este mapa enseña su posición principal y ofrece una herramienta más de navegación. Normalmente se posiciona en el ángulo inferior derecho y puede ser reducido con un clic en el botón que tiene en su borde. El rectángulo dibujado en este mapa puede ser movido para cambiar la posición del mapa principal.

Enlace: Ofrece un enlace al mapa. Esto permite poder guardar en un simple enlace el estado de la navegación del mapa. Así, se puede guardar como favorito en el navegador o enviarlo por correo.

Barra de herramienta del mouse: Es una simple barra de herramientas que permite escoger cómo utilizar el ratón entre navegación y zum con ventana.

Línea a escala: Enseña una escala gráfica sobre el mapa.

Posición del mouse: Enseña las coordenadas actuales del cursor sobre el mapa.

2.2 Metodologías de Desarrollo

“Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, evolutivo, incremental, entre otros). Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto y guías para uso de herramientas de apoyo. Habitualmente se utiliza el término “método” para referirse a técnicas, notaciones y guías asociadas, que son aplicables a una (o algunas) actividades del proceso de desarrollo, por ejemplo, suele hablarse de métodos de análisis y/o diseño”. (Hill, 1997)

Según su filosofía de desarrollo se clasifican en pesadas o tradicionales y ligeras o ágiles.

2.2.1 Metodologías Pesadas de Desarrollo

Las metodologías tradicionales o pesadas del desarrollo del software se guían expresamente a través de un plan durante el proceso de desarrollo, en el cual se generan gran cantidad de artefactos y roles mediante un modelado y la documentación detallada, apoyándose en poderosas herramientas de trabajo y notaciones definidas en la arquitectura del software. (SÁNCHEZ, 2004)

2.2.1.1 Rational Unified Process (RUP)

Es un proceso de ingeniería de software planteado por Kruchten (1996), cuyo objetivo es producir software de alta calidad, es decir, que cumpla con los requerimientos de los usuarios dentro de una planificación y presupuesto establecido. Cubre el ciclo de vida y desarrollo de software.

Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Posee varios beneficios, entre ellos permite desarrollar aplicaciones sacando el máximo provecho de las nuevas tecnologías, mejorando la calidad, el rendimiento, la reutilización, la seguridad y el mantenimiento del software mediante una gestión sistemática de los riesgos; permite la producción de software que cumpla con las necesidades de los usuarios, a través de la especificación de los requisitos, con una agenda y costo predecible; enriquece la productividad en equipo y proporciona prácticas óptimas de software a todos sus miembros; permite llevar a cabo el proceso de desarrollo práctico, brindando amplias guías, plantillas y ejemplos para todas las actividades críticas; unifica todo el equipo de desarrollo de software y mejora la comunicación al brindar a cada miembro del mismo una base de conocimientos, un lenguaje de modelado y un punto de vista de cómo desarrollar software; optimiza la productividad de cada miembro del equipo al poner al alcance la experiencia derivada de miles de proyectos y muchos líderes de la industria. Deja plasmado una larga documentación donde se recoge todo el proceso evolutivo del Software a través de sus fases y flujos de trabajo.

4.1.1.2 Microsoft Solution Framework (MSF)

Esta es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en

los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas.(Gattaca, 2009)

MSF tiene las siguientes características: adaptable, pues es usado en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar. Es escalable, puede organizar equipos tan pequeños entre tres o cuatro personas, así como también, proyectos que requieren cincuenta personas o más. Es flexible, es utilizada en el ambiente de desarrollo de cualquier cliente. Presenta tecnología agnóstica, porque puede ser usada para desarrollar soluciones basadas sobre cualquier tecnología. Además se basa en el control de entregables por parte de los desarrolladores al proyecto.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto: Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el Modelo de Aplicación.

2.2.2 Metodologías Ágiles de Desarrollo

Las metodologías ágiles se enfocan generalmente en intentar evadir las guías de las metodologías tradicionales. Centran sus procesos de desarrollo a los individuos que interactúan con el proyecto así como con el equipo de trabajo, obteniendo buenos resultados, es decir, se desarrolla un buen software en cuanto a su funcionamiento pero no genera una documentación consistente. (LETELIER, P., 2003)

2.2.2.1 Extreme Programming (XP)

Está centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP está guiada por una rápida programación y se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, en la reutilización de código, en la realización de pruebas a los principales procesos con el objetivo de tratar de obtener los posibles errores futuros. (Beck, y otros, 1999)

2.2.2.2 SCRUM

Metodología desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle, es un término utilizado en el juego de rugby que llevado a la ingeniería significa que el equipo se agrupará para llegar con el apoyo de

cada uno de los miembros del proyecto a obtener un producto con calidad que es el objetivo fundamental. Define un proceso empírico, iterativo e incremental de desarrollo, está especialmente indicada para proyectos con un rápido cambio de requisitos. El principio básico es que es muy difícil contar desde el principio con un catálogo completo de funcionalidades, ya que los requisitos van surgiendo conforme el propietario del producto y los usuarios del mismo van haciendo sucesivas aportaciones. El desarrollo de software se realiza mediante iteraciones, denominadas sprints. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. Otra característica importante son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (Calderón, 2007)

2.2.3 RUP, metodología de desarrollo para el SIG

Teniendo en cuenta lo descrito anteriormente es importante destacar que MSF a pesar de estar dedicada principalmente a proyectos de varias dimensiones, se basa en el control de entregables por parte de los desarrolladoresen vez de estar orientada a procesar una documentación, además se basa en procesos yno en casos de uso,por lo que no es factible que se utilicepara este sistema. Se descarta el uso de la metodología XP puesto que con ella es necesario tener un intercambio constante entre los usuarios y los desarrolladores, por lo que el sistema no se puede desarrollar utilizando dicha metodología porque no se cuenta con un cliente específico. Al igual que SCRUM evita la generación documental, ambas se basan más en el desarrollo que en la organización y deben unirse a otras metodologías para lograr sus objetivos, por lo que SCRUM también se descarta. Por la importancia que tiene para esta investigación la elaboración de una eficiente documentación se selecciona RUP, puesto que es una metodología orientada al documento, cubre todo el ciclo de vida y desarrollo de software, además como es caracterizado por ser iterativo e incremental, permite el refinamiento constante del sistema y la adición de nuevas funcionalidades a cada una de las iteraciones, y como es dirigido por casos de usos y centrado en la arquitectura asegura que el resultado final del producto cumpla con los requisitos planteados por el cliente o usuario. Este trabajo tiene que quedar con total documentación para cuando un cliente se interese por el mismo entienda todo el proceso de desarrollo.

2.3 Lenguaje Unificado de Modelado (UML)

RUP utiliza como lenguaje para modelado a UML, este es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema. Está compuesto por diversos elementos gráficos que se combinan para conformar diagramas, no es una guía para realizar el análisis y diseño orientado

aobjetos, es decir, no es un proceso. UML es un lenguaje que permite la modelación de sistemas con tecnología orientada a objetos. (Universidad de las Ciencias Informáticas, 2011)
Este lenguaje es fácil de aprender y usar, conforma la colección de las mejores técnicas de ingeniería que han probado ser un éxito en el modelado de sistemas grandes y complejos. (Jiménez Lezama, 2005)

Las ventajas que posibilita la utilización de UML son diversas, por ejemplo el uso de los lenguajes visuales facilita la asimilación y entendimiento del contenido, se minimiza el tiempo invertido en el desarrollo del sistema, la documentación del proyecto se confecciona de una forma ordenada y guiada por casos de uso.

2.4 Herramientas CASE

Para superar las dificultades existentes del uso de las diversas tecnologías, la industria de computadoras ha desarrollado un soporte automatizado para el desarrollo y mantenimiento de software. Este es llamado Computer Aided Software Engineering (CASE), lo que se define como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un Software. (Scribd, 2010)

Son aplicaciones informáticas que están destinadas a aumentar la productividad en el desarrollo de software y reducir los costos en cuanto a tiempo y dinero.

2.4.1 Rational Rose

Rational Rose es una herramienta de diseño orientada a objetos, que da soporte al modelado visual, es decir, que permite representar gráficamente el sistema, permitiendo hacer énfasis en los detalles más importantes, centrándose en los casos de uso y enfocándose hacia un software de mayor calidad, empleando un lenguaje estándar común que facilita la comunicación. Está bajo licencia privativa. (Murillo Alfaro, 1999)

Esta herramienta proporciona mecanismos para realizar la Ingeniería Inversa, cubre todo el ciclo de vida de un proyecto: concepción y familiarización del modelo, construcción de los componentes, transición a los usuarios y certificación de las distintas fases, brinda la posibilidad de que varias personas trabajen a la vez, permitiendo que cada desarrollador opere en un espacio de trabajo privado.

2.4.2 Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (Visual Paradigm International, 2009)

Dentro de sus principales características se encuentran: posibilita el uso de varios idiomas, soporta aplicaciones Web, es un producto de calidad, es fácil de instalar, actualizar y tener gran compatibilidad.

Es una herramienta profesional multiplataforma, que proporciona abundantes tutoriales de UML, así como demostraciones interactivas y proyectos de este lenguaje de modelado. Posee como peculiaridad sobre el resto de las demás herramientas que cuenta con una potente funcionalidad para la creación de interfaces de usuarios de las aplicaciones. (Visual Paradigm International, 2010)

2.4.3 Visual Paradigm, herramienta CASE para el SIG

Ambas herramientas presentan grandes potencialidades para el modelado de la aplicación pero se descarta Rational Rose al ser privativa, se selecciona Visual Paradigm puesto que es una herramienta multiplataforma, que permite dibujar todos los tipos de diagramas de clases, presenta amplia facilidad de uso, abundantes tutoriales de UML y una potente funcionalidad para la creación de interfaces.

2.5 Lenguaje de Programación

2.5.1 Java

Teniendo en cuenta las tecnologías previas a utilizar para el desarrollo de este Sistema de Información Geográfica se selecciona el lenguaje Java por ser el que utiliza GWT, el mismo es soportado por IDEs de programación libres. Java es un lenguaje de programación con el que se puede realizar cualquier tipo de programa. En la actualidad es un lenguaje muy extendido y cada vez cobra más importancia tanto en el ámbito de Internet como en la informática en general. Está desarrollado por la compañía SUM Microsystems con gran dedicación y siempre enfocado a cubrir las necesidades tecnológicas más

punteras. Algunas de las principales características de Java son: es multiplataforma, es enfocado hacia la Programación Orientada a Objetos y presenta un alto nivel de seguridad en su código.

Java se ha convertido en un recurso inestimable ya que permite a los desarrolladores:

- Desarrollar software en una plataforma y ejecutarlo en prácticamente cualquier otra plataforma.
- Crear programas para que funcionen en un navegador web y en servicios web.
- Desarrollar aplicaciones para servidores como foros en línea, tiendas, encuestas, procesamiento de formularios HTML, entre otros.
- Combinar aplicaciones o servicios que usan el lenguaje Java para crear servicios o aplicaciones totalmente personalizados.
- Desarrollar potentes y eficientes aplicaciones para teléfonos móviles, procesadores remotos, productos de consumo de bajo coste y prácticamente cualquier tipo de dispositivo digital. (Lenguajes de Programación, 2010)

2.6 Entorno Integrado de Desarrollo (IDE)

2.6.1 Eclipse

Es una plataforma universal para integrar herramientas de desarrollo, con una arquitectura abierta y basada en pluggins. Eclipse da soporte a todo tipo de proyectos que abarcan todo el ciclo de vida del desarrollo de aplicaciones, incluyendo soporte para modelado. El entorno integrado de desarrollo (IDE) de Eclipse emplea módulos para proporcionar toda su funcionalidad al frente de la plataforma del cliente, a diferencia de otros entornos monolíticos donde las funcionalidades están todas incluidas, las necesite el usuario o no. Está escrito en Java. (Sitio Oficial de Eclipse, 2003)

Presenta características principales como:

- Editor visual con resaltado de sintaxis.
- Compilación incremental de código.
- Depura código que resida en una máquina remota. (SWT-JFace-Eclipse, 2010)

2.6.2 Zend Studio for Eclipse

Zend Studio o Zend Development Environment es un completo entorno de desarrollo integrado para el lenguaje de programación PHP. Está escrito en Java, y está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux.

