Universidad de las Ciencias Informáticas

Facultad 6

HERRAMIENTA PARA LA GESTIÓN DEL FICHERO *MAPFILE* DE *MAPSERVER* 6.4

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

Autores:

María del Carmen Quesada Fuentes

Teresita Aurora Remón Mantilla

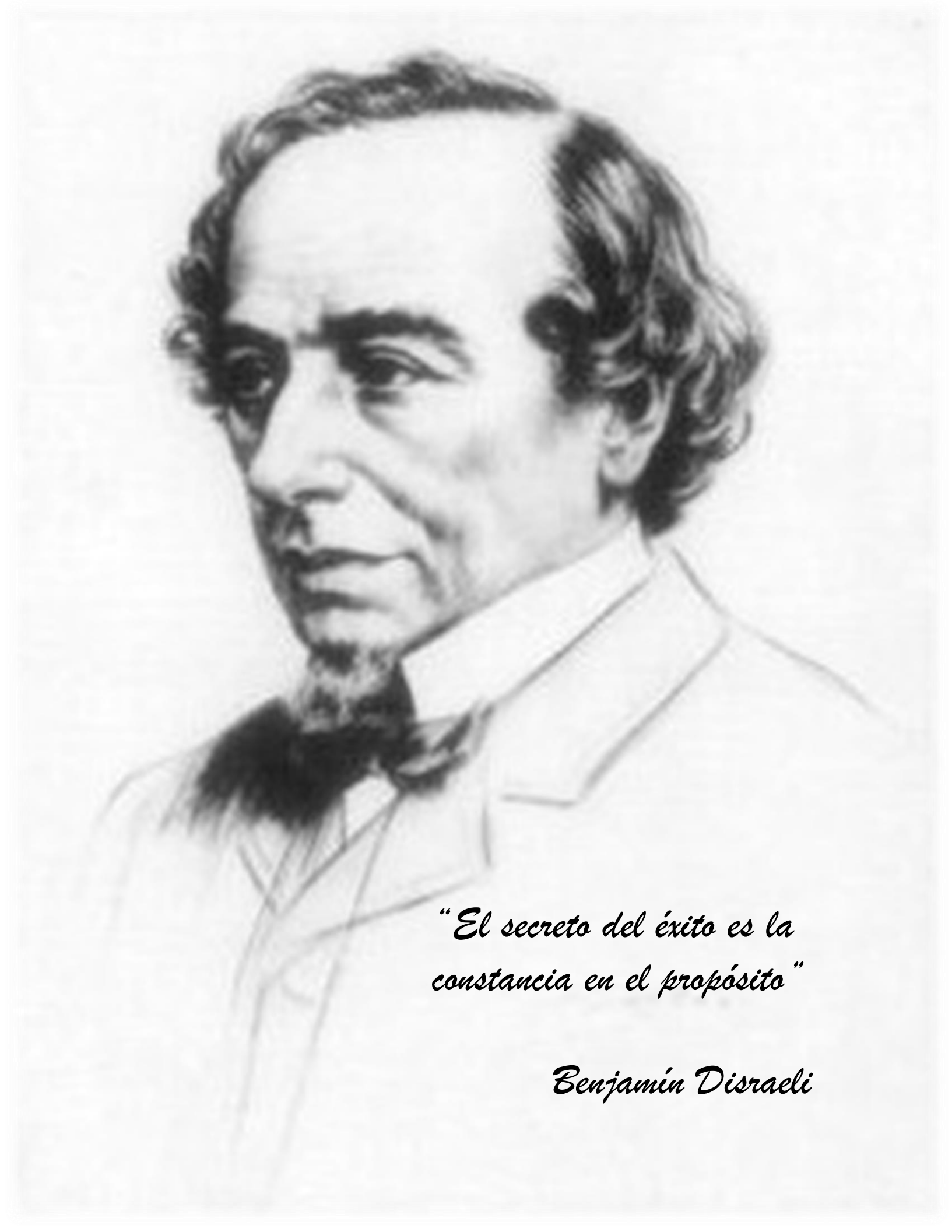
Tutores:

MSc. Grethell Castillo Reyes

Ing. Alejandro O. Hernández Cebrián

Ing. Yoan Mandina Verdecia

La Habana, julio de 2016
"Año 58 de la Revolución"



"El secreto del éxito es la constancia en el propósito"

Benjamín Disraeli

Declaración de Autoría

Declaro por este medio que nosotras, María del Carmen Quesada Fuentes y Teresita Aurora Remón Mantilla, somos las autoras de este trabajo y que autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste, firmamos la presente declaración jurada de autoría en La Habana a los _____ días del mes de _____ del año **2016.**

Teresita Aurora Remón Mantilla
Autor

_____

María del Carmen Quesada Fuentes
Autor

MSc. Grethell Castillo Reyes
Tutor

_____

Ing. Alejandro Orgelio Cebrián Hernández
Tutor

Ing. Yoan Mandina Verdecia
Tutor

Datos de Contacto

Tutores

Nombres: Grethell

Apellidos: Castillo Reyes

Centro de trabajo: Universidad de las Ciencias Informáticas

Correo electrónico: gcreyes@uci.cu

Año de graduación: 2012

Título de la especialidad: Máster en Informática Aplicada

Nombres: Alejandro Orgelio

Apellidos: Hernández Cebrián

Centro de trabajo: Universidad de las Ciencias Informáticas

Correo electrónico: aohernandez@uci.cu

Año de graduación: 2009

Título de la especialidad: Ingeniería en Ciencias Informáticas

Nombres: Yoan

Apellidos: Mandina Verdecia

Centro de trabajo: Universidad de las Ciencias Informáticas

Correo electrónico: ymandina@uci.cu

Año de graduación: 2014

Título de la especialidad: Ingeniería en Ciencias Informáticas

Dedicatoria

Teresita

Les dedico el resultado de mi voluntad y perseverancia para lograr mi objetivo de estudiar esta carrera.

A la persona que constituye el motor que impulsa a mi familia mi Mamá ***María Elena Mantilla Prendes*** *sin su educación no habría llegado hasta aquí.*

A mi Papá ***Rolando Remón González*** *por ser ejemplo y sacrificio en todo momento.*

María del Carmen

Le dedico esta tesis al sol que iluminó mis días hasta el último momento en que sus rayitos de luz no pudieron seguir alumbrando mi vida desde cerca, esa eres tu mi Abuelita ***Carmen****, eres el ser más especial de mi vida y en cualquier lugar de allá arriba que te encuentres sé que estas mirándome, guiándome y principalmente muy orgullosa de mi, por llegar hacerme algo en esta vida ,espero q te sientas feliz de tu nieta te amo.*

Agradecimientos de Teresita

Agradezco infinitamente a las personas que más amo en la vida, **mis padres**, María Elena y Rolandito, por darme todo el amor, comprensión, ayuda, preocupación y su apoyo incondicional, sin Uds. no hubiera llegado hasta donde estoy, cada logro que he alcanzado se los debo a Uds., y ¿con quienes mejores para compartir este triunfo?, si no es con Uds. dos.

A mi tía Regla que aun estando lejos siempre está al tanto de mis estudios, por darme ánimos para seguir, por brindarme su apoyo incondicional para que pueda estudiar y aprender a valorar el sentido verdadero de las cosas y aunque no pudo estar presente hoy sé que se siente orgullosa de sobrina preferida y la única.

Le agradezco eternamente a mi tío Lázaro que ya no está presente pero siempre me apoyó incondicionalmente para que llegara a donde estoy ahora, yo sé que desde allá arriba me sigues cuidando.

Agradezco especialmente a Jorge y Papito que se convirtieron en mis segundos papás, me han dado el cariño y la ayuda que no me imaginé, por estar pendientes de mí en los momentos más duros y difíciles y decirme "mi niña no te preocupes que tú te vas a graduar".

A mis primos de Cienfuegos y de Matanzas que todas las semanas preguntan y están al tanto de mi tesis y como me ha ido en estos años.

Quiero agradecerles a mis primos Tere, Papo y Peter que tanto han hecho por mí desde niña y ahora más en mi carrera, siempre me han alentado para que me haga toda una profesional.

A la gente de mi adorado barrio, los que siempre todos los fines de semana me preguntan cómo voy con la tesis. El chino por decir que soy una hija más de él y estar al tanto de mí, a mi amiga Yoya que es como mi hermana mayor siempre preocupada por mí y dándome

chucherías para que su flaquita no pase hambre, a mi amiga Evelyn dándome consejos para mi exposición. Gracias a mis choferes Rudy y Pedro que tantos viajes me han hecho hasta aquí y hasta a cargar bultos los he puesto

Durante estos años he conocido personas muy agradables a las cuales estoy agradecida, algunos los conozco de antes y otros ya casi terminando, como son tantos el documento completo no me alcanzaría para mencionarlos a todos, pero no se me pueden quedar sin mencionar a mi amigo Carlitos que siempre ha estado ahí para ayudarme en los momentos míos de crisis y que lo seguirá estando porque me tiene que aguantar hasta que dios quiera, ya que mi mamá lo adoptó como su hijo. A las chicas donde viví en el 109: Laro, Laurita, Eli, Yane con las que hice una linda amistad que espero que dure hasta siempre, con Uds. pasé los mejores años de la escuela. A mi amiga Yeline por soportarme, pelearme y malcriarme desde 1er año, que era como tener a mi mamá ahí conmigo. A las niñas de mi apto de ahora sobre todo a Dayneluski y Kielam con las que vengo cargando desde 2do, que ya me parece una eternidad y hasta cariño les cogí. A mis machos del aula que viven fastidiándome. A mi compañera de tesis María por ayudarme en todo lo posible.

Gracias a Fifí que ha estado al pendiente de mí desde primer año, ya es como mi familia, le tengo mucho aprecio y cariño y sé que sin tu ayuda me hubiera costado mucho llegar hasta aquí

A Yoan que más que mi tutor se convirtió en un amigo, al que fastidio todos los días, le quemo es móvil cuando no lo veo y hasta malcriado lo tengo con cafecito todas las tardes, gracias por toda la ayuda que nos brindaste en nuestra tesis. A mis tutores Grethell y Alejandro gracias por la ayuda que nos ofrecieron a pesar de lo ocupado que estaban en el proyecto. A nuestro oponente Miguel que nos ayudó bastante en nuestro documento al igual que los miembros del tribunal.

A todos los presentes, los que no pudieron estar y los que se me faltaron por mencionar, Gracias.

Agradecimientos de María

Les agradezco a todas las personas que de una forma u otra fueron capaces de aportar su granito de arena en la tesis y en mi vida

A mi madre Marbellis, sin su amor, paciencia y dedicación, no pudiera caminar por las fronteras de la vida, eres la mejor madre del mundo, además de madre fuiste una gran amiga donde siempre me apoyaste en mis problemas y aun estando lejos de ti en estos años de la universidad día a día nos comunicábamos y esa risa de nosotras estremecía las paredes de ambos lados que no parecía que habláramos como madre e hija sino como grandes amigas que somos, gracias por contarme todos tus secretos y problemas y siempre pensar que yo te ayudaría a resolverlo, un beso te quiero.

A mi padre Pedro, no tengo palabras para describirte, eres la persona más especial que dios ha puesto en mi vida no sabes lo feliz que me hace tener un padre, así como tú, que da la vida por mí y nunca te ha importado las circunstancias de la vida para darme lo que yo quiera. Hoy me siento muy feliz porque sé que estás orgullosa de tu niñita que se ha hecho algo en la vida como tú siempre deseaste y gracias por complacerme en ser tu única niña a la que siempre has mimado y cuidado, gracias por ser hoy lo que soy y eso te le debo a ti papá, eres el mejor papá del mundo eso te lo aseguro te quiero.

A mis hermanos Pedro Luis, Ale gracias por estar todo el tiempo a mi lado desde la infancia, son mis hermanos mayores con los he convivido siempre. gracias por cuidarme y protegerme en esta vida por ser la más pequeña de nosotros, aunque nos fajemos los quiero mucho. A mis hermanos Gonzalo y Yoandri por ser los más pequeños de mis hermanos a los debo cuidar, proteger y mimar los quiero mucho. A mi hermana Roxana no por ser la última eres la menos importante, al contrario, eres la mejor hermana y amiga que tengo, por ser mi única hermanita y la más pequeña siempre te voy apoyar y ayudar. Gracias por andar siempre detrás de mí, sin importarte dejar planteado a tu novio por andar conmigo

cada vacaciones y fin de año que voy, por eso le doy gracias adiós por tenerte como mi hermana te quiero.

A mi abuelito Pedro por ser mi segundo papá, gracias por cuidarme y darme todo en esta vida no sé qué será de mi cuando no existas te amo abuelo.

A Yoan por ser alguien muy especial en mi vida y que nunca voy a olvidar, gracias por todo lo que hiciste por mí, siempre me acompañó en las buenas y en las malas apoyándome con mis trabajos de controles, con mis pruebas finales y principalmente con mi tesis y gracias a él le debo hoy todo mi esfuerzo de seguir adelante en la universidad y ser hoy lo que soy una ¨Ingeniera¨.

A mi compañera de tesis Teresita por toda su dedicación y realización en este trabajo gracias por tu colaboración, ayuda y por aguantarme todos los días.

A mis tutores Grethell y Alejandro, oponente Miguel y miembros del tribunal por todo su apoyo, confianza y ayuda que hicieron posible la culminación de este trabajo.

A mis tíos del alma Emilia, Ariel, Maribel, Francisco, Diosnelis por toda su preocupación por que yo fuese alguien en esta vida.

A mis primos Arianna, Idalmis, Geikel, Jarol, Franciquito, Jorge David, Geikel Ariel, Anthony y Nicol, por ser los mejores primos del mundo y con los que he compartido toda mi infancia y adolescencia los quiero.

Yanara más que mi prima eres mi amiga gracias por compartir algunos de estos fines de semana a tu lado y gracias por confiar tus secretos en mí, eres muy especial te quiero

Eddian gracias por cada fin de semana que estaba en tu casa me hiciste reír tanto, con esas ocurrencias tuyas q no tenían cuando parar te quiero.

Agradecimientos

A mis segundos padres aquí en la habana Diana y Pury, que me apoyaron y me tuvieron como una hija más en sus vidas, dándome buenos consejos en mis malos y buenos momentos que tuve.

A mi novio José porque más que novio hizo función de un padre, donde día a día me enseñaba algo nuevo de la vida y siempre me apoyo con mis estudios, gracias por hacerme tan feliz en tan poco tiempo.

A mis amigas más cercanas Eliza, Dayana con las que he compartido buenos y malos momentos en la universidad Eliza gracias por esos espaguetis deliciosos que hacías a cada rato y ese Facebook que la mayoría de mis fotos son contigo, besos amigas, y Dayana, aunque no lo creas te aprecio mucho, no importa los momentos malos que tuvimos, sino que esa amistad estaba inculcada en nuestros corazones y no se pudo destruir, te quiero.

A mi mejor amiga y hermana Ivonne gracias amiga por aparecer en mi vida y acompañarme en mis buenos y malos momentos sé que nuestra amistad es verdadera, somos como dice mi novio la soga y el caldero, gracias por ser mi amiga y espero que esta amistad no termine aquí y que perdure por siempre, sabes más que una amiga tienes un a hermano te quiero.

A Yuya gracias por estar a mi lado cuando te necesité, eres un buen pañuelo que toda persona quisiera tener a su lado en los malos momentos, gracias amiga. A mis amigos Eddy, Dennis, Darian, Manuel por ser los compañeros de aula más magníficos que tuve, gracias por acompañarme a estudiar y compartir la mayoría de las fiestas conmigo, los quiero mucho. A mis compañeros de aula felicidades a todos porque llegamos a cumplir el sueño de todo universitario. A las niñas de mi apto Claudia, Eliza, Flavia y Cano con las que he convivido todo este tiempo y tuvimos buenos y malos momentos juntos las extrañaré.

A la mejor tía de la UCI Migdalia, gracias por estar a mi lado todo el tiempo mimándome como una hija más, yo sé que en el día si no me veías me extrañabas muchas gracias por soportar mis malcriadeces, eres mi mamá aquí en la universidad te quiero mucho.

Resumen

El uso de los Sistemas de Información Geográfica (SIG) se ha expandido de forma extraordinaria por todo el mundo y han pasado del total desconocimiento a la práctica cotidiana. En la Universidad de las Ciencias Informáticas (UCI), específicamente en el centro Geoinformática y Señales Digitales (GEYSED) de la Facultad 6 se desarrolla la aplicación informática GeneSIG, devenida plataforma para la personalización de sistemas de información geográfica, en conjunto con la Empresa de Tecnologías de la Información para la Defensa (XETID) y el Grupo Empresarial GEOCUBA. Para realizar la configuración del mapa, se utiliza un fichero de configuración de *MapServer* denominado *MapFile*, dicho fichero se genera utilizando un subsistema de catálogo de mapas, pero esto solo es posible para la versión 5.6 de *MapServer*. Actualmente el equipo utiliza la versión 6.4 de *MapServer*, que implican la no utilización de la herramienta lo que trae consigo un mayor consumo de tiempo al realizar la generación manual del fichero e incidencias en la planificación y tiempo de respuesta a las necesidades de los clientes. Con la presente investigación se desarrolló la herramienta *ManagerMaps* para la gestión del fichero *MapFile* en SIG basados en *MapServer* 6.4. Para el desarrollo de esta herramienta se utilizó como lenguaje de programación, *JavaScript* y *PHP*, como *framework* de desarrollo, *Symfony* y *Bootstrap*, el Sistema Gestor de Base de Datos (SGBD) *PostgreSQL* y la metodología de desarrollo *AUP*. De la explotación futura de esta herramienta se deriva la importante ventaja de generar automáticamente el fichero de configuración *MapFile* para la versión 6.4 de *MapServer*.

Palabras claves: *MapFile, MapServer,* SIG.

Abstract

The use of Geographic Information Systems (GIS) has been expanded dramatically throughout the world and has moved from total ignorance to everyday practice. In the University of Informatic Sciences (UCI), specifically at the center Geoinformatics and Digital Signal from Faculty 6 is being developed a computer application named GeneSIG, which has become a platform for customizing GIS, altogether with Empresa de Tecnologías de la Información para la Defensa (XETID) and Grupo Empresarial GEOCUBA. To configure the map, a *MapServer* configuration file called *MapFile* is used, in the present file is generated using a map catalog subsystem, but this is only possible for the version 5.6 of *MapServer*. Currently the team uses higher versions involving the non-use of the tool, which brings more time-consuming because of the manual generation of the file and incidents in planning and response time of customer needs. With this research was developed the ManagerMaps tool for managing maps on Geographic Information Systems based on *MapServer* 6.4. For the development of this tool it was used as a programming language, JavaScript and PHP, as a development framework, Symfony and Bootstrap, the Database Management System (DBS) PostgreSQL and AUP development methodology. From the future use of this tool is derived the important advantage of automatically generating the configuration file named *MapFile* for *MapServer* 6.4.

Key Words: GIS, MapFile, MapServer.

