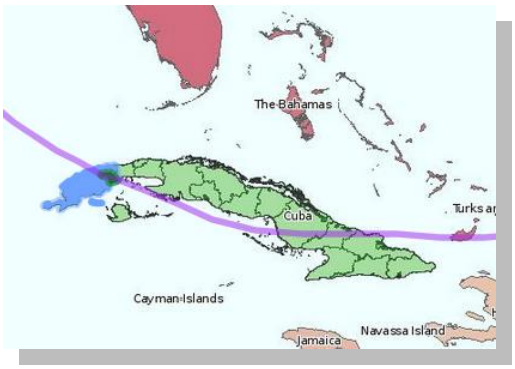
Universidad de las Ciencias Informáticas
Facultad 6

Sistema de Información Geográfica para el análisis semántico de la trayectoria de Huracanes en Cuba



Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Autor: Pedro Manuel Salas Leyva

Tutor (es): MsC. Yuniel Eliades Proenza Arias

Ing. Grethell Castillo Reyes

"Año 57 de la Revolución"
La Habana, Cuba
Junio 2015

"Un hombre provisto de papel, lápiz y goma, y con una sujeción a una disciplina estricta, es en efecto una máquina de Turing universal"

Alan Mathison Turing

Declaración de autoría

Declaro ser el único autor de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Pedro Manuel Salas Leyva

Autor

_____

MsC. Yuniel Eliades Proenza Arias

Tutor

_____

Ing. Grethell Castillo Reyes

Tutor (a)

Datos de contacto

Tutor: MsC. Yuniel Eliades Proenza Arias.

Edad: 33.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Títulos: Ingeniero Informático, Máster en Ingeniería de Software e Inteligencia Artificial.

Síntesis del Tutor: Graduado de Ingeniería Informática en el año 2006 de la Universidad de Holguín y CUJAE. Máster en Ingeniería de Software e Inteligencia Artificial por la Universidad de Málaga en el 2011. Se ha desempeñado como analista y desarrollador de aplicaciones Web y Desktop. Tiene experiencia en el trabajo con Sistemas de Información Geográfica y el desarrollo de Ontologías.

E-mail: yproenza@uci.cu.

Tutor (a): Ing. Grethell Castillo Reyes.

Edad: 25.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas (UCI).

Títulos: Ingeniera en Ciencias Informáticas.

Síntesis de la Tutor (a): Graduada de Ingeniera en Ciencias Informáticas en el 2012 en la UCI. Se ha desempeñado como desarrolladora y líder del proyecto “GeneSIG 2”. Posee experiencia en el desarrollo de aplicaciones web y en el trabajo con la plataforma GeneSIG. Tiene experiencia en el desarrollo de Sistemas de Información Geográfica.

E-mail: gcreyes@uci.cu.

Dedicatoria

Dedico esta tesis a la memoria de mis tíos Rafael y Santiago Salas quienes donde quiera que estén, sé que se sentirán orgullosos de su sobrino.

A mi mamá Miladys y mi abuela Inés por todo su amor, sacrificio y dedicación desde el 14 de julio de 1989.

Agradecimientos

Agradezco a mi mamá Miladys por todo lo que me ha dado toda la vida, por esperar siempre lo mejor de mí y desear el título más que yo.

A mi abuela Inés que desea el título más que mi mamá, que siempre ha estado cuando la he necesitado sin importar lo difícil que fuera, para darme su amor y cariño. A mi abuelo Ángel que ha sido el mejor soldado de la retaguardia como dice a veces y que sin él, no lo hubiese logrado.

A mi papá Pedro José, por apoyarme siempre y confiar que podía lograrlo incluso cuando yo no lo hacía. A mis hermanos Pedro Ernesto, Rafael, Emanuel y Susana por todo su amor. A mi abuelo Santiago por quererme tanto y desear el título tanto como mi abuela Inés.

A mis tías Magalys, Niuris, Zadys, Miriam, Cristina, Kenia y Ana Alicia por quererme como a sus propios hijos. A mi tío Ronny por ser mi ejemplo y mi inspiración. A mis tíos Yimy y Javier, Ernesto, José Ramón, Tino, Guillermo, Sandor y Lucas por todo su apoyo. A mis tíos Santy y Rafael que no pudieron verme pero que fueron un ejemplo e inspiración.

A mis tutores, que sin ellos no habría sido posible nada de esto.

A mis amigos Luisma, Yili, Isis, Anita y Tere por ser los mejores del mundo. A Dayana, Ana, Nelson, Ernesto, Janier y Jairo.

A la gente de mi aula. A mis profes, y el Mayor de los agradecimientos para Andy que ha sido mi columna vertebral desde que nos conocimos. A Adela, Anita y Andrés.

Resumen

Los Sistemas de Información Geográfica han alcanzado un alto grado de popularidad y aceptación en el mundo, siendo ampliamente utilizados en diversas áreas de la economía y la sociedad. Estos sistemas permiten visualizar eventos sobre mapas, posibilitando ubicarlos en el lugar o región que ocurrieron, así como evaluar su impacto en un área determinada. A pesar de las grandes ventajas que ofrecen a partir de la vinculación de la información espacial como valor adicional, existen necesidades mayores de consulta de información. Cuando la búsqueda no está directamente asociada a un parámetro puramente espacial o está asociado a semántica espacial, los Sistemas de Información Geográfica convencionales no pueden ayudar. Debido a esto, surgen los denominados Sistemas de Información Geográfica Guiados o Gobernados por Ontologías. Estos sistemas, además de enriquecer la información socio-política con información espacial, incorporan un valor mayor, el procesamiento de información semántica. La capacidad de procesamiento semántico radica en la utilización de ontologías para este fin. Las ontologías funcionan como una base de conocimientos en dichos sistemas, en las cuales se representa cada individuo con toda su información. La presente investigación tiene como principal objetivo, desarrollar un Sistema de Información Geográfica para el análisis semántico de la trayectoria de los huracanes que han afectado a Cuba. Para el cumplimiento de este objetivo se realiza un estudio preliminar sobre el tema para destacar aspectos importantes, posibilitando en etapas posteriores, el modelado, diseño e implementación de la solución informática. Finalmente se verifica y valida el resultado generado, con el fin de determinar el cumplimiento de las expectativas de los mismos sobre la solución informática propuesta.

Palabras Claves: huracanes, ontologías, Sistema de Información Geográfica Guiado por una Ontología, semántica espacial.

Índice de contenidos

Índice de figuras

Índice de tablas

Introducción

Con el transcurso de los años el hombre se ha visto ante situaciones en la cuales debe decidir entre varias alternativas de la vida cotidiana. Por tal motivo, ha tratado de apoyarse en los avances tecnológicos para desarrollar herramientas que le ayuden a escoger las opciones más factibles. De esta manera surgen varios sistemas informáticos orientados fundamentalmente a contribuir al proceso de la toma de decisiones, un ejemplo lo constituyen los Sistemas de Información Geográfica (SIG).

Actualmente el desarrollo de los SIG ha alcanzado límites inesperados en diversas esferas de la sociedad. Pueden ser utilizados en la gestión eficiente del transporte, control de flotas, análisis de riesgos, impacto ambiental, entre otras áreas críticas de la economía y la sociedad. Los mismos incorporan tecnologías e información socioeconómica y espacial, la cual es gestionada sobre mapas, permitiendo interactividad y facilidad para la toma de decisiones. Las ventajas y comodidades son disímiles, pues una vez desarrollado resulta muy sencillo visualizar eventos sobre los mapas, permitiendo ubicarlos en el lugar o región que ocurrieron, así como evaluar su impacto en un área determinada o la población.

El estudio de los huracanes resulta de gran importancia para Cuba que por su posición geográfica es asediada cada año por estos fenómenos. Debido a la intensidad de sus vientos, la velocidad de traslación, la forma alargada y estrecha de Cuba los huracanes ocasionan grandes pérdidas a la economía y los recursos materiales del país. Por tal motivo se hace necesario conocer el comportamiento de esto fenómenos, las trayectorias que han seguido, la cantidad de regiones que han afectado entre otros aspectos. De esta manera se podrían aplicar un grupo de medidas para mitigar los efectos de los ciclones en un área cuando aún estén lejanos a esta. De igual forma resulta de gran relevancia conocer que huracanes han ocasionado más daños según el punto por donde entraron, la trayectoria que siguieron, la presión entre otras características. De ser posible conocer todos estos aspectos antes de la entrada de un huracán al país sin lugar a dudas los efectos devastadores de los mismos serian considerablemente reducidos permitiendo al estado ahorrar gran cantidad de recursos y dinero.

Los SIG gozan de una amplia aceptación y popularidad tanto a nivel nacional como internacional. La Universidad de la Ciencias Informáticas (UCI) cuenta con un centro de desarrollo de software denominado, Geo-informática y Señales Digitales (GEYSED) perteneciente a la Facultad 6, en el cual se han desarrollado exitosamente varios de estos sistemas. Entre los proyectos productivos del centro GEYSED se encuentra GeneSIG cuyo objetivo fundamental es potenciar y mejorar la plataforma de igual

nombre mediante la inclusión de nuevas funcionalidades y componentes. De esta manera sería posible desarrollar sistemas más robustos con un alto valor económico y social.

Con las personalizaciones que se realizan sobre la plataforma GeneSIG, conocidas como Aplicativos, es posible realizar diferentes tipos de análisis espaciales. Incluso con las características que tiene hoy la plataforma se podría incluir análisis de carácter espacio-temporal, que es el tipo de análisis necesario para el estudio de los huracanes y de los objetos móviles en general.

El principal problema radica en que, si se necesita conceptualizar el análisis, llevándolo más al significado de la información por sobre la información espacial o espacio-temporal en sí, los aplicativos no satisfacen la necesidad. Se hace necesario realizar análisis para responder preguntas como:

- ¿Qué huracanes pasaron ***cerca*** de la Habana entre 1990 y 2000?
- ¿Cuántos huracanes ***intensos*** afectaron Camagüey ***antes*** de 1980?
- ¿Qué ***tormentas tropicales*** han *afectado* a Cuba?

Por otro lado, si fuese necesario interoperar entre diferentes sistemas de información geográfica, con diferentes fuentes de datos de huracanes o un mismo sistema usando disímiles bases de datos - ¿cómo se eliminarían las ambigüedades existentes entre los entes de información? - considerando que conceptualmente pudieran haberse definido datos similares utilizando nombres o identificadores distintos, aunque siempre referentes a huracanes. ¿Cómo responder a las preguntas anteriormente ejemplificadas utilizando orígenes diversos de información? Las ontologías juegan un papel fundamental en este campo, permitiendo conceptualizar dominios y permitir la descripción de la información a través de lo que representa, más allá de la manera en que se almacene. En el dominio geográfico y espacio-temporal sucede de igual manera, fundamentalmente cuando es necesario ubicar geográficamente un determinado elemento o ente, pero no a partir de coordenadas puramente geográficas, sino, a partir de elementos referenciales: **cerca de**, **moverse a**, **junto a**, entre otros. Además de ello, es posible, a partir de los metadatos y descripción de las bases de datos, integrar esquemas y mejorar la traducción de las consultas, acercándolas al lenguaje natural (Billen, y otros, 2011).

Los SIG personalizados sobre GeneSIG ofrecen grandes ventajas mediante el procesamiento de información geoespacial. Sin embargo según (Fuentenegra, 2012) urge la necesidad de encontrar una solución que permita la integración de datos geográficos de una manera mucho más abstracta en la que el conocimiento juegue un rol esencial. Esto permite que se explote con mayor efectividad la información semántica que se encuentra embebida en los datos almacenados. Por otro lado (Puebla, 2012) plantea que la incorporación de la gestión de información semántica asociada a los datos geográficos de

GeneSIG es una necesidad para aumentar la eficiencia en las operaciones de búsqueda y elevar la satisfacción de los usuarios de los sistemas que se desarrollarán con GeneSIG.

De igual forma cuando se desean analizar con GeneSIG entidades cuyo comportamiento no es solo espacial sino espacio-temporal como los ciclones tropicales se requiere de la utilización de funciones matemáticas bastante complejas. Debido a que la plataforma GeneSIG no incorpora el análisis semántico surgen determinadas limitaciones. Por ejemplo se deja de procesar información semántica asociada a las entidades que se desean analizar. Esto ocasiona que se realice un análisis incompleto de la información empobreciendo el criterio para la toma de decisiones

Por la **situación problemática** antes expuesta se propone realizar la investigación a partir del siguiente **problema científico** a resolver:

¿Cómo mejorar el análisis del impacto de huracanes sobre determinadas áreas o la población, en los Sistemas de Información Geográfica desarrollados en el Centro GEYSED?

A raíz del problema de investigación se delimita como **objetivo general**: Desarrollar un Sistema de Información Geográfica para el análisis semántico de la trayectoria de Huracanes en Cuba.

Se define como **objeto de estudio**, Sistemas de Información Geográfica.

Partiendo del **objetivo general** antes propuesto, el **campo de acción** de esta investigación es Sistema de Información Geográfica para el análisis semántico de la trayectoria de Objetos Móviles.

Para el desarrollo de la investigación se tienen en cuenta las siguientes **preguntas científicas**:

- ¿Cómo afecta al desarrollo de los SIG en el centro GEYSED la incapacidad de realizar análisis semántico?
- ¿Cómo influye la incorporación de ontologías sobre los SIG desarrollados en el centro GEYSED sobre los resultados del análisis?
- ¿Cómo mejora la interpretación de los resultados a partir del análisis semántico?

Para el cumplimiento de la investigación serán desarrolladas las siguientes **tareas**:

- Elaboración del marco teórico de la investigación a partir del estado del arte existente sobre el tema actual.
- Análisis detallado sobre los Sistemas de Información Geográfica y de los Sistemas de Información Geográfica Gobernados por Ontologías.

- Selección de las metodologías, herramientas de software a utilizar y lenguajes de programación para el desarrollo del sistema informático.
- Generación de los artefactos que corresponden a los flujos de trabajo definidos por la metodología seleccionada.
- Implementación y pruebas al sistema informático desarrollado.
- Validación de la propuesta realizada.

Como **métodos de investigación** científica se emplearon los siguientes:

Métodos teóricos:

Análisis-histórico lógico: Este método se utiliza para realizar un análisis del estado del arte relacionado con el desarrollo de los SIGGOs, sus principales conceptos, ventajas y desventajas, elementos que los componen y estructuración de la información de estos sistemas.

Analítico-sintético: Este método se emplea para analizar la información encontrada sobre los rasgos distintivos de los SIGGOs, para poder extraer elementos generales y específicos tratando de establecer una relación entre los diferentes criterios que puedan existir y tributen al desarrollo de los mismos.

La **estructura** de este trabajo de investigación consta de introducción, tres capítulos, conclusiones, recomendaciones, bibliografía y anexos:

Capítulo #1. Fundamentación teórica: Recoge aspectos concernientes al objeto de estudio, se hace una revisión de la bibliografía donde se analizan los principales conceptos sobre las ontologías y los SIGGOs, observando sus ventajas, limitaciones y los requisitos necesarios a tener en cuenta en su elaboración, además de valorar la situación actual.

Capítulo #2. Características y diseño del Sistema de Información Geográfica para el Análisis de la Trayectoria de Huracanes: Contiene información asociada al sistema, descripción de los conceptos del negocio representados mediante el modelo de dominio, especificación de los requisitos funcionales y no funcionales del software. Se centra además en el modelo del diseño, así como la evaluación de los resultados parciales obtenidos en la investigación.

Capítulo #3. Implementación y prueba del Sistema de Información Geográfica para el Análisis de la Trayectoria de Huracanes: Incluye los diagramas de despliegue y de componentes del flujo de trabajo de Implementación y del flujo de trabajo de Pruebas; los modelos de prueba con la descripción de los casos de prueba y la validación de la solución propuesta.

Capítulo #1. Fundamentación teórica

1.1 Introducción

En este capítulo se enuncia un grupo de conceptos básicos que sustentan la presente investigación, los cuales están encaminados a lograr una mejor comprensión de la misma. Se realiza un estudio de los componentes y funcionalidades tanto de los SIGs como los SIGGOs existentes, así como de las herramientas y tecnologías apropiadas para su desarrollo. También se muestran algunas de las principales ventajas y desventajas de ambos sistemas. De igual manera, se realiza un estudio de cómo poder manejar una ontología sobre la plataforma GeneSIG para logar integrar OntoMov a la solución propuesta.

