UNIVERSIDAD DEL AZUAY

Facultad de Ciencias de la Administración

Escuela de Ingeniería de Sistemas

Modelo de un Sistema Georreferenciado de Gestión de Cobros y Entrega de Pedidos

Tesis previa a la obtención del título de
Ingeniero de Sistemas

Autores: John Eduardo Álvarez Ullaguari
Mario Andrés Campos Crespo

Director: Ing. Paúl Ochoa

Cuenca, Ecuador

Enero 2011

DEDICATORIA

Dedico la presente tesis en primer lugar a Dios de quien he recibido muchas dones y bendiciones como el haberme dado a los seres que más amo en este mundo: mi esposa, Diana y mi pequeña hija Alisson, por ser la fuente de mi inspiración y motivación para superarme cada día más y así poder luchar para que la vida nos depare un futuro mejor.

Andrés

Mi tesis la dedico con todo mi amor y cariño a mi Dios que me brindo la oportunidad de vivir y de regalarme una familia maravillosa.

Con mucho cariño en especial a mis padres que me dieron la vida y han estado conmigo en todo momento. Gracias a mi Papá Jhon Alvarez por enseñarme con su ejemplo lo que cuesta luchar en los largos caminos de la vida y a mi Mamá Eufemia Ullaguari, por enseñar como se debe luchar con sus consejos, guía y paciencia, aunque hemos pasado momentos difíciles siempre han estado apoyándome y brindándome todo su amor, por todo esto les agradezco de todo corazón el que estén conmigo a mi lado.

De la misma manera a mis hermanos Rocío Alvarez y Andres Alvarez los quiero con todo mi corazón y este trabajo que me llevó mucho sacrificio es para ustedes, solamente les estoy devolviendo todo me han dado en este tiempo.

A mis profesores y compañeros por la confianza y apoyo brindado durante todo este tiempo, que el señor todo poderoso les ilumine y por todo el tiempo vivido me les llevo en el corazón.

John

AGRADECIMIENTOS

Agradecemos primero a Dios por la inteligencia y sabiduría que nos dio al nacer.

A todas aquellas personas que de forma directa e indirecta nos ayudaron para la realización de esta tesis.

Al Ing. Paúl Ochoa por ser nuestro tutor y guía, dándonos su asesoramiento y ayudarnos a coordinar el avance de nuestra tesis.

Al Ing. Diego Pacheco por compartir con nosotros sus conocimientos y ayudarnos a solventar algunas dificultades que se nos presentaron en la implementación de nuestro proyecto.

ÍNDICE DE CONTENIDOS

INDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

Los sistemas de Georreferenciación son ampliamente utilizados para resolver problemas de la planificación y gestión. Una de sus aplicaciones se desarrolla en este trabajo, en el cual se utilizan procedimientos para generar rutas óptimas de gestión de cobros y entrega de pedidos en la ciudad de Cuenca, publicando sus resultados en un entorno interactivo Web.

El proyecto fue elaborado utilizando la herramienta ArcGIS para construir las capas de los mapas, y otras herramientas y servicios de código abierto (Open Source) como: PostgreSQL, con su extensión PostGIS para manejar información georreferenciada, MapServer para publicar los mapas, el cliente ligero de programación mscross, lenguajes PHP, Javascript, XML, para la interacción con la interfaz del usuario, librerías Pgrouting.

ABSTRACT

Georeferencing systems are widely used to resolve planning and management problems. One of its applications is developed in this work. In which procedures to generate optimum routes for the management of payments and deliveries of orders in the city of Cuenca, puslishing its results in an interactive Web environment.

The project was created using ArcGIS to construct the mapping layers, and other Open source tools and services such as; PostgreSQL, with its extention PostGIS to manage georeferenced information, MapServer to publish the maps, mscross light-client programming, PHP, Javascript, and XML languages for the interaction with the user interface, and Pgrouting libraries.

INTRODUCCIÓN

La tendencia actual para representar los mapas generados en los sistemas de información geográfica (SIG), es hacerlo en ambientes web, ya que resulta un medio idóneo para divulgar la información particularmente en lo que se refiere a la representación visual de territorios, rutas viales y demás elementos geográficos. En el desarrollo actual de los SIG es deseable y necesario dar un valor agregado a todo esta información dando la capacidad de generar procesos automatizados y de transacciones que proporcionen resultados rápidos y confiables para de este modo facilitar a los administradores y altos mandos de una organización la toma de decisiones de forma oportuna y personalizada según sus necesidades.

Por ello el proyecto se constituye en una herramienta en la cual se pueda obtener información geográfica de los puntos de entrega y el mismo aplicativo nos sugiera rutas optimas de recorrido para gestión de cobros y entrega de pedidos a clientes, mejorando el desempeño de los cobradores control y el seguimiento de los ejecutivos, con loc cual contribuye al desarrollo de las empresas que brindan estos servicios.

El documento se encuentra desarrollado en cuatro capítulos, en el primero se presenta un marco teórico sobre algunas generalidades de los SIG y las herramientas de software libre (OpenSource) utilizadas en la implementación del prototipo. El Capítulo 2 trata sobre diferentes aspectos relacionados a la recolección y levantamiento de información a partir de entrevistas, reuniones, además de la identificación de procesos, operaciones a ser utilizados en la implementación del software. El Capítulo 3 se centra en el Análisis, diseño e implementación de la aplicación, aquí especificamos los pasos que seguimos durante la construcción e implementación del mismo. Por último, el Capítulo 4 consta de un manual de usuario, para atender las necesidades de los usuarios y así sacarle mayor provecho del software.

CAPITULO 1

MARCO TEORICO

INTRODUCCIÓN.

En la actualidad, los Sistemas de Información Geográfica (SIG) han logrado ampliar su utilidad práctica debido al auge de las aplicaciones web, cada vez más enfocadas al usuario final, especialmente a lo que tiene que ver con la gestión de servicios, con información cartográfica e interfaces gráficas más enriquecidas.