Índice de Contenidos

Introducción 1

Capítulo 1: Fundamentación teórica de la solución propuesta. 5

1.1 Introducción 5

1.2 Conceptos asociados al dominio de investigación 5

1.2.1 Partes fundamentales de los SIG 5

1.2.2 Servidores de Mapas 6

1.2.3 Mapa digital 11

1.4 Tecnologías a utilizar 16

1.4.6 Metodología de desarrollo de software 16

1.4.2 Lenguajes de programación 18

1.4.4 Lenguaje unificado de modelado (UML) 21

1.4.5 Herramientas CASE 22

1.4.6 Servidor Web 23

1.4.7 Servidor de mapa 23

1.5 Conclusiones del Capítulo 24

Capítulo 2: Análisis y diseño. 25

2.1 Introducción 25

2.2 Modelo del Dominio 25

2.7 Descripción de la Solución 27

2.3 Requisitos del Software 28

2.3.1 Requisitos funcionales 28

2.3.2 Requisitos no funcionales 30

2.4 Patrón Arquitectónico 31

2.4.1 Patrón Modelo-Vista-Controlador (MVC) 32

2.5 Patrones de diseño 33

2.5.1 GRASP 33

2.5.2 Patrones GOF 35

2.6 Modelo de Datos 35

2.7 Modelo de diseño 37
2.7.1 Diagrama de Clases del Diseño 37
2.7 Conclusiones del capítulo 38
Capítulo 3: Implementación y prueba. 39
3.1 Introducción 39
3.2 Modelo de implementación 39
3.2.1 Diagrama de Componentes 39
2.7 Modelo de Despliegue 40
3.4 Modelo de pruebas 41
3.4.1 Diseño de Casos de Prueba 41
3.4.2 Resultados de las pruebas realizadas 45
3.5 Conclusiones del capítulo 48
Conclusiones 49
Recomendaciones 50
Referencias 51
Bibliografía 54
Anexos 57
Interfaces de la aplicación 57

Índice de Figuras

Fig. 1: Estructura del fichero *MapFile* 8
Fig. 2: Interfaz para crear el fichero *MapFile* en LiberMaps 12
Fig. 3: Interfaz para objeto web de LiberMaps 13
Fig. 4: Interfaz de MapStorer 14
Fig. 5: Interfaz de visualizar mapa de MapStorer 15
Fig. 6: Diagrama del modelo de dominio 26
Fig. 7: Modelo-Vista-Controlador 33
Fig. 8: Diagrama del modelo de dato 36
Fig. 9: Diagrama del modelo de clase del diseño Modificar fichero *MapFile*. 37
Fig. 10: Diagrama de componentes Modificar fichero *MapFile*. 40
Fig. 11: Diagrama de despliegue 41
Fig. 12: Resultados de pruebas de caja negra 46
Fig. 13: Interfaz principal de la aplicación 57
Fig. 14: interfaz para Insertar un nuevo símbolo 58
Fig. 15: Interfaz con los *OUTPUTFORMAT*s existentes. 59
Fig. 16: Contenido del objeto *.map*. 60

Índice de Tablas

Tabla 1: Fases de la metodología AUP 17
Tabla 2: Descripción de las clases del diagrama del modelo de dominio 26
Tabla 3: Caso de prueba Importar fichero *MapFile* 42
Tabla 4: Variables del Caso de prueba Importar fichero *MapFile* 45
Tabla 5: Registro de defectos y dificultades detectados 46

Introducción

A medida que se ha desarrollado la especie humana y su necesidad de comunicación, se han ampliado sus fronteras en todas las ramas de la sociedad. De forma tal que para delimitarlos surge la Geografía como una ciencia del conocimiento. El volumen de información geográfica que se acumula a nivel mundial se hace cada vez mayor, lo que conlleva a utilizar una vía de acceso rápido para buscar los datos deseados por el usuario, con este, y otros fines se desarrollan los Sistemas de Información Geográfica (en lo adelante SIG), enfatizando en la ubicación de la información en el espacio.

Los SIG son los encargados de capturar, almacenar, integrar, manipular, analizar y visualizar todos los datos que están espacialmente referenciados a un área determinada (Sánchez, M, y otros, 2007)

Por la necesidad de tener referenciadas determinadas variables, existe un mayor uso de los SIG, de ahí que desde su creación hasta la actualidad se hayan convertido en una herramienta muy útil. Pueden emplearse en diversas actividades relacionadas con la tecnología de computadoras, para integrar una gran variedad de datos capaces de crear una imagen de la geografía como características de: mapas, medio ambiente y socioeconómicas de una zona de estudio determinada (Carrasco, 2013).

Los SIG son una herramienta multipropósito con aplicaciones en los campos más diversos. Esta tecnología permite gestionar los datos espaciales, y surgió como resultado de la necesidad de disponer rápidamente de información para resolver problemas y contestar a preguntas de modo inmediato. Son usados generalmente en gestión de recursos naturales, medio ambiente, por citar ejemplos (Burrough, y otros, 1998).

En Cuba se llevan a cabo un conjunto de procesos de transformaciones tecnológicas, para esto se estudian tecnologías que se encuentran estrechamente relacionadas con los SIG y su vinculación y adaptación con los softwares libres existentes en el mundo. Existen diferentes instituciones nacionales que contribuyen al crecimiento informático del país y que a la vez se destacan por el uso de los SIG, entre las cuales se encuentran: GEOCUBA, Ministerio de Ciencia Tecnología y Medio Ambiente (CITMA), e incluso en las Fuerzas Armadas Revolucionarias (FAR), cada una de ellas orientadas a esferas como son: la telefonía, la minería, la meteorología, el transporte, la salud, y lugares estratégicos, para situar puntos para la defensa de la patria.

La Universidad de las Ciencias Informáticas (UCI), uno de los centros educacionales de altos estudios, donde se combina el aprendizaje, la investigación y la producción de forma organizada, ayuda al país para

el desarrollo de su economía y en el proceso de informatización. El centro GEYSED de la Facultad 6 de la UCI desarrolla la aplicación informática GeneSIG, devenida plataforma para la personalización de SIG, en conjunto con la XETID y el GEOCUBA.

MapServer es un servidor de mapas de código abierto para la publicación de datos espaciales y aplicaciones cartográficas interactivas para la web. Este posee un fichero de configuración denominado *MapFile* el cual define los recursos que serán utilizados por *MapServer* para la representación del mapa. El equipo de desarrollo cuenta con la herramienta *LiberMaps* para la edición del fichero de configuración de *MapServer*, (el *MapFile*), así como la personalización de la información georreferenciada de acuerdo a la necesidad del usuario. En la actualidad, el equipo de desarrollo experimenta algunas limitaciones en el proceso de implementación que inciden en la planificación y el tiempo de respuesta a las necesidades de los clientes, aún más, en las posibilidades de error y la calidad del producto final obtenido. Esto se debe a que la herramienta LiberMaps está concebida para configurar el fichero *MapFile* sólo para la versión 5.6 de *MapServer*. Actualmente en el proyecto se utiliza la versión 6.4 de *MapServer*, que implica la no utilización de esta herramienta debido principalmente a cambios en la estructura de los objetos y atributos contenidos dentro del *MapFile*, lo que trae como consecuencia un mayor consumo de tiempo al realizar la generación manual de dicho fichero.

De la situación antes expuesta se plantea el siguiente **problema científico a resolver:** ¿Cómo disminuir las limitaciones de tiempo en la gestión del fichero *MapFile* de *MapServer* 6.4?

En función de resolver el problema planteado se propone como **Objetivo general:**

Implementar una herramienta para la gestión del fichero *MapFile* de *MapServer* 6.4.

La investigación tiene como **objeto de estudio:**

La gestión del fichero *MapFile* en *MapServer* y como **campo de acción:** la gestión del fichero *MapFile* de *MapServer* 6.4.

De acuerdo con el objetivo general, se definen una serie de **tareas de la investigación** encaminadas a su exitoso cumplimiento:

- Elaboración del marco teórico – metodológico de la investigación y la introducción de la tesis.

- Identificación de las principales soluciones existentes en el tratamiento del problema planteado mediante la realización de un estudio de los referentes teóricos – prácticos que preceden la realización de esta investigación.
- Argumentación de las herramientas y tecnologías a utilizar en la modelación y desarrollo del sistema.
- Argumentación de la metodología de software a utilizar en el proceso de desarrollo del sistema.
- Análisis y diseño de la propuesta de solución.
- Implementación del sistema.
- Desarrollo del proceso de pruebas de la solución implementada.

Para guiar el proceso de investigación se definen las siguientes **preguntas de investigación:**

¿Cuáles son las características y estructura del archivo de configuración de *MapServer*?

¿Cuáles son las soluciones existentes para generar el archivo *MapFile*?

Con la implementación de la herramienta para la gestión del fichero *MapFile*, ¿será resuelto el problema actual?

Durante el desarrollo de la investigación se decide usar los siguientes **métodos científicos:**

Métodos Teóricos

- **Analítico – Sintético:** Permite la descomposición de un todo complejo en sus partes y cualidades. La síntesis establece la unión entre las partes previamente analizadas y posibilita descubrir relaciones y características generales entre los elementos de la realidad. Este método se emplea para la valoración de soluciones existentes que respondan al campo de acción. Además se utiliza en la selección de las herramientas y tecnologías empleadas en el desarrollo de la solución (Álvarez de Zayas, 1995). Ayudó a sintetizar el análisis realizado a toda la documentación consultada para el desarrollo de la investigación.
- **Histórico - Lógico:** Estudia la trayectoria real de los fenómenos y acontecimientos en el transcurso de su historia. Investiga las leyes generales del funcionamiento y desarrollo de los fenómenos. Este método se utiliza para estudiar la evolución de los conceptos asociados a las herramientas que gestionan el fichero *MapFile* de *MapServer* y permite la definición de términos propios. Se emplea también para estudiar la evolución y las tendencias de las tecnologías y demás herramientas a utilizar (Álvarez de Zayas, 1995). Este método se usó para realizar un estudio de las herramientas encargadas de la gestión del fichero *MapFile* desarrollados anteriormente en el

ámbito nacional e internacional, sirviendo de base en la investigación e implementación de la herramienta desarrollada.

Método Empírico

- **Observación:** Con este método es posible distinguir directamente los hechos de la realidad objetiva. Permite conocer el proceso de limitado como objeto de estudio, lo cual contribuyó a tener un registro visual más detallado de lo que se quiere y hace falta hacer y cómo hay que hacerlo. Se detectaron algunas limitaciones en el proceso de implementación que inciden en la planificación y el tiempo de respuesta a las necesidades de los clientes y como consecuencia un consumo mayor de tiempo al realizar la generación manual del fichero *MapFile*. De utilizar esta variante como única alternativa de solución se genera otra situación adicional: los desarrolladores deben conocer para cada versión de *MapServer*, la estructura sintáctica del fichero *MapFile*.

El trabajo se encuentra distribuido en tres capítulos. En el Capítulo 1 se realiza una fundamentación de la teoría de la investigación. Se describen términos técnicos de importancia y se realiza un estudio de sistemas con características similares a las requeridas. Se concluye este capítulo con las herramientas y tecnologías a utilizar para el desarrollo de la herramienta. Se inicia el Capítulo 2 definiendo las características de la solución propuesta. Se muestra el modelo del dominio, se identifican los requisitos funcionales y no funcionales con los que debe cumplir la herramienta. Se define la arquitectura y se muestran los resultados obtenidos en el desarrollo del proceso del diseño. Por último, en el Capítulo 3 se abordan los temas referentes a la implementación y prueba de la herramienta. Se muestran los diagramas de despliegue y de componentes, así como la descripción de las pruebas realizadas al sistema y los resultados obtenidos.

Capítulo 1: Fundamentación teórica de la solución propuesta.

1.1 Introducción

En el presente capítulo se describe lo que constituye la fundamentación teórica de la solución propuesta que sustenta la presente investigación, la cual está encaminada a lograr una mejor comprensión de la realización de esta indagación, para obtener una herramienta con calidad. Para ello es necesario utilizar tecnologías que agilicen el proceso de desarrollo de software y que estén enfocadas en la obtención de un producto. Se describe la utilización de la metodología de desarrollo de software, lenguajes de programación, herramienta CASE, lenguaje de modelado, *frameworks* y otros, conformando así la propuesta de solución de ésta.

1.2 Conceptos asociados al dominio de investigación

1.2.1 Partes fundamentales de los SIG

Un SIG está compuesto de tres subsistemas fundamentales:

1. Subsistema de datos, que permite la entrada, salida y gestión de datos, así como el acceso a los mismos por parte de los restantes subsistemas.
2. Subsistema de visualización y creación cartográfica, que permite la interacción con los datos mediante mapas, leyendas, así como la edición de dichos datos.
3. Subsistema de análisis, que provee los métodos y procesos para el análisis de los datos.

Además, cuenta con cinco elementos básicos:

1. Datos: son los que propician toda la información geográfica necesaria para la creación de un SIG.
2. Métodos: son las acciones aplicadas sobre los datos, para llevar a cabo el análisis pertinente respecto a los mismos.
3. *Software*: es el instrumento que garantiza la implementación de los métodos y gestiona los datos.
4. *Hardware*: es el equipo necesario para ejecutar el *software*.
5. Personas: es el equipo informático capaz de diseñar y utilizar el *software* (Olaya, 2010).

1.2.2 Servidores de Mapas

Los servidores de mapas son parte importante en el proceso de interacción con los datos espaciales, debido a que permiten a los usuarios visualizarlos y consultar su información geográfica por medio de una aplicación espacial. En la actualidad existe una amplia variedad de servidores de mapas comerciales, de código abierto y de servidores gratuitos; un servidor de mapas es como el motor que permite la visualización de mapas en una página web (Mitchell, 2005).

El servidor de mapas le brinda al usuario la posibilidad de interactuar con la información gráfica constituida por distintas capas de información. Con el objetivo de proporcionar información geoespacial, las organizaciones están optando por un servidor de soluciones innovadoras basadas en los SIG, que proporcionan contenidos y capacidades a través de servicios Web.

Las características principales de un servidor de mapas son las siguientes:

- Servicios Web de SIG que permiten utilizar información georreferenciada (mapas) vía *Web*, siendo ésta la principal, pero no la única funcionalidad del Servidor de Mapas.
- Geoprocesamiento, las herramientas disponibles en forma local (*buffer*, clip, etc.) pueden estar disponibles en un mapa publicado mediante el desarrollo de *software web*.
- Desarrollo de herramientas, al tener la capacidad de desarrollar una interfaz para servidor de mapas, se tiene la posibilidad de realizar cambios en forma dinámica a cada capa, pudiendo agregar o eliminar sus elementos (puntos, líneas, polígonos entre otros).
- Consulta y búsqueda de datos, se puede realizar una búsqueda mediante criterios definidos por el usuario (puntos de muestreo en cierta región, hospedantes correspondientes a una plaga, información histórica con parámetros de tiempo definidos).
- Generación de estadísticas, los datos consultados por el usuario pueden ser representados también en forma de gráficos que permitan realizar un análisis concreto de los mismos.
- Carga y descarga de información georreferenciada, la flexibilidad ofrecida por el servidor de mapas permite realizar la carga de capas que el usuario tenga de forma local, para poder visualizarlas de forma automática dentro del servidor de mapas. También se puede realizar la descarga de la información georreferenciada disponible, de forma local. (Sinavef, 2011).

MapServer

MapServer es uno de los servidores frecuentemente utilizados en su variedad de versiones. Es de código abierto para la publicación de datos espaciales y aplicaciones cartográficas interactivas para la *web*.

Originalmente desarrollado a mediados de los 90's en la Universidad de Minnesota, *MapServer* es publicado bajo una Licencia tipo MIT, y funciona en los principales sistemas operativos (Windows, Linux, Mac OS X). Es un motor de renderización de mapas que funciona en un entorno web. Su propósito es la visualización dinámica de mapas a través de Internet. Permite compartir de manera visual la información geográfica, de forma dinámica y en tiempo real, además de que es posible compartir datos con otras aplicaciones. (MapServer, 2008)

El funcionamiento de *MapServer* se concibe generalmente detrás de un servidor web. Dicho servidor recibe las peticiones de los mapas por parte del cliente, éste las envía a *MapServer*, quien los construye y se los pasa al servidor *web* que es el encargado de mostrar la respuesta al cliente. (MapServer, 2008)

Algunas de sus características principales son:

- Capacidad de ejecutar en varios sistemas operativos (*Windows, Linux, Mac OS X*, etc.
- Soporte para lenguajes de *scripting* populares y entornos de desarrollo (*PHP, Python, Perl, Ruby, Java, .NET*).
- Salida de aplicación completamente personalizable.
- Muchos entornos de aplicaciones de código abierto listas para su uso.

MapFile

El *MapFile* define los recursos que serán utilizados por *MapServer* para la representación del mapa, muestra y consulta de parámetros, también contiene información acerca de cómo se debe dibujar el mapa, la leyenda y el resultado de realizar una consulta. Por tanto, define parámetros de los datos, el despliegue y las consultas, que serán usados en una aplicación con *MapServer*, se puede hablar del fichero *MapFile* como un archivo de configuración que tiene extensión *".map"*. Es el archivo principal desde donde *MapServer* obtiene el tamaño del mapa, las capas que va a representar, la simbología y cómo se verá el mapa en general. Este archivo de configuración tiene varias secciones que conforman el mapa en general. Cada sección comienza con el nombre de la sección y culmina con la etiqueta *END*. Dentro de cada sección se especifican determinados parámetros en la forma ATRIBUTO - VALOR que puede ser o no de caracteres nulo. (Espinosa A, 2008)

Estructura del fichero *MapFile* (MapServer, 2008):

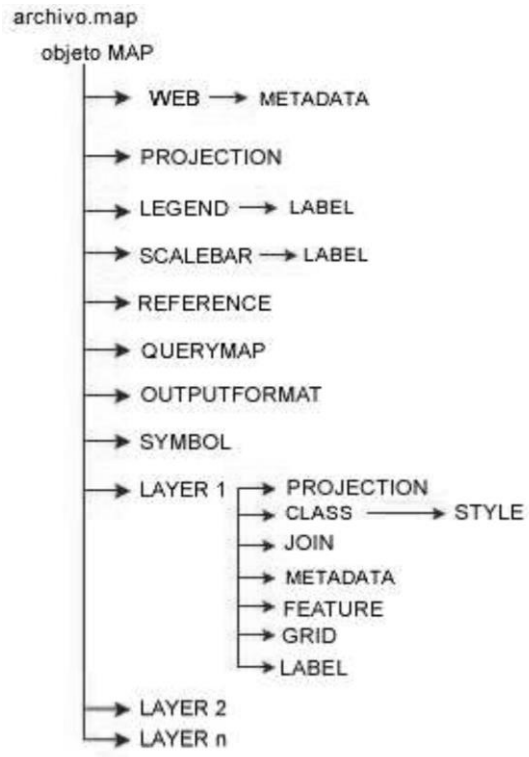


Fig. 1: Estructura del fichero *MapFile*

- Objeto *MAP*

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

- Objeto *WEB*

Este objeto define cómo operará la interfaz *web* con respecto a la representación del mapa, o sea, es el objeto que permitirá visualizar la imagen creada por *MapServer* insertándola en una página *web*. Consta de alguna de las siguientes propiedades: *HEADER, TEMPLATE, FOOTER, MINSCALE, MAXSCALE.*

- Objeto *PROJECTION*

Define la proyección de los mapas y las capas que el servidor generará.

- Objeto *LEGEND*

A partir de las propiedades que se definen en esta sección *MapServer* es capaz de generar la leyenda del mapa. La leyenda es una imagen cuyo formato depende del formato definido para la creación del mapa.

