UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

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

Título: *"Módulo para la edición de una ontología del dominio geográfico en la plataforma GeneSIG."*

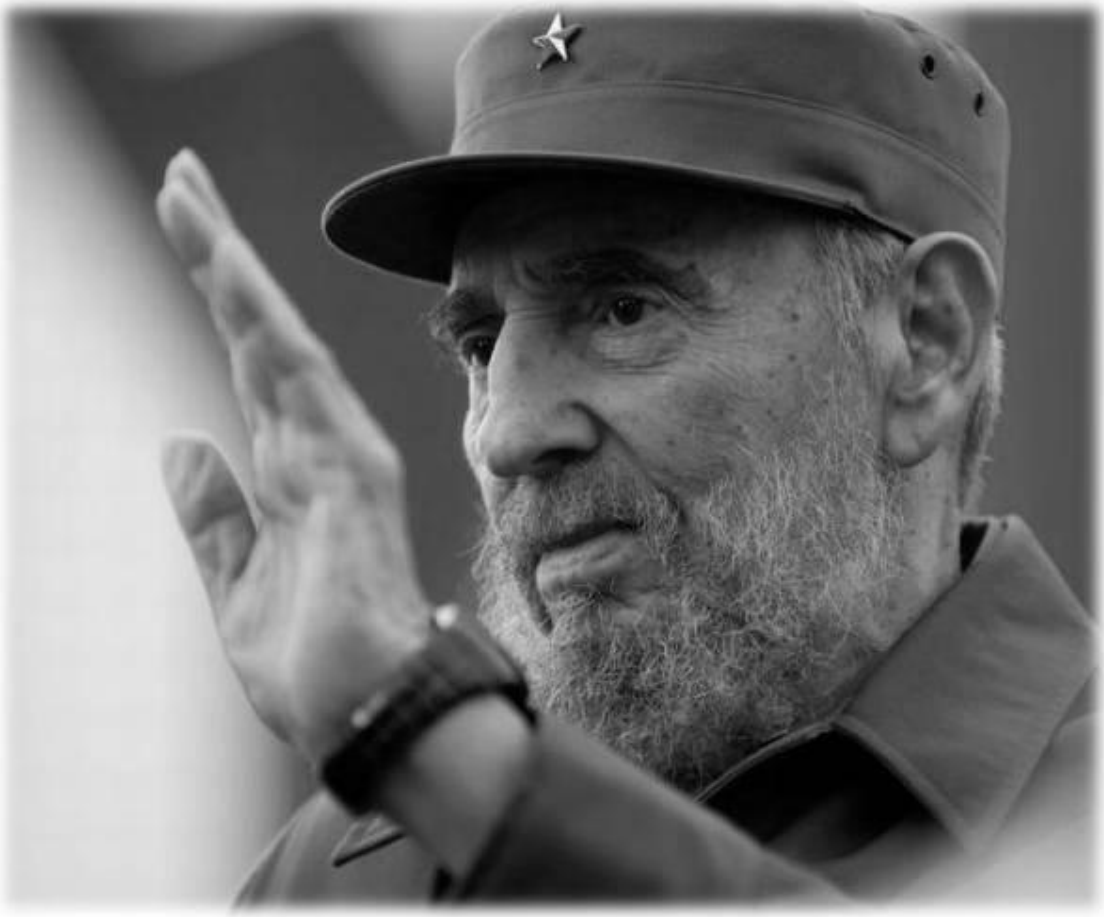
Autores: *Blanca Mayra Lastres Romero*

Luis Orlando Rico Blanco

Tutor: *Ing. Adrián Gracia Aguila*

La Habana, junio 2013

"Año 55 de la Revolución".



"Se pueden adquirir conocimientos y conciencia a lo largo de toda la vida, pero jamás en ninguna otra época de su existencia una persona volverá a tener la pureza y el desinterés con que, siendo joven, se enfrenta a la vida."

Fidel Castro Ruz

Declaración de Autoría

Declaro ser los autores 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 firmamos la presente a los ____ días del mes de _____ del año _____.

Blanca Mayra Lastres Romero

_____

Firma del Autor

Luis Orlando Rico Blanco

_____

Firma del Autor

Adrián Gracia Aguila

_____

Firma del Tutor

Datos de Contactos

Tutor: Ing. Adrián Gracia Águila.

Correo electrónico: agracia@uci.cu

Ingeniero en Ciencias Informáticas graduado en la Universidad de las Ciencias Informáticas (UCI). Actualmente se desempeña como profesor de la Universidad de las Ciencias Informáticas y Líder del proyecto Aplicativos SIG.

Agradecimientos.

Blanca

Quiero agradecer a todas aquellas personas que formaron parte de este sueño que hoy se hace realidad.

A mis padres Carlos y Obdulia por ser lo más grande que tengo, mi razón de ser, mi ejemplo a seguir y apoyarme en todo lo que me he propuesto hacer, los amo muchísimo. A mis hermanos Yury y Carli que son mis tesoros, por quererme tanto.

A Eva y Esmerido quienes también son mis padres, por siempre poder contar con ellos apoyarme en todo y no tener reparos en quererme como una hija.

A toda mi familia, mis tíos y primos por estar siempre presente cuando hemos necesitado su ayuda.

A mis amistades de la universidad que compartieron conmigo estos cinco años a mis compañeras de apartamento y en especial a Yeny y Arianna por todos los momentos que pasamos juntas y que nunca voy a olvidar. A Luis por ser un compañero de tesis sin igual, por confiar siempre en mí. A todos los profesores que me ayudaron y me orientaron a lo largo de la carrera y en especial durante la tesis. A Adrián mi tutor por apoyarnos en todo. A los miembros del tribunal y el oponente Alain por encaminarnos con sus consejos oportunos.

Agradecimientos

Luis

Primero agradecer a mis padres, especialmente a mi mamá por su apoyo incondicional en todos los momentos de mi vida, por todos sus sacrificios y que gracias a ella estoy hoy por convertirme en ingeniero. A mi viejo, que aunque no ha estado mucho tiempo conmigo en estos últimos años a sabido apoyarme y ayudarme en los momentos que lo he necesitado.

A mi abuela y mi tío por ser mis segundos padres y siempre darme todo su amor y cariño sin pedir nada a cambio.

A mi hermanita (que es la hermana más linda que tengo), por compartir su vida conmigo, y por sacrificarse todos estos años para que todos tengamos un futuro mejor.

A mi novia, por estar este ultimo año conmigo dándome apoyo y ayudándome en todo lo que ha podido sin pedirme nada a cambio, por darme todo su amor y su cariño y enseñarme que hay que luchar en la vida por las cosas que uno quiere.

A mi tutor por siempre estar ahí dispuesto a apoyarnos y brindarnos su ayuda sin importar las reiteradas veces que lo necesitamos. También a Alain que nos apoyó siempre y cada vez que necesitamos su ayuda estuvo ahí para nosotros.

A mi compañera de tesis Blanca , por confiar siempre en mí.

También quiero agradecer a mis amigos que me han apoyado en todo lo que he necesitado durante mi estancia aquí en la Universidad, a Robert, al Polito, al Tabito y a Marvel.

También quiero agradecer al piquete de la Redbull .

En general a todos mis compañeros de aula que hemos compartido los 5 años.

Agradecer a esta Revolución y al Comandante en Jefe Fidel Castro Ruz por darme la oportunidad de convertirme en un ser alguien útil a la sociedad.

Dedicatoria

Blanca

A mis padres Obdulia y Carlos por siempre confiar en mí y porque éste también es su sueño.

Luis

Dedico este trabajo a las seis personas más importantes del mundo: mi mamá, mi abuela, mi tío, mi hermana, mi papá y mi novia. Todo lo que soy y seré es gracias a ustedes, a su amor incondicional, a su confianza puesta en mí, a todo lo lindo que me han dado siempre. Ustedes me han permitido soñar, son lo mejor que tengo en la vida.

Resumen

En la Línea de Productos de Software (LPS) Aplicativos SIG, perteneciente al centro de desarrollo GEYSED, se desarrollan diferentes aplicaciones que tienen como objetivo gestionar información geográfica. En la misma, se trabaja con la plataforma GeneSIG la cual no permite la integración de datos geográficos en la que se explote con mayor efectividad la información semántica. Para ello, se hace necesario almacenar en un sistema de representación del conocimiento la información geográfica que esta maneja. Las ontologías son unos de los sistemas de representación del conocimiento más utilizado, por lo que con una ontología del dominio geográfico se podría transformar la información almacenada en conocimiento, pero para poder integrar las mismas con la Plataforma se hace necesario el uso de herramientas de tercero para su edición o realizar el proceso manualmente. Es por ello que la presente investigación tiene como objetivo realizar un módulo para la edición de ontologías del dominio geográfico, para ello se utiliza como lenguajes del lado del servidor Java y PHP, como servidor de mapa MapServer 5.6, Sistema Gestor de Base de Datos PostgreSQL 8.4, como Entorno de Desarrollo Integrado NetBeans 7.2.1 y servidor de aplicaciones Apache 2.2.

Palabras claves: información semántica, ontología, ontología de dominio, representación del conocimiento.

Índice de Contenido.

Índice de Contenido.

Índice de Tablas

Índice de Imágenes

Introducción

Con el transcurso de los años el Internet ha tenido avances significativos, dentro de los cuales se encuentra la *World Wide Web (WWW)* mayormente conocida como la red de redes. Esta surgió debido a la gran cantidad de información que transitaba por la red y la necesidad de lograr que los datos fueran accesibles y enlazables. La *web* fue creada alrededor de 1989 por Tim Berners-Lee con la ayuda de Robert Cailliau, desde ese entonces Berners-Lee ha guiado el desarrollo de estándares *web*.

En estos momentos la *web* es el medio de mayor difusión de intercambio personal. Uno de los principales problemas de la misma es la dificultad para encontrar la información. Esto se debe a que generalmente los buscadores actuales buscan palabras claves en las páginas, sin atender a la sinonimia o polisemia de estas, lo que provoca que la mayoría de las veces sea necesario consultar una gran cantidad de páginas hasta encontrar la información que se desea, es decir, es principalmente un problema de interoperabilidad semántica.

La *web* semántica surge con el propósito de superar las limitaciones actuales de la *web* tradicional. La misma se puede ver como un conjunto de tecnologías para la organización y representación del conocimiento digital que permitirá a las personas y a los ordenadores trabajar en conjunto. Según (Tim Berners-Lee; James Hendler; Ora Lassila, 2001) La *web* semántica no es una *web* separada, sino una extensión de la actual, en la cual la información se relaciona a un significado bien definido. Uno de los elementos fundamentales dentro de la arquitectura de la *web* semántica son las ontologías que se utilizan para representar el conocimiento.

Al igual que la WWW los Sistemas de Información Geográfica (SIG) han evolucionado, en el pasado, intercambiar la información geográfica era tan simple como enviar mapas de papel o datos, a través del correo. Con el transcurso de los años esta situación fue cambiando debido al incremento exponencial del volumen de datos geográficos que se generaban y la necesidad de que los mismos fueran compartidos, por ejemplo, un gran porciento de la información que maneja una empresa, o en especial una institución puede ser georeferenciada, es decir, se puede localizar en un lugar determinado por referencias geográficas.

Dada la necesidad planteada anteriormente comienzan a surgir los SIG los cuales según (Braken, 1992) pueden ser concebidos como una especialización de un sistema de bases de datos, caracterizado por su capacidad de manejar datos geográficos, que están georeferenciados y los cuales pueden ser visualizados como mapas. Hoy día los SIG han alcanzado un gran auge pues brindan grandes utilidades a la hora de utilizar los recursos naturales, ya sea, en las comunicaciones, en la salud, la ingeniería, en la industria y en la agricultura entre otros sectores. Por lo que esta tecnología ha sido acogida por las empresas, instituciones y universidades, por solo citar algunos ejemplos.

En Cuba estas herramientas han sido introducidas en algunas áreas, ejemplo en el sistema de salud para el cual se creó el SIGEpi realizado por Martínez Piedra el cual se utiliza para la ejecución de estudios epidemiológicos, también se cuenta con el SIG-ESAC, este es para la gestión de estadísticas de salud de Cuba. Se puede mencionar además la implementación de un SIG para la gestión y ordenamiento del Ecosistema Sabana-Camagüey, en la creación del mismo participaron la Dra. María A. García Cisneros, la Dra. Maribel Páez Moro y El Dr. Pedro Martínez Fernández.

La Universidad de las Ciencias Informática (UCI) no se ha quedado fuera cuando se trata de los SIG. Hace algún tiempo se vienen llevando a cabo diferentes proyectos como el SIGUCI, el cual tiene como objetivo la representación geoespacial de la información asociada a la universidad, el SIGRutas para el servicio de transporte obrero de la universidad y el SIGSalud para la gestión de la información georeferenciada en las instalaciones de la salud pública de Venezuela. Estos proyectos representan no solo una fuente de ingresos al país, sino que han contribuido a ganar prestigio a la Universidad.

El trabajo con los SIG se complica debido a que los términos que se manejan en ellos, por lo general, no son entendidos por los usuarios que en muchas ocasiones no son ingenieros, ni especialistas informáticos. Para facilitar esta situación aparecen las geo-ontologías la cuales según (Santos, y otros, 2009) una geo-ontología es una ontología que ofrece una descripción de entidades geográficas y difiere de otras ontologías por la presencia predominante de relaciones semánticas del contenido de la información geográfica de servicios *web,* con el fin de mejorar el descubrimiento y recuperación en la *web*.

Las geo-ontologías son estructuras que relaciona, modifica y asocia información de datos sobre otros datos. Con los grandes volúmenes de información que se estaban manipulando la comunidad científica no podía conformarse con el mero hecho de compartirla, sino que necesitaba lograr la interoperabilidad entre

los SIG, específicamente interoperabilidad semántica.

Según (Kashyap, y otros, 1996) el problema de la interoperabilidad semántica es la identificación de objetos semánticamente similares que pertenecen a bases de datos diferentes y la resolución de sus diferencias esquemáticas plantea que "las ontologías proveen una comprensión compartida y consensuada del conocimiento de un dominio, que puede ser comunicado entre personas y sistemas heterogéneos. Por ello, el desarrollo de ontologías, se presenta como el instrumento adecuado para alcanzar la integración semántica en el entorno de las infraestructuras de los datos espaciales, es así como se han concebido los SIG de nueva generación o Sistemas de Información Geográfica Gobernados por Ontologías (SIGGO), en los que las ontologías sean un componente activo dentro de su arquitectura".

En la UCI de conjunto con las Fuerzas Armadas Revolucionarias (FAR) y GeoCuba se desarrolla desde el año 2009 una plataforma tecnológica, sobre *software* libre, para el desarrollo de SIG en la *web*, denominada GeneSIG. Esta plataforma no permite la integración de datos geográficos en la que se explote con mayor efectividad la información semántica. Para ello, se hace necesario almacenar en un sistema de representación del conocimiento la información geográfica que esta maneja.

Las ontologías son una forma de representación del conocimiento, por lo que con una ontología del dominio geográfico se podría transformar la información almacenada en conocimiento, pero para poder integrar las mismas con la Plataforma se hace necesario el uso de herramientas de tercero para su edición o realizar el proceso manualmente, lo cual no permite que se obtenga, a través de GeneSIG, las coordenadas geográficas. Esto trae consigo que la información geográfica almacenada sea inexacta e inconsistente, además de que se puede encontrar en diferentes formatos en una misma ontología, es decir, que no se encuentre estandarizada, esto hace que no se exploten las bondades de GeneSIG.

Dada la problemática expuesta anteriormente surge como **problema a resolver** ¿Cómo favorecer la estandarización, precisión y consistencia de la información almacenada en una ontología del dominio geográfico desde la plataforma de GeneSIG?

Para darle solución al problema se tiene como **objetivo general** desarrollar un módulo para la edición de una ontología del dominio geográfico en la plataforma GeneSIG**.** El **objeto de estudio** de la investigación

estará centrado en la representación de la información en una ontología, con el **campo de acción** enfocado en la representación de la información en una ontología del dominio geográfico.

Para dar cumplimiento a los objetivos se plantean las siguientes **tareas de la investigación:**

1. Valorar los editores de ontologías existentes para la *web*.
2. Identificar los componentes de *software* a reutilizar en la propuesta de solución.
3. Fundamentar la metodología de desarrollo de software a usar en el proceso.
4. Modelar los requisitos funcionales y no funcionales de la propuesta de solución.
5. Diseñar la propuesta de solución.
6. Implementar la propuesta de solución.
7. Probar la propuesta de solución.

