UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 6

Título: Sistema de Catálogo para la Gestión de Mapas en la Web Versión 2.0

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autora: **Grethell Castillo Reyes**

Tutor: **MsC.Romanuel Ramón Antunez**

Co-tutor: **MsC.Yusnier Valle Martínez**

Mayo 2012

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste, firmo la presente declaración jurada de autoría en La Habana a los __ días del mes de __ del año ___.

_____	_____
Firma de la Autora	Firma del Tutor

DATOS DE CONTACTO

TUTOR: Romanuel Ramón Antunez. (email: rramon@uci.cu). Ingeniero en Ciencias Informáticas (2008), Instructor (2010), Máster en Informática Aplicada (2011).

CO-TUTOR: Yusnier Valle Martínez. (email: yvm@uci.cu). Licenciado en Ciencia de la Computación (2004), Asistente (2008), Máster en Informática Aplicada (2009).

AGRADECIMIENTOS

A la Universidad de las Ciencias Informáticas, fruto de la Revolución y de Fidel, por darme la oportunidad de formarme como profesional.

A mis tutores Romanuel y Valle por guiarme en todo momento, por estar pendiente de cada paso dado, por su apoyo y recomendaciones y por su esfuerzo para lograr que los resultados fueran los mejores.

A mis abuelitos, aunque la vida me los arrebatara antes de que pudieran verme convertida en lo que soy hoy, a ustedes les debo todo. Gracias por cuidarme, por educarme, por amarme siempre como a una hija. Siempre los llevo en mi corazón.

A mi papá que ha sido mi ejemplo y mi motor impulsor. El que yo haya llegado hasta aquí es fruto de su esfuerzo y dedicación. Ahora es mi turno de seguir tus pasos. Te quiero.

A mis tías y tíos por ser para ustedes una hija, por todo el amor que me han dado. A mis hermanos, y a mi familia en general por estar siempre pendiente de mí, y de mi carrera universitaria.

A mi compañero Guillermo por todo el apoyo que me ha brindado. Gracias por tener siempre tus brazos extendidos para mi, por estar presente en los momentos difíciles, por todo el amor que me has brindado. Te quiero.

A Hamilé, Eduardo y Alex por toda la ayuda que me han dado, por su apoyo incondicional en todo momento, por ser una familia tan especial, por todo su amor y apoyo, los quiero. Gracias por todo.

A mi hermanita Alexaidis por ser amiga, consejera y hasta mi psicóloga, sin ti estos 5 años hubiesen sido muy difíciles. A toda tu familia por tener los brazos siempre abiertos para mi.

A los muchachos del equipo de trabajo de GeneSIG. A Armando, Rangel, Rolando y Yampier por su ayuda incondicional en todo momento.

A mis amigas inseparables Amelia y Jessica por estar siempre pendientes de mi y por estar en mi camino para apoyarme.

A todas las amistades ganadas en estos 5 años, por haber sido como una familia muy grande, porque a pesar de la distancia los llevaré en el corazón. Gracias a todos, en especial a Ruby, Pacheco, Marcelino, Enrique, Yuri y Lisdany, los quiero.

A todos los que de una forma u otra han contribuido a mi formación como profesional y humano en general.

DEDICATORIA

A la memoria de mis abuelos Elba y Gilberto, por ser mis padres y amigos de siempre, los llevo en mi corazón.

A mi papá Emilio, por ser mi guía y apoyo, por ser quien me impulsa día a día, te quiero.

A mis tías Elida, Leonor, Emilia, Elsa y a mi tío Iluminado, por quererme como a una hija más, por ese amor tan grande que me han brindado, los amo.

A la memoria de mi mamá María Elena.

RESUMEN

La plataforma soberana GeneSIG, diseñada para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada, actualmente no personaliza la información geográfica que representa, aspecto de gran importancia para garantizar la seguridad de la información. Por otro lado, en la edición del fichero mapfile a través del catálogo de mapas LiberMaps 1.0, existen procedimientos con cierto grado de similitud y que pueden llegar a ser engorrosos por la cantidad de campos de información a tener en cuenta. Además, en el catálogo se almacenan gran cantidad de mapas, por lo que en varias ocasiones se le hace difícil al usuario encontrar el mapa que desea editar. Para darle solución a estos problemas, mediante la presente investigación se implementó la versión 2.0 del catálogo de mapas LiberMaps, que cuenta con los mecanismos necesarios para integrarse con Sistemas de Información Geográfica, como lo es GeneSIG y de esta manera brindarle la personalización de la información teniendo en cuenta perfiles definidos por usuarios y roles. Además, permite la realización de búsquedas y cuenta con funcionalidades que agilizan el proceso de creación de los mapas. La implementación del sistema se llevó a cabo en un ambiente web, utilizando tecnologías libres. Las pruebas realizadas a la aplicación tuvieron resultados satisfactorios, lo que garantiza que la herramienta contribuya a la implementación de la Infraestructura de Datos Espaciales de la República de Cuba.

PALABRAS CLAVES: Ambiente web, Catálogo, GeneSIG, LiberMaps, Mapas, Mapfile, Sistemas de Información Geográfica, Tecnologías libres.

Índice general

Índice de figuras

Índice de tablas

Introducción

En el naciente siglo XXI la humanidad ha estado inmersa en un completo auge del desarrollo de las TIC[1]. En años recientes en el país los esfuerzos se han ido enfocando en el desarrollo de SIG[2], así como en la implementación de la IDERC[3] [Delgado, 2005]. Estos sistemas han sido capaces de evolucionar de forma rápida y similar al crecimiento de la tecnología, contribuyendo así al desarrollo de la sociedad de la información.

Los SIG son una tecnología reciente fundamentada en el uso de datos espaciales[4] y que se aplica cada vez más a un mayor número de disciplinas [Llopis, 2008]. Han sido una revolución en el campo de la geografía y son muy útiles en diversas áreas como la cartografía[5], la navegación, la aviación, en la gestión de recursos y de activos, la gestión de información geológica y en general son herramientas de apoyo para la toma de decisiones en varias entidades de todo el mundo, pues son sistemas que permiten disponer rápidamente de información necesaria para resolver problemas de forma inmediata.

En un SIG se usan herramientas de gran capacidad de procesamiento gráfico y alfanumérico. Las mismas van dotadas de procedimientos para captura, almacenamiento, análisis y visualización de la información georreferenciada [Salinas, 2007]. Una de estas herramientas son los catálogos de información geográfica, también necesarios para la construcción de una IDE[6]. Estos permiten que usuarios o aplicaciones de software puedan

[1] Tecnología de la Información y la Comunicación.
[2] Sistema de Información Geográfica.
[3] Infraestructura de Datos Espaciales de la República de Cuba.
[4] Información con componente espacial en el sentido geográfico. Entidades u objetos abstraídos del espacio geográfico real.
[5] Ciencia que estudia las formas y técnicas de representación de la geografía sobre un mapa.
[6] Infraestructura de Datos Espaciales.

buscar la información existente en algún lugar dentro de un entorno computacional distribuido. Los catálogos son la solución para publicar descripciones de datos geográficos mediante métodos estándares que posibilitan la realización de búsquedas a lo largo de múltiples servidores.

Por el gran interés que los SIG han cobrado a nivel mundial y en el país, un grupo de trabajo del Centro GEYSED[7] perteneciente a la facultad 6 de la UCI[8], en conjunto con miembros de la empresa GEOCUBA y de la UCID[9] han desarrollado un sistema para la creación de SIG, la plataforma soberana GeneSIG, desarrollada completamente con herramientas y tecnologías libres cumpliendo con la política de migración a software libre y de soberanía tecnológica que impulsa el país. Cuenta con las funcionalidades comunes de estos sistemas, que resultan esenciales en el comportamiento básico de los mismos.

La plataforma GeneSIG utiliza como servidor de mapas a Mapserver, entorno de desarrollo de código abierto. Mapserver permite la creación de aplicaciones SIG con el fin de visualizar, consultar y analizar información geográfica a través de la red mediante la tecnología IMS[10] [Ledea, 2010]. El núcleo de este servidor es un fichero con extensión ".map", denominado mapfile, que contiene toda la información sobre el mapa que representa el SIG, dígase los objetos que lo componen, sus atributos y relaciones.

De manera paralela a la realización de GeneSIG, se crearon dos sistemas. El primero de ellos, LiberMaps 1.0, brinda un conjunto de funcionalidades para la creación y edición del fichero mapfile de Mapserver, y gestiona la personalización de la información geográfica. El segundo es el SyGMe[11], módulo implementado para el catálogo de mapas LiberMaps, que tiene como función principal la gestión de metadatos geográficos[12]. Además, permite documentar la información geográfica y facilita la organización y localización de los datos.

En la edición de los ficheros ".map" a través de LiberMaps 1.0, existen procedimientos con cierto grado de similitud y que pueden llegar a ser engorrosos por la cantidad de

[7]Centro de Desarrollo Geoinformática y Señales Digitales.
[8]Universidad de Ciencias Informáticas.
[9]Unidad de Compatibilización e Integración de Software para la Defensa.
[10]Internet Map Server.
[11]Sistema de Gestión de Metadatos.
[12]Información que caracteriza los datos espaciales.

campos de información a tener en cuenta. Por ejemplo, a la hora de crear varios ficheros con similares datos el proceso se torna un poco repetitivo, disminuyendo de esta manera la usabilidad del sistema. Por otro lado, en el catálogo se almacenan gran cantidad de mapas, por lo que en varias ocasiones se le hace difícil al usuario encontrar el mapa que desea editar. Uno de los servicios más importantes que debe existir en una IDE, según [OGC, 2007], es el servicio de búsqueda de información geográfica a través de metadatos. En este caso, el SyGMe permite gestionar colecciones de metadatos a través de los cuales se documenta y localiza la información geográfica, pero el catálogo de mapas LiberMaps 1.0 no cuenta con los mecanismos necesarios para poder acceder a ellos y de esta manera materializar el servicio de búsqueda que el mismo requiere.

Por último, y no menos importante, actualmente los usuarios de la plataforma GeneSIG visualizan de igual manera toda la información representada por un mapa, siendo este aspecto de gran importancia cuando se trata del manejo de información sensible y que no todos los usuarios deben conocer. Por esta razón, la plataforma necesita contar con un mecanismo que permita personalizar la información y el acceso a los mapas, de acuerdo a perfiles definidos por usuarios y roles.

A partir de la situación planteada se detecta el siguiente **problema de la investigación**:

¿Cómo facilitar la gestión de mapas en la web para SIG basados en Mapserver?

Para darle solución al problema planteado se define como **objeto de estudio** el proceso de catalogación de mapas.

Planteándose como **objetivo general** desarrollar un sistema capaz de localizar, editar, importar y exportar ficheros de configuración de mapas para Mapserver.

Por lo que se define como **campo de acción** el proceso de catalogación de mapas en la web para aplicaciones basadas en Mapserver.

En función del objetivo planteado se infiere la siguiente **idea a defender**: El desarrollo de un sistema capaz de localizar, editar, importar y exportar ficheros de configuración de mapas para Mapserver, facilitará la gestión de mapas en la web para SIG basados en Mapserver.

Para darle cumplimiento al objetivo antes planteado se han definido las siguientes **tareas** que guían el proceso de investigación:

1. Identificación de las variantes de solución existentes en el tratamiento del problema planteado mediante la realización de un estudio de los referentes teóricos - prácticos que preceden la realización de esta investigación.
2. Caracterización de las tecnologías y herramientas en las que se apoya la solución del sistema justificando su elección.
3. Realización del análisis y diseño del sistema.
4. Implementación del sistema.
5. Validación del sistema.
6. Elaboración de una propuesta de integración entre LiberMaps y sistemas desarrollados por la línea de los SIG.
7. Elaboración de la documentación técnica del proceso de desarrollo del sistema con vistas a su socialización.

Se espera obtener como **posible resultado**: El catálogo de mapas LiberMaps en su versión 2.0 integrado a los sistemas GeneSIG y SyGMe.

Para darle cumplimiento a las tareas anteriores se utilizarán los siguientes **métodos de la investigación científica** [Álvarez, 2000]:

Métodos teóricos: Se utiliza el método **analítico - sintético** que permite consultar la bibliografía especializada en cuanto al tema abordado e identificar elementos claves que contribuyan a la solución del problema científico planteado, permite sintetizar conceptos que ayudarán a comprender la solución del problema; el método **histórico - lógico** para el estudio crítico de los trabajos anteriores que constituyen referentes teórico - prácticos en el tratamiento del problema planteado, y tomarlos como base de comparación con los resultados alcanzados; el método de **modelación** para realizar diagramas y representar

todo el proceso de la gestión de los ficheros mapfile en el catálogo LiberMaps facilitando un mejor entendimiento de la solución a implementar.

Métodos empíricos: Se utiliza el método de **análisis documental** en la revisión de la literatura especializada para consultar la información necesaria en el proceso de investigación.

Estructura del documento

El presente documento se encuentra dividido en tres capítulos:

Capítulo 1. Fundamentos de la catalogación de mapas : Se describen los conceptos que sustentan la investigación con el objetivo de brindar elementos esenciales que contribuyan a un mejor entendimiento de la solución que se propone. Se hace un análisis más profundo de la situación problemática y el objeto de estudio de la investigación. Se justifica la elección de las tecnologías y herramientas existentes actualmente que son necesarias y contribuyen al desarrollo de la solución.

Capítulo 2. Descripción de la solución propuesta: Se describen los principales artefactos relacionados con la modelación del dominio y el levantamiento de requisitos, según indica la metodología seleccionada para guiar el proceso de desarrollo.

Capítulo 3. Implementación y validación: Se presentan los artefactos ingenieriles relacionados con el diseño, implementación y validación del sistema. Se abunda sobre el funcionamiento de los frameworks seleccionados y se presenta una estrategia de arquitectura para la integración del catálogo con otros sistemas desarrollados por la línea de los SIG.

Capítulo 1

Fundamentos de la catalogación de mapas

Con el objetivo de facilitar la comprensión del alcance de la investigación, en el presente capítulo se exponen conceptos fundamentales asociados al dominio del problema planteado que sustentan la investigación. Además, se realiza un análisis detallado de los referentes teórico - prácticos que preceden la realización de este trabajo y que contribuyen a esclarecer su objeto de estudio. Se da una descripción más detallada de la situación problemática para lograr un mejor entendimiento de la necesidad que existe. Además se presenta la selección de las herramientas y tecnologías necesarias para el desarrollo de la solución, justificando su elección.

1.1. Conceptos asociados al dominio del problema

1.1.1. Dato espacial

Un dato espacial es un dato que además de tener una serie de características como color, forma y otras, posee también una ubicación en el espacio (coordenadas x, y, z) [Costa, 2010]. Los datos espaciales se distinguen por tres componentes esenciales: la localización (expresada a través de las coordenadas), la relación (expresada a través de la

topología[1]) y la descripción (expresada a través de atributos propios)[Guevara, 2001].

En general, los datos espaciales son sinónimo de información geográfica y constituyen uno de los elementos más importantes en los que se apoya la toma de decisiones en la actualidad, ya que gran parte de la información que se maneja en las entidades y empresas a nivel mundial, posee características espaciales.

1.1.2. Infraestructura de Datos Espaciales

Para una mayor eficiencia en la recolección, administración, acceso, entrega y utilización de los datos espaciales se crean las IDE. [Delgado, 2009] plantea que una infraestructura está formada por redes, y estas, a su vez por sistemas, y que por tanto una IDE es una infraestructura de información, es compartir información geográfica en la web. Es el medio en que diversos proveedores comparten información y a donde acuden los usuarios a consumirla.

Se llega a la conclusión de que una IDE es el conjunto de datos geográficos y de servicios, estándares, y tecnologías que permiten procesar, almacenar, difundir y optimizar el manejo de dichos datos, útiles para la toma de decisiones.

1.1.3. Sistema de Información Geográfica

En una IDE existen un conjunto de tecnologías que permiten procesar, almacenar, distribuir y mejorar la utilización de la información geográfica. Una de estas tecnologías es llamada SIG.