La sección contiene las siguientes propiedades: *IMAGECOLOR, POSITION, STATUS, FONT, BUFFER, MINDISTANCE, PARTIALS.*

- Objeto *SCALEBAR*

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

- Objeto *LABEL*

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

- Objeto *REFERENCE*

El conjunto de propiedades que se definen en esta sección permiten conformar el mapa de referencia, mapa que comprende la extensión total de la zona que incluirá el servicio *WMS*. Las propiedades que contiene son: *IMAGE, EXTENT, SIZE, STATUS, MARKER, MARKERSIZE, MINBOXSIZE, COLOR, OUTLINECOLOR.*

- Objeto *QUERYMAP*

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

- Objeto *OUTPUTFORMAT*

Define los formatos de salida disponibles, incluye formatos como P*NG13, GIF14, JPEG15 y SWF16.* Contiene las siguientes propiedades: *NAME, DRIVER, IMAGEMODE, MIMETYPE, EXTENSION, TRANSPARENT, FORMATOPTION.*

- Objeto *SYMBOL*

Esta sección define un símbolo del mapa. Un símbolo es un caracter cartográfico configurable a través de la modificación de los atributos que lo componen, puede declararse tanto en la propia estructura de un fichero *MapFile* o como un archivo independiente. Las propiedades que lo componen son: *NAME, ANTIALIAS, CHARACTER, TRASNPARENT, POINTS, FILLED, FONT, IMAGE, ANCHORPOINT.*

- Objeto *LAYER*

Define las propiedades de las capas del mapa que se generará. Es el objeto más utilizado en el fichero *MapFile*, describe las capas que se utilizarán para la representación del mapa. Contiene los objetos *CLASS, METADATA, PROJECTION, JOIN, GRID, FEATURE* y *LABEL* y además las siguientes propiedades: *NAME, GROUP, TYPE, TYPERASTER, DATA, CONNECTIONTYPE, LABELITEM, HEADER, FOOTER, TRANSPARENCY, TOLERANCE*.

- Objeto *CLASS*

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

- Objeto *STYLE*

Determina un conjunto de propiedades específicas para un objeto *CLASS*. Una clase puede tener varios estilos. Contiene las siguientes propiedades *LABEL, ANGLE, BACKGROUNDCOLOR, BACKGROUNDSHADOWCOLOR, COLOR, FONT, FORCE, MAXSIZE, MINSIZE, MINDISTANCE, OFFSET, PARTIAL, POSITION, SHADOWSIZE, SIZE, TYPE, IMAGECOLOR, STATUS, SYMBOLSET, BUFFER, LINEJOIN, PATTERN, GAP, LINECAP, LJMAXSIZE, INITIALGAP.*

- Objeto *METADATA*

Es utilizado por el objeto *MAP* para indicar los metadatos en general del servicio y por el objeto *LAYER* para indicar metadatos específicos para cada capa de información, con estos metadatos se confecciona el archivo de capacidades.

Cambios hasta la versión 6.0

- En la versión 6.0 los parámetros relacionados con el estilo se trasladaron desde el elemento *SYMBOL* hacia el elemento *STYLE* de *CLASS*, estos son:
  - *PATTERN* (anteriormente llamado *STYLE*), *GAP, LINECAP, LINEJOIN, LINEJOINMAXSIZE*
- En la versión 6.0 no está disponible el elemento *SYMBOL TYPE* de la versión 5.6 porque ya no es necesario.
- La etiqueta *VALIDATION* será reconocida en la versión 6.0

Cambios hasta la versión 6.4

- Se añadió el parámetro *ANCHORPOINT* a *SYMBOL*.
- Se añadió el parámetro *INITIALGAP* a *STYLE*.
- El comportamiento del parámetro *GAP* de *STYLE* fue modificado para especificar el centro de separación central.

1.2.3 Mapa digital

El concepto de mapa proviene del término latín *mappa*. Se trata de una representación gráfica y métrica de una porción de territorio sobre una superficie bidimensional, que por lo general suele ser plana, aunque también puede ser esférica como en el caso de los globos terráqueos (García y Bellido, 2012). Estas características se suman a que el mapa puede ser además analógico (cuando es impreso sobre papel) o digital (codificado en cifras, almacenado en un ordenador y presentado en una pantalla). Es la representación de cualquier estructura de datos usada para reflejar cartográficamente una variable espacial, nominal o cuantitativa, independientemente del modelo de datos utilizado que puede ser vectorial o raster. (BELMONTE 2009).

Un mapa digital está formado típicamente por capas, por ejemplo, la representación de los ríos, divisiones territoriales, asentamientos y carreteras pueden ser representadas en capas individuales.

Un mapa digital es un conjunto de datos que representan información espacial y atributos, almacenados en el ordenador. Constituye el almacenamiento de información espacial como dibujos electrónicos hechos a base de elementos gráficos sencillos (líneas, puntos, círculos, etc.) organizados en capas, con el objetivo de una salida impresa o por pantalla.

1.3 Soluciones existentes

El acceso al conocimiento geográfico más fácil y disponible, ha encauzado a muchos especialistas del mundo a buscar alternativas para mejorar las tecnologías SIG y con ello el tratamiento para gestionar los mapas. La forma de crear, mostrar, acceder, compartir e interactuar con cada uno, está en dependencia de las características específicas de cada sistema.
Es de interés conocer durante la investigación, la capacidad que tienen algunas herramientas de realizar estas operaciones, por lo que es necesario realizar un análisis de las propiedades de estas, y las ventajas que proporcionan su uso. A continuación, se hace referencia a algunas herramientas para la gestión de mapas, que se utilizan en la actualidad:

- **LiberMaps**

LiberMaps se concibe como un producto con autonomía suficiente para funcionar de manera independiente a un SIG, pero que al mismo tiempo cuenta con los protocolos necesarios para comunicarse con sistemas externos y brindar servicios que permitan personalizar tanto los datos como las funcionalidades para cada usuario. El objetivo de integrar el catálogo de mapas con un SIG (en este caso

GeneSIG), es lograr la personalización del SIG de acuerdo a los usuarios, teniendo en cuenta el perfil de cada uno, controlando el acceso a los datos sensibles (Castillo, R, y otros, 2013).

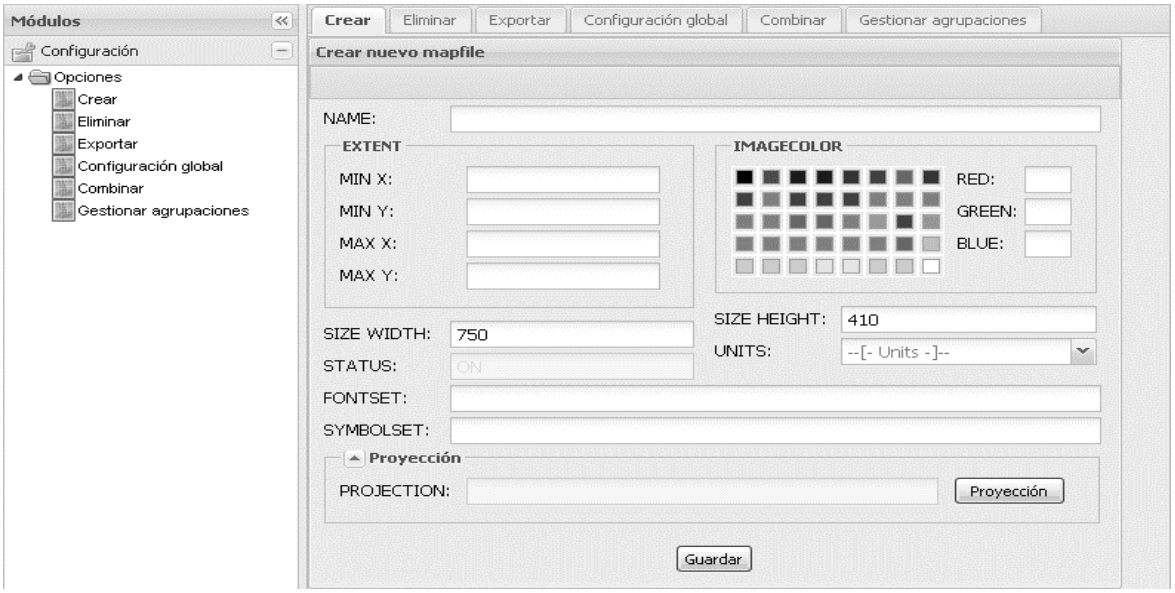


Fig. 2: Interfaz para crear el fichero *MapFile* en *LiberMaps*

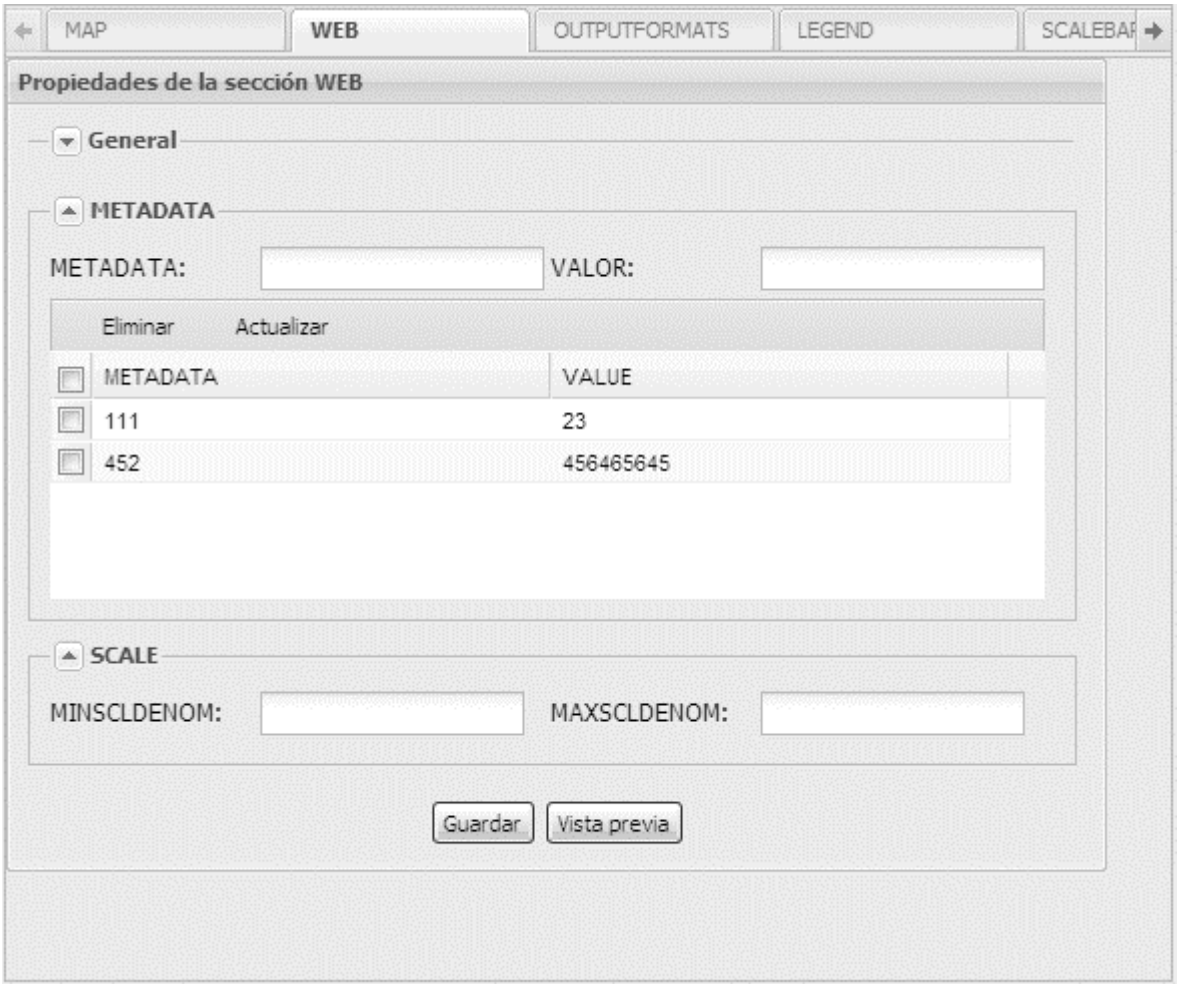


Fig. 3: Interfaz para generar el objeto web de *LiberMaps*

- **MapStorer**

Es una herramienta de código abierto, útil para manejar proyectos de *MapServer*. Permite a los usuarios crear el fichero *MapFile*. Provee funcionalidades que permiten manipularlo, por ejemplo, crear uno nuevo, adicionarle capas, crear clases para las capas, exportar e importar el fichero. Aun (Puebla Martínez, y otros, 2012) así, tiene varias debilidades, no permite la creación y edición del objeto *SYMBOL*, uno de los objetos más importantes del mapa, pues se dice que sin él es como si no existiera.

Características de *MapStorer*

- *MapStorer* es un / Proyecto *Free Software OpenSource*.
- *MapStorer* es un sistema apoyado de base de datos para la gestión de proyectos *MapServer*.
- Las asignaciones de proyectos pueden ser creados y editados a través de un uso fácil de *front-end web*.

- Los archivos de configuración de *MapServer* (*MapFile*), son creados sobre la marcha, con previa visualización oportuna de los proyectos de mapeo creadas y editadas.
- Reduce la edición manual de archivos de texto.
- Organiza el fichero *MapFile*.

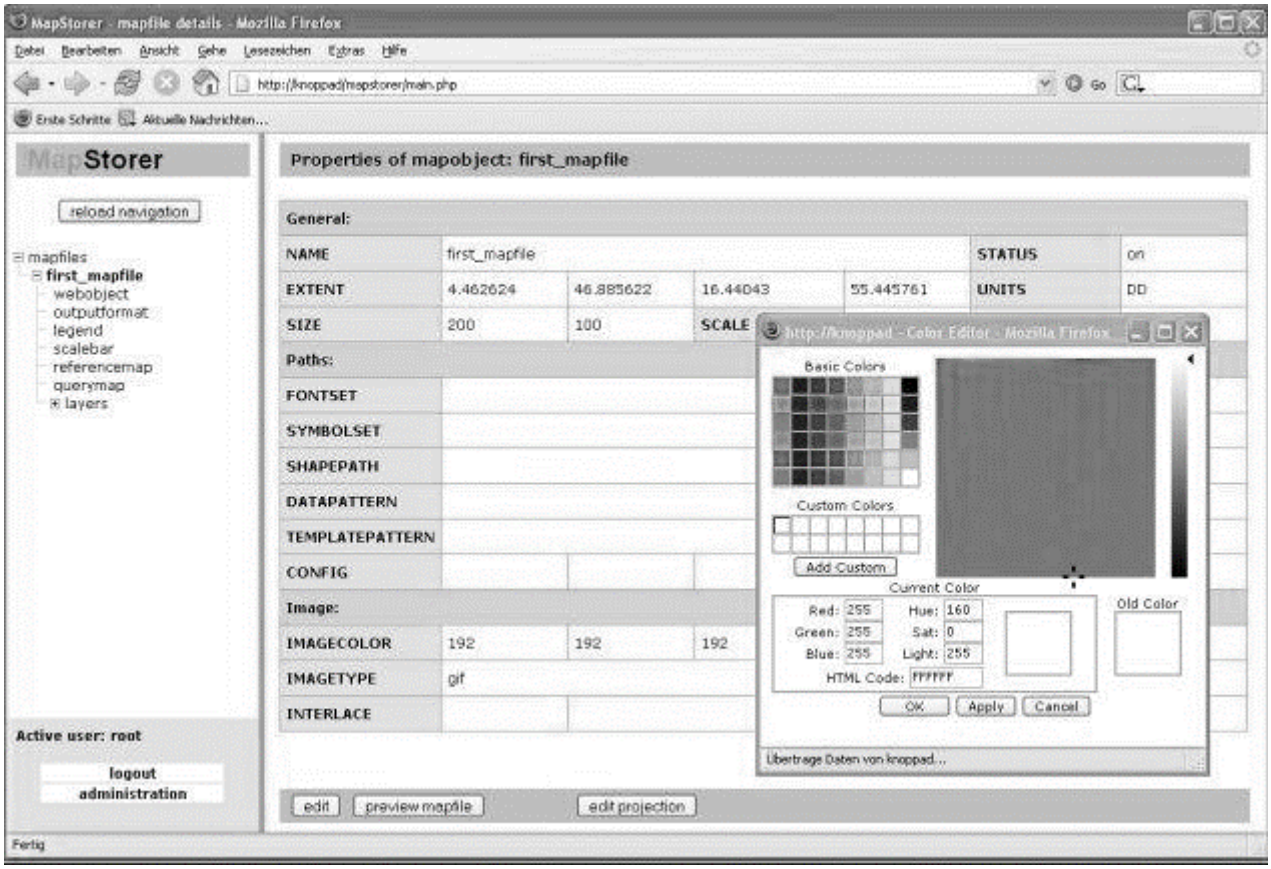


Fig. 4: Interfaz de *MapStorer*

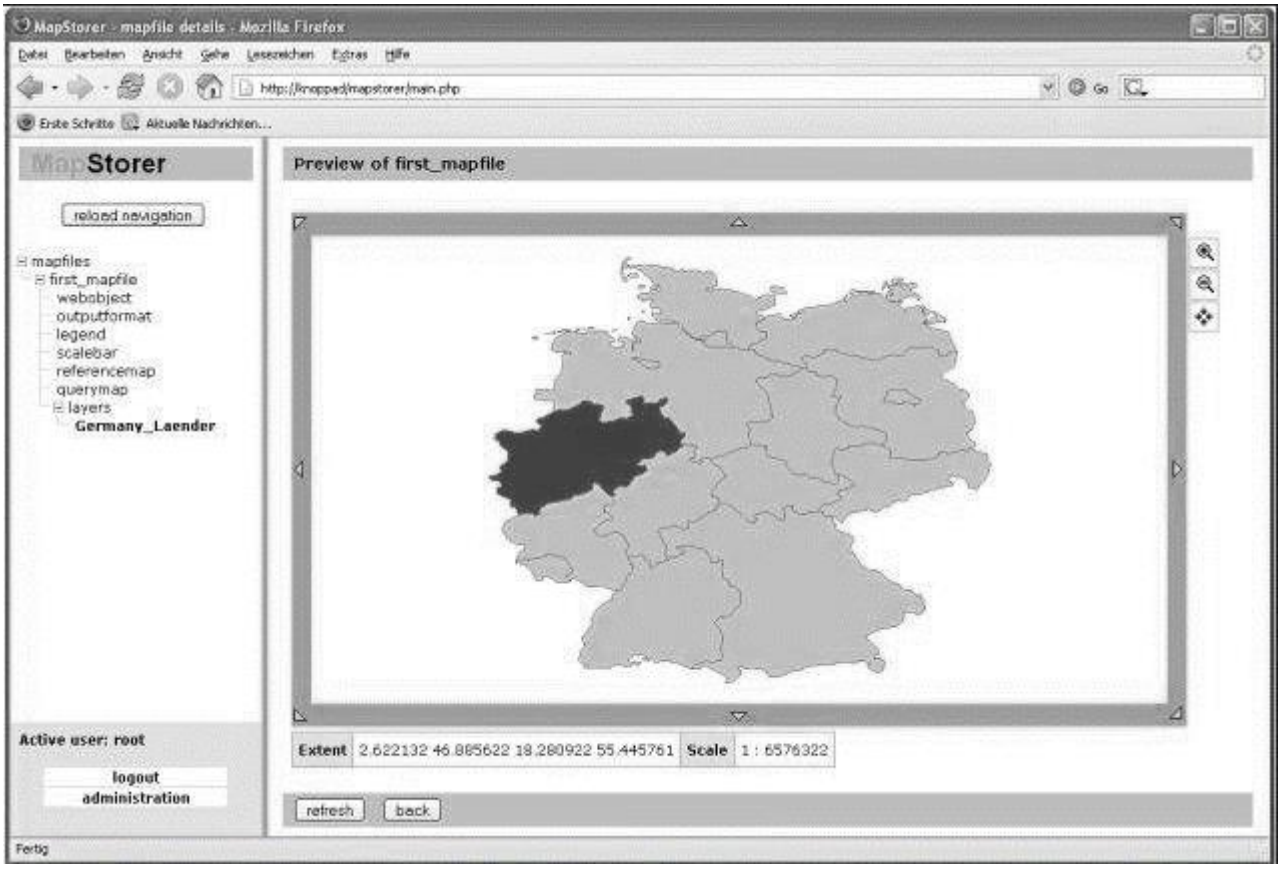


Fig. 5: Interfaz de visualizar mapa de *MapStorer*

Después de haber realizado un análisis de las principales herramientas que permiten la edición del fichero *MapFile*, se decide descartar a *MapStorer* porque no permite la creación y edición del Objeto *SYMBOL*, uno de los objetos más importantes del mapa, *LiberMaps* se descartó puesto que utiliza la versión 5.6 de *MapServer* y el equipo de trabajo está utilizando la versión 6.4 de dicho servidor; por lo que se decide implementar la herramienta *ManagerMaps* para la gestión de mapas en SIG basados en *MapServer*. A pesar de que las herramientas analizadas no presentan una solución definitiva al problema a resolver, contribuyeron de una forma u otra en la construcción de la herramienta *ManagerMaps*. De *MapStorer* se utilizó la forma en que representa una vista previa del mapa que se va construyendo y, por último, de *LiberMaps* surgieron la mayoría de los requisitos funcionales y no funcionales de la aplicación ya que esta herramienta era la que se utilizaba en el proyecto y sus funcionalidades son similares a las de la herramienta que se desea construir.