Como **posibles resultados** de la investigación se esperan:

- Un módulo para la edición de una ontología del dominio geográfico en la plataforma GeneSIG.
- La documentación técnica del módulo obtenido.
- El documento científico.

Como **idea a defender** si se desarrolla un módulo para la edición de una ontología del dominio geográfico en la plataforma GeneSIG, se favorecerá la estandarización, precisión y consistencia de la información almacenada en una ontología del dominio geográfico.

Los **Métodos Teóricos** permiten estudiar las características del objeto de investigación y facilitan la construcción de la hipótesis de investigación. Dentro de estos se encuentran presente en esta investigación:

- **Histórico-Lógico:** En el estudio de los editores de ontologías existentes en la actualidad.
- **Modelación:** En la realización de los artefactos generados por la metodología de desarrollo.

También se encuentran los **Métodos Empíricos** en los cuales el contenido procede de la experiencia, dentro de estos se encuentra:

- **Entrevista:** Se efectúa una entrevista informativa, no estructurada focalizada en el tema a investigar. Las cual se realizará a los trabajadores de la LPS Aplicativos SIG, para recopilar información referente a la plataforma de GeneSIG y para identificar los requisitos funcionales y no funcionales con que contará el módulo a construir para la misma.

Población: los 12 trabajadores de la LPS Aplicativos SIG.
Unidad de estudio: trabajador de la LPS Aplicativos SIG.
Muestra: 7 trabajadores que representan el 58%.
Técnica de muestreo: No probabilística.

Dentro de esta técnica el **Muestreo intencional:** debido a que el investigador escoge según su buen juicio a los integrantes de la muestra que son representativos o con probabilidad de dar mayor información.

Capítulo #1: Se realiza la fundamentación teórica y el estudio del arte relacionados con el tema a investigar. En el mismo son expuestos los principales conceptos como geo-ontología, SIG, interoperabilidad semántica y otros elementos para entender mejor el problema a resolver.

Capítulo #2: Se abordará sobre las herramientas a utilizar para el desarrollo de la aplicación y se seleccionará y especificará la metodología de desarrollo a utilizar para dar solución a la problemática planteada.

Capítulo #3: Se identificarán los requisitos tanto funcionales como no funcionales y se crearán los diagramas y artefactos correspondientes a dicha metodología.

Capítulo #4: Está dedicado a la implementación y realización de pruebas a la aplicación.

Capítulo I: Estudio del arte y Fundamentación Teórica

1.1. Introducción

En el presente capítulo se desarrolla el estudio del arte y la fundamentación teórica, con el objetivo de ayudar al investigador a concretar el trabajo, pues en el mismo quedan definidos algunos elementos y conceptos necesarios para buscar la respuesta al problema a resolver. Dentro de estos elementos se pueden observar los relacionados con las ontologías, la representación de la información en una ontología y dentro de ellas específicamente los estándares de representación de la información geográfica en una ontología, los SIG y otros términos relacionados con la investigación.

1.2. Definiciones y características asociadas al dominio del problema.

1.2.1. Web Semántica:

Según (Henst, 1997) La *Web* Semántica es una extensión de la *WWW* en la que los contenidos de la *web* pueden ser expresados mucho más que en un lenguaje natural y también en un formato que puedan ser entendidos interpretado y usado por diferentes *software,* permitiéndoles buscar, compartir e integrar información más fácil.

Según (Soltero, y otros, 2006) La *Web* representa un enorme repositorio de información formado de un conjunto de fragmentos que de alguna manera están integrados e interrelacionados, asociados a dominios particulares y a las empresas. Sin embargo, la información como se almacena actualmente, generalmente no tiene un claro significado que facilite su recuperación y manipulación de manera automatizada o manual. La idea principal de la *Web* Semántica es tratar de resolver esta deficiencia.

Según (Valdés, 2007) La visión de la *Web* Semántica es ampliar los principios de la *web* desde los documentos a los datos. La misma permitirá satisfacer mayor potencial a las *web*, permitiendo que los datos sean compartidos con eficiencia por grandes comunidades, y sea procesada automáticamente por las herramientas y manualmente.

Según (López, 2008) La *Web* Semántica debe encargarse de resolver las limitaciones de los sistemas de representación de conocimiento tradicionales, creando lenguajes de reglas suficientemente expresivos como para permitir a la *web* razonar tan ampliamente como se desee.

Se puede concluir que la *Web* Semántica permite resolver los problemas de la *web* tradicional, facilitando la reutilización de la información, permitiendo además integrar y compartir la misma de forma tal que pueda ser procesada por las máquinas.

1.2.2. Interoperabilidad Semántica:

La interoperabilidad semántica es un prerrequisito para hallar y acceder efectivamente a datos relevantes en diferentes contextos de aplicación (Klien, y otros, 2005).

La interoperabilidad semántica de la información y los servicios geográficos se ha convertido en una de las principales líneas de investigación de las Ciencias de la Información Geográfica (Rubal, 2005). Según (Klien, y otros, 2005) esta puede lograrse mediante el uso de geo-ontologías de dominio y anotaciones semánticas de datos geográficos que utilicen dichas geo-ontologías.

La interoperabilidad semántica puede ser tratada como una reconciliación ontológica, es decir, encontrando relaciones entre entidades que pertenecen a diferentes ontologías (Shvaiko, y otros, 2007).

Se concluye que la interoperabilidad semántica es la capacidad que poseen los sistemas de información de intercambiar datos, para obtener beneficios mutuos.

1.2.3. Ontologías:

Según (Edgar, y otros, 2012) una ontología define los términos a utilizar para describir y representar un área de conocimiento. Son utilizadas por las personas, las bases de datos y las aplicaciones que necesitan compartir un dominio de información. Incluyen además definiciones de conceptos básicos del dominio, y las relaciones entre ellos que son útiles para los ordenadores.

Según (Navarro, 2012) una ontología define un conjunto de primitivas con las cuales se modela un dominio de conocimiento o discurso, en el que, dichas primitivas son típicamente clases, atributos y relaciones entre las clases.

Otra definición describe a una ontología como una descripción formal y explícita de conceptos en un dominio de discurso (clases), propiedades de los conceptos que describen características y atributos (*slots*) y restricciones sobre las propiedades (*facets*) (Navarro, 2012).

Se puede concluir que las ontologías son las encargadas de definir términos para representar un área del conocimiento, las mismas incluyen conceptos básicos y relaciones entre ellos que son útiles para los ordenadores debido a que son utilizadas por las personas, las aplicaciones y bases de datos, con la finalidad de facilitar la comunicación y el intercambio de información entre diferentes entidades y sistemas, pues

ellas se pueden usar como repositorios para la organización de la información. Además se establecen las siguientes clasificaciones para una ontología:

(Uschol, 1996) ofrece tres dimensiones sobre las cuales varían los tipos de ontologías:

Formalidad: Se refiere al grado de formalismo del lenguaje usado para expresar la conceptualización.

Ontología altamente informal: Expresada en lenguaje natural (por ejemplo un glosario de términos).

Ontología informal estructurada: Utiliza lenguaje natural estructurado y restringido, que permite reducción de la ambigüedad.

Ontología semiformal: Usa un lenguaje de definición formal, como puede ser ontolingua.

Ontología rigurosamente formal: La definición de términos se lleva a cabo de manera meticulosa usando semántica formal, teoremas, y pruebas de estas propiedades como solidez y compleción.

Propósito: Se refiere a la intención de uso de la ontología.

Ontologías para comunicación entre personas: Una ontología informal no ambigua puede ser suficiente.

Ontologías para interoperabilidad entre sistemas: Para llevar a cabo traducciones entre diferentes métodos, lenguajes, *software*... En estos casos la ontología se usa como un formato de intercambio de conocimiento.

Ontologías para beneficiar la ingeniería de sistemas: Cuando las ontologías benefician las aplicaciones *software* apoyando aspectos como la reutilización de componentes *software* en un dominio de interés, la adquisición de conocimiento, la fiabilidad de los sistemas al proporcionar consistencia en el conocimiento utilizado, o la especificación de los sistemas *software* identificando los requisitos y definiendo especificaciones para las tecnologías de la información.

Materia: Para expresar la naturaleza de los objetos que la ontología caracteriza.

Ontologías de dominio: Caracterizan disciplinas específicas, tales como medicina, finanzas, química, biología... con independencia de los problemas o tareas relevantes de dichas disciplinas.

Ontologías de tarea, de método o de resolución de problemas: Conceptualizan el problema o la tarea a resolver en un dominio.

Ontologías de representación o metaontologías: El objeto que se caracteriza es un lenguaje de representación de conocimiento.

(Guarino, 1998) clasifica las ontologías de acuerdo con su dependencia y relación con una tarea específica desde un punto de vista, diferenciando los siguientes tipos de ontologías:

Ontologías de Alto Nivel o Genéricas: Describen conceptos muy generales como espacio, tiempo, acción... que son independientes de un problema o dominio particular. Parece razonable tener ontologías de alto nivel unificadas para una gran cantidad de comunidades de usuarios. En relación con los sistemas de información, estas ontologías describirían conceptos básicos.

Ontologías de Dominio: Describen un vocabulario relacionado con un dominio genérico especializando los conceptos introducidos en la ontología de nivel superior.

Ontologías de Tareas o de Técnicas básicas: Describen una tarea, actividad o artefacto especializando las ontologías de alto nivel.

Ontologías de Aplicación: Son las ontologías más específicas. Describen conceptos que dependen de las ontologías de dominio y de tarea, siendo con frecuencia especializaciones de ambas ontologías. Los conceptos en estas ontologías a menudo se corresponden con los roles propios de las entidades del dominio mientras que realizan una cierta actividad.

En (Van Heijst, 1997) se clasifican las ontologías de acuerdo a la cantidad y tipo de estructura de la conceptualización, distinguiendo:

Ontologías terminológicas: Especifican los términos que son usados para representar el conocimiento en el universo del discurso. Suelen ser usadas para unificar vocabulario en un campo determinado.

Ontologías de información: Especifican la estructura de almacenamiento de bases de datos. Ofrecen un marco para el almacenamiento estandarizado de información.

Ontologías de modelado de conocimiento: Especifican conceptualizaciones del conocimiento. Contienen una rica estructura interna y suelen estar ajustadas al uso particular del conocimiento que describen.

Entre las diferentes clasificaciones de ontologías mencionadas anteriormente la clasificación de Guarino es la que se utilizará, dentro de esta específicamente la ontología de Dominio, ya que la investigación actual se centra en la creación de un módulo para la edición de una ontología del dominio geográfico.

1.2.4. Geo-ontología:

Según (Santos, y otros, 2009) una geo-ontología es una ontología que ofrece una descripción de entidades geográficas y difiere de otras ontologías por la presencia predominante de relaciones topológicas que son tan importantes en el ámbito espacial.

Según (Voronisky, 2009) las geo-ontologías cumplen con todas la características de una ontología convencional, pero tienen además propiedades propias de este dominio, por ejemplo, un par de coordenadas (x, y) que representa la posición geográfica de un objeto. También incluye una serie de relaciones topológicas con una semántica determinada, por ejemplo, en el siguiente planteamiento: “el río cruza el bosque”, se puede encontrar los conceptos (objetos) río y bosque, y también se encuentra una relación cruza que actúa sobre los objetos río y bosque.

En (Santos, y otros, 2007) se plantea que las geo-ontologías son estructuras en las que se puede llevar a cabo la integración entre datos, metadatos y conocimiento espacial.

Se concluye que una geo-ontología es una ontología que cuenta además con propiedades propias, como coordenadas y relaciones tanto topológicas como espaciales.

1.2.5. Sistemas de Información Geográfica:

Existen diversas definiciones sobre los sistemas de información geográfica (SIG), dentro de las cuales se encuentran:

Conjunto de herramientas para seleccionar, almacenar, recuperar, transformar y exhibir datos espaciales del mundo real para un sistema particular con propósito definido (Garrete, y otros, 2009).

Según (Garrete, y otros, 2009) un SIG es un sistema computarizado que proporciona cuatro sistemas de capacidades para manejar datos georeferenciados: entrada de datos, administración de datos (almacenaje y recuperación), manipulación, análisis y salida de datos.

Un SIG abarca tecnología de la información, gestión de la información, asuntos legales y de negocios, y conceptos específicos de materias de un gran abanico de disciplinas, pero es implícito en la idea de SIG como una tecnología usada para tomar decisiones en la solución de problemas que tenga al menos una parte de componente espacial (López, 2008).

Se puede definir que un SIG es la unión del *software, hardware* y datos geográficos, que permite a los usuarios manipular, analizar, integrar y desplegar información geográficamente referenciada, asociada a un territorio.

Características principales de un SIG según (López, 2008) son:

- La capacidad de visualización de información geográfica compleja a través de mapas.
- La funcionalidad de los SIG como una base de datos sofisticada, en la que se mantiene y relaciona información espacial y temática.
- La diferencia con las bases de datos convencionales estriba 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.
- Son una tecnología de integración de información.
- Se han desarrollado a partir de innovaciones tecnológicas habidas en campos especializados, de la geografía y otras ciencias (tratamiento de imágenes, análisis fotogramétricos, cartografía automática, etc.), para constituir un sistema único.
- 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, etc.

1.3. Representación de la información en una ontología.

En la actualidad se hace necesario saber organizar, analizar, procesar, filtrar y representar la información, con la finalidad de facilitar la comunicación y el intercambio de la misma entre los diferentes sistemas y entidades. Por ello es importante saber utilizar las formas de representación del conocimiento. Dentro de estos se encuentran las ontologías. De forma general, para representar la información en una ontología hay que tener en cuenta la estructura y el lenguaje a utilizar. La estructura de una ontología está formada por los siguientes elementos (Navarro, 2012):

- **Conceptos:** Son las ideas básicas que se intentan formalizar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc. La forma más común de definir los conceptos es mediante clases, que no son más que un molde en el cual se definen las propiedades y el estado interno que tendrán los objetos.
- **Propiedades (*slots*):** Las propiedades son las características intrínsecas a cada concepto, que permiten discernir claramente entre uno y otro.
- **Relaciones:** Representan la interacción y enlace entre los conceptos del dominio y especifican las dependencias entre unos y otros. Una relación está definida por un dominio, formado por una o más clases a las que se les aplica la relación directamente, mientras que el rango está definido por una o más clases que pueden ser aplicadas a la relación.
- **Funciones:** Son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología.
- **Instancias:** Representan objetos determinados de una clase, son las implementaciones específicas de un concepto, con valores concretos a sus propiedades, que permiten establecer unicidad entre todos ellos.
- **Axiomas:** Son teoremas que se declaran sobre relaciones que deben cumplir los elementos de la ontología. Por ejemplo: “Si A y B son instancias de la clase C, entonces A no puede ser subclase de B”.

Existen diferentes lenguajes o conjunto de lenguajes ontológicos en los que se puede implementar una ontología, a continuación aparecen algunos de ellos (Romero, 2010):

RDF: *(Resource Description Framework)* se considera una base para procesar metadatos; proporciona interoperabilidad entre aplicaciones que intercambian información legible por máquina en la *web*. RDF destaca por la facilidad para habilitar el procesamiento automatizado de los recursos *web*. El modelo de datos RDF no hace ninguna aseveración sobre la estructura de un documento que contiene información RDF, permitiendo que las instrucciones puedan aparecer en cualquier orden dentro de una ontología. Tampoco provee primitivas de modelado para definir las relaciones entre propiedades y recursos. Esta limitación es solucionada mediante el lenguaje para describir vocabulario, conocido como RDF Schema.

RDF Schema*:* Es un vocabulario utilizado para describir relaciones entre propiedades y clases de recursos RDF, con una semántica para la generalización y jerarquización tanto de propiedades como de clases. Las primitivas de RDF(S) son agrupadas en clases, propiedades, contenedores de clases y propiedades, colecciones, vocabulario de refinación (transformación de algo abstracto en concreto) y

propiedades de utilidad. RDF(S) provee las primitivas básicas, necesarias para modelar ontologías, existe un balance adecuado entre sus capacidades de expresividad y razonamiento. Su desarrollo se realizó buscando un núcleo estable que pudiera ser fácilmente extendido. RDFS es usado ampliamente por diferentes herramientas como Protégé, Mozzila y Amaya.