No requiere la instalación previa de PHP ni del entorno de ejecución de Java, tiene phpDoc y Cliente FTP integrado, plegado de código (comentarios, bloques de phpDoc, cuerpo de funciones y métodos e implementación de clases), soporte para PHP 4, PHP 5, y para control de versiones usando CVS o Subversion si el desarrollador lo elige. Detecta errores de sintaxis en tiempo real y tiene una Instalación de barras de herramientas para Internet Explorer y Mozilla Firefox. (Desarrollo Web, 2005)

2.6.3 Eclipse, IDE de programación para el SIG

Se escoge como IDE a Eclipse, puesto que da soporte a todo tipo de proyectos que abarcan todo el ciclo de vida del desarrollo de aplicaciones, incluyendo soporte paramodelado. Tiene como característica fundamental para este SIG que está escrito en Java y que además depura código que resida en una máquina remota, característica imprescindible de la tecnología GWT. Se descarta Zend Studio for Eclipse porque es un completo entorno de desarrollo integrado para el lenguaje PHP, el cual no es utilizado en esta aplicación.

2.7 Servidor Web

2.7.1 Servidor Apache HTTP 2.0

El proyecto Apache HTTP Server es un esfuerzo por desarrollar y mantener un servidor HTTP de código abierto para los sistemas operativos modernos, incluyendo UNIX y Windows NT. El objetivo del mismo es proveer un seguro, eficiente y extensible servidor que proporcione servicios HTTP en acuerdo con los estándares actuales de HTTP. (Sitio Oficial de Apache, 2008)

Esta nueva versión de Apache tiene las siguientes características:

Hilos en UNIX: En los sistemas UNIX con soporte a hilos POSIX, Apache puede correr ahora en un modo multiproceso híbrido de forma multihilo, lo que hace que mejore la estabilidad.
Soporte Multiprotocolo: Apache tiene ahora una estructura en aras de soportar servicios por múltiples protocolos.

Mejor soporte en plataformas no UNIX: Apache 2.0 es más rápido y más estable en plataformas no UNIX como BeOS, OS/2 y Windows. Con la introducción de los módulos de multi-procesamiento (MPMs) específicos para cada plataforma y el Apache Portable Runtime (APR).

Soporte IPv6: En los sistemas que soportan IPv6 Apache ahora también tiene este soporte, sus sockets de escucha ya lo soportan por defecto.

Filtrado: Los módulos de Apache pueden ahora ser escritos como filtros que pueden actuar en el flujo de contenido desde el servidor o hacia él.

Configuración simplificada: Muchas de las directivas que pueden ser confusas han sido simplificadas.

Soporte nativo a Windows NT Unicode: Apache 2.0 en Windows NT ahora utiliza utf-8 para la codificación de los nombres de ficheros.

Actualizada la biblioteca de expresiones regulares: Apache 2.0 incluye a Perl Compatible Regular Expression Library (PCRE), para la evaluación de todas las expresiones regulares se utiliza la más poderosa sintaxis de Perl 5.

2.8 Conclusiones Parciales

Después de un análisis de varias metodologías se selecciona RUP para guiar el proceso de desarrollo del sistema, esta metodología deja bien documentado la vida del SIG. También se hace una caracterización de las tecnologías GWT y OpenLayers, donde se recopila toda la información necesaria para la integración de las mismas en la construcción de la aplicación. El lenguaje de programación que se utilizará es Java, pues es el que utiliza GWT. Para seleccionar la herramienta CASE a utilizar se tuvo en cuenta que fuera multiplataforma, por lo que se escogió a Visual Paradigm. Además es escogido el IDE de desarrollo Eclipse, puesto que da soporte a todo tipo de proyectos y además está escrito en Java, algo que es muy importante para la realización de este SIG, puesto que es el lenguaje en el que se programa utilizando GWT.

CAPÍTULO 3: Presentación de la Solución Propuesta

Introducción

Luego de haber analizado las herramientas y tecnologías a utilizar para el desarrollo de la solución, en este capítulo se presenta el análisis y la modelación realizada para implementar un SIG integrando las tecnologías GWT y OpenLayers. En el presente capítulo se abordan todos los aspectos relacionados con el dominio de la solución. Además, se exponen los requisitos funcionales y no funcionales, los cuales son la base para el diseño y la implementación. Se identifican actores del sistema así como sus relaciones con los casos de uso (CU), expresados en el modelo de casos de uso del sistema. También se describen los casos de uso identificados a partir de sus requisitos funcionales asociados.

3.1 Modelo de Dominio

Un Modelo de Dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan los eventos que suceden en el entorno que trabaja el sistema. (Jacobson, y otros, 2004)

Es necesario aclarar, para mayor comprensión, por qué modelar un dominio y no un negocio. En la presente investigación los procesos involucrados no representan procesos reales, dado que el resultado final debe ser un producto genérico que sirva como base para el desarrollo de otros SIG solicitados por clientes interesados en aplicaciones de este tipo que integren las tecnologías GWT y OpenLayers. Los casos de uso modelados son genéricos dentro de la estructura conceptual de un sistema de información geográfica, constituyendo una generalización de las acciones fundamentales, obtenidas como resultado del estudio de los SIG. La solución propuesta no tiene un cliente, por lo que un negocio a modelar es inexistente. Por estas razones, se decidió realizar un modelo conceptual de dominio para proporcionar una mayor comprensión del área de acción de la solución propuesta, el cual tiene el objetivo ayudar a comprender los conceptos que utilizan y con los que deberá trabajar la aplicación, así como la relación entre ellos.

3.1.1 Modelo conceptual del dominio.

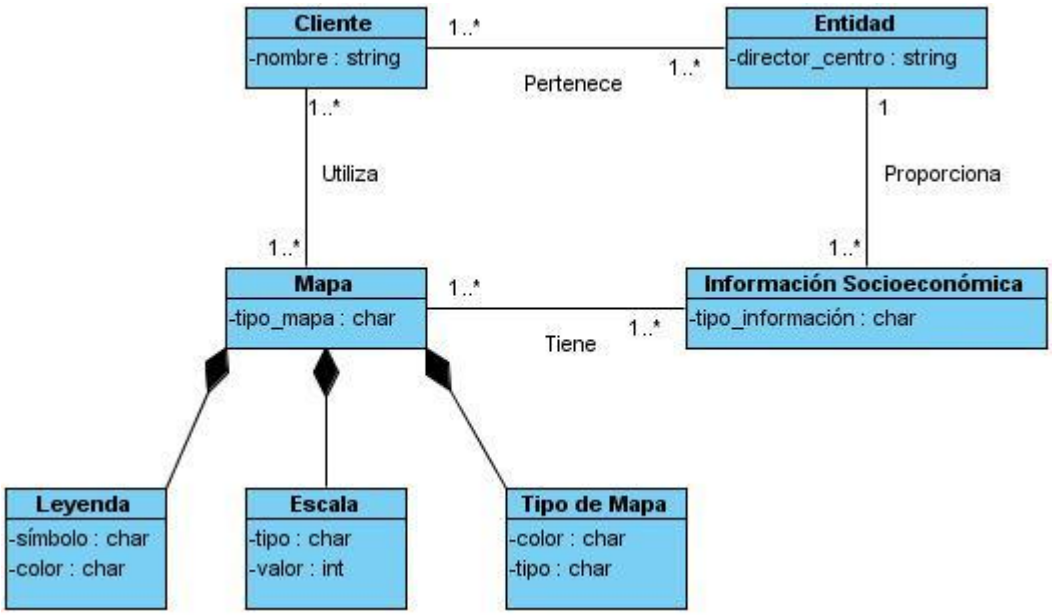


Figura 2. Modelo de Dominio

3.1.2 Descripción de las clases del dominio

Cliente

Es la persona que trabaja en una empresa, organismo u organización que necesite utilizar o consultar algún tipo de información incluida en un mapa.

Entidad

Es la 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. El mapa tiene propiedades métricas, esto 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. (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.

3.1.3 Breve descripción del diagrama

Los mapas están compuestos por varias escalas representativas, por leyendas que permiten un mejor entendimiento de los mismos, así como por la tipografía, que está referida a la variedad de mapas existentes de acuerdo con su especificación, que pueden ser de división Político-Administrativa, Vegetación, Yacimientos geológicos, entre otros. También estos mapas tienen toda la información socioeconómica, tratada en los distintos tipos de datos referentes a todos los sectores de la economía y la sociedad. Los mapas son utilizados por los diferentes clientes pertenecientes a una entidad específica, la misma proporciona toda la información socioeconómica que se le agregará posteriormente a los mapas.

3.2 Requisitos

Una etapa inicial y muy importante dentro del proceso de la Ingeniería de Software basada en RUP son los Requisitos, que es donde se lleva a cabo el proceso de descubrir, analizar, escribir y verificar los servicios y restricciones del sistema de software que se desea producir. Este proceso se realiza mediante la obtención, el análisis, la especificación, la validación y la administración de los requisitos del software. Los requisitos de software son las necesidades de los clientes, los servicios que los usuarios desean que proporcione el sistema de desarrollo y las restricciones con las que debe cumplir.

Al lograr la identificación de los requisitos iniciales de la aplicación y realizar el análisis de los mismos se limaron las ambigüedades y se determinaron los requisitos funcionales y no funcionales presentados a continuación:

3.2.1 Requisitos Funcionales

Son los que presentan las funcionalidades que el sistema deberá ser capaz de realizar, incluyendo las acciones que podrán ser ejecutadas por el usuario, las acciones ocultas que debe realizar el sistema, y las condiciones extremas a determinar por el mismo. (Martínez, y otros, 2008)

RF1 Mover Mapa

Con este requerimiento se desea que el usuario pueda mover el mapa hacia todas las latitudes.

RF2 Realizar Zum

Con este requerimiento se quiere que el usuario pueda ampliar o disminuir el área del mapa a partir de la ejecución de las diferentes variantes de zum.

RF 2.1 Zum +: Este requisito tiene como función acercar la imagen del mapa a la vista del usuario.

RF 2.2 Zum - : Este requisito tiene la función alejar la imagen del mapa de la vista del usuario.

RF3 Recentrar Mapa

Con este requerimiento se quiere que el usuario sitúe el mapa en su centro, haciéndolo coincidir con el centro de visualización de la aplicación.

RF4 Medir Distancia

Con este requerimiento se quiere que el usuario pueda realizar trazos entre puntos en el mapa y visualice la distancia calculada hasta el momento y luego de culminar la medición visualice la distancia total calculada.

RF5 Calcular Área de una región

Con este requerimiento se quiere que el usuario pueda ir construyendo un polígono en el mapa para poder visualizar el cálculo del área del mismo, tanto la parcial mientras va trazando el polígono y la total cuando culmina.

RF6 Añadir marca

Con este requerimiento se quiere que el usuario pueda incrustar una marca en el mapa en determinado punto señalado por el mismo.

3.2.2 Requisitos No Funcionales

Los requisitos no funcionales son los que especifican propiedades o cualidades que el producto debe tener, como restricciones del entorno o de la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad, fiabilidad, entre otras. Estas propiedades se ven como las características que hacen al producto agradable, usable, rápido o confiable.(Jacobson, y otros, 2004)

A continuación se definen los siguientes:

RNF 1 Usabilidad

El sistema podrá ser usado por personas con conocimientos básicos en el manejo de ordenadores.

RNF 2 Eficiencia

La velocidad de procesamiento de la información, la actualización y la recuperación dependerán de la cantidad de información que tenga que procesar la aplicación, no sobrepasando los tres o cuatro segundos.

RNF 3 Restricciones de diseño

El producto de software debe estar soportado sobre una arquitectura cliente-servidor.

Se deben emplear los estándares establecidos por las tecnologías GWT y OpenLayers.

RNF 4 Interfaces de usuario

El sistema debe tener una apariencia profesional y un diseño gráfico sencillo y amigable.

RNF 5 Requerimientos del Software

La construcción de la aplicación funcionará bajo los conceptos de arquitectura cliente/servidor. Por tanto, el servidor del usuario final debe tener como requerimientos mínimos de software:

Para las PCs clientes:

- El sistema es transparente al Navegador web que utilice el cliente.
- Puede tener instalado cualquier sistema operativo.
- El navegador Web tiene que tener soporte JavaScript.

Para los Servidores:

- Sistemas operativos GNU/Linux, Windows XP o superior.
- Servidor Web Apache HTTP v2.0 o superior.
- OpenLayers v2.10.
- GWT v1.3.

RNF 6 Requerimientos del Hardware

Para las PCs clientes:

- Conexión 50 MB.
- 256 MB de memoria RAM.
- Procesador 512 MHz.

Para los servidores:

- Conexión 1GB.
- Procesador 3 GHz.

- RAM de 2 GB.
- Espacio en disco disponible de 1 GB.

RNF 7 Rendimiento

El tiempo de respuesta está dado por la cantidad de información a procesar, no deberá sobrepasarse de tres segundos.