1.2 Sistemas de Información Geográfica

“Un Sistema de Información Geográfica (SIG, para sus siglas en español o GIS, para sus siglas en inglés) consiste en información de naturaleza diversa sobre un determinado territorio, almacenada en un conjunto de bases de datos tanto gráficas como alfanuméricas, cuya relación con el territorio se realiza a través de un sistema de referencia geográfico y se gestiona a través de uno o varios programas informáticos específicos; el conjunto es soportado por un sistema de computadores y por un personal especializado.” (Fallas, 2011)

“Un SIG puede ser concebido como una especialización de un sistema de bases de datos, caracterizado por su capacidad de manejar datos geográficos, que están geo-referenciados y los cuales pueden ser visualizados como mapas.” (Braken, 1992)

“Un SIG es un intento más o menos logrado según los casos de constituir una visión esquemática de una realidad compleja.” (Bosque-Sendra, 1992)

Se puede definir SIG como una base de datos especializada que contiene objetos geométricos (Cebrian, 1994)

“Un conjunto de hardware, software y datos geográficos para capturar, manipular, analizar y mostrar información geográficamente referenciada.” (ESRI, 1995)

Debido a la gran diversidad de definiciones propuestas para los SIG y luego de estudiar varias de ellas, el autor de la presente investigación asume la propuesta por el National Center for Geographic Information and Analysis (NCGIA):“Un SIG es un sistema de hardware, software y procedimientos elaborados para facilitar la obtención, gestión, manipulación, análisis, modelado, representación y

salida de datos especialmente referenciados, para resolver problemas complejos de planificación y gestión" (NCGIA, 1990)

1.2.1 Funciones básicas

Un SIG funciona como una base de datos con información geográfica (datos alfanuméricos) que se encuentra asociada por un identificador común a los objetos gráficos de un mapa digital. De esta forma, señalando un objeto se conocen sus atributos e inversamente, así como preguntando por un registro de la base de datos se puede saber su localización en la cartografía.

La razón fundamental para utilizar un SIG es la gestión de información espacial. El sistema permite separar la información en diferentes capas temáticas y las almacena independientemente, permitiendo trabajar con ellas de manera rápida y sencilla, facilitando al profesional la posibilidad de relacionar la información existente a través de la topología de los objetos, con el fin de generar otra nueva que no podríamos obtener de otra forma.

Las principales cuestiones que puede resolver un SIG, ordenadas de menor a mayor complejidad, son:

- Localización: preguntar por las características de un lugar concreto.
- Condición: el cumplimiento o no de unas condiciones impuestas al sistema.
- Tendencia: comparación entre situaciones temporales o espaciales distintas de alguna característica.
- Rutas: cálculo de rutas óptimas entre dos o más puntos.
- Pautas: detección de pautas espaciales.
- Modelos: generación de modelos a partir de fenómenos o actuaciones simuladas.

1.2.2 Procedimientos matemáticos utilizados en los SIG para el análisis espacial

Cálculos geométricos en los SIG

Casi la totalidad de los análisis espaciales realizados por los SIG están sustentados por el uso de cálculos geométricos sencillos, los cuales son la base para la construcción de algoritmos más complejos. Esto se debe a que prácticamente todos los cálculos geométricos que se realizan en estos sistemas se basan en la posición y las relaciones topológicas entre objetos (Ramón, et al., 2010).

Cálculos geométricos según métricas Euclidianas

La idea de distancia es fundamental para todo análisis espacial, pues en casi todos los procedimientos se incluyen en este concepto. Además, de calcular la distancia entre dos puntos, puede calcularse entre geometrías. Es el caso entre dos rectas en el plano, es igual a la distancia entre un punto cualquiera sobre la primera hasta otro punto ortogonal a él en la segunda, siempre y cuando sean paralelas, en caso contrario la distancia es nula, pues siempre existirá un punto en que ambas se encuentran (Ramón, et al., 2010).

Cálculos geodésicos

Cuando los cálculos son referidos al plano y como tradicionalmente se han realizado en los SIG se pueden emplear los métodos euclidianos. Sin embargo, cuando se trabaja con un sistema de coordenadas diferente a las cartesianas estos resultados son completamente erróneos (Ramón, et al., 2010).

En (Pesquer, et al., 2003) se plantean tres algoritmos para solucionar estos problemas que pueden generar los cálculos euclidianos.

El Problema Inverso de la Geodesia, consiste en determinar el acimut y la longitud de la línea geodésica que separe dos puntos sobre el elipsoide. Existen múltiples métodos para su resolución, pero uno de los más empleados en la literatura científica es el método iterativo de Bessel, el cual se explica en (Zakatov, 1981).

El problema directo de la Geodesia, consiste en determinar un punto destino problema a partir de un punto origen, una distancia y un acimut conocidos. Para este también existen diversos métodos conocidos para su resolución siendo uno de los más utilizados la integración de Runge-Kutta de 4to orden, también explicado en (Zakatov, 1981).

El tercer algoritmo es referido al cálculo de áreas; donde se propone transformar las coordenadas de los vértices del polígono a analizar, a una proyección equivalente y se propone específicamente una Cilíndrica Equal-Área (Pesquer, 2003).

1.2.3 Componentes que lo conforman

La composición de los SIG está dividida en cinco elementos conceptuales: el hardware, el software, la información, el personal capacitado y los métodos de análisis o desarrollo (Ortiz, 2002).

- **Hardware:** este elemento requiere varios tipos de dispositivos, desde equipos computadores centralizados para el almacenamiento de los datos, hasta capturadores de información geográfica. Su organización ha de determinarse por un hardware específico y de buenas

prestaciones para cumplir con las necesidades de las aplicaciones asociadas al procesamiento y análisis.

- **Software:** los componentes principales desde el punto de vista del software son: los sistemas de manejo de BD, la interfaz gráfica de usuarios (GUI, por sus siglas en inglés) para el fácil acceso a las herramientas, las herramientas de captura y manejo de información geográfica. También las herramientas para el soporte de consultas, análisis y visualización de los datos.
- **Información:** la constituyen los llamados datos espaciales y alfanuméricos, que pueden obtenerse por recursos propios o a través de proveedores específicos (ficheros, bases de datos externas o servicios). Los datos que se representan en un SIG están determinados por un conjunto de capas de información que al combinarse entre sí crean situaciones espaciales. Estas capas se construyen de acuerdo a la proyección gráfica obtenida de forma vectorial o ráster. Las capas vectoriales las constituyen elementos como los puntos, líneas, polígonos y las ráster se determinan por conjuntos de celdas de información que corresponden a los píxeles de la pantalla (ver Figura 1).

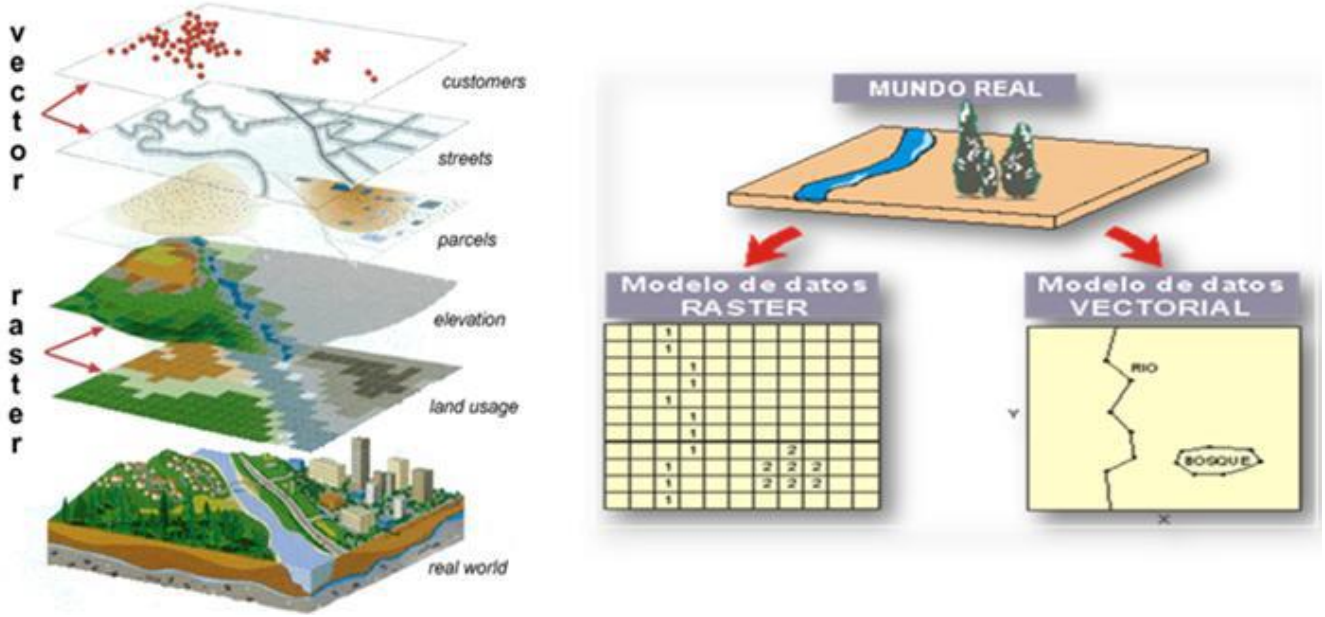


Figura 1. Forma en que se representa la información en un SIG (ORTIZ, 2002).

- **Personal capacitado:** las tecnologías SIG son de valor limitado si no se cuenta con especialistas de esa rama, son estos los que garantizan el manejo adecuado del sistema y ejecutan los planes de implementación del mismo. Sin personal experto en el desarrollo de

estas aplicaciones se corre el riesgo de procedimientos y análisis erróneos que afectaría en gran medida las actividades de toma de decisiones.

- **Métodos de análisis:** para todo SIG, el empleo de métodos de análisis sobre la información espacial y alfanumérica que maneja es primordial. El aprovechamiento de la combinación de los datos del entorno real y la representación espacial con el nivel de cálculo y procedural de un sistema informático, posibilita la determinación de modelos y situaciones muy útiles para cualquier análisis que se requiera.

1.2.4 Características principales de un SIG

Un grupo de características sobre los SIG son importantes destacar y a continuación se mencionan:

- La capacidad de visualización de información geográfica compleja a través de mapas.
- La funcionalidad de los SIG como una BD sofisticada, en la que se mantiene y relaciona información espacial y temática.
- La diferencia con las BD convencionales radica en que toda la información contenida en un SIG está unida a entidades geográficamente localizadas. Por ello, en un SIG la posición de las entidades constituye el eje del almacenamiento, recuperación y análisis de los datos.
- Representan una tecnología de integración de información.
- Se han desarrollado a partir de innovaciones tecnológicas sobre campos especializados de la geografía y otras ciencias (tratamiento de imágenes, análisis fotogramétricos, cartografía automática, entre otras), para constituir un sistema único más potente que la suma de las partes.
- Permiten unificar la información en estructuras coherentes.
- Este carácter integrador y abierto, hace de los SIG un área de contacto entre variados tipos de aplicaciones informáticas, destinadas al manejo de información con propósitos y formas diversas; por ejemplo: programas estadísticos, gestores de bases de datos, programas gráficos, hojas de cálculo, procesadores de texto, entre otros.
- Los límites y diferencias entre los SIG se enfocan en, los programas de diseño asistido por computadoras, los de cartografía temática y los de tratamiento de imágenes que son especialmente difusos. Aunque sus diferencias estriban sobre todo en el modelo de datos y en las capacidades de análisis de información espacial.

1.3 Ontologías

“Una ontología estará formada por una taxonomía relacional de conceptos, es decir, un conjunto de conceptos organizados jerárquicamente y por un conjunto de axiomas o reglas de inferencia mediante los cuales se podrá inferir nuevo conocimiento, es una forma de modelar sentencias que son siempre ciertas” (García, et al., 2004).

“Una ontología es un conjunto estructurado de términos que describen algún dominio o tema. La idea es que una ontología proporciona el esqueleto de una base de conocimientos” (Swartout, y otros, 1997).

Debido a la gran cantidad de definiciones que se han propuesto para las ontologías y luego de estudiar varias de estas, el autor de la presente investigación asume la definición dada por Tom Gruber que plantea: “Las ontologías son representaciones formales de un dominio o conceptualización que está generalizada o es de dominio abierto” (Gruber, 1993), la misma será la usada a partir de este momento durante toda la investigación.

1.3.1 Componentes básicos

Las ontologías según (Gruber, 1993) tienen un conjunto de componentes imprescindibles, entre los que se destacan:

- **Conceptos:** define lo que se puede traducir, y ofrece a los usuarios para escribir ontologías.
- **Relaciones:** representan las interacciones entre los conceptos del dominio.
- **Funciones:** son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de una ontología.
- **Instancias:** representan objetos determinados de un concepto.
- **Axiomas:** son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología.

1.3.2 Clasificación de las Ontologías

Según (Guarino, 1995) las ontologías se clasifican de acuerdo a su dependencia y relación con una tarea específica desde un punto de vista. A continuación se describen estas clasificaciones:

- Ontologías de Alto Nivel o Genéricas: describen conceptos más generales. En relación con los sistemas de información, estas ontologías describirían conceptos básicos. Por ejemplo, una teoría describiría partes y todos, así como sus relaciones con la topología.

- Ontologías de Dominio: describen un vocabulario relacionado con un dominio genérico. Por ejemplo, podría ser una descripción de datos y entidades relacionados con la sensorización remota con un ambiente urbano.
- Ontologías de Tareas o de Técnicas básicas: describen una tarea, actividad o artefacto. Por ejemplo, la evaluación de la contaminación sonora en ambientes urbanos o la descripción de características generales de componentes, procesos o funciones.
- Ontologías de Aplicación: Describen conceptos que dependen tanto de un dominio específico como de una tarea específica.

1.3.3 Ontología OntoMov

OntoMov: es una ontología para el dominio de objetos móviles, la cual fue desarrollada por el grupo Khaos de la Universidad de Málaga, España. La ontología incluye varios conceptos del dominio de objetos en movimiento, sobre todo los conceptos básicos asociados a los tres tipos de objetos en movimiento existentes: puntos móviles, líneas de movimiento y regiones en movimiento. Por otro lado, los conceptos de trayectoria asociada, también fueron considerados por su importancia en el análisis del movimiento.

Dicha ontología considera varias relaciones espaciales, temporales y espacio temporales entre conceptos. Los principales esfuerzos se centran en la representación de las propiedades dinámicas, es decir, las que cambian con el tiempo. En el análisis del movimiento, el cambio de posición es el elemento central, pero hay algunas otras propiedades dinámicas de importancia. La evolución de las propiedades dinámicas se puede representar utilizando piezas temporales (porciones), la creación de instancias de la clase TimeSlice y asociarlo a un intervalo válido. Ese intervalo válido, debe ser una instancia de la clase TimeInterval, en el caso de OntoMov se usa la clase de intervalo equivalente, importado de OWL (acrónimo del inglés, Web Ontology Language)-Time. El corte temporal debe estar asociado a su pertenencia de clase, utilizando la propiedad tsTimeSliceOf.

Con OntoMov se pueden representar las entidades que utilizan el movimiento de objetos, conceptos de dominio en movimiento. Permite la representación de varias entidades reales de diferentes ámbitos, como mover objetos y analizando su comportamiento por su semántica de movimiento. El primer conjunto de datos es de EE.UU NHC (Estados Unidos, Centro Nacional de Huracanes, por su traducción al español), base de datos de huracanes desde 1851 hasta el 2010, publicado en formato legible fácil. El uso de sintaxis abstracta de OWL describe cómo representar estos datos espacio temporales en OntoMov (Proenza, et al., 2012).

1.4 Sistemas de Información Geográfica Gobernados por Ontologías

"Un SIG que utiliza ontologías para generar nueva información a partir de un conjunto previo de datos es lo que se denomina SIGGO. En los SIGGO las ontologías son un componente más, como lo es la base de datos temáticos o espaciales que interviene y coopera de la misma manera para alcanzar los objetivos para los cuales fue creada" (Larin, y otros).

"Los SIGGO no son más que SIG en los que se incorporan el uso de ontologías como un componente activo. Los SIGGO son construidos utilizando clases derivadas de las ontologías y como consecuencias de esto se extrae el conocimiento embebido en estas para aportar mayor eficiencia y robustez al sistema" (Larin, y otros).

1.4.1 Fases que componen un SIGGO

Los SIGGO tienen dos fases que son fundamentales en su estructura (Garea-Llano, y otros, 2007):

- **Fase de generación del conocimiento:** comprende la especificación de las ontologías utilizando un editor de ontologías, la generación de nuevas ontologías a partir de las ya existentes y la traducción de ontologías a componentes de software.
- **Fase de uso del conocimiento:** se apoya en los productos obtenidos en la fase anterior: una serie de ontologías especificadas en un lenguaje formal y en una serie de clases. Las ontologías están disponibles para ser navegadas por el usuario final y para su utilización en la generación de aplicaciones, análisis y finalmente las posibles alternativas de decisión.

