UNIVERSIDAD DEL AZUAY

FACULTAD CIENCIAS DE LA ADMINISTRACION

ESCUELA DE INGENIERÍA EN SISTEMAS

GEOINFORMACION DE LA CIUDAD DE CUENCA EN LA INTERNET

Tesis previa a la obtención del titulo de
Ingeniero en Sistemas

AUTOR: Javier Andrés García Galarza

DIRECTOR: Ing. Paúl Ochoa A

CUENCA, ECUADOR
2007

DEDICATORIA

Dedicado a mis padres Leonardo y Cecilia quienes han sabido brindar todo el apoyo y confianza para cumplir mis metas, gracias a su amor y colaboración me ha sido posible llegar a esta parte del camino.

A mi esposa María Rosa, la mejor compañera y amiga en mi vida, gracias por haber iluminado mi camino con apoyo y las ganas, gracias por tu amor y comprensión.

AGRADECIMIENTO

Un agradecimiento especial al Ing. Paúl Ochoa, por la gran ayuda que ha sido clave para la elaboración de este trabajo, aportando de forma desinteresada con ideas y empeño para obtener los mejores resultados en el proyecto.

Las opiniones vertidas en éste trabajo de investigación son propiedad del autor y están bajo la responsabilidad del mismo.

_____

Javier Andrés García G.

Índice de Contenidos

REFERENCIAS

Índice de Ilustraciones

Índice de Tablas

RESUMEN

El presente trabajo de tésis plantea la creación de una aplicación desarrollada mediante uso de software libre o de bajo costo en la que podamos visualizar y consultar información geográfica de la ciudad de Cuenca por medio de la Web, utilizando información espacial recopilada de diferentes fuentes, obteniendo como producto final un mapa interactivo de la ciudad con herramientas de navegación para desplazarnos en el mapa, opciones de búsqueda de direcciones y lugares de interés permitiendo ubicar cualquier dirección o lugar dentro de la ciudad, y la consulta de información a cerca de éstos lugares que sea de utilidad para el usuario.

ABSTRACT

This thesis proposes the creation of an application developed though the use of free or low-cost software where we can visualize and consult geographical information of Cuenca through th Web, usinf spatial information gathered from different sources. The final product will be an interactive map with navigation tools to move on it, and options for the search of addresses and places of interest which will allow to locate any address or place inside the city and consult information about places of interest for the user.

INTRODUCCIÓN

El creciente y rápido desarrollo de la Internet y la accesibilidad a diversa información ha dado lugar a una nueva generación de Sistemas de Información Geográfica. Los SIG basados en Internet combinan las posibilidades de tomar decisiones y análisis geográfico con la personalización, accesibilidad, y poder interactivo de la Internet.

Los sistemas basados en web tienen la potencialidad de facilitar el funcionamiento de aplicaciones distribuidas. Laurini (2004) propone el uso de la Internet como "Un medio para intercambiar información, ideas, mapas entre todos los actores".

Un SIG basado en web con algunas funciones como zoom, pan, intercambio de capas temáticas y funciones de búsqueda y consulta, puede permitir a un usuario obtener información particular sobre sus necesidades que pueden ser totalmente individuales

El Objetivo de este trabajo es investigar y evaluar las opciones que tenemos para publicar información geográfica en la Web, realizando una aplicación en la que, utilizando la mejor alternativa de software, obtengamos una representación virtual de la ciudad de Cuenca, permitiéndonos navegar por un mapa dinámico realizando funciones como zoom, pan, consultas y búsquedas que nos ayuden a la localización de direcciones y lugares georeferenciados; todo esto mediante un navegador con acceso a Internet.

Para cumplir estos objetivos es necesario realizar una recolección y levantamiento de información (capítulos 1 y 2) existente que utilizaremos para la visualización y consultas, para luego hacer un análisis para identificar los requisitos, arquitectura y datos que debe tener la aplicación escogiendo las herramientas más apropiadas para su desarrollo.

Luego diseñaremos el comportamiento de la aplicación modelando un diseño conceptual para la base de datos (capítulos 3 y 4), organización y propiedades de las

capas de información espacial, diseñando los diferentes módulos y componentes en cliente y servidor que comprenderán la aplicación Web.

La implementación de sistema (capítulo 5) consistirá en el desarrollo de los modelos que se realizó en el diseño y que cumplan con los requerimientos que nos hemos propuesto para cumplir con los objetivos planteados en ésta investigación.

Finalmente (capítulo 6) se plantean las conclusiones y recomendaciones que pueden establecerse con miras a desarrollos futuros sobre la prublicación de mapas en la Web.

CAPITULO 1

INTRODUCCIÓN A SISTEMAS DE INFORMACIÓN GEOGRÁFICA Y SIG BASADO EN WEB.

Introducción

En la actualidad los Sistemas de Información Geográfica (SIG) en la Internet se han masificado de forma sorprendente gracias a sitios Web como Google, Yahoo, Microsoft y gobiernos de países y ciudades que han desarrollado aplicaciones con capacidades geográficas para publicar información de tipo turístico e informativo, ofreciendo a los usuarios un servicio de mapa digital donde pueden consultar sitios de interés, buscar direcciones, etc.

Los visores de mapas sobre Web que se ejecutan en navegadores comunes ("browsers") permiten la recuperación y visualización de información espacial rápida y fácil por muchos usuarios, sin que para ellos sea necesario tener conocimiento sobre herramientas SIG.

A continuación en este capitulo vamos a revisar algunas definiciones y conceptos que nos ayudaran a comprender la importancia de un SIG basado en Web.

1.1 Sistemas de Información Geográfica (SIG).

1.1.1 Que es un SIG

Existen muchas y diferentes definiciones que describen a los Sistemas de Información Geográfica dependiendo del campo al que estos están aplicados,

Un Sistema de Información Geográfico es una organizada colección de hardware, software, datos geográficos y personal preparado; para la captura, almacenamiento,

actualización, análisis y visualización de información geográficamente referenciada (ESRI, 1995).

Cho (1998) nos da una lista de definiciones básicas que describen a los SIG como "Sistemas computarizados que son usados para guardar y manipular información geográfica" y "un conjunto de funciones autómatas para profesionales con avanzadas capacidades para levantar, manipular y mostrar datos ubicados geográficamente".

Una definición muy apropiada es la que encontramos en Wikipedia

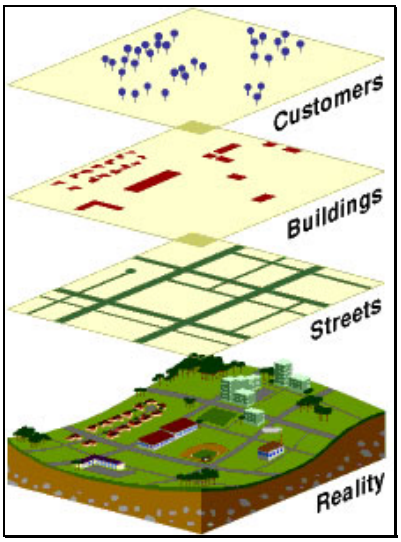
(Wikipedia-SIG) "Un Sistema de Información Geográfica (SIG o GIS, en su acrónimo inglés) es una integración organizada de hardware, software, datos geográficos y personal, diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada con el fin de resolver problemas complejos de planificación y gestión. También puede definirse como un modelo de una parte de la realidad referido a un sistema de coordenadas terrestre y construido para satisfacer unas necesidades concretas de información."

Los SIG están en la actualidad ampliamente usados para resolver una variedad de problemas como prevención de desastres naturales, estudio de suelos, planes de contingencia ante desastres, control de la población y muchos mas campos donde el uso de estas herramientas dan un gran aporte a la comunidad, junto con el uso del World Wide Web (WWW), los Sistemas de Información Geográfica en la Web podrían ser el futuro para permitir a un extenso número de personas el acceso a funcionalidades SIG, permitiendo a la comunidad la participación en aplicaciones, realizar consultas y análisis de la información.

1.1.2 Como un SIG almacena y usa los datos

Un SIG representa la información como características y eventos en una colección de capas como nos muestra la siguiente imagen.

Fig. 1. Representación de un SIG



Fuente: www.esri.com

Esta es la característica mas importante de un SIG, por ejemplo un sistema de información tradicional puede almacenar toda la información de todas las escuelas, probablemente incluyendo direcciones, pero el usuario no podría conocer donde exactamente se encuentra una determinada escuela, preguntas simples como “Cuales son las esuelas que ese encuentran en la calle x” no es imposible de contestar con un sistema tradicional.

1.1.3 Como un SIG muestra los datos.

Los datos de un SIG son compuestos por entidades graficas (líneas, puntos, polígonos) y una tabla asociada con cada entidad. Las entidades graficas muestran una ubicación en un sistema de referencia generalmente representado en un mapa. Individualmente cada entidad grafica tiene un registro asociado en la tabla.

1.1.4 Que puede hacer un GIS

Un Gis nos ofrece una gran variedad de herramientas para manipular, consultar, analizar y visualizar datos geográficos.

El tipo de herramientas analíticas que un SIG ofrece están relacionadas con análisis espacial y geográfico. La ubicación de un elemento o evento es el objetivo del análisis.

Antes de la utilización de herramientas SIG, el análisis geográfico era muy complicado y algunas veces imposible, en la actualidad preguntas críticas de gran complejidad son respondidas por herramientas SIG, por ejemplo:

- ¿Cual es la mejor ruta en una emergencia?
- ¿Cuáles propiedades en el área están a 20 Km de un desastre?
- ¿Qué tipos de suelos están en determinada área?
- ¿Cuál es la mejor ruta en automóvil de un lugar a otro?

Como podemos ver, una aplicación GIS puede responder a una gran variedad de preguntas relacionadas con elementos ubicados geográficamente o eventos sean naturales o realizados por el hombre.

1.1.5 Quien usa Sistemas de Información Geográfica.

Los SIG son usados por muchas industrias, negocios comerciales, transportación, salud, agricultura, gobierno, y muchos campos de aplicación, Empresas de Agua Potable usan GIS como una base de datos espacial de tuberías y alcantarillas, los gobiernos pueden usar GIS para manejar y actualizar predios y propiedades, Instituciones pueden también usarlo para provisión de servicios tales como salud, agua, educación ,etc. Tomando en cuenta la cantidad, distribución de la población, facilidades de acceso, etc.

Hace poco, principalmente en países de gran movimiento económico los SIG están siendo usados en negocios comerciales, para identificar posibles mercados y ayudar a determinar la mejor ubicación para nuevos locales basados en factores estadísticos, ubicación de la competencia, fácil acceso y ubicación de los clientes.

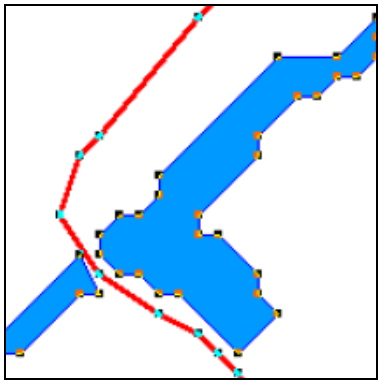
1.1.6 Representación de Información Geográfica

La información geográfica con la cual se trabaja en los SIG. Puede encontrarse en dos tipos de presentaciones o formatos: Celular o ráster y Vectorial.

1.1.6.1 Modelo Vectorial.

Los gráficos vectoriales se conforman de elementos geométricos tales como puntos, líneas, curvas o polígonos, Estos datos frecuentemente tienen asociada una tabla de información, una por cada tipo (punto, línea o polígono).

Fig2. Gráfico Vectorial



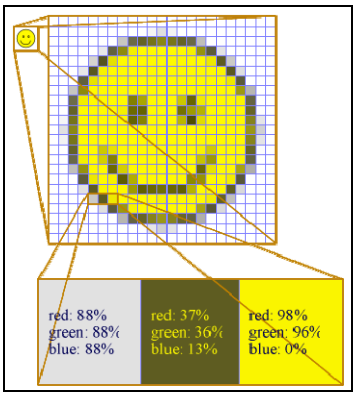
Fuente www.esri.com

1.1.6.2 Modelo Ráster.

Una imagen rasterizada es una estructura que representa una matriz con filas y columnas, donde cada celda o elemento es un píxel o punto de color definido individualmente; cada celda tiene un valor asociado, por ejemplo la altura, tipo de suelo, etc.

Los gráficos rasterizados se distinguen de los gráficos vectoriales en que estos últimos representan una imagen a través del uso de objetos geométricos como curvas y polígonos, no del simple almacenamiento del color de cada pixel.

Fig. 3. Gráfico Ráster



Fuente: http://es.wikipedia.org/wiki/R%C3%A1ster

1.2 Aplicaciones Geográficas en la Web.

1.2.1 Internet y WWW

Aunque popularmente es un poco complicado tener una definición de Internet, el concepto es muy fácil, es una colección de redes, una red de redes, computadoras compartiendo información digital vía un común grupo de interconexiones y protocolos de comunicaciones.

Podemos encontrar definiciones e historia en http://es.wikipedia.org/wiki/Internet

En la actualidad casi cualquier persona puede conectar su computador a Internet e inmediatamente comunicarse con otras computadoras y usuarios en la red; Internet conecta dos tipos de computadores: servidores, quienes sirven los documentos y clientes quienes reciben y muestran documentos.
La mayoría de información existente en el mundo está ahora disponible a través de Internet, mucha de la cual está relacionada con los SIG.

Probablemente el más fascinante desarrollo de las tecnologías de la información es el avance del Internet y las tecnologías que World Wide Web (WWW) implican, en este documento no entraremos en detalle sobre la Internet y WWW. En lugar trataremos de enfocar el impacto de estas tecnologías en el futuro del desarrollo de aplicaciones GIS.

Así podemos identificar dos tendencias que promueven la difusión de aplicaciones SIG en un medio como Internet.
Primero, el rápido crecimiento de los usuarios de los SIG, requieren que los datos geográficos puedan ser distribuidos eficiente y efectivamente para un amplio número de usuarios; Segundo, la necesidad de que los SIG sean difundidos para un gran rango de aéreas, crea la necesidad de aplicaciones mas flexibles, con requerimientos mas orientados al usuario final, con interfaces mas intuitivas e interactivas.

1.2.2 SIG basados en web

El acceso a un SIG a través de Internet es un efectivo ambiente de trabajo que nos permite tener capacidades multi-usuario, multi-plataforma, puede mantener requerimientos para la mayoría de usuarios a los que la aplicación ha sido dirigida, las principales ventajas de un SIG enfocado a Internet con respecto a programas SIG de escritorio son:

- La considerable reducción de costos en licencias de software.
- Reducir le necesidad de profesionales para instalación, soporte y mantenimiento de software especializado.
- Reduce la curva de aprendizaje del usuario.
- Disminuye la complejidad que imponen los paquetes de software de Información Geográfica.

El WWW es un excelente medio para generar aplicaciones cliente/servidor gracias a tres factores que son muy importantes:

- Flexibilidad: Se puede construir aplicaciones y distribuirlas a los usuarios de manera simple

- Mantención: Actualizaciones a los clientes son controladas y distribuidas de manera simple y centralizada.

- Accesibilidad: Multiplataforma, se puede ingresar ala aplicación por medio de un web browser, sin importar que tipo de Sistema Operativo tenga el usuario.

El auge de Internet y específicamente el servicio WWW (World Wide Web) ha creado una gran expectativa para el acceso a la información geográfica sobre Web a través de navegadores comunes ("browsers"). Los visores de mapas sobre Web incluyen tanto la presentación de mapas de propósito general como herramientas sofisticadas interactivas y personalizables. Su propósito es permitir la recuperación de información espacial rápida y fácilmente por muchos usuarios, requiriendo mínimas herramientas de lectura de mapas, como por ejemplo, análisis visuales rudimentarios, sin la alta complejidad que imponen por lo general los paquetes de software de Sistemas de Información Geográfica.

1.2.3 SIG público

(STRAND, *Eric J*) A medida que la tecnología SIG se adapte a Internet, ofrecerá más servicios dentro de la toma colaborativa de decisiones espaciales (CSDM, por Collaborative Spatial Decision Making) Las CSDM se pueden utilizar como ayuda para idear políticas de toma de decisiones públicas que involucren cuestiones espaciales que afecten a las comunidades y a la sociedad, incluyendo la zonificación para el uso de tierras, terrenos para la construcción, protección ambiental y administración de recursos naturales.

Cuando el SIG se emplea democráticamente para ayudar a los ciudadanos a entender las propuestas y sus consecuencias espaciales, evaluar alternativas y seleccionar soluciones apropiadas, se lo denomina ″SIG público″. A medida que el SIG evoluciona con el Web, la posibilidad del SIG público se hace cada vez más real. El Web y el SIG público les darán más poder a los ciudadanos, y aumentarán la eficiencia social y económica para las comunidades y la sociedad en general.

1.3 Alternativas para publicación de información geográfica en Internet

Es importante definir las opciones con las que contamos para el desarrollo de una aplicación SIG basada en web, ya que la arquitectura que se vaya a usar depende de las características del proyecto en el que se va a aplicar, teniendo en cuenta el formato (ráster/vector), cantidad, calidad de la información que deseamos publicar.

La distribución y visualización de información espacial en la web se puede realizar mediante algunos formatos (ver Tabla 1), aunque se trata de estandarizar el uso de GML (Geographic Markup Lenguage) que esta de acuerdo a las especificaciones de OpenGIS Consortium, como el lenguaje estándar para la visualización de información geográfica vectorial por medio del web.

Taba 1: Formatos para visualización de Información Geográfica

• SVF (Simple Vector Format)
• DWF (Drawing Web Format)
• SWF (ShockWave Flash)
• PGML (Precision Graphics Markup Language)
• WebCGM
• VML (Vector Markup Language)
• SVG (Scalable Vector Graphics)
• VRML (Virtual Reality Modeling Language)
• HGML (Hyper Graphics Markup Language)
• DrawML
• GML (Geography Markup Language)

Fuente: "http://es.wikipedia.org/wiki/Web_Mapping"

Desde el punto de vista de la arquitectura, un mapa para la Web esta formado por dos componentes básicos: cliente y servidor.
El Servidor es responsable de proveer la información geográfica de acuerdo a la petición que realice el cliente, el cliente realiza la petición (request) y el servidor le regresa la información representada en algún formato (GML, SVG, JPG, PNG, etc.) para que sea mostrado en el lado del cliente.

El Cliente provee al usuario una interfaz grafica, su principal tarea es receptar las entradas del usuario y comunicarse con el servidor a través del protocolo http.
La funcionalidad y características del cliente dependen de: tipo de aplicación, perfil del usuario al que es dirigido, y la opción tecnológica que se va a usar.
El desarrollo de las tecnologías web permite ir poniendo cada vez más interactividad y procesamiento al lado del cliente**, l**a interfaz para el usuario se puede desarrollar en Applets de Java, DHTML, HTML, ActiveX, etc.

Tabla 2: Comparación de técnicas para implementar el cliente

	DHTML;HTML, JavaScript	**Flash,Adobe Flex, SVG, ActiveX**	**Java Applet**
Requiere Plug-in	No	Si	Si
Compatibilidad con Navegadores	Problemático con versiones anteriores de browser	100% si el plug-in esta presente	100% si el plug-in esta presente
Velocidad para cargar interfaz	Alta	Media	Baja
Bajar contenido de diferentes Servidores	Posible	Imposible	Imposible
Facilidad de depurar y mantener	Difícil	Medio	Alto
Interfaz Interactiva	Alta	Alta	Alta

Fuente: Autor

Existe un gran numero de soluciones comerciales y gratuitas para realizar aplicaciones SIG basadas en web, se diferencian por el o los formatos que soportan, la velocidad para cargar y mostrar los datos, las capacidades de navegación y consulta que puedan ofrecer al usuario final, pero principalmente difieren en la arquitectura en que se basan las diferentes aplicaciones.

1.3.1 Arquitecturas para desarrollar un SIG en Internet.

Podemos clasificarlas en mapping estático, mapping estático sensitivo, mapping dinámico, cabe mencionar que los términos dinámico o estático hacen referencia a la interacción que se realiza con el servidor para la visualización de los datos geográficos.

1.3.1.1 Web Mapping estático.

La forma mas simple de presentar información espacial en un sitio web en Internet es incluir una imagen (GIF, PNG, etc.) dentro de un archivo HTML usando la etiqueta "img" (e.j. <IMG SRC="mapa.gif">); esta imagen puede ser generada desde algún software SIG, o simplemente capturada desde algún dispositivo de hardware como un escáner.

En este tipo de aplicaciones no se obtiene ninguna funcionalidad, sino solamente nos permite la visualización de la imagen que representa un mapa.

1.3.1.2 Web mapping estático sensitivo en **HTML.- (**Serra del Pozo, Paúl 2002**)** Estos mapas consultables desde un "browser" o navegador de internet, consisten en páginas simples en formato HTML y de imágenes GIF o JPG dentro de una etiqueta MAP (zonas de una imagen con "hiperenlace"). Estas páginas peden ser creadas por SIG profesionales como MapInfo o ArcView, o simple desarrollo de paginas HTML.

Esta etiqueta nos permite definir áreas dentro de la imagen, donde se puede mostrar un texto definido por la propiedad ALT, cuando el puntero del ratón es movido dentro de estas áreas; También se puede realizar un vinculo a alguna URL que puede ser definida en la propiedad HREF cuando el se realice un clic en el algún lugar de la imagen.

Estas páginas web no acceden a ficheros de información geográfica, como los servidores de mapas, sino que leen ficheros HTML. Constituyen una solución estática, pese a que permite al usuario una cierta interactividad: herramientas de visualización, un tipo de consulta alfanumérica simple y alguna respuesta al hacer clic en los elementos cartográficos, como obtener sus datos correspondientes, una fotografía o un enlace de internet, etc. Estos tipos de interacciones los puede definir el creador de la página web.

1.3.1.3 Web mapping estáticos sensitivos basados XML.- se trata de aplicaciones en las que si tenemos interacción con el mapa que se visualiza en el navegador, aunque el principio es que se trata de información estática ya que es descargada completamente desde el servidor y visualizada en el navegador cuando toda la información está disponible en el cliente (browser), generalmente es necesario un plugin en el navegador, que permita visualizar e interactuar con la información

No consulta la información geográfica en formato nativo, sino que accede a ficheros que proceden de la cartografía en formato nativo y que son actualizados con la periodicidad que disponga el administrador de los datos.

(Serra del Pozo, Paúl 2002**)** La ventaja de este sistema es que la cartografía que el usuario solicita tiene un carácter más dinámico que las imágenes JPG o GIF de las soluciones HTML o de los servidores de mapas (alguno de los cuales, por otro lado, también pueden enviar al cliente información en formato XML): el usuario puede realizar determinadas funciones “en local”, ya que dispone del fichero en su memoria virtual. Por ejemplo, al hacer un “clic" sobre cualquier código de parcela, el “browser” ofrece sus datos correspondientes y aparece la parcela en tamaño aumentado, con un color de selección. En definitiva, ciertas funciones como los “zooms” o activación y desactivación de capas se ejecutan sólo en el ordenador local, sin necesidad de una nueva petición al servidor, con el consecuente ahorro de recursos del servidor y tráfico a través de internet. También permite la selección de ventanas de zoom especificando las coordenadas mínimas y máximas de la zona a ampliar.

Como ejemplos tenemos:

Aplicaciones Flash o Flex que tienen su propio formato para representar información vectorial y se puede desarrollar herramientas que nos proporcionen algún tipo de funcionalidad mediante programación en ActionScript.

MapViewSVG.- es una extensión para el software ArcGis de ESRI, esta herramienta convierte mapas de ArcView/ArcGis en formato SVG, y es visualizada en el navegador mediante un plug-in (Corel, Adobe SVG Viewer), mediante esta aplicación se puede tener una cierta interacción con alguna base de datos in embargo esta es solamente para recuperar datos de atributos mas no información geográfica.

1.3.1.4 Mapping Dinámico:

Esta es la opción que permite al usuario la máxima interacción con la información geográfica. El usuario tiene acceso a los recursos de la Web, se desplaza libremente por toda la información con herramientas funcionales, cambia la representación gráfica en línea, enlaza elementos gráficos con informaciones procedentes de bases de datos, y trabaja en tiempo real con funciones de análisis

Estas soluciones están generalmente basadas en una arquitectura que consta de tres capas: cliente, servidor de mapas y manejador de base de datos.

Componentes de la arquitectura

Cliente: es ejecutado en navegadores, estos manejan la comunicación con el sitio web usando el protocolo http,
Cualquier navegador que soporte el estándar HTML puede actuar como cliente. Será necesario que soporte Applet (Plug-in) de Java, tecnología ActiveX, Flash, o SVG si los servicios a los que se accede requieren estos componentes.

Al seleccionar alguna alternativa para implementar al lado del cliente debemos tener en cuenta algunos aspectos, como por ejemplo el formato de la información; el uso de formato vectorial es mucho mas rápido comparado con el formato ráster, y se puede transportar mas eficientemente por medio de la red, pero para soportar algún tipo de formato vectorial es necesario una aplicación que se ejecute dentro del web browser, estas aplicaciones son comúnmente llamadas plug-ins.

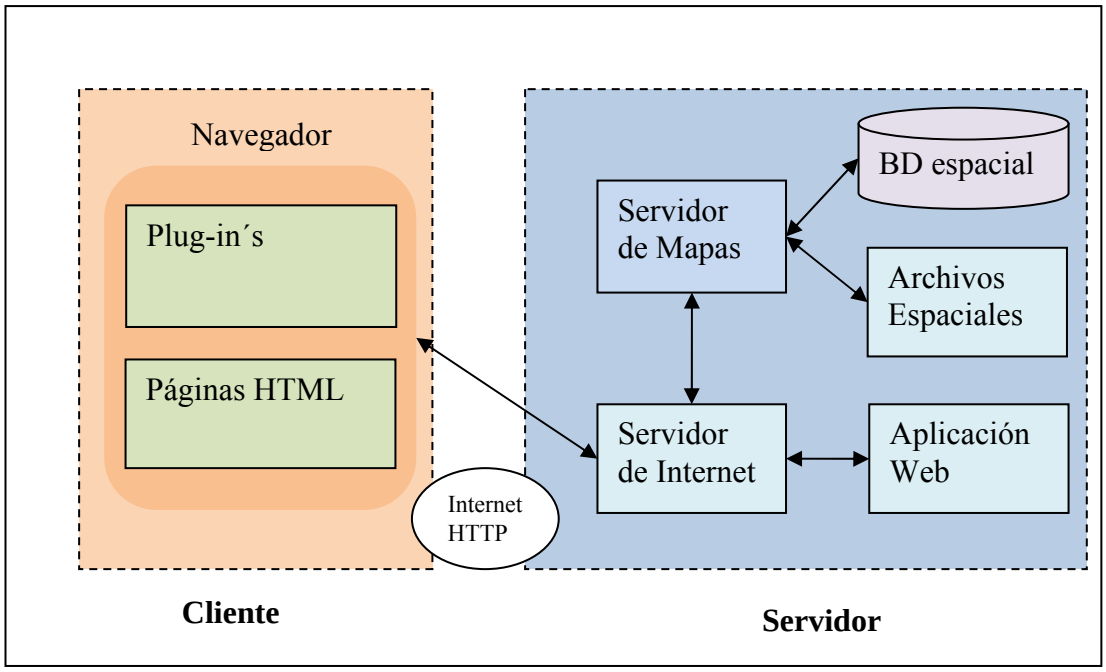
Servidor de Mapas: un servidor de mapas es usado para manejar las peticiones del cliente, generalmente el servidor de mapas está enlazado con el servidor de Internet que es el que en realidad maneja las peticiones http (http request) que son generadas por el navegador del cliente, el servidor de mapas es independiente de la implementación que se realice en el lado del cliente.

Existen algunas alternativas para implementar u servidor de mapas: ArcView IMS, MapObjects IMS, ArcIMS, Map&Guide, Geomedia Web, UMN MapServer, OGC Mapview, Geoclip, GeoServer, etc.

Manejador de Bases de Datos:

El ultimo componente en la arquitectura es un Manejador de bases de datos (DBMS por sus siglas en inglés), en la mayoría de los casos se usa una sistema de base de datos que soporte datos espaciales como PostGres/PostGis, Oracle Stpatial, etc., ya que agregan funcionalidad para manejar datos espaciales.

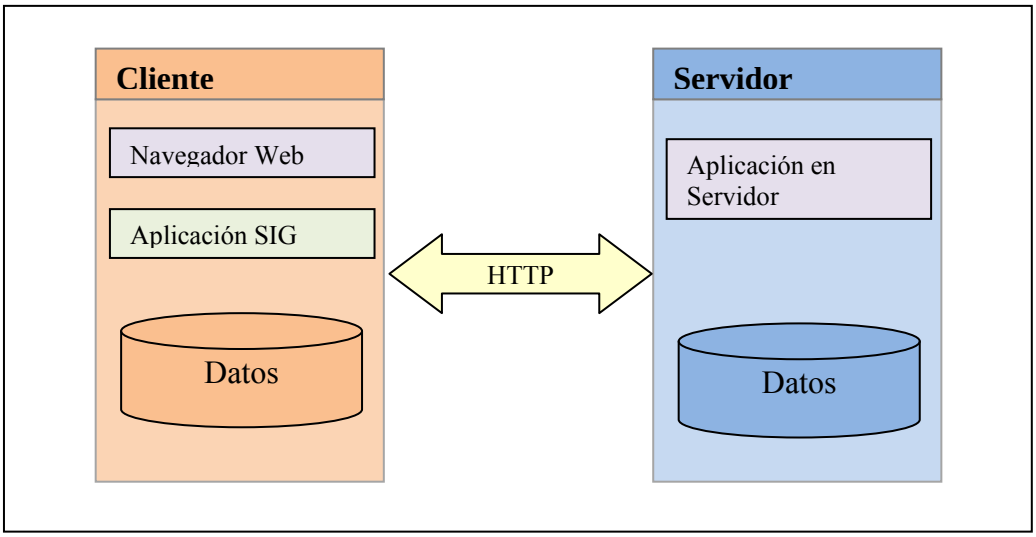
Fig. 4: Diagrama de la Arquitectura Mapping Dinámico



Fuente: Autor

A este tipo de arquitecturas las podemos dividir en dos tipos, client-side y Server-side, en una aplicación enfocada al cliente (client-side) el cliente (Web browser) es el que tiene el soporte para realizar las funcionalidades SIG que la aplicación necesite, mientras que en una aplicación enfocada al lado del servidor (Server-side) el web browser es solamente utilizado para generar la interacción con el servidor para generar los resultados, que luego son visualizados en el cliente.

Fig. 5: Arquitectura enfocada al cliente

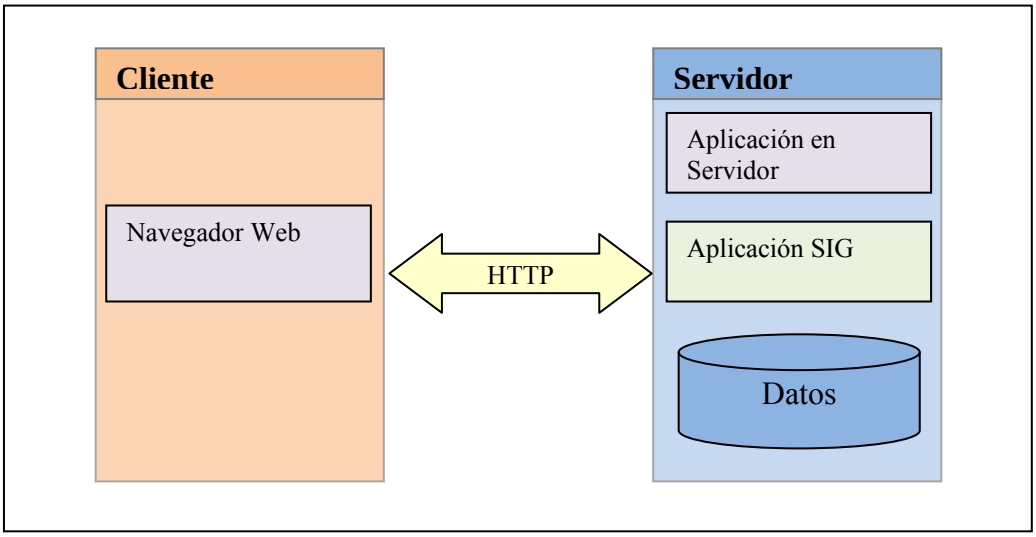


Fuente: Autor

Un ejemplo de una aplicación enfocada al servidor, sería: un usuario envía una petición al servidor, como por ejemplo una dirección, en el servidor se procesa la respuesta y es enviada hacia el cliente en forma de una imagen en un documento HTML, la respuesta que genera el servidor puede ser visualizada por cualquier navegador sin necesidad de instalar ningún componente adicional.

Esta arquitectura tiene muchas ventajas ya que los datos y la aplicación están centralizados en el servidor y esto significa facilidad en el desarrollo, publicación y mantenimiento de la aplicación.

Fig. 6: Arquitectura enfocada al servidor.



Fuente: Autor

Una comparación de las ventajas y desventajas de estas dos arquitecturas es mostrada en la siguiente tabla.

Tabla 3 Figura: Arquitectura enfocada al servidor.

Ventajas Server-Side	**Desventajas Server-Side**
Fácil Publicación	Interfaz grafica de usuario(GUI) limitada
Fácil Mantenimiento	Solamente trabaja con formato ráster en cliente
Mantiene Estándares	
Funciona en navegadores estándar	
Ventajas Client-Side	**Desventajas Client-Side**
Se puede usar formatos vectoriales	Difícil desarrollo
Mejor calidad de Imagen	Requiere Software Adicional en cliente(plug-ins)

Buena Interfaz grafica de usuario	Tiempo de descarga del Plug-in
	No obedece a estándares.
	Dependiendo del Plug-in, puede dar incompatibilidad con los navegadores

Fuente: Autor

1.3.4 Comparación de las Alternativas para la publicación de SIG en la Web:

La arquitectura y tecnologías a usar dependen de los requerimientos de cada sistema, y del perfil de los usuarios a los que la aplicación está dirigida.

En la tabla Tabla 4 podemos ver un resumen de las características de las alternativas que tenemos para publicar información geográfica por medio del Internet,

Tabla 4: Alternativas para publicar información geográfica en la Internet

	Mapping Estático	**Mapping Estático Sensitivo HTML**	**Mapping Estático Sensitivo XML**	**Mapping Dinámico**
Perfil del Cliente o Usuario Final	No Experto	No Experto	No Experto	No Experto
Medios Necesarios para la Implementación	Servidor Web, Ficheros de Imágenes	Servidor Web, modulo para crear los archivos HTML	Servidor Web, modulo para crear archivos XML, PlugIn en Cliente	Servidor Web, servidor de mapas, servidor de datos(opcional)
Costo de Implementación en Tiempo	Muy baja	Muy Bajo, Cargar archivos al servidor	Baja, generar archivos(SVG,XML)	Alto, Implementación, y Configurar servidor de mapas
Conocimientos de Programación	Nada	Nada	Muy Bajo	Necesario
Funciones	Ninguna	Limitada, visualización y consulta simples	Visualización y Consulta Semi Avanzada	Avanzadas, visualización y consultas complejas, routing, análisis espacial
Velocidad de respuesta en Cliente	Muy Rápida	Rápida	Rápida con archivos pequeños, Lenta con archivos grandes	Menos rápida

Fuente: Autor

1.4 Estándares para Publicación de Información Geográfica

1.4.1 Open Geospatial Consortium

(WIKIPEDIA-OGC) “El Open Geospatial Consortium (OGC) fue creado en 1994 y agrupa a más de 250 organizaciones públicas y privadas. Su fin es la definición de estándares abiertos e interoperables dentro de los Sistemas de Información Geográfica. Persigue acuerdos entre las diferentes empresas del sector que posibiliten la interoperación de sus sistemas de geoprocesamiento y facilitar el intercambio de la información geográfica en beneficio de los usuarios. Anteriormente fue conocido como Open GIS Consortium”.

1.4.2 Especificaciones

Las especificaciones más importantes surgidas del OGC son:

GML, acrónimo inglés de *Geography Markup Language* (*Lenguaje de Marcado Geográfico*.). Es un sublenguaje de XML descrito como una gramática en XML Schema para el modelaje, transporte y almacenamiento de información geográfica. Su importancia radica en que a nivel informático se constituye como una lengua franca para el manejo y trasvase de información entre los diferentes software que hacen uso de este tipo de datos, como los Sistema de Información Geográfica.

GML se diseñó a partir de la especificación abstracta producida por el grupo OpenGIS, ahora Open Geospatial Consortium, y de la serie de documentos ISO 19100. GML no contiene información específica sobre como se debe hacer la visualización de los datos representados. Para ello se utilizan estilos que se relacionan a GML y se describen en otros sublenguajes de XML. Otras extensiones manejadas por GML incluyen SMIL para definir elementos de interacción y XPointer para representar metadatos.

Web Map Service (WMS) produce mapas de datos espaciales referidos de forma dinámica a partir de información geográfica. Este estándar internacional define un "mapa" como una representación de la información geográfica en forma de un archivo de imagen digital conveniente para la exhibición en una pantalla de ordenador. Un mapa no consiste en los propios datos. Los mapas producidos por WMS se generan normalmente en un formato de imagen como PNG, GIF o JPEG, y

ocasionalmente como gráficos vectoriales en formato SVG (Scalable Vector Graphics) o WebCGM (Web Computer Graphics Metafile).

El estándar define tres operaciones:

1. Devolver metadatos del nivel de servicio.
2. Devolver un mapa cuyos parámetros geográficos y dimensionales han sido bien definidos.
3. Devolver información de características particulares mostradas en el mapa (opcionales).

Las operaciones WMS pueden ser invocadas usando un navegador estándar realizando peticiones en la forma de URLs (Uniform Resource Locators). El contenido de tales URLs depende de la operación solicitada. Concretamente, al solicitar un mapa, la URL indica qué información debe ser mostrada en el mapa, qué porción de la tierra debe dibujar, el sistema de coordenadas de referencia, y la anchura y la altura de la imagen de salida. Cuando dos o más mapas se producen con los mismos parámetros geográficos y tamaño de salida, los resultados se pueden solapar para producir un mapa compuesto. El uso de formatos de imagen que soportan fondos transparentes (e.g., GIF o PNG) permite que los mapas subyacentes sean visibles. Además, se puede solicitar mapas individuales de diversos servidores.

