UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Facultad 6

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

Complemento generador de archivos MapFile para GeoQ (GAMF-GeoQ)

AUTORES: Bárbaro Pérez Mallén

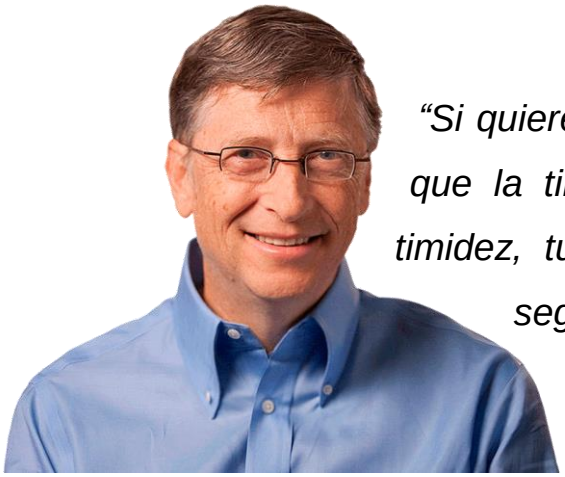
Thaylisis García Martínez

TUTORES: Msc. Daniel Echevarría González

Ing. Sandor Escobar Ruiz

La Habana, 23 de junio del 2014

"Año 56 de la Revolución"



“Si quieres hacer algo grande en tu vida, debes recordar que la timidez está solo en la mente. Si piensas con timidez, tus actos serán similares. Pero si piensas con seguridad, actuarás de la misma forma. Por ello nunca dejes que la timidez conquiste tu mente”

Bill Gates

Declaración de Autoría:

Declaramos por este medio que nosotros, Bárbaro Pérez Mallén con carnet de identidad 90112705724 y Thaylisis García Martínez con carnet de identidad 91100725296, somos autores de la presente tesis y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste se firma la presente a los ____ días del mes de **Junio** del año **2014**.

_____	_____
Firma del Autor	Firma del Autor
Bárbaro Pérez Mallén	Thaylisis García Martínez
_____	_____
Firma del Tutor	Firma del Tutor
Daniel Echevarría González	Sandor Escobar Ruiz

Datos de Contacto

Msc. Daniel Echevarría González

Graduado de la Universidad de la Habana en el año 1997, pertenece al Departamento Geoinformática del Centro de Geoinformática y Señales Digitales (GEYSED) de la Facultad 6. Se desempeña en el proyecto Aplicativos SIG, donde es el Profesor. Correo electrónico: danielec@uci.cu.

Ing. Sandor Escobar Ruiz

Graduado de la Universidad de las Ciencias Informáticas (UCI) en el año 2012, pertenece al Departamento Geoinformática del Centro de Geoinformática y Señales Digitales (GEYSED) de la Facultad 6. Se desempeña como jefe en el proyecto SIG-Desktop. Correo electrónico: sescobar@uci.cu.

Agradecimientos

Dedicatoria

Resumen

Con el avance de las Tecnologías de la Información y las Comunicaciones (TICs), los Sistemas de Información Geográfica (SIG) han cobrado un gran interés, por las múltiples aplicaciones que tienen este tipo de plataformas. Así mismo se han incrementado las empresas que se dedican a la creación del software. La Universidad de las Ciencias Informáticas (UCI) es uno de esos centros que se vinculan a la producción de software, donde se encuentran los proyectos SIG-Desktop que se dedica al desarrollo de SIG para entornos de escritorios utilizando como base a la plataforma GeoQ, y Aplicativos SIG que desarrolla SIG para entorno Web, empleando a la plataforma GeneSIG junto a MapServer.

La actual investigación, surge a partir de la necesidad que existe de lograr compatibilidad entre las representaciones de los mapas que realizan GeoQ y MapServer. Asumiendo como objetivo, desarrollar un complemento generador de archivos MapFile para la plataforma GeoQ a partir de un archivo de proyecto. Permitiendo así portar las configuraciones de los mapas de un SIG a otro. El generador de MapFile, está desarrollado en el lenguaje de programación C++, auxiliándose en el marco de trabajo QT, y utilizando *Q*T *Creator* como Entorno Integrado de Desarrollo (IDE). Para lograr la meta propuesta, fue utilizado como metodología para el desarrollo de software, el Proceso Unificado Ágil (AUP), utilizando las técnicas de modelación establecidas por el Lenguaje Unificado de Modelado (UML)**.**

Palabras Claves

C++, MapServer, GeoQ, MapFile y representación.

Abstract

With the advancement in the Information and Communications Technology (ICTs), the Geographic Information Systems (GIS) has become of great interest for the many applications that this type of platform have. Also the amount of companies dedicated to the creation of software have increased. The University of Informatics Sciences (UIS) is one of those centers that are linked to the production of software, in SIG-Desktop projects are dedicated to the development of GIS desktop environments using as a basis the GeoQ platform, and AplicativosSIG developing Web GIS environment, using GeneSIG platform next to MapServer.

The current research arises from need that exists of achieving compatibility between the performance of the maps that GeoQ and MapServer accomplish. Assuming as objective, to develop a plugin generator for MapFile files to the GeoQ platform from a project file. Permitting porting configurations maps from one GIS to another. The MapFile generator, is developed in the programming language C++, with the framework QT, and using QT Creator as Integrated Development Environment (IDE). To achieve the proposed goal, was used as a methodology for software development, Agile Unified Process (AUP), using the modeling techniques set up in the Unified Modeling Language (UML).

Key Words

C++, GeneSIG, GeoQ, MapFile and representation.

Índice de Contenido:

Índice de Contenido

Índice de Tablas:

Índice de Figuras:

Introducción

El desarrollo de las Nuevas Tecnologías de la Información y las Comunicaciones (NTIC), está en constante progreso. Actualmente, han alcanzado una importancia social muy elevada, al punto de ser imprescindibles en el avance de la humanidad. Las mismas se encuentran concentradas principalmente en la Informática, las Telecomunicaciones y el Internet. Específicamente en el grupo de la informática, existen los Sistema de Información Geográfica (SIG), que manejan información geográfica y cartográfica. Estos sistemas son utilizados para resolver problemas complejos de planificación y gestión de información geográfica, como la gestión de los recursos, y otros. Han revolucionado la forma de crear y manejar los mapas, brindando la posibilidad de analizar y gestionar territorios, a través de la cartografía digital de forma muy rápida.

Gracias a su continuo avance y el aumento de las necesidades de uso, ha crecido el desarrollo de estos sistemas sobre plataformas o tecnologías libres. Hoy en día, el software libre juega un papel fundamental en el proceso de “independencia tecnológica” en que se encuentran numerosos países en progreso, debido a las ventajas, características y flexibilidades que brinda. Cuba es uno de esos países que se encuentra dedicado a ese proceso, con un gran futuro en la explotación de nuevas tecnologías de la ciencia y la computación.

Teniendo el país como objetivo, la independencia tecnológica y la informatización de la sociedad, en la Universidad de las Ciencias Informática (UCI), se han desarrollado sistemas para el avance de los SIG en la Web, un ejemplo es la plataforma denominada GeneSIG. También existe GeoQ, que está desarrollado para entornos de escritorio. Ambas son utilizadas por el Departamento de Geoinformática, perteneciente al Centro de Desarrollo Geoinformática y Señales Digitales (GEYSED) de la Facultad 6. Este cuenta con varios proyectos, entre ellos SIG-Desktop y Aplicativos SIG. Los cuales, utilizan en su proceso de desarrollo de SIG a GeoQ y GeneSIG respectivamente. Este último emplea además el servidor de mapas MapServer para la publicación de datos espaciales y aplicaciones cartográficas para la Web.

Actualmente estos proyectos para facilitar el trabajo que realizan, tienen la posibilidad de compartir la información que representan con el servidor MapServer y GeoQ. Esto se lleva a cabo utilizando como punto en común a los orígenes de datos, ya sea a través de bases de datos (SDE, PostGIS u Oracle), archivos

con formato Shapefile[1], entre otros. En estos momentos, las vistas de los mapas que generan ambas herramientas para un mismo origen de datos no coinciden, como se explica la Fig**.** 1. Este escenario se debe a que manejan las configuraciones de los estilos, las simbologías y los tamaños[2] de modos diferentes.

Fig. 1: Mapa conceptual sobre las vistas que generan actualmente GeoQ y MapServer.

Hoy en día para lograr semejanzas en estas vistas es realmente difícil, debido a que en la configuración de los estilos y simbología entra en juego cierto volumen de información, que puede llegar a confundir a una persona. Estas aproximaciones se efectúan creando para MapServer las configuraciones que se visualizan en GeoQ y viceversa. Todo este procedimiento se realiza de forma manual, ocasionando además grandes pérdidas de tiempo y posibles errores en la transformación, pudiendo obtener estilos y deformaciones no deseadas.

A partir de la situación descrita anteriormente, surge el siguiente **problema a resolver**: ¿Cómo lograr una correcta interoperabilidad entre el servidor de mapas MapServer y la plataforma GeoQ? o La incompatibilidad de las vistas de mapas generadas entre GeoQ y MapServer, provoca la necesidad de lograr una correcta interoperabilidad.

Se define como **objeto de estudio**: El proceso de representación de mapas, enmarcando como **campo de acción:** Los archivos de configuración que intervienen en el proceso de representación de mapas. Para solucionar el problema planteado y limitar el alcance de la investigación, se traza como **objetivo general**:

[1] **Shapefile.** Es un formato multiarchivo que consta como mínimo de tres archivos, uno principal (.shp), uno de índice (.shx) y una tabla en formato dBASE (.dbf).

[2] Se refiere a las unidades de medidas que puede tener determinado elemento, por ejemplo el tamaño de un símbolo en GeoQ se da en milímetros, mientras que en MapServer se mide por unidades.

Introducción

Desarrollar un plugin[3] para generar archivos MapFile en la plataforma GeoQ, a partir de un archivo de proyecto con extensión .qgs.

Para el cumplimiento del objetivo general se definieron los **objetivos específicos** que se proponen a continuación:

1) Analizar cómo se efectúa la representación de los mapas, para identificar sus principales características y patrones.
2) Definir los elementos del análisis y el diseño, para la elaboración del complemento de GeoQ.
3) Desarrollar la solución propuesta y validarla mediante pruebas.

Se desarrollarán las siguientes **tareas investigativas** para dar cumplimiento a los objetivos trazados:

1. Análisis del proceso de representación que despliegan los principales servidores de mapas.
2. Análisis de los principales archivos de configuración de los mapas.
3. Caracterización de la metodología de desarrollo, lenguajes y tecnologías a utilizar en la construcción de solución, así como de diferentes herramientas de generación de MapFile.
4. Identificación de la arquitectura y patrones de diseño que seguirá la aplicación.
5. Implementación de las funcionalidades identificadas para el complemento de GeoQ.
6. Ejecución de pruebas que validen la solución.

Como **posibles resultados** de esta investigación se encuentran:

- La elaboración de una herramienta para generar archivos MapFile en GeoQ.
- Documentación asociada al proceso de desarrollo del complemento.

Para dar cumplimiento a las tareas trazadas en este trabajo, se hace uso de diferentes **métodos de la investigación,** con el fin de obtener una mejor organización y estructura de la misma. Entre estos se encuentran los teóricos, que facilitan la construcción de modelos de investigación. También existen los

[3] **Complemento o plugin**. Un complemento es una aplicación que se relaciona con otra para aportarle una funcionalidad nueva.

empíricos, que representan un nivel de la investigación cuyo contenido, procede de la experiencia y es sometido a cierta elaboración racional (Universidad de las Ciencias Informáticas, 2010):

Métodos teóricos:

- ✓ **Analítico–Sintético**: para el estudio de la bibliografía relacionada con el proceso de representación cartográfica, con el objetivo de extraer las principales características que ahí se encuentran y se relacionan con este. Además, se utiliza para caracterizar las tendencias y tecnologías actuales, con el objetivo de seleccionar las herramientas más utilizadas en el área de desarrollo.
- ✓ **Modelación**: se emplea para representar los principales diagramas y características que propone la metodología de desarrollo, y que permitirán una mejor comprensión del generador de archivos MapFile para GeoQ.

Métodos empíricos:

- ✓ **Entrevista**: se lleva a cabo una entrevista, con el objetivo de entender el funcionamiento actual de creación de los archivos MapFile tanto fuera, como dentro del contexto de la integración de las plataformas GeneSIG y GeoQ. Para ello se entrevista a los líderes de los proyectos SIG-Desktop y Aplicativos SIG.

Este trabajo, para una mayor comprensión está desglosado en 3 capítulos, los cuales a su vez están divididos en epígrafes y subepígrafes. En el **Capítulo 1 "Fundamentación teórica de la representación cartográfica"**, se describen los temas relacionado con el problema que sustenta la investigación, realizando un análisis más detallado, para lograr mayor conocimiento de lo que se va a tratar en la solución del problema. Se fundamenta el objeto de estudio, su definición y características y se analizan soluciones que respondan en alguna medida al problema de la investigación, planteando sus ventajas y desventajas.

Por su parte el **Capítulo 2 "Análisis y diseño del complemento GAMF-GeoQ"**, define la metodología que se utilizará en desarrollo de la investigación. Además argumenta y valora las principales herramientas, tecnologías y que se utilizan para la construcción de la solución. También se describe el modelo del dominio, se determinan los requisitos funcionales y no funcionales con los que la aplicación debe cumplir, evidenciando el alcance de la investigación. Así mismo, se determinan los casos de uso del sistema, se realiza el diagrama de clases del dominio y se definen los estilos y patrones arquitectónicos.

Finalmente en el **Capítulo 3 “Implementación y prueba del complemento GAMF-GeoQ”**, se describe la solución, especificando el modelo de despliegue y de implementación. También se describe la contribución que aporta el plugin al proceso de representación cartográfica. Se especifican los estándares de codificación utilizados y se diseñan y aplican las pruebas al sistema.

Capítulo 1: Fundamentación teórica de la representación cartográfica

Con el objetivo de esclarecer el alcance de la investigación, en este capítulo se analizan los conceptos relacionados con el problema que sustenta la investigación. Se realiza una descripción del objeto de estudio, explicando el proceso de representación cartográfica y que elementos intervienen. También realiza un estudio del arte relacionado con el tema tratado, con el fin de analizar los distintos sistemas que existan y determinar cuáles son las mejores características a utilizar.

1.1 Definiciones Generales

Esta investigación se desarrolla en un entorno, donde es necesario conocer la definición de ciertos elementos que en el contexto, se relacionan de una forma con el objetivo de la investigación. Dichos elementos, básicamente responden al dominio del problema mediante una lógica de negocio, que incluye el uso de una serie de conceptos, entre los que se puede citar:

Sistemas de Información Geográficas

"Es un sistema de hardware, software y procedimientos diseñados para soportar la captura, administración, manipulación, análisis, modelamiento y graficación de datos u objetos referenciados espacialmente, para resolver problemas complejos de planeación y administración. Una definición más sencilla es: Un sistema de computador capaz de mantener y usar datos con localizaciones exactas en una superficie terrestre" (Carmona, y otros, 1999).

Según (Star, y otros, 1990) un SIG es un sistema de información diseñado para trabajar con datos referenciados mediante coordenadas espaciales o geográficas. En otras palabras, es tanto un sistema de base de datos con capacidades específicas para datos georreferenciados, como un conjunto de operaciones para trabajar con esos datos. En cierto modo, es un mapa de orden superior.

Finalmente, se llega a concluir que un SIG es un sistema que integra la tecnología e información geográfica, que permite gestionar los mapas de manera rápida y sencilla, aprovechando al máximo la información que puedan brindar. Sus principales funciones se resumen en capturar, analizar, almacenar, editar y representar datos georreferenciados.

Fundamentación teórica de la representación cartográfica

Cartografía

Según (Franco Maass, y otros, 2002) "*está estrechamente relacionada con la naturaleza, características y disponibilidad de los datos. Esta ciencia tiene por objeto el estudio de los mapas como método especial de representación de la realidad geográfica e incluye, desde el estudio multilateral de la esencia de los mapas, los procesos de su elaboración y confección, hasta los de su reproducción masiva y uso en problemas aplicados.* "

Según (Sanchez Menendez, 2009) "*Trata del establecimiento de cartas de todo tipo y engloba todas las fases de trabajo, desde los primeros levantamientos hasta la impresión final de los mapas. Se encarga de plasmar en una superficie plana toda la información posible acerca de una zona determinada de la superficie terrestre, representando con mayor o menor detalle las características más importantes que pudiesen existir, conservando en lo posible las relaciones angulares y lineales existentes entre los elementos representados"*.

Teniendo en cuenta lo descrito anteriormente, se puede decir que la cartografía es el conjunto de estudios y técnicas que intervienen en la formación o análisis de mapas que representan la Tierra, parte de ella, o cualquier parte del Universo.