Son varios los conceptos que se tienen en la actualidad sobre los SIG, algunos se refieren a sus componentes y funciones, otros a su capacidad para la toma de decisiones, otros a su función como base de datos, pero llegando a un consenso entre todos se puede resumir que un SIG se define como cualquier sistema de información capaz de integrar, almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada. En un sentido más genérico, los SIG son herramientas que permiten a los usuarios crear

[1]Disciplina matemática que estudia las propiedades de los espacios topológicos.

consultas interactivas, analizar la información espacial, editar datos, mapas y presentar los resultados de todas estas operaciones [Bravo, 2000].

Resumiendo, se entiende como SIG al conjunto de información geográfica y socio-económica y a las herramientas informáticas, ya sea hardware y software que permiten la gestión de dicha información y la visualización, consulta e impresión de los mapas relacionados.

1.1.4. Catálogo de mapas

Una de las razones fundamentales de ser de un SIG es la visualización de la información geográfica que maneja a través de mapas. Según la [RAE, 2010] un mapa es una representación geográfica de una parte de la superficie terrestre, en la que se presenta información relativa a una ciencia determinada. Es construido mediante gráficas y métricas, de forma bidimensional o esférica.

En un SIG se usan herramientas que van dotadas de gran capacidad para el procesamiento de información y que además son fundamentales en la construcción de una IDE, una de estas herramientas son los catálogos. Un catálogo es una relación ordenada en la que se incluyen o describen de forma individual libros, documentos, personas, objetos, etc., que están relacionados entre sí [RAE, 2010].

Teniendo en cuenta el concepto dado anteriormente de mapa y de catálogo, se llega a la conclusión de que un catálogo de mapas es una relación ordenada en la que se incluyen o describen de forma individual los mapas.

1.1.5. Tecnología webmapping

Existen numerosas tecnologías además de los SIG que permiten el manejo de información geográfica. La tecnología webmapping es aquella que hace referencia a las herramientas que facilitan la manipulación de información geoespacial a través de la web y que se enmarcan en la implantación de servicios y productos cartográficos en Internet [Padrón, 2001]. A este tipo de herramienta se le conoce como Sistemas Webmapping o

Servidores de Mapas.

1.2. Servidor de mapas Mapserver

Existen numerosos servidores de mapas que pueden ser utilizados para manipular información geográfica a través de la web. Mapserver es uno de los servidores frecuentemente utilizados en su variedad de versiones.

Mapserver es un motor de renderización[2] de mapas que funciona en un entorno web como un script CGI[3] o como una aplicación independiente a través de una API[4] accesible desde varios lenguajes de programación [Kropla, 2005] y además es de carácter libre. Su propósito es la visualización dinámica de mapas a través de Internet [MapServer, 2008]. Tiene la capacidad de funcionar en diversos sistemas operativos. Cuenta con extensiones que le permiten escuchar peticiones de mapas formuladas según los protocolos WMS[5], WFS[6] y WCS[7] de la OGC[8] y de enviar respuestas según el protocolo. Su utilización resulta muy importante para muchos, ya que permite compartir de manera visual la información geográfica, de forma dinámica y en tiempo real, además de que es posible compartir datos con otras aplicaciones utilizando las especificaciones OpenGIS[9] del OGC.

El funcionamiento de Mapserver se concibe generalmente detrás de un servidor web. Dicho servidor recibe las peticiones de los mapas por parte del cliente, este las envía a Mapserver, quien los construye y se los pasa al servidor web que es el encargado de mostrar la respuesta al cliente, como se muestra en la Figura 1.1[10].

[2]Término derivado del inglés rendering usado para referirse al proceso de generar una imagen desde un modelo.
[3]Interfaz de Entrada Común (del término en inglés CGI: Common Gateway Interface).
[4]Interfaz de Programación de Aplicaciones (del término en inglés API: Application Program Interface).
[5]Web Map Services.
[6]Web Feature Services.
[7]Web Coverage Service.
[8]Open Geospatial Consortium.
[9]Estándares internacionales orientados a Sistemas de Información Geográfica.
[10]Fuente: Elaboración propia basada en la figura 4.1 de [O'Reilly, 2005].

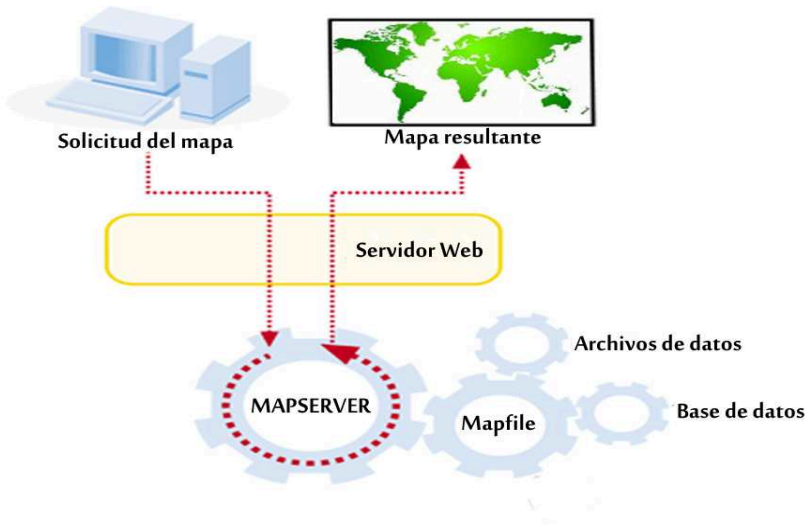

Figura 1.1: Funcionamiento del servidor de mapas Mapserver.

1.2.1. El mapfile

Mapserver es como una máquina que necesita combustible para poder funcionar [O'Reilly, 2005]. En este caso el combustible que necesita se denomina mapfile.

El mapfile define los recursos que serán utilizados por Mapserver para la representación del mapa, muestra y consulta de parámetros, también contiene información acerca de cómo se debe dibujar el mapa, la leyenda y el resultado de realizar una consulta. Por tanto define parámetros de los datos, el despliegue y las consultas, que serán usados en una aplicación con Mapserver, se puede hablar del mapfile como un archivo de configuración que tiene extensión ".map"[Espinosa, 2008]. Es el archivo principal desde donde Mapserver obtiene el tamaño del mapa, las capas que va a representar, la simbología y cómo se verá el mapa en general, por eso se dice que es el corazón o el núcleo de dicho servidor.

Este archivo de configuración tiene varias secciones que conforman el mapa en general como se muestra en la Figura 1.2[11]. Cada sección comienza con el nombre de la sección y culmina con la etiqueta END. Dentro de cada sección se especifican determinados parámetros en la forma ATRIBUTO - VALOR que pueden ser o no de carácter nulo.

Secciones del mapfile [MapServer, 2008]

[11]Fuente: Elaboración propia basada en [Ballari, 2006].

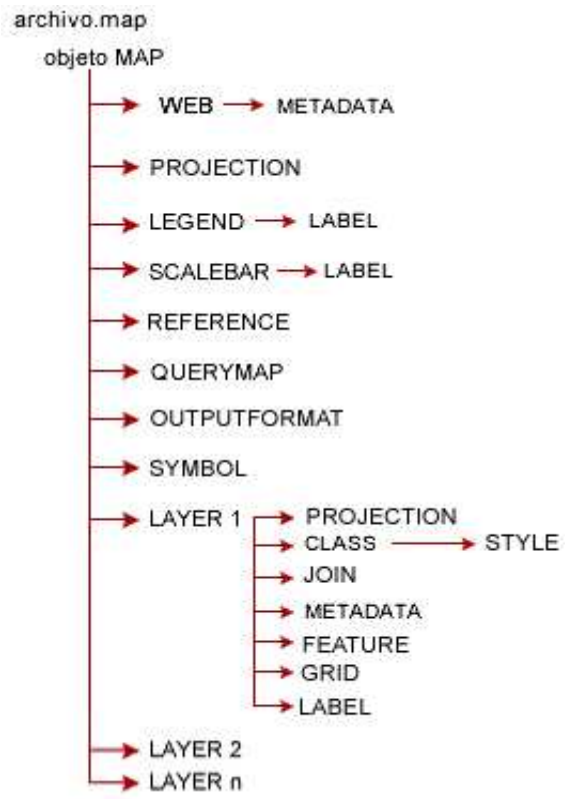

Figura 1.2: Esquema que representa las secciones que componen un mapfile.

Objeto MAP

Es el objeto principal del mapfile, contiene anidados los objetos WEB, PROJECTION, LEGEND, SCALEBAR, REFERENCE, QUERYMAP, OUTPUTFORMAT, SYMBOL y LAYER, y además propiedades generales que son esenciales en la representación del mapa: NAME, SIZE, STATUS, EXTENT, UNITS, IMAGECOLOR, IMAGETYPE, SHAPEPATH, FONSET, OFFSITE, SYMBOLSET.

Objeto WEB

Este objeto define cómo operará la interfaz web con respecto a la representación del mapa, o sea, es el objeto que permitirá visualizar la imagen creada por Mapserver insertándola en una página web. Mediante sus propiedades se especifica en qué directorio Mapserver ubica las imágenes que renderiza, se indica qué hacer cuando una consulta falla o cuando se produce algún error en la representación del mapa, se restringe la escala

a la que se debe mostrar el mapa si el usuario intenta llevarlo a una escala no permitida (ya sea muy pequeña o muy grande). Anida la sección METADATA y además las siguientes propiedades: HEADER, TEMPLATE, FOOTER, MINSCALE, MAXSCALE, IMAGEPATH, IMAGEURL, EMPTY URL.

Objeto PROJECTION

Define la proyección de los mapas y las capas que el servidor generará. Mapserver utiliza la librería PROJ4 "*Geographic Projection Library*"para tal fin. Puede ser definida utilizando los parámetros de la proyección o la codificación que propone el EPSG[12].

Objeto LEGEND

A partir de las propiedades que se definen en esta sección Mapserver es capaz de generar la leyenda del mapa. La leyenda es una imagen cuyo formato depende del formato definido para la creación del mapa. La sección contiene las siguientes propiedades: IMAGECOLOR, KEYSIZE, KEYSPACING, POSITION, STATUS, FONT, SYMBOLSET, BUFFER, MINDISTANCE, PARTIALS, SHADOWSIZE.

Objeto SCALEBAR

Define cómo será construida la escala gráfica. Esta sección contiene las siguientes propiedades: STYLE, STATUS, SIZE, COLOR, UNITS, INTERVALS, TRANSPARENT, POSITION, BACKGROUNDCOLOR, IMAGECOLOR, OUTLINECOLOR.

Objeto LABEL

Es usado por los objetos LAYER, LEGEND y SCALEBAR para definir una etiqueta con el objetivo de colocar algún tipo de anotación en el mapa. Las propiedades que lo definen son: ANGLE, BACKGROUNDCOLOR, BACKGROUNDSHADOWCOLOR, BACKGROUNDSHADOWSIZE, COLOR, FONT, FORCE, MAXSIZE, MINSIZE, MINDISTANCE, OFFSET, OUTLINECOLOR, PARTIAL, POSITION, SHADOWCOLOR, SHADOWSIZE, SIZE, TYPE.

[12]European Petroleum Survey Group.

Objeto REFERENCE

El conjunto de propiedades que se definen en esta sección permiten conformar el mapa de referencia, mapa que comprende la extensión total de la zona que incluirá el servicio WMS, sobre él se representará una marca en la zona que se visualiza actualmente, actualizándose interactivamente. Las propiedades que contiene son: IMAGE, EXTENT, SIZE, STATUS, MARKER, MARKERSIZE, MINBOXSIZE, COLOR, OUTLINECOLOR.

Objeto QUERYMAP

Esta sección define un mecanismo para recibir el resultado de las consultas. Contiene las siguientes propiedades: COLOR, STATUS, SIZE, STYLE.

Objeto OUTPUTFORMAT

Define los formatos de salida disponibles, incluye formatos como PNG[13], GIF[14], JPEG[15] y SWF[16]. Contiene las siguientes propiedades: NAME, DRIVER, IMAGEMODE, MIMETYPE, EXTENSION, TRANSPARENT, FORMATOPTION.

Objeto SYMBOL

Esta sección define un símbolo del mapa. Un símbolo es un caracter cartográfico configurable a través de la modificación de los atributos que lo componen, puede declararse tanto en la propia estructura de un mapfile o como un archivo independiente. El mapa puede tener varios símbolos y de la misma forma se dice que no tener símbolos es no tener mapa. Las propiedades que lo componen son: NAME, ANTIALIAS, CHARACTER, TRASNPARENT, LINEJOIN, POINTS, PATTERN, TYPE, FILLED, FONT, GAP, LINECAP, IMAGE, LJMAXSIZE.

Objeto LAYER

Define las propiedades de las capas del mapa que se generará. Es el objeto más utilizado en el mapfile, describe las capas que se utilizarán para la representación del mapa. Contie-

[13]Portable Network Graphics.
[14]Graphics Interchange Format.
[15]Joint Photographic Experts Group.
[16]Small Web Format.

ne los objetos CLASS, METADATA, PROJECTION, JOIN, GRID, FEATURE y LABEL y además las siguientes propiedades: NAME, GROUP, TYPE, TYPERASTER, DATA, CONNECTIONTYPE, CLASSITEM, LABELITEM, HEADER, FOOTER, TRANSPARENCY, TOLERANCE, TILEINDEX, PROCESSING.

Objeto CLASS

Este objeto determina un conjunto de propiedades específicas para una capa, define clases temáticas para las mismas. Cada capa debe tener al menos una clase. Anida el objeto STYLE y contiene las siguientes propiedades: NAME, COLOR, OUTLINECOLOR, EXPRESSION.

Objeto STYLE

Determina un conjunto de propiedades específicas para un objeto CLASS. Una clase puede tener varios estilos. Contiene las siguientes propiedades LABEL, ANGLE, BACKGROUNDCOLOR, BACKGROUNDSHADOWCOLOR, BACKGROUNDSHADOWSIZE, COLOR, FONT, FORCE, MAXSIZE, MINSIZE, MINDISTANCE, OFFSET, OUTLINECOLOR, PARTIAL, POSITION, SHADOWCOLOR, SHADOWSIZE, SIZE, TYPE, IMAGECOLOR, KEYSIZE, KEYSPACING, STATUS, SYMBOLSET, BUFFER, MINDISTANCE, PARTIALS.

Objeto JOIN

Este objeto permite obtener información de más de una fuente de datos para representar una capa determinada del mapa, o sea con su utilización se puede indicar la dirección de una fuente de datos (base de datos o fichero shape[17]) que junto a la que se especifica mediante la propiedad DATA del objeto LAYER permitirán representar dicho objeto. Contiene las siguientes propiedades: NAME, CONNECTION, CONNECTIONTYPE, FROM, TABLE, TEMPLATE, TO, TYPE.

[17]Formato vectorial de almacenamiento digital donde se guarda la localización de los elementos geográficos y los atributos asociados a ellos.

Objeto METADATA

Es utilizado por el objeto MAP para indicar los metadatos en general del servicio y por el objeto LAYER para indicar metadatos específicos para cada capa de información. Los servidores que implementan los estándares WMS y WFS utilizan estos datos para manejar información del mapa en general, para definir parámetros propios de la conexión, el formato en que se quiere que el servidor entregue la imagen renderizada, entre otros parámetros. Además, con estos metadatos se confecciona el archivo de capacidades[18].

Objeto FEATURE

Este objeto permite agregar geometrías a las capas que lo incluyan, independientemente de las que se definan en la fuente de datos que indica la propiedad DATA. Contiene las siguientes propiedades: POINTS, ITEMS, TEXT, WKT.

Objeto GRID

Este objeto permite que se representen rejillas en el mapa que indican los grados de latitud y longitud del mismo. Contiene las siguientes propiedades: LABELFORMAT, MINARCS, MAXARCS, MININTERVAL, MAXINTERVAL, MINSUBDIVIDE, MAXSUBDIVIDE.

1.3. Catalogación de mapas

El auge que ha tenido el uso y la búsqueda de la información geográfica en el mundo ha traído consigo que se creen herramientas que faciliten su manejo y organización. En la actualidad, estas herramientas se han convertido en instrumentos imprescindibles en varias de las entidades que hoy las utilizan. Una de estas herramientas son los ya mencionados catálogos. [Costa, 2010] plantea que son de gran utilidad a la hora de organizar adecuadamente una determinada información. Este tipo de herramienta es necesaria para

[18] Archivo con formato XML[19] que contiene los metadatos de los servicios y de las capas de información que contiene.

localizar, identificar, valorar y obtener información de un tema en particular.

