Universidad de las Ciencias Informáticas

Facultad 6

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Título: "Módulo *raster* para la plataforma GeneSIG"

Autor: Ana María Quesada Esquijarosa

Tutor(es): MSc. Daniel Echevarría González

Ing. Alejandro Orgelio Hernández Cebrian

La Habana, junio de 2015

"Año 57 de la Revolución

"No se vive celebrando victorias, sino superando derrotas."

Ernesto "Che" Guevara

DECLARACIÓN DE AUTORÍA

Declaro ser la legítima autora del trabajo titulado: “Módulo *raster* para la plataforma GeneSIG”, y reconozco a la Universidad de las Ciencias Informáticas (UCI) 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

Ana María Quesada Esquijarosa

_____

Firma del Tutor

MSc. Daniel Echevarría González

_____

Firma del Tutor

Ing. Alejandro Orgelio Hernández Cebrian

Síntesis del Tutor

Nombre y Apellidos: MSc. Daniel Echevarría González.

Correo electrónico: danielec@uci.cu.

Año de graduado: 1995.

Profesión: Licenciatura en Geografía

Breve descripción: Graduado en la de la Facultad de Geografía de la Universidad de La Habana, Master en Ciencias en la Facultad de Geografía de la U/H en el año 1998. Profesor Asistente en el año 2012. Actualmente se desempeña como especialista de la LPS Aplicativos SIG.

Síntesis del Tutor

Nombre y Apellidos: Alejandro Orgelio Hernández Cebrian.

Correo electrónico: aohernandez@uci.cu.

Año de graduado: 2009.

Profesión: Ingeniero en Ciencias Informáticas.

Breve descripción: Graduado en la Universidad de las Ciencias Informáticas. Actualmente Líder del grupo de Bases de Datos Espaciales de la LPS Aplicativos SIG.

Síntesis del Autor

Nombre y Apellidos: Ana María Quesada Esquijarosa

Correo electrónico: amquesada@estudiantes.uci.cu.

Dedicatoria

A mi padre, el hombre más grande que he conocido en mi vida, aunque ya no esté entre nosotros. Para ti el resultado de este trabajo y todo mi amor.

Agradecimientos

A mi padre, por su ejemplo, por ser mi guía y darme el aliento para seguir adelante durante estos cinco años sin él, a pesar de los momentos duros que me ha tocado vivir.

A mi madre, por ser la mejor madre del mundo, por su apoyo, su cariño y su amor, por estar siempre ahí para mí a pesar de la distancia.

A mi hermano, por estar ahí para mí y ayudarme siempre.

A Bide, por criarme, quererme y consentirme durante todos estos años como a una hija.

Al resto de mi familia por creer en mí y apoyarme durante mi carrera.

A mis niñas, Yai, Mayi y Danaysi, mil gracias por su amistad, por ser como hermanas para mí durante estos años, por los momentos compartidos, por el apoyo incondicional.

A mis amigos del alma, Luisma, Pako, Luis Felipe, gracias por sus consejos, por cuidarme y protegerme siempre.

A mi novio Dayron, papito gracias por tu amor y por soportarme durante esta etapa tan difícil que fue el proceso de tesis, sé que para ambos fue un reto.

A los que también se convirtieron en parte de mi familia en esta universidad, Yany, Mayre, Nesto.

A aquellos que para mí fueron más que compañeros de grupo y a los que me llevaré siempre en mi corazón.

A mi familia de la FEU, Lio, Enier, Bernardo, Osviel, Luis Enrique, Guille, Lijandy, Yaisel, gracias por convertirse en grandes amigos y por las experiencias vividas.

A mis profesores durante estos 5 años, que han sido los encargados de formar a la persona que soy hoy, no solo en el plano académico, sino también en el personal, ha sido un honor ser una más de sus estudiantes.

A mis tutores por su apoyo durante este proceso tan duro y complicado para mí.

A todos los que aportaron su granito de arena en el desarrollo de esta investigación.

Al tribunal y al oponente por sus consejos y recomendaciones en aras de lograr un trabajo con la calidad requerida.

A todos ustedes, muchas gracias.

Resumen

La plataforma GeneSIG, utilizada como base para el desarrollo de Sistemas de Información Geográfica con perfiles más específicos, actualmente no cuenta con un modelo de datos almacenado en bases de datos para la información en formato *raster*. Para dar solución a este problema, mediante la presente investigación se implementó un módulo para la plataforma que garantiza el almacenamiento de información en formato *raster* en una base de datos espacial, su edición, georreferenciación y exportación como imagen. El proceso de desarrollo estuvo guiado por la metodología AUP, y con el objetivo de evitar incompatibilidades entre el módulo y la plataforma se mantuvo una arquitectura orientada a objetos y basada en componentes. La implementación de dicho componente se llevó a cabo en un ambiente web, utilizando tecnologías libres. Las pruebas realizadas al módulo arrojaron resultados satisfactorios, garantizando su correcto funcionamiento.

Palabras claves: almacenamiento, base de datos, edición, exportación, formato *raster,* GeneSIG, georreferenciación, Sistema de Información Geográfica.

Abstract

GeneSIG platform, which is used as a base for developing Geographic Information Systems with more specific profiles, doesn't have at the moment any stored data model on databases for handling information in raster format. With the goal of solving this problem, through the current investigation it was implemented a module for the platform that permits storing raster information into a spatial database, as well as editing it, georeference it and export it as an image. The development process was guided by the AUP methodology, and in order to avoid incompatibilities between the module and the platform an object-oriented and component-based architecture was kept. The component was implemented on a web environment using free technologies. The resulting module was widely tested obtaining satisfactory feedback which guarantees its correct functioning.

Key words: storing, database, edit, export, raster format, GeneSIG, georeference, Geographic Information System.

Índice

Índice de tablas

Índice de figuras

Introducción

La demanda de información en la sociedad actual ha crecido exponencialmente, al igual que la necesidad de innovación para resolver innumerables problemáticas. Esto ha incidido de forma directa en el desarrollo de nuevas tecnologías, incluyendo a los Sistemas de Información Geográfica (SIG), los cuales brindan un conjunto de facilidades aplicables a varias disciplinas de la investigación científica. La Open Geospatial Consortium (OGC) define un SIG como *"un sistema informático para capturar, almacenar, verificar, integrar, manipular, analizar y mostrar los datos relacionados con las posiciones en la superficie de la Tierra"* (OGC, 2014). Los SIG brindan gran cantidad de información para el apoyo a la toma de decisiones en distintas ramas de la actividad humana. Son empleados en la gestión de recursos naturales, la gestión de riesgos, la ecología, los negocios y la mercadotecnia, las ciencias sociales, la planificación, la actividad militar, entre otros sectores que se centran en el manejo de información geoespacial.

La información manipulada se estructura de dos formas, mediante el uso de vectores y mediante el uso de matrices. El primer caso se refiere a un SIG vectorial y el segundo a un SIG *raster*. Según Javier Domínguez, *"un sistema vectorial es aquel en el que el territorio se representa a partir de vectores, éstos se localizan en el espacio mediante pares de coordenadas coincidentes con su origen y destino"* (Domínguez, 2000). Además define un SIG *raster* como *"aquel que realiza sus cálculos a través de una estructura matricial, en la que cada celda o píxel tiene un valor y una localización determinadas"* (Domínguez, 2000).

En Cuba, diversas instituciones tienen entre sus proyectos la creación de SIG; la Universidad de las Ciencias Informáticas (UCI) es un ejemplo de ello. En la facultad 6 de dicha institución se encuentra el centro de desarrollo de software Geoinformática y Señales Digitales (GEYSED), y como parte de este la LPS[1] Aplicativos SIG cuenta con una aplicación informática que es utilizada como base para el desarrollo de SIG con perfiles más específicos, la plataforma GeneSIG.

Esta plataforma es un sistema informático que pone a disposición de desarrolladores y usuarios un sistema orientado a la web, de código abierto y tecnologías libres, para la creación de SIG. Su arquitectura es basada en componentes, lo que permite adaptar sus funcionalidades a cualquier negocio mediante la

[1] Línea de Productos de Software

reutilización de estos. GeneSIG está implementada usando los lenguajes de programación PHP[2] y JavaScript basándose sobre el *framework* CartoWeb. Además hace uso de ExtJS, AJAX[3] y CSS[4].

Dentro de las funcionalidades que brinda la plataforma GeneSIG se encuentran la recuperación y representación de datos almacenados en bases de datos espaciales, y la interacción plena del usuario con los mapas. En la actualidad la generalidad de la información recuperada desde bases de datos espaciales es de tipo vectorial, los objetos manipulados están conformados por polígonos, líneas y puntos como componentes bases. De igual forma, la mayoría de los SIG desarrollados por la LPS Aplicativos SIG, usando como base a GeneSIG, son de tipo vectorial. No obstante existen necesidades concretas, como la superposición de capas vectoriales y capas *raster* de determinadas regiones de un mapa, que requieren la carga de información rasterizada por lo que se ha desarrollado un procedimiento específico para este fin.

Este procedimiento consiste en la carga de una imagen guardada físicamente en uno de los directorios del sistema, lo cual no garantiza realizar el almacenamiento de información en formato *raster* hacia una base de datos espacial. Por otro lado se desechan importantes potencialidades surgidas de tener un modelo *raster* en una base de datos como es la posibilidad de tener la información geográfica de manera distribuida. De igual forma existen muchas operaciones que se realizan sobre la información geográfica, entre ellas el cálculo de distancias entre dos puntos y la conversión de las proyecciones, que no se pueden explotar. Muchos sistemas gestores de bases de datos implementan estas funcionalidades por lo que distribuyendo la información sobre bases de datos se pueden aprovechar, permitiendo un aumento del rendimiento y la robustez de las aplicaciones.

El procedimiento empleado no favorece un análisis preciso de la información geográfica manejada. Por ejemplo: suponiendo que existan capas vectoriales, dígase puntos, sobre una capa con información en formato *raster* que represente un núcleo urbano, resulta muy engorroso e impreciso hacer un análisis geográfico entre los objetos de la capa vectorial y la imagen que forma la base del mapa.

---

[2] *Hypertext Preprocessor* (PHP, por sus siglas en inglés, Lenguaje Procesador de Hipertextos)

[3] *Asynchronous JavaScript And XML* (AJAX, por sus siglas en inglés, JavaScript Asíncrono y XML)

[4] *Cascading Style Sheets* (CSS, por sus siglas en inglés, Hoja de estilo en cascada)

Existen tecnologías que permiten el análisis sobre modelos de datos almacenados en bases de datos. El gestor PostgreSQL sirve de soporte para datos espaciales en un SIG mediante su extensión espacial PostGIS, que es la utilizada en GeneSIG para el manejo de información vectorial. Esta extensión define un conjunto de funciones implementadas que son básicas para el análisis espacial dentro de la plataforma. Al no contar la información rasterizada con un modelo almacenado en base de datos resulta imposible de explotar en términos de desarrollo los beneficios de esta tecnología. Esta situación trae consigo malos resultados en la toma de decisiones donde se empleen los SIG como herramientas de gestión.

A partir de la problemática descrita anteriormente se deriva entonces el siguiente **problema de investigación**: El procedimiento actual para vincular los datos en formato *raster* con la plataforma GeneSIG no garantiza el almacenamiento de información en formato *raster* en una base de datos espacial, así como su edición, georreferenciación y exportación.

De acuerdo al problema planteado se define como **objeto de estudio** el tratamiento de información geográfica en la plataforma GeneSIG, delimitando el **campo de acción** hacia el tratamiento de información rasterizada en la plataforma GeneSIG. El **objetivo general** es desarrollar un módulo para la plataforma GeneSIG que facilite el tratamiento de información en formato *raster* a partir de su almacenamiento en una base de datos espacial, así como su edición, georreferenciación y exportación.

Para dirigir correctamente el proceso investigativo se definen las siguientes **preguntas de investigación**:

- ¿Cuál es la fundamentación teórica existente sobre el manejo de información en formato *raster* en los Sistemas de Información Geográfica?
- ¿Cómo diseñar y modelar de forma flexible un módulo para la plataforma GeneSIG que permita el tratamiento de información en formato *raster* a partir del almacenamiento, edición, georreferenciación y exportación de la misma?
- ¿Cómo implementar el módulo de gestión de información en formato *raster* garantizando su integración final con la plataforma GeneSIG?
- ¿El módulo garantiza el almacenamiento, edición, georreferenciación y exportación de la información en formato *raster* en la plataforma GeneSIG?

Para dar cumplimiento al objetivo general se proponen las siguientes **tareas investigativas**:

1. Elaboración de un estudio del arte de sistemas que posibiliten el trabajo con información en formato *raster* para definir características y tendencias del manejo de esta información.

2. Caracterización del modelo de almacenamiento de datos en formato *raster* para su despliegue en un servidor de base de datos.
3. Fundamentación de la metodología de software, las herramientas y tecnologías a utilizar en el proceso de desarrollo.
4. Modelado de los requisitos funcionales y no funcionales de la propuesta de solución para definir sus características.
5. Diseño de la propuesta de solución para guiar el proceso de implementación.
6. Implementación de la propuesta de solución para dar cumplimiento al objetivo planteado.
7. Validación de la propuesta de solución a través de pruebas para demostrar su correcto funcionamiento.

Para darle cumplimiento a las tareas anteriores se utilizarán los siguientes **métodos de la investigación científica**:

Métodos teóricos:

Se basan en la utilización del pensamiento en sus funciones de deducción, análisis y síntesis (Barchini, 2005). En la presente investigación los métodos teóricos a utilizar son:

- Método analítico - sintético: Es utilizado para consultar la bibliografía especializada en cuanto al manejo de información en formato *raster* e identificar elementos claves que contribuyan a la solución del problema científico planteado, permite sintetizar conceptos que ayudarán a comprender la solución del problema.

- Método sistémico: Utilizado en el estudio de la plataforma GeneSIG para comprender su estructura, forma de trabajo y complejidad, con el objetivo de definir la forma de implementación que se debe llevar a cabo en el desarrollo del módulo propuesto.

Métodos empíricos:

Se aproximan al conocimiento del objeto mediante sus conocimientos directos y el uso de la experiencia (Barchini, 2005). En la presente investigación el método empírico a utilizar es:

- Estudio de caso: Se determina su uso pues se requiere la utilización de la plataforma GeneSIG como base para la implementación del módulo propuesto, y la posterior realización de pruebas para determinar su efectividad.

Se espera obtener como **posible resultado**:

1. Módulo *raster* para la plataforma GeneSIG que va a contar con una serie de interfaces donde el usuario podrá escoger la base de datos donde desea guardar su imagen en formato *raster*, editar dicha imagen, georreferenciarla y exportarla en los formatos PNG o JPEG.