RNF 8 Portabilidad

El sistema será multiplataforma y su puesta en marcha debe ser compatible con cualquier sistema operativo.

3.3 Descripción del Sistema

Luego de haber realizado la descripción de los requisitos funcionales, los cuales constituyen las funcionalidades esenciales del sistema, se le da paso a la siguiente etapa de conceptualización de un software, que es la descripción del Modelo de Casos de Uso del Sistema (MCUS). El mismo está formado por actores del sistema, casos de uso del sistema, y sus relaciones.

Los requisitos funcionales se agrupan según sus características, utilidad y usabilidad conformando así los casos de uso, los cuales no son más que fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores. Estos casos de uso describen cómo se comporta y funciona un sistema específico, mediante un documento que tiene redactado de manera secuencial las acciones relevantes que realiza un actor en interacción con el sistema hasta completar un proceso, sin darle importancia a los detalles de la implementación. (Jacobson, 2000)

Los actores se definen como los roles que puede tener un usuario, pueden ser humanos, otros sistemas, máquinas, hardware, entre otros, que interactúan con un sistema para de esta forma intercambiar datos, aunque en algunos casos pueden constituir un recipiente pasivo de información. Un actor no es parte del sistema en desarrollo, es un agente externo que interactúa con el mismo en pos de obtener un resultado esperado. (Jacobson, 2000)

El sistema cuenta con el actor que a continuación se presenta:

Actor	Descripción
Usuario	Es la persona que interactúa con el sistema y utiliza las funcionalidades que brinda el mismo.

Figura 3. Actor del sistema

3.3.1 Modelo de Casos de Uso del Sistema

A partir de la identificación de los actores que interactúan con la aplicación, así como la recopilación del conjunto de funcionalidades escritas en forma de requerimientos que a su vez han sido agrupados en casos de uso según sus peculiaridades, se conforma el Modelo de casos de uso del sistema, el mismo es la entrada fundamental para el diseño, implementación y pruebas. El modelo que fundamenta el desarrollo del sistema propuesto es el que se presenta a continuación:

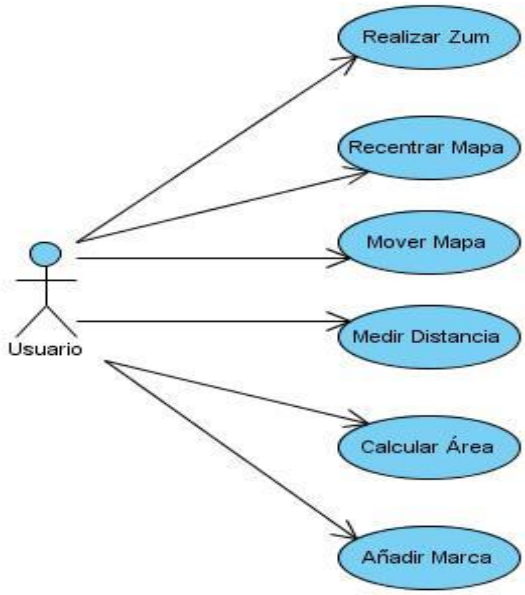


Figura 4. Modelo de Caso de Uso del Sistema

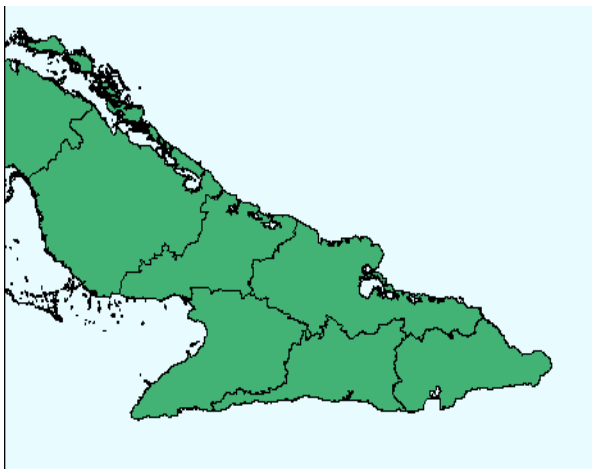
3.3.2 Descripción textual de los Casos de Uso del Sistema

Cada uno de los casos de uso identificados presenta características particulares que para su mayor entendimiento se hace necesario describirlas textualmente. Además, para lograr un mejor rendimiento del tiempo en la construcción del software es recomendable identificar los casos de uso arquitectónicamente significativos, que en la descripción textual se diferencian por la prioridad de crítico. A continuación se muestran las descripciones correspondientes a los casos de uso arquitectónicamente significativos, así como la correspondiente a los otros casos de uso.

3.3.2.1 CU Mover Mapa

Caso de Uso:	Mover Mapa
Actores:	Usuario
Propósito	Este caso de uso se lleva a cabo con el objetivo de desplazar el mapa al norte, sur, este y oeste.
Resumen:	El caso de uso se inicia cuando el actor desea mover el mapa, y termina cuando el sistema muestra el mapa hacia la dirección que seleccionó el usuario.
Precondiciones:	
Referencias	RF 1
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario da click en uno de los botones: norte, sur, este, oeste.	2. El sistema mueve el mapa hacia la dirección indicada por el actor.
	3. El sistema procesa la información y

	devuelve el área de representación corregida en dependencia de la dirección seleccionada.
Prototipo de Interfaz	
 	

3.3.2.2 CU Realizar Zum

Caso de Uso:	Realizar Zum
Actores:	Usuario
Propósito	Este caso de uso se lleva a cabo con el objetivo de incrementar y disminuir el área de visualización del mapa.
Resumen:	El caso de uso se inicia cuando el usuario desea acercar o alejar el mapa, culmina cuando el sistema muestra el área del mapa aumentada o disminuida en la pantalla.
Precondiciones:	
Referencias	RF 2
Prioridad	Crítico
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona una de las opciones de	2. El sistema realiza la operación según la

visualización del mapa: “Acercar” “Alejar”	opción seleccionada por el actor. Si seleccionó “Acercar”, ver sección “Zum +”. Si seleccionó “Alejar”, ver sección “Zum -”.
	3. El sistema procesa la información y actualiza el área que se representa del mapa.
Sección “Zum +”	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción “Acercar” de la barra de herramientas.	2. El sistema realiza la operación de zum+ seleccionada por el actor, donde acerca la imagen del mapa a la vista del usuario. El sistema disminuye la escala.
	3. El sistema realiza lo que se describe en el paso 3 del flujo normal de eventos.
Prototipo de Interfaz	

Sección “Zum -”	
Acción del Actor	**Respuesta del Sistema**
1. El actor selecciona la opción “Alejar” de la barra de herramientas.	2. El sistema realiza la operación de zum-seleccionada por el actor, donde aleja la imagen del mapa de la vista del usuario. El sistema aumenta la escala.
	3. El sistema realiza lo que se describe en el paso 3 del flujo normal de eventos.
Prototipo de Interfaz	

3.3.2.3 CU Recentrar Mapa

Caso de Uso:	Recentrar Mapa
Actores:	Usuario
Propósito	Este caso de uso se lleva a cabo con el objetivo de hacer corresponder el centro de visualización con el centro del mapa su centro.
Resumen:	El caso de uso se inicia cuando el actor desea poner el mapa en su centro y termina cuando el sistema hace coincidir el centro del mapa con el centro de visualización de la aplicación.
Precondiciones:	
Referencias	RF 3

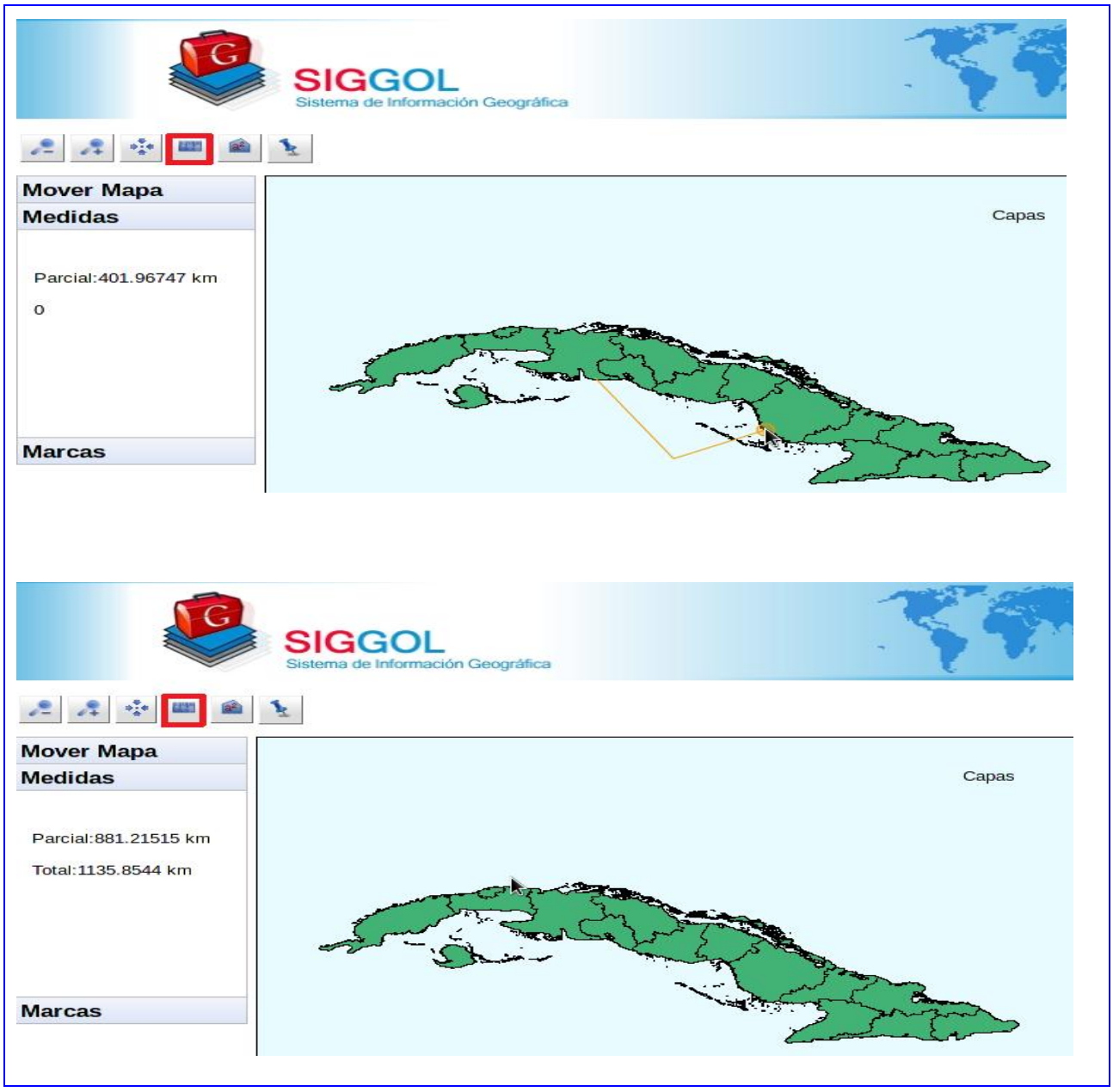
Prioridad	Secundario
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción “Recentrar Mapa”.	**2.** El sistema muestra el mapa ubicado en su centro.
Prototipo de Interfaz	

3.3.2.4 Medir Distancia

Caso de Uso:	Medir Distancia
Actores:	Usuario

Propósito	Este caso de uso se lleva a cabo con el objetivo de medir la distancia entre dos o más puntos en el mapa.
Resumen:	El caso de uso se inicia cuando el actor desea medir la distancia existente entre dos o más puntos marcados en el mapa, y termina cuando el sistema muestra los valores correspondientes.
Precondiciones:	
Referencias	RF 4
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona en la barra de herramientas la opción “Medir Distancia”.	**2.** El sistema permite al usuario comenzar a medir la distancia entre los puntos.
3. El usuario marca una línea de punto a punto.	**4.** El sistema va mostrando los puntos y la línea que se forma entre ellos, mostrando la distancia parcial medida hasta el momento.
5. El usuario da doble click para terminar la edición de la distancia a medir.	**6.** El sistema muestra la distancia total alcanzada en la medición.
Prototipo de Interfaz	