Mapa

Según (Domínguez Bravo, 2000) *"Un mapa es una representación de la realidad y no la realidad misma. Para representar esa realidad deberemos de utilizar unas convenciones. En primer lugar la realidad a representar es generalmente volumétrica y por lo tanto implica un cambio de tres dimensiones a dos. Este cambio de tres a dos dimensiones se suele suplir describiendo la tercera dimensión como un atributo (así por ejemplo una cota de una montaña tendría una localización de coordenadas x e y que podemos leer sobre el mapa y un atributo, z, que sería la altura)"*.

Se puede concluir que un mapa es una representación gráfica de un territorio que no tiene que obligatoriamente reflejar como luce este, sino que puedes ser usado para mostrar la densidad poblacional, o los límites de determinado lugar.

1.2 Proceso de representación cartográfica

En la representación digital de todo mapa existen secciones que conforman las características básicas de estos, como la leyenda, el título, la escala, el mapa de ubicación y las referencias. Estos pueden ser presentados de diferentes maneras o ser obviados, dependiendo del uso que se le haya destinado. Toda la información geográfica se representa y administra por SIG mediante tres estructuras de datos principales: clases de entidad, tablas de atributos y capas[4] (Esri, 2012). Además estos sistemas cuentan con archivos que permiten guardar la configuración de los proyectos y estilos de mapas con los que trabajan. Estos son consultados al inicio del proceso e interpretados por el motor de renderización, que en algunos sistemas son de obligatoria inclusión, mientras que otros proporcionan inicialmente estilos aleatorios.

Normalmente, un SIG se utiliza para administrar varias capas distintas, cada una contiene información de una colección de entidades en particular (por ejemplo, carreteras), a la que se hace referencia geográficamente sobre la superficie de la Tierra. El mapa resultante del proceso se basa en una serie de datos, cada uno de ellos con una representación geográfica especificada. Por ejemplo, los elementos individuales pueden representarse como entidades (tales como puntos, líneas y polígonos), como imágenes mediante ráster, como superficies mediante entidades, y como atributos descriptivos contenidos en tablas (Esri, 2012).

Las representaciones geográficas se organizan en una serie de capas. La mayoría de estas son grupos de elementos geográficos simples, por ejemplo, una red de carreteras, una colección de límites de parcelas, tipos de suelo, una superficie de elevación, imágenes de satélite de una fecha determinada, ubicaciones de pozos o la superficie del agua. Las colecciones de datos espaciales normalmente se organizan como capas de clases de entidad o basados en ráster.

Pasos generales para la representación

De manera general el proceso inicia con el archivo de configuración, o varios de estos debido a que algunos sistemas utilizan más de uno. A partir de los cuales se obtienen el origen de los datos cartográficos, luego los estilos y demás configuraciones que se deben aplicar. Esta información es procesada por el motor de

[4] Es una colección de entidades homogéneas para cada tema.

renderizado obteniendo como resultado la imagen del mapa. Estos pasos son ejecutados por los sistemas de diferentes formas, ajustándose a la arquitectura y las capacidades particulares de cada una. La Fig. 2 muestra el proceso de representación y los elementos que intervienen, a partir de información obtenida de (Martínez Piedra, 2005) y consideraciones de los autores.

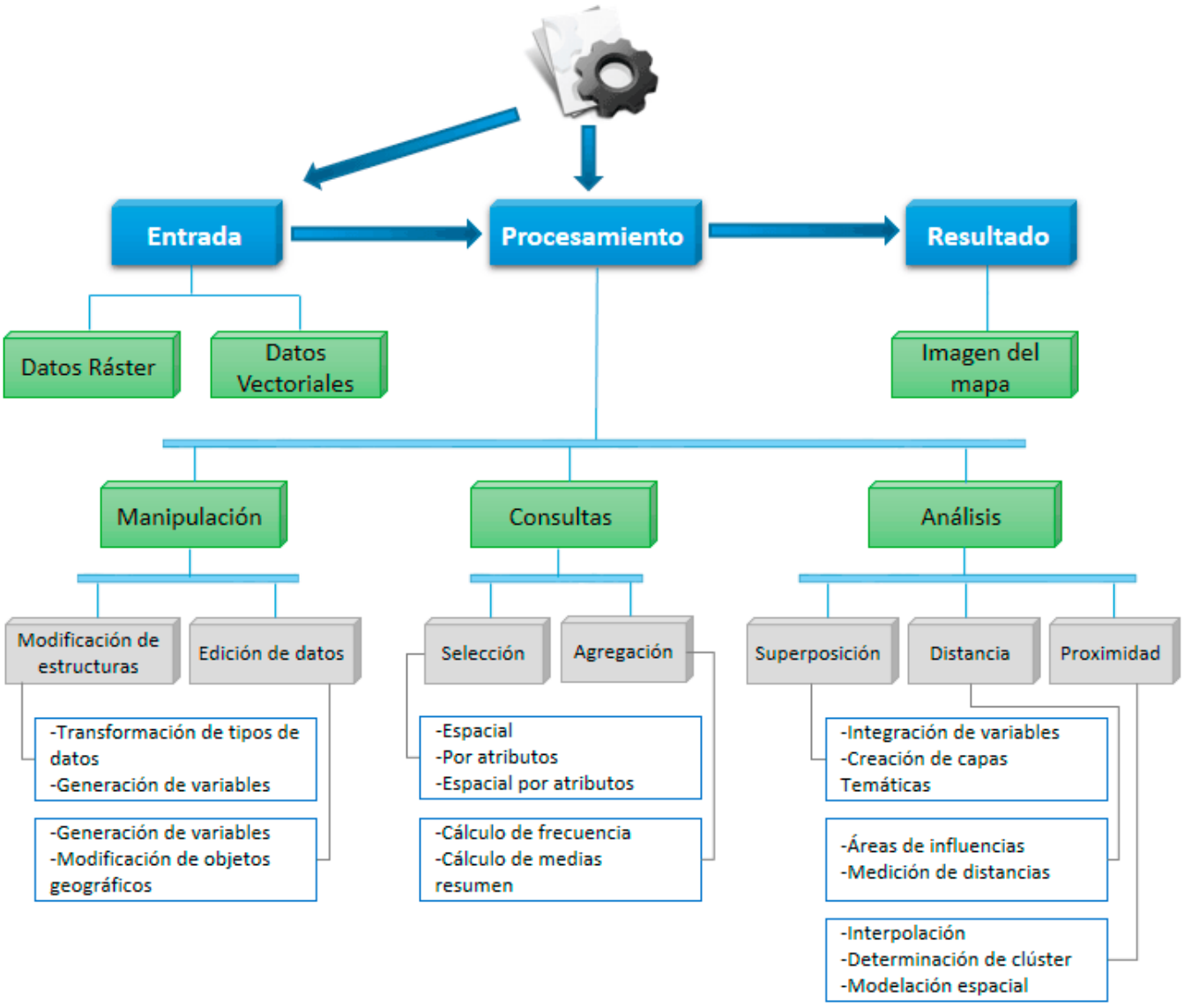


Fig. 2: Proceso de representación cartográfica.

Los datos de entrada con formato **vectorial**, consiste en cadenas de coordenadas y usa tres tipos de elementos gráficos para representar los objetos geográficos de un mapa: punto (nodo con coordenada x,y), línea (segmento con coordenadas x1,y1 … xn,yn), área (polígono con coordenadas x1,y1 x2,y2 … xn,yn x1,y1) (Martínez Piedra, 2005). La Fig. 3 muestra un ejemplo de estos.

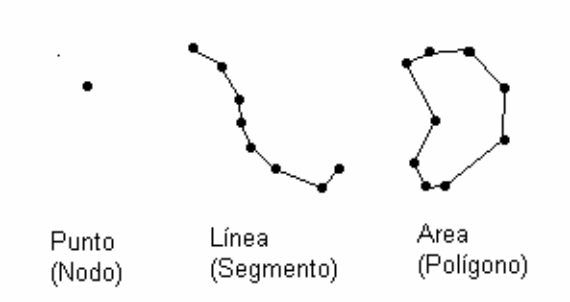


Fig. 3: Tipo de dato vector.

El formato **r**áster almacena datos espaciales en una matriz generada al dividir una imagen gráfica en una rejilla regular de celdas. Cada elemento de la matriz almacena un atributo que identifica cada celda, la que se representa por un píxel de la imagen. La posición de las celdas en la matriz provee información sobre la ubicación de datos espaciales (Martínez Piedra, 2005). La Fig. 4 muestra un ejemplo de estos.

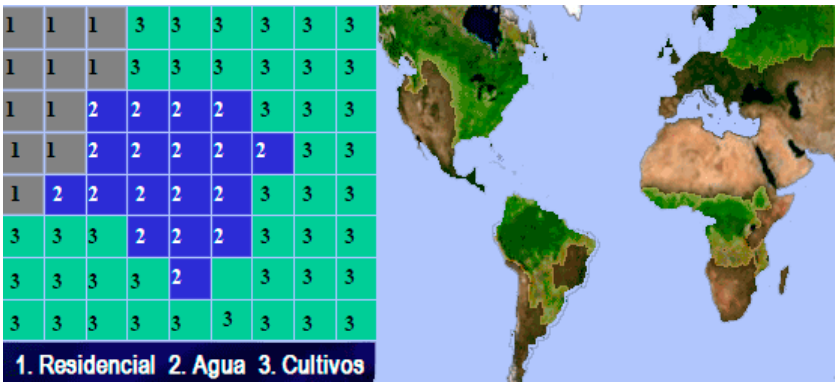


Fig. 4: Tipo de dato ráster.

1.3 Archivos de configuración de mapas.

Existen dos tipos de SIG, uno desarrollado para entornos de escritorios y otro destinado a la visualización de mapas en la WEB. Cada uno adapta el proceso de representación antes descrito según sus características y la arquitectura que posea. Estos tienen la posibilidad de compartir la información que representan y lograr interoperabilidad, utilizando como punto en común a los orígenes de datos ya sea a través de bases de datos, archivos Shapefile, entre otros. Además, cuentan con diferentes formas de configurar los proyectos y estilos de los mapas con los que trabajan. En esta investigación es de interés estudiar herramientas de código abierto (*open source*), con el objetivo de impulsar y divulgar el proceso de soberanía tecnológica que se lleva a cabo en la universidad.

Los SIG para entornos de escritorios hoy en día se pueden llevar a cabo en ordenadores personales o estaciones de trabajo, ya sea de forma individual o en una arquitectura cliente-servidor más complejo. Esta última es la utilizada por los SIG WEB y ha cobrado importancia muy rápidamente en los últimos tiempos, especialmente en lo que al acceso a datos se refiere. La Fig. 5 muestra de forma general la arquitectura cliente-servidor utilizada en los servidores de mapas (Muñoz, 2009).

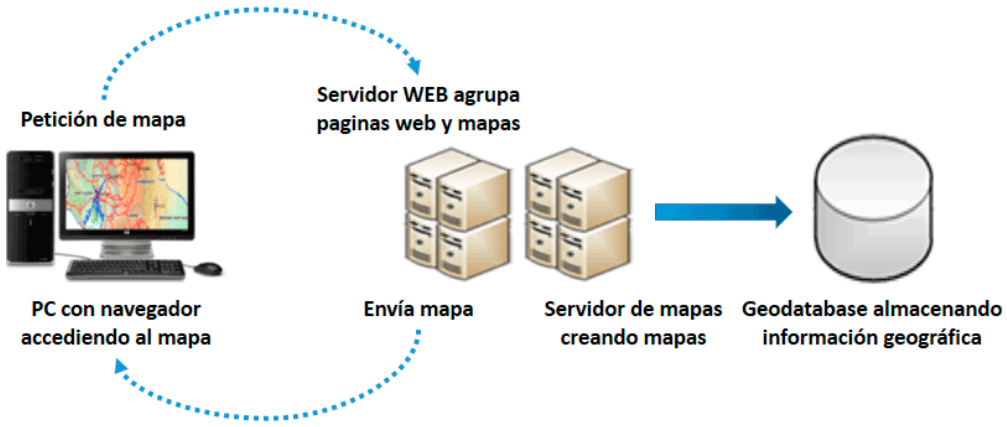


Fig. 5: Arquitectura del servidor de mapas (Muñoz, 2009).

Actualmente existen varios tipos de ficheros de configuración que permiten a los diferentes SIG guardar proyectos, exportar los estilos y símbolos de los mapas, y utilizarlos en otros trabajos o computadoras diferentes. Además mediante ellos es que estos sistemas pueden lograr una correcta interoperabilidad, ya

que cada SIG lo hace de forma distinta y en estos se refleja la manera en que se debe manejar la información. Estos archivos son consultados al inicio del proceso de representación cartográfica, y contienen una serie de parámetros comunes como la dirección del origen de datos, los estilos, la leyenda, la escala, la extensión a mostrar y que proyección utilizar. A continuación se hace mención de distintos tipos de aplicaciones que hacen uso de estos.

La herramienta gvSIG es un SIG para escritorios, que utiliza para guardar la configuración del entorno de trabajo ficheros con extensión .gvp. Este contiene además las configuraciones de los estilos y símbolos, o pueden ser exportados en archivos con extensión .style y .sym respectivamente. Todos estos presentan una estructura XML[5] (Kareaga Cantero, 2008).

Otro programa es el servidor de mapas GeoServer, que permite además de compartir editar datos geoespaciales. Este servidor cuenta con una herramienta que permite gestionar todas las configuraciones y estilos, que son guardadas en archivos con extensión .xml y .sld respectivamente y cuentan con estructura XML (Muñoz, 2009).

También existe el archivo .axl comúnmente llamado ArcXML, el cual consta como su nombre lo indica con estructura XML. Es utilizado por el servidor de mapas ArcIMS para la creación y configuración de las características del mapa. Este fichero es creado antes de enviar una solicitud a la aplicación del servidor y es utilizado por el servidor espacial para generar la imagen del mapa (East, y otros, 2002).

En esta investigación para poder dar solución al problema que sustenta la investigación y alcanzar el objetivo propuesto, es necesario abordar un poco más sobre los archivos de configuración de proyectos .qgs y MapFile. En este último se debe hacer mayor énfasis ya que es el resultado que se espera obtener con el complemento. Esta fundamentación se realiza sobre la base de que estos ficheros inciden directamente en la interoperabilidad entre GeoQ y MapServer.

[5] **Lenguajes de marcas extensibles (e*Xtensible Markup Language,* XML).**

1.3.1 Archivo de proyecto .qgs

Este archivo se guarda todos los estilos, símbolos, y demás configuraciones del proyecto, y pertenece al SIG para entornos de escritorio GeoQ. Además esta plataforma permite exportar los estilos y símbolos en ficheros independientes con extensión .qml y .xml respectivamente. En este caso también presentan internamente una estructura XML. El .qgs dentro de su estructura tiene por sección principal a qgis, la cual anida a todas las demás y define algunas características generales como el título y la extensión. Presenta una gran cantidad de elementos, por lo que en la Fig. 6 se muestran la estructura de los principales para un proyecto. A continuación se describen algunos de estos:

- **qgis**: Define el inicio de un proyecto de GeoQ y contiene elementos como title, projectlayers, legend, properties, entre otros.
- **properties**: En este definen las características generales del mapa y contiene elementos como Gui, PositionPrecision, SpatialRefSys, Paths, DefaultStyles, entre otros.
- **legend**: Se define la leyenda del mapa, que capas se van a visualizar y en qué orden se mostrarán (al contrario del .map, las que se encuentren en la parte inferior se representarán debajo de las que estén en la parte superior). Define el objeto legendlayer.
- **projectlayers**: Contiene al conjunto de capas definidas por el elemento maplayer que comprende la descripción de una capa con sus características y estilos.
- **title**: Define el nombre del proyecto de GeoQ.
- **mapcanvas**: Define las unidades de medidas del mapa, la extensión y la proyección. En él se definen secciones como units, extent, projections, destinationsrs, entre otros.
- **symbols**: Contiene al conjunto de símbolos definidas por el elemento symbol y en él se encuentra una estructura que comprende a los objetos lowervalue, uppervalue, label, outlinecolor, fillcolor, entre otros. Esta distribución depende del tipo de símbolo utilizado que puede ser singleSymbol, categorizedSymbol, RuleRenderer y graduatedSymbol. Estos a su vez permiten una serie de estilos de tipo SimpleFill, PointPatternFill, SimpleMarker, PointPatternFill, MarkerLine, SimpleLine, SVGFill, LinePatternFill, GradientFill y CentroidFill.