El servicio WMS permite así la creación de una red de servidores distribuidos de mapas, a partir de los cuales los clientes pueden construir mapas a medida. Las operaciones WMS también pueden ser invocadas usando clientes avanzados SIG, realizando igualmente peticiones en la forma de URLs. Existe software libre, como la aplicación gvSIG, que permite este acceso avanzado a la información remota, añadiendo la ventaja de poder cruzarla con información local y disponer de una gran variedad de herramientas SIG.

Web Feature Service o **WFS** es un servicio estandar, que ofrece un interfaz de comunicación que permite interactuar con los mapas servidos por el estandar WMS, como por ejemplo, editar la imagen que nos ofrece el servicio WMS o analizar la imagen siguiendo criterios geográficos.

Para realizar estas operaciones se utiliza el lenguaje GML que deriva del XML, que es el estandar a traves del que se transmiten la ordenes **WFS**.

Web Coverage Service es un estándar para intercambiar datos geoespaciales, Coverage hace referencia a datos como fotografías aéreas, mapa de elevaciones, etc. WCS permite a los usuarios acceder remotamente a datos de cobertura.

WCS permite obtener archivos de cobertura junto con su información detallada, permite consultas complejas a través de estos datos

1.5 Conclusiones

Con el avance de las tecnologías de la información, y la popularidad que han tomado en los últimos años portales web con información espacial, han dado lugar a un gran numero de alternativas para la publicación de información geográfica por medio de Internet, apoyadas en diferentes tecnologías, que permiten implementar aplicaciones con cierta funcionalidad de análisis y visualización de información que se encuentra georeferenciada, permitiendo así la distribución a un gran numero de usuarios, que pueden o no tener conocimientos de Sistemas de Información Geográfica.

Gracias a la implementación de Servidores de Mapas es posible realizar aplicaciones que accedan dinámicamente a la información geográfica y permiten realizar análisis espacial que toma un gran potencial cuando usamos a los servidores de mapas integrados con bases de datos espaciales.

Se puede predecir que el desarrollo de los Sistemas de Geoinformación Geográfica basados en web van a tener un gran avance en los próximos años, creando nuevas alternativas y mejorando las capacidades de análisis, navegación y consultas, acercándose cada vez más a las capacidades que las aplicaciones SIG de escritorio nos pueden brindar. Esto poyado con la creación de estándares para la publicación de información espacial crea la posibilidad de realizar aplicaciones distribuidas que accedan a diferentes servidores y fuentes de datos creando así lo que se llaman

Infraestructura de Datos Espaciales dando lugar al surgimiento de una nueva generación de Sistemas de Información Geográfica.

CAPITULO 2

RECOLECCIÓN Y LEVANTAMIENTO DE INFORMACIÓN

Introducción

La primera labor que debemos hacer es identificar que capas de información necesitamos para le sistema, revisando diferentes formatos y fuentes de información que tenemos disponible, integrando datos provenientes de diferentes organizaciones, instituciones y trabajos de investigación que ya han sido realizados.

2.1 Recolección de Información

2.1.1 Base Cartográfica

La base cartográfica de este proyecto consiste en los archivos o bases de datos que representan geográficamente características de la ciudad de Cuenca, estos archivos los hemos conseguido del Instituto de Estudios de Régimen Seccional del Ecuador (IERSE), se encuentran en el Sistema de Coordenadas Geográficas PSAD56/UTM Zona 17 por lo tanto es sistema que se usará para todo el proyecto

Los archivos son:

Tabla 5 Archivos Recolectados

Cartografía	Formato	Tipo	Fuente *	Observaciones	
Manzanas.shp	shape	polígono	IERSE	PSAD56	Manzanas de la ciudad
Predios.shp	shape	polígono	IERSE	PSAD56	Predios de la ciudad
Vialidad.shp	shape	línea	IERSE	PSAD56	Av., calles, retornos, etc.
Rios.shp	shape	línea	IERSE	PSAD56	Rios principales, secundarios, quebradas
Alimenticios_Cuenca_1k_UTM_SAM56.shp	shape	punto	LOPEZ – MARTINEZ	PSAD56	Lugares alimenticios en la ciudad

Alojamiento_Cuenca_1k_UTM_SAM56.shp	shape	línea	LOPEZ – MARTINEZ	PSAD56	Lugares de alojamiento
Apoyo_Turista_Cuenca_1k_UTM_SAM56.shp	shape	línea	LOPEZ – MARTINEZ	PSAD56	Información y apoyo al turista
Cultura_Cuenca_1k_UTM_SAM56.shp	shape	punto	LOPEZ – MARTINEZ	PSAD56	Sitios culturales en la ciudad
Esparcimiento_Aire_Libre_Cuenca_1k_UTM_SAM56.shp	shape	punto	LOPEZ – MARTINEZ	PSAD56	Lugares al aire libre
Sitios_Recreacion_Cuenca_1k_UTM_SAM56.shp	shape	punto	LOPEZ – MARTINEZ	PSAD56	Sitios de recreación

Fuente: Autor

*Fuentes:
IERSE Instituto de Estudios de Régimen Seccional del Ecuador Cuenca-Ecuador

LOPEZ Fernanda – MARTINEZ Geovany, 2007, Cuenca-Ecuador

Manzanas

Nombre del Archivo: manzanas.shp

Tabla 6: Tabla de Atributos de manzanas.shp

CAMPO	DESCRIPCION	TIPO	LARGO
FID	Identificador del objeto	Object ID	4
SHAPE	Geometría del objeto	Polygon	
AREA	Superficie del Polígono	Double	18,7
PERIMETER	Perímetro del polígono	Double	18,7
ZONA	Código de Zona	String	3
SECTOR	Código de Sector	String	3
MANZANA	Código de Manzana	String	4
CLAVECATAS	Código Catastral	Double	10,1

Fuente: Autor

Predios

Nombre del Archivo: predios.shp

Tabla 7: Tabla de Atributos de: predios.shp

CAMPO	DESCRIPCION	TIPO	LARGO
FID	Identificador del objeto	Object ID	4
Shape	Geometría del objeto	Polygon	
AREA	Superficie del Polígono	Double	18,7
PERIMETER	Perímetro del polígono	Double	18,7
ZONA	Código de Zona	String	3
SECTOR	Código de Sector	String	3
MANZANA	Código de Manzana	String	4
PREDIO	Código del predio	String	4

COD_PREDIO	Código Unificado	Number	16

Fuente: Autor

Vialidad

Nombre del Archivo: vialidad.shp

Tabla 8: Tabla de Atributos de vialidad.shp

CAMPO	DESCRIPCION	TIPO	LARGO
FID	Identificador del objeto	Object ID	4
Shape	Geometría del objeto	Polygon	
Parroquia	Parroquia en donde se ubica la calle	Text	25
TEXT	Nombre de la Calle	Text	100

Fuente: Autor

Ríos

Nombre del Archivo: Rios_Principales_Azuay_50k_UTM_SAM56.shp

Tabla 9: Tabla de Atributos de: Rios_Principales_Azuay_50k_UTM_SAM56.shp

CAMPO	DESCRIPCION	TIPO	LARGO
FID	Identificador del objeto	Object ID	4
Shape	Geometría del objeto	Polygon	
Nombre	Nombre del río	Text	100

Fuente: Autor

2.1.2 Base Alfanumérica

La información alfanumérica es la que obtenemos de Bases de Datos, archivos de texto, hojas de cálculo, etc., en las cuáles tenemos una o más columnas que se relacionan con la información cartográfica que utilizaremos.

Predios.dbf

Esta tabla contiene toda la información de 1651 predios dentro de la ciudad, a continuación el listado de los principales campos contenidos en este archivo:

Tabla 10: Tabla de Atributos de: Predios.dbf

CAMPO	DESCRIPCION	TIPO	LARGO
A_ZONA	Código de Zona	Texto	20
A_SECTOR	Código de Sector	Texto	20
A_MANZANA	Código Manzana	Texto	20
A_PREDIO	Código Predio	Texto	20
CALLE	Código Calle Principal	Texto	5
ENTRE	Código Calle Intersección	Texto	5
Y	Código Calle Intersección	Texto	5
NUMERO	Numero del Predio	Texto	6
PARROQUIA	Código Parroquia	Texto	6
APELLIDOS	Apellidos Propietario	Texto	50
NOMBRES	Nombres Propietario	Texto	50
CEDULA	Numero Cedula Propietario	Texto	13
TELEFONO	Teléfono predio	Texto	20

Fuente: Autor

2.2 Conclusión

Ver, localizar y consultar datos es el objetivo de la mayoría de aplicaciones SIG, para esto debemos contar con la información y datos que satisfagan las necesidades del usuario final, por esta razón se realizó la recopilación de datos en trabajos y proyectos ya realizados obteniendo así los archivos de vías, manzanas, ríos, predios y de información turística, que van a ser clave para la visualización del mapa y para las búsquedas que se van a implementar en las etapas posteriores del proyecto.

Integrar la información es necesario ya que permite tener mayor potencialidad en las consultas, y aumenta características que podemos mostrar en la aplicación.

CAPITULO 3:
ANÁLISIS

Introducción.

Se estudiará las características del proyecto, definición de necesidades del usuario para conseguir una aplicación que tenga como objetivo llegar a un gran número de usuarios, utilizando la mejor alternativa tecnológica, que nos permita realizar una aplicación con software libre o de bajo costo que cumpla con los objetivos de realizar la búsqueda de direcciones y lugares de interés en la ciudad.

3.1 Análisis de factibilidad

3.1.1 Factibilidad Operacional

En el desarrollo de la investigación preliminar se ha encontrado que no existe un portal web en donde podamos ubicar geográficamente direcciones y lugares de interés para la ciudad de Cuenca, el desarrollo de una aplicación con estas características ayudara a agilizar los procesos de búsquedas de información dentro de la ciudad, facilitando a quienes necesitan información representada geográficamente en un mapa.

3.1.2 Factibilidad Técnica

El auge que han tenido últimamente el desarrollo aplicaciones con contenido geográfico en la web ha dado lugar a que tengamos varias alternativas de todo tipo para llevar a cabo este tipo de desarrollos, basadas en diferentes arquitecturas, tipo de implementación y que van desde software libre a grandes aplicaciones comerciales que nos permiten la publicación de mapas en la Internet.

En lo que respecta al desarrollo de la interfaz de usuario también tenemos la posibilidad de elegir entre varias opciones, dependiendo de las características que queramos implementar en el cliente y el conocimiento de desarrollo de aplicaciones web que tengamos.

3.1.3 Factibilidad económica

Tomando en cuenta que para el desarrollo de la aplicación hemos seleccionado la utilización de software libre o de bajo costo, llevar a cabo el desarrollo e implementación tiene un margen de inversión económica prácticamente nula, por lo tanto no realizaremos un análisis económico del proyecto.

3.2 Análisis de Requerimientos

Analizaremos las características más importantes que deberá tener nuestro proyecto,
Los requerimientos han sido distribuidos en dos niveles, como nivel1 los requerimientos funcionales de la aplicación que serán los que nos ayuden a lograr los objetivos principales del proyecto estos son la búsqueda de direcciones y la búsqueda de lugares de interés, y de nivel2 aquellos que van a servir para la navegación en el mapa como por ejemplo actuar/desactivar capas, herramientas de navegación, y herramientas de consulta.

3.2.1 Requerimientos Funcionales

<u>Requerimientos de Nivel 1</u>

R1.- El sistema deberá realizar búsquedas por dirección: Calle Principal y calle Intersección

R2.- El sistema deberá realizar búsquedas por: calle principal y número de predio

R3.- El sistema deberá realizar búsquedas por: calle principal, numero de predio y calle intersección.

R4.- El sistema deberá realizar búsquedas de lugares de interés por el nombre del lugar.

R5- El sistema deberá realizar búsquedas de lugares de interés por palabras claves del lugar.

R6.- El sistema listará los resultados encontrados en forma de una lista.

R7.- El sistema ubicará los resultados encontrados en el mapa.

<u>Requerimientos de Nivel 2</u>

R8.- El sistema proveerá al usuario herramientas para la navegación en el mapa como son: zoom+, zoom-, zoom rectangular, vista completa, desplazamiento, mapa de referencia.

R9.- El sistema mostrará una leyenda donde el usuario podrá activar/desactivar capas de información.

R10.- El sistema podrá imprimir la vista actual del mapa en formatos de imágenes.

R11.- El sistema deberá tener una herramienta para consultar información de lugares en el mapa.

3.2.2 Requerimientos no funcionales

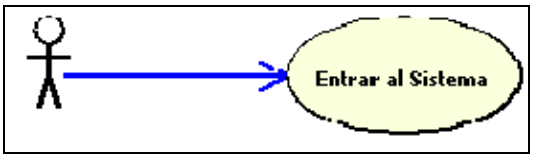
R12.- El sistema deberá acceder a archivos vectoriales y bases de datos para la visualización en el mapa.

R13.- El sistema deberá proveer una interfaz interactiva, sin recargar la página del cliente en cada petición del usuario.

3.3 Casos de Usos

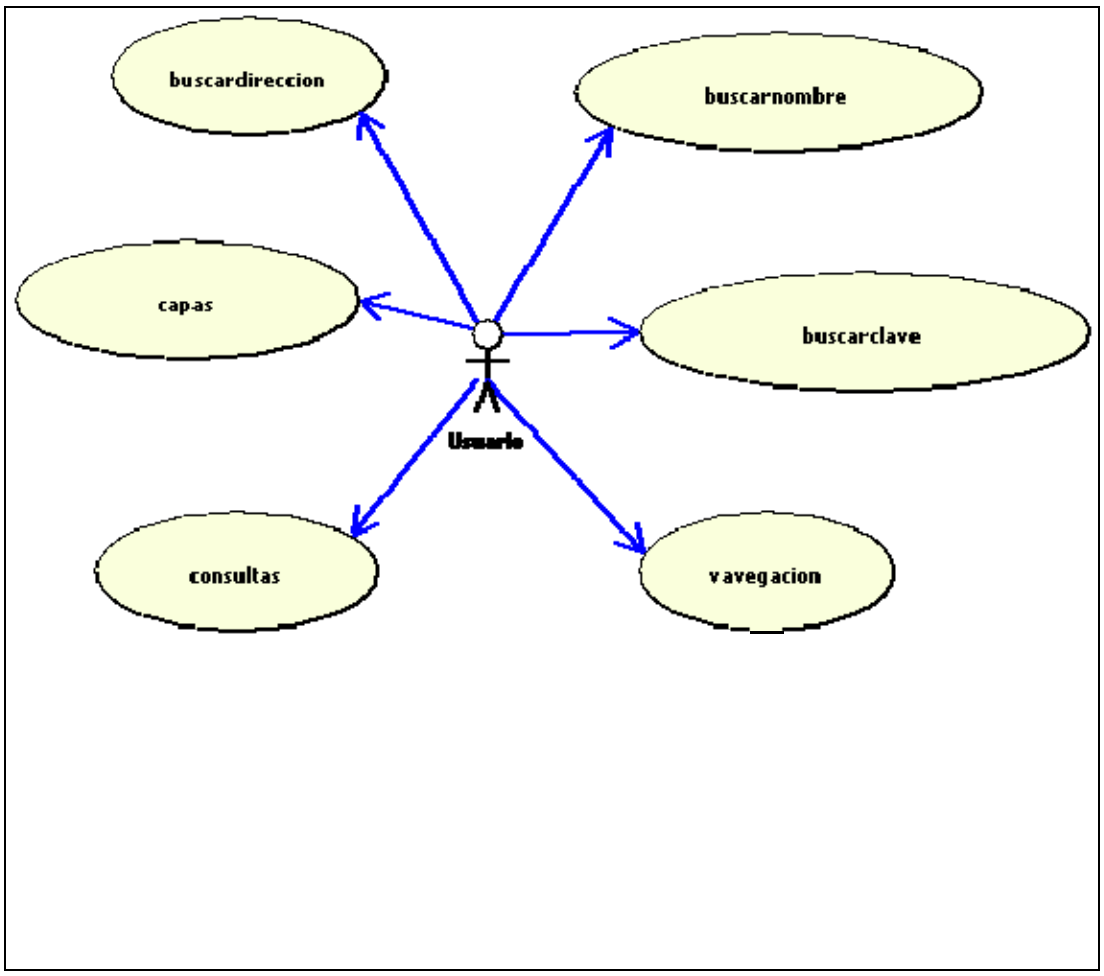
3.3.1 Diagrama de Casos de Usos

Fig. 7: Caso uso CU-1



Fuente: Autor

Fig. 8: Caso uso CU-2



Fuente: Autor

3.3.2 Descripción de Casos de Usos

Tabla 11: Descripción de Casos de Usos

Código	Nombre	Descripción
CU-01	entrar	Usuario entra al sistema desde una url.
CU-02	buscardireccion	El usuario busca una dirección.
CU-03	buscarnombre	El usuario busca un lugar pos su nombre.
CU-04	buscarclave	El usuario busca un lugar pos sus palabras claves.
CU-05	capas	El usuario activa/desactiva capas de información y se reflejan en el mapa.
CU-06	consultas	El usuario realiza consultas de lugares en el mapa.
CU-07	navegación	El usuario utiliza la herramientas de navegación para desplazarse, alejarse, acercarse en el mapa

Fuente: Autor

3.3.3 Detalle de Casos de Usos

Tabla 12: Caso de Uso CU-01

CASO DE USO	CU-01
AUTOR	usuario
PREREQUISITOS	
FLUJO DE EVENTOS	- usuario entra la url de la aplicación - servidor genera el mapa - aplicación muestra el mapa
RESULTADO	- la aplicación se carga, queda lista para usar

Fuente: Autor

Tabla 13: Caso de Uso CU-02

CASO DE USO	CU-02
AUTOR	**buscardireccion**
PREREQUISITOS	- la aplicación se encuentre cargada
FLUJO DE EVENTOS	- usuario selecciona buscar por dirección - usuario ingresa dirección - usuario presiona botón buscar
RESULTADO	- la aplicación lista los resultados - la aplicación muestra los resultados en el mapa

Fuente: Autor

Tabla 14: Caso de Uso CU-03

CASO DE USO	CU-03
AUTOR	**buscarnombre**
PREREQUISITOS	- la aplicación se encuentre cargada
FLUJO DE EVENTOS	- usuario selecciona buscar por nombre - usuario ingresa nombre a buscar - usuario presiona botón buscar
RESULTADO	- la aplicación lista los resultados - la aplicación muestra los resultados en el mapa

Fuente: Autor

Tabla 15: Caso de Uso CU-04

CASO DE USO	CU-04
AUTOR	**buscarclave**
PREREQUISITOS	- la aplicación se encuentre cargada
FLUJO DE EVENTOS	- usuario selecciona buscar por palabra clave - usuario ingresa nombre palabras clave a buscar - usuario presiona botón buscar
RESULTADO	- la aplicación lista los resultados - la aplicación muestra los resultados en el mapa

Fuente: Autor

Tabla 16: Caso de Uso CU-05

CASO DE USO	CU-05
AUTOR	**capas**
PREREQUISITOS	- la aplicación se encuentre cargada
FLUJO DE EVENTOS	- usuario activa/desactiva capas de información - usuario aclara/obscurece las capas - usuario pone al frente o atrás las capas
RESULTADO	- la aplicación muestra las capas seleccionadas.

Fuente: Autor

Tabla 17: Caso de Uso CU-06

CASO DE USO	CU-06
AUTOR	**consultas**
PREREQUISITOS	- la aplicación se encuentre cargada
FLUJO DE EVENTOS	- usuario selecciona le herramienta de consulta - usuario hace clic en un lugar en el mapa
RESULTADO	- la aplicación le muestra la información de ese lugar - se puede hacer zoom al lugar consultado - se puede sacar información detallada del lugar consultado

Fuente: Autor

Tabla 18: Caso de Uso CU-07

CASO DE USO	CU-07
AUTOR	**navegación**
PREREQUISITOS	- la aplicación se encuentre cargada
FLUJO DE EVENTOS	- usuario selecciona una herramienta de navegación (zoom+,zomm-, zoom rectangular, Vista completa, desplazamiento, mapa de referencia) - usuario selecciona un punto/rectángulo en el mapa
RESULTADO	- aplicación genera el nueva extensión del mapa - la herramienta usada queda seleccionada si es desplazamiento o zoom rectangular.

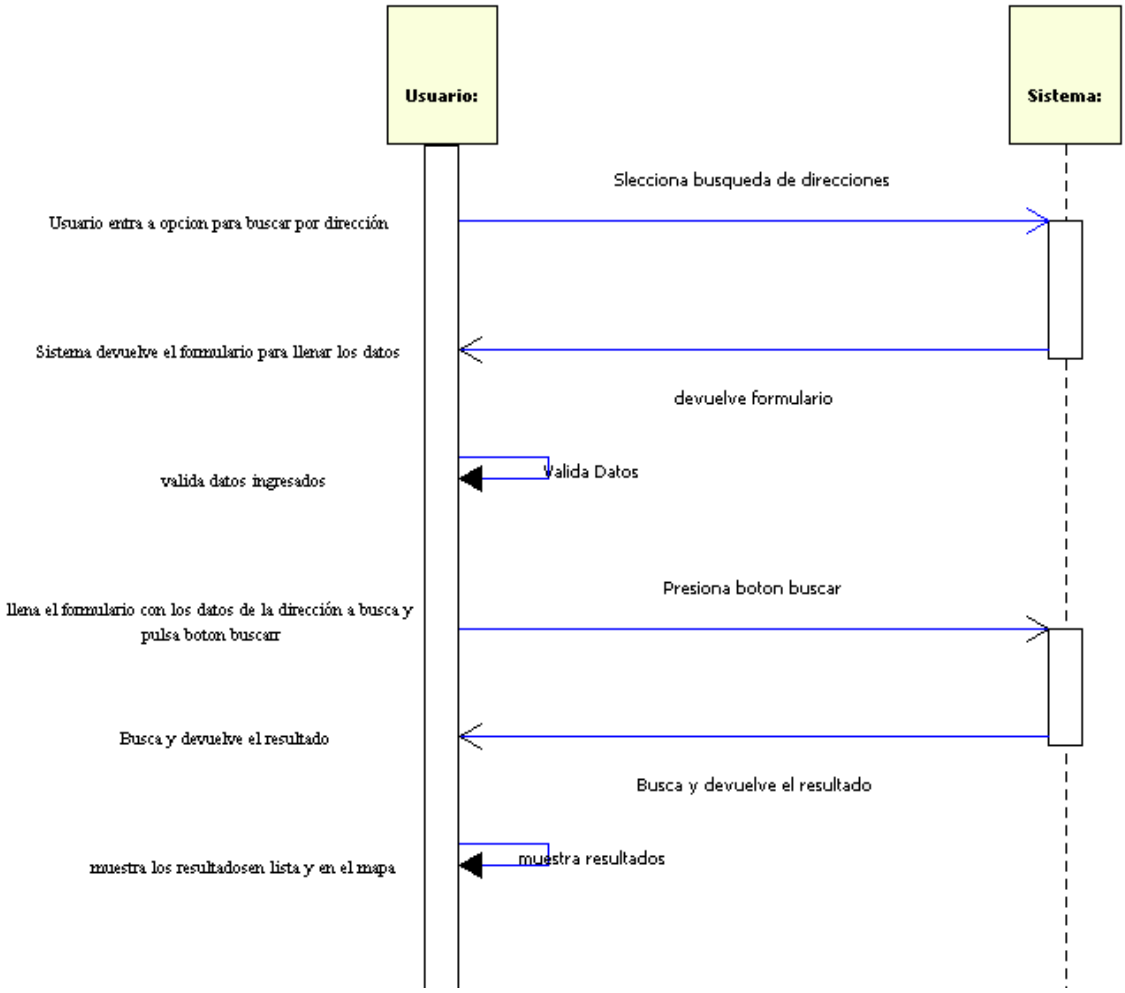
Fuente: Autor

3.4 Diagramas de Secuencia

Los Casos de usos expresados en diagramas de secuencia.

Caso de Uso CU-02 buscardireccion expresado en diagrama de secuencias

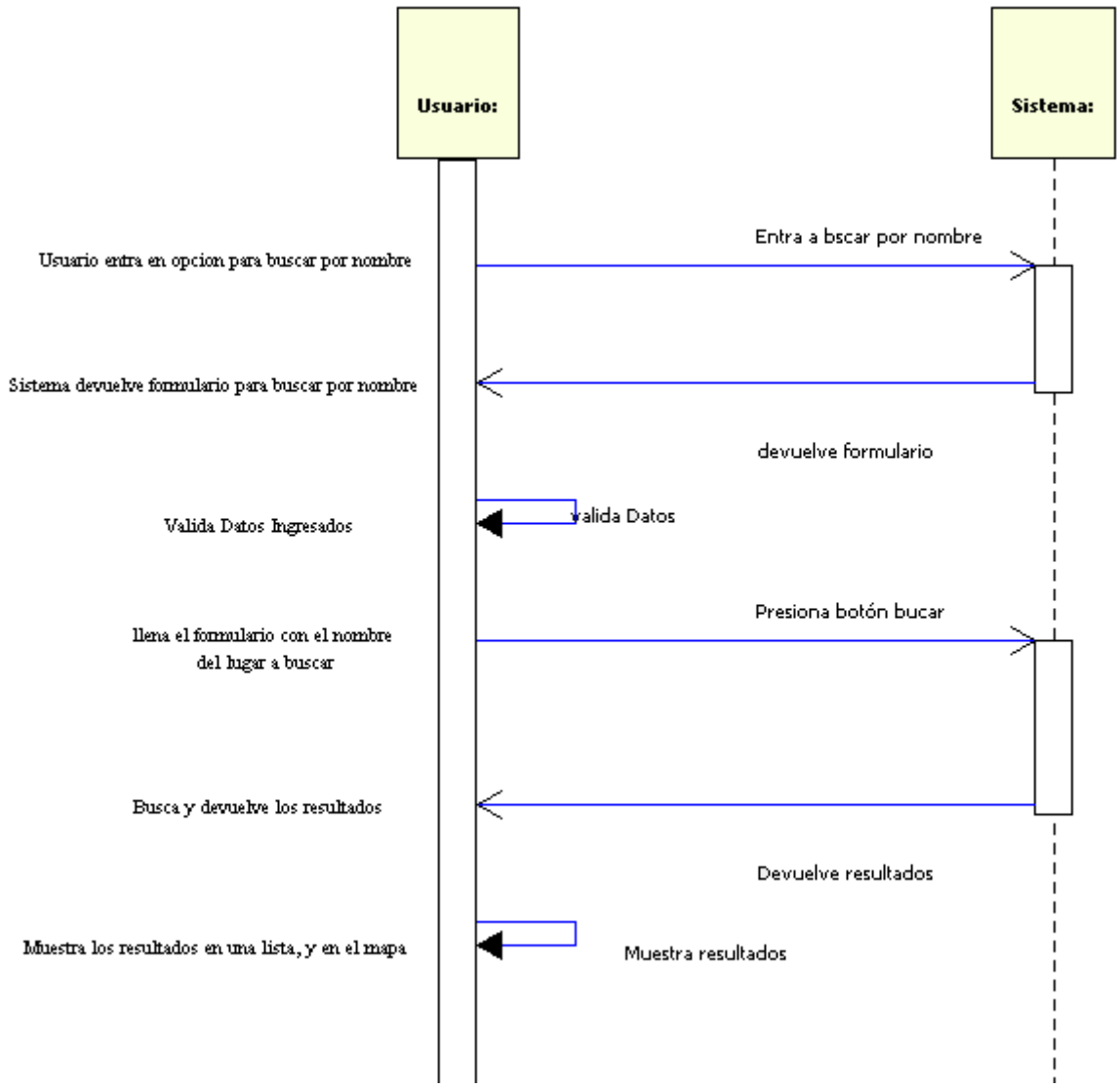
Fig. 9: Diagrama de Secuencia para CU-02



Fuente: Autor

Caso de Uso CU-03 buscarnombre expresado en diagrama de secuencias

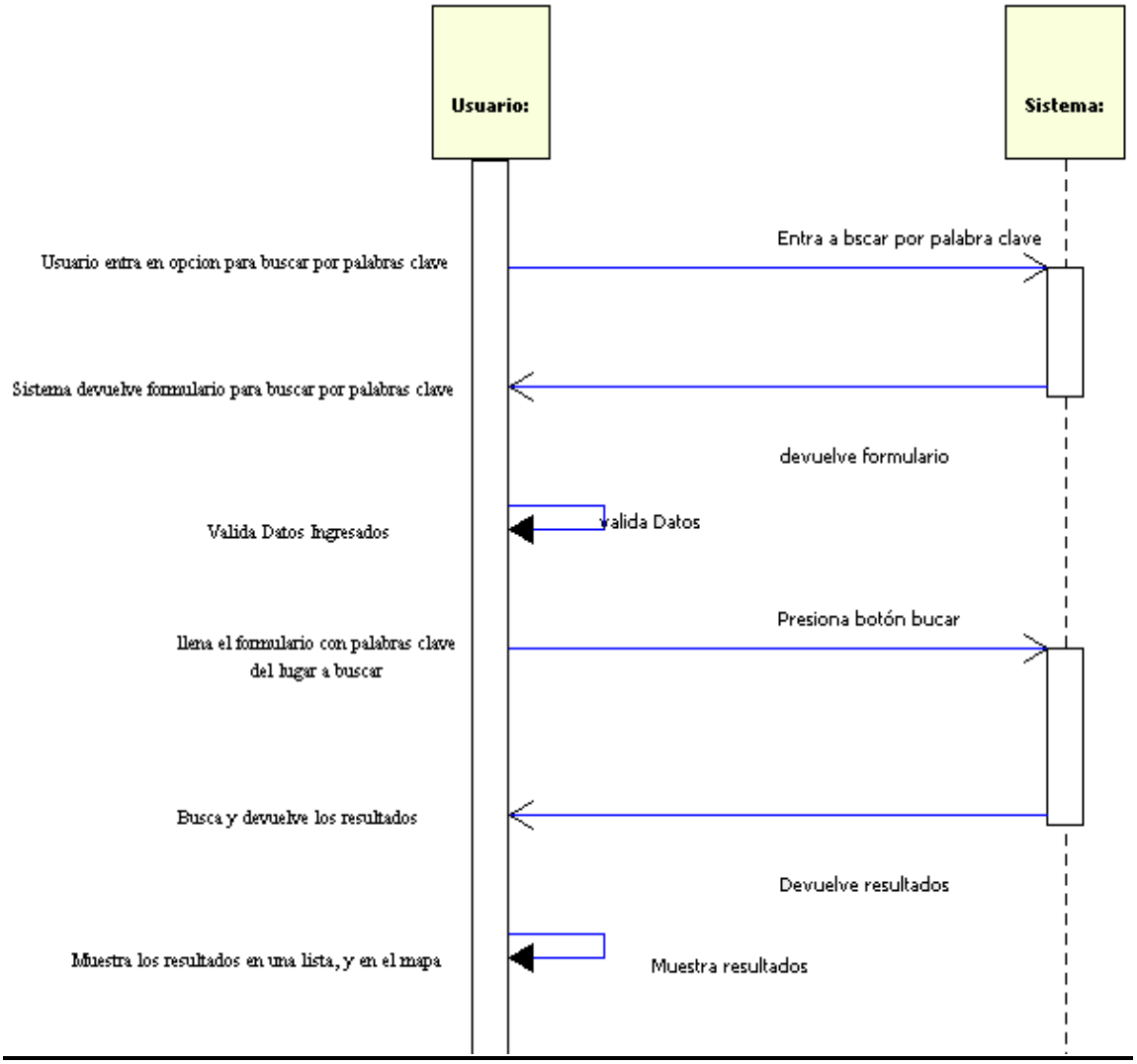
Fig. 10: Diagrama de Secuencia para CU-03



Fuente: Autor

Caso de Uso CU-04 buscarclave expresado en diagrama de secuencias

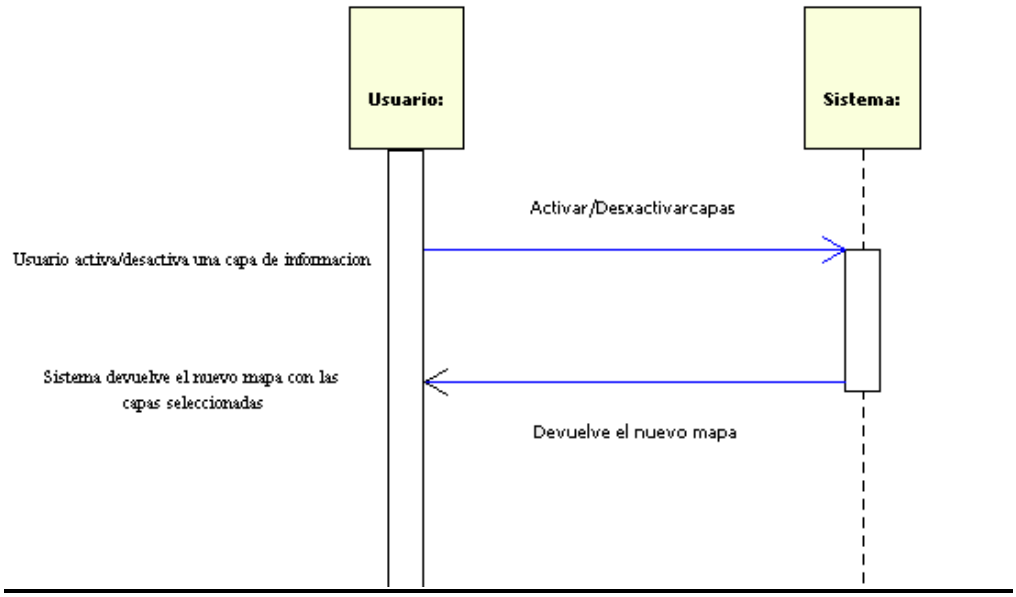
Fig. 11: Diagrama de Secuencia para CU-04



Fuente: Autor

Caso de Uso CU-05 capas expresado en diagrama de secuencias

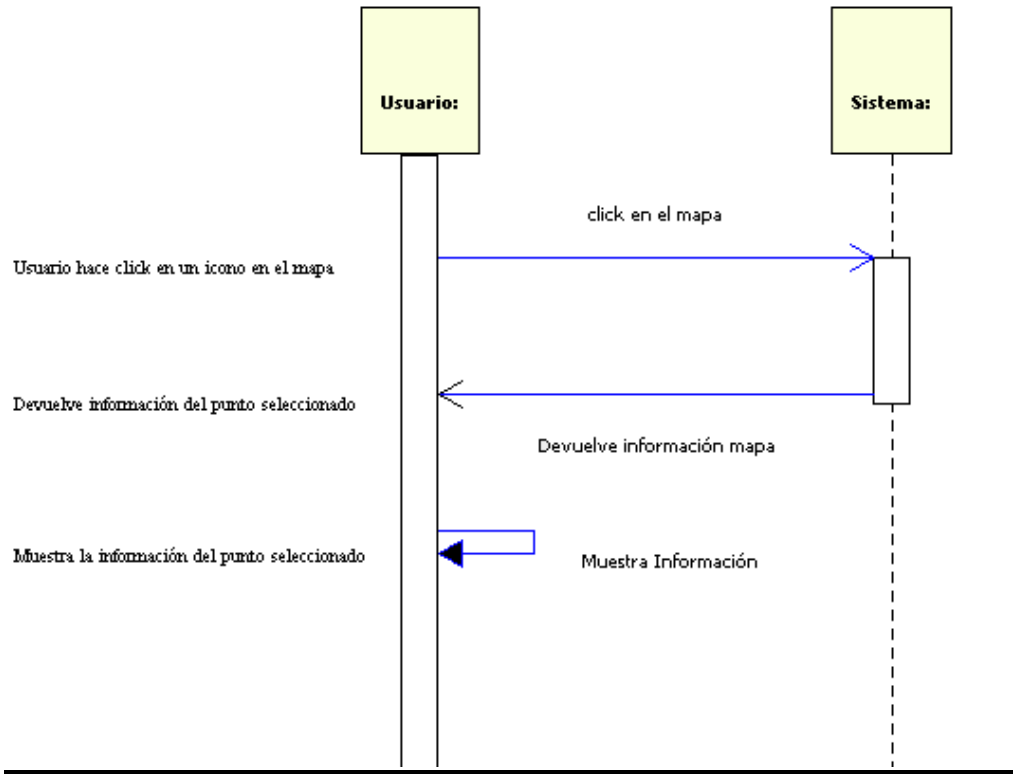
Fig. 12: Diagrama de Secuencia para CU-05



Fuente: Autor

Caso de Uso CU-06 consultas expresado en diagrama de secuencias

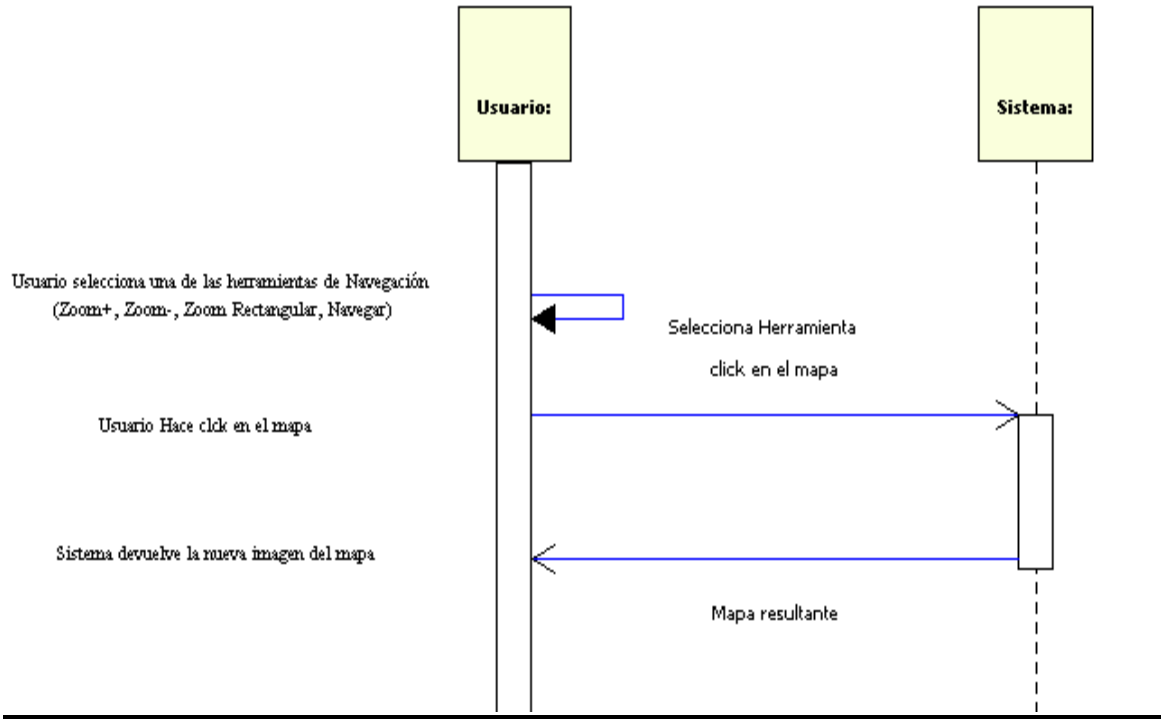
Fig. 13: Diagrama de Secuencia para CU-06



Fuente: Autor

Caso de Uso CU-07 navegación expresado en diagrama de secuencias

Fig. 14: Diagrama de Secuencia para CU-07



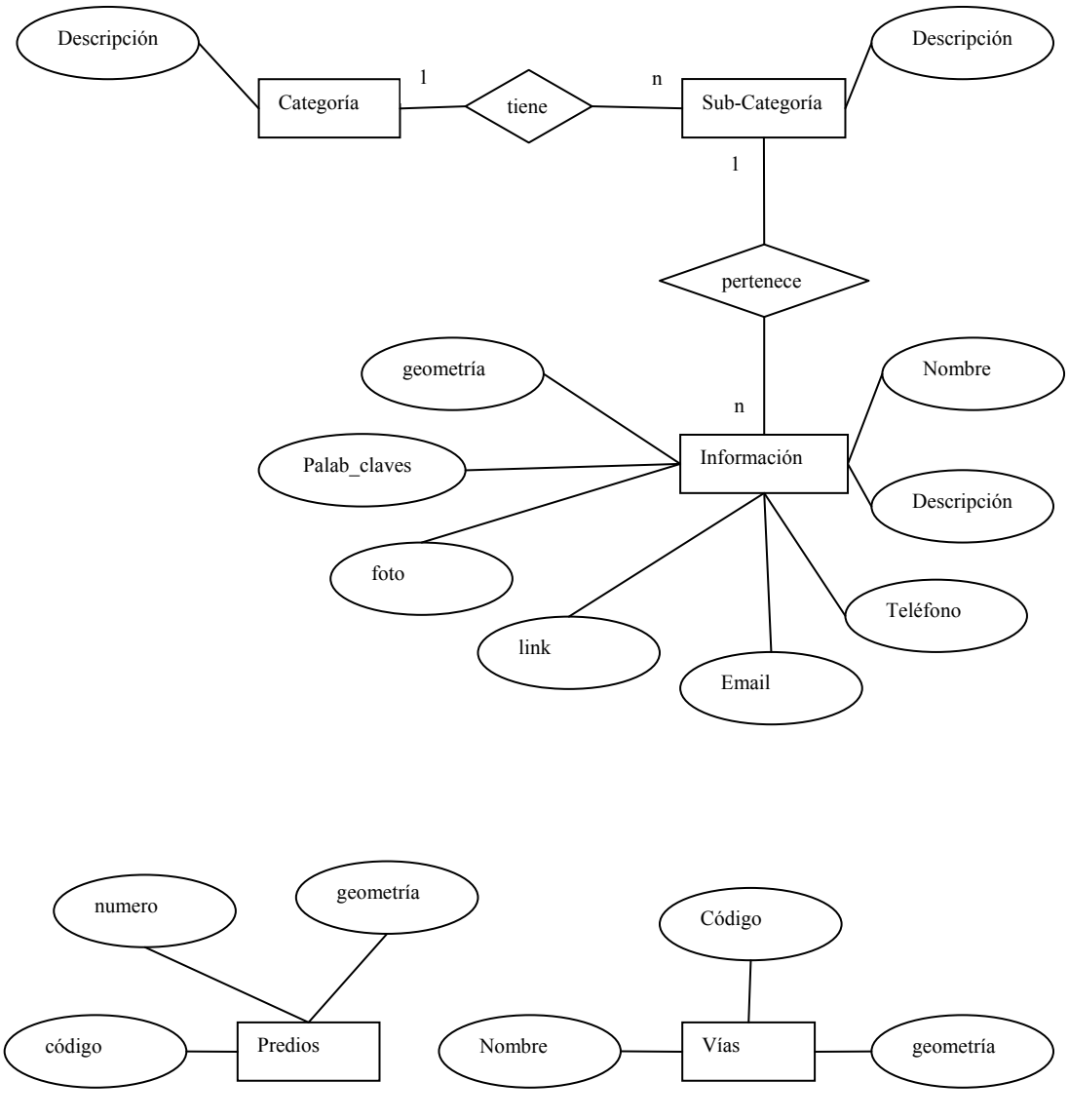
Fuente: Autor

3.5 Modelo de Datos

A continuación realizaremos el modelo de datos con los requerimientos de la aplicación que mas adelante serán utilizados para crear el diseño de la base de datos. En éste modelo incluiremos la definición de entidades y atributos a través del diagrama Entidad-Relación.