OWL*:* *(Ontology Web Language)*: Es un lenguaje de marcado para publicar y compartir datos usando ontologías a través de la *web*, añade más vocabulario para describir clases (conceptos), propiedades (de tipos de datos y de objetos), relaciones entre clases e individuos (instancias). Se decidió utilizar este lenguaje en el módulo a desarrollar debido a que OWL tiene mayor capacidad para expresar significado y semántica que XML, RDF, y RDF(S). De esta forma, OWL va más allá de estos lenguajes en su capacidad para representar contenido interpretable por una máquina en la *web*. OWL proporciona tres sub-lenguajes OWL Lite, OWL DL y OWL Full cada uno con nivel de expresividad mayor que el anterior, diseñados para ser utilizados por comunidades específicas de desarrolladores y usuarios.

1.4. Representación de la información en una ontología del dominio geográfico.

Para la representación de la información geográfica se deben tener en cuenta los diferentes estándares establecidos por la OGC. Estos tienen como objetivo facilitar la captura, almacenamiento, comprobación y actualización de la información geográfica digital referida a posiciones del mundo real a través de las tecnologías de la información. Dentro de estos estándares se encuentran:

- ***Keyhole Markup Language* (KML):** el cual permite guardar información geográfica en 3 dimensiones. KML es un lenguaje XML que se centra en la visualización geográfica que incluye no sólo la presentación de datos gráficos en el mundo, sino también el control de la navegación del usuario en el sentido de a dónde ir y dónde buscar (OGC).

- ***Geography Markup Language* (GML):** el cual sirve como un lenguaje de modelado de sistemas geográficos, así como un formato de intercambio abierto para transacciones geográficas en Internet. Dentro de las ventajas que el mismo presenta se encuentra que está basado en XML y orientado a la representación de objetos geográficos, permite el modelaje, transporte y almacenamiento de información geográfica a partir de XML, también permite la verificación automática de la integridad de los datos, puede ser leído por herramientas públicas o genéricas, se edita fácilmente y es utilizado por MapServer.

Además se debe tener en cuenta los sistemas de coordenadas para la representación de la información geográfica, dentro de los cuales se encuentran el *Universal Transversal de Mercator* (UTM) que se expresa en metros y que facilita cálculos de distancia y superficie y el *World Geodetic System 84* (WGS84) que es un sistema de coordenadas geográficas mundial que permite localizar cualquier punto de la Tierra (sin necesitar otro de referencia) por medio de tres unidades dadas longitud, latitud y altitud (OGC). Este último es el que se estará utilizado para capturar las coordenadas en esta aplicación, debido a que es uno de los sistemas de coordenadas más utilizados mundialmente, por ejemplo el API de Google Maps y Open Street Map utilizan este sistema, además la LPS Aplicativos SIG utiliza como base este sistema de coordenadas.

1.5. Análisis de Soluciones existentes

En los últimos años se han creado diversas herramientas para la gestión de ontologías, a continuación se presentan las características más relevantes de algunas de ellas:

1.5.1. Ontolingua

Es una de las herramientas que se basan en el uso del navegador web. Se puede describir como un proyecto que proporciona un ambiente colaborativo distribuido, para crear, editar y modificar ontologías, disponible desde un entorno web. El servidor Ontolingua contiene una librería de ontologías compartidas por los usuarios, que fomenta muy bien el principio de reutilización de ontologías (Navarro, 2012). Incluye una API para integrar las ontologías del servidor con agentes preparados para Internet y su diseño modular, ofrece un conjunto de librerías que permiten a los usuarios ensamblar rápidamente una nueva ontología (Barrera, y otros, 2012).

1.5.2. Chimara

Esta herramienta permite crear y mantener ontologías en la web. Proporciona un ambiente distribuido para navegar, crear, editar, modificar y usar ontologías. Utiliza diferentes formatos para la carga de las bases de conocimientos. Reorganiza taxonomías y resuelve conflictos de nombres y edición de términos. Facilita la combinación permitiendo al usuario subir ontologías a su espacio de trabajo (Barrera, y otros, 2012).

1.5.3. Protégé

La herramienta para edición de ontologías más ampliamente difundida entre la comunidad científica es Protégé un ambiente para el desarrollo de sistemas basados en conocimiento que ha evolucionado hacia

un conjunto de herramientas de propósito más general, de distribución libre y con código abierto (Navarro, 2012). Esta herramienta permite crear y editar ontologías sobre RDFS, OWL y XML Schema, dentro de un entorno gráfico, usable e intuitivo. Utiliza la representación basada en marcos e implementada en CLIPS. Protégé en su núcleo, implementa un rico conjunto de estructuras de modelado de conocimiento y acciones que apoyan la creación, visualización y manipulación de ontologías en varios formatos de representación. Es una plataforma que puede ser extendida con gráficos, diagramas, componentes animados para acceder a aplicaciones embebidas en sistemas de bases de conocimientos (Barrera, y otros, 2012). Además es una herramienta que se actualiza con bastante regularidad y a la que se le pueden añadir módulos y plugins con nuevas funcionalidades (López, 2008).

Según (García, 2007) el editor de ontologías Protégé permite:

- Generar distintos tipos de documentación sobre la ontología, tales como documentos en HTML o estadísticas sobre la ontología.
- Es posible importar ontologías en diversos formatos a partir de bibliotecas de ontologías como DAML y *Ontolingua Server*, entre otras.
- Por último, con Protégé también es posible crear entornos de trabajo colaborativo, que facilitan la edición de la ontología por varios usuarios a la vez.

La plataforma Protégé soporta dos formas principales de ontologías de modelado (Investigación, 2011):

El editor **Protégé-Frames** permite a los usuarios construir y poblar ontologías que están basadas en marcos, de acuerdo con el protocolo abierto de *Knowledge Base Connectivity* (OKBC). En este modelo, una ontología consiste en un conjunto de clases organizados en una jerarquía subsunción para representar conceptos sobresalientes de un dominio, un conjunto de ranuras asociadas a las clases para describir sus propiedades y relaciones, y un conjunto de instancias de esas clases ejemplares individuales de los conceptos que tienen valores específicos para sus propiedades.

El editor **Protégé-OWL** permite a los usuarios construir ontologías para la Web Semántica, en particular en el *W3C Web Ontology Language* (OWL). "Una ontología OWL puede incluir descripciones de clases, propiedades y sus instancias. Dada una ontología, la semántica de OWL formales especifica cómo derivar

sus consecuencias lógicas, hechos, es decir, literalmente, no presentes en la ontología, sino que entraña la semántica.

WebProtégé: es un editor de ontología de código abierto basado en la Web. Proporciona una interfaz fácil de usar y altamente configurable que se puede adaptar para el uso de expertos de dominio. Tiene soporte para la edición basada en formularios y la colaboración en toda regla (Investigación, 2011).

1.5.4. WebODE

Es una plataforma de desarrollo de ontologías que ha sido desarrollada por el Grupo de Ingeniería Ontológica de la Universidad Politécnica de Madrid. Contiene un editor de ontologías que es una aplicación web que ha sido construida sobre una interfaz de acceso (ODE API), y que integra distintos servicios de construcción de ontologías como edición, navegación, documentación, mezcla, razonamiento, que integra la mayor parte de los servicios ofrecidos por la plataforma, además posee un sistema de gestión de conocimientos, un generador automático de portales Web semánticos, una herramienta de anotación de recursos web y una herramienta de edición de servicios Web semánticos (Corcho, y otros, 2004).

1.6. Conclusiones Parciales.

Después de analizar los conceptos asociados al problema a resolver se observa la importancia de las geo-ontologías para los Sistemas de Información Geográfica ya que permiten relacionar, modificar y asociar información de datos sobre otros datos. Las soluciones explicadas anteriormente no resuelven el problema de la investigación debido a que ninguna de ellas se puede integrar a la plataforma de GeneSIG. Además de que no están diseñadas para trabajar específicamente con ontologías de dominio geográfico.

Capítulo II: Herramientas a utilizar en la construcción de la aplicación.

2.1. Introducción

En este capítulo se fundamentan las tecnologías que serán utilizadas en el desarrollo del módulo para la plataforma GeneSIG. Se abordarán la metodología a utilizar que guiará el proceso de desarrollo del módulo, el lenguaje de modelado para la creación de los diagramas y los lenguajes y herramientas de programación para la implementación del mismo.

2.2. RUP

Es una metodología de desarrollo de software en la cual se intentan recoger todos los aspectos a tener en cuenta en el ciclo de vida del software la misma está compuesta por cuatro fases (EVA):

1. **Inicio:** define el ámbito y objetivos del proyecto así como la funcionalidad y capacidades del producto.

2. **Elaboración:** la funcionalidad como el dominio del problema se estudia a profundidad y se define una arquitectura básica planificando el proyecto considerando recursos disponibles.

3. **Construcción:** el producto se desarrolla a través de iteraciones involucrando tareas de análisis, diseño e implementación.

4. **Transición:** en esta etapa se libera el producto y se entrega al usuario para un uso real. Se incluyen tareas de marketing, empaquetado, instalación, configuración, capacitación, soporte, mantenimiento, etc. Los manuales de usuario se completan y refinan con la información anterior.

También esta metodología cuenta con disciplinas para la organización de las tareas como son

1. **Modelado del negocio:** comprende la estructura y la dinámica de la organización, así como los problemas actuales e identifica posibles mejoras, de los procesos del negocio. Utiliza los Diagramas de Actividad y de Clases.

2. **Requerimientos:** establece y especifica los requisitos del sistema, define los límites del sistema, y una interfaz de usuario.

3. **Análisis y diseño:** define la arquitectura del sistema, además el diseño refina el análisis para poder implementar los diagramas de clases, los diagramas de colaboración, el de clases, el de secuencia, el de estados, y el modelo de la arquitectura.

4. **Implementación:** implementa las clases de diseño como componentes transformándolo en código fuente, asigna y prueba los componentes individualmente, integra los componentes en un sistema ejecutable. Utiliza los diagramas de componentes.

5. **Pruebas:** verifica la integración de los componentes, que todos los requisitos han sido implementados.

6. **Despliegue:** en esta disciplina se realizan las actividades de probar el software en su entorno final, empaquetarlo, distribuirlo e instalarlo, así como la capacitación.

7. **Gestión y configuración de cambios:** es un apoyo al control de cambios evitan confusiones costosas como la compostura de algo que ya se había arreglado y aseguran que los resultados no entren en conflicto con alguna de las siguientes actividades:

- **Actualización simultánea:** Es la actualización de algo elaborado con anterioridad, sin saber que alguien más lo está actualizando.

- **Notificación limitada:** Al realizar alguna modificación, no se deja información sobre lo que se hizo, por lo tanto no se sabe quién, como, y cuando se hizo.

- **Versiones múltiples:** No saber con exactitud, cual es la última versión, y al final no se tiene un orden sobre que modificaciones se han realizado a las diversas versiones.

8. **Gestión del proyecto:** planea, dirige al personal, ejecutar acciones y supervisar proyectos.

9. **Entorno:** se enfoca sobre las actividades necesarias para configurar el proceso que engloba el desarrollo de un proyecto y describe las actividades requeridas, provee a la organización que desarrollará un ambiente en el cual basarse, el cual provee procesos y herramientas para poder desarrollar el software.

Características de RUP:

1 **Guiado por casos de uso:** donde los casos de uso definen lo que el usuario desea a partir de la captura de requisitos y la modelación del negocio.

2 **Centrado en la arquitectura:** característica que brinda una visión completa del sistema, se describen los procesos del negocio que son más importantes, para comprenderlo, desarrollarlo y producirlo de una forma eficaz.

3 **Iterativo e incremental:** donde cada fase se desarrolla en iteraciones, de forma tal que se pueda dividir en pequeños proyectos mejorando su comprensión y desarrollo.

Como metodología de desarrollo de software para la realización del módulo para la edición de una ontología del dominio geográfico en la plataforma GeneSIG se ha seleccionado RUP. La misma se utiliza para analizar, implementar y documentar los sistemas, lo que es más factible para el equipo de trabajo teniendo en cuenta que se necesita garantizar la continuidad del proyecto, lo que constituye una ventaja.

Esta metodología se basa en el desarrollo iterativo y el modelado visual para describir un sistema, lo cual permite incorporar al proceso de desarrollo de software un mejor control de los requerimientos y cambios. RUP se va a centrar en una gestión cuidadosa de requisitos, que es necesario realizar para el desarrollo del módulo que se propone. En RUP no se hace necesaria la presencia del cliente en todas las iteraciones. En cada iteración puede verificarse la calidad del sistema desde el comienzo.

2.3. Lenguaje Unificado de Modelado (UML) 2.0

UML surge como respuesta al problema de contar con un lenguaje estándar para el modelado de software. Está basado en una notación gráfica la cual permite: especificar, construir, visualizar y documentar los objetos de un sistema programado (EVA).

Características de UML:

1 Es un lenguaje consolidado, fácil de aprender y permite la documentación de todo el ciclo de desarrollo del sistema.

2 Puede ser usado en todas las etapas de desarrollo del sistema y su representación gráfica puede ser usada para comunicarse con los usuarios o entre el equipo de desarrollo.

3 Se ha venido adoptando en diferentes medios empresariales y académicos como el lenguaje estándar para el análisis y diseño de los sistemas de software.

El UML proporciona ventajas por lo que ha sido seleccionado para modelar la aplicación, pues permite generar código a partir de los modelos y a la inversa, también permite documentar todo el ciclo de vida del módulo para el desarrollo de futuras versiones, además de que mediante los diagramas se muestre de manera entendible toda la estructura y el comportamiento del mismo. GeneSIG es un producto de gran tamaño por lo que UML juega un papel primordial en este proceso de desarrollo.

2.4. Visual Paradigm 8.0.

Esta herramienta acelera el desarrollo de aplicaciones, ya que sirve de puente visual entre arquitectos, analistas y diseñadores de sistemas de información, haciendo el trabajo más fácil y dinámico (Cuervo & Moreno, 2005). La misma es utilizada hoy en día para el modelado de los diagramas UML.

Visual Paradigm es una herramienta CASE para el modelado UML muy potente, gratuita, fácil de instalar, utilizar y actualizar. Permite dibujar todo tipo de diagramas UML, revertir código fuente a modelos UML, generar código fuente desde los diagramas UML. Incluye los objetos más recientes de UML además de diagramas de casos de uso, diagramas de clase, diagramas de componentes, reversa instantánea para Java, C++, DotNet Exe/DLL, XML, XML Schema, y Corba IDL, ofrece soporte para Rational Rose, integración con Microsoft Visio, además de generar reportes y documentación en HTML/PDF (Casals & Squires, 2013).

Debido a que se desarrollará un trabajo bajo las políticas de software libre, decir, que dicha herramienta no es libre, pero la universidad cuenta con una licencia comercial, por lo que se utilizará para el modelado de la aplicación como herramienta CASE el Visual Paradigm.

2.5. Plataforma GeneSIG 1.5

La plataforma GeneSIG fue desarrollada por el centro de desarrollo de GEySED, perteneciente a la facultad 6 de la Universidad de las Ciencias Informáticas (UCI) en conjunto a equipos de desarrollo de GeoCuba y UCID centro que se encuentra dentro de la universidad. Está implementada con herramientas y tecnologías libres, su principal objetivo es la representación geoespacial de la información y realizar análisis sobre la misma.

La plataforma soberana GeneSIG es un Sistema de Información Geográfica Web basado en estándares OpenGIS que incluye funcionalidades operativas de las aplicaciones de esta tecnología. Algunos de los

componentes de la plataforma son la base de datos geoespacial, visor web interactivo de acceso a los datos, servicio de catálogo y módulo de análisis. Además de contar con una interfaz bastante sencilla y de fácil manejo, permitiendo la personalización y uso de todos los componentes que la integran.

A esta plataforma se le integrará el módulo que se está desarrollando con el presente trabajo por lo que se tiene en cuenta las herramientas utilizadas en la construcción de la plataforma para que el módulo sea compatible con la misma.

2.6. Gestor de bases de datos PostgreSQL 8.4 (PostGIS 1.5.2).

PostgreSQL, llamado Sistema Gestor de Bases de Datos Relacionales Orientadas a Objetos, es usado para manejar grandes cantidades de información, y está basado en el modelo relacional, aunque incorpora conceptos del modelado orientado a objetos. Es distribuido bajo licencia BSD y con su código fuente disponible libremente, es multiplataforma, soporta múltiples transacciones, integridad de datos, presenta una estabilidad muy alta, gran seguridad de los datos, potencia, robustez, facilidad de administración e implementación de estándares estas son algunas de sus características principales (Martinez, 2009).