2. Documentación referente al proceso de desarrollo del módulo para el tratamiento de información en formato *raster* a implementar.

El presente trabajo de diploma está estructurado en 3 capítulos:

En el primer capítulo, titulado **"Fundamentos teóricos sobre el procesamiento de datos en formato *raster*"**, se realiza un análisis detallado de la situación problemática; es definido el marco conceptual de la investigación y se estudian otras soluciones existentes asociadas al dominio de la problemática que pueden dar respuesta al problema a resolver. Se analizan las tecnologías y herramientas que serán empleadas en la elaboración del módulo que dará solución al problema planteado. Seguidamente en el capítulo 2 que lleva por título **"Descripción del módulo *raster* para la plataforma GeneSIG"** se realiza el modelado del dominio del problema, se plantean los requisitos funcionales y no funcionales, así como los artefactos generados en el modelo del sistema. Concluyendo la investigación en el capítulo 3, titulado **"Implementación del módulo *raster* para la plataforma GeneSIG"**, que está centrado en la definición de las clases de diseño, sus relaciones y la estructuración de los diagramas de clases del diseño. Se establece el modelo de datos, el de despliegue y el de implementación, concluyendo con la validación del módulo a partir de las pruebas necesarias para demostrar que la solución es correcta.

Capítulo 1. Fundamentos teóricos sobre el procesamiento de datos en formato *raster*

En este capítulo se realiza un análisis detallado de la situación problemática; es definido el marco conceptual de la investigación y se estudian otras soluciones existentes asociadas al dominio de la problemática que pueden dar respuesta al problema a resolver. También se analizan las tecnologías y herramientas que serán empleadas en la elaboración del módulo que dará solución a la problemática planteada.

1.1 Sistema de Información Geográfica

Existen diversos criterios en cuanto a la definición de un SIG. Víctor Olaya lo define como *"un sistema que integra tecnología informática, personas e información geográfica, y cuya principal función es capturar, analizar, almacenar, editar y representar datos georreferenciados"* (Olaya, 2010). Javier Domínguez plantea que *"un SIG se puede definir como aquel método o técnica de tratamiento de la información geográfica que nos permite combinar eficazmente información básica para obtener información derivada*" (Domínguez, 2000). La OGC lo define como *"un sistema informático para capturar, almacenar, verificar, integrar, manipular, analizar y mostrar los datos relacionados con las posiciones en la superficie de la Tierra"* (OGC, 2014). A partir del análisis de los conceptos brindados se asume el que brinda la OGC pues en este se abordan la mayoría de los procesos que lleva a cabo una herramienta de este tipo.

Para el estudio práctico de los SIG en (Olaya, 2010) se plantea un esquema clásico de cinco componentes, los cuales son:

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

Las relaciones que se establecen entre estos componentes también forman parte de los SIG, siendo su correcta organización fundamental para el óptimo funcionamiento de estos sistemas. Los datos son necesarios para que el resto de los componentes puedan ejercer su papel en el sistema, y deben ser los

adecuados para satisfacer las necesidades de los usuarios y estos obtengan los resultados deseados de la mejor manera.

Los SIG permiten realizar análisis complejos de la realidad espacial. Son herramientas muy versátiles y no existe prácticamente ningún ámbito de trabajo en el que no puedan, de un modo u otro, ser de utilidad. Un SIG puede emplearse como herramienta modeladora, ya que con él pueden modelarse realidades geográficas complejas al hacer uso de las funciones de análisis espacial implementadas en el mismo. La toma de decisiones representa una utilización particular de esta capacidad de modelado, donde el elemento clave es la interpretación de los resultados. Otra forma de empleo es como herramienta para la difusión de información geográfica, pues es capaz de exponer datos geográficos a un público más amplio y hacerlo de la mejor manera.

Por la gran cantidad de áreas de utilización de los SIG, la creación de sistemas de este tipo ha pasado a ser el objetivo de un vasto número de empresas. Entre las principales a nivel internacional se encuentran ESRI, Intergraph, MapInfo, Bentley Systems, Autodesk y Smallworld. En el ámbito nacional, la UCI como productora de software y siguiendo las especificaciones de la OGC ha desarrollado la plataforma GeneSIG.

1.2 Plataforma GeneSIG

Una plataforma de desarrollo de software es *"una pieza importante de software, como un entorno operativo, en la que distintos programas de aplicación más pequeños pueden ser diseñados para funcionar"* (InterActiveCorp, 2014). Un ejemplo lo constituye la plataforma GeneSIG, producto encaminado a realizar la representación y análisis geoespacial de información geográfica y su estructura arquitectónica permite personalizar sus funcionalidades a cualquier negocio que lo requiera a través de la reutilización de sus componentes. También puede ser usada como una aplicación SIG, ya que permite consultar, consumir y manipular bases de datos espaciales de diversos formatos y orígenes, como son: datos vectoriales, datos de Sistemas de Posicionamiento Global (GPS por sus siglas en inglés), Servicios de Mapas Web (WMS por sus siglas en inglés) y Servicios de Geometrías (WFS por sus siglas en inglés) (Parodis, y otros, 2012).

Cumple técnicamente con las especificaciones OpenGIS que establece el OGC que garantizan la interoperabilidad global entre los SIG y en consecuencia con la política de migración a software libre y de soberanía tecnológica que impulsa Cuba (Varen, 2012). Su arquitectura está basada en componentes, lo

que permite adaptar sus funcionalidades a cualquier negocio mediante la reutilización de estos. GeneSIG está implementada usando los lenguajes de programación PHP y JavaScript, está basada en el framework CartoWeb, siendo este la columna vertebral de la plataforma, y hace uso de ExtJS, AJAX y CSS.

Algunas de las funcionalidades con las que cuenta la plataforma son:

- Recuperación y representación de datos almacenados en bases de datos espaciales: Permite realizar consultas espaciales sobre la base de datos y mostrar los resultados en un mapa con un formato definido.
- Interacción plena del usuario con los mapas: Permite las siguientes funciones sobre el mapa:
  - Navegación (hacia delante y hacia detrás).
  - Identificación de objetos geográficos.
  - Mover centro del mapa.
  - Navegación a través del mapa de referencia.
- Análisis de rutas: Generación de caminos mínimos sobre grafos de rutas.
- Medición de distancias y superficies: Se definen las unidades de medida para estas operaciones y luego es dibujada sobre el mapa la región a la cual se le aplicará.
- Localización de objetos geográficos: Permite la búsqueda y localización de objetos a partir de las coordenadas que los definen o de una jerarquía territorial.
- Generación de mapas temáticos: El principal análisis geográfico es realizado a través de la generación de mapas temáticos partiendo de un criterio o variable. Este proceso puede realizarse por colores y por gráficos de barra y pastel.

Como se mencionó con anterioridad, GeneSIG emplea la extensión PostGIS para el manejo de información geográfica. Esta extensión define un conjunto de funciones implementadas que son básicas para el análisis espacial dentro de la plataforma incluyendo algunas de las ya mencionadas. En la actualidad es utilizada solamente para el manejo de datos vectoriales, pues la plataforma no tiene implementada una forma de recuperar y visualizar información en formato *raster* de una base de datos espacial.

1.3 Información en formato *raster*

En su forma más simple, el formato *raster* se representa a partir una matriz de celdas (o píxeles) organizadas en filas y columnas (o una cuadrícula) en la que cada celda contiene un valor que representa

información, como la temperatura. La información en formato *raster* la constituyen fotografías aéreas digitales, imágenes de satélite e imágenes digitales.

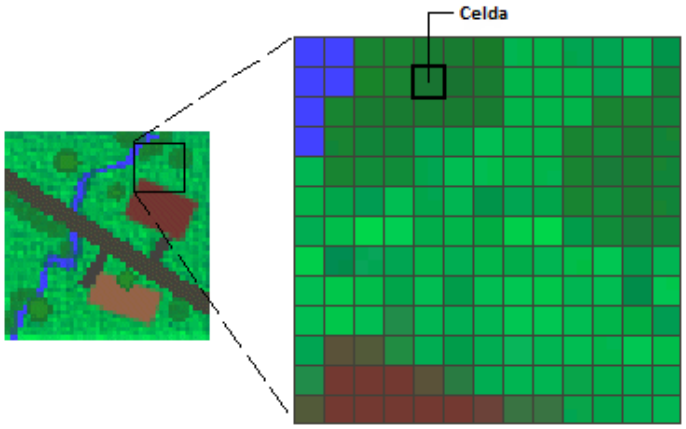

Figura 1. Representación del formato *raster*

Los datos almacenados en formato *raster* representan fenómenos del mundo real:

- Los datos temáticos representan entidades como, por ejemplo, el uso del suelo o los datos del suelo.
- Los datos continuos representan fenómenos como la temperatura o elevación, o datos espectrales, entre ellos imágenes de satélite y fotografías aéreas.

Los datos temáticos y continuos se pueden visualizar en el mapa en forma de capas de datos junto con otros datos geográficos, pero a menudo se utilizan como datos de origen para el análisis espacial. Las imágenes en formato *raster* suelen utilizarse como atributos en tablas: pueden visualizarse con datos geográficos y se utilizan para transmitir información adicional acerca de las entidades geográficas de mapas.

Si bien la estructura de datos en formato *raster* es simple, es excepcionalmente útil para una amplia variedad de aplicaciones. En un SIG, los usos de los datos en formato *raster* se pueden dividir en cuatro categorías principales, en forma de mapa base, de mapa de superficie, de mapas temáticos y en forma de atributos de una entidad (Esri, 2014).

- En forma de mapas base

Un uso común de los datos en formato *raster* en un SIG es en forma de visualización de fondo para otras capas de entidades. Por ejemplo, las ortofotografías que se visualizan debajo de otras capas ofrecen al usuario la garantía de que las capas del mapa se alinean espacialmente y representan tanto objetos

reales como información adicional. Las dos fuentes principales de mapas base en formato *raster* son las ortofotografías de fotografías aéreas e imágenes de satélite. A continuación se muestra una imagen en formato *raster* utilizada como mapa base en el producto SIGUCI para los datos de los edificios en la UCI.

Figura 2. Datos en formato *raster* en forma de mapa base

- En forma de mapas de superficie

Los datos en formato *raster* son apropiados para representar cambios continuos en un entorno (superficie). Ofrecen un método efectivo para almacenar la continuidad en forma de superficie. También proporcionan una representación de superficies con espacios regulares. Los valores de elevación que se miden desde la superficie de la Tierra son la aplicación más común de los mapas de superficie, pero otros valores, como las precipitaciones, la temperatura, la concentración y la densidad de población, también pueden definir superficies que se pueden analizar espacialmente. En la siguiente imagen se visualiza un mapa del Instituto Goddard para Estudios Espaciales (Goddard Institute for Space Studies o GISS, por su acrónimo en inglés), de la NASA, donde se muestra la evolución de anomalías en las temperaturas superficiales globales desde 1880 hasta 2012. En rojo las temperaturas más altas en el Ártico, península Antártica y los continentes.

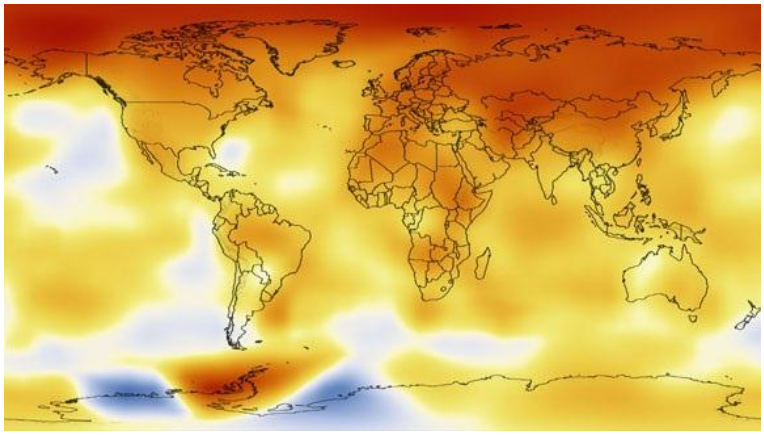

Figura 3. Datos en formato *raster* en forma de mapa de superficie (Phillips, 2010)

- En forma de mapas temáticos

Las imágenes que representan datos temáticos se pueden derivar al analizar otros datos. Una aplicación de análisis común consiste en clasificar una imagen de satélite por categorías de cobertura de suelo. Básicamente, esta actividad agrupa los valores de datos multiespectrales en clases (como tipo de vegetación) y asigna un valor categórico. A continuación se muestra un ejemplo de datos en formato *raster* clasificados en el que se representa el uso del suelo.

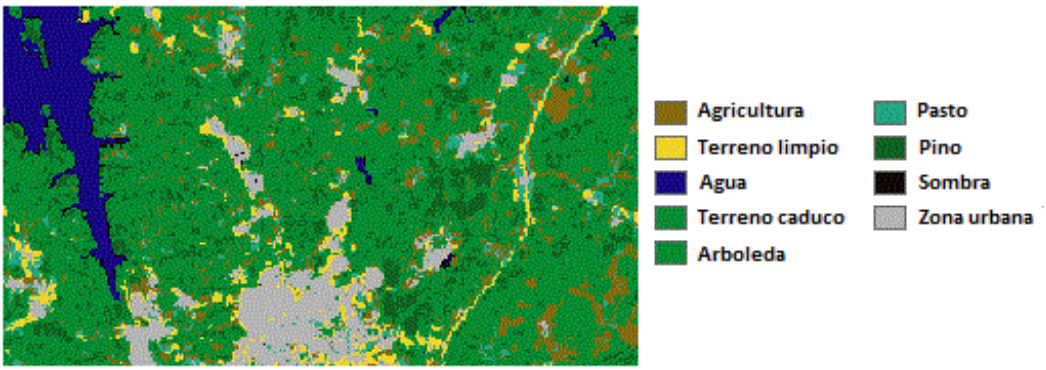

Figura 4. Datos en formato *raster* en forma de mapas temáticos

- En forma de atributos de una entidad

Los datos en formato *raster* utilizados como atributos de una entidad pueden ser fotografías digitales relacionadas con un objeto o ubicación geográfica. A continuación se muestra una imagen digital de un árbol de gran tamaño, que podría utilizarse como atributo de una capa de paisaje que puede mantener una ciudad.

Figura 5. Datos en formato *raster* en forma de atributo de una entidad

Entre las principales ventajas de almacenar los datos en formato *raster* se encuentran:

- Estructura de datos simple: matriz de celdas con valores que representan una coordenada y que, en ocasiones, se encuentra vinculada a una tabla de atributos.
- Formato potente para análisis espacial y estadístico avanzado.
- Capacidad de representar superficies continuas y llevar a cabo análisis de superficie.
- Capacidad de almacenar puntos, líneas, polígonos y superficies de manera uniforme.
- Capacidad de llevar a cabo superposiciones rápidas con conjuntos de datos complejos.