Diagrama Entidad-Relación

Fig. 15: Diagrama Entidad Relación



Fuente: Autor

3.6 Análisis de Selección de Software

La selección del software apropiado puede convertirse en una tarea dificultosa, no es fácil de determinar cuál es el software que mejor se adapte a nuestras necesidades ya que existe un gran número de opciones en el mercado. La selección se debe realizar basada en los requerimientos funcionales, pero también evaluar vendedor, implementación, tecnología y los impactos que éstos tienen en nuestro proyecto.

En el mundo de la industria de software nos enfrentamos a un gran dilema en lo que respecta a tendencias de la industria, lo que el día de hoy es lo mas actualizado, el día de mañana se convierte en algo obsoleto, un análisis de selección de software debe enfocarse en el futuro, teniendo previsto el soporte, versiones y mantenimiento que el proveedor del software puede ofrecer a futuro.

En este documento vamos a analizar dos paquetes de software, MapViewSVG y el servidor de mapas MapServer, para esto antes de realizar una comparación entre estos dos programas, realizaré una breve descripción de cada uno.

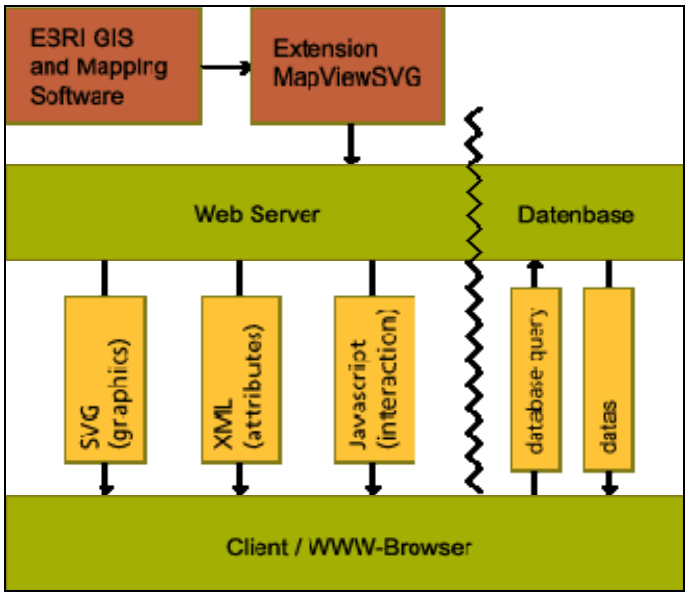
3.6.1 MapViewSVG

MapViewSVG es una extensión para ArcGis 9.x o ArcView Gis 3.x de ESRI, este programa nos da la facilidad de convertir mapas generados en ArcGis a formato SVG (Scalable Vector Graphics), que es un formato para representar información vectorial, nosotros podemos usar estos archivos SVG para publicarlos en Internet o en un CD-ROM, también ofrece la posibilidad de visualizar los datos en páginas con formato HTML mediante el uso de algunas tecnologías conocidas actualmente como AJAX.

MapViewSVG también soporta consulta de atributos en varias formas, en el evento mouseover, identificando el atributo en el mapa o en la tabla de atributos ya que existe un enlace entre la tabla de atributos y la geometría

Esquema de funcionamiento

Fig. 16: Funcionamiento MapViewSVG



Fuente: http://www.mapviewsvg.com

En la documentación en la página web de MapViewSVG [1] nos dice que la consulta a bases de datos es solamente para recuperar atributos alfanuméricos.

Con MapViewSVG podemos construir dos tipos de soluciones:

Solución HTML: ésta solución tiene mas opciones que la solución SVG, muestra la información de los archivos más rápido, la interfaz de usuario es definida con Hojas de estilo, la desventaja es que solamente Microsoft Internet Explorer y Mozilla Firefox 1.5 permiten esta funcionalidad.

En el sitio web oficial del producto recomiendan esta opción solamente para Intranets.

Solución SVG: ésta solución trabaja solamente con formato SVG, como ya dijimos antes para que un navegador soporte este tipo de formatos vectoriales es necesario de un Plug-In (Corel, Adobe-SVG) que aumente las capacidades del navegador,

[1] http://www.mapviewsvg.com

Herramientas adicionales

- Tiling and Dynamic Loading

  Esta herramienta ofrece la posibilidad de cargar dinámicamente, esto reduce la cantidad de tiempo en cargara la aplicación la primera vez.
- Location Search

  Es una herramienta que nos ofrece la capacidad de buscar direcciones, para esta función la aplicación accede a una base de datos mysql, esta herramienta está solamente disponible para la solución en Html.
- Share Geometry

  Nos da la posibilidad de visualizar diferentes valores de campos dentro de una geometría SVG.
- Database Access

  Nos da la posibilidad de consultar atributos alfanuméricos de una geometría en lugar de archivos Xml.

Listado de Características

- Exportar datos a formato SVG.
- Soporte de archivos de imágenes.
- ArcView GIS 3.x para archivos Shape, Annotation, Coverages, Layers, ArcSDE connection, Raster.
- Activar/Desactivar Capas
- Buscar características construyendo consultas (solamente Solución HTML)
- Tabla de atributos de una capa (solamente solución HTML)
- Escala dependiendo de la capa
- Links
- ToolTips texts
- Barra de escala
- Herramienta de medición de distancias y coordenadas.

3.6.2 MapServer

MapServer [2] es un ambiente de desarrollo de código abierto para construir aplicaciones web espaciales,

MapServer fue originalmente desarrollado por la universidad de Minnesota (UMN) en cooperación con la NASA, el software es mantenido por un creciente número de desarrolladores alrededor del mundo y es auspiciado por un diverso grupo de organizaciones que lo mejoran y mantienen.

MapServer nos da la posibilidad de tener mapas personalizados en nuestras aplicaciones web, ofreciendo acceso a una gran variedad de formatos ráster y vectoriales.

Posee una estructura cliente-servidor, donde el usuario solo necesita estar conectado a Internet y utilizando un navegador podrá tener acceso a las informaciones georeferenciadas. El procesamiento es realizado del lado del servidor.

Características

- Salida de Información
  - Dibujo y ejecución de escalas de acuerdo al nivel de zoom
  - Etiquetas a elementos, incluyendo manejo de colisiones.
  - Interfaz de usuario personalizable.
  - Formato de salida personalizable.
  - Soporta tipos de letras TrueType.
  - Elementos de mapa automáticos (barra de escala, mapa de referencia, leyenda)
  - Uso de expresiones lógicas y regulares.
- Soporta varios ambientes de desarrollo
  - Php, Python, Perl, Ruby, Jaca and C#
- Suporte Multi-Plataforma
  - Linux, Windows, Mac OS X, Solarios y otros
- Soporte para múltiples formatos
  - Tiff/GeoTiff, Eppl7, y otros vía GDAL.
  - ESRI shapefiles, PostGis, ESRI ArcSDE, Oracle Spatial, Mysql y otros vía OGR.
  - Soporta las especificaciones Open Geospatial Consortium(OGC)
    - WMS, WFS
    - WMC, WCS, SLD, GML, SOS
- Soporte para múltiples proyecciones
  - Soporte diferentes proyecciones con la librería Proj4.

[2] http://mapserver.gis.umn.edu

3.6.3 Comparación

Cuando evaluamos una posible solución de software, lo principal que debemos considerar es la capacidad de cada alternativa para mantener los requerimientos funcionales que nos hemos planteado en fases anteriores del proyecto.

3.6.3.1 Método de evaluación

Después de tener identificados nuestros requerimientos funcionales y haber dado un rápido vistazo a las características de los productos, necesitamos un mecanismo para comparar las características de estos dos paquetes para cumplir con nuestros objetivos. La herramienta que se va a usar en este documento va a ser una matriz de decisión.

Una matriz de decisión (LITKE-2002) básicamente es un arreglo en donde en el eje x tenemos la lista de alternativas o soluciones a evaluar, y en el eje y tenemos una lista de criterios con sus respectivos pesos dependiendo de la importancia.

Una matriz de decisión nos permite estructurar y resolver el problema realizando los siguientes:

1.- Especificando y priorizando nuestras necesidades con una lista de criterios.
2.- Evaluando, valorando y comparando las diferentes soluciones.
3.- Seleccionando la solución que más cumpla con los criterios que hemos seleccionado.

La clave al usar una matriz de decisión es la valoración del puntaje, usando un set definido de calores discretos, en nuestro ejemplo vamos a basarnos en la información de la Tabla 19

Tabla 19: Tabla para valorar requerimientos.

Valoración	Descripción
1.0	Satisface completamente el criterio de selección o requerimiento
0.5	Satisface parcialmente el criterio de decisión o requerimiento.

0.0	Desconocido o nulo, la alternativa ni satisface ni no satisface la decisión o requerimiento.
-0.5	No Satisface parcialmente el criterio de decisión o requerimiento
-1.0	No Satisface completamente el criterio de decisión o requerimiento.

Fuente: Autor

3.6.3.2 Evaluación

A continuación la matriz de decisión que se ha construido:

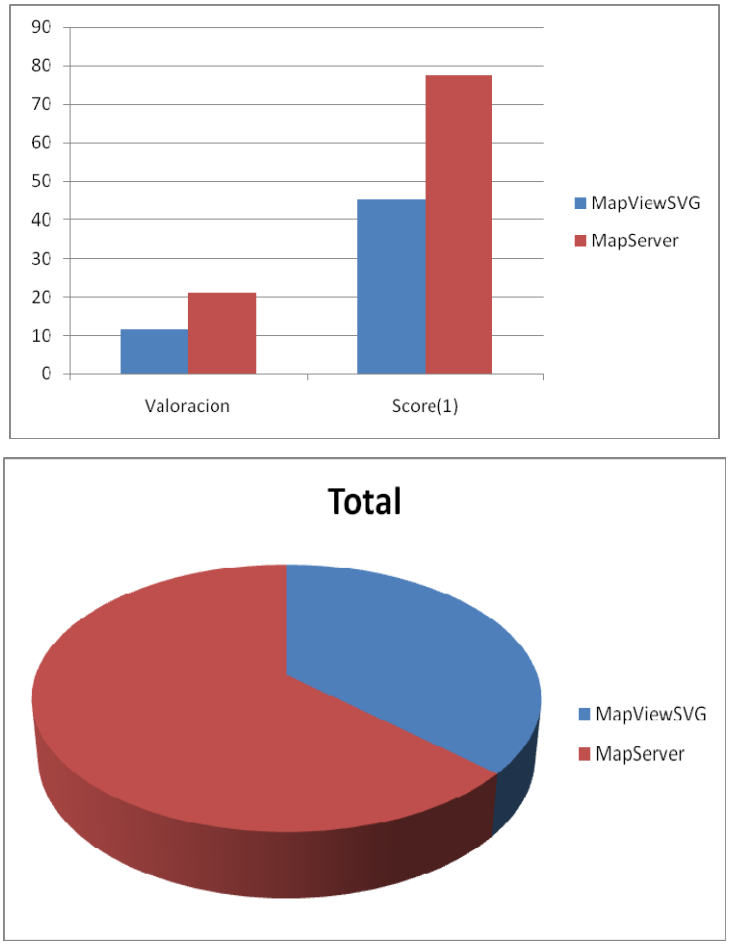
Pesos: están valorados en una escala de 0 a 5.

Tabla 20: Matriz de Decisión

		MapViewSVG		MapServer	
CRITERIO	**Pesos**	Valoración	Score(1)	Valoración	Score(1)
Requiere Plug-in	4	-0,5	**-2**	1	**4**
Compatibilidad con Navegadores	4	1	**4**	-1	**-4**
Velocidad para cargar interfaz	5	-0,5	**-2,5**	0,5	**2,5**
Bajar contenido de diferentes Servidores	1	-1	**-1**	1	**1**
Facilidad de depurar y mantener	2	1	**2**	-0,5	**-1**
Possibilities for proveer una interfaz intuitiva	5	1	**5**	1	**5**
Nivel de dificultad en implementación	2	1	**2**	-0,5	**-1**
Tiempo de implementación	3	1	**3**	-0,5	**-1,5**
Conocimientos Programación	2	1	**2**	-0,5	**-1**
Velocidades de Respuesta en Cliente	4	-0,5	**-2**	1	**4**
Funciones Avanzadas	4	-0,5	**-2**	1	**4**
Integración con Bases de Datos alfanuméricas	4	1	**4**	1	**4**
Interacción con Bases de Datos Geográficas	4	-1	**-4**	1	**4**
Interfaz usuario personalizada	4	0,5	**2**	1	**4**
Interacción con otros sistemas	1	0	**0**	1	**1**
Búsqueda en Bases de Datos	4	1	**4**	1	**4**
Herramientas de Navegación	5	1	**5**	1	**5**
Tipos de Archivos Soportados	3	1	**3**	1	**3**
Tipos de Salida Soportados	3	0,5	**1,5**	1	**3**
Multiplataforma Desarrollo	1	-1	**-1**	1	**1**
Multiplataforma Cliente	3	1	**0**	0,5	**0**
Personalizar Búsquedas	4	-0,5	**-2**	1	**4**
Búsqueda Direcciones	5	1	**5**	1	**5**
Buscar por Nombres	5	1	**5**	1	**5**
Activar/Desactivar Capas	4	1	**4**	1	**4**
Consulta en Mapa	5	1	**5**	1	**5**
Madurez Producto/Vendedor	4	1	**4**	1	**4**
Soporte Técnico	3	1	**3**	0,5	**1,5**
Extendible	2	-1	**-2**	1	**2**
Costo/Licencias	4	-1	**-4**	1	**4**
Facilidad de Uso Cliente	4	1	**4**	0,5	**2**
			0		**0**
Total	108	11,5	**45**	21	**77,5**

(1) Score = Valoración *Peso

Fig. 17: Gráficos Comparativos



Fuente: Autor

En base a los resultados obtenidos con el análisis realizado en la matriz de decisión podemos optar por implementar la aplicación usando el servidor de mapas Mapserver, ya que es el que mas se apega a nuestras necesidades.

3.7 Conclusión

Después de realizar el análisis tenemos identificados claramente cuáles son las características con las que nuestro producto debe contar para cumplir con sus objetivos, y hemos seleccionado el software apropiado para cumplir con todos los requerimientos que nos hemos propuesto para brindar al usuario un sistema de gran utilidad todo mediante técnicas formales que nos brindan seguridad para continuar con las siguientes etapas del proyecto.

Se ha llegado a la conclusión que usar un Servidor de Mapas es la mejor opción cuando se requiere desarrollar una aplicación que requiere un cierto nivel de análisis y consulta sobre los datos espaciales, ya que nos permiten personalizar la aplicación y acceder a diferentes fuentes de datos para la visualización, búsquedas y consultas.

CAPITULO 4
DISEÑO

Introducción.

En la fase de diseño convertiremos la abstracción del análisis en componentes del sistema que luego son las que usaremos e implementaremos en la siguiente etapa del proyecto, la principal actividad del diseño es refinar el análisis y realizar componentes que obedezcan las reglas de la arquitectura sobre la que vamos a desarrollar el sistema.

4.1 Diseño de Datos

En todo sistema de información los datos deben ser almacenados para poder ser procesados y producir salidas que se convierten en información, ésta información es la que nuestro sistema va a mostrar al usuario localizada en el mapa de la ciudad.

Como se explicó en la etapa de Recolección de Información contamos con archivos de tipo Shape de ESRI con las capas que necesitamos para la visualización del mapa, lo que necesitamos ahora es tener una estructura de datos que nos permita realizar las búsquedas (Requerimientos R1-R5) que son la parte principal del sistema.

4.1.1 Diccionario de Datos

Tabla: Categoría.- contiene lista de categorías para clasificar la información de lugares de interés.

Tabla 21: Diccionario de Datos Tabla Categoría

Campo	***Descripción***	***Tipo***	***Tamaño***
IdCategoria	Identificador único de la categoría	entero	
desCategoria	Nombre de la categoría	cadena	50

Fuente Autor

Tabla: SubCategoria.- contiene lista de SubCategorías para clasificar la información de lugares de interés en sub categorías.

Tabla 22: Diccionario de Datos Tabla SubCategoria

Campo	*Descripción*	*Tipo*	*Tamaño*
IdSubCategoria	Identificador único de la sub categoría	entero	
desSubCategoria	Nombre de la sub categoría	cadena	50
idCategoria	Código que relaciona con tabla categoría	entero	

Fuente Autor

Tabla: Información.- contiene lista de lugares de interés clasificados por sub categoría,

Tabla 23: Diccionario de Datos Tabla Información

Campo	*Descripción*	*Tipo*	*Tamaño*
Gid	Identificador único de cada lugar	entero	
Nombre	Nombre de la lugar	cadena	80
Descripcion	Descripcion del lugar	texto	
Direccion	Dirección del lugar	cadena	200
Telefono	Teléfono del lugar	cadena	20
Email	Email del lugar	cadena	100
Link	Enlace a pagina web del lugar	cadena	100
Foto	URL de foto del lugar en formato grande	cadena	100
Foto_mini	URL de foto del lugar en formato pequeño	cadena	
HTML	Contenido HTML sobre el lugar	text	
Tags	Palabras Clave para buscar al lugar	text	
idSubCategoria	Código de SubCategoria que relaciona con la tabla SubCategoria	entero	
the_geom	Campo que contiene la geometría del punto en el mapa del lugar	geometry	

Fuente Autor

Tabla: predios.- contiene la información de los predios de la ciudad, debe ser importada desde el archivo predios.shp y el archivo predios.dbf.

Tabla 24: Diccionario de Datos Tabla Predios

Campo	*Descripción*	*Tipo*	*Tamaño*
Gid	Identificador único de cada predio	entero	
area	Área del predio	decimal	10,2
zona	Código de zona	entero	
sector	Código de sector	entero	

manzana	Código manzana	entero	
predio	Código predio	entero	
Cod_predio	Código de Predio	entero	
calle	Código Calle principal	cadena	10
entre	Código Calle intersección	cadena	10
y	Código calle intersección	cadena	10
numero	Número del predio	Cadena	
parroquia	Nombre de la parroquia	cadena	50
apellidos	Apellidos propietario del Predio	Cadena	50
nombres	Nombres propietario del Predio	Cadena	50
Cedula	Cedula propietario del predio	Cadena	13
telefono	Teléfono propietario del predio	Cadena	50
the_geom	Campo que contiene la geometría del polígono en el mapa del predio	geometry	

Fuente Autor

Tabla: vías.- contiene la información de los predios de la ciudad, debe ser importada desde el archivo vias.shp.

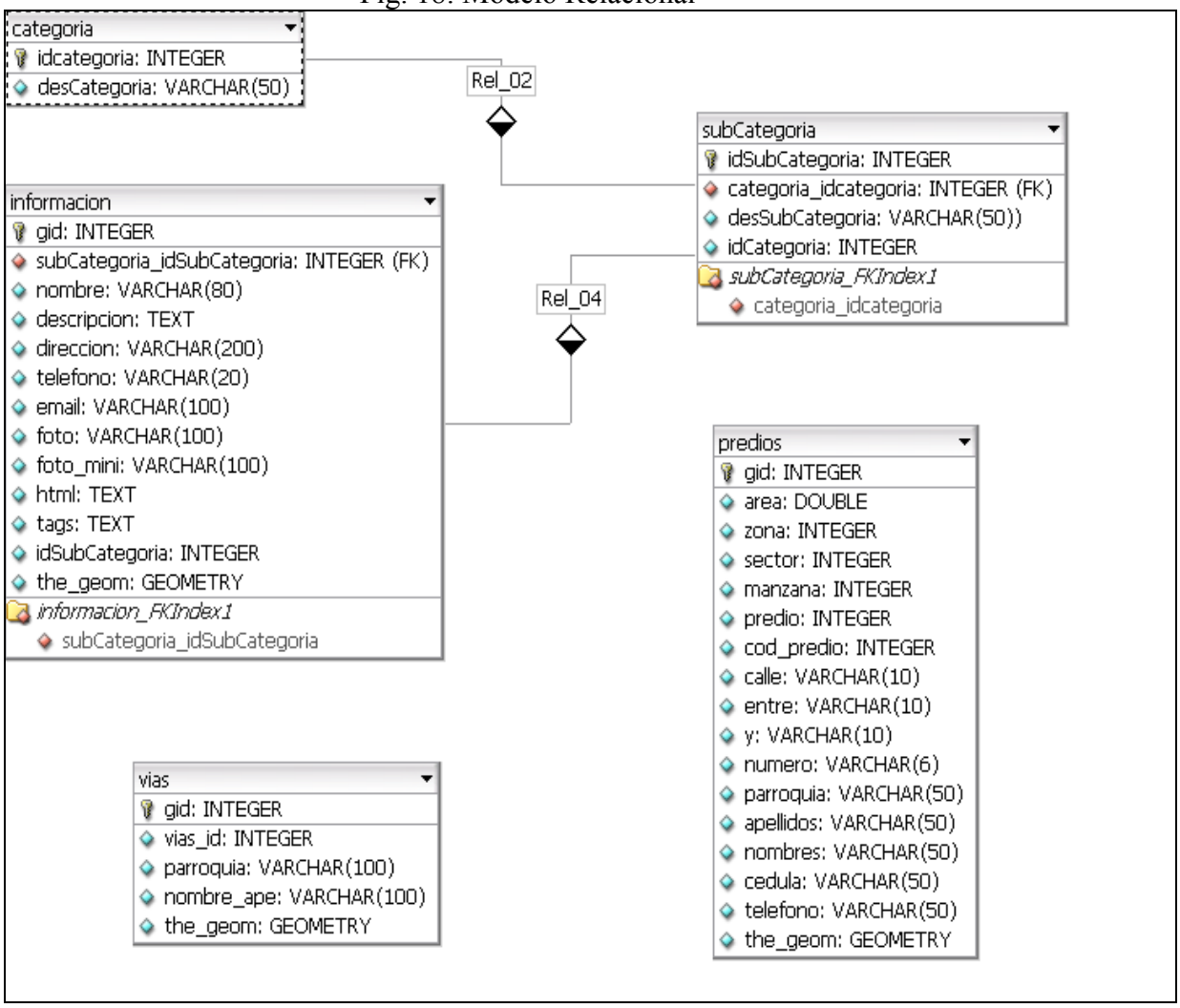
Tabla 25: Diccionario de Datos Tabla Vías

Campo	***Descripción***	***Tipo***	***Tamaño***
Gid	Identificador único de cada vía	entero	
Vías_id	Identificador de la vía	entero	
parroquia	Nombre parroquia pertenece la vía	cadena	100
Nombre_ape	Nombre de la Vía	cadena	100
the_geom	Campo que contiene la geometría de la línea en el mapa de la vía.	geometry	

Fuente Autor

4.1.2 Relaciones de las tablas

Fig. 18: Modelo Relacional



Fuente Autor

4.2 Diseño de Capas Temáticas

La búsqueda de direcciones y sitios de interés es el objetivo principal de este trabajo de tesis, los resultados de éstas búsquedas deben ser representadas en el mapa de la ciudad, por esta razón debemos realizar la representación grafica del mapa valiéndonos de los archivos vectoriales en formato shape que hemos obtenido, y realizar un diseño de la estructura y el orden en el que agruparemos las capas para la visualización en la aplicación, así como otros detalles de visualización.

Organización Jerárquica

Es común en las aplicaciones SIG ubicar a cada capa temática dentro de una jerarquía, y ésta posición en la jerarquía es una importante propiedad a la hora de visualizar gráficamente, así por ejemplo una capa que está más abajo contendrá a otras que están en niveles más altos de jerarquía, esto visualmente significa que las capas con jerarquía más alta son visualizadas encima de las que poseen un nivel más bajo de prioridad.

De esta forma clasificaremos las capas de archivos vectoriales, como se muestra en la siguiente tabla.

También concentraremos las capas en grupos que van a ser los que podamos activar y desactivar, cuando seleccionemos las opciones de activar/desactivar un grupo éstas acciones van a tener efecto en todas las capas de dicho grupo.

Tabla 26: Jerarquía de Capas

Capa	***Grupo***	***Nivel***
Manzanas	Cuenca	0
Ríos Principales	Cuenca	2
Vías Principales	Cuenca	1
Predios	Predios	3
Predios*	Datos	0
Vialidad	Vialidad	4

Fuente Autor

* La razón por la que la capa de predios se encuentra en dos grupos es porque el grupo de datos nos va a servir para realizar consultas directamente sobre la tabla de atributos del archivo shape.

Atributos de Propiedades

Los atributos de propiedades hacen referencia a un campo en la tabla de atributos de la capa en donde tenemos almacenado el valor de algún atributo que define el comportamiento visual de la capa, por ejemplo podemos tener atributos con información de Etiquetas, valores de aéreas y perímetros, ángulos, escalas, etc.

En las capas que nosotros vamos a utilizar solamente utilizaremos el atributo Text de la capa vialidad.shp, para etiquetar los nombres de las vías con su respectivo nombre.

Tabla 27: Atributos de Propiedades

Capa	***Atributo***	***Descripción***	***Propiedad***
Vialidad	TEXT	Nombre de la vía	Etiqueta de la capa vialidad

Fuente Autor

Clasificación por atributo de una Capa

Un objeto en la capa tiene diferentes propiedades basadas en su locación geográfica o en algún atributo, éstos valores de atributos nos pueden ser muy útiles al momento de generar el mapa, nosotros podemos usar estas propiedades para cambiar el aspecto visual de estos objetos según éstos atributos, logrando así una visualización más clara separando a la capa en clases, esta separación de clases se realiza en base a algún atributo del archivo, y podemos dar diferentes comportamientos visuales a cada clase.

Tabla 28: Clasificación por Atributo

Capa	**Atributo**	**Descripción**	**Propiedad**
Manzanas	ZONA	Zona donde está ubicada la vía	Color del objeto manzana
Vialidad	TIPO_DE_VI	Tipo de Vía	Tamaño de Letra en Etiqueta

Fuente Autor

Escalas de visualización de las Capas

Otra de las propiedades importantes a revisar son las escalas a las que podemos visualizar cada una de las capas de información, es necesario indicar este tipo de configuración a la capa, ya que según su contenido y tipo de geometría afecta de diferentes formas a la visualización de todo el mapa, por ejemplo en la capa de

predios mostrada en una escala 1:15000 nos es casi imposible identificar un predio, mientras que a una escala 1:2000 obtenemos una vista mucho más clara del mapa.

Tabla 29: Escalas de Visualización

Capa	Grupo	Escala Máxima	Escala Mínima
Predios	Predios	5000	0

Fuente Autor

El resto de capas no tienen restricción en cuanto a la escala de visualización.

4.3 Diseño de Programa

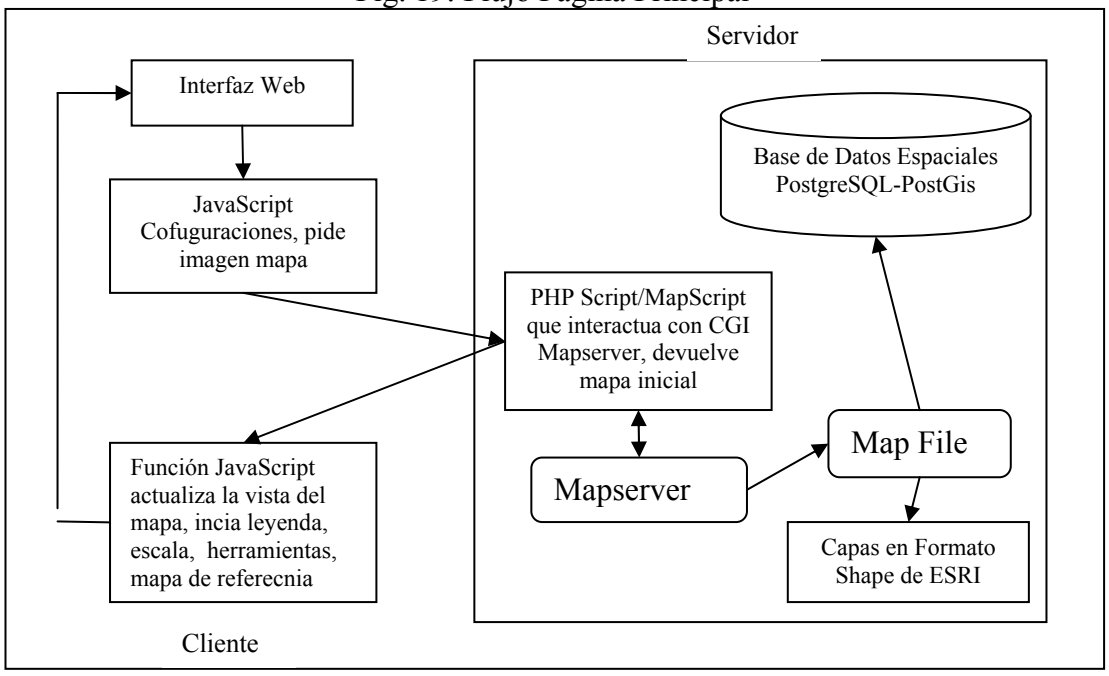
El propósito del diseño del programa es modelar las especificaciones de la aplicación de tal manera que al ser codificadas satisfagan los requerimientos que nos hemos planteado

4.3.1 Flujos de datos del Sistema

Los siguientes gráficos modelan el flujo de información del sistema.