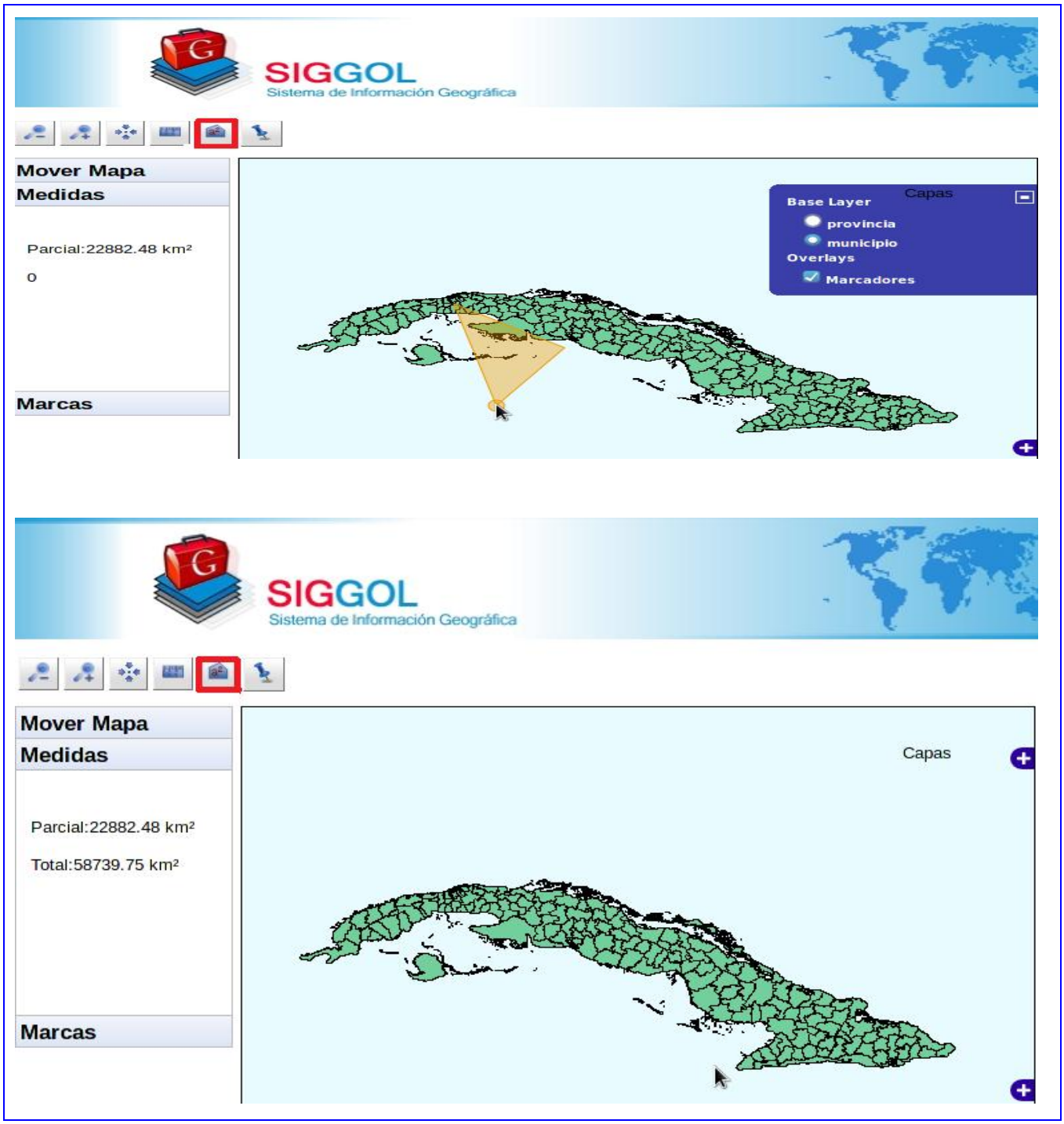


3.3.2.5 Calcular Área

Caso de Uso:	Calcular Área.
Actores:	Usuario

Propósito	Este caso de uso se lleva a cabo con el objetivo de formar una región mediante un polígono para poder visualizar el cálculo del área de la misma.
Resumen:	El caso de uso se inicia cuando el usuario desea obtener el área de una región trazada en el mapa, y termina cuando el sistema muestra los valores correspondientes.
Precondiciones:	
Referencias	RF 5
Prioridad	Crítico

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción “Calcular Área”.	**2.** El sistema permite comenzar a editar la región en forma de polígono para calcular el área.
3. El usuario delimita el polígono en el mapa al que quiere calcularle el área.	**4.** El sistema muestra el polígono al que se la calculará el área, mostrando la distancia parcial medida hasta el momento.
5. El usuario da doble click para terminar la edición del polígono al que se le está calculando el área.	**6.** El sistema muestra el área total calculada en el polígono delimitado.
Prototipo de Interfaz	

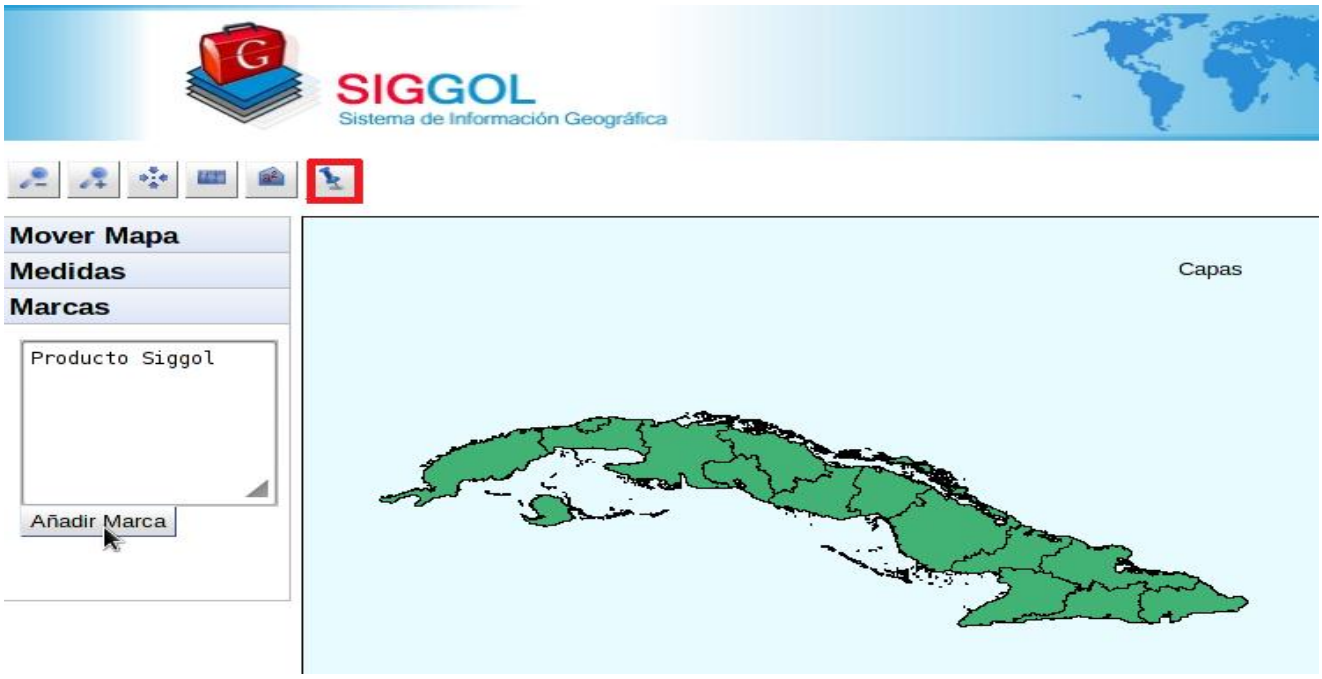

SIGGOL
Sistema de Información Geográfica
Mover Mapa
Medidas
Parcial:22882.48 km²
0
Marcas
Capas
Base Layer
provincia
municipio
Overlays
Marcadores
SIGGOL
Sistema de Información Geográfica
Mover Mapa
Medidas
Parcial:22882.48 km²
Total:58739.75 km²
Marcas
Capas

3.3.2.6 Añadir Marca

Caso de Uso:	Añadir Marca
Actores:	Usuario
Propósito	Este caso de uso se lleva a cabo con el objetivo de añadir un marcador en el mapa en determinado lugar deseado por el usuario.
Resumen:	El caso de uso se inicia cuando el actor desea añadir una marca en el mapa en determinado punto y termina cuando el sistema muestra la marca en el mapa.
Precondiciones:	
Referencias	RF 6
Prioridad	Secundario

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la herramienta “Añadir Marca”.	**2.** El sistema despliega el panel Añadir Marca que se encuentra en la parte izquierda de la aplicación.
3. El usuario escribe el texto de la marca y pulsa el botón Añadir Marca.	**5.** El sistema muestra la marca añadida en el mapa.
4. El usuario da click en el lugar en el mapa donde desea añadir la marca.	**7.** El sistema muestra el texto referente a la marca.
6. El usuario si desea ver la información referente a la marca da click derecho encima de la marca añadida.	

Prototipo de Interfaz



3.4 Conclusiones Parciales

En el presente capítulo se arribó a la propuesta de solución a través de la modelación del dominio y el planteamiento de los requisitos funcionales y no funcionales, quedando explícito qué es lo que el sistema debe hacer, así como sus cualidades o propiedades. Cumpliéndose este objetivo parcial, se crea una base sólida para el diseño e implementación del sistema, ganando claridad en su concepción. Se logró profundizar en el conocimiento de las funcionalidades básicas (críticas) que debe poseer este Sistema de Información Geográfica.

CAPÍTULO 4: Construcción de la solución propuesta

Introducción

En este capítulo se presentan los principales elementos correspondientes a la construcción de la solución propuesta. Se exponen los artefactos pertenecientes a las etapas del flujo de trabajo de diseño, implementación y prueba. Se ofrece una vista más detallada del sistema mostrando la arquitectura propuesta. Además, se abordan los patrones de diseño y se presenta el modelo de implementación a través del diagrama de despliegue obtenido del diseño y los resultados obtenidos en las pruebas de caja negra desarrolladas al producto.

4.1 Patrones de Diseño Utilizados

Un patrón de diseño es un conjunto de reglas que describen cómo afrontar tareas y solucionar problemas que surgen durante el desarrollo de software. (Larman, 1999) Son el esqueleto de las soluciones a problemas comunes. Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Estos son independientes del lenguaje en el que se utilicen (siempre y cuando el lenguaje sea orientado a objetos). Es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular. Un patrón de diseño identifica: Clases, Instancias, Roles, Colaboraciones y la distribución de responsabilidades. Los patrones de diseño se encuentran divididos en dos grupos: patrones GRASP y patrones GOF.

A continuación se muestran algunos de los patrones más relevantes utilizados en el diseño de la propuesta de solución.

4.1.1 Patrones de Diseño (GRASP)

GRASP son patrones generales que describen los principios fundamentales de diseño de objetos de software para asignación de responsabilidades, es el acrónimo de General Responsibility Assignment Software Patterns. Constituyen un apoyo para la enseñanza que ayuda a entender el diseño de objeto esencial y aplica el razonamiento para el diseño de una forma sistemática, racional y explicable.

Para el desarrollo de la solución se utilizaron los patrones GRASP siguientes: experto, creador, bajo acoplamiento, alta cohesión y el controlador.

4.1.1.2 Patrón Experto

Se utiliza en el diseño para asignar responsabilidades a la clase que mayor información posee para cumplir una tarea. Esto trajo como beneficio que se conservara el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide.
Dentro del sistema se utilizará este patrón en clases como DataPicker que debe ser la encargada de proporcionar los datos cargados de una fuente de datos así como modificar los mismos.

4.1.1.2 Patrón Creador

Se utiliza para guiar la asignación de responsabilidades relacionadas con la creación de objetos, esta tarea es muy frecuente en los sistemas orientados a objetos. El patrón Creador indica que la clase incluyente del contenedor o registro es idónea para asumir la responsabilidad de crear el objeto contenido o registrado.
Este patrón se utilizó en los casos de la instanciación de las clases dentro de la interfaz, los controladores y otros que contienen objetos como Map, MapWidget, entre otros.

4.1.1.3 Patrón Bajo Acoplamiento

Se utiliza para asignar una responsabilidad de modo que su colocación no incremente el acoplamiento tanto que produzca los resultados negativos propios de un alto acoplamiento.
Este patrón soporta el diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. No puede considerarse en forma independiente de otros patrones como Experto o Alta Cohesión, sino que más bien ha de incluirse como uno de los principios del diseño que influyen en la decisión de asignar responsabilidades.
Este patrón se utilizará en la asignación de responsabilidades a clases de manera que se divida por funcionalidades de forma tal que un cambio en una de estas genere poco cambio en otros, es el caso del controlador que tiene un conjunto de métodos para cada funcionalidad por separado reduciendo la dependencia de las mismas.