PostGIS es una extensión de PostgreSQL que le añade soporte de objetos geográficos, debido a que está construido sobre PostgreSQL, PostGIS hereda automáticamente sus características, así como los estándares abiertos. A continuación se puede observar las ventajas que tiene la utilización de PostGIS (Ramsey, 2012):

- PostGIS es software libre, tiene licencia GNU *General Public License* (GPL).
- Es compatible con los estándares de OGC.
- Soporta tipos de datos espaciales, índices espaciales y tiene cientos de funciones espaciales.
- Permite importar y exportar datos a través de varias herramientas conversoras (shp2pgsql, pgsql2shp, ogr2ogr, dxf2postgis).

Se estará utilizando debido a que es el gestor de Base de Datos con que trabaja la plataforma GeneSIG y la Línea de Producción de Software AplicativoSIG.

Capítulo II: Herramientas a utilizar en la construcción de la aplicación.

2.7. Lenguajes de programación del lado del servidor

Java.

Java es un lenguaje de programación muy extendido y que cada vez cobra más importancia en el ámbito de la informática. Una de sus principales características es que es un lenguaje independiente de la plataforma. Java es un lenguaje de programación de propósito general, concurrente, basado en clases y orientado a objetos (Jalón, y otros, 2002). Mencionar además que Java se ha convertido en un lenguaje con una implantación masiva en todos los entornos (personales y empresariales). Java permite un rápido desarrollo de la interfaz de usuario, escribir código libre de errores, nuevo soporte HTML5, soporte para múltiples idiomas, amplio conjunto de Plugins, mejor soporte para las últimas tecnologías Java, código inteligente y edición rápida, fácil y eficiente gestión de proyectos, es robusto, seguro, multitarea y portable.

Por lo que se decidió utilizarlo debido a que permite el desarrollo de aplicaciones bajo el esquema de Cliente-Servidor, como de aplicaciones distribuidas, lo que lo hace capaz de conectar dos o más computadoras u ordenadores, ejecutando tareas simultáneamente.

PHP 5.3.

PHP es el acrónimo de *Hypertext Prepocessor* este lenguajes de programación, se creó para la generación de páginas *web*, este lenguaje es libre y gratuito, es decir, está amparado bajo el movimiento código abierto (*open source*), tiene extensiones para soportar múltiples bases de datos y además es multiplataforma. Generalmente se ejecuta en un servidor *web*, tomando el código en PHP como su entrada y creando páginas *web* como salida. Algunas de las principales características de PHP (Hanze, y otros, 2008):

1 Puede ser utilizado en cualquiera de los principales Sistemas Operativos del mercado, incluyendo Linux, muchas variantes Unix (incluyendo HP-UX, Solaris y OpenBSD), Microsoft Windows, Max OSX, RISC OS.

2 Soporta la mayoría de servidores *web* hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server y muchos otros.

3 Tiene módulos disponibles para la mayoría de los servidores, para aquellos otros que soporten el Estándar CGI, PHP puede usarse como procesador CGI.

4 Tiene la posibilidad de usar programación procedimental o programación orientada a objetos.

5 La característica más potente es que tiene soporte para una gran cantidad de base de datos.

2.8. Lenguaje de programación del lado del cliente

EXTJS 3.0.

ExtJS es una biblioteca o conjunto de librerías JavaScript para el desarrollo de aplicaciones web interactivas, usa tecnologías AJAX, DHTML y DOM. La misma permite realizar interfaces de usuario, es generalmente usada en sistemas que requieren altos niveles de interacción con el usuario, la comunicación con el servidor se realiza en segundo plano, permitiendo solicitar o enviar datos y procesar la información recibida en tiempo real, es fácil de usar, muy parecidas a las conocidas aplicaciones de escritorio. Esto permite a los desarrolladores web concentrarse en la funcionalidad de las aplicaciones en vez de en las advertencias técnicas, además posee licencias Open Source GPL y comerciales.

2.9. Entorno de Desarrollo Integrado (IDE) NetBeans 7.2.1

NetBeans es un Entorno de Desarrollo Integrado (IDE) disponible para Windows, Mac, Linux y Solaris. Ofrece todas las funciones de los IDE avanzados como diseño de interfaces, asistentes para la conexión con bases de datos, creación automática de propiedades y clases, etc. El proyecto NetBeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones *web,* de escritorio y móviles utilizando la plataforma Java, así como PHP, JavaScript, Ajax, Groovy y Grails, y C / C + +.

En NetBeans IDE 7.2.1 se ha mejorado el rendimiento y la experiencia de codificación. La versión incluye también notables características como la integración con el generador de escena para la creación de formas visualmente JavaFX, soporte para múltiples frameworks PHP, soporte actualizado Groovy, y muchas otras mejoras en Java EE, Maven y C/C + +(Oracle Corporation, 2013).

2.10. OpenJDK 6

OpenJDK es la versión libre de la plataforma de desarrollo Java bajo concepto de lenguaje orientado a objetos. Fue creada por la empresa denominada Sun Microsystems. Esta implementación se encuentra catalogada dentro de la licencia GPL de GNU con una excepción de enlaces, por lo que algunos de los componentes de los folders de clases y sitios web de Java se ultiman de los términos de la licencia para poder ser considerados dentro de la versión estipulada como GNU.

2.11. SOAP (Simple Object Access Protocol).

Ofrece los mecanismos de comunicación básicos para el envío de mensajes en formato XML, permitiendo la invocación remota de servicios. Normalmente funciona sobre HTTP. El mismo consiste de tres partes, un sobre, un conjunto de reglas de codificación y una conversión para representar llamadas a procedimientos y respuestas. El protocolo SOAP tiene tres características principales: extensibilidad, neutralidad e independencia. SOAP define cómo organizar información de forma estructurada para intercambiarla entre distintos sistemas.

La utilización de los servicios web posee gran importancia debido a la independencia de la que proveen a las aplicaciones, que hace posible que tanto el cliente como el servidor sean tratados de manera diferenciada. Esto permite que puedan ser implementados por separado y los cambios que surjan en uno no afectarán al otro (Pelechano, 2005).

2.12. Servidor de mapas MapServer 5.6.

Es un entorno de desarrollo de código abierto para la publicación de datos espaciales y aplicaciones cartográficas interactivas. Dentro de sus principales características se encuentran: dibujo y etiquetado dependiente de la escala, símbolo y color adaptable, acceso en función de las características a datos sobre atributos, generación automática de leyendas, y utilización de datos en forma de mosaico, salida avanzada de cartografía, soporta scripts desarrollados en php, java, C# entre otros, es multiplataforma, soporta múltiples formatos ráster y vectorial.

2.13. Servidor de aplicaciones Apache 2.2.

Apache es uno de los servidores Web utilizados en la red, desde hace mucho tiempo, únicamente le hace competencia un servidor de Microsoft, el IIS. Por lo que este servidor es uno de los mayores triunfos del software libre, que tanto gusta a los usuarios de LINUX.

Algunas características importantes del servidor Apache son:

- Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con el API de programación de módulos, para el desarrollo de módulos específicos.
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.

- Se desarrolla de forma abierta.
- Gracias a ser modular, se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor.

Las funcionalidades más elementales se encuentran en el módulo base, siendo necesario un módulo multiproceso para manejar las peticiones. Se han diseñado varios módulos multiproceso para cada uno de los sistemas operativos sobre los que se ejecuta el Apache, optimizando el rendimiento y la rapidez del código.

El resto de las funcionalidades del servidor se consiguen por medio de módulos adicionales que se pueden cargar. Para añadir un conjunto de utilidades al servidor, simplemente hay que añadirle un módulo, de forma tal que no sea necesario volver a instalar el software.

Se estará utilizando el servidor de aplicaciones Apache para cargar y desplegar la aplicación que se estará desarrollando.

2.14. Servidor Web Glassfish 3.1.2.

Servidor de aplicaciones desarrollado por *Sun Microsystems* que implementa las tecnologías definidas en Java EE y permite ejecutar aplicaciones que siguen esta especificación. Dentro de las principales características se encuentran que es gratuito y de código libre, además cabe destacar su velocidad, alta escalabilidad, manejo centralizado de *clusters* e instancias, bajo consumo de memoria y un excelente panel de administración. Su versión comercial es denominada *Sun Glassfish Enterprise Server*.

2.15. OWL-API

De todas las soluciones estudiadas anteriormente se decidió utilizar la OWL-API debido a que esta es de código abierto y está disponible bajo la licencia LGPL, además es para OWL 2 y posee una eficiente memoria de Referencia de Aplicación. También se le pueden integrar razonadores y proporciona casi total apoyo a las tareas de gestión de RDF y OWL estándar, tales como agregar, eliminar y consultar triples. La misma permitió reutilizar los métodos que ya están implementados en el módulo a construir.

2.16. Conclusiones

Se analizaron las diferentes herramientas y metodologías que se usarán para el desarrollo del presente trabajo, las cuales se eligieron teniendo en cuenta las necesidades y características propias del módulo que se desea construir así como las facilidades y ventajas que brindan cada una de ellas. Las

herramientas y metodologías seleccionadas para el desarrollo del presente trabajo fueron en el caso de las metodologías la propuesta seleccionada fue RUP debido a que la plataforma GeneSIG es un proyecto de gran envergadura y se necesita registrar todo el proceso de desarrollo del módulo detalladamente, haciendo uso de UML como lenguaje de modelado y Visual Paradigm como herramienta CASE, mientras que NetBeans fue elegido en el caso de la plataforma de desarrollo. Como lenguaje de programación se utilizarán del lado del servidor PHP 5.3 y Java, del lado del cliente EXTJS 3.0 y como gestor de bases de datos PostgreSQL con PostGIS.

Capítulo III: Presentación de la solución propuesta.

3.1. Introducción

Durante el desarrollo de este capítulo se realizará el levantamiento de requisitos funcionales y no funcionales, los mismos guiarán el proceso de desarrollo del futuro producto hasta su implementación. Además se realizará el modelo de dominio, el diagrama de caso uso del sistema y la descripción de los casos de uso del sistema.

3.2. Modelo de Dominio.

El modelo de dominio se utiliza para un mejor entendimiento del área bajo análisis como paso previo al diseño de un sistema, es utilizado como un medio para comprender el sector industrial al cual el sistema va a servir. Relacionará objetos en el dominio del sistema con otro, definirá conceptos y términos (ver Figura # 1).

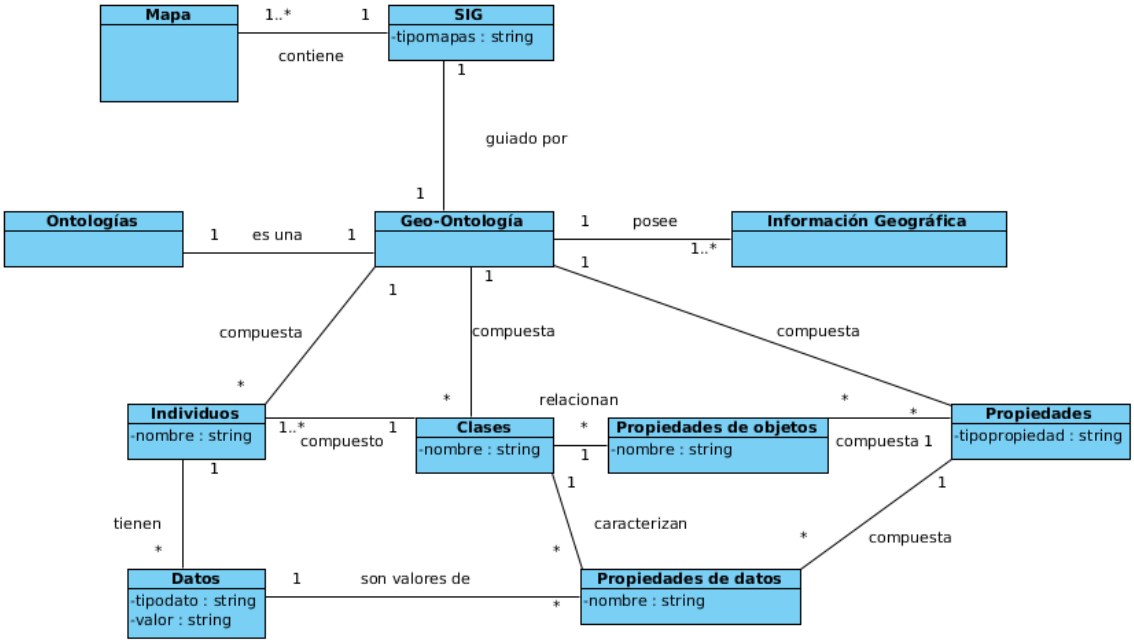


Figura # 1 Modelo de Dominio.

3.2.1. Definiciones de clases del modelo de dominio.

Ontología: Son herramientas encargadas de definir términos para representar un área del conocimiento.

Geo-ontología: Es una ontología de dominio geográfico.

Individuos: Representan objetos determinados de un concepto.

Clases: Son la base de la descripción del conocimiento en las ontologías ya que describen los conceptos (ideas básicas que se intentan formalizar) del dominio.

Propiedades: Son las características o atributos que describen a los conceptos.

Mapa: Es una representación gráfica y métrica de una porción de territorio.

Información geográfica: Es aquellos datos espaciales georeferenciados requeridos como parte de las operaciones científicas, administrativas o legales.

SIG: Es una integración organizada de *hardware*, *software* y dato geográfico diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada.

Datos: Son los valores que tiene un individuo asociado a una propiedad de datos de una clase.

Propiedades de objeto: Representan relaciones que existen entre las clases.

Propiedades de datos: Son atributos que caracterizan a una clase.

3.3. Levantamiento Requisitos.

El Levantamiento de Requisitos se realiza con el objetivo de entender y comprender mejor el problema que se necesita solucionar, existen los Requisitos Funcionales (RF) los cuales especifican los detalles más relevantes que la aplicación debe presentar y los Requisitos No Funcionales (RNF) que son las cualidades que la misma debe poseer para un mejor funcionamiento.

3.3.1. Requisitos funcionales.

RF 1.Mostrar jerarquías de clases: Esta funcionalidad permite mostrar la jerarquía de clase en forma de árbol.

RF 2. Adicionar una clase: El sistema debe ser capaz de adicionar una clase como hija de una clase seleccionada. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre de la clase (Formato: Alfanumérico, Obligatorio: Sí).

RF 3. Eliminar una clase: El sistema debe ser capaz de eliminar una clase seleccionada.

- Clase (Formato: Selección, Obligatorio: Sí).

RF 4. Listar clases Equivalentes: El sistema debe mostrar todas las clases equivalentes a partir de una clase seleccionada.

- Clase (Formato: Selección, Obligatorio: Sí).

RF 5. Adicionar clases Equivalentes: El sistema debe permitir dada una clase seleccionada adicionar una clase equivalente. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre clase equivalente (Formato: Alfanumérico, Obligatorio: Sí).

RF 6. Listar clases disjuntas: El sistema debe mostrar todas las clases disjuntas, a partir de una clase seleccionada.

- Clase (Formato: Selección, Obligatorio: Sí).

RF 7. Adicionar clases disjuntas: Esta funcionalidad permite dada una clase seleccionada adicionar una clase disjunta. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre clase equivalente (Formato: Alfanumérico, Obligatorio: Sí).

RF 8. Mostrar jerarquías de propiedades de objetos: La funcionalidad permite mostrar las jerarquías de las propiedades de objetos en forma de árbol.

RF 9. Adicionar propiedad de objetos: El sistema debe ser capaz de adicionar una propiedad de objeto como hija de una propiedad de objeto seleccionada. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre de la propiedad de objeto (Formato: Alfanumérico, Obligatorio: Sí).

RF 10. Eliminar propiedad de objetos: El sistema debe ser capaz de eliminar una propiedad de objeto seleccionada.

- Propiedad de objeto (Formato: Selección, Obligatorio: Sí).

RF 11. Listar Dominios de (Intersección): A partir de una propiedad de objeto seleccionada, el sistema debe ser capaz de mostrar sus dominios de (Intersección).

- Propiedad de objeto (Formato: Selección, Obligatorio: Sí).

RF 12. Adicionar Dominios de (Intersección): Esta funcionalidad debe ser capaz dada una propiedad de objeto seleccionada, adicionar dominios de (Intersección). Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre del dominio de (intersección) (Formato: Alfanumérico, Obligatorio: Sí).

RF 13. Listar Rangos de (Intersección): Esta funcionalidad debe permitir a partir de una propiedad de objeto mostrar sus rangos de (Intersección).

- Propiedad de objeto (Formato: Selección, Obligatorio: Sí).