1.4 Tecnologías a utilizar

El desarrollo de la herramienta que se utilizará, se sustenta dentro de la tecnología *web*, a través del lenguaje de programación del lado del servidor *PHP* 5, y del lado del cliente *JavaScript*, en ambos casos apoyado en el *framework Symfony* 2.8 y *Bootstrap* 3.

Como gestor de base de datos *PostgreSQL* 9.3 por ser libre, soportar datos espaciales con la extensión *PostGIS* 2.1, además de herencia, creación de funciones, etc. Tiene una gran comunidad de usuarios que permiten su continuo desarrollo, lo que constituye un factor importante para la continuidad del proyecto.

1.4.6 Metodología de desarrollo de software

Una metodología es un conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar un nuevo *software*, además de constituir una asistencia fundamental cuando se da mantenimiento o actualización a un *software* implementado. (Pressman, 2010).

Una metodología de desarrollo de *software*, es aquella que hace posible la planificación, organización y construcción de un sistema o proyecto, con independencia de su temática o complejidad. Actualmente estas metodologías son una guía en el proceso de desarrollo de las aplicaciones informáticas, permitiendo que se obtengan resultados con la mayor calidad, rapidez y eficiencia posible, para evitar cometer errores futuros". (Pressman, 2002).

Las metodologías de desarrollo de *software* surgen como alternativa al problema de lo difícil que resulta la construcción de un producto informático. Estas imponen un proceso disciplinado sobre el desarrollo de *software* con el fin de hacerlo más predecible y eficiente.

AUP-UCI (Agile Unified Process)

La metodología empleada para el desarrollo de esta investigación se basó en el método de desarrollo ágil AUP. El Proceso Unificado Ágil (*AUP*) es una versión simplificada de *RUP* (Proceso Unificado de Software). Éste describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en *RUP*. En la UCI se decide hacer una variación de esta metodología, de forma tal que se adapte al ciclo de vida definido para la actividad productiva en la universidad, logrando estandarizar el proceso de desarrollo de software. De las cuatro fases que propone *AUP* (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la

misma, se unifican las restantes 3 fases de *AUP* en una sola titulada Ejecución y se agrega la fase de Cierre (Sánchez, 2015). Esta metodología se utilizó por las características que presenta y por ser la utilizada para guiar el proceso de desarrollo del *software* en el proyecto, permitiendo una mejor compatibilidad en la documentación (Sánchez, M, y otros, 2007). Para una mayor comprensión se muestra la siguiente tabla:

Tabla 1: Fases de la metodología AUP

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del *software*.
Construcción		
Transición		
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.4.2 Lenguajes de programación

Un lenguaje de programación es un lenguaje diseñado para describir el conjunto de acciones consecutivas que un equipo debe ejecutar. Está formado por un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. También se puede definir como un idioma artificial diseñado para ordenar tareas a realizar por máquinas como las computadoras, que puede usarse para crear programas que controlen el comportamiento físico y lógico de la computadora. Dentro de los lenguajes de programación se encuentran los de bajo nivel y los de alto nivel que se caracterizan por expresar los algoritmos necesarios para la creación de programas informáticos de manera que facilitan la comunicación entre un humano y la computadora mediante signos convencionales cercanos al lenguaje natural. Algunos de los lenguajes de programación más empleados en la actualidad son *C++, Java, C#*, entre muchos otros.

JavaScript 1.2, del lado del cliente

JavaScript se define como un lenguaje orientado a objetos, basado en prototipos, imperativo y dinámico. Es utilizado principalmente del lado del cliente permitiendo crear efectos atractivos y dinámicos en las páginas web. Está basado en acciones que poseen menos restricciones. Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros (Valdés Pérez, 2007). Además, admite hacer validaciones del lado del cliente facilitando esto un menor tiempo de respuesta y evita efectuar llamadas incorrectas al servidor.

HTML 5, del lado del cliente

Nuevo concepto para la construcción de sitios *web* y aplicaciones en una era que combina dispositivos móviles, computación en la nube y trabajos en red. *HTML5* propone estándares para cada aspecto de la *web* y también un propósito claro para cada una de las tecnologías involucradas. A partir de ahora, *HTML* provee los elementos estructurales, *CSS* se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista. *JavaScript* tiene todo el poder necesario para proveer dinamismo y construir aplicaciones *web* completamente funcionales.

HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Nunca fue declarado oficialmente, pero, incluso cuando algunas *APIs* (Interfaz de Programación de Aplicaciones) y la especificación de *CSS3* por completo no son parte del mismo, es considerado el producto de la combinación de *HTML*, *CSS* y *JavaScript*. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la especificación de *HTML5*. *HTML* está a cargo de la estructura, *CSS*

presenta esa estructura y su contenido en la pantalla y *JavaScript* hace el resto que es extremadamente significativo (Gauchat, 2012).

Hypertext Preprocessor (PHP) 5.5.9, del lado del servidor

PHP es un lenguaje de programación interpretado que se utiliza para la generación de páginas web de forma dinámica. Es de código abierto, gratuito y multiplataforma lo que quiere decir que puede ser ejecutado en la mayoría de los sistemas operativos tales como son *Unix* y *Windows*. Este lenguaje es usado principalmente en interpretación del lado del servidor (Álvarez, 2001). Permite la conexión a diferentes tipos de servidores como: *MySQL, PostgreSQL, Oracle, ODBC, DB2, Microsoft SQL Server, Firebird y SQLite*. Él ofrece integración con varias bibliotecas existentes (Vázquez Mariño, 2008).

Algunas de las principales características de *PHP* (Hanze, 2008):

- Puede ser utilizado en cualquiera de los principales Sistemas Operativos del mercado, incluyendo *Linux*, muchas variantes Unix (incluyendo *HP-UX, Solaris* y *OpenBSD*), *Microsoft Windows, Max OSX, RISC OS.*
- Soporta la mayoría de servidores *web* hoy en día, incluyendo *Apache, Microsoft Internet Information Server, Personal Web Server* y muchos otros.
- Utiliza la programación orientada a objetos.
- La característica más potente es que tiene soporte para una gran cantidad de base de datos.

Symfony 2.8 como framework de desarrollo de PHP

Symfony es un completo *framework* diseñado para optimizar, gracias a sus características, el desarrollo de las aplicaciones *web*. Empieza por separar la lógica de negocio, la lógica de servidor y la presentación de la aplicación *web*, proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación *web* compleja, y automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación (Symfony, 2015).

Symfony está desarrollado completamente con *PHP* y ha sido probado con éxito en sitios como *Yahoo! Answers, delicious, DailyMotion* y otros de primer nivel. *Symfony* es compatible con la mayoría de gestores de bases de datos, como *MySQL, PostgreSQL, Oracle y SQL Server* de *Microsoft*. Se puede ejecutar tanto en plataformas **nix (Unix, Linux*), como en plataformas *Windows*.

Symfony 2 ha cambiado la carga automática de clases con respecto a sus versiones anteriores para ser más universal, más rápida e independiente de la necesidad de vaciar la caché. Esto provocó que el dilema

del rendimiento de *Symfony* 1 fuera resuelto logrando evacuar esta situación. Así mismo esta versión facilita el trabajo con la aplicación al lograr una integración entre sus componentes, o sea que puedes desarrollar una sola aplicación donde se maneje la parte gráfica y la controladora en una misma área. (Symfony, 2009).

Framework de desarrollo Bootstrap 3.1

Bootstrap es un *framework* o conjunto de herramientas de *software* libre para diseño de sitios y aplicaciones *web*, originalmente creado por *Twitter*. Los diseños creados con *Bootstrap* son simples, limpios e intuitivos, esto le da agilidad a la hora de cargar y al adaptarse a otros dispositivos. El *framework* trae varios elementos con estilos predefinidos fáciles de configurar: formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en *HTML* y *CSS*, así como, extensiones de *JavaScript* opcionales adicionales. (Otto, y otros, 2006)

1.4.3 Sistemas Gestores de Bases de Datos

Un Sistema de Gestión de Bases de Datos (SGBD) es un conjunto de programas informáticos que permiten el almacenamiento, modificación y extracción de la información en una base de datos, además de proporcionar herramientas para añadir, borrar, modificar y analizar los datos. También proporcionan métodos para mantener la integridad de los mismos, para administrar el acceso de usuarios a los datos y para recuperar la información si el sistema se corrompe (Valderrey Sanz, 2011)

PostgreSQL 9.3 como Sistemas Gestores de Bases de Datos

PostgreSQL es un SGBD relacional, robusto, con escalabilidad, su código fuente está disponible libremente, *PostgreSQL* se distribuye bajo la Licencia *PostgreSQL*, una licencia liberal de código abierto, similar a la *BSD* o licencias del *MIT* (Martínez, 2013). Posee una amplia variedad de plataformas tales como: *Linux, Windows, Solaris, Mac OS X* y otros. Soporta la realización de transacciones seguras, vistas, uniones, claves extranjeras, procedimientos almacenados, triggers, tipos de datos definidos por el usuario y otras operaciones que se realizan en ellos. El almacenamiento es confiable y consistente. La manipulación es potente, flexible y eficiente. Está considerada como la base de datos de código abierto más avanzada del mundo (Guerrero Martínez, 2010). Se decidió utilizar este SGBD para trabajar por las características con las que cuenta, además es el utilizado por la plataforma GeneSIG.

A continuación, algunas de las características más importantes y soportadas por *PostgreSQL* (Momjian, 2000):

- Soporta distintos tipos de datos y permite la creación de tipos propios.
- Múltiples métodos de autentificación.
- Es multiplataforma.
- Soporta almacenamiento de objetos grandes como gráficos, videos, sonidos, entre otros.

PostGIS: la extensión geográfica del gestor seleccionado

Con la finalidad de que la base de datos *PostgreSQL* soporte objetos geográficos se ha desarrollado el módulo *PostGIS*, convirtiéndola en una base de datos espacial que se puede utilizar en Sistemas de Información Geográfica (Moreta, 2009). Añade soporte para objetos geográficos que permite consultas de ubicación para ser ejecutadas en *SQL* (Ramsey, y otros, 2015).

PostGIS 2.1: Es una versión estable y liberada de *PostGIS*, extensión de *PostgreSQL* que la convierte en una Base de Datos Espacial. Posee soporte de datos *raster* y vectoriales. Es compatible con el servidor de mapas *MapServer* (Ramsey, y otros, 2015)

1.4.4 Lenguaje unificado de modelado (UML)

El lenguaje unificado de modelado es la notación, principalmente gráfica, que usan los métodos para expresar un diseño. El modelado permite especificar el comportamiento deseado en la construcción de un sistema, comprender mejor lo que se construye y se descubren oportunidades de simplificación y reutilización (Booch, y otros, 2000).

UML es un lenguaje estándar de modelado visual que se usa para especificar, construir, documentar y visualizar artefactos de un sistema de *software*. Este lenguaje proporciona un vocabulario que incluye tres categorías: elementos, relaciones y diagramas, conteniendo aspectos conceptuales tales como procesos de negocios, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes de software reutilizables (Booch, y otros, 2000). *UML* es el estándar de la industria en la actualidad debido a un conjunto de rasgos principales dentro de los que se pueden mencionar (Rumbaugh, y otros, 2000)

- Permite modelar sistemas utilizando técnicas orientadas a objetos.
- Es un lenguaje muy expresivo que cubre las vistas necesarias para desarrollar y luego desplegar los sistemas.
- Ampliamente utilizado por la industria del *software*.
- Reemplaza a decenas de notaciones empleadas por otros lenguajes.

- Comportamiento del sistema: casos de usos, diagramas de secuencia, de colaboración, que sirve para evaluar el estado de las máquinas.

Por lo antes mencionado se tomó la decisión de utilizar *UML* para el modelado de la herramienta, ya que es de fácil compresión para su uso y menos engorroso realizar cambios una vez que se haya comenzado a desarrollar el *software*.

1.4.5 Herramientas CASE

Son un conjunto de herramientas y métodos asociados que proporcionan asistencia automatizada en el proceso de desarrollo del *software* a lo largo de su ciclo de vida, mediante su uso se reduce el tiempo, costo del desarrollo y mantenimiento del *software*, así como se mejora su calidad. (Pressman, 2010)

Visual Paradigm for UML 8.0

Visual Paradigm es una herramienta *CASE*, que propicia un conjunto de ayudas para el desarrollo de programas informáticos dando soporte al modelado visual con *UML* desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación. Es soportada por varios sistemas operativos tales como: *Ubuntu, Windows* y otros. Fue concebida para soportar el ciclo de vida completo del proceso de desarrollo del *software* a través de la representación de varios tipos de diagramas. Diseñada para una amplia gama de usuarios interesados en la construcción de sistemas de *software* de forma fiable a través de la utilización de un enfoque orientado a objetos (International, 2008).

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

Visual Paradigm ofrece:

- Entorno de creación de modelos conformes a *UML*.
- Diseño centrado en casos de uso y enfocado al negocio que genera un *software* de mayor calidad.
- Capacidades de ingeniería directa (versión profesional) e inversa.
- Modelo y código que permanece sincronizado en todo el ciclo de desarrollo.

- Disponibilidad de múltiples versiones, para cada necesidad.
- Disponibilidad de integrarse en los principales Entornos de Desarrollo Integrado (*IDE*).
- Disponibilidad en múltiples plataformas.
- Extensible mediante desarrollo de nuevos Módulos (*plugins*).

1.4.6 Servidor Web

Un servidor *web* es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario. El servidor *web* es el encargado de contestar a estas peticiones de forma adecuada, entregando como resultado una página *web* o información de todo tipo en correspondencia con los comandos solicitados (textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de música.) (The Apache Software Foundation, 2016).

El servidor *HTTP Apache* 2.4 es un servidor *web HTTP* de código abierto para plataformas *Unix (BSD, GNU/Linux*, entre otros), *Microsoft Windows, Macintosh* y otras, que implementa el protocolo *HTTP*/1.1.

La arquitectura del servidor *Apache* es muy modular. Consta de una sección *Core* y diversos módulos que aportan muchas de las funcionalidades que podrían considerarse básicas para un servidor *web*. Se emplea principalmente para enviar páginas *web* estáticas y dinámicas en la *World Wide Web*. Muchas aplicaciones *web* están diseñadas asumiendo como ambiente de implantación a *Apache* o que utilizarán características propias de este servidor *web*.

1.4.7 Servidor de mapa

MapServer 6.4 es un motor de procesamiento de datos geográficos escrito en C. Permite crear “mapas de imágenes geográficas”, es decir, mapas que pueden dirigir a los usuarios hacia el contenido. Actualmente *MapServer* es mantenido por un creciente número de desarrolladores de todo el mundo. Es apoyado por un grupo diverso de organizaciones que patrocinan las mejoras y el mantenimiento, además es administrado al interior de *OSGeo (Open Source Geospatial*). *MapServer* incluye algunas salidas cartográficas avanzadas, tales como (MapServer, 2008):

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

1.5 Conclusiones del Capítulo

El análisis de las soluciones existentes arrojó que no existen propuestas de implementación que satisfagan los requerimientos del problema de la investigación actual. Aun así, varias de sus funcionalidades fueron contempladas en la propuesta actual. Las herramientas y tecnologías utilizadas (libres y multiplataforma) para resolver el problema planteado permiten que *ManagerMaps* sea un producto informático que contribuye a la soberanía tecnológica que impulsa el país y la universidad.

Capítulo 2: Análisis y diseño de la solución propuesta.

2.1 Introducción

Para guiar el proceso de desarrollo de la solución informática es necesario comprender su contexto e identificar las condiciones o capacidades que la misma debe tener. En este capítulo se plantea la representación del respectivo modelo de dominio, así como los requisitos funcionales y no funcionales que debe cumplir el sistema para considerarse exitoso ante los usuarios finales y el patrón arquitectónico a emplear. El diseño está guiado mediante la utilización de patrones de asignación de responsabilidades y grupo de los cuatros que fundamentan la representación originada junto con el modelo del diseño. También la estructuración, aplicación y relación de estos elementos se establece mediante la concepción, orientada a objetos y basada en componentes, garantizando una arquitectura de *software* estable.

2.2 Modelo del Dominio

El modelo de dominio representa las clases más significativas del sistema. Su objetivo fundamental es, a través del Diagrama de clases del dominio, facilitar la comprensión del contexto y de los requisitos del sistema. Se utilizan diagramas *UML* para su descripción (Jacobson, y otros, 2000). Es la representación visual de los conceptos u objetos más importantes de un negocio, sus características y las relaciones entre dichos conceptos. Es el mecanismo fundamental para comprender el dominio del problema y para establecer conceptos comunes. Es un diccionario visual del dominio del problema.

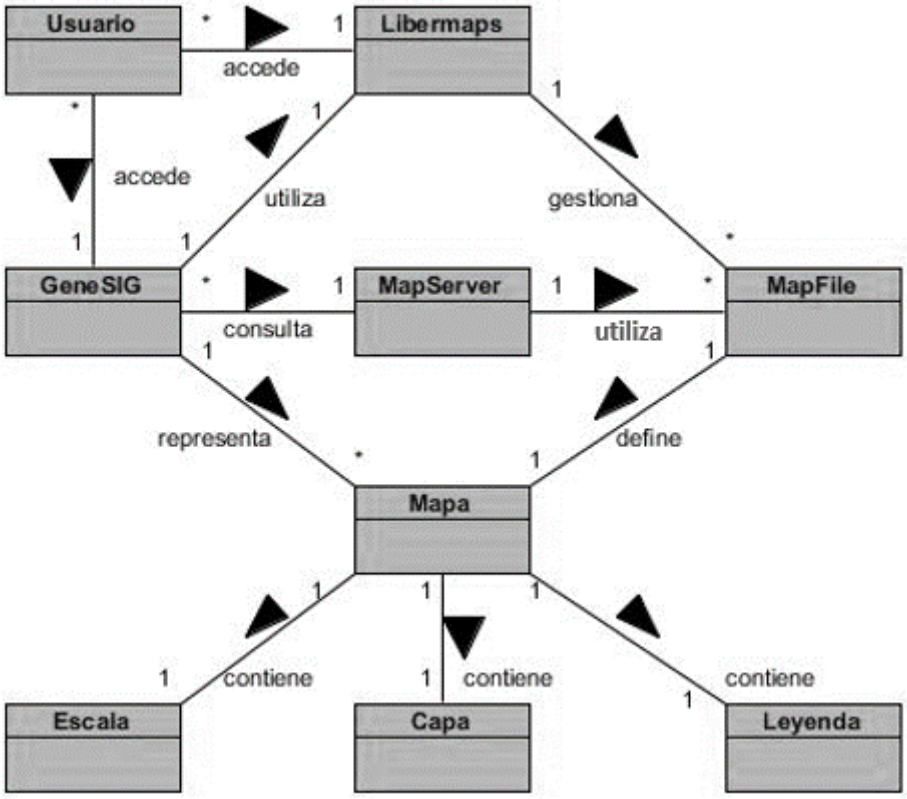


Fig. 6: Diagrama del modelo de dominio

El SIG, en este caso la plataforma GeneSIG realiza consultas al servidor de mapas *MapServer* que contiene el fichero de configuración de mapas *MapFile* por cada una de las consultas que realiza el SIG. Cada uno de estos ficheros define un mapa que GeneSIG representará. Los mapas contienen leyenda, escala, un conjunto de capas y un conjunto de atributos que lo describen. Por otro lado, el usuario es quien gestiona los ficheros *MapFile* a través de su interacción con el catálogo de mapas *LiberMaps* en su versión 1.0.

