Universidad de las Ciencias Informáticas

Facultad 6

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

Título: "Módulo de configuración de *MapCache* para la plataforma GeneSIG"

Autor: Yaima Del Campo Peña

Tutor(es): MSc. Gerdys Ernesto Jiménez Moya

Ing. Alain León Companioni

La Habana, junio de 2015

"Año 57 de la Revolución"

"La única lucha que se pierde es la que se abandona. Si avanzo sígueme, si me detengo empújame, si retrocedo mátame."

"Seamos realistas y hagamos lo imposible."

Ernesto Che Guevara.

Declaración de Autoría

Declaro ser la legítima autora del trabajo titulado: “Módulo de configuración de Mapcache para la Plataforma GeneSIG”, y reconozco a la Universidad de las Ciencias Informáticas (UCI) con los derechos patrimoniales de la misma, con carácter exclusivo.

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

_____

Firma del Autor

Yaima Del Campo Peña

_____

Firma del Tutor

MSc. Gerdys Ernesto Pérez Moya

_____

Firma del Tutor

Ing. Alain León Companioni

Datos de Contacto

Nombre y Apellidos: Gerdys Ernesto Jiménez Moya

Correo electrónico: gejimenez@uci.cu

Año de graduado: 2008

Profesión: Ingeniero en Ciencias Informáticas

Nombre y Apellidos: Alain León Companioni

Correo electrónico: acompanioni@uci.cu

Año de graduado: 2010

Profesión: Ingeniero en Ciencias Informáticas

Agradecimientos

A mis padres por todo su amor, compresión, dedicación, apoyo y por recordarme en todo momento que yo si puedo y que lo lograría.

A mi novio José por todo su cariño, por estar siempre ahí apoyándome y sobre todo por tu paciencia con mi carácter y por darme fuerzas cuando creía que todo estaba perdido.

A toda mi familia en especial:

A mi abuelita Esther.

A mis tíos Onelia y Leo que fueron mis segundos padres aquí en La Habana, gracias por todo el apoyo y el cariño brindado.

A mi hermano Yole que ha sido un ejemplo a seguir para mí y a mi sobrino Brian que más que todo es mi hermanito pequeño, para ti todo mi cariño y amor.

A mi tío Arturo por apoyarme siempre e impulsarme a seguir adelante, tú eres mi otro papá; gracias por enseñarme que la vida te pone trabas pero lo importante es vencerlas y seguir adelante.

A mi primo Henry que me abrió las puertas de la UCI y me ayudó mucho a adaptarme a la universidad.

En especial a mi suegra Ida Emilia por quererme como a una hija.

A todos los amigos que conocí en la universidad, los que fueron compañeros de clases y a mis compañeras de apartamento en especial:

Anita y Danaysi que son mis amigas desde primer año para agradecerles no alcanzan las palabras. Con ustedes compartí buenos y malos momentos, largas horas de estudio y también momentos de alegría y fiesta. Gracias por su apoyo, por quererme y aceptarme como soy; gracias por ser mis amigas.

Agradecimientos

A Mayi y Paneque, otros dos de mis buenos amigos, Mayi eres una persona maravillosa, con una fuerza increíble, gracias por ser una amiga incondicional y por enseñarme a ser fuerte en la vida.

A mi gran amigo Nelson, gracias por ayudarme a seguir adelante.

A Maire, Cándido, Ramón y Susell que no pudieron terminar esta travesía, pero que significaron mucho en mi vida universitaria.

A todos ustedes gracias por su amistad, nunca los olvidaré.

A todos los que me ayudaron a llegar hasta aquí, los profesores de la carrera y a los compañeros y profesores que colaboraron con sugerencias y revisiones del documento.

Un agradecimiento muy especial a mis tutores, por hacerme correr tanto detrás de ellos y por su clásica frase "Tranquila todo va a salir bien". Gracias por el apoyo brindado y ser cómplices de este gran esfuerzo, por ofrecerme sus conocimientos y ayudarme a estar aquí hoy. Gracias por el tiempo que sacaron de su complicada agenda para dedicármelo y sobre todo por tener tanta paciencia conmigo.

Agradecerles también a todos los que he conocido durante estos cinco años de carrera, algunos muy buenos amigos, pero lo importante es que de todos aprendí algo.

Gracias a todos aquellos que estuvieron pendientes del desarrollo de mi Tesis, que en algún momento preguntaron: ¿Cómo va la Tesis?

Agradecerles también a todas las instructoras que me han acogido como una hija más en especial a Migdalia.

A todos muchas gracias.

Dedicatoria

Por su apoyo cariño y amor quiero dedicar esta tesis a mis padres por estar siempre presentes, confiar en mí y no dejar ni un segundo de apoyarme a seguir adelante.

A mi novio José por estar presente cuando el camino se tornó difícil y por su paciencia y dedicación.

Resumen

MapCache es un servidor de teselas que almacena las celdas de las imágenes solicitadas para su posterior uso, evitando así una sobrecarga en el servidor y contribuyendo a mejorar la rapidez en la visualización de los mapas. Actualmente es utilizado junto al servidor de mapas *MapServer* en la plataforma GeneSIG y el proceso de configuración de este servidor de teselas es complejo debido a que se realiza a través de un fichero *XML* de forma manual. Este fichero se encuentra en el servidor de aplicaciones, al cual solo pueden acceder aquellos que tengan credenciales, esto trae consigo que si algún especialista no las posee y necesita configurar el fichero deba realizarlo a través de un tercero que sí tenga acceso. Otro factor negativo es que existen pocos especialistas que dominan la configuración del fichero y el tiempo de configuración es elevado. Para darle solución a estos problemas mediante la siguiente investigación se implementó un módulo de configuración de *MapCache* para la plataforma GeneSIG, que permite configurar todos los parámetros del servidor de teselas a través de una interfaz visual. Además cuenta con una opción que permite adicionar un conjunto de celdas como capa a GeneSIG y visualizar en un mapa esta acción. El proceso de desarrollo de software estuvo guiado por la metodología *AUP*. La implementación se llevó a cabo utilizando herramientas y tecnologías de código abierto. Las pruebas de software aplicadas al módulo arrojaron resultados satisfactorios, permitiendo mejorar el proceso de configuración de *MapCache*.

Palabras claves: configuración, GeneSIG, *MapCache,* servidor de teselas.

Abstract

MapCache tiles is a server that stores the cells of the images ordered for later use, thus avoiding overloading the server and helping to improve the speed of displaying maps. It is currently used besides MapServer on GeneSIG platform and its configuration process is complex because it is done manually through an XML file. This file is in the application server, which can only be accessed by those with credentials, this entails that if a specialist does not has the credentials and it is en need to configure the file must do so through a third person who does have it. Another negative factor is that there are few specialists who master the configuration file and the setup time is high. For solving these problems through the present investigation a MapCache configuration module for GeneSIG platform was implemented, that allow to configure all server parameters tiles through a visual interface. It also has an option to add a set of cells as GeneSIG layer on a map and visualize this. The software development process was guided by AUP methodology. The implementation was carried out using tools and open source technologies. Software testing applied to the module yielded satisfactory results, allowing better MapCache configuration process.

Keywords: configuration, GeneSIG, MapCache, server tiles.

Índice

Índice de tablas

Índice de figuras

Introducción

El desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC), ha permitido la creación de herramientas que facilitan el trabajo con la información geográfica y la visualización de los mapas. Surgen así los Sistemas de Información Geográfica (SIG), "… *sistemas que integran tecnología informática, personas e información geográfica. Su función principal es capturar, analizar, almacenar, editar y representar datos referenciados geográficamente…"* (Olaya, 2010).

Los SIG actúan como elemento central para garantizar que todas las personas implicadas accedan a la información que contienen y lo hagan de la manera correcta. Tienen la capacidad de exponer de forma adecuada y a un gran público, datos geográficos. Además se pueden usar como una herramienta para la toma de decisiones en función de ciertas variables, y para modelar realidades geográficas complejas (Olaya, 2010). Debido a estas características, estos sistemas son útiles y versátiles desde diferentes puntos de vista y han tenido gran impacto en diversos sectores de la sociedad como la agricultura (Uva y Campanella, 2009), la salud (Cuéllar y otros, 2009) y el transporte (Bosque y Gómez, 2001).

En un SIG, la visualización de la información geográfica constituye un elemento fundamental, no solo porque facilita la interpretación de la información, sino que lo hace de una manera intuitiva para las personas. Los servidores de mapas son parte importante en este proceso, debido a que permiten a los usuarios visualizar los datos espaciales y consultar su información geográfica por medio de una aplicación (Monge y otros, 2010). En la actualidad existe una amplia variedad de servidores de mapas, entre ellos, *MapServer* (OsGeo, 2014a) y *GeoServer* (OsGeo, 2014b).

Aun cuando los servidores de mapas permiten visualizar los datos espaciales, es necesario mejorar la visualización de los mapas que estos brindan para un mejor entendimiento por parte de los usuarios. El *WMS*[1] es un estándar de la *OGC*[2] (OGC, 2014), que facilita el procesamiento de la información geográfica en los SIG. Su función principal es generar mapas de datos referenciados espacialmente a través de internet de forma dinámica, a partir de información geográfica. Los mapas producidos por *WMS* se crean

[1] Servicio de Mapas en la Web (del término en inglés WMS: Web Map Service)

[2] Open Geospatial Consortium

en un formato de imagen digital como *PNG*[3], *GIF*[4] o *JPEG*[5], o como gráficos vectoriales en formato *SVG*[6] o *WebCGM*[7] (OGC, 2006).

El principal inconveniente de este servicio es que raramente hay dos peticiones *WMS* iguales. Todos los usuarios del servicio realizan solicitudes, que aunque a veces son muy similares, no resultan idénticas ni en escala ni en ámbito geográfico. Debido a lo anterior, el servidor no puede aprovechar las respuestas anteriores para despachar rápidamente las peticiones nuevas. Además con el aumento del número de usuarios, el servidor recibe de manera concurrente diversas demandas para mapas similares, pudiendo provocar que colapse el servicio (Masó y otros, 2010).

Otra desventaja es que no aprovecha los mecanismos de *cache*, que permiten evitar nuevas consultas al servidor, almacenando en un servidor intermedio o en el propio cliente, las peticiones idénticas durante un tiempo determinado (Masó y otros, 2010). Como solución a dicho problema surge la idea de tener las imágenes pre-generadas basándose en un modelo de teselas[8]; con tal fin se desarrolló la especificación de *WMS* con teselas, *WMTS*[9] (Gámez, 2008) que actualmente es un estándar (Masó y otros, 2010) implementado por disímiles aplicaciones entre ellas *MapCache* (MapServer, 2014).

MapCache es un servidor de teselas, que fue creado como un producto independiente teniendo como objetivo brindarle al servidor de mapas *MapServer* un servidor de teselas. Actúa como *proxy* interceptando las peticiones que se realizan al servidor de mapas e implementa el modelo de teselas (Inarejos, 2014). *MapCache* almacena las celdas de las imágenes solicitadas para su posterior uso, evitando así una sobrecarga en el servidor y contribuyendo a mejorar la rapidez en la visualización de los mapas.

---

[3] Gráficos de Red Portátiles (del término en inglés PNG: Portable Network Graphics)

[4] Formato de Intercambio de Gráficos (del término en inglés GIF: Graphics Interchange Format)

[5] Grupo Conjunto de Expertos en Fotografía (del término en inglés JPEG: Joint Photographic Experts Group)

[6] Gráficos Vectoriales Redimensionables (del término en inglés SVG: Scalable Vector Graphics)

[7] Computer Graphics Metafile

[8] Teselas, celdas, baldosas, mosaicos, pequeñas imágenes (del término en inglés : tiles)

[9] Servicio de Mapas Teselados (del término en inglés WMTS: Tile Map Tiling Services)

La Línea de Productos de Software (LPS) Aplicativos_SIG del Centro Geoinformática y Señales Digitales (GEySED) desarrolla SIG a partir de la plataforma GeneSIG. Esta plataforma está desarrollada con herramientas y tecnologías de código abierto (Pantoja y Torres, 2012) y sirve como base para la creación de SIG para entornos web. En ella la visualización de los mapas se realiza utilizando el servidor de mapas *MapServer*, cuya capacidad de funcionar en diversos sistemas operativos lo hace uno de los más utilizados por los usuarios y se utiliza el servidor de teselas *MapCache.*

Actualmente el proceso de configuración de *MapCache* en la LPS es complejo pues se realiza a través de un fichero *XML*[10], de forma manual. Este fichero se encuentra en el servidor de aplicaciones, al cual no todos los profesionales de la línea tienen acceso, esto trae consigo que si se desea hacer algún cambio puntual en la configuración, se deba realizar a través de algún profesional que tenga acceso al servidor. Otro factor negativo es la poca experiencia del equipo de trabajo, pues existen pocos especialistas que dominan la configuración del fichero, y al realizarse de forma manual, esta les exige tener un amplio dominio de cada uno de sus atributos y los valores que puedan llegar a tomar.

Por otra parte, como no se permite visualizar el mapa que se está configurando, no se puede observar cómo influyen en él los cambios que se están aplicando y existe la posibilidad de introducir errores. Además el tiempo de configuración es elevado y sobrepasa los 30 minutos. Estos elementos provocan que no se exploten al máximo todas las ventajas que puede brindar *MapCach*e y por tanto influye negativamente en la visualización de los mapas en los aplicativos realizados por la LPS.

Debido a lo anteriormente mencionado, se deriva entonces el siguiente **problema a resolver**: ¿Cómo mejorar el proceso de configuración de *MapCache* para la plataforma GeneSIG?

Para darle solución al problema planteado se define como **objeto de estudio** la configuración de ficheros con formato *XML***,** delimitando el **campo de acción** hacia la configuración del fichero *XML* de *MapCache*. El **objetivo general** es desarrollar un módulo de configuración de *MapCache* para la plataforma GeneSIG.

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

Se plantea como **hipótesis de la investigación** que si se desarrolla un módulo de configuración de *MapCache* se mejorará el proceso de configuración de *MapCache* para la plataforma GeneSIG.

Para darle cumplimiento al objetivo general se proponen las siguientes **tareas de investigación**:

1. Caracterización del proceso de configuración de los ficheros *XML* para comprender cómo se realiza este proceso.
2. Análisis de la configuración de los servidores de teselas, sus limitaciones y fortalezas para identificar aspectos a tener en cuenta en el desarrollo del módulo.
3. Caracterización de las principales herramientas, tecnologías y metodologías a utilizar para la construcción de la propuesta de solución.
4. Modelado de los requisitos funcionales y no funcionales de la propuesta de solución para definir sus características.
5. Diseño de la solución propuesta para guiar la implementación.
6. Implementación de la solución propuesta para dar respuesta al objetivo de la investigación.
7. Validación de la propuesta de solución para comprobar su correcto funcionamiento.

A lo largo del desarrollo de la investigación se evidencia el uso de los siguientes métodos científicos:

Métodos teóricos:

- Analítico – Sintético (León y González, 2011): Se utiliza para realizar un estudio en profundidad de la bibliografía especializada, analizando cada componente que integra el objeto planteado por separado para luego sintetizarlo en la confección de la solución propuesta.

- Análisis Histórico – Lógico (León y González, 2011): Se realiza un estudio del arte relacionado con el objeto de la investigación para analizar las soluciones que existen para resolver el problema planteado y así conocer cómo se comporta actualmente la configuración de ficheros con formato *XML*.

- Método hipotético-deductivo (León y González, 2011): Para la formulación de la hipótesis y arribar a conclusiones.

Al concluir la investigación se esperan los siguientes resultados:

- Un módulo que permita la configuración de *MapCache* para la plataforma GeneSIG.
- La documentación ingenieril asociada al desarrollo del módulo.

El presente trabajo se encuentra estructurado en tres capítulos:

Capítulo 1: Fundamentación teórica sobre la configuración de *MapCache*. Se establecen conceptos asociados al dominio del problema, se analizan las soluciones existentes hasta el momento ante problemas similares. Además se precisan las tecnologías y herramientas a utilizar para dar solución al problema planteado, así como la metodología de desarrollo de software que se utilizará para la implementación del módulo.

Capítulo 2: Presentación del módulo de configuración de *MapCache* para la plataforma GeneSIG. Se realiza la descripción de la solución propuesta, para ello se efectúa el modelo de dominio, así como la especificación de los requisitos y los casos de uso.

Capítulo 3: Construcción del módulo de configuración de *MapCache* para la plataforma GeneSIG. Se realiza la construcción de la solución propuesta y se establecen los patrones arquitectónicos. Además se confeccionan el modelo de diseño, el modelo de despliegue, el modelo de implementación y se aplican las pruebas de software.

Capítulo 1. Fundamentación teórica sobre la configuración de *MapCache.*

En este capítulo se describen los conceptos fundamentales asociados al dominio del problema planteado, con el objetivo de facilitar la comprensión del alcance de la investigación. Se analizan las herramientas y soluciones existentes que puedan servir de base a la solución. Además se propone la metodología a emplear para guiar el proceso de desarrollo de software, así como las herramientas y tecnologías a utilizar para la elaboración del módulo que dará solución a la problemática planteada.

1.1. Sistemas de Información Geográfica

"*Un SIG es una herramienta que brinda a las labores de uso y manejo de información geográfica toda la potencia de un ordenador, pues ha sido diseñada específicamente para trabajar con este tipo particular de información*" (Olaya, 2010). Los SIG facilitan a los usuarios el trabajo con la información geográfica, permitiendo visualizarla en los mapas.

1.1.1. Componentes de los SIG

Víctor Olaya analiza los SIG desde dos enfoques distintos, uno como un sistema compuesto por subsistemas y otro de acuerdo a los elementos que lo compone (Olaya, 2010). Mientras que Álvaro de J. Carmona y Jhon Jairo Monsalve lo hacen desde un solo enfoque, de acuerdo a los elementos que lo componen (Carmona y Monsalve, 1999). A partir de estos dos criterios se asume el primero para analizar los componentes de un SIG.