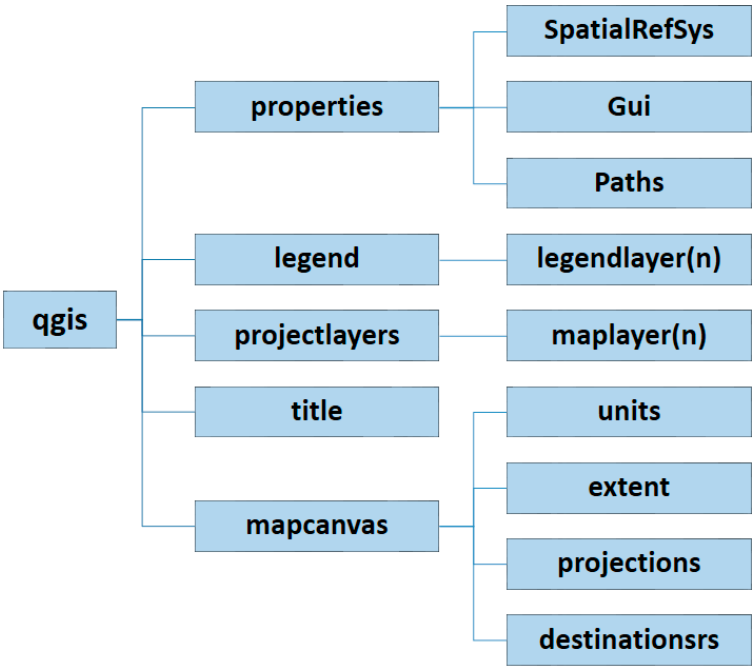


Fig. 6: Principales elementos de un archivo de proyecto de GeoQ.

1.3.2 MapFile

El MapFile es un componente importante de MapServer, es un archivo con extensión .map en formato texto, que contiene todas las definiciones y configuraciones iniciales necesarias para la ejecución del servidor. Define las relaciones entre objetos, señala donde se encuentran los datos y define cómo y cuáles son los objetos que se presentaran. Este archivo es consultado por el servidor en cada interacción del usuario con este. Los .map, no contienen el mapa concretamente, sino sus datos geo-referenciales, es decir, este relaciona el fichero gráfico del mapa con las coordenadas reales. (Manso Callejo, y otros, 2009).

Se puede simplificar diciendo que, el MapFile define los datos a ser usados, muestra y consulta parámetros. También contiene información sobre los estilos y símbolo que contendrá el mapa a representar. En general define cómo los datos serán presentados al usuario. La Fig. 7 muestra la estructura del servidor y su interacción con este.

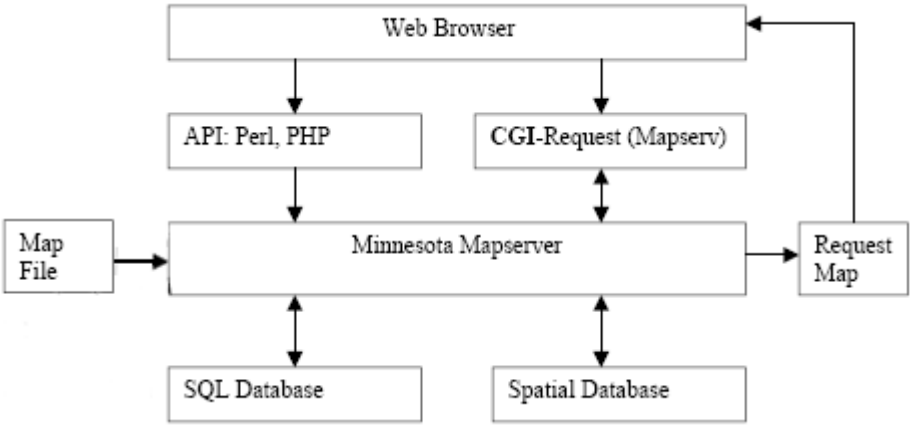


Fig. 7: Estructura de MapServer (Albornoz Venegas, 2006).

Dentro de su estructura, la sección principal es el objeto MAP, la cual anida a todas las demás y define las características generales del mapa como el nombre, las unidades de medidas del mapa, el tamaño, entre otras. Cada objeto se inicia con el nombre que lo define y termina con la palabra END. En este se definirán una serie de parámetros, algunos son de obligatoria inclusión, mientras que otros son opcionales o tienen un valor asignado por defecto. La Fig. 8 muestra todas las secciones que se definen en un MapFile, mientras que las principales son (Manso Callejo, y otros, 2009):

- PROJECTION: Se encarga de definir la proyección de los mapas a representar. Es necesario especificar dos objetos PROJECTION uno en el objeto MAP para la imagen de salida y otro para cada capa en el objeto LAYER.
- WEB: Define como operará la interfaz web anidando al objeto METADATA.
- METADATA: Deben definirse dos objetos METADATA, uno en el objeto MAP donde contendrá los datos de servicio general y otro en cada objeto LAYER con datos específicos de cada capa de información.
- LAYER: Se debe definir un objeto LAYER por cada capa de información que presente el servicio.
- CLASS: Define la apariencia y las características del etiquetado, cada capa debe tener al menos una clase.
- LABEL Permite definir una etiqueta, con la cual es posible colocar la toponimia u otro tipo de anotación en el mapa, a partir de datos alfanuméricos.
- LEGEND: Permite generar la simbología de forma automática.

- SCALEBAR: Define como se construirá la escala gráfica.
- SYMBOL: Las definiciones de símbolo se pueden incluir dentro de la MapFile principal o, más comúnmente, en un archivo separado. De esta manera se designa con la palabra clave SYMBOLSET, como parte del objeto MAP. Esta configuración es recomendada e ideal para reutilizar a través de múltiples aplicaciones MapServer.
- STYLE: Múltiples estilos se pueden aplicar dentro de un objeto CLASS o LABEL y tiene parámetros para la simbolización. Está destinado a la lógica separada de miradas y su objetivo final es que sea reutilizable como los símbolos, aunque todavía no es soportado.

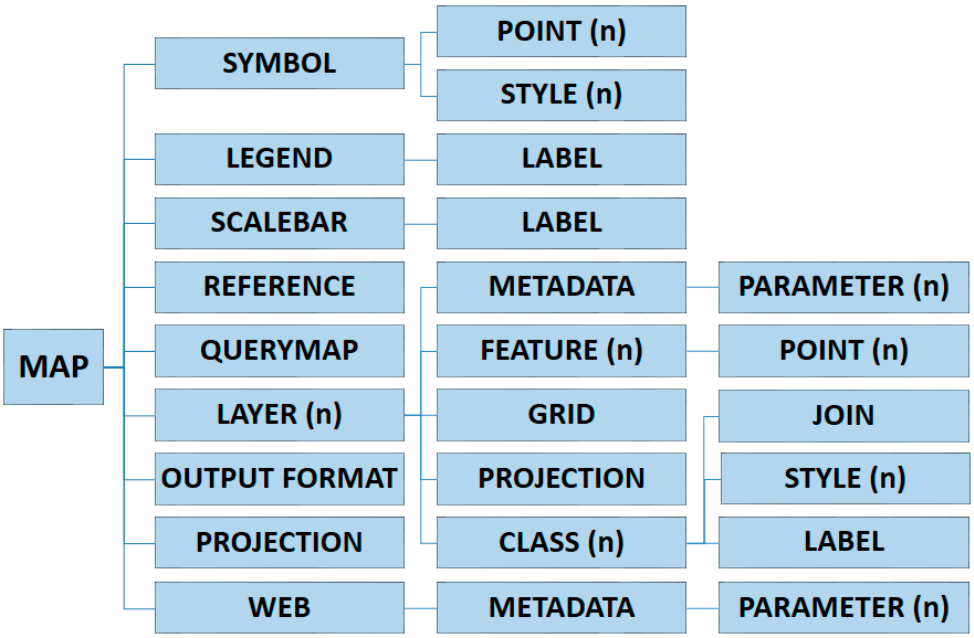


Fig. 8: Estructura del MapFile (Albornoz Venegas, 2006).

Principales propiedades contenidas en los objetos LAYER, CLASS y STYLE

Se decidió para brindar mayor comprensión de los archivos MapFile explicar algunas de las propiedades de los objetos que más importancia tienen para lograr resolver el problema que sustenta la investigación. Estas secciones son las que intervienen directamente en la representación de los estilos y que permitirán lograr la correcta interoperabilidad de MapServer con cualquier otro sistema. Además son estas configuraciones las que el usuario finalmente visualiza. A continuación se muestran sus principales características.

Tabla 1: Principales propiedades del objeto LAYER

Propiedad	Descripción
NAME	[string] Nombre corto para la capa. Este nombre es el vínculo entre el archivo map y la interfaz web, deben ser idénticos.
GROUP	[name] Nombre de un grupo o conjunto de capas.
TYPE	[point | line | polygon | circle | annotation | raster | query] Especifica cómo los datos podrían ser dibujados. Debe coincidir con el tipo de archivo Shapefile. Por ejemplo, un archivo Shapefile de polígonos, podrá ser dibujado como una capa de puntos, pero una Shapefile de puntos no podrá ser dibujada como polígono.
STATUS	[on | off | default] Configura el estado actual de la capa.
DATA	[filename] | [sde parameters] | [postgis table/column] | [oracle table/column] Nombre completo del archivo de datos espaciales a ser procesado. Si se trata de archivos Shapefile, no es necesario incluir la extensión.
DUMP	[true | false] Permite que MapServer genere la descarga en formato GML. Por defecto es false.
CONNECTION	[string] Cadena de conexión a bases de datos para acceder a datos remotos. Puede ser una conexión SDE, PostGIS u Oracle.
CONNECTIONTYPE	[local | sde | ogr | postgis | oraclespatial | wms] Tipo de conexión. Por defecto es local. Este parámetro debe incorporarse en el caso que se desee incluirse una capa remota.
CLASS	Señal de comienzo del objeto CLASS
CLASSITEM	[atributte] Nombre del ítem en tabla de atributos a usar como filtro para aplicar el objeto CLASS.
LABELITEM	[atributte] Nombre del ítem en tabla de atributos a usar como anotación.
HEADER	Nombre del archivo plantilla, para ser usado como encabezado de la plantilla de respuesta a consultas.

TEMPLATE	Nombre del archivo plantilla a utilizar, en la que se representarán los resultados de peticiones. Página web visible por el usuario.
FOOTER	Nombre del archivo plantilla, para ser usado como cierre de la plantilla de respuesta a consultas.
METADATA	**Inicio del objeto METADATA**
MINSCALE	Escala mínima para la cual la interfaz es válida. Cuando un usuario peticiona un mapa a escala más pequeña, MapServer retorna el mapa a esta escala.
MAXSCALE	Escala máxima para la cual la interfaz es válida. Cuando un usuario peticiona un mapa a escala más grande, MapServer retorna el mapa a esta escala.
PROJECTION	**Comienzo del Objeto PROJECTION de la capa de información**
TRANSPARENCY	[integer] Establece un nivel de transparencia para la capa. El valor es un porcentaje de 0 a 100 donde 100 es opaco y 0 es totalmente transparente.
TOLERANCE	[integer] Sensibilidad para las consultas basadas en puntos.
TILEINDEX	Archivo Shapefile que contiene los rectángulos envolventes de cada una de las piezas que forman el mosaico.

Tabla 2: Principales propiedades del objeto CLASS

Propiedad	Descripción
BACKGROUNDCOLOR	[R] [G] [B] Color para usar por los símbolos no transparentes.
COLOR	[R] [G] [B] Color a usar para dibujar las entidades.
EXPRESION	[string] Soporta expresiones de comparación, expresiones regulares y expresiones lógicas simples, para definir las clases. Si no se define ninguna expresión, se considerará todas las entidades dentro de la misma clase.
LABEL	Señal de comienzo del objeto LABEL.
OUTLINECOLOR	[R] [G] [B] Color a usar para la línea externa de polígonos. No es soportado por líneas.
NAME	[string]

	Nombre a ser utilizado en la generación de leyenda para esta clase. Si no se incluye ningún nombre, no aparecerá esta clase en la leyenda.

Tabla 3: Principales propiedades del objeto STYLE

Propiedad	Descripción
ANGLE	Angulo en grados, para girar el símbolo (en sentido anti horario).
COLOR	[R] [G] [B] Color a usar para dibujar las entidades.
GAP	[double] GAP especifica la distancia entre símbolos (centro a centro). Para polígonos rellenos, especifica la distancia entre símbolos tanto en la dirección X como la Y. Para las líneas, la distancia entre símbolos sobre esta.
MAXSCALEDENOM	Escala mínima a la que el estilo se dibuja. Se da como el denominador de la fracción de la escala real, por ejemplo para un mapa a una escala de 1:24.000 uso 24000.
MINSCALEDENOM	Escala maxima a la que el estilo se dibuja. Se da como el denominador de la fracción de la escala real, por ejemplo para un mapa a una escala de 1:24.000 uso 24000.
OUTLINECOLOR	[R] [G] [B] Color a usar para la línea externa de polígonos. No es soportado por líneas.
GEOMTRANSFORM	[bbox | end | labelpnt | labelpoly | start | vértices | <expression>] Se utiliza para indicar que la función actual se transforma antes de que se aplica el estilo real.
SYMBOL	[integer | string | filename | url] Símbolo a usar para representar las características. integer es el índice del símbolo en el conjunto de símbolos, empezando en 1. string es el nombre del símbolo (como se define mediante SYMBOL parámetro NAME). filename especifica la ruta a un archivo que contiene un símbolo. url especifica la dirección de un archivo que contiene un símbolo de mapa de pixels. Por ejemplo, una imagen PNG.

Renderizado múltiple y superposición

Para mostrar una carretera con una línea de borde negro, dos carriles de color amarillo y un carril central de color rojo. Esto requiere la combinación de varias firmas. Efectos cartográficos complejos pueden ser alcanzados por la presentación de los mismos datos de vectores con diferentes símbolos, tamaños y

colores. Esto se puede hacer creando varias capas con el objeto LAYER, sin embargo, afecta el rendimiento de la aplicación, como todo proceso de representación de la misma geometría tomará más tiempo de procesador. La solución más óptima es el uso varios STYLE para crear símbolos complejos mediante la superposición (McKenna, y otros, 2009). La Fig. 9 muestra la estructura de símbolos cartográficos que se pueden configurar en el MapFile y representarlos en MapServer.

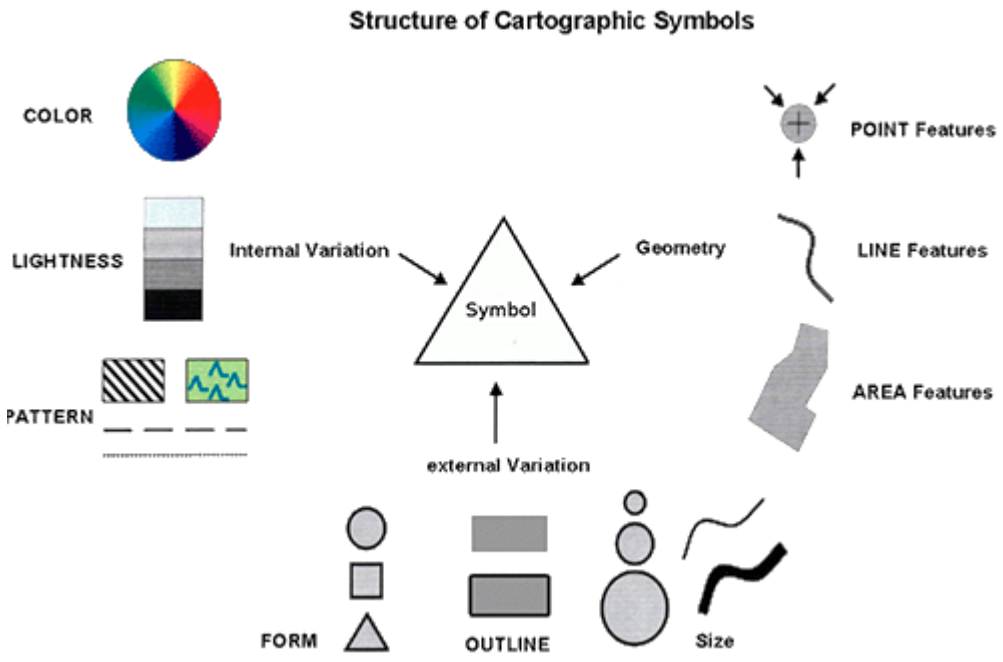


Fig. 9: Estructura de símbolos cartográficos (McKenna, y otros, 2009).