1.3.1. Metadato

En los catálogos de información geográfica existen elementos que ayudan a documentar los datos y permiten el procesamiento, interpretación y almacenamiento de los mismos, estos son los llamados metadatos.

Un metadato es una información que describe entre otras características, la calidad, distribución, actualidad y referencia espacial de un conjunto de datos [Costa, 2010]. Además de que es un objeto que describe otro objeto, también puede utilizarse para acceder o encontrar dicho objeto, pues funcionan como un vocabulario descriptivo común. Autores como [Pombert, 2006] y [Ortiz, 1999] se refieren a los metadatos como "dato de los datos".

1.3.2. Servicio de catálogo

Al servicio que soporta el descubrimiento y acceso a la información geoespacial a través de catálogos se le conoce de diferentes maneras dentro de la comunidad geoespacial, servicio de catálogo es una de ellas.

En concreto, servicio de catálogo es aquel que responde a peticiones de metadatos en un catálogo que cumple con ciertos criterios de navegación y búsqueda. Su objetivo es descubrir datos geoespaciales a través de las propiedades descritas por sus metadatos [Nebert, 2001].

1.3.3. Especificación de servicios de catálogo de la OGC

La especificación de servicios de catálogo de la OGC establece un marco general para la implementación de servicios de catálogo que se pueden aplicar para satisfacer peticiones en una amplia variedad de dominios [OGC, 2007].

Los servicios de búsqueda CWS[20] son los que permiten el acceso a los catálogos con el

[20]Catalog Web Services.

uso de metadatos. Es uno de los servicios más importantes que debe existir en una IDE. El mismo debe cumplir con el requisito de ser interoperable[21]. Se apoya en estándares del OpenGIS y permite la publicación y el acceso a catálogos digitales de datos y servicios geoespaciales a través de metadatos. El CWS tiene una capa de metadatos que almacena en la base de datos los metadatos necesarios para responder a solicitudes de catálogo. Los metadatos incluyen columnas espaciales, que pueden ser consultados y procesados [OGC, 2007].

1.3.4. Uso del servicio CWS

El servicio CWS se utiliza de la siguiente manera:

Un usuario interesado en localizar información geográfica utiliza una interfaz de usuario para la búsqueda, hace una solicitud de búsqueda, especificando las preguntas sobre los datos con unas determinadas propiedades. La petición de búsqueda pasa al catálogo y este formula una pregunta al servidor de catálogo registrado. El servidor de catálogo gestiona una colección de entradas de metadatos. Dentro de las entradas de metadatos hay instrucciones sobre cómo llegar a los datos espaciales que se han descrito [Nebert, 2001].

1.4. Situación Problemática

En la UCI, se ha creado la plataforma GeneSIG, una herramienta informática que tiene como función principal realizar la representación geoespacial de la información asociada a negocios específicos y permitir realizar análisis sobre dicha información. Al mismo tiempo que se desarrollaba la plataforma GeneSIG se implementó el catálogo de mapas LiberMaps, herramienta muy útil para aquellos que se apoyan en SIG que utilizan el servidor de mapas Mapserver, como es el caso de GeneSIG, pues el corazón de este servidor es un fichero con extensión ".map"que contiene toda la información sobre el mapa que representa el SIG. Es precisamente la edición dinámica de este fichero la razón de ser

[21] Habilidad de dos o más sistemas o componentes para intercambiar información y utilizar la información intercambiada.

fundamental del catálogo de mapas LiberMaps, pues muchos de los usuarios que deben configurar este fichero lo hacen manualmente, tornándose extremadamente engorroso teniendo en cuenta el volumen de datos a ser modificados. Además se implementó también el SyGMe, módulo para la gestión de metadatos geográficos, que permite documentar la información geográfica y facilita una mejor organización y localización de los datos.

Actualmente GeneSIG no cuenta con un mecanismo para personalizar la información que representa. Por su parte, LiberMaps, gestiona su seguridad y en función de eso se configura, pero precisamente su existencia se debe a que sirva como intermediario entre el servidor de mapas y el SIG, permitiendo hacer dinámico el acceso a la información espacial y además garantizando la seguridad de los datos que se manejen en el SIG.

En la edición de los ficheros ".map" a través de LiberMaps 1.0, existen procedimientos con cierto grado de similitud y que pueden llegar a ser engorrosos por la cantidad de campos de información a tener en cuenta. Por ejemplo, a la hora de crear varios ficheros con similares datos el proceso se torna un poco repetitivo, disminuyendo de esta manera la usabilidad del sistema. Por otro lado, en el catálogo se almacenan gran cantidad de mapas, por lo que en varias ocasiones se le hace difícil al usuario encontrar el mapa que desea editar. Uno de los servicios más importantes que debe existir en una IDE, según [OGC, 2007], es el servicio de búsqueda de información geográfica a través de metadatos. En este caso, el SyGMe permite gestionar colecciones de metadatos a través de los cuales se documenta y localiza la información geográfica, pero el catálogo de mapas LiberMaps 1.0 no cuenta con los mecanismos necesarios para poder acceder a ellos y de esta manera materializar el servicio de búsqueda que el mismo requiere.

Por esta razón se ha detectado la necesidad de implementar una nueva versión del catálogo de mapas LiberMaps, que se comunique con la plataforma GeneSIG para lograr manejar el acceso a los datos y a las funcionalidades según los usuarios y roles de ambos sistemas; y con el SyGMe, para documentar la información que se maneja en el catálogo a través de metadatos y lograr la realización de búsquedas de los mapas del catálogo. Además debe contar con funcionalidades que permitan agilizar el proceso de creación de los mapas.

1.5. Análisis de las soluciones existentes

Actualmente se pueden encontrar herramientas que brindan servicios de catálogo y otras que permiten la edición del mapfile, las más utilizadas son:

GeoNetwork, sistema de código abierto para la catalogación de información geoespacial, productos cartográficos y metadatos. Permite la búsqueda a través de múltiples catálogos mediante de los servicios que brinda y provee un visor de mapas para combinar servicios web. Brinda la posibilidad de visualizar una o más capas de los mapas que el servidor ofrece. Permite crear y exportar mapas, además de gestionar las cuentas de los usuarios y grupos.

ArcCatalog, es uno de los softwares propietarios más utilizados en el mundo de los SIG. Se utiliza para organizar la información de un SIG, entre ellas los mapas, a los que se puede acceder mediante búsquedas y además permite gestionar colecciones de metadatos basados en estándares, útiles para documentar la información.

MapStorer, herramienta de código abierto, útil para manejar proyectos de Mapserver. Permite a los usuarios crear el mapfile. Provee funcionalidades que permiten manipularlo, por ejemplo crear uno nuevo, adicionarle capas, crear clases para las capas, exportar e importar el fichero. Pero aún así tiene varias debilidades, solo acepta la proyección EPSG: 4326, y no permite la creación y edición de uno de los objetos más importantes del mapa, pues se dice que sin él es como si no existiera, el objeto SYMBOL. Después de haber realizado un análisis de las principales herramientas que permiten la edición del mapfile y la catalogación de información geográfica, se decide descartar a ArcCatalog por su condición de herramienta propietaria pues a pesar de que tiene grandes potencialidades no contribuye con la soberanía tecnológica por la que aboga el país. Se decide implementar la versión 2.0 del catálogo de mapas **LiberMaps**, decisión en la que tuvo gran peso la opinión de los especialistas del grupo de trabajo de la plataforma GeneSIG. Esta nueva versión debe basarse en la funcionalidades que brinda MapStorer para la edición del archivo mapfile y en las que brinda GeoNetwork para la realización de búsquedas de información geográfica.

1.6. Herramientas y tecnologías a utilizar

Una vez definida la solución al problema antes planteado se hace necesario seleccionar las herramientas y tecnologías con las que será desarrollada. En el presente epígrafe se justifica la elección de las tecnologías y herramientas existentes actualmente que son necesarias y contribuyen al desarrollo de esta tarea. La selección que se propone es parte del resultado de una investigación desarrollada en el transcurso de los años 2009 - 2010, titulada "Diseño de la arquitectura base del catálogo de mapas LiberMaps", realizada con el objetivo de definir la arquitectura del sistema. La solución estará desarrollada con herramientas y tecnologías libres y sustentada sobre tecnología web.

1.6.1. Metodología de desarrollo de software

El proceso de desarrollo de software se torna una tarea ardua en la mayoría de las ocasiones. Para lograr obtener un producto con la calidad requerida es necesario tener una guía que imponga cierta disciplina y funcione como un hilo conductor en el proceso de desarrollo.

Con este objetivo han sido creadas las metodologías de desarrollo de software, las cuales proporcionan las guías para poder conocer todo el camino a recorrer desde antes de empezar la implementación, con lo cual se asegura la calidad del producto final, así como el cumplimiento en la entrega del mismo en un tiempo estipulado [Chacón, 2006].

Entre las metodologías existentes se define RUP[22] como la más adecuada para el desarrollo, por las ventajas que ofrece su uso al garantizar un proceso de desarrollo organizado, robusto y correctamente documentado. Para la construcción del sistema se necesita una metodología conocida por el equipo de proyecto, que permita integrar las etapas de desarrollo fácilmente y que sea compatible o tenga relación con UML[23], que dirija las tareas de cada miembro del equipo por separado pero a la vez como un todo y que ofrezca criterios para el control y medición de la calidad del producto. Todas estas características están

[22]Proceso Unificado Racional (del término en inglés RUP: Rational Unified Process).
[23]Lenguaje Unificado de Modelado (del término en inglés UML: Unified Model Language).

presentes en RUP.

Además de las características antes mencionadas, existen tres elementos fundamentales que definen a RUP [Jacobson, 2000] y que son de gran importancia para el proceso de desarrollo, ellas son:

- Es una metodología guiada por casos de uso, lo que facilita el desarrollo del sistema a partir de los requisitos, ya que los mismos inician el proceso de desarrollo y le proporcionan un hilo conductor.
- Define un proceso iterativo incremental, lo que da la posibilidad de organizar el desarrollo por fases, condición que ayuda a observar como va evolucionando el producto de manera que se puedan mitigar los riesgos de forma temprana y corregir las deficiencias.
- Es centrada en la arquitectura, lo que indica cómo debe construirse el sistema.

La metodología RUP define cuatro fases para el desarrollo del software: Inicio, Elaboración, Construcción y Transición, dentro de las cuales existen nueve flujos de trabajo [Progress, 2009] que tienen mayor o menor peso en dependencia de la fase en que se encuentre el desarrollo, ellos son: Modelación del negocio, Requisitos, Análisis y diseño, Implementación, Prueba, Despliegue, Administración, Configuración de cambios, Administración del proyecto y Ambiente. Los seis primeros flujos son conocidos como flujos de ingeniería y los tres últimos como flujos de apoyo.

1.6.2. Lenguaje de modelado

Además de lo esencial que resulta utilizar una metodología de software como hilo conductor de todo el ciclo de vida del sistema, también es importante tener en cuenta que en un proceso de desarrollo de software es necesario contar con algún elemento que describa el aspecto y la conducta del producto, estos elementos son llamados lenguajes de modelado.

UML es uno de los lenguajes de modelado de gran utilidad para el desarrollo, pues ofrece un modo estándar de visualizar, especificar, construir y documentar los artefactos de un sistema. Su objetivo es lograr una aplicación de software robusta, flexible y escalable. [Jacobson, 2000].

UML V2.0 es el lenguaje de modelado que se seleccionó para el desarrollo de la solución, pues resulta muy eficiente cuando se emplea RUP como metodología de desarrollo, su combinación es la más utilizada para realizar el análisis, implementación y documentación de los sistemas orientados a objetos, como es el caso [Ledea, 2010]. UML es utilizado a nivel internacional tanto para modelar software, hardware u organizaciones del mundo real. Además está preparado para usarse con todos los métodos de desarrollo y etapas del ciclo de vida de un software.

1.6.3. Herramienta CASE

Con el objetivo de apoyar y automatizar la metodología de software y el lenguaje de modelado se emplean las herramientas CASE[24]. Estas, son sistemas que facilitan el diseño y la documentación de las actividades del proceso de desarrollo de un software.

Visual Paradigm V6.4, se considera una de las herramientas CASE más adecuada para trabajar en software libre, esta propiedad es básicamente, la que constituyó un hecho determinante en su selección para ser utilizada en el proceso de modelado del catálogo de mapas LiberMaps. Posee una interfaz amigable y fácil de utilizar [Ledea, 2010]. El hecho de que es una herramienta de apoyo al lenguaje UML que soporta todo el ciclo de desarrollo del software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue, es otra de las razones que se tuvo en cuenta para determinar que es la herramienta indicada para realizar el modelado del sistema.

[24]Ingeniería de Software Asistida por Computadora (del término en inglés CASE: Computer Aided Software Engineering).

1.6.4. Lenguaje del lado del servidor

El lenguaje de programación del lado del servidor escogido es PHP[25] en su versión 5, pues por sus características se considera que posee la mejor mezcla entre flexibilidad y rendimiento para la realización de páginas web dinámicas. Además cuenta con abundante documentación lo que permite un mayor entendimiento del mismo. Este lenguaje está asociado con una extensa biblioteca que va creciendo conforme se realizan nuevas versiones, la cual permite la realización de disímiles tareas como la encriptación, el acceso a base de datos y el tratamiento de ficheros. Es un lenguaje abierto, multiplataforma, presenta un amplio soporte para la Programación Orientado a Objetos [Ledea, 2010]. Además soporta varios SGBD[26], como MySQL, Oracle y PostgreSQL, está integrado con varias bibliotecas externas que permiten por ejemplo generar documentos PDF[27] y hacer análisis de código XML, y es atendido por una amplia comunidad de desarrolladores que lo actualizan frecuentemente con extensiones y mejoras y además encuentran y reparan errores de funcionamiento del lenguaje.

1.6.5. Lenguaje del lado del cliente

JavaScript V1.5 es compatible con la mayoría de los navegadores, esto es lo que lo convierte en el lenguaje del lado del cliente más utilizado. Es por ello que se propone para su utilización, además de la posibilidad que tiene de poder incluirse en cualquier documento, por ejemplo PHP, entre otros [Ledea, 2010]. Es un lenguaje interpretado, lo que significa que no requiere de compilación. Se caracteriza por ser sólido, robusto e increíblemente poderoso. [Camy, 2002] plantea que utiliza pocos recursos de memoria, es totalmente gratuito y garantiza un balance en la carga del servidor, ya que se ejecuta directamente en el cliente, el servidor no es solicitado para ello. Además posee gran cantidad de documentación en línea y permite el preprocesado de información antes de enviarla al servidor.

[25] PreProcesador de Hipertexto (del término en inglés PHP: Hypertext PreProcessor).
[26] Sistema Gestor de Base de Datos.
[27] Formato de Documento Portátil (del término en inglés PDF: Portable Document Format).

1.6.6. Sistema gestor de base de datos

Para almacenar la información que se manipula en un gran porciento de aplicaciones, se utilizan los SGBD, programas que garantizan la integridad y seguridad de la información y que sirven como intermediarios entre las aplicaciones y los datos.

El gestor de base de datos elegido fue PostgresSQL V8.3, pues permite el soporte para datos espaciales en un SIG mediante la extensión PostGIS[28], conjuntamente con la gestión de objetos geográficos, aspecto fundamental dadas las características de la aplicación que se desea desarrollar. Cuenta con soporte para la creación de tipos, triggers, vistas, herencia, reglas y procedimientos almacenados, así como el hecho de que escala muy bien cuando aumenta el número de CPU[29] y la cantidad de RAM[30], debido a su arquitectura de diseño [Ledea, 2010]. Tiene soporte nativo para los lenguajes más populares del medio, como PHP, seleccionado anteriormente. Es una herramienta soberana, con una gran comunidad de usuarios, denominada PGDG[31], que permite su continuo desarrollo. Cuba cuenta con una comunidad de desarrollo de dicho gestor integrada por profesionales y estudiantes de la UCI, lo que constituye un factor importante en cuanto a la continuidad de la implementación del sistema.