Como se mencionaba con anterioridad el manejo de información en formato *raster* en GeneSIG se realiza a partir de la carga de una imagen guardada físicamente en uno de los directorios de la plataforma, por lo cual no se garantiza realizar el almacenamiento de datos en formato *raster* hacia una base de datos espacial. Esto impide explotar las funcionalidades que posee la plataforma a partir de su trabajo con PostGIS, así como la realización de diversas operaciones que se realizan sobre la información geográfica, entre ellas el cálculo de distancias y la conversión de las proyecciones.

1.4 Análisis de sistemas que manejan información *raster*

Existen disímiles sistemas que permiten el tratamiento de información en formato *raster*, cada uno de ellos con sus características y peculiaridades. Para la presente investigación se tomarán en cuenta las características que puedan ser útiles en el desarrollo del módulo que se espera obtener. A continuación se describen tres de estos sistemas y sus características principales; dos de ellos propietarios, ArcGIS y MapInfo, considerados actualmente como las herramientas más potentes para el manejo de información

geográfica; y el tercero de código abierto, Quantum Gis (QGIS). Se seleccionaron estos sistemas para su estudio porque en el caso de ArcGIS es uno de los más difundidos a nivel mundial y es actualmente el que más funcionalidades ofrece a los usuarios. MapInfo Professional y QGIS por su parte son utilizados actualmente en Aplicativos SIG para el análisis de información en formato *raster*, por lo que los especialistas están familiarizados con su funcionamiento.

1.4.1. ArcGIS

ArcGIS (Esri, 2014) es una plataforma de información que permite crear, analizar, almacenar y difundir datos, modelos, mapas y globos en tres dimensiones (3D), poniéndolos a disposición de todos los usuarios según las necesidades de la organización. Como sistema de información es accesible desde clientes de escritorio, navegadores web, y terminales móviles que se conectan a servidores de departamentos, corporativos o con arquitecturas de computación en la nube.

Consta de tres componentes:

- ArcGIS Desktop - Conjunto de aplicaciones SIG integrados.
- ArcSDE Gateway - Interfaz para el manejo de bases de datos geográficas multiusuarios.
- ArcIMS - Aplicación web para la distribución de datos y servicios.

Tiene un modelo de datos geográficos de alto nivel para la representación de información espacial como polígonos, líneas, puntos y *raster*. Permite la implementación del modelo de datos basado en archivos o en una base de datos espacial. El modelo de datos basado en archivos permite formatos como *coverages*, *shapefiles*, imágenes, mallas y redes triangulares irregulares, estas dos últimas son las que proporcionan el soporte de la información espacial en formato *raster*.

ArcSDE es una aplicación de bases de datos relacionados. Esta aplicación permite el manejo de la información geográfica en el manejador de base de datos que prefiera y funciona como el servidor para el acceso de información geográfica desde ArcGIS Desktop y otras aplicaciones. Funciona como el puente entre el resto de las aplicaciones ArcGIS y su base de datos relacional.

1.4.2. MapInfo Professional

MapInfo Professional (Pitney Bowes, 2014) es un sistema de información geográfica de escritorio producido por Pitney Bowes Software (anteriormente MapInfo Corporation) que se utiliza para la

cartografía y análisis de localización. Permite a los usuarios visualizar, analizar, editar e interpretar datos de salida para revelar relaciones, patrones y tendencias.

Entre las principales características de MapInfo se encuentran el soporte para imágenes en formato *raster*, pues permite el uso de imágenes de mapa de bits, tales como mapas escaneados de papel, imágenes de satélite y fotografías, para proporcionar capas de contenidos detallados para sus mapas. La estratificación es una de sus características más usadas, se basa en su capacidad para combinar datos de fuentes muy diferentes, incluso con diferentes formatos y proyecciones, en la misma ventana del mapa. Una vez combinados en la ventana del mapa, las relaciones que sólo existen geográficamente entre los diferentes conjuntos de datos se pueden visualizar y consultar; las capas pueden ser vectorial y *raster* juntos. A pesar de que permite el manejo y análisis de información en formato *raster*, no almacena estos datos en bases de datos espaciales.

1.4.3. QGIS

QGIS (QGIS, 2014) es un SIG de código abierto, multiplataforma, desarrollado en QT Toolkit y C++. Se publica bajo la GNU GPL[5]. Es un proyecto oficial de la Open Source Geospatial Foundation. Puede ver y superponer datos en formato vectorial y *raster* en diferentes proyecciones sin conversión a un formato interno o común. Los formatos admitidos incluyen tablas de PostgreSQL con capacidad espacial, archivos en formato *raster* e imágenes admitidas por la biblioteca GDAL[6], datos en formato vectorial, de bases de datos, entre otros. Es capaz de guardar capturas de pantalla como imágenes georreferenciadas. No permite el almacenamiento de datos en formato *raster* en una base de datos, el trabajo con estos lo hace a través de la carga de un archivo que puede ser de tipo ArcInfo Binary Grid, ArcInfo ASCII Grid, GeoTIFF, ERDAS IMAGINE, entre otros.

A partir del estudio de las herramientas se puede arribar a las siguientes conclusiones: MapInfo y QGIS no permiten el almacenamiento de datos en formato *raster* en una base de datos espacial. ArcGis si permite

[5] *GNU General Public License* (GNU GPL, por sus siglas en inglés, Licencia Pública General de GNU)

[6] *Geospatial Data Abstraction Library* (GDAL, por sus siglas en inglés). Biblioteca de software para la lectura y escritura de formatos de datos geoespaciales.

el almacenamiento de información en formato *raster*, pero la única forma de utilizarlo sería a partir del tercer módulo con que cuenta, el cual presenta la funcionalidad de generar mapas que son exportados como servicios a internet. GeneSIG pudiera consumir estos para ser visualizados, pero sería muy engorroso este procedimiento, pues habría que depender de dos herramientas, ArcGis para la generación del servicio y GeneSIG para consumirlo, además de una conexión permanente a internet. Por otra parte este sistema es distribuido bajo una licencia privativa por lo que no está acorde a las políticas de desarrollo de software definidas en la UCI.

Por las razones planteadas con anterioridad se llega a la conclusión de que ninguna de estas herramientas puede ser utilizada en la solución de la presente investigación, por lo que se decide desarrollar un módulo *raster* para la plataforma GeneSIG a partir del uso de tecnologías libres. Es importante mencionar que QGIS se utilizará como guía y ejemplo en el desarrollo de los procesos de georreferenciación y edición de la proyección. Esta herramienta permite georreferenciar una imagen en formato *raster*, a pesar de no estar almacenada en una base de datos espacial, así como asignarle una proyección y obtener la que posee, aspectos que se deben tener en cuenta en el desarrollo del módulo.

1.5 Herramientas y tecnologías a utilizar

Una vez definida la solución al problema antes planteado se hace necesario seleccionar las herramientas y tecnologías con las que será desarrollado el módulo *raster* para la plataforma GeneSIG. En el presente epígrafe se justifica la elección de las tecnologías y herramientas existentes actualmente que son necesarias y contribuyen al desarrollo de esta tarea. La solución estará desarrollada con herramientas y tecnologías libres, y sustentada sobre tecnología web, utilizando como plataforma base GeneSIG v1.5.

1.5.1. Metodología de desarrollo de software

El proceso de desarrollo de software se torna una tarea ardua en la mayoría de las ocasiones. Para lograr obtener un producto con la calidad requerida es necesario tener una guía que imponga cierta disciplina y funcione como un hilo conductor en el proceso de desarrollo. Con este objetivo han sido creadas las metodologías de desarrollo de software, las cuales proporcionan las guías para poder conocer todo el camino a recorrer desde antes de empezar la implementación, con lo cual se asegura la calidad del producto final, así como el cumplimiento en la entrega del mismo en un tiempo estipulado.

Se define AUP[7] como la metodología más adecuada para el desarrollo del módulo, esta se basa en la gestión de riesgos, proponiendo que aquellos componentes con alto riesgo tengan más prioridad que los demás y sean desarrollados en etapas tempranas del proyecto. Desarrolla prototipos ejecutables durante la fase de elaboración del producto, demostrando la validez de la arquitectura para los requisitos claves del producto y determinando los riesgos técnicos. En AUP se establecen cuatro fases, Concepción, Elaboración, Construcción y Transición, que transcurren de manera consecutiva y que acaban con hitos claros alcanzados. El proceso AUP establece un modelo más simple que el que aparece en la metodología RUP[8] por lo que reúne en una única disciplina las disciplinas de Modelado de Negocio, Requisitos y Análisis y Diseño. El resto de las disciplinas (Implementación, Pruebas, Despliegue, Gestión de Configuración, Gestión y Entorno) coinciden con las restantes de RUP.

Se eligió esta metodología por ser muy apropiada para guiar proyectos de una complejidad y volumen no muy altos y que necesiten una rápida implementación, los cuales son los aspectos fundamentales a tener en cuenta para el desarrollo del producto a obtener. AUP proporciona un desarrollo del software rápido y eficiente, con una generación de artefactos media que satisface los requerimientos para la construcción del módulo, a la vez que permite un ahorro de tiempo considerable. De igual forma se tiene en cuenta que es la metodología utilizada en la LPS Aplicativos SIG, por lo que se debe mantener en aras de garantizar que los especialistas comprendan los artefactos generados y el proceso seguido en el desarrollo.

Esta metodología cuenta con las siguientes fases, como se mencionó anteriormente:

1. Concepción: Identificación del alcance y dimensión del proyecto, propuesta de la arquitectura y del presupuesto del cliente.
2. Elaboración: Confirmación de la idoneidad de la arquitectura.
3. Construcción: Desarrollo incremental del sistema, siguiendo las prioridades funcionales de los implicados.
4. Transición: Validación y despliegue del sistema.

[7] *Agile Unified Process* (AUP, por sus siglas en inglés, Proceso Unificado Ágil)

[8] *Rational Unified Process* (RUP, por sus siglas en inglés, Proceso Racional Unificado)

Para el desarrollo de las fases anteriores las disciplinas de AUP a cumplir son las siguientes:

- Modelo: Entender el negocio de la organización, el dominio del problema que aborda el proyecto y definir una solución viable.
- Aplicación: Transformar el modelo en código ejecutable y realizar un nivel básico de las pruebas.
- Prueba: Realizar una evaluación objetiva para garantizar la calidad. Esto incluye encontrar defectos, validar que el sistema automatizado funciona según lo previsto y verificar que se cumplan los requisitos.
- Despliegue: Realizar un plan para la presentación del sistema y ejecutarlo para hacer que el sistema se encuentre a disposición de los usuarios finales.
- Gestión de Configuración: Realizar la gestión de acceso a artefactos de su proyecto. Esto incluye no sólo el seguimiento de las versiones del artefacto en el tiempo, sino también el control y la gestión de cambios para ellos.
- Gestión del Proyecto: Dirigir las actividades que se llevan a cabo en el proyecto. Esto incluye la gestión de los riesgos, la dirección de personas (la asignación de tareas, el seguimiento de los progresos, entre otros), y coordinar con las personas para garantizar que se entrega a tiempo y dentro del presupuesto.
- Ambiente: Apoyar el resto de los esfuerzos por garantizar que el proceso sea adecuado, la orientación (normas y directrices) y herramientas (hardware, software, entre otros) están disponibles para el equipo según lo necesiten.

1.5.2. Lenguaje de modelado

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

UML[9] es uno de los lenguajes de modelado de gran utilidad para el desarrollo, pues ofrece un modo estándar de visualizar, especificar, construir y documentar los artefactos de un sistema. Su objetivo es lograr una aplicación informática robusta, flexible y escalable (Jacobson, y otros, 2000).

Será utilizado en su versión 2.0 para la realización de los entregables que propone la metodología seleccionada. Los artefactos a entregar son el modelo del dominio y modelo de casos de uso para la definición y detalle de los requisitos funcionales. El modelo de diseño, incluyendo un modelo de despliegue, un modelo de objetos y un modelo de datos. Finalmente será presentado también un modelo de diagrama de componentes de implementación como parte del modelo de implementación.

1.5.3. Herramienta de modelado

Con el objetivo de apoyar y automatizar la metodología de software y el lenguaje de modelado se emplean las herramientas CASE[10]. Son sistemas que facilitan el diseño y la documentación de las actividades del proceso de desarrollo de un software.

Visual Paradigm v8.0, se considera una de las herramientas CASE más adecuada para realizar el proceso de modelado de un software, esta propiedad es básicamente la que constituyó un hecho determinante en su selección para ser utilizada en el modelado de la propuesta de solución. Posee una interfaz amigable y fácil de utilizar. El hecho de que es una herramienta UML que soporta todo el ciclo de desarrollo del software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue, es otra de las razones que se tuvo en cuenta para determinar que es la herramienta indicada para realizar el modelado del módulo.

1.5.4. Lenguaje de programación

PHP v5.3 es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma, rápido, con una gran biblioteca de funciones y mucha documentación. Un lenguaje del lado del servidor es

---

[9] *Unified Modeling Language* (UML, por sus siglas en inglés, Lenguaje Unificado de Modelado)

[10] *Computer Aided Software Engineering* (CASE, por sus siglas en inglés, Ingeniería de Software Asistida por Computadora) son diversas aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software

aquel que se ejecuta en el servidor web, justo antes de que se envíe la página a través de Internet al cliente. Las páginas que se ejecutan en el servidor pueden realizar accesos a bases de datos, conexiones en red y otras tareas para crear la página final que verá el cliente. El cliente solamente recibe una página con el código HTML resultante de la ejecución de PHP. Como la página resultante contiene únicamente código HTML, es compatible con todos los navegadores.

Una de sus grandes cualidades es su versatilidad al momento de escribir código, su sencillez en la sintaxis e inclusive su seguridad; es gratuito y fácil de aprender. Posee una gran variedad de funciones que pueden ser utilizadas para mejorar el rendimiento de los programas y es un lenguaje de uso muy común en la web.

1.5.5. Entorno de Desarrollo Integrado

NetBeans es un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) que permite diseñar aplicaciones de forma fácil con solo arrastrar objetos a la interfaz de un formulario. Es una plataforma pensada para escribir, compilar, depurar y ejecutar programas. NetBeans no solo permite el desarrollo de aplicaciones de escritorio, también permite el desarrollo de aplicaciones para la web. La programación en este IDE se realiza a través de componentes modulares o módulos. Las aplicaciones construidas a partir de módulos pueden ser extendidas ya que estos permiten ser desarrollados independientemente por otros desarrolladores de software, de ahí que sea una aplicación flexible/extensible.