RF 14. Adicionar Rangos de (Intersección) : Esta funcionalidad debe ser capaz dado una propiedad de objeto seleccionada, adicionar un rango de (Intersección). Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre del rango de (Intersección). (Formato: Alfanumérico, Obligatorio: Sí).

RF 15. Listar propiedades equivalentes de objetos: Esta funcionalidad permite a partir de una propiedad de objeto seleccionada mostrar todas las propiedades de objetos equivalentes.

- Propiedad de objeto (Formato: Selección, Obligatorio: Sí).

RF 16. Adicionar propiedades equivalentes de objetos: Esta funcionalidad debe ser capaz dada una propiedad de objeto seleccionada, adicionar una propiedad de objeto equivalente. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre la propiedad de objeto equivalente (Formato: Alfanumérico, Obligatorio: Sí).

RF 17. Listar propiedades inversas: Esta funcionalidad debe permitir a partir de una propiedad de objeto seleccionada que el sistema muestre todas las propiedades inversas.

- Propiedad de objeto (Formato: Selección, Obligatorio: Sí).

RF 18. Adicionar propiedades inversas: El sistema debe ser capaz dado una propiedad de objeto seleccionada adicionar una propiedad inversa. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre de la propiedad inversa (Formato: Alfanumérico, Obligatorio: Sí).

RF 19. Listar propiedades disjuntas: Esta funcionalidad debe permitir a partir de una propiedad de objeto seleccionada que el sistema muestre todas las propiedades disjuntas.

- Propiedad de objeto (Formato: Selección, Obligatorio: Sí).

RF 20. Adicionar propiedades disjuntas: Esta funcionalidad debe ser capaz dado una propiedad de objeto seleccionada, adicionar una propiedad disjunta. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre de la propiedad disjunta (Formato: Alfanumérico, Obligatorio: Sí).

RF 21. Mostrar jerarquía de propiedades de datos: Esta funcionalidad permite mostrar jerarquías de propiedades de datos en forma de árbol.

RF 22. Adicionar propiedades de datos: El sistema debe ser capaz de adicionar una propiedad de datos como hija de una propiedad de datos seleccionada. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre de la propiedad de datos (Formato: Alfanumérico, Obligatorio: Sí).

RF 23. Eliminar propiedades de datos: Esta funcionalidad debe permitir eliminar una propiedad de dato seleccionada.

- Propiedad de datos (Formato: Selección, Obligatorio: Sí).

RF 24. Listar dominios: Esta funcionalidad debe permitir a partir de una propiedad de datos seleccionada, mostrar sus dominios.

- Propiedad de datos (Formato: Selección, Obligatorio: Sí).

RF 25. Adicionar dominios: Esta funcionalidad debe ser capaz dada una propiedad de datos seleccionada, adicionar dominios. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre del dominio (Formato: Alfanumérico, Obligatorio: Sí).

RF 26. Listar propiedades equivalentes: Esta funcionalidad permite a partir de una propiedad de datos seleccionada, mostrar todas las propiedades equivalentes.

- Propiedad de datos (Formato: Selección, Obligatorio: Sí).

RF 27. Adicionar propiedades equivalentes: Esta funcionalidad debe ser capaz dada una propiedad de datos seleccionada, adicionar propiedades equivalente. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre propiedad equivalente (Formato: Alfanumérico, Obligatorio: Sí).

RF 28. Listar propiedades disjuntas: Esta funcionalidad debe permitir a partir de una propiedad de datos seleccionada, que el sistema muestre todas las propiedades disjuntas.

- Propiedad de datos (Formato: Selección, Obligatorio: Sí).

RF 29. Adicionar propiedades disjuntas: Esta funcionalidad debe ser capaz dado una propiedad de datos seleccionada, adicionar una propiedad disjuntas. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Nombre de la propiedad disjunta (Formato: Alfanumérico, Obligatorio: Sí).

RF 30. Listar individuo: Esta funcionalidad permite mostrar una lista de individuos en forma de árbol dada una clase.

- Clase (Formato: Selección, Obligatorio: Sí).

RF 31. Adicionar individuo: El sistema debe ser capaz de adicionar un individuo como hijo de un individuo seleccionado. Para ejecutar esta funcionalidad se tiene que tener en cuenta los siguientes criterios de entrada:

- Nombre del individuo (Formato: Alfanumérico, Obligatorio: Sí).

RF 32. Eliminar individuo: Esta funcionalidad debe ser capaz de eliminar un individuo seleccionado.

- Individuo (Formato: Selección, Obligatorio: Sí).

RF 33. Listar individuos equivalentes: Esta funcionalidad debe permitir a partir de un individuo seleccionado, mostrar un listado de todos los individuos equivalentes.

- Individuo (Formato: Selección, Obligatorio: Sí).

RF 34. Adicionar un individuo equivalente: Esta funcionalidad debe permitir a partir de un individuo seleccionado adicionar un individuo equivalente. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Individuo (Formato: Alfanumérico, Obligatorio: Sí).

RF 35. Listar diferentes individuos: Esta funcionalidad debe permitir a partir de un individuo seleccionado, mostrar un listado de todos los individuos que sean diferentes.

- Individuo (Formato: Selección, Obligatorio: Sí).

RF 36. Adicionar un individuo diferente: Esta funcionalidad debe permitir a partir de un individuo seleccionado, adicionar un individuo con características diferentes. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Individuo (Formato: Alfanumérico, Obligatorio: Sí).

RF 37. Listar afirmaciones de propiedades de objeto: Esta funcionalidad debe permitir a partir de un individuo seleccionado, mostrar un listado de todas las afirmaciones de propiedades de objeto.

- Individuo (Formato: Selección, Obligatorio: Sí).

RF 38. Adicionar afirmaciones de propiedades de objeto: Esta funcionalidad debe permitir a partir de un individuo seleccionado, adicionar afirmaciones de propiedades de objeto. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Afirmación de propiedad de objeto (Formato: Alfanumérico, Obligatorio: Sí).

RF 39. Listar afirmaciones negativas de propiedades de objeto: Esta funcionalidad debe permitir a partir de una propiedad de objeto seleccionada, mostrar un listado de todas las afirmaciones negativas de propiedades de objeto.

- Individuo (Formato: Selección, Obligatorio: Sí).

RF 40. Adicionar afirmaciones negativas de propiedades de objeto: Esta funcionalidad debe permitir a partir de una propiedad de objeto, adicionar afirmaciones negativas de propiedades de objeto. Para ejecutar esta funcionalidad se tiene que tener en cuenta el siguiente criterio de entrada:

- Afirmación negativa de propiedad de objeto (Formato: Alfanumérico, Obligatorio: Sí).

RF 41. Listar afirmaciones de propiedades de datos: Esta funcionalidad debe permitir a partir de una clase y un individuo seleccionado, mostrar un listado de todas las afirmaciones de dato.

- Clase (Formato: Selección, Obligatorio: Sí).
- Individuo (Formato: Selección, Obligatorio: Sí).

RF 42. Adicionar Afirmaciones de propiedades de datos: Esta funcionalidad debe permitir a partir de una clase y un individuo seleccionado, adicionar una afirmación de propiedades de datos. Para ejecutar esta funcionalidad se tiene que tener en cuenta los siguientes criterios de entrada:

- Propiedad del dato (Formato: Selección, Obligatorio: Sí).
- Tipo de dato (Formato: Selección, Obligatorio: Si).
- Valor (Formato: Alfanumérico, Obligatorio: No).

3.3.2. Requisitos no funcionales.

Usabilidad

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

Fiabilidad

- La información y las funcionalidades del sistema estarán disponibles y el usuario podrá acceder a ellas las 24 horas de los 7 días de la semana.

Eficiencia

- El tiempo de respuesta estará dado por la cantidad de información a procesar, entre mayor cantidad de información mayor será el tiempo de procesamiento. El mismo no debe exceder los 15 segundos.
- Al igual que el tiempo de respuesta, la velocidad de procesamiento de la información, la actualización y la recuperación dependerán de la cantidad de información que tenga que procesar la aplicación.

Restricciones de diseño

- El producto de software final debe diseñarse sobre una arquitectura cliente-servidor.
- Se debe lograr un producto altamente configurable y extensible, teniendo en cuenta que se desarrollará para adjuntarlo a la Plataforma GeneSIG.

Interfaz

Interfaces de usuario

El sistema debe:

- Tener una apariencia profesional y un diseño gráfico sencillo con la utilización de las tonalidades de los colores representativos de la plataforma GeneSIG.

Interfaces de hardware

Para las PCs clientes:

- Se requiere tengan tarjeta de red.
- Al menos 128 MB de memoria RAM.
- Procesador 512 MHz como mínimo.

Para los servidores:

- Se requiere tengan tarjeta de red.
- El Servidor de la aplicación tenga como mínimo 2GB de RAM y 80 GB de disco duro.
- El Servidor de BD tenga como mínimo 2GB de RAM y 80 GB de disco duro.
- El servidor de mapas tenga como mínimo 2GB de RAM.
- Procesador 3 GHz como mínimo.

Interfaces de software

Para las PCs clientes:

- Un navegador como Mozilla Firefox u otro navegador que cumpla con los estándares W3C.
- Sistema operativo: GNU/Linux o Windows.
- Para los Servidores:
- Sistema operativo: GNU/Linux Ubuntu Server 11.04.
- 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.
- MapServer 5.2.2 o superior, con extensión PHP Mapscript.

Requisito de licencia

- De acuerdo a las licencias de las herramientas que se proponen a utilizar para el desarrollo del módulo se puede decir que el mismo posee una arquitectura de modelo libre, que permitirá la utilización, modificación y distribución de las mismas por terceros sin necesidad de obtener la autorización de sus respectivos titulares.

3.4. Descripción de los actores del sistema.

Tabla 1 Descripción de los actores del sistema

Actor	Descripción
Cliente	Es aquel cliente que tendrá acceso a todas las funcionalidades del sistema.

3.5. Diagrama de caso uso del sistema.

A partir de la obtención de los requisitos funcionales se obtiene una vista externa del sistema, que dio paso a la construcción del diagrama de caso de uso del sistema (ver Figura # 2) donde se ha representado el actor que interactúa con el sistema a través de los casos de uso, los cuales se describen a continuación.

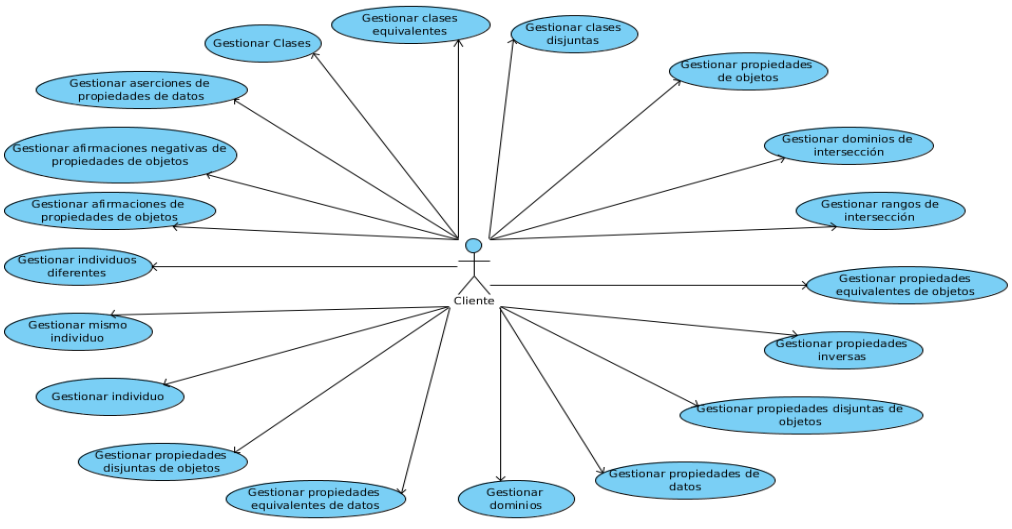


Figura # 2 Diagrama de caso de uso del sistema.

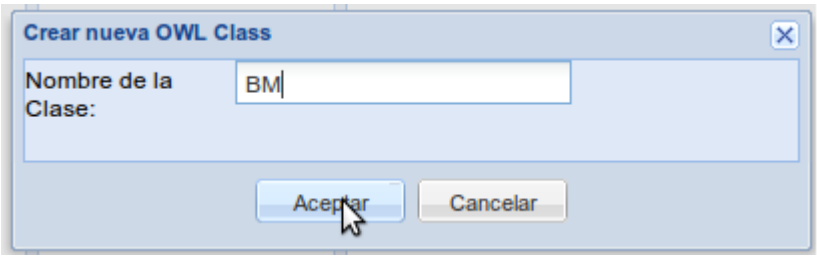
3.6. Descripción de los casos de usos del sistema.

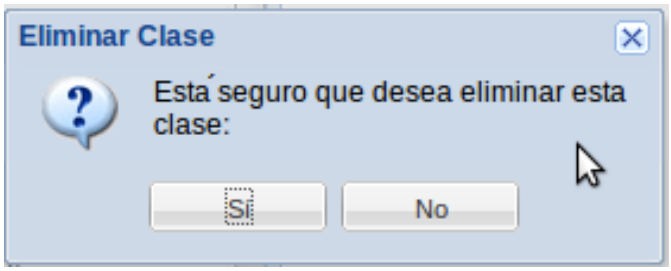
Gestionar Clases

Tabla 2 Descripción del CU Gestionar Clase.

Caso de Uso:	Gestionar Clase
Actores:	Cliente
Propósito	Este caso de uso se lleva a cabo con el objetivo de que el cliente pueda ver el listado de clase, adicionar una clase y eliminar una clase.
Resumen:	El caso de uso comienza cuando el usuario desea realizar alguna operación con las clases, ya sea adicionar, eliminar o listar clases.
Precondiciones:	**-**
Referencias	RF 1, RF 2 y RF 3
Prioridad	Alta

Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando el cliente presiona la pestaña Clases en la aplicación.	2. Se muestra en pantalla las opciones: Adicionar. Eliminar.
3. Selecciona la opción deseada. seleccionó la opción Adicionar, ver Sección “Adicionar clase”. Si seleccionó la opción Eliminar, ver sección “Eliminar clase”.	4. Termina el CU.
Sección “Adicionar clase”	

	1. Se muestra la ventana “Crear nueva OWL Class”.
2. Introduce el nombre de la clase.	3. Verifica que el campo no esté en blanco.
4. Presiona el botón (Aceptar).	5. Adiciona la clase, Volver al Flujo Normal de Eventos 4.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3. Campo en blanco, No se activa el botón (Aceptar). Volver al flujo 2 de la sección “Adicionar clase”.
4. Presiona el botón (Cancelar).	4.1 Cierra la ventana y se termina el caso de uso.
Prototipo de Interfaz	
	
Sección “Eliminar clase”	
1. Selecciona la clase que desea eliminar y presiona el botón (Eliminar).	2. Se muestra una ventana con las opciones “Sí” o “No”.
3. Presiona el botón “Sí”.	4. Elimina la clase y se muestra el listado con las clases existentes. Volver al Flujo Normal de Eventos 4.

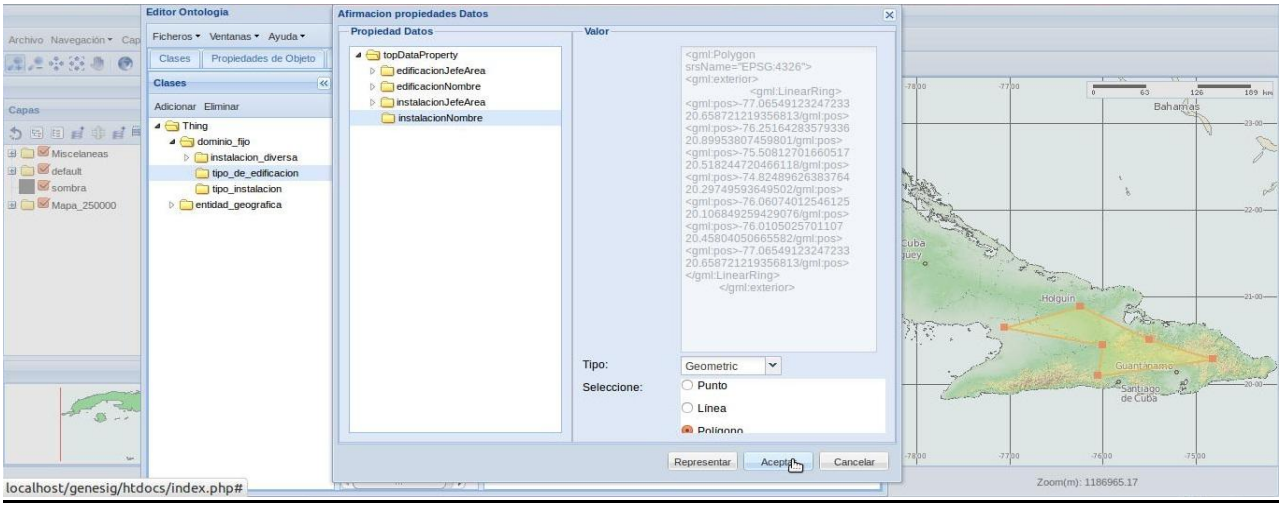
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
3. Presiona el botón “No”.	3.1 Cierra la ventana y se termina el CU.
Prototipo de Interfaz	
	
Pos-condiciones	El sistema muestra las clases existente.