1.6.7. Servidor web

Apache Server V2.0 es uno de los servidores web más populares a nivel internacional, se caracteriza generalmente por su gran robustez, flexibilidad, estabilidad y eficiencia. Para el desarrollo y posterior uso del sistema, Apache Server es el servidor web seleccionado por las múltiples ventajas que ofrece, por ser de carácter libre y además por ser compatible con múltiples sistemas operativos como Linux y Windows entre los más usados, lo que lo hace prácticamente universal [Ledea, 2010]. Además, este servidor permite la publicación de documentos PHP de la misma forma que se hace en Internet. Su condición de software

[28] Módulo que añade soporte de objetos geográficos a la base de datos PostgreSQL, convirtiéndola en una base de datos espacial para su utilización en SIG.
[29] Unidad Central de Procesamiento (del término en inglés CPU: (Central Processing Unit).
[30] Memoria de Acceso Aleatorio (del término en inglés RAM: Random Access Memory).
[31] PostgreSQL Global Development Group.

libre hace que pueda adaptarse a diferentes entornos y necesidades, no necesita grandes recursos para funcionar y está respaldado por una comunidad de desarrollo amplia.

1.6.8. Servidor de mapas

Para el manejo de información geográfica a través de la web además de ser necesario un servidor web, se requiere también de un servidor de mapas, quien es el encargado de realizar la operaciones necesarias con la información geoespacial y enviar el resultado de estas operaciones al cliente.

Como servidor de mapas se seleccionó a Mapserver V5.6, pues de todas las tecnologías de código abierto que permiten la representación geoespacial, este es hasta el momento el más estable. Además, su funcionalidad aumenta si se combina su instalación con PostgresSQL como gestor de base de datos y PHP como lenguaje de programación [Ledea, 2010]. La biblioteca MapScript, que provee el servidor, puede ser utilizada por lenguajes interpretados como PHP, seleccionado anteriormente.

1.6.9. Framework Symfony

Se seleccionó al framework Symfony V1.2 para la construcción del sistema, por ser consecuente con el lenguaje definido para el lado del servidor, pues está desarrollado completamente con PHP5. Mediante este framework, el acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos. Esto le permite un alto nivel de abstracción y una fácil portabilidad. Este framework automatiza la tarea repetitiva de crear módulos para manipular datos mediante el uso de objetos Propel[32]. Si el modelo de objetos está bien definido, es posible incluso generar de forma automática la parte de administración completa de un sitio web [Ledea, 2010].

Es un framework muy utilizado a nivel internacional, bien documentado y multiplataforma. Es lo suficientemente flexible como para adaptarse a los casos más complejos, sigue la mayoría de mejores prácticas y patrones de diseño para la web, es independiente

[32]ORM[33] Open Source para PHP5 que permite acceder a la base de datos utilizando un conjunto de objetos, que proporcionan una interfaz sencilla para almacenar y recuperar datos.

del SGBD y fácil de extender, lo que permite su integración con librerías desarrolladas por terceros; por tanto, permitirá que el proceso de desarrollo del sistema sea robusto y organizado. Un aspecto clave a tener en cuenta para el desarrollo del sistema es la seguridad de la información. Esta es otra de las razones por la que se escoge Symfony, pues el mismo permite la creación de aplicaciones seguras. La gestión de la caché reduce el ancho de banda utilizado y la carga del servidor.

1.6.10. Framework ExtJS

ExtJS V2.2 fue el framework de AJAX[34] seleccionado para apoyar el lenguaje del lado del cliente JavaScript, pues permite la creación de aplicaciones complejas de forma más eficiente. Este framework permite distribuir la carga de procesamiento, por lo que se logra establecer un balance entre el cliente y el servidor. Esto posibilita que el servidor, al tener menos carga, pueda atender más clientes a la vez [Ledea, 2010]. Una de sus mayores ventajas es que la comunicación es asíncrona, brindando la libertad de cargar información de manera imperceptible para el usuario. Otro factor a considerar en su uso es la eficiencia de la red, el tráfico de red puede disminuir al permitir que la aplicación elija qué información desea transmitir al servidor y viceversa. Según [Sencha, 2011] es un framework de código abierto que posibilita la reutilización de código, y que provee APIs para cada una de sus versiones que facilitan el trabajo del desarrollador. Es neutral al lenguaje que se use en el servidor. Además posee una gran comunidad de desarrollo.

1.6.11. Entorno de Desarrollo Integrado

Además de los frameworks de desarrollo, existen otras herramientas como los entornos de desarrollo integrado, que permiten y facilitan la creación de aplicaciones de softwares. Estos son un conjunto de herramientas informáticas que proveen un marco de trabajo para uno o varios lenguajes de programación.

Netbeans V7.0 se selecciona como herramienta de entorno de desarrollo integrado

[34] JavaScript Asíncrono y XML (del término en inglés AJAX: Asynchronous JavaScript And XML).

siendo consecuentes con uno de los lenguajes y frameworks previamente propuestos, pues brinda la posibilidad de desarrollar aplicaciones web utilizando PHP y ofrece soporte para Symfony [Ledea, 2010]. Además es un proyecto de código abierto sin restricciones para su uso. Permite desarrollar las aplicaciones mediante módulos, lo que facilita la posterior extensión de dichas aplicaciones. A pesar de estar escrito en Java[35] puede utilizarse para escribir, compilar, depurar y ejecutar programas en otros lenguajes de programación como es el caso de PHP. Ofrece autocompletado de código, marcado de error para el lenguaje PHP y tiene un potente debugger[36] integrado, con facilidad de utilización.

1.7. Conclusiones parciales

El creciente auge que ha tenido el uso de la información geográfica para la toma de decisiones en gran parte de las esferas socioeconómicas a nivel mundial, ha demostrado que contar con sistemas que faciliten su manipulación es de vital importancia. El estudio realizado en el presente capítulo ratifica que estas aplicaciones constituyen un campo de investigación muy activo y amplio. Se identificaron funcionalidades básicas de las principales herramientas existentes a nivel internacional que permiten la edición del archivo mapfile y la catalogación de información geográfica. Teniendo en cuenta las necesidades actuales se tomó la decisión de implementar la versión 2.0 del catálogo de mapas LiberMaps, apoyado en las principales funcionalidades detectadas en los sistemas MapStorer y Geonetwork. Finalmente, ya definidas las herramientas y tecnologías a utilizar para el desarrollo de la solución, se puede afirmar que los aspectos abordados en el presente capítulo proveen la base de conocimientos necesaria para presentar una propuesta de solución.

[35] Lenguaje de programación orientado a objetos.
[36] Programa que permite identificar y corregir los errores de otro programa informático.

Capítulo 2

Descripción de la solución propuesta

En el presente capítulo comienza la elaboración de la solución propuesta: la versión 2.0 del catálogo de mapas LiberMaps. Se describen los principales artefactos relacionados con la modelación del dominio y el levantamiento de requisitos, según indica la metodología RUP. Se presenta específicamente el modelo de dominio y los principales conceptos asociados a él. Se describen detalladamente los requisitos funcionales y no funcionales con los que debe cumplir el sistema, los actores, el diagrama de casos de uso y su descripción.

2.1. Modelo de dominio

Teniendo en cuenta que no se tienen bien definidos los procesos del negocio, dado que el catálogo de mapas LiberMaps es un sistema concebido para comercializar con cualquier empresa o entidad, se realizará una modelación del dominio y se procederá a explicar cada uno de los conceptos que forman parte del mismo, conformando así un diccionario.

Según [Jacobson, 2000] un modelo de dominio captura los tipos más importantes de objetos en el contexto del sistema. Los objetos del dominio representan las cosas que existen o los eventos que suceden en el entorno en el que trabaja el sistema. Todo ello se representa a través de clases relacionadas, mediante el lenguaje UML, con el objetivo de tener una mejor comprensión de la estructura y dinámica de la organización, los problemas actuales dentro de esta, e identificar las mejoras potenciales.

2.1.1. Diagrama de clases del dominio

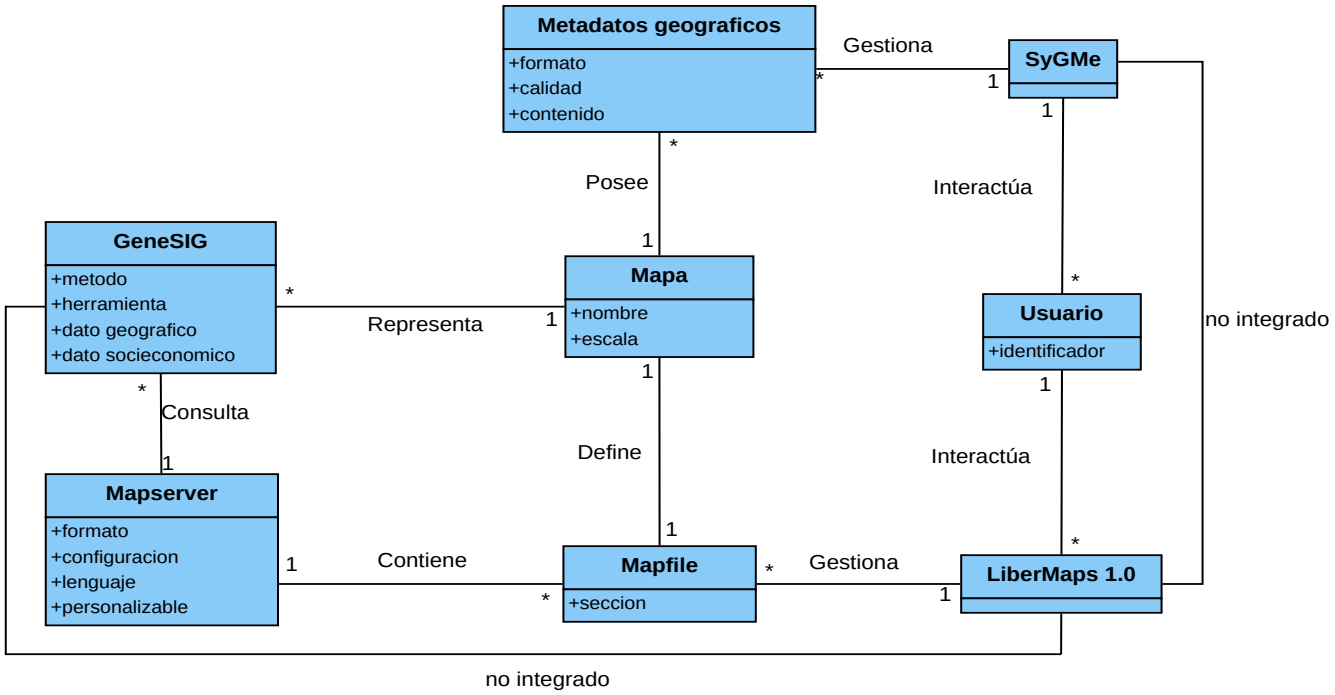

Figura 2.1: Diagrama de clases del dominio del catálogo de mapas.

2.1.2. Breve descripción del modelo de dominio

El SIG, en este caso la plataforma GeneSIG, realiza consultas al servidor de mapas Mapserver que contiene el fichero de configuración de mapas o Mapfile por cada una de las consultas que realiza el SIG. Cada uno de estos ficheros define un mapa que GeneSIG representará. Los mapas poseen metadatos geográficos que los describen y ayudan a su documentación. Por su parte, el usuario es quien gestiona los ficheros mapfile y los metadatos a través de su interacción con el catálogo de mapas LiberMaps en su versión 1.0 y el SyGMe respectivamente. Actualmente no existe comunicación entre el catálogo de mapas LiberMaps y GeneSIG, así como entre el catálogo de mapas y SyGMe.

2.1.3. Glosario de términos del dominio

GeneSIG: Plataforma para la creación de SIG, capaz de integrar, almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada.

Mapa: Es una representación geográfica de una parte de la superficie terrestre, en la que se presenta información relativa a una ciencia determinada. Es construido mediante gráficas y métricas, de forma bidimensional o esférica.

Mapserver: Servidor de Mapas que es consultado por el SIG para visualizar mapas.

Mapfile: Es el archivo principal desde donde Mapserver obtiene las características del mapa que representa el SIG, por ejemplo de qué tamaño será el mapa, las capas que va a representar, la simbología y cómo se verá el mapa en general, por eso se dice que es el corazón o el núcleo de dicho servidor.

Metadatos geográficos: Es una información que describe entre otras características, la calidad, distribución, actualidad y referencia espacial de un conjunto de datos geográficos.

Usuario: Persona, organismo u organización que necesite acceder a los metadatos de un mapa a través del SyGMe o que necesita configurar un mapa a través del mapfile que lo representa utilizando para ello al catálogo de mapas LiberMaps 1.0.

SyGMe: Sistema de Gestión de Metadatos que permite mantener el control de los metadatos geográficos que poseen los mapas.

LiberMaps 1.0: Catálogo de mapas que permite mantener el control y gestionar cada archivo mapfile de Mapserver.

2.2. Modelo del sistema

Luego de haber establecido una relación entre las clases que conforman el dominio del sistema, así como la descripción de los principales conceptos a través del modelo de dominio, se presentan los principales artefactos del flujo de trabajo Requisitos que tiene como objetivo principal guiar el desarrollo hacia el sistema correcto.

2.2.1. Requisitos funcionales

Los requisitos funcionales son condiciones o capacidades con los que debe cumplir el sistema, son acuerdos entre el cliente y los desarrolladores sobre lo que debe o no debe hacer el sistema [Jacobson, 2000]. Estos guiarán el proceso de desarrollo del catálogo, pues constituyen la base sobre la que se construirá el sistema y finalmente conforman el conjunto de funcionalidades con las que debe contar. A continuación se presentan, para ver una descripción más detallada de cada uno de los requisitos se debe consultar el documento de Especificación de requisitos de software del catálogo de mapas LiberMaps V2.0.

- RF[1] 1. Generar mapfile de forma automática.
- RF 2. Importar mapfile.
- RF 3. Obtener metadatos de un mapa.
- RF 4. Obtener metadatos de una capa.
- RF 5. Combinar Objeto LAYER.
- RF 6. Crear metadato del objeto WEB.
- RF 7. Eliminar metadato del objeto WEB.
- RF 8. Crear metadato del objeto LAYER.

[1]Requisito Funcional.

- RF 9. Eliminar metadato del objeto LAYER.
- RF 10. Buscar mapfile.
- RF 11. Autenticar Usuario.
- RF 12. Cambiar sistema de autenticación.
- RF 13. Obtener mapfile mediante servicios web.
- RF 14. Obtener mapfile por usuario mediante servicios web.
- RF 15. Obtener capas de un mapfile y un usuario mediante servicios web.
- RF 16. Exportar mapfile mediante servicios web.
- RF 17. Almacenar mapfile mediante servicios web.
- RF 18. Obtener metadatos de un mapa mediante servicios web.
- RF 19. Obtener metadatos de una capa mediante servicios web.

2.2.2. Requisitos no funcionales

Además de los requisitos funcionales deben especificarse los requisitos no funcionales. Según [Jacobson, 2000] son requisitos que especifican propiedades del sistema, como restricciones del entorno o la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad. A continuación se presentan:

Requisitos de Usabilidad

- **RNF**[2] **1.** Se emplearán componentes que indiquen al usuario el estado de los procesos que por su complejidad requieran de un tiempo de procesamiento apreciable.
- **RNF 2.** El software tendrá siempre visible la opción de Ayuda, lo que posibilitará un mejor aprovechamiento por parte de los usuarios de sus funcionalidades.

[2]Requisito No Funcional.

Requisitos de Fiabilidad

- **RNF 3.** El sistema debe estar disponible todo el tiempo para sus usuarios, descontando el tiempo que se encuentre en mantenimiento y la ocurrencia de alguna falla externa (Ejemplo: problemas de electricidad).
- **RNF 4.** El tiempo medio de reparación en caso de fallos es de 7 días.

Requisitos de Eficiencia

- **RNF 5.** El tiempo de respuesta, la velocidad de procesamiento de la información, la actualización y la recuperación estará dado por la cantidad de información a procesar, entre mayor cantidad de información, mayor será el tiempo de procesamiento.

Requisitos de soporte

- **RNF 6.** El período de soporte así como las restricciones asociadas se manejarán entre el equipo de desarrollo y los clientes.

Restricciones de diseño

- **RNF 7.** El producto de software final debe diseñarse sobre una arquitectura cliente-servidor.
- **RNF 8.** Se deben emplear los estándares establecidos (diseño de base de datos y codificación). Para ver una descripción más detallada se deben consultar los documentos: Estándares de codificación PHP y Estándares de diseño de bases de datos del catálogo de mapas LiberMaps V2.0.