Teniendo en cuenta el primer enfoque se definen cuatro subsistemas fundamentales: Subsistema de datos, de visualización y creación cartográfica, el de análisis y el de gestión. El primero es el encargado de la gestión de datos dentro del SIG, así como de las operaciones de entrada y salida de estos. Permite a los otros subsistemas tener acceso a los datos y realizar sus funciones en base a ellos. El segundo tiene como función la creación de representaciones a partir de los datos, permitiendo la interacción entre ellos y cuenta con las funcionalidades de edición. El tercer subsistema es el que contiene los métodos y procesos para realizar el análisis de los datos geográficos; mientras que el último es el encargado de gestionar la interacción de los restantes subsistemas (Olaya, 2010).

Otro punto de vista para analizar los SIG es de acuerdo a los elementos básicos que lo componen. Se definen cinco elementos fundamentales: Datos, Métodos, Personas, Software y Hardware. Los datos son la materia prima necesaria para el trabajo en un SIG y los que contienen la información geográfica vital para su existencia. Los métodos son el conjunto de metodologías y formulaciones a aplicar sobre los datos. El hardware es el equipo de cómputo necesario para ejecutar el software, aplicación que maneja los métodos y trabaja con los datos. Por último el motor de un SIG, las personas, quienes son las encargadas de diseñar y utilizar el software (Olaya, 2010).

Para reflejar correctamente la nueva realidad de los SIG se propone modificar este esquema clásico quedando de la siguiente manera:

- Datos.
- Procesos. Métodos enfocados al análisis de los datos.
- Visualización. Métodos y fundamentos relacionados con la representación de los datos.
- Tecnología. Software y hardware.
- Factor organizativo. Engloba los elementos relativos a la coordinación entre personas, datos y tecnología, o la comunicación entre ellos (Olaya, 2010).

Cada uno de estos componentes es de vital importancia en el desarrollo de un SIG. La interacción de unos con otros de manera organizada permite lograr su correcto funcionamiento. Debido al aporte que brinda a la investigación se analizará la visualización con mayor profundidad.

1.1.2. Visualización en los SIG

En un SIG la visualización juega un papel importante y por ello estos disponen de funcionalidades para representar la información geográfica. Trabajar en un SIG añade, entre otros elementos, el hecho de que la información se encuentra almacenada según un modelo dado (*raster* o vectorial). El usuario puede crear cartografía para otros pero, en la mayoría de los casos, la crea para sí mismo para poder emplearla como una herramienta más a la hora de desarrollar su trabajo. La representación visual que se produce en un SIG puede estar pensada para ser empleada en diversos ámbitos o puede tener una funcionalidad concreta dentro de un campo de aplicación dado.

Por último es necesario destacar que la representación visual realizada por un SIG se deriva a partir de datos numéricos y son estos datos los que se convierten en elementos gráficos, tales como líneas o

puntos. El beneficiario es el usuario del mapa, que recibe la representación visual de estos, y es esta visualización la que le transmite, en función de su calidad, la información geográfica. En relación con esto un SIG está diseñado para satisfacer dos necesidades fundamentales: crear la cartografía a partir de los datos y visualizar de la mejor manera posible los datos con los que se trabaja, para que esta visualización aporte valor añadido a los datos de cara al desarrollo de la labor de ese usuario (Olaya, 2010).

1.2. Modelo de teselas

Este modelo consiste en dividir en una pirámide de mayas el mapa cartográfico ofrecido por el servidor. En ella cada nivel corresponde con una resolución o escala del mapa y todas las teselas tienen un tamaño fijo en píxeles (Gámez, 2008), por ello es necesario utilizar este modelo para mejorar los servicios de mapas. Actualmente se pueden implantar proyectos que resuelven esta problemática, entre los que destacan los servidores de teselas. A continuación se describen algunas de las especificaciones que se han desarrollado para el servicio de teselas: *WMS-C*[11] *y WMTS* (García y Castro, 2010), esta última ya convertida en estándar.

1.2.1. WMS-C

Esta especificación tiene como principal objetivo encontrar una manera de optimizar la entrega de las imágenes de mapa a través de Internet. La propuesta ofrece idealmente algún medio por el cual los usuarios puedan acceder a las teselas de los servidores existentes, de tal manera que las imágenes se pueden almacenar en *cache*, en el servidor o en un lugar intermedio, o incluso ser completamente pre-generadas, si se desea (Pérez y Lallanilla, 2014).

1.2.2. WMTS

El *WMTS* de *OGC* proporciona un enfoque complementario al *WMS;* mientras que este último es una solución ideal para datos dinámicos centrándose en la renderización de mapas personalizados, el *WMTS* sirve datos pre-renderizados donde las escalas han sido restringidas a un conjunto discreto de teselas, que siguen una geometría de malla regular. El conjunto de teselas permite la ejecución de un servicio

[11] Servicio de Mapas en la Web- en cache (del término en inglés WMS-C: Web Map Service - Cached)

WMTS utilizando un servidor web, que devuelve los archivos existentes y el uso de mecanismos de red estándar de escalabilidad, tales como *cache* distribuidos. Usa un modelo de teselas parametrizados de tal manera que un usuario puede hacer peticiones de un conjunto discreto de valores y recibir rápidamente fragmentos de imágenes pre-renderizadas (teselas) del servidor, que generalmente no requieren de ninguna manipulación posterior para ser mostrados en pantalla (Masó y otros, 2010).

El estándar define tres operaciones básicas: *GetCapabilities*, *GetTile* y *GetFeatureInfo*, siendo esta última opcional. Estas operaciones permiten obtener los recursos básicos: el documento de metadatos del servicio, una tesela (de un determinado *tile matrix set* (conjunto de matrices de teselas*)* a una escala y sistema de referencia determinados) y el documento que tiene la información de un punto sobre una tesela (que contiene información de los objetos particulares mostrados en el mapa). Cada tesela de una matriz se identifica por el índice de columna (*TileCol*) y el índice de fila *(TileRow)*; estos índices tienen su origen 0,0 en la tesela izquierda y superior de la matriz y se incrementan hacia la derecha y hacia abajo respectivamente. En el *WMTS* se definen además tres maneras diferentes para obtener los recursos; una basada en la obtención directamente a partir de su *URL*[12]*.* Las otras dos basadas en la obtención a partir de operaciones, una siguiendo una sintaxis *KVP*[13] y otra siguiendo una sintaxis *SOAP*[14] (Masó y otros, 2010).

1.3. Servidores de teselas

Los servidores de teselas ofrecen la cartografía renderizada únicamente en juegos de teselas con un número limitado de resoluciones, casi siempre además empleando sistemas de almacenamiento intermedio (*cache*) (Díaz y otros, 2014). Estos sistemas realizan el proceso de duplicar un conjunto de datos con el objetivo de reducir el tiempo de acceso a la información original. A continuación se analiza el servidor de teselas *MapCache* por su importancia para la investigación y además *GeoWebCache* que al igual que el primero se configura a través de un fichero con formato *XML.*

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

[13] Par clave-valor (del término en inglés KVP: Key-Value Pairs)

[14] Protocolo de Acceso Simple a Objeto (del término en ingles SOAP: Simple Object Access Protocol)

1.3.1. MapCache

A partir de la versión 6.0 *Mapserver* también incluye potentes funciones de teselas a través del proyecto *MapCache* (Sphinx, 2014b). *MapCache* nació como proyecto independiente en el año 2010, está implementado en C/C++ y actúa como *proxy* interceptando las peticiones que se realizan al servidor de mapas y es un servidor que implementa un modelo de teselas. En la Figura 1 se puede observar cómo se crean las teselas, los niveles hacen referencia al nivel de *zoom*.

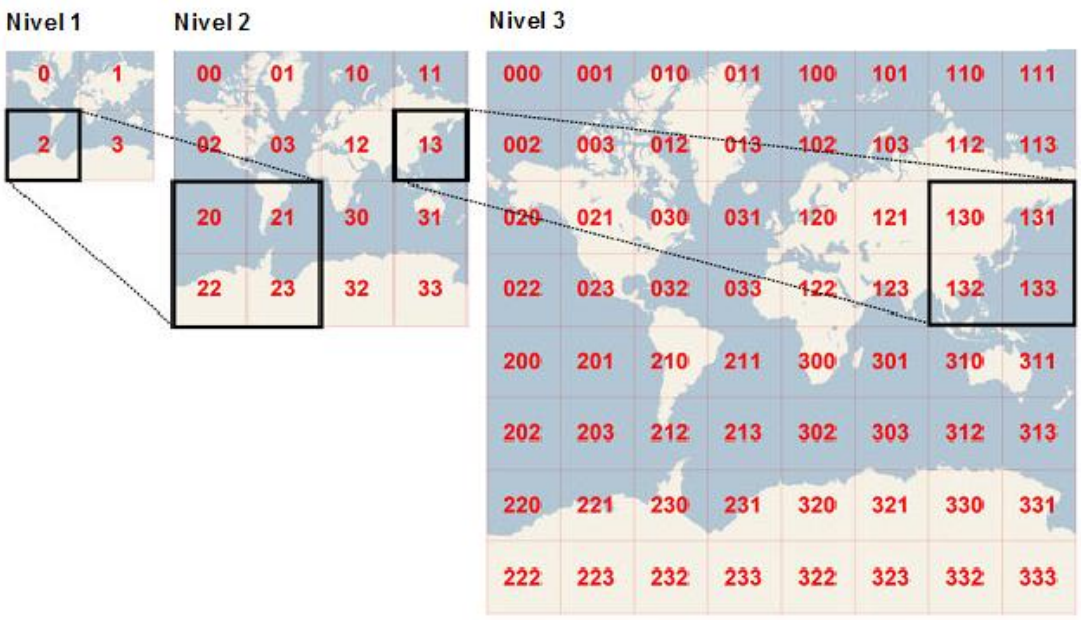

Figura 1. Representación de la creación de teselas.

La primera vez que se realiza una petición de una tesela, para un *zoom* concreto, el servidor comprueba si ya dispone de esa tesela para ese zoom en su sistema de *cache*, si es así la recupera y la devuelve al usuario. En caso de que no disponga de ella, realiza una petición al servidor de mapas para la extensión y el *zoom* de la tesela que ha sido solicitada (petición mediante *WMS*). De esta manera si un usuario vuelve a solicitar esa tesela para el mismo *zoom*, esta ya estará generada (Inarejos, 2014).

Otra característica importante de *MapCache* es la posibilidad de generar las teselas independientemente de las peticiones del usuario, esto se denomina *seeds* (semillas). La herramienta *seeds,* permite hacer peticiones por línea de comando que pregeneran teselas para capas determinadas, que se almacenan en

cache (Inarejos, 2014). La configuración de *MapCache* se realiza a través de un fichero con formato *XML*, mapcache.xml, en el que se determina cómo *MapCache* atenderá las peticiones (Sphinx, 2014a). A continuación se muestran las etiquetas de este fichero:

Matriz (*Grid*):
Es una matriz de celdas del mapa en un área determinada y contiene una extensión geográfica, resoluciones y tamaños de celdas.

Fuente de datos (*Source*):
Especifica el servicio donde se consultarán las teselas. Típicamente es un servidor *WMS* accesible a través de una *url*. Actualmente no hay otras fuentes de datos implementadas.

Cache:
Es el elemento que especifica dónde y cómo se ubicarán las teselas que guarda *MapCache*, es posible configurar diferentes ubicaciones.

Formato (*Format*):
Es un formato de imagen que se utilizará para devolver los datos a los usuarios y almacenarlos en la *cache*.

Conjunto de celdas (*Tileset*):
Es el elemento principal en *MapCache* donde se configuran las teselas que podrán ser consultadas. Corresponde con un conjunto de teselas que provienen de una fuente de datos, son almacenados en una *cache* y se devuelven en un formato de imagen.

Servicios (Services):
Los servicios son tipos de peticiones a las que *MapCache* va a responder. Se debe habilitar al menos uno.

1.3.2. GeoWebCache

GeoWebCache es un proyecto de código abierto implementado en *Java*, que optimiza el renderizado del servicio *WMS*. Actúa como un *proxy* entre el cliente y el servidor, guardando los mapas para cuando sean solicitados. En la medida en que nuevos mapas son solicitados, intercepta estas llamadas y envía las teselas pre-renderizadas si están almacenadas, o llama al servidor para que las renderice si es necesario.

De esta forma, una vez que un mapa está guardado, la velocidad de renderizado se incrementa varias veces, mejorando la experiencia del usuario (García y otros, 2012). Además implementa *WMS-C* (García y Castro, 2010) y plenamente *WMTS* utilizando *KVP* (OpenGeo, 2014).

GeoWebCache surge con el objetivo de cubrir la necesidad en *GeoServer* de un servidor de teselas que permitiera pregenerar y acelerar la cartografía servida por este producto. Con el tiempo ha crecido en funcionalidad y se puede considerar un producto independiente, aunque en general se utilice conjuntamente con *GeoServer* (Díaz y otros, 2014). Su configuración se realiza a través de un fichero *XML*, pero tiene la ventaja que posibilita realizar esta configuración por medio de una interfaz web (Open Source Geospatial Foundation, 2014). A continuación se muestran algunos de los parámetros que se configuran (GeoWebCache, 2014). (Ver Figura 2).

- Tipo de operación (*Type of operation):* Generalmente se selecciona “Seed”, o sea, pre-generación.
- Conjunto de matriz *(Grid Set)*: En este punto se selecciona el sistema de referencia del mapa.
- Formato *(Fomat)*: Formato de imagen que se utiliza en las llamadas del mapa al servidor.
- Comenzar agrandado y parar agrandado (*Zoom start y Zoom stop*): Niveles de *zoom* para los que se generan las teselas.
- *Estilos (STYLES)*: El estilo con el que se generan las teselas.
- Cuadro delimitador *(Bounding box)*: Extensión de los datos que define las teselas que se generan. Por defecto se toma la extensión de la capa (GeoWebCache, 2014).

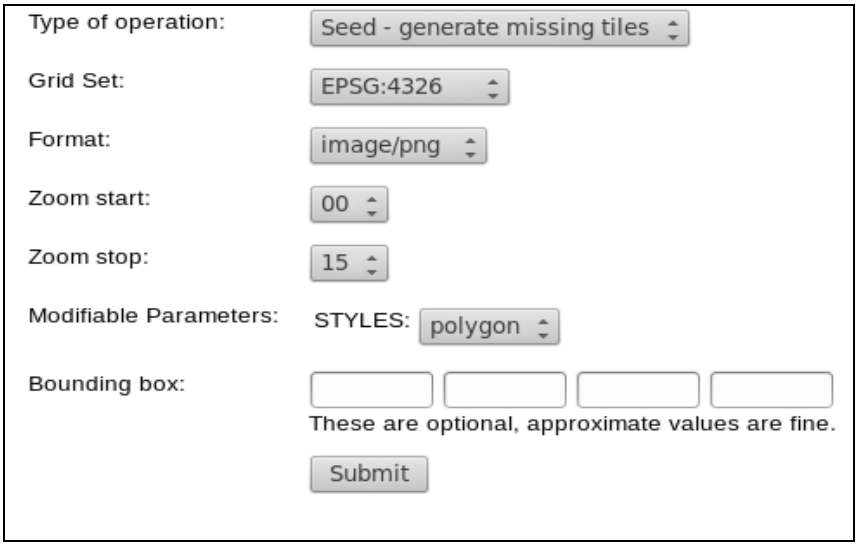

Figura 2. Formulario de pre-generación de teselas de GeoWebCache.

1.3.3. TileCache

TileCache es una implementación de un *WMS-C,* disponible bajo la licencia *BSD*[15] por *MetaCarta*. Está desarrollado *en Python* y funciona como *proxy* entre el cliente y el servidor *WMS.* En el caso de uso más simple, solo requiere acceso de escritura al disco, la capacidad de ejecutar *scripts CGI*[16] de *Python* y un *WMS* que se desea almacenar en *cache*. Con estos recursos, se puede crear una *cache* local de cualquier servidor *WMS* y utilizar el resultado en cualquier *WMS-C* de lado del cliente, como *OpenLayers* (Contributors, 2010).

TileCache tiene dos modalidades de funcionamiento. La primera hace la tesela usando *MapServer* o *Mapnik* como una copia de la representación final. La segunda puede almacenar en *cache*, teselas *WMS* descargadas desde un servicio *WMS* remoto. Soporta dos mecanismos de *cache* diferentes, el primero*, DiskCache*, almacena en memoria las teselas como archivos de imagen. El segundo mecanismo, *MemoryCache,* almacena datos en una instancia *memcache* o *cluster*. *TileCache* cuenta con una herramienta basada en web que permite hacer su configuración a través de un navegador (Contributors, 2010); pero el proyecto lleva ya bastante tiempo estancado sin prácticamente actividad por lo que no sería aconsejable instalarlo en nuevos proyectos (Díaz y otros, 2014).

A partir del estudio realizado sobre estos tres servidores de teselas y considerando que no es recomendable instalar *TileCache* en proyectos nuevos; se estableció una comparación entre *GeoWebCache* y *MapCache* teniendo en cuenta una serie de aspectos: lenguaje de programación en el que están desarrollados, cómo realizan su configuración, si se puede realizar vía web, y si están integrados con *Mapserver* 6.0 que es el servidor de mapas que utiliza la plataforma GeneSIG (ver Tabla 1).

[15] Berkeley Software Distribution.

[16] Interfaz de entrada común (del término en inglés CGI: Common Gateway Interface).

Tabla 1. Comparación entre los servidores de teselas GeoWebCache y TileCache.