Tabla 2: Descripción de las clases del diagrama del modelo de dominio

	Definición de clases del diagrama del dominio
GeneSIG	Plataforma para la creación de SIG, capaz de integrar, almacenar, editar, analizar, compartir y mostrar la información geográficamente referenciada.
Usuario	Es la persona que interactúa con el SIG en caso este GeneSIG.

MapServer	Servidor de Mapas que es consultado por el SIG para visualizar mapas.
MapFile	Es el archivo principal desde donde *MapServer* obtiene las características del mapa que representa el SIG
Mapa	Es la representación gráfica de una porción de territorio; tiene propiedades métricas y permite medir distancias, ángulos y otros.
Escala	Relación que existe entre las medidas de distancia expresadas en el mapa y las mismas de manera real; relación matemática entre las dimensiones en el mapa y la porción de terreno que representan.
Capa	La información temática que se usa para elaborar los SIG está estructurada mediante capas.
Leyenda	Descripción para entender las simbologías que aparecen en el mapa cuyo significado el usuario desconoce.
LiberMaps	Catálogo de mapas que permite mantener el control y gestionar cada fichero *MapFile* de *MapServer*.

2.7 Descripción de la Solución

La herramienta ManagerMaps tiene el objetivo de editar y exportar el fichero *MapFile*, el cual es el archivo fundamental para el servidor de mapas *MapServer*. Debe permitir al usuario gestionar cada uno de los componentes que posee dicho fichero, estos son MAPA, ESCALA, LEYENDA, MAPA CONSULTA, OBJETO WEB, MAPA REFERENCIA, SÍMBOLO, CAPA, CLASE, ESTILO, ETIQUETA Y METADATO. Además, debe permitir gestionar las relaciones entre estos.

La herramienta obtiene los datos a través de la importación de un fichero *.map* o de la inserción de estos manualmente por parte del usuario. Además de generar el fichero *MapFile* se debe mostrar al usuario una vista previa del mapa generado por *MapServer* a partir del fichero.

2.3 Requisitos del Software

Según (Pressman, 2010) los requisitos de *software* constituyen las necesidades de los clientes, las funcionalidades que debe cumplir el sistema software y las restricciones que debe cumplir. Los mismos se clasifican en funcionales y no funcionales. A continuación, se presentan los requisitos del sistema para ver una descripción más detallada de cada uno de estos:

2.3.1 Requisitos funcionales

Los requisitos funcionales son condiciones o capacidades con los que debe cumplir el sistema, son acuerdos entre el cliente y los desarrolladores sobre lo que debe o no debe hacer el sistema (Jacobson, y otros, 2000).

RF1: Importar fichero *MapFile*. Permite importar un fichero con extensión *.map* desde una ubicación determinada y almacenarlo en la base de datos.

RF2: Exportar fichero *MapFile*. Permite al usuario seleccionar un fichero *MapFile* de los existentes y exportar de dicho fichero con extensión *.map* la información asociada al fichero *MapFile* (mapa) almacenada en la base de datos.

RF3: Modificar objeto *map*. Permite modificar todos los datos asociados al objeto *MAP* del fichero *MapFile*, como Proyección, Configuración, Extensión, Tamaño, Color y características generales. Además, permite mostrar una vista previa del mismo.

RF4: Eliminar fichero *MapFile*. Permite eliminar el mapa con cada dato asociado a éste.

RF5: Crear *LAYER*. Permite al usuario adicionar las capas del fichero *MapFile* a partir de un conjunto de datos asociados a las mismas como Nombre, Proyección, Conexión, Tipo, entre otras características generales.

RF6: Modificar *LAYER*. Permite modificar los datos asociados al objeto *LABEL* del fichero *MapFile* como Posición, Tamaño, Color y características generales.

RF7: Eliminar *LAYER*. Permite al usuario eliminar una o varias capas correspondientes al fichero *MapFile*.

RF8: Buscar *LAYER*. Permite al usuario obtener una vista de las capas que forman parte del fichero *MapFile*.

RF9: Reordenar *LAYER*. Permite establecer el orden lógico de las capas del fichero *MapFile* a partir de criterios que ya se tienen definidos.

RF10: Adicionar *metadata* a un *LAYER*. Permite crear los metadatos del objeto *LAYER* del fichero *MapFile* a partir de su nombre y descripción.

RF11: Adicionar *SYMBOL*. Permite al usuario la adición de los objetos *SYMBOL*S del fichero *MapFile* a partir de un conjunto de datos asociados al mismo como Nombre, Tipo, *Linecap, Points*, entre otros.

RF12: Modificar *SYMBOL*. Permite al usuario cambiar los valores del objeto.

RF13: Eliminar *SYMBOL*. Permite al usuario eliminar todos los objetos *SYMBOL*S del fichero *MapFile* que desee.

RF14: Buscar *SYMBOL*. Permite que el usuario busque un símbolo del mapa.

RF15: Adicionar *OUTPUTFORMAT*. Permite al usuario realizar la adición de los objetos *OUTPUTFORMAT*S del fichero *MapFile* a partir de un conjunto de datos asociados al mismo como Nombre, Formato, Extensión, entre otros.

RF16: Modificar *OUTPUTFORMAT*. Permite cambiar el *OUTPUTFORMAT* como como *PNG, GIF, JPEG, GeoTIFF, SVG, PDF* y *KML*.

RF17: Eliminar *OUTPUTFORMAT*. Permite al usuario eliminar uno o varios objetos *OUTPUTFORMAT*S del fichero *MapFile*, pero siempre asegurando que quede al menos uno.

RF18: Buscar *OUTPUTFORMAT*. Busca cuál es el tipo de formato de salida que tiene el objeto.

RF19: Adicionar *class* a un *LAYER*. Permite adicionar una clase dentro de una capa determinada, esta clase agrupa un conjunto de características de la capa.

RF20: Modificar *class* a un *LAYER* Permite modificar las características de la clase de esa capa.

RF21: Eliminar *class* a un *LAYER*. Descarta la clase que tiene la capa seleccionada.

RF22: Buscar *class*. Busca la clase deseada por el usuario de la lista de clases existentes.

RF23: Adicionar *style* de una *class*. Permite adicionar un estilo a una capa determinada, dentro del estilo se especifican un conjunto de parámetros como son el color, *SYMBOL*, backgroundcolor, etc.

RF24: Modificar *style* de una *class*. Permite modificar las características del estilo a una capa determinada.

RF25: Eliminar *style* de una *class*. Elimina el estilo que tiene la capa.

RF26: Buscar *style*. Busca un estilo determinado dada la lista de estilos que existen.

RF27: Modificar objeto *web*. Permite modificar los datos asociados al objeto *WEB* del fichero *MapFile*, como Escala y Características Generales. Además, permite mostrar una vista previa del mismo.

RF28: Adicionar *metadata* al objeto *web*. Permite crear los metadatos del objeto *WEB* del fichero *MapFile* a partir de su nombre y descripción.

RF29: Modificar objeto *querymap*. Permite modificar los datos asociados al objeto *QUERYMAP* del fichero *MapFile*, como Tamaño, Color, Estilo y Características Generales. Además, permite mostrar una vista previa del mismo.

RF30: Modificar *reference*. Permite modificar los datos asociados al objeto *REFERENCE* del fichero *MapFile*, como Tamaño, Color, Extensión, Marcadores y Características Generales. Además, permite mostrar una vista previa del mismo.

RF31: Modificar *scalebar*. Permite cambiar valores de la escala gráfica a petición del usuario, ya sea cambiar el tamaño, el color, visualizarla en metros, kilómetros, entre otras cosas.

RF32: Modificar *legend*. Permite modificar los datos asociados al objeto *LEGEND* del *MapFile*, como Tamaño, Color y Características Generales. Además, permite mostrar una vista previa del objeto *LEGEND*.

RF33: Modificar *label*. Permite al usuario usado poner cualquier tipo de anotación en el mapa.

RF34: Visualizar el mapa. Muestra una vista previa de un mapa determinado.

RF35: Asignar elemento. Asigna un elemento del mapa a otro.

2.3.2 Requisitos no funcionales

Además de los requisitos funcionales deben especificarse los requisitos no funcionales. 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, fiabilidad, seguridad, integridad, tiempo de respuesta y recuperación ante fallos.

Requisitos de Interfaz de usuario:

- El sistema debe tener indicadores que permitan conocer al usuario las acciones que debe realizar, por ejemplo, botones con íconos sugerentes y alternativa textual.

Restricciones de diseño e implementación:

- Lenguaje de programación: *JavaScript 1.2* y *PHP* 5.5.9
- Lenguaje de marcado: *HTML* 5.
- Gestor de base de datos: *PostgreSQL* 9.3 y *PostGIS* 2.1.

Requisitos de Software:

Para las PC clientes:

- Un navegador como *Mozilla Firefox* con versión 16.0 o superior hasta 46.
- Sistema operativo *GNU/Linux*, *Windows* 7 o 8.

Para las PC servidor:

- Sistema operativo *GNU/Linux*, *Ubuntu Server* 14.04
- Servidor *Web Apache* 2.4, con módulo *PHP* 5.5.9 configurado con la extensión *pgsql* incluida.
- *PostgreSQL* 9.3 como Sistema Gestor de Base de Datos.
- *PostGIS* 2.1 como soporte de datos espaciales.
- *MapServer* 6.4 con extensión *PHP MapScript*

Requisitos de Hardware:

Para las PC clientes:

- 512 MB de *RAM* como mínimo.
- 1 GB de capacidad en disco duro.

Para las PC servidor:

- 1 GB de *RAM* y 40 *GB* de disco duro como mínimo para el Servidor de mapas.
- 2 GB de *RAM* y 60 *GB* de disco duro como mínimo para el servidor de Base de Datos
- 3.0 *GHz* de Procesador como mínimo.

Requisitos de licencia:

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

2.4 Patrón Arquitectónico

La arquitectura de *software* permite representar de forma concreta la estructura y funcionamiento interno de un sistema (Clements, y otros, 2003). Para su diseño se debe crear y representar componentes que

interactúen entre ellos y tengan asignadas tareas específicas, además de organizarlos de forma tal que se logren los requisitos establecidos. Se puede comenzar con patrones de soluciones ya probados y utilizar modelos que han funcionado. Estas soluciones examinadas se conocen como patrones arquitectónicos, que van de lo general a lo particular.

Un patrón arquitectónico impone una transformación en el diseño de una arquitectura, algunos de sus elementos fundamentales son: su alcance se enfoca en un aspecto determinado de la arquitectura, impone una regla sobre la arquitectura puesto que describe la manera en la que el *software* manejará algún aspecto de su funcionalidad a nivel de infraestructura, abarcan aspectos específicos dentro del contexto de la arquitectura. Estos patrones, por lo general, definen la estructura de un sistema (Pressman, 2010).

2.4.1 Patrón Modelo-Vista-Controlador (MVC)

En el patrón arquitectónico MVC se separa la lógica de negocio de la interfaz de usuario y se aumenta la posibilidad de evolución por separado de ambos aspectos. Mediante su uso, es posible incrementar la reutilización y la flexibilidad de las aplicaciones debido a la separación de los aspectos antes mencionados. Las modificaciones a las vistas no afectan en absoluto a los otros módulos de la aplicación.

El uso del *framework* que utiliza MVC obliga a dividir y organizar el código de acuerdo a las convenciones establecidas por el *framework*. El código de la presentación se guarda en la vista, el código de manipulación de datos se guarda en el modelo y la lógica de procesamiento de las peticiones constituye el controlador. Aplicar el patrón MVC a una aplicación resulta muy útil además de restrictivo.

En el controlador se encuentran las acciones, las cuales son el núcleo de la aplicación, pues contienen toda la lógica de ella. Estas acciones utilizan el modelo y precisan las variables para la vista. Al realizarse una petición web en una aplicación *Symfony*, la *URL* define una acción y los parámetros de la petición.

La vista es la encargada de originar las páginas que son mostradas como resultado de las acciones, donde se encuentra el *layout*, que es común para todas las páginas de la aplicación. La vista en *Symfony* está conformada por varias partes, preparadas cada una de ellas especialmente para ser fácilmente transformable por la persona que normalmente trabaja con cada aspecto del diseño de las aplicaciones.

Se divide en las capas de acceso a datos y la capa de abstracción de la base de datos. La primera representa la lógica del negocio del sistema, es decir, las reglas del negocio y los requisitos funcionales que debe cumplir el sistema. La segunda capa brinda la posibilidad de no generar sentencias *SQL*, lo que

les facilita el trabajo a los programadores. Además de permitir que, si se desea cambiar el gestor de base de datos, esto sea posible sin tener que realizar cambios en la lógica del negocio, para ello se auxilia de la (Mapeo de Objetos Relacional) *ORM Doctrine*.

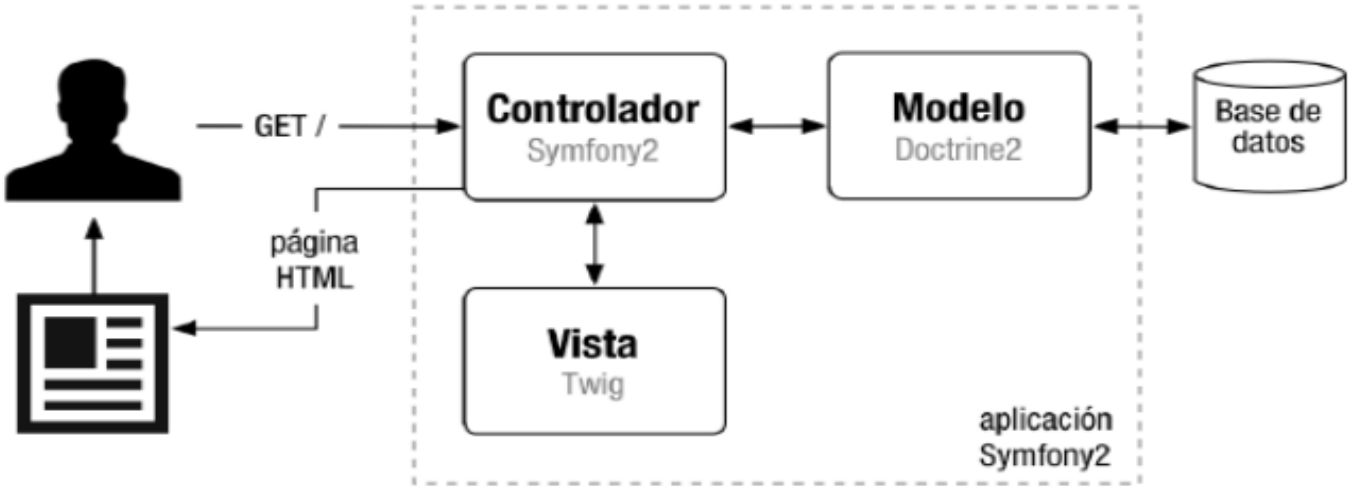


Fig. 7: Modelo-Vista-Controlador

2.5 Patrones de diseño

Un patrón de diseño describe una estructura de diseño que resuelve un problema de diseño particular dentro de un contexto específico. Permite al diseñador obtener una descripción la cual le facilita determinar si el patrón es aplicable al trabajo actual, si se puede reutilizar y así ahorrar tiempo de diseño, y si el patrón puede servir como guía para desarrollar un patrón similar pero con diferente estructura y funcionalidad (Pressman, 2010).

2.5.1 GRASP

Los patrones *GRASP* describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. *GRASP* es un acrónimo que *significa General Responsibility Assignment Software Patterns*. Aunque se considera que más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación recomendable en el diseño de *software* (Potencier, y otros, 2009).

- **Experto**

Es uno de los patrones que más se utiliza cuando se trabaja con Symfony, con la inclusión de la librería Doctrine para mapear la Base de Datos. *Symfony* utiliza esta librería para realizar su capa de abstracción en el modelo, encapsular toda la lógica de los datos y generar las clases con todas las funcionalidades comunes de las entidades, las clases de abstracción de datos poseen un grupo de funcionalidades que están relacionadas directamente con la entidad que representan y contienen la información necesaria de la tabla que representan. Un ejemplo es la clase *MapFile* que representa a la tabla *MapFile* de la Base de Datos.

- **Creador**

Se asigna la responsabilidad de que una clase B cree un objeto de la clase A. En la clase *Actions* se encuentran las acciones definidas para el sistema y se ejecutan en cada una de ellas. En dichas acciones se crean los objetos de las clases que representan las entidades, lo que evidencia que la clase *Actions* es "creador" de dichas entidades. En la aplicación se evidencia el uso de este patrón en la clase *DefaultController*, donde se crea un objeto de la clase *MapFile*.

- **Alta cohesión**

Cada elemento del diseño debe realizar una labor única dentro del sistema. Se utiliza mediante la estructuración jerárquica de las clases evitando la sobrecarga en funcionalidades de las mismas, con una mejor distribución de responsabilidades establecidas en el marco del negocio. *Symfony* permite la organización del trabajo en cuanto a la estructura del proyecto y la asignación de responsabilidades con una alta cohesión. Un ejemplo de ello es la clase *DefaultController*, la cual está formada por varias funcionalidades que están estrechamente relacionadas, siendo la misma la responsable de definir las acciones para las plantillas y colaborar con otras para realizar diferentes operaciones, instanciar objetos y acceder a las propiedades.

- **Bajo acoplamiento**

Se utiliza para lograr la mínima dependencia entre las clases del sistema, lo cual disminuye considerablemente el flujo de datos, propiciando rapidez en la ejecución de sus funcionalidades (MORENO 2012). Las clases que implementan la lógica del negocio y de acceso a datos se encuentran en el modelo como *MapFile*, las cuales no tienen asociaciones con las de la vista (*principal.html.twig*) o el controlador (*DefaultController*), lo que proporciona que la dependencia en este caso sea baja.

- **Controlador**

Asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Este patrón se evidencia en la clase *DefaultController* la cual mediante las *Actions* es la encargada de controlar todas las peticiones.