Requisitos para la documentación de usuarios en línea y ayuda del sistema

- **RNF 9.** El software tendrá siempre la posibilidad de ayuda disponible para cualquier tipo de usuario, lo que le permitirá un avance considerable en la explotación de la aplicación en todas sus funcionalidades.

Requisitos de Interfaz de usuario

- **RNF 10.** El sistema debe tener indicadores que permitan conocer al usuario las acciones que debe realizar, por ejemplo botones con iconos sugerentes y alternativa textual.
- **RNF 11.** El sistema debe permitir al usuario transitar de una tarea a otra sin necesidad de obligarlo a realizar acciones innecesarias o no deseadas, por ejemplo para llegar de una tarea a otra el usuario no debe dar más de 3 clicks.

Requisitos de hardware

Para las estaciones clientes

- **RNF 12.** Se requiere tengan tarjeta de red.
- **RNF 13.** Al menos 128 MB de memoria RAM.
- **RNF 14.** Se requiere al menos 100 MB libres del disco duro.
- **RNF 15.** Procesador 512 MHz como mínimo.

Para los servidores

- **RNF 16.** Se requiere tarjeta de red.
- **RNF 17.** El Servidor de Mapas debe tener como mínimo 2 GB de RAM y 2 GB de disco duro.
- **RNF 18.** El Servidor de base de datos debe tener como mínimo 2 GB de RAM y 10 GB de disco duro.
- **RNF 19.** Procesador 3 GHz como mínimo.

Requisitos de software

Para las estaciones clientes

- **RNF 20.** Un Navegador (Mozilla Firefox, Zafari u otro navegador) que cumpla con los estándares W3C[3].
- **RNF 21.** Sistema operativo: GNU/Linux, Windows o Mac OS.

Para los servidores

- **RNF 22.** Sistemas operativos GNU/Linux o Windows Server 2000 o superior.
- **RNF 23.** Servidor Web Apache 2.0 o superior, con módulo PHP5 configurado con la extensión PGSQL[4] incluida.
- **RNF 24.** PostgresSQL V8.3 como SGBD, con la extensión PostGis V1.3.5 como soporte de datos espaciales.
- **RNF 25.** Mapserver 5.6, con extensión PHP Mapscript.

Requisitos de licencia

- **RNF 26.** De acuerdo a los tipos de licencias de los componentes y herramientas que se proponen a utilizar para el desarrollo del catálogo de mapas LiberMaps se puede catalogar legalmente su arquitectura de modelo libre, permitiendo la utilización, modificación y distribución de las mismas por terceros sin necesidad de obtener la autorización de sus respectivos titulares.

Requisitos Legales, de Derecho de Autor y otros

- **RNF 27.** Como producto, GeneSIG se distribuye amparado por las normativas legales establecidas en el país para la definición de los derechos de autor emitidas por el Decreto - Ley No. 14 de 1977.

[3]World Wide Web Consortium.

[4]Lenguaje Estructurado de PostgreSQL (del término en inglés PGSQL: PostgreSQL Structured Query Language).

Estándares aplicables

- **RNF 28.** El sistema será desarrollado bajo estándares OpenGIS como aseguramiento de la parte científica y en el desarrollo se codificará y modelará siguiendo los patrones de las normativas ISO[5], tanto de codificación como de diseño de bases de datos.

2.2.3. Diagrama de casos de uso del sistema

Una vez recopilados los requisitos funcionales del sistema es necesario conformar el DCUS[6]. [Pressman, 2002] define los casos de uso como un conjunto de escenarios que identifican una línea de utilización para el sistema que va a ser construido y que facilitan una descripción de cómo el sistema se usará. Por su parte, un DCUS muestra la relación entre los casos de uso y los actores del sistema. Para el catálogo de mapas se definen 12 casos de uso. A continuación se muestra el diagrama:

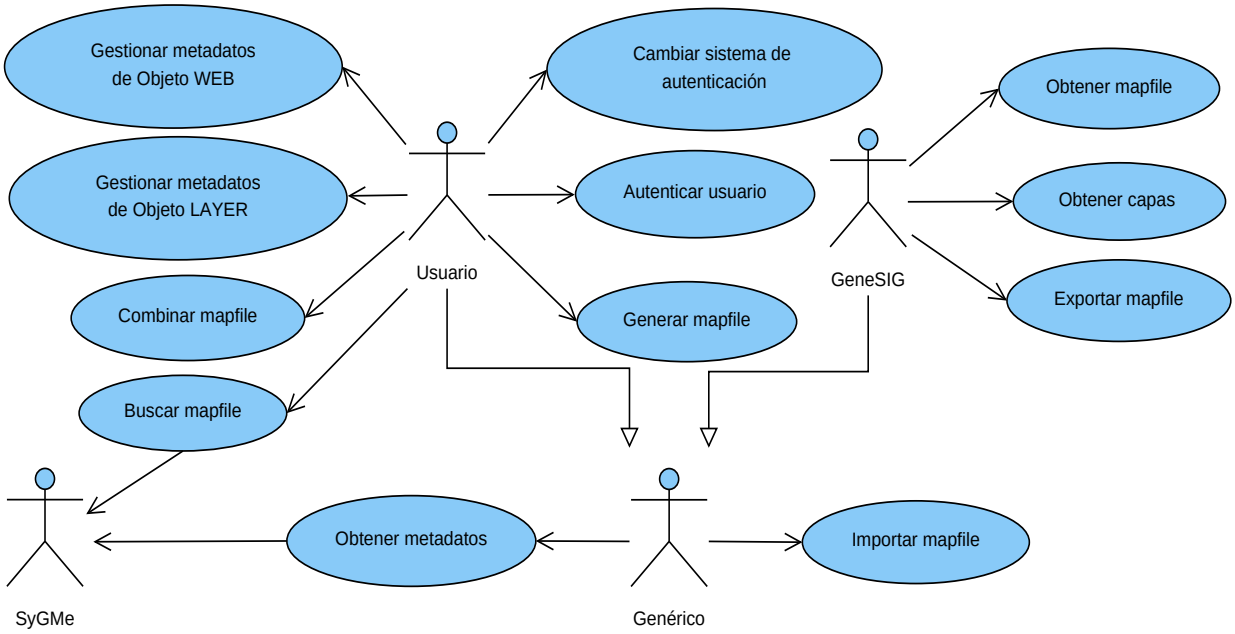

Figura 2.2: Diagrama de casos de uso del catálogo de mapas.

[5]Organización Internacional para la Estandarización (del témino en inglés ISO: International Standard Organization).
[6]Diagrama de Casos de Uso del Sistema.

2.2.4. Descripción de los actores del sistema

Los actores son personas, organizaciones u otros sistemas automatizados que interactúan directamente con la aplicación y están identificados con un rol en dependencia del papel que asuman dentro de la misma. Son los que interactúan con el sistema a través de los casos de uso que inicializan. Para el catálogo se definen 4 actores que a continuación se describen:

Actor	**Descripción**
Usuario	Usuario responsable de manejar los ficheros mapfile del catálogo de mapas. Puede importar, exportar, editar, eliminar, organizar y replicar los ficheros .map que se almacenan en el catálogo, según el nivel de acceso que tenga a las funcionalidades.
GeneSIG	Usuario que representa a la plataforma GeneSIG. Interactúa con los ficheros mapfile que se gestionan en el catálogo a través de los servicios web que brinda el mismo (casos de uso que inicializa) con el objetivo de personalizar el acceso a los mapas.
Genérico	Actor ficticio de los que heredan los actores Usuario y GeneSIG para inicializar casos de uso que resultan comunes para ambos.
SyGMe	Representa al Sistema de Gestión de Metadatos a través del cual el catálogo obtiene los metadatos de un mapa o una capa y a partir de ellos puede realizar búsquedas.

Tabla 2.1: Descripción de los actores del sistema

2.2.5. Descripción de los casos de uso del sistema

Dado que los casos de uso son la entrada esencial para realizar el análisis, diseño, implementación y pruebas del sistema, se hace necesario realizar una descripción de los mismos. Debido a que es muy amplia la información que se maneja en el catálogo de mapas, a continuación se presenta una breve descripción del caso de uso Buscar mapfile, para ver la descripción de los restantes casos de uso consultar el Anexo A.

Caso de uso:	**Buscar mapfile**
Actores:	Usuario
Resumen:	El caso de uso inicia cuando el usuario desea buscar un mapfile. El usuario selecciona el módulo "Configuración", luego la opción "Buscar mapfile", con el objetivo de buscar un mapfile a partir de los metadatos que gestiona el SyGMe.
Precondiciones:	El usuario debe estar autenticado y poseer los privilegios necesarios para acceder a esta opción.
Referencias:	RF 10
Prioridad:	Crítico
Flujo normal de los eventos	
1. El caso de uso comienza cuando el usuario selecciona la opción "Buscar mapfile"del módulo "Configuración".	2. El sistema muestra una ventana para que el usuario introduzca el valor de uno de los descriptores de los metadatos.
3. El usuario elige el descriptor del metadato e inserta su valor.	4. El sistema realiza la búsqueda y lista los ficheros mapfile encontrados.
5. El usuario elige un mapfile y pulsa el botón "Editar".	6. El sistema lo despliega en el panel izquierdo del menú navegación y muestra la ventana para su edición.
Flujo alterno de los eventos	
3.1 El usuario omite alguno de los campos obligatorios para realizar la búsqueda.	4.1 El sistema muestra un mensaje comunicando que el formulario no es válido y marca los campos requeridos en rojo.
3.2 El usuario introduce el valor del descriptor elegido incorrectamente.	4.2 El sistema muestra un mensaje comunicando que el valor no es válido e indica cómo debe ser.
Postcondiciones:	Se encuentra el mapfile.

Tabla 2.2: Descripción del caso de uso Buscar mapfile

2.3. Conclusiones parciales

Los artefactos que se generaron en el presente capítulo permiten contar con una documentación amplia y organizada de los principales conceptos que se manejan en el sistema y sus relaciones, así como las restricciones con las que cuenta el mismo para su futura implementación. Se generó el modelo de dominio del sistema, ya que no se tienen bien definidos los procesos del negocio, lo que permitió identificar los problemas existentes. En el levantamiento de requisitos se obtuvieron las funcionalidades con las que debe contar el sistema, y a partir de ellos se modeló la vista de casos de uso del sistema, que permite definir como será el comportamiento de los usuarios con respecto a la aplicación. Se considera que luego de haber presentado la propuesta de solución se está en condiciones de pasar a la implementación de la misma.

Capítulo 3

Implementación y validación

En el presente capítulo se presentan los artefactos ingenieriles relacionados con el diseño, implementación y validación del sistema: el modelo de diseño, el modelo de datos, el modelo de implementación, el modelo de despliegue y el modelo de casos de prueba. Se abunda sobre el funcionamiento de los frameworks seleccionados, y se explica cuáles son los patrones utilizados para el desarrollo del sistema. Se presenta una estrategia de arquitectura para la integración del catálogo con otros sistemas desarrollados por la línea de los SIG.

3.1. Diseño del catálogo

Con la finalidad de situar el dominio del problema dentro de la perspectiva de los desarrolladores, RUP propone realizar el análisis y diseño del sistema. [Pressman, 2002] afirma que el análisis tiene la misión de estudiar los requisitos, refinarlos y estructurarlos, con el objetivo de conseguir una comprensión más precisa de los mismos. Por su parte, el diseño es una representación abstracta de lo que se va a construir, contribuye a formar una arquitectura sólida y ayuda a crear un plano para la implementación. Mediante él se modela el sistema para que soporte todos los requisitos incluyendo los no funcionales.

El propósito y objetivo del análisis debe alcanzarse de algún modo en todo proyecto. Pero la manera exacta de ver y de emplear el análisis puede diferir de un proyecto a

otro y una de las variantes que se pueden emplear es no utilizar en absoluto el modelo de análisis para describir los resultados del análisis. En cambio el proyecto analiza los requisitos como parte integrada de la captura de requisitos o en el diseño [Jacobson, 2000]. [IBM, 2007] hace referencia a los seis principios que define RUP. Uno de ellos plantea que la metodología se debe ajustar al tamaño del proceso y a las necesidades del proyecto.

Considerando este principio que afirma que la metodología es altamente configurable, y que por tanto hay artefactos que pueden obviarse durante el desarrollo, se decide hacer una transición directa del flujo de trabajo de requisitos al diseño, sin necesidad de realizar el análisis. Se llega a esta conclusión pues el resultado que se obtendrá mediante el modelado del análisis no tendrá similitud con el diseño y no constituirá una base para realizar el mismo, dado que el framework Symfony tiene sus particularidades con respecto al diseño y arquitectura del software; los requisitos son bien conocidos y se cuenta con cierta comprensión de ellos; el hecho de que los lenguajes de programación, los componentes reutilizables, los frameworks, y en general, las tecnologías sobre las que se estará desarrollando el sistema sean ya elementos conocidos, hace prescindible la necesidad de refinar los requisitos acercándolos más al lenguaje del desarrollador, que es el principal objetivo de realizar el análisis. Además, en el proyecto GeneSIG y en la versión 1.0 del catálogo de mapas no se utilizó el modelo de análisis como parte del expediente de proyecto.

En los siguientes subepígrafes se dará una breve descripción de como está diseñada la estructura del sistema utlizando los frameworks anteriormente seleccionados.

3.1.1. Modelo de integración entre LiberMaps y GeneSIG

El catálogo de mapas LiberMaps se concibe como un producto con autonomía suficiente como para funcionar de manera independiente a un SIG, pero que al mismo tiempo debe contar con los protocolos necesarios para comunicarse con sistemas externos y brindar servicios que permitan personalizar tanto los datos como las funcionalidades a cada usuario. El objetivo de integrar el catálogo de mapas con un SIG es lograr la personalización del SIG de acuerdo a los usuarios, teniendo en cuenta el perfil de cada uno,

controlando el acceso a los datos sensibles y a las funcionalidades disponibles en cada caso. En este epígrafe se presenta un modelo para la integración del catálogo de mapas con la plataforma GeneSIG, aunque también es válido para la integración con cualquier sistema desarrollado por la línea de los SIG.

El diagrama UML que se presenta en la figura 3.1 constituye una representación básica de cómo deberá ser la comunicación entre el catálogo de mapas LiberMaps y la plataforma GeneSIG, de la misma forma pudiera realizarse la integración con otro SIG. La comunicación entre los componentes se realizará mediante el protocolo SOAP[1], basado en XML y creado precisamente para la comunicación entre aplicaciones. Es independiente de la plataforma que lo utilice.

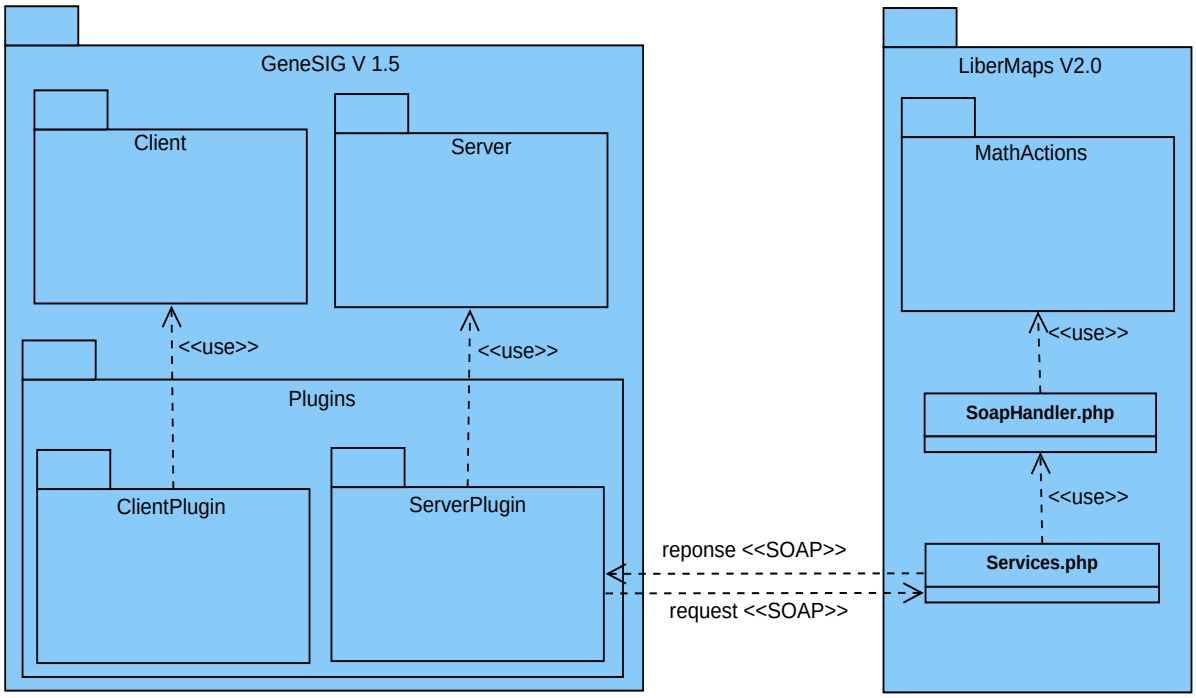

Figura 3.1: Modelo para la integración entre LiberMaps y GeneSIG.