4.1.1.4 Patrón Alta Cohesión

Se utiliza para las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Es un patrón evaluativo que el desarrollador aplica al valorar sus decisiones de diseño, se da una alta cohesión funcional cuando los elementos de un componente colaboran para producir algún comportamiento bien definido.

4.1.1.5 Patrón Controlador

Se utiliza al manejar la clase controladora como manipuladora de los eventos del sistema. El patrón controlador se encarga de asignar la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase.
Se debe utilizar la misma clase de controlador con todos los eventos del sistema en el mismo caso de uso. Este patrón se evidencia en la dependencia de algunas clases que necesitan elementos que brindan otras, tal como es la clase Controller que necesita que WMSContructor le devuelva una instancia completa de una capa WMS, evitando así, tener que construirla ella misma.

4.2 Vista lógica de la arquitectura de la solución

La vista lógica es una abstracción de acuerdo a criterios diferentes que pueden usar su propia notación y definir de forma independiente el significado de los componentes, relaciones, argumentos, principios y pautas; es la base sobre la cual los arquitectos construyen modelos de aplicación que representan la perspectiva lógica de la arquitectura de una aplicación.

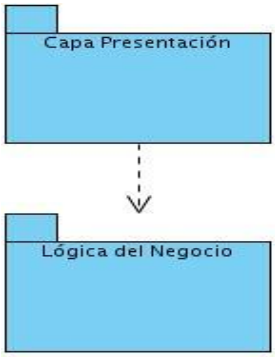


Figura 5. Vista lógica de la Arquitectura

El sistema ha sido diseñado teniendo en cuenta el patrón en capas que define una organización jerárquica tal, que cada capa proporciona servicios a la capa inmediata superior y se sirve de las prestaciones que le brinda la inmediata inferior, estableciéndose así la capa de presentacióny la capa de lógica del negocio.
Este sistema no presenta la capa de acceso a datos puesto que no se cuenta con una base de datos ni alguna otra fuente de datos.

Capa de presentación: Es la que ve el usuario (también se la denomina “capa de usuario”), presenta el sistema al usuario, le comunica la información y captura la información del usuario en un mínimo de proceso (realiza un filtrado previo para comprobar que no hay errores de formato). Esta capa se comunica únicamente con la capa de negocio. También es conocida como interfaz gráfica y debe tener la característica de ser “amigable” (entendible y fácil de usar) para el usuario.

Capa de negocio: Es donde residen los programas que se ejecutan, se reciben las peticiones del usuario y se envían las respuestas tras el proceso. Se denomina capa de negocio (e incluso de lógica del negocio) porque es aquí donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados.

4.3 Modelo de Diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de la implementación tienen impacto en el sistema a considerar. (Jacobson, y otros, 2004) En esta etapa lo más importante es la elaboración de los diagramas de clases del diseño.

4.3.1 Diagramas de Clases del Diseño

El diagrama de clases del diseño describe gráficamente las especificaciones de las clases de software y de las interfaces en una aplicación. (Larman, 1999)
Incluye la siguiente información:

- Clases, asociaciones y atributos.
- Interfaces, con sus operaciones y constantes.
- Métodos.
- Navegabilidad.

- Dependencias.

Para la creación del diagrama de clases del diseño primeramente se identifican las entidades software necesarias para desarrollar el sistema, se definen las relaciones existentes entre ellas y los atributos y métodos según la necesidad de la aplicación. Este diagrama es el más importante de la fase de diseño de un software, el cual brinda una visión general para comenzar con la implementación del producto.

A continuación se muestran los diagramas de clases del diseño de la investigación:

4.3.1.1 Diagrama General de Clases del Diseño

Se realizó un diagrama general de clases de diseño donde se representa el sistema desde una vista amplia con cada una de las clases del sistema. Luego se representa el diagrama para cada CU en particular, expresando los elementos comunes del general y los específicos de cada uno.