Se selecciona NetBeans v7.4 como IDE de desarrollo porque permite realizar la instalación y la actualización de forma simple, posee características visuales para el desarrollo web y permite crear aplicaciones web con PHP 5.

1.5.6. Sistema Gestor de Bases de Datos (SGBD)

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

El gestor de base de datos PostgreSQL es el utilizado en la plataforma GeneSIG para el manejo de bases de datos espaciales. Permite el soporte para datos espaciales en un SIG mediante su extensión espacial PostGIS, conjuntamente con la gestión de objetos geográficos, aspecto fundamental dadas las características del módulo que se desea desarrollar. Tiene soporte nativo por parte de los lenguajes más

populares del medio, como PHP. En esta investigación se hará uso de PostgreSQL v9.3 pues es la que se relaciona con la versión de PostGIS necesaria para el tratamiento de información en formato *raster*.

PostGIS

Con PostGIS se pueden usar todos los objetos que aparecen en la especificación OpenGIS como puntos, líneas, polígonos, multilíneas, multipuntos, y colecciones geométricas. Los objetos SIG soportados por PostGIS son de características simples definidas por OpenGIS.

Se selecciona PostGIS v2.1 pues brinda soporte para datos en formato *raster* y permite realizar su análisis y procesamiento. Otras de sus características son la posibilidad de obtener un nuevo *raster* a partir de partes (o bandas) de otros y la conversión de una geometría PostGIS a una *raster*. Proporciona acceso tanto al *raster* completo como a bandas y píxeles de este.

1.5.7. Servidor web

Apache Server v2.4.7 es uno de los servidores web más populares a nivel internacional, se caracteriza generalmente por su gran robustez, flexibilidad, estabilidad y eficiencia. Para el desarrollo y posterior uso del módulo, Apache Server es el servidor web seleccionado por las múltiples ventajas que ofrece. Es modular, de carácter libre, compatible con múltiples sistemas operativos, como Linux y Windows, y es muy popular, por lo que es más fácil conseguir ayuda y soporte. Su condición de software libre hace que pueda adaptarse a diferentes entornos y necesidades, no necesita grandes recursos para funcionar y está respaldado por una comunidad de desarrollo amplia.

1.5.8. Servidor de mapas

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

entorno de desarrollo de código abierto para la creación de aplicaciones SIG en entornos de red con el fin de visualizar, consultar y analizar información geográfica mediante la tecnología IMS[11].

Se selecciona Mapserver v6.4 como servidor de mapas, pues de todas las tecnologías de código abierto que permiten la representación geoespacial, este es hasta el momento el más estable. Es una de las herramientas de código abierto más completas y eficientes para realizar aplicaciones. Es sumamente versátil, a pesar de ser originalmente orientada a trabajar con Linux, puede trabajar también con Mac OS X y Windows. Una de las características más destacables de MapServer es que es muy fácil de utilizar, sencillo de manipular y permite realizar análisis de manera rápida y eficiente.

1.5.9. Bibliotecas

ExtJS

ExtJS es una biblioteca de JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML[12] y DOM[13]. Permite crear aplicaciones complejas utilizando componentes predefinidos, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno. La ventana flotante que provee ExtJS es excelente por la forma en la que funciona, al moverla o redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido.

Se selecciona ExtJS v3.4, principalmente porque es la soportada por la versión a utilizar de la plataforma GeneSIG. Usar un motor de renderizado[14] como ExtJS permite tener además estos beneficios:

a. Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.

[11] *Internet Map Server* (IMS, por sus siglas en inglés, Servidor de mapas en internet)

[12] *Dynamic HTML* (DHTML , por sus siglas en inglés, HTML Dinámico)

[13] *Document Object Model* (DOM, por sus siglas en inglés, Modelo de Objetos del Documento)

[14] Término derivado del inglés *render* usado para referirse al proceso de generar una imagen desde un modelo.

b. Comunicación asíncrona. En este tipo de aplicación el motor de renderizado puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente lo note.
c. Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija qué información desea transmitir al servidor y viceversa, sin embargo la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico.

Geospatial Data Abstraction Library (GDAL)

GDAL es una biblioteca de acceso a datos para la lectura y escritura de una gran variedad de formatos de datos geoespaciales. Presenta un modelo abstracto de datos único para todos los formatos soportados. Viene con una gran variedad de utilidades de líneas de comandos para la traducción y procesamiento de datos, que son especialmente útiles para los usuarios finales, en contraste con los programadores que utilizan la biblioteca (Warmerdam, 2008). La biblioteca se divide en un medio *raster* y uno vectorial, cada uno con su propio modelo de datos y API[15]. Se selecciona GDAL v1.10 pues es ampliamente utilizado en el mundo geoespacial de código abierto incluyendo, pero no limitado, a paquetes como MapServer y QGIS. También se utiliza en diversos grados por varios productos de software propietario, incluyendo ArcGIS.

1.6 Conclusiones parciales

La manera actual de mostrar los datos en formato *raster* en la plataforma GeneSIG impide un análisis completo de la información geográfica visualizada, por lo que el módulo a desarrollar debe garantizar el almacenamiento de este tipo de datos en bases de datos espaciales. Con la investigación de las características y ventajas que proporcionan las herramientas y tecnologías para el desarrollo de software quedan sentadas las bases teóricas y tecnológicas sólidas, de gran ayuda en el diseño e implementación del módulo.

[15] *Application Programming Interface* (API, por sus siglas en inglés, Interfaz de Programación de Aplicaciones) es el conjunto de subrutinas, funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.

Capítulo 2. Descripción del módulo *raster* para la plataforma GeneSIG

El presente capítulo muestra el comienzo de la elaboración del módulo *raster* para la plataforma GeneSIG. Se describen, según indica la metodología AUP, los principales artefactos relacionados con el modelado del dominio y el levantamiento de requisitos. Se presenta específicamente el modelo del dominio y los principales conceptos asociados al mismo. Se plantean los requisitos funcionales y no funcionales con los que debe cumplir el módulo, así como los casos de uso y la descripción detallada de cada uno de ellos.

2.1 Modelo de dominio

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

Con el objetivo de representar los conceptos más significativos en el dominio del problema y partiendo de que no se tienen bien definidos los procesos del negocio, se realizará el modelado del dominio y se procederá a explicar cada uno de los conceptos que forman parte del mismo.

2.1.1. Diagrama de clases del dominio

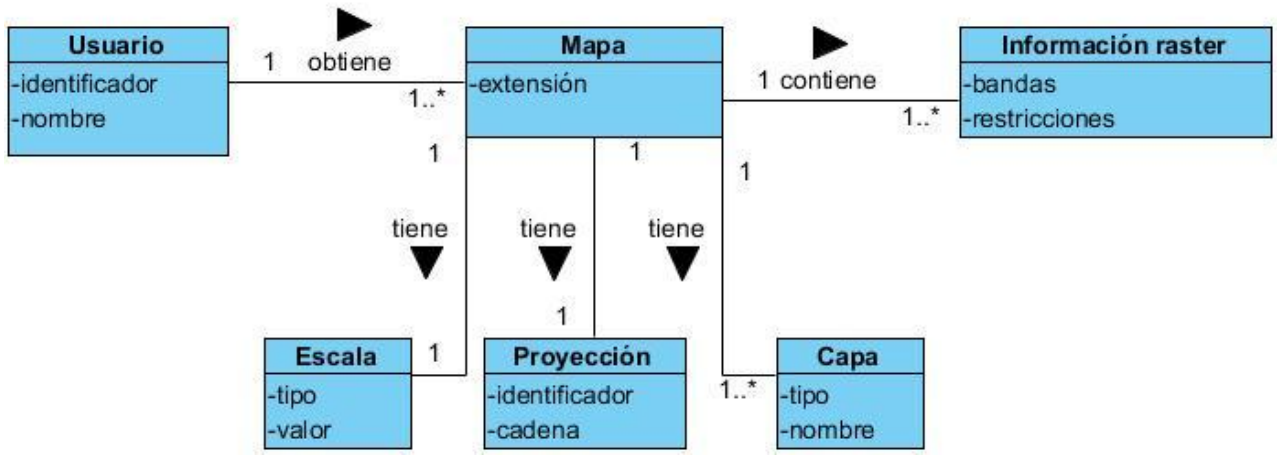

Figura 6. Diagrama de clases del dominio

2.1.2. Descripción del modelo de dominio

El usuario realiza la búsqueda de determinada ubicación geográfica, obteniendo así una representación de un mapa que contiene información en formato *raster*. Los mapas están compuestos por varios atributos como escala, proyección y capas, que permiten un mejor entendimiento y análisis de los mismos.

2.1.3. Definición de las clases del modelo de dominio

Usuario. Persona que accede a la plataforma GeneSIG con el objetivo de interactuar con un mapa representado por información *raster*.

Mapa. Representación gráfica y métrica de una porción de territorio sobre la superficie bidimensional, generalmente plana, pero que puede ser también esférica como ocurre en los globos terráqueos.

Información *raster*. La información en formato *raster* la constituyen fotografías aéreas digitales, imágenes de satélite e imágenes digitales.

Escala. Relación entre la distancia que separa dos puntos en un mapa y la distancia real de esos dos puntos en la superficie terrestre. En los mapas, la escala puede expresarse de tres modos distintos: en forma de proporción o fracción, con una escala gráfica o una expresión en palabras y cifras. Cuanto mayor es la escala, más se aproxima al tamaño real de los elementos de la superficie terrestre.

Proyección. Es un sistema de representación gráfico que establece una relación ordenada entre los puntos de la superficie curva de la Tierra y los de una superficie plana (mapa). Estos puntos se localizan auxiliándose en una red de meridianos y paralelos, en forma de malla.

Capa. Conjunto lógico de elementos temáticos descritos y almacenados en una biblioteca. Estas capas organizan la biblioteca según temas.

2.2 Requisitos de software

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. Existen varias técnicas que facilitan la comunicación con los clientes del producto con el fin de lograr su satisfacción entre las que se encuentran entrevistas, cuestionarios, tormenta de ideas y observación. Sobre la base del análisis de las soluciones existentes en el capítulo anterior, la representación de los conceptos u objetos del mundo real mediante el

modelo de dominio y la necesidad de realizar una correcta captura de los requisitos, para evitar demoras en la construcción del módulo y errores en los mismos se realizará una tormenta de ideas con especialistas de Aplicativos SIG.

2.2.1. Técnica de recopilación de requisitos

Es una herramienta de trabajo grupal que facilita el surgimiento de nuevas ideas sobre un tema o problema determinado. Consiste en reuniones de grupos no muy numerosos (máximo 10 personas), con el objetivo de acumular ideas e información sin evaluar las mismas. Posibilita que el proceso de recopilación de requisitos se realice de forma más eficiente. La tormenta de ideas permitió el intercambio con un grupo de profesionales de la LPS que trabajan sobre la base de información en formato *raster*. Este debate permitió determinar qué funcionalidades deben estar presentes en el módulo a desarrollar.

2.2.2. Especificación de requisitos funcionales

Los requisitos funcionales (RF) son condiciones o capacidades con los que debe cumplir el sistema, son acuerdos entre el cliente y los desarrolladores sobre lo que debe o no debe hacer el sistema (Jacobson, y otros, 2000). Constituyen la base sobre la que se construirá el módulo, conformando el conjunto de funcionalidades con las que debe contar y guiando el proceso de desarrollo. A continuación se exponen los RF que el módulo a desarrollar debe poseer:

RF 1. Añadir punto de control del terreno (PCT). Esta funcionalidad permite añadir un punto de referencia en la imagen.

RF 2. Mover PCT. Esta funcionalidad permite mover un punto de referencia hacia otra posición en la imagen.

RF 3. Eliminar PCT. Esta funcionalidad permite eliminar un punto de referencia de la imagen.

RF 4. Mostrar tabla de PCT. Esta funcionalidad muestra una tabla con las coordenadas de los puntos de referencia insertados hasta el momento.

RF 5. Georreferenciar imagen. Esta funcionalidad permite georreferenciar una imagen en formato *raster* a partir de los PCT especificados.

RF 6. Cargar imagen. Esta funcionalidad permite al usuario cargar una imagen desde un archivo para realizar su georreferenciación.

RF 7. Redimensionar mapa. Esta funcionalidad permite redimensionar un mapa en formato *raster* a un nuevo ancho y alto.

RF 8. Cambiar proyección del mapa. Esta funcionalidad permite establecer una relación ordenada entre los puntos de la superficie curva de la Tierra y los del mapa.

RF 9. Exportar mapa como imagen. Esta funcionalidad permite exportar como una imagen un mapa en formato *raster* que se encuentre almacenado en la base de datos. Los formatos de conversión son JPEG y PNG.

RF 10. Guardar imagen *raster* en una base de datos. Esta funcionalidad permite guardar en base de datos una imagen en formato *raster*.

RF 11. Mostrar lista de conexiones a la base de datos. Esta funcionalidad muestra una tabla con los datos de una conexión a la base de datos (IP del servidor de bases de datos, puerto, usuario y nombre de la base de datos), exceptuando la contraseña.

RF 12. Añadir conexión a la base de datos. Esta funcionalidad permite agregar una nueva conexión a una base de datos.

RF 13. Editar conexión a la base de datos. Esta funcionalidad permite editar los parámetros de una conexión existente.

RF 14. Eliminar conexión a la base de datos. Esta funcionalidad permite eliminar una conexión existente.

2.2.3. Especificación de requisitos no funcionales

Además de los requisitos funcionales deben especificarse los requisitos no funcionales (RNF). Según (Jacobson, y otros, 2000) son requisitos que especifican propiedades del sistema, como restricciones del entorno o la implementación, rendimiento, dependencias de la plataforma, facilidad de mantenimiento, extensibilidad y fiabilidad. A continuación se presentan los RNF que debe cumplir el módulo a desarrollar:

RNF 1. Software

Se requiere para las estaciones de trabajo cliente las siguientes condiciones:

- Navegador web que cumpla con los estándares W3C[16].

Se requiere para las estaciones de trabajo servidoras las siguientes condiciones:

- Sistema Operativo: GNU/Linux Ubuntu Server 14.04.
- Servidor Web Apache 2.4.7, con módulo PHP 5 configurado con la extensión pgsql incluida.
- PostgreSQL 9.3 como Sistema Gestor de Base de Datos.
- PostGis 2.1 como extensión de PostgreSQL para el manejo de datos espaciales.
- MapServer 6.4 con extensión PHP mapscript.

RNF 2. Hardware

Se requiere para las estaciones de trabajo cliente las siguientes condiciones:

- Debe poseer tarjeta de red.
- Procesador: 512MHz.
- Memoria RAM: 128Mb.
- Disco Duro: 40Gb.

Se requiere para las estaciones de trabajo servidoras las siguientes condiciones:

- Debe poseer tarjeta de red.
- Procesador: 3Ghz.
- Memoria RAM: 2Gb.