Descripción de los principales componentes del modelo

Los componentes que se representan en la figura 3.1 se describen a continuación:

[1]Protocolo de Acceso a Objetos Simples (del término en inglés SOAP: Simple Object Access Protocol).

- **GeneSIG 1.5**: El paquete representa a la plataforma GeneSIG como cliente del servicio. Este sistema utiliza para su desarrollo el framework Cartoweb, que divide al servidor en dos subsistemas: un cliente, representado por el paquete ***Client*** y un servidor representado por el paquete ***Server***. El cliente es el encargado de la interacción con los datos provenientes del navegador, tiene la tarea de procesarlos y enviar al servidor la información correspondiente al tratamiento espacial. La plataforma se divide en varios plugins, paquetes modulares de archivos que se utilizan para llevar a cabo una acción especializada. El paquete ***Plugins*** representa al módulo de GeneSIG que hará uso de los servicios web que brinda el catálogo de mapas.

- **LiberMaps 2.0**: El paquete representa al catálogo de mapas como proveedor de los servicios para la integración con GeneSIG u otro SIG. Contiene los archivos principales que genera el framework Symfony, a través del plugin *ckWebServicePlugin* para la implementación de servicios web utilizando SOAP. El fichero ***Services.php*** es el script del controlador generado en conjunto con el WSDL[2] utilizado para describir la interfaz pública de los servicios web, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios. Este fichero utiliza la clase ***SoapHandler.php***, que será registrada como manejador de los requerimientos de los servicios y que a su vez hace uso del paquete ***MathActions***, módulo de la aplicación donde se ubica la clase Action que implementa las funcionalidades que constituyen servicios.

Descripción de los servicios para la integración

GetCatalog: Mediante este servicio se obtienen todos los ficheros mapas almacenados en la base de datos del catálogo.

- **Entrada:** No tiene parámetros de entrada.

[2]Lenguaje de Descripción de Servicios Web (del término en inglés WSDL: Web Service Description Language).

- **Salida:** XML con una relación de todos los mapas del catálogo.

GetMapsByUser: Mediante este servicio se obtienen los mapas a los que tiene acceso un usuario determinado.

- **Entrada:** Objeto *UserData* que tiene como parámetros el nombre del usuario y la contraseña.
- **Salida:** XML con una relación de todos los mapas del catálogo a los que tiene acceso el usuario que se indica (se muestra el identificador y nombre del mapa, además de las capas que lo integran).

GetLayersByUser: Mediante este servicio se obtienen todas las capas de un mapa determinado a las que tiene acceso un usuario.

- **Entrada:** Objeto *LayersByUser* que tiene como parámetros el identificador del mapa y un objeto *UserData* formado por el nombre del usuario y la contraseña.
- **Salida:** XML con una relación de las capas a las que tiene acceso el usuario que se indica (se muestra el identificador y el nombre de cada capa).

ExportMapByUser: Este servicio permite exportar un mapa almacenado en el catálogo.

- **Entrada:** Objeto *ExportData* que tiene como parámetros el identificador del mapa que se desea exportar y un objeto *UserData* formado por el nombre y contraseña de un usuario con los privilegios necesarios para realizar esta acción en el catálogo.
- **Salida:** XML con el mapa que se indica, incluyendo todos los objetos que lo componen.

ImportMapByUser: Este servicio permite almacenar un mapa en la base de datos del catálogo.

- **Entrada:** Objeto *ImportData* que tiene como parámetros una cadena con la ubicación del archivo .map y un objeto *UserData* formado por el nombre y contraseña de un usuario con los privilegios necesarios para realizar esta acción en el catálogo.

- **Salida:** No tiene salida, se almacena el mapa de manera persistente en la base de datos del catálogo.

GetMetadataByMap: Este servicio permite obtener los metadatos de un mapa determinado.

- **Entrada:** Objeto *MetadataByMap* que tiene como parámetros el identificador del mapa y un objeto *UserData* formado por el nombre del usuario y la contraseña.
- **Salida:** XML con los metadatos del mapa que se indica, incluyendo algunos de sus descriptores (identificador, nombre, norma que lo define y fecha de creación del metadato).

GetMetadataByLayer: Este servicio permite obtener los metadatos de una capa determinada.

- **Entrada:** Objeto *MetadataByLayer* que tiene como parámetros el identificador de la capa y un objeto *UserData* formado por el nombre del usuario y la contraseña.
- **Salida:** XML con los metadatos de la capa que se indica, incluyendo algunos de sus descriptores (identificador, nombre, norma que lo define y fecha de creación del metadato).

3.1.2. Estructura del catálogo con Symfony

El framework Symfony está basado en el patrón arquitectónico MVC[3], formado por tres niveles: el modelo, representa la información con la que se trabaja, es el encargado de manejar la lógica de los datos; la vista, permite al usuario interactuar con la información a través de una página web, y el controlador que es el que contiene toda la lógica del negocio, pues es quien recibe las peticiones del usuario y según ellas trasforma al modelo y a la vista. Symfony genera la estructura de los proyectos que lo emplean. Es por ello que la estructura del catálogo de mapas quedaría organizada como se explica a continuación.

[3]Modelo Vista Controlador.

Modelo: Se divide en las capas de acceso a datos y la capa de abstracción de la base de datos. La primera representa la lógica del negocio del sistema, es decir, las reglas del negocio y los requisitos funcionales que debe cumplir el sistema. La segunda capa brinda la posibilidad de no tener que generar sentencias SQL[4], lo que les facilita el trabajo a los programadores. Además de permitir que si se desea cambiar el gestor de base de datos, esto sea posible sin tener que realizar cambios en la lógica del negocio, para ello se auxilia del ORM Propel.

En las aplicaciones realizadas con Symfony, el acceso y la modificación de los datos almacenados en la base de datos se realiza mediante objetos; por lo que nunca se accede de forma explícita a la base de datos. Además, el framework genera por cada tabla de la base de datos, cuatro clases: Clase, ClasePeer, BaseClase, BaseClasePeer, como se muestra en la figura 3.2. Estas clases son las que se encargan en conjunto de realizar todo el acceso de los datos y la lógica de la aplicación.

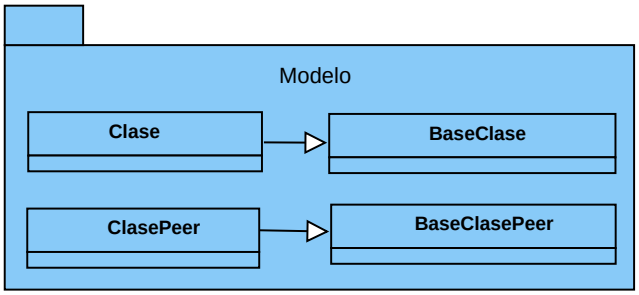

Figura 3.2: Estructura de las clases del modelo generadas por Symfony.

- Clase y ClasePeer: Se encargan de toda la lógica del negocio.
- BaseClase: Contiene todos los atributos definidos en la tabla y un conjunto de métodos ya implementados que gestionan el acceso a los datos, aceleran y simplifican el trabajo del equipo.
- BaseClasePeer: Contiene un conjunto de métodos estáticos que complementan el acceso y la lógica de los datos.

[4]Lenguaje de Consulta Estructurado (del término en inglés SQL: Structured Query Language).

Vista: En Symfony, la vista está formada principalmente por plantillas en PHP. Es aquí donde se transforma el modelo en una página Web con la que el usuario interactúa. Consta de tres partes fundamentales:

- Layout (plantilla global): Contiene el código HTML[5] que es común en todas las páginas, evitando de esta forma tener que repetirlo en todas las páginas.
- Complemento de las acciones (plantillas): Toman los resultados de la acción y se insertan en el cuerpo del layout para generar finalmente la página Web como resultado de la petición de un usuario.
- Páginas clientes y formularios: Son las páginas que se generan finalmente y con las que el usuario interactúa.

Controlador: Se encarga de procesar las interacciones del usuario y realizar los cambios apropiados en el modelo o en la vista. Es el responsable del manejo de las peticiones del usuario, cargar la configuración de la aplicación, el manejo de la seguridad, entre otras tareas. Se divide en un controlador frontal y las acciones.

- Acciones: Se encargan de obtener los resultados del modelo y definen variables para la vista. Representan una actividad específica que el sistema debe realizar, para ofrecer un resultado determinado a una petición. Las funciones que las componen se definen como *executeNombreAccion* y se encuentran agrupadas por módulos.
- Controlador Frontal: Es el único punto de entrada de la aplicación, carga la configuración y determina las acciones a ejecutarse. El controlador frontal, al igual que el layout es común para todas las acciones de la aplicación.

3.1.3. Estructura del catálogo con ExtJS

Las interfaces de la aplicación se realizarán con ExtJS. La utilización de esta librería de JavaScript se expresa a través del paquete ExtJS que representa los principales componentes de la propia librería, figura 3.3.

[5]Lenguaje de Marcado de Hipertexto (del término en inglés HTML: HyperText Markup Language).

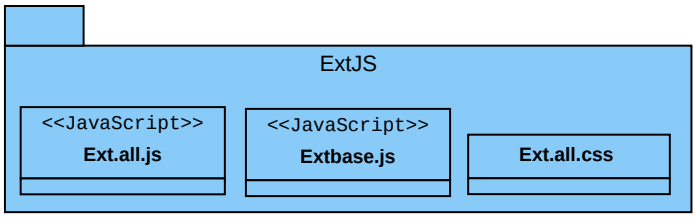

Figura 3.3: Paquete de la librería ExtJS.

Se utilizan un número de ficheros JS comunes para cada caso de uso, figura 3.4. Estos ficheros serán los encargados de configurar algunos elementos generales del funcionamiento de la aplicación. Al mismo tiempo en cada caso de uso se estará haciendo uso de otros ficheros JS específicos para cada uno, dichos ficheros se diseñarán en el propio caso de uso al que pertenezcan.

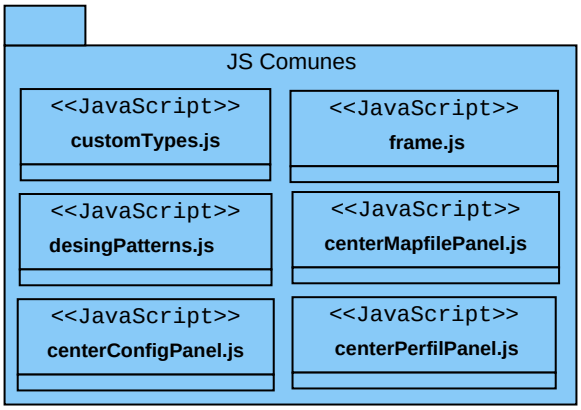

Figura 3.4: Paquetes JS utilizados en la aplicación.

3.1.4. Patrones de diseño utilizados

Los patrones de diseño constituyen la base para la búsqueda de soluciones a problemas comunes en el proceso de desarrollo de software. Se caracterizan por su reusabilidad, flexibilidad y aplicabilidad a diferentes problemas de diseño en diversas circunstancias.

Por su importancia en la aplicación, a continuación se describen los patrones utilizados en el diseño de la GUI[6] del catálogo de mapas LiberMaps, así como otros que implemen-

[6]Interfaz Gráfica de Usuario (del término en inglés GUI: Graphical User Interface).

ta el framework Symfony, como son los patrones GRASP[7], que describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades y los patrones GOF[8], ambos muy conocidos pues se utilizan en el desarrollo de gran cantidad de aplicaciones.

Patrones utilizados en la interfaz gráfica:

Patrón Fachada (Facade): El conjunto de formularios de edición de datos realiza funcionalidades comunes sobre los mismos a través de la realización de pedidos AJAX, invocando determinadas acciones que procesan las solicitudes realizadas al servidor. El patrón Fachada puede ser utilizado para proveer una interfaz sencilla y unificada que actúe como intermediaria entre los formularios (considerados clientes) y las acciones en el servidor.

Patrón Observador (Observer): Los formularios del catálogo cuentan con una gran cantidad de listas desplegables con valores de tablas de una base de datos. Los valores de estas tablas tienen la particularidad de que no son estáticos, sino que están sujetos a cambios por parte de los usuarios. En ExtJS, una vez que una lista desplegable es cargada desde el servidor, sus valores se almacenan en la caché del navegador para no repetir el llamado cada vez que se desplieguen. Por tanto, si se produce un cambio de alguno de sus valores en la tabla correspondiente, es necesario "informárselo"de alguna forma para que se recargue nuevamente. El patrón Observador se caracteriza por tener un objeto (el *Subject*) que mantiene una lista de objetos dependientes (los *Observers*), a los cuales notifica de cualquier cambio de su estado.

Patrones GOF implementados por Symfony [Ledea, 2010]:

Patrón Fábrica Abstracta: Permite trabajar con objetos de distintas familias de manera que las familias no se mezclen entre sí y haciendo transparente el tipo de familia concreta que se esté usando. Por ejemplo, en el caso de que Symfony necesite crear un

[7]Patrones Generales de Asignación de Responsabilidad de Software (del término en inglés GRASP: General Responsibility Assignment Software Patterns).

[8]Banda de los Cuatro (del término en inglés GOF: Gang of Four).

nuevo objeto para una petición, busca en la definición de la factoría el nombre de la clase que se debe utilizar para esta tarea.

Patrón Instancia Única (Singleton): Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Symfony lo hace a través de la llamada a *sfContext::getInstance()*.

Patrones GRASP implementados por Symfony [Ledea, 2010]:

Patrón Creador: Se asigna la responsabilidad de que una clase B cree un objeto de la clase A. Las clases Actions de Symfony son las responsables de crear los objetos que instancian las clases del modelo. Por tanto, son "creadoras"de dichas entidades.

Patrón Experto: La responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados. Symfony implementa este patrón a través de la capa de abstracción de datos que provee el ORM Propel que encapsula toda la lógica de los datos y genera el esquema para la base de datos junto a todas las clases del modelo y las funcionalidades que son comunes.

Patrón Controlador: Asigna la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. En este caso el framework utiliza un controlador frontal para manejar las peticiones web. Este controlador utiliza el enrutamiento para asociar el nombre de una acción y un módulo utilizando la URL[9] definida por el usuario.

Patrón Alta Cohesión: Este patrón define que cada elemento del diseño debe realizar una labor única dentro del sistema.

3.1.5. Diagramas de clases del diseño

Con el objetivo de manejar las clases que participan en el diseño de un caso de uso y las relaciones que entre ellas existen se utilizan los diagramas de clases del diseño. De esta forma se pueden representar los elementos que intervienen en una realización de un caso de uso [Jacobson, 2000]. En la figura 3.5 se presenta el diagrama de clases del diseño del caso de uso Buscar mapfile, para ver los diagramas del resto de los casos de uso consultar

[9]Localizador Uniforme de Recursos (del término en inglés URL: Uniform Resource Locator).

el Anexo B.