1.4 Análisis de soluciones existentes

Después de llevar a cabo una búsqueda por Internet, se pudo comprobar que las aplicaciones que permitan homologar sus vistas con MapServer, desarrolladas en C++ o lenguajes para entornos de escritorios, son realmente escasas. Esto se debe, a que generalmente la gestión de archivos MapFile, es necesaria para SIG que quieren publicar información cartográfica con este servidor en la WEB, por lo que se basan en tecnologías de este tipo. A continuación se describen varias herramientas que generan estos archivos:

MapServer Export

Es un plugin escrito en Python para el SIG Quantum GIS, actualmente QGIS. Para poder utilizarlo, debe tener Python en su sistema y QGIS debe haber sido compilado. Esta herramienta opera en un archivo de salvado de proyecto de QGIS (.qgs), ya sea en el contenido actual de la vista del mapa y la leyenda o en otro proyecto que ya haya sido guardado, en este último caso, se debe especificar la dirección donde se encuentra el archivo de proyecto QGIS. Sin embargo, es posible terminar con un archivo de mapa no funcional, dependiendo del interés de su uso. Aunque MapServer Export es bueno en la creación de un MapFile desde un archivo de proyecto, puede requerir algunos ajustes en los estilos y símbolos para obtener los resultados que se desea. Además presenta una interfaz muy simple e intuitiva, que facilita la exportación al usuario (Sherman, y otros, 2009).

Este plugin no resuelve el problema que sustenta la investigación, debido a que no es compatible con las versiones superiores a la 1.7 de QGIS y por consiguiente tampoco con GeoQ. Esto último se debe a que no tiene soporte para manejar ni los estilos, ni la nueva forma de renderizado de símbolos. Además, con este complemento se puede obtener .map no funcionales. A pesar de que no es empleado en el desarrollo de la solución propuesta, se utilizaron algunas de sus características como: iniciar el complemento y cargar automaticamente el proyecto actual en el que está trabajando.

Gix Export Tool

Gix Export es una extensión implementada sobre un lenguaje de programación propio para el SIG ArcView (ahora se conoce como ArcGIS) llamado Avenue. Esta extensión es de código abierto y ayuda a los desarrolladores, a migrar proyectos ESRI[6] ArcView, a las alternativas libres para SIG, como MapServer, JUMP, ArcExplorer, QGIS y Thuban. Esta extensión, permite exportar una vista de un archivo shapefile, al formato ".map" (Gix Project, 2004). Entre sus principales características se tienen:

- Creación automática de archivos .map, .fnt (fuentes) y .sym (simbolos) para MapServer.
- Creación automática de archivos .jpg para referencias en el mapa de MapServer.
- Exportación de temas como SQL listos para subir a las bases de datos de PostgreSQL con capacidades espaciales (utilizando PostGIS).

[6] **ESRI**. *Environmental Systems Research Institute*, Instituto de Investigación de sistemas ambiental.

Este complemento no será utilizado debido a que solamente se encuentra disponible para sistemas operativos Windows. Además, solamente permite exportar vistas de archivos Shapefile, obviando la posibilidad de que los datos cartográficos se encuentren en una base de datos como Postgres. Aunque tampoco será empleada en la solución, se extrajo la característica de creación automática de los símbolos para ser implementada en los que se utilizan en GeoQ.

LiberMaps

LiberMaps es un catálogo de mapas, que permite mantener el control y gestionar cada archivo MapFile que se encuentre en su catálogo, desarrollado en ExtJS y PHP. Esta mejora el proceso de edición de ficheros de mapas utilizados en los SIG que utilizan como servidor de mapas a MapServer. Cuenta con un conjunto de funcionalidades que le permite editar un .map completamente. Fue desarrollado como un entorno WEB por lo que no se encuentra disponible para usuarios sin conexión a una red informática (Castillo Reyes, y otros, 2013). Entre sus principales funciones se encuentran:

- Gestionar MapFile: permite crear y eliminar un *MapFile*.
- Generar MapFile: permite la creación de un *MapFile* a partir de datos predeterminados .
- Exportar MapFile: permite al usuario seleccionar un mapa de los existentes y exportar a un fichero con extensión *.map* la información asociada al mapa almacenada en la base de datos.

Esta herramienta aunque sirvió de referencia para determinar las estructuras y configuraciones que tienen los archivos MapFile, no se puede emplear como solución del problema planteado debido a que está pensada para entornos WEB. También, de esta aplicación se extrajo la característica de visualizar la imagen del mapa que se genera con el MapFile en el que se está trabajando, y de esta manera brindarle al usuario la posibilidad de saber si lo que está creando es lo que desea o es correcto.

1.5 Conclusiones Parciales

Después de realizar un estudio del arte, sobre las diferentes herramientas que existen para generar un archivo MapFile, se puede concluir que ningunas de las estudiadas resuelve el problema que sustenta la investigación. Aunque estas aplicaciones son de código libre, lo que permite estudiar la forma y la lógica que deben tener los programas de este tipo. Por lo tanto, se facilita la confección de un generador de

archivos MapFile propio para GeoQ, utilizando características que estos presentan como guía en el desarrollo del complemento.

El estudio realizado en este capítulo, constituye una orientación necesaria, para conocer cuáles son las rutas a tomar durante el desarrollo del producto y así obtener mejor rendimiento, logrando un trabajo de mayor calidad. Después de especificar cuáles eran, se puede afirmar que los cimientos para iniciar la construcción de la futura aplicación, están correctamente establecidos y se puede comenzar con el análisis y diseño del complemento GAMF-GeoQ.

Capítulo 2: Análisis y diseño del complemento GAMF-GeoQ

El software que se propone en esta investigación, es una herramienta que procura tener aplicación dentro del contexto de un problema real. Por lo tanto, debe seguir un proceso de análisis y diseño que suministre las bases bajo las cuales se va a implementar. Es por eso que en este capítulo, se describen las herramientas, tecnologías y metodología de desarrollo de software a utilizar para el generador de archivos MapFile. También se definen los principales conceptos del dominio, se realiza una especificación de requisitos, se define la arquitectura del sistema y los patrones de diseño a utilizar. Además, se realiza el levantamiento de requisitos funcionales y no funcionales, así como la descripción de los actores del sistema y la elaboración de los diagramas de casos de usos, para luego ser detallados textualmente.

2.1 Metodologías de desarrollo de software

En todo sistema de desarrollo de software, es de suma importancia seguir alguna especificación que permita a los desarrolladores, tener un método que haga que todas sus etapas, no sean solo coherentes, sino también las más consecuentes. Con tal objetivo, surgieron varias metodologías de desarrollo[7] que permiten coordinar el proceso de estos de una mejor manera, evitando tener que comenzar de cero si un cliente pide una modificación en el negocio a última hora. (Figueroa, y otros)

Comparación entre las metodologías ágiles y tradicionales

Decidir que metodología usar es de suma importancia, debido a que esta guía a los desarrolladores durante todo el proceso de construcción del software. Para ello, debemos conocer cuáles son las particularidades de estas, y decidirnos por la que mejor se adapte a las características del proyecto y del equipo de desarrollo. Para ayudar a tomar la decisión, los autores realizaron un cuadro comparativo, exponiendo las principales

[7] **Metodologías ágiles y tradicionales.** Entre los principales métodos ágiles se encuentran Extreme Programming (XP), SCRUM, ICONIX, Cristal Methods, AUP, entre otras. Entre los principales métodos tradicionales se encuentran Microsoft Solution Framework (MSF), MÉTRICA 3, Proceso Unificado de Desarrollo (Rational Unified Process, RUP) y Método de Desarrollo de Sistemas Dinámicos (Dinamic Systems Development Method, DSDM).

características de las dos ramas de metodologías existentes. La tabla número 4 muestra dicha comparación. (Letelier, y otros, 2006)

Tabla 4: Comparación de metodologías

Metodologías Ágiles	Metodologías Tradicionales
Preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Procesos menos controlados, con pocos principios.	Proceso muy controlado, numerosas normas.
Contrato flexible e incluso inexistente.	Contrato prefijado.
El cliente es parte del desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10) y trabajando en el mismo sitio.	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.	Más Artefactos.
Menor énfasis en la arquitectura de software.	La arquitectura de software es esencial.
Impuestas internamente (por el equipo de desarrollo).	Impuestas externamente.

Proceso Unificado Ágil (*Agile Unified Process*, AUP)

El AUP es un método que trata de establecer el contexto ágil, mediante el aumento incremental de desarrollo con las fases del proceso unificado. Mantiene todos los elementos del desarrollo ágil, al tiempo que define una arquitectura y requisitos desde el principio del ciclo de vida de entrega de software. Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones usando técnicas ágiles y conceptos que aún se mantienen válidos en el Proceso Unificado de Rational (*Rational Unified Process*, RUP) (Edeki, 2013). En la Fig. 10 se evidencian las fases y disciplinas con que propone AUP.

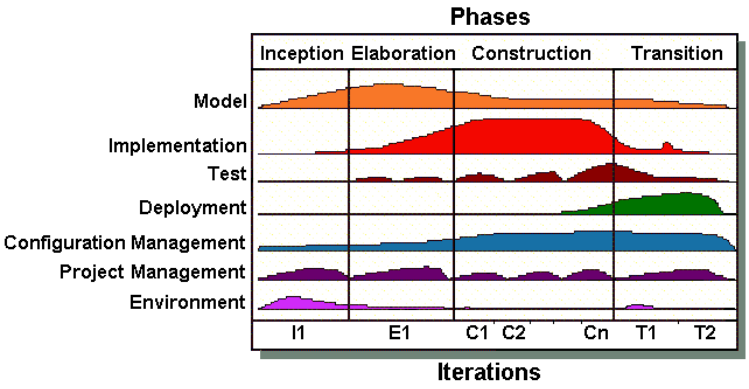


Fig. 10: Ciclo de Vida de AUP (Ambler, 2014).

AUP fue seleccionada para el desarrollo de esta investigación como resultado de la comparación anterior, ya que las metodologías ágiles se ajustan a las características del proyecto y al equipo de desarrollo. Ejemplo de ello es que se cuenta con solo 2 integrantes y poco tiempo de desarrollo (seis meses). Por otra parte, el complemento será utilizado por desarrolladores, por lo que no necesitan toda la documentación que genera una tradicional para entender cómo funciona. Esto evidencia que desarrolladores y clientes se ponen en comunicación directa y continua, integrando este último al equipo, lo que facilita la definición de prioridades y el esclarecimiento de dudas, elevando así la calidad del producto a desarrollar. Además AUP guarda relación con RUP, que es la utilizada por el proyecto SIG-Desktop para la implementación de GeoQ, permitiendo agregar la documentación generada al expediente de proyecto sin problemas de compatibilidad. También permite ver el crecimiento del software mediante las iteraciones y mantener el foco en las actividades de alto valor.

2.2 Herramientas y tecnologías a utilizar

Para llevar a cabo la realización de cualquier software, es necesario contar con herramientas de modelados, lenguajes de programación y tecnologías como base de desarrollo. La selección de estas puede ser una tarea complicada, debido a la cantidad de opciones que en la actualidad existen para lograr un mismo objetivo. A continuación, se describen las principales herramientas y tecnologías que serán utilizadas en el desarrollo del complemento, dejando claro cuáles fueron los puntos de inflexión que propiciaron la elección de estas.

UML V2.0 como lenguaje de modelado

Según (Schmuller, 2000) El lenguaje de modelado unificado (*Unified Modeling Language,* UML) *"Es un sistema de notación que se ha convertido en estándar en el mundo del desarrollo de sistemas... Está constituido por un conjunto de diagramas*[8]*, y proporciona un estándar que permite al analista de sistemas generar un anteproyecto de varias facetas que sean comprensibles para los clientes, desarrolladores y todos aquellos que está involucrados en el proceso de desarrollo. Es necesario contar con todos esos diagramas dado que cada uno se dirige a cada tipo de persona implicada en el sistema. Un modelo UML indica qué es lo que supuestamente hará el sistema, mas no cómo lo hará"*.

Este lenguaje es utilizado en combinación con la metodología de desarrollo, para graficar los diferentes artefactos que esta genera y mostrar de manera sencilla el dominio del problema. Permitiendo simplificar la complejidad sin perder información, para que usuarios, líderes y desarrolladores puedan comprender claramente las características de la aplicación (Rearte, 2002).

Visual Paradigm V8.0 para UML

Para apoyar los diagramas que genera UML en el transcurso de este trabajo, se utiliza la herramienta de Ingeniería de Software Asistida por Computadoras (C*omputer Aided Software Engineering,* CASE) Visual Paradigm. Esta permite modelar todo tipo de diagramas UML, ya que proporciona instrumentos específicos para ello. Esto también permite la estandarización de la documentación, ya que se ajusta a la que esta establece. Además cuenta con abundantes tutoriales, demostraciones interactivas y proyectos (Norvell, 2010).

Tecnologías para el desarrollo

Las tecnologías de desarrollo que se adoptan para este trabajo, son las que utiliza el proyecto SIG-Desktop. Estas herramientas suponen una ventaja para los autores del trabajo, ya que al estar vinculadas a ellas en el trabajo del proyecto, las conocen y no se tendría que invertir tiempo para capacitarse en otras. También

[8] Los diagramas con los que consta UML son los casos de uso, clases, objetos, secuencia, colaboración, estado, actividades, componentes y despliegue o implementación.

se analizaron y tomaron en cuenta varios criterios, entre los que se encuentran las ventajas que proporcionan.

C++ como lenguaje de programación

C++ es un lenguaje de programación que deriva del **C**, por lo que soporta prácticamente cualquier código escrito en este último. También incorpora grandes adelantos, como la programación orientada a objetos (POO)[9], programación genérica, excepciones y sobrecarga de operadores. Cuenta con varias características, como distinguir entre mayúsculas y minúsculas, todas las sentencias y declaración de variables terminan en punto y coma, las palabras clave están siempre en minúsculas, entre otras. Cuenta con soporte multiplataforma (Linux, Windows, MacOS, entre otras) y es completamente libre (Stroustrup, 2013).

QT 4.7.1 como framework de desarrollo

QT es un marco de trabajo *(framework)* de desarrollo de aplicaciones gráficas multiplataforma usando como lenguaje nativo a C++. Cuenta con múltiples librerías y clases que están bien documentadas, lo que las hacen más fáciles de usar (Thelin, 2007).

Qt Creator V2.4.1 como entorno de desarrollo

Qt Creator, es un Entorno Integrado de Desarrollo (*Integrated Development Environment,* IDE) de código abierto para desarrollar aplicaciones en C++, basándose en el *framework* QT. Permite el desarrollo de aplicaciones en entornos Windows, Mac OS y Linux. Proporciona herramientas para el diseño y desarrollo de aplicaciones complejas. Además, presenta características como el resaltado de sintaxis y autocompletado de código, la comprobación de código estático y toques de estilo a medida que escribe y plegado de código (Summerfield, y otros, 2008).

[9] **POO (Programación Orientada a Objetos).** Es una nueva filosofía de programación que se basa en la utilización de objetos. Sus mecanismos básicos son: objetos, mensajes, métodos y clases.

2.3 Modelo del Dominio

Un MD no es más que un grupo de clases relacionadas, que siguen las normas del UML para representar los objetos del mundo real en un dominio de interés. El MD se centra en una parte del negocio, la vinculada con el ámbito del proyecto; tiene por objetivo primordial, ayudar a entender los conceptos que utilizan los usuarios y los conceptos con que trabajan y que serán con los que deberá trabajar la aplicación. De manera general, se puede decir que un MD es una representación de las clases conceptuales del mundo real, no de componentes de software. La Fig. 11 muestra el diagrama de modelo de dominio referente al problema que se desea dar solución.