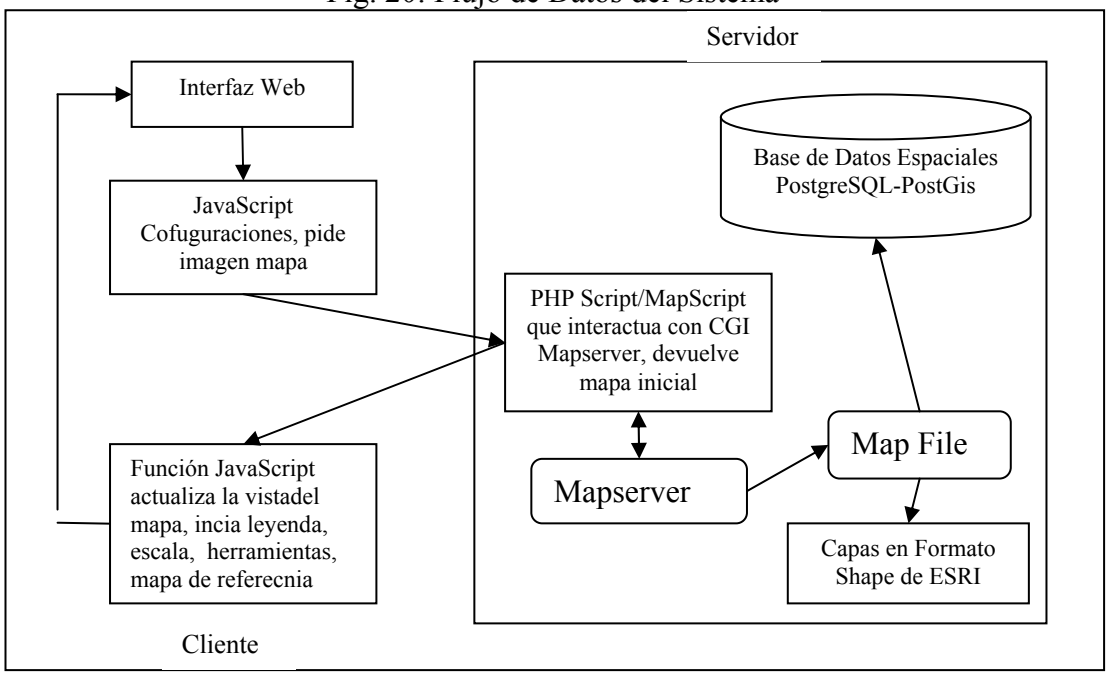
Flujos de datos pagina inicial

Fig. 19: Flujo Página Principal



Fuente Autor

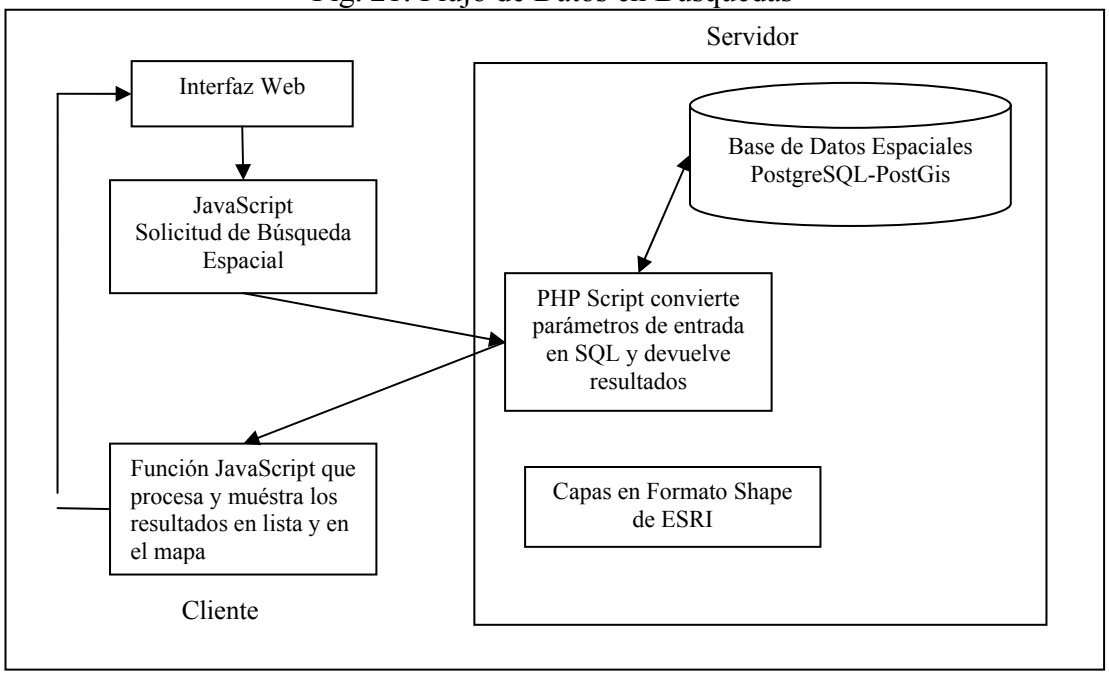
Fig. 20: Flujo de Datos del Sistema



Fuente Autor

Flujo de datos en las búsquedas

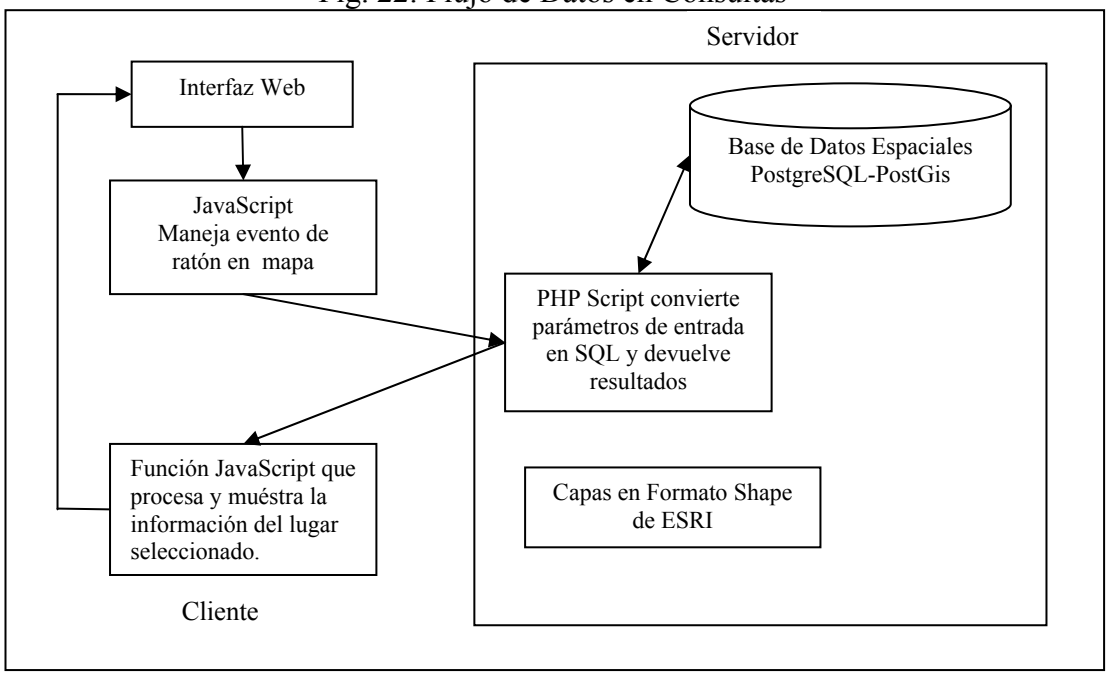
Fig. 21: Flujo de Datos en Búsquedas



Fuente Autor

Flujo de datos en consultas

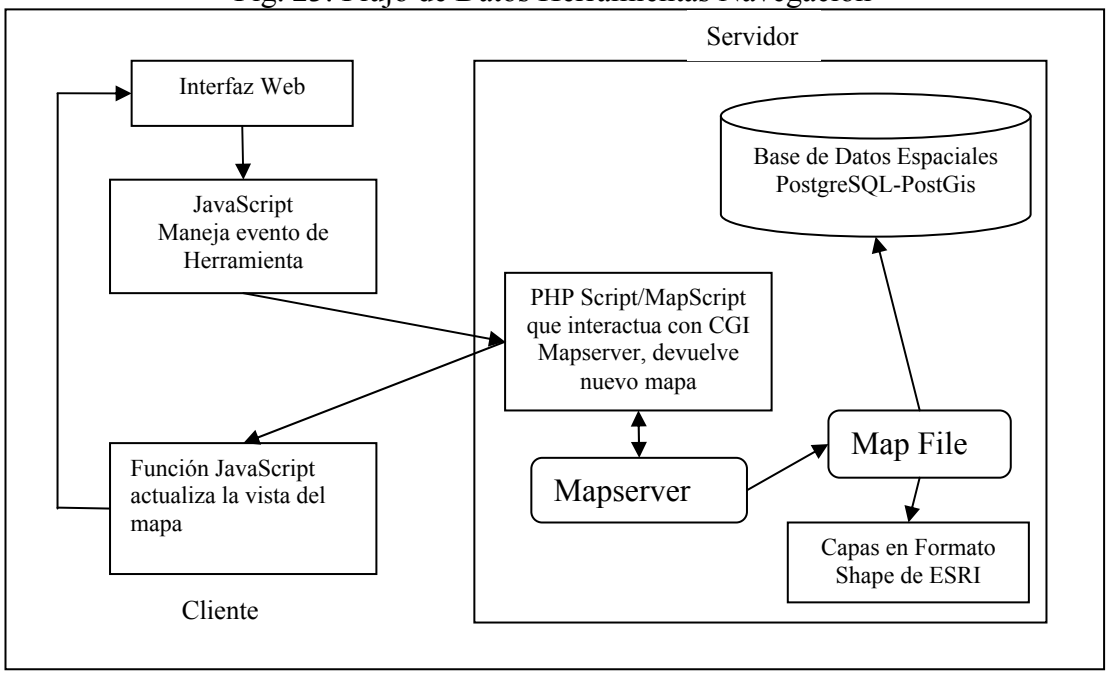
Fig. 22: Flujo de Datos en Consultas



Fuente Autor

Flujos de Información en herramientas de navegación.

Fig. 23: Flujo de Datos Herramientas Navegación



Fuente Autor

4.3.2 Módulos del Sistema

Controlador Principal

En este módulo se encuentran las funciones que inicializan, todas las herramientas y otros módulos que luego vamos a utilizar, también se incluirán configuraciones si es que fuera necesario.

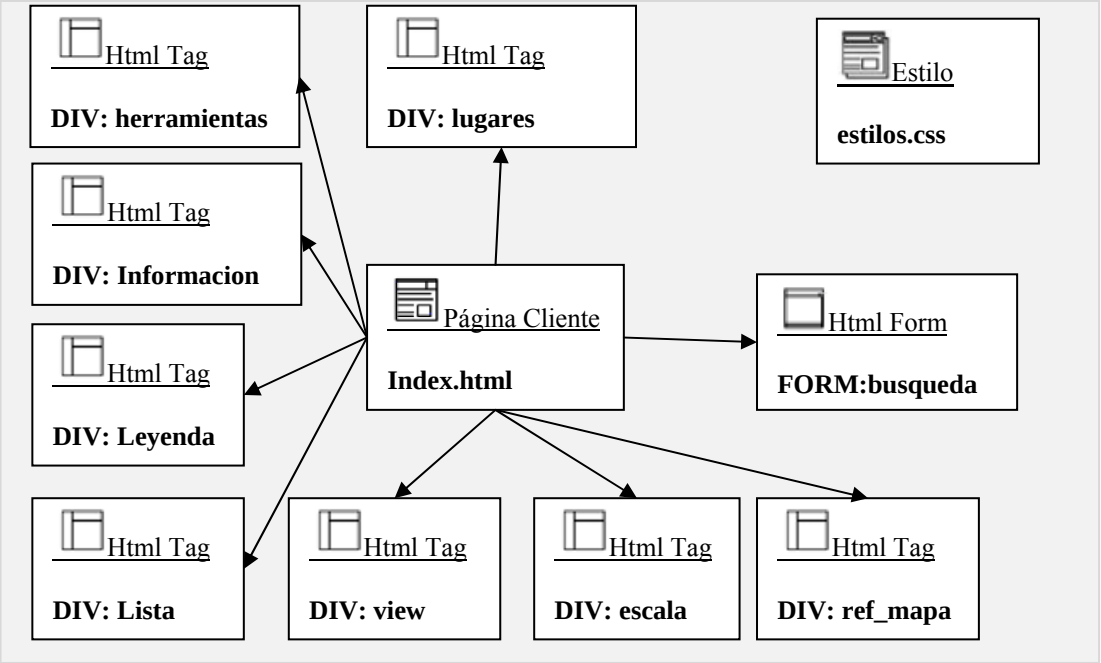
Fig. 24: Controlador Principal

Fuente Autor

Vista

Este modulo contiene la parte de visualización en el navegador del cliente, la página estará dividida en varias etiquetas <DIV>, donde ubicaremos sus respectivos componentes, la ubicación y propiedades de cada etiqueta div estarán manejadas en la hoja de estilos estilo.css, también tenemos dentro de la página una etiqueta <FORM> donde se ubicarán los elementos para que el cliente llene la información de las búsquedas.

Fig. 25: Vista

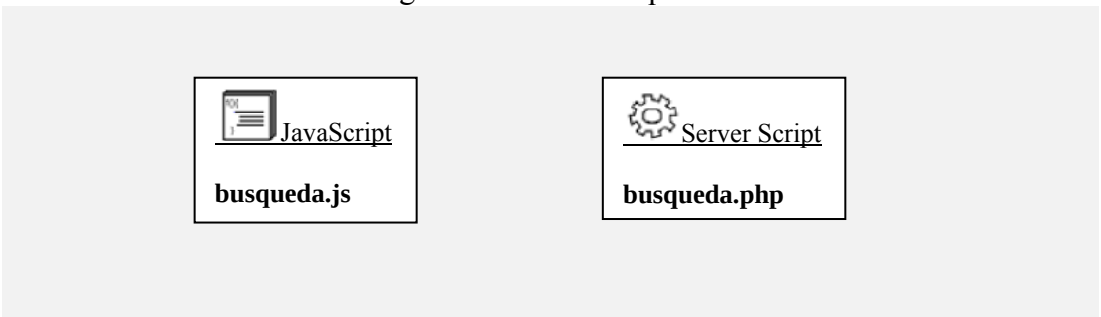


Fuente Autor

Búsquedas

El módulo de búsqueda es el que contendrá los archivos necesarios para realizar las búsquedas de direcciones y lugares, en el archivo JavaScript búsqueda.js tendremos las funciones que recibirán los parámetros y realicen la llamada al archivo búsqueda.php que al en el lado del servidor se encargará de acceder a la base de datos y retornar el resultado para actualizar la vista en el cliente.

Fig. 26: Módulo Búsquedas



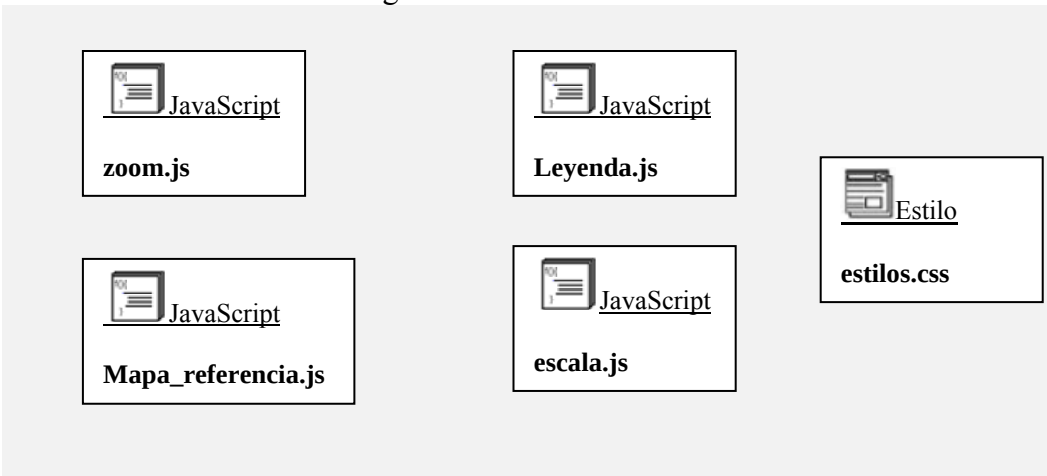
Fuente Autor

Herramientas

Contiene los archivos que controlan las funciones que realizan cada una de las herramientas, estas funciones interactuarán con Mapserver a través de MapScript

para generar la vista del nuevo mapa, y estas mismas funciones serán las encargadas de actualizar la vista en el e cliente

Fig. 27: Módulo Herramientas

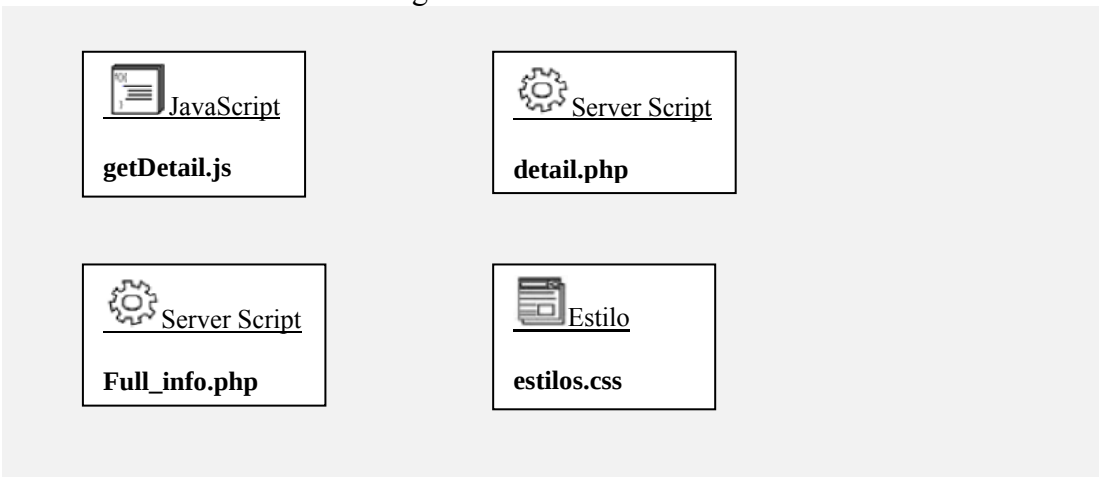


Fuente Autor

Consulta

En este modulo implementaremos las funciones que nos permitan la consulta de lugares que aparezcan en el mapa, para esto cuando el usuario realice un clic en un lugar en el mapa se ejecutará las funciones dentro de getDetail.js, estas funciones recuperan la información del lugar realizando una llamada al archivo detail.php, y luego actualiza la vista mostrando la información obtenida dentro de la etiqueta div con id=”informacion”, también tendremos una función dentro de getDetail.js que abra la página full_info.php con toda la información del lugar seleccionado

Fig. 28: Módulo Consulta



Fuente Autor

4.3.3 Diseño de Aplicación Web

El diseño de la aplicación web nos va a servir para conceptualizar los casos de uso y los diagramas de secuencia en un esquema de funcionamiento del sistema, en donde podremos modelar la interacción con los diferentes módulos y componentes que implementaremos.

PaginaCliente.- una instancia de la página del cliente es la capa de presentación en la cual se realizará la interacción con las peticiones del cliente a través del navegador.

JavaScript.- hace referencia a módulos escritos en javascript que controlan las peticiones del cliente, y a su vez realizan las peticiones a scripts o archivos en el servidor, también actualizan la vista (presentación) asíncronamente.

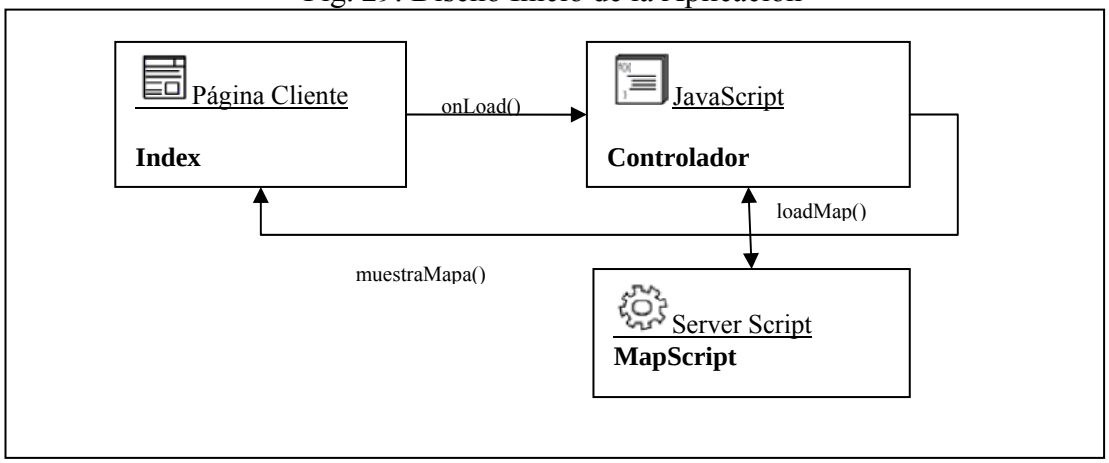
Server Script.- Son scripts que se ejecutan en el servidor dando como resultado páginas web dinámicas o información, generalmente contienen scripts que interactúan con recursos en el servidor como pueden ser sistemas externos o bases de datos.

HTML Form.- un formulario es una colección de campos de entrada de información en la página del cliente, que sirven para recolectar información del cliente para ser procesada.

Inicio de la Aplicación

Se realiza cuando el usuario entra al url de la aplicación, se ejecuta el evento onLoad() llama a las funciones javascript que son el controlador principal del programa, en donde se cargan todas las herramientas, configuraciones y se inicializa el mapa.

Fig. 29: Diseño Inicio de la Aplicación

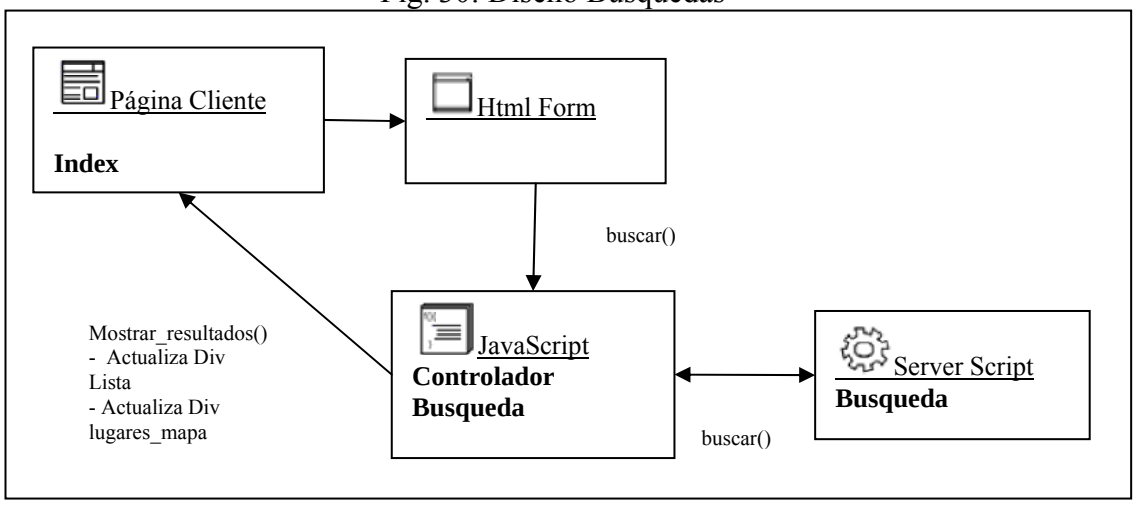


Fuente Autor

Búsquedas

El usuario accede al formulario para realizar una búsqueda, llena los datos necesarios y mediante el botón buscar ejecuta la función javaScript en el archivo que contiene las funciones controladores de las búsquedas, éstas invocan a un archivo Php en el servidor que realiza la búsqueda en la base de datos, retorna los resultados al controlador y en el controlador se genera los resultados, primero una lista de todos los resultaos y se actualiza la etiqueta div que tenga el id=”Lista” , luego generamos los puntos en el mapa con los resultados de la búsqueda actualizando la etiqueta div de id=”lugares_mapa”

Fig. 30: Diseño Búsquedas

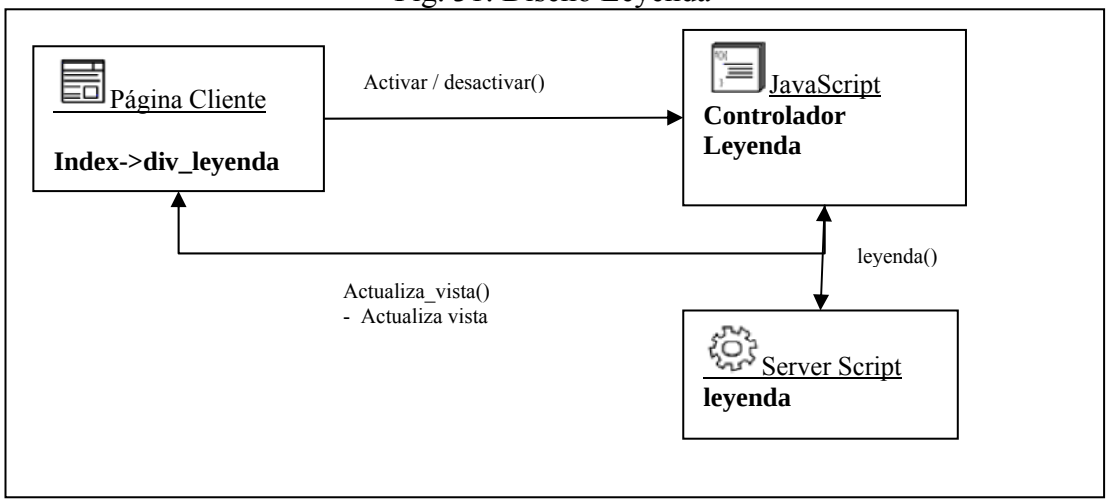


Fuente Autor

Actualizar Leyenda

La etiqueta div id= "leyenda" fue generada cundo la aplicación se cargó, cada ítem en la leyenda tiene un componente checkBox que ejecuta los eventos activar/desactivar capas implementados dentro del controlador de la leyenda, aquí se invoca a las funcione MaScript que realizan la actualización de la imagen del mapa que es devuelta al controlador para que actualice la vista en el cliente.

Fig. 31: Diseño Leyenda

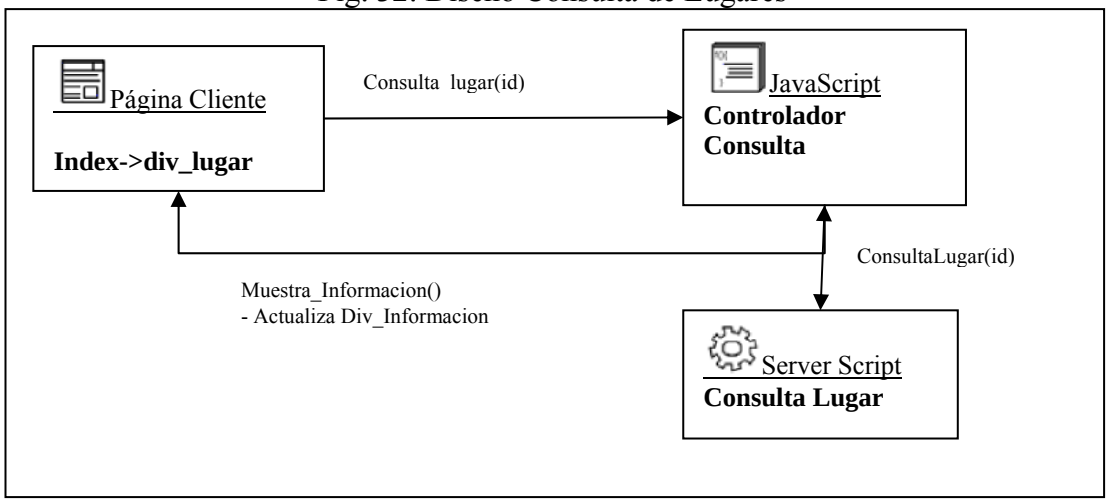


Fuente Autor

Consulta de Lugares

En la aplicación dentro del div id= “lugares” tendremos los lugares que hemos seleccionado en la leyenda o los lugares que han sido resultado de alguna de las búsquedas, cuando se generan estos puntos en el mapa deben tener la llamada al evento clic implementada en el controlador Consulta, pasando como paramento su propio id de lugar, el controlador de Consulta realizará una llamada asíncrona a un script php que retorne las información del lugar seleccionado, y luego actualizará la etiqueta div id=”información” que es donde mostraremos los resultados en la pagina del cliente

Fig. 32: Diseño Consulta de Lugares

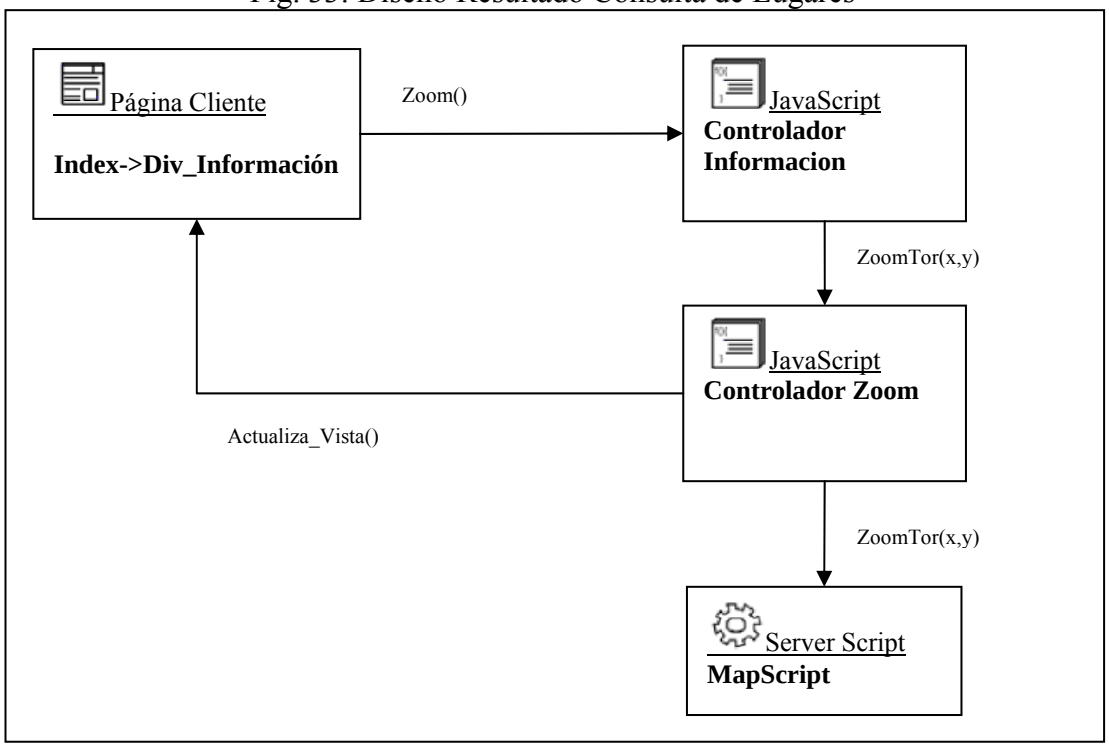


Fuente Autor

En el resultado que hemos obtenido después de la consulta, actualizamos dinámicamente la etiqueta div id=”información”, aquí vamos a tener dos controles, que ejecutarán funciones escritas en el controlador Información,

La función Zoom, que nos permitirá realizar un zoom al lugar seleccionado, utilizando para esto el controlador de la herramienta Zoom, que interactúa con MapScript para actualizar la vista del mapa.

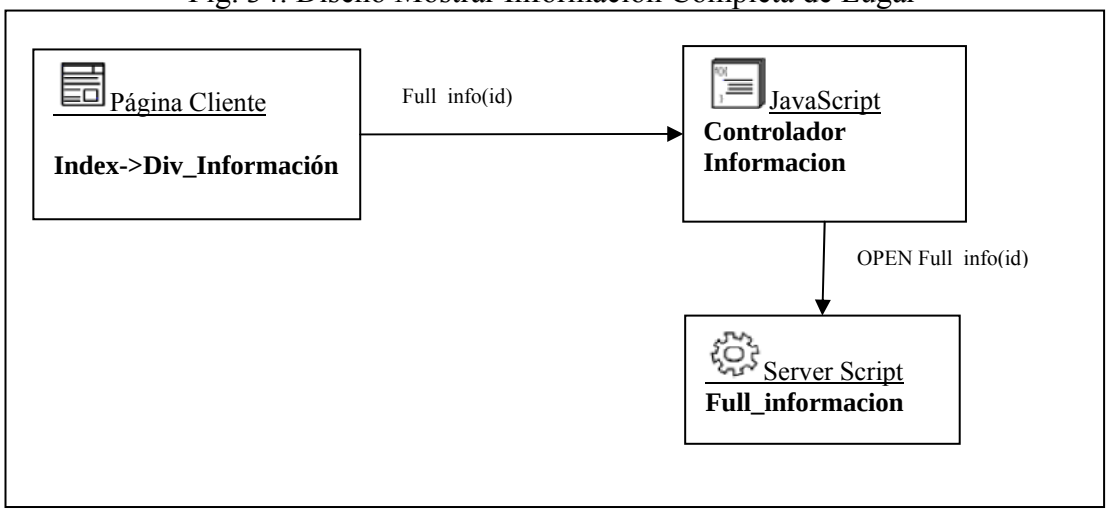
Fig. 33: Diseño Resultado Consulta de Lugares



Fuente Autor

La función full_info (), que mostrara en una nueva ventana el script full_info.php que nos muestra toda la información de el lugar seleccionado.

Fig. 34: Diseño Mostrar Información Completa de Lugar

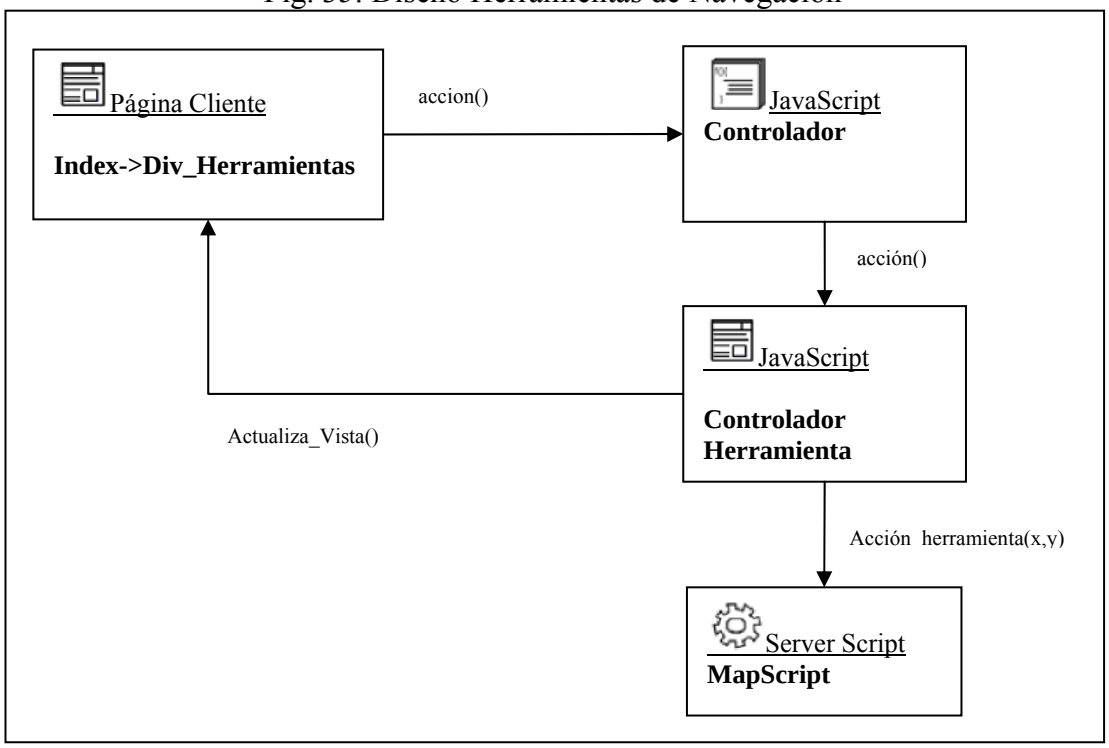


Fuente Autor

Herramientas

El div id=”herramientas” que se inicializó junto con la aplicación contiene los íconos de cada herramienta, estos iconos realizan las llamadas a sus respectivos controladores implementaos en javaScript, y cada uno de estos realizarán las llamadas a las funciones MapScript dependiendo de qué función realice cada herramienta, y finalmente el controlador de la herramienta actualizará la vista en el lado del cliente.

Fig. 35: Diseño Herramientas de Navegación



Fuente Autor

4.3.4 Diseño de Funciones.

A continuación se realizará un diseño de las funciones que utilizaremos para realizar las búsquedas y consultas.

Búsqueda por Dirección.

```
Buscar_direccion (principal, intersección, numero)
{
   If(numero=””)
   {
        R=Seleccionar intersección(v1.the_geom, v2.the_geom), x, y
        De vías v1, vías v2
        Condition v1.Text like $principal Y v2.Text like $Intersection
```

```
        Y intersecta(v1.the_geom, v2.the_geom);

    }else
    {
        R=Seleccionar intersección(v1.the_geom, v2.the_geom), x, y
        De vías v1, vías v2, Predio p
        Condicion v1.Text like $principal Y v2.Text like $Interseccion
        Y intersecta(v1.the_geom, v2.the_geom)
        Y p.numero= $numero
        Y Buffer(p.the_geom) intersecta a principal

    }


 Desde i=0 hasta R.tamaño
{
        x= R(i).x
        y= R(i).y
}

}
```

Busqueda por nombre.

```
Buscar_nombre (nombre)
{
        R=Seleccionar id,  x(the_geom) , y(the_geom)
        De informacion
        Condicion nombre like $nombre
        Desde i=0 hasta R.tamaño
        {
                x= R(i).x
                y= R(i).y
                id? R(i).id
        }

}
```

Busqueda por Palabras Clave.

```
Buscar_claves (cadena)
{
        R=Seleccionar id,  x(the_geom) , y(the_geom)
        De informacion
        Condicion tags like $cadena
        Desde i=0 hasta R.tamaño
        {
                x= R(i).x
                y= R(i).y
                id= R(i).id
        }
```

```
}
```

Consulta

```
consulta (id)
{
        R=Seleccionar *
        De informacion
        Condicion id=$id

        Desde i=0 hasta R.tamaño
        {
                Nombre= r(i).Nombre
                Direccion= r(i).Direccion
                telefono= r(i).Telefono
        }

}
```

Mostrar Lugares en Mapa

```
mostrar (x, y, id)
{
        Mapa= obtener_mapa();
        img= nuevo Elemento(img)
        img.source= "pin.gif"
        mapa.insertarImagen(img,x,y)
}
```

4.3.5 Conclusiones

Después de haber realizado el diseño de la aplicación tenemos claro que es lo que debemos hacer, y cómo funciona el sistema para satisfacer los objetivos que nos hemos planteado, teniendo en mente la estructura de la base de datos expresada en un diagrama relacional, que luego transformaremos en tablas con sus respectivos campos y tipos de datos, también realizamos el diseño de la estructura que va a tener la aplicación web, estableciendo cuáles son los archivos que necesitamos para la parte del cliente y los archivos en el servidor y, por último, los diseños de la funciones que luego serán implementadas y permitirán realizar las búsquedas y consultas.

CAPITULO 5
IMPLEMENTACIÓN

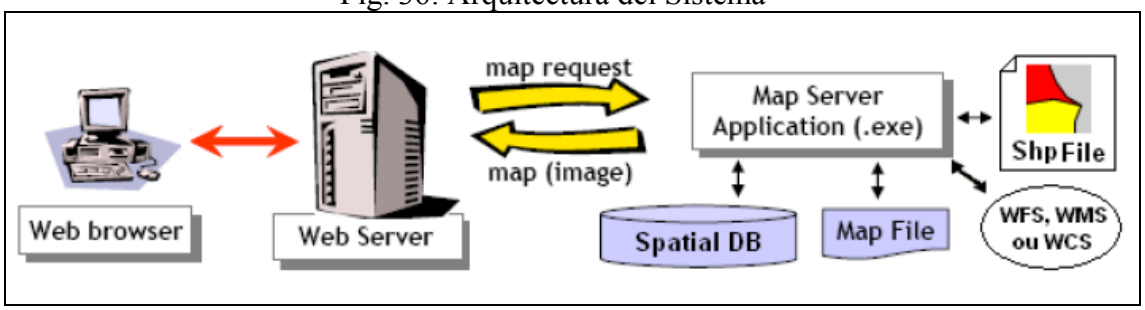
Introducción.

Completadas las etapas del análisis, en donde identificamos las características que debe tener nuestro sistema, y el diseño, donde hemos definido la arquitectura que vamos a usar, el siguiente paso es realizar la implementación de la aplicación donde se plasma el resultado de las etapas anteriores, en un sistema físico que podamos ejecutar desde un navegador; convirtiendo el diseño en el código de la aplicación. Para esto, realizaremos la instalación del servidor de mapas, servidor web, servidor de bases de datos y la aplicación en el servidor y cliente; teniendo como resultado la aplicación web que cumpla con los requisitos planteados.