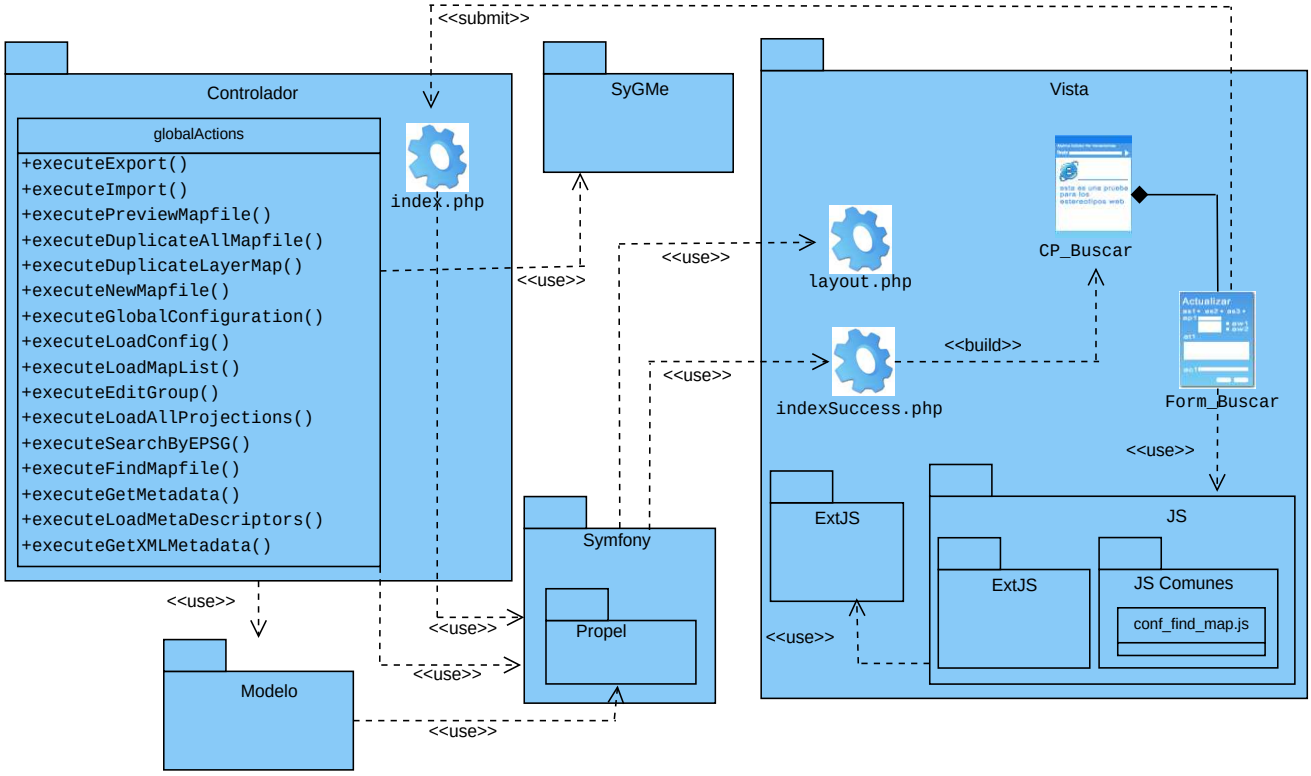

Figura 3.5: Diagrama de clases del diseño del caso de uso Buscar mapfile.

3.2. Modelo de datos

Para lograr que se gestionen correctamente los datos y que el usuario acceda a la información de manera rápida y eficiente se requiere de un buen diseño de la base de datos. [Pressman, 2002] plantea que el modelado de datos responde a una serie de preguntas importantes para cualquier aplicación de procesamiento de datos. Los métodos de modelado de datos hacen uso del diagrama de clases persistentes, Anexo C figura C.1, que hace referencia a las entidades lógicas y a sus atributos; y al DER[10], Anexo C figura C.2 que permite identificar objetos de datos y sus relaciones mediante una notación gráfica. En las figuras 3.6 y 3.7 se muestra una pequeña representación de las clases y tablas que conforman estos diagramas respectivamente.

[10]Diagrama Entidad Relación.

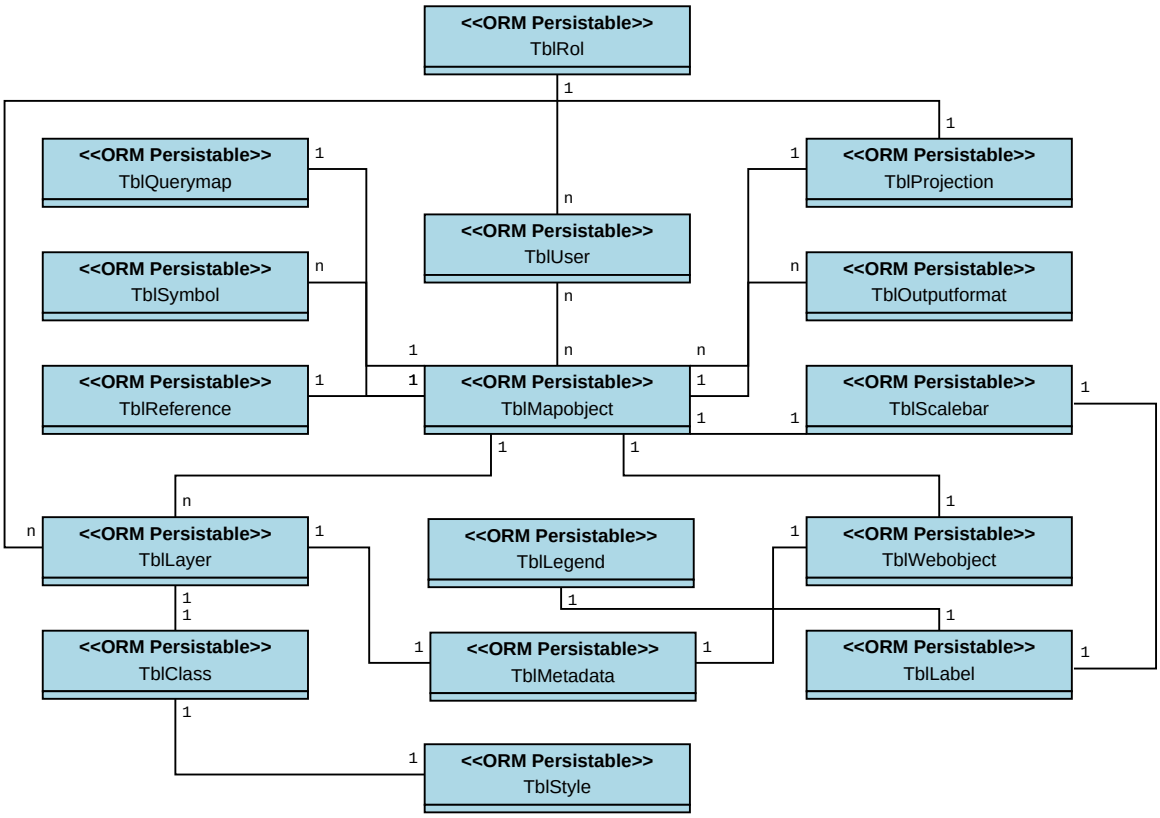

Figura 3.6: Representación de las clases persistentes de la base de datos.

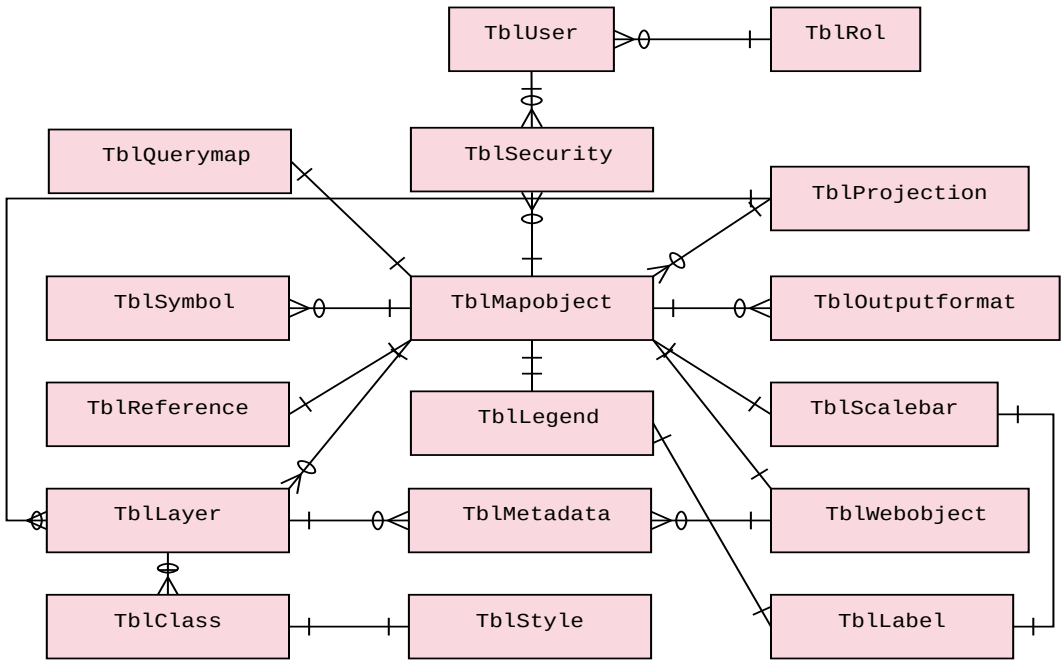

Figura 3.7: Representación de las tablas del Diagrama Entidad - Relación.

3.3. Modelo de despliegue

El modelo de despliegue provee un mapa de cómo debe distribuirse la instalación del sistema. Con el objetivo de representar los recursos de cómputo que necesita el catálogo de mapas para su correcto funcionamiento, en la figura 3.8 se presenta el diagrama de despliegue. Para la instalación del sistema se requiere de un servidor web, Apache, donde se ejecutarán tareas como la construcción de interfaces de usuarios y procesamiento de datos. Junto con el servidor web se encontrará el servidor de mapas, en este caso Mapserver. Se requiere además de un servidor de base de datos donde se ejecutará el SGBD PostgreSQL. Estos dos servidores se comunicarán a través del protocolo de transmisión de datos TCP/IP[11]. El cliente podrá acceder al sistema mediante el protocolo de transferencia de hipertexto HTTP[12]. La comunicación entre el servidor del catálogo de mapas con el servidor donde está instalada la plataforma GeneSIG será vía SOAP.

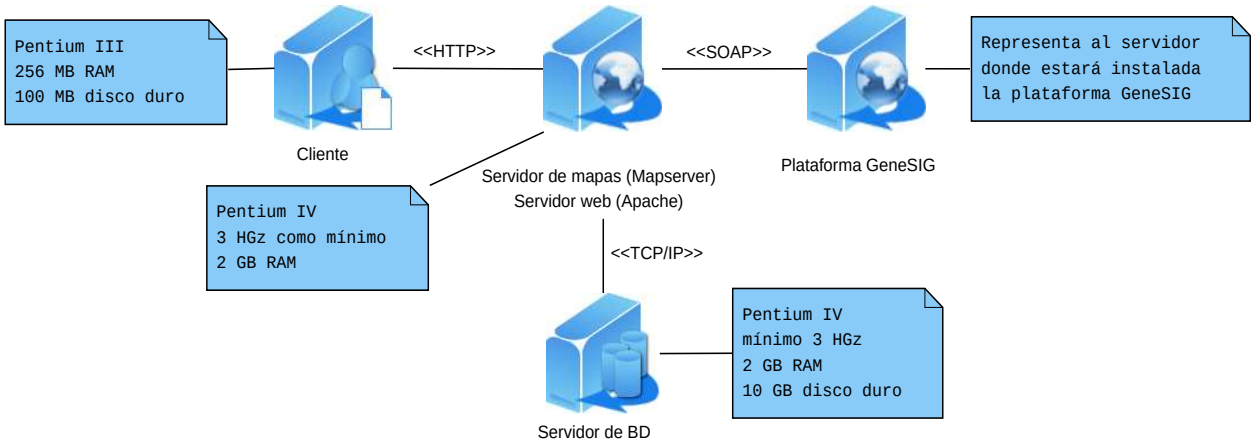

Figura 3.8: Diagrama de despliegue del catálogo de mapas.

3.4. Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes (base de datos, ejecutables, módulos o ficheros). Describe también cómo se organizan los componentes de acuerdo con los mecanismos

[11]Protocolo de Control de Transmisión/Protocolo de Internet (del inglés TCP/IP: Transmission Control Protocol/Internet Protocol).

[12]Protocolo de Transferencia de Hipertexto (del término en inglés HTTP: Hypertext Transfer Protocol).

de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros [Jacobson, 2000]. El diagrama de la figura 3.9 es un diagrama genérico que representa los principales componentes del sistema utilizados en la implementación.

- Template: Aquí se encuentra el componente *layout.php* que es la plantilla que se le aplica a todas las páginas. Esta hace uso de los ficheros definidos dentro del paquete ExtJS para conformar la vista y del componente *sfView.class.php* del paquete Symfony para mostrar dicha vista.

- Módulo: Este paquete está integrado por un componente *actions.class.php* (representando al controlador) y un componente *indexSuccess.php* (representando a la vista). El primero se relaciona con las clases *Clase.php* y *ClasePeer.php* del paquete Modelo para el acceso a los datos de la base de datos, además hace uso del componente *sfAction.class.php* que trae Symfony por defecto. Por otro lado, la vista utiliza el componente *layout.php* que está en el paquete Template para darle forma a la página.

- Modelo: Contiene las clases responsables de manejar la información con las tablas de la base de datos. Utiliza el componente Propel para la abstracción a la base de datos.

- Symfony: Las librerías a usar para el desarrollo de la aplicación teniendo en cuenta el patrón MVC se encuentran en esta carpeta: *sfView.class.php* para mostrar las vistas, *sfAction.class.php* para la controladora y Propel para el modelo.

- ExtJS: Este paquete contiene los principales componentes que conforman la librería JavaScript.

- GeneSIG: Representa a la plataforma GeneSIG que hace uso del componente *Services.php* para acceder a los servicios web, que a su vez hace uso del componente *SoapHandler.php*, clase que controla los parámetros de cada servicio y que se comunica con el componente *actions.class.php* donde se implementan los servicios.

- SyGMe: Representa al Sistema de Gestión Metadatos del cual el catálogo hace uso para obtener los metadatos que dicho sistema maneja.
- PostgreSQL: Representa a la base de datos del catálogo.

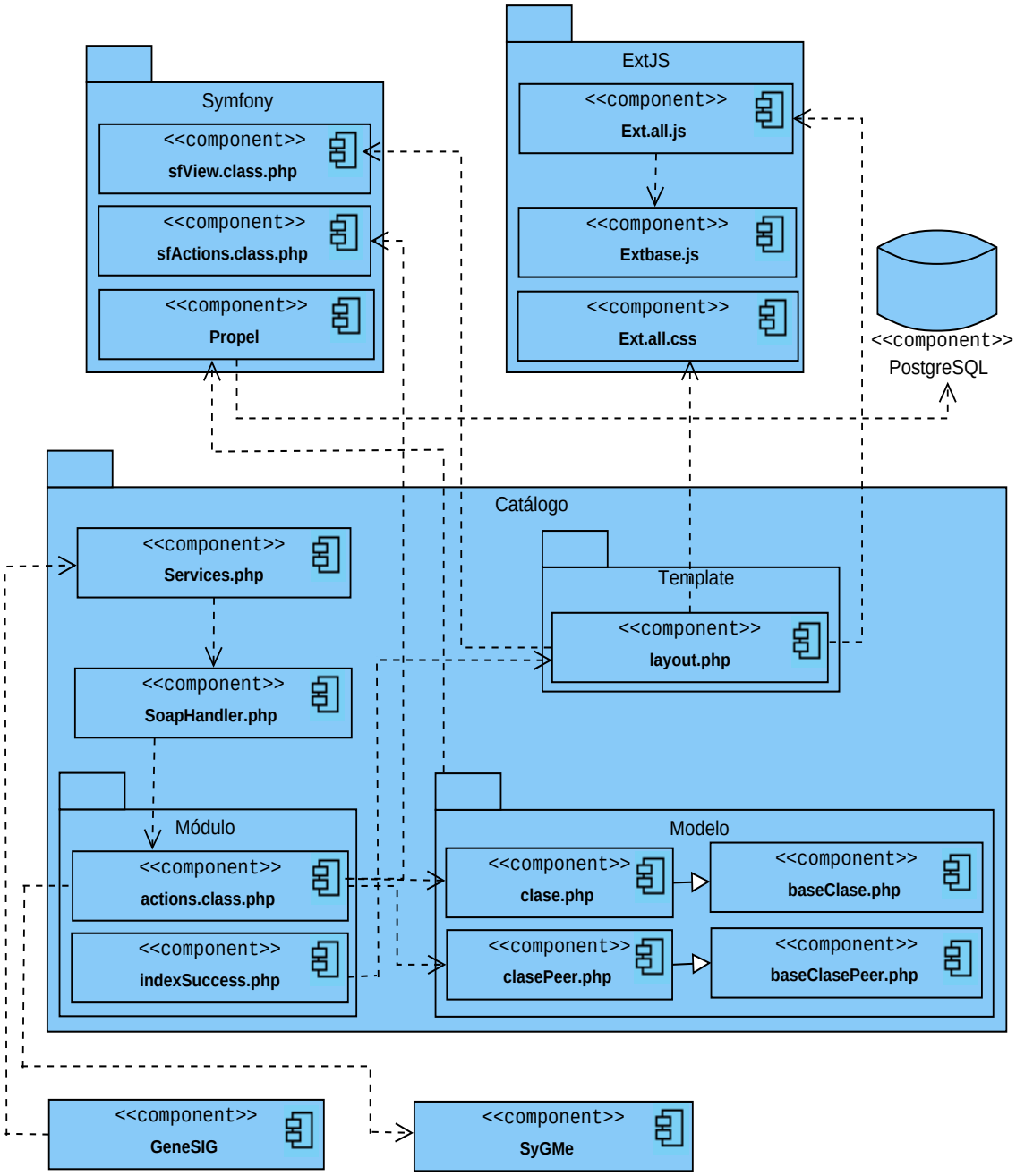

Figura 3.9: Diagrama de componentes del catálogo de mapas.