1.5 Tendencias y tecnologías

Debido a que la solución propuesta se integra a la plataforma soberana GeneSIG y la misma define en sus líneas bases el uso de un grupo de herramientas y tecnologías. Para el desarrollo de la solución informática se emplean estas herramientas y tecnologías, las cuales se enuncian a continuación:

1.5.1 Plataforma de Desarrollo

La Plataforma GeneSIG 2.0 es desarrollada por la Universidad de las Ciencias Informáticas (UCI), la empresa Geocuba y la Empresa de tecnologías de información para la Defensa (XETID). Esta aplicación SIG Web, es capaz de soportar una amplia gama de funcionalidades relacionadas con la gestión de datos espaciales, por parte de usuarios especializados y nuevos consumidores de servicios de datos geográficos. GeneSIG, tiene una dualidad funcional, que lo habilita para su utilización como una aplicación SIG, a través de la cual los usuarios pueden consumir, manipular y consultar bases de datos cartográficas digitales, de diversos formatos y orígenes, como por ejemplo:

datos vectoriales, datos ráster, datos GPS (global positioning system), bases de datos espaciales en formato postgis, Servicios de Mapas Web y Servicios de Geometrías.

De igual forma posibilita el desarrollo de proyecto SIG (personalizaciones) orientadas a solventar los requisitos de un negocio dado. Entre las ventajas que ofrece GeneSIG, se encuentra la portabilidad, pues permite que el cliente opere en varios Sistemas Operativos sin necesidad de modificar el código que se encuentra en el servidor.

1.5.2 Proceso de Desarrollo

En correspondencia con las líneas bases del proyecto "GeneSIG" se define ProDeSoft (Proceso de Desarrollo de Software) como proceso de desarrollo de software a emplear en la solución que se propone. ProDeSoft se basa en una combinación de varios modelos de procesos: el basado en componentes, el iterativo y el incremental.

Este proceso se encuentra disponible en su versión 1.5 y según su especificación (UCID, 2012), plantea que el ciclo de vida de un proyecto está compuesto por cinco fases: inicio, modelación, construcción, explotación experimental y despliegue (ver Figura 2).

Figura 2. Ciclo de vida de un proyecto, metodología ProDeSoft.

Durante la fase de **Inicio** se logra una visión preliminar de la problemática a resolver y se definen los recursos relevantes para la ejecución del proyecto. Es decir, se describen los objetivos y el alcance del proyecto, se identifican los involucrados y ejecutores (entidades involucradas), se estima de manera general las actividades a realizar durante todo el ciclo de desarrollo del proyecto (Cronograma General), se establece la estrategia a seguir para realizar la modelación del negocio y la captura de requisitos y de ser necesario se estiman los recursos materiales que deberán ser adquiridos.

En la fase de **Modelación** se capturan las partes esenciales del sistema, donde se identifican los procesos de negocio fundamentales y se aceptan los requisitos funcionales, obteniéndose la línea base de la arquitectura y una estrategia de construcción de la aplicación aprobada por los implicados en el proyecto. El hito fundamental de esta fase es la liberación de la arquitectura de sistema, datos y despliegue.

En la fase de **Construcción** se aclaran los requisitos restantes y se completa el desarrollo del sistema sobre una base estable de la arquitectura. Las fases anteriores sólo dieron una arquitectura básica que es aquí refinada de manera incremental conforme se construye el producto. En esta fase todas las características, componentes, y requisitos deben ser integrados, implementados, y probados en su totalidad, obteniendo una versión liberada del producto.

Durante la fase de **Explotación Experimental** se convierte la versión liberada del producto en una solución estable, donde se eliminan los errores que surgen durante las pruebas y se obtiene una certificación funcional y de seguridad del producto.

En la fase de **Despliegue** se instala y configura el sistema para un ambiente de producción real, se capacita al personal que usará la aplicación y se continúa dando soporte durante la explotación del sistema, culminando de ser preciso con transferencias tecnológicas.

1.5.3 Lenguaje de modelado

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) es utilizado para la especificación, la visualización, la construcción y la documentación de los artefactos de los sistemas de software y también para otros tipos de sistemas (Lovelle, 2005).

Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML permite modelar de manera conceptual procesos de negocio y funciones de sistema, también actividades específicas como escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. UML facilita un material de apoyo que le permita al lector definir diagramas propios, y a su vez, entender el modelado de diagramas ya existentes.

UML ofrece una serie de ventajas según (Meliá, 2008) las cuales se enuncian a continuación:

- Dispone de una semántica y sintaxis precisa, permitiendo conocer de forma no ambigua lo que el diagrama indica.
- Tiene una alta capacidad expresiva lo que posibilita representar todos los aspectos de la arquitectura web.

- Tiene capacidad para representar el sistema a diferentes niveles de abstracción, así cada uno de los miembros de desarrollo, arquitectos, diseñadores y analistas pueden enfocarse en los problemas que les ocupan olvidándose de los demás aspectos.
- Los modelos pueden ser intercambiables entre diferentes herramientas que tengan soporte de UML, así los modelos pueden ser reutilizados

En la solución propuesta se emplea para modelar los diagramas que se enuncian a continuación:

- Diagrama de Conceptos para representar el contexto del negocio.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Componentes para modelar la parte reemplazable y programable de un sistema informático.
- Diagramas de Despliegue para modelar la distribución física del sistema.

1.5.4 Herramientas de modelado

Visual Paradigm para UML 8.0 es una herramienta de modelado profesional que soporta el ciclo de vida completo del desarrollo de software orientado al desarrollo y la construcción de objetos, a las mejoras, la realización de comprobaciones y el despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad a un menor coste. Permite dibujar todos los tipos de diagramas de clases, generar código inverso, generar código desde diagramas y generar documentación.

A partir del estudio realizado se identificaron varias ventajas que ofrece Visual Paradigm:

- Entorno de creación de diagramas para UML 2.1.
- Uso de un lenguaje estándar común a todo el equipo de desarrollo que facilita la comunicación.
- Capacidades de ingeniería directa e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.
- Disponibilidad de integrarse en los principales Entornos de Desarrollo Integrado (IDE, por sus siglas en inglés).
- Disponibilidad en múltiples plataformas.

Visual Paradigm para UML en su versión 8.0 incluye siete nuevas mejoras:

- Importa diagramas de Microsoft Office Visio a Visual Paradigm.
- Soporta patrones de diseño.
- Soporta las tres formas del Diagrama Entidad-Relación: conceptual, lógica y física.
- Mejora la trazabilidad de elementos utilizando el historial de revisiones.
- Soporta líneas anidadas.
- Muestra como elemento estereotipado del modelo un ícono de imagen.
- Muestra y oculta elementos del diagrama según se desee.

Esta herramienta de Ingeniería de Software Asistida por Computadora (CASE, pos sus siglas en inglés) es una potente plataforma de desarrollo visual compatible con el ciclo completo de una aplicación. Además, combina el modelado con excelentes herramientas para la generación de códigos, ingeniería inversa y la interoperabilidad con otras aplicaciones (Paradigm, 2011).

1.5.5 Servidores web

Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario. El servidor web es el encargado de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo en correspondencia con los comandos solicitados (textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.). En este punto se considera necesario realizar la siguiente aclaración: mientras que comúnmente se utiliza la palabra servidor para referirse a una computadora con un software servidor instalado, en estricto rigor un servidor es el software que permite la realización de las funciones descritas.

El servidor HTTP Apache es un servidor web HTTP de código abierto para plataformas Unix (BSD, GNU/Linux, entre otros), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1. La arquitectura del servidor Apache es muy modular. El servidor consta de una sección Core y diversos módulos que aportan muchas de las funcionalidades que podrían considerarse básicas para un servidor web. Apache es usado principalmente para enviar páginas web estáticas y dinámicas en la World Wide Web. Muchas aplicaciones web están diseñadas asumiendo como ambiente de implantación a Apache o que utilizarán características propias de este servidor web.

1.5.6 Lenguaje de programación para aplicaciones web

Es un conjunto de símbolos, caracteres y reglas (programas) que les permiten a las personas comunicarse con la computadora. Los lenguajes de programación tienen un conjunto de instrucciones que permiten realizar operaciones de entrada/salida, cálculo, manipulación de textos, lógica/comparación y almacenamiento/recuperación. Entre ellos pueden señalarse Delphi, Visual Basic, Pascal, Java, entre otros.

Lenguajes del lado del cliente

Los lenguajes del lado del cliente basan su procesamiento en el cliente web, es decir, que se ejecutan en el navegador del usuario. En este sentido existen varios lenguajes, pero la presente investigación asumirá ExtJS para la solución que se propone, lo cual se justifica seguidamente:

ExtJS 3.0

Actualmente existen múltiples librerías de JavaScript que permiten realizar todo tipo de maravillas en el navegador web. ExtJS es una de estas librerías JavaScript de alto rendimiento que permite construir aplicaciones complejas haciendo uso de tecnologías como: AJAX, DHTML y DOM. Esta librería incluye:

- Componentes de interfaz de usuario de alto performance y personalizables.
- Modelo de componentes extensibles.
- Una API (Application Programming Interface) fácil de usar.
- Licencias de códigos abiertos y comerciales.

ExtJS permite crear Aplicaciones Ricas en Internet (RIA, por sus siglas en inglés) mediante JavaScript, lo que significa que estas se ejecutan en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, aumentando la interactividad, velocidad y usabilidad en las aplicaciones (García, y otros, 2009).

Una de las grandes ventajas de utilizar ExtJS es que permite crear aplicaciones complejas utilizando componentes predefinidos, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno. El análisis de las funcionalidades de ExtJS como framework ha permitido comprobar las potencialidades de la misma para crear una interfaz de usuario funcional y compatible con la

plataforma GeneSIG. Se incorporan a este criterio los beneficios que permiten el uso de ExtJS y que a continuación se relacionan:

- **Relación entre Cliente-Servidor balanceado:** se distribuye la carga de procesamiento, permitiendo que el servidor pueda atender más clientes al mismo tiempo.
- **Eficiencia de la red:** disminuye el tráfico en la red pues las aplicaciones cuentan con la posibilidad de elegir qué datos desea trasmitir al servidor y viceversa (criterio este que puede variar con el uso de aplicaciones de pre-carga).
- **Comunicación asíncrona:** en este tipo de aplicación el motor de render puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.

Lenguajes del lado del servidor

Los lenguajes de lado del servidor son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Existen varios lenguajes, pero solamente se asumen Java y PHP para el desarrollo de la solución que se propone, los cuales se justifican a continuación:

PHP

Es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, y con extensa documentación. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red, y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de la PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores.

Está dotado de una extensa librería de funciones, lo que le permite realizar numerosos tipos de aplicaciones web. La librería de funciones cubre desde cálculos matemáticos complejos hasta tratamiento de conexiones de red, por citar algunos ejemplos. Algunas de las más importantes capacidades de PHP son:

- Compatibilidad con las bases de datos más comunes, como MySQL, mSQL, Oracle, Informix, ODBC y PostgresSQL.
- Incluye funciones para el envío de correo electrónico, cargar archivos, crear dinámicamente en el servidor imágenes en formato GIF, incluso animadas y una lista interminable de utilidades adicionales.

Lenguajes para implementar el servicio web

Java

Java fue diseñado por la compañía Sun Microsystems Inc, con el propósito de crear un lenguaje que pudiera funcionar en redes computacionales heterogéneas (redes de computadoras formadas por más de un tipo de computadora, ya sean PC, MAC's, estaciones de trabajo, entre otros), y que fuera independiente de la plataforma en la que se vaya a ejecutar. Esto significa que un programa de Java puede ejecutarse en cualquier máquina o plataforma. El lenguaje fue diseñado con las siguientes características en mente:

- **Simple**: elimina la complejidad de los lenguajes como "C++" y da paso al contexto de los lenguajes modernos orientados a objetos. La filosofía de programación orientada a objetos es diferente a la programación convencional.
- **Robusto**: el sistema de Java maneja la memoria de la computadora por sí solo. No hay que preocuparse por apuntadores, memoria que no se esté utilizando, entre otros. Java realiza todo esto sin necesidad de que uno se lo indique.
- **Portable:** como el código compilado de Java es interpretado, un programa compilado de Java puede ser utilizado por cualquier computadora que tenga implementado el intérprete de Java.
- **Independiente a la arquitectura:** al compilar un programa en Java, el código resultante es un tipo de código binario conocido como byte-code. Este código es interpretado por diferentes computadoras de igual manera, solamente hay que implementar un intérprete para cada plataforma. De esa manera Java logra ser un lenguaje que no depende de una arquitectura computacional definida.
- **Multithreaded:** un lenguaje que soporta múltiples threads, es un lenguaje que puede ejecutar diferentes líneas de código al mismo tiempo.
- **Dinámico:** Java no requiere que se compilen todas las clases de un programa para que este funcione. Si se realiza una modificación a una clase, Java se encarga de realizar un Dynamic Bynding o un Dynamic Loading para encontrar las clases.

Está diseñado para soportar aplicaciones que serán ejecutadas en los más variados entornos de red, estaciones de trabajo, sobre arquitecturas distintas y con sistemas operativos diversos. Para acomodar requisitos de ejecución tan variados, el compilador de Java genera byte-codes: un formato intermedio indiferente a la arquitectura, diseñado para transportar el código eficientemente a múltiples plataformas hardware y software.

Librería para manejar la ontología

OWLAPI es una programación de aplicaciones de alto nivel de interfaz (API) para trabajar con ontologías OWL, ha estado disponible desde 2003, y ha sufrido una serie de revisiones de diseño, en particular, el seguimiento de la evolución de OWL en sí. Tiene un diseño flexible que permite a terceros para proporcionar implementaciones alternativas para todos los componentes principales. OWLAPI está implementado en Java y está disponible como código abierto. Permite a los desarrolladores trabajar en un nivel apropiado de abstracciones, aislando desde los problemas potenciales relacionados con, serialización y el análisis de estructuras de datos.

Con el uso de OWLAPI una ontología es simplemente visto como un conjunto de axiomas y anotaciones. Los nombres y las jerarquías de las interfaces para las entidades, expresiones de clase y axiomas de la OWLAPI se corresponden estrechamente con la especificación estructural, relacionando la especificación de alto nivel. Soporta la carga de ontologías con una variedad de sintaxis. Sin embargo, ninguna de las interfaces de modelo en OWLAPI reflejan, o están sesgados a cualquier sintaxis concreta o modelo. Por ejemplo, a diferencia de otras API como Jena, la representación de expresiones de clases y axiomas no está en el nivel de RDF.

El modelo de OWLAPI proporciona acceso a una ontología a través de una serie de definiciones, interfaces y clases. Las interfaces de modelo son de sólo lectura, en la que no proporcionan funcionalidad explícita para el cambio de las estructuras de datos subyacentes El modelo proporciona una visión centrada en el axioma de ontologías OWL (McGuinness, 2004).

1.5.7 Gestores de bases de datos

El Instituto de Ingenieros en Electricidad, Electrónica y Computación (IEEE, por sus siglas en inglés) define como una BD como “una colección de datos interrelacionados almacenados conjuntamente en uno o más ficheros de computadora”.

Los Sistemas Gestores de Base de Datos (SGBD) son software muy específico, encargados de servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Estos sistemas están compuestos por un lenguaje de definición de datos, un lenguaje de manipulación de datos y un lenguaje de consultas.

Los SGBD tienen como finalidad, manejar de una forma clara, sencilla y ordenada un grupo de datos que consecutivamente se convertirán en información, para una buena manipulación de los mismos. Deben cumplir con una variedad de objetivos tales como: abstracción de información, independencia,

redundancia mínima, consistencia, seguridad, integridad, respaldo y recuperación, control de concurrencia y tiempo de respuesta (Zambrano, 2008).

PostgreSQL versión 8.4

Este es un Sistema de Gestión de Bases de Datos Relacionales Orientada a Objetos (ORDBMS, por sus siglas en inglés). Es una herramienta desarrollada por la Universidad de California, Berkeley. Se considera por sus características como uno de los gestores de bases de datos de licencia libre y código abierto más avanzado de la actualidad por contar con más de diez años de evolución. PostgreSQL utiliza el modelo cliente-servidor así como el multiproceso en vez del multihilo, lo que garantiza la estabilidad del sistema. Con este multiproceso en caso de fallos no se afecta el sistema del todo ya que solo se afecta dicho proceso y no los demás (PostgreSQL, 2013).

Algunas de sus características son:

- Creación de sistemas con gran robustez y alto nivel de escalabilidad.
- Soporte completo para claves foráneas, vistas, triggers y procedimientos.
- Permite herencia entre tablas.
- Maneja diversos tipos de datos (boolean, integer, char).
- Soporta objetos y volúmenes de datos de gran tamaño.

Entre las ventajas se encuentran:

- Es multiplataforma, puede utilizarse en sistemas operativos como Linux y Windows.
- Provee documentación muy bien organizada, detallada, pública y libre.
- Posee comunidades de desarrollo muy activas, en Cuba existe una comunidad de PostgreSQL.
- Soporte nativo para lenguajes muy utilizados como PHP.

PostGIS 1.5

“Con la finalidad de que la base de datos PostgreSQL soporte objetos geográficos se ha desarrollado el módulo PostGIS, convirtiéndola en una base de datos espacial que se puede utilizar en Sistemas de Información Geográfica” (Moreta, 2009).