5.1 Arquitectura

El siguiente gráfico muestra un esquema de la arquitectura que vamos a montar para la aplicación.

Fig. 36: Arquitectura del Sistema



En donde distinguimos los siguientes componentes:

- Web Server.- es el servidor web, éste es el que maneja las peticiones (request) del usuario, el servidor web mantiene la comunicación entre el cliente, el servidor de mapas y la aplicación en el servidor web, también devuelve los resultados de las peticiones (response) al cliente.

- MapServer.- el servidor de mapas maneja las funciones que interactúan con la información geográfica para producir el mapa de salida(zoom, panning, activar/desactivar capas, etc.) generando el mapa resultante

- Cliente.- El cliente se ejecuta en un navegador, éste solicita los recursos del servidor web, si la petición es una función del mapa, el servidor web interactúa con el servidor de mapas, y el cliente muestra en el navegador la respuesta del servidor web.

- Spatial Data Base.- es la base de datos que vamos a usar para almacenar información que puede tener datos geográficos, para nuestra aplicación vamos a usar la base de datos PostgreSQL con la extensión PostGis.

- Shape Files.- son los archivos vectoriales que contienen la información georeferenciada para generar la imagen del mapa de la ciudad, contamos con los archivos de manzanas, predios, ríos y vías.

- MapFile.- es el archivo de configuración de nuestro mapa en el que configuramos los archivos o tablas con los que vamos a generar el mapa, escalas, colores, leyenda, y todo lo que tiene que ver con el mapa resultante y su visualización.

5.2 Servidor de Web.

Este componente de la arquitectura recibe las peticiones (request) del cliente, las procesa y devuelve una respuesta (response) que es devuelta al cliente para que la procese o visualice

En ésta aplicación voy a usar el servidor Servidor HTTP Apache, que es un servidor Http de código abierto y multiplataforma, es un servidor muy modular, y nos da la posibilidad de aumentar sus características con la instalación de módulos.

Instalación bajo Windows.

En la página oficial http://www.apache.org bajamos el Instalador binario para Windows y lo ejecutamos.

Llenar correctamente los parámetros esenciales de configuración:

- Network Domain o dominio de red si lo tenemos
- Nombre del servidor en minúsculas o dirección IP del computador.
- E-mail del administrador del servidor web.

Después de finalizada la instalación podemos probar el servidor entrando en el navegador a la siguiente dirección http://localhost y si todo ha ido bien obtenemos lo siguiente.

Fig. 37: Servidor Apache

Fuente Autor

5.3 MapServer.

Mapserver es una herramienta SIG orientada a la publicación y uso de mapas en línea vía web, es un desarrollo de código abierto para construir aplicaciones con capacidades espaciales en internet, una de las principales ventajas de MapServer es que puede implementarse en la mayoría de plataformas sean comerciales o no,

Librerías

Existe una gran cantidad de librerías que pueden ser usadas por MapServer, la mayoría de estas librerías son opcionales,

Estas librerías proveen acceso a bases de datos, soporte para Servicios Web (WMS, WFS, etc.), y formatos para el formato de salida de la información

Entre las más importantes podemos citar las siguientes:

FreeType.- motor para rénder de fuentes.

LibJPEG.- es usada por MapServer para generar imágenes JPEG

Libpng.- usada para generar imágenes PNG

Zlib.- librería de compresión de datos

GDAL (Geospatial Data Abstraction Library).- permite importar y proyectar imágenes ráster georeferenciadadas.

OGR.- permite leer y escribir en una variedad de formatos vectoriales

Proj.4.- posee funciones para manejar proyecciones cartográficas y realizar conversiones.

Shapelib- es una librería escrita en C para crear y manipular archivos Shape

PostgreSQL Client Libraries.- dan la posibilidad de accede a bases de datos PostGIS desde MapSever.

Oracle Spatial Client Libraries- dan la posibilidad de accede a bases de datos Oracle Spatial.

Ming .- permite a MapServer crear formatos de salida SWF(Shockwave Flash)

PDFLib.- permite a MapServer realizar capturas en formato PDF

Todas estas librerías están disponibles para descargarse gratuitamente y son de código abierto.

Tabla 30: Donde obtener las librerías

Package	Location
MapServer	http://mapserver.gis.umn.edu/dload.html
GD	www.boutell.com/gd
FreeType	www.freetype.org

libJPEG	www.ijg.org/files
libpng	www.libpng.org/pub/png
zlib	www.gzip.org/zlib
GDAL	http://gdal.maptools.org
Proj.4	http://proj.maptools.org
shapelib	http://shapelib.maptools.org

Fuente KROPLA- 2005

5.3.1 Instalación

Desde la página de descarga del sitio oficial de MapServer http://mapserver.gis.umn.edu/dload.html podemos obtener el software, para un desarrollo más rápido podemos descargar paquetes en donde ya vienen integradas las librerías que se revisó anteriormente, inclusive que contiene ya integrado el servidor web apache.

Para Linux éste paquete se llama FGS Linux Installer y para ambiente Windows el paquete se llama MS4W (MapServer for Windows)

Nosotros usaremos MS4W, la instalación es muy sencilla solamente tenemos que descargar el archivo ms4w.2.2.g.zip y descomprimirlo en cualquier directorio, para registrar al Servidor apache como un servicio en el sistema operativo ejecutamos el archivo apache-install.bat que se encuentra en la raíz del directorio.

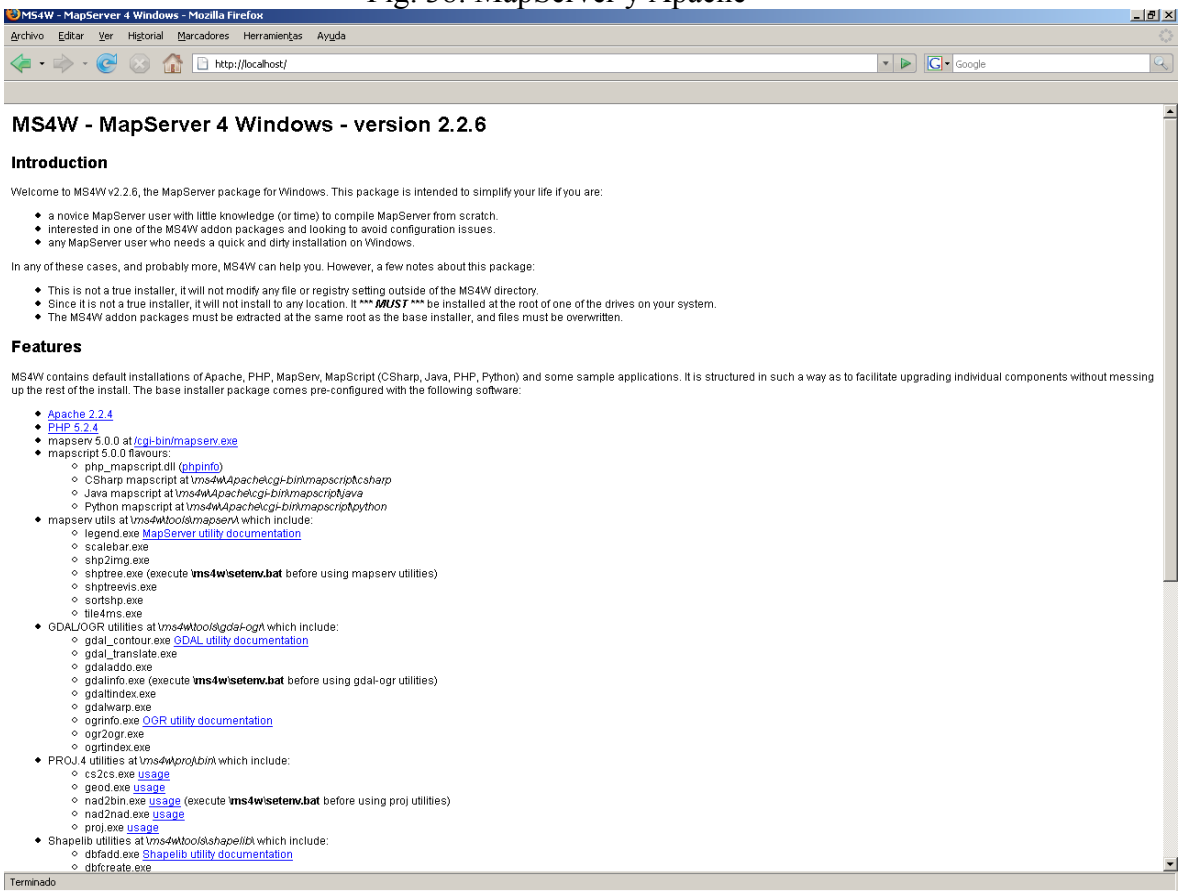
El paquete MS4W contiene un servidor web pre configurado que contiene

- Apache HTTP Server version 2.2.3
- PHP version 5.1.6 or 4.4.4
- MapServer CGI 5
- MapScript 5 (CSharp, Java, PHP, Python)
- Includes support for Oracle 10g, and SDE 9.1 data (if you have associated Client/dlls)
- MrSID support built-in
- GDAL/OGR Utilities

- etc

Podemos realizar una prueba del servidor entrando a http://localhost en el navegador.

Fig. 38: MapServer y Apache



Fuente: Autor

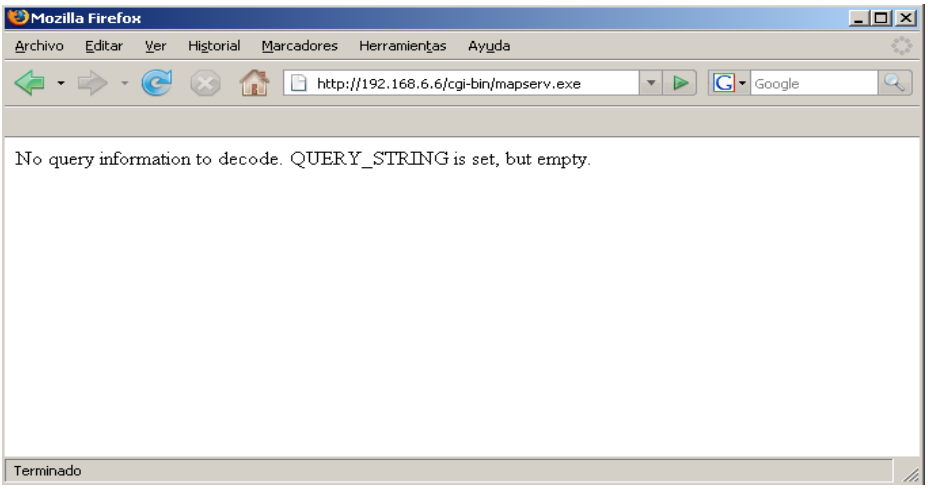
En ésta página podemos encontrar una lista de las librerías que se encuentran instaladas y enlaces a documentación y ejemplos de cada una.

A continuación probaremos que el CGI de mapserver esté funcionando correctamente para esto introducimos la siguiente dirección en el navegador:

http://localhost/cgi-bin/mapserv.exe

Si es que está funcionado correctamente debemos obtener un mensaje como el siguiente.

Fig. 39: MapServer CGI



Fuente Autor

Con estos pasos tenemos funcionando el servidor web y el servidor de mapas juntos.

5.3.2 Funcionamiento de MapServer

Mapserver es un programa escrito en C, que funciona como un programa CGI a través de un servidor Http para que este pueda tener acceso desde Internet. Este programa genera una imagen de mapa obtenida de acuerdo a los parámetros que nosotros le proporcionemos.

El programa mapserver utiliza los siguientes recursos para obtener una imagen de mapa como salida:

- Servidor Web.- en nuestro caso el servidor HTML Apache
- Un archivo de mapa Map File con extensión ".map" que es en donde configuramos los datos que nos van a servir de entrada para generar el mapa y las salidas (Mapa, leyenda, barra de escala, mapa de referencia, formato de salida, etc) que va a generar MapServer, en el Anexo 1 podemos enconrar una referencia del Map File.
- Un Template file (opcional) para la presentación de la salida en el navegador del cliente.

5.3.3 Creación del Archivo de Mapa (MapFile)

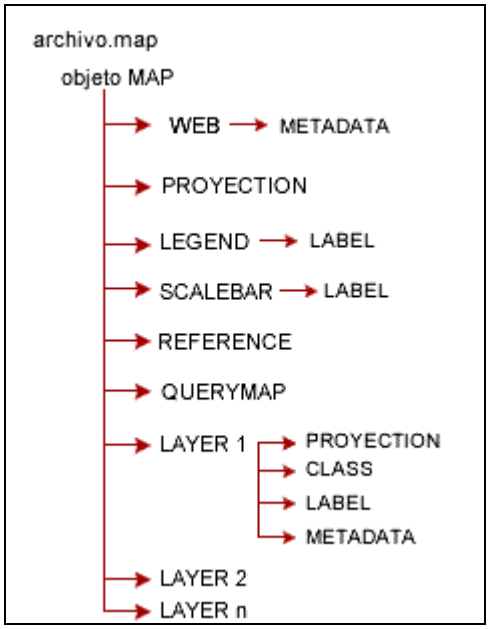
El archivo de configuración de un mapa en MapServer es un archivo plano de texto con extensión ".map", en el que incluiremos una serie de definiciones de objetos que determinarán la apariencia y comportamiento del mapa que se mostrará en el navegador

Un archivo .map consta de varios tipos de objetos que están organizados jerárquicamente, cada Objeto se inicia con el nombre del objeto y termina con la palabra END, cada objeto tiene un conjunto de propiedades de tipo propiedad-valor.

Los comentarios son introducidos al principio de la línea con el carácter #

La siguiente figura nos muestra la jerarquía y los tipos de objetos que puede contener un Map File.

Fig. 40: Jerarquía de Objetos en Map File



Fuente Autor

En el Anexo 1 encontraremos una complete referencia de la estructura y sintaxis de un Map File obtenida de la página oficial de MapServer http://mapserver.gis.umn.edu/docs/reference/mapfile.

Para empezar el Map File ponemos las etiquetas MAP y END que es el objeto principal

```
MAP
END
```

A continuación vamos a poner las propiedades del objeto MAP, incluyendo los objetos que usaremos para nuestro mapa, como son,

- Leyenda
- Mapa de Referencia
- Barra de Escala
- Capas de Información

```
MAP
  NAME cuecna_map  # Nombre del mapa
  STATUS ON
  SIZE 400 300 # tamaño de visualización por defecto
  EXTENT 720481.1942734885 9677818.107311556 724019.5534738856 9679931.24505935
#georeferenciacion del mapa
  UNITS METERS
  IMAGECOLOR 255 255 255
  IMAGETYPE png # tipo de formato de salida

  WEB  # Inicio de Objeto WEB
    IMAGEPATH "/tmp/ms_tmp/"
    IMAGEURL "/ms_tmp/"
  END

  LEGEND # Incicio de Objeto Leyend
    TRANSPARENT TRUE
    KEYSIZE 15 15
    STATUS ON
    LABEL
            TYPE BITMAP
            SIZE large
            COLOR 0 0 0
    END
  END
  SCALEBAR
    TRANSPARENT TRUE
  END

  REFERENCE # Inicio del mapa de referencia
          IMAGE 'C:/datos/cuenca5.gif'  # Imagen para referencia
          EXTENT 713294.90 9675046.51 735511.76 9687183.73 #extensión en la cual se presenta
          SIZE 200 100
          STATUS ON
          MINBOXSIZE 3
          MAXBOXSIZE 100
```

```
            COLOR 120 0 0
            OUTLINECOLOR 0 0 0
            MARKERSIZE 3
            MARKER 'star'
 END

LAYER # Capa de Rios Principales
     NAME riosprin
      TYPE POLYGON
      STATUS ON
      DATA 'C:/datos/riosprin.shp'
      PROJECTION  # Proyeccion de la capa *
           "init=epsg:24877"
      END
 CLASS
  NAME 'Rios Principales'
  STYLE
   COLOR 0 197 255
   OUTLINECOLOR 115 178 255
  END #STYLE
 END #CLASS
 GROUP "Cuenca"  # Grupo de Capas al que pertenece
END #LAYER

  LAYER # Capa de manzanas
      NAME manzanas
      TYPE POLYGON
      STATUS ON
      DATA 'C:/datos/manzanas.shp'
      PROJECTION
           "init=epsg:24877"
      END
     COLOR 230 237 245
     OUTLINECOLOR 255 255 255
     GROUP "Cuenca"
 END

 LAYER  # capa areas verdes
       NAME verdes
        TYPE POLYGON
        STATUS ON
        DATA 'C:/datos/verdes.shp'
        PROJECTION
            "init=epsg:2317"
        END
   CLASS
    NAME 'Areas Verdes'
    STYLE
     COLOR 158 215 194
     END #STYLE
   END #CLASS
   GROUP "Cuenca"
END #LAYER

LAYER # Capa de Predios
            NAME predios
            TYPE LINE
            STATUS OFF
            DATA 'C:/datos/predios.shp'
            PROJECTION
```

```
            "init=epsg:24877"
        END
        CLASS
          NAME "Predios2"
          STYLE
              #COLOR 215 238 219
              OUTLINECOLOR 255 235 190
          END
        END
        MAXSCALEDENOM 5000
        MINSCALEDENOM 0
        GROUP "Predios"
END

LAYER
    NAME vialidad
     TYPE LINE
     STATUS ON
     DATA 'C:/datos/vias.shp'
     PROJECTION
        "init=epsg:24877"
     END
 LABELCACHE on
 LABELITEM 'NOMBRE_APE'
 CLASS
  NAME 'Vialidad'
  LABEL
        TYPE TRUETYPE
        FONT "activa"
        PARTIALS FALSE
     SIZE 8
     COLOR 84 84 84
     POSITION CC
     ANGLE AUTO
     BUFFER 10
     OUTLINECOLOR 233 233 210
     PRIORITY 10
  END
 END #CLASS
 GROUP "Vialidad"
END #LAYER

END  #Fin de Objeto MAP
```

Proyección de la capa *.- La proyección init=epsg: 24877 hace referencia a la proyección que usará la librería Proj.4 para la georeferenciar las capas, el código 24877 pertenece a PSAD56 UTM Zona 17 S

Luego de haber definido el archivo de mapa, lo podemos probar utilizando el CGI de mapserver, lo podemos hacer de la siguiente manera:
Ingresamos al navegador y accedemos a la siguiente dirección

http://localhost/cgi-bin/mapserv.exe?mode=map&mapsize=800+600&map=C:\ms4w\apps\ka-map-1.0\map\cuenca.map&LAYERS=all

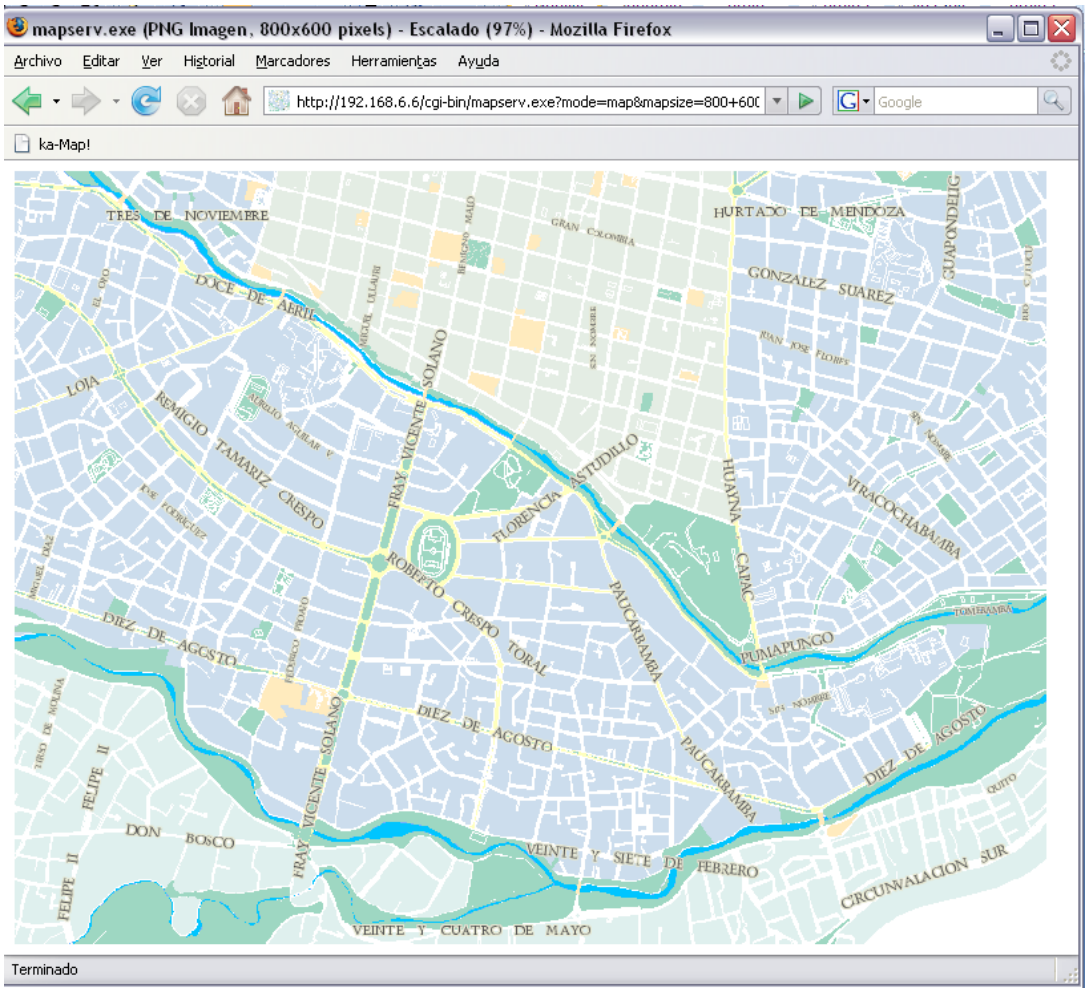
Con esta URL estamos invocando al CGI mapserv.exe con los siguientes parámetros:
Mode.- indica la forma en el que mapserver funciona, con el parámetro map le decimos a MapServer que genere una imagen en el servidor.

Mapsize.- indicamos el tamaño de visualización del mapa, si no especificamos éste parámetro la imagen resultante se visualizara del tamaño especificado en el Map File.
Map.- éste parámetro nos sirve para indicarle a mapserver en donde está ubicado el archivo Map File (.map) dentro del servidor.

Layers- en éste parámetro podemos pasar una lista de las capas que queremos mostrar, en el ejemplo con la palabra ALL le decimos que visualice todas las capas.

Y como resultado obtenemos la siguiente salida.

Fig. 41: Prueba del Map File



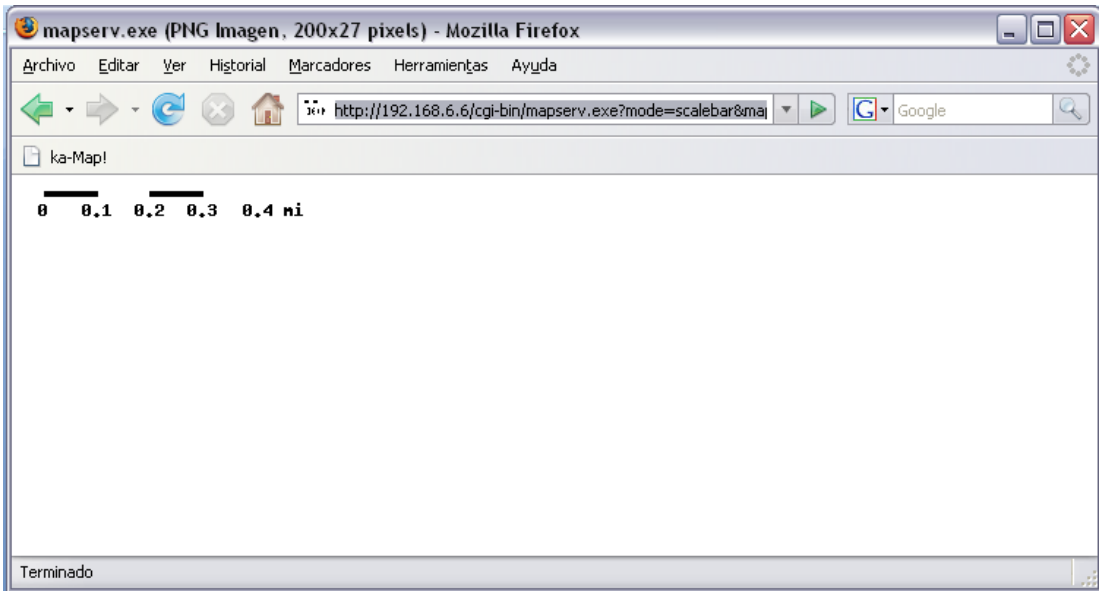
Fuente: Autor

Ahora obtendremos la imagen de la escala de la vista actual, solamente modificamos el parámetro mode= scalebar

http://192.168.6.6/cgi-bin/mapserv.exe?mode=scalebar&mapsize=800+600&map=C:\ms4w\apps\ka-map-1.0\map\cuenca.map&LAYERS=all

Y obtenemos la siguiente salida

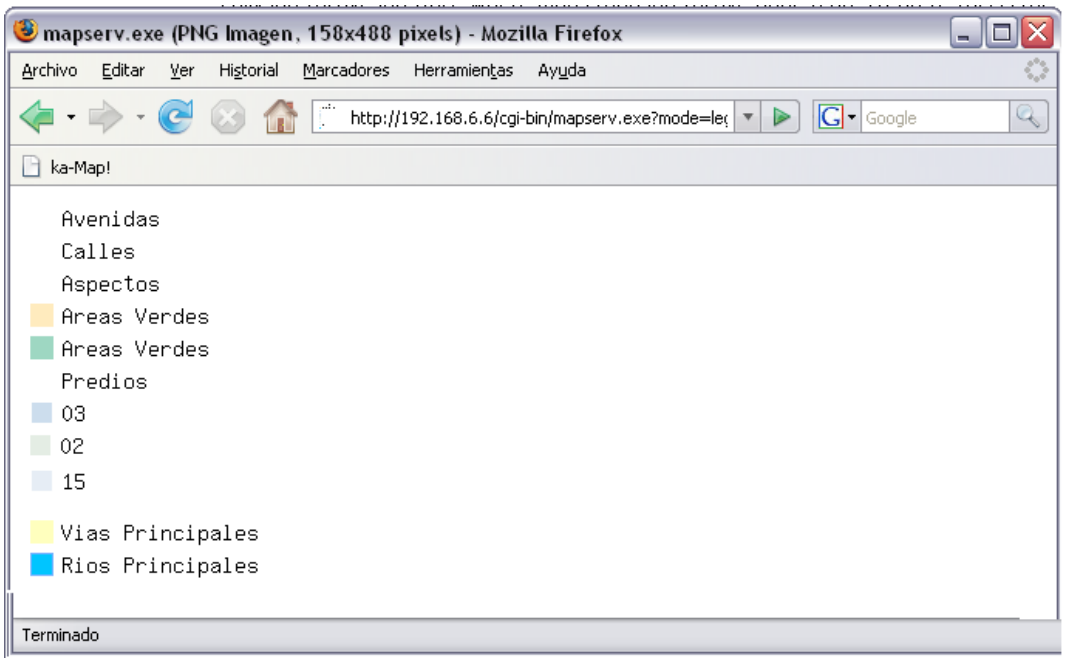
Fig. 42: Generar Barra de Escala



Fuente: Autor

Para Obtener la Leyenda mode= legend

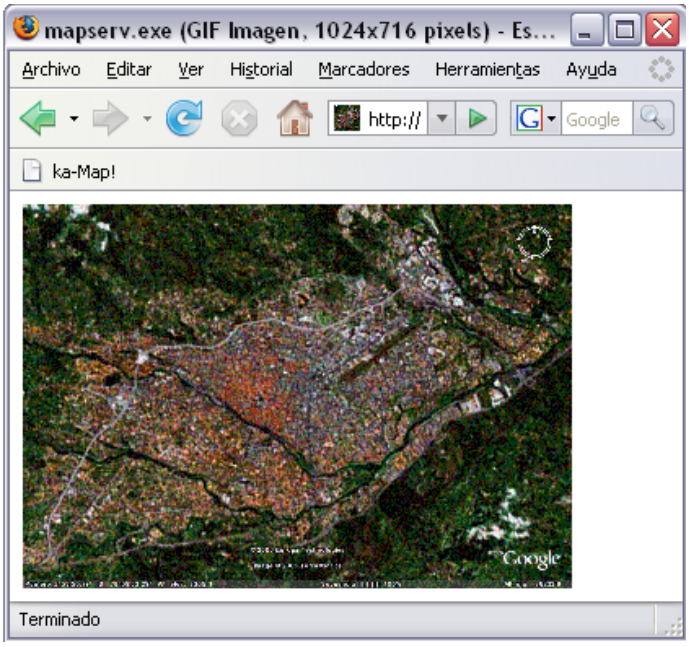
Fig. 43: Generar Leyenda



Fuente: Autor

Para Obtener el Mapa de referencia mode=reference

Fig. 44: Generar Mapa de Referencia



Fuente: Autor

De igual forma se puede usar mode=ZoomIn o mode= ZommOut para realizar funciones de zoom sobre el mapa resultante.

5.4 Base de Datos

5.4.1 PostgreSQL es un sistema de administración de base de datos objeto-relacional (ORDBMS, por sus siglas en inglés) basado en Postgres v4.2 desarrollado en la Universidad de California en el Departamento de Ciencias de la Computación de Berkeley.

PostgreSQL es un descendiente de código abierto de este código original de Berkeley. Soporta SQL92 y SQL99 y ofrece muchas características modernas:

- Consultas complejas
- Foreign keys
- Triggers
- Vistas
- Integridad transaccional
- Control de concurrencia multiversión.

A su vez, PostgreSQL puede ser extendido por el usuario en múltiples formas; por ejemplo, agregando nuevos tipos de datos, funciones, operadores, métodos de indexación, funciones de agregación y lenguajes procedurales.

Además, debido a la licencia libre, PostgreSQL puede ser usado, modificado y distribuido libre de cargos para cualquier propósito, sea privado, comercial o académico.

5.4.2 PostGIS.- es una extensión al sistema de base de datos PostgreSQL, que nos da la posibilidad de usar objetos georeferenciados, postGIS posee soporte para indexar estos datos geográficos y funciones para realizar análisis.
PostGIS: Es una extensión al sistema de base de datos objeto-relacional PostgreSQL. Permite el uso de objetos *GIS* (Geographic information systems).

Está publicado bajo licencia GNU, con PostGIS podemos usar todos los objetos que aparecen en la especificación OpenGIS como puntos, líneas, polígonos, multilíneas, multipuntos, y colecciones geométricas.

Lenguajes de consulta espacial

Las bases de datos espaciales no tienen un conjunto de operadores que sirvan como elementos básicos para la evaluación de consultas ya que estas manejan un volumen extremadamente grande de objetos complejos no ordenados en una dimensión. Es por esto que existen algoritmos complejos para evaluar predicados espaciales. Las consultas son realizadas generalmente en SSQL (Spatial SQL)

Las tres categorías fundamentales de consultas en un sistema de información espacial son:

- Consultas exclusivamente de propiedades espaciales. Ejemplo: "Traer todos los pueblos que son cruzados por un río".
- Consultas sobre propiedades no espaciales. Ejemplo: "Cuantas personas viven en Cuenca".
- Consultas que combinan propiedades espaciales con no espaciales. Ej.: "Traer todos los vecinos de un cuadra localizada en Parque Calderón"

Entre los tipos de funciones que podemos usar están:

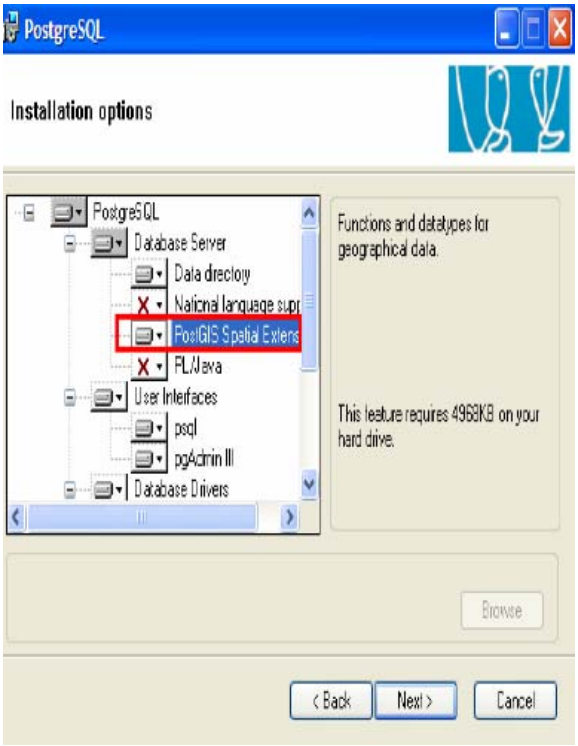
- distancia
- contigüidad
- contenido
- área
- longitud
- intersección
- unión
- buffer

Podemos encontrar una referencia a las funciones PostGIS en el ANEXO 2

5.4.3 Instalación

Podemos descargar de http://www.postgresql.org/, ejecutamos el archivo de instalación, escogemos instalación personalizada y activamos la instalación de la extensión PosGIS, y terminamos la instalación.

Fig. 45: Instalación PostgreSQL/PostGIS

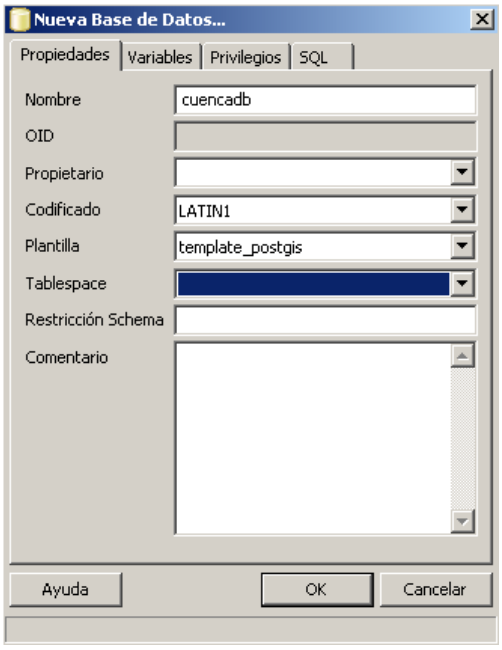


Fuente: Autor

Dentro de la carpeta /bin que se encuentra en el directorio en el que realizamos la instalación se encuentra el archivo pgAdmin3.exe que ejecuta una herramienta visual para la gestión de nuestra base de datos.

Para crear nuestra base de datos escogemos la opción Nueva Base de Datos, le damos un nombre y escogemos como plantilla a la base de datos que se creó con la instalación de PostGis llamada “template_postgis”, también podemos realizar esta tarea mediante SQL.

Fig. 46: Crear Base de Datos



Fuente: Autor

```
CREATE DATABASE cuencadb WITH TEMPLATE = template_postgis
ENCODING = 'LATIN1';
```

5.4.4 Creación de Tablas

De igual forma podemos crear nuestras tablas mediante sentencias de creación de datos SQL, o usando el asistente gráfico.

```
CREATE TABLE categoria (
    idcategoria integer NOT NULL,
    "desCat" character varying(50),
    descategoria character varying(40),
    ver "char" DEFAULT '0'::"char"
);


CREATE TABLE subcategoria (
    idsubcategoria integer NOT NULL,
    dessubcategoria character varying(40),
    idcategoria integer,
    ver "char" DEFAULT '0'::"char"
```

```
);

CREATE TABLE informacion (
    gid integer NOT NULL,
    codigo integer,
    nombre character varying(80),
    descripcion character varying(200),
    direccion character varying(250),
    telefono character varying(25),
    email character varying(100),
    link character varying(100),
    foto character varying(100),
    foto_mini character varying(100),
    html text,
    tags text,
    idsubcat smallint,
    the_geom geometry
);
```

Creamos un tipo personalizado para devolver en las búsquedas

```
CREATE TYPE desc_results AS (
        gid integer,
        nombre character varying(80),
        descripcion character varying(200),
        descripcionbr character varying(200),
        direccion character varying(250),
        telefono character varying(25),
        email character varying(100),
        link character varying(100),
        foto character varying(100),
        foto_mini character varying(100),
        html text,
        idsubcat smallint,
```

```
        x double precision,
        y double precision,
);
```

Creamos dos funciones para buscar lugares en la tabla información y que devuelvan filas del tipo desc_results

```
CREATE FUNCTION buscarnombre(text) RETURNS SETOF desc_results
   AS $_$
SELECT gid,nombre,descripcion,headline('simple',descripcion,q) as
descripcionbr,direccion,telefono,email,link,foto,foto_mini,html,idsubcat,X(the_geom
) as x, Y(the_geom) as y
  FROM informacion
  WHERE nombre like “%text%”;
$_$
   LANGUAGE sql;

CREATE FUNCTION buscarclave(text) RETURNS SETOF desc_results
   AS $_$
SELECT gid,nombre,descripcion,headline('simple',descripcion,q) as
descripcionbr,direccion,telefono,email,link,foto,foto_mini,html,idsubcat,X(the_geom
) as x, Y(the_geom) as y
  FROM informacion
  WHERE tags like ‘%text%’ DESC;
$_$
   LANGUAGE sql;
```

5.4.5 Importar Archivos Shape

Para realizar las búsquedas vamos a necesitar crear las tablas de predios y vías correspondientes a los archivos predios.shp y vías.shp.