Servidores de teselas	***GeoWebCache***	***MapCache***
Lenguaje de Programación	*Java*	C/C++
Configuración	fichero con formato *XML*	fichero con formato *XML*
Configuración vía web	Sí	No
Integración con *MapServer 6.0*	No	Sí

Una vez realizada la comparación entre estos dos servidores de teselas se llega a la conclusión de utilizar *GeoWebCache* como guía para la creación de las interfaces visuales del módulo, pues este servidor de teselas brinda una aplicación que permite configurar el fichero *XML* vía web, facilitando el trabajo con el mismo.

1.4. Configuración de ficheros XML

XML es un lenguaje de marcación jerárquico promovido por *W3C*[17] en un formato de texto simple y muy flexible, derivado de *SGML*[18], que originalmente fue diseñado para cumplir con los retos de la publicación electrónica a gran escala (W3C, 2013). Ofrece la posibilidad de representar de forma neutra la información, independientemente del sistema operativo y el lenguaje de programación que se utilice (Tendero y Simarro, 2013). *XML* permite la internacionalización, es decir puede trabajar con cualquier conjunto de caracteres, facilitando la comprensión de estos (Jeudi, 2008).

El principal concepto de *XML* es el de envolver el texto con elementos de anotación, llamados etiquetas, que califican el texto; la etiqueta se compone de dos partes la de apertura y la de cierre, además estos

[17] World Wide Web Consortium

[18] Lenguaje de Marcado Generalizado Estándar (del término en inglés SGML: Standard Generalized Markup Language)

elementos pueden tener atributos en parejas de nombre y valor (Cordero, 2014). Estas etiquetas contienen el nombre del nodo entre los símbolos de mayor y menor (<nombre de la etiqueta>) y la de cierre lleva una barra antepuesta al nombre (</nombre de la etiqueta>) (García, 2006). Además *XML* permite que cada comunidad pueda construir su propio vocabulario de etiquetas y las reglas para relacionarlas en dependencia del problema que se desee resolver (Palmirani y Vitali, 2012).

XML es un lenguaje de marcación abierto y gracias a esto constituye un óptimo formato de datos para almacenar información a largo plazo. Simplifica los cambios y hace más sencillo el transporte de los datos (Cordero, 2014). Permite compartir información entre sistemas o fuentes de datos heterogéneas, por ejemplo, páginas web, distintas bases de datos (Jeudi, 2008). Posee una estructura jerárquica que permite definir bien las estructuras de los documentos (W3C, 2013). Además lo caracteriza su rigor, ya que utiliza una sintaxis y define sus reglas a través de una gramática; de este modo define una regla de comportamiento de una etiqueta y este comportamiento no puede ser violado por el usuario (Palmirani y Vitali, 2012).

Los elementos *XML* presentan la importante peculiaridad de ser extensibles, pueden extenderse para llevar más información y tienen la gran ventaja de que pueden aportar más información sin echar a perder las aplicaciones (Cordero, 2014). Todas estas características que posee *XML* hacen que sea de gran utilidad actualmente y muchas aplicaciones se han diseñado entorno a las posibilidades que ofrece este lenguaje. Los ficheros en formato *XML* pueden ser utilizados como base de datos (Bourret, 2010) o fichero de configuración (Mestras, 2013).

1.4.1. Herramientas específicas para el trabajo con ficheros XML

El uso intensivo de los ficheros *XML* en el desarrollo de aplicaciones hace necesaria la utilización de herramientas específicas (bibliotecas) para acceder y manipular este tipo de archivos de manera potente, flexible y eficiente. Estas herramientas reducen los tiempos de desarrollo de aplicaciones y permiten manejar los ficheros *XML* de forma simple y sin cargar innecesariamente el sistema. Existen dos clasificaciones para este tipo de herramientas, las que validan el fichero y las que no. Las primeras comprueban que el documento *XML* esté bien formado y sea válido con respecto a un esquema definido,

entre las que se encuentra *JAXB*[19]; mientras que las segundas sólo verifican si el documento está bien formado según la definición *XML* y no en relación a un esquema, ejemplo de estas son *SAX*[20] y *DOM*[21] (Tendero y Simarro, 2013).

JAXB es una biblioteca de *Java*, que convierte el contenido de un documento *XML* en una estructura de clases *Java;* este documento debe tener un esquema *XML* asociado y por tanto el contenido de este debe ser válido con respecto a ese esquema (Sun Microsystems, 2001). Dado un esquema, que especifica la estructura de los datos *XML*, el compilador *JAXB* genera un conjunto de clases de Java que contienen todo el código para analizar los documentos *XML* basados en el esquema. Una aplicación *JAXB* es específica de un esquema; no se puede utilizar para procesar los documentos *XML* que se basan en otro esquema. *JAXB* proporciona una manera rápida y conveniente de crear uniones de dos vías entre los documentos *XML* y los objetos *Java* (Ort y Mehta, 2003).

SAX es otra tecnología para poder acceder a *XML* desde lenguajes de programación. Funciona por eventos y métodos asociados, y aunque *SAX* tiene el mismo objetivo que *DOM* esta aborda el problema desde una óptica diferente, y sólo permite leer del fichero (Chengkai Li, 2009). Por lo general se usa SAX cuando la información almacenada en los documentos *XML* es clara, está bien estructurada y no se necesita hacer modificaciones (Tendero y Simarro, 2013). Originalmente fue una *API*[22] únicamente para el lenguaje de programación *Java*, pero existen versiones de *SAX* para otros lenguajes de programación. SAX no almacena completamente el documento en memoria, lo que hace que sea más rápido y que utilice menos memoria (Brownell, 2008).

DOM es una interfaz de programación que permite analizar y manipular dinámicamente y de manera global el contenido, el estilo y la estructura de un documento. Tiene su origen en el *W3C* (W3C, 2005). Para trabajar con un documento *XML* primero se almacena en memoria en forma de árbol con nodos que

[19] Java Architecture for XML Binding.

[20] Simple API for XML.

[21] Documento de Modelado de Objetos (del término en inglés DOM: Document Object Model).

[22] Interfaz de programación de aplicaciones (del término en inglés: Application Programming Interface).

representan la estructura descrita por dicho documento. Una vez creada en memoria esta estructura, los métodos del *DOM* permiten recorrer los nodos del árbol y analizar a qué tipo particular pertenece. En función del tipo de nodo, se ofrece una serie de funcionalidades para poder trabajar con la información que contienen (Tendero y Simarro, 2013). DOM debe ser utilizado en aplicaciones donde se acceda al documento repetidamente pues permite navegación en cualquier dirección y modificaciones arbitrarias (García, 2011). A continuación se muestra una comparación sobre las 3 herramientas mencionadas anteriormente (ver Tabla 2).

Tabla 2. Comparación entre las bibliotecas para el trabajo con ficheros XML.

Herramientas	***JAXB***	***SAX***	***DOM***
Tipo de herramientas	Valida el fichero	No valida el fichero	No valida el fichero
Leer del fichero	Sí	Sí	Sí
Analizar y manipular el contenido y estructura del documento.	No	No	Sí

Una vez realizada la comparación entre *JAXB*, *SAX* y *DOM* y teniendo en cuenta que solo se necesita verificar si el archivo mapcache.xml está bien formado, se utilizó una herramienta que no valida el fichero. De estas herramientas se descarta la posibilidad de utilizar *SAX*, pues tiene la desventaja de que solo brinda la opción de leer los datos, y como se necesita configurar el fichero se empleó *DOM* que sí permite analizar y manipular dinámicamente de manera global el contenido y estructura del documento.

DOM posee *APIs* en varios lenguajes de programación, destacándose *JDOM*[23] para *Java* (Tendero y Simarro, 2013), fue la *API* que facilitó el trabajo con el fichero de configuración de *MapCache*. *JDOM* surge en el año 2000 para el manejo optimizado de datos *XML*. Es una *API* específica para *Java* (no está pensada para otros lenguajes de programación) permitiendo crear, analizar y manipular cómodamente documentos *XML*, representándolos como un árbol completo, que consta de elementos, atributos, nodos y

[23] Documento de Modelado de Objetos en Java (del término en inglés JDOM: Java Document Object Model)

secciones entre otros, y está disponible todo el tiempo. *JDOM* puede acceder y modificar en cualquier momento la parte que se desee de este árbol y los diferentes tipos de nodos son representados por clases concretas (Dorantes y otros, 2006).

1.5. Herramientas y tecnologías a utilizar

La correcta selección de las herramientas y tecnologías a utilizar en el desarrollo de un software garantiza que se logre un producto de alta calidad. Teniendo en cuenta que el módulo de configuración de *MapCache* está integrado a GeneSIG en su versión 1.5, este se acoge a las tecnologías y herramientas utilizadas en la creación de la plataforma.

1.5.1. Metodología de desarrollo de software

Las metodologías de desarrollo de software sirven de guía en todo el ciclo de desarrollo de un software, de ahí la importancia que tiene definir correctamente cuál es la más adecuada utilizar. En el desarrollo de la solución propuesta se utiliza la metodología *AUP*[24] conocida también *como RUP*[25] ágil, empleada actualmente en la LPS Aplicativos_SIG *y* sobre la que se tiene conocimiento y práctica. *AUP* es flexible y propone los mismos roles y artefactos que *RUP*, a diferencia de que no hay necesidad de generar toda la documentación que se requiere en cada flujo de trabajo.

Las metodologías ágiles tienen como prioridad la completa satisfacción del usuario mediante tempranas y continuas entregas de software que le aporte un valor. Estas proponen mantener una constante comunicación entre el usuario y el equipo de desarrollo de software. Además se hace uso de un mínimo de trabajo de ingeniería de software (Pressman, 2010).

AUP es una versión simplificada de *RUP* que describe una aproximación al desarrollo de aplicaciones, combinando conceptos propios de *RUP* con técnicas ágiles, con el objetivo de mejorar la productividad. Al igual que en *RUP* en *AUP* se establecen cuatro fases que transcurren de manera consecutiva y que

[24] Proceso Unificado Ágil (del término en inglés AUP: Agile Unified Process)

[25] Proceso Unificado Racional (del término en inglés RUP: Rational Unified Process)

acaban con hitos claros alcanzables. El proceso *AUP* establece un modelo más simple que el que aparece en *RUP,* por lo que reúne en una misma disciplina las disciplinas de Modelado de Negocio, Requisitos, Análisis y Diseño. El resto de las disciplinas (Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno), se mantienen igual a las restantes de *RUP* (AUP, 2015).

1.5.2. Lenguaje de modelado UML 2.0

Un lenguaje de modelado es un conjunto de normas estandarizadas que permiten la construcción de modelos, los cuales constituyen una ayuda para comprender el sistema. Para el desarrollo de la solución se escogió *UML*[26], sus objetivos son muchos pero se pueden sintetizar sus funciones:

Visualizar: Permite expresar de forma gráfica un sistema que pueda ser entendido fácilmente.

Especificar: Permite especificar cuáles son las características del sistema antes de su construcción.

Documentar: Los propios elementos gráficos sirven como documentación del sistema desarrollado que pueden servir para su futura revisión (Hernández, 2012).

1.5.3. Herramienta Case Visual Paradigm for UML 8.0

Visual Paradigm for UML 8.0 es una herramienta *CASE*[27] que soporta el modelado mediante *UML* y controla que este sea correcto. Durante todo el ciclo de vida del software brinda asistencia a los analistas y desarrolladores (González y Torres, 2012). Es fácil de instalar, tiene la característica de ser multiplataforma, además en la universidad se cuenta con la licencia para su uso. *Visual Paradigm for UML* 8.0 también proporciona abundantes tutoriales, demostraciones interactivas y proyectos de *UML*. Esta herramienta permitió la creación de todos los diagramas que guiaron el desarrollo del módulo, garantizando así la calidad de mismo.

[26] Lenguaje Unificado de Modelado (del término en inglés UML: Unified Modeling Language)

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

1.5.4. Lenguaje de programación PHP 5

PHP[28] en su versión 5 es un lenguaje de programación de uso general de código abierto del lado del servidor. Puede ser compilado y ejecutado en varias plataformas utilizando el mismo código fuente. Dispone de una amplia documentación que facilita al usuario su entendimiento, siendo fácil de utilizar (Vázquez, 2008); cuenta con una biblioteca, *PHP Mapscript*, que permite a *PHP* trabajar con *MapServer*. Por las características antes descritas, *PHP* 5 es el lenguaje de programación escogido para utilizar en el desarrollo del módulo. Además en la decisión también se tuvo en cuenta que dicho módulo será integrado a la plataforma GeneSIG, la cual emplea *PHP* en el desarrollo de sus componentes.

1.5.5. Lenguaje de programación Java 7.0

Java es un lenguaje de programación orientado a objetos de propósito general, cuya principal característica es que es compilado e interpretado (Belmonte, 2005). Es independiente de la plataforma de desarrollo, es decir, cualquier programa creado a través de *Java* podrá funcionar correctamente en ordenadores de todo tipo y con sistemas operativos distintos. Actualmente es uno de los lenguajes de programación más populares (Adtriboo, 2014). Es completamente libre y gratuito por lo que no es necesario comprar licencias de ningún tipo y tiene una sintaxis amigable. Además cuenta con la *API JDOM* que facilita el trabajo con ficheros de formato *XML,* en este caso se utilizó *JDOM* 2.0.4*.* Por estas características es el lenguaje de programación utilizado para la creación de servicios en el desarrollo de la aplicación.

1.5.6. ExtJS 3.0

Es una biblioteca de *JavaScript* para el desarrollo de aplicaciones enriquecidas para la web o *RIA*[29]*,* haciendo uso extensivo de *AJAX*[30]. Fue creado en sus inicios como una extensión de *YUI*[31] otro *framework*

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

[29] Aplicaciones de internet enriquecidas (del término en inglés RIA: Rich Internet Applications)

[30] JavaScript asíncrono y XML (acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML)

[31] Yahoo User Interface

bastante similar. Su potencia reside en la colección de componentes para el diseño de interfaces. Cuenta con una comunidad y las versiones que lanza al mercado son mejores que las anteriores. Debido a sus políticas de código abierto es aceptado por numerosos usuarios (Terry, 2012). Su uso en el módulo provee una interfaz rica en componentes y fácil de usar. Además permite capturar y administrar los eventos externos realizados por el usuario.

1.5.7. NetBeans 8.0

Un *IDE*[32] es una aplicación que proporciona servicios integrales para facilitar la creación de software. En el desarrollo del módulo se utiliza el *IDE NetBeans* 8.0, que permite a los programadores escribir, compilar, depurar y ejecutar programas. Es además un producto libre y sin restricciones de uso que brinda la posibilidad de desarrollar aplicaciones web de forma rápida y fácil (Oracle, 2015). Otra de las ventajas de usarlo es que a pesar de estar escrito en *Java* soporta varios lenguajes como *PHP y JavaScript* que son los lenguajes utilizados en el desarrollo de la propuesta de solución.

1.5.8. Servidor Web Apache 2.2

Es un proyecto de código abierto, de uso gratuito y multiplataforma que destaca por su seguridad y rendimiento (The Apache Software Foundation, 2015). Continuamente adaptado a los nuevos estándares *HTTP*,[33] *Apache* 2.2 es un servidor web muy utilizado, flexible, rápido y eficiente. Tiene la infraestructura necesaria para servir distintos protocolos (The Apache Software Foundation, 2013). Permite personalizar la respuesta ante errores que se puedan dar en el servidor, siendo posible configurarlo para que ejecute un determinado *script* cuando ocurra un error en concreto.

1.5.9. Servidor de Mapas MapServer 6.0

MapServer es una plataforma de código abierto para la publicación de datos espaciales y aplicaciones cartográficas interactivas para la web. Es multiplataforma y tiene soporte a un gran número de estándares de la *OGC* entre ellos *WMS*. Posibilita la automatización de los elementos del mapa (barra de escala,

[32] Entorno de desarrollo integrado (del término en inglés IDE: Integrated development environment)

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

mapa de referencia y leyenda), soporta múltiples formatos de datos *raster* y vectorial (MapServer, 2014). Su modo de funcionamiento está basado en la generación de lado del servidor web de imágenes estáticas (*JPEG, GIF, PNG*, etc.) como resultado del proceso de las peticiones realizadas por los clientes (Díaz, 2005).

1.6. Conclusiones parciales

En los SIG la visualización es un elemento fundamental porque facilita la interpretación de la información geográfica. El servicio *WMS* permite mejorar la visualización de los mapas para un mejor entendimiento por los usuarios, facilitando el procesamiento de la información geográfica en los SIG, pero a pesar de sus ventajas no permite responder adecuadamente a un número elevado de peticiones simultáneas. Para dar solución a este problema, se pueden implantar servidores de teselas que aplican el modelo de teselas y además generalmente un sistema de *cache*, permitiendo acceder rápidamente a la información solicitada.

Teniendo en cuenta las necesidades actuales de la LPS se decidió utilizar *GeoWebCache* como guía para la creación de las interfaces del módulo y la *API JDOM* para facilitar el trabajo con el fichero *XML de* configuración de *MapCache*; para ello se seleccionaron las herramientas y tecnologías necesarias para la implementación. El estudio realizado en este capítulo provee los conocimientos necesarios para pasar a presentar la propuesta de solución.

Capítulo 2. Presentación del módulo de configuración de *MapCache*

Antes de pasar a implementar el módulo es necesario saber claramente qué se va a hacer y cómo. En este capítulo, con el objetivo de describir las principales características que tiene el módulo se presentan los artefactos relacionados con la modelación del dominio y la captura de los requisitos. Se describen los requisitos funcionales y no funcionales con los que deberá cumplir el módulo, el actor del sistema y los casos de uso.