Para el trabajo con esta librería las funciones más importantes son los Constructores, los Editores y las Salidas de geometrías, además de los Operadores, las Relaciones espaciales y medidas, las

Funciones geométricas de procesamiento y las de Manejo de geometrías. Esta última trabaja con alguna de las sentencias más empleadas como son:

- **AddGeometryColumn:** agrega una columna geométrica a una tabla ya existente.
- **DropGeometryColumn:** elimina una columna de una tabla de geometría espacial.
- **UpdateGeometrySRID:** actualiza el SRID (Identificadores Espaciales de Referencias, por sus siglas en inglés) de todas las características de una columna de geometría.

1.5.8 Consultas a la ontología

Para realizar las consultas a la ontología espacio-temporal OntoMov se emplea Sparql como lenguaje estandarizado para consulta de grafos.

Sparql: Es un lenguaje de consulta capaz de obtener información desde grafos RDF (Resource Description Framework). Es la propuesta de estándar del W3C. Proporciona facilidades para (Rodil-Garrido, 2006):

- Extraer información en forma de URIs, nodos blancos y literales.
- Extraer sub-grafos RDF.
- Construir nuevos grafos RDF basados en los grafos incluidos en la consulta.

Con un lenguaje de consulta RDF, como Sparql, los desarrolladores y usuarios finales se encontrarán con una gran cantidad de información (recursos reconocibles) teniendo la facilidad para representar y utilizar los resultados obtenidos de manera más eficiente, sustentándose así su utilidad en la necesidad de recuperar y organizar la información de diferentes fuentes.

Sparql tiene especificaciones que explican diferentes partes de su funcionalidad; en general, consiste en un lenguaje de consulta, un formato para las respuestas, y un medio para el transporte de consultas y respuestas (Valencia-Castilo).

1.5.9 Entornos de desarrollo integrados

Un IDE es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos. En la actualidad las herramientas que usualmente componen un IDE son las siguientes: un editor de texto, un compilador, un intérprete, herramientas para la automatización, un depurador, un sistema de ayuda para la construcción de interfaces gráficas de usuario y opcionalmente un sistema de control de versiones.

NetBeans IDE 7.4

Es una aplicación de código abierto diseñada para el desarrollo de aplicaciones fácilmente portables entre las distintas plataformas, haciendo uso de la tecnología Java. NetBeans IDE4 dispone de soporte para crear interfaces gráficas de forma visual, desarrollo de aplicaciones web, control de versiones, colaboración entre varias personas, creación de aplicaciones compatibles con teléfonos móviles, resaltado de sintaxis y por si fuera poco, sus funcionalidades son ampliables mediante la instalación de packs. Entre las características que proporciona NetBeans se encuentran:
Desarrollo de aplicaciones multiplataforma sobre: MacOS, Windows, Linux. Add-ons para desarrollo Móvil, desarrollo Web gráfico e integración con SOA. Crecimiento de plataforma por medio de plugins. Entre los plugins que existen se tienen los siguientes:

- Herramientas que incluyen el lenguaje Java utilizadas para la mejora de desarrollo de aplicaciones.
- Herramientas de modelado UML.
- Herramientas XML.

NetBeans fue elegido como IDE debido a que simplifica las tareas de programación, permitiendo centrarse en las particularidades de la aplicación a desarrollar y evitando la complejidad inherente a cualquier desarrollo sobre una plataforma gráfica. Presenta gran área de edición donde el desarrollador interactúa con el código fuente, panel que permite la gestión de los archivos del proyecto y panel de visualización de los mensajes de error, resultado de una compilación.

1.5.10 Servidor de mapas

MapServer 5.6 es un motor de procesamiento de datos geográficos escrito en C. Permite crear “mapas de imágenes geográficas”, es decir, mapas que pueden dirigir a los usuarios hacia el contenido. Actualmente MapServer es mantenido por un creciente número de desarrolladores de todo el mundo. Es apoyado por un grupo diverso de organizaciones que patrocinan las mejoras y el mantenimiento, además es administrado al interior de OSGeo (Open Source Geospatial). MapServer incluye algunas salidas cartográficas avanzadas, tales como:

- Ejecución de la aplicación y dibujo de elementos según la escala.
- Automatización de los elementos del mapa (barra de escala, mapa de referencia y leyenda).
- Soporte a los lenguajes de scripting y ambientes de desarrollo más populares entre los que se encuentran PHP y Java

- Soporte multiplataforma Linux, Windows, Mac OS X, Solaris.
- Soporte a un gran número de estándares del OGC: WMS (cliente/servidor), WFS no transaccional (cliente/servidor), WMC, WCS, SLD, GML, SOS, OM.
- Múltiples formatos de datos vectoriales y ráster (TIFF/GeoTIFF, EPPL7 y otros por medio de GDAL).
- Archivos shapefile de ESRI, PostGIS, ESRI ARCSDE, Oracle Spartial, MySQL.
- Soporte de proyecciones cartográficas.

1.6 Conclusiones parciales

El estudio de las características, componentes y funcionalidades de los Sistemas de Información Geográficas, así como el manejo de las Ontologías permitieron determinar un conjunto de herramientas y tecnologías necesarias para el desarrollo de la presente investigación, lo cual condujo a la selección de estos elementos de la manera más adecuada posible. De igual modo, se analizaron las principales ventajas y desventajas de estas tecnologías para lograr una correcta integración entre estas y proponer una solución viable. El análisis del estado actual permitió determinar las funciones que contribuyen al desarrollo de sistemas de este tipo. Además, se estudiaron las características de la plataforma soberana GeneSIG, lo cual permite que la solución propuesta sea compatible con la misma.

Capítulo #2. Características y diseño del sistema

2.1 Introducción

Para guiar el proceso de desarrollo de la solución informática es necesario comprender su contexto e identificar las condiciones o capacidades que la misma debe tener. En este capítulo se definen las características del negocio a través de los principales conceptos o entidades destacadas. Esto posibilita la definición de los requisitos funcionales y no funcionales, así como modelar la solución propuesta a través de las clases del diseño. El diseño está guiado mediante la utilización de patrones de asignación de responsabilidades y grupo de los cuatros que fundamentan la representación originada. También la estructuración, aplicación y relación de estos elementos se establece mediante la concepción, orientada a objetos y basada en componentes, garantizando una arquitectura de software estable y de un alto grado de robustez. Además, se representa el modelo de datos que será manejado por la aplicación.

2.2 Propuesta de solución

La solución propuesta es un SIG personalizado sobre la plataforma soberana GeneSIG el cual incorpora la ontología OntoMov. Esto permite incorporar al sistema y por consecuencia a GeneSIG un grupo de ventajas y funcionalidades de OntoMov, dotándolo de la capacidad de realizar análisis semántico. Como resultado se obtiene un SIG capaz de procesar información semántica y espacial de manera simultánea. Este sistema es capaz de realizar consultas a la ontología en leguaje Sparql, las cuales devuelven un conjunto de datos generados por OntoMov que son almacenados, procesados y visualizador por el SIG. Esto permite obtener la información de una manera sencilla, sin necesidad de implementar un método puramente espacial para procesar la información mediante el uso de funciones más complejas. De igual forma, permite generar nueva información mediante el uso de los razonadores semánticos para realizar inferencia, posibilitando analizar instancias cuya información asociada no está almacenada en la BD o está de manera incompleta. Si bien esta solución informática no cumple con la definición completa de SIGGO si constituye una parte de estos sistemas. Lo cual permite la integración con otros componentes que aporten los elementos que el sistema no tiene para ser un SIGGO en su totalidad.

2.3 Modelo de dominio

El modelado de dominio o conceptual (ver Figura 3) tiene como objetivo comprender y describir los conceptos principales dentro del contexto organizacional y operacional, lo cual permite definir los

procedimientos, requisitos y roles más significativos. Esto ayuda a usuarios y desarrolladores a tener un vocabulario común, contribuyendo con el proceso de desarrollo de la aplicación.

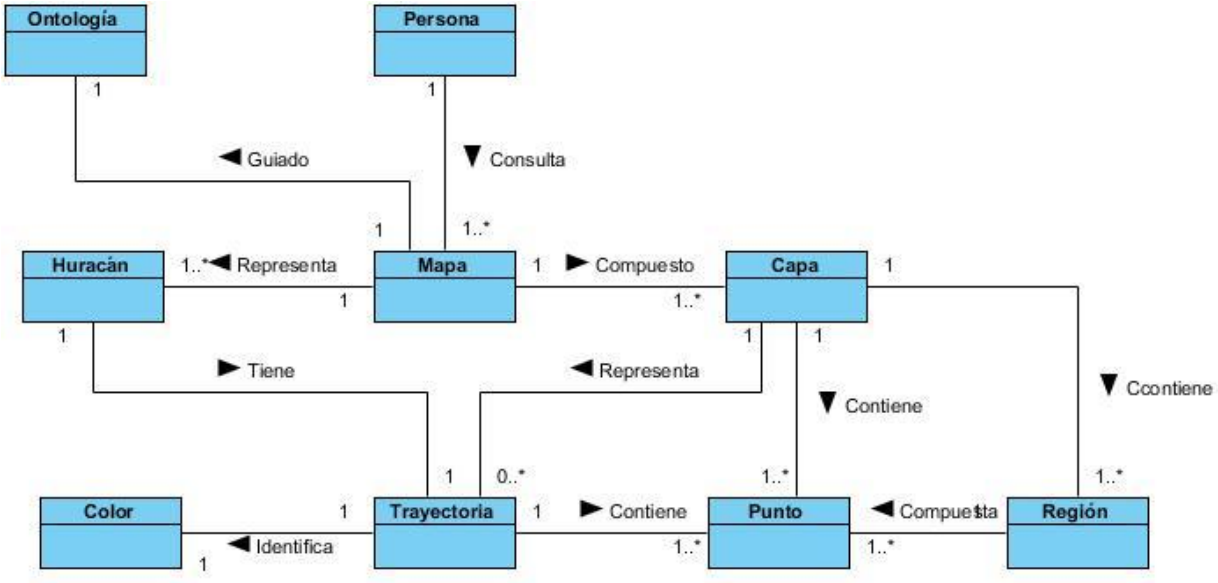


Figura 3. Modelo conceptual del sistema.

2.4 Descripción de los conceptos relacionados en el modelo de dominio

Concepto	Descripción
Persona	Es quien consulta la información representada sobre los mapas.
Mapa	Es la entidad que consulta por una persona, contiene las capas, se representan los huracanes y es guiado por una ontología.
Ontología	Se encarga de realizar el análisis semántico de cada una de las entidades.
Huracán	Son la entidad fundamental, se representan sobre los mapas y tienen una trayectoria asociada.
Capa	Está contenida dentro de los mapas, sobre esta se pueden representar trayectorias y contienen puntos y regiones.
Color	Es una característica de la trayectoria de los huracanes.
Trayectoria	Constituye una línea formada por cada uno de los puntos por donde pasó el huracán durante su traslación.
Punto	Es una localización sobre las capas la cual tiene condenadas en los ejes X y Y.

Región	Es un polígono que se representa sobre las capas y está compuesto por un conjunto de puntos que forman a una provincia y/o un municipio en el mapa.

Tabla 1. Descripción de los conceptos del dominio del negocio.

2.5 Requisitos del sistema

La Especificación de Requisitos de Software (ERS) es una descripción completa del comportamiento del sistema que se va a desarrollar. Un requisito funcional (RF) puede expresarse de dos formas: de alto nivel o de usuario y de bajo nivel o de sistema. Un requisito de alto y/o bajo nivel puede tener implícitos varios requisitos que responden al más general y por lo tanto, estos también tienen que ser especificados. Además, la ERS también contiene requisitos no funcionales o complementarios (RNF). Los no funcionales son requisitos que imponen restricciones en el diseño o la implementación (como por ejemplo restricciones en el diseño o estándares de calidad).

2.5.1 Requisitos Funcionales

Los requisitos funcionales definen los servicios que un sistema debe proveer, su comportamiento a las diferentes entradas y situaciones. Para la aplicación informática en cuestión se definen los siguientes RF por procesos:

Proceso 1. Realizar análisis sobre huracán.

RF 1. Seleccionar huracán.

RF 2. Representar trayectoria de huracán.

RF 3. Clasificar huracán.

Proceso 2. Realizar análisis sobre provincias.

RF 4. Seleccionar provincia.

RF 5. Representar provincia.

RF 6. Clasificar provincia.

RF 7. Representar huracanes cercanos a la provincia.

RF 8. Representar huracanes que afectaron la provincia.

Proceso 3. Realizar análisis sobre municipios.

RF 9. Seleccionar municipio.

RF 10. Representar municipio.

RF 11. Clasificar municipio.

RF 12. Representar huracanes cercanos al municipio.

RF 13. Representar huracanes que afectaron al municipio.

A continuación se muestran las especificaciones de los requisitos funcionales críticos de la solución informática propuesta:

- Especificación del **RF 1.** Seleccionar huracán:

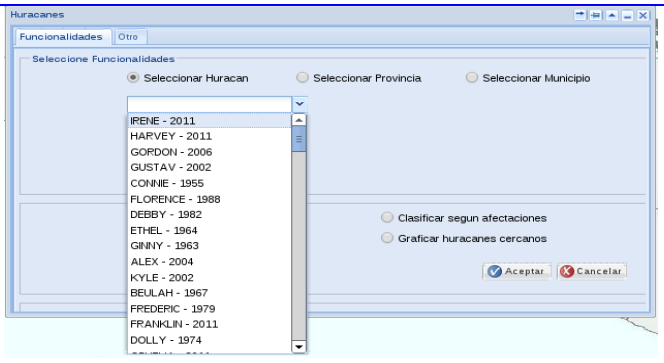
Conceptos tratados	**Conceptos**	**Atributos**
	Huracán.	-
Precondiciones	**Precondiciones**	**Pre-requisitos**
	Se debe haber creado un mapa.	No procede.
Descripción	1. Se selecciona un huracán dando clic sobre la lista desplegable “seleccionar huracán”. 2. Se escoge el nombre del huracán que se desea seleccionar. 3. Se hace clic sobre el botón aceptar.	
Validaciones	Solo se activará la lista desplegable “seleccionar huracán” si está seleccionada la opción “seleccionar huracán”.	
Post-condiciones	Quedará seleccionado el huracán que se desee analizar.	
Post-requisitos	Representar trayectoria de huracán, Clasificar huracán, Determinar huracanes cercanos.	
Prototipo de interfaz		

Tabla 2. Especificación del RF 1. Seleccionar huracán.

- Especificación del **RF 2.** Representar trayectoria de huracán:

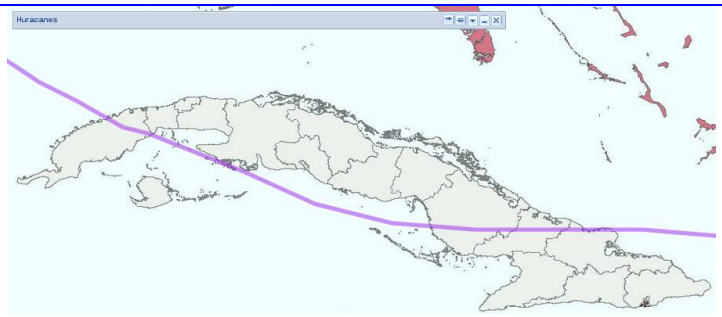
Conceptos tratados	Conceptos	Atributos
	Huracán, Trayectoria, Capa, Mapa.	-
Precondiciones	**Precondiciones**	**Pre-requisitos**
	Se debe haber seleccionado un huracán.	Seleccionar huracán.
Descripción	1. Se selecciona la opción representar trayectoria y se da clic sobre el botón aceptar. 2. Se crea una capa con la trayectoria del huracán llamando a la función “addVector.layer” especificando los datos y atributos requeridos. 3. Se adiciona al mapa mediante la función “addLayer”. 4. Se muestra la capa en el mapa.	
Validaciones	No se mostrará la trayectoria si no existe o si son incorrectas o no se adiciona la capa al mapa.	
Post-condiciones	Se adiciona la capa al mapa y se visualiza.	
Post-requisitos	No procede	
Prototipo de interfaz		

Tabla 3. Especificación del RF 2. Representar trayectoria de huracán.

- Especificación del **RF 3.** Clasificar huracán:

Conceptos tratados	Conceptos	Atributos
	Huracán, Capa, Mapa.	-
Precondiciones	**Precondiciones**	**Pre-requisitos**
	Se debe haber seleccionado un	Seleccionar huracán.