El procedimiento es el siguiente. Tomando en cuenta que nuestra base de datos se llama cuencadb y que el nombre del shape es vias.shp y que queremos generar la

tabla vías en la base de datos. Lo primero que tienen que hacer usar la utilidad shp2pgsql.

cd C:\Archivos de programa\PostgreSQL\8.2\bin
shp2pgsql -s 24877 c:\datos\vias.shp vias> C:\vias.sql

de igual forma para el archivo de predios
shp2pgsql -s 24877 c:\datos\predios.shp vias> C:\predios.sql

Se ha usado el código 24877 para especificar la proyección EPGS: 24877

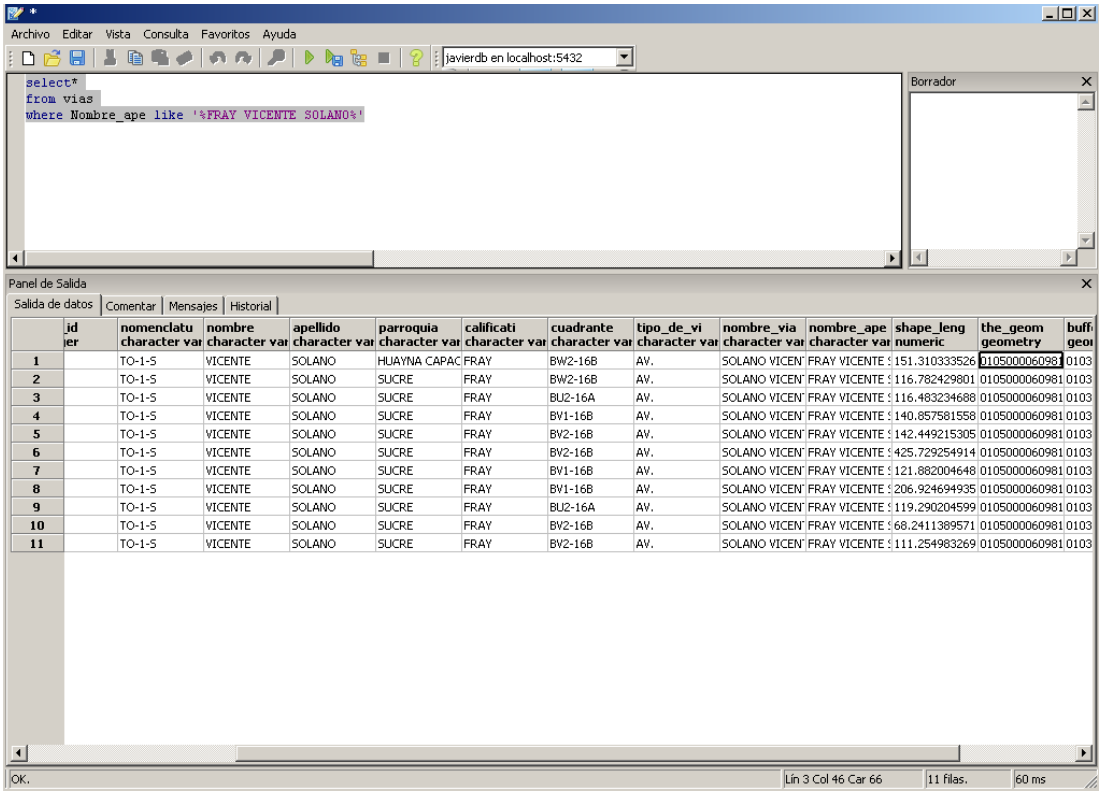
PostGIS SRID:24877
+proj=utm +zone=17 +south +ellps=intl +units=m +no_defs

Después de haber ejecutado estos comandos, tenemos creadas las tablas de vías y predios, cada una con una columna que representa la geometría de cada fila.

Podemos realizar una prueba para comprobar que todo esté correcto.
Realizamos la siguiente consulta.

Select *
From vías
Where Nombre_ape like '%FRAY VICENTE SOLANO%'

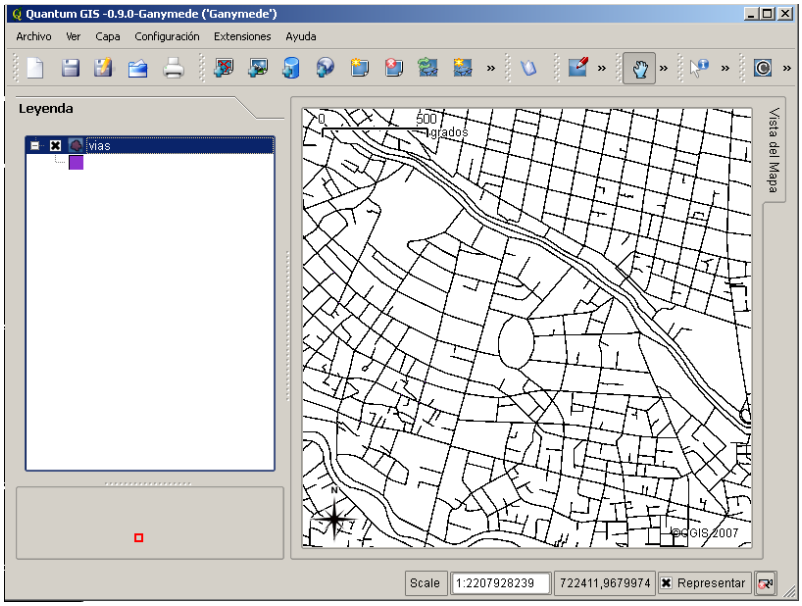
Fig. 47: Consulta en PgAdmin



Fuente: Autor

O probamos con algún software con soporte para PostGIS como en la siguiente imagen se ha creado una conexión a la tabla vías en el software QGis.

Fig. 48: Visualizar Tabla PostGIS en QGis



Fuente: Autor

5.5 Aplicación Web

El siguiente paso después de haber creado nuestro mapa, es desarrollar la aplicación web que interactúe con el usuario ofreciéndole herramientas gráficas para navegación en el mapa y para las consultas y búsquedas.

Para éste propósito se usaran librerías javaScript que interactúen con las librerías PhpMapScript, archivos Php y un archivo HTML que va a ser el archivo al que el usuario va a acceder desde el navegador.

5.5.1 MapScript

Es una colección de scripts que interactúan directamente con el API de MapServer, estos scripts contienen funciones escritas en un cierto lenguaje, en nuestro caso usaremos PhpMapScript 5.0, El módulo de MapScript permite hacer muchas operaciones en datos espaciales a través de MapServer, incluyendo funciones de lectura/escritura a archivos shape, a bases de datos, cambios de proyecciones, y muchas otras funcionalidades.

5.5.2 AJAX (Asynchronous JavaScript and XML)

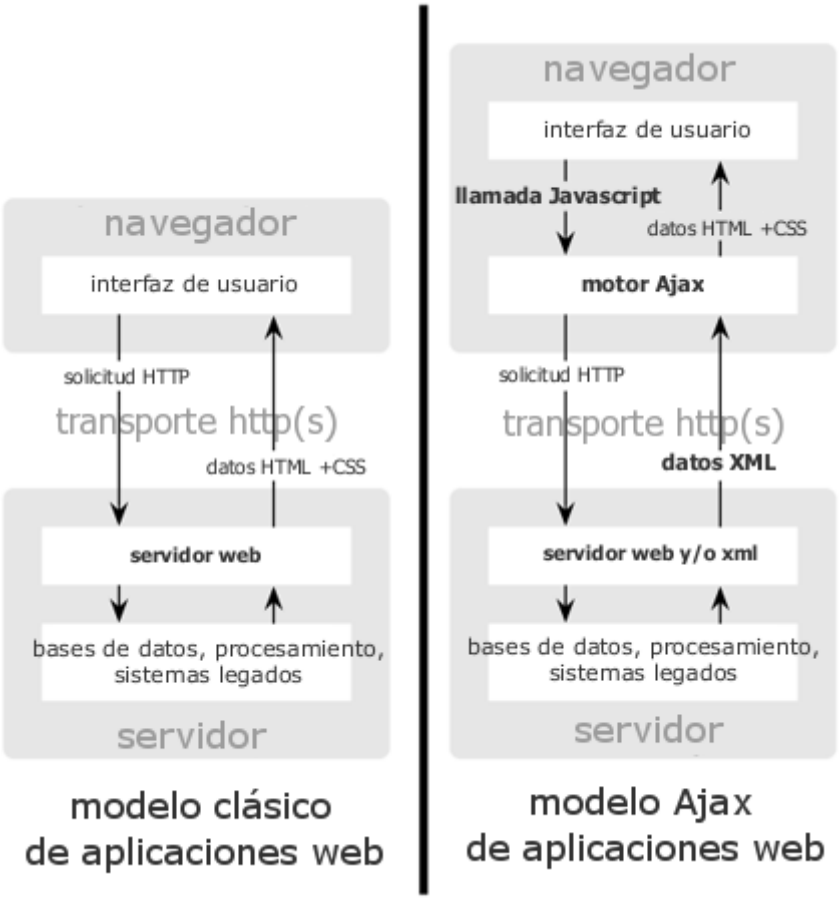
En un conjunto de tecnologías (JavaScript+CSS+DOM+XMLHttpRequest) para el desarrollo de aplicaciones web que ofrecen al usuario una interacción más dinámica, por ejemplo en una aplicación web normal al llenar y enviar un formulario se envía la información al servidor y éste devuelve toda una página de respuesta que es cargada nuevamente por el navegador, usando AJAX se realiza una petición asíncrona mediante javaScipt y el resultado devuelto por el servidor solamente actualiza la parte que le corresponde , así se gasta menos ancho de banda, es más rápido y más eficiente, los procesos HTTP requeridos se sustituyen con un motor AJAX escrito en JS.

Las tecnologías que usa Ajax son:

- Presentación.- XHTML and CSS;
- Pantallas dinámicas e interacción usando Document Object Model;
- Manipulación e intercambio de datos usando XML and XSLT;
- Recuperación asíncrona de datos usando XMLHttpRequest;

- JavaScript enlaza toda la aplicación.

Fig. 49: Comparación modelo clásico / modelo Ajax



Fuente: Garrett-2005

5.5.3 Página Principal

Después de haber integrado las funciones MapScript con las funciones en javaScript obtenemos la interfaz del usuario con las herramientas que mediante funciones javaScript acceden a las funciones MapScript que ejecutan las funciones de zoom, pan, escala, referencia dentro de la aplicación que el navegador está ejecutando.

El archivo de interfaz del usuario es el siguiente.

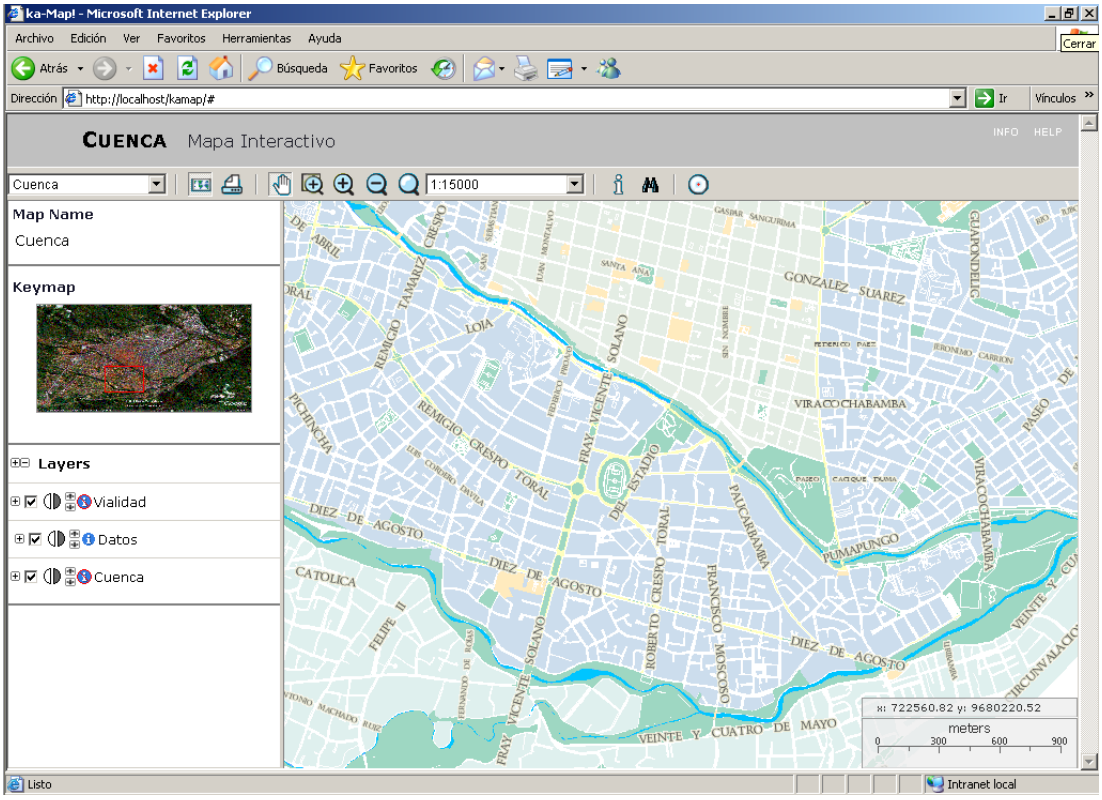
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>ka-Map!</title>
<script type="text/javascript" src="DHTMLapi.js"></script>
```

```
<script type="text/javascript" src="xhr.js"></script>
<script type="text/javascript" src="Map.js"></script>
<script type="text/javascript" src="Keymap.js"></script>
<script type="text/javascript" src="Legend.js"></script>
<script type="text/javascript" src="Tool.js"></script>
<script type="text/javascript" src="Query.js"></script>
<script type="text/javascript" src="tools/rubberzoom/RubberZoom.js"></script>
<script type="text/javascript" src="MouseTracker.js"></script>
<script type="text/javascript" src="scalebar/scalebar.js"></script>
<link rel="stylesheet" type="text/css" href="scalebar/scalebar-thinner.css">
<link href="tools/Explorer/screen.css" rel="stylesheet" type="text/css" media="all">
<link href="tools/Explorer/tools.css" rel="stylesheet" type="text/css" media="all">
<link href="tools/tooltip/tooltip.css" rel="stylesheet" type="text/css" media="all">
</head>
<body onload="myOnLoad();">
<div id="page">
<div id="explorer">
        <div class="kmTitle">Cuenca</div>
        <div class="kmSubtitle">Mapa Interactivo</div>
        <div class="minimenu"><a href="#"
onClick="showContent('info_no_html.html');"><span>info</span></a> <a href="#"
onClick="showContent('help_no_html.html');"><span>help</span></a></div>
        <form name="toolbar">
        <div id="toolbar">
            <img alt="Separator" class="separator"
src="images/icon_set_explorer/separator.png">
            <img id="toolMapinfo" onclick="switchService(this.id)" title="Click to see the Map Info"
alt="Click to see Map Info" src="images/a_pixel.gif">
            <img id="toolPrint" onclick="switchService(this.id)" title="Click to Print this
View" alt="Print" src="images/a_pixel.gif">
            <img alt="Separator" class="separator" src="images/icon_set_explorer/separator.png">
            <img id="toolPan" onclick="switchMode(this.id)" title="Click and drag to Navigate the
Map" alt="Click and drag to Navigate the Map" src="images/a_pixel.gif">
            <img id="toolZoomRubber" onclick="switchMode(this.id)" title="rubber zoom"
alt="rubber zoom" src="images/a_pixel.gif" >
            <img id="toolZoomIn" onclick="myZoomIn()" title="zoom in" alt="zoom in"
src="images/a_pixel.gif" onMouseDown="this.id='toolZoomInDown';"
onMouseUp="this.id='toolZoomIn';">
            <img id="toolZoomOut" onclick="myZoomOut()" title="zoom Out" alt="zoom Out"
src="images/a_pixel.gif" onMouseDown="this.id='toolZoomOutDown';"
onMouseUp="this.id='toolZoomOut';">
            <img id="toolZoomFull" onclick="getFullExtent()" title="zoom to full extents" alt="zoom
to full extents" src="images/a_pixel.gif" onMouseDown="this.id='toolZoomFullDown';"
onMouseUp="this.id='toolZoomFull';">
            <select name="scales"
onchange="mySetScale(this.options[this.selectedIndex].value)"></select>
            <img alt="Separator" class="separator" src="images/icon_set_explorer/separator.png">
            <img id="toolQuery" onclick="switchMode(this.id);switchService(this.id);" title="Click
and drag or double click to query the Map" alt="Click and drag or double click to query the Map"
src="images/a_pixel.gif" >
            <img id="toolSearch" onclick="switchService(this.id);" title="Search places in the
database" alt="Search places in the database" src="images/a_pixel.gif" >
            <img alt="Separator" class="separator" src="images/icon_set_explorer/separator.png">
            <img id="toolLink" onclick="switchService(this.id);" title="Send this view to a friend"
alt="Send this view to a friend" src="images/a_pixel.gif" >
        </div>
        </form>
</div>
<div id="layoutFrame">
        <div id="service">
```

```
                <div id="mapInfo">
                        <h5>Map Name</h5>
                        <div id="mapTitle"></div>
                        <hr>
                        <h5>Keymap</h5>
                        <div id="keymap"></div>
                        <div id="legend"></div>
                        <div id="GNmetadata"></div>
                        <hr>
                </div><!-- end mapInfo -->
                <div id="curtain">
                        <h5>The application is loading.<br> Please wait a moment...</h5>
                </div><!-- end curtain -->
        </div><!-- end service -->
        <div id="viewport">
                <div id="scaleReference">
                <div id="scaleBackground" class="transparentBackground"></div>
                <div id="scalebar"></div>
                <!--<div id="scale">current scale</div>-->
                </div>
                        <div id="geoPositionReference">
                        <div id="geoPositionBackground" class="transparentBackground"></div>
                                <h5>Coords</h5>
                                        <div id="geoPosition">
                                                x: 0.00
                                                y: 0.00
                                        </div>
                        </div>
</div><!-- end viewport -->
</div><!-- end layoutFrame -->
</div><!-- end page -->
</div>
</body>
</html>
```

En la página principal hemos realizado el diseño de la aplicación a la que el usuario va a acceder a través de un navegador, ésta página está compuesta de etiquetas “div” las cuáles actualizaremos asíncronamente mediante funciones javaScript y su comportamiento visual manejado mediante hojas de estilo. La página resultante se muestra en la siguiente imagen.

Fig. 50: Página principal en el Navegador



Fuente: Autor

5.5.4 Búsqueda de Direcciones

Para realizar la búsqueda de direcciones usaremos las tablas de predios y vías que están creadas en la base de datos desde los archivos tipo Shape.

La consulta para buscar la intersección entre dos vías sería la siguiente.

```
Select  principal,interseccion, X(Centroid(Punto)) as X, Y(Centroid(Punto)) as Y," as numero
from
(
  select v1.nombre_ape as principal, v2.nombre_ape as interseccion,intersection( v1.the_geom,
v2.the_geom) as Punto
  from vias v1, vias v2
  where v1.nombre_ape like '%SOLANO%'
  and v2.nombre_ape like '%REMIGIO%'
  and v1.the_geom && v2.the_geom
  and intersects(v1.the_geom,v2.the_geom)
```

```
)A order by principal,interseccion
```

Con lo cuál obtenemos la siguiente salida:

"FRAY VICENTE SOLANO";"REMIGIO TAMARIZ CRESPO";721762.311857793;9678973.70778672;""

Esta salida nos da las coordenadas X y Y para ubicar el punto de la intersección.

De igual forma para buscar una dirección con su número y calle principal la consulta sería de la siguiente forma

```
select  a.principal, X as X, Y as Y, numero as numero
from
(
  select v.nombre_ape as principal, p.numero,p.APELLIDOS , X(Centroid(intersection(v.the_geom,
p.the_geom))) as X, Y(Centroid(intersection(v.the_geom, p.the_geom))) as Y
  from predios p, vias v
  where p.numero='17079' and v.nombre_ape like '%CORDOVA%'
  and v.the_geom && p.the_geom and intersects(v.the_geom, p.the_geom)
)A order by a.principal,interseccion
```

Búsqueda.php

El script en PHP que va a realizar las búsquedas es el siguiente:

```
<?php
$v_principal = iconv("UTF-8", "ISO-8859-13", $_REQUEST['principal']);
$v_interseccion = iconv("UTF-8", "ISO-8859-13", $_REQUEST['interseccion']);
$numero=$_REQUEST['numero2'];



        $conexion= pg_pconnect("host=localhost port=5432 dbname=cuencadb user=
                                        postgres password=postgres");
        if (!conexion)
        {
          echo "Error en conexion a la Base de Datos";
```

```
            exit;
        }
if( $numero=="" and $v_interseccion<>"")
{
        $sql="select  principal,interseccion, X(Centroid(Punto)) as X, Y(Centroid(Punto)) as Y," as
numero
        from
        (
          select v1.nombre_ape as principal, v2.nombre_ape as interseccion,intersection(
v1.the_geom, v2.the_geom) as Punto
          from vias v1, vias v2
          where v1.nombre_ape like '%".$v_principal."%'
          and v2.nombre_ape like '%".$v_interseccion."%'
          and v1.the_geom && v2.the_geom
          and intersects(v1.the_geom, v2.the_geom)
        )A order by principal,interseccion";
        $resultado = pg_Exec($conexion,$sql);
        prin_inter($v_principal,$v_interseccion,$numero,$resultado);

}else
{
        if ($numero<>"" and empty($v_interseccion))
        {
                $sql="select  a.principal, " as interseccion, X as X, Y as Y, numero as numero from (
select v.nombre_ape as principal, p.numero,p.APELLIDOS , X(Centroid(intersection(v.the_geom,
p.the_geom))) as X, Y(Centroid(intersection(v.the_geom, p.the_geom))) as Y from predios p, vias v
where p.numero='".$numero."' and v.nombre_ape like '%".$v_principal."%' and v.the_geom &&
p.the_geom and intersects(v.the_geom, p.the_geom) )A order by a.principal,interseccion";
        $resultado = pg_Exec($conexion,$sql);
        prin_numero($v_principal,$numero,$resultado);
        }
        else
        {
          $sql="select  principal,interseccion, X as X, Y as Y,numero as numero from ( select
v1.nombre_ape as principal,p.numero as numero, v2.nombre_ape as interseccion,
X(Centroid(intersection(v1.the_geom, p.the_geom))) as X, Y(Centroid(intersection(v1.the_geom,
p.the_geom))) as Y from vias v1, vias v2, predios p  where v1.nombre_ape like '%".$v_principal."%'
and v2.nombre_ape like '%".$v_interseccion."%' and v1.the_geom && v2.the_geom and
intersects(v1.the_geom, v2.the_geom) and p.numero='".$numero."' and v1.the_geom && p.the_geom
```

```
and intersects(v1.the_geom, p.the_geom) )A group by principal,interseccion, X, Y,numero order by
principal,interseccion";
            $resultado = pg_Exec($conexion,$sql);
            prin_numero_inter($v_principal,$v_interseccion,$numero,$resultado);

        }
}
pg_close($conexion);
exit;
/*********funcones*************************/
        function prin_inter($v_principal,$v_interseccion,$numero,$resultado){
                        $filas = pg_NumRows($resultado);
                        $vPrincipal="";
                        $vInterseccion="";
                        $vX=0;
                        $vY=0;
                        $num_reg=0;
                        for($j=0;$j<$filas;$j++)
                        {
                         $x=pg_result($resultado,$j,2);
                         $y=pg_result($resultado,$j,3);
                         $principal=pg_result($resultado,$j,0);
                         $interseccion=pg_result($resultado,$j,1);
                         echo "$principal|$interseccion|$numero|$x|$y\n";
                        }
        }

        function prin_numero($v_principal,$numero,$resultado){
        $filas = pg_NumRows($resultado);
        $vPrincipal="";
        $vX=0;
        $vY=0;
        $num_reg=0;
        for($j=0;$j<$filas;$j++)
        {
         $x=pg_result($resultado,$j,2);
         $y=pg_result($resultado,$j,3);
         $principal=pg_result($resultado,$j,0);
         echo "$principal|$interseccion|$numero|$x|$y\n";
        }
```

```
        }
        function prin_numero_inter($v_principal,$v_interseccion,$numero,$resultado){
                        $filas = pg_NumRows($resultado);
                        $vPrincipal="";
                        $vInterseccion="";
                        $vX=0;
                        $vY=0;
                        $num_reg=0;
                        for($j=0;$j<$filas;$j++)
                        {
                         $x=pg_result($resultado,$j,2);
                         $y=pg_result($resultado,$j,3);
                         $principal=pg_result($resultado,$j,0);
                         $interseccion=pg_result($resultado,$j,1);
                         echo "$principal|$interseccion|$numero|$x|$y\n";
                         }
                        }
        }
?>
```

Buscar.js

La función en JavaScript que realiza la llamada asíncronamente al archivo PHP es el siguiente.

```
kaSearch.prototype.searchAddress=function(search_query,search_query2, numero,numero2 ){
if (search_query.length <= 0){
    alert("Ingrese calle Principal!");
    return;
  }
  if (search_query2.length <= 0 && (numero.length <= 0 || numero2.length <= 0) ){
    alert("Ingrese calle de Interseccion o Numero");
    return;
  }
  if (search_query.length > 0 && (search_query2.length> 0 || (numero.length > 0 && numero2.length
> 0) ))
  {
     element = document.getElementById ('search-results-list');
     element.innerHTML = "<h3>Processing search.<br>Please wait...</h3><hr>";
```

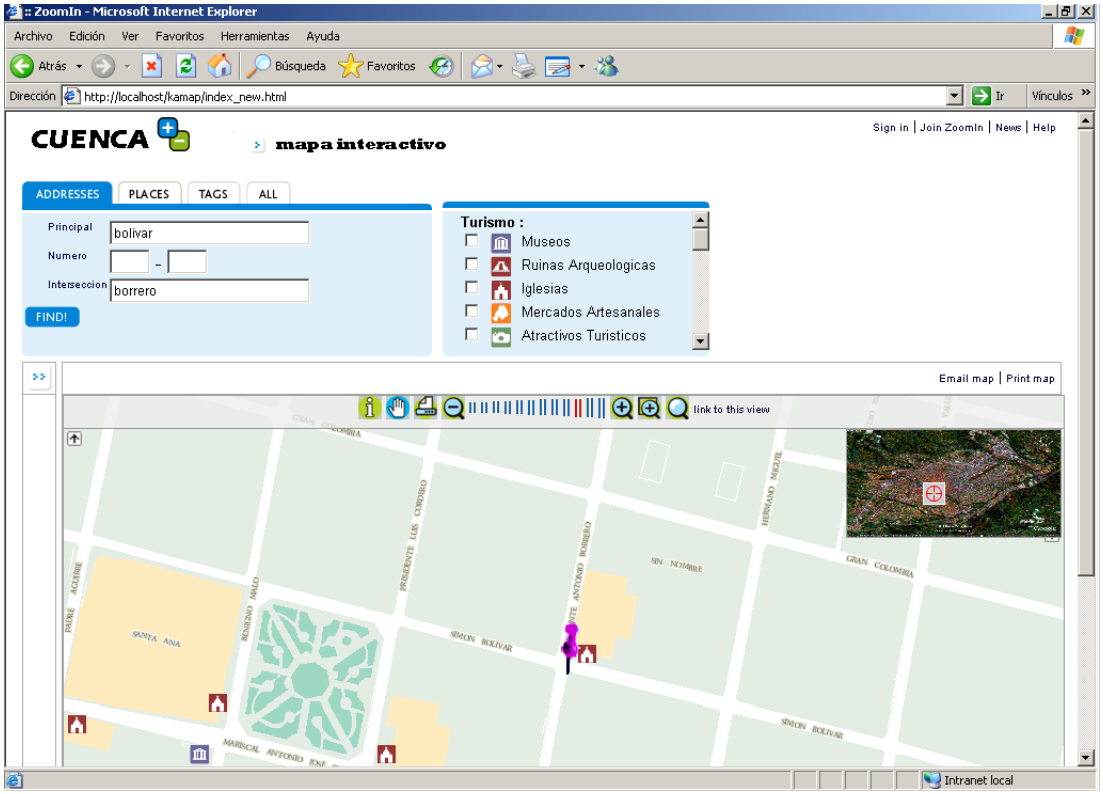
```
    var url
="tools/busqueda/busqueda.php?xmlRequest=true&principal="+searchstring1+"&interseccion="+sear
chstring2+"&numero="+numero+"&numero2="+numero2;
            try {
                    httpReq.open("GET", url, false);
                    if ( isIE ) {
                      httpReq.onreadystatechange = this.setPointsSearch;
                    }
                    else {
                      httpReq.onload = this.setPointsSearch;
                    }
                    httpReq.send(null);
            } catch (e) {
                    alert("searchAddress"+e);
            }
   }
};
```

Es la función que procesa y muestra el resultado en el mapa

```
kaSearch.prototype.setPointsSearch= function() {
  if ( httpReq.readyState == 4 ) {
    for (var i = 0; i < lines.length - 1; i++) {
      cols = lines[i].split("|");
      var icon                          = document.createElement("img");
      var principal             = cols[0];
      var interseccion          = cols[1];
      var numero                = cols[2];
      var x= cols[3];
      var y= cols[4];
      var imgn = document.createElement( 'img' );
       imgn.src = 'img/pin2.gif';
       imgn.style.left='-5px';
       imgn.style.top='-40px';
       imgn.style.position='absolute';
       div.appendChild(imgn);
       myKaMap.addObjectGeo( canvasSearch, x, y, div )
   if (isIE) {
        imgn.setAttribute("onclick", function() { toolTip.showTip(this.vx,this.vy,this.vid); });
```

```
    } else {
      imgn.setAttribute("onclick", "toolTip.showTip(this.vx,this.vy,this.vid)");
    }
  if(i==0)
          resultado="<tr><td colspan='2'>No se han encontrado
          resultado= resultado +"</table>"
          element = document.getElementById ('search-results-list');
         element.innerHTML = "";
          element.innerHTML=resultado;
  }
};
```

Fig. 51: Búsqueda de Dirección



Fuente: Autor

5.5.5 Búsqueda de Lugares de Interés.

Para la búsqueda de lugares por nombre y por clave, utilizaremos las funciones que creamos

Buscarnombre (text).- para la búsqueda de lugares por su nombre

Buscarclave (text).- para la búsqueda de lugares por palabras clave

BusquedaLugar.php

Este archivo realiza la búsqueda en la base de datos invocando a las funciones.

```
<?php
$cadena = iconv("UTF-8", "ISO-8859-13", $_REQUEST['cad']);
$tipo = iconv("UTF-8", "ISO-8859-13", $_REQUEST['type']);
$conexion= pg_pconnect("host=localhost port=5432 dbname=javierdb user=postgres
password=postgres");
if (!conexion)
{
  echo "Error en conexion a la Base de Datos";
  exit;
}
if( $tipo=="0" ) ///Busqueda por Nombre de Negocio
{
        $sql="select * from buscarNombre('$cadena')";
        $resultado = pg_Exec($conexion,$sql);
}else ///Busqueda por palabra clave
{
        $sql="select * from buscarclave('$cadena')";
                $resultado = pg_Exec($conexion,$sql);
        }

$filas = pg_NumRows($resultado);
$num_reg=0;
for($j=0;$j<$filas;$j++)
{
        $id=pg_result($resultado,$j,0);
        $nombre=pg_result($resultado,$j,1);
        $descripcion=pg_result($resultado,$j,2);
        $direccion=pg_result($resultado,$j,4);
        $telefono=pg_result($resultado,$j,5);
        $mail=pg_result($resultado,$j,6);
        $link=pg_result($resultado,$j,7);
        $foto_mini=pg_result($resultado,$j,9);
        $idSubCat=pg_result($resultado,$j,11);
        $x=pg_result($resultado,$j,12);
        $y=pg_result($resultado,$j,13);
echo"$id|$nombre|$descripcion|$direccion|$telefono|$mail|$link|$foto_mini|$idSubCat|$x|$y\n";
}
pg_close($conexion);
```

```
exit;
?>
```

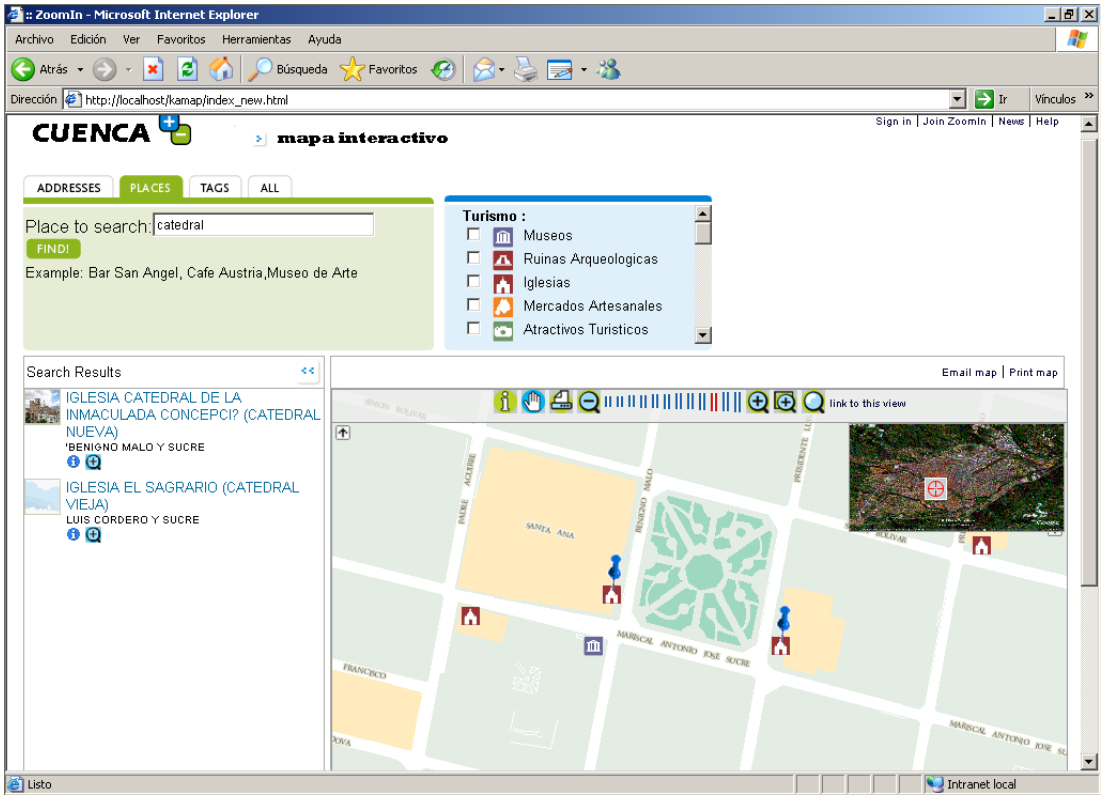
Búsqueda.js

Es la función que realiza la llamada asíncrona a BusquedaLugar.php

```
kaSearch.prototype.searchLugar=function(search_query, type ){
if (search_query.length <= 0){
      alert("Debe ngresar la cadena a buscar");
      return;
   }
   element = document.getElementById ('search-results-list');
   element.innerHTML = "<h3>Processing search.<br>Please wait...</h3><hr>";
   var url
="tools/busqueda/busquedaLugar.php?xmlRequest=true&cad="+searchstring+"&type="+type;
          try {
                              httpReq.open("GET", url, false);
                              if ( isIE ) {
                                httpReq.onreadystatechange = this. setPointsSearch;
                              }
                              else {
                                httpReq.onload = this. setPointsSearch;
                              }
                              httpReq.send(null);
           } catch (e) {
                    alert("searchItems"+e);
  }
};
```

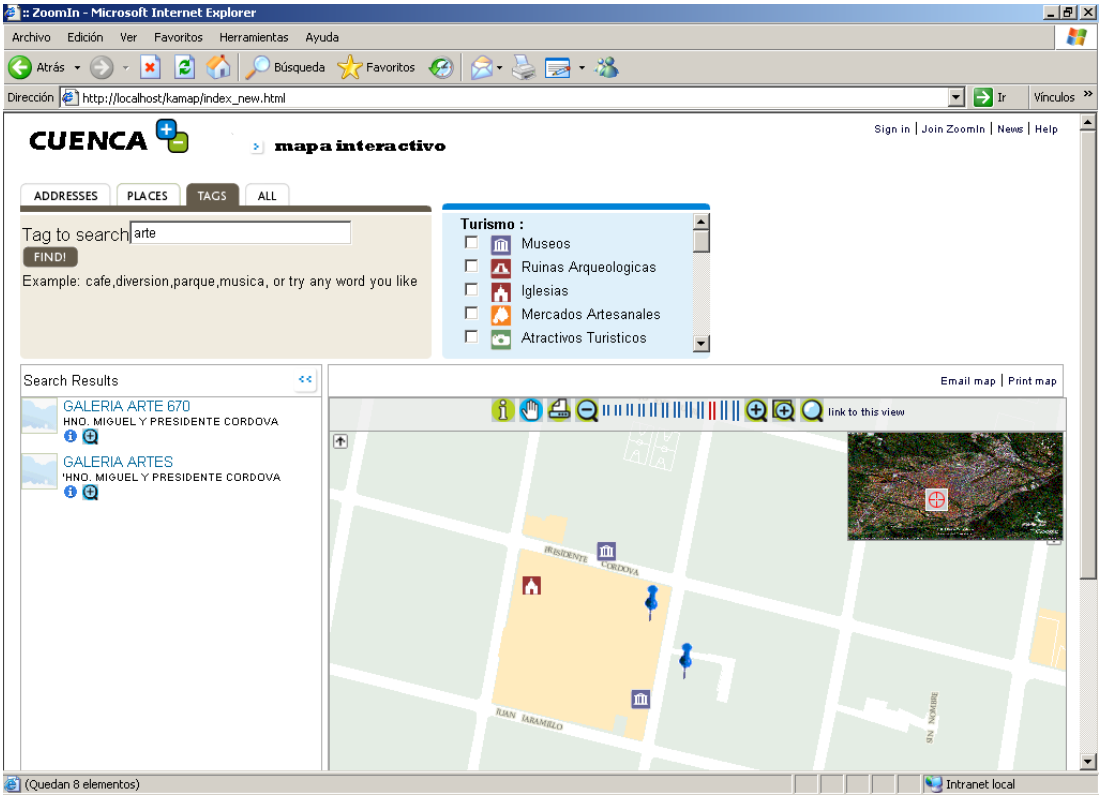
Para ubicar los resultados en el mapa usaremos la misma función “setPointsSearch” que usamos para la búsqueda de direcciones.

Fig. 52: Resultado Búsqueda Lugar por Nombre



Fuente: Autor

Fig. 53: Resultado búsqueda lugar por palabra clave



Fuente: Autor

5.5.6 Consulta de Lugar en el Mapa

Si observamos la función kaSearch.prototype.setPointsSearch podemos notar que la imagen que se crea como resultado realiza una llamada en el evento "onclick" a la función "toolTip.showTip (this.vx, this.vy, this.vid)" que es la que muestra la información de la imagen en la que se hizo clic.

MyToolTip.js