2.5.2 Patrones GOF

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

- **Patrón Decorator**

Añadir responsabilidades adicionales a un objeto dinámicamente, proporcionando una alternativa flexible a la especialización mediante herencia, cuando se trata de añadir funcionalidades. El archivo nombrado *layout* es el que contiene el *Layout* de la página. Este archivo, conocido también como plantilla global, guarda el código *HTML* que es usual en todas las páginas del sistema, para no tener que repetirlo en cada página. El contenido de la plantilla se integra en el *layout*, o si se mira desde el otro punto de vista, el layout decora la plantilla. Este procedimiento es una implementación del patrón Decorator. Se puede evidenciar en la clase *layout.html.twig* hereda de la clase *base.html.twig.*

2.6 Modelo de Datos

Para lograr que se gestionen correctamente los datos y que el usuario acceda a la información de manera rápida y eficiente se requiere de un buen diseño de la base de datos. (Pressman, 2010) Plantea que el modelado debe responder a una serie de preguntas importantes para cualquier aplicación de procesamiento de datos. El modelo de datos siguiente muestra cómo se representan las tablas en la base de datos.

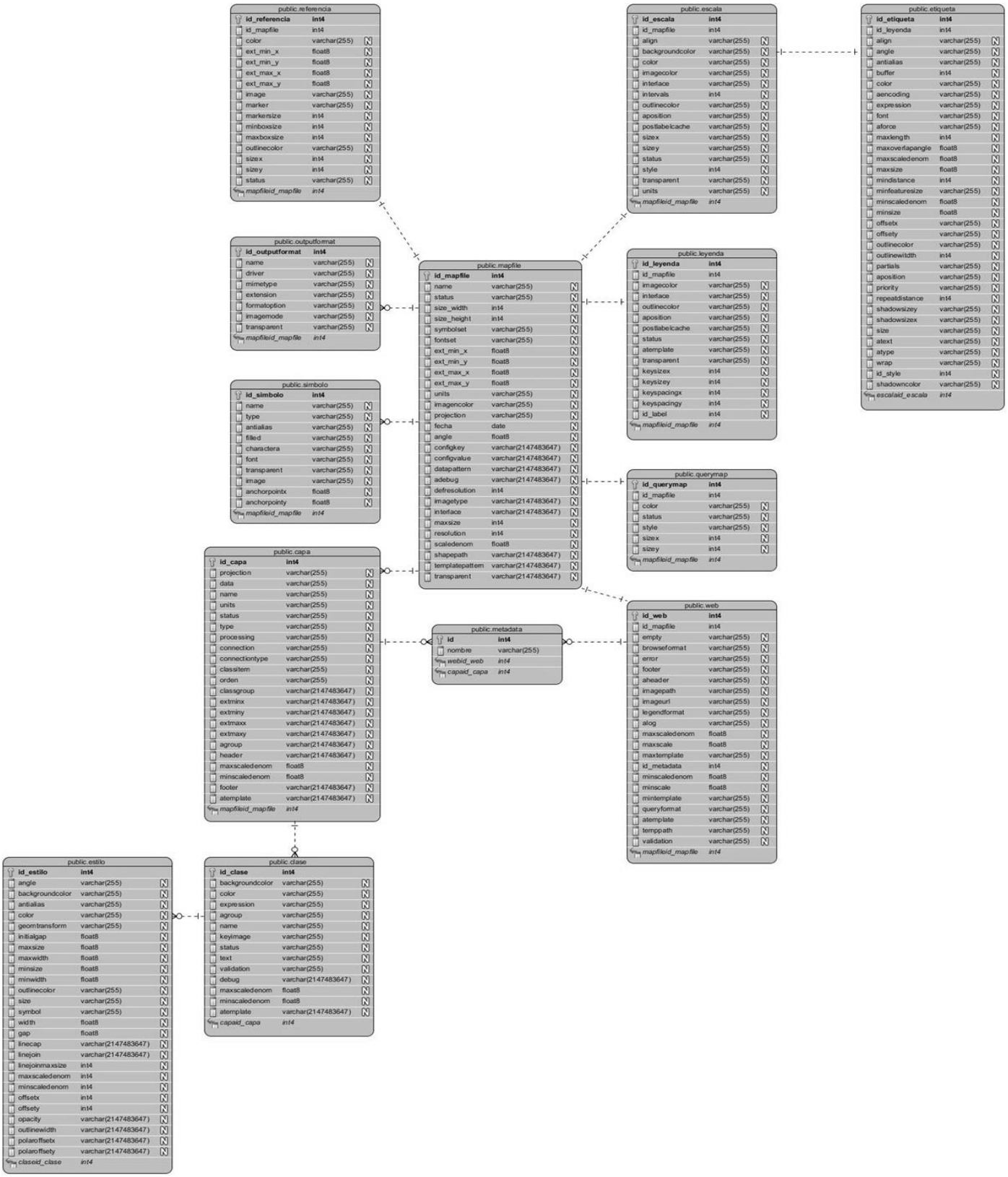


Fig. 8: Diagrama del modelo de dato

2.7 Modelo de diseño

Es la representación física de los Requisitos Funcionales mediante un modelo de objetos. Centraliza su atención en el impacto que tiene en el sistema los requisitos tanto funcionales como no funcionales, así como otras restricciones del entorno de implementación (Jacobson, y otros, 2000).

2.7.1 Diagrama de Clases del Diseño

Una clase del diseño es una abstracción de una clase real o construcción similar en la implementación del sistema. El lenguaje que se utiliza en dichas clases es el mismo que se emplea para la implementación del sistema. Según (Jacobson, y otros, 2000), un diagrama de clases corresponde a la realización de un Requisito Funcional, mostrando las clases, subsistemas y las relaciones involucradas en el mismo, también tiene en cuenta los requisitos que influyen sobre determinada clase, objetos o subsistemas. En la siguiente figura se presenta el diagrama de clases del diseño del requisito funcional Modificar fichero *MapFile*.

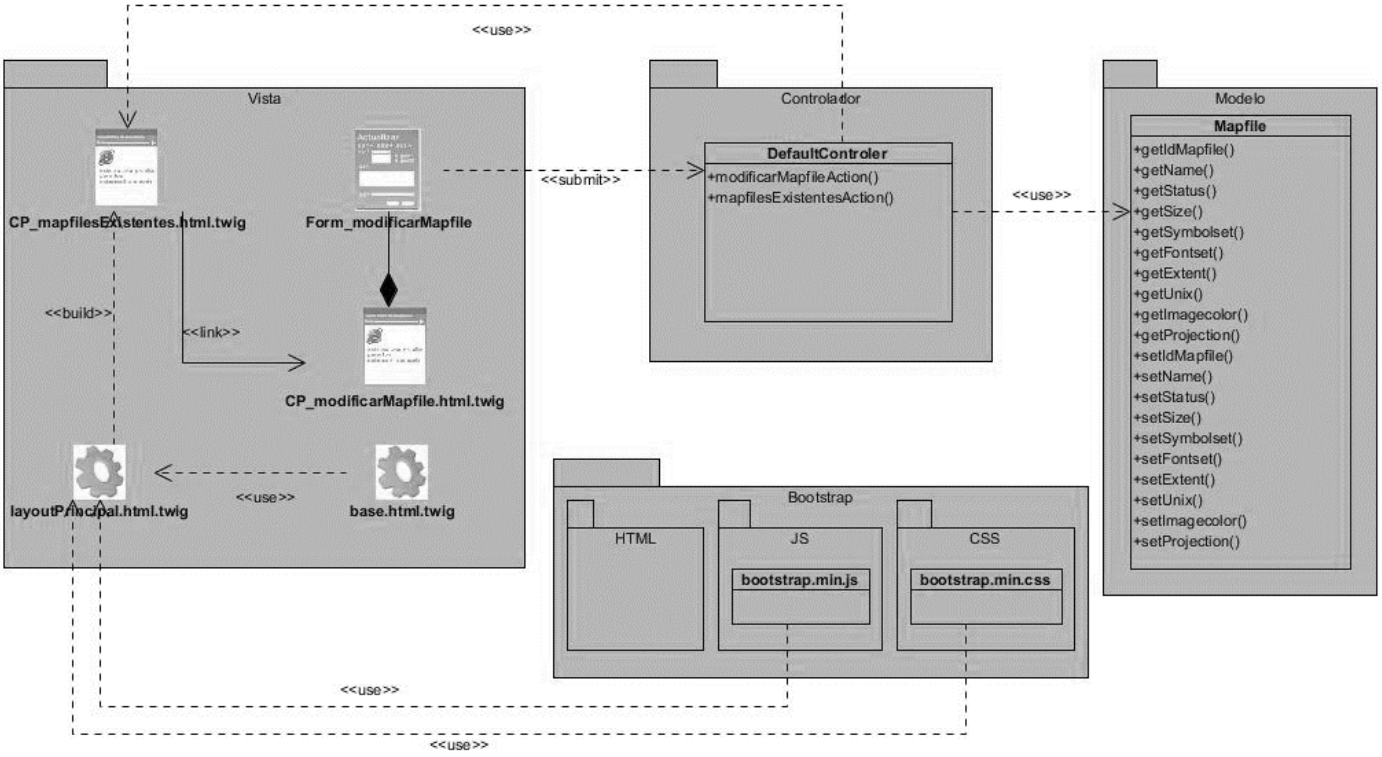


Fig. 9: Diagrama del modelo de clase del diseño Modificar fichero *MapFile*.

La figura 9 muestra el diagrama de clases del diseño, en el mismo se muestran las clases implicadas en la realización del requisito modificar *MapFile*. La clase *mapfilesExistentes.html.twig* muestra al usuario todos

los *MapFile*s existentes en la base de datos, esta hereda *de layoutPrincipal.html.twig* que a su vez hereda de la clase *base.html.twig* que proporciona *Symfony* 2. Estas interfaces son construidas utilizando las clases *bootstrap.min.js* y *bootsrap.min.css* proporcionadas por el *framework Bootstrap*. La clase *modificarMapfile.html.twig* muestra los datos del *MapFile* seleccionado y estos son enviados mediante el formulario modificar*Mapfile* a través de un *submit* a *DefaultController* donde serán procesados por el *Action modificarMapfileAction()*. La clase *DefaultController* se nutre de la entidad *MapFile* para obtener la información y retorna los resultados de la petición a la clase *mapfilesExistentes.html.twig*.

2.7 Conclusiones del capítulo

La modelación del dominio, la definición de los requisitos del *software* (funcionales y no funcionales) y la realización del diagrama de clases del diseño exponiendo los diferentes patrones de asignación de responsabilidades y del grupo de los cuatro, utilizados en la solución, estos contribuyen al entendimiento del equipo de desarrollo lo que representa una ventaja para la correcta ejecución de las restantes etapas en la construcción de la solución propuesta. El modelo de datos diseñado respondió a las exigencias que posee la solución propuesta.

Capítulo 3: Implementación y prueba de la solución propuesta.

3.1 Introducción

En este capítulo se muestra el flujo de trabajo de implementación, el cual forma parte de la fase de construcción. Como resultado se muestra el diagrama de componente para visualizar con mayor facilidad la estructura general de la herramienta y el diagrama de despliegue para modelar la arquitectura del sistema. Además, se definen las pruebas necesarias que se le aplicarán a la herramienta y se muestran los resultados obtenidos.

3.2 Modelo de implementación

Describe cómo los elementos del modelo del diseño se implementan en términos de componentes (ficheros de código fuente, ejecutables, entre otros). El modelo de implementación también representa la manera en que se organizan los componentes, de acuerdo con los mecanismos de estructuración y modularización disponibles en el entorno de implementación, así como, en el lenguaje o los lenguajes de programación utilizados, y la forma en que dependen los componentes unos de otros (Jacobson, y otros, 2000).

3.2.1 Diagrama de Componentes

Muestra el sistema de *software* dividido en componentes y las relaciones existentes entre estos, forma parte del modelo de implementación. Un diagrama de componentes permite visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que ellos proporcionan y utilizan a través de las interfaces (Jacobson, y otros, 2000).

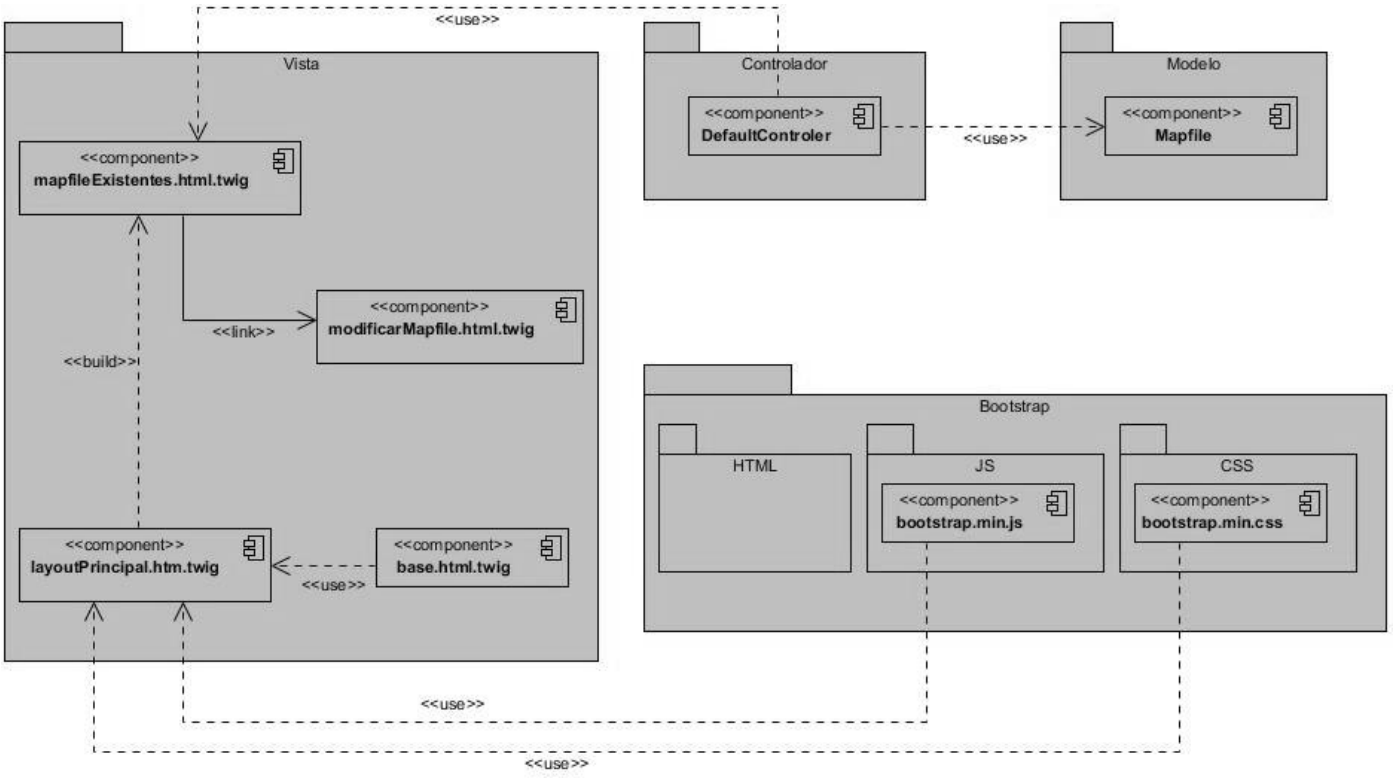


Fig. 10: Diagrama de componentes Modificar fichero *MapFile*.

2.7 Modelo de Despliegue

El diagrama de despliegue es un modelo de objetos que describe la topología de la arquitectura física del sistema. A través de la distribución de funcionalidades entre nodos interconectados y sus enlaces son elementos de *hardware* sobre los que puede ejecutarse el *software*, donde cada uno representa un dispositivo de procesamiento y cada enlace, los mecanismos de comunicación que se establecen entre dichos nodos y sobre cada uno de ellos se colocan, a modo de artefactos (elementos físicos simples) los elementos componentes del *software* (Jacobson, 2000). En la figura se presenta el diagrama de Despliegue. Para la instalación del sistema se requiere de un servidor *web*, *Apache*, donde se ejecutarán tareas como la construcción de interfaces de usuarios y procesamiento de datos. Junto con el servidor web se encontraría el servidor de mapas, en este caso *MapServer*. Se requiere además de un servidor de base de datos donde se ejecutaría el SGBD *PostgreSQL*. Estos dos servidores se comunicarán a través del protocolo de transmisión de datos *TCP/IP*. El cliente podrá acceder al sistema mediante el protocolo de transferencia de hipertexto *HTTP*.

Fig. 11: Diagrama de despliegue

3.4 Modelo de pruebas

Durante el flujo de trabajo de pruebas se verifica el resultado final de la implementación probando la estructura, tanto en la construcción interna como intermedia, así como las versiones finales del sistema a ser entregado. Uno de los artefactos generados en esta etapa es el modelo de pruebas que se describe como una colección de casos de pruebas, procedimientos de prueba y componentes de prueba. Las pruebas son actividades en las cuales un sistema o componente es ejecutable bajo condiciones o requerimientos específicos permitiendo que los resultados sean observados y registrados, estas se realizan con el objetivo de encontrar deficiencias existentes en el *software* (Booch, y otros, 2000).

3.4.1 Diseño de Casos de Prueba

Un diseño de caso de prueba (DCP) está compuesto por un conjunto de entradas, respuesta que emite el sistema de acuerdo a esas entradas y el flujo central que indica el camino del escenario descrito. Estos son desarrollados para verificar el cumplimiento total o parcial de un requisito. Las entradas representan las variables que se pueden especificar y las mismas contienen: V, I, o N/A. V indica válido, I indica inválido, y N/A que no es necesario proporcionar un valor del dato en este caso, ya que es irrelevante. A continuación, se presenta en la Tabla 3, un ejemplo de los DCP realizados para comprobar el funcionamiento de la solución, específicamente para el proceso Importar fichero *MapFile*.

Las pruebas de sistema deben ser ejecutadas idealmente por un equipo de pruebas ajeno al equipo de desarrollo. La obligación de este equipo, consiste en la ejecución de actividades de prueba en donde se debe verificar que la funcionalidad total de un sistema fue implementada de acuerdo a los documentos de especificación definidos.

Teniendo en cuenta las condiciones de la solución informática propuesta para la gestión del fichero *MapFile*, se realizó el método de caja negra utilizando la técnica de partición de equivalencia.

El método de caja negra también denominado pruebas de comportamiento, se centran en verificar el cumplimiento de los requisitos funcionales del *software*, se emplean cuando se conoce la función específica para la que se diseñó la solución y se aplican a la interfaz del *software*, permitiendo obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales del programa (Pressman, 2005), con el fin de encontrar la mayor cantidad de no conformidades existentes en el producto. La técnica empleada para el método de caja negra es, partición de equivalencia, que según define Pressman: "se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar" (Pressman, 2005).