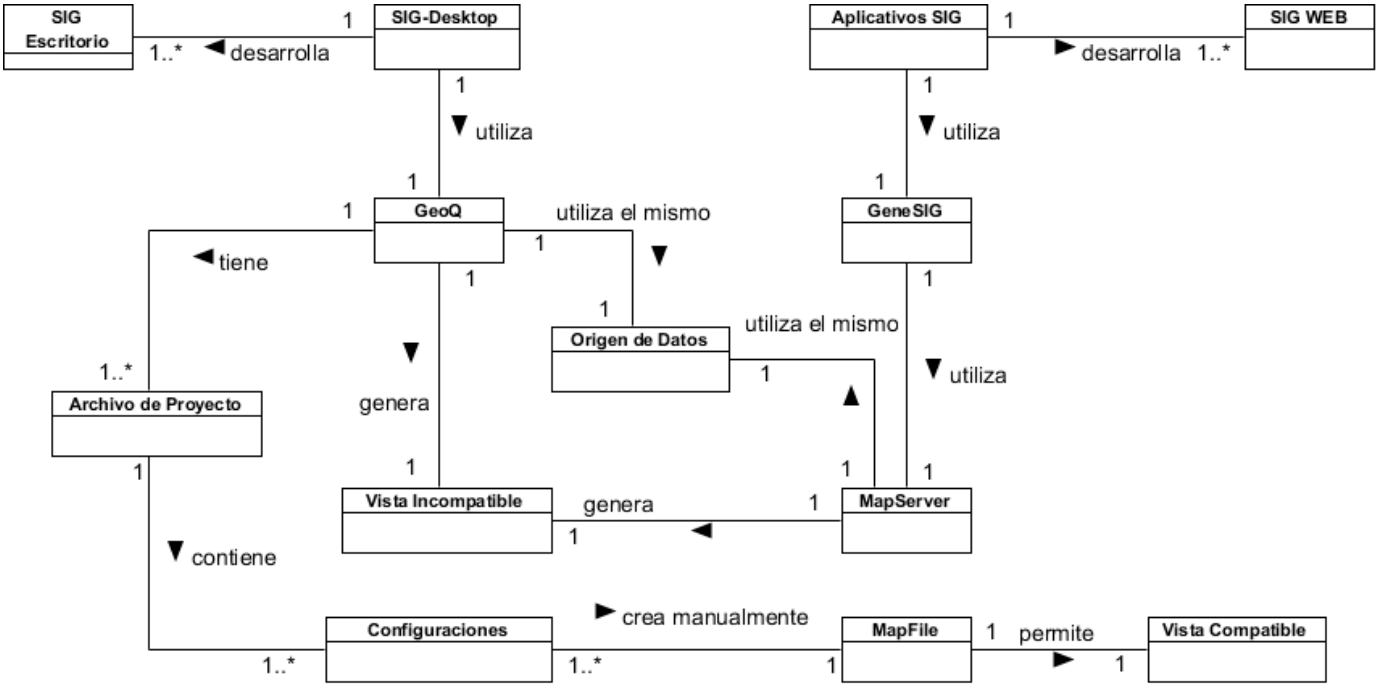


Fig. 11: Diagrama del Modelo del Dominio.

Descripción del modelo del dominio

En la figura 15 se explica que los proyectos SIG-Desktop utilizando la plataforma GeoQ, y Aplicativos SIG empleando a GeneSIG junto a MapServer, desarrollan SIG para entornos de escritorios y Web respectivamente. Actualmente, en estos proyectos la información que se representa con el servidor

MapServer y GeoQ genera vistas incompatibles, aun desde un mismo origen de datos. Es por esta razón que a partir de las configuraciones que contiene un archivo de proyecto, se tiene que crear manualmente un MapFile que permita obtener una vista compatible.

Definiciones y abreviaturas del modelo del modelo del dominio

- **SIG Escritorio:** SIG para entornos de escritorios.
- **SIG WEB:** SIG para entornos en la Web (*World Wide Web*).
- **SIG-Desktop:** Es un proyecto que se dedica a la realización de SIG para escritorios.
- **Aplicativos SIG:** Es un proyecto que se dedica a la realización de SIG para la Web.
- **GeoQ:** Es un SIG de escritorio que utiliza el Proyecto SIG-Desktop como base para el desarrollo de otros SIG de escritorio.
- **GeneSIG:** Es un SIG para la Web utilizado por el proyecto Aplicativos SIG como base para el desarrollo de otros SIG para la Web.
- **Origen de Datos:** Información cartográfica del mapa que se quiere representar.
- **MapFile:** Contiene todas las configuraciones y estilos que se extraen del archivo de proyecto.
- **MapServer:** Servidor de mapas de código abierto utilizado por la plataforma GeneSIG.
- **Configuraciones:** Son las configuraciones que deben ser portadas de GeoQ a MapServer.
- **Archivo de Proyecto:** Representa al archivo de proyecto de GeoQ con extensión .qgs.
- **Vista Incompatible:** Los mapas que se generan no son iguales, los estilos son distintos.
- **Vista Compatible:** El mapa que se puede generar en MapServer con el MapFile creado tendrá una vista compatible con la de GeoQ.

2.4 Especificación de los requisitos

La Especificación de Requisitos Software (ERS), es una descripción del sistema que va a desarrollar. Deben recogerse tanto las necesidades de clientes y usuarios, como los requisitos que debe cumplir el software a desarrollar para satisfacer dichas necesidades. Estas necesidades se agrupan en los Requisitos Funcionales (RF) que es una definición de una función del sistema, y los Requisitos no Funcionales (RNF) que son aquellas cualidades o propiedades que la aplicación debe tener. Son estas propiedades las que hacen que el programa sea confiable, eficiente, ágil y atractivo a los ojos del usuario. El lenguaje usado en su descripción debe ser informal para que sea fácil de entender por cualquier persona.

Técnicas de extracción de requisitos

Para la extracción de requisitos, existen varias técnicas que posibilitan que el proceso de descubrir, analizar y verificar servicios y restricciones, se realice de manera eficiente. Ejemplos de estas son la entrevista, cuestionarios, revisión de documentos, análisis de productos de competencia y tormentas de ideas. Este último es el utilizado en este trabajo para llevar a cabo le ERS.

La tormenta de ideas o brainstorming es una técnica basada en la exposición de manera informal y libre de todas las ideas en torno a un tema o problema planteado que ayuda a estimular la creatividad. El fundamento del método es que muchas mueren por la crítica destructiva a que se ven sometidas, antes de que maduren o se perfeccionen. Además esta práctica prohíbe la crítica, la autocrítica y la autocensura. Es muy importante para lograr estimular libremente la imaginación, crear un ambiente que beneficie la comunicación y la motivación de los miembros del grupo, para de esa forma conseguir una libre manifestación (Gallagher, 2009).

Requisitos funcionales

RF.1 Leer archivo de proyecto de GeoQ.

Esta funcionalidad permite que el usuario pueda leer un proyecto de GeoQ con extensión .qgs, previamente guardado o usando el proyecto actual sobre el que se está trabajando. El usuario debe especificar la dirección donde se encuentra el archivo .qgs. El sistema mostrará el archivo de proyecto GeoQ en un cuadro editor de texto.

RF.2 Análisis Sintáctico del Proyecto.

El sistema debe realizar un análisis sintáctico (*parsing*) del archivo .qgs que se quiere exportar a MapServer. Posteriormente mostrará un árbol con todas las propiedades del archivo .qgs.

RF.3 Generar archivo MapFile.

Esta funcionalidad permite que el usuario pueda generar archivos de mapas, a partir de un proyecto de GeoQ y sus propiedades. El sistema mostrará el archivo de mapas (.map) en un cuadro de edición de texto.

RF.4 Insertar propiedad al archivo MapFile.

Esta funcionalidad permite al usuario insertar propiedades a un .map que ha sido cargado o generado previamente. El sistema mostrará el MapFile modificado.

RF.5 Guardar archivo MapFile.

Esta funcionalidad permite al usuario guardar el MapFile generado o modificado hacia una ruta especificada por el mismo. El sistema guardara el archivo .map y mostrara un mensaje indicando que se exporto correctamente el MapFile.

- Mensaje (Formato: Alfabético).

RF.6 Visualizar imagen del mapa.

Esta funcionalidad permite al usuario comprobar que el archivo MapFile funciona correctamente, mostrando en una ventana la imagen del mapa que se genera con MapServer.

RF.7 Guardar símbolos.

El sistema debe ser capaz de guardar los símbolos en un archivo independiente del MapFile, en caso de que el usuario lo requiera. Se debe especificar la ruta completa donde se guardará el archivo.

RF.8 Guardar capas.

El sistema debe ser capaz de guardar las capas independientes del MapFile, en caso de que el usuario lo requiera hacia una ruta especificada por este.

Requisitos no funcionales

Usabilidad: El plugin contará con una ayuda que servirá de apoyo en caso de no entender que pasos se deben ejecutar. También contará con botones en los elementos donde se deba especificar direcciones, para poder así explorar mejor los objetos y su uso sea más placentero para el usuario.

Interfaces de Usuario: El complemento presenta un diseño sencillo brindando al usuario una apariencia agradable e intuitiva, utilizando etiquetas que sugieren la acción que realiza.

Interfaces de Hardware: Para el complemento corra en las PCs, se requieren al menos 128 MB de memoria RAM, 40 GB de disco duro y que el procesador tenga una velocidad de 512 MHz como mínimo. Esto se debe a que corre sobre GeoQ, por lo que debe adoptar los requerimientos de hardware de este.

Interfaces de Software: El generador de archivos MapFile para GeoQ es un sistema multiplataforma que puede ejecutarse en los siguientes sistemas operativos GNU/Linux, Mac y Windows.

Requisitos de Licencia: GAMF-GeoQ se publica bajo La Licencia Pública General de GNU (GNU GPL). Desarrollar este producto bajo esta licencia, quiere decir que se puede inspeccionar y modificar el código fuente y garantiza que nuestros usuarios, siempre tendrán acceso a un programa gratuito.

Requisitos Legales, Derechos de Autor y otros: El sistema debe ajustarse y regirse por la ley, decretos leyes, decretos, resoluciones y manuales (órdenes) establecidos, que norman los procesos que serán automatizados. Se adopta herramientas de desarrollo libres, y en el caso de que sea necesario hacer uso de al menos una propietaria tendrá avalada su licencia.

2.5 Modelo de casos de uso del sistema

El modelo de Casos de Uso (CU) ayuda al cliente, a los usuarios y a los desarrolladores a llegar a un acuerdo sobre cómo utilizar el sistema. La mayoría de estos tienen muchos tipos de usuarios, cada uno se representa como un actor que utiliza el sistema al interactuar con ellos. Además son fragmentos de funcionalidad que el sistema ofrece para aportar un resultado de valor para sus actores (Jacobson, 2000). La Fig. 12 muestra el diagrama de caso de uso para el complemento GAMF-GeoQ.

Descripción de los actores del sistema

Tabla 5: Descripción del actor

Actor	Descripción

	Persona que va a exportar un proyecto de GeoQ con extensión .qgs a un archivo de mapa con extensión .map para ser utilizada por MapServer.

Diagrama de casos de uso del sistema

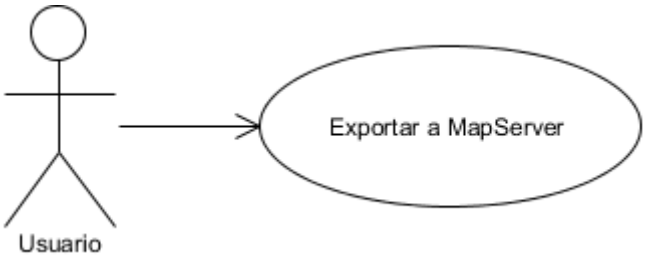


Fig. 12: Diagrama de Casos de Uso.

Descripción textual del caso de uso del sistema

Tabla 6: Descripción del caso de uso "Exportar a MapServer"

Caso de Uso	Exportar a MapServer
Actores	Usuario
Resumen	El caso de uso inicia cuando el usuario selecciona la opción "Exportar a MapServer". El sistema muestra una interfaz donde se realizan todas las configuraciones necesarias para exportar un proyecto de GeoQ a un archivo .map.
Precondiciones	Debe existir un archivo .qgs
Referencia	RF.1, RF.2, RF.3, RF.4, RF.5, RF.6, RF.7, RF.8, RF.9
Prioridad	Crítico.
Postcondiciones	Se exporta el archivo .qgs a un .map
Flujo Normal de Eventos	

Acción del Actor	Respuesta del Sistema
1) El caso de uso inicia cuando el usuario selecciona la opción “Exportar a MapServer” dentro del menú “Complementos”.	2) El sistema guarda automáticamente el proyecto si se le ha realizado algún cambio, en caso de que nunca haya sido guardado ver Flujo Alterno (FA): “Guardar proyecto actual”.
	3) El sistema muestra la interfaz “Exportar a MapServer”.
	4) El sistema realiza el análisis sintáctico del proyecto de GeoQ, muestra su estructura xml en un cuadro editor de texto, muestra un árbol con las propiedades del .qgs y habilita las propiedades para la generación del archivo .map. En caso de error en el análisis sintáctico ver FA: “No se carga el archivo .qgs”.
5) El usuario realiza las configuraciones iniciales del archivo .map que se va a generar. Sino ver FA: “Mantener propiedades por defecto del .map”.	
6) El usuario indica la ruta donde se guardará un archivo con los símbolos independientes del MapFile. Sino ver FA: “No se separan los símbolos”.	
7) El usuario indica la ruta donde se guardarán las capas independientes del MapFile. Sino ver FA: “No se separan las capas”.	
8) El usuario ejecuta la opción de “Generar”.	9) El sistema genera el .map y lo muestra en un cuadro de edición de texto.
10) El usuario gestiona las propiedades del archivo .map. Sino ver FA: “No se gestionan las propiedades”.	11) El sistema modifica las propiedades del archivo .map que fueron gestionadas.

12) El usuario ejecuta la opción “Visualizar imagen del mapa”. Sino ver FA: “No se visualiza”.	13) El sistema muestra en una ventana la imagen del mapa que se genera con MapServer. En caso de error ver FA: “Error al visualizar”.
14) El usuario indica la ruta donde se guardará el MapFile generado.	
15) El usuario ejecuta la opción “Guardar”.	16) El sistema guarda el archivo MapFile y muestra un mensaje de confirmación “Archivo exportado correctamente”. Sino ver FA: “Sobrescribir archivo MapFile” o FA: “Error al guardar MapFile”. Finaliza el caso de uso.
Flujo Alterno de Eventos	
2) Evento: Guardar proyecto actual	
Acción del Actor	**Respuesta del Sistema**
	2.1) El sistema muestra una ventana donde se especifica la dirección en la que será guardado.
2.2) El usuario indica la dirección para guardar.	2.3) El sistema indica a GeoQ que guarde el archivo en la dirección que dio el usuario. Ir a la acción 3.
4) Evento: No se carga el archivo .qgs	
	4.1) El sistema muestra un mensaje “Error de lectura o al cargar el archivo de proyecto”. Finaliza el caso de uso.
5) Evento: Mantener propiedades por defecto del .map	
5.1) El usuario mantiene las propiedades por defecto del .map que se va a generar. Ir a la acción 6.	

6) Evento: No se separan los símbolos	
6.1) El usuario no indica la ruta donde se guardarán los símbolos por lo tanto se incluyen en el MapFile. Ir a la acción 7.	
7) Evento: No se separan los capas	
7.1) El usuario no indica la ruta donde se guardarán los capas por lo tanto se incluyen en el MapFile. Ir a la acción 8.	
10) Evento: No se gestionan las propiedades	
10.1) El usuario no realiza ningún cambio en el MapFile generado. Ir a la acción 12.	
12) Evento: No se visualiza	
12.1) El usuario no ejecuta la opción “Visualizar imagen del mapa”. Ir a la acción 14.	
13) Evento: Error al visualizar	
	13.1) El sistema no muestra la imagen del mapa y en su lugar muestra un mensaje “Existen errores en la estructura del MapFile”. Ir a la acción 14.
16) Evento: Sobrescribir archivo MapFile	
	16.1) Si ya existe un .map con el mismo nombre del que desea guardar el sistema muestra un mensaje “El MapFile… ya existe. ¿Desea sobrescribirlo?”.
16.2) El usuario selecciona la opción “Si” o “No”.	16.3) En caso de que el usuario seleccione “Si” se sobrescribe el archivo .map y finaliza el caso de uso. 16.4) En caso de que el usuario seleccione “No” finaliza el caso de uso.

16) Evento: Error al guardar MapFile	
	16.1) Si el archivo no se guardó correctamente se muestra un mensaje “Error al intentar crear o abrir el archivo”. Finaliza el caso de uso.

2.6 Descripción de la arquitectura

La arquitectura de software nace por la necesidad de manejar la complejidad de los sistemas de mediana o gran envergadura. En la medida que las aplicaciones crecen en dificultad, se hace necesario establecer medios para el manejo de estas. Actualmente es posible encontrar numerosas definiciones del término arquitectura de software, cada una con planteamientos diversos y comunes. Según (Pressman, 2005) *“es la estructura u organización de los componentes del programa (módulos), la manera en qué estos componentes interactúan, y la estructura de datos que utilizan los componentes”*.

La plataforma GeoQ cuenta con una arquitectura extensible de complementos o plugins. Esto permite que se añadan nuevas funciones a la aplicación. Estos mecanismos por lo general, están basados en el patrón Fachada, para proporcionar una interfaz de interacción con el sistema, mediante la cual se debe acceder a los métodos principales de una o varias clases bases. Esta integración se logra por medio de la clase QgisInterface provista por su núcleo. Gracias a estas facilidades, integrar al sistema el complemento de generación de archivos de mapas, es una tarea sencilla.

Para el desarrollo del generador de archivos MapFile, se seleccionaron del estilo arquitectónico llamada y retorno las arquitecturas Orientada a Objetos y la de N-Capas. En las orientadas a objetos los componentes se basan en los principios orientados a objetos (encapsulamiento, herencia y polimorfismo). Pero la principal característica de esta es que se puede modificar la implementación de un objeto sin afectar la interfaz.