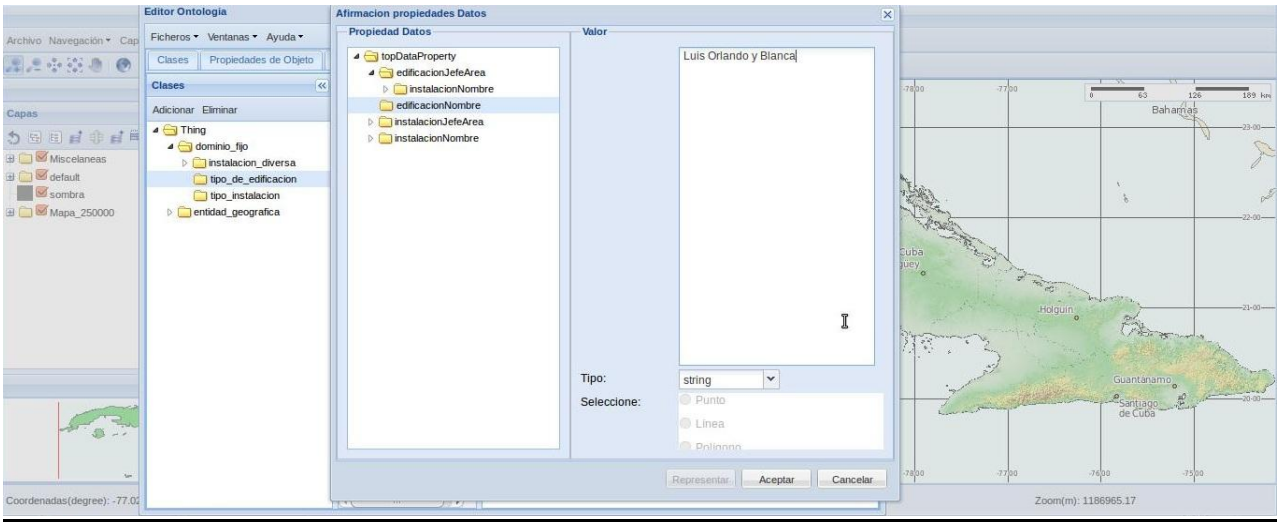
Gestionar aserciones de propiedades de datos.

Tabla 3 Descripción CU Adicionar aserciones de propiedades de datos.

Caso de Uso:	Gestionar aserciones de propiedades de datos.
Actores:	Cliente
Propósito	Este caso de uso se lleva a cabo con el objetivo de que el cliente pueda adicionar aserciones (valores) de propiedades de datos.
Resumen:	El caso de uso comienza cuando el cliente desea realizar alguna operación con las aserciones de propiedades de datos, ya sea adicionar o listar dichas propiedades.
Precondiciones:	-
Referencias	RF 41 y FR 42

Prioridad	Alta
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. El caso de uso inicia cuando selecciona dentro de la sección "Individuos" la clase y el individuo al cual desea "Adicionar afirmaciones propiedades de datos". Luego selecciona dentro de la pestaña Afirmación propiedad datos el botón adicionar.	2. Se muestra en pantalla la ventana "Afirmaciones propiedades de Datos"
3. El cliente selecciona los datos para realizar la aserción (Propiedad de datos y el tipo de dato). • Si el cliente escoge el tipo de dato geométrico Ver sección "Tipo geométrico". • Si escoge cualquier otro tipo Ver sección "Otro tipo".	4. Termina el CU.
Sección "Tipo geométrico"	
	1. Se desactiva el campo valor, y activa el panel para seleccionar si lo que desea es una línea, un polígono o un punto.
2. Selecciona lo que desea representar.	3. Verifica que todos los campos estén seleccionados.
4. Presiona el botón (Representar).	5. Se desactiva la ventana principal, permitiendo seleccionar las coordenadas en el mapa.
6. Presiona el botón (Aceptar).	7. Se adiciona la aserción y la muestra. Volver al Flujo Normal de

	Eventos 4.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3. Muestra un cartel “Datos inválidos” y no se activa el botón (Representar). Volver a la Sección “Tipo geométrico” al flujo 2.
6. Presiona el botón (Cancelar).	6.1 Se cierra la ventana y termina el CU.
Prototipo de Interfaz	
	
Sección “Otro Tipo”	
	1 Se desactivan el panel que contiene las opciones punto, línea y polígono además del botón (Representar).
2 Selecciona la propiedad de datos e introduce el Valor.	3 Verifica que los campos estén llenos.
4 Presiona el botón (Aceptar).	5 Se adiciona la afirmación y la muestra. Volver al Flujo Normal de Evento 4

Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
	3. Muestra un cartel “Datos inválidos”.
4. Presiona el botón (Cancelar).	4.1 Se cierra la ventana y termina el CU.
Prototipo de Interfaz	
	
Pos-condiciones	El sistema muestra las aserciones existentes.

Para observar las demás descripciones diríjase al artefacto “Módulo OntoEdit Modelo del Sistema v1.0 nuevo”.

3.7. Conclusiones

Con el desarrollo de este capítulo se obtuvieron como resultado el modelo de dominio para un mejor entendimiento del módulo a realizar, se definieron los requisitos tanto funcionales como no funcionales para lograr mayor eficiencia y seguridad en la aplicación a construir. Se realizó el diagrama de caso de uso del sistema y el documento con la descripción de cada caso de uso para una mejor comprensión del funcionamiento del módulo.

Capítulo IV: Construcción de la solución propuesta.

4.1. Introducción

El presente capítulo está centrado en el proceso de desarrollo del módulo propuesto. En el mismo se incluye el modelo de diseño dentro del cual se presentarán los diagramas de clase del diseño, el modelo de despliegue además se expondrán los patrones de diseños utilizados para lograr una mayor calidad en el módulo a construir.

4.2. Arquitectura de software.

La arquitectura de software está compuesta por un conjunto de patrones necesarios para guiar la construcción de un software, permitiendo a los desarrolladores del software compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación.

4.2.1. Estilos Arquitectónicos.

Un estilo arquitectónico es una lista de diferentes componentes que describen los patrones o las interacciones a través de ellos. Un estilo afecta a toda la arquitectura de *software* y puede combinarse en la propuesta de solución (Almeira, y otros, 2007).

Existen diferentes estilos, el que se utiliza es el estilo de Llamada y Retorno. Este estilo pone mayor atención a la hora de modificar el sistema y en la escalabilidad del mismo ya que permite al diseñador del software construir una estructura de programa relativamente fácil de modificar y ajustar a escala. Se basa en la bien conocida abstracción de procedimientos/funciones/métodos. Las ventajas de Llamada y retorno son que al permitir la descomposición en módulos disminuye la complejidad, además persiguen escalabilidad (Almeira, y otros, 2007).

Las principales características de este estilo son:

- Hilo de control simple soportado por los lenguajes de programación.
- Usa una estructura implícita de subsistemas.
- Razonamiento jerárquico, cambios en una subrutina implican cambios en las subrutinas que hacen la invocación.

- Pretenden incrementar el desempeño distribuyendo el trabajo en múltiples procesadores.

4.2.2. Patrón arquitectónico.

Arquitectura Orientada a Objeto.

Los componentes de este estilo son los objetos, o más bien instancias de los tipos de datos abstractos. Se basa en los principios de encapsulamiento, herencia y polimorfismo. Son así mismo 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 además las interfaces están separadas de las implementaciones (Reynoso, y otros, 2004).

Arquitectura Basada en Componente.

Un componente es una pieza de código pre-elaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar. Este modelo posee características del modelo espiral, es evolutivo por naturaleza y exige un enfoque interactivo para la creación de software, además de que posibilita alcanzar un alto nivel de reutilización de software. Este patrón está dividido en 4 etapas la de análisis y comparación de proceso de desarrollo de software basado en componente, análisis arquitectural, identificación de componentes y especificación de los principales componentes.

4.2.3. Patrones de diseño.

4.2.3.1. Patrones de Principios Generales para Asignar Responsabilidades (GRASP).

Estos patrones son importantes a la hora de asignar responsabilidades si se quiere diseñar eficazmente el software orientado a objetos. También sirven de ayuda para entender y aplicar el razonamiento en el diseño de objeto. Existen diferentes patrones de diseño, dentro de ellos los que se utilizan para el desarrollo del módulo propuesto son:

Experto: El patrón experto en información indica que la responsabilidad de la implementación de un método debe ser en la clase que contiene la información necesaria para cumplir con esta responsabilidad. En el diagrama de clase del diseño perteneciente al caso de uso “Gestionar clase” se puede observar la utilización de este patrón, en la clase “GeoOntoClassManager”, ya que ella posee la información suficiente para listar, adicionar y eliminar una clase (ver figura. #3).

Creador: El patrón Creador guía la asignación de responsabilidades relacionadas con la creación de objetos, una tarea muy común. La intención básica del patrón Creador es encontrar un creador que

necesite conectarse al objeto creado en alguna situación. Eligiéndolo como el creador se favorece el bajo acoplamiento (Larman, 2002).

Bajo Acoplamiento: El patrón de Bajo Acoplamiento es un principio a tener en mente en todas las decisiones de diseño, debido a que el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, por lo que es necesario mantener un Bajo Acoplamiento para lograr que las clases sean más independientes, y así reducir el impacto del cambio.

Alta cohesión: Alta cohesión es cuando una clase tiene una responsabilidad moderada en un área funcional y colabora con otras clases para llevar a cabo las tareas. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla. Su alto grado de funcionalidad, combinada con una reducida cantidad de operaciones, también simplifica el mantenimiento y las mejoras. La ventaja que significa una gran funcionalidad también soporta un aumento de la capacidad de reutilización (Larman, 2002).

4.2.3.2. Patrones GOF (Pandilla de los Cuatro).

Los patrones GOF (Gang Of Four) son dirigidos al desarrollo de sistemas orientados a objetos y se encuentran divididos en tres grupos: los de creación, utilizados en la abstracción de cómo es creado un objeto, los de estructura que indican cómo se encuentran compuestas las clases y los de comportamiento, utilizados en la asignación de responsabilidades en las clases.

Patrón Fachada: Una Fachada es un objeto "front-end" que es el único punto de entrada para los servicios de un subsistema; la implementación y otros componentes del subsistema son privados y no pueden verlos los componentes externos. La Fachada proporciona Variaciones Protegidas frente a los cambios en las implementaciones de un subsistema. El patrón Fachada es sencillo y se utiliza ampliamente. Oculta un subsistema detrás de un objeto (Larman, 2002). Permitirá simplificar la interfaz del módulo que se propone y que será parte de la plataforma GeneSIG.

Patrón Singleton (Instancia única): Este patrón consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella. El mismo provee una única instancia global gracias a que la propia clase es responsable de crear la única instancia, además permite el acceso global a dicha instancia mediante un método de clase y declara el constructor de clase como privado para que no

sea instanciable directamente. Este patrón se pone de manifiesto en la clase ServerOntoEdit con el fin de garantizar que allá una única instancia para la conexión al Servidor de Servicios.

Patrón Command (Comando): se utiliza para encapsular un comando en un objeto para que pueda ser almacenado, pasado a métodos y devuelto igual que cualquier otro objeto. Este patrón se hace notorio en la clase AjaxHelper, la cual se encarga de encapsular las peticiones que se hacen al servidor sin la necesidad de conocer el contenido de las mismas.

4.3. Diagrama de clase del diseño.

El diagrama de clases del diseño se utiliza para describir de forma gráfica las especificaciones de las clases y de las interfaces en una aplicación. El mismo está compuesto por clases, atributos, asociaciones e interfaces. A continuación se representan los diagramas de clases del diseño perteneciente a los casos de usos Gestionar clase (ver Figura #3) y Adicionar aserciones de propiedades de objetos (ver Figura #4).

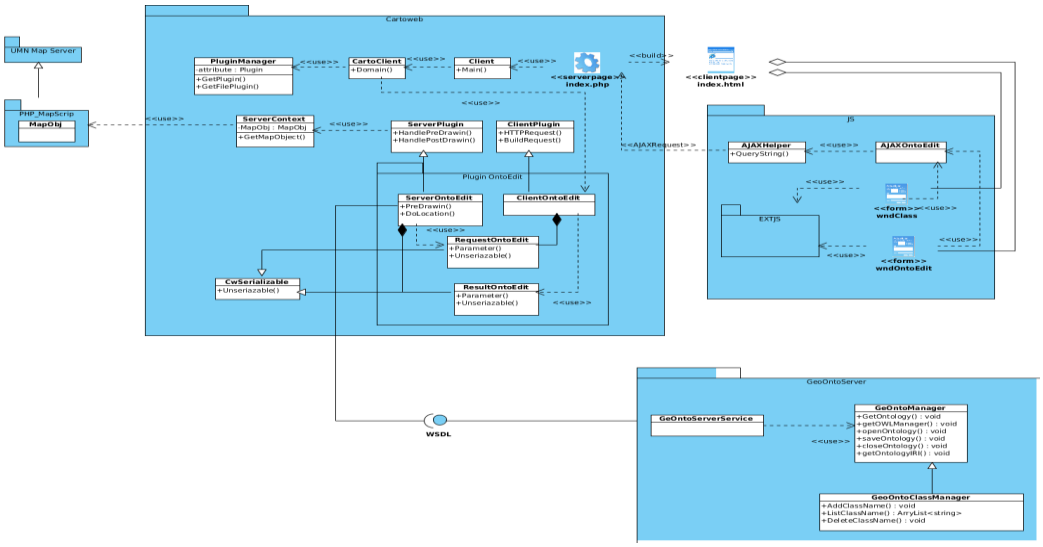


Figura # 3 Diagrama de Clase del Diseño (Gestionar clases).

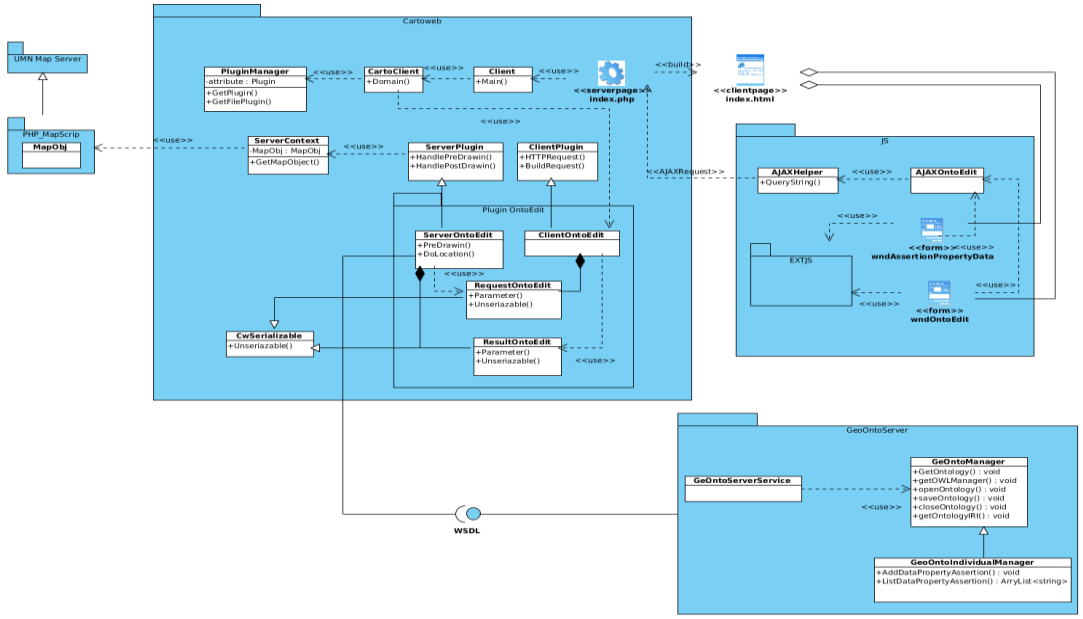


Figura # 4 Diagrama de Clase del Diseño (Adicionar aserciones de propiedades de datos).