En este capítulo, se anotan brevemente algunos conceptos y características básicas de los SIG y herramientas que fueron utilizadas en la implementación y que nos permitieron tener elementos de decisión según los requerimientos que se presentaron en nuestro proyecto.

1.1. Definición, funciones y aplicación de los SIG

Un SIG es una herramienta capaz de combinar información gráfica (mapas) y alfanumérica (estadísticas) para obtener una información derivada sobre el espacio. (CIEMAT, 2000)

La utilización de un SIG se justifica en cinco aspectos básicos:

- Un SIG nos permite realizar comparaciones entre escalas y perspectivas emulando una cierta capacidad de representación de diferentes lugares al mismo tiempo.
- Un SIG nos permite diferenciar entre cambios cualitativos y cuantitativos; aportándonos una gran capacidad de cálculo.
- Un SIG nos permite gestionar un gran volumen de información a diferentes escalas y proyecciones.
- Un SIG integra espacialmente datos tabulares y geográficos junto a cálculos sobre variables (topología).
- Un SIG admite multiplicidad de aplicaciones y desarrollos; poniendo a nuestra disposición herramientas informáticas estandarizadas.

Los campos de aplicación de los SIG son numerosos así podríamos hablar de aplicaciones socioeconómicas, forestales, catastrales, etc. Pero también van encaminados a aquellos desarrollos informáticos o a la construcción de productos específicos para resolver un proyecto como es el caso de nuestro proyecto de tesis.

1.1.1. Software GIS: ArcGis.

ArcGis es una serie integrada de software de Sistemas de Información Geográfica (SIG), creada por el Instituto de Investigación ESRI (*Enviromental Sensitivies Research Institute*), que trabaja como un motor compilador de información geográfica alfanumérica (Bases de Datos) y gráfica (Mapas), con lo cual permite visualizar, crear, manipular y gestionar información geográfica como lugares, direcciones, áreas urbanas y rurales; regiones, etc.

Figura 1. Software ArcGis
Fuente: (ESRI, 2002)

Además, el software contiene herramientas de administración de bases de datos para tabular la información, como la *metadata*, la *geodatabase* que dan un soporte adecuado en la creación y la organización de un proyecto GIS.

De nuestra experiencia con el software vemos que tiene una interfaz gráfica amigable, en la cual se puede desplegar de manera rápida la información geográfica. A sí mismo el aprendizaje del software es rápido, teniendo algunos conocimientos de Sistemas de Información Geográfica previos, contiene una gran ayuda en línea con muchos recursos disponibles en: http://support.esri.com/index.cfm?fa=software.gateway

1.2. Software Gestor de base de datos: PostgreSQL.

PostgreSQL es un Sistema de Gestión de Bases de Datos Objeto-Relacionales (*ORDBMS*), fue desarrollado en la Universidad de California en Berkeley Computer Science Department. Distribuido bajo licencia BSD (*Berkeley Software Distribution*) y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado y en sus últimas versiones no tiene nada que envidiarle a otras bases de datos comerciales como Oracle o DB2.

Desarrollado en lenguaje C, con implementación de algunas extensiones de orientación a objetos. Posee control de concurrencia multiversión, lo que mejora sensiblemente las operaciones de bloqueo y transacciones en sistemas multiusuario, también soporte para vistas, claves foráneas, integridad referencial, disparadores, procedimientos almacenados, subconsultas, y casi todos los tipos y operadores soportados actualmente.

PostgreSQL utiliza un modelo cliente/servidor y usa "multiprocesos" en vez de "multihilos" para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. (Martínez, 2009).

1.2.1. La extensión PostGIS para PostgreSQL

PostGIS es una extensión al sistema de base de datos objeto-relacional PostgreSQL. Permite el uso de objetos GIS(*Geographic Information Systems*).

PostGIS incluye soporte para índices GiST basados en R-Tree, y funciones básicas para el análisis de objetos GIS. Esta creado por Refractions Research Inc, como un proyecto de investigación de tecnologías de bases de datos

espaciales. Está publicado bajo licencia GNU. Con PostGIS podemos usar todos los objetos que aparecen en la especificación OpenGIS como puntos, lineas, polígonos, multilíneas, multipuntos, y colecciones geométricas. (Ramsey, 2008)

1.3. Servidor de Mapas: MapServer

Un servidor de mapas es una Aplicación que publica información espacial, georreferenciada a través de Internet. Cuenta con gran parte de la funcionalidad SIG, es decir la posibilidad de navegar por el mapa, encontrar e identificar elementos, trabajar con medidas y escalas, seleccionar y crear áreas de influencia en torno a un objeto entre otras.

MapServer es un servidor de mapas que permite publicar e interactuar con mapas digitales desde páginas Web. MapServer es un ambiente de desarrollo (*OpenSource*) para construir las aplicaciones del Internet espacialmente-habilitadas. (Jaramillo, 2005). El servidor MapServer se encarga de leer la información de mapas, presentarlo en páginas Web y permitir la interacción del usuario con el mapa.

Las principales características (Ortega y Flores, 2009), incluyen:

- Soporte para visualizar y consultar cientos de rasters, vectores y formatos de base de datos.
- Máxima interacción de los usuarios con la información geográfica.
- Soporte de varios lenguajes tipo script y habientes de desarrollo como (PHP, Python, Perl, Ruby, Java, .NET)
- Soporta diversos tipos de proyecciones.

1.4. Lenguajes y técnicas de programación Web: HTML, PHP, JavaScript, XML, AJAX

1.4.1. HTML

Utiliza una codificación genérica, la cual hace uso de TAGS o etiquetas. Proporciona al usuario la información en una manera interactiva, haciendo uso del hypertexto , o texto con enlaces hacia otros lugares del Web, o

hacia inserciones de multimedia (videos, sonidos, gráficos, etc.). Además es universal y no depende del sistema operativo que se esté utilizando. (Posada, 2006).

1.4.2. PHP

El lenguaje PHP es un lenguaje de programación de estilo clásico, es decir que es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones, etc. No es un lenguaje de marcas como podría ser HTML o XML, es más bien similar a los lenguajes JavaScript o C. La ventaja que tiene PHP sobre otros lenguajes de programación es que nos permite intercalar las sentencias PHP en las paginas HTML.