[16] *World Wide Web Consortium*, abreviado W3C, es un consorcio internacional que produce recomendaciones para la web mundial.

- Disco Duro: 160Gb

Se debe realizar una salva semanal de los registros generados por PostgreSQL para poder eliminarlos del servidor de bases de datos. Estos ficheros llegan a ocupar gran cantidad de espacio del disco duro debido al procesamiento de archivos en formato *raster* de gran tamaño.

RNF 3. Interfaz

Se desea que la interfaz externa del producto sea de fácil navegación por el usuario, sencilla y legible, que mantenga los estándares y características de la plataforma soberana GeneSIG como son: los colores, la estructura de los botones, el estilo y tamaño de letra, ya que el módulo a construir va a ser incorporado a la plataforma. Para lograr una interfaz de usuario amigable, atractiva y funcional para el usuario final, es necesario tener en cuenta algunos principios de diseño de interfaz de usuario que serán listados a continuación.

- Las funcionalidades deberán estar al alcance de un clic y representadas por íconos asociados a la acción que se realiza, de manera que cualquier persona con un mínimo dominio de la informática pueda hacer uso del sistema.
- Garantizar la legibilidad de manera que exista contraste de los colores de los textos con el fondo y el tamaño de la fuente sea lo suficientemente adecuado a la vista del usuario.
- Los mensajes mostrados al usuario deben ser concisos y de fácil comprensión.
- Los menús y etiquetas de botones deben comenzar con la palabra más importante.

2.3 Modelo de Casos de Uso del Sistema (CUS)

Una vez recopilados los requisitos funcionales del sistema es necesario conformar el Diagrama de Casos de Uso del Sistema (DCUS). Un DCUS muestra la relación entre los casos de uso y los actores del sistema.

2.3.1. Descripción de los actores que interactúan con el sistema

Un actor es un usuario del sistema, esto incluye usuarios humanos y otros sistemas computacionales. A continuación, se mencionan los actores que van a interactuar con el módulo a construir, definiendo el rol que le ocupa dentro del mismo.

Tabla 1. Descripción de los actores del sistema

Actor	**Descripción**
Usuario	Representa los usuarios que van a interactuar con el módulo.

2.3.2. Identificación de los CUS

(Pressman, 2005) define los casos de uso como un conjunto de escenarios que identifican una línea de utilización para el sistema que va a ser construido y que facilitan una descripción de cómo el sistema se usará. A continuación se identifican los casos de uso que hacen referencia a los requisitos funcionales que requiere el módulo a construir.

Tabla 2. Casos de uso del sistema

Referencia a requisitos	**Nombre del caso de uso**
RF: 1, 2, 3, 4, 5 y 6	Georreferenciar
RF: 7 y 8	Editar mapa *raster*
RF: 9	Exportar mapa *raster* como imagen
RF: 10	Guardar imagen *raster* en base de datos
RF: 11, 12, 13 y 14	Gestionar conexiones a bases de datos

2.3.3. Diagrama de CUS

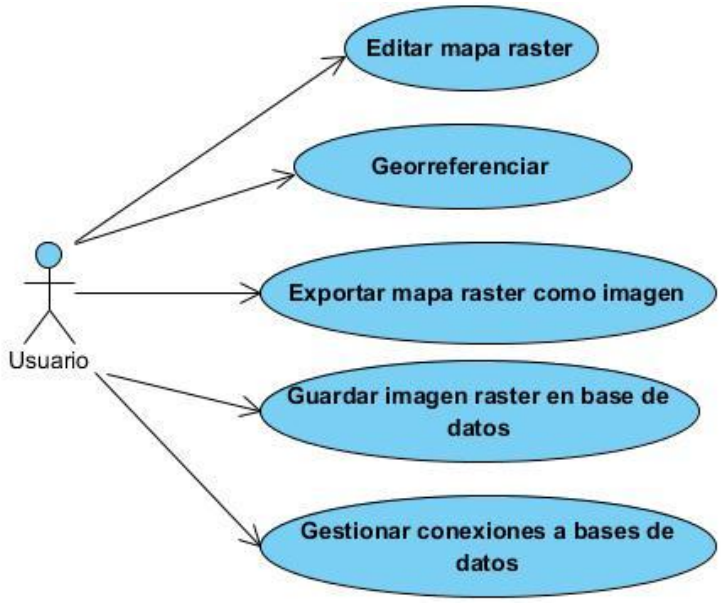

Figura 7. Diagrama de CUS

2.3.4. Descripción extendida de los CUS

Debido a la existencia de varios CUS y que las descripciones textuales son muy extensas, se propone la del caso de uso Guardar imagen *raster* en base de datos a continuación y las de los restantes pueden ser consultadas en el documento “Módulo raster. Especificación de casos de uso” del expediente de proyecto.

Tabla 3. Descripción del CUS Guardar imagen *raster* en base de datos

Caso de uso	Guardar imagen *raster* en base de datos	
Objetivo	Almacenar la imagen en formato *raster* en una base de datos.	
Actores	Usuario: (Inicia)	
Resumen	El CUS permite almacenar en una base de datos, con parámetros especificados por el usuario, el mapa en formato *raster*.	
Complejidad	Media	
Prioridad	Crítico	
Referencias	RF 10	
Precondiciones	Existe una conexión válida a la base de datos en la que se pretende guardar la imagen.	
Poscondiciones	Se almacena en base de datos la información en formato *raster* del mapa.	
Flujo de eventos		
Flujo básico Guardar imagen en formato *raster* en base de datos		
	Actor	**Sistema**
1	Selecciona en la barra de herramientas la opción *Raster*.	
2		Muestra la interfaz correspondiente a la ventana principal del módulo.
3	Selecciona la pestaña Guardar en BD	
4		Muestra la interfaz correspondiente para guardar el mapa en la base de datos.
5	Inserta los datos necesarios para guardar la imagen: Nombre de la base de datos en la que la va a guardar, contraseña para la conexión a dicha base de datos, la proyección que tendrá el mapa y el archivo de la imagen, y presiona el botón “Aceptar”.	

6		Comprueba que no existan campos vacíos.
7		Comprueba que el fichero seleccionado por el usuario sea una imagen en el formato permitido (JPEG, PNG, TIFF)
8		Comprueba que el nombre de la tabla no contenga caracteres especiales.
9		Almacena el mapa en la base de datos, terminando así el caso de uso.
Flujos alternos		
Nº Evento 5. El usuario cancela la acción		
	Actor	**Sistema**
5.1	Presiona el botón “Cancelar”	
5.2		Reinicia todos los campos del formulario.
Nº Evento 6. Existen campos vacíos		
	Actor	**Sistema**
6.1		Señala en rojo los campos que no deben ser vacíos y al colocar el cursor sobre estos muestra un mensaje, “Campo obligatorio”.
Nº Evento 7. Formato no válido		
	Actor	**Sistema**
7.1		Emite un mensaje “El formato del fichero no es válido”.
Nº Evento 8. Nombre de tabla con caracteres especiales		
	Actor	**Sistema**
8.1		Emite un mensaje “El nombre de la tabla no es válido”

Relaciones	**CU Incluidos**	No procede
	CU Extendidos	No procede
Requisitos no funcionales	RNF 1, RNF 2, RNF 3 Y RNF 4	
Asuntos pendientes	No procede	

2.4 Conclusiones parciales

A pesar de que no se tienen bien definidos los procesos del negocio se lograron representar los conceptos más significativos en el dominio del problema. La realización de una tormenta de ideas permitió un mejor análisis y comprensión del problema y las necesidades planteadas en la investigación, arrojando como resultado los requisitos funcionales y no funcionales del módulo a desarrollar. La descripción detallada de los casos de uso identificados constituye una guía en el proceso de desarrollo de las funcionalidades del módulo.

Capítulo 3. Implementación del módulo *raster* para la plataforma GeneSIG

El presente capítulo muestra los artefactos ingenieriles relacionados con el diseño, implementación y validación del módulo a desarrollar. Entre los principales elementos que muestran se encuentran la arquitectura de software seleccionada y el modelo de diseño, incluyendo los patrones arquitectónicos y de diseño utilizados respectivamente. De igual manera se incluyen el modelo de datos, el de despliegue, el de implementación y las pruebas realizadas al módulo.

3.1 Arquitectura de software

La definición de arquitectura de software más usada, asumida como oficial y adoptada también por grandes compañías desarrolladoras de software, es la que brinda el estándar de la IEEE[17] que expresa lo siguiente: "*La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución"* (IEEE, 2000).

La arquitectura de software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión de este. La vista arquitectónica es una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.

Teniendo en cuenta que se empleará la plataforma GeneSIG como herramienta base para el desarrollo, y que esta ha sido desarrollada haciendo uso del *framework* CartoWeb es conveniente mantener la misma arquitectura para evitar futuras incompatibilidades. En este caso CartoWeb posee una arquitectura orientada a objetos y basada en componentes.

[17] *Institute of Electrical and Electronics Engineers* (IEEE, por sus siglas en inglés, Instituto de Ingeniería Eléctrica y Electrónica). Es una asociación mundial de técnicos e ingenieros dedicada a la estandarización y el desarrollo en áreas técnicas

3.1.1. Patrones arquitectónicos

Un patrón arquitectónico impone una transformación en el diseño de la arquitectura, su alcance es más específico ya que se concentra en un aspecto, en lugar de hacerlo en toda la arquitectura. Un patrón aplica una regla sobre la arquitectura, describe la manera en que el software maneja algún aspecto de su funcionalidad al nivel de la infraestructura, abarca aspectos específicos del comportamiento dentro del contexto de la arquitectura y es usado para determinar la forma de la estructura general de un sistema.

El módulo *raster*, se ha desarrollado bajo los patrones arquitectónicos orientado a objetos y basado en componentes. El primero define el sistema como un conjunto de objetos que cooperan entre sí en lugar de un conjunto de procedimientos. Según (Reynoso, y otros, 2004) *"los componentes del estilo se basan en principios orientados a objetos: encapsulamiento, herencia y polimorfismo. Las interfaces están separadas de las implementaciones. Las representaciones de los datos y las operaciones están encapsuladas en un tipo abstracto de datos u objeto. La comunicación entre los componentes es a través de mensajes"*.

El patrón arquitectónico basado en componentes, por otra parte, define cómo organizar el modelo en componentes funcionales, exponiendo interfaces de comunicación que contienen métodos, eventos y propiedades. El mismo permite que se pueda representar el módulo como un componente que es incorporado a la plataforma GeneSIG, de manera que no altere los restantes componentes ni la estructura de la misma; brindando la posibilidad de que el sistema sea flexible y fácil de personalizar.

3.2 Modelo de diseño

El modelo de diseño es un modelo de objetos que describe la realización física de los casos de uso centrándose en cómo los requisitos funcionales y no funcionales, junto con otras restricciones relacionadas con el entorno de implementación, tienen impacto en el módulo, siendo la principal vía de acceso en la actividad de implementación (Jacobson, y otros, 2000).

3.2.1. Diagrama de paquetes

En el lenguaje UML, se modelan diferentes diagramas para presentar el diseño de la aplicación. Uno de estos diagramas es el de paquete. Un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones (Pressman, 2000). Estos diagramas suministran una descomposición de la jerarquía lógica de un sistema. Los paquetes están

normalmente organizados para maximizar la coherencia interna dentro de cada uno y minimizar el acoplamiento externo entre ellos. Se propone de esta forma, el siguiente diagrama de paquetes para el módulo *raster*.

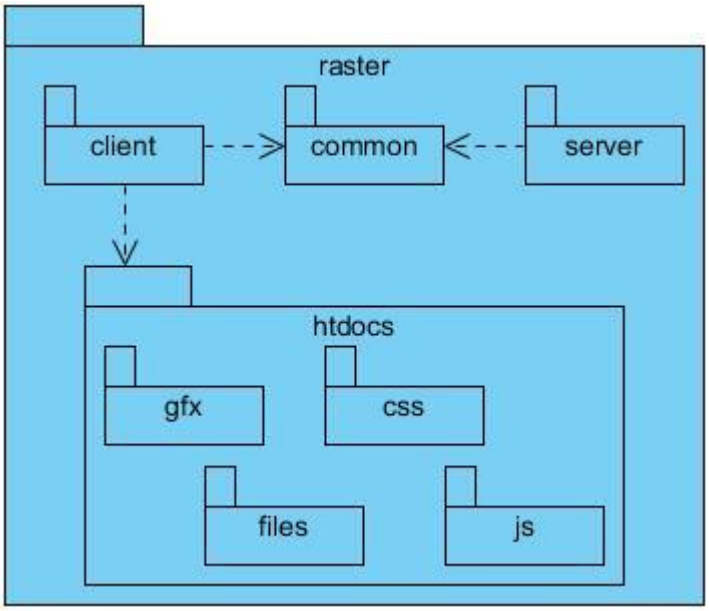

Figura 8. Diagrama de paquetes

Breve descripción de los paquetes del módulo *raster*.

client/ contiene todos los componentes (de extensión php) que implementan el módulo CartoClient del CartoWeb.

server/ contiene todos los componentes (de extensión php) que implementan el módulo CartoServer del CartoWeb.

common/ contiene todos los componentes que implementan aquellas clases que servirán de puente para la comunicación entre client.php y server.php.

htdocs/gfx/ contiene las imágenes que serán utilizadas en el módulo.

htdocs/css/ contiene las hojas de estilo que serán utilizadas en el módulo.

htdocs/js/ contiene todos los componentes (de extensión js) que implementan el módulo cliente en javascript del CartoWeb.

htdocs/files/ contiene temporalmente las imágenes que el usuario seleccione para guardar en la base de datos.

3.2.2. Diagrama de clases del diseño

Las clases del diseño representan una abstracción de una o varias clases en la implementación del sistema. El lenguaje utilizado para especificar una clase del diseño es el mismo que el lenguaje de programación utilizado, los métodos tienen correspondencia directa con el correspondiente método de la implementación de clases, puede aparecer como un estereotipo que se corresponde con una construcción en el lenguaje de programación dado.

Los diagramas de clases del diseño se emplean en el modelado de las vistas del diseño del sistema para describir las especificaciones de las interfaces y las clases del software, lo que facilita el trabajo de los implementadores. Teniendo en cuenta lo antes expuesto, se propone el siguiente diagrama de clases del diseño para el caso de uso Guardar imagen *raster* en base de datos. Los diagramas de los CUS restantes pueden ser consultados en el documento “Módulo raster. Modelo de diseño” del expediente de proyecto.