Para mayor entendimiento de estos diagramas diríjase al artefacto Módulo para la plataforma de GeneSIG Modelo de Diseño.

4.4. Diseño de Base de Datos.

Con un correcto diseño de la base de datos se tiene mayor acceso a una información exacta y actualizada. Su objetivo fundamental es obtener un conjunto de datos y un conjunto de operaciones sobre ellos, que permitan satisfacer las necesidades de los usuarios.

Diagrama de clase persistente del sistema.

La información que manejará el módulo será persistente, por esto estará soportada por una base de datos relacional. Seguidamente se muestra el diseño de la base de datos del módulo propuesto a través de los diagramas de clases persistentes (ver Figura #5) y el esquema de la base de datos generados a partir de éste, es decir, el modelo Entidad-Relación (ver Figura #6).

Figura # 5 Diagrama de clase persistente del sistema.

Modelo de Entidad-Relación.

Figura # 6 Diagrama de Entidad-Relación.

4.5. Modelo de Despliegue.

Se utiliza para modelar el *hardware* utilizado en la implementación del sistema y las relaciones entre sus componentes. También se utiliza para visualizar la distribución de los componentes de *software* en los nodos físicos. Este modelo se compone de nodos, componentes y asociaciones. El mismo se ha diseñado para que un ingeniero en *software* pueda especificar la plataforma sobre la que se ejecuta el *software* del sistema (ver Figura #7).

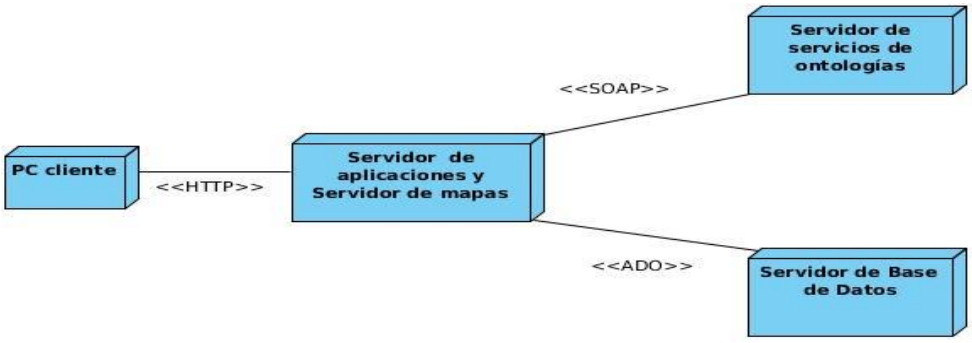


Figura # 7 Modelo de Despliegue.

4.6. Diagrama de componente.

El diagrama de componente o modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes. Además describe cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y de modularización disponibles en el entorno de implementación y en el lenguaje de programación utilizado, y la dependencia entre los componentes (ver Figura #8).

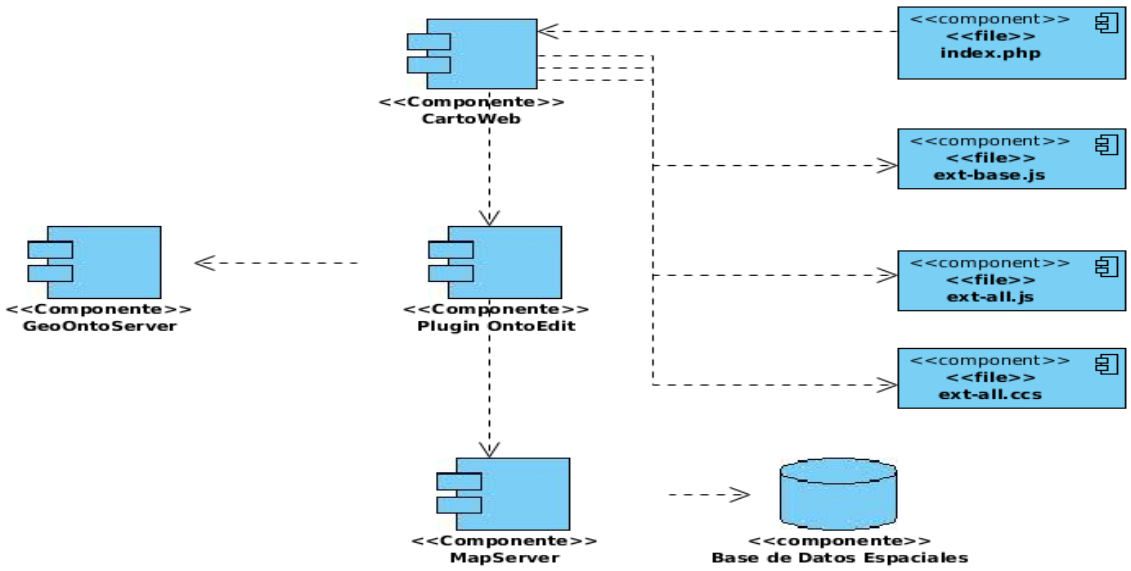


Figura # 8 Diagrama de Componente.

4.7. Pruebas.

Las pruebas son un proceso de ejecución de un programa que se realiza con la intención de detectar errores. Decir que los casos de prueba se realizan con el propósito de identificar y comunicar las condiciones que se llevarán a cabo en la prueba. Los mismos son necesarios para verificar que la aplicación tenga éxito y que cumpla con los requisitos del producto. Las pruebas realizadas al sistema serán de funcionalidad aplicando el método de caja negra y como se utiliza la metodología RUP serán basadas en casos de uso.

El método de caja negra consiste en pruebas que se les realiza a la interfaz de la aplicación del *software*, teniendo en cuenta solamente las entradas y las salidas de dicha aplicación. Las pruebas de caja negra tratan de encontrar diferentes errores como funciones incorrectas o faltantes, errores de interfaz, errores

en estructuras de datos o en acceso a bases de datos externas, errores de comportamiento o desempeño y errores de inicialización y término. Dentro de este método se pueden aplicar diferentes técnicas como la de particiones equivalentes, la misma se basa en una evaluación de clases de equivalencia para una condición de entrada. Estas clases de equivalencia representan un conjunto de estados válidos y no válidos para las condiciones de entrada.

Se definieron los casos de pruebas para los casos de uso Gestionar clase y Gestionar aserciones de propiedades de datos.

Caso de uso: Gestionar clases.

Descripción general: El caso de uso comienza cuando el usuario desea realizar alguna operación con las clases, ya sea adicionar, eliminar o listar clases.

Secciones a probar en el CU.

Tabla 4 Secciones a probar en el CU Gestionar Clase.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1: Gestionar clase	EC 1.1: Gestionar clase satisfactoriamente.	El sistema realiza la acción según la operación de gestión (Adicionar, Eliminar), seleccionada por el usuario.	Presionar la pestaña “Clases”. Seleccionar la sección correspondiente a la acción que desea realizar.
SC2: Adicionar clase	EC 2.1: Adicionar clase satisfactoriamente.	El sistema adiciona la clase y la muestra en el treepanel que se encuentra debajo de dicha opción.	Presionar la pestaña “Clases”. Seleccionar la sección “Adicionar”. Llenar los datos correspondientes. Presionar el botón (Aceptar).
	EC 2.2 Se dejan campos en blancos.	El sistema no activa la opción “Aceptar” hasta que se llene el campo	Presionar la pestaña “Clases”. Seleccionar la sección

		correspondiente.	"Adicionar". .No se activa el botón (Aceptar) hasta que se llene el campo correspondiente.
	EC 2.3 Seleccionar la opción "Cancelar".	El sistema cierra la ventana "Crear nueva OWL Class" y muestra en el treepanel que se encuentra debajo de la pestaña "Clases" las clases existentes.	Presionar la pestaña "Clases". Seleccionar la sección "Adicionar". Llenar los campos correspondientes. Seleccionar la opción "Cancelar".
SC3: Eliminar clase	EC 3.1: Eliminar clase satisfactoriamente	Muestra un mensaje de confirmación que permite elimina o no la clase del sistema.	Presionar la pestaña "Clases". Seleccionar la clase que se desea eliminar. Presionar el botón "Eliminar". Presionar el botón "Sí".
	EC 3.2: No se confirma la eliminación de la clase.	El sistema muestra el mensaje de confirmación y no se mantiene la clase por lo que se mantiene registrada en el sistema.	Presionar la pestaña "Clases". Seleccionar la clase que se desea eliminar. Presionar el botón "Eliminar". presionar el botón "No".

Descripción de variables.

Tabla 5 Descripción de variables del CU Gestionar Clase.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción

1	Clase	Campo de texto en la opción “Adicionar”. Treepanel para la opción “Eliminar”.	No	Este campo no debe estar en blanco.

Matriz de datos.

EC2: Adicionar clase.

Tabla 6 Matriz de datos de la EC2 Adicionar clase.

ID del escenario	Escenario	V1	Respuesta del Sistema	Resultado de la Prueba
EC 2.1	Adicionar clase satisfactoriamente.	V (Blanca)	El sistema adiciona la clase, y la muestra en el treepanel que se encuentra debajo de la pestaña “Clases”.	Satisfactorio
EC 2.2	Se dejan campos en blancos.	V ()	El sistema no activa loa opción “Aceptar”.	Satisfactorio

Matriz de datos.

EC3: Eliminar clase.

Tabla 7 matriz de datos de la EC2 Adicionar clase.

ID del escenario	Escenario	V1	Respuesta del Sistema	Resultado de la Prueba
EC 3.1	Eliminar clase satisfactoriamente	V (Blanca)	El sistema muestra un mensaje de confirmación para eliminar la clase.	Satisfactorio
EC 3.2:	No se confirma la eliminación del usuario.	*NA*	El sistema oculta el mensaje de confirmación y mantiene a la clase registrada en el sistema.	Satisfactorio

Caso de uso: Gestionar aserciones de propiedades de datos.

Descripción general: El caso de uso comienza cuando el usuario desea realizar alguna operación con las aserciones de propiedades de datos, ya sea adicionar, listar dichas propiedades.

Secciones a probar en el CU.

Tabla 8 Secciones a aprobar en el CU Gestionar aserciones de propiedades de datos.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo central
SC1 Gestionar aserciones de propiedades de datos.	EC 1.1: Gestionar aserciones satisfactoriamente.	El sistema realiza la acción según la operación de gestión (Adicionar, listar), seleccionada por el usuario.	Presionar la pestaña “Individuos”. Seleccionar la clase y el individuo. Seleccionar la opción correspondiente.
SC2: Adicionar afirmaciones de propiedades de datos.	EC 2.1: Adicionar afirmaciones de propiedades de datos satisfactoriamente.	El sistema muestra la ventana “Afirmaciones propiedades de datos” para que el usuario seleccione los datos.	Presionar la pestaña “Individuos”. Seleccionar la clase y el individuo. Presiona el botón “Adicionar”. Seleccionar los datos.
	EC 2.2 Seleccionar tipo de datos geométricos satisfactoriamente.	Se desactiva el campo valor, y activa el panel para seleccionar si lo que desea es una línea, un polígono o un punto para representarlo en el mapa.	Presionar la pestaña “Individuos”. Seleccionar la clase y el individuo. Presiona el botón “Adicionar”. Seleccionar los datos. Presionar el botón (Representar). Seleccionar en el mapa. Presionar el botón (Aceptar).
	EC 2.3: Campos sin seleccionar	Muestra un cartel “Datos inválidos” y no se activa el botón	Presionar la pestaña “Individuos”.

	satisfactoriamente.	(Representar).	Seleccionar la clase y el individuo. Presiona el botón “Adicionar”. Muestra un cartel y no se activa el botón (Representar).
	EC 2.4: Seleccionar otro tipo de datos satisfactoriamente.	Se activa el campo valor para que se introduzcan las coordenadas.	Presionar la pestaña “Individuos”. Seleccionar la clase y el individuo. Presiona el botón “Adicionar”. Seleccionar los datos y llenar el campo valor. Presionar el botón (Adicionar).
	EC2.5: Campos sin seleccionar satisfactoriamente.	No se activa el botón (Aceptar).	Presionar la pestaña “Individuos”. Seleccionar la clase y el individuo. Presiona el botón “Adicionar”. Muestra un cartel datos inválidos.
	EC 2.6: presionar el botón (Cancelar) satisfactoriamente.	Se cierra la ventana “Afirmaciones propiedades de datos” y se muestran las afirmaciones existentes.	Presionar la pestaña “Individuos”. Seleccionar la clase y el individuo. Presiona el botón “Adicionar”. Presionar el botón (Cancelar).

Descripción de variables.

Tabla 9 Descripción de variable en el CU Adicionar aserciones de propiedades de datos.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Seleccionar propiedades de datos.	Treepanel	No	Se debe selecciona una propiedad de datos.

2	Seleccionar tipo propiedad geométrica.	Lista desplegable	No	Se debe seleccionar una propiedad geométrica.
3	Seleccionar una figura geométrica.	Radiobuton.	No	Se debe seleccionar una figura geométrica.
4	Introducir el Valor	Campo de teto	No	Se debe introducir el valor.

Matriz de Datos.

Tabla 10 SC 2 Adicionar aserciones de propiedades de datos.

ID del escenario	Escenario	V1	V2	V3	V4	Respuesta del Sistema.	Resultado de la Prueba.
EC 2.2	Seleccionar tipo de datos geométricos satisfactoriamente.	V (Blanca)	V (geométrico)	V (Línea)	V NA	El sistema activa el botón (Representar).	Satisfactorio
EC 2.3	Se dejan campos sin seleccionar	V ()	V (-)	V (Línea)	V NA	El sistema no activa el botón (Representar).	Satisfactorio
		V (Blanca)	V ()	V (Línea)	V NA		
		V (Blanca)	V (-)	V ()	V NA		
EC 2.4	Seleccionar otro tipo de datos satisfactoriamente.	V (Blanca)	V NA	V NA	V (Valor)	El sistema activa el botón (Aceptar).	Satisfactorio
EC 2.5	Campos sin	V	V	V	V	El sistema no	Satisfactorio

	seleccionar satisfactoriamente.	() V (Blanca)	NA V NA	NA V NA	(Valor) V ()	activa el botón (Aceptar).	

Se realizaron 3 iteraciones de las pruebas de caja negra las cuales dieron como resultados:

En la primera iteración se obtuvo como resultado un total de 15 no conformidades de ellas 5 significativas relacionadas con errores de funcionalidad de la aplicación, las 10 restantes fueron no significativas asociadas a problemas ortográficos y de redacción.

En la segunda iteración disminuyen las no conformidades a un total de 4, siendo 2 significativa asociadas a problemas con las funcionalidades de la aplicación y 2 no significativas pertenecientes a errores ortográficos y de redacción.

En la tercera iteración de pruebas no se detectaron no conformidades, siendo el resultado de las pruebas realizadas al software satisfactorias. Por lo que se concluye que el módulo realizado cumple con los requisitos funcionales y con el objetivo propuesto.

4.8. Conclusiones.

En el presente capítulo se aplicaron los diferentes patrones de diseño y se elaboró un diagrama de clases del diseño que sirvió para mostrar la realización física de los requisitos funcionales y no funcionales. Mediante este diagrama se definieron los principales métodos y atributos a desarrollar durante el proceso de implementación del sistema. Se obtuvieron además los modelos de clases persistentes y el modelo de Entidad-Relación para describir la información que será almacenada.

La realización del diagrama de despliegue sirvió para modelar las relaciones físicas de los distintos nodos que componen el sistema y con la realización de las pruebas al *software* se obtuvo como resultado que la aplicación cumple de manera satisfactoria con las funcionalidades identificadas durante el proceso de desarrollo del *software*. Con las iteraciones de las pruebas se logra refinar las no conformidades encontradas, permitiendo que el sistema cuente con la calidad requerida.

Conclusiones Generales.

Con el estudio realizado para lograr la creación del módulo para edición de una ontología del dominio geográfico en la plataforma GeneSIG se llegó a las siguientes conclusiones:

- Las soluciones existentes no cumplen con el problema a resolver.
- Los artefactos que registran el desarrollo del módulo, permitirá futuras actualizaciones del mismo.
- El módulo Editor de Ontología logra solucionar el problema de la investigación científica.
- El sistema cumple con los requisitos funcionales y no funcionales, debido a que la herramienta es capaz de mostrar los resultados esperados.

Recomendaciones.