2.1. Descripción de la propuesta de solución

La propuesta de solución tiene como objetivo realizar la configuración del servicio de *MapCache* a través de la plataforma GeneSIG. Para ello es necesario realizar un mecanismo de comunicación entre una aplicación web y un servicio del sistema operativo. La mejor forma para lograr lo anterior es realizarlo a través de servicios web para garantizar la interoperabilidad con otras aplicaciones. La configuración de *MapCache* se realiza mediante un fichero *XML*, para este tipo de formato existen herramientas específicas para leer y escribir en un fichero, según se describió en el Capítulo 1.

La solución se divide en varios pasos:

1. Leer el fichero mapchache.xml y cargar en memoria esa configuración.
2. Se exponen mediante servicios web los métodos que permiten la modificación del fichero de configuración.
3. Estos servicios se consumen desde el módulo de GeneSIG de configuración de *MapCache*, el cual debe permitir entre otras funcionalidades, agregar un conjunto de celdas como capa a la plataforma.
4. Al realizar algún cambio desde la plataforma mediante los servicios web se escriben los mismos en el fichero de configuración.

2.2. Modelo del dominio

Teniendo en cuenta que no se tienen bien definidos los procesos del negocio se realiza una modelación del dominio, (ver Figura 3) y se explican cada uno de los conceptos que forman parte de este. Un modelo de dominio permite representar las clases conceptuales que sean significativas en el dominio del problema. Utilizando la notación *UML*, representa los objetos del dominio o clases conceptuales, las asociaciones entre ellas y sus atributos (Larman, 2003).

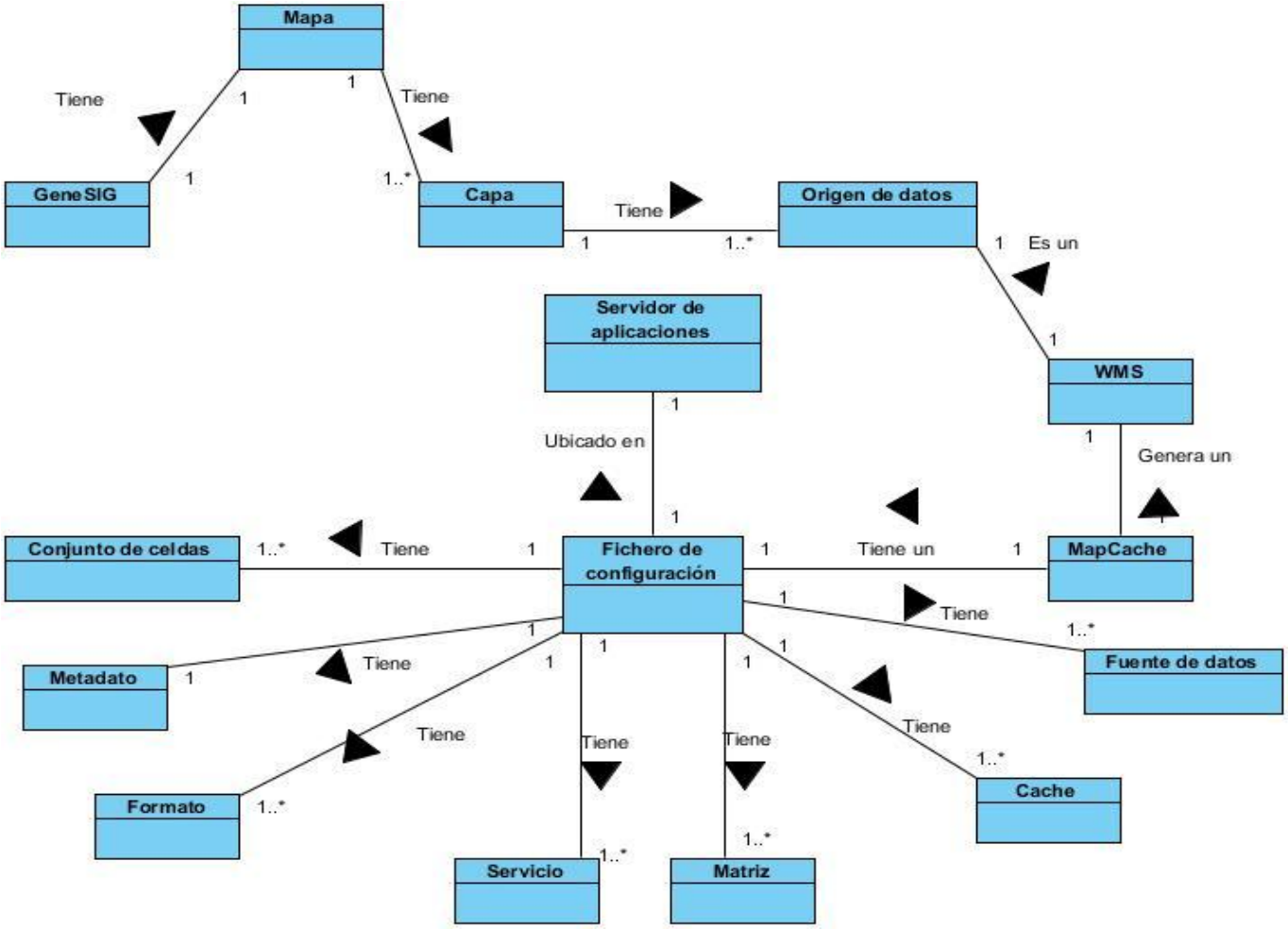

Figura 3. Diagrama de clases del dominio.

2.2.1. Descripción de las clases del modelo del dominio

***Cache*:** Localización donde se almacenarán las teselas recibidas.

Capa: Es una colección de geometrías homogéneas, es decir, tienen el mismo conjunto de atributos. Constituye la unidad básica de un mapa.

Conjunto de celdas: Es el elemento principal en la configuración de *MapCache* y corresponde con un conjunto de celdas que provienen de una fuente de datos, son almacenados en una *cache* y se devuelven en un formato de imagen.

Fichero de configuración: Fichero con formato *XML* en el que se configuran todos los parámetros de *MapCache*.

Formato: Es el formato de imagen que se utilizará para devolver los datos a los usuarios y almacenarlos en la *cache*.

Fuente de datos: Es un servicio que *MapCache* puede consultar para obtener los datos. Típicamente es un servidor *WMS* accesible a través de una *url*.

GeneSIG: Plataforma para la creación de SIG para entornos web.

Mapa: Es una representación geográfica de una porción de la superficie terrestre.

MapCache: Servidor de teselas que utiliza *MapServer*.

Matriz: Es una matriz de celdas del mapa en un área determinada y contiene una extensión geográfica, resoluciones y tamaños de celdas.

Metadatos: Información del servicio de *MapCache*.

Origen de datos: De donde las capas obtienen el conjunto de geometrías.

Servicios: Elemento donde se configura el tipo de servicio que utilizará *MapCache.*

Servidor de aplicaciones: Servidor con los recursos y configuraciones necesarias para ejecutar los

servicios de *MapCache*.

WMS: Servicio de mapas en la web que facilita el procesamiento de la información geográfica en los SIG.

2.2.2. Breve descripción del modelo del dominio

La plataforma GeneSIG tiene un mapa, un mapa está compuesto por varias capas que tienen varios orígenes de datos y un origen de datos es un *WMS*. *MapCache* genera un *WMS* y tiene un fichero de configuración que se encuentra ubicado en el servidor de aplicaciones. Este fichero de configuración tiene varias entradas: conjunto de celdas, matriz de celdas, *cache*, formato de imagen, metadatos y servicios.

2.3. Modelo del sistema

Luego de haber explicado las clases que conforman el dominio del sistema y establecer una relación entre ellas; con el objetivo de guiar el desarrollo hacia una solución que satisfaga las necesidades del cliente, se definieron los requisitos funcionales y no funcionales que debe cumplir el sistema.

2.3.1. Técnicas de captura de requisitos

Los requisitos de software son “la descripción de los servicios proporcionados por el sistema y sus restricciones operativas”. Se pueden clasificar en funcionales y no funcionales (Sommerville, 2005) y se identifican a partir de las necesidades que tiene el usuario. Por la importancia que tiene la captura de estos, existen varias técnicas que facilitan la comunicación con los clientes del producto, con el fin de lograr su satisfacción. Algunas de estas técnicas son: entrevistas, cuestionarios, tormenta de ideas, observación y estudio de documentaciones. En este caso, para obtener los requisitos que debe cumplir el sistema, se utilizó la tormenta de ideas.

La tormenta de ideas consiste en reuniones en grupo que no debe ser muy numeroso (máximo 10 personas), con el objetivo de acumular ideas e información sin evaluar las mismas. Como técnica de requisitos es sencilla de usar y de aplicar, ayudando a obtener una visión general de las necesidades del sistema (Escalona y Koch, 2002). La tormenta de ideas permitió el intercambio con el grupo de profesionales de la LPS que tienen acceso a configurar el fichero de *MapCache.* Este debate permitió definir las funcionalidades con las que cuenta el sistema, con el objetivo de facilitarles a estos la configuración de *MapCache*.

2.3.2. Requisitos funcionales (RF)

"Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y cómo se debe comportar en situaciones particulares" (Sommerville, 2005). Estos conforman el conjunto de funcionalidades con las que debe contar el sistema por lo que constituyen su base. En el desarrollo del módulo se identificaron 24 requisitos funcionales que se muestran a continuación, para mayor información consultar el artefacto *"Módulo Configuración MapCache Especificación de requisitos de software"* ubicado en el expediente de proyecto.

RF 1 Visualizar los conjuntos de celdas existentes en el fichero de configuración de *MapCache*.

RF 2 Adicionar un conjunto de celdas en el fichero de configuración de *MapCache*.

RF 3 Modificar un conjunto de celdas existente en el fichero de configuración de *MapCache*.

RF 4 Eliminar un conjunto de celdas existente en el fichero de configuración de *MapCache*.

RF 5 Visualizar las fuentes de datos existentes en el fichero de configuración de *MapCache*.

RF 6 Adicionar una fuente de datos en el fichero de configuración de *MapCache*.

RF 7 Modificar una fuente de datos existente en el fichero de configuración de *MapCache*.

RF 8 Eliminar una fuente de datos existente en el fichero de configuración de *MapCache*.

RF 9 Visualizar las *cache* existentes en el fichero de configuración de *MapCache*.

RF 10 Adicionar una *cache* en el fichero de configuración de *MapCache*.

RF 11 Modificar una *cache* existente en el fichero de configuración de *MapCache*.

RF 12 Eliminar una *cache* existente en el fichero de configuración de *MapCache.*

RF 13 Visualizar los formatos existentes en el fichero de configuración de *MapCache*.

RF 14 Adicionar un formato en el fichero de configuración de *MapCache.*

RF 15 Modificar un formato existente en el fichero de configuración de *MapCache*.

RF 16 Eliminar un formato existente en el fichero de configuración de *MapCache.*

RF 17 Visualizar las matrices de celdas existentes en el fichero de configuración de *MapCache*.

RF 18 Adicionar una matriz de celdas en el fichero de configuración de *MapCache*.

RF 19 Modificar una matriz de celdas existente en el fichero de configuración de *MapCache*.

RF 20 Eliminar una matriz de celdas existente en el fichero de configuración de *MapCache*.

RF 21 Modificar metadatos del fichero de configuración de *MapCache.*

RF 22 Adicionar un conjunto de celdas como capa a GeneSIG.

RF 23 Recargar configuración de *MapCache.*

RF 24 Modificar estado de los servicios.

2.3.3. Requisitos no funcionales (RFN)

Además de los requisitos funcionales se identificaron los requisitos no funcionales. *"Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema"* (Sommerville, 2005). A continuación se presentan:

RNF 1 Tipo de usuario final

- Debido a las acciones que se realizarán en la aplicación, deberá ser manejada por usuarios con conocimientos básicos de *MapCache.*

RNF 2 Usabilidad

- La aplicación debe tener una estructura que permita trabajar con rapidez y accesibilidad, mostrando una interfaz amigable e intuitiva para el usuario. Debe ser una aplicación de fácil aprendizaje por el usuario, donde los mensajes y campos de entrada utilizarán nombres intuitivos para el mismo.

RNF 3 Interfaz de usuario

El sistema debe:

- Tener un diseño sencillo donde no sea necesario mucho entrenamiento para utilizar el sistema.
- Ser intuitivo.

RNF 4 Requisitos de Hardware

Para las PCs clientes:

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

Para los servidores:

- Se requiere tengan tarjeta de red.
- Procesador de 3 GHz como mínimo.
- El servidor de mapas tiene que tener como mínimo 2 GB de RAM y 40 GB de disco duro.

RNF 5 Requisitos de Software

Para las PCs clientes:

- Un navegador como *Mozilla Firefox* u otro navegador que cumpla los estándares de *W3C*.
- Sistema operativo: *GNU / Linux Ubuntu* 12.04.

Para los servidores:

- Sistema operativo *GNU / Linux Ubuntu Server* 12.04.
- Servidor Web *Apache* 2.0, con módulo *PHP* 5.
- *MapServer* 6.0, con extensión *PHP Mapscript*.

2.3.4. Diagrama de Casos de Usos del Sistema (DCUS)

Una vez definidos los requisitos funcionales del sistema, es necesario agrupar estos en casos de usos. Un caso de uso representa la manera en que un usuario final interactúa con el sistema en diferentes condiciones (Pressman, 2005); mientras que el Diagrama de Casos de Usos del Sistema muestra los actores, los casos de usos y las relaciones entre ellos. Para el desarrollo del módulo se agruparon los requisitos funcionales en 8 casos de usos, que se muestran a continuación en la Figura 4:

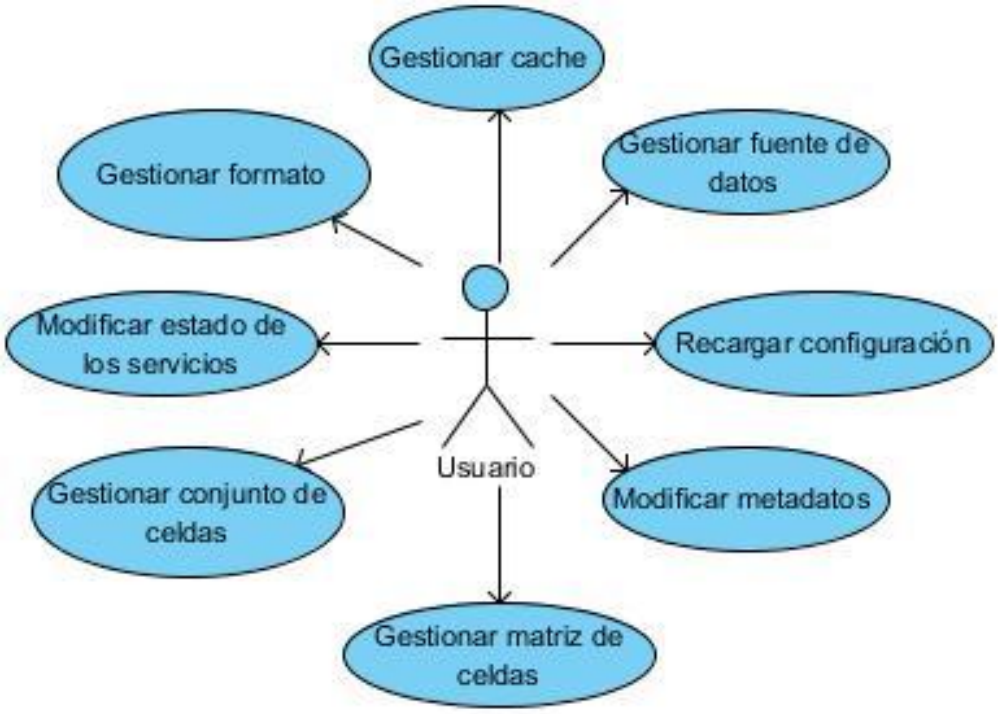

Figura 4. Diagrama de Casos de Usos del Sistema.

2.3.5. Actores del sistema

Los actores son las personas que se comunican con el sistema, representando los papeles que juegan estas conforme el sistema opera (Pressman, 2005). El sistema cuenta con un actor que se describe a continuación. (Ver Tabla 3).

Tabla 3. Descripción del actor del sistema.

Actor	**Descripción**
Usuario	Usuario que configurará el servidor de teselas *MapCache* y agregará los conjuntos de teselas como capa a GeneSIG.

2.3.6. Descripción de los casos de uso del sistema

Teniendo en cuenta que el sistema cuenta con 8 casos de usos, a continuación solo se presenta la descripción del caso de uso Gestionar conjunto de celdas, (ver Tabla 4). El resto de los casos de usos con

su correspondiente descripción se podrá consultar en el artefacto "*Módulo Configuración MapCache Especificación de casos de uso*" ubicado en el expediente de proyecto.

Tabla 4. Descripción del Caso de Uso Gestionar conjunto de celdas.