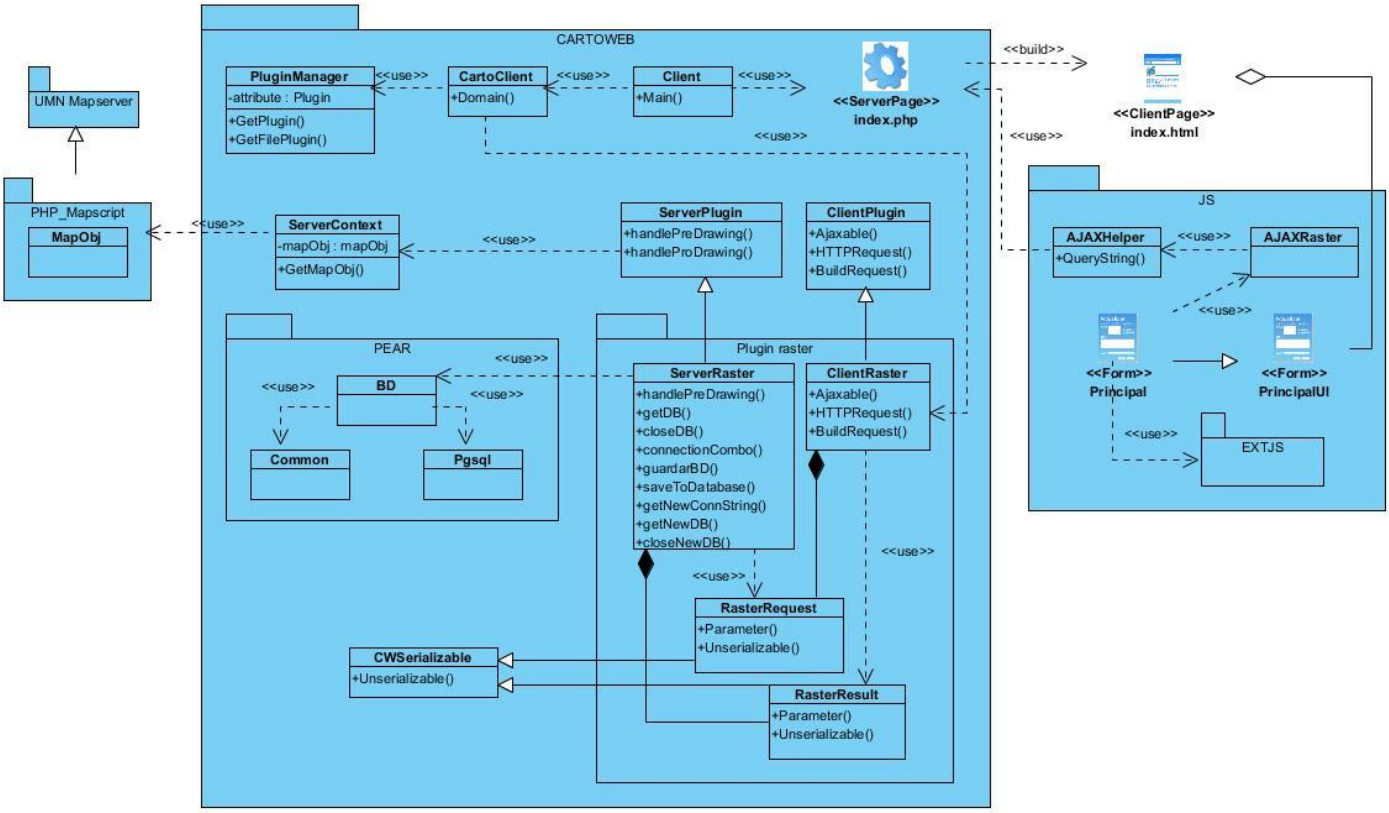

Figura 9. Diagrama de clases del diseño. CUS Guardar imagen *raster* en base de datos

3.2.3. Patrones de diseño

Los patrones de diseño constituyen la base para la búsqueda de soluciones a problemas comunes en el proceso de desarrollo de software. Se caracterizan por su reusabilidad, flexibilidad y aplicabilidad a diferentes problemas de diseño en diversas circunstancias. El patrón es un esquema de solución que se aplica a un tipo de situación, esta aplicación del mismo no es mecánica, sino que requiere de adaptación y matrices.

En el diseño de la propuesta de solución se aplican los patrones GRASP[18], que permiten describir los principios fundamentales de diseño de objetos para la asignación de responsabilidades.

Experto: Se aplica para asignar una responsabilidad al experto en información: la clase que cuenta con la información necesaria para cumplir la responsabilidad. En este caso, se tiene la clase ClientRaster, que es experta en procesar los datos que son enviados a través del navegador web, la clase ServerRaster, que es la responsable de la interacción con el servidor de mapas y el servidor de bases de datos, y la clase raster.ajax, experta en el comportamiento por la parte cliente del módulo. El uso de este patrón permite que se conserve el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide.

Creador: Se aplica para la asignación de responsabilidades a las clases relacionadas con la creación de objetos, de forma tal que una instancia de un objeto sólo pueda ser creada por el objeto que contiene la información necesaria para ello. En este caso el patrón se refleja en las clases ClientRaster y ServerRaster, encargadas de crear una instancia de las clases RasterRequest y RasterResult, clases que describen las variables que contienen la información que forma parte de los valores de entrada de la solicitud que realiza el ClientRaster al ServerRaster y viceversa.

Bajo Acoplamiento: El acoplamiento es una medida de la fuerza en que una clase está conectada a otras, que las conoce y recurre a ellas. En este caso, se refleja el bajo acoplamiento, en cada una de las clases del módulo *raster*, con el objetivo de que una no dependa de muchas otras, de esta forma, no se afectan las clases por cambios de otros componentes, son fáciles de entender por separado y fáciles de reutilizar.

Alta Cohesión: La cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. En este caso, se garantiza que cada una de las clases del módulo *raster* presente alta cohesión, de manera que las clases posean la característica de tener las responsabilidades estrechamente relacionadas y que no realicen un trabajo enorme. El uso de este patrón permite que se

[18] *General Responsibility Assignment Software Patterns* (GRASP, por sus siglas en inglés, Patrones generales de software para asignar responsabilidades)

pueda mejorar la claridad y facilidad en que se entiende el diseño, se simplifique el mantenimiento y existan mejoras de funcionalidad.

Controlador: Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. En este caso, se encuentra reflejado en la clase raster.ajax, encargada del comportamiento del módulo y de gestionar todos los eventos que ocurren en el mismo.

3.3 Modelo de datos

Un modelo de datos se puede definir como un conjunto de conceptos, reglas y convenciones bien definidos que permiten aplicar una serie de abstracciones a fin de describir y manipular los datos de un cierto mundo real que se desea almacenar en la base de datos (BD).

La BD a utilizar por el módulo a desarrollar dependerá de la personalización que se quiera hacer de la plataforma GeneSIG. Teniendo en cuenta que no se encuentran definidas las fronteras del negocio, se establece que el SGBD se utilizará para almacenar la información de la cartografía que se vaya a utilizar en el momento de cargar un mapa. Por tal motivo, se considera que no es necesario restringir el uso de la BD a tablas previamente definidas, cuando se conoce que la BD va a variar en dependencia de la cartografía que se utilice. Por otra parte, debe quedar incluida, en cada una de las bases de datos de las personalizaciones en la que se incluya el módulo *raster*, la tabla Conexiones, encargada de almacenar las conexiones que gestione el usuario. A continuación se muestra el Diagrama Entidad-Relación propuesto.

Figura 10. Diagrama Entidad-Relación

3.4 Modelo de implementación

El modelo de implementación describe cómo los elementos del modelo de diseño se implementan en términos de componentes (base de datos, ejecutables, módulos o ficheros). De igual manera describe

cómo se organizan los componentes de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación y en el lenguaje o lenguajes de programación utilizados y cómo dependen los componentes unos de otros (Jacobson, y otros, 2000).

Un componente es una unidad modular que puede reemplazarse en su propio entorno y cuyos elementos internos quedan ocultos. Puede poseer interfaces proporcionadas, a través de las cuales se puede obtener acceso a sus funciones, e interfaces necesarias, en las que se definen las funciones o servicios de otros componentes que son necesarios. Mediante la conexión de las interfaces proporcionadas y las interfaces necesarias de distintos componentes puede construirse un componente mayor. Un sistema de software completo se puede concebir como un componente.

El diagrama de componentes permite concebir el diseño atendiendo a los bloques principales y ayuda al equipo de desarrollo a entender un diseño existente y crear uno nuevo. Al establecer el sistema como una colección de componentes con interfaces proporcionadas y necesarias bien definidas se garantiza la correcta separación entre los componentes. A su vez, se facilita la comprensión de los cambios al modificar los requisitos. A continuación se propone el diagrama de componentes para el módulo *raster*, representando de forma física el sistema.

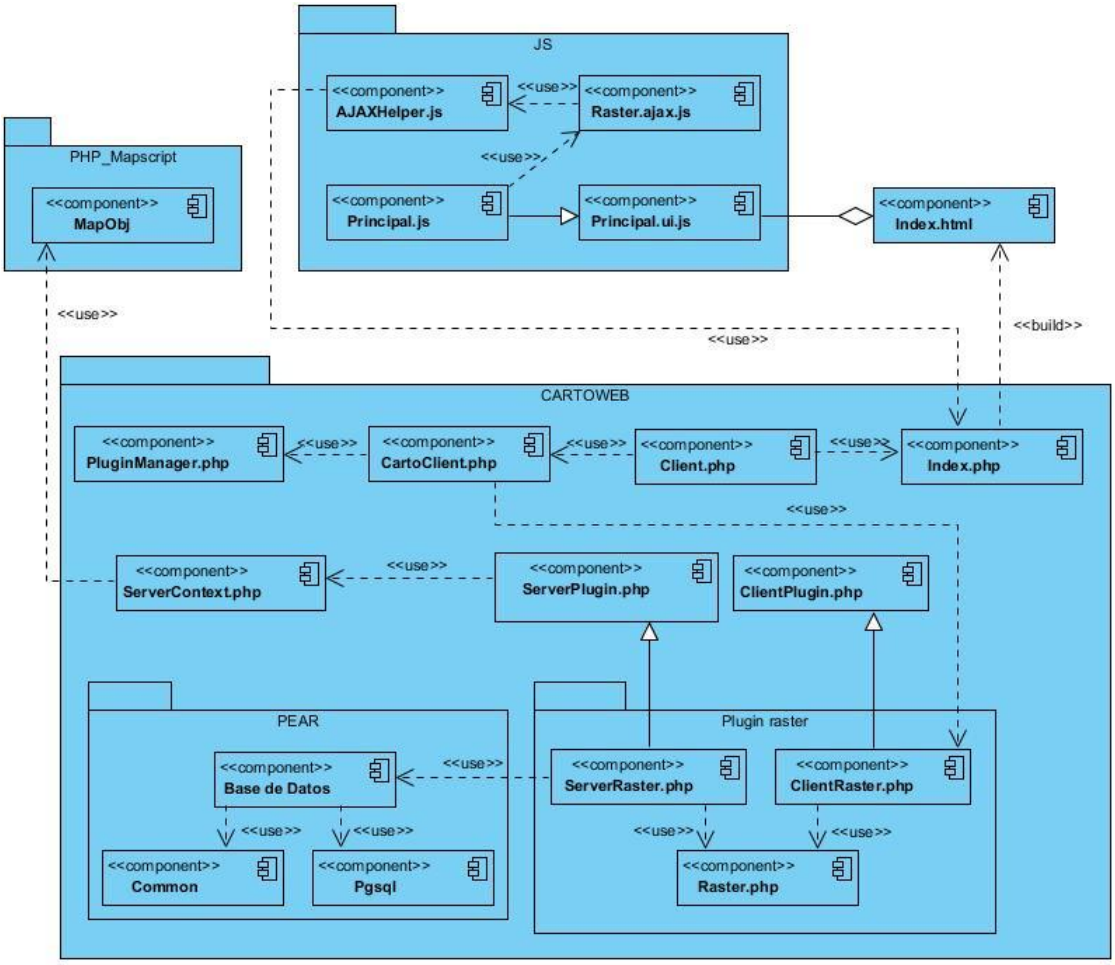

Figura 11. Diagrama de componentes

3.5 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 (Jacobson, y otros, 2000), permitiendo modelar mediante este diagrama la vista de despliegue estática, describir la arquitectura en tiempo de ejecución de procesadores, dispositivos y los componentes de software que ejecutan esta arquitectura, además de la topología del sistema, estructura de hardware y el software que se ejecuta en cada unidad. Sus principales elementos son los nodos, las conexiones y los elementos de anotación.

Nodos: Es un elemento físico que existe en tiempo de ejecución, representando un recurso computacional que, por lo general, dispone de algo de memoria y con frecuencia capacidad de

procesamiento. Un conjunto de componentes puede residir en un nodo y puede también migrar de un nodo a otro.

Conexiones: El tipo de comunicación es representado por un estereotipo que identifica el protocolo de comunicación o el tipo de red usado.

Elementos de anotación: Los elementos de anotación son las partes explicativas de los modelos UML. Son comentarios que se pueden aplicar para describir, clasificar y hacer observaciones sobre cualquier elemento de un modelo. El tipo principal de anotación es la nota que simplemente es un símbolo para mostrar restricciones y comentarios junto a un elemento o un conjunto de elementos.

Se muestra a continuación el diagrama de despliegue, representando cómo quedará desplegado el módulo *raster*.