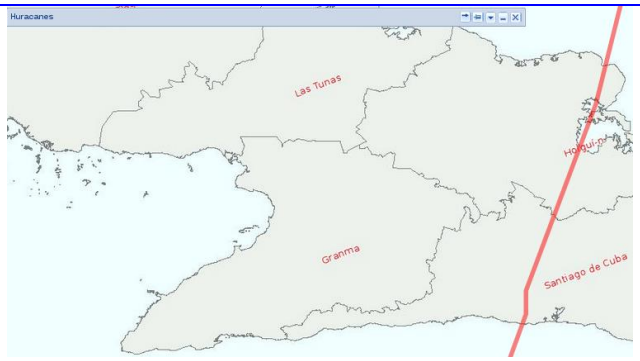
	huracán.	
Descripción	1. Se selecciona la opción clasificar huracán y se da clic sobre el botón aceptar. 2. Se consulta la ontología para obtener la clasificación del huracán seleccionado. 3. Se representa sobre el mapa la trayectoria del huracán seleccionado con el color rojo si la clasificación fue de muy devastador, de amarillo si fue de devastador y de verde si fue de poco devastador. Además, se muestra un mensaje indicando la clasificación del huracán. Ejemplo: "La clasificación del huracán fue de muy devastador". 4. Se muestra la capa en el mapa.	
Validaciones	No se mostrará la trayectoria si no existe o si son incorrectas o no se adiciona la capa al mapa.	
Post-condiciones	Se adiciona la capa al mapa y se visualiza.	
Post-requisitos	No procede.	
Prototipo de interfaz		

Tabla 4. Especificación del RF 3. Clasificar huracán.

- Especificación del **RF 4.** Seleccionar provincia:

Conceptos tratados	**Conceptos**	**Atributos**
	Región.	-
Precondiciones	**Precondiciones**	**Pre-requisitos**
	Se debe haber creado un mapa.	No procede.

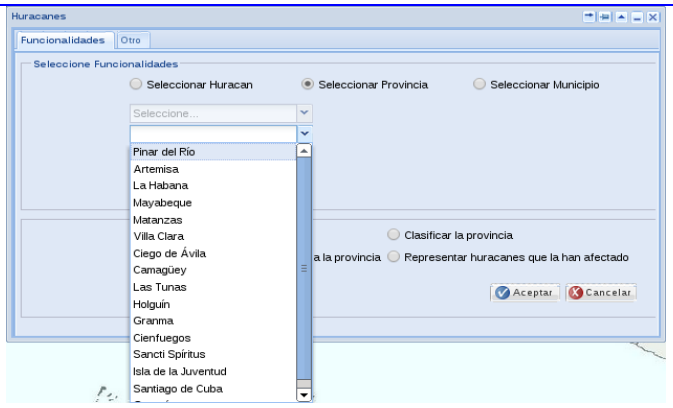
Descripción	1. Se selecciona una provincia dando clic sobre la lista desplegable “seleccionar provincia”. 2. Se escoge el nombre de la provincia que se desea seleccionar. 3. Se hace clic sobre el botón aceptar.
Validaciones	Solo se activará la lista desplegable “seleccionar provincia” si está seleccionada la opción “seleccionar provincia”.
Post-condiciones	Quedará seleccionado el huracán que se desee analizar.
Post-requisitos	Representar provincia, Clasificar provincia, Representar huracanes cercanos a la provincia, Representar huracanes que afectaron la provincia.
Prototipo de interfaz	

Tabla 5. Especificación del RF 4. Seleccionar provincia.

- Especificación del **RF 5.** Representar provincia:

Conceptos tratados	**Conceptos**	**Atributos**
	Región, Capa, Mapa.	-
Precondiciones	**Precondiciones**	**Pre-requisitos**
	Se debe haber seleccionado una provincia.	Seleccionar provincia.
Descripción	1. Se selecciona la opción representar provincia y se da clic sobre el botón aceptar. 2. Se crea una capa provincia llamando a la función	

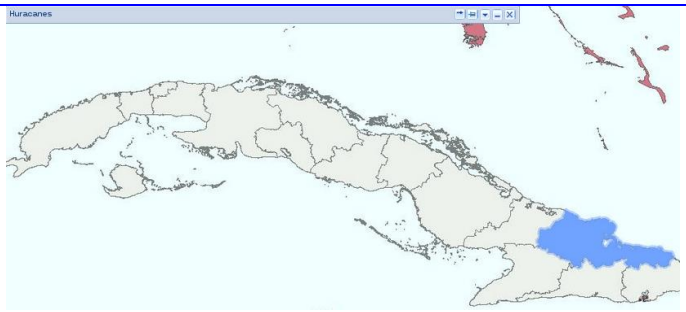
	"addVector.layer" especificando los datos y atributos requeridos. 3. Se adiciona a la capa la geometría mediante la función "addfeature". 4. Se muestra la capa en el mapa.
Validaciones	No se muestra la capa cuando la geometría no está contenida en la BD o la provincia no existe, o no se agrega la capa.
Post-condiciones	Se adiciona la capa al mapa y se visualiza.
Post-requisitos	No procede.
Prototipo de interfaz	

Tabla 6. Especificación del RF 5. Representar provincia.

- Especificación del **RF 6.** Clasificar provincia:

Conceptos tratados	**Conceptos**	**Atributos**
	Huracán, Región, Capa, Mapa.	-
Precondiciones	**Precondiciones**	**Pre-requisitos**
	Se debe haber seleccionado una provincia.	Seleccionar provincia.
Descripción	1. Se selecciona la opción clasificar provincia y se da clic sobre el botón aceptar. 2. Se consulta la ontología para obtener la clasificación de la provincia. 3. Se representa sobre el mapa la provincia seleccionada con el color rojo si la clasificación fue de muy afectada, de amarillo si fue de medianamente afectada y de verde si fue	

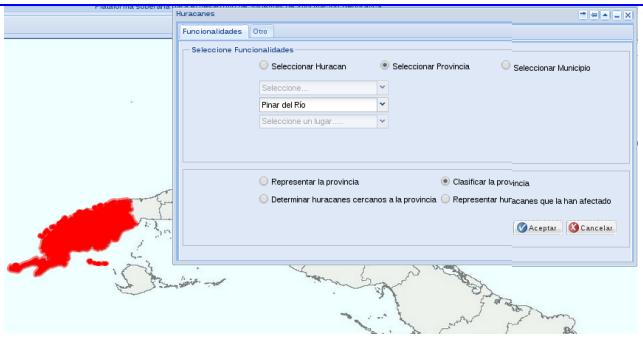
	de poco afectada. Además, se muestra un mensaje indicando la clasificación de la provincia. Ejemplo: "La clasificación de la provincia fue de muy afectada". 4. Se muestra la capa en el mapa.
Validaciones	No se muestra la capa cuando la provincia no existe o no se agrega la misma.
Post-condiciones	Se adiciona la capa al mapa y se visualiza.
Post-requisitos	No procede.
Prototipo de interfaz	

Tabla 7. Especificación del RF 6. Clasificar provincia.

- Especificación del **RF 7.** Representar huracanes cercanos a la provincia:

Conceptos tratados	**Conceptos**	**Atributos**
	Huracán, Región, Capa, Mapa.	-
Precondiciones	**Precondiciones**	**Pre-requisitos**
	Se debe haber seleccionado una provincia.	Seleccionar provincia.
Descripción	1. Se selecciona la opción representar huracanes cercanos a la provincia y se da clic sobre el botón aceptar. 2. Se consulta la ontología para obtener los huracanes que han pasado cerca de la provincia seleccionada. 3. Se agregan a la capa los huracanes que devolvió la ontología. 4. Se muestra la capa en el mapa.	

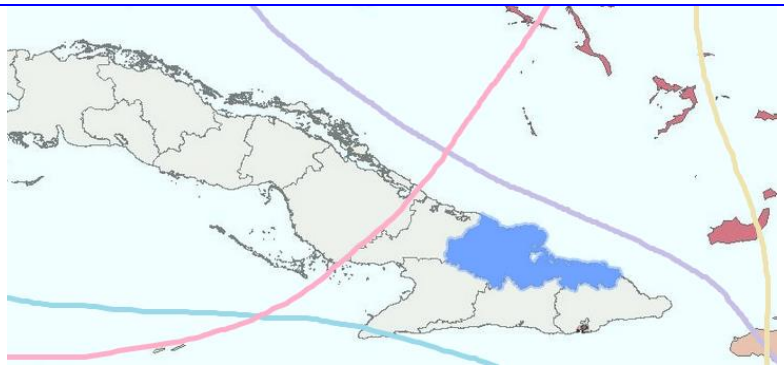
Validaciones	Solo se muestran los huracanes que pasaron cerca de la provincia dentro de un rango de 50km pero que no pasaron sobre esta.
Post-condiciones	Se adiciona la capa al mapa y se visualiza.
Post-requisitos	No procede.
Prototipo de interfaz	

Tabla 8. Especificación del RF 7. Representar huracanes cercanos a la provincia.

- Especificación del **RF 8.** Representar huracanes que afectaron la provincia:

Conceptos tratados	**Conceptos**	**Atributos**
	Huracán, Región, Capa, Mapa.	-
Precondiciones	**Precondiciones**	**Pre-requisitos**
	Se debe haber seleccionado una provincia.	Seleccionar provincia.
Descripción	1. Se selecciona la opción representar huracanes que afectaron la provincia y se da clic sobre el botón aceptar. 2. Se consulta la ontología para obtener los huracanes que han pasado sobre la provincia seleccionada. 3. Se agregan a la capa los huracanes que devolvió la ontología. 4. Se muestra la capa en el mapa.	
Validaciones	Solo se mostraran los huracanes que pasaron sobre la provincia seleccionada.	
Post-condiciones	Se adiciona la capa al mapa y se visualiza.	

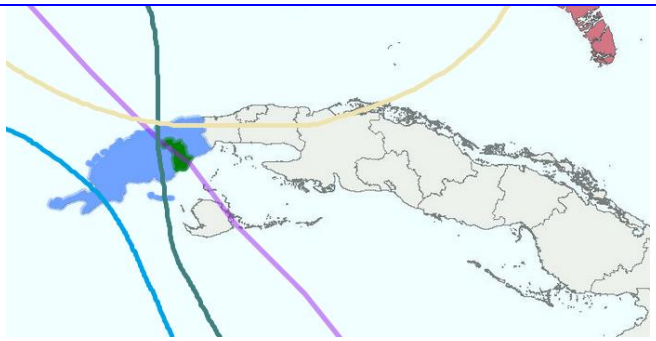
Post-requisitos	No procede.
Prototipo de interfaz	

Tabla 9. Especificación del RF 8. Representar huracanes que afectaron la provincia.

2.5.2 Requisitos No Funcionales

Los requisitos no funcionales son propiedades o cualidades que el producto debe tener. Son las características que hacen al producto atractivo, usable, rápido y confiable. Dichos requisitos resultan fundamentales en la evaluación de las características no funcionales del software como: seguridad y confiabilidad.

RNF 1. Interfaz gráfica o apariencia externa:

El sistema debe poseer una interfaz gráfica uniforme incluyendo pantallas, menús y opciones. Tanto los títulos de los componentes de la interfaz, como los mensajes para interactuar con los usuarios, deberán ser en idioma español y tener una apariencia uniforme guiada por los colores gris, blanco y azul claro. Los mensajes definidos deberán ser lo suficientemente informativos para dar a conocer la severidad de los mismos. Los colores de la trayectoria de los huracanes se podrán apreciar en correspondencia con la intensidad de su paso: rojo, naranja y verde. Las regiones identificadas como provincias se representarán en azul claro y los municipios de color verde.

RNF 2. Usabilidad:

El sistema debe permitir una navegación sencilla, tanto a los usuarios con conocimientos avanzados de informática como a los usuarios más inexpertos, esto se logrará a partir de una estructura de la información correcta.

RNF 3. Software:

Algunos de los requerimientos mínimos de software son los siguientes:

Para las PCs Clientes:

- Un navegador web.

- Sistema operativo: GNU/Linux y Windows.

Para las PCs Servidores:

- Sistema operativo GNU/Linux Debian 6.
- Servidor Web Apache 2.0 o superior, con módulo PHP 5 configurado con la extensión pgsql incluida.
- PostgreSQL como Sistema Gestor de Base de Datos.
- PostGIS como extensión de PostgreSQL como soporte de datos espaciales.
- MapServer 5.2.2 o superior, con extensión PHP MapScript.

RNF 4. Hardware:

Para las PCs Clientes:

- Se requiere tengan tarjeta de red.
- Al menos 1 GB de memoria RAM.
- Procesador 512 MHz como mínimo.

Para los Servidores:

- Se requiere tarjeta de red.
- El Servidor de Mapas debe tener como mínimo 2 GB de RAM.
- El Servidor de base de datos debe tener como mínimo 2 GB de RAM.
- Procesador 3 GHz como mínimo.

RNF 5. Restricciones de diseño e implementación:

- Plataforma GeneSIG.
- Lenguaje de programación: JAVA, JavaScript y PHP.
- Lenguaje de marcado: HTML.
- Como gestor de base de datos: PostgresSQL 8.4 y PostGIS.

2.6 Arquitectura del sistema

“La arquitectura es una vista estructural de alto nivel, que define estilo o combinación de estilos para una solución. Se puede decir que la arquitectura es esencial para éxito o fracaso de un proyecto.

Además, la arquitectura de un software es necesaria para comprender el sistema, organizar el desarrollo del mismo, fomentar la reutilización y controlar la evolución del proyecto" (Pressman, 2005).

La Plataforma GeneSIG emplea la arquitectura orientada a objeto y la arquitectura basada en componentes. A continuación se realiza una breve descripción de estas:

Arquitectura Orientada a Objetos: "nombres alternativos para este estilo han sido Arquitecturas Basadas en Objetos, Abstracción de Datos y Organización Orientada a Objetos. Los componentes del estilo se basan en principios orientados a objetos: encapsulamiento, herencia y polimorfismo. Las interfaces están separadas de las implementaciones. Las representaciones de los datos y las operaciones están encapsuladas en un tipo abstracto de datos u objeto. La comunicación entre los componentes es a través de mensajes" (Reynoso, y otros, 2004).

Arquitectura Basada en Componentes: "se centra en el diseño y construcción de sistemas computacionales que utilizan componentes de software reutilizables. Define la composición de software como "el proceso de construir aplicaciones mediante la interconexión de componentes de software a través de sus interfaces (de composición)", abogaba por la utilización de componentes prefabricados sin tener que desarrollarlos de nuevo" (Robaina, 2008).

2.6.1 Estilo Arquitectónico

Un estilo arquitectónico: "Expresa componentes y las relaciones entre estos, con las restricciones de su aplicación y la composición asociada, así como también las reglas para su construcción. Se consideran como un tipo particular de estructura fundamental para un sistema de software, conjuntamente con un método asociado que especifica cómo construirlo incluyendo información acerca de cuándo usar la arquitectura que describe, sus invariantes y especializaciones, así como las consecuencias de su aplicación" (Camacho, y otros, 2004).

Los estilos arquitectónicos son un tipo particular de estructura que definen los posibles patrones a usar en la implementación de la aplicación. Los estilos más conocidos son: Flujo de Datos, Centrado en datos y, Llamada y Retorno que es el que se emplea en este trabajo. Los componentes y las relaciones entre estos se realiza a través de este estilo que: "Permite construir una estructura de programa relativamente fácil de modificar y ajustar. Esta familia de estilos enfatiza la modificabilidad y la escalabilidad. Es el estilo más generalizado en sistemas de gran escala. Miembros de la familia son las arquitecturas de programa principal y subrutina, los sistemas basados en llamadas a procedimientos remotos, los sistemas orientados a objeto y los sistemas jerárquicos en capas" (Reynoso, y otros, 2004).

2.6.2 Patrones de arquitectura

Un patrón se define como una solución probada con éxito que aparece una y otra vez ante determinado tipo de problema en un contexto dado. Los patrones se definen por un nombre, un problema, una solución y las consecuencias de su aplicación. Este define una posible solución correcta para un problema de diseño dentro de un contexto dado, describiendo las cualidades invariantes de todas las soluciones (Pressman, 2006).

A continuación se realiza una breve descripción de los patrones arquitectónicos utilizados en la solución:

Patrón de arquitectura orientada a objetos: El patrón de arquitectura orientada a objetos define el sistema como un conjunto de objetos que cooperan entre sí en lugar de un conjunto de procedimientos. Los componentes del estilo se basan en principios orientados a objetos: encapsulamiento, herencia y polimorfismo. Son las unidades de modelado, diseño e implementación, y los objetos y sus interacciones son el centro de las incumbencias en el diseño de la arquitectura y en la estructura de la aplicación. Las interfaces están separadas de las implementaciones y en cuanto a las restricciones, puede admitirse o no que una interfaz pueda ser implementada por múltiples clases.

Patrón de arquitectura basado en componentes: Las características principales de este patrón son la modularidad, la reusabilidad y la compatibilidad. El mismo describe un acercamiento al diseño de sistemas como un conjunto de componentes que exponen interfaces bien definidas y que colaboran entre sí para resolver problemas. Los componentes son diseñados de forma que puedan ser reutilizados en distintos escenarios en disímiles aplicaciones, aunque algunos componentes son realizados para una tarea específica.

Luego de haber analizado la arquitectura de la plataforma GeneSIG y haber decidido mantener la misma para el sistema a desarrollar, se hace necesario entonces definir los patrones de diseño que se pueden emplear en la implementación de la solución.