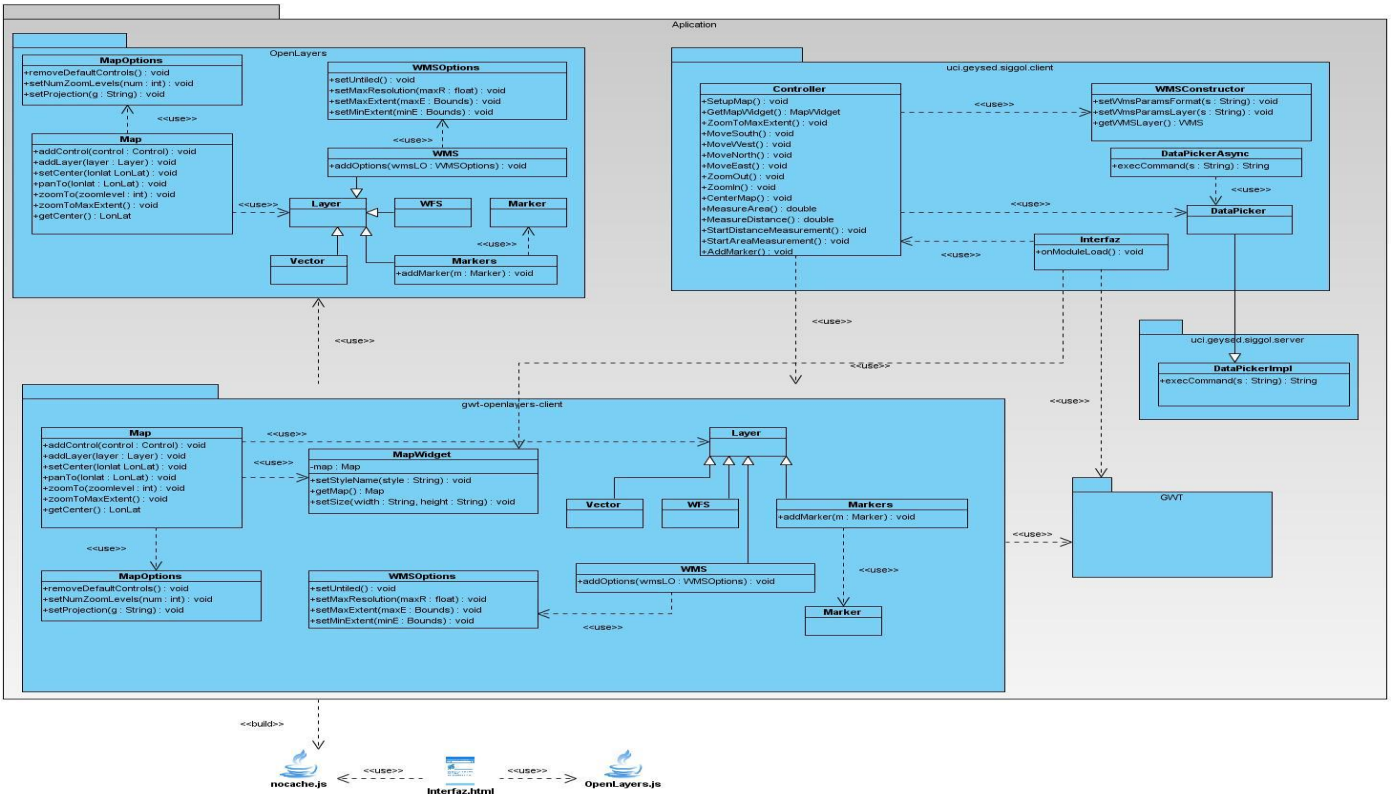


Figura 6. Diagrama General de Clases del Diseño

4.3.1.2 DCD del CU Realizar Zum

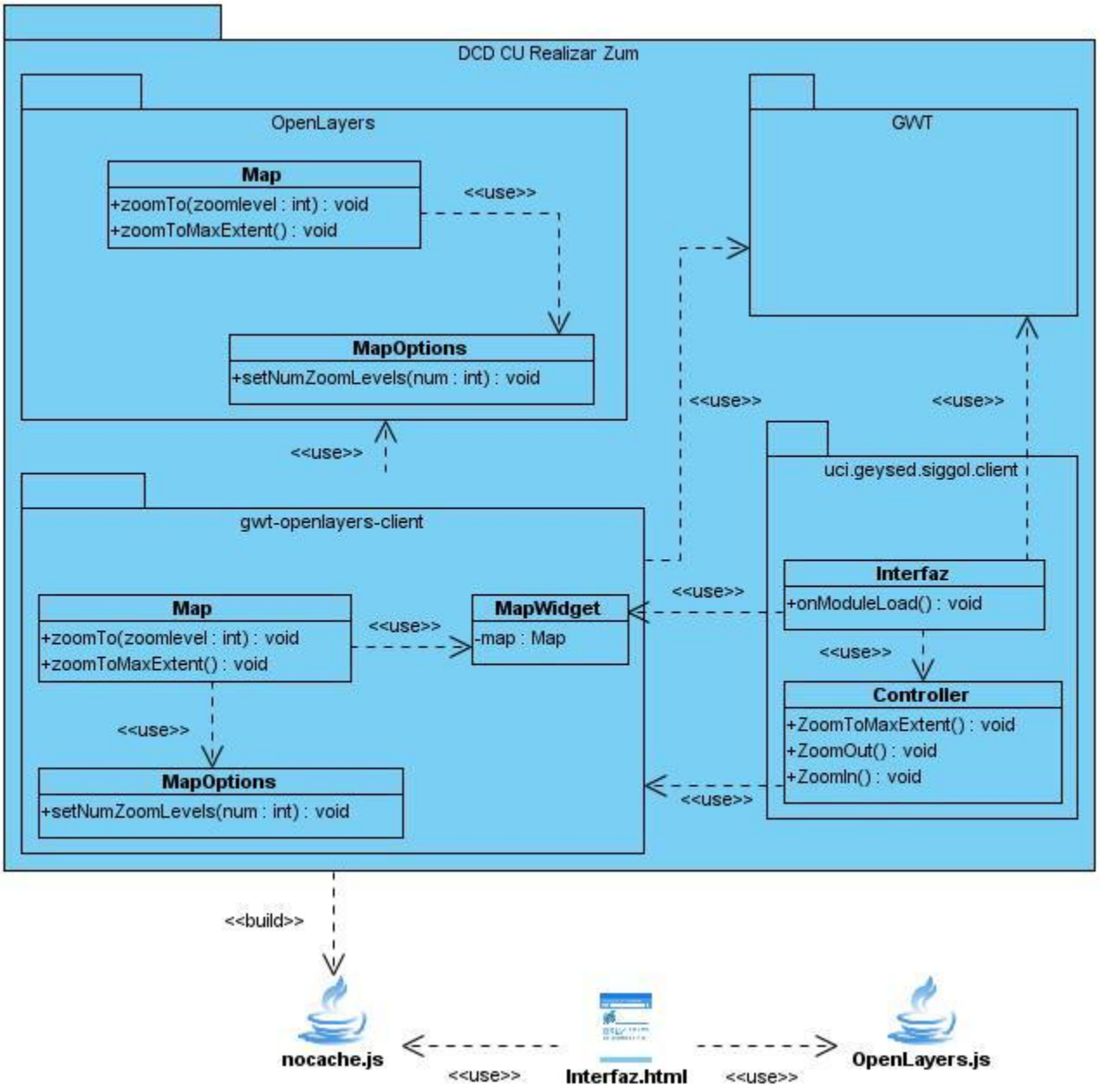


Figura 7. DCD del CU Realizar Zum

4.3.1.3 DCD del CU Mover Mapa

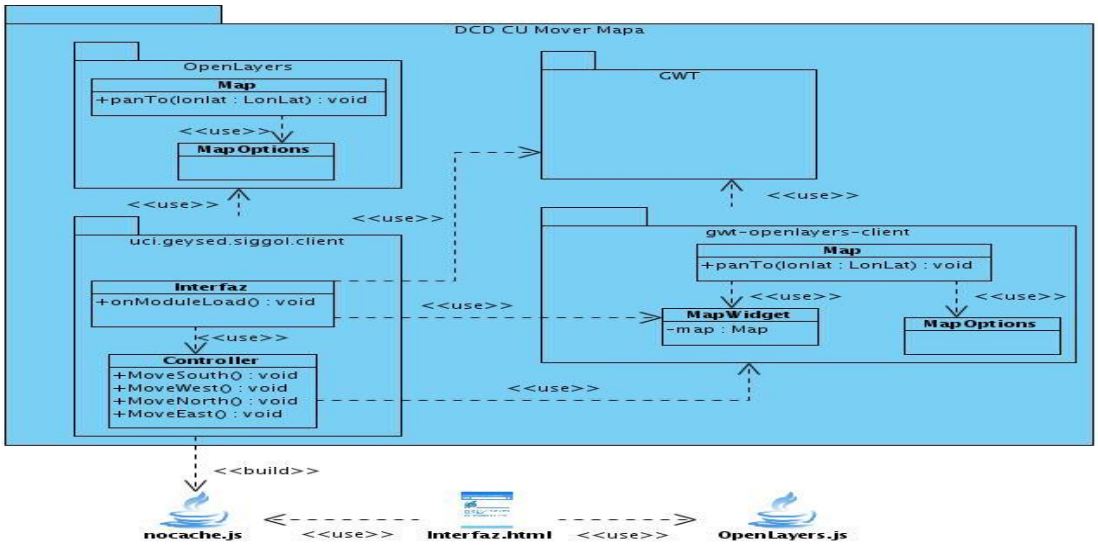


Figura 8. DCD del CU Mover Mapa

4.3.1.4 DCD del CU Medir Distancia

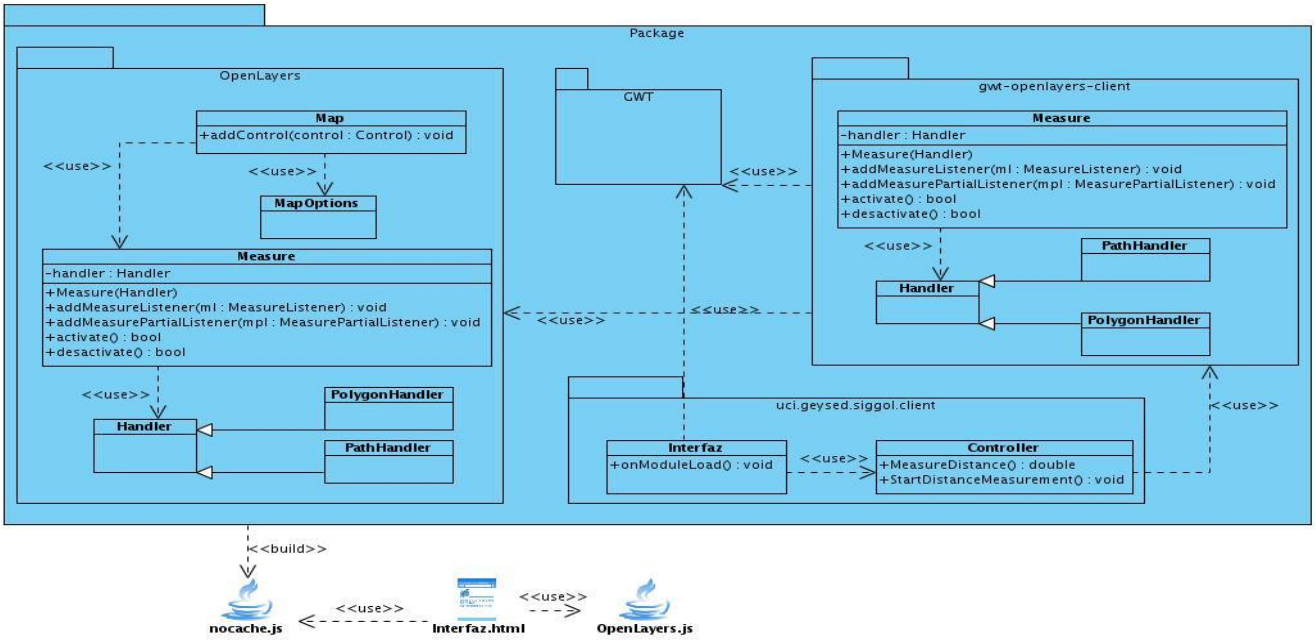


Figura 9. DCD del CU Medir Distancia

4.3.1.5 DCD del CU Calcular Área

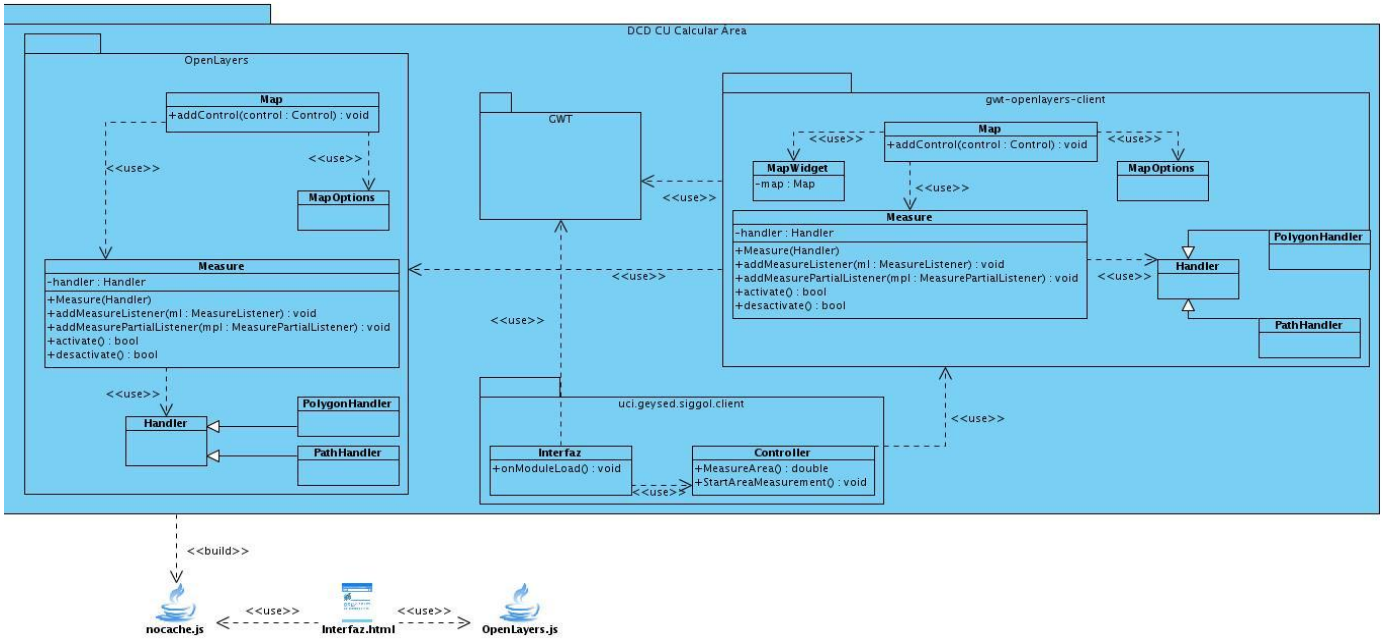


Figura 10. DCD del CU Calcular Área

4.3.1.6 DCD del CU Añadir Marca

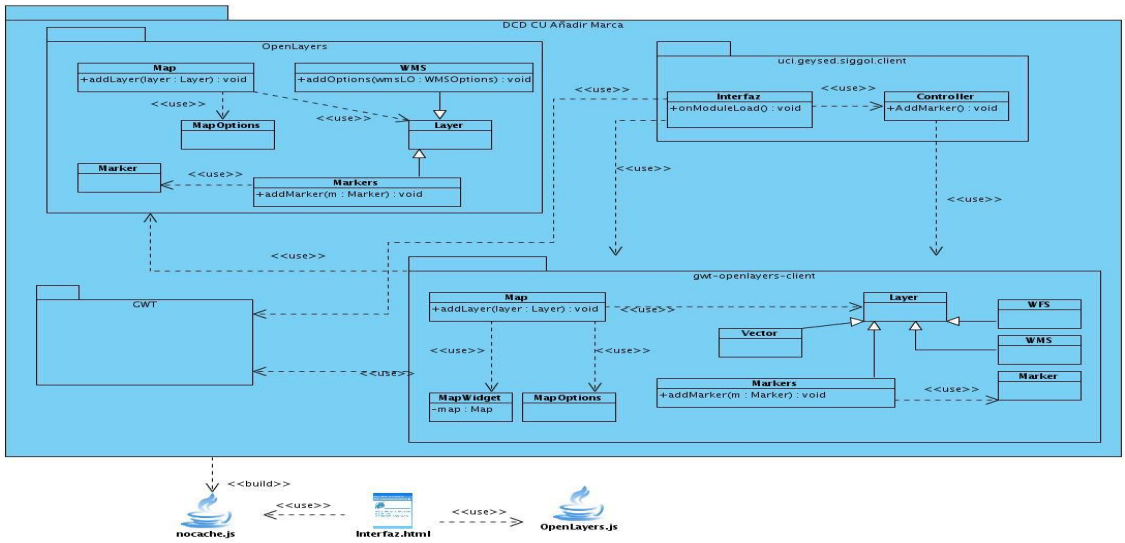


Figura 11. DCD del CU Añadir Marca

4.3.1.7 DCD del CU Recentrar Mapa

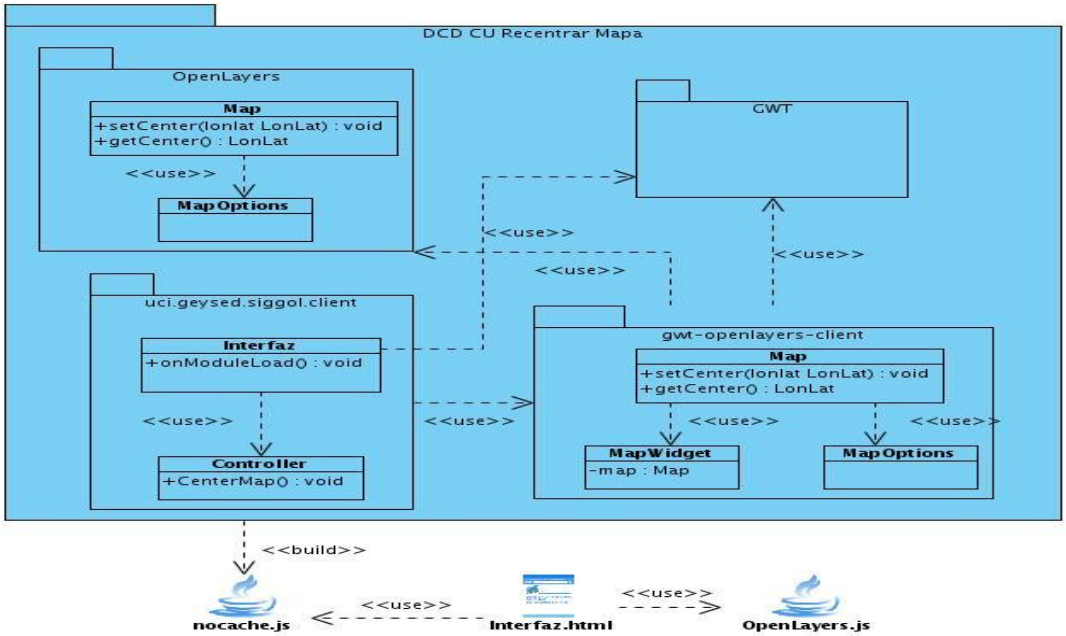


Figura 12. DCD del CU Recentrar Mapa

4.4 Modelo de Implementación

El modelo de implementación describe cómo se implementan las clases del sistema en términos de componentes. Describe además cómo se organizan los componentes de acuerdo al lenguaje o lenguajes de programación utilizados, y cómo dependen algunos componentes de otros.

4.4.1 Diagrama de Componentes

El Diagrama de Componentes muestra un conjunto de elementos del modelo tales como componentes, subsistemas de implementación y sus relaciones. Se utiliza para modelar los aspectos físicos de un sistema así como la vista de implementación estática de un sistema, y muestra la organización y las dependencias lógicas entre un conjunto de componentes software, sean éstos componentes de código fuente, librerías, binarios o ejecutables.