3.5. Modelo de pruebas

Una vez finalizada la implementación de la versión 2.0 del catálogo de mapas LiberMaps, se hace necesario comprobar que su funcionamiento sea adecuado, por lo que se debe verificar que las operaciones internas se ajustan a las especificaciones planteadas. Para lograrlo se aplicarán las pruebas de caja negra.

Esta técnica de prueba se centra en los requisitos funcionales del software, permitiendo obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Las mismas son de gran utilidad pues permiten encontrar en el software varios tipos de errores: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos externas, errores de rendimiento y errores de inicialización y de terminación. Además, con su aplicación se comprueba si el sistema es sensible a ciertos valores de entrada y qué efectos tiene sobre el mismo la entrada de ciertas combinaciones de datos [Pressman, 2002], aspectos que son necesarios a la hora de validar el buen funcionamiento del sistema.

En concreto se hará uso de la técnica de partición equivalente, que se basa en una evaluación de las clases de equivalencia para una condición de entrada. [Pressman, 2002] afirma que una clase de equivalencia representa un conjunto de estados válidos o inválidos para condiciones de entrada. En los siguientes subepígrafes se muestra la descripción de las pruebas aplicadas al caso de uso Buscar mapfile, para ver las pruebas aplicadas al resto de los casos de uso consultar el Anexo D.

3.5.1. Descripción general del CUS Buscar mapfile

El caso de uso comienza cuando el usuario accede al sistema para buscar un mapfile.

Condición de ejecución: El usuario debe estar autenticado en el sistema y poseer los privilegios necesarios para ejecutar esta acción.

Sección	**Escenarios**	**Descripción de la sección**

SC1: Buscar mapfile	EC 1.1: Buscar mapfile con éxito	El usuario autenticado accede a la funcionalidad y el sistema muestra una interfaz para introducir los datos requeridos. Cuando el usuario oprime el botón "Buscar"se hace una búsqueda y se listan los ficheros mapfile encontrados.
	EC 1.2: Buscar mapfile falla	El sistema emite un mensaje de error cuando se deja algunos de los campos en blanco, el valor de los campos no es válido o no se encuentra ningún mapfile.

Tabla 3.1: Escenarios del caso de uso Buscar mapfile

3.5.2. Descripción de las variables del CUS Buscar mapfile

No	**Nombre del campo**	**Clasificación**	**Valor nulo**	**Descripción**
1	Descriptor	Lista desplegable	No	Permite seleccionar el descriptor de un metadato.
2	Valor	Campo de texto	No	Acepta una cadena de caracteres que indica el valor del descriptor seleccionado.
3	Mapfile	Grid Panel	Si	Muestra un listado con los ficheros mapfile encontrados.
4	Buscar	Botón	No	Permite realizar la búsqueda del mapfile.

Tabla 3.2: Variables del caso de uso Buscar mapfile

3.5.3. Matriz de datos del CUS Buscar mapfile

Escenario	**Var 1**	**Var 2**	**Respuesta**	**Resultado**	**Flujo central**

EC 1.1: Buscar mapfile con éxito	(Título) V	(Guantánamo) V	El sistema busca el mapfile a partir del descriptor y el valor indicado.	Satisfactorio	1. El usuario accede al sistema para buscar un mapfile. 1.1. El sistema muestra una interfaz para introducir los datos: descriptor y valor. 2. El usuario inserta los datos anteriores. 2.1. El sistema realiza la búsqueda y lista el resultado.
	(Fecha creación) V	(2012-03-30) V	El sistema busca el mapfile a partir del descriptor y el valor indicado.	Satisfactorio	
EC 1.2: Buscar mapfile falla	(vacío) I	(Guantánamo) V	El sistema muestra un mensaje de error indicando que debe especificar el metadato.	Satisfactorio	
	(Titulo) V	(vacío) I	El sistema muestra un mensaje de error indicando que debe especificar el metadato.	Satisfactorio.	
	(Titulo) V	(aaaa) V	El sistema muestra un mensaje de error indicando que no se ha encontrado el mapa.	Satisfactorio	

	(Fecha creación) V	(30-03-2012) I	El sistema muestra un mensaje de error indicando que la fecha debe tener el formato AAAA-MM-DD.	Satisfactorio	

Tabla 3.3: Matriz del caso de uso Buscar mapfile

3.6. Conclusiones parciales

Los artefactos generados en el presente capítulo constituyen la base sobre la que se construye la versión 2.0 del catálogo de mapas LiberMaps. A través del modelo de diseño se logró representar la estructura del sistema teniendo en cuenta los frameworks sobre los que se desarrolla. Se cuenta con una especie de guía para la instalación del sistema como resultado de generar el modelo de despliegue con los diferentes nodos físicos donde finalmente se instalará el mismo. Por su parte, la elaboración del modelo de datos permitió representar los elementos del problema y sus relaciones entre sí. El modelo de componentes diseñado permitió representar los elementos antes descritos en el diseño en términos de componentes de software, permitiendo mostrar así los componentes necesarios para la implementación del sistema. Además, las pruebas realizadas al software han mostrado su correcto funcionamiento.

Conclusiones

En los últimos años el uso de la informática para facilitar el trabajo con la información geográfica ha aumentado considerablemente, haciendo necesario facilitar su creación, manipulación y visualización. Mediante la presente investigación se ha logrado desarrollar la versión 2.0 del catálogo de mapas LiberMaps, herramienta que facilita la búsqueda y edición de la información georreferenciada, siendo este el objetivo principal de dicha investigación.

Con la creación de esta nueva versión se logra contar con un sistema que permite editar dinámicamente los ficheros de configuración mapfile, imprescindibles en el funcionamiento de Mapserver, servidor de mapas que utiliza GeneSIG. Pero que además cuenta con las funcionalidades necesarias para lograr la comunicación con aplicaciones SIG, como lo es GeneSIG, con el objetivo de personalizar tanto los datos como las funcionalidades a cada usuario. La comunicación que mantiene con el SyGMe permite la realización de búsquedas, funcionalidad que resulta de gran importancia cuando se trata de grandes cantidades de información.

El sistema fue desarrollado utilizando herramientas multiplataforma, lo que permite que pueda ser desplegada sobre entornos libres. El hecho de haber sido desarrollada sobre una arquitectura web garantiza la explotación de los beneficios de la red, minimizando el costo en cuanto a hardware y software de las entidades que utilicen el producto. Su creación contribuye a la implementación de la IDERC, y permite contar con herramientas cubanas que faciliten no sólo el manejo de la información georreferenciada, sino que además responda a los intereses del país, y que esté respaldada por un equipo de desarrollo cubano que pueda dar soporte y mantenimiento a esta solución.

Recomendaciones

Una vez vencidos los objetivos de esta investigación, y teniendo en cuenta las experiencias obtenidas a lo largo de su desarrollo se recomienda a la institución:

- Generalizar el campo de aplicación del catálogo de mapas LiberMaps para la integración con otros SIG desarrollados por la línea en el centro.
- Seguir trabajando en el desarrollo del sistema, añadiéndole nuevas funcionalidades que faciliten el trabajo con los ficheros mapfile.

Referencias bibliográficas

[Ballari, 2006] Ballari, D. (2006). Puesta en marcha y explotación de geoservicios del opengeospatial consortium: Curso teórico-práctico con tecnologías open source. page 20.

[Bravo, 2000] Bravo, J. D. (2000). *Breve Introducción a la Cartografía y a los Sistemas de Información Geográfica*.

[Camy, 2002] Camy, L. I. (2002). *La biblia de JavaScript. [En línea] http://bibliodoc.uci.cu/pdf/reg00541.pdf*.

[Chacón, 2006] Chacón, J. C. R. (2006). Aplicación de la metodología rup para el desarrollo rápido de aplicaciones. guatemala : Universidad de san carlos de guatemala. facultad de ingeniería. escuela de ingeniería en ciencias.

[Costa, 2010] Costa, A. L. (2010). *Módulo para la gestión de metadatos geográficos de LiberMaps*. PhD thesis, Universidad de las Ciencias Informáticas.

[Delgado, 2005] Delgado, T. (2005). *Infraestructuras de Datos Espaciales en países de bajo desarrollo tecnológico. Implementación en Cuba. Tesis presentada en opción al grado científico de Doctor en Ciencias Técnicas.* PhD thesis.

[Delgado, 2009] Delgado, T. (2009). La infraestructura de datos espaciales de la república de cuba, avances y perspectivas. novena conferencia cartográfica regional de las naciones unidas para américa.

[Espinosa, 2008] Espinosa, A. M. (2008). Fundamentos del mapserver, mapscript, postgis y su integración con el cartoweb. page 30.

[Guevara, 2001] Guevara, A. (2001). Esquema metodológico para el diseño e implementación de un sistema de información geográfica.

[IBM, 2007] IBM, C. (2007). *Ayuda de Rational Unified Process.*

[Jacobson, 2000] Jacobson, I. (2000). *El proceso unificado de desarrollo software. Madrid. Series Editors.*

[Kropla, 2005] Kropla, B. (2005). *Beginning MapServer Open Source GIS Development.*

[Ledea, 2010] Ledea, L. M. (2010). *Diseño de la arquitectura base del catálogo de mapas LiberMaps.* Universidad de las Ciencias Informáticas.

[Llopis, 2008] Llopis, J. P. (2008). Sistemas de información geográfica aplicados a la gestión del territorio. [en línea] http://www.editorial-club-universitario.es/libro.asp?ref=3775.

[Álvarez, 2000] Álvarez, C. (2000). *Metodología de la Investigación Científica. Santiago de Cuba. Universidad de Oriente.*

[MapServer, 2008] MapServer (2008). *Mapserver documentation release 5.6. [En línea] http://mapserver.gis.umn.edu.*

[Nebert, 2001] Nebert, D. D. (2001). *Global Spatial Data Infrastructure. El Recetario IDE. [En línea] http://www.gsdi.org.*

[OGC, 2007] OGC (2007). *OpenGIS® Catalogue Services Specification. [En línea] http://www.opengeospatial.org/standards.*

[O'Reilly, 2005] O'Reilly (2005). *Web Mapping Illustrated @ Team.*

[Ortiz, 1999] Ortiz, V. (1999). Nuevas perspectivas para la catalogación: Metadatos versus marc. [en línea] http://redc.revistas.csic.es/index.php/redc/article/viewarticle/338.

[Padrón, 2001] Padrón, D. (2001). Empleo de servidores cartográficos en internet.

[Pombert, 2006] Pombert, A. T. (2006). ¿catalogación en el entorno digital?: una breve aproximación a los metadatos. [en línea] http://scielo.sld.cu/scielo.php.

[Pressman, 2002] Pressman, R. (2002). *Ingeniería del Software. Un enfoque práctico. Quinta edición.*

[Progress, 2009] Progress (2009). Progress, software y servicios. [en línea] http://www.progress.com.py/rup.php.

[RAE, 2010] RAE (2010). *Diccionario de la Lengua Española. Vigésima segunda edición. [En línea] http://www.rae.es/rae.html.*

[Salinas, 2007] Salinas, J. (2007). *Información Geográfica, Software Libre e Infraestructuras de Datos Espaciales.*

[Sencha, 2011] Sencha (2011). Extjs. documentación ext js javascript library. [en línea] http://extjs.com/deploy/dev/docs.

Acrónimos

API Interfaz de Programación de Aplicaciones (del término en inglés API: Application Program Interface).

AJAX JavaScript Asíncrono y XML (del término en inglés AJAX: Asynchronous JavaScript And XML).

CASE Ingeniería de Software Asistida por Computadora (del término en inglés CASE: Computer Aided Software Engineering).

CGI Interfaz de Entrada Común (del término en inglés CGI: Common Gateway Interface).

CPU Unidad Central de Procesamiento (del término en inglés CPU: (Central Processing Unit).

CWS Catalog Web Services.

DCUS Diagrama de Casos de Uso del Sistema.

DER Diagrama Entidad Relación.

EPSG European Petroleum Survey Group.

GEYSED Centro de Desarrollo Geoinformática y Señales Digitales.

GIF Graphics Interchange Format.

GOF Banda de los Cuatro (del término en inglés GOF: Gang of Four).

GRASP Patrones Generales de Asignación de Responsabilidad de Software (del término en inglés GRASP: General Responsibility Assignment Software Patterns).

GUI Interfaz Gráfica de Usuario (del término en inglés GUI: Graphical User Interface).

HTML Lenguaje de Marcado de Hipertexto (del término en inglés HTML: HyperText Markup Language).

HTTP Protocolo de Transferencia de Hipertexto (del término en inglés HTTP: Hypertext Transfer Protocol).

IDE Infraestructura de Datos Espaciales.

IDERC Infraestructura de Datos Espaciales de la República de Cuba.

IMS Internet Map Server.

ISO Organización Internacional para la Estandarización (del témino en inglés ISO: International Standard Organization).

JPEG Joint Photographic Experts Group.

MVC Modelo Vista Controlador.

OGC Open Geospatial Consortium.

ORM Mapeo Objeto - Relacional (del término en inglés ORM: Object Relation Mapping).

PDF Formato de Documento Portátil (del término en inglés PDF: Portable Document Format).

PGDG PostgreSQL Global Development Group.

PHP PreProcesador de Hipertexto (del término en inglés PHP: Hypertext PreProcessor).

PNG Portable Network Graphics.

PGSQL Lenguaje Estructurado de PostgreSQL (del término en inglés PGSQL: PostgreSQL Structured Query Language).

RAM Memoria de Acceso Aleatorio (del término en inglés RAM: Random Access Memory).

RF Requisito Funcional.

RNF Requisito No Funcional.

RUP Proceso Unificado Racional (del término en inglés RUP: Rational Unified Process).

SGBD Sistema Gestor de Base de Datos.

SIG Sistema de Información Geográfica.

SQL Lenguaje de Consulta Estructurado (del término en inglés SQL: Structured Query Language).

SyGMe Sistema de Gestión de Metadatos.

SOAP Protocolo de Acceso a Objetos Simples (del término en inglés SOAP: Simple Object Access Protocol).

SWF Small Web Format.

TIC Tecnología de la Información y la Comunicación.

TCP/IP Protocolo de Control de Transmisión/Protocolo de Internet (del inglés TCP/IP: Transmission Control Protocol/Internet Protocol).

UCI Universidad de Ciencias Informáticas.

UCID Unidad de Compatibilización e Integración de Software para la Defensa.

UML Lenguaje Unificado de Modelado (del término en inglés UML: Unified Model Language).

URL Localizador Uniforme de Recursos (del término en inglés URL: Uniform Resource Locator).

WCS Web Coverage Service.

WFS Web Feature Services.

WMS Web Map Services.

WSDL Lenguaje de Descripción de Servicios Web (del término en inglés WSDL: Web Service Description Language).

W3C World Wide Web Consortium.

XML Lenguaje de Marcas Extensible (del término en inglés XML: Extensible Markup Language).