El programa PHP es ejecutado en el servidor con lo cual accede a recursos como por ejemplo una base de datos. Los resultados son enviados al navegador. El resultado es normalmente un pagina HTML. Aunque es independiente del navegador, para que sus páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

1.4.3. JAVASCRIPT

Se trata de un lenguaje de tipo script compacto, basado en objetos y guiado por eventos diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet.

Los programas JavaScript van incrustados en los documentos HMTL, y se encargan de realizar acciones en el cliente, como pueden ser pedir datos, confirmaciones, mostrar mensajes, crear animaciones, comprobar campos. Es dinámico, responde a eventos en tiempo real. Con esto podemos cambiar totalmente el aspecto de nuestra página al gusto del usuario, evitándonos tener en el servidor una página para cada gusto, hacer cálculos en base a variables cuyo valor es determinado por el usuario, etc.

1.4.4. XML

XML es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un papel fundamental en el intercambio de una gran variedad de

datos. Es un lenguaje muy similar a HTML_pero su función principal es describir datos y no mostrarlos como es el caso de HTML. XML es un formato que permite la lectura de datos a través de diferentes aplicaciones. Sirve para estructurar, almacenar e intercambiar información. (W3C, 2005)

Ejemplo de documento XML:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<libro>
 <titulo></titulo>
 <capitulo>
  <titulo></titulo>
  <seccion>
   <titulo></titulo>
  </seccion>
 </capitulo>
</libro>
```

1.4.5. AJAX

Acrónimo de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML* (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (*Rich Internet Applications*). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano.

De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. Cuando estas tecnologías se combinan en un modelo AJAX, es posible lograr aplicaciones web capaces de actualizarse continuamente sin tener que volver a cargar la página completa. Esto crea aplicaciones más rápidas y con mejor respuesta a las acciones del usuario. (Wikipedia, AJAX, 2010)

1.5. Clientes ligeros Web para SIG

Los clientes ligeros web para SIG son aplicaciones de internet que se encargan de visualizar información geográfica y permiten su manipulación a través de herramientas básicas de navegación y análisis. (Carrillo, 2009).

Existen varios proyectos de software libre y de código abierto que facilitan la administración, el desarrollo y la personalización de este tipo de aplicaciones, las cuales consumen servicios web y comunican al usuario con tareas avanzadas que se realizan en el servidor.

La mayoría de proyectos gira en torno a dos paradigmas: UMN MapServer y OpenLayers. Los clientes UMN aprovechan características como: mapa, escala, mapa de referencia, herramientas de navegación básica, identificación de objetos espaciales; y su Interfaz de Programación de Aplicaciones (API) llamada MapScript que ha sido implementada en diferentes lenguajes de programación como PHP, Python, Perl y Ruby., entre estos clientes tenemos AppForMap, GeoMOOSE y msCross. Por otra parte, la nueva generación de clientes utiliza OpenLayers debido a su óptimo rendimiento en tareas de renderización de mapas en la web, entre estos tenemos AppForMap e i3Geo y Flamingo (i3Geo).

A continuación mostraremos una tabla comparativa entre el cliente msCross que utilizamos para implementar en nuestro proyecto y el cliente OpenLayers, mostrando en forma de ventajas y desventajas diferentes parámetros como son: su licencia, su país de origen, su soporte comercial, los lenguajes de programación que permiten, su independencia con respecto a programas servidores de mapas y el manejo de metadatos que poseen, entre otros:

Parámetro / Cliente	msCross (UMN MapServer)	OpenLayers