DCP1: Importar fichero *MapFile*

Tabla 3: Caso de prueba Importar fichero *MapFile*

Escenario	Descripción	Variables				Respuesta del sistema	Flujo central
		#MapFile	***Name***	***Projection***	**Fecha**		
1.1 Importar fichero *MapFile*.	Importa correctamente el fichero *MapFile* de extensión *.map*.	N/A	N/A	N/A	N/A	Se le muestra un mensaje informándole al usuario que el fichero se ha importado correctamente.	1. *ManagerMaps*. 2. Seleccionar la opción Importar. 3. Se muestra un explorador para que el usuario seleccione el fichero con extensión *.map* que se desea importar. 4. Dar clic en abrir. 5. Se muestra un cartel al usuario informándole que el archivo se ha importado correctamente.
1.2 Seleccionar un archivo	El usuario selecciona un archivo que no tiene la	N/A	N/A	N/A	N/A	Al seleccionar un fichero y presionar el botón abrir la	1. *ManagerMaps*. 2. Seleccionar la opción Importar. 3. Se muestra un explorador

cuya extensión no es un *.map*.	extensión *.map*					aplicación comprueba que el fichero tiene la extensión *.map* y de no ser así le muestra un mensaje al usuario informándole que el archivo no es válido.	para que el usuario seleccione el fichero de extensión *.map* que se desea importar. 4. Dar clic en abrir. 5. Se muestra un cartel al usuario informándole que el archivo seleccionado no es válido.
1.3 Cancelar la operación Importar.	Cancela la operación de importar *MapFile*.	N/A	N/A	N/A	N/A	Se cancela la operación de importar y se le muestra al usuario un mensaje informándole que la operación ha sido cancelada.	1. *ManagerMaps*. 2. Seleccionar la opción Importar. 3. Se muestra un explorador para que el usuario seleccione el fichero de extensión *.map* que se desea importar. 4. Dar clic en abrir. 5. Cuando el usuario está seleccionando el archivo a importar este selecciona la opción cancelar.
1.4 Seleccionar un archivo cuya configuración no es válida.	El usuario selecciona un archivo que no tiene una configuración correcta.	N/A	N/A	N/A	N/A	Al seleccionar un fichero y presionar el botón abrir la aplicación comprueba que el archivo está configurado correctamente y de no ser así le muestra un	1. *ManagerMaps*. 2. Seleccionar la opción Importar. 3. Se muestra un explorador para que el usuario seleccione el fichero con extensión *.map* que se desea importar. 4. Dar clic en abrir. 5. Se muestra un cartel al usuario informándole

						mensaje al usuario informándole que el fichero tiene una configuración incorrecta.	que el fichero tiene una configuración incorrecta.

Tabla 4: Variables del Caso de prueba Importar fichero *MapFile*

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
#MapFile	Numérico	Campo de texto	No	Cadena de números que corresponden al identificador del *MapFile*, con capacidades de 1 hasta 50 caracteres.
Nombre	Letras	Campo de texto	No	Conjunto de caracteres de solo letras con capacidades de 1 hasta 50 caracteres. El nombre que tiene el fichero.
Projection	Alfanumérico	Campo de texto	No	Conjunto de caracteres alfanuméricos
Fecha	Numérico	Campo de texto	No	Conjunto de caracteres numéricos escritos de la forma: DD-MM-AA, corresponden a la fecha en que se creó el *MapFile*.

3.4.2 Resultados de las pruebas realizadas

Una vez diseñados los casos de prueba para cada requisito funcional, se procedió a la ejecución de las pruebas. Los resultados se pueden observar en la Figura 12, donde se muestran la cantidad de no conformidades encontradas. Con la realización de las pruebas de caja negra aplicando la técnica de partición equivalente. Fueron detectadas un total de 7 no conformidades de funcionalidad durante 3 iteraciones, las cuales fueron corregidas a medida que se fue avanzando en el proceso de prueba:

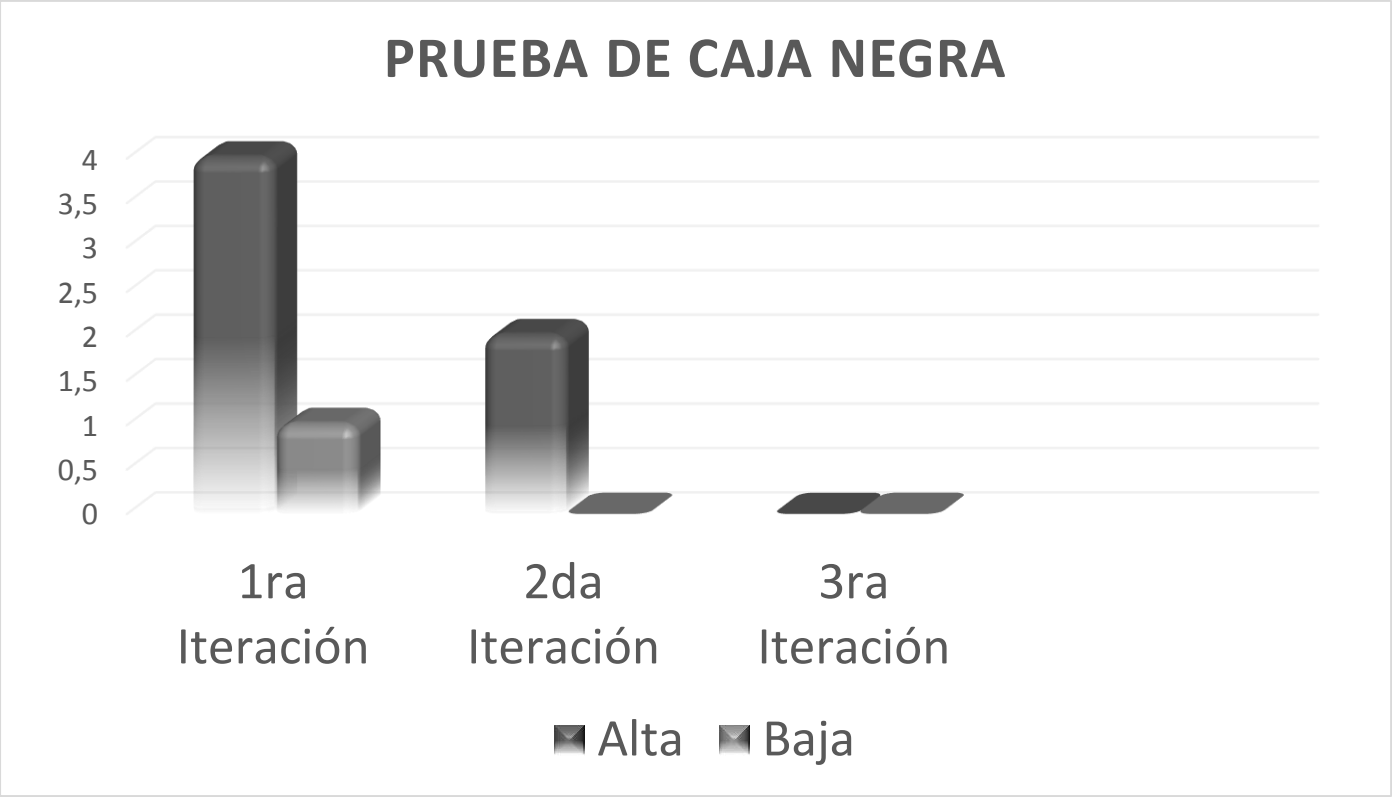


Fig. 12: Resultados de pruebas de caja negra

A continuación, en el "**Registro de defectos y dificultades detectados"** se muestran ejemplos de las no conformidades detectadas durante las pruebas a la aplicación y la respuesta del equipo de desarrollo:

Tabla 5: Registro de defectos y dificultades detectados

Elemento	No.	No conformidad (NC)	Alta	Baja	Tipo de error	Resp. equipo de desarrollo
Aplicación	1	Cuando el usuario selecciona la opción Asignar elemento *LAYER*, la aplicación permite insertar la misma *LAYER* en el mapa más de una vez.	X		Funcionalidad	Se validó que al asignar un elemento *LAYER* a un mapa, esta no se haya asignado al mapa con anterioridad.
Aplicación	2	Cuando se inserta un nuevo objeto *SYMBOL*, el	X		Funcionalidad	Se modificó el *link* del botón que

		botón Ver todos los objetos *SYMBOL* no realiza ninguna acción.				estaba escrito de forma incorrecta.
Aplicación	3	Cuando el usuario selecciona un fichero *.map* para importarlo, la aplicación no valida que el tipo de fichero sea el correcto.	X		Funcionalidad	Al enviar el fichero para el servidor se validó que este tuviera la extensión *.map* y que su tamaño no excediera los 2 *MB*.
Aplicación	4	Cuando se exporta el fichero *.map*, la aplicación no muestra un cartel al usuario informándole que el fichero se ha exportado.		X	Funcionalidad	Al exportar el archivo *MapFile* en la página *mapfilesExistentes.html.twig* se mostró un cartel anunciando que el fichero *MapFile* se ha exportado correctamente.
Aplicación	5	Al seleccionar la opción visualizar mapa, este no se muestra porque existen errores en la estructura del fichero *MapFile* seleccionado para visualizarlo.	X		Funcionalidad	Al escribir la propiedad *Extent* del objeto *Map*, esta no se había escrito correctamente.
Aplicación	6	Al seleccionar la opción visualizar mapa de un fichero *MapFile* que posee símbolos, estos no se muestran al visualizarlos.	X		Funcionalidad	La opción insertar elemento *SYMBOL* al fichero *MapFile* no estaba implementada de manera correcta

						por lo que no se asignaban los símbolos al *MapFile*.
Aplicación	7	Al crear un nuevo *OUTPUTFORMAT*, la aplicación no valida que el atributo *Size* sea un valor numérico.	X		Funcionalidad	En la clase nuevo*Outputformat.html.twig* se validó que el atributo *Size* sea un valor numérico.

3.5 Conclusiones del capítulo

Como resultado de este capítulo se obtuvo el diagrama de componentes el cual muestra la estructura física que tiene el componente de *software* y el diagrama de despliegue que permitió modelar la arquitectura del sistema. Con la realización de las pruebas de caja negra aplicando la técnica de partición equivalente se determinaron 5 no conformidades en su primera iteración y en la segunda iteración 2, a las cuales se le dieron solución. El proceso de pruebas desarrollado permitió identificar y resolver los errores en la implementación aumentando la calidad del producto final obtenido.

Conclusiones

- El análisis de las soluciones existentes arrojó que no existen propuestas de implementación que satisfagan los requerimientos del problema de la investigación actual. Aun así, varias de sus funcionalidades fueron contempladas en la propuesta actual.
- Las herramientas y tecnologías utilizadas (libres y multiplataforma) para resolver el problema planteado permiten que *ManagerMaps* sea un producto informático que contribuye a la soberanía tecnológica que impulsa el país y la universidad.
- La documentación técnica generada durante la investigación facilita que un futuro la herramienta pueda ser ampliada y perfeccionada.
- Se cumplió el objetivo planteado logrando desarrollar una herramienta para la gestión del fichero *MapFile* de *MapServer* 6.4 en los SIG.
- El modelo arquitectónico utilizado, y la utilización de cada patrón de diseño proporciona uniformidad, robustez y flexibilidad a la solución desarrollada.
- La aplicación de las pruebas de caja negra permitió detectar los errores obtenidos durante el proceso de implementación aumentando la calidad del producto final obtenido.

Recomendaciones

Se recomienda agregarle a la aplicación, la carga de una plantilla con la estructura de las versiones superiores de *MapServer* para su edición y futura explotación, de esta manera se brinda la posibilidad de darle soporte a versiones superiores del *MapServer* 6.4.

Referencias

Álvarez de Zayas, Carlos. 1995. *Metodología de la investigación científica.* 1995.

Álvarez, Miguel Ángel. 2001. *Desarrollo Web.* 2001.

ArcCatalog. [En línea] http://desktop.arcgis.com/es/desktop/latest/manage-data/using-arccatalog/what-is-arccatalog-.htm.

Booch, Grady, Jacobson, Ivar y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

Burrough, P y MCDonnell, R. 1998. *Principles of Geographic Systems.* 1998.

Carrasco, A S. 2013. *sistema de Información Geográfico para el Grupo Estatal AZCUBA.* La Habana : s.n., 2013.

Castillo, R, G y Martínez, V. 2013. *Herramienta para la gestión de mapas en Sistemas de Información Geográfica.* 2013.

Clements, P, Bass, L y Kazman, R. 2003. *Software Architecture in Practice.* 2003.

—. 2003. *Software Architecture in Practice. .* s.l. : SEI Series in Software Engineering: Addison Wesley, 2003.

Diewald, Verena y Christl, Arnulf. 2007. *Practical Introduction to MapStorer.* 2007.

Espinosa A, M. 2008. *Fundamentos del mapserver, mapscript, postgis y su integraci´on con el cartoweb.* 2008.

—. 2008. *Fundamentos del mapserver, mapscript, postgis y su integración con el cartoweb.* 2008.

García y Bellido, Antonio. 2012. *España y los españoles hace dos mil años: según la Geografía de Strábon.* Buenos Aires : s.n., 2012.

Gauchat, Juan Diego. 2012. *El gran libro de HTML5, CSS3 y Javascript.* Barcelona : s.n., 2012. 08007.

GeoNetwork. 2007-2009. *GeoNetwork Opensource El manual completo.* 2007-2009.

Guerrero Martínez, Rafael. 2010. PostgreSQL-es. *PostgreSQL-es.* [En línea] 2 de octubre de 2010. http://www.postgresql.org.es.

2009. gvSIG Asociación. [En línea] Generalidad Valenciana, 2009. http://web.gvsig-training.com.

International, Visual Paradigm. 2008. Visual Paradigm. [En línea] 2008. http://www.visual-paradigm.com/product/vpuml/features/.

Jacobson, et al. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

Jacobson, I, Booch, G y et al. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

Ledesma, Rubén. 2008. *Introducción al Bootstrap.* 2008.

Manso Callejo, Miguel Angel y Ballari, Daniela. *Anexo: El archivo Map.* Madrid : s.n.

Noviembre 2015. *MapServer 6.4 Documentation.* Noviembre 2015.

MapServer. 2008. *MapServer Documentation Release 6.4.1.* Minesota : s.n., 2008.

MARABOLI ROSSELOTT, MARCELO. 2003. *MANUAL de programación en PHP.* VALPARAÍSO : s.n., 2003.

Mariño, Carlos Vázquez. Septiembre 2008. *Programación en PHP 5.* Ferrol : s.n., Septiembre 2008.

Martínez, Rafael. 2013. PostgreSQL: Portal en español sobre PostgreSQL. [En línea] 2013. http://www.postgresql.org.es.

Mestras, Juan Pavón. 2014. *Bootstrap 3.0. Aplicaciones Web/Sistemas Web.* Madrid : s.n., 2014.

Mitchell, T. 2005. *Web Mapping Ilustrated: Using Open Source GIS Toolkits.* 2005.

Montiel, Alexis Puente. 2014. Miscelánea Natural. [En línea] 2014. http://www.miscelaneanatural.org/informatica/sig/la-trampa-del-java-o-por-que-gvsig-no-es-realmente-un-sig-libre.

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

Olaya, V. 2010. *Sistema de Información Geográfica.* 2010.

OSGeoLive. [En línea] http://live.osgeo.org/es/quickstart/geonetwork_quickstart.html.

Otto, Mark y Thornton, Jacob. 2006. *Bootstrap 3.* 2006.

Pacheco, Nacho. 2011. *Symfony2-es.* 2011.

Palomo Duarte, Manuel y Montero Pérez, Ildefonso. *Programación en PHP a través de ejemplos.* Sevilla : s.n.

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

Parmenter, Barbara. 2009. *Using ArcCatalog to Preview Data and Examine Metadata.* 2009.

Pérez, Damián Valdés. 2007. Maestros del Web. [En línea] 2007. http://www.maestrosdelweb.com.

PostgreSQL. 2013. PostgreSQL Sitio oficial del Servidor de Base de datos. [En línea] 2013.

Potencier, Fabien y Weaver, Ryan. 2009. *Symfony 2.* 2009.

Pressman. 2010. *Ingeniería del Software, un enfoque práctico.* Madrid : s.n., 2010.

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

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

—. 2012. *Plataforma GENESIG: dando pasos hacia un entorno geosemántico.* La Habana : s.n., 2012.

Ramsey, P y Santilli, S. 2015. *Postgis Documentation.* 2015.

Reyna, A. 2005. *El uso de los sistemas de información geográfica en el análisis demográfico de situaciones de desastre.* 2005.

Rodríguez, Tamara Sánchez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

Rumbaugh y Booch. 2000. *El Lenguaje Unificado de Modelado.* Madrid : s.n., 2000.

S, Belmonte, S y Nuñez, V. 2009. *Desarrollo de modelos hidrológicos con herramientas SIG.* 2009.

Sæther Bakken, Stig, y otros. 2002. *Manual de PHP.* 2002.

Sánchez, M, R P y Baños, V, Y. 2007. *Sistema generador de Mapas Temñaticos y Grñaficos.* 2007.

Sinavef. 2011. *Servidor de Mapas.* Guadalajara : s.n., 2011.

Softonic. softonic. *softonic.* [En línea] http://gvsig.softonic.com/.

Star, J y Estes, E, J. 1990. *Geographic Information Systems.* 1990.

Symfony. 2015. *Symfony en pocas palabras.* 2015.

—. 2009. Symfony.es. [En línea] 2009. http://symfony.es/noticias/2009/03/06/asi-seran-las-novedades-de-symfony-20/.

The Apache Software Foundation. 2016. Apache: HTTP Server Project. [En línea] 2016. http://httpd.apache.org/ABOUT_APACHE.html.

Valderrey Sanz, Pablo. 2011. *Administración de Sistemas Gestores de Base de Datos.* Madrid : s.n., 2011.

Valdés Pérez, Damián. 2007. Maestros del Web. *Maestros del Web.* [En línea] 3 de julio de 2007. http://www.maestrosdelweb.com.

Vázquez Mariño, Carlos. 2008. *Programación en PHP 5.* Ferrol : s.n., 2008.

ZANINOTTO, F y POTENCIER, F. 2009. librosweb.es. Symfony 1.1, la guía definitiva. [En línea] 2009. http://www.librosweb.es/symfony_1_1.

Bibliografía

Álvarez de Zayas, Carlos. 1995. *Metodología de la investigación científica.* 1995.

Álvarez, Miguel Ángel. 2001. *Desarrollo Web.* 2001.

ArcCatalog. [En línea] http://desktop.arcgis.com/es/desktop/latest/manage-data/using-arccatalog/what-is-arccatalog-.htm.

Booch, Grady, Jacobson, Ivar y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* 2000.

Burrough, P y MCDonnell, R. 1998. *Principles of Geographic Systems.* 1998.

Carrasco, A S. 2013. *sistema de Información Geográfico para el Grupo Estatal AZCUBA.* La Habana : s.n., 2013.

Castillo, R, G y Martínez, V. 2013. *Herramienta para la gestión de mapas en Sistemas de Información Geográfica.* 2013.

Clements, P, Bass, L y Kazman, R. 2003. *Software Architecture in Practice.* 2003.

—. 2003. *Software Architecture in Practice. .* s.l. : SEI Series in Software Engineering: Addison Wesley, 2003.

Diewald, Verena y Christl, Arnulf. 2007. *Practical Introduction to MapStorer.* 2007.

Espinosa A, M. 2008. *Fundamentos del mapserver, mapscript, postgis y su integraci´on con el cartoweb.* 2008.

—. 2008. *Fundamentos del mapserver, mapscript, postgis y su integración con el cartoweb.* 2008.

García y Bellido, Antonio. 2012. *España y los españoles hace dos mil años: según la Geografía de Strábon.* Buenos Aires : s.n., 2012.

Gauchat, Juan Diego. 2012. *El gran libro de HTML5, CSS3 y Javascript.* Barcelona : s.n., 2012. 08007.

GeoNetwork. 2007-2009. *GeoNetwork Opensource El manual completo.* 2007-2009.

Guerrero Martínez, Rafael. 2010. PostgreSQL-es. *PostgreSQL-es.* [En línea] 2 de octubre de 2010. http://www.postgresql.org.es.

2009. gvSIG Asociación. [En línea] Generalidad Valenciana, 2009. http://web.gvsig-training.com.

International, Visual Paradigm. 2008. Visual Paradigm. [En línea] 2008. http://www.visual-paradigm.com/product/vpuml/features/.