Caso de uso	**Gestionar conjunto de celdas.**
Objetivo	Gestión de los conjuntos de celdas.
Actores	Usuario
Resumen	El caso de uso inicia cuando el usuario desea crear un nuevo conjunto de celdas, modificar un conjunto de celdas existente, eliminar un conjunto de celdas o adicionar un conjunto de celdas como capa a GeneSIG.
Complejidad	Alta
Prioridad	Crítica
Precondiciones	Existe conectividad con el servidor donde está alojado el servicio de configuración de *MapCache*.
Postcondiciones	En dependencia de la opción escogida por el usuario: • Se crea un nuevo conjunto de celdas. • Se modifica un conjunto de celdas existente. • Se elimina un conjunto de celdas existente. • Se añade un conjunto de celdas como capa a GeneSIG.
Flujo de eventos	

Flujo básico Gestionar conjunto de celdas.		
	Actor	**Sistema**
1	Selecciona la opción "Configurar *MapCache*" ubicada en la barra de herramientas.	Muestra una ventana principal con un menú lateral izquierdo donde están las opciones “Conjunto de celdas”, “Fuente de datos”, “Matriz de celdas”, “Formato”, “Servicio”, “Metadato” y “*Cache*”, estas opciones corresponden a las entradas del fichero de configuración de *MapCache*, y un menú lateral derecho donde se muestra un listado de la entrada que esté seleccionada en ese momento y las opciones “Adicionar”, “Modificar”, “Eliminar” y “Agregar capa a GeneSIG”.
2	Selecciona la opción “Conjunto de celdas”.	El sistema muestra el listado de los conjuntos de celdas que ya están en el sistema y las opciones “Adicionar”, “Modificar”, “Eliminar” y “Añadir capa a GeneSIG”.
3	Selecciona que opción desea realizar: a. “Adicionar” b. “Modificar” c. “Eliminar” d. “Agregar capa a GeneSIG”	En dependencia de la opción seleccionada por el usuario, muestra la interfaz correspondiente: **a.** Ir a la **sección “Adicionar nuevo conjunto de celdas”.** **b.** Ir a la **sección “Modificar conjunto de celdas”.** **c.** Ir a la **sección “Eliminar conjunto de celdas”.** Ir a la **sección “Adicionar conjunto de celdas como capa a GeneSIG”.** **Termina el caso de uso**

Flujos alternos		
Nº Evento 2. El usuario cierra la ventana "Configurar *MapCache*".		
	Actor	**Sistema**
2.1	Cierra la ventana principal correspondiente a la opción "Configurar *MapCache*".	Cierra la ventana principal correspondiente a la opción "Configurar *MapCache*".
Sección 1: "Adicionar conjunto de celdas".		
Flujo básico Adicionar satisfactoriamente un conjunto de celdas.		
	Actor	**Sistema**
3	Presiona el botón "Adicionar".	Muestra una ventana para que el usuario introduzca los datos necesarios para crear un nuevo conjunto de celdas.
4	Introduce los datos y presiona el botón "Aceptar".	Valida los datos, crea el conjunto de celdas y actualiza el listado de los conjuntos de celdas.
Flujos alternos		
Nº Evento 4. Datos en blanco.		
	Actor	**Sistema**
4.1	Deja campos obligatorios en blanco.	Muestra un mensaje indicando que el formulario tiene campos obligatorios en blanco y señala estos campos en color rojo.

Nº Evento 4. Datos incorrectos.		
	Actor	**Sistema**
4.2	Introduce datos con un formato incorrecto.	Muestra un mensaje indicando que se han introducido datos con un formato incorrecto y señala en color rojo los campos con errores.
Nº Evento 4. Cancelar la operación.		
	Actor	**Sistema**
4.3	Presiona el botón “Cancelar”.	Cierra la ventana correspondiente a la adición de un conjunto de celdas.
Nº Evento 4. Cerrar ventana “Adicionar conjunto de celdas”.		
	Actor	**Sistema**
4.4	Cierra la ventana “Adicionar conjunto de celdas”.	Cierra la ventana correspondiente a la adición de un conjunto de celdas.
Sección 2 “Modificar conjunto de celdas”		
Flujo básico Modificar satisfactoriamente un conjunto de celdas.		
	Actor	**Sistema**
3	Selecciona del listado de conjuntos de celdas, el conjunto de celdas que desea modificar y presiona el botón “Modificar”.	Muestra una ventana para que el usuario introduzca los datos necesarios para modificar el conjunto de celdas

		seleccionado.
4	Introduce los datos y presiona el botón “Aceptar”.	Valida los datos, modifica el conjunto de celdas y actualiza el listado de los conjuntos de celdas.
Flujos alternos		
Nº Evento 4. Datos en blanco		
	Actor	**Sistema**
4.1	Deja campos obligatorios en blanco.	Muestra un mensaje indicando que el formulario tiene campos obligatorios en blanco y señala estos campos en color rojo.
Nº Evento 4. Datos incorrectos		
	Actor	**Sistema**
4.2	Introduce datos con un formato incorrecto.	Muestra un mensaje indicando que se han introducido datos con un formato incorrecto y señala en color rojo los campos con errores.
Nº Evento 4. Cancelar la operación.		
	Actor	**Sistema**
4.3	Presiona el botón “Cancelar”.	Cierra la ventana correspondiente a la modificación de un conjunto de celdas.
Nº Evento 4. Cerrar ventana “Modificar conjunto de celdas”.		

	Actor	**Sistema**
4.4	Cierra la ventana “Modificar conjunto de celdas”.	Cierra la ventana correspondiente a la modificación de un conjunto de celdas.
Sección 3: “Eliminar conjunto de celdas”.		
Flujo básico Eliminar satisfactoriamente un conjunto de celdas.		
	Actor	**Sistema**
3	Selecciona del listado de conjuntos de celdas, el conjunto de celdas que desea eliminar y presiona el botón “Eliminar”.	Muestra un mensaje para confirmar la eliminación con los botones “Aceptar” y “Cancelar”.
4	Presiona el botón “Aceptar”.	Muestra un mensaje indicando que se ha eliminado el conjunto de celdas satisfactoriamente y actualiza el listado de conjuntos de celdas.
Nº Evento 4. Cancelar eliminación.		
	Actor	**Sistema**
4.1	Presiona el botón “Cancelar”.	Oculta el mensaje de confirmación y mantiene el conjunto de celdas registrado en el sistema.
Nº Evento 4. Cerrar ventana de confirmación.		
	Actor	Sistema
4.2	Cierra la ventana “Confirmación”.	Oculta el mensaje de confirmación y mantiene el conjunto

		de celdas registrado en el sistema.
	Sección 4: "Adicionar conjunto de celdas como capa a GeneSIG".	
	Flujo básico Adicionar satisfactoriamente un conjunto de celdas como capa a GeneSIG.	
	Actor	**Sistema**
5	Selecciona del listado de conjuntos de celdas, el conjunto de celdas que desea añadir como capa a GeneSIG y presiona el botón "Agregar capa a GeneSIG".	Adiciona como capa a GeneSIG y recarga el mapa de GeneSIG.

2.4. Conclusiones parciales

La elaboración del modelo del dominio del sistema, teniendo en cuenta que no se tienen bien definidos los procesos del negocio, permitió entender el funcionamiento de la LPS para la configuración de *MapCache* y dar continuidad a la captura de los requisitos de software. Los requisitos contribuyeron a definir las funcionalidades con las que debe contar el módulo, y a partir de ellos se elaboró el diagrama de casos de uso del sistema permitiendo definir cómo interactuará el usuario con la aplicación. Una vez realizada la presentación de la propuesta de solución, ya están creadas las bases para dar paso a la fase de implementación.

Capítulo 3. Construcción del módulo de configuración de *MapCache*

En el presente capítulo se explica la arquitectura del módulo y los patrones arquitectónicos y de diseño que se emplearon para el desarrollo del mismo. Además se presentan el modelo de diseño, el modelo de implementación y el modelo de despliegue, artefactos relacionados con el diseño e implementación. Se aborda también sobre la validación de la solución a partir de las pruebas realizadas.

3.1. Arquitectura de desarrollo de software

La arquitectura de desarrollo de software en su forma más simple: *"es la estructura u organización de los componentes del programa (módulos), la manera en que estos componentes interactúan y la estructura de datos que utilizan componentes"* (Pressman, 2005). GeneSIG tiene su estructura basada en el *framework CartoWeb* que posee una arquitectura cliente-servidor. La plataforma cumple entonces con la misma estructura definida por *CartoWeb,* pero utiliza solo los paquetes a los cuales le realizará cambios o aportes funcionales. Teniendo en cuenta que el módulo a desarrollar como ya se dijo anteriormente es para GeneSIG y se define como un *plugin*, este adquiere la arquitectura definida por la plataforma.

3.1.1. Patrones arquitectónicos

"Un patrón arquitectónico provee un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para la organización de las relaciones entre ellos. Estos patrones son plantillas para arquitecturas de software concretas, que especifican las propiedades estructurales de una ampliación con amplitud de todo el sistema y tienen un impacto en la arquitectura de subsistemas"(Camacho y otros, 2004).

La selección de los patrones arquitectónicos es una decisión fundamental en el desarrollo de un software pues permiten darle solución a un problema en específico. En la solución de la investigación se emplean Arquitectura Basada en Componentes y Orientada a Objetos sobre las que se basa la plataforma GeneSIG.

Arquitectura Orientada a Objetos: Se basa en los principios de la Programación Orientada a Objetos (POO): encapsulamiento, herencia y polimorfismo. Sus componentes son los objetos, o más bien instancias de los tipos de dato abstracto. Las interfaces están separadas de las implementaciones y se puede modificar la implementación de un objeto sin afectar a sus clientes (Camacho y otros, 2004).

Arquitecturas Basadas en Componentes: "*Define la composición de software como el proceso de construir aplicaciones mediante la interconexión de componentes de software a través de sus interfaces (de composición), abogaba por la utilización de componentes prefabricados sin tener que desarrollarlos de nuevo*." Esta arquitectura también se conoce como el paradigma de ensamblar componentes y escribir código para que ellos funcionen (Robaina y Sordo, 2008). Simplifica las pruebas y el mantenimiento del sistema, permitiendo agregar o actualizar componentes cuando sea necesario sin tener que afectar el resto del sistema (Terrero, 2005).

3.2. Patrones de diseño

Un patrón de diseño describe la estructura comúnmente recurrente de los componentes en comunicación, que resuelve un problema general de diseño en un contexto particular y tienden a ser independientes de los lenguajes y paradigmas de programación (Camacho y otros, 2004). En el desarrollo de la investigación se emplean varios patrones *GRASP*[34]*,* que describen los principios fundamentales de diseño de objetos para la asignación de responsabilidades, entre los que se encuentran:

Experto en información: Es utilizado para asignar responsabilidades. Esta responsabilidad es asignada a la clase que tiene mayor información para llevar a cabo las funcionalidades necesarias. Este patrón se pone de manifiesto en la clase *Tileset* que es experta en manejar los datos del fichero correspondientes a esta entrada y en la clase *ClientMapcache.php* la cual es experta en manejar las peticiones, siendo esta su responsabilidad. El uso de este patrón permite que se conserve el encapsulamiento, donde cada objeto contiene sus propios atributos y funcionalidades para cumplir su tarea.

Creador: Se utiliza para la creación de responsabilidades relacionadas con la creación de objetos. Se evidencia su uso en la clase *PluginManager* que crea un objeto para acceder al *plugin* encargado de atender la petición de la interfaz.

Controlador: Responsable de recibir o manejar un evento del sistema. Está presente en la clase *Main*, que controla la petición realizada desde la interfaz para acceder a los datos del fichero que atenderán

[34] Patrones Generales de Software para Asignación de Responsabilidades (del término en inglés GRASP: General Responsibility Assignment Software Patterns)

dicha petición y en la clase *PluginManager*, que controla la petición realizada desde la interfaz para poder acceder al *plugin* que atenderá dicha petición.

Alta Cohesión: Asignar a las clases responsabilidades que trabajen sobre una misma área de la aplicación y que no tengan mucha complejidad, evitando así que una clase sea la única responsable de muchas tareas en áreas funcionales muy heterogéneas.

Bajo acoplamiento: Se asignan las responsabilidades de forma tal que cada clase se comunique con el menor número de clases, minimizando el nivel de dependencia.

Los patrones alta cohesión y bajo acoplamiento están estrechamente relacionados y son utilizados por el sistema en la mayoría de las clases.

Además de los patrones *GRASP*, se emplearon dos de los patrones *GoF*[35], conocidos también como "pandilla de los cuatro": *Singleton* y *Command*.

Singleton: Establece la conexión mediante una instancia única a la clase, controlando así el acceso a la misma (Larman, 2003). Se muestra en la clase *ServerContex*t utilizando el método *GetMapObject*, para garantizar una única instancia del objeto mapa en toda la ejecución del programa.

Command: Encapsula las peticiones a través de un objeto, lo que permite realizar operaciones como gestionar las acciones de dicho objeto (Larman, 2003). Se evidencia en la clase *AJAXHelper* que es la encargada de comunicar las interfaces de usuario con el servidor.

3.3. Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en como los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el sistema a considerar (Jacobson y otros, 2000). La elaboración de un correcto modelo del diseño facilita la realización de un software con alta calidad.

[35] Gang-of-Four

3.3.1. Diagrama de paquetes

En *UML* se modelan varios diagramas que permiten presentar el diseño de la aplicación. Uno de estos diagramas es el de paquetes que muestra como el sistema está dividido en agrupaciones lógicas y las dependencias entre ellas (Pressman, 2003). Este diagrama representa una vista general de cómo está formado el sistema y facilita el entendimiento de este, (ver Figura 5).

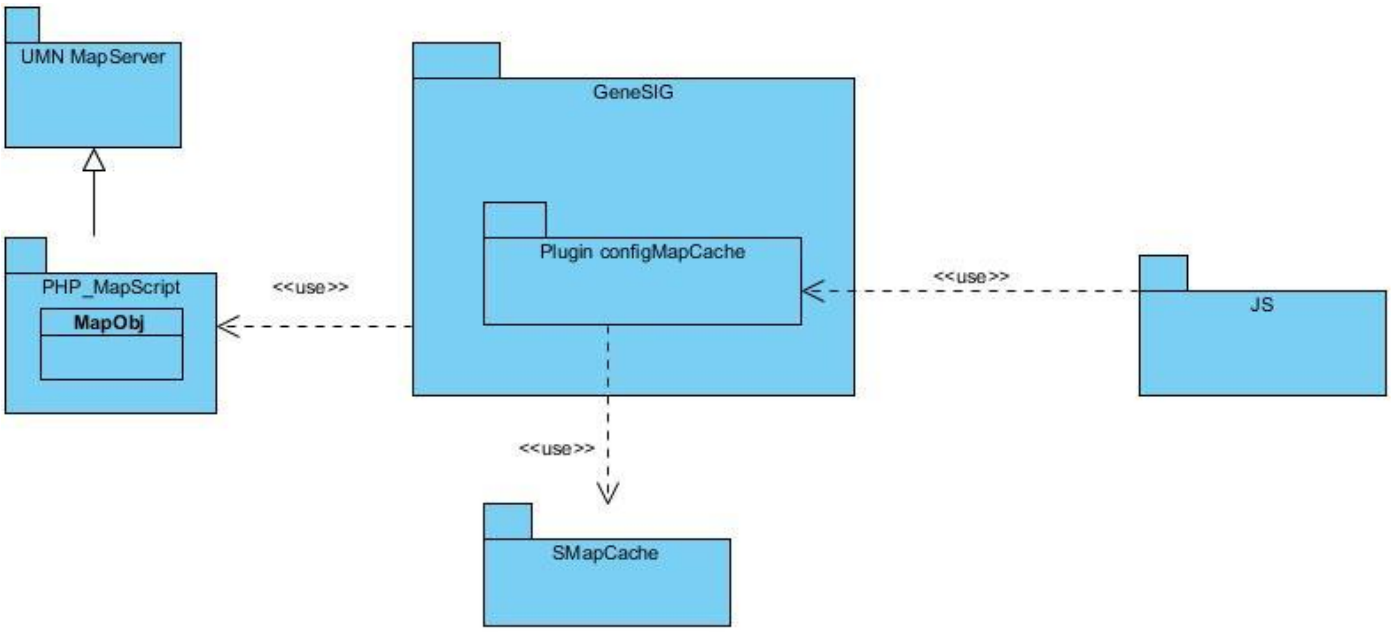

Figura 5. Diagrama de paquetes del sistema.

3.3.2. Diagrama de clases del diseño del caso de uso "Gestionar conjunto de celdas"

El diagrama de clases del diseño constituye la representación de las clases del sistema, su estructura y organización. A continuación se presenta el diagrama de clases del diseño correspondiente al caso de uso Gestionar conjunto de celdas (ver Figura 6), el resto de los diagramas correspondientes a los demás casos de uso se podrán consultar en el artefacto *"Módulo Configuración MapCache Modelo de diseño"* ubicado en el expediente de proyecto.

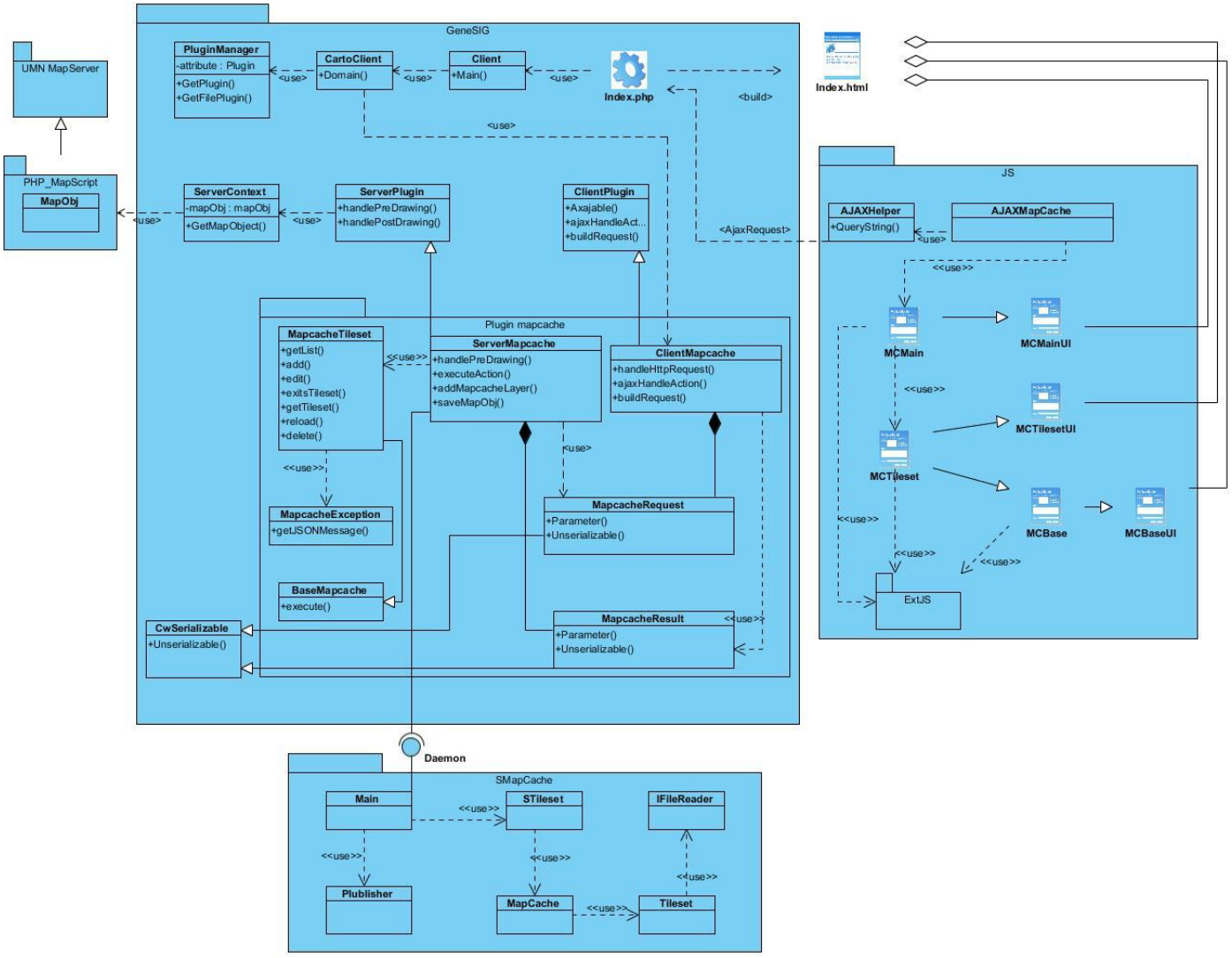

Figura 6. Diagrama de clases del diseño del caso de uso "Gestionar conjunto de celdas".