Las arquitecturas en capas están organizadas jerárquicamente, cada capa proporciona servicios a la capa que está encima y sirve de cliente a la que está abajo (Varma, 2009). Como una variante del patrón N-Capas, se definió una en 2 capas, como se representa en la Fig. 13. La primera seria la capa de presentación que es la encargada de mostrar el sistema al usuario. Además comunica y captura la información en un mínimo proceso y se relaciona únicamente con la segunda denominada capa de negocio. En esta es donde

se ejecutan todas las acciones del usuario y se envían las respuestas tras el procedimiento. También es la que se encarga de todo el proceso de generación y modificación de archivos MapFile.

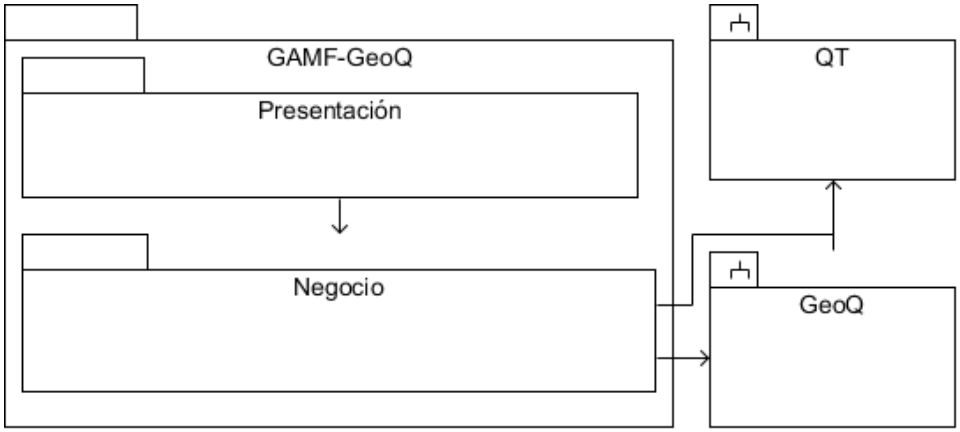


Fig. 13: Arquitectura del Complemento GAMF-GeoQ.

2.7 Modelo de Diseño

Diseñar, es crear una estructura que organiza la lógica en el sistema, de modo que un cambio en una parte, no siempre requerirá una alteración en otra. Un mal diseño es todo lo contrario, un cambio conceptual requiere transformaciones en muchas partes de la aplicación y con el tiempo, el costo se vuelve abrumador y no se podría agregar una nueva función sin afectar a otras. (Beck, 1999)

Diagrama de clases

Los diagramas de clases, muestran de manera estática, las distintas clases que componen un sistema de software y sus relaciones, ya sean asociaciones, generalizaciones o dependencias. En ellas se pueden apreciar los distintos atributos y métodos que la componen, sin embargo no especifican los métodos mediante los que se invocan entre ellas. En la Fig. 14, se muestra el diagrama de clases del producto que se va a desarrollar.

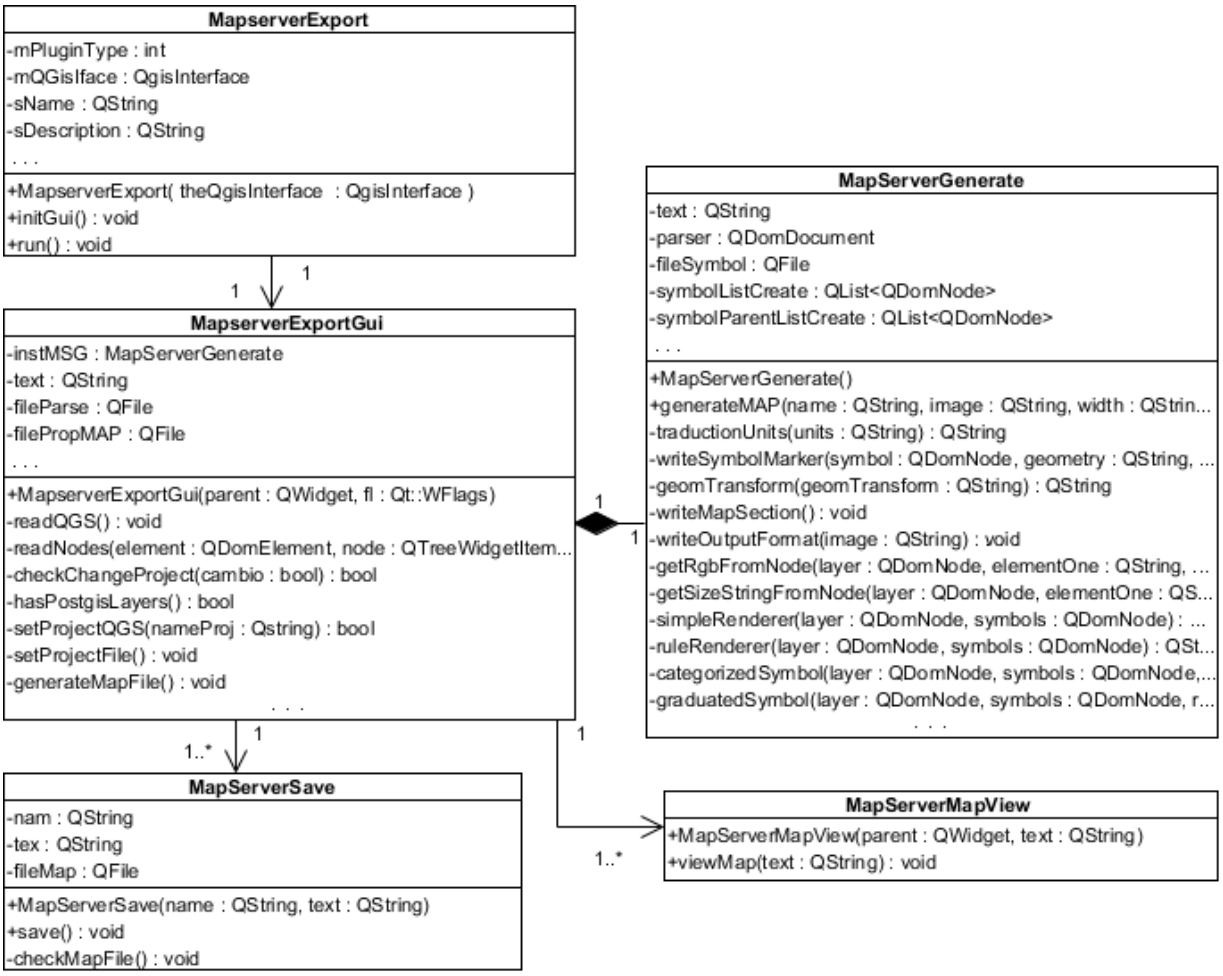


Fig. 14: Diagrama de clases del diseño del CU "Exportar a MapServer".

2.8 Patrones de Diseño

Los patrones de diseño ayudan a organizar las clases para guiar todos los procesos de creación de software, ya que brindan soluciones aprobadas a problemas de desarrollos similares. Según (Gamma, y otros, 1995) "*Es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular".* También permiten reutilizar soluciones, simplifica la comunicación entre diseñadores y facilita el aprendizaje al diseñador inexperto, además tienen nombres estándar.

Los patrones generales de software para asignación de responsabilidades (*General Responsibility Assignment Software Patterns*, GRASP), más que patrones propiamente dichos, son una serie de "buenas prácticas" de aplicación. Estos son recomendable en el diseño de programas que se utilizan para asignar responsabilidades. Para el desarrollo del generador de archivos MapFile se utilizaron los siguientes patrones de GRASP (Tabares, 2010):

Experto: Diseñado para asignar la responsabilidad de realizar determinada tarea, a la clase que conoce la información necesaria para ejecutarla. Esto facilita la comprensión del sistema, su mantenimiento y adaptación a los cambios con reutilización de componentes. Este patrón se evidencia en la clase MapServerView, que es la que conoce toda la información necesaria para ejecutar la visualización del mapa que se genera a partir del archivo MapFile.

Creador: Aporta un principio general para la creación de objetos, una de las actividades más frecuentes en programación. Plantea la necesidad de que una clase tenga la responsabilidad de crear una instancia de otra, siempre y cuando los utilice específicamente. El uso de este patrón se evidencia en la clase MapServerExportGui, la cual crea objetos de la clase MapServerGenerate, MapServerSave y MapServerView.

Controlador: Permite asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. Se utilizó para decidir qué clase es la responsable de recibir o manejar un evento del sistema, por ejemplo la clase MapserverExportGui, es la encargada de recibir los datos que son enviados desde la interfaz y enviarlos a las distintas clases según el método llamado.

Alta Cohesión: Es una medida que establece cuán vinculadas y adecuadas están las responsabilidades de una clase, de manera que no realice un trabajo colosal; una clase con baja cohesión realiza un trabajo exagerado, haciéndola difícil de comprender, reutilizar y conservar. Este patrón se pone de manifiesto en la clase MapServerSave, que se ocupa únicamente de salvar el archivo MapFile.

Bajo Acoplamiento: Es una medida de la fuerza con que una clase se relaciona con otras, porque las conoce y recurre a ellas; una clase con bajo acoplamiento no depende de muchas otras, sin embargo una con alto refleja varios inconvenientes. En la clase MapServerGenerate se demuestra el uso de este patrón, porque no depende de ninguna otra clase para realizar sus funcionalidades.

2.9 Conclusiones Parciales

El uso de las arquitecturas orientadas a objetos y en dos capas, además de patrones de diseño y estándares de codificación para la organización e implementación de la solución propuesta permitirán facilitar los cambios que podría tener la aplicación en el futuro y fomentar la reutilización de su código. Además la confección de los diagramas de diseño y clase utilizados para representar la estructura interna del sistema, facilitarán la ejecución de tareas de mantenimiento del código fuente.

El proceso de planificación y diseño realizado en este capítulo, forma parte de la orientación que debe tener el equipo de desarrollo para tener éxito en el proyecto. Luego de haber rebasado correctamente estas fases, se puede afirmar que las bases para comenzar con el desarrollo de la aplicación, están adecuadamente determinadas.

Capítulo 3: Implementación y pruebas del complemento GAMF-GeoQ

En el presente capítulo, se describe los elementos relacionados al flujo de trabajo de implementación de GAMF-GeoQ que define la metodología AUP. Se realizan los diagramas de despliegue y de componentes de implementación resultantes de esta etapa de construcción. Además, se describen detalladamente las pruebas de caja negra e integración que se le realizan al sistema, en busca de errores que pueda presentar en el funcionamiento y no se haya tomado en cuenta, o pasado por alto durante la implementación. También se exponen las contribuciones del complemento en la homogeneización de los mapas entre GeoQ y MapServer.

3.1 Modelo de Implementación

El Modelo de Implementación, es una visión general de lo que se debe implementar y está conformado por un conjunto de componentes y subsistemas, que forman la estructura física de la implementación del sistema. Estos pueden ser datos, archivos, ejecutables, código fuente y los directorios. Básicamente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos.

Diagrama de Componentes

Permite visualizar con más facilidad, la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Según (Schmuller, 2000) "*En lugar de representar una entidad conceptual como una clase o estado, un diagrama de componentes representa a un elemento real: un componente de software. Estos componentes se encuentran en las computadoras, no en la mente del analista"*.

El diagrama de componentes de GAMF-GeoQ, es la representación de la forma en que los elementos físicos del sistema serán separados, como lo son: archivos, cabeceras, módulos y paquetes. Muestra la organización y las dependencias que existen en GAMF-GeoQ, como muestra la Fig. 15.

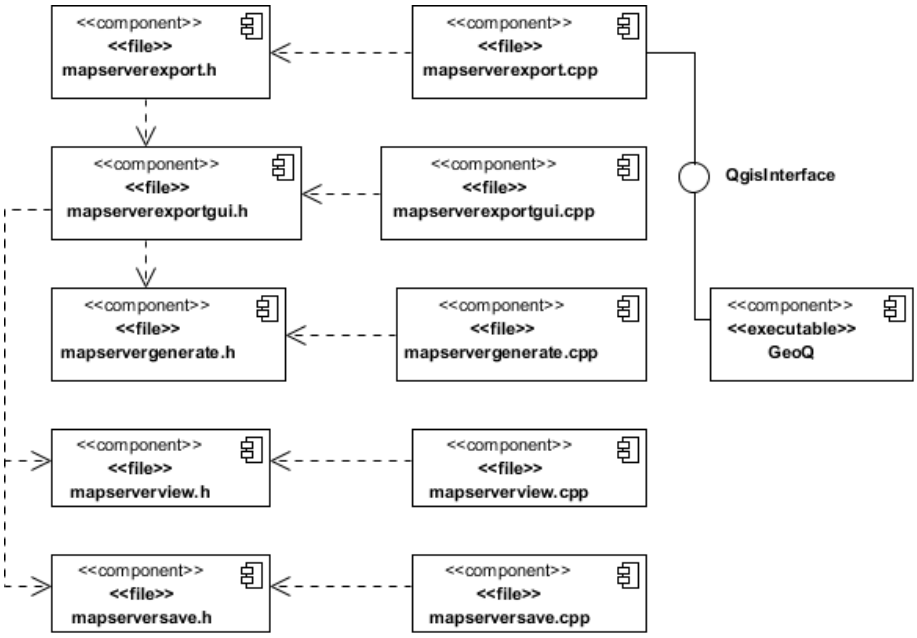


Fig. 15: Diagrama de Componentes de GAMF-GeoQ.

3.2 Modelo de Despliegue

Un diagrama de despliegue ilustra la forma en que luce un sistema físicamente cuando sea fusionado, estos están compuesto por nodos donde cada uno se representa por un cubo. Una línea asocia a dos cubos y simboliza una conexión entre ellos. Los tipos de nodos son procesador y dispositivo. Según (Schmuller, 2000), modela la arquitectura en tiempo de ejecución de un sistema. Esto muestra la configuración de los elementos de hardware (nodos) y cómo los artefactos del software se trazan en esos nodos.

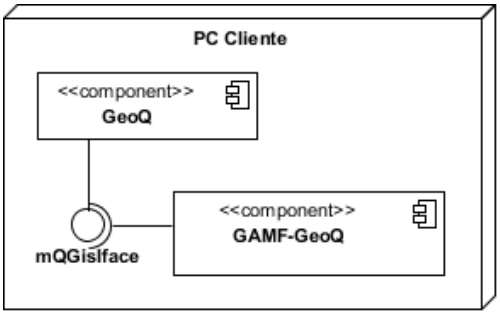


Fig. 16: Diagrama de Despliegue de GAMF-GeoQ.

En la Fig. 16, se muestra el diagrama de despliegue del plugin Generador de archivos MapFile para GeoQ, donde se muestra el nodo PC Cliente en el cual debe estar instalado la plataforma GeoQ y además, esta tiene que estar integrada con el componente GAMF-GeoQ. Con esta herramienta el usuario podrá generar los .map a partir de un proyecto de GeoQ.

3.3 Estándares de codificación

Un estándar de codificación no es más que un conjunto de reglas que se siguen para la generación de código fuente, con el propósito de reflejar un estilo armonioso, como si un único programador hubiera escrito todo de una sola vez. Al comenzar un proyecto de software, es necesario establecer un patrón, para asegurarse de que todos los programadores del proyecto trabajen de forma coordinada. Usar técnicas sólidas y realizar buenas prácticas de programación con vistas a generar un código de alta calidad, es de gran importancia para obtener un buen rendimiento. Todo esto posibilita que se obtenga un sistema fácil de comprender y mantener.

Estilo de codificación utilizado:

- Los nombres de las variables y funciones estarán escritos con el estándar “lowerCamelCase”, que se utiliza cuando la primera letra de cada una de la primera palabra es minúscula.
- Los nombres de las clases estarán escritos con el estándar “UpperCamelCase” que se utiliza para establecer que la primera letra de cada una de las palabras es mayúscula.
- Se utilizarán llaves para indicar la estructura del cuerpo de control, independientemente del número de declaraciones que contenga.
- Todas las funciones y clases estarán comentadas, para explicar el flujo del código y el propósito de las funciones o variables.