Licencia	GNU GPL (OpenSource)	BSD-style (OpenSource)
País de origen	Italia	Estados Unidos
Soporte OSGEO (Open Source Geospatial Foundation)	No	Si (Graduado)
Lenguaje en el que está escrito	Javascript	Javascript
Cliente Web AJAX	Si	Si
¿Incluye componente de metadatos?	No	Si (especificación CSW)
Permite implementar servicios WFS y WMS	Si	Si

Mapas dinámicos	Dinámicos	Altamente Dinámicos
Permite dibujar líneas, polígonos, puntos	No	Si
Página oficial	http://sourceforge.net/projects/mscross/	http://openlayers.org
Documentación	http://datacrossing.crs4.it/en_Documentation_mscross.html	http://trac.openlayers.org/wiki/Documentation http://openlayers.org/dev/examples/

Tabla 1. Comparación entre Clientes msCross y OpenLayers
Fuente: (García y Autores)

1.5.1. El cliente ligero msCross

Inicialmente desarrollado para ser una interfaz JavaScript para UMN MapServer. msCross es un cliente WEB AJAX. Fue desarrollado para permitir a los usuarios visualizar dinámicamente capas de información geográfica en la web. Corresponde a los denominados WMS (Web Map Service). Fue desarrollado para ser multi-navegador. El objetivo principal es permitir crear mapas simples con un conjunto de aplicaciones API de Google, usando sólo software libre.

Figura 2. Cliente msCross
Fuente: (Cau y Manca, 2006)

Características principales:

- Software libre, distribuido bajo licencia GPL (Open Source)

- Funciona del lado del cliente
- Multi-navegadores / multi-plataforma
- Uso simple (sólo un archivo javascript, no requiere instalación)
- Fácilmente adaptable y extensible con programación.
- Basado en tecnología AJAX (Asynchronous JavaScript y XML)
- Permite interactuar con capas de puntos.
- Soporte al cliente WFS, OGC (Web Feature Service, Open Geospatial Consortium).
- Fácil personalización de barra de herramientas.
- Nuevo modo de depuración

Ventajas:

- Maneja menor cantidad de archivos.
- Más eficiente en internet pues utiliza menor cantidad de recursos.

Desventajas:

- Limitada cantidad de herramientas disponibles.
- Se necesita mayor programación para agregar funcionalidad extra.

Algunos proyectos que utilizan msCross se pueden ver en el sitio web de *SourceForge:* http://sourceforge.net/projects/mscross

CONCLUSIONES.

En la actualidad existen diferentes tecnologías, lenguajes y técnicas de programación tanto de lado del cliente como del servidor, y muchas herramientas Web que podemos utilizar en nuestros desarrollos. Los Servidores de Mapas son la evolución de los Sistemas de Información Geográfica hacia Internet y los lenguajes de programación HTML, PHP, JavaScript, XML, AJAX, cliente AJAX MsCross y otros, permiten al usuario tener una interacción dinámica con el contenido cartográfico. En internet hay un amplio marco teórico y práctico sobre las mismas y algunos de estos desarrollos fueron utilizados e implementados en este proyecto.

CAPITULO 2.

RECOLECCIÓN Y LEVANTAMIENTO DE INFORMACIÓN

INTRODUCCIÓN.

Una de las etapas primordiales en la realización de un proyecto de software, es la recolección e identificación de la información necesaria para la elaboración del mismo, puesto que constituye la base para el análisis, diseño e implementación de la aplicación. De las reuniones y entrevistas mantenidas hemos obtenido información explicativa sobre las herramientas SIG y el manejo básico de la gestión de cobros y cartera. Al mismo tiempo se detalla la cartografía base de la ciudad de Cuenca con la que hemos trabajado y los procesos de análisis de redes y depuración de la información obtenida.

2. Recolección de la información

2.1. Entrevistas y reuniones

A través del Ingeniero Paúl Ochoa y de los integrantes del departamento de investigación y desarrollo del Instituto de Estudios de Régimen Seccional del Ecuador (IERSE). Fueron abordados y aclarados aspectos sobre el uso de las herramientas SIG que son implementados en diferentes portales Web.