Para un mejor entendimiento, se procede a presentar de manera separada las clases que conforman los paquetes antes mencionados. En primer lugar se definen las clases del paquete *SMapCache* que contiene los métodos que permiten escribir y leer los datos del fichero de configuración *XML* de *MapCache* (ver Figura 7), y se exponen a través de servicios, garantizando la interoperabilidad. Desde la plataforma el componente *configMapCache* maneja las peticiones de GeneSIG y se encarga de acceder a los servicios mediante *SOAP* (ver Figura 8).

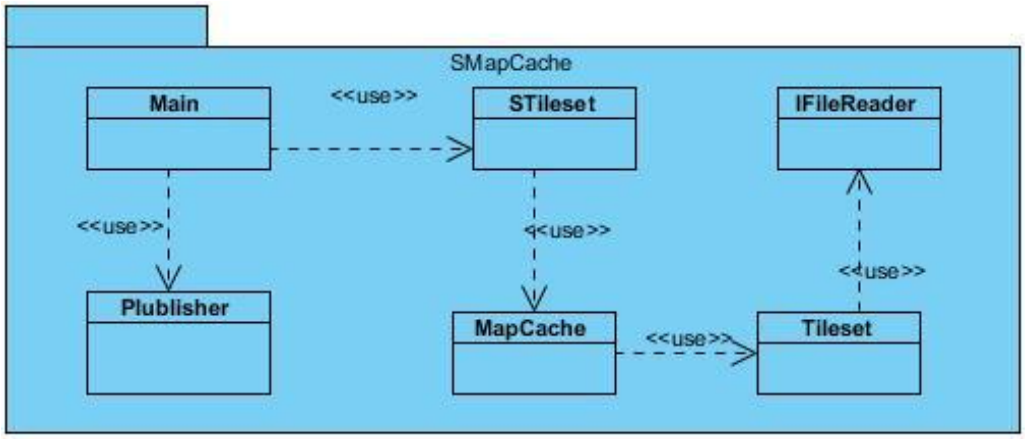

Figura 7. Clases del diseño del paquete SMapCache.

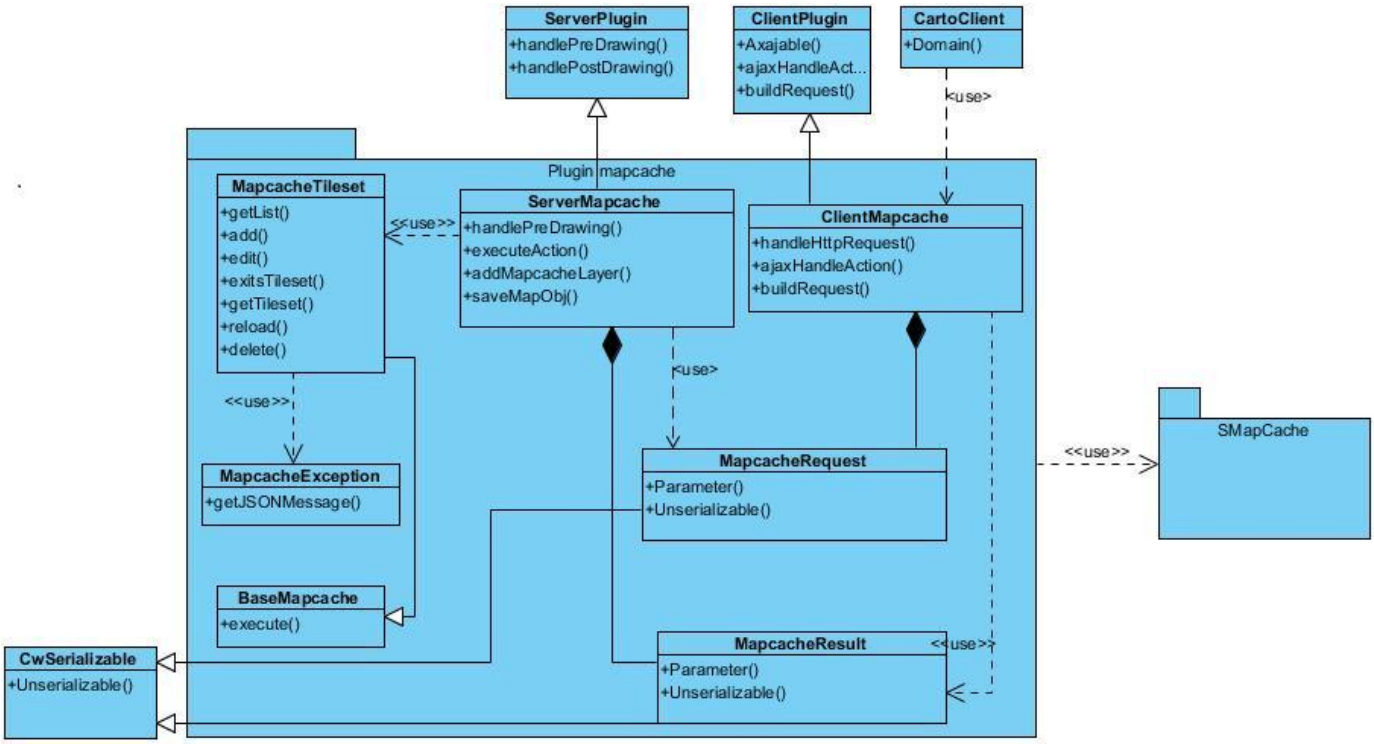

Figura 8. Clases del diseño del paquete Plugin mapcache.

En el paquete GeneSIG se encuentran las clases encargadas de hacer funcionar la plataforma, donde *ServerPlugin, ClientPlugin* y *CwSerializable* garantizan el funcionamiento de los componentes de la plataforma y constituyen las clases bases de estos (ver Figura 9); mientras que en el paquete JS es donde se encuentran las interfaces visuales que utilizan *ExtJS* para crear los componentes visuales, y dentro hay una clase *AJAXHelper* que tiene un mecanismo para enviar los datos al servidor *(*ver Figura 10*).*

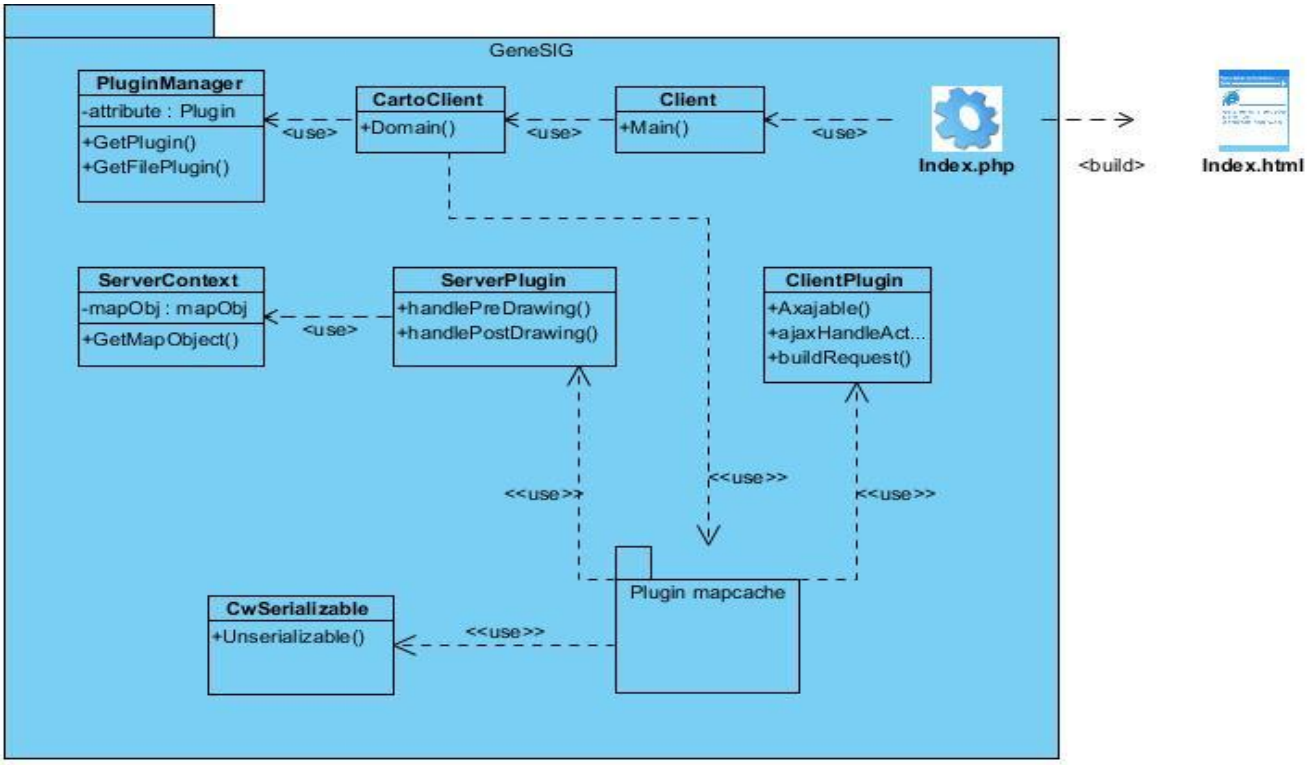

Figura 9. Clases del diseño del paquete GeneSIG.

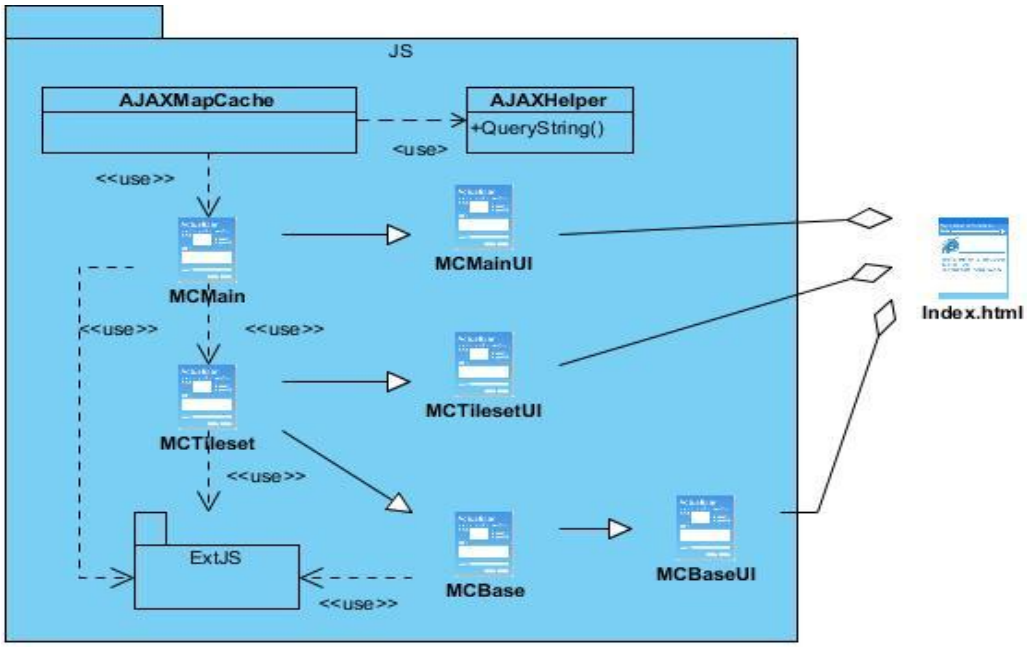

Figura 10. Clases del diseño del paquete JS.

3.3.3. Modelo de Despliegue

"*El modelo de despliegue es un modelo de objetos que describe la distribución física del sistema en términos de cómo se distribuye la funcionalidad entre los nodos de cómputo*"(Jacobson y otros, 2000). Con el objetivo de representar los recursos de cómputo que necesita el módulo para su correcto funcionamiento se muestra el diagrama de despliegue (ver Figura 11). Debido a que la solución propuesta se desarrolla sobre tecnologías web, se utilizó el protocolo de comunicación *HTTP*.

Protocolo *HTTP*: Protocolo de comunicaciones estándar que comunica servidores y clientes. El cliente envía un mensaje de petición y el servidor contesta con un mensaje de respuesta, cuyo contenido es en función de la petición realizada por el cliente.

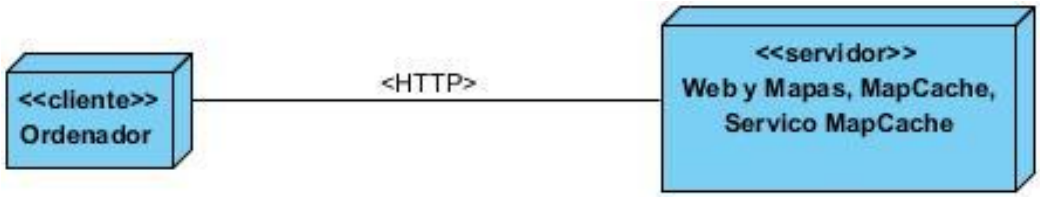

Figura 11. Diagrama de Despliegue

3.4. Modelo de Implementación

El modelo de implementación permite representar como se implementan en términos de componentes, los elementos del modelo de diseño. Describe también como dependen los componentes unos de otros, y como se organizan de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en los lenguajes de programación utilizados (Jacobson y otros, 2000). El diagrama de componentes que se muestra a continuación (ver Figura 12) permite representar los principales componentes del sistema utilizados en la implementación del caso de uso Gestionar conjunto de celdas.

Cada elemento del diseño se representa como un componente en el diagrama excepto en el caso de las clases *MapcacheRequest* y *MapcacheResult* que se representan en el componente Mapcache.php. El resto se representan en un componente con el nombre de la clase y la extensión en dependencia del tipo de componente: js para las clases de interfaz visual, php para las clases del lado del servidor y *Java* para las clases del servicio. De igual manera sucede en la implementación del resto de los casos de uso.

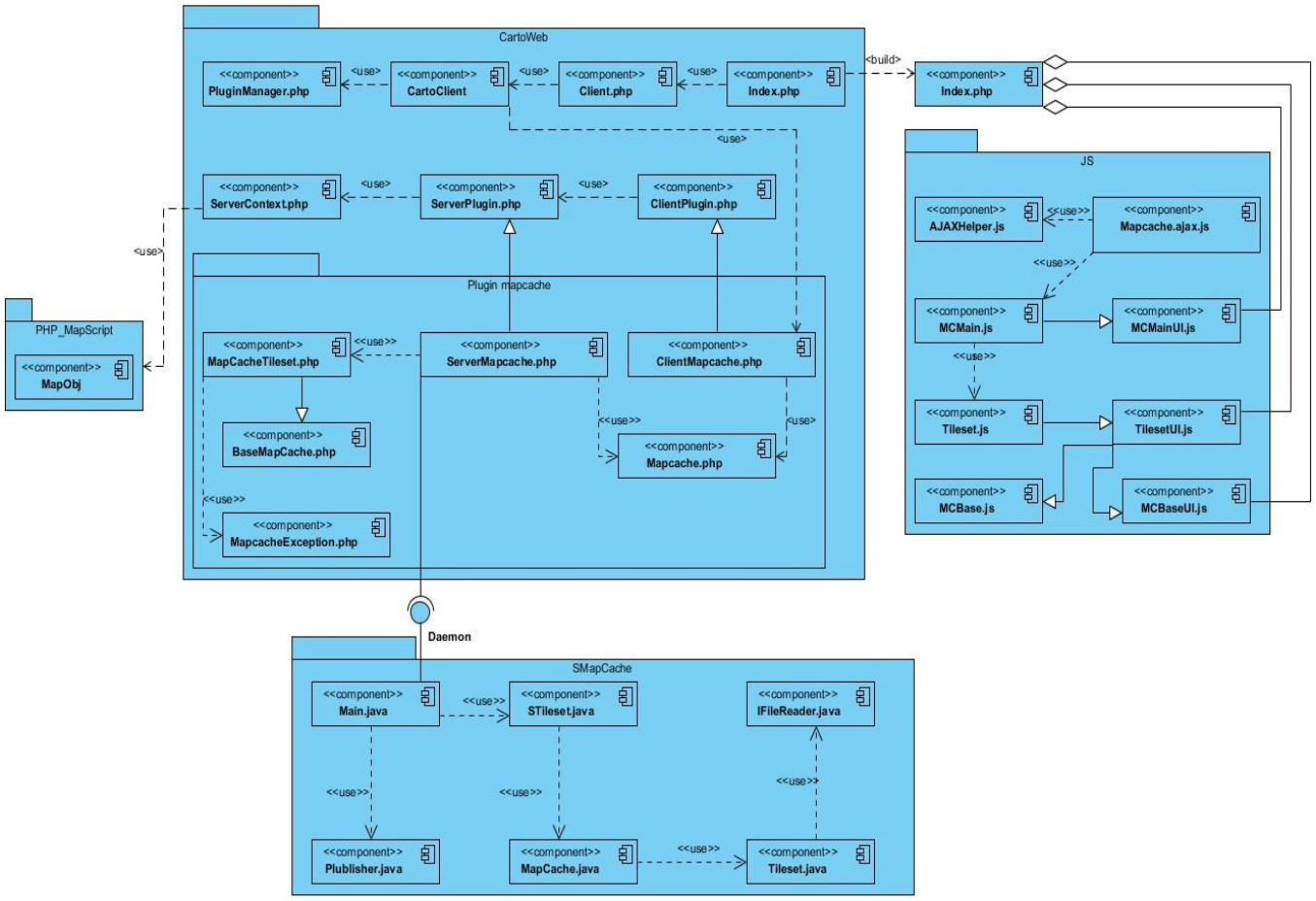

Figura 12. Diagrama de Componentes del Caso de Uso "Gestionar conjunto de celdas".