2.7 Diseño del Sistema

Siguiendo las buenas prácticas de la metodología ProDeSoft, la disciplina de Diseño y Arquitectura se especifica detalladamente. Por tanto, a continuación se presentan algunos aspectos importantes que se tuvieron en cuenta asociados a esta disciplina:

2.7.1 Patrones de diseño

Los patrones de diseño representan un conjunto de estrategias y buenas prácticas que facilitan la creación de un software al describir un problema y su solución. En este trabajo se emplean los siguientes Patrones Generales de Asignación de Responsabilidades (GRASP, pos sus siglas en inglés):

Experto: se encarga de asignar una responsabilidad al experto en información, es decir, la clase que cuenta con la información necesaria para cumplir la responsabilidad. En la solución de la aplicación informática se puede evidenciar en la clase **huracanResult**, ya que es la entidad que contiene toda la información (parámetros inmediatos) concerniente a un huracán.

Creador: tiene la responsabilidad de identificar quién debe ser el responsable de la creación de nuevos objetos o clases. La nueva instancia deberá ser creada por la clase que tiene la información necesaria para realizar la creación del objeto, usa directamente las instancias creadas del objeto, o maneja varias instancias de la clase. El uso de este patrón se puede apreciar en el diseño de la solución cuando se le solicita información a la clase **ClientHuracan** y esta se encarga de crear instancias de la clase **huracanRequest**. De igual forma la clase **ServerHuracan** crea instancias de la clase **huracanResult**.

Alta Cohesión: se encarga de asignar a las clases responsables que trabajen sobre una misma área de la aplicación sin mucha complejidad. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. En el diseño de la solución existen las clases clientes y las clases servidoras, las cuales se encargan de realizar cada una su función, esto permite que las clases no estén saturadas y que se garantice a través de la asignación de responsabilidades, que el sistema presente alta cohesión en sus clases.

Bajo Acoplamiento: es el encargado de asignar las responsabilidades de forma que las clases se comuniquen con el menor número de clases que sea posible. En el modelo de diseño se muestra que no existen dependencias fuertes entre las clases, permitiendo así que al producirse una modificación en alguna de estas, se tenga la mínima repercusión posible en el resto de las clases y pueda ser reutilizado el código.

Los patrones de Grupo de los Cuatro (GoF, por sus siglas en inglés) resultan otra clasificación de los patrones de diseño que pueden ser utilizados en una solución informática. En la solución propuesta se pusieron en práctica los siguientes patrones GoF:

Singlenton: en el diseño de clases es necesario aplicar la solución de este patrón, que no es más que garantizar el acceso único a una clase mediante una única instancia. Por este medio se puede controlar el acceso a las clases. El patrón Singlenton se evidencia al modificar el framework CartoWeb, donde el objetivo del mismo es crear el objeto “mapa” para que no se cree cada vez que se hace un envío en la aplicación.

En el paquete OntoMov se encuentran las clases **Loader** y **Sparql** que se encargan de interactuar con las entidades de la ontología, también se encuentra la clase **MainControl** que se comporta como un servicio web. También se encuentra el programa principal **Main** que publica una interfaz SOAP en WSDL como instancia única (patrón Singlenton) de la clase servicio web **MainControl**.

Command: este patrón permite encapsular las peticiones a través de un objeto, lo que permite realizar operaciones como gestionar las acciones de dicho objeto. Se utiliza para la comunicación a través de las interfaces de usuario, específicamente a través de la clase **AJAXHelper** que es la encargada de comunicar las interfaces con el servidor. Uno de los aspectos más importantes en el sistema son las interfaces gráficas de usuario, ya que el usuario interactúa constantemente con ellas y por eso principalmente se aplica este patrón GoF a la solución.

2.7.2 Diagrama de Clases del diseño

Una clase del diseño es una abstracción de una clase real o construcción similar en la implementación del sistema. El lenguaje que se utiliza en dichas clases es el mismo que se emplea para la implementación del sistema. Se especifican los atributos y las operaciones. Además, se pueden realizar interfaces si tienen sentido para la programación y los métodos tienen correspondencia con las operaciones que fueron utilizadas en la implementación.
A continuación se presentan en las Figuras 4 y 5 los Diagramas de Clases del diseño de los Procesos 1 y 2, Realizar análisis sobre huracán y Realizar análisis sobre provincia respectivamente:

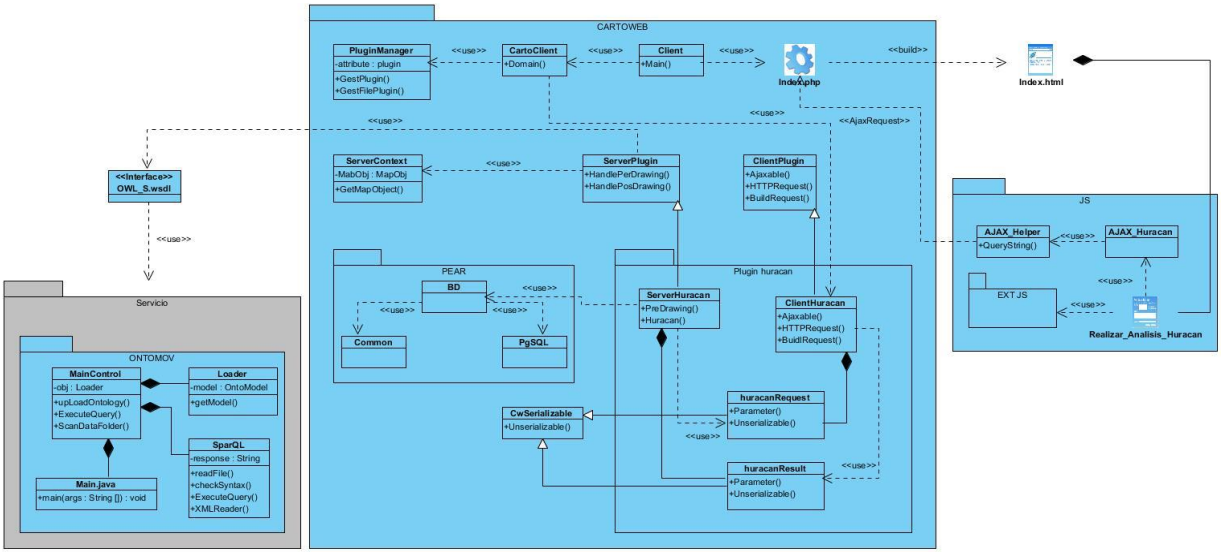


Figura 4. Diagrama de Clases del diseño del Proceso 1. Realizar análisis sobre huracán.

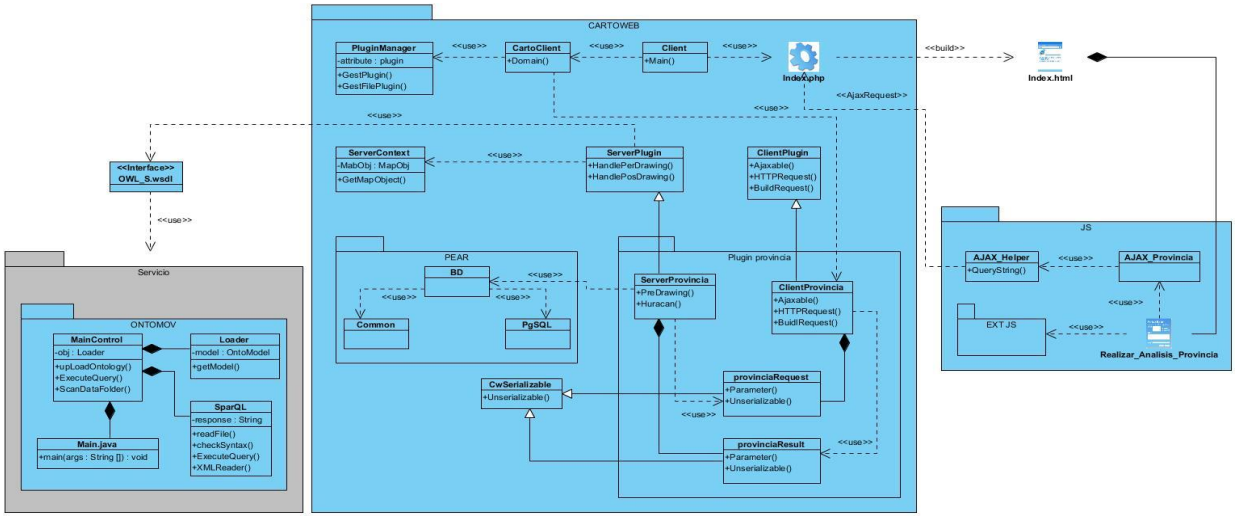


Figura 5. Diagrama de Clases del diseño del Proceso 2. Realizar análisis sobre provincia.

2.7.3 Descripción de las clases del diseño

Cada uno de los diagramas de Clases de diseño que emplea el paquete CartoWeb posee clases que son comunes, así como también las clases que son JavaScript y PHP. Para una mejor comprensión del propósito de cada clase del diseño de la solución, se describen las mismas del Proceso 1. Realizar análisis sobre huracán:

- **index.php:** tiene como propósito controlar la realización del RF en sí, recibe las peticiones realizadas por el cliente, gestiona las mismas y manda a construir la ClientPage.

- **Client:** contiene todos los archivos específicos de PHP del lado de CartoClient y permite la interacción entre la index.php y la CartoClient.
- **CartoClient:** integra y recoge todos los datos y funciones realizadas por cada una de las .js que intervienen en el RF y se definen una serie de variables globales que van a ser utilizadas por la aplicación.
- **PluginManager:** clase que se utiliza para gestionar la base de plugins.
- **ClientPlugin:** contiene las interfaces necesarias para los plugins del lado del cliente.
- **ServerPlugin:** esta clase proporciona la base de herramientas para el desarrollo de plugins.
- **ServerContex:** es la contenedora de la información común que ha de ser utilizada por la parte cliente y la servidora, empleando la información seleccionada como un objeto para un fácil manejo de los datos.
- **BD:** es la clase encargada de establecer la conexión con el servidor de base de datos para procesar los objetos a editar.
- **Common:** es la encargada de administrar las conexiones a la base de datos para ejecutar las consultas a la misma satisfactoriamente, esto incluye tratamiento de los datos.
- **Pgsql:** gestiona desde PHP las funciones de PostgreSQL.
- **CwSerializable:** se encarga de serializar todas aquellas clases que pueden ser serializadas, permitiendo la comunicación entre el Client y el Server del plugin.
- **AJAX_Helper:** tiene como propósito enviar las respuestas de los plugins “AJAX”, para alimentar a los plugins que responden a las peticiones del usuario.
- **Index.html:** es la encargada de mostrar en el mapa la región localizada.
- **AJAX_Huracan:** es la encargada de gestionar el pedido y respuesta a las peticiones del usuario por Ajax.
- **Realizar_Analisis_Huracan:** tiene como objetivo dado un huracán, conocer las trayectorias del mismo, la intensidad o clasificación que sugiere, la presión, cuán próximo a otros huracanes ha estado, el impacto socio-económico y las regiones que ha afectado.
- **ServerHuracan:** es la clase servidora que tiene como principal función la conexión con la base de datos y con el paquete que propicia el trabajo con la ontología para realizar las consultas requeridas y enviar las respuestas necesarias al **ClientHuracan**.

- **ClientHuracan:** se encarga de recoger y seleccionar de las .js contenidas en el paquete JS, toda la información correspondiente a los datos a precisar, entrados a través de los formularios, y los envía al **ServerHuracan**.
- **HuracanRequest:** es una clase común encargada de transportar los datos recogidos en **ClientHuracan** desde la interfaz y transportarlos a la clase **ServerHuracan**.
- **HuracanResult**: es una clase común encargada de transportar los datos generados en **ServerHuracan** a la clase **ClientHuracan**.
- **Loader** y **Sparql:** son las clases que se encargan de interactuar con las entidades de la ontología.
- **MainControl:** es la clase que se comporta como un servicio web y maneja todas las funcionalidades principales de la ontología.
- **Main:** es la clase que publica una interfaz SOAP en WSDL como instancia única de la clase servicio web **MainControl**.

2.8 Diseño de la Base de Datos

El diseño de la BD tiene como principal propósito asegurarse de que los datos persistentes son almacenados y consistentes. Además, se encarga de definir el comportamiento que debe ser implementado en la base de datos.

Para definir las clases persistentes se tuvieron en cuenta los conceptos identificados en el dominio del negocio que iban a persistir en el tiempo, pero esto no quiere decir que todos los definidos sean transformados en clases de este tipo. Demostrando esto que la persistencia es la capacidad de un objeto de mantener su valor en el espacio y en el tiempo.

A continuación se presenta el diagrama de clases persistentes modelado para la solución:

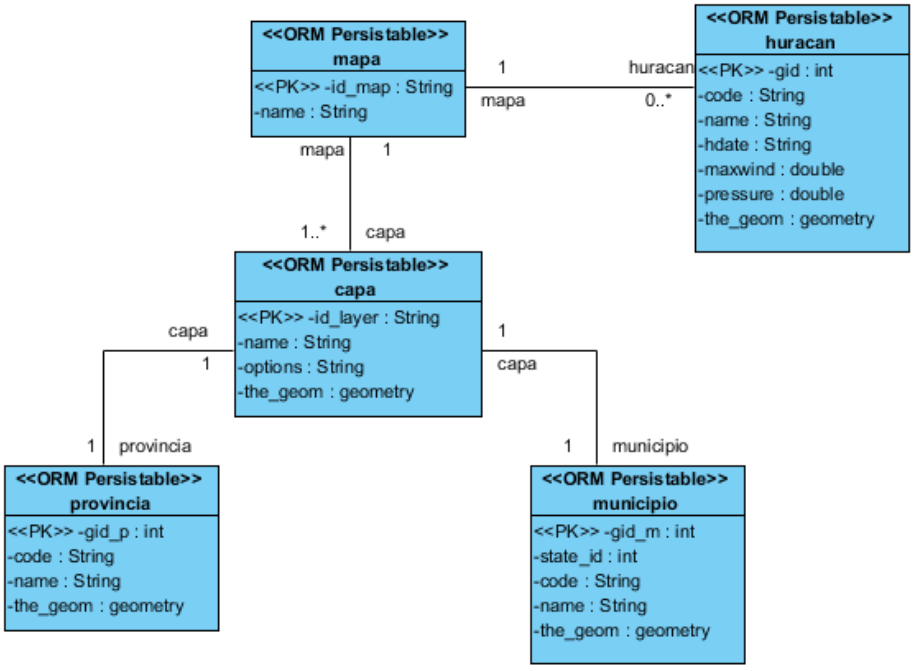


Figura 6. Diagrama de Clases Persistentes.

A partir del diagrama de clases persistentes anteriormente obtenido se generó el siguiente modelo físico de la BD:

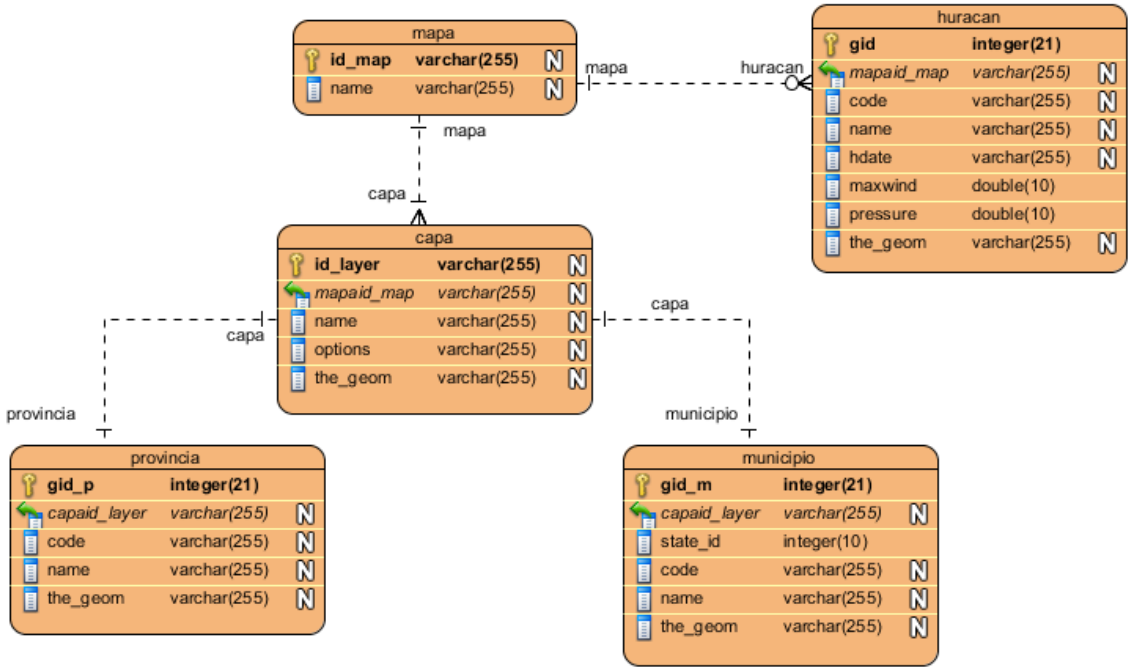


Figura 7. Modelo físico de la base de datos.