Se destacó y señalo la importancia que tienen ya que permiten al usuario interactuar de forma dinámica con la información cartográfica publicada, es decir, el usuario puede ubicar dentro del mapa diversas locaciones geográficas como un determinado sector, calle, avenida, etc., estos objetos geográficos a su vez poseen toda la información espacial y sus atributos dentro de una base de datos geográfica, con lo cual nos asegura el trabajar con información consistente y real.

De las reuniones mantenidas con el Eco. Esteban Vintimilla, gerente de cartera y crédito en Marcimex. En lo referente a la información de gestión de cobros.

Fueron tomadas algunas ideas a saber que uno de los pasos más importantes dentro del ciclo de cobranza, es hacer un seguimiento. Por lo que el cobrador debe asegurarse de que el deudor cumpla con su compromiso y que muchos ejecutivos en ventas creen en promesas de pago, que no siempre son recaudaciones. Por ello es importante que las mismas se encuentren debidamente soportadas y recibir un seguimiento adecuado para confirmar los compromisos de los deudores y reducir el índice de incumplimientos. Esto comprende a los clientes y cobradores que se encuentran organizados por zona, y para el manejo de cartera donde se consideran los créditos, vencimientos, abonos, pagos, deudas para los cuales entre más atributos posean es mejor puesto que toda la información requerida quedara debidamente registrada y almacenada.

2.2. Base cartográfica

La información cartográfica de este proyecto, está conformada por archivos y bases de datos que contienen información geográfica de la ciudad de Cuenca, constituida por los archivos de Manzanas, Calles y Vías. Estos archivos son proporcionados por el IERSE, se encuentran en el Sistema de Coordenadas Geográficas PSAD56/UTM Zona 17.

CARTOGRAFÍA	FORMATO	TIPO	FUENTE	OBSERVACIONES	
MANZANAS.SHP	SHAPE	POLÍGONO	IERSE	PSAD56	MANZANAS DE LA CIUDAD
VIAS_CUENCA.SHP	SHAPE	LINEA	IERSE	PSAD56	VIAS DE LA CIUDAD
CALLES.SHP	SHAPE	LÍNEA	IERSE	PSAD56	AV., CALLES, RETORNOS, ETC.
PATH.SHP	SHAPE	LÍNEA	AUTORES	PSAD56	CAPA PARA DIBUJAR LA RUTA

Tabla 2. Archivos Recolectados
Fuente: (Autores)

2.2.1 Cartografía Base

Manzanas

Nombre del Archivo: 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

Tabla 3. Tabla de Atributos de Manzanas.shp
Fuente: (Autores)

Vías_cuenca

Nombre del Archivo: vías_cuenca.shp

CAMPO	DESCRIPCION	TIPO	LARGO
FID	Identificador del objeto	Object ID	4
SHAPE	Geometría del objeto	Line	
NOMBRE	Nombre de la via	String	50
CATEGORIA	Tipo de via	String	20
ONEWAY	Sentido de la via	String	9
METERS	Metraje de la via	Numeric	-

Tabla 4. Tabla de Atributos de vías_cuenca.shp
Fuente: (Autores)

Calles

Nombre del Archivo: calles.shp

CAMPO	DESCRIPCION	TIPO	LARGO
FID	Identificador del objeto	Object ID	4
SHAPE	Geometría del objeto	Line	
NOMBRE	Nombre de la calle	String	50

Tabla5. Tabla de Atributos de calles.shp
Fuente: (Autores)

Path (Layer que se utiliza para dibujar la ruta)

Nombre del Archivo: path.shp

CAMPO	DESCRIPCION	TIPO	LARGO
FID	Identificador del objeto	Object ID	4
SHAPE	Geometría del objeto	Line	

Tabla6. Tabla de Atributos de path.shp
Fuente: (Autores)

2.3. Modelo de Redes para cálculo de rutas: Redes de Transporte

El análisis con modelo de redes se utiliza para calcular rutas que sean óptimas para la planificación; en nuestro caso específico es utilizado para crear rutas óptimas de un punto a otro por las calles de la ciudad de Cuenca.

Figura 3. Ruta Ciudad de Cuenca
Fuente: (Espinoza y Espinoza, 2010)

Este tipo de redes son conocidas como redes de Transporte y permiten definir exactamente la información de rutas entre dos puntos específicos (por ejemplo la ruta más corta entre la calle Lamar y Juan Montalvo, etc.,) características propias del transporte de una área determinada, esta