3.5. Pruebas de software

Una vez finalizada la implementación del módulo, se hace necesario comprobar que su funcionamiento sea correcto, verificando que las funcionalidades se ajusten a las especificaciones planteadas. Para ello se realizan las pruebas de software, con el fin de detectar defectos y asegurar que estos sean corregidos antes de entregar el software al cliente. El objetivo es diseñar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores. Estos determinan un conjunto de entradas, condición de ejecución y resultados esperados para un objetivo en particular; para generarlos existen técnicas que proporcionan criterios distintos. En este caso se aplican técnicas de caja negra, para probar las interfaces

del módulo desarrollado y un pre-experimento para hacer un análisis antes y después de aplicar la propuesta de solución.

3.5.1. Pruebas de caja negra

Las pruebas de cajas negras o funcionales son realizadas a la interfaz del software y para aplicarlas solo se necesita conocer la funcionalidad del mismo. Estas pruebas se centran en los requisitos funcionales, permitiendo derivar conjuntos de condiciones de entradas que ejercitarán por completo dichos requerimientos. Para seleccionar el conjunto de entradas con el que trabajar, hay que tener en cuenta que existen algunos que causan comportamientos erróneos en el sistema, y como consecuencia producen una serie de salidas que revelan la presencia de defectos, teniendo como objetivo final encontrar un colección de datos cuya probabilidad de pertenecer a dicho grupo sea la más alta posible. Para confeccionar los casos de prueba existen distintos criterios, algunos de ellos son: particiones de equivalencia y análisis de valores límite (Juristo y otros, 2006). Se seleccionó para aplicar el criterio de particiones de equivalencia.

El método seleccionado para detectar los errores divide el dominio de entrada de un programa en clases de datos de los que se puede derivar casos de prueba, reduciendo así el número total a desarrollar. El diseño de los mismos consta de dos pasos, primero identificar las clases de equivalencia, que representan un conjunto de valores válidos y no válidos para una condición de entrada, y luego de identificadas se crean los casos de prueba (Juristo y otros, 2006).

El diseño de los casos de prueba se basa en la descripción de los casos de uso. Cada planilla de caso de prueba recoge la especificación de un caso de uso, dividido en secciones y estas en escenarios, detallando las funcionalidades descritas en él y describiendo cada variable que recoge el caso de uso que se esté analizando. La descripción de los casos de pruebas aplicados a los casos de uso definidos, se pueden consultar en la carpeta "*Módulo Configuración MapCache Casos de Prueba"* ubicada en el expediente del proyecto.

Resultados de las pruebas de Caja Negra

"El módulo de configuración de *MapCache* para la plataforma GeneSIG" fue sometido a una primera iteración de pruebas, en la que se detectaron 8 no conformidades en la primera iteración: 3 errores ortográficos y 5 funcionalidades incorrectas (Ver figura 7), que fueron corregidas; luego se procede a una segunda iteración con el objetivo de verificar que las no conformidades antes detectadas estuvieran totalmente resueltas y no generaran otras, obteniéndose resultados satisfactorios.

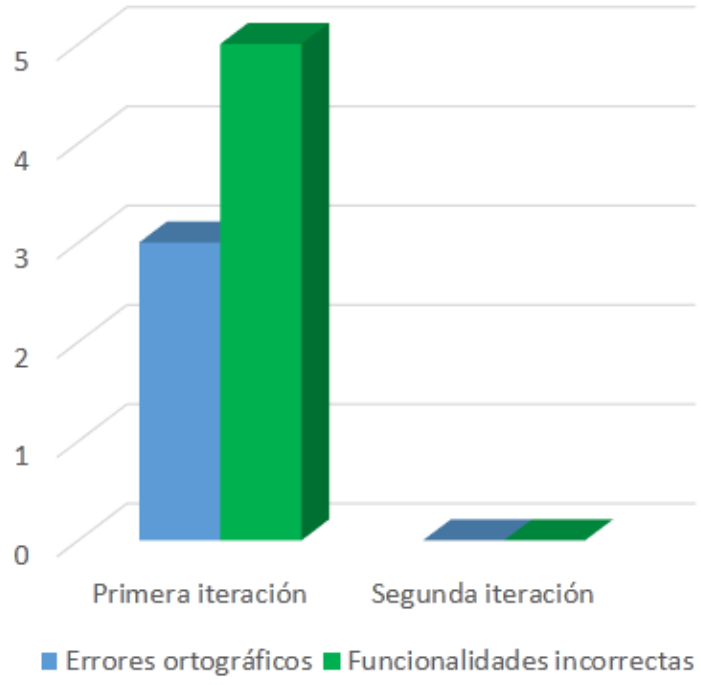

Figura 13. Resultados de las pruebas de caja negra

3.5.2. Pre-experimento

Como parte de la validación de la propuesta de solución se realizó un pre-experimento que permitió hacer un análisis previo y posterior a la aplicación de la propuesta de solución y para ello se definieron las variables: tiempo de configuración, errores introducidos y nivel de complejidad. Se elaboró un instrumento de diagnóstico (Ver Anexo 1) que a través de la aplicación de una encuesta se obtienen los criterios de medida manejados en las variables definidas. A continuación se describen los elementos que conforman el pre-experimento (Sampieri y otros, 2006):

G O1 X O2

G: Grupo de experimentación compuesto por los especialistas del proyecto Aplicativos_SIG y GeneSIG.

O1: Observación sobre la configuración de *MapCache* antes de aplicar la propuesta de solución.

X: Aplicación de la propuesta al grupo de experimentación a través del despliegue del módulo de configuración de *MapCache*.

O2: Observación sobre la configuración de *MapCache* después de aplicar la propuesta de solución.

La población estuvo conformada por todos los especialistas de los proyectos Aplicativos_SIG y GeneSIG que constituyen un total de 23 especialistas (Ver Anexo 2). Para la aplicación del instrumento de diagnóstico se seleccionó una muestra de 7 profesionales que se corresponden con los especialistas que tienen acceso al fichero de configuración de *MapCache.* Se utilizó el método no probabilístico intencional (León y González, 2011) y dicha muestra representa el 30,4 % de la población, por tanto constituye un valor significativo para el análisis.

3.5.3. Resultados de la encuesta aplicada

Para la aplicación de la encuesta, a la muestra seleccionada se les pidió que configuraran *MapCache* a través del fichero *XML* y luego se desplegó el módulo en la estación de trabajo de cada uno, para que interactuaran con él. De esta manera se recogen sus impresiones antes y después de utilizar el sistema y se evalúa el tiempo de configuración, los errores introducidos y el nivel de complejidad; obteniéndose resultados satisfactorios, modelando en gráficos de pastel el análisis realizado para cada una de las variables.

En la primera pregunta, enfocada en medir el tiempo de configuración, se obtuvieron resultados que demostraron que antes de aplicar la propuesta de solución, el tiempo de configuración era elevado, en la mayoría de los casos de 30 a 60 minutos y en otros más de 60; mientras que luego de aplicarla estos valores oscilaban entre 5 y 30 minutos (ver Figura 14). Una vez analizados estos resultados, se concluye que al utilizar el módulo de configuración de *MapCache* para la plataforma GeneSIG el tiempo empleado para configurar este servidor de teselas no sobrepasa de los 30 minutos, disminuyendo considerablemente este factor con respecto a la configuración directa en el fichero *XML.*

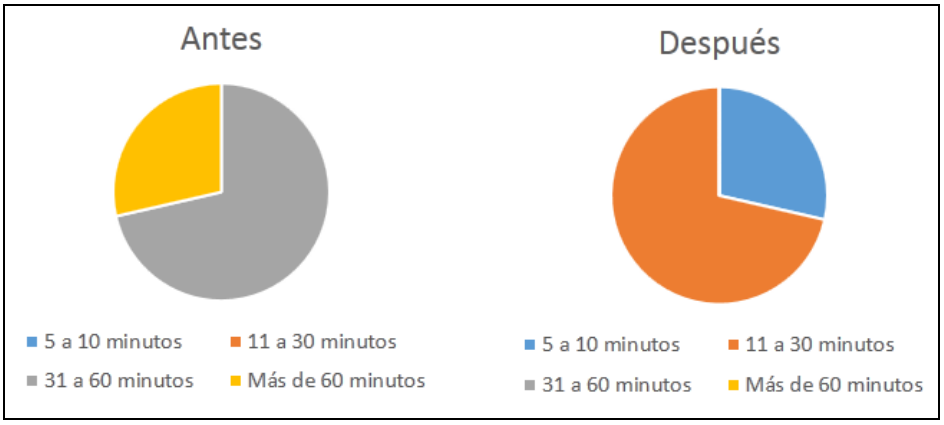

Figura 14. Análisis del tiempo empleado en la configuración de MapCache.

En la segunda pregunta se evaluaba mediante la aplicación de la encuesta otro factor importante, la cantidad de errores introducidos en la configuración. Como resultado se obtuvo que cuando la configuración se realizó directamente en el fichero *XML*, los especialistas cometieron una cantidad de errores que variaba en su mayoría entre los 11 y 20 errores, alcanzándose en algunos casos más de 20. Por su parte al realizar el mismo proceso luego de aplicada la propuesta de solución los errores no sobrepasaron de 10 y en algunas casos hubo ausencia total de los mismos. (Ver Figura 15).

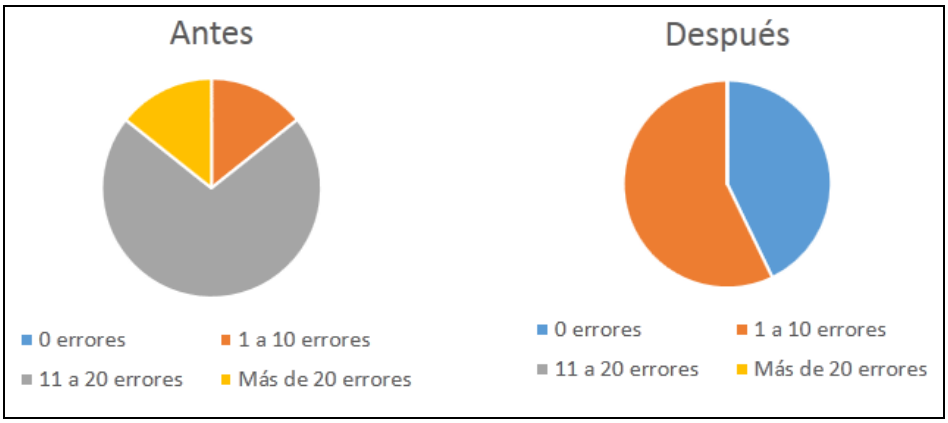

Figura 15. Análisis de la cantidad errores introducidos en la configuración de MapCache.

En la tercera y última pregunta se analizó la complejidad al realizar la configuración, un factor muy importante en el análisis de la usabilidad de la configuración de MapCache, arrojando los siguientes resultados. Previo a la aplicación de la propuesta de solución los niveles de complejidad eran mayormente altos y en ocasiones medios; luego de aplicada la solución disminuyeron considerablemente siendo en su mayoría bajos y en algunos casos medios (ver Figura 16), permitiendo demostrar que el módulo

desarrollado brinda a los especialistas una interfaz sencilla y fácil de usar que permite realizar la configuración vía web, evitando el difícil trabajo con un fichero *XML*.

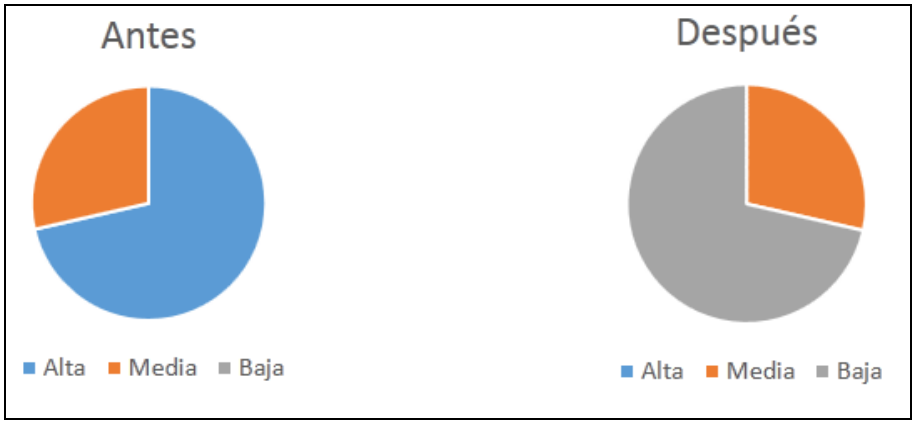

Figura 16. Análisis de los niveles de complejidad al realizar la configuración de MapCache

Por todos los resultados anteriormente expuestos se puede afirmar que el módulo desarrollado mejora el proceso de configuración de *MapCache*, pues al aplicarlo el tiempo de configuración disminuyó considerablemente y no sobrepasa los 30 minutos; el rango de errores introducidos es pequeño variando de 0 a 10 errores; además se redujo la complejidad al realizar la configuración, de ser mayormente alta a baja. El módulo propone una vía sencilla para introducir los datos en el fichero *XML* a través de interfaces intuitivas y permite añadir los conjuntos de teselas creados como capa a GeneSIG y visualizarlas en el mapa.

Conclusiones parciales

El empleo adecuado de los patrones arquitectónicos posibilitó realizar una correcta implementación. A través del modelo de diseño se logró representar la estructura del sistema y se generó el modelo de despliegue. Por su parte la elaboración del modelo de componentes posibilitó representar en términos de componentes, los elementos antes descritos en el diseño. Además las pruebas permitieron encontrar errores que fueron solucionados, garantizando el correcto funcionamiento del sistema. De su aplicación se concluye que el tiempo de configuración disminuyó considerablemente y no sobrepasa los 30 minutos; el rango de errores introducidos es pequeño variando de 0 a 10 errores; se redujo la complejidad al realizar la configuración, de ser mayormente alta a baja; estos datos permiten demostrar que la propuesta de solución mejora el proceso de configuración de *MapCache*.

Conclusiones generales

Del contenido del trabajo presentado, de los antecedentes revisados en la literatura y su análisis se arriban a las siguientes conclusiones:

- El servicio *WMS* permite mejorar la visualización de los mapas para un mejor entendimiento por parte de los usuarios, facilitando el procesamiento de la información geográfica en los SIG.
- El análisis realizado sobre las soluciones existentes, permitió conocer las características de los servidores de teselas y las herramientas útiles para la configuración de ficheros *XML*.
- Con la aplicación del proceso de desarrollo *AUP* se obtuvieron los artefactos generados del proceso ingenieril durante el desarrollo de la solución.
- Con la implementación del sistema, se logró obtener un software que permite la configuración de *MapCache* en la plataforma GeneSIG, lo cual contribuirá a mejorar el rendimiento de los profesionales que realizan personalización en la LPS Aplicativos_SIG.
- Una vez realizadas las pruebas de software pertinentes, se logró una aplicación que cumple con todos los requisitos funcionales definidos y que satisface las necesidades del cliente; se demostró que mejoró el proceso de configuración de *MapCache* a partir de que disminuyeron los tiempos empleados para realizar las configuraciones, los errores introducidos así como la complejidad en la realización de dicha actividad.

Recomendaciones

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

- Extender el desarrollo de la configuración de *MapCache*, independiente de la plataforma GeneSIG, para consumir los servicios que posibilitan escribir y leer desde un fichero *XML.*
- Realizar cursos de entrenamiento para los usuarios que laboran con la plataforma GeneSIG para que desarrollen habilidades sobre la configuración de *MapCache* usando la propuesta de solución.

Referencias bibliográficas

ADTRIBOO 2014. Java, características y ventajas de uno de los lenguajes de programación más populares |. [En línea]. [Consulta: 16 noviembre 2014]. Disponible en: http://blog.adtriboo.com/2014/01/21/java-ventajas-caracteristicas-lenguaje-programacion-mas-populares/.

AUP Ingeniería de Software. [En línea] 2015. [Consulta: 16 abril 2015]. Disponible en: http://ingenieriadesoftware.mex.tl/63758_AUP.html.

BELMONTE, O. 2005. Introducción al lenguaje de programación Java.