Con el objetivo de mejorar y concluir una versión más completa del módulo propuesto, en este documento se propone a modo de recomendaciones:

- Realizar otros tipos de pruebas al sistema.
- Obtener a partir de la Plataforma las relaciones topológicas entre los individuos.
- Permitir crear, importar y exportar una geo-ontología.

Glosario de términos.

Ajax: *Asynchronous JavaScript And XML* no es una tecnología porque es la unión de un conjunto de tecnologías para el desarrollo *Web* que se ejecutan en el cliente.

Caso de uso (CU): Secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema.

GeoCuba: Grupo empresarial dedicado a la elaboración, producción y venta de planos y mapas con diversos fines. Además se encarga de realizar estudios geográficos, de impacto ambiental e investigaciones científicas en el campo de la geociencia, desarrollando productos informativos terminados con alta calidad y fiabilidad.

Metodología: Conjunto de métodos que se siguen en una disciplina científica, en un estudio o en una exposición doctrinal.

PHP: El lenguaje PHP (acrónimo de "*PHP: Hypertext Preprocessor*") es un lenguaje de script interpretado utilizado para la generación de páginas *Web* dinámicas, embebido en páginas HTML y ejecutado en el servidor.

Polisemia: Pluralidad de significados de una palabra.

Plug-ins: Aditamento para agregar a un equipo.

SIGGO: Sistemas de Información Geográfica Gobernados por Ontologías las ontologías son una componente más, como lo es la base de datos temática o espacial que interviene y coopera de la misma manera para alcanzar los objetivos para los cuales fue creado el SIG.

Sinonimia: Coincidencia de significados entre dos o más vocablos.

Referencias Bibliográficas

Almeira, Adriana Sandra, Cavenago, Vania Perez y Rosanigo, Zulema Beatriz. 2007. *Referencia Arquitectura de Software: Estilos y patrones.* Argentina : s.n., 2007.

Barrera, Milagros, Núñez, Haydemar y Ramos, Esmeralda. 2012. *Ingeniería Ontológica.* Caracas: s.n., 2012.

Braken, I. & Webster, C. 1992. *Information technology in geography and planning.* Londers & New York: Routledge: s.n., 1992.

Casals, V. M., & Squires, E. R. (2013). *SLD069 Sistemas para el manejo de órdenes de servicio por los especialistas del centro de Ingenería Clínica y Electromedicina.* La Habana.

Corcho, Oscar, y otros. 2004. *Construcción de ontologías legales con la metodología METHONTOLOGY y la herramienta WebODE.* Madrid: s.n., 2004.

Cuervo, M. C., & Moreno, O. Y. (2005). *Herramientas Libres para modelar software.*

EVA. Entorno Virtual de Aprendizaje. 2000. 24 de 05 de 2013 <http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/RUP/RUP.pdf>.

García, María de la Nava Maroto. 2007. *Las relaciones conceptuales en la terminología de los productos cerámicos y su formalización mediante un editor de ontologías.* 2007. Tesis doctoral.

Garrete, Jose Luis Almazán, Palomino, Carmen y Caba., Hilda Araceli. 2009. *Sistema de Información Geográfica en la Gestión Integral del Litoral.* Madrid : s.n., 2009.

Guarino, N. 1998. *Formal Ontology and Information Systems.* Amsterdam: IOS Press, 1998.

Henst, Christian Van Der. 1997. Maestros del web. [En línea] 1997. [Citado el: 19 de 11 de 2012.] http://www.maestrosdelweb.com/editorial/web-semantica-y-sus-principales-caracteristicas/.

Hanze, Xiomara Beatriz Heredia y Guerrero, Santa Judith Vera. 2008. *Estudio de PHP Y MYSQL para el desarrollo del portal Web para el municipio de Esmeraldas.* Escuela Superior Politécnica de Chimborazo. Esmeraldas: s.n., 2008. Tesis de grado

Investigación. Stanford Center for Biomedical Informatics. 2011. sitio de protégé: WebProtégé: Un editor de ontologías en colaboración y herramientas de adquisición de conocimientos para la Web, Web Semántica Journal. [En línea] 2011. [Citado el: 14 de 01 de 20013.] http:// protege.stanfo.

Bibliografía.

Jalón, Javier García de, Mingo, Iñigo y Calleja, Jesús. 2002. *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2002.

Kashyap, Vipul y Sheth, Amit. 1996. *Semantic Heterogeneity in Global Information System: The Role of Metadata, Context and Ontologies, in Coorperative Information Systems: Current Trends And Directions.* [ed.] Schlageter, G M. P. Papazoglou. London: Academic Press, 1996.

Klien, Eva y Michael Lutz. 2005. *The Role Spatial Relations in Automating Semantic Annotation of Geodata.* 2005. Vol. 3693.

Larman, Craig. 2002. *UML y Patrones una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2002.

López, Gerardo Sarabia. 2008. *Búsqueda y ponderación de información contenida en Bases de datos Espaciales, utilizando jerarquía. Centro de Investigación en computación*. México: s.n., 2008. Tesis.

Martinez, Rafael. 2009. postgreSQL-es. [En línea] 2009. [Citado el: 25 de 02 de 2013.] http://www.postgresql.org.es/sobre_postgresql.

Navarro, Juan Francisco García. 2012. *Analítica Visual aplicada a la Ingeniería de Ontologías.* Salamanca: s.n., 2012.

Oracle Corporation. 2013. NetBeans. [En línea] 2013. [Citado el: 28 de 02 de 2013.] http://netbeans.org/features/index.html.

Pelechano, Vicente. 2005. *Servicios Web. Estándares, Extensiones y Perspectivas de Futuro.* Valencia: s.n., 2005.

Ramsey, Paul. *Mappinggis.* [En línea] [Citado el: 25 de 02 de 2013.] http://mappinggis.com/2012/09/19/por-que-utilizar-postgis/.

Rayo, María Belén, y otros. 2011. *Gestión de la composición semántica de servicios web para el dominio de turismo.* Granada: s.n., 2011.

Reynoso, Carlos y Kiccillof, Nicolás. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

Ricardo, Edgar Rojas, y otros. 2012. *Aplicación informática para gestionar repositorios, establecer semejanzas y caracterizar ontologías.* La Habana: s.n., 20 de 4 de 2012, Revista Cubana de Ciencias Informáticas.

Romero, Julio César Vizcarra. 2010. *Repositorio Semántico de datos espaciales.* México: s.n., 2010.

Rubal, Martin. 2005. *Mappings for cognitive Semantic Interoperability.* Alemania: s.n., 2005.

Bibliografía.

Santos, Rafael Oliva y Garea-Llano, Eduardo. 2007. *Geo-ontologies as Integration Structuresof Knowledge, Data, and Metadata.* Mexico: s.n., 2007. Second International Conference on Geospatial Semantics.

Santos, Rafael Oliva, Llano, Eduardo Garea y Pérez., Francisco Maciá. 2009. *Ideas para una arquitectura de persistencia basada en geo-ontologías para anotaciones semánticas de datos geográficos.* La Habana: s.n., 2009.

Santos, Rafael Oliva, Pérez, Francisco Maciá y Llano, Eduardo Garea. 2009. *Esquema de un modelo de Integración de datos, metadatos y conocimiento geográfico.* La Habana: s.n., 2009.

Shvaiko, Pavel y Euzenat, Jerome. 2007. *Ontology Matching.* 2007. Vol. 5332.

Soltero, Alonso Pérez, Valenzuela, Mario Barceló y Sánchez, Guzmán G. Alfonso. 2006. *La web semántica como apoyo a la gestión del conocimiento y al modelo organizacional.* 12. Chile: s.n., 2006.

Tim Berners-Lee; James Hendler; Ora Lassila. **Web, The Semantic. 2001.** 2001.

Uschol, M. 1996. *Building Ontologies: Toward a Unified Methodology.* Cambridge: s.n., 1996. Proceedings of 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems.

Valdés, Damián Pérez. 2007. *Web semántica y sus principales características.* [En línea] 26 de 6 de 2007. [Citado el: 19 de 11 de 2012.]

http://www.maestrosdelweb.com/editorial/web-semantica-y-sus-principales-caracteristicas./.

Van Heijst, Schreiber, Wielinga. 1997. *Using Explicit Ontologies in KBS Development.* Italia: s.n., 1997.

Voronisky, Francisco Vera, Llano, Eduardo Garea. 2009. *Alineamiento de ontologías en el dominio geoespacial.* La Habana: s.n., 2009.

Bibliografía Consultada

Almeira, Adriana Sandra, Cavenago, Vania Perez y Rosanigo, Zulema Beatriz. 2007. *Referencia Arquitectura de Software: Estilos y patrones.* Argentina : s.n., 2007.

Barrera, Milagros, Núñez, Haydemar y Ramos, Esmeralda. 2012. *Ingeniería Ontológica.* Caracas: s.n., 2012.

Braken, I. & Webster, C. 1992. *Information technology in geography and planning.* Londers & New York: Routledge: s.n., 1992.

Casals, V. M., & Squires, E. R. (2013). *SLD069 Sistemas para el manejo de órdenes de servicio por los especialistas del centro de Ingenería Clínica y Electromedicina.* La Habana.

Cibulsky, Indalecio Fructuoso Bezos. 2006. *Ejemplos de Ontologías – Ontologías aplicadas a la Información Geográfica.* Santa Fe Argentina: s.n., 2006.

Corcho, Oscar, y otros. 2004. *Construcción de ontologías legales con la metodología METHONTOLOGY y la herramienta WebODE.* Madrid: s.n., 2004.

Cuervo, M. C., & Moreno, O. Y. (2005). *Herramientas Libres para modelar software.*

Díaz, Alvaro Antonio Penroz. 2005. *Graphical User Interface (GUI) para el programa servidor de mapas MapServer 4.6.1.* 2005. Tesis.

EVA. Entorno Virtual de Aprendizaje. 2000. 24 de 05 de 2013

<http://eva.uci.cu/file.php/161/Documentos/Materiales_complementarios/UD_1_Procesos/Metodologias/RUP/RUP.pdf>.

García, María de la Nava Maroto. 2007. *Las relaciones conceptuales en la terminología de los productos cerámicos y su formalización mediante un editor de ontologías.* 2007. Tesis doctoral.

Garrete, Jose Luis Almazán, Palomino, Carmen y Caba., Hilda Araceli. 2009. *Sistema de Información Geográfica en la Gestión Integral del Litoral.* Madrid : s.n., 2009.

Guarino, N. 1998. *Formal Ontology and Information Systems.* Amsterdam: IOS Press, 1998.

Glassfish.softonic.com. [En línea] [Citado el: 20 de 02 de 2013.] http://glassfish.softonic.com.

Henst, Christian Van Der. 1997. Maestros del web. [En línea] 1997. [Citado el: 19 de 11 de 2012.] http://www.maestrosdelweb.com/editorial/web-semantica-y-sus-principales-caracteristicas/.

Bibliografía.

Hanze, Xiomara Beatriz Heredia y Guerrero, Santa Judith Vera. 2008. *Estudio de PHP Y MYSQL para el desarrollo del portal Web para el municipio de Esmeraldas.* Escuela Superior Politécnica de Chimborazo. Esmeraldas: s.n., 2008. Tesis de grado

Investigación. Stanford Center for Biomedical Informatics. 2011. sitio de protégé: WebProtégé: Un editor de ontologías en colaboración y herramientas de adquisición de conocimientos para la Web, Web Semántica Journal. [En línea] 2011. [Citado el: 14 de 01 de 20013.] http:// protege.stanfo.

Kashyap, Vipul y Sheth, Amit. 1996. *Semantic Heterogeneity in Global Information System: The Role of Metadata, Context and Ontologies, in Coorperative Information Systems: Current Trends And Directions.* [ed.] Schlageter, G M. P. Papazoglou. London: Academic Press, 1996.

Klien, Eva y Michael Lutz. 2005. *The Role Spatial Relations in Automating Semantic Annotation of Geodata.* 2005. Vol. 3693.

Larman, Craig. 2002. *UML y Patrones una introducción al análisis y diseño orientado a objetos y al proceso unificado.* 2002.

López, Gerardo Sarabia. 2008. *Búsqueda y ponderación de información contenida en Bases de datos Espaciales, utilizando jerarquía. Centro de Investigación en computación*. México: s.n., 2008. Tesis.

Martinez, Rafael. 2009. postgreSQL-es. [En línea] 2009. [Citado el: 25 de 02 de 2013.]

http://www.postgresql.org.es/sobre_postgresql.

Minera, Francisco. 2010. *PHP6 sitios dinámicos con el lenguaje más dinámico.* Argentina: s.n., 2010.

Navarro, Juan Francisco García. 2012. *Analítica Visual aplicada a la Ingeniería de Ontologías.* Salamanca: s.n., 2012.

OGC. OGC. [En línea] [Citado el: 03 de 05 de 20013.]

http://www.opengeospatial.org/standards/kml .

OGC. OGC. [En línea] [Citado el: 03 de 05 de 2013.]

http://www.opengeospatial.org/standards/gml .

Oracle Corporation. 2013. NetBeans. [En línea] 2013. [Citado el: 28 de 02 de 2013.]

http://netbeans.org/features/index.html.

Osiris Next. [En línea] http://on.cs.unibas.ch/owls-api/.

Pelechano, Vicente. 2005. *Servicios Web. Estándares, Extensiones y Perspectivas de Futuro.* Valencia: s.n., 2005.

Ramsey, Paul. [En línea] [Citado el: 25 de 02 de 2013.]

http://mappinggis.com/2012/09/19/por-que-utilizar-postgis/.

Rayo, María Belén, y otros. 2011. *Gestión de la composición semántica de servicios web para el dominio de turismo.* Granada: s.n., 2011.

Reynoso, Carlos y Kiccillof, Nicolás. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* Buenos Aires : s.n., 2004.

Ricardo, Edgar Rojas, y otros. 2012. *Aplicación informática para gestionar repositorios, establecer semejanzas y caracterizar ontologías.* La Habana: s.n., 20 de 4 de 2012, Revista Cubana de Ciencias Informáticas.

Romero, Julio César Vizcarra. 2010. *Repositorio Semántico de datos espaciales.* México: s.n., 2010.

Rubal, Martin. 2005. *Mappings for cognitive Semantic Interoperability.* Alemania: s.n., 2005.

Ruiz, Miguel Jesús Torres. 2007. *Representación Ontológica Basada en Descriptores Semánticos Aplicada a Objetos Geográficos.* México: s.n., 2007. Resumen de tesis doctoral.

Santos, Rafael Oliva y Garea-Llano, Eduardo. 2007. *Geo-ontologies as Integration Structuresof Knowledge, Data, and Metadata.* Mexico: s.n., 2007. Second International Conference on Geospatial Semantics.

Santos, Rafael Oliva, Llano, Eduardo Garea y Pérez., Francisco Maciá. 2009. *Ideas para una arquitectura de persistencia basada en geo-ontologías para anotaciones semánticas de datos geográficos.* La Habana: s.n., 2009.

Santos, Rafael Oliva, Pérez, Francisco Maciá y Llano, Eduardo Garea. 2009. *Esquema de un modelo de Integración de datos, metadatos y conocimiento geográfico.* La Habana: s.n., 2009.

Sierra, Virginia y Alvarez, Carlos. *Metodología de la Investigación Científica .*

Shvaiko, Pavel y Euzenat, Jerome. 2007. *Ontology Matching.* 2007. Vol. 5332.

Soltero, Alonso Pére, Valenzuela, Mario Barceló y Sánchez, Guzmán G. Alfonso. 2006. *La web semántica como apoyo a la gestión del conocimiento y al modelo organizacional.* 12. Chile: s.n., 2006.

The OWL API. [En línea] http://owlapi.sourceforge.net/.

Tim Berners-Lee; James Hendler; Ora Lassila. **Web, The Semantic. 2001.** 2001.

Uschol, M. 1996. *Building Ontologies: Toward a Unified Methodology.* Cambridge: s.n., 1996. Proceedings of 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems.

Valdés, Damián Pérez. 2007. Web semántica y sus principales características. [En línea] 26 de 6 de 2007. [Citado el: 19 de 11 de 2012.]

http://www.maestrosdelweb.com/editorial/web-semantica-y-sus-principales-caracteristicas./.

Van Heijst, Schreiber, Wielinga. 1997. *Using Explicit Ontologies in KBS Development.* Italia: s.n., 1997.

Voronisky, Francisco Vera, Llano, Eduardo Garea. 2009. *Alineamiento de ontologías en el dominio geoespacial.* La Habana: s.n., 2009.