información se puede manejar dentro de un shapefile, o una geodatabase, la información manejada dentro de esta red es la representación de un flujo totalmente libre al interior de la red, pudiendo surgir variables independientes dentro de la misma red, es decir si entre el aeropuerto y el parque central, existen semáforos, o están las calles obstaculizadas obviamente estas variables afectaran a la ruta. (Barrientos, 2007)

CONCLUSIONES.

Para poder llevar a cabo cualesquier tipo de proyecto informático es indispensable tener como fuente de información los conocimientos de los "expertos" o personas involucradas en el proyecto quienes a través de entrevistas y reuniones nos esclarecieron los temas relacionados con nuestro proyecto que fueron de mucha importancia para el análisis posterior.

También hay que considerar que un SIG distribuye la información geográfica que dispone en un conjunto de capas, pues esta estructura le permite realizar un análisis adecuado de sus características temáticas y espaciales, por lo que es imprescindible una adecuada recolección, selección e integración de las capas a ser utilizadas, las mismas que están conformadas por un conjunto de metadatos, que nos permiten realizar consultas que facilitan la ubicación de información geográfica por parte del usuario.

Las capas con las que estamos trabajando, son las indispensables para la construcción del mapa temático, su visualización y búsquedas, ya que poseen la información cartográfica que requiere el usuario.

CAPITULO 3.

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO

INTRODUCCIÓN.

Después de evaluar y analizar el conjunto de herramientas y técnicas de programación requeridas para llevar a cabo nuestra aplicación, y levantada la información de las capas cartográficas requeridas, procederemos al desarrollo de nuestro prototipo. En este capítulo detallamos cada una de las etapas efectuadas para la realización del proyecto, describiendo los pasos esenciales que debemos realizar, así como las dificultades que hemos tenido durante su ejecución y las soluciones dadas a las mismas. En forma general, en el capítulo se describen tres aspectos fundamentales que se llevaron a cabo, que son:

- La publicación del mapa de la ciudad de Cuenca con las diferentes capas temáticas que lo conforman, a través del cliente ligero msCross utilizando el servidor de mapas MapServer.
- Instalación y configuración de la librería PgRouting para la funcionalidad de ruteo a la base de datos.
- Implementación de las diferentes funciones e interfaces para la Gestión de Cobros y Entrega de Pedidos.

3.1. Servicio de publicación de mapas (WMS)

Un Servicio de publicación de mapas (WMS) nos permite producir mapas de forma dinámica a partir de información geográfica vectorial o raster presentando la información como imágenes digitales susceptibles de ser visualizadas en pantalla. En la realización de nuestro proyecto necesitamos utilizar estos servicios que nos permitan la visualización del mapa de la ciudad de Cuenca. Los diferentes temas que involucran la publicación de mapas en la Web han sido tratados ampliamente en varias monografías y trabajos de tesis, por lo cual nuestro objetivo es explicar en forma breve las

herramientas principales y configuraciones realizadas para así dar una orientación básica sobre estos temas.

3.2. Geodatabase generado desde ArcGis

A más de las capas de manzanas y calles que permiten referenciar el mapa de la ciudad de Cuenca, requerimos de una capa temática de información de aquellas vías de la ciudad en donde se realizará la creación de la ruta óptima que deberán seguir los cobradores desde las diferentes agencias hacia los clientes.

La capa de vías: vías_cuenca.shp corresponden a un sector de la ciudad (podría aplicarse a toda la ciudad) de que serán utilizadas para generar la ruta y que fue obtenida a partir del Geodatabase generado con la extensión Network Analyst de la herramienta ArcGis 9.2.