2.9 Conclusiones parciales

En este capítulo se realiza el análisis del negocio mediante conceptos utilizando el modelo de dominio como técnica de esta disciplina. Se especifican los requisitos funcionales y no funcionales para tener un mayor grado de detalle sobre las características del sistema. Se detallan las categorías de los requisitos del software posibilitando un mejor entendimiento sobre las características a implementar y una línea base arquitectónica robusta. Se realizó el diagrama de clases del diseño exponiendo los diferentes patrones de asignación de responsabilidades utilizados en la solución. También se diseñó el modelo de datos respondiendo a las exigencias y a las clases persistentes que posee la solución propuesta.

Capítulo #3. Implementación y prueba del sistema

3.1 Introducción

Una vez presentados los elementos referentes al diseño del sistema, en esta sección se tratará lo relacionado a la implementación y pruebas del mismo. Se hará énfasis, como en el capítulo anterior, a la representación de los componentes que forman la parte programable y reemplazable de la solución informática, específicamente del RF 3. Se presentará la distribución física de la solución propuesta para lograr una adecuada, estructurada y organizada ejecución de la misma. Además, se estarán generando las pruebas aplicadas a la solución informática mediante el método de prueba, caja negra, utilizando la técnica de partición equivalente, así como los resultados que demuestran el grado de satisfacción de la aplicación informática desarrollada.

3.2 Estilo de programación

Los estilos de programación definen la estructura y apariencia física del código, lo que facilita su comprensión, mantenimiento y lectura. En la implementación se utilizaron diferentes estilos que se describen a continuación:

3.2.1 Definición de clases

El nombre de las clases está escrito en inglés-español y su llave de apertura se encuentra en la próxima línea del nombre (ver Figura 8).

```php
<?php
class ClientHuracanes extends ClientPlugin implements GuiProvider, ServerCaller, Ajaxable
{
    //peticiones
    public $action;

    //respuestas
    public $data;
    public $msg;

    public function __construct()
    {
        parent::__construct();
    }

    protected function renderFormPrepare()
    {
    }
```

Figura 8. Definición de clases.

3.2.2 Estilo de métodos

Los métodos tendrán letra inicial minúscula, en caso de ser dos palabras se mantendrán unidas y la segunda comenzará con mayúscula; seguidamente se colocan dos puntos entre espacios y la palabra reservada **function**. En caso de pasarle elementos por parámetros, estos se escribirán con minúscula (ver Figura 9).

```
    /**
     * Funcion que activa el menu la opcion de comenzar a trabajar con la
     * herramienta
     *
     */
    executeActionButton: function () {
        var _this = AjaxPlugins.Huracanes;
        if (!_this.isPluginsLoader()) {
            loadDynamicConfig.executeLoadPlugin('huracanes', function () {
                _this.Controller.init();
                _this.isLoader = true;
            });
            if (AjaxPlugins.hasOwnProperty('huracanes'))
                loadDynamicConfig.executeLoadPlugin('huracanes');
        } else
            _this.Controller.init();

    },
    call:function (){
        Genesig.ajax.triggerLight('Huracanes.getNombresHurr', {});
    },

    isPluginsLoader: function () {
        return this.isLoader;
    }
};
```

Figura 9. Estilo de los métodos.

3.2.3 Declaración de variables

Las variables se escribirán con minúscula y estarán antecedidas de la palabra reservada **var**. El signo igual (=) estará ubicado entre espacios (ver Figura 10).

3.2.4 Estructuras de control

Dentro de las estructuras de control se pueden encontrar if, for, while, do while, entre otras. Se utilizaron, fundamentalmente, las condicionales, ***if***.

```
var _this = AjaxPlugins.Huracanes;
if (!_this.isPluginsLoader()) {
    loadDynamicConfig.executeLoadPlugin('huracanes', function () {
        _this.Controller.init();
        _this.isLoader = true;
    });
    if (AjaxPlugins.hasOwnProperty('huracanes'))
        loadDynamicConfig.executeLoadPlugin('huracanes');
} else
    _this.Controller.init();
```

Figura 10. Declaración de variables y estructura de control *if*.

3.3 Diagrama de componentes

Este tipo de diagrama representa la parte modular**,** desplegable y reemplazable de un sistema que encapsula implementación. También expone un conjunto de interfaces que funcionan como elementos intermediarios de comunicación entre componentes que posibilitan la independencia funcional.

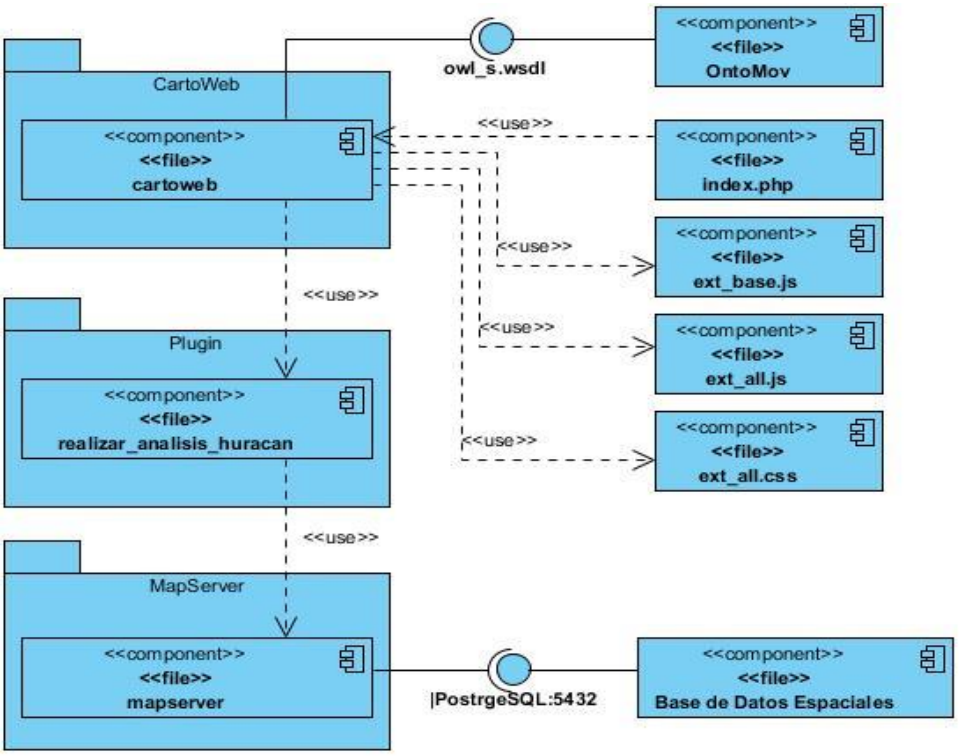


Figura 11. Diagrama de componentes del Proceso 1. Realizar análisis sobre huracán.

3.4 Diagrama de despliegue

Un diseño del despliegue del sistema permite establecer una correspondencia entre la arquitectura de software y hardware a utilizar, modelando la disposición física de los componentes que integran al sistema, también mostrando las conexiones físicas entre el hardware y los respectivos componentes. A continuación se presenta el diagrama de despliegue de la solución:

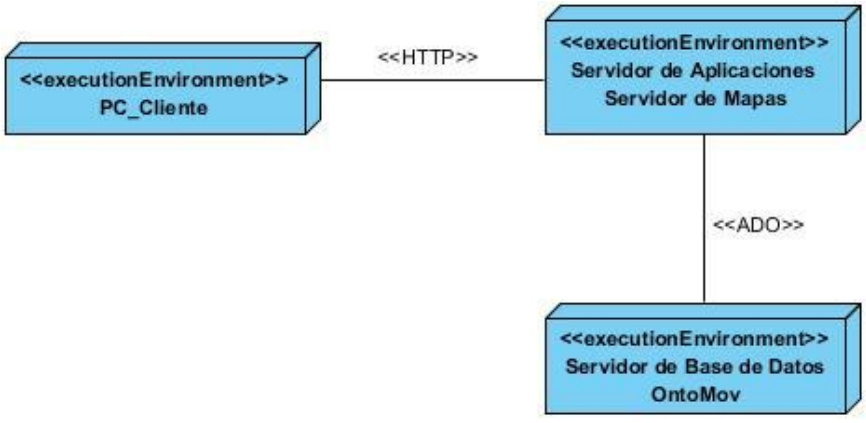


Figura 12. Diagrama de despliegue del sistema.

3.5 Pruebas

Las pruebas representan actividades claves que ayudan a entregar el producto con la calidad requerida para satisfacer las necesidades del cliente, cumplir con sus expectativas y con la certeza de que el producto cumple las especificaciones definidas.

3.5.1 Diseño de Casos de Prueba

Un diseño de caso de prueba (DCP) está compuesto por un conjunto de entradas, respuesta que emite el sistema de acuerdo a esas entradas y el flujo central que indica el camino del escenario descrito. Estos son desarrollados para verificar el cumplimiento total o parcial de un requisito. Las entradas representan las variables que se pueden especificar y las mismas contienen: V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante. A continuación se presenta en la Tabla 10, un ejemplo de los DCP realizados para comprobar el funcionamiento de la solución, específicamente para del Proceso 1. Realizar análisis sobre huracán:

- **DCP 1.** Realizar análisis sobre huracán:

Escenario	Descripción	Variables				Respuesta del sistema	Flujo Central
		Seleccionar huracán	Representar huracán	Nombre del huracán	Clasificación		
EC 1.1. Seleccionar huracán.	Seleccionar un huracán de la lista desplegable.	V Activar radiobutton	N/A -	V Flora	N/A -	Muestra una lista desplegable con los nombres de los huracanes almacenados en la base de datos.	Se selecciona el radiobutton “Seleccionar huracán”, acto seguido se escoge del comboBox el huracán deseado.
EC 1.2. Representar trayectoria de huracán.	Representar sobre el mapa la trayectoria de un huracán seleccionado.	V Activar radiobutton	V Activar radiobutton	V Flora	N/A -	Grafica sobre el mapa la trayectoria del huracán seleccionado.	Se selecciona el radiobutton “Representar trayectoria de huracán”, después el radiobutton “Seleccionar huracán”

							en caso de ser necesario, acto seguido se escoge del comboBox el huracán deseado y se presiona la opción del botón “Aceptar”.
EC 1.3. Clasificar huracán.	Representar la clasificación de los huracanes según su intensidad (rojo si la clasificación fue muy devastador, de naranja si fue devastador, de amarillo si fue medianamente devastador y de verde si fue de poco devastador).	V Activar radiobutton	N/A -	V Flora	V Activar radiobutton	Representa el huracán seleccionado del color correspondiente de acuerdo a la clasificación del mismo y muestra un mensaje indicando esta información, “El huracán es clasificado de muy devastador”, “El huracán es clasificado de devastador”, entre otros.	Se selecciona el radiobutton “Clasificar huracán según afectaciones”, luego el radiobutton “Seleccionar huracán” en caso de ser necesario, posteriormente se selecciona del comboBox el huracán deseado y se presiona la opción del botón “Aceptar”.
EC 1.4. Cancelar realización de análisis sobre huracán.	Anular cualquier acción que conlleve realizar algún análisis sobre un huracán descrito anteriormente.	N/A -	N/A -	N/A -	N/A -	Cierra la ventana emergente “Huracanes” y regresa al mapa base.	Se selecciona la opción del botón “Cancelar” de la ventana emergente “Huracanes”.

Tabla 10. Caso de prueba Realizar análisis sobre huracán.

No.	Variable	Valor Nulo	Descripción
1	Seleccionar huracán	No	Es un radiobutton que permite habilitar la lista desplegable o comboBox “Seleccione huracán” y el fieldsed “FuncionesHurr” que contiene las funcionalidades asociadas a los huracanes.
2	Representar huracán	No	Es un radiobutton que habilita la funcionalidad “Representar huracán”.
3	Nombre del huracán	No	Es un comboBox que contiene los nombres de los huracanes almacenados en la BD.
4	Clasificación	No	Es un radiobutton que habilita la funcionalidad “Clasificar huracán”.

Tabla 11. Variables del Caso de prueba Realizar análisis sobre huracán.

3.5.2 Resultados de las pruebas realizadas

Teniendo en cuenta las condiciones de la solución informática propuesta para la representación de huracanes, regiones sobre la plataforma GeneSIG y la consulta inmediata de información sobre estos elementos, se realizaron diversas pruebas de acuerdo a los niveles de aplicación asociados a esta disciplina, de las cuales se exponen sus resultados:

Las pruebas de caja negra también denominadas pruebas de comportamiento, se centran en verificar el cumplimiento de los requisitos funcionales del software, se emplean cuando se conoce la función específica para la que se diseñó la solución y se aplican a la interfaz del software, permitiendo obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa (Pressman, 2005), con el fin de encontrar la mayor cantidad de no conformidades existentes en el producto. Dentro de las técnicas empleadas por las pruebas de caja negra se utilizó, partición de equivalencia, que según define Pressman: “se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar” (Pressman, 2005).

Una prueba de unidad es la prueba enfocada a los elementos testeables más pequeños del software. La prueba de unidad siempre está orientada a caja blanca (Oré B., 2009). El diseño de casos de pruebas de

una unidad comienza una vez que se ha desarrollado, revisado y verificado en su sintaxis el código a nivel fuente. Es una forma de probar el correcto funcionamiento de un módulo o componente del código. Esto sirve para asegurar que cada elemento de estos funcione correctamente por separado. Los errores más frecuentes encontrados en este nivel de prueba están relacionados con las interfaces de comunicación entre componentes, interfaces entrada/salida, estructuras de datos locales, cálculos y flujos de control.

Para comprobar el funcionamiento independiente de cada funcionalidad se empleó el método de caja blanca, guiado por la técnica del camino básico (ver Anexos: 1, 2 y 3). Una vez determinados los caminos independientes de los requisitos funcionales importantes de la solución, se diseñaron casos de pruebas en función de dichos caminos, cuyos resultados están contemplados en la gráfica que se presenta en la Figura 13.

Otras de las pruebas aplicadas a la solución son las pruebas de integración. Una prueba de integración es ejecutada para asegurar que los componentes en el modelo de implementación operen correctamente cuando son combinados para ejecutar una funcionalidad global determinada (Oré B., 2009). Se prueba un paquete o un conjunto de paquetes del modelo de implementación. Los errores más comunes guardan relación con la comunicación a través de las interfaces, acumulación notable de errores de cálculo, acceso incoherente a estructuras de datos globales, tiempos de respuestas, entre otros.

Las pruebas de integración son la fase del testeo de software en la cual módulos individuales de software son combinados y testeados como un grupo. Estas pruebas se deben desarrollar luego de las unitarias y de forma incremental. Se conoce como integración incremental cuando el programa se construye y se prueba en pequeños segmentos en los que los errores son más fáciles de aislar y corregir. Estas pruebas se pueden plantear desde un punto de vista estructural o funcional. Las pruebas funcionales de integración son similares a las pruebas de caja negra. En la disciplina de pruebas aplicada a la solución se utilizó esta variante con el objetivo de encontrar fallos en la respuesta de un componente cuando su operación depende de los servicios prestados por otro(s) componente(s).

En un primer momento se aisló la funcionalidad “Clasificar huracán” correspondiente al Proceso 1. Realizar análisis sobre huracán que depende de los resultados que devuelve OntoMov al realizar una consulta Sparql, pues dicha ontología es la que contiene las relaciones posicionales entre un huracán y una región. Al realizar esta separación se pudo apreciar que la funcionalidad correspondiente no tenía sentido y que es indispensable mantener la dependencia. De igual manera se realizó un análisis similar a otras de las funcionalidades de la solución, tales como: “Clasificar provincia”, “Clasificar municipio”, “Representar huracanes que han afectado a una provincia”, entre otros, demostrando que las mismas son dependientes de OntoMov. También los resultados obtenidos con la aplicación de estas pruebas se encuentran reflejados en la gráfica que se presenta en la Figura 13.

Una vez diseñados los casos de prueba para cada requisito funcional, como se observa en el ejemplo de la Tabla 6, se procedió a la ejecución de las pruebas, teniendo en cuenta que la solución tiene características especiales, pues en la mayoría de los casos las entradas son eventos del ratón.

Los resultados se pueden observar en la Figura 13, donde se muestran la cantidad de casos de prueba usados y las no conformidades encontradas, las cuales fueron corregidas. Se realizaron tres iteraciones, luego de las cuales no se encontraron no conformidades, cumpliéndose correctamente los requisitos funcionales.