BOSQUE, J. y GÓMEZ, M. 2001. Cálculo de rutas óptimas para el transporte de residuos tóxicos y peligrosos.

BOURRET, R., 2010. *XML DataBases* [en línea]. 2010. S.l.: s.n. Disponible en: http://www.rpbourret.com/xml/XMLDatabaseProds.htm.

BROWNELL, D. 2008. *SAX2*. S.l. : O'Reilly. ISBN 0-596-00237-8.

CAMACHO, E., CARDESO, F. y GABRIEL NUÑEZ 2004. Arquitecturas de Software. Guía de estudio.

CARMONA, A. de J. y MONSALVE, J.J. 1999. Sistemas de Información Geográficos.

CHENGKAI LI. 2009. *«XML Parsing, SAX/DOM».* [En línea]. S.l.: Department of Computer Science and Engineering, University of Texas at Arlington. Disponible en: ranger.uta.edu/~cli/.../XMLParsing_ChengkaiLi.pdf.

CONTRIBUTORS, T. 2010. TileCache, from MetaCarta Labs. [En línea]. [Consulta: 20 noviembre 2014]. Disponible en: http://tilecache.org/.

CORDERO, J.J.T. 2014. *La guía definitiva de XML: ¡¡XML, JSON y mucho más!!* S.l.: s.n.

CUÉLLAR LUNA, L., CONCEPCIÓN ROJAS, M., RAMÍREZ, B., ÁLVAREZ VALDEZ, Á. y DÍAZ, C. 2009. Los sistemas de información geográfica y su empleo en un sistema de vigilancia integrado para la prevención del dengue en un municipio de ciudad de La Habana. *No 9*, pp. 166-183. ISSN 1578-5157.

DÍAZ, A.A.P. 2005. *Graphical User Interface (GUI) para el programa servidor de mapas MapServer 4.6.1*. Trabajo para optar al título de ingeniero de ejecución en informática. Temuco Chile: Universidad de la Frontera.

DÍAZ, A., REYNA, M.A. de y SANZ, J. 2014. Servidores — Panorama del SIG Libre. [En línea]. [Consulta: 21 noviembre 2014]. Disponible en: http://panorama-sig-libre.readthedocs.org/es/latest/servidores/.

DORANTES, L.M., GARCÍA, T. y GARCÍA, C.F., 2006. *Desarrollo de aplicaciones Web con XML y Java*. 2006. S.l.: IEEE 4º Congreso Internacional en Innovación y Desarrollo Tecnológico.

ESCALONA, M.J. y KOCH, N. 2002. Ingeniería de Requisitos en Aplicaciones para la Web – Un estudio comparativo. ,

GÁMEZ, I. 2008. Escalabilidad en servicios de mapas. Modelo de teselas en cache con OpenLayers.

GARCÍA, J.E.G. 2011. *Sistema de gestión de diálogos para Entrenadores Aduaneros*. Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas. La Habana: Universidad de las Ciencias Informáticas.

GARCÍA, R.B. 2006. *Verificación de firmas digitales en documentos XML de tamaño arbitrario.* Trabajo fin de carrera. Boadilla del Monte: Universidad Politécnica de Madrid. Facultad de Informática.

GARCÍA, R. y CASTRO, J.P. de 2010. Benchmarking de implementaciones WMS-C de software libre.

GARCÍA, R., CASTRO, J.P. de, REGUERAS, L.M., VERDÚ, E. y VERDÚ, M.J. 2012. Desarrollo de un sistema inteligente para la precarga automática de teselas en GeoWebCache a partir de un catálogo de fenómenos geográficos.

GeoWebCache. 2008-2014, [en línea]. [Consulta: 15 noviembre 2014]. Disponible en: http://geowebcache.org/.

GONZÁLEZ, L.C. y TORRES, E.R.P. 2012. Extensión de Visual Paradigm for UML para el desarrollo dirigido por modelos de aplicaciones de gestión de información. , vol. 5, no. nº 10.

HERNÁNDEZ, E. 2012. El Lenguaje Unificado de Modelado (UML). [En línea]. [Consulta: 15 noviembre 2014]. Disponible en: www.disca.upv.es/enheror/pdf/ActaUML.PDF.

INAREJOS, R.O. 2014. *Sistema de visualización SIG del Servei Meteorològic de Catalunya.* S.l.: UNIVERSITAT OBERTA DE CATALUNYA.

JACOBSON, I., BOOCH, G. y JAMES RUMBAUGH 2000. *El Proceso Unificado de Desarrollo de Software*. Madrid: s.n.

JEUDI 2008. XML | Observatorio Tecnológico. [En línea]. [Consulta: 13 enero 2015]. Disponible en: http://recursostic.educacion.es/observatorio/web/fr/software/programacion/675-xml.

JURISTO, N., MORENO, A.M. y VEGAS, S. 2006. *Técnicas de Evaluación de Software*. S.l.: UNIVERSIDAD SIMON BOLIVAR.

LARMAN, C. 2003. *Uml y Patrones.Una Introducción al Análisis y Diseño Orientado a Objetos y al Proceso Unificado.* Segunda Edición. S.l.: s.n. ISBN 84-205-3438-2.

LEÓN, R.A.H. y GONZÁLEZ, S.C. 2011b. *El proceso de investigación científica.* Ciudad de La Habana: Editorial Universitaria. ISBN 978-959-16- 1307-3.

MAPSERVER 2014. MapCache — MapServer 6.4.1 documentation. [En línea]. [Consulta: 15 noviembre 2014]. Disponible en: http://mapserver.org/el/mapcache/index.html.

MASÓ, J., JULIÀ, N. y PONS, X. 2010. El nuevo estándar internacional OGC-WMTS. Oportunidades de aplicación y rendimiento versus OGC-WMS.

MESTRAS, J.P., 2013. *Java EE – Apache Tomcat Aplicaciones Web/Sistemas Web*. 2013. S.l.: s.n.

MONGE, L.Á., TORRES, J.P., LÓPEZ, L.E. y NAVARRO, C.X. 2010. Análisis comparativo de servidores de mapas. , ISSN 1578-5157.

OGC 2006. OpenGIS® Web Map Server Implementation Specification.

OGC 2014. OGC Standards | OGC. [En línea]. [Consulta: 2 noviembre 2014]. Disponible en: http://www.opengeospatial.org/standards/is.

OLAYA, V. 2010. Sistemas de Información Geográfica.

OPENGEO 2014. WMTS - Web Map Tiling Service — GeoWebCache 1.6.0 User Manual. [En línea]. [Consulta: 17 noviembre 2014]. Disponible en: http://geowebcache.org/docs/1.6.0/services/wmts.html.

OPEN SOURCE GEOSPATIAL FOUNDATION 2014. Caching with GeoWebCache — GeoServer 2.6.x User Manual. [En línea]. [Consulta: 17 diciembre 2014]. Disponible en: http://docs.geoserver.org/2.6.x/en/user/geowebcache/index.html.

ORACLE 2015. Bienvenido a NetBeans y www.netbeans.org, Portal del IDE Java de Código Abierto. [En línea]. [Consulta: 16 febrero 2015]. Disponible en: https://netbeans.org/index_es.html.

ORT, E. y MEHTA, B. 2003. Java Architecture for XML Binding (JAXB). [En línea]. [Consulta: 12 enero 2015]. Disponible en: http://www.oracle.com/technetwork/articles/javase/index-140168.html.

OSGEO 2014a. GeoServer. [En línea]. [Consulta: 2 noviembre 2014]. Disponible en: http://geoserver.org/.

OSGEO 2014b. Mapserver. [En línea]. [Consulta: 2 noviembre 2014]. Disponible en: http://mapserver.org/es/.

PALMIRANI, M. y VITALI, F., 2012. *XML Legislativo: Principios e Instrumentos Técnicos*. 2012. S.l.: s.n.

PANTOJA, Y. y TORRES, S. 2012. *Modelo de desarrollo basado en líneas de productos de software para Sistemas de Información Geográfica sobre la base de la Plataforma GeneSIG*. Maestría. La Habana: Universidad de las Ciencias Informáticas.

PÉREZ, M.A. y LALLANILLA, R.E. 2014. Sistema para la visualización y construcción de caché de mapas.

PRESSMAN, R.S. 2003. *Ingeniería del software. Un enfoque práctico.* S.l.: s.n.

PRESSMAN, R.S. 2005. *Ingeniería del Software.Un enfoque práctico*. Sexta. S.l.: s.n.

PRESSMAN, R.S. 2010. *Software Engenineering*. Seventh. S.l.: s.n.

ROBAINA, I.L. y SORDO, V.L.R. 2008. *Propuesta del Diseño Arquitectónico del Simulador de Sistemas Biológicos: BioSyS.* S.l.: Universidad de las Ciencias Informáticas.

SAMPIERI, R.H., COLLADO, C.F. y PILAR BAPTISTA LUCIO 2006. Concepción o elección del diseño de investigación. . Cuarta Edición. S.l.: s.n., ISBN: 970-10-5753-8.

SOMMERVILLE I. 2005. *Ingeniería del Software*. Séptima edición. S.l.: s.n.

SPHINX 2014a. Configuration File — MapServer 6.4.1 documentation. [En línea]. [Consulta: 17 abril 2015]. Disponible en: http://mapserver.org/es/mapcache/config.html.

SPHINX 2014b. MapServer Suite Products — MapServer 6.4.1 documentation. [En línea]. [Consulta: 20 noviembre 2014]. Disponible en: http://mapserver.org/es/products.html.

SUN MICROSYSTEMS, INC., 2001. *The Java Architecture for XML Binding User's Guide*. 2001. S.l.: s.n.

TENDERO, J.E.C. y SIMARRO, F.M. 2013. Capítulo 1 Manejo de ficheros. *Acceso a datos .CFGC*. S.l.: RA-MA

TERRERO, J.C. 2005. Desarrollo de Software basado en Componentes. [En línea]. [Consulta: 18 enero 2015]. Disponible en: https://msdn.microsoft.com/es-es/library/bb972268.aspx#mainSection.

TERRY, N.D.R. 2012. *Diseñador y generador de formularios de formularios web v2.0*. La Habana: Universidad de las Ciencias Informáticas.

THE APACHE SOFTWARE FOUNDATION 2013. Visión general de las nuevas funcionalidades de Apache 2.0 - Servidor HTTP Apache. [En línea]. [Consulta: 11 junio 2015]. Disponible en: http://httpd.apache.org/docs/2.0/es/new_features_2_0.html.

THE APACHE SOFTWARE FOUNDATION. 2015. Welcome! - The Apache HTTP Server Project. [En línea]. [Consulta: 10 enero 2015]. Disponible en: http://httpd.apache.org/.

UVA, M.A. y CAMPANELLA, O. 2009. AP-SIG: un SIG con funciones específicas para Agricultura de Precisión.

VÁZQUEZ, C. 2008. Programación en PHP5.

W3C 2005. W3C Document Object Model. [En línea]. [Consulta: 9 junio 2015]. Disponible en: http://www.w3.org/DOM/.

W3C 2013. Extensible Markup Language (XML). [En línea]. [Consulta: 28 noviembre 2015]. Disponible en: http://www.w3.org/XML/.

Glosario de términos

BSD: es una licencia de software libre permisiva, o sea es flexible respecto a la distribución, de modo que el software puede ser redistribuido como software libre o propietario, siendo libre la licencia original del autor.

CartoWeb: aplicación de publicación *WebGIS* construida en *PHP* sobre *UMN MapServer* que explota *AJAX*.

CGI: es una importante tecnología de la *WWW* que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en el servidor.

Clúster*:* es el conjunto o conglomerado de computadoras construidos mediante la utilización de hardwares comunes.

DiskCache: es la *cache* en disco.

KVP: es un conjunto de dos elementos de datos vinculados: una clave, que es un identificador único para algún elemento de los datos, y el valor, que puede ser los datos que se identifica o un puntero a la ubicación donde están estos.

Mapnik: es una herramienta para el renderizado de un mapa.

Memcache: es una memoria de clave y valor para almacenar en pequeños trozos de datos arbitrarios (cadenas, objetos).

MemoryCache: es la *cache* en memoria.

Modelo *raster*: supone la existencia de un área de estudio sobre la cual se sobrepone un sistema de cuadrículas, donde cada unidad se denomina celda y tienen la misma forma.

Modelo vectorial: constituye una codificación de los datos geográficos en la que se representa una variable geográfica por su geometría, independientemente de su escala.

OGC: es una organización que tiene como misión el promover el uso de estándares y tecnologías abiertas en el área de los sistemas y tecnologías de la información geográfica y áreas afines.

Proxy: servidor (programa o sistema informático) que sirve de intermediario en las peticiones de recursos que realiza un cliente a otro servidor.

REST: es un estilo de arquitectura de software para sistemas hipermedia distribuidos como la *World Wide Web*.

RIA: son aplicaciones web que tienen la mayoría de las características de las aplicaciones de escritorio tradicionales. Estas aplicaciones utilizan un navegador web estandarizado para ejecutarse y por medio de complementos o mediante una máquina virtual se agregan las características adicionales.

Script: es un programa usualmente simple, que por lo regular se almacena en un archivo de texto plano.

SOAP: es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos *XML*.

SVG: es un lenguaje para describir gráficos vectoriales bidimensionales, tanto estáticos como animados (estos últimos con ayuda de *SMIL*), en *XML*.

WebCGM: es un formato de imagen de tipo vectorial, es decir, que almacena un conjunto de líneas y figuras. Esto es óptimo para diagramas complejos en que se requiere hacer *zoom* o visualizar por capas.

W3C: es un consorcio internacional que produce recomendaciones para la World Wide Web (WWW).

WWW: Red informática mundial.

YUI: serie de bibliotecas escritas en *JavaScript*, para la construcción de aplicaciones interactivas (*RIA*).

Zoom: agrandar, ampliar.

Anexos

***Anexo 1*:** Encuesta aplicada a una muestra de los especialistas de los proyectos Aplicativos_SIG y GeneSIG.

Instrumento para diagnóstico

Nombre de la persona: _____

Área donde labora: _____ **Rol que desempeña:** _____

Años de experiencia: _____

Terminologías

Tiempo de configuración: Intervalo de tiempo dedicado para configurar el fichero mapcache.xml.

Errores introducidos: Cantidad de errores introducidos durante el proceso de configuración del fichero mapcache.xml.

Nivel de complejidad: Nivel de dificultad al realizar el proceso de configuración del fichero mapcache.xml.

Desarrollo
Por favor, le pedimos que responda las siguientes preguntas, agradeciéndole de antemano su colaboración.

Las preguntas están dirigidas para dos momentos; el primero, antes de aplicar la propuesta y el segundo para verificar el criterio de los usuarios luego de aplicar la propuesta.

1. Indique qué tiempo se tarda para realizar la configuración de *MapCache* durante el desarrollo de nuevos aplicativos (Marque con una **X** la opción seleccionada):

Antes		Después	
5 a 10 minutos		5 a 10 minutos	
11 a 30 minutos		11 a 30 minutos	
31 a 60 minutos		31 a 60 minutos	
Más de 1 hora		Más de 1 hora	

2. Seleccione el rango de errores introducidos en la realización de la configuración de *MapCache* durante el desarrollo de nuevos aplicativos (Marque con una **X** la opción seleccionada):

Antes		**Después**	
0		0	
1 - 10		1 - 10	
11 - 20		11 - 20	
Más de 20		Más de 20	

3. ¿Cómo considera el nivel de complejidad al realizar la configuración de un fichero de *MapCache* durante el desarrollo de nuevos aplicativos? (Marque con una **X** la opción seleccionada):

Antes		**Después**	
Alto		Alto	
Medio		Medio	
Bajo		Bajo	

***Anexo 2*:** Listado de los especialistas de los proyectos Aplicativos_SIG y GeneSIG que conformaron la población del pre-experimento, los señalados de color gris son los 7 especialistas seleccionados como muestra para aplicar el instrumento de diagnóstico.

No	Nombre y Apellidos	Rol que desempeña	Proyecto al que pertenece	Años de experiencia
1	Alain León Companioni	Jefe de proyecto	Aplicativos_SIG	5
2	Alejandro O. Hernández Cebrián	Diseñador de base de datos	Aplicativos_SIG	6
3	Aliana López Costa	Diseñador de base de datos	Aplicativos_SIG	5
4	Ariel Labrada Delgado	Programador	GeneSIG	3
5	Daniel Echeverría González	Cartógrafo	Aplicativos_SIG	18
6	Grethell Castillo Reyes	Jefe de proyecto	GeneSIG	3
7	Joel Macías Roque	Programador	Aplicativos_SIG	5
8	Laritza Asán Caballero	Diseñadora de interfaz	Aplicativos_SIG	2
9	Leiny Ruíz Cervera	Analista	GeneSIG	3
10	Listley Castell Espinosa	Analista	Aplicativos_SIG	3

11	Miguel Milán Isaac	Programador	Aplicativos_SIG	2
12	Miosotis Aida Troche	Administrador de configuración	Aplicativos_SIG	2
13	Sandor Escobar Ruíz	Arquitecto	Aplicativos_SIG	3
14	Yampier Medida Tarancón	Analista	Aplicativos_SIG	6
15	Yordan Hernández Hernández	Programador	GeneSIG	2
16	Vladimir López Salvador	Programador	GeneSIG	1
17	Yoan Mandina Verdecia	Programador	GeneSIG	1
18	Rafael Enrique Naranjo Leonard	Programador	GeneSIG	1
19	Fidelky Alcantara Martínez	Programador	Aplicativos_SIG	1
20	Yoan Díaz Cubas	Programador	GeneSIG	1
21	Celia Torres Reyes	Analista	GeneSIG	1
22	Arletty Silvera Boffill	Programador	GeneSIG	1
23	Camilo Madariaga Dozaret	Programador	GeneSIG	1