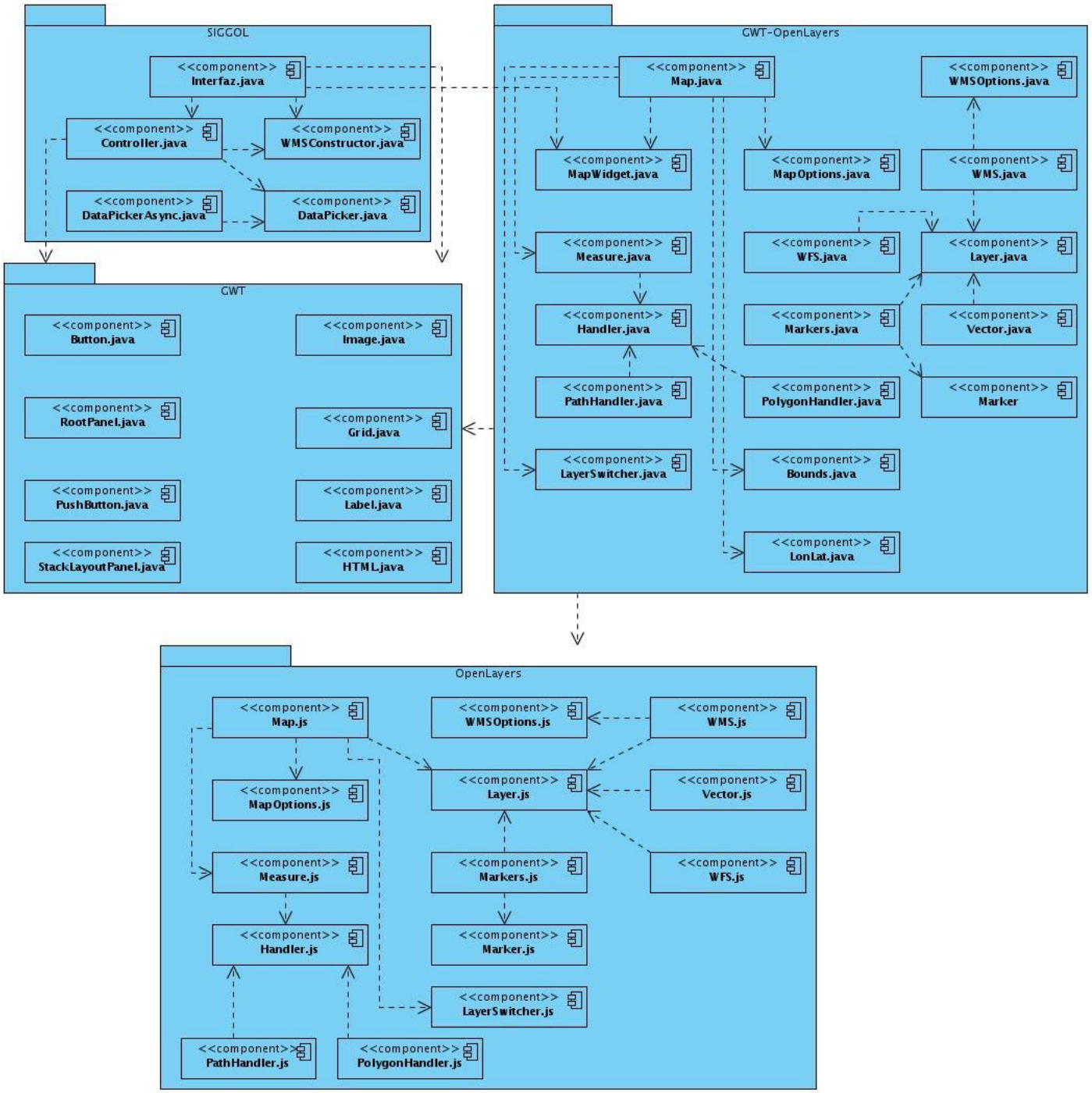


Figura 13. Diagrama de Componentes

4.5 Modelo de Despliegue

El Modelo de Despliegue es un modelo de objetos que muestra las relaciones físicas de los distintos nodos que componen un sistema y cómo se reparten los componentes sobre dichos nodos. La vista de despliegue representa la disposición de las instancias de componentes de ejecución en instancias de nodos conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria.

A continuación se muestra el modelo de despliegue del sistema:

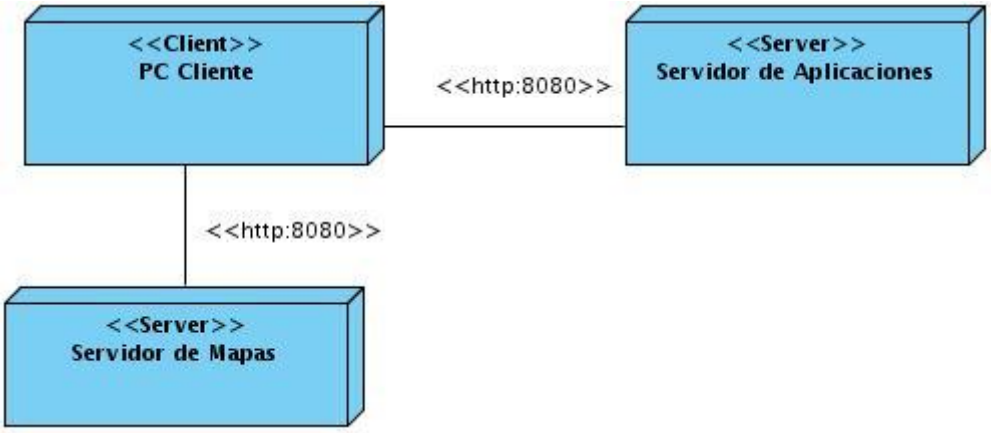


Figura 14. Modelo de Despliegue

4.6 Pruebas del sistema propuesto

Las pruebas de software se integran dentro de las diferentes fases del ciclo de vida del producto dentro de la Ingeniería de software. Estas son un elemento crítico para la garantía de la calidad del producto y representa una revisión final de las especificaciones del diseño y de la codificación. El objetivo principal de las pruebas es asegurar que el sistema cumpla con las especificaciones requeridas y eliminar los posibles defectos que este pudiera tener.

4.6.1 Diseño de Casos de Prueba

Un caso de prueba constituye un conjunto de condiciones o variables bajo las cuales se determina si un requisito es parcial o completamente satisfactorio. Su propósito es especificar una forma de probar el

sistema que incluya las entradas, los resultados esperados y las condiciones bajo las que ha de probarse. Los casos de prueba ayudan a validar que el sistema desarrollado realice las funciones para las que ha sido creado en base a los requisitos del usuario, por lo que resulta importante diseñar al menos un caso de prueba para cada requisito funcional del sistema con el objetivo de asegurar que todas las funcionalidades sean comprobadas.

4.6.2 Pruebas de Caja Negra

La prueba de Caja Negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. (Pressman, 2007)

Muchos autores consideran que estas pruebas permiten encontrar: (Pressman, 2007)

- ✓ Funciones incorrectas o ausentes.
- ✓ Errores de interfaz.
- ✓ Errores en estructuras de datos o en accesos a las Bases de Datos externas.
- ✓ Errores de rendimiento.
- ✓ Errores de inicialización y terminación.

Para desarrollar la prueba de caja negra existen varias técnicas, entre ellas están: (Pressman, 2007)

1. Técnica de la Partición de Equivalencia: Esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
2. Técnica del Análisis de Valores Límites: Esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
3. Técnica de Grafos de Causa-Efecto: Es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

La técnica que se empleó en el sistema es la Técnica de la Partición de Equivalencia. Dentro del método de Caja Negra esta técnica es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico. La

partición equivalente se dirige a la definición de casos de pruebas que descubran clases de errores, reduciendo así el número de clases de prueba que hay que desarrollar.
Objetivo: El objetivo es encontrar el mayor número posible de errores con una cantidad razonable de esfuerzo, aplicado sobre un lapso de tiempo realista.

Alcance: El proceso de pruebas de caja negra se va a centrar principalmente en los requisitos funcionales del software para verificar la interacción con el usuario.

Descripción: Estas pruebas permiten evaluar todos los requisitos funcionales del sistema.

Casos de Prueba: Se utiliza para verificar que se hayan cumplido los requerimientos funcionales establecidos anteriormente. Se le realizó la prueba de caja negra a cada caso de uso.

4.6.3 Resultados

Fueron probados seis casos de uso, los que representan el cien por ciento del total de requisitos funcionales. Para cada caso de uso se tuvo en cuenta los diferentes escenarios que pudieran existir, los cuales implican cada una de las posibles interacciones del usuario o combinaciones de situaciones posibles con el fin de evaluar el comportamiento del sistema ante cada funcionalidad.
El resultado de cada prueba coincide con la especificación de requisitos lo que demuestra la correcta implementación de los mismos.
Como ejemplo que muestra lo dicho anteriormente se refleja la sección y el escenario probado para el caso de uso **Añadir Marca**.

Nombre de la sección	**Escenarios de la sección**	**Descripción de la funcionalidad**	**Flujo Central**
SC 1: Añadir Marca.	EC 1.1: Añadir Marca.	Esta funcionalidad permite al usuario añadir una marca en el mapa en determinado punto con la información deseada referente a esta marca y termina cuando el sistema muestra la marca en el mapa.	• El usuario selecciona la herramienta “Añadir Marca”. • El sistema despliega un panel en el lado izquierdo de la aplicación donde aparece el cuadro de texto para que el usuario escriba el contenido de la marca. • El usuario introduce la información y da click en el botón Añadir Marca. • Seguidamente el usuario da click en el mapa donde desea añadir la marca. • El sistema recibe la información y muestra la marca añadida al mapa en el lugar especificado con la información agregada.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Texto de la marca	Campo de texto	No	Se le pueden introducir números, letras y otros tipos de caracteres, siempre hay que introducirle texto a la marca.

ID del escenario	Escenario	Variable 1	Respuesta del Sistema	Resultado de la Prueba
EC 1.1	Añadir Marca	V Aquí está Villa Clara	El sistema permite al usuario añadir una marca en el mapa en determinado punto con la información deseada referente a esta marca y termina cuando el sistema muestra la marca en el mapa.	Resultado Satisfactorio

Ver los demás casos de prueba en los Anexos.

4.7 Conclusiones Parciales

En este capítulo se representaron todos los diagramas de clases del diseño que contribuyeron a una exitosa implementación de todos los casos de uso. Se representó el diagrama de componentes para tener una visión de la estructura física de la aplicación. Se describieron los patrones de diseño utilizados y la arquitectura propuesta que definieron buenas prácticas de programación. También se representó el diagrama de despliegue que muestra cómo están distribuidos los dispositivos de software entre los diferentes nodos. Por último se realizaron las pruebas de caja negra al sistema donde se obtuvo resultados satisfactorios.

CONCLUSIONES

Mediante la planificación de varias tareas al inicio de la investigación, se realizó un Sistema de Información Geográfica de forma exitosa y organizada, del cual se concluyen los siguientes aspectos:

El estudio de diversos SIG en los diferentes ámbitos arrojó que no existe un sistema que integre las tecnologías GWT y OpenLayers, por lo que a partir de ahí se defendió la idea propuesta y se desarrolló la presente investigación. La caracterización y comparación de varias herramientas y tecnologías ayudaron a seleccionar las más indicadas para la realización de la aplicación, lo que contribuyó a un correcto proceso de desarrollo. Con la identificación de los requisitos funcionales quedó explícito lo que el sistema debía hacer, los cuales llevados a casos de uso mostraron un mejor entendimiento acerca del funcionamiento del software, y con las condicionantes de los requisitos no funcionales se evidenció claramente las propiedades y cualidades del producto. A raíz de una arquitectura diseñada y otros patrones de diseño se logró la implementación de una aplicación escalable y a la medida, teniendo el usuario en todo momento el control sobre la aplicación. Para evaluar el comportamiento del sistema ante cada funcionalidad se realizaron pruebas de caja negra, donde los resultados fueron satisfactorios para un cien por ciento de todos los casos de uso planteados.

Se logró el objetivo general de la investigación por lo que el sistema dio solución a las necesidades del Departamento Geoinformática. La aplicación está lista para su utilización y con la capacidad de mejorar ampliamente en funcionalidades adaptándola a las condiciones del cliente que la solicite.

RECOMENDACIONES

Se recomienda que se le agregue a la aplicación:

- Soporte para base de datos.

- Elementos de comunicación con el servidor.

- Otras funcionalidades.

REFERENCIAS BIBLIOGRÁFICAS

An Overview on Current Free and Open Source Desktop. **Steiniger, Stefan and Bocher, Erwan. 2008.** T2N 1N4, Canadá : Ecole Central de Nantes, 2008.

Asociación Java Hispano. 2009. Java con Google Web Toolkit. [En línea] 2009. [Citado el: 20 de Enero de 2011.] http://www.javaHispano.org.

Báez, Rafael González. 2010. *Cine, Cartografía y Matemáticas.* 2010. 28.

Beck y Kent. 1999. *Extreme Programming Explained.* Embrace Change : s.n., 1999.

Calderón, Amaro. 2007. *Metodologías ŸAgiles.* Trujillo Perú : s.n., 2007. s.n.

Carrillo, German. 2010. Geo Tux. [En línea] 30 de Noviembre de 2010. [Citado el: 4 de Diciembre de 2010.] http://geotux.tuxfamily.org/index.php?option=com_myblog&show=comparaci%F3n-de-clientes-web-para-sig-v.5.html&Itemid=59.