Ejemplo de código fuente

En la Fig. 17, se muestra un fragmento de código del método writeMapLayers, encargado de escribir las capas del archivo MapFile y que refleja los estándares utilizados.

```
// Escribe las capas del mapa
void MapServerGenerate::writeMapLayers(){
    QString resultMSG;

    // Se obtiene las capas de la leyenda para ser capaz de determinar el orden más tarde
    QDomNodeList legendLayerFile = this->parser.documentElement().elementsByTagName("legendlayerfile");
    QDomNodeList legendOrder;

    // Se obtiene la lista de nodos maplayer
    QDomNodeList layers = this->parser.documentElement().elementsByTagName("maplayer");

    for(int i = layers.length()-1; i >= 0; i--){
        QString epsg;
        QString dataString;
        QString providerString;
        QString layerId;
        QString layerName;
        // Los atributos de la etiqueta maplayer contienen los ajustes dependientes de escala,
        // Visibilidad, y el tipo de capa
        this->mapText += "  LAYER\n";

        for (QDomNode node = layers.at(i).lastChild(); !node.isNull(); node = node.previousSibling()){

            if (node.nodeName() == "layername"){
                // Se añade el nombre de la capa y se remplazan los espacios por underscore para cumplir con wms
                layerName = node.firstChild().nodeValue().replace("\"", "").replace(" ","_");
                this->mapText += "    NAME \"" + layerName + "\"\n";
            }
            // layerName no es único en qgis
            if (node.nodeName() == "id"){
                layerId = node.firstChild().nodeValue();
            }
```

Fig. 17: Ejemplo de código fuente del Método "writeMapLayers".

3.4 Pruebas de Software

Las pruebas del software son un conjunto de métodos y técnicas para garantizar la calidad del mismo. En este proceso se ejecutan pruebas dirigidas a componentes, o al sistema en su totalidad, con el objetivo de medir el grado en que la aplicación cumple con los requerimientos. Se puede definir como la ejecución del código usando combinaciones de entradas, en un determinado estado, para revelar defectos.

Prueba de Integración

Para una correcta integración del plugin GAMF-GeoQ con la plataforma GeoQ, es primordial que se copie el componente con la estructura adecuada y todas las funcionalidades implementadas, en la carpeta que la plataforma tiene habilitada para los complementos. Esto se debe, a que cada vez que se ejecuta va a esa carpeta y lee todo su contenido en busca de nuevos complementos. Posteriormente el usuario debe dar clic en el menú “complementos” de la barra de herramientas, después “administrar e instalar complementos”. Después de realizar esta acción se mostrará un formulario con todos los complementos habilitados y los disponibles, se debe marcar el complemento con nombre “Exportar a

MapServer". Seguidamente, éste aparecerá en el menú "complementos", para ser ejecutado siempre que el usuario requiera exportar un .map a partir de un proyecto de GeoQ. La Fig. 18 muestra la correcta integración del complemento a la plataforma.

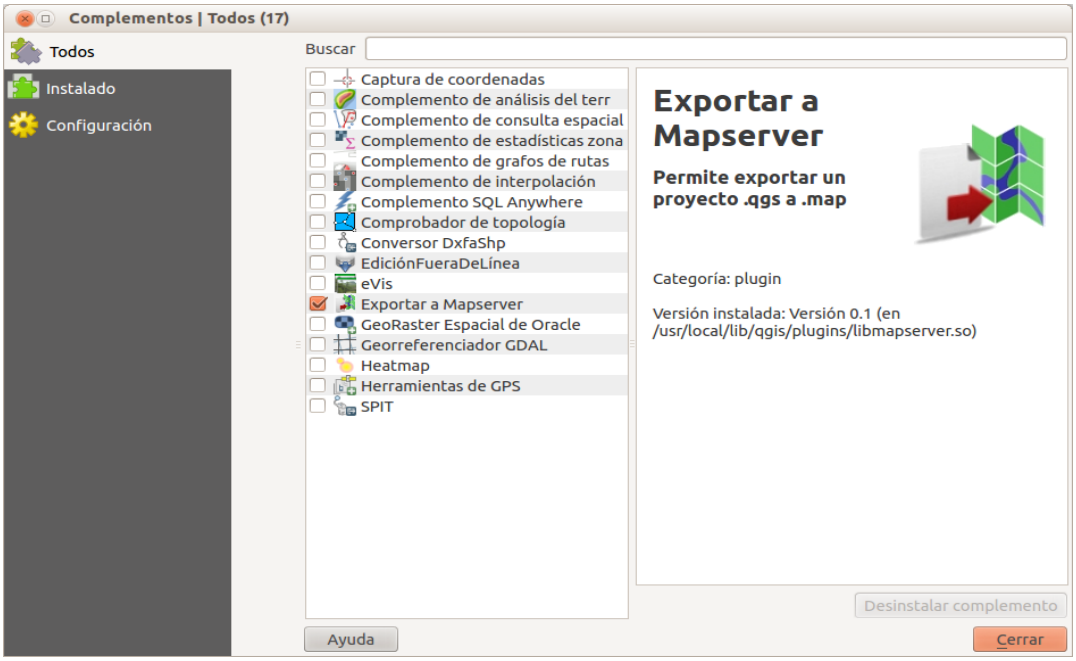


Fig. 18: Administrador de complementos de GeoQ.

Pruebas de Caja Negra

Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, obviando el comportamiento interno y la estructura del programa. Según (Pressman, 2005) estas se centran principalmente en los requisitos funcionales de la aplicación. Además, permiten obtener un conjunto de condiciones de entrada, que ejerciten completamente todas las condiciones que debe cumplir un programa.

En los casos de pruebas se combinan los posibles datos válidos e inválidos, que son necesarios para verificar el correcto funcionamiento del complemento GAMF-GeoQ. Estas se realizan a nivel de sistema, de tipo funcional, donde se emplea el enfoque estructural o de caja negra, específicamente dando uso a la técnica de particiones equivalentes. Estas técnicas son muy efectivas, pues permiten examinar los valores de las entradas existentes en el software para detectar los errores de manera inmediata. A continuación se realiza el diseño de caso de prueba correspondiente al caso de uso Exportar a MapServer.

Descripción General: El caso de uso inicia cuando el usuario selecciona la opción “Exportar a MapServer”. El sistema muestra una interfaz donde se realizan todas las configuraciones necesarias para exportar un proyecto de GeoQ a un archivo .map.

Condiciones de Ejecución: Debe existir un archivo .qgs.

Secciones a probar del caso de uso:

Tabla 7: Caso de prueba del CU “Exportar a MapServer”.

Nombre de la sección	Escenarios de la sección	Descripción de la funcionalidad	Flujo Central
SC1: Exportar a MapServer	EC 1.1: Exportar a MapServer satisfactoriamente.	El sistema muestra una interfaz, donde se realizan todas las configuraciones necesarias para exportar un proyecto de GeoQ a un archivo .map. Genera el MapFile y lo guarda hacia una dirección especificada.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se carga el archivo .qgs. - Realizar todas las configuraciones iniciales del archivo .map que se va a generar. - Indicar la ruta donde se guardará el archivo que contiene los símbolos del MapFile. - Indicar la ruta donde se guardarán las capas del MapFile. - Presionar el botón “Generar”. - Gestionar las propiedades del archivo .map. - Presionar el botón “Visualizar imagen del mapa”. - Indicar la ruta donde se guardará el MapFile generado. - Presionar el botón “Guardar”.

			- Se muestra un mensaje de confirmación “Archivo exportado correctamente”.
	EC 1.2: Guardar proyecto actual.	El sistema guarda el proyecto actual de GeoQ en caso de que este no haya sido guardado con anterioridad.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se muestra un mensaje indicando guardar el proyecto actual. - Seleccionar la opción “Si”. - Indicar la ruta donde se guardará el proyecto. - Dar clic en el botón “Aceptar”. - Se guarda el proyecto actual.
	EC 1.3: Cargar proyecto previamente guardado.	El sistema carga un proyecto previamente guardado desde una dirección donde se encuentra el archivo .qgs. Muestra el archivo de proyecto GeoQ en un cuadro editor de texto.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Indicar la ruta donde se encuentra el proyecto a través de la opción “Archivo de entrada .qgs”. - Dar clic en el botón “Aceptar”. - Se carga el proyecto de GeoQ.
	EC 1.4: No se carga el .qgs.	El sistema no carga el .qgs por un error sintáctico y muestra en mensaje en función del error generado.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se muestra un mensaje “Error de lectura o al cargar el archivo de proyecto”.
	EC 1.5: Mantener propiedades por defecto del .map.	No se modifican las propiedades por defecto para el archivo .map que se va a generar.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se carga el archivo .qgs.

			- Mantener las propiedades por defecto del .map que se va a generar. - Presionar el botón “Generar”.
	EC 1.6: No se separan los símbolos.	El sistema no guardará el archivo con los símbolos independientes del MapFile, por lo que estos se incluyen en el mismo.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se carga el archivo .qgs. - El usuario no indica la ruta donde se guardarán los símbolos. - Presionar el botón “Generar”.
	EC 1.7: No se separan las capas.	El sistema no guardará las capas independientes del MapFile, por lo que estas se incluyen en el mismo.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se carga el archivo .qgs. - El usuario no indica la ruta donde se guardarán las capas. - Presionar el botón “Generar”.
	EC 1.8: No se gestionan las propiedades	No se insertan propiedades al MapFile generado.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se carga el archivo .qgs. - Presionar el botón “Generar”. - No se realiza ningún cambio en el MapFile generado. - Indicar la ruta donde se guardará el MapFile generado. - Presionar el botón “Guardar”.
	EC 1.9: No se visualiza	No se ejecuta la opción de visualizar la imagen del MapFile generado.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se carga el archivo .qgs. - Presionar el botón “Generar”.

			- No se ejecuta la opción “Visualizar imagen del mapa”. - Indicar la ruta donde se guardará el MapFile generado. - Presionar el botón “Guardar”.
	EC 1.10: Error al visualizar	No se visualiza la imagen del MapFile generado y en su lugar muestra un mensaje “Existen errores en la confección del MapFile”.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se carga el archivo .qgs. - Presionar el botón “Generar”. - Presionar el botón “Visualizar imagen del mapa”. - No se muestra la imagen del mapa y en su lugar muestra un mensaje “Existen errores en la confección del MapFile”.
	EC 1.11: Sobrescribir archivo MapFile	El sistema sobrescribe el MapFile existente por el nuevo generado.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se carga el archivo .qgs. - Presionar el botón “Generar”. - Se muestra un mensaje “El MapFile… ya existe. ¿Desea sobrescribirlo?”. - Seleccionar la opción “Si”. - Se sobrescribe el MapFile.
	EC 1.12: Error al guardar MapFile	El sistema muestra un mensaje indicando un error al intentar guardar el MapFile generado.	- Seleccionar la opción “Exportar a MapServer” dentro del menú “Complementos”. - Se carga el archivo .qgs. - Presionar el botón “Generar”. - Indicar la ruta donde se guardará el MapFile generado. - Presionar el botón “Guardar”.

			- Se muestra un mensaje “Error al intentar crear o abrir el archivo”.

Tabla 8: Descripción de las variables del caso de prueba “Exportar a MapServer”.

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Nombre	Cadena de caracteres	No	Se debe definir el nombre del mapa que se desea generar.
2	Ancho	Numérico	No	Se debe definir la anchura que tendrá el mapa.
3	Alto	Numérico	No	Se debe definir la altura que tendrá el mapa.
4	Tipo de imagen	Lista desplegable	No	Se debe seleccionar el tipo de imagen.
5	Unidades	Lista desplegable	No	Se debe seleccionar la unidad de medida que tendrá el mapa.
6	Conjunto de fuentes	Ruta	Sí	Se debe especificar la ruta del archivo que contendrá el conjunto de fuentes.
7	Símbolos separados	Ruta	Sí	Se debe especificar la ruta para guardar los símbolos que se generan.
8	Capas separadas	Ruta	Sí	Se debe especificar la ruta para guardar las capas que se generan.
9	Archivo de entrada (.qgs)	Ruta	No	Se debe especificar la ruta para cargar el archivo de proyecto previamente guardado.
10	Archivo de salida (.map)	Ruta	No	Se debe especificar la ruta para guardar el MapFile generado.

Matriz de Datos:

SC1: Exportar a MapServer

Tabla 9: Matriz de datos del caso de prueba “Exportar a MapServer”.

ID del EC	EC 1.1	EC 1.2	EC 1.3	EC 1.4
Escenario	Exportar a MapServer satisfactoriamente.	Guardar proyecto actual.	Cargar proyecto previamente guardado.	No se carga el .qgs.
V1	V(MapFile)	NA	NA	NA
V2	V(700)	NA	NA	NA
V3	V(700)	NA	NA	NA
V4	V(png)	NA	NA	NA
V5	V(kilómetros)	NA	NA	NA
V6	V()	NA	NA	NA
V7	V(/home/barbaro/Escritorio/symbol.sym)	NA	NA	NA
V8	V(/home/barbaro/Escritorio/)	NA	NA	NA
V9	NA	NA	V(/home/barbaro/Escritorio/proyecto.qgs)	NA
V10	V(/home/barbaro/Escritorio/)	NA	NA	NA
Respuesta del Sistema	El sistema muestra un mensaje indicando que se exporto correctamente el MapFile.	El sistema guarda el proyecto actual de GeoQ.	El sistema muestra el archivo de proyecto GeoQ en un cuadro editor de texto.	El sistema muestra un mensaje “Error de lectura o al cargar el archivo de proyecto”.
Resultado de la prueba	Satisfactoria	Satisfactoria	Satisfactoria	Satisfactoria

ID del EC	EC 1.5	EC 1.6	EC 1.7	EC 1.8
Escenario	Mantener propiedades por defecto del .map.	No se separan los símbolos.	No se separan las capas.	No se gestionan las propiedades
V1	V(GeoQ-MAP)	V(GeoQ-MAP)	V(GeoQ-MAP)	V(GeoQ-MAP)
V2	V(650)	V(650)	V(650)	V(650)
V3	V(650)	V(650)	V(650)	V(650)

V4	V(agg)	V(agg)	V(agg)	V(agg)
V5	V(dd)	V(dd)	V(dd)	V(dd)
V6	V()	V()	V()	V()
V7	V()	V()	V(/home/barbaro/Escritorio/symbol.sym)	V()
V8	V()	V(/home/barbaro/Escritorio/)	V()	V()
V9	NA	NA	NA	NA
V10	NA	NA	NA	V(/home/barbaro/Escritorio/)
Respuesta del Sistema	No se modifican las propiedades por defecto para el archivo .map que se va a generar.	El sistema no guardará el archivo con los símbolos independientes del MapFile, por lo que estos se incluyen en el mismo.	El sistema no guardará las capas independientes del MapFile, por lo que estas se incluyen en el mismo.	No se insertan propiedades al MapFile generado.
Resultado de la prueba	Satisfactoria	Satisfactoria	Satisfactoria	Satisfactoria
D del EC	EC 1.9	EC 1.10	EC 1.11	EC 1.12
Escenario	No se visualiza	Error al visualizar	Sobrescribir archivo MapFile	Error al guardar MapFile
V1	V(GeoQ-MAP)	V(GeoQ-MAP)	V(GeoQ-MAP)	V(GeoQ-MAP)
V2	V(650)	V(650)	V(650)	V(650)
V3	V(650)	V(650)	V(650)	V(650)
V4	V(agg)	V(agg)	V(agg)	V(agg)
V5	V(dd)	V(dd)	V(dd)	V(dd)

V6	V()	V()	V()	V()
V7	V()	V()	V()	V()
V8	V()	V()	V()	V()
V9	NA	NA	NA	NA
V10	V(/home/barbaro/Escritorio/)	NA	V(/home/barbaro/Escritorio/)	V(/home/barbaro/Escritorio/)
Respuesta del Sistema	No visualiza la imagen del MapFile generado.	El sistema muestra un mensaje “Existen errores en la confección del MapFile”.	El sistema sobrescribe el MapFile existente por el nuevo generado.	El sistema muestra un mensaje indicando un error al intentar guardar el MapFile generado.
Resultado de la prueba	Satisfactoria	Satisfactoria	Satisfactoria	Satisfactoria
ID del EC	**EC 1.9**	**EC 1.10**	**EC 1.11**	**EC 1.12**
Escenario	No se visualiza	Error al visualizar	Sobrescribir archivo MapFile	Error al guardar MapFile
V1	V(GeoQ-MAP)	V(GeoQ-MAP)	V(GeoQ-MAP)	V(GeoQ-MAP)
V2	V(650)	V(650)	V(650)	V(650)
V3	V(650)	V(650)	V(650)	V(650)
V4	V(agg)	V(agg)	V(agg)	V(agg)
V5	V(dd)	V(dd)	V(dd)	V(dd)
V6	V()	V()	V()	V()
V7	V()	V()	V()	V()
V8	V()	V()	V()	V()
V9	NA	NA	NA	NA

V10	V(/home/barbaro/Escritorio/)	NA	V(/home/barbaro/Escritorio/)	V(/home/barbaro/Escritorio/)
Respuesta del Sistema	No visualiza la imagen del MapFile generado.	El sistema muestra un mensaje “Existen errores en la confección del MapFile”.	El sistema sobrescribe el MapFile existente por el nuevo generado.	El sistema muestra un mensaje indicando un error al intentar guardar el MapFile generado.
Resultado de la prueba	Satisfactoria	Satisfactoria	Satisfactoria	Satisfactoria

Resultado de las pruebas de caja negra

Una vez diseñado el caso de prueba, este se ejecuta al sistema, con el propósito de comprobar que el software cumple con todas las funcionalidades establecidas y que está en condiciones de ser entregado al cliente. El componente GAMF-GeoQ fue sometido a una primera iteración de pruebas, la que arrojó cinco no conformidades, luego de corregidas las mismas se realiza una segunda iteración obteniendo resultados satisfactorios como se evidencia en la Fig. 19.