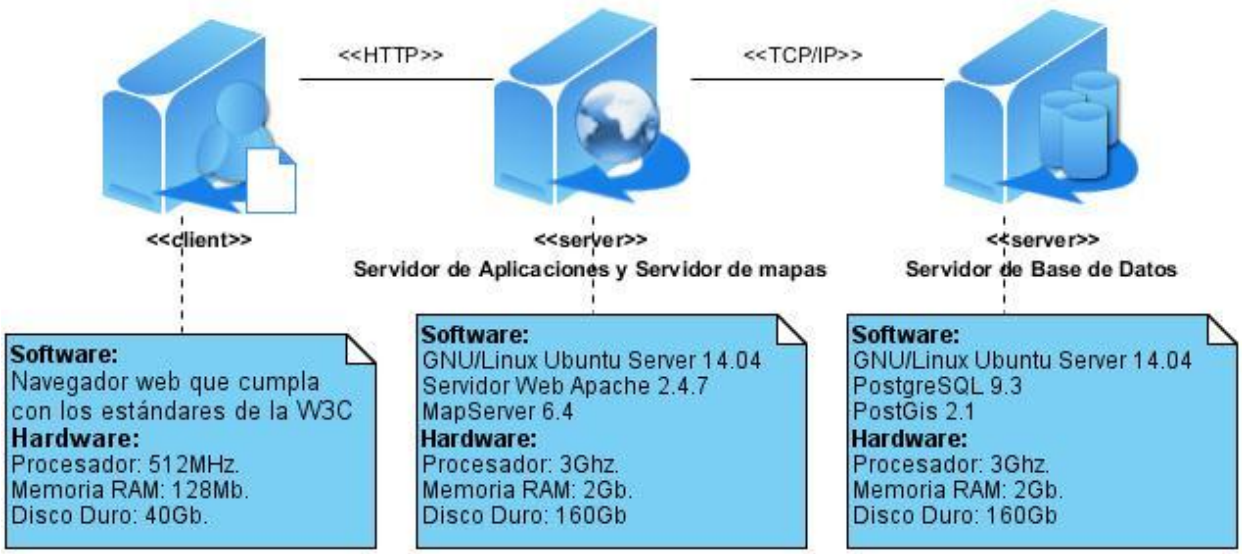

Figura 12. Diagrama de despliegue

3.6 Validación y verificación

La verificación y validación es el nombre que se da a los procesos de comprobación y análisis que aseguran que el software que se desarrolla está acorde a su especificación y cumple las necesidades de los clientes (Drake, y otros, 2009). La verificación y la validación no son la misma cosa, el papel de la verificación comprende comprobar que el software está de acuerdo con su especificación. Se comprueba

que el sistema cumple los requerimientos funcionales y no funcionales que se le han especificado. La validación es un proceso más general. Se debe asegurar que el software cumple las expectativas del cliente. Va más allá de comprobar si el sistema está acorde con su especificación, para probar que el software hace lo que el usuario espera a diferencia de lo que se ha especificado.

Dentro del proceso de verificación y validación se utilizan dos técnicas de comprobación y análisis de sistemas, las inspecciones de software y las pruebas del software. La primera es una técnica estática que analiza y comprueba las representaciones del sistema como el documento de requerimientos, los diagramas de diseño y el código fuente del programa. La segunda es una técnica dinámica que consiste en contrastar las respuestas de una implementación del software a series de datos de prueba y examinar las respuestas del software y su comportamiento operacional, para comprobar que se desempeñe conforme a lo requerido (Drake, y otros, 2009).

Las técnicas estáticas sólo pueden comprobar la correspondencia entre un programa y su especificación y no puede probar que el software es de utilidad operacional, y mucho menos que las características no funcionales del software son las correctas. Por lo tanto, para validar un sistema de software, siempre se requieren llevar a cabo ciertas pruebas.

3.6.1. Pruebas de software

Concretamente la prueba de software se puede definir como una actividad en la cual un sistema o uno de sus componentes se ejecutan en circunstancias previamente especificadas, registrándose los resultados obtenidos (Juristo, y otros, 2005). Seguidamente se realiza un proceso de evaluación en el que los resultados obtenidos se comparan con los resultados esperados para localizar fallos en el software.

Las técnicas de evaluación dinámica o prueba proporcionan distintos criterios para generar casos de prueba que provoquen fallos en los programas. Estas técnicas se agrupan en Técnicas de Caja Blanca o Estructurales, que se basan en un minucioso examen de los detalles procedimentales del código a evaluar; y Técnicas de Caja Negra o Funcionales, que realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. Estas son técnicas complementarias que han de aplicarse al realizar una prueba dinámica, ya que pueden ayudar a identificar distintos tipos de fallas en un programa.

Debido a las características del módulo desarrollado y del sistema para el que se realizó, es necesario incluirlo en la plataforma GeneSIG para realizar las pruebas de caja negra. Luego de comprobar con las pruebas de caja blanca que el módulo por si solo funciona correctamente, se configurarán los archivos establecidos por la plataforma para incluirlo como un *plugin* de la misma. Una vez se configure el acceso al módulo desde GeneSIG se podrá proceder a la realización de las pruebas de caja negra.

3.6.2. Pruebas de Caja Blanca o Estructurales

Con el objetivo de llevar a cabo la Estrategia de Pruebas de Unidad, que comprueba que el módulo, entendido como una unidad funcional de un programa independiente, está correctamente codificado, se realizan en esta investigación las pruebas de Caja Blanca o Estructurales.

A este tipo de prueba se le conoce también como Técnicas de Caja Transparente o de Cristal. Este método se centra en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa. Se examina así la lógica interna del programa sin considerar los aspectos de rendimiento. El objetivo es diseñar casos de prueba para que se ejecuten, al menos una vez, todas las sentencias del programa, y todas las condiciones tanto en su vertiente verdadera como falsa. Existen distintos criterios de cobertura lógica, que permiten decidir qué sentencias o caminos se deben examinar con los casos de prueba. El Criterio de cobertura de caminos asegura que los casos de prueba diseñados permiten que todas las sentencias del programa sean ejecutadas al menos una vez y que las condiciones sean probadas tanto para su valor verdadero como falso.

Una de las técnicas empleadas para aplicar este criterio de cobertura es la Prueba del Camino Básico. Esta técnica se basa en obtener una medida de la complejidad del diseño procedimental de un programa (o de la lógica del programa). Esta medida es la complejidad ciclomática de McCabe, y representa un límite superior para el número de casos de prueba que se deben realizar para asegurar que se ejecuta cada camino del programa (Juristo, y otros, 2005).

Diseño de casos de prueba de Caja Blanca.

A continuación se muestra la prueba del camino básico aplicada al método saveToDatabase de la clase ServerRaster del módulo, para la cual se siguieron una serie de pasos lógicos.

```php
public function saveToDatabase($connection, $pass, $name, $projection, $file){
    $success = false;
    $data = array();
    $con = $this->getDB();
    $query = "select servidor, puerto, nombre, usuario from conexiones where nombre='" . $connection . "'";
    $result = $con->query($query);
    $this->closeDB($con);
    if ($result->fetchInto($data)) {
        $host = $data[0];
        $port = $data[1];
        $database = $data[2];
        $user = $data[3];

        $conStr = $this->getNewConnString($database, $pass);
        $con2 = $this->getNewDB($conStr);
        if ($con2) {
            $query2 = "select * from " . $name;
            $result2 = pg_query($con2, $query2);
            $this->closeNewDB($con2);
            if (!$result2) {
                $cmd = sprintf('raster2pgsql -s %s -I -C -M %s -F -t 100x100 %s | PGPASSWORD="%s" psql -h %s -p %s -U %s -d %s',
                    $projection,
                    $file,
                    $name,
                    $pass,
                    $host,
                    $port,
                    $user,
                    $database);
                $out = shell_exec($cmd);
                $success = strpos($out, "CREATE TABLE") ? true : false;
                unlink($file);
            }
        }
    }
    return $success;
}
```

Figura 13. Método saveToDatabase de la clase ServerRaster

El método recibe por parámetros el nombre de la base de datos en la que se va a almacenar la imagen, su contraseña, el nombre de la tabla que será creada, la proyección con que se guardará y la dirección de la imagen a guardar. Selecciona los parámetros de conexión de la base de datos introducida por el usuario, y ejecuta el comando de consola que permite guardar la imagen en la base de datos. Retorna true en caso de guardar satisfactoriamente la imagen y false en caso contrario.

Representación del método en un grafo de flujo.

El grafo de flujo se utiliza para representar el flujo de control lógico del método. Para ello se utilizan los tres elementos siguientes:

- Nodos: representan cero, una o varias sentencias en secuencia. Cada nodo comprende como máximo una sentencia de decisión.

- Aristas: líneas que unen dos nodos.

- Regiones: áreas delimitadas por aristas y nodos.

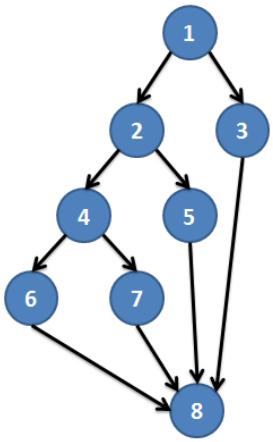

Figura 14. Grafo de flujo del método saveToDatabase de la clase ServerRaster

Cálculo de la complejidad ciclomática.

Existen varias formas de calcular la complejidad ciclomática V(G) de un método a partir de un grafo de flujo G, las cuales deben arrojar el mismo resultado para asegurar que el cálculo de la complejidad sea correcto.

- V(G) = Aristas - Nodos + 2.
  V(G) = 10 – 8 + 2
  V(G) = 4

- V(G) = Nodos de predicado + 1.
  V(G) = 3 + 1
  V(G) = 4

- V(G) = Número de regiones del grafo.
  V(G) = 4

Conjunto básico de caminos independientes.

El valor de la complejidad ciclomática nos da un límite superior para el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. Por tanto a continuación se identifica el conjunto básico de caminos independientes.

Camino 1: 1-2-4-6-8

Camino 2: 1-2-4-7-8

Camino 3: 1-2-5-8

Camino 4: 1-3-8

Estos 4 caminos constituyen el camino básico para el grafo de flujo correspondiente.

Casos de prueba que fuerzan la ejecución de cada camino.

A cada camino obtenido de la prueba de camino básico se le realiza un caso de prueba.

Número del camino: 1

Caso de prueba:

connection = nombre de una base de datos a la cual se pueda conectar el usuario.

pass = contraseña del usuario que tiene acceso a la base de datos.

name = nombre de una tabla que no exista en la base de datos.

projection = cualquier valor de proyección.

file = cualquier dirección válida a un archivo de imagen.

Resultado esperado: Se almacena la imagen en la base de datos y retorna *true*.

Resultado de la prueba: Satisfactorio.

Número del camino: 2

Caso de prueba:

connection = nombre de una base de datos a la cual se pueda conectar el usuario.

pass = contraseña del usuario que tiene acceso a la base de datos.

name = nombre de una tabla que ya existe en la base de datos.

projection = cualquier valor de proyección.

file = cualquier dirección válida a un archivo de imagen.

Resultado esperado: Retorna *false*.

Resultado de la prueba: Satisfactorio.

Número del camino: 3

Caso de prueba:

connection = nombre de una base de datos a la cual no se puede conectar el usuario por errores en los datos de conexión proporcionados.

pass = contraseña del usuario que tiene acceso a la base de datos.

name = nombre de una tabla que no exista en la base de datos.

projection = cualquier valor de proyección.

file = cualquier dirección válida a un archivo de imagen.

Resultado esperado: Retorna *false*.

Resultado de la prueba: Satisfactorio**.**

Número del camino: 4

Caso de prueba:

connection = nombre de una base de datos que no existe.

pass = valor de contraseña.

name = nombre de una tabla de la base de datos.

projection = valor de proyección.

file = dirección válida a un archivo de imagen.

Resultado esperado: Retorna *false*.

Resultado de la prueba: Satisfactorio.

3.6.3. Pruebas de Caja Negra o Funcionales

Estas pruebas se basan en la especificación del programa o componente a ser probado para elaborar los casos de prueba. El componente se ve como una “Caja Negra” cuyo comportamiento sólo puede ser determinado estudiando sus entradas y las salidas obtenidas a partir de ellas (Juristo, y otros, 2005). No obstante, como el estudio de todas las posibles entradas y salidas de un programa sería impracticable se selecciona un conjunto de ellas sobre las que se realizan las pruebas. El objetivo es encontrar una serie de datos de entrada cuya probabilidad de pertenecer al conjunto de entradas que causan un comportamiento erróneo sea lo más alto posible.

Para confeccionar los casos de prueba de Caja Negra existen distintos criterios, el seleccionado para realizar las pruebas al módulo *raster* es el de Particiones de equivalencia. (Pressman, 2005) presenta la partición de equivalencia como un método de prueba que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. A continuación se muestra la descripción de las pruebas aplicadas al caso de uso Guardar imagen *raster* en base de datos, las pruebas aplicadas al

resto de los casos de uso pueden ser consultadas en el documento “Módulo raster. Diseño de casos de prueba” del expediente de proyecto.

Diseño del caso de prueba para el CUS Guardar imagen *raster* en base de datos.

Descripción general

El caso de uso inicia cuando el usuario desea guardar una imagen en formato *raster* en una base de datos espacial y termina cuando ha sido completada la acción.

Condiciones de ejecución.

Existe una conexión válida a la base de datos en la que se pretende guardar la imagen.

Casos de prueba.

Sección (SC): Guardar imagen *raster* en base de datos

Tabla 4. Casos de prueba del CUS Guardar imagen *raster* en base de datos

Escenario	Descripción	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	Respuesta del sistema	Flujo central
EC 1.1: Guardar imagen en la base de datos con éxito.	El usuario selecciona el nombre de la base de datos, la contraseña a dicha base de datos, la proyección que tendrá el mapa, el archivo de la imagen, introduce el nombre que tendrá la tabla de la base de datos y da clic en el botón “Aceptar”.	V	V	V	V	V	El sistema inserta la imagen en formato *raster* en la base de datos y emite un mensaje “La imagen ha sido guardada exitosamente”.	Seleccionar la opción “Raster” de la barra de herramientas de GeneSIG. Seleccionar la pestaña “Guardar en BD”. Insertar los datos correspondientes. Clic en el botón “Aceptar”.
		prueba	postgres	4326	/.../subir.jpeg	prueba1		
EC 1.2: Guardar imagen en base de datos dejando campos vacíos.	El usuario no introduce todos los datos correspondientes y da clic en el botón “Aceptar”.	I	I	V	I	I	El sistema señala en rojo los campos que no deben ser vacíos.	Seleccionar la opción “Raster” en la barra de herramientas de GeneSIG. Seleccionar la pestaña “Guardar en BD”. Clic en el botón “Aceptar”.
		(vacío)	(vacío)	4326	(vacío)	(vacío)		

EC 1.3: Guardar imagen en base de datos seleccionando un fichero con un formato no válido.	El usuario introduce todos los datos, seleccionando un fichero con un formato no válido y da clic en el botón “Aceptar”.	V prueba	V postgres	V 4326	I /.../video.avi	V prueba2	El sistema emite un mensaje “El formato del fichero no es válido”.	Seleccionar la opción “Raster” en la barra de herramientas de GeneSIG. Seleccionar la pestaña “Guardar en BD”. Seleccionar un fichero cuyo formato sea diferente de 'jpeg', 'png' o 'tif'. Insertar los restantes datos correctamente. Clic en el botón “Aceptar”.
EC 1.4: Guardar imagen en base de datos con un nombre de tabla con caracteres especiales.	El usuario introduce todos los datos, escribiendo el nombre de la tabla con caracteres especiales y da clic en el botón “Aceptar”.	V prueba	V postgres	V 4326	V /...subir.jpeg	I prueba*1	El sistema emite un mensaje “El nombre de la tabla no es válido”	Seleccionar la opción “Raster” en la barra de herramientas de GeneSIG. Seleccionar la pestaña “Guardar en BD”. Insertar un nombre para la tabla que contenga caracteres especiales. Insertar los restantes datos correctamente. Clic en el botón

								"Aceptar".
EC 1.5: Cancelar la opción de guardar la imagen en la base de datos.	El usuario da clic en el botón "Cancelar".	N/A	N/A	N/A	N/A	N/A	El sistema reinicia todos los campos del formulario.	Seleccionar la opción "Raster" en la barra de herramientas de GeneSIG. Seleccionar la pestaña "Guardar en BD". Clic en el botón "Cancelar".