```
kaToolTip.prototype.showTip=function(x,y,id){
        this.moveDetail(x,y, id,1);
}
```

Y ésta llama a la función moveDetail()

```
kaToolTip.prototype.moveDetail = function(){
        var x = parseFloat(arguments[0]);
        vary = parseFloat(arguments[1]);
        var aPix = myKaMap.geoToPix( x, y )
```

```
        this.id= parseInt(arguments[2]);
        this.type= parseInt(arguments[3]);
        this.coordX = x;
        this.coordY = y;
        var aPix = myKaMap.geoToPix( x, y );
        var url ="scripts/getDetail.php?id="+this.id+"&type="+this.type;
        try {
                httpReq.open("GET", url, false);
                if ( isIE ) {
                  httpReq.onreadystatechange = this.setDetail;
                }
                else {
                  httpReq.onload = this.setDetail;
                }
                httpReq.send(null);
  } catch (e) {
          alert("kaToolTip.prototype.moveDetail:"+e);
  }
}
```

La función moveDetail () realiza la llamada asíncrona a el archivo getdetail.php para mostrar los resultados en la página principal.

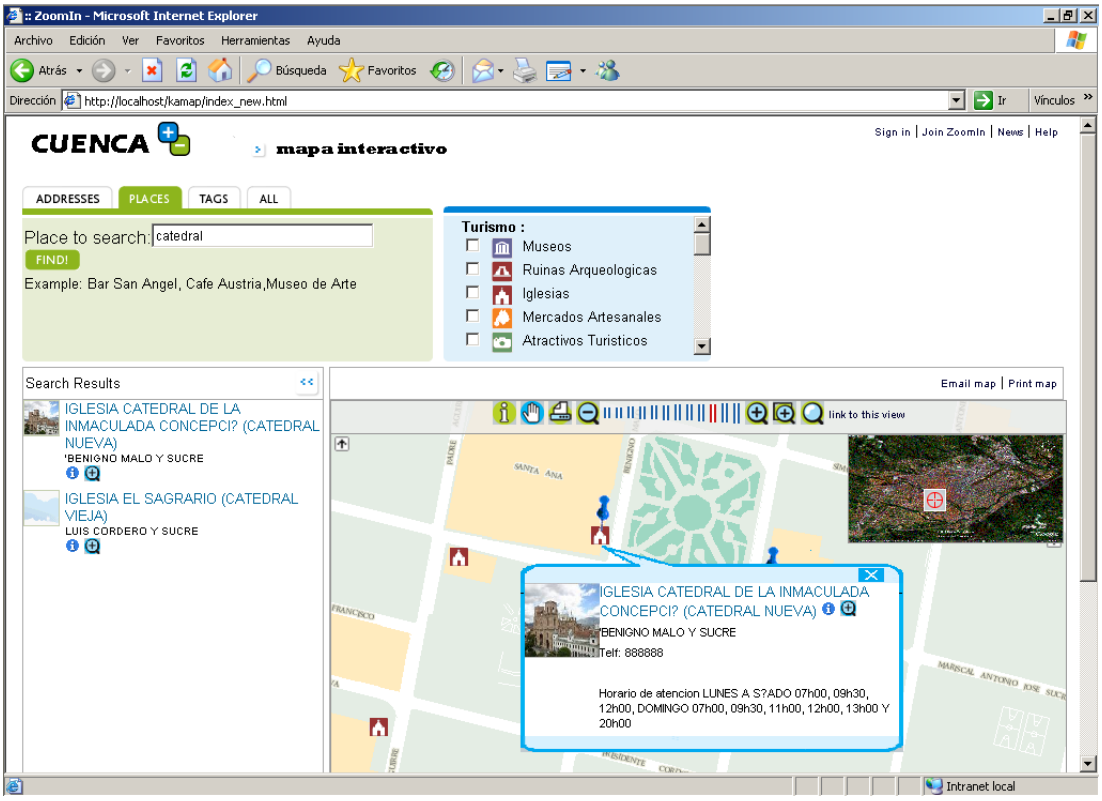
getDetail.php

```
<?php
	$v_id = iconv("UTF-8", "ISO-8859-13", $_REQUEST['id']);
	$conexion= pg_pconnect("host=localhost port=5432 dbname=javierdb user=postgres
password=postgres");
	if (!conexion)
	{
	  echo "Error en conexion a la Base de Datos";
	  exit;
	}
	$sql="select  i.gid, i.nombre,i.direccion , i.telefono, i.link,i.descripcion, i.foto_mini,
X(the_geom) as x, Y(the_geom) as y from informacion i where i.gid=$v_id";
	$resultado = pg_Exec($conexion,$sql);
	$filas = pg_NumRows($resultado);
	if($filas==1){
		$id=pg_result($resultado,0,0);
		$nombre=pg_result($resultado,0,1);
		$direccion =pg_result($resultado,0,2);
		$telefono =pg_result($resultado,0,3);
		$link =pg_result($resultado,0,4);
		$descripcion =pg_result($resultado,0,5);
		$foto_mini =pg_result($resultado,0,6);
		$x =pg_result($resultado,0,7);
		$y =pg_result($resultado,0,8);
	}
	if($foto_mini=="")
		$foto_mini="fotos/default_place_image.gif";

echo "<table width='350px'  height='100%'    align='center' cellpadding='0' cellspacing='0'
bgcolor='#FFFFFF'>
		<tr>
			<td width='50px' align='center' >
				<img src='$foto_mini'  width='70px' height='70px'
title='place photo'  />
			</td>
			<td   height='100%' width='100%' align='left'>
				<div class='detailTitle' >
					<a  class='detailTitle'  title='click to see full details for this
place' >
```

```
                                                    $nombre </a>
                                                    <img id=general-button alt='Click para ver mas
informacion de este lugar' title='Click para ver mas informacion de este lugar'
src='images/icon_query_on.png'onclick='$onclickInfo' />
                                                    <img alt='Acercar en el mapa'  title='Acercar en
el mapa'  src='images/icon_set_nomad/tool_zoomin_2.png' id=general-button onclick='$onclick'
width='15' height='15'>
                                        </div>
                                        <div class='detailItem'>$direccion</div>
                                        <div class='detailItem' >Telf: $telefono </div>
                                <div  class='detailItem'><a  class='detailTitle'  title='click to see this
place page' href='http://www.comelon.com'>
                                                    $link</a></div>
                                <div class='detailItem' > $descripcion </div></td>
                        </tr>
                </table>";
?>
```

Fig. 54: Resultado Consulta de Lugar en el Mapa



Fuente: Autor

5.6 Conclusiones

Después de haber implementado e instalado los diferentes componentes de la arquitectura, obtenemos la aplicación a la que el usuario accederá a través de un browser permitiéndole navegar en el mapa de la ciudad, realizar búsquedas de direcciones y lugares de interés dentro del mapa, y consultar información de estos lugares; para cumplir estos objetivos se instaló un servidor web que maneja las peticiones del cliente y devuelve los resultados, un servidor de mapas que es el que maneja la interacción con la información geográfica y un manejador de bases de datos con capacidades espaciales que almacena la información necesaria. La interacción de estos componentes nos permitió la interacción del usuario con la información espacial y alfanumérica dando como resultado un Sistema de Información para la ciudad de Cuenca en donde podemos ubicar direcciones y lugares representados en un mapa.

En la parte del cliente se obtuvo una interfaz muy dinámica ya que se realizó la implementación usando la metodología AJAX, que es el uso de varias tecnologías que nos permitieron desarrollar una interfaz muy intuitiva, fácil de usar y sin recargas de toda la página a la que el cliente tiene acceso, dando como resultado una aplicación muy fácil y agradable de usar.

CAPITULO 6: CONLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

Después de haber realizado la implementación del Sistema, hemos obtenido una aplicación Web en la que se ha publicado información espacial georeferenciada de la ciudad de Cuenca en modo interactivo, de tal forma que es factible utilizar herramientas de navegación que permiten al usuario final desplazarse por el mapa realizando funciones espaciales como zoom, pan, activación/desactivación de capas, etc. Además se implementó un módulo de búsqueda, que nos brinda la posibilidad de buscar una dirección o un lugar dentro de la ciudad de Cuenca facilitando al usuario final la ubicación específica de un punto de la ciudad a través de un mapa digital.

Éste tipo de aplicaciones son posibles realizar gracias a la popularidad que han tomado últimamente las aplicaciones con capacidades geográficas en la Internet, esto ha contribuido a que se hayan desarrollado varias alternativas libres o propietarias que nos facilitan la publicación información espacial en la Web con múltiples alternativas en lo que se refiere a arquitectura, lenguaje de programación, formato de visualización, etc.

La principal característica del sistema es visualizar información geo-referenciada, para esto se ha recopilado archivos de información en formato Shape de ESRI que usamos para la presentación del mapa en diferentes capas, y para exportar éstos archivos a una base de datos. Los archivos que se han utilizado para realizar éstas actividades se refieren a predios, ríos, vías, información turística y manzanas de la ciudd de Cuenca. Ésta integración de diferentes capas de información dan un gran realce a la visualización y consultas que se ha desarrollado.

En la fase de análisis identificamos las características funcionales y no funcionales que el sistema debería satisfacer, tomando decisiones de arquitectura y modelando los datos y el comportamiento de la aplicación, en ésta fase realizamos un análisis comparativo entre dos paquetes de software que nos permiten publicar información espacial en la Web, MapViewSVG y MapServer, luego de tabular datos en una matriz de decisión se pudo concluir que, para implementar una aplicación que

cumpla con nuestros requerimientos, la mejor opción es usar una arquitectura de Web Mapping Dinámico mediante la utilización de el servidor de mapas MapServer (Tabla 19).

En el diseño del Sistema de Geoinformación para la ciudad de Cuenca, se obtuvo el diagrama relacional de la base de datos identificando los nombres y tipos de campos en las diferentes tablas de nuestra base de datos, así también se diseñó la forma en que se iba a presentar la imagen del mapa seleccionando las capas, sus prioridades y propiedades de visualización que hicieron que el mapa resultante tenga un aspecto y funcionamiento muy atractivo, y por último en ésta fase diseñamos los componentes y su funcionamiento dentro de la arquitectura, que nos dieron una clara idea de cómo realizar la implementación realizando un sistema modular.

La implementación del Sistema se realizó basándonos en los modelos y diagramas que realizamos en las etapas de análisis y diseño, implementando la arquitectura que habíamos previsto, para esto instalamos un servidor Web Apache que funciona integrado con el servidor de mapas MapServer, luego instalamos el manejador de bases de datos PostgreSQL con su extensión PostGIS que nos permitió trabajar directamente con objetos referenciados desde la base de datos, creamos el esquema de nuestra base de datos, las estructuras de nuestras tablas y por último importamos las tablas de predios y vías desde sus respectivos archivos Shape.

La aplicación Web a la que el cliente tiene acceso desarrollamos mediante la metodología AJAX, desarrollando módulos en JavaScript que acceden a asíncronamente a la base de datos mediante scripts en PHP, la metodología AJAX es el uso conjunto de múltiples tecnologías que nos permiten realizar interfaces mas intuitivas y sin recargas constantes en el lado del cliente.

El resultado final es una aplicación que se ejecuta en un navegador web, que nos permite visualizar y navegar sobre el mapa de la ciudad de Cuenca, permitiéndonos la visualización y búsqueda de lugares de interés con la capacidad de recuperar información alfanumérica de cada uno de estos lugares, y la opción de buscar cualquier dirección dentro de la ciudad pasando como parámetro la calle principal, el

numero y/o la calle intersección, posicionando y mostrando estos resultados en el mapa dentro del navegador.

6.2 RECOMENDACIONES

Tomando en cuenta que en la Universidad se realizan constantes desarrollos en lo que respecta a aplicaciones SIG, y éstas cada ves se han ido enfocando a su publicación en la Internet, es de mucha importancia unificar éstos trabajos en un servidor de mapas, teniendo de ésta manera una única interfáz en la que podamos consultar diferentes tipos de cartografías provenientes de diferentes trabajos de investigación., construyendo así una Infraestuctura de Datos Espaciales (IDE), y dando la posibilidad a futuros desarrollos a integrarse a ésta interfaz.

Se debería promover el uso de software libre en instituciones educativas y públicas ya que en la mayoría de los casos podemos optar con opciones de iguales características y desempeño mediante el uso de software gratuito, teniendo asi un gran ahorro de recursos que a veces puede llegar a ser varios miles de dólares en costos de licencias sin mencionar los costos de soporte. En el ámbito de los Sistemas de Información Geográfica se cuentan con muchas opciones para desarrollos de aplicaciones de escritorio y para la Web.

Con la gran popularidad que ha tomado en los últimos años la publicación de información georeferenciada en la Web, se han creado varios estándares para éste proposito; los futuros desarrollo deberían apegarse a éstos estándares ya que mediante la creación de estos servicios vía web permiten integrar información obteniéndolas de distintas fuentes (servidores) y también nos permiten publicar nuestros propios servicios que se pueden hacer públicos para que otras instituciones puedan consumir estos servicios via Web.

REFERENCIAS

Bibliografía

BANDOR, Michael S. 2006. "Quantitative Methods for Software Selection and Evaluation" , Software Engineering Institute. http://www.sei.cmu.edu/publications/documents/06.reports/06tn026/06tn026.htm

BRINKHOFF, Thomas. 2000. The Impacts of Map-Oriented Internet Applications.

CCONALLEM Jim, 2002. "Building Web Applications with UML Second Edition"

CHO, G. 1998. 'Geographic Information Systems and the Law.' John Wiley & Sons, New Jersey, United States of America.

ESRI, 1995. "Understanding GIS- The ARC/INFO Method". GeoInformation International, UK: pp1-2.

GARRETT Jesse James, 2005. "Ajax A New Approach to Web Applications", http://adaptivepath.com/ideas/essays/archives/000385.php, February 18.

HARMON John E. – ANDERSON Steven J. 2003. "The Design and Implementation of Geographic Information Systems"

KROPLA, Bill. 2005. "Beginning MapServer: Open Source GIS Development"

LITKE, Christian - PELLETIER, Michael. 2002. "Build it or Buy it? How to perform a cost-benefit analysis for IT projects." The Fabricator (March 28, 2002).

LOPEZ Fernanda – MARTINEZ Geovany. 2007. "Difusión de los lugares turísticos de la provincia del Azuay por medio de Herramientas Geomaticas", Cuenca

OCHOA ARIAS Paúl. Tutorial de Prácticas ArcGis". 2006

SERRA DEL POZO, Paúl. 2002 a. ‘Alternativas a los Servidores de Mapas’, http://www.mappinginteractivo.com

SERRA DEL POZO, Paúl. 2002 b. ‘Cinco Servidores de Mapas’, http://www.mappinginteractivo.com.

STRAND, Eric J. 1998. “Los usuarios de SIG observan, se pasean, y esperan la evolución de la World Wide Web”. Publicado en el número 1 de GeoInformación, Septiembre/Octubre de 1998, página 46.

WEB

Apache, “http://www.apache.org”

MapServer, “http://mapserver.gis.umn.edu/”

MapViewSVG, http://www.mapviewsvg.com

Open Geospatial Consortium, “http://www.opengeospatial.org/”

PostGIS, “http://postgis.refractions.net/”

PostgresSQL” http://www.postgresql.org/”

Quantum Gis, “http://www.qgis.org/”

Wikipedia, OGC,“http://es.wikipedia.org/wiki/Open_Geospatial_Consortium”

Wikipedia, SIG, “ http://es.wikipedia.org/wiki/SIG”

ANEXO 1 Mapfile Reference (http://mapserver.gis.umn.edu/docs/reference/mapfile)
MapFiles are the basic configuration mechanism for MapServer. This document contains a rundown of the keys and values that describe a mapfile.

1. Introduction

Important notes on using mapfiles.

The Mapfile is the heart of MapServer. It defines the relationships between objects, points MapServer to where data are located and defines how things are to be drawn.

There are some important concepts that you must understand before you can reliably use mapfiles to configure MapServer. First is the concept of a layer. A layer is the combination of data plus styling. Data, in the form of attributes plus geometry, are given styling using CLASS and STYLE directives.

Notes

- The Mapfile is NOT case-sensitive.
- Strings containing non-alphanumeric characters or a MapServer keyword MUST be quoted. It is recommended to put ALL strings in double-quotes.
- There is a maximum of 200 layers per mapfile. This can be changed by editing the map.h file to change the value of MS_MAXLAYERS to the desired number and recompiling. Here are other important default limits:
  - MAXCLASSES 250 (set in map.h)
  - MAXSTYLES 5 (set in map.h)
  - MAXSYMBOLS 64 (set in mapsymbol.h)
- File paths may be given as absolute paths, or as paths relative to the location of the mapfile. In addition, data files may be specified relative to the SHAPEPATH.
- The mapfile has a hierarchical structure, with the Map object being the "root". All other objects fall under this one.
- Comments are designated with a #.
- Attributes are named using the following syntax: [ATTRIBUTENAME] ... Note that the name of the attribute included between the square brackets *IS CASE SENSITIVE*. Generally ESRI generated shapefiles have their attributes (.dbf column names) all in upper-case for instance, and for PostGIS, *ALWAYS* use lower-case.
- MapServer Regular Expressions are used through the operating system's C Library. For information on how to use and write Regular Expressions on your system, you should read the documentation provided with your C Library. On Linux, this is GLibC, and you can read "man 7 regex" ... This man page is also available on most UNIX's. Since these RegEx's are POSIX compliant, they should be the same on Windows as well, so windows users can try searching the web for "man 7 regex" since man pages are available all over the web.

2. Class

Defines thematic classes for a given layer and each layer must have at least one class. In cases with more than one class, membership is determined using attribute values and expressions. Starts with the keyword CLASS and terminates with the keyword END.

BACKGROUNDCOLOR [r] [g] [b]
: Color to use for non-transparent symbols.

COLOR [r] [g] [b]
: Color to use for drawing features.

DEBUG [on|off]
: Enables debugging of the class object. Verbose output is generated and sent to the standard error output (STDERR) or the MapServe r logfile if one is set using the LOG parameter in the WEB object.

EXPRESSION [string]

Four types of expressions are now supported to define class membership. String comparisons, regular expressions, simple logical expressions, and string functions. If no expression is given, then all features are said to belong to this class.

- String comparisons are case sensitive and are the fastest to evaluate. No special delimiters are necessary although string must be quoted if they contain special characters. (As a matter of good habit, it is recommended you quote all strings).
- Regular expressions function just like previous versions of MapServer. However, you must now delimit a regular expression using /regex/. No quotes should be used.
- Logical expressions allow you to build fairly complex tests based on one or more attributes and therefore are only available with shapefiles. Logical expressions are delimited by parentheses "(expression)". Attribute names are delimited by square brackets "[ATTRIBUTE]". These names are case sensitive and must match the items in the shapefile. For example: EXPRESSION ([POPULATION] > 50000 AND '[LANGUAGE]' eq 'FRENCH') ... The following logical operators are supported: =,>,<,<=,>=,=,or,and,lt,gt,ge,le,eq,ne. As you might expect this level of complexity is slower to process.
- One string function exists: length(). This obviously computes the length of a string. An example follows:
- EXPRESSION (length('[NAME_E]') < 8)

String comparisons and regular expressions work from the classitem defined at the layer level. You may mix expression types within the different classes of a layer.

JOIN
Signals the start of a JOIN object.

KEYIMAGE [filename]
Full filename of the legend image for the CLASS. This image is used when building a legend (or requesting a legend icon via MapScript or the CGI application).

LABEL
Signals the start of a LABEL object.

MAXSCALE [double]
Maximum scale at which this CLASS is drawn.

MAXSIZE [integer]
Maximum size in pixels to draw a symbol. Default is 50.

MINSCALE [double]
Minimum scale at which this CLASS is drawn.

MINSIZE [integer]
Minimum size in pixels to draw a symbol. Default is 0.

NAME [string]
Name to use in legends for this class. If not set class won't show up in legend.

OUTLINECOLOR [r] [g] [b]
Color to use for outlining polygons and certain marker symbols. Line symbols do not support outline colors.

SIZE [integer]
Height, in pixels, of the symbol/pattern to be used. Only useful with scalable symbols. For vector (and ellipse) symbol types the default size is based on the range of Y values in the POINTS defining the symbol. For pixmaps, the default is the vertical size of the image. Default size is 1 for TTF symbols.

STYLE
Signals the start of a STYLE object. A class can contain multiple styles.

SYMBOL [integer|string|filename]

The symbol name or number to use for all features if attribute tables are not used. The number is the index of the symbol in the symbol file, starting at 1, the 5th symbol in the file is therefore symbol number 5. You can also give your symbols names using the NAME keyword in the symbol definition file, and use those to refer to them. Default is 0, which results in a single pixel, single width line, or solid polygon fill, depending on layer type.

You can also specify a gif or png filename. The path is relative to the location of the mapfile.

TEMPLATE [filename]
Template file or URL to use in presenting query results to the user.
TEXT [string]
Static text to label features in this class with. This overrides values obtained from the LABELTIEM. The string may be given as an expression delimited using the ()'s. This allows you to concatenate multiple attributes into a single label. For example: ([FIRSTNAME],[LASTNAME]).

You can also "stack" 2 symbols to achieve interesting effects. You define the second symbol, which effectively sits "on top" of the symbol normally defined above.

The following parameters allow you to define the symbol, and they are equivalent to their non-overlay counterparts:

- OVERLAYBACKGROUNDCOLOR
- OVERLAYCOLOR
- OVERLAYOUTLINECOLOR
- OVERLAYSIZE
- OVERLAYMINSIZE
- OVERLAYMAXSIZE
- OVERLAYSYMBOL

3. Feature

Defines inline features. You can use inline features when it's not possible (or too much trouble) to create a shapefile. Inline features can also be built via urls or forms. Starts with the keyword FEATURE and terminates with the keyword END.
POINTS

A set of xy pairs terminated with an END, for example:

POINTS 1 1 50 50 1 50 1 1 END

Note that with POLYGON/POLYLINE layers POINTS must start and end with the same point (i.e. close the feature).

TEXT [string]
String to use for labeling this feature.
WKT [string]

A geometry expressed in OpenGIS Well Known Text geometry format. This feature is only supported if MapServer is built with OGR or GEOS support.

WKT "POLYGON((500 500, 3500 500, 3500 2500, 500 2500, 500 500))"
WKT "POINT(2000 2500)"

Note

Inline features should be defined as their own layers in the mapfile. If another CONNECTIONTYPE is specified in the same layer, MapServer will always use the inline features to draw the layer and ignore the other CONNECTIONTYPEs.

4. Grid

The GRID object defines a map graticule as a LAYER. Starts with the keyword GRID and terminates with the keyword END.

LABELFORMAT [DDMM|DDMMSS]
: Format of the label. "DDMM" for degrees minutes, and "DDMMSS" for degrees, minutes, seconds. The default is decimal display of whatever SRS you're rendering the GRID with.

MINARCS [double]
: The minimum number of arcs to draw. Use this parameter to get more lines. Optional.

MAXARCS [double]
: The maximum number of arcs to draw. Use this parameter to get fewer lines. Optional.

MININTERVAL [double]
: The minimum number of intervals to try to use. The distance between the grid lines, in the units of the grid's coordinate system. Optional.

MAXINTERVAL [double]
: The maximum number of intervals to try to use. The distance between the grid lines, in the units of the grid's coordinate system. Optional.

MINSUBDIVIDE [double]
: The minimum number of segments to use when rendering an arc. If the lines should be very curved, use this to smooth the lines by adding more segments. Optional.

MAXSUBDIVIDE [double]
: The maximum number of segments to use when rendering an arc. If the graticule should be very straight, use this to minimize the number of points for faster rendering. Optional, default 256.

The following is an example of a GRID object in use:

```
 LAYER
  NAME "grid"
  METADATA
    "DESCRIPTION" "Grid"
  END
  TYPE LINE
  STATUS ON
  CLASS
    NAME "Graticule"
    COLOR 0 0 0
    LABEL
      COLOR  255 0 0
      FONT fritqat
      TYPE truetype
      SIZE 8
      POSITION AUTO
      PARTIALS FALSE
      BUFFER 5
      OUTLINECOLOR 255 255 255
    END
  END
  PROJECTION
     "init=epsg:4326"
  END
  GRID
    LABELFORMAT DDMM
    MAXARCS 10
    MAXINTERVAL 10
    MAXSUBDIVIDE 2
  END
END # Layer
```

5. Include

Defines a file to be included in the mapfile parsing.

When this directive is encountered parsing switches to the included file immediately. As a result the included file can be comprised of any valid mapfile syntax. For example:

INCLUDE 'myLayer.map'

Performance does not seem to be seriously impacted with limited use, however in high performance instances you may want to use includes in a pre-processing step to build a production mapfile. The C pre-processor can also be used (albeit with a differnt syntax) and is far more powerful.

Notes

- Supported in versions 4.10 and higher.
- The name of the file to be included **MUST be quoted** (single or double quotes).
- Includes may be nested, up to 5 deep.
- File location can be given as a full path to the file, or (in MapServer >= 4.10.1) as a path relative to the mapfile.
- Debugging can be problematic since: 1) the file an error occurs in does not get output to the user and 2) the line number counter is not reset for each file. Here is one possible error that is thrown when the include file cannot be found:
- msyylex(): Unable to access file. Error opening included file "parks_include.map"

Example

```
MAP
 NAME 'include'
 EXTENT 0 0 500 500
 SIZE 250 250

 INCLUDE "test_include_symbols.map"
 INCLUDE "test_include_layer.map"
END
```

where test_include_symbols.map contains:

```
SYMBOL
  NAME 'square'
  TYPE VECTOR
  FILLED TRUE
  POINTS 0 0 0 1 1 1 1 0 0 0 END
END
```

and test_include_layer.map contains:

```
LAYER
 TYPE POINT
 STATUS DEFAULT
 FEATURE
  POINTS 10 10 40 20 300 300 400 10 10 400 END
 END
 CLASS
  NAME 'Church'
  COLOR 0 0 0
  SYMBOL 'square'
  SIZE 7
```

```
    STYLE
      SYMBOL "square"
      SIZE 5
      COLOR 255 255 255
    END
    STYLE
      SYMBOL "square"
      SIZE 3
      COLOR 0 0 255
    END
  END
END
```

6. Join

Defines how a specific join is handled. Starts with the keyword JOIN and terminates with the keyword END.

Description

Joins are defined within a **LAYER** object. It is important to understand that JOINs are *ONLY* available once a query has been processed. You cannot use joins to affect the look of a map. The primary purpose is to enable lookup tables for coded data (e.g. 1 => Forest) but there are other possible uses.

Supported Formats

- DBF/XBase files
- CSV (comma delimited text file)
- PostgreSQL and PostGIS tables
- MySQL tables

Mapfile Parameters:

CONNECTION [string]
: Parameters required for the join table's database connection (not required for DBF or CSV joins). The following is an example for PostgreSQL:

```
CONNECTION "host=127.0.0.1 port=5432 user=postgres password=postgres dbname=somename"
```

CONNECTIONTYPE [string]
: Type of connection (not required for DBF or CSV joins). The following is an example for PostgreSQL:

```
CONNECTIONTYPE ogr
```

FROM [item]
: Join item in the dataset. This is case sensitive.

NAME [string]
: Unique name for this join. Required.

TABLE [filename|tablename]
: For file-based joins this is the name of XBase or comma delimited file (relative to the location of the mapfile) to join TO. For PostgreSQL and MySQL support this is the name of the PostgreSQL/MySQL table to join TO.

TEMPLATE [filename]
: Template to use with one-to-many joins. The template is processed once for each record and can only contain substitutions for items in the joined table. Refer to the column in the joined table in your template like [joinname_columnname], where joinname is the NAME specified for the JOIN object.

TO [item]
: Join item in the table to be joined. This is case sensitive.

TYPE [ONE-TO-ONE|ONE-TO-MANY]
: The type of join. Default is one-to-one.

Example1: Join from SHP file to DBF file

Mapfile Layer

```
LAYER
  NAME prov_bound
  TYPE POLYGON
  STATUS DEFAULT
  DATA prov.shp
  CLASS
    NAME "Province"
    STYLE
      OUTLINECOLOR 120 120 120
      COLOR 255 255 0
    END
  END
  TEMPLATE "../htdocs/cgi-query-templates/prov.html"
  HEADER "../htdocs/cgi-query-templates/prov-header.html"
  FOOTER "../htdocs/cgi-query-templates/footer.html"
  JOIN
    NAME "test"
    TABLE "../data/lookup.dbf"
    FROM "ID"
    TO "IDENT"
    TYPE ONE-TO-ONE
  END
END # layer
```

7. Label

This object is used to define a label, which is in turn usually used to annotate a feature with a piece of text. Labels can however also be used as symbols through the use of various TrueType fonts.

ANGLE [double|auto|follow]

Angle, given in degrees, to draw the label or AUTO to allow the software to compute the angle, AUTO is valid for LINE layers only. FOLLOW was introduced in version 4.10 and tells map server to compute a curved label for appropriate linear features (see RFC 11 for specifics).

ANTIALIAS [true|false]

Should text be antialiased? Note that this requires more available colors, decreased drawing performance, and results in slightly larger output images.

BACKGROUNDCOLOR [r] [g] [b]

Color to draw a background rectangle (i.e. billboard). Off by default.

BACKGROUNDSHADOWCOLOR [r] [g] [b]

Color to draw a background rectangle (i.e. billboard) shadow. Off by default.

BACKGROUNDSHADOWSIZE [x][y]

How far should the background rectangle be offset? Default is 1.

BUFFER [integer]

Padding, in pixels, around labels. Useful for maintaining spacing around text to enhance readability. Available only for cached labels. Default is 0.

COLOR [r] [g] [b]

Color to draw text with.

ENCODING [string]

Supported encoding format to be used for labels. If the format is not supported, the label will not be drawn. Requires the iconv library (present on most systems). The library is always detected if present on the system, but if not the label will not be drawn.

Required for displaying international characters in MapServer. More information can be found at: http://www.foss4g.org/FOSS4G/MAPSERVER/mpsnf-i18n-en.html.

FONT [name]

Font alias (as defined in the FONTSET) to use for labeling.

FORCE [true|false]

Forces labels for a particular class on, regardless of collisions. Available only for cached labels. Default is false.

MAXSIZE [integer]

Maximum font size to use when scaling text (pixels). Default is 256.

MINDISTANCE [integer]

Minimum distance between duplicate labels. Given in pixels.

MINFEATURESIZE [integer|auto]

Minimum size a feature must be to be labeled. Given in pixels. For line data the overall length of the displayed line is used, for polygons features the smallest dimension of the bounding box is used. "Auto" keyword tells MapServer to only label features that are larger than their corresponding label. Available for cached labels only.

MINSIZE [integer]

Minimum font size to use when scaling text (pixels). Default is 4.

OFFSET [x][y]

Offset values for labels, relative to the lower left hand corner of the label and the label point. Given in pixels. In the case of rotated text specify the values as if all labels are horizontal and any rotation will be compensated for.

OUTLINECOLOR [r] [g] [b]

Color to draw a one pixel outline around the text.

PARTIALS [true|false]

Can text run off the edge of the map? Default is true.

POSITION [ul|uc|ur|cl|cc|cr|ll|lc|lr|auto]

Position of the label relative to the labeling point (layers only). First letter is "Y" position, second letter is "X" position. "Auto" tells MapServer to calculate a label position that will not interfere with other labels. With points and polygons, MapServer selects from the 8 outer positions (i.e. excluding cc). With lines, it only uses lc or uc, until it finds a position that doesn't collide with labels that have already been drawn. If all positions cause a conflict, then the label is not drawn (Unless the label's FORCE a parameter is set to "true"). "Auto" placement is only available with cached labels.

PRIORITY [integer]|[item_name]

The priority parameter (added in v5.0) takes an integer value between 1 (lowest) and 10 (highest). The default value is 1. It is also possible to bind the priority to an attribute (item_name) using square brackets around the [item_name]. e.g. "PRIORITY [someattribute]"

Labels are stored in the label cache and rendered in order of priority, with the highest priority levels rendered first. Specifying an out of range PRIORITY value inside a map file will result in a parsing error. An out of range value set via MapScript or coming from a shape attribute will be clamped to the min/max values at rendering time. There is no expected impact on performance for using label priorities.

SHADOWCOLOR [r] [g] [b]

Color of drop shadow.

SHADOWSIZE [x][y]

Shadow offset in pixels.

SIZE [integer]|[tiny|small|medium|large|giant]

Text size. Use "integer" to give the size in pixels of your TrueType font based label, or any of the other 5 listed keywords to bitmap fonts.

TYPE [bitmap|truetype]

Type of font to use. Generally bitmap fonts are faster to draw then TrueType fonts. However, TrueType fonts are scalable and available in a variety of faces. Be sure to set the FONT parameter if you select TrueType.

WRAP [character]

Character that represents an end-of-line condition in label text, thus resulting in a multi-line label.

8. Layer

The most used object in a MapFile, this one describes layers used to make up a map. Layers are drawn in their order of appearance in the MapFile (first layer is at the bottom, last in on top).

CLASS

Signals the start of a CLASS object.

CLASSITEM [attribute]

Item name in attribute table to use for class lookups.

CONNECTION [string]

Database connection string to retrieve remote data.

An SDE connection string consists of a hostname, instance name, database name, username and password separated by commas.

A PostGIS connection string is basically a regular PostgreSQL connection string, it takes the form of "user=nobody password=****** dbname=dbname host=localhost port=5432"

An Oracle connection string: user/pass[@db]

CONNECTIONTYPE [local|sde|ogr|postgis|oraclespatial|wms]

Type of connection. Default is local. See additional documentation for any other type.

DATA [filename]|[sde parameters][postgis table/column][oracle table/column]

Full filename of the spatial data to process. No file extension is necessary for shapefiles. Can be specified relative to the SHAPEPATH option from the Map Object.

If this is an SDE layer, the parameter should include the name of the layer as well as the geometry column, i.e. "mylayer,shape,myversion".

If this is a PostGIS layer, the parameter should be in the form of "<columnname> from <tablename>", where "columnname" is the name of the column containing the geometry objects and "tablename" is the name of the table from which the geometry data will be read.

For Oracle, use "shape FROM table" or "shape FROM (SELECT statement)" or even more complex Oracle compliant queries! Note that there are important performance impacts when using spatial subqueries however. Try using MapServer's FILTER whenever possible instead. You can also see the SQL submitted by forcing an error, for instance by submitting a DATA parameter you know won't work, using for example a bad column name.

DEBUG [on|off]

Enables debugging of the layer object. Verbose output is generated and sent to the standard error output (STDERR) or the MapServer logfile if one is set using the LOG parameter in the WEB object.

DUMP [true|false]

Switch to allow mapserver to return data in GML format. Usefull when used with WMS GetFeatureInfo operations. "false" by default.

FEATURE

Signals the start of a FEATURE object.

FILTER [string]

This parameter allows for data specific attribute filtering that is done at the same time spatial filtering is done, but before any CLASS expressions are evaluated. For OGR and shapefiles the string is simply a mapserver regular expression. For spatial databases the string is a SQL WHERE clause that is valid with respect to the underlying database.

For example: FILTER "type='road' and size <2"

FILTERITEM [attribute]
Item to use with simple FILTER expressions. OGR and shapefiles only.
FOOTER [filename]
Template to use *after* a layer's set of results have been sent. Multiresult query modes only.
GRID
Signals the start of a GRID object.
GROUP [name]
Name of a group that this layer belongs to. The group name can then be reference as a regular layer name in the template files, allowing to do things like turning on and off a group of layers at once.
HEADER [filename]
Template to use *before* a layer's set of results have been sent. Multiresult query modes only.
LABELANGLEITEM [attribute]
Item name in attribute table to use for class annotation angles. Values should be in degrees.
LABELCACHE [on|off]
Specifies whether labels should be drawn as the features for this layer are drawn, or whether they should be cached and drawn after all layers have been drawn. Default is on. Label overlap removal, auto placement etc... are only available when the label cache is active.
LABELITEM [attribute]
Item name in attribute table to use for class annotation (i.e. labeling).
LABELMAXSCALE [double]
Maximum scale at which the layer is labeled.
LABELMINSCALE [double]
Minimum scale at which the layer is labeled.
LABELREQUIRES [expression]

Sets context for labeling this layer, for example:

LABELREQUIRES "![orthoquads]"

means that this layer would NOT be labeled if a layer named "orthoquads" is on. The expression consists of a boolean expression based on the status of other layers, each [layer name] substring is replaced by a 0 or a 1 depending on that layer's STATUS and then evaluated as normal. Logical operators AND and OR can be used.

LABELSIZEITEM [attribute]
Item name in attribute table to use for class annotation sizes. Values should be in pixels.
MAXFEATURES [integer]
Specifies the number of features that should be drawn for this layer in the CURRENT window. Has some interesting uses with annotation and with sorted data (i.e. lakes by area).
MAXSCALE [double]
Maximum scale at which this layer is drawn.
METADATA

This keyword allows for arbitrary data to be stored as name value pairs. This is used with OGC WMS to define things such as layer title. It can also allow more flexibility in creating templates, as anything you put in here will be accessible via template tags.

Example:

```
METADATA
  title "My layer title"
  author "Me!"
END
```

MINSCALE [double]
Minimum scale at which this layer is drawn.
NAME [string]
Short name for this layer. Limit is 20 characters. This name is the link between the mapfile and web interfaces that refer to this name. They must be identical. The name should be

unique, unless one layer replaces another at different scales. Use the GROUP option to associate layers with each other.

OFFSITE [r] [g] [b]

Sets the color index to treat as transparent for raster layers.

POSTLABELCACHE [true|false]

Tells MapServer to render this layer after all labels in the cache have been drawn. Useful for adding neatlines and similar elements. Default is false.

PROCESSING [string]

Passes a processing directive to be used with this layer. The supported processing directives vary by layer type, and the underlying driver that processes them. Here we see the SCALE and BANDs directives used to autoscale raster data and alter the band mapping. All raster processing options are described in the Raster Data Access HOWTO.