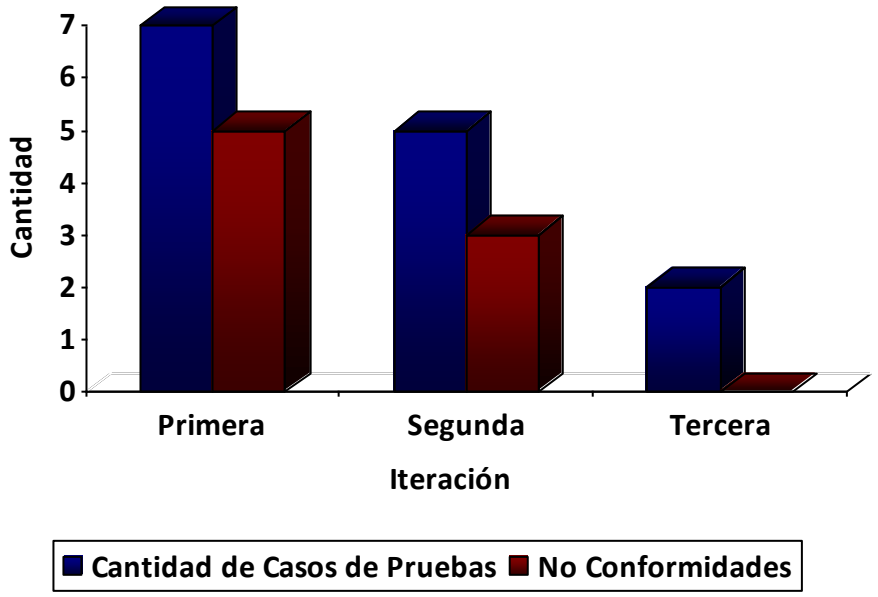


Figura 13. Pruebas de Caja Negra. Cantidad de Casos Prueba y No Conformidades por cada iteración.

3.6 Conclusiones parciales

Se explicó el estilo de programación utilizado, declaración de clases, variables y métodos, para lograr un mejor entendimiento del código resultante. Se realizaron los diagramas de despliegue y componentes para lograr una representación abstracta de la distribución física del sistema, así como de los elementos que encapsulan la implementación. Se diseñaron los casos de prueba y se validó la solución propuesta analizando los resultados generados en los DCP, lo que permitió obtener una solución informática viable para su utilización.

Conclusiones generales

Una vez culminada la investigación es posible afirmar que se dio cumplimiento al objetivo general trazado, por lo que se concluye que:

- A través del estudio del estado del arte se pudieron determinar los métodos, herramientas y tecnologías factibles para desarrollar un SIG personalizado sobre la plataforma GeneSIG capaz consulta una ontología del dominio de objetos móviles, al igual que la necesidad y factibilidad de desarrollar la investigación.
- La elección de Prodesoft como proceso de desarrollo de software permitió guiar y documentar el ciclo de vida de la aplicación, permitiendo que esta cumpliera con los estándares y estilos definidos en las líneas bases de GeneSIG.
- La utilización de las tecnologías y herramientas definidas en las líneas bases de GeneSIG permitió que la solución se integre a esta plataforma sin provocar incompatibilidades.
- La realización de las pruebas mediante la técnica de partición de equivalencia permitió detectar los posibles errores, para así dar cumplimiento de los requisitos funcionales y no funcionales establecidos y obtener un producto con la calidad adecuada.
- La integración de la ontología espacio-temporal OntoMov posibilitó incluir nuevas funcionalidades al Sistema de Información Geográfica para el Análisis Semántico de la Trayectoria de Huracanes.

Recomendaciones

Una vez culminada la investigación y teniendo en cuenta las experiencias obtenidas en el desarrollo de la misma, se proponen las siguientes recomendaciones:

- Extender el uso de OntoMov a otros Sistemas de Información Geográficas que se desarrollen en el proyecto “GeneSIG 2” del centro GEYSED.
- Incorporar validaciones a la solución informática propuesta mediante el uso de otras pruebas.
- Realizar análisis semánticos para otros tipos de objetos móviles aprovechando las características que propicia OntoMov.

Bibliografía referenciada

(INTECO), Instituto Nacional de Tecnologías de la Comunicación. 2009. *Ingeniería de software: Metodologías y ciclos de vida.* España : s.n., 2009.

(UCID), Unidad de Compatibilización Integración y Desarrollo. 2012. *Proceso de Desarrollo y Gestión de Proyectos de Software.* 2012.

Billen, Rolan, y otros. 2011. Ontologies in The Geographic Information Sector. [ed.] Gilles Falquet, y otros. *Ontologies in Urban Development Projects.* Advanced Information and Knowledge Processing Series. s.l.: Springer London, 2011, págs. 83 - 103.

Bosque-Sendra, J. 1992. *Sistemas de Información Geográfica.* Madrid : Rialp, 1992.

Braken, I. 1992. *Information technology in geography and planning.* Londres & New York : Routledge, 1992.

Camacho, E, Nuñez, F y Cardeso, G. 2004. *Arquitecturas de Software, Guías de Estudio.* 2004.

Cebrian, J. 1994. *GIS Concepts.* Cáceres : Departamento de Geografía y Ordenación del Territorio de la Universidad de Extremadura, 1994.

ESRI. 1995. ESRI (Enviromental Systems Research Institute). What is GIS? [En línea] 1995. http://www.gis.com/content/what-gis..

Fallas, Jorge. 2011. *SISTEMAS DE INFORMACIÓN GEOGRÁFICA. SISTEMAS DE INFORMACIÓN GEOGRÁFICA. SISTEMAS DE INFORMACIÓN GEOGRÁFICA. SISTEMAS DE SISTEMAS DE INFORMACIÓN GEOGRÁFICA.* 2011.

Fuentenegra, Adrián P., Águila, Adrián G. y Escalona Labrada, Kizzy Y. 2012. *Generación de consultas para la manipulación de Geo-ontologías desde la plataforma GeneSIG. 2012.*

García, R, Fernández, E y Rodríguez, D. 2009. Plataforma para Formación de Investigadores a Distancia. [En línea] 2009. http://laboratorios.fi.uba.ar/lsi/donofrio-uminsky-trabajoprofesional.pdf.

García, Ramón Martínez, Lerache, Jorge y M. Marcela, Bruno. 2004. *Ontología para el Aprendizaje y Compartición de Conocimientos entre Sistemas Autónomos.* 2004.

Garea-Llano, E y Gil-Rodríguez, J. 2007. *Los Sistemas de Información Geográfica Gobernados por Ontologías como Herramienta para la Interpretación Semántica de la Información Espacial y su integración a la IDERC.* La Habana : s.n., 2007.

González,, J. 2001. El Lenguaje de Modelado Unificado. [En línea] 2001. http://www.docirs.cl/uml.htm.

Gruber, Tomas. 1993. *Translation approach to portable ontologies.* s.l. : Knowledge acquisition, 1993.

Guarino, N. 1995. *Ontologies and knowledge bases: towards a terminological clarification.* Roma : Laboratory for Applied Ontology, 1995.

Larin, R y Garea, E. *INTEGRACIÓN SEMÁNTICA DE DATOS ESPACIALES CON SISTEMAS DE INFORMACIÓN GEOGRÁFICA.* La Habana : s.n.

Lovelle, S. 2005. *Perfil para representar una arquitectura de componentes en UML.* La Habana : CUJAE, 2005.

McGuinness, Deborah L. y Harmelen, Frank V. 2004. *OWL Web Ontology Language Overview. 2004.*

Meliá, S. 2008. Un Método de Desarrollo Dirigido por Modelos de Arquitectura para Aplicaciones Web. [En línea] 2008. http://www.dlsi.ua.es/~santi/papers/thesis_definitiva.PDF.

Moreta, O. 2009. *Diseño e implementación de Mapa Interactivo utilizando Web Mapping y Base de Datos Espacial.* Quito : s.n., 2009.

NCGIA. 1990. *National Center for Geographical Information and Analysis.* Santa Bárbara : Universidad de California, 1990.

Oré B., Alexander. 2009. *Quality Assurance & Software Testing.* [Online] 2009. [Cited: Febrero 04, 2014.]. http://www.calidadysoftware.com/testing/pruebas_unitarias1.php.

Ortiz, Gabriel. 2002. ¿Qué son los Sistemas de Información Geográfica? [En línea] 2002. http://www.gabrielortiz.com/art.asp?Info=012#Topolog%C3%ADas..

Paradigm, Visual. 2011. Visual Paradigm. [En línea] 2011. http://www.visual-paradigm.com.

Pesquer, Lluís Mayos, Pons, Xavier Fernández y Masó, Joan Pau. 2003. International Society for Photogrametry and Remote Sensing. [En línea] 2003. http://www.isprs.org/publications/related/semana_geomatica05/front/abstracts/Dimarts8/G08_abs.pdf.

PostgreSQL. 2013. PostgreSQL Sitio oficial del Servidor de Base de datos. [En línea] 2014. [Citado el: 12 de noviembre de 2014.] http://www.dirphp.com/dir/Software_y_servidores_PHP/PostgreSQL_Sitio_oficial_del_Servidor_de_Base_de_datos_51.html.

Pressman, R. 2005. *Ingeniería de Software.* 2005.

Pressman, R. 2006. *Ingeniería de Software. Un enfoque práctico. 5ta Edición.* 2006.

Proenza, Yuniel Eliades., Navas, Ismael. y Aldana, José F. 2012. *OntoMov: the ontology for moving objects.* 2012.

Puebla Martínez, Manuel E. y Águila, Adrián G. 2012. *Plataforma GENESIG: dando pasos hacia un entorno geosemántico.* 2012.

Ramón, Romanuel Antunez y Hernández, Lidisy Montero. 2010. *Algoritmo para el cálculo del área de polígonos curvos.* La Habana, Cuba : JCJ-ICIMAF, 2010.

Robaina, I. 2008. *Propuesta del Diseño Arquitectónico del Simulador de Sistemas Biológicos.* La Habana : s.n., 2008.

Rodil-Garrido, A. 2006. s.l. : Universitat Oberta de Catalunya, 2006.

Swartout, B, Patil, R y Knight, K. 1997. *Toward distributed use of large-scale ontologies.* s.l. : In AAAI-97 Spring Symposium Series on Ontological Engineering, 1997.

Valencia-Castillo, E. *Recuperación y organización de la información a través de RDF usando SPARQL*.

Zakatov, P. S. . 1981. Curso de Geodesia Superior. [En línea] 1981. [Citado el: 27 de Enero de 2015.] http://www.alipso.com/monografias4/Aplicaciones-de-la-Geodesia.

Zambrano, Raquel Ramírez. 2008. *Sistemas Gestores de Base de Datos.* ISSN 1988-6047.

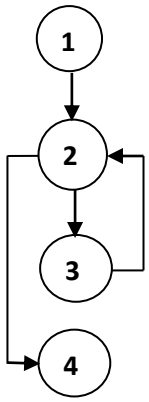
Anexos

Anexo 1: Salidas de las pruebas de caja blanca aplicadas a la función seleccionar huracán:

```
public function getHuracanes() {
1   Utils::checkDbError($this->db, 'No es posible conectarse a la base de datos.');
    $this->db->setFetchMode(DB_FETCHMODE_ASSOC); //devolver como array asociativo

1   $query = "SELECT DISTINCT code,hdate,name/*,gid,asewkt(transform(ST_SetSRID(hurr,the_geom, 4326),4326)) AS geom */ FROM hurr WHERE name !
    $result = $this->db->getAll($query);
    Utils::checkQueryError($result);

1   $arrayRecorrido = array();

2   foreach ($result as $row) {
3       $arrayRecorrido[] = array($row['name'].' - '.substr( $row['hdate'],0,4),$row['code']);
    }

4   echo json_encode($arrayRecorrido);
    die;
}
```

Figura 14. Método para seleccionar un huracán.

Grafo (G) de flujo resultante:



Complejidad Ciclomática:

- **V (G)** = R (cantidad de regiones) = 2.
- **V (G)** = A (cantidad de aristas) – N (cantidad de nodos) + 2 = 4 – 4 + 2 = 2.
- **V (G)** = P (cantidad de nodos predicados) + 1 = 1 + 1 = 2.

Caminos (C) independientes:

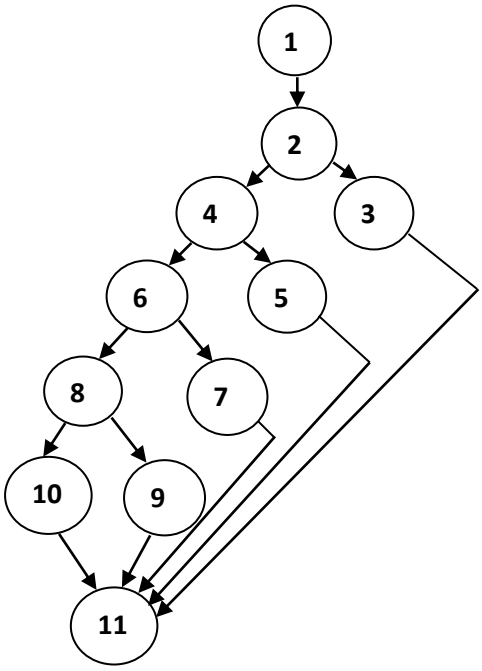
- C1 = 1, 2, 3, 2, 4.
- C2 = 1, 2, 4.

Anexo 2: Salidas de las pruebas de caja blanca aplicadas a la función clasificar huracán:

```
ClasifHur:{
    buildPostRequest:function (argObject){
1       var action = "action=" + argObject['actionName'] + "&data=" + argObject["data'
        return  action;
    },
1   onAfterAjaxCall: function (argObject, pluginOutput) {

1       var _this = AjaxPlugins.Huracanes;
        var data = _this.getontOlogyResult(hurr);
1       var intens = _this.getIntenc(hurr);

2       if((data>=3)&(intens>200)){
3           alert("El huracan es clasificado de muy devastador");
            _this.drawHurricaneClasif(hurr);
3           _this.ventana.collapse();

        }
4       else if((data>=3)&((intens>100)&(intens<200))){
5           alert("El huracan es clasificado de devastador");
            _this.drawHurricaneClasif(hurr);
5           _this.ventana.collapse();
        }
6       else if((data>=1)&((intens>100)&(intens<200))){
7           alert("El huracan es clasificado de medianamente devastador");
            _this.drawHurricaneClasif(hurr);
7           _this.ventana.collapse();
        }
8       else{
            alert("El huracan es clasificado de poco devastador");
9           _this.drawHurricaneClasif(hurr);
9           _this.ventana.collapse();
10      }
11  }
}
```

Figura 15. Método para clasificar un huracán.

Grafo (G) de flujo resultante:



Complejidad Ciclomática:

- **V (G)** = R (cantidad de regiones) = 5.
- **V (G)** = A (cantidad de aristas) – N (cantidad de nodos) + 2 = 14 – 11 + 2 = 5.
- **V (G)** = P (cantidad de nodos predicados) + 1 = 4 + 1 = 5.

Caminos (C) independientes:

- C1 = 1, 2, 3, 11.
- C2 = 1, 2, 4, 5, 11.
- C3 = 1, 2, 4, 5, 6, 7, 11.
- C4 = 1, 2, 4, 6, 8, 9, 11.
- C5 = 1, 2, 4, 6, 8, 10, 11.

Anexo 3: Casos de Pruebas de los caminos independientes críticos anteriormente identificados:

Proceso:
Realizar análisis sobre huracán.
Caso de Prueba:
Seleccionar huracán.
Camino independiente:
1, 2, 3, 2, 4.
Entradas:
Elegir el radiobutton “Seleccionar Huracán” y luego la lista desplegable o comboBox “Seleccionar huracán”.
Resultados esperados:
Se debe mostrar una lista desplegable con los nombres de los huracanes almacenados en la BD. Haciendo uso del foreach se van agregando al store de la lista todos los nombres de los huracanes, la fecha en que afectaron y el código de este.
Condiciones de ejecución:
Se debe haber seleccionado la opción del radiobutton “Seleccionar Huracán”.

Tabla 12. Caso de Prueba. Seleccionar huracán para el camino básico: 1, 2, 3, 2, 4.

Proceso:
Realizar análisis sobre huracán.
Caso de Prueba:
Clasificar huracán.
Camino independiente:
1, 2, 3, 11.
Entradas:
Elegir de la lista desplegable el nombre de un huracán. Se guarda en una variable el resultado del método “getontOlogyResult” el cual devuelve a partir de una consulta Sparql la cantidad de provincias que ha afectado el huracán seleccionado. Se guarda en una variable el resultado del método

"getIntenc" que devuelve mediante una consulta SQL la intensidad de los vientos del huracán seleccionado. Se comparan los resultados de las variables antes mencionadas con valores predefinidos.
Resultados esperados:
Se grafica la trayectoria del huracán seleccionado en color rojo y se muestra un mensaje notificando la clasificación del huracán como muy devastador.
Condiciones de ejecución:
Se debe haber seleccionado la opción del radiobutton "Seleccionar Huracán".

Tabla 13. Caso de Prueba. Clasificar huracán para el camino básico: 1, 2, 3, 11.