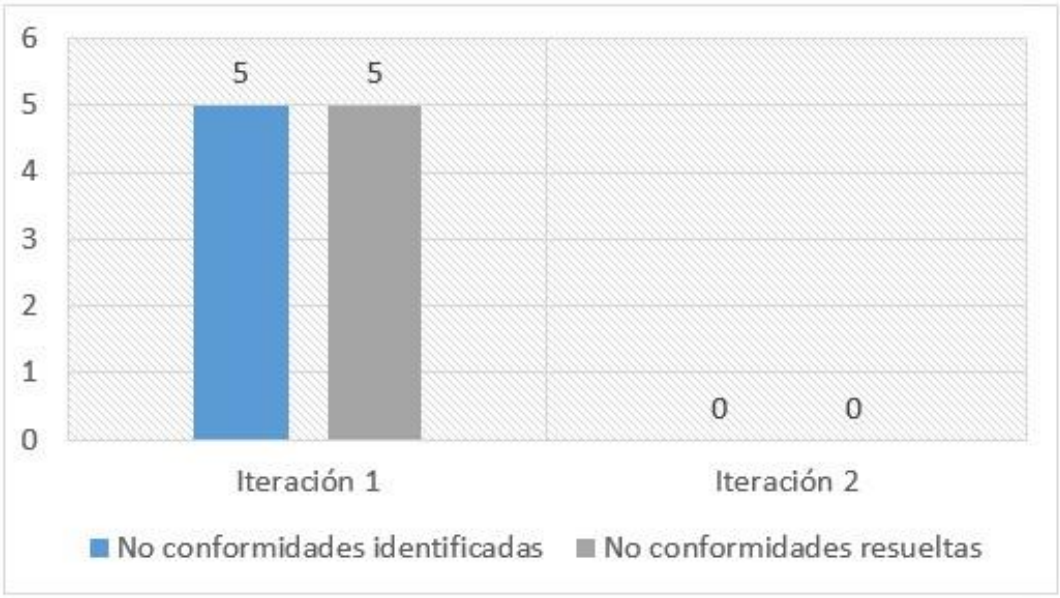


Fig. 19: Resultado de las pruebas de caja negra.

3.5 Contribuciones del plugin GAMF-GeoQ

El desarrollo del plugin presentado como solución al problema que sustenta la investigación, se encuentra encaminado a brindar una serie de ventajas en la interoperabilidad del proceso de representación cartográfica da la plataforma GeoQ y MapServer. Además mejora considerablemente, el tiempo empleado en la confección de los archivos MapFile y elimina posibles errores en su estructura. Es fundamental aclarar que para limitar el alcance de la investigación, se pretende compatibilizar las representaciones en una sola dirección, o sea desde la plataforma GeoQ a partir de la versión 2.0, hacia la 5.6 de MapServer. También será compatible con las versiones de QGIS superiores a la 1.8 que traen incorporado la nueva forma de manejar los símbolos.

Una vez que el complemento se encuentre integrado a la plataforma, podemos hacer uso del mismo siguiendo los pasos que fueron expuestos en la descripción del CU “Exportar a MapServer”. Como resultado se obtiene un archivo MapFile, que contiene correctamente todas las configuraciones y símbolos que se utilizaron en GeoQ, para dar estilos a los orígenes de datos que provengan de ficheros Shapefile, de una base de datos Postgres, o una consulta WMS. La Fig. 20 muestra el flujo de eventos que refleja cómo pueden lograr dicha interoperabilidad entre MapServer y GeoQ.

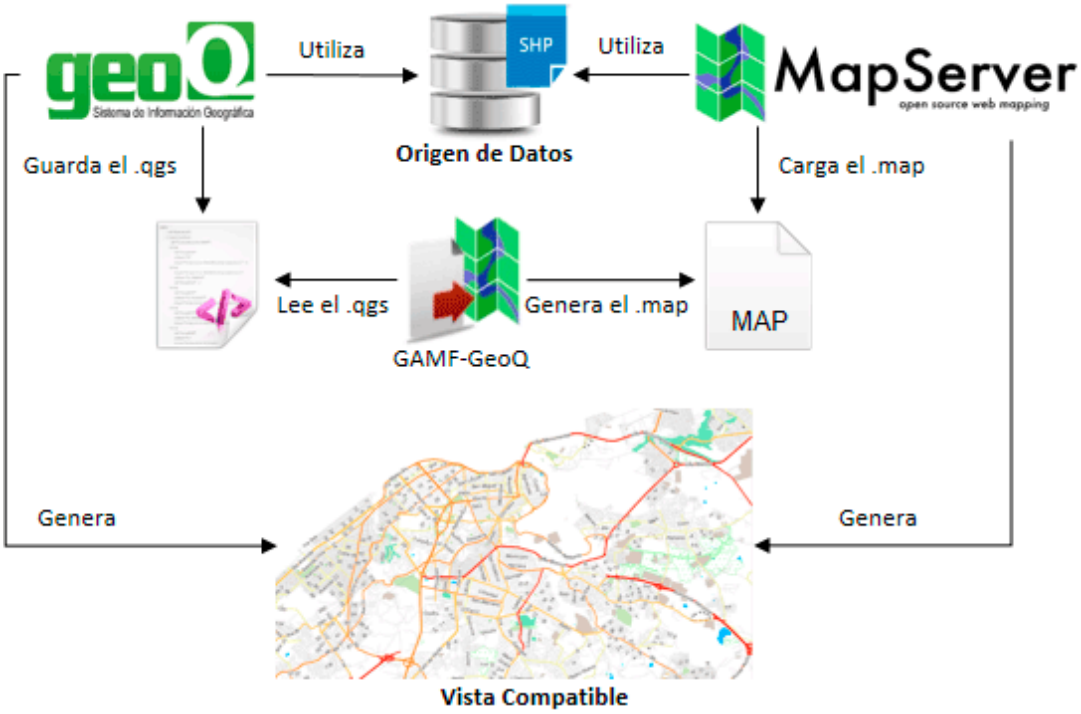


Fig. 20: Flujo de eventos que refleja interoperabilidad entre GeoQ y MapServer.

3.6 Conclusiones Parciales

El modelo del diagrama de componentes al reflejar la estructura general de la solución a través de la interacción entre sus diferentes componentes, permite comprender como debe funcionar el sistema en tiempo de ejecución, durante la generación de archivos MapFile. Además los resultados de las pruebas de software aplicadas a la solución, demostraron que en diferentes escenarios las funcionalidades implementadas se ejecutan correctamente y que la aplicación desarrollada cumple con la calidad requerida para ser utilizada por los especialistas de los proyectos SIG-Desktop y Aplicativos SIG.

Conclusiones Generales

La presente investigación presenta el complemento GAMF-GeoQ, que no es más que una herramienta para la generación de archivos MapFile en la plataforma GeoQ, a partir de un archivo de proyecto de este último. Este plugin le añade un valor práctico al SIG, permitiendo a los especialistas del proyecto Aplicativos SIG realizar los estilos de cualquier mapa en GeoQ, luego exportarlo y visualizarlo en MapServer. Para finalizar, a continuación se exponen las principales conclusiones a las que se arribaron durante la presente investigación:

- Después de realizar un estudio del arte, sobre las diferentes herramientas que existen para generar un archivo MapFile, se puede concluir que ningunas de las estudiadas resuelve el problema que sustenta la investigación. Aunque son de código libre, lo que permitió estudiar la forma y la lógica que siguen este tipo de aplicaciones. Lo que facilitó la confección de un generador de archivos MapFile propio para GeoQ. Para ello se tomaron características que estos presentan como guía en el desarrollo del complemento.
- También el uso de las arquitecturas orientadas a objetos y en dos capas, además de patrones de diseño y estándares de codificación para la organización e implementación de la solución propuesta permitirán facilitar los cambios que podría tener la aplicación en el futuro y fomentar la reutilización de su código. Además la confección de los diagramas de diseño y clase utilizados para representar la estructura interna del sistema, facilitarán la ejecución de tareas de mantenimiento del código fuente.
- De igual forma el modelo del diagrama de componentes al reflejar la estructura general de la solución a través de la interacción entre sus diferentes componentes, permite comprender como debe funcionar el sistema en tiempo de ejecución, durante la generación de archivos MapFile. Además los resultados de las pruebas de software aplicadas a la solución, demostraron que en diferentes escenarios las funcionalidades implementadas se ejecutan correctamente y que aplicación desarrollada cumple con la calidad requerida para ser utilizada por los especialistas de los proyectos SIG-Desktop y Aplicativos SIG.

Recomendaciones

Durante el desarrollo del sistema han surgido ideas, que contribuirían a que el generador de archivos MapFile gane en fortaleza, por lo que se recomienda:

- Incorporar en el sistema soporte para generar archivos MapFile de manera genérica, o sea en dependencia de la versión de MapServer.
- Incorporar en el sistema la posibilidad de exportar las fuentes que usa GeoQ, que no son más que las fuentes del sistema y utilizarlas como las disponibles para MapServer.
- Incorporar la opción de importar un archivo MapFile y convertirlo en un proyecto de GeoQ.
- Incorporar la posibilidad de exportar un proyecto de GeoQ que contenga orígenes de datos provenientes de otras bases de datos como Oracle o consultas WFS.

Referencias Bibliográficas

Regents of the University of Minnesota. 2013. *MapServer open source web mapping.* [En línea] 2013. [Citado el: 4 de Diciembre de 2013.] http://mapserver.org/mapfile/.

Ambler, Scott W. . 2014. The Agile Unified Process. *Ambysoft.* [En línea] 2014. [Citado el: 25 de Marzo de 2014.] http://www.ambysoft.com/unifiedprocess/agileUP.html.

Beck, Kent. 1999. *eXtreme Progammnig explained.* 1999.

Brian Harley, John. 2005. *La nueva naturaleza de los mapas: ensayos sobre la historia de la cartografía.* s.l. : Fondo de Cultura Económica, 2005.

Camacho, Erika, Cardeso, Fabio y Nuñez, Gabriel. 2004. *ARQUITECTURAS DE SOFTWARE.* 2004.

Carmona, Alvaro y Monsalve, Jhon Jairo. 1999. *SISTEMAS DE INFORMACIÓN GEOGRÁFICOS.* 1999.

Castillo Reyes, Grethell, Valle Martínez, Yusnier y Ramón Antunez, Romanuel. 2013. *Herramienta para la gestión de mapas en Sistemas de Información Geográfica.* La Habana : s.n., 2013. 2306-2495.

Domínguez Bravo, Javier. 2000. *Breve Introducción a la Cartografía y a los Sistemas de Información Geográfica (SIG).* Madrid : CIEMAT, 2000. 1135-9420.

East, Russell, y otros. 2002. *The Architecture of ArcIMS, a Distributed Internet Map Server.* New York : s.n., 2002.

Edeki, Charles. 2013. *Agile Unified Process.* New York : s.n., 2013. 2321-8363.

Esri. 2012. ArcGIS Resources. [En línea] 2012. [Citado el: 25 de Diciembre de 2013.] http://resources.arcgis.com/es/help/getting-started/articles/026n0000000r000000.htm.

EUI - FI . Universidad Politécnica de Valencia. 2003. *Introducción a Herramientas CASE y System Architect.* 2003.

Figueroa, Roberth G., Solís, Camilo J. y Cabrera, Armando A. *Metodologias Tradicionales VS. Metodologías Ágiles.* s.l. : Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

Franco Maass, Sergio y Valdez Pérez, Eugenia. 2002. *Principios Básicos de Cartografía y Cartografía automatizada.* Toluca : s.n., 2002. 968-835-833-9.

Gallagher, Shaun. 2009. *Brainstorming: Views and Interviews on the Mind.* California : s.n., 2009. 1845400232.

Referencias Bibliográficas

Gamma, Erich, y otros. 1995. *Dessign Patterns. Elements of Reusable Object-Oriented Software.* s.l. : Addison Wesley, 1995.

Gix Project. 2004. *gix Export Tool ArcView extension.* [En línea] 2004. [Citado el: 2 de Diciembre de 2013.] http://gix.sourceforge.net/.

Gutiérrez González, Rocío y Sánchez González, Raúl. 2002. *Comparativa entre las biblotecas gráficas GTK + y QT.* 2002.

Highsmith, Jim. 2002 . *Agile Software Development Ecosystems.* s.l. : Addison Wesley, 2002 . 0-201-76043-6.

Jacobson, Ivar, Booch, Grady and Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* s.l. : Pearson educación, 2000. 84-78-29-036-2.

Kareaga Cantero, Iñaki. 2008. *Diseño y Desarrollo de herramientas de validación topologica en el entorno gvSIG.* Catalunya : s.n., 2008.

Ledea, L. M. 2010. *Diseño de la arquitectura base del catálogo de mapas. Universidad de las Ciencias Informáticas.* 2010.

Letelier, Patricio y Penadés, M. Carmen. 2006. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Valencia : s.n., 2006. 46022.

Manso Callejo, Miguel Angel y Ballari, Daniela. 2009. *Instalación de MapServer como WMS, WFS y WCS.* 2009.

Martínez Piedra, Ramón. 2005. *Sistemas de Información Geográfica Aplicaciones y métodos analíticos en la prevención y control del Dengue.* La Habana : s.n., 2005.

McKenna, Jeff, Fawcett, David y Butler, Howard. 2009. *MapServer Documentation Release 5.6.1.* 2009.

Méndez Roldán, Isuel. 2010. *El servidor de mapas MapServer, una solución recomendada para la representación de información geoespacial.* 2010.

Muñoz, Moncayo. 2009. *Servidores de Mapas.* 2009.

Norvell, Theodore. 2010. *Visual Paradigm for UML Tutorial.* 2010.

Olivares Flores, Linda I. 2008. *Manual de Programación en Lenguaje C++.* 2008.

Oyala, Víctor. 2011. *Sistemas de Informacion Geográficas.* 2011.

Referencias Bibliográficas

Pressman, Roger S. 2005. *Ingenieria de software un enfoque practico. Quinta Edición.* Nueva York : Editorial McGraw-Hill, 2005.

Rearte, Emilio. 2002. *Bases de Datos y UML.* 2002.

Schmuller, Joseph. 2000. *Aprendiendo UML en 24 horas.* Mexico : Pearson Educacion, 2000. 968-444-463-X.

Sherman, Gary E., y otros. 2009. *QGIS User Guide.* 2009.

Stallman, Richard M. 2004. *Software libre para una sociedad libre.* 2004. ISBN: 84-933555-1-8.

Star, J. y Estes, John E. 1990. *Geographic Information Systems: An Introduction.* s.l. : Prentice Hall, 1990.

Stroustrup, Bjame. 2013. *The C++ Programming Language fourth edition.* 2013.

Summerfield, Mark y Blanchette, Jasmin. 2008. *C++ GUI Programming.* 2008. 0-13-235416-0.

Tabares, Ricardo Botero. 2010. *Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación.* 2010.

Thelin, Johan. 2007. *Foundations of Qt Development.* New York : s.n., 2007. 1-59059-831-8.

Ticheler, Jeroen. 2007. *ESRI.* [En línea] 7 de Enero de 2007. [Citado el: 2 de Diciembre de 2013.] http://arcscripts.esri.com/details.asp?dbid=12766.

Universidad de las Ciencias Informáticas. 2010. *Metodología de la Investigación tema3: El diseño metodológico de la investigación científica.* La Habana : s.n., 2010.

Varma, Vasudeva. 2009. *Software Architecture.* Delhi : s.n., 2009. 978-81-317-0749-4.

Venegas, Albornoz y Andrés, Mario. 2006. *Catastro complementado con información sobre uso de recintos y publicación WEB utilizando MapServer como herramienta del plan maestro de la univiesidad de Santiago de Chile.* Santiago de Chile : s.n., 2006.

Anexos

Entrevista realizada a los jefes de los proyectos SIG-Desktop y Aplicativos-SIG.

1) ¿Cómo se crean actualmente los archivos MapFile?
2) Al integrarse las plataformas GeoQ y GeneSIG, ¿cómo hacen para portar el o los mapas que se gestionan en GeoQ y se van a utilizar?
3) ¿Qué versión del MapServer utilizan para sus proyectos?
4) ¿Existen configuraciones que se pueden hacer en GeoQ y no en MapServer? ¿Cuáles son?
5) ¿Les beneficiaria una herramienta que portara las configuraciones de los mapas de GeoQ hacia MapServer?