Jacobson, et al. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

Jacobson, I, Booch, G y et al. 2000. *El Proceso Unificado de Desarrollo de Software.* Madrid : s.n., 2000.

Ledesma, Rubén. 2008. *Introducción al Bootstrap.* 2008.

Manso Callejo, Miguel Angel y Ballari, Daniela. *Anexo: El archivo Map.* Madrid : s.n.

Noviembre 2015. *MapServer 6.4 Documentation.* Noviembre 2015.

MapServer. 2008. *MapServer Documentation Release 6.4.1.* Minesota : s.n., 2008.

MARABOLI ROSSELOTT, MARCELO. 2003. *MANUAL de programación en PHP.* VALPARAÍSO : s.n., 2003.

Mariño, Carlos Vázquez. Septiembre 2008. *Programación en PHP 5.* Ferrol : s.n., Septiembre 2008.

Martínez, Rafael. 2013. PostgreSQL: Portal en español sobre PostgreSQL. [En línea] 2013. http://www.postgresql.org.es.

Mestras, Juan Pavón. 2014. *Bootstrap 3.0. Aplicaciones Web/Sistemas Web.* Madrid : s.n., 2014.

Mitchell, T. 2005. *Web Mapping Ilustrated: Using Open Source GIS Toolkits.* 2005.

Montiel, Alexis Puente. 2014. Miscelánea Natural. [En línea] 2014. http://www.miscelaneanatural.org/informatica/sig/la-trampa-del-java-o-por-que-gvsig-no-es-realmente-un-sig-libre.

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

Olaya, V. 2010. *Sistema de Información Geográfica.* 2010.

OSGeoLive. [En línea] http://live.osgeo.org/es/quickstart/geonetwork_quickstart.html.

Otto, Mark y Thornton, Jacob. 2006. *Bootstrap 3.* 2006.

Pacheco, Nacho. 2011. *Symfony2-es.* 2011.

Palomo Duarte, Manuel y Montero Pérez, Ildefonso. *Programación en PHP a través de ejemplos.* Sevilla : s.n.

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

Parmenter, Barbara. 2009. *Using ArcCatalog to Preview Data and Examine Metadata.* 2009.

Pérez, Damián Valdés. 2007. Maestros del Web. [En línea] 2007. http://www.maestrosdelweb.com.

PostgreSQL. 2013. PostgreSQL Sitio oficial del Servidor de Base de datos. [En línea] 2013.

Potencier, Fabien y Weaver, Ryan. 2009. *Symfony 2.* 2009.

Pressman. 2010. *Ingeniería del Software, un enfoque práctico.* Madrid : s.n., 2010.

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

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

—. 2012. *Plataforma GENESIG: dando pasos hacia un entorno geosemántico.* La Habana : s.n., 2012.

Ramsey, P y Santilli, S. 2015. *Postgis Documentation.* 2015.

Reyna, A. 2005. *El uso de los sistemas de información geográfica en el análisis demográfico de situaciones de desastre.* 2005.

Rodríguez, Tamara Sánchez. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* La Habana : s.n., 2015.

Rumbaugh y Booch. 2000. *El Lenguaje Unificado de Modelado.* Madrid : s.n., 2000.

S, Belmonte, S y Nuñez, V. 2009. *Desarrollo de modelos hidrológicos con herramientas SIG.* 2009.

Sæther Bakken, Stig, y otros. 2002. *Manual de PHP.* 2002.

Sánchez, M, R P y Baños, V, Y. 2007. *Sistema generador de Mapas Temñaticos y Grñaficos.* 2007.

Sinavef. 2011. *Servidor de Mapas.* Guadalajara : s.n., 2011.

Softonic. softonic. *softonic.* [En línea] http://gvsig.softonic.com/.

Star, J y Estes, E, J. 1990. *Geographic Information Systems.* 1990.

Symfony. 2015. *Symfony en pocas palabras.* 2015.

—. 2009. Symfony.es. [En línea] 2009. http://symfony.es/noticias/2009/03/06/asi-seran-las-novedades-de-symfony-20/.

The Apache Software Foundation. 2016. Apache: HTTP Server Project. [En línea] 2016. http://httpd.apache.org/ABOUT_APACHE.html.

Valderrey Sanz, Pablo. 2011. *Administración de Sistemas Gestores de Base de Datos.* Madrid : s.n., 2011.

Valdés Pérez, Damián. 2007. Maestros del Web. *Maestros del Web.* [En línea] 3 de julio de 2007. http://www.maestrosdelweb.com.

Vázquez Mariño, Carlos. 2008. *Programación en PHP 5.* Ferrol : s.n., 2008.

ZANINOTTO, F y POTENCIER, F. 2009. librosweb.es. Symfony 1.1, la guía definitiva. [En línea] 2009. http://www.librosweb.es/symfony_1_1.

Anexos

Interfaces de la aplicación

A continuación, se muestran algunas interfaces de la aplicación consideradas las más importantes. En la figura 13 se muestra la interfaz principal, la cual posee a la derecha un menú con todas las funcionalidades del sistema y a la derecha se aprecia los ficheros *MapFile* creados recientemente.

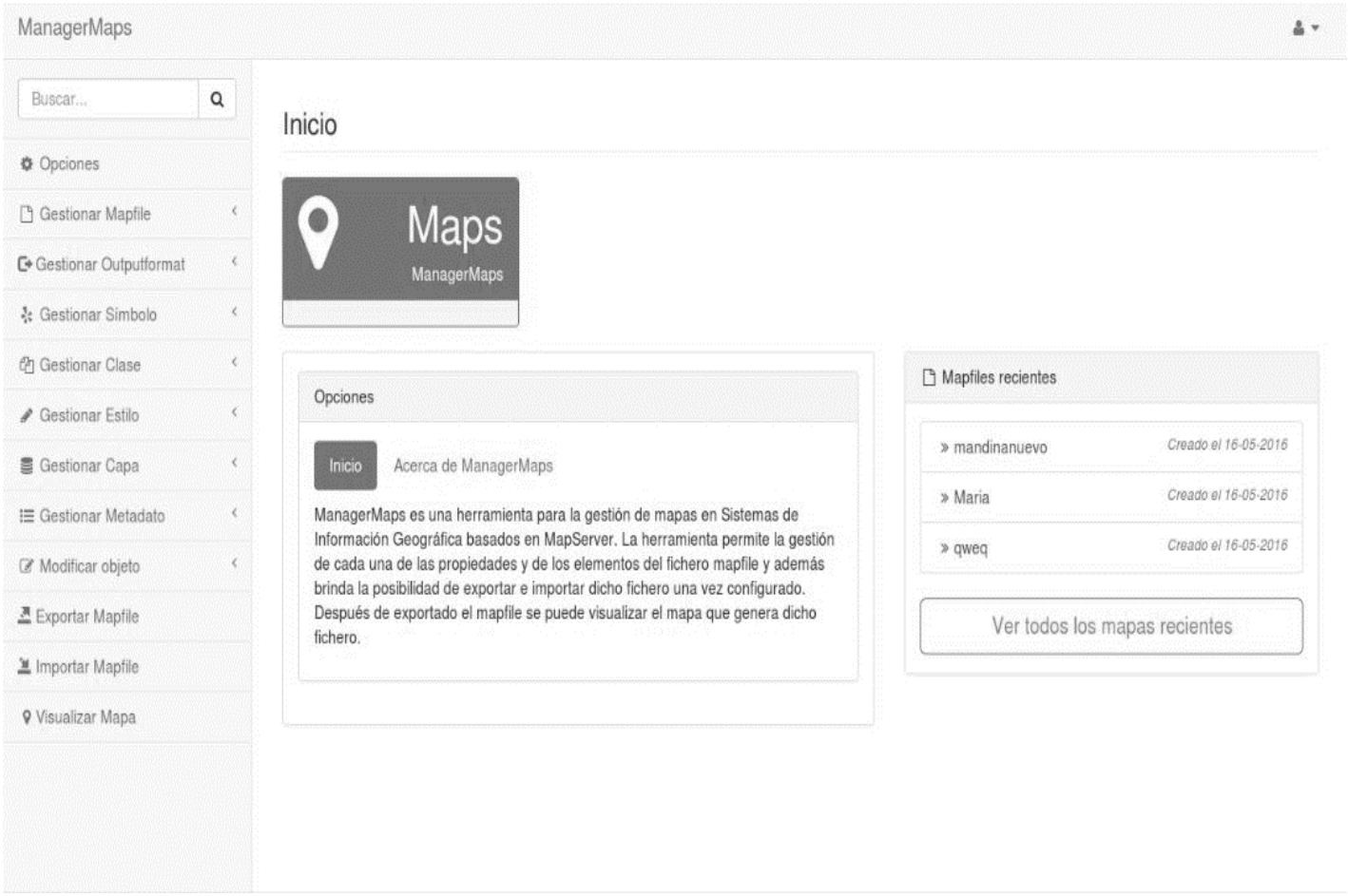


Fig. 13: Interfaz principal de la aplicación

La figura 14 muestra la interfaz de cómo se inserta un nuevo símbolo, en la misma se puede apreciar un formulario donde el usuario debe introducir los atributos que este posee.

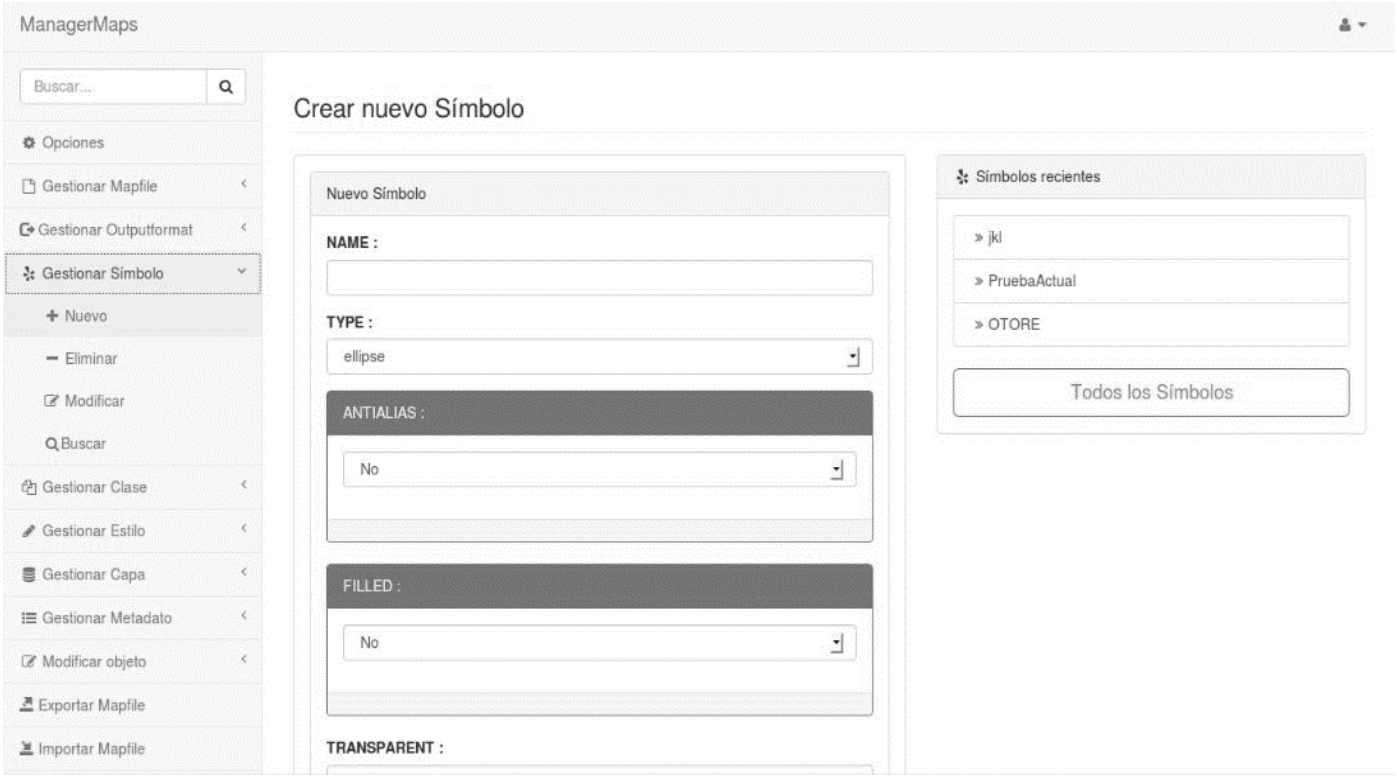


Fig. 14: interfaz para Insertar un nuevo símbolo

La figura 15 muestra en una tabla todos los *OUTPUTFORMAT*s existentes.

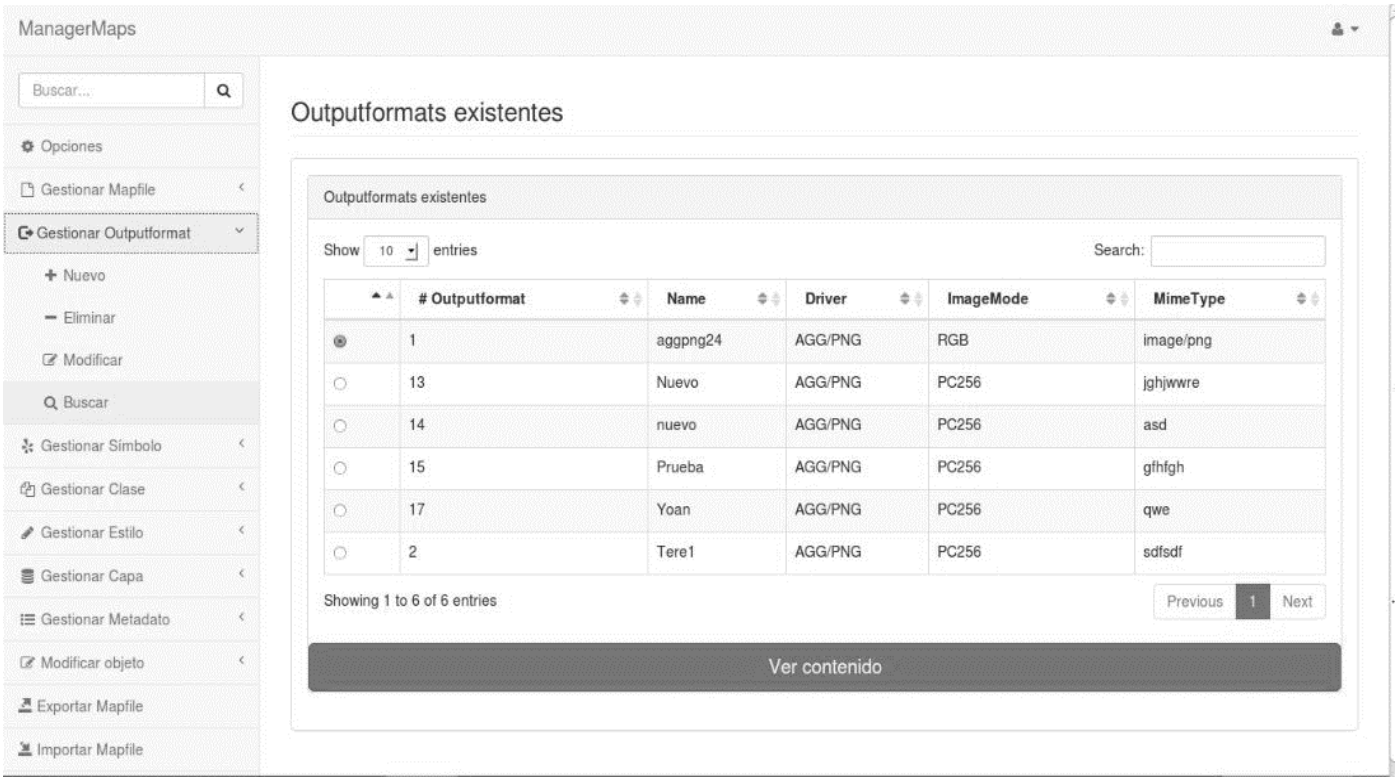


Fig. 15: Interfaz con los *OUTPUTFORMAT*s existentes.

La figura 16 muestra el contenido que posee el *MapFile* una vez que se genere.

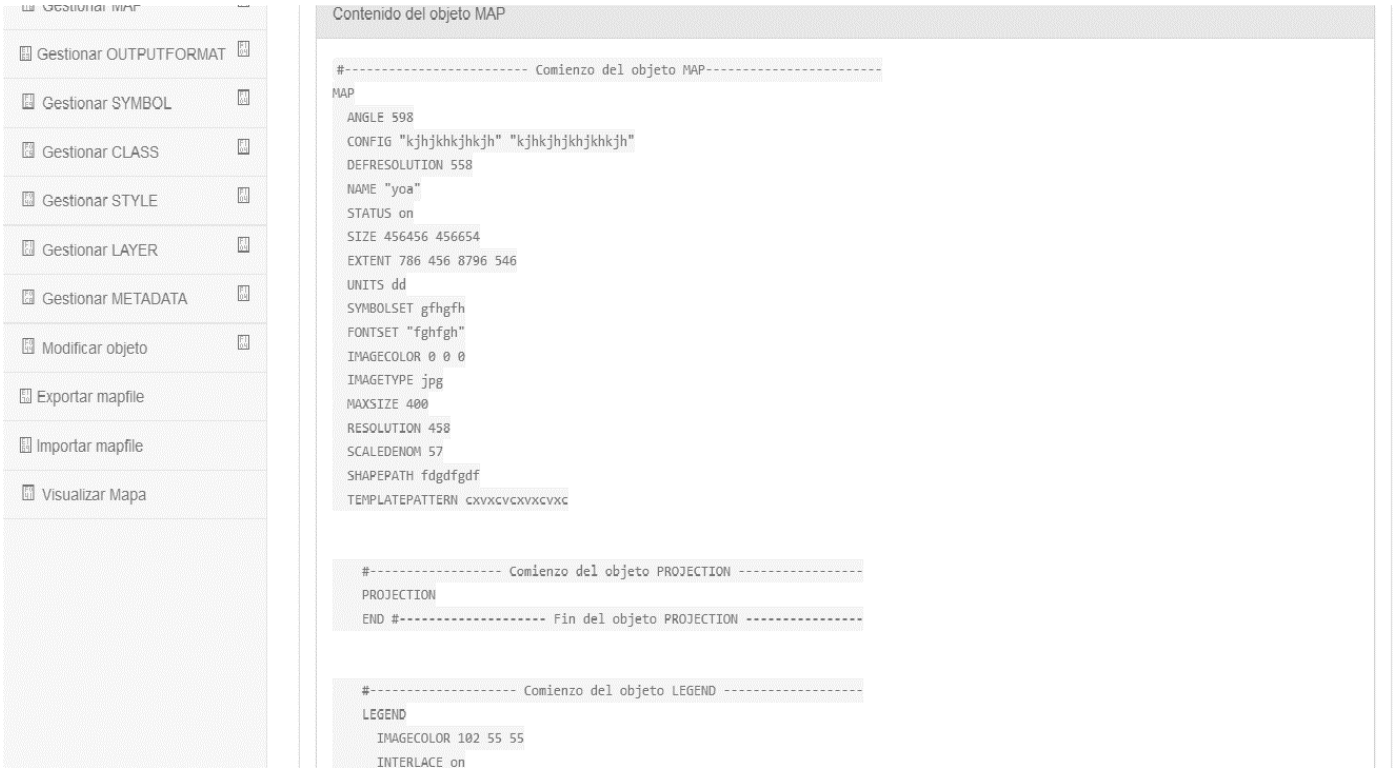


Fig. 16: Contenido del objeto *.map*.