```
PROCESSING "SCALE=AUTO"
PROCESSING "BANDS=3,2,1"
```

This is also where you can enable connection pooling for certain layer layer types. Connection pooling will allow MapServer to share the handle to an open database or layer connection throughout a single map draw process. Additionally, if you have FastCGI enabled, the connection handle will stay open indefinitely, or according to the options specified in the FastCGI configuration. Oracle, ArcSDE, OGR and PostGIS currently support this approach.

```
PROCESSING "CLOSE_CONNECTION=DEFER"
```

PROJECTION

Signals the start of a PROJECTION object.

REQUIRES [expression]

Sets context for displaying this layer (see LABELREQUIRES).

SIZEUNITS [pixels|feet|inches|kilometers|meters|miles|

Sets the unit of CLASS object SIZE values (default is pixels). Useful for simulating buffering.

STATUS [on|off|default]

Sets the current status of the layer. Often modified by MapServer itself. Default turns the layer on permanently.

Some notes regarding the STATUS values:

- In CGI mode, layers with STATUS DEFAULT cannot be turned off using normal mechanisms. It is recommended to set layers to STATUS DEFAULT while debugging a problem, but set them back to ON/OFF in normal use.
- For WMS, layers in the server mapfile with STATUS DEFAULT are always sent to the client.

STYLEITEM [attribute]

Item to use for feature specific styling. This is *very* experimental and OGR only at the moment.

SYMBOLSCALE [double]

The scale at which symbols and/or text appear full size. This allows for dynamic scaling of objects based on the scale of the map. If not set then this layer will always appear at the same size. Scaling only takes place within the limits of MINSIZE and MAXSIZE as described above.

TEMPLATE [file|url]

Used as a global alternative to CLASS TEMPLATE.

TILEINDEX [filename|layername]

Name of the tileindex file or layer. A tileindex is similar to an ArcInfo library index. The tileindex contains polygon features for each tile. The item that contains the location of the tiled data is given using the TILEITEM parameter. When a file is used as the tileindex for shapefile or raster layers, the tileindex should be a shapefile. For CONNECTIONTYPE OGR layers, any OGR supported datasource can be a tileindex. Normally the location should contain the path to the tile file relative to the shapepath, not relative to the tileindex itself. If the DATA parameter contains a value then it is added to the end of the location. When a tileindex layer is used, it works similarly to directly referring to a file, but any supported feature source can be used (ie. postgres, oracle).

NOTE: All files in the tileindex should have the same coordinate system, and for vector files the same set of attributes in the same order.

TILEITEM [attribute]
: Item that contains the location of an individual tile, default is "location".

TOLERANCE [double]
: Sensitivity for point based queries (i.e. via mouse and/or map coordinates). Given in TOLERANCEUNITS. If the layer is a POINT or a LINE, the default is 3. For all other layer types, the default is 0. To restrict polygon searches so that the point must occur in the polygon set the tolerance to zero.

TOLERANCEUNITS [pixels|feet|inches|kilometers|meters|miles|dd]
: Units of the TOLERANCE value. Default is pixels.

TRANSPARENCY [integer|alpha]

Sets the transparency level of all classed pixels for a given layer. The value can either be an integer in the range (0-100) or the named symbol "ALPHA". Although this parameter is named "transparency", the integer values actually parameterize layer opacity. A value of 100 is opaque and 0 is fully transparent.

The "ALPHA" symbol directs the mapserver rendering code to honor the indexed or alpha transparency of pixmap symbols used to style a layer. This is only needed in the case of RGB output formats, and should be used only when necessary as it is expensive to render transparent pixmap symbols onto an RGB map image.

TRANSFORM [true|false ul|uc|ur|lc|cc|lr|ll|lc|lr]

Tells MapServer whether or not a particular layer needs to be transformed from some coordinate system to image coordinates. Default is true. This allows you to create shapefiles in image/graphics coordinates and therefore have features that will always be displayed in the same location on every map. Ideal for placing logos or text in maps. Remember that the graphics coordinate system has an origin in the upper left hand corner of the image, contrary to most map coordinate systems.

Version 4.10 introduces the ability to define features with coordinates given in pixels (or percentatges, see UNITS), most often inline features, relative to something other than the UL corner of an image. That is what 'TRANSFORM FALSE' means. By setting an alternative origin it allows you to anchor something like a copyright statement to another portion of the image in a way that is independent of image size.

TYPE [point|line|polygon|circle|annotation|raster|query|chart]

Specifies how the data should be drawn. Need not be the same as the shapefile type. For example, a polygon shapefile may be drawn as a point layer, but a point shapefile may not be drawn as a polygon layer. Common sense rules. Annotation means that a label point will be calculated for the features, but the feature itself will not be drawn although a marker symbol can be optionally drawn. this allows for advanced labeling like numbered highway shields. Points are labeled at that point. Polygons are labeled first using a centroid, and if that doesn't fall in the polygon a scanline approach is used to guarantee the label falls within the feature.

Lines are labeled at the middle of the longest arc in the visible portion of the line. Query only means the layer can be queried but not drawn.

In order to differentiate between POLYGONs and POLYLINEs (which do not exist as a type), simply respectively use or ommit the COLOR keyword when classifying. If you use it, it's a polygon with a fill color, otherwise it's a polyline with only an OUTLINECOLOR.

For CHART layers, see the Dynamic Charting howto.

A circle must be defined by a a minimum bounding rectangle. That is, 2 points that define the smallest square that can contain it. These 2 points are the two opposite corners of said box.

The following is an example using inline points to draw a circle:

```
 LAYER
   NAME 'inline_circles'
   TYPE CIRCLE
   STATUS ON
   FEATURE
     POINTS
       74.01 -53.8
       110.7 -22.16
     END
   END
   CLASS
     STYLE
       COLOR 0 0 255
     END
   END
END
```

9. Legend

Defines how a legend is to be built. Legend components are built automatically from class objects from individual layers. Starts with the keyword LEGEND and terminates with the keyword END.

The size of the legend image is NOT known prior to creation so be careful not to hard-code width and height in the <IMG> tag in the template file.

IMAGECOLOR [r] [g] [b]
: Color to initialize the legend with (i.e. the background).

INTERLACE [on|off]
: **Deprecated** Default is [on]. This keyword is now depcrecated in favour of using the FORMATOPTION "INTERLACE=ON" line in the OUTPUTFORMAT declaration.

LABEL
: Signals the start of a LABEL object

OUTLINECOLOR [r] [g] [b]
: Color to use for outlining symbol key boxes.

POSITION [ul|uc|ur|ll|lc|lr]
: Where to place an embedded legend in the map. Default is lr.

KEYSIZE [x][y]
: Size of symbol key boxes in pixels. Default is 20 by 10.

KEYSPACING [x][y]
: Spacing between symbol key boxes ([y]) and labels ([x]) in pixels. Default is 5 by 5.

POSTLABELCACHE [true|false]
: Tells MapServer to render this legend after all labels in the cache have been drawn. Useful for adding neatlines and similar elements. Default is false.

STATUS [on|off|embed]
: Is the legend image to be created.

TEMPLATE [filename]

HTML legend template file. Refer to the HTML Legend howto.

TRANSPARENT [on|off]

Deprecated Should the background color for the legend be transparent. This flag is now deprecated in favour of declaring transparency within OUTPUTFORMAT declarations. Default is off.

10. Map

A MAP object defines the master object of the mapfile. It defines application/map wide parameters.

ANGLE [double]

Angle, given in degrees, to rotate the map. Default is 0. The rendered map will rotate in a clockwise direction. The following are important notes:

- Requires a PROJECTION object specified at the MAP level and for each LAYER object (even if all layers are in the same projection).
- Requires MapScript (SWIG, PHPMapscript). Does not work with CGI mode.
- If using the LABEL object's ANGLE or the LAYER object's LABELANGLEITEM parameters as well, these parameters are relative to the map's orientation (i.e. they are computed after the MAP object's ANGLE). For example, if you have specified an ANGLE for the map of 45, and then have a layer LABELANGLEITEM value of 45, the resulting label will not appear rotated (because the resulting map is rotated clockwise 45 degrees and the label is rotated counter-clockwise 45 degrees).
- More information can be found on the MapRotation Wiki Page.

CONFIG [key] [value]

This can be used to define the location of your EPSG files for the PROJ.4 library. Setting the [key] to PROJ_LIB and the [value] to the location of your EPSG files will force PROJ.4 to use this value. Using CONFIG allows you to avoid setting environment variables to point to your PROJ_LIB directory. Here are some examples:

1. Unix
2. CONFIG "PROJ_LIB" "/usr/local/share/proj/"
3. Windows
4. CONFIG "PROJ_LIB" "C:/somedir/proj/nad/"

Any other value will be passed on to CPLSetConfigOption(). This provides customized control of some GDAL and OGR driver behaviours. Details on such options would be found in specific GDAL driver documentation.

DATAPATTERN [regular expression]

This defines a regular expression to be applied to requests to change DATA parameters via URL requests (i.e. map_layername_data=...). If a pattern doesn't exist then web users can't monkey with support files via URLs. This allows you to isolate one application from another if you desire, with the default operation being very conservative. See also TEMPLATEPATTERN.

DEBUG [on|off]

Enables debugging of the map object. Verbose output is generated and sent to the standard error output (STDERR) or the MapServer logfile if one is set using the LOG parameter in the WEB object. Apache users will see timing details for drawing in Apache's error_log file.

Requires MapServer to be built with the DEBUG=MSDEBUG option (--with-debug configure option).

EXTENT [minx] [miny] [maxx] [maxy]

The spatial extent of the map to be created. In most cases you will need to specify this, although mapserver can sometimes (expensively) calculate one if it is not specified.

FONTSET [filename]

Filename of fontset file to use. Can be a path relative to the mapfile, or a full path.

IMAGECOLOR [r] [g] [b]

Color to initialize the map with (i.e. background color). When transparency is enabled (TRANSPARENT ON) for the typical case of 8-bit pseudocolored map generation, this color will be marked as transparent in the output file palette. Any other map components drawn in this color will also be transparenct, so for map generation with transparency it is best to use an otherwise unused color as the background color.

IMAGEQUALITY [int]

Deprecated Use FORMATOPTION "QUALITY=n" in the OUTPUTFORMAT declaration to specify compression quality for JPEG output.

IMAGETYPE [gif|png|jpeg|wbmp|gtiff|swf|userdefined]

Output format to generate. See details in the OUTPUTFORMAT section for available formats. The name here must match the 'NAME' of a user defined or internally generated OUTPUTFORMAT section.

INTERLACE [on|off]

Deprecated Use FORMATOPTION "INTERLACE=ON" in the OUTPUTFORMAT declaration to specify if the output images should be interlaced.

LAYER

Signals the start of a LAYER object.

LEGEND

Signals the start of a LEGEND object.

MAXSIZE [integer]

Sets the maximum size of the map image. This will override the default value. For example, setting this to 2048 means that you can have up to 2048 pixels in both dimensions (i.e. max of 2048x2048).

NAME [name]

Prefix attached to map, scalebar and legend GIF filenames created using this MapFile. It should be kept short.

PROJECTION

Signals the start of a PROJECTION object.

QUERYMAP

Signals the start of a QUERYMAP object.

REFERENCE

Signals the start of a REFERENCE MAP object.

RESOLUTION [int]

Sets the pixels per inch for output, only affects scale computations and nothing else, default is 72.

SCALE [double]

Computed scale of the map. Set most often by the application.

SCALEBAR

Signals the start of a SCALEBAR object.

SHAPEPATH [filename]

Path to the directory holding the shapefiles or tiles. There can be further subdirectories under SHAPEPATH.

SIZE [x][y]

Size in pixels of the output image (i.e. the map).

STATUS [on|off]

Is the map active? Sometimes you may wish to turn this off to use only the reference map or scale bar.

SYMBOLSET [filename]

Filename of the symbolset to use. Can be a path relative to the mapfile, or a full path.

SYMBOL

Signals the start of a SYMBOL object.

TEMPLATEPATTERN [regular expression]

This defines a regular expression to be applied to requests to change TEMPLATE parameters via URL requests (i.e. map_layername_template=...). If a pattern doesn't exist then web users can't monkey with support files via URLs. This allows you to isolate one application from

another if you desire, with the default operation being very conservative. See also DATAPATTERN.

TRANSPARENT [on|off]

Deprecated Use FORMATOPTION "TRANSPARENT=ON" in the OUTPUTFORMAT declaration to specify if the output images should be transparent.

UNITS [feet|inches|kilometers|meters|miles|dd]

Units of the map coordinates. Used for scalebar and scale computations.

WEB

Signals the start of a WEB object.

11. OutputFormat

This section discusses how output formats are defined and selected.

A map file may have zero, one or more OUTPUTFORMAT object declarations, defining available output formats supported including formats like PNG, GIF, JPEG, GeoTIFF and Flash (SWF).

If OUTPUTFORMAT sections declarations are not found in the map file, the following implicit declarations will be made. Only those for which support is compiled in will actually be available. The GeoTIFF depends on building with GDAL support, and the Flash (SWF) depends on compiling with support for the MING library.

```
OUTPUTFORMAT
  NAME gif
  DRIVER "GD/GIF"
  MIMETYPE "image/gif"
  IMAGEMODE PC256
  EXTENSION "gif"
END
OUTPUTFORMAT
  NAME png
  DRIVER "GD/PNG"
  MIMETYPE "image/png"
  IMAGEMODE PC256
  EXTENSION "png"
END
OUTPUTFORMAT
  NAME jpeg
  DRIVER "GD/JPEG"
  MIMETYPE "image/jpeg"
  IMAGEMODE RGB
  EXTENSION "jpg"
END
OUTPUTFORMAT
  NAME wbmp
  DRIVER "GD/WBMP"
  MIMETYPE "image/wbmp"
  IMAGEMODE PC256
  EXTENSION "wbmp"
END
OUTPUTFORMAT
  NAME swf
  DRIVER "SWF"
  MIMETYPE "application/x-shockwave-flash"
  EXTENSION "swf"
  IMAGEMODE PC256
  FORMATOPTION "OUTPUT_MOVIE=SINGLE"
END
OUTPUTFORMAT
  NAME GTiff
```

```
  DRIVER "GDAL/GTiff"
  MIMETYPE "image/tiff"
  IMAGEMODE RGB
  EXTENSION "tif"
END
```

NAME [name]

The name to use use in the IMAGETYPE keyword of the map file to select this output format.(optional)

DRIVER [name]

The name of the driver to use to generate this output format. Some driver names include the definition of the format if the driver supports multiple formats. For GD the possible driver names are "GD/Gif", "GD/PNG", "GD/WBMP" and "GD/JPEG". For flash the driver is just called "SWF". For output through GDAL the GDAL shortname for the format is appeneded, such as "GDAL/GTiff". Note that PNG, JPEG and GIF output can be generated with either GDAL or GD (GD is generally more efficient).(manditory)

IMAGEMODE [PC256/RGB/RGBA/INT16/FLOAT32]

Selects the imaging mode in which the output is generated. Does matter for non-raster formats like Flash. Not all formats support all combinations. For instance GD/GIF supports only PC256. (optional)

- PC256: Produced a pseudocolored result with up to 256 colors in the palette (traditional MapServer mode)
- RGB: Render in 24bit Red/Green/Blue mode. Supports all colors but does not support transparency.
- RGBA: Render in 32bit Red/Green/Blue/Alpha mode. Supports all colors, and alpha based transparency.
- BYTE: Render raw 8bit pixel values (no presentation). Only works for RASTER layers (through GDAL) and WMS layers currently.
- INT16: Render raw 16bit signed pixel values (no presentation). Only works for RASTER layers (through GDAL) and WMS layers currently.
- FLOAT32: Render raw 32bit floating point pixel values (no presentation). Only works for RASTER layers (through GDAL) and WMS layers currently.

MIMETYPE [type]

Provide the mime type to be used when returning results over the web. (optional)

EXTENSION [type]

Provide the extension to use when creating files of this type. (optional)

TRANSPARENT [ON/OFF]

Indicates whether transparency should be enabled for this format. Note that transparency does not work for IMAGEMODE RGB output. Not all formats support transparency (optional). When transparency is enabled for the typical case of 8-bit pseudocolored map generation, the IMAGECOLOR color will be marked as transparent in the output file palette. Any other map components drawn in this color will also be transparent, so for map generation with transparency it is best to use an otherwise unused color as the background color.

FORMATOPTION [option]

Provides a driver or format specific option. Zero or more FORMATOPTION statement may be present within a OUTPUTFORMAT declaration. (optional)

- GD/JPEG: The "QUALITY=n" option may be used to set the quality of jpeg produced (value from 0-100).
- GD/PNG: The "INTERLACE=[ON/OFF]" option may be used to turn interlacing on or off.
- GD/GIF: The "INTERLACE=[ON/OFF]" option may be used to turn interlacing on or off.

- GDAL/GTiff: Supports the TILED=YES, BLOCKXSIZE=n, BLOCKYSIZE=n, INTERLEAVE=[PIXEL/BAND] and COMPRESS=[NONE,PACKBITS,JPEG,LZW,DEFLATE] format specific options.
- GDAL/*: All FORMATOPTIONs are passed onto the GDAL create function. Options supported by GDAL are described in the detailed documentation for each GDAL format

12. Projection

This object defines the coordinate system to display your data.

To set up projections you must define two projection objects: one for the output image (in the MAP object) and one for each layer (in the LAYER objects) to be projected. MapServer relies on the PROJ.4 library for projections. Projection objects therefore consist of a series of PROJ.4 keywords, which are either specified within the object directly or referred to in an *epsg* file. An epsg file is a lookup file containing projection parameters, and is part of the PROJ.4 library.

The following two examples both define the same projection (UTM zone 15, NAD83), but use 2 different methods:

Example1: Inline Projection Parameters

```
PROJECTION
  "proj=utm"
  "ellps=GRS80"
  "datum=NAD83"
  "zone=15"
  "units=m"
  "north"
  "no_defs"
END
```

Example2: epsg Projection Use

```
PROJECTION
   "init=epsg:26915"
END
```

(this refers to an epsg lookup file that contains a '26915' code with the full projection parameters)

Example3: Inline Projection Parameters

```
PROJECTION
  "proj=latlong"
  "ellps=WGS84"
  "datum=WGS84"
END
```

Example4: epsg Projection Use

```
PROJECTION
   "init=epsg:4326"
END
```

Important Notes

- If all of your data in the mapfile is in the same projection, you DO NOT have to specify any projection objects. MapServer will assume that all of the data is in the same projection.

- If you specify a MAP-level projection, and then only one other LAYER projection object, MapServer will assume that all of the other layers are in the specified MAP-level projection.
- Always refer to the epsg file in lowercase, because it is a lowercase filename and on Linux/Unix systems this parameter is case sensitive.

For More Information

- If you get projection errors, refer to the MapServer Error page to check if your exact error has been discussed.
  - Search the MapServer-users email list archives, odds are that someone has faced your exact issue before.
- See the PROJ.4 user guides for complete descriptions of supported projections and coordinate systems.
- Refer to the Cartographical Map Projections page for background information on projections.

13. QueryMap

Defines a mechanism to map the results of a query. Starts with the keyword QUERYMAP and terminates with the keyword END.

COLOR [r] [g] [b]
: Color in which features are highlighted. Default is yellow.

SIZE [x][y]
: Size of the map in pixels. Defaults to the size defined in the map object.

STATUS [on|off]
: Is the query map to be drawn?

STYLE [normal|hilite|selected]

: Sets how selected features are to be handled. Layers not queried are drawn as usual.

  - Normal: Draws all features according to the settings for that layer.
  - Hilite: Draws selected features using COLOR. Non-selected features are drawn normally.
  - Selected: draws only the selected features normally.

14. Reference Map

Defines how reference maps are to be created. Starts with the keyword REFERENCE and terminates with the keyword END.

Three types of reference maps are supported. The most common would be one showing the extent of a map in an interactive interface. It is also possible to request reference maps as part of a query. Point queries will generate an image with a marker (see below) placed at the query point. Region based queries will depict the extent of the area of interest. Finally, feature based queries will display the selection feature(s) used.

COLOR [r] [g] [b]
: Color in which the reference box is drawn. Set any component to -1 for no fill. Default is red.

EXTENT [minx][miny][maxx][maxy]
: The spatial extent of the base reference image.

IMAGE [filename]
: Full filename of the base reference image. Must be a GIF image.

MARKER [integer|string]
: Defines a symbol (from the symbol file) to use when the box becomes too small (see MINBOXSIZE and MAXBOXSIZE below). Uses a crosshair by default.

MARKERSIZE [integer]
: Defines the size of the symbol to use instead of a box (see MARKER above).

MINBOXSIZE [integer]
: If box is smaller than MINBOXSIZE (use box width or height) then use the symbol defined by MARKER and MARKERSIZE.

MAXBOXSIZE [integer]
: If box is greater than MAXBOXSIZE (use box width or height) then draw nothing (Often the whole map gets covered when zoomed way out and it's perfectly obvious where you are).

OUTLINECOLOR [r] [g] [b]
: Color to use for outlining the reference box. Set any component to -1 for no outline.

SIZE [x][y]
: Size, in pixels, of the base reference image.

STATUS [on|off]
: Is the reference map to be created? Default it off.

15. Scalebar

Defines how a scalebar should be built. Starts with the keyword SCALEBAR and terminates with the keyword END.

Scalebars currently do not make use of TrueType fonts. The size of the scalebar image is NOT known prior to rendering, so be careful not to hard-code width and height in the <IMG> tag in the template file. Future versions will make the image size available.

BACKGROUNDCOLOR [r] [g] [b]
: Color to use for scalebar background, not the image background.

COLOR [r] [g] [b]
: Color to use for drawing all features if attribute tables are not used.

IMAGECOLOR [r] [g] [b]
: Color to initialize the scalebar with (i.e. background).

INTERLACE [true|false]
: Should output images be interlaced? Default is [on]. This keyword is now depcrecated in favour of using the FORMATOPTION "INTERLACE=ON" line in the OUTPUTFORMAT declaration.

INTERVALS [integer]
: Number of intervals to break the scalebar into. Default is 4.

LABEL
: Signals the start of a LABEL object

OUTLINECOLOR [r] [g] [b]
: Color to use for outlining individual intervals. Set any component to -1 for no outline which is the default.

POSITION [ul|uc|ur|ll|lc|lr]
: Where to place an embedded scalebar in the image. Default is lr.

POSTLABELCACHE [true|false]
: For use with embedded scalebars only. Tells the MapServer to embed the scalebar after all labels in the cache have been drawn. Default is false.

SIZE [x][y]
: Size in pixels of the scalebar. Labeling is not taken into account.

STATUS [on|off|embed]
: Is the scalebar image to be created, and if so should it be embedded into the image? Default is off. (Please note that embedding scalebars require that you define a markerset. In essence the scalebar becomes a custom marker that is handled just like any other annotation.)

STYLE [integer]
: Chooses the scalebar style. Valid styles are 0 and 1.

TRANSPARENT [on|off]
: Should the background color for the scalebar be transparent. This flag is now deprecated in favour of declaring transparency within OUTPUTFORMAT declarations. Default is off.

UNITS [feet|inches|kilometers|meters|miles]
: Output scalebar units, default is miles. Used in conjunction with the map's units to develop the actual graphic. Note that decimal degrees are not valid scalebar units.

16. Style

This object holds parameters for symbolization. Multiple styles may be applied within a class.

This object is new in 4.0 and is intended to seperate logic from looks. The final intent is to have named styles (Not yet supported) that will be re-usable through the mapfile. This is the new, preferred way of defining the appearance of an object, notably a class.

ANGLE [double]
: Angle, given in degrees, to draw the line work. Default is 0. For symbols of Type HATCH, this is the angle of the hatched lines. For its use with hatched lines, see Example#8 in the SYMBOL examples.

ANGLEITEM [string]
: Attribute/field that stores the angle to be used in rendering. Angle is given in degrees with 0 meaning no rotation.

ANTIALIAS [true|false]
: Should TrueType fonts and Cartoline symbols be antialiased.

BACKGROUNDCOLOR [r] [g] [b]
: Color to use for non-transparent symbols.

COLOR [r] [g] [b]
: Color to use for drawing features.

MAXSIZE [integer]
: Maximum size in pixels to draw a symbol. Default is 50.

MINSIZE [integer]
: Minimum size in pixels to draw a symbol. Default is 0.

MINWIDTH [integer]
: Minimum width in pixels to draw the line work.

OFFSET [x][y]
: Offset values for shadows, hollow symbols, etc ...

OUTLINECOLOR [r] [g] [b]
: Color to use for outlining polygons and certain marker symbols. Line symbols do not support outline colors.

SIZE [integer]
: Height, in pixels, of the symbol/pattern to be used. Only useful with scalable symbols. Default is 1. For symbols of Type HATCH, the SIZE is the distance between hatched lines. For its use with hatched lines, see Example#8 in the SYMBOL examples.

SIZEITEM [string]
: Attribute/field that stores the size to be used in rendering. Value is given in pixels.

SYMBOL [integer|string|filename]

: The symbol name or number to use for all features if attribute tables are not used. The number is the index of the symbol in the symbol file, starting at 1, the 5th symbol in the file is therefore symbol number 5. You can also give your symbols names using the NAME keyword in the symbol definition file, and use those to refer to them. Default is 0, which results in a single pixel, single width line, or solid polygon fill, depending on layer type.

: You can also specify a gif or png filename. The path is relative to the location of the mapfile.

WIDTH [integer]
: Width refers to the thickness of line work drawn, in pixels. Default is 1. For symbols of Type HATCH, the WIDTH is how thick the hatched lines are. For its use with hatched lines, see Example#8 in the SYMBOL examples.

17. Variable Substitution

New in MapServer 4.0, variables can be substituted within mapfile parameter values. At this time, cookie and CGI parameter values are supported. This allows mapserver mapfiles to be aware of a user's cookies (Good for implementing security), or non-mapserver request parameters (Good for integrating with other systems).

Syntax: '%' + variable name + '%'

Example 1. Connecting securely to a Spatial Database

You want to map some senstitive data held in a PostGIS database. The username and password to be used for the database connection are held in 2 cookies previously set by a seperate authentication mechanism, "uid" and "passwd".

CONNECTION "user=%uid% password=%passwd% dbname=postgis"

Example 2. Handling temporary files

You have a user based discovery application that generates shapefiles and stores them in a user's home directory on the server. The "username" comes from a cookie, the "filename" comes from a request parameter.

DATA "/home/%username%/tempshp/%filename%"

This feature is only available in the CGI version of MapServer through a mapfile pre-processor. If you are using MapScript, you will have to code the substitution logic into your application yourself (By writing your own pre-processor).

18. Web

Defines how a web interface will operate. Starts with the keyword WEB and terminates with the keyword END.

EMPTY [url]
: URL to forward users to if a query fails. If not defined the value for ERROR is used.

ERROR [url]
: URL to forward users to if an error occurs. Ugly old MapServer error messages will appear if this is not defined

FOOTER [filename]
: Template to use AFTER anything else is sent. Multiresult query modes only.

HEADER [filename]
: Template to use BEFORE everything else has been sent. Multiresult query modes only.

IMAGEPATH [path]
: Path to the temporary directory fro writing temporary files and images. Must be writable by the user the web server is running as. Must end with a / or depending on your platform.

IMAGEURL [path]
: Base URL for IMAGEPATH. This is the URL that will take the web browser to IMAGEPATH to get the images.

LOG [filename]
: File to log MapServer activity in. Must be writable by the user the web server is running as.

MAXSCALE [double]
: Maximum scale at which this interface is valid. When a user requests a map at a bigger scale, MapServer automatically returns the map at this scale. This effectively prevents user from zooming too far out.

MAXTEMPLATE [file|url]
: Template to be used if above the maximum scale for the app, useful for nesting apps.

METADATA

: This keyword allows for arbitrary data to be stored as name value pairs. This is used with OGC WMS to define things such as layer title. It can also allow more flexibility in creating templates, as anything you put in here will be accessible via template tags. Example:

```
METADATA
  title "My layer title"
  author "Me!"
END
```

MINSCALE [double]
: Minimum scale at which this interface is valid. When a user reuqests a map at a smaller scale, MapServer automatically returns the map at this scale. This effectively prevents the user from zooming in too far.

MINTEMPLATE
: Template to be used if above the minimum scale for the app, useful for nesting apps.

OUTPUTFORMAT [mime-type]
: Format of the query output. Default is "text/html". This is experimental, the use of the OUTPUTFORMAT object is recommended instead.

TEMPLATE [filename|url]
: Template file or URL to use in presenting the results to the user in an interactive mode (i.e. map generates map and so on ...)

Document information (version, author, etc.)

Release: 4.6

Last Updated: 2005/12/15

Author: Jean-François Doyon

Contact: jdoyon(at)ccrs.nrcan.gc.ca

Author: Jeff McKenna

Contact: jmckenna(at)dmsolutions.ca

Author: Steve Lime

Contact: steve.lime(at)dnr.state.mn.us

ANEXO 2 Referencia PostGIS.
Funciones OpenGIS:

1. AddGeometryColumn(varchar,varchar,varchar,integer,varchar,integer)
Sintaxis:
AddGeometryColumn(<nombre_db>,<nombre_tabla>,<nombre_columna>,<srid>,<type>,<dimension>)
Añade una columna geométrica a una tabla existente. SRID debe ser un valor entero que referencia una valor en la tabla *SPATIAL_REF_SYS*. El tipo debe ser una cadena en mayúsculas que indica el tipo de geometría, como, '*POLYGON*' o '*MULTILINESTRING*'.
2. DropGeometryColumn(varchar,varchar,varchar)
Sintaxis:DropGeometryColumn(<nombre_db>,<nombre_Tabla>,<nombre_columna>)
Elimina una columna geométrica de una tabla espacial.
3. AsBinary(geometry)
Devuelve la geometría pasándola a formato *well-known-binary* de OGC, usando la codificación *endian* del servidor donde se ejecuta la base de datos.
4. Dimension(geometry)
Devuelve 2 si la geometría es de 2D y 3 si es 3D.
5. Envelope(geometry)
Retorna un *POLYGON* que representa la caja circunscrita de la geometría.
6. GeometryType(geometry)
Retorna el tipo de geometría como una cadena. Ejemplo:
'*LINESTRING*','*POLYGON*','*MULTIPOINT*',etc.
7. X(geometry)
Encuentra y devuelve la coordenada X del primer punto de *geometry*. Devuelve NULL si no hay puntos.
8. Y(geometry)
Encuentra y devuelve la coordenada Y del primer punto de *geometry*. Devuelve NULL si no hay puntos.
9. Z(geometry)
Encuentra y devuelve la coordenada Z del primer punto de geometry. Devuelve NULL si no hay puntos.
10.NumPoints(geometry)
14.InteriorRingN(geometry,integer)
Devuelve el enésimo circulo interior del primer polígono en la *geometry*. Y NULL sino hay polígonos.
15.IsClosed(geometry)
Devuelve cierto si punto final = punto inicial de la geometría.
16.NumGeometries(geometry)
Si *geometry* es una *GEOMETRYCOLLECTION* devuelve el numero de geometrías que la componen. En caso contrario devuelve NULL.
17.Distance(geometry,geometry)
Devuelve la distancia cartesiana entre dos geometrías en unidades proyectadas.
18.AsText(geometry)
Devuelve una representación textual de una geometría. Ejemplo:
POLYGON(0 0,0 1,1 1,1 0,0 0)
19.SRID(geometry)
Devuelve un numero entero que es el identificador del sistema de referencia espacial de una geometría.
20.GeometryFromText(varchar,integer)
Sintaxis: GeometryFromText(<geom>,<SRID>) convierte un objeto de la representación textual a un objeto geometría.
21.GeomFromText(varchar,integer)
Igual que *GeometryFromText* .
22.SetSRID(geometry)
Establece el valor del *SRID* de una geometría al entero dado. Usado en la construcción de cajas circunscritas para consultas.
23.EndPoint(geometry)

Devuelve un objeto punto que representa el ultimo punto en la geometría.
24.StartPoint(geometry)
Devuelve un objeto punto que representa el primer punto en la geometría.
24.Centroid(geometry)
Devuelve un punto que representa el *centroide* de la geometría.
Otras funciones:
1. A<&B
Devuelve verdadero si la caja que circunscribe a A superpone o esta a la derecha de la de B.
2. A&>B
Devuelve verdadero si la caja que circunscribe a A superpone o esta a la izquierda de la de B.
3. A<<B
Devuelve verdadero si la caja que circunscribe a A esta estrictamente a la derecha de la de B.
4. A>>B
Devuelve verdadero si la caja que circunscribe a A esta estrictamente a la izquierda de la de B.
5. A~=B
Es equivalente al operador "igual que". Compara las 2 geometrías característica por característica y si todas coinciden devuelve verdadero.
6. A~B
Devuelve verdadero si la caja circunscrita de A esta contendida en la de B.
7. A&&B
Si la caja circunscrita de A se superpone a la de B devuelve Verdadero.
8. area2d(geometry)
Devuelve el área de una geometría si es un *POLYGON* o *MULTIPOLYGON*.
9. asbinary(geometry,'NDR')
Devuelve la geometría en el formato binario de OGC, usando codificacion littleendian. Se usa para sacar datos de la bd sin convertirlos a texto.
10.asbinary(geometry,'XDR')
Devuelve la geometría en el formato binario de *OGC*, usando codificación *bigendian*. Se usa para sacar información de la base datos sin convertirla a texto.
11.box3d(geometry)
Devuelve una *BOX3D* que representa la máxima extensión de la geometría.
12.collect(geometry)
Devuelve un objeto *GEOMETRYCOLLECTION* a partir de un conjunto de geometrías. Es una función de Agregación en la terminología de PostgreSQL.
Ejemplo:
SELECT COLLECT(GEOM) FROM GEOTABLE GROUP BY ATTRCOLUMN
Devuelve una colección separada para cada valor distinto de *ARRTCOLUMN*.
13.distance_spheroid(point,point,spheroid)
Devuelve la distancia lineal entre 2 latitud/longitud puntos dados de un spheroid. Ver el apartado length_spheroid().
14.extent(geometry)
Es una función de Agregación en la terminología de PostgreSQL. Trabaja sobre una lista de datos, de la misma manera que las funciones sum() y mean().
Ejemplo:
SELECT EXTENT(GEOM) FROM GEOMTABLE
Devuelve una BOX3D dando la máxima extensión a todas las características de la tabla.
SELECT EXTENT(GEOM) FROM GEOMTABLE GROUP BY CATEGORY
Devuelve un resultado extendido para cada categoria.
15.find_srid(varchar,varchar,varchar)
Sintaxis: find_srid(<db/esquema>,<tabla>,<columna>)
Devuelve el *SRID* de una columna dada buscando esta en la tabla *GEOMETRY_COLUMNS*. Si la columna geométrica no ha sido añadida con la función AddGeometryColumns() no funcionará.
16.force_collection(geometry)
Convierte una geometría en una *GEOMETRYCOLLECTION*. Esto se hace para

simplificar la representación WKB .

17.force_2d(geometry)

Fuerza la geometría a 2D así la representación de salida tendrá solo las coordenadas X e Y. Se usa porque OGC solo especifica geometrías 2D.

18.force_3d(geometry)

Fuerza la geometría a 3D. Así la todas las representaciones tendrán 3 coordenadas X,Y y Z.

19.length2d(geometry)

Devuelve la longitud 3d de la geometría si es una *linestring* o *multilinestring*.

20.length3d(geometry)

Devuelve la longitud 2d de la geometría si es una *linestring* o *multilinestring*.

21.length_spheroid(geometry,spheroid)

Calcula la longitud de una geometría en un elipsoide. Se usa si las coordenadas de la geometría están en latitud/longitud. El elipsoide es un tipo separado de la base de datos y se puede construir como:

SPHEROID [<NAME>,<SEMI-MAJOR AXIS>,<INVERSE FLTTENING>]

Ejemplo:

SPHEROID [“GRS_1980”,6378137,298,257222101]

Ejemplo de calculo:

SELECT
LENGTH_SPHEROID(
geometry_column,
'SPHEROID[“GRS_1980”,6378137,298.257222101]'
)
FROM geometry_table;

22.length3d_spheroid(geometry,spheroid)

Calcula la longitud de una geometría en un elipsoide, teniendo en cuenta la elevación.

23.max_distance(linestring,linestring)

Devuelve la distancia mas larga entre dos *linestring*.

24. mem_size(geometry)

Retorna el tamaño en bytes de la geometría.

25.npoints(geometry)

Devuelve el numero de puntos en la geometría.

26.nrings(geometry)

Si la geometría es un polígono o multipolígono devuelve el numero de círculos de la geometría.

27.num_sub_objects(geometry)

Devuelve el numero de objetos almacenados en la geometría. Se usa en Multigeometrías y *GEOMETRYCOLLECTION*s.

28.perimeter2d(geometry)

Devuelve el perímetro 2d de la geometría, si esa geometría es un polígono o un multipolígono.

29.perimeter3d(geometry)

Devuelve el perímetro 3d de la geometría, si esa geometría es un polígono o un multipolígono.

30.point_inside_circle(geometry,float,float,float)

Sintaxis:point_inside_circle(<geometry>,<circle_center_x>,<circle_center_y>,<radius>)

Devuelve verdadero si la geometría es un punto y esta dentro del circulo.

31.postgis_version()

Devuelve la versión de las funciones postgis instaladas en la bases de datos.

32.summary(geometry)

Devuelve un resumen en texto del contenido de esa geometría.

33.transform(geometry,integer)

Devuelve una nueva geometría con sus coordenadas transformadas la SRID dada por el parámetro integer. SRID debe existir en la tabla SPATIAL_REF_SYS.

34.translate(geometry,float8,float8,float8)

Traslada la geometría a la nueva localización usando los valores pasados como

desplazamientos X,Y,Z.

35.truly_inside(geometryA,geometryB)

Devuelve verdadero si alguna parte de B esta dentro de la caja circunscrita de A.

36.xmin(box3d) ymin(box3d) xmin(box3d)

Devuelve la mínima coordenada de la caja circunscrita.

37.xmax(box3d) ymax(box3d) zmax(box3d)

Devuelve la máxima coordenada de la caja circunscrita.