COMAS, D. y RUIZ, E. 1993. *Datos espaciales y sus fundamento en los sistemas de información geográfica.* Barcelona : s.n., 1993.

Desarrollo Web. 2005. Desarrollo Web. [En línea] 2005. [Citado el: 16 de 01 de 2011.] http://www.desarrolloweb.com/articulos/392.php.

Echeverría, X. V. 2007. ¿ QUÉ ES UN SIG ? [En línea] 2007. http://www.navactiva.com/web/es/descargas/pdf/atic/sig1.pdf.

Entorno Virtual de Aprendizaje. Entorno Virtual de Aprendizaje. [En línea] UCI. [Citado el: 20 de enero de 2010.] http://eva.uci.cu/mod/resource/view.php?id=33748..

Gattaca. 2009.*Presentación de la Metodología Microsoft Solution Framework (WSF).* España : s.n., 2009.

Gis Europe*.* **Huntingdon. 1992.** 1992, REVISTA GEOEUROPE .

Google Groups. Geoserver. 2010. [En línea] 2010. [Citado el: 2 de 12 de 2010.] http://groups.google.com/group/geoserver-es.

Gutiérrez Puebla, Javier y Gould, Michael. 1994.*¿Qué son los Sistemas de Información Geográfica?* 1994.

Hill, McGraw. 1997. *Ingeniería del Software: Un enfoque práctico.* 1997.

Hirsch. 2008. *Metodologías de desarrollo.* Madrid : s.n., 2008.

2010. http://www.metacarta.com. [En línea] 2010. [Citado el: 11 de noviembre de 2010.] http://www.metacarta.com/about-us-overview.htm.

HUXHOLD, W. E. y LEVINSOHN. 1995. *Managing geographic information system .* New York : Oxford Universi ty Press, 1995.

Ibero, General Ibáñez de. 2009. Cartografía. [En línea] 2009. [Citado el: 09 de Diciembre de 2010.] http://www.cnig.es/.

Informáticas, Departamento Geoinformática de la Universidad de las Ciencias. 2010. siguci.uci.cu. [En línea] 2010. http://siguci.uci.cu.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2004. *Proceso Unificado del Desarrollo de Sofware .* Habana : Félix Varela, 2004.

Jiménez Lezama, Juan Carlos. 2005. *"MODELADO DE UN BEAN DE BÚSQUEDA MEDIANTE UML".* 2005.

Larman, Craig. 1999. UML y Patrones,Introducción al análisis y diseño orientado a objetos. Mexico : s.n., 1999.

Lenguajes de Programación. 2010. Lenguajes de Programación. [En línea] 2010. [Citado el: 24 de 01 de 2011.] http://www.lenguajes-de-programacion.com/programacion-java.shtml.

LETELIER, P. 2003. Metodologías Ágiles en el Desarrollo de Software. [En línea] 2003. [Citado el: 25 de Enero de 2011.] http://issi.dsic.upv.es/tallerma.

Maguire, D.J., M.F., Goodchild y D.W., Rhind. 1997. *"Geographic Information Systems: principles, and applications".* New York : Universidad de Heidelberg, 1997.

MapGuide Open Source. OSGeo. 2007. OSGeo. [En línea] 06 de 03 de 2007. [Citado el: 2 de 12 de 2010.] 2007-03-06.

Martínez, Alejandro y Martínez, Raúl. 2008. *Guía a Rational Unified Process.* 2008.

MCDONNELL, BURROUGH. 1997. *LOS SISTEMAS DE INFORMACIÓN GEOGRÁFICA.* 1997.

Microsystems, Sun. 2005. http://java.sun.com. [En línea] 12 de Junio de 2005. [Citado el: 11 de Noviembre de 2010.] http://java.sun.com/docs/books/jvms/second_edition/jvms-clarify.html.

Murillo Alfaro, Félix. 1999. *Herramientas Case.* Sevilla : s.n., 1999.

—. 2010. Quantum GIS. [En línea] 2010. [Citado el: 2 de 12 de 2010.] http://www.qgis.org/.

Pressman, Roger S. 2007. Ingeniería de Software. Un enfoque práctico. 2007.

Profesores, Consejo de. 2007. *Introducción a los Sistemas de Información.* México : s.n., 2007.

Rodríguez, Hendrik Cesar. 2008. *Guía metodológica para el desarrollo de Sistemas de Información Geográfica en la Universidad de las Ciencias Informáticas.* Ciudad de la Habana : s.n., 2008.

SANABRIA., R. D. y RODRÍGUEZ, N. 1998. *LOS SISTEMAS DE INFORMACION GEOGRÁFICA.* 1998.

SANCHEZ, M. A. M. 2004. Metodologías De Desarrollo De Software. [En línea] 2004. http://www.informatizate.net/articulos/pdfs/metodologias_de_desarrollo_de_software_07062004.pdf..

Silva, Dr. José Luis Batista. 2009. Mapping Interactivo. [En línea] 2009.

http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=1051.

Sitio Oficial de Apache. 2008. http://www.apache.org. [En línea] 2008. [Citado el: 05 de 02 de 2011.] http://www.apache.org/docs/2.2/new_features_2_0.html.

Sitio Oficial de Eclipse. 2003. Eclipse Platform Technical Overview. [En línea] 2003. [Citado el: 30 de 01 de 2011.] http://www.eclipse.org/whitepapers/eclipse-overview.pdf. .

Sitio oficial de Java. Sitio Oficial de Java. [En línea] http://www.java.com/es/about/.

Sitio Oficial de OpenLayers. 2010. OpenLayers. [En línea] 2010. [Citado el: 18 de Enero de 2011.] http://www.openlayers.org.

Toolkit, Google Web. 2009. *Documentación de GWT.* España : s.n., 2009.

—. 2010. Google Web Toolkit. [En línea] 2010. [Citado el: 26 de Enero de 2011.] http://code.google.com/intl/es/webtoolkit/overview.html..

UCI. 2009. Revista GeoNews. [En línea] 15 de Febrero de 2009. http://revistageonews.wordpress.com/2009/02/15/proyecto-sig-uci/.

Universidad de las Ciencias Informáticas. 2011. Entorno Virtual de Aprendizaje. [En línea] 2011. [Citado el: 19 de Enero de 2011.] http://eva.uci.cu/mod/assignment/view.php?id=37221.

Visual Paradigm International. 2010. Analyst sends diagrams to ElaborView server. [En línea] 2010. [Citado el: 28 de 01 de 2011.] http://www.visual-paradigm.com/product/vpuml/editions/standard.js.

BIBLIOGRAFÍA

Calderón, Amaro. 2007. *Metodologías ŸAgiles.* Trujillo Perú : s.n., 2007. s.n.

COMAS, D. y RUIZ, E. 1993. *Datos espaciales y sus fundamento en los sistemas de información geográfica.* Barcelona : s.n., 1993.

Desarrollo Web. 2005. Desarrollo Web. [En línea] 2005. [Citado el: 16 de 01 de 2011.] http://www.desarrolloweb.com/articulos/392.php.

Echeverría, X. V. 2007. ¿ QUÉ ES UN SIG ? [En línea] 2007. http://www.navactiva.com/web/es/descargas/pdf/atic/sig1.pdf.

Entorno Virtual de Aprendizaje. Entorno Virtual de Aprendizaje. [En línea] UCI. [Citado el: 20 de enero de 2010.] http://eva.uci.cu/mod/resource/view.php?id=33748..

Gattaca. 2009. *Presentación de la Metodología Microsoft Solution Framework (WSF).* España : s.n., 2009.

Gutiérrez Puebla, Javier y Gould, Michael. 1994.*¿Qué son los Sistemas de Información Geográfica?* 1994.

Hill, McGraw. 1997. *Ingeniería del Software: Un enfoque práctico.* 1997.

HUXHOLD, W. E. y LEVINSOHN. 1995. *Managing geographic information system .* New York : Oxford Universi ty Press, 1995.

Ibero, General Ibáñez de. 2009. Cartografía. [En línea] 2009. [Citado el: 09 de Diciembre de 2010.] http://www.cnig.es/.

Informáticas, Departamento Geoinformática de la Universidad de las Ciencias. 2010. siguci.uci.cu. [En línea] 2010. http://siguci.uci.cu.

Jacobson. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000. s.n..

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2004.*Proceso Unificado del Desarrollo de Sofware .* Habana : Félix Varela, 2004.

Jiménez Lezama, Juan Carlos. 2005. *"MODELADO DE UN BEAN DE BÚSQUEDA MEDIANTE UML".* 2005.

Larman, Craig. 1999. UML y Patrones,Introducción al análisis y diseño orientado a objetos. Mexico : s.n., 1999.

Lenguajes de Programación. 2010. Lenguajes de Programación. [En línea] 2010. [Citado el: 24 de 01 de 2011.] http://www.lenguajes-de-programacion.com/programacion-java.shtml.

LETELIER, P. 2003. Metodologías Ágiles en el Desarrollo de Software. [En línea] 2003. [Citado el: 25 de Enero de 2011.] http://issi.dsic.upv.es/tallerma.

Maguire, D.J., M.F., Goodchild y D.W., Rhind. 1997. *"Geographic Information Systems: principles, and applications".* New York : Universidad de Heidelberg, 1997.

MapGuide Open Source. OSGeo. 2007. OSGeo. [En línea] 06 de 03 de 2007. [Citado el: 2 de 12 de 2010.] 2007-03-06.

Marañón, Gonzalo Álvarez. 2009. Características del lenguaje C#. [En línea] 2009. [Citado el: 21 de 01 de 2001.] http://www.iec.csic.es/CRIPTONOMICON/java/quesc#.html.

Martínez, Alejandro y Martínez, Raúl. 2008. *Guía a Rational Unified Process.* 2008.

MCDONNELL, BURROUGH. 1997. *LOS SISTEMAS DE INFORMACIÓN GEOGRÁFICA.* 1997.

Microsystems, Sun. 2005. http://java.sun.com. [En línea] 12 de Junio de 2005. [Citado el: 11 de Noviembre de 2010.] http://java.sun.com/docs/books/jvms/second_edition/jvms-clarify.html.

Rodríguez, Andrea. 2008. *Curso de Bases de Datos Espaciales.* Universidad de Concepción : s.n., 2008.

Rodríguez, Hendrik Cesar. 2008. *Guía metodológica para el desarrollo de Sistemas de Información Geográfica en la Universidad de las Ciencias Informáticas.* Ciudad de la Habana : s.n., 2008.

Scribd. 2010. www.scribd.com/doc. *www.scribd.com/doc.* [En línea] 2010. [Citado el: 23 de 01 de 2011.] http://www.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE. .

Silva, Dr. José Luis Batista. 2009. Mapping Interactivo. [En línea] 2009. http://www.mappinginteractivo.com/plantilla-ante.asp?id_articulo=1051.

Sitio Oficial de Apache. 2008. http://www.apache.org. [En línea] 2008. [Citado el: 05 de 02 de 2011.] http://www.apache.org/docs/2.2/new_features_2_0.html.

Sitio Oficial de Eclipse. 2003. Eclipse Platform Technical Overview. [En línea] 2003. [Citado el: 30 de 01 de 2011.] http://www.eclipse.org/whitepapers/eclipse-overview.pdf. .

Sitio oficial de Java. Sitio Oficial de Java. [En línea] http://www.java.com/es/about/.

Sitio Oficial de OpenLayers. 2010. OpenLayers. [En línea] 2010. [Citado el: 18 de Enero de 2011.] http://www.openlayers.org.

SWT-JFace-Eclipse. 2010. SWT-JFace-Eclipse. [En línea] 2010. [Citado el: 28 de 01 de 2011.] http://www.java2s.com/Code/Java/SWT-JFace-Eclipse/CatalogSWT-JFace-Eclipse.html.

Toolkit, Google Web. 2009. *Documentación de GWT.* España : s.n., 2009.

—. 2010. Google Web Toolkit. [En línea] 2010. [Citado el: 26 de Enero de 2011.] http://code.google.com/intl/es/webtoolkit/overview.html..

GLOSARIO DE TÉRMINOS

SIG: Sistema de Información Geográfica

GEySED: Centro de Desarrollo de la Universidad de las Ciencias Informáticas, específicamente de la Facultad 6 que incluye los departamentos Geoinformática y Señales Digitales.

GWT: Framework de desarrollo en Java de código abierto para aplicaciones web.

OpenLayers: Es una librería JavaScript que permite mostrar un mapa dinámico y marcadores cargados desde cualquier fuente de datos.

IDE: Entorno de Desarrollo Integrado.