Descripción de las variables

Tabla 5. Variables del CUS Guardar imagen *raster* en base de datos

No	Nombre de campo	Clasificación	Valor nulo	Descripción
1	Conexión a la BD	comboBox	No	Permite seleccionar la base de datos donde desea guardar la imagen.
2	Contraseña de la BD	textField	No	Permite introducir la contraseña para conectarse a la base de datos seleccionada.
3	Proyección	comboBox	No	Permite seleccionar la proyección que tendrá el mapa que contiene la imagen.
4	Dirección del fichero	fileUploadField	No	Permite seleccionar la imagen a guardar en la base de datos.
5	Nombre de la tabla	textField	No	Permite introducir el nombre de la tabla que se creará para almacenar la imagen en la base de datos.

3.6.4. Resultados de las pruebas

La gráfica presentada a continuación muestra los resultados obtenidos de acuerdo a la cantidad de no conformidades detectadas al aplicar las pruebas en cada una de las iteraciones realizadas. En todos los casos estas no conformidades corresponden a la interfaz. Una vez concluido el proceso de validación y verificación se comprobó que el módulo cumple con las especificidades planteadas al inicio de la investigación, logrando obtener los resultados esperados.

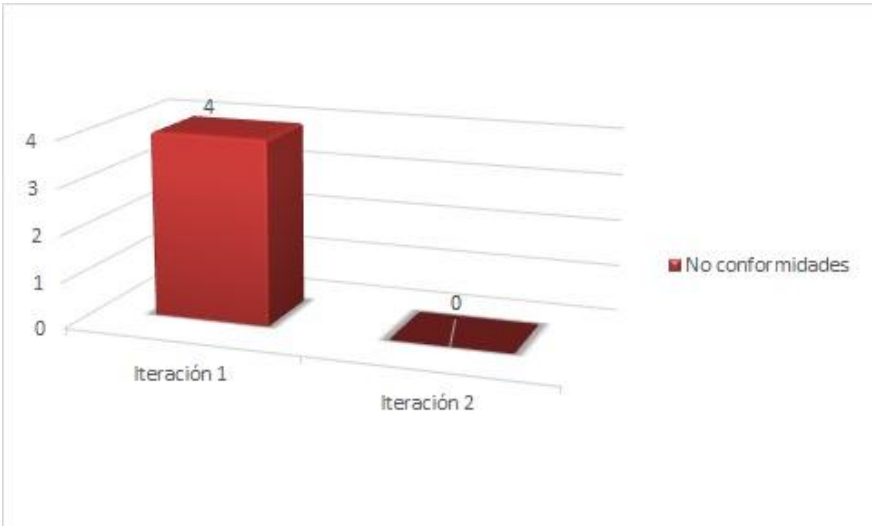

Figura 15. Resultados de las pruebas realizadas al módulo

3.7 Conclusiones parciales

Los artefactos generados en el presente capítulo constituyen la base en el desarrollo del módulo *raster* para la plataforma GeneSIG. La utilización de una arquitectura orientada a objetos y basada en componentes garantiza la integración del módulo con la plataforma sin alterar el correcto funcionamiento de los restantes componentes. El proceso de validación y verificación llevado a cabo garantiza que la propuesta de solución desarrollada realice el correcto almacenamiento, edición, georreferenciación y exportación de la información en formato *raster* en la plataforma.

Conclusiones generales

Mediante la presente investigación se ha logrado desarrollar un módulo para la plataforma GeneSIG que favorece el tratamiento de información en formato *raster* a partir de su almacenamiento en una base de datos espacial, así como su edición, georreferenciación y exportación, siendo este el objetivo principal de dicha investigación. El estudio de los principales conceptos relacionados con la información en formato *raster* y los sistemas que la manejan permitió sentar las bases para el desarrollo del módulo. Los artefactos generados durante el proceso de desarrollo de software proveen la documentación necesaria para futuras actualizaciones y desarrollo de nuevas funcionalidades. Las herramientas y tecnologías utilizadas en el desarrollo del módulo obedecen a criterios de selección de tecnologías libres y multiplataforma, adecuándose a las políticas que impulsan la universidad y el país. El diseño y ejecución de las pruebas de caja blanca y caja negra permitió comprobar el correcto funcionamiento del componente.

Recomendaciones

Agregar a la plataforma GeneSIG módulos de análisis de terreno en cuanto a pendientes, orientación y relieve.

Referencias bibliográficas

Barchini, Graciela Elisa. 2005. *Métodos "I + D" de la Informática.* Santiago del Estero : s.n., 2005. Vol. 2(5).

Black, Rex. 2009. *Managing the Testing Process: Practical Toolsand Techniques for Managing Software and Hardware.* Third Edition. Indianapolis : Wiley Publishing, Inc., 2009. ISBN: 978-0-470-40415-7.

Domínguez Bravo, Javier. 2000. *Breve Introducción a la Cartografía y a los Sistemas de Información Geográfica.* Madrid : CIEMAT, 2000.

Drake, José M. y López, Patricia. 2009. *Verificación y Validación.* 2009.

Esri. 2014. ArcGIS Platform. [En línea] 2014. [Citado el: 14 de noviembre de 2014.] http://help.arcgis.com/es/arcgisdesktop/10.0/help/index.html#/na/00r900000092000000/.

IEEE. 2000. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* 2000. ISBN 0-7381-2519-9.

InterActiveCorp. 2014. Dictionary.com. [En línea] 2014. [Citado el: 21 de noviembre de 2014.] http://dictionary.reference.com/browse/software+platform.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison-Wesley, 2000.

Juristo, Natalia, Moreno, Ana M. y Vegas , Sira. 2005. *Técnicas de evaluación de software.* 2005.

OGC. 2014. OGC Official Site. [En línea] 2014. [Citado el: 14 de noviembre de 2014.] http://www.opengeospatial.org/ogc/glossary.

Olaya, Víctor. 2010. *Sistemas de Información Geográfica.* 2010.

Parodis, Denyse y Comas, Yassel. 2012. *Manual de Usuario Plataforma soberana para el desarrollo de Sistemas de Información Geográfica GENESIG.* La Habana : s.n., 2012.

Phillips, Tony. 2010. Ciencia@NASA. [En línea] 2010. [Citado el: 13 de abril de 2015.] ciencia.nasa.gov/ciencias-especiales/15jan_warming/.

Pitney Bowes. 2014. MapInfo. [En línea] 2014. [Citado el: 14 de noviembre de 2014.] http://www.mapinfo.com/.

Pressman, Roger S. 2000. *Ingeniería del Software. Un enfoque práctico.* Quinta Edición. Madrid : McGraw-Hill, 2000.

—. 2005. *Ingeniería del Software. Un enfoque práctico.* Sexta Edición. s.l. : McGraw-Hill, 2005. pág. 900. ISBN 9701054733.

QGIS. 2014. QGIS. [En línea] 2014. [Citado el: 14 de noviembre de 2014.] http://qgis.org/es/site/#.

Reynoso, Carlos y Kiccillof, Nicolás. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2004.

Sommerville, Ian. 2005. *Ingeniería del Software.* Séptima Edición. Madrid : PEARSON EDUCACIÓN, 2005. ISBN 84-7829-074-5.

Varen Caballero, Eliani. 2012. *Estrategia para la implementación de Sistemas de Información Geográfica del petróleo sobre la base de la plataforma GeneSIG.* La Habana : s.n., 2012. Trabajo de diploma.

Warmerdam, Frank. 2008. The geospatial data abstraction library. *Open Source Approaches in Spatial Data Handling.* s.l. : Springer Berlin Heidelberg, 2008.

Bibliografía

Barchini, Graciela Elisa. 2005. *Métodos "I + D" de la Informática.* Santiago del Estero : s.n., 2005. Vol. 2(5).

Black, Rex. 2009. *Managing the Testing Process: Practical Toolsand Techniques for Managing Software and Hardware.* Third Edition. Indianapolis : Wiley Publishing, Inc., 2009. ISBN: 978-0-470-40415-7.

Bloomfield, Robin, y otros. 2005. *Validation, Verification and Certification of Embedded Systems.* 2005. ISBN 92-837-1146-7.

Domínguez Bravo, Javier. 2000. *Breve Introducción a la Cartografía y a los Sistemas de Información Geográfica.* Madrid : CIEMAT, 2000.

Doxygen. 2015. GDAL - Geospatial Data Abstraction Library. [En línea] 2015. http://www.gdal.org/.

Drake, José M. y López, Patricia. 2009. *Verificación y Validación.* 2009.

Esri. 2014. ArcGIS Platform. [En línea] 2014. [Citado el: 14 de noviembre de 2014.] http://help.arcgis.com/es/arcgisdesktop/10.0/help/index.html#/na/00r900000092000000/.

—. 2014. ArcGIS Resource Center. [En línea] 2014. [Citado el: 21 de noviembre de 2014.] http://www.esri.com/software/arcgis.

Frederick, Shea, Ramsay, Colin y Blades, Steve. 2008. *Learning Ext JS.* Birmingham : Packt Publishing, 2008. ISBN 978-1-847195-14-2.

Garcia, Jesus D. 2009. *ExtJS in action.* s.l. : Manning Publications, 2009.

Hernández, Roberto, Fernández, Carlos y Baptista, Pilar. 2006. *Metodología de la investigación.* Cuarta Edición. s.l. : McGraw-Hill, 2006. ISBN 970-10-5753-8.

Hernández, Rolando Alfredo y Coello, Sayda. 2011. *El proceso de investigación científica.* Ciudad de La Habana : Editorial Universitaria, 2011. ISBN 978-959-16-.

IEEE. 2000. *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.* 2000. ISBN 0-7381-2519-9.

InterActiveCorp. 2014. Dictionary.com. [En línea] 2014. [Citado el: 21 de noviembre de 2014.] http://dictionary.reference.com/browse/software+platform.

Issi, Lázaro. 2002. *JavaScript.* Madrid : Ediciones Anaya Multimedia, 2002.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : Addison-Wesley, 2000.

Juristo, Natalia, Moreno, Ana M. y Vegas , Sira. 2005. *Técnicas de evaluación de software.* 2005.

Larman, Craig. 2003. *UML y Patrones.* Segunda Edición. s.l. : Prentice Hall, 2003.

Lores, Linet y Monné, Diana. 2009. *Aplicación de las pruebas de liberación al SIGM.* UCI. La Habana : s.n., 2009.

OGC. 2014. OGC Official Site. [En línea] 2014. [Citado el: 14 de noviembre de 2014.] http://www.opengeospatial.org/ogc/glossary.

Olaya, Víctor. 2010. *Sistemas de Información Geográfica.* 2010.

Parodis, Denyse y Comas, Yassel. 2012. *Manual de Usuario Plataforma soberana para el desarrollo de Sistemas de Información Geográfica GENESIG.* La Habana : s.n., 2012.

Phillips, Tony. 2010. Ciencia@NASA. [En línea] 2010. [Citado el: 13 de abril de 2015.] ciencia.nasa.gov/ciencias-especiales/15jan_warming/.

PHP Group. 2014. PHP: Hypertext Preprocessor. [En línea] 2014. http://php.net/.

Pitney Bowes. 2014. MapInfo. [En línea] 2014. [Citado el: 14 de noviembre de 2014.] http://www.mapinfo.com/.

Pressman, Roger S. 2000. *Ingeniería del Software. Un enfoque práctico.* Quinta Edición. Madrid : McGraw-Hill, 2000.

—. 2005. *Ingeniería del Software. Un enfoque práctico.* Sexta Edición. s.l. : McGraw-Hill, 2005. pág. 900. ISBN 9701054733.

QGIS. 2014. QGIS. [En línea] 2014. [Citado el: 14 de noviembre de 2014.] http://qgis.org/es/site/#.

Reynoso, Carlos y Kiccillof, Nicolás. 2004. *Estilos y Patrones en la Estrategia de Arquitectura de Microsoft.* 2004.

Sommerville, Ian. 2005. *Ingeniería del Software.* Séptima Edición. Madrid : PEARSON EDUCACIÓN, 2005. ISBN 84-7829-074-5.

Varen Caballero, Eliani. 2012. *Estrategia para la implementación de Sistemas de Información Geográfica del petróleo sobre la base de la plataforma GeneSIG.* La Habana : s.n., 2012. Trabajo de diploma.

Warmerdam, Frank. 2008. The geospatial data abstraction library. *Open Source Approaches in Spatial Data Handling.* s.l. : Springer Berlin Heidelberg, 2008.

Glosario de términos

Cartografía: Técnica de trazar mapas o cartas geográficas. Ciencia que estudia los mapas y cartas geográficas y cómo realizarlos.

Sistema de Información Geográfica (SIG): sistema que integra tecnología informática, personas e información geográfica, y cuya principal función es capturar, analizar, almacenar, editar y representar datos georreferenciados.

SIG vectorial: un sistema vectorial es aquel en el que el territorio se representa a partir de vectores, éstos se localizan en el espacio mediante pares de coordenadas coincidentes con su origen y destino.

SIG *raster*: aquel que realiza sus cálculos a través de una estructura matricial, en la que cada celda o píxel tiene un valor y una localización determinados.

Base de datos espacial: es un sistema administrador de bases de datos que maneja datos existentes en un espacio o datos espaciales.

Rasterización: es el proceso por el cual una imagen descrita en un formato gráfico vectorial se convierte en un conjunto de píxeles o puntos para ser desplegados en un medio de salida digital, como una pantalla de computadora, una impresora electrónica o una Imagen de mapa de bits (bitmap).

Geoportal: Sitio de internet o equivalente, que presta servicios de proveedor de acceso a los servicios de información geográfica.

Georreferenciación: es la técnica de posicionamiento espacial de una entidad en una localización geográfica única y bien definida en un sistema de coordenadas y datum específicos. Es una operación habitual dentro de los Sistemas de Información Geográfica (SIG) tanto para objetos *raster* como para objetos vectoriales.

Ortofotografía: es una presentación fotográfica de una zona de la superficie terrestre, en la que todos los elementos presentan la misma escala, libre de errores y deformaciones, con la misma validez de un plano cartográfico.

Dataset: es una colección de datos que contiene las unidades de datos individuales organizadas en una específica.

Geodatabase: Modelo que permite el almacenamiento físico de la información geográfica, ya sea en archivos dentro de un sistema de ficheros o en una colección de tablas en un Sistema Gestor de Base de Datos.