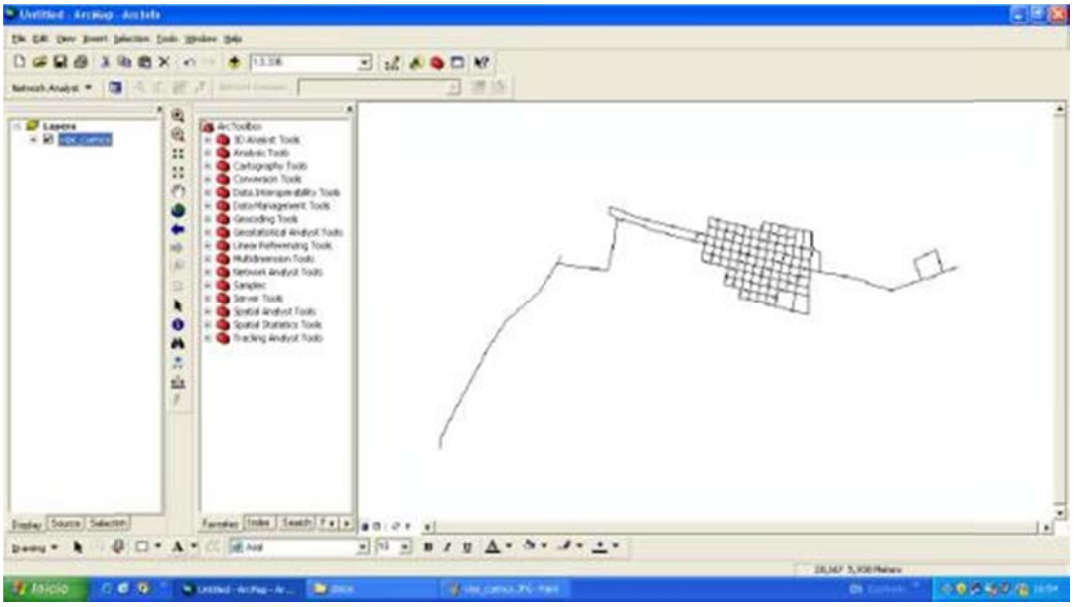


Figura 4. Geodatabase creada desde ArcGis

Fuente (Autores)

Para una explicación detallada de cómo utilizar la extensión Network Analyst de ArcGis y la creación de un Geodatabase, Feature Dataset, etc se puede consultar el trabajo desarrollado por las Hermanas Espinoza, en su tesis: "Sistema de Redes Viales de la Ciudad de Cuenca y del Ecuador para la navegación con GPS" (Espinoza M. Andrea, Espinoza M. Gabriela, 2010)

3.3. Generación del archivo de texto .map

Para poder publicar el mapa con las distintas capas temáticas en nuestro Servidor de Mapas, en primer lugar es necesario que con la información disponible, procedamos a generar el archivo de texto .map que es el principal archivo en la configuración de MapServer pues contiene los parámetros que definen las capas disponibles en el servicio, el estilo que se representarán, su simbología, formato se generará la imagen, el sistema de referencia, etc.

Para generar este archivo debemos exportar los datos desde ArcMap agregando la extensión "MXD to Web Map Service configuration file" cuyas bibliotecas y los respectivos pasos para poder registrarlas y configurarlas se pueden obtener de la url: http://arcscripts.esri.com/details.asp?dbid=12766

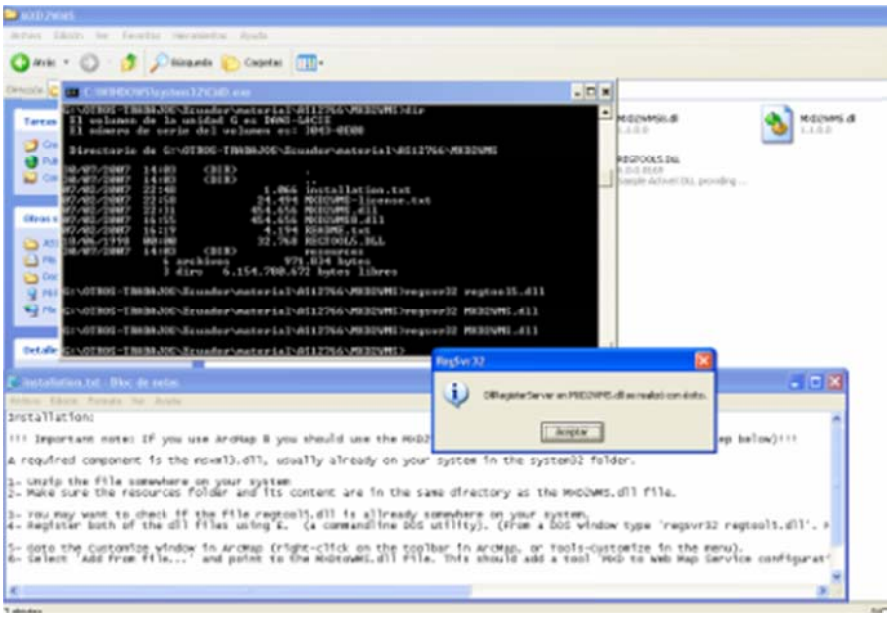


Figura 5: Registrar extensión MXD
Fuente: (Pacheco & Sellers, 2008)

Una vez conseguido registrar la extensión MXD correctamente y añadido el icono en ArcMap, damos clic en este y se visualizará una ventana en la cual seleccionamos las capas que queremos que se generen y luego guardamos el archivo con la extensión .map

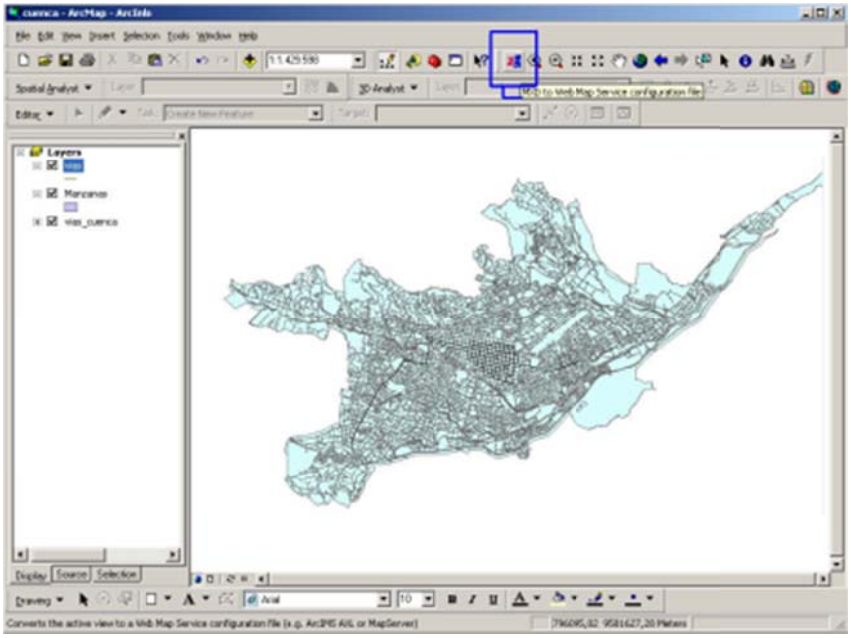


Figura 6: Exportar Archivo .map desde ArcMap
Fuente (Autores)

3.4. Instalación de MapServer y visualización del mapa

En el caso de MapServer la instalación es sencilla, será necesario descargar el software que en nuestro caso corresponde al paquete MS4W y consultar la documentación que es muy amplia, para encontrar estos recursos y otras herramientas disponibles puede dirigirse a la url: http://maptools.org/ms4w/index.phtml?page=downloads.html

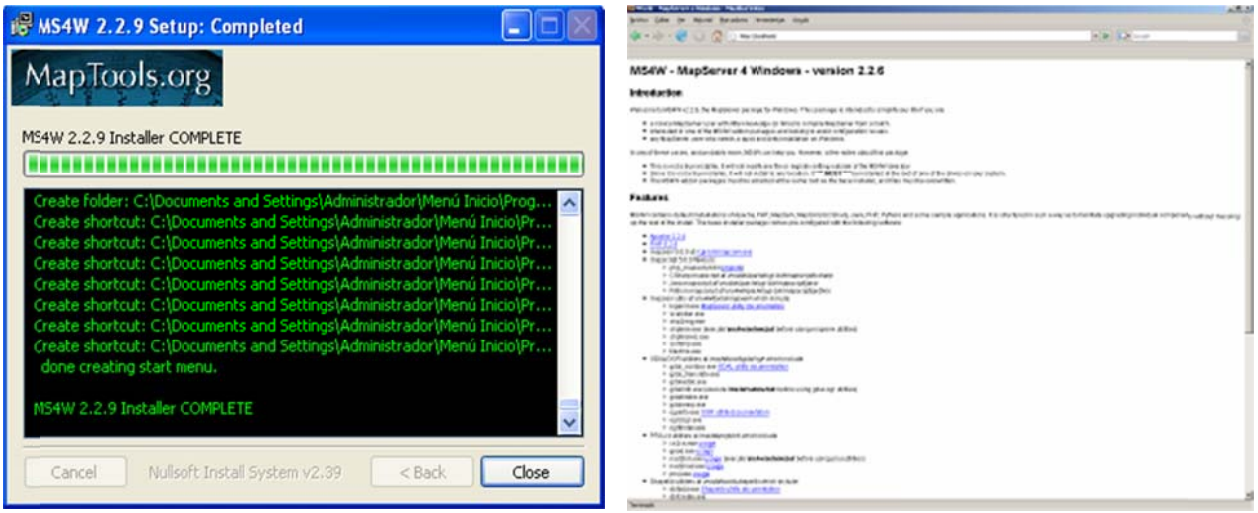


Figura 7: Instalación y ejecución de Mapserver en Windows
Fuente: (Ortega & Flores, 2009)

Con el propósito de probar nuestro Servidor de Mapas recién instalado y una vez definido correctamente nuestro archivo *MapFile* al que le hemos denominado **cuencaesf.map** y cuyo código fuente completo lo podemos ver en el **Anexo 3.** Probaremos la visualización sin problemas del mapa ingresando la siguiente URL en el browser:

> http://localhost///cgi-bin/mapserv.exe?mode=map&map=/ms4w/Apache/htdocs/cuenca/cuencaesf.map&mapext=713606.8460999994+9675184.5942+733223.2157000004+9686954.41596&mapsize=1000+600&layers=SARDINIA%20manzanas%20vias_cuenca%20&undefined

Con esta URL estamos invocando al CGI mapserv.exe, los parámetros son:

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= Coloque la dirección en donde se encuentra el archivo .map

Layers= Especifique el nombre de cada una de las capas que requiera visualizar, en nuestro caso hacemos referencia a las capas manzanas y vías_cuenca

El resultado que observará en su navegador, será similar al siguiente:

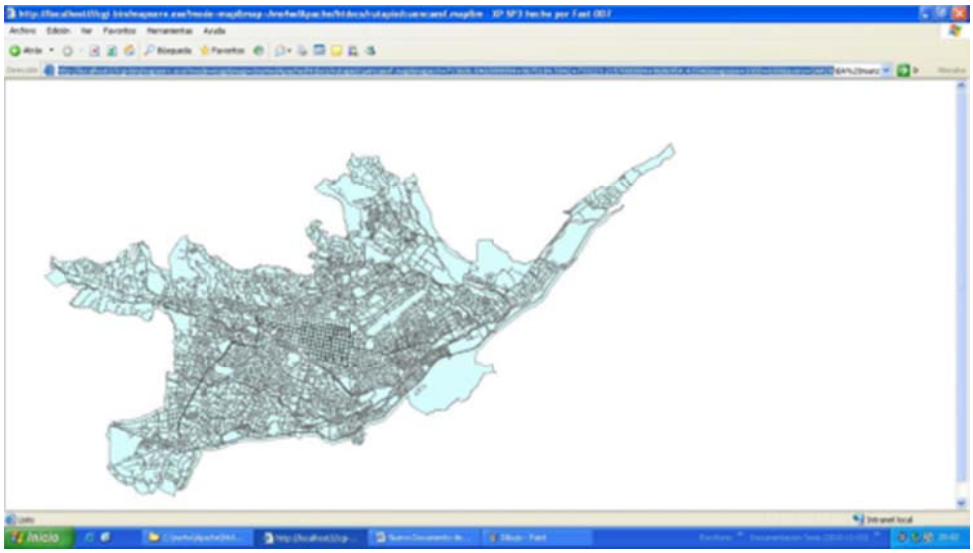


Figura 8: Visualización del mapa generado en MapServer
Fuente: (Autores)

En caso de tener problemas con el archivo .map al momento de publicar se recomienda seguir los pasos y correciones que se que se detallan en el **Anexo 1.**

3.4.1. Conexión de Mapserver con PostgreSQL y PostGIS

Las tablas de la base de datos contienen información cartográfica almacenada. Con el propósito de visualizar esta información en el browser se debe editar el .map y añadir por cada layer las siguientes líneas, por ejemplo para el layer vías_cuenca:

```
CONNECTIONTYPE postgis
CONNECTION "host=localhost dbname=cuenca
user=postgres password=postgres port=5432"
DATA "the_geom from vias_cuenca"
```

3.4.2. Definir proyección de las capas

Para verificar la proyección de las capas editar nuevamente el .map y añadir por cada layer las siguientes líneas, por ejemplo para el layer vías_cuenca:

```
LAYER
        NAME 'vias_cuenca'
        GROUP 'vias_cuenca'
```

```
        CONNECTIONTYPE postgis CONNECTION "host=localhost
dbname=cuenca user=postgres password=postgres port=5432"
        DATA "the_geom from vias_cuenca"

        PROJECTION
            "init=epsg:24877"
        END #end projection

        TYPE line
        STATUS ON
        TOLERANCE 8 #default is 3 for raster, 0 for vector
        #TOLERANCEUNITS meters #default is meters,
[pixels | feet | inches | kilometers | meters | miles | dd]

        TEMPLATE "query.html"

        # These classes are based on the first set of symbols from a
group renderer. Not optimal yet!
            CLASS
                COLOR 0 0 0
                MINSIZE 2
                MAXSIZE 2
            END #end style
    END #end layer
```

3.5. Instalación de PostgreSQL con la extensión PostGIS

Para una instalación personalizada de PostgreSQL con la extensión PostGIS descargue el software y consulta la documentación para ello dirigirse al sitio oficial en: http://www.postgresql.org/

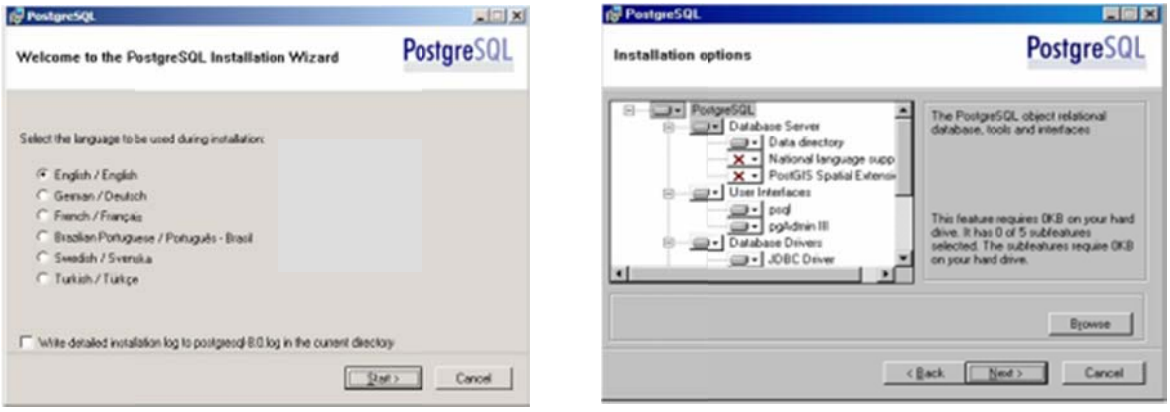


Figura 9: Instalación PostgreSql con extensión PostGIS
Fuente: (Autores)

No olvide escoger la extensión PostGis. El uso de la extensión PostGis de PostgreSQL permite el manejo y manipulación de la información geográfica. Luego establezca un usuario y contraseña para configurar el servicio PostgreSQL.

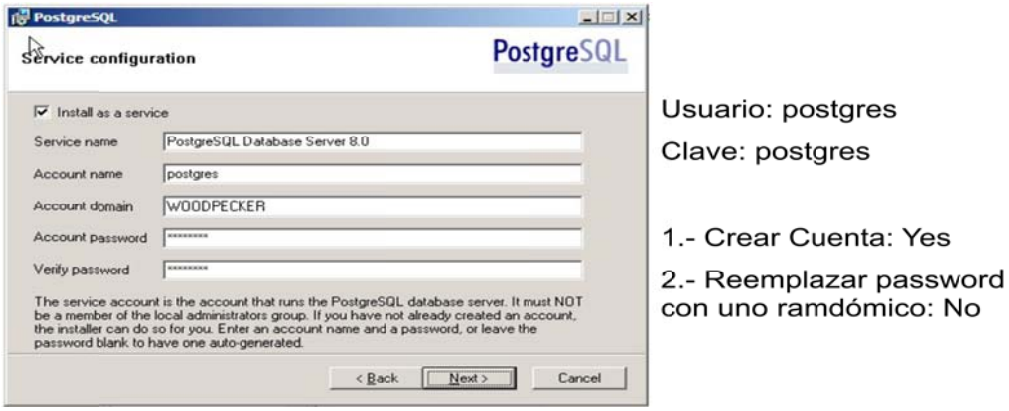


Figura 10: Configuración de usuario y contraseña
Fuente: (Pacheco, 2008)

3.6. Publicación con msCross

msCross obtiene la imagen del mapa preguntando directamente a mapserver, y genera el url necesario para asignarlo a un elemento de HTML. Esta interfaz fue desarrollada para permitir que los usuarios muestren dinámicamente las capas geográficas de la información sobre la Web y creen usos de Estilo, usando un Software Libre. (Mariño y Moncayo, 2008)

En nuestro proyecto implementamos esta librería en bloques tipo javascript desde algunos archivos .php en donde se llama al archivo mscross.js para que incluya todas las funciones y herramientas necesarias para publicar la información.

Aquí se muestra un ejemplo en donde se llama al archivo .map y se muestran todas las capas necesarias:

```
<title>Hoja de Rutas</title>
<script src="../mscross.js" type="text/javascript"></script>
<script type="text/javascript">
        myMap1 = new msMap(
document.getElementById('map_tag'),'standardUp');
```

```
        myMap1.setCgi( '/cgi-bin/mapserv.exe' );
        myMap1.setFullExtent( 713606.8461, 733223.2157, 9675184.5942,
9686001.686);
        myMap1.setMapFile(
'/ms4w/Apache/htdocs/rutapie/cuencaesf.map' );
        myMap1.setLayers( 'manzanas','calles','vias_cuenca','path' );
        myMap1.redraw();
```

Luego se procede a colocar todas las capas que fueron exportadas al archivo map. Para ello utilizamos **casillas de verificación**, las mismas que muestran los nombres de las capas a visualizar y se llama a una función para que muestre en el mapa la casilla que seleccione.

Las capas que utilizamos son: Manzanas, calles, vías_cuenca, path. La porcion de codigo etiquetado dentro de un form y el resultado en la interfaz, se muestran a continuacion:

```
<!--CAPAS TEMATICAS-->
<form id="select_layers" name="select_layers">
          <input onClick="chgLayers()" type="checkbox" value="manzanas" name="layer[0]" checked />
          Manzanas<br>
          <input onClick="chgLayers()" type="checkbox" value="calles" name="layer[1]" />
          calles<br>
          <input onClick="chgLayers()" type="checkbox" value="vias_cuenca" name="layer[2]" checked/>
          vias_cuenca<br>
          <input onClick="chgLayers()" type="checkbox" value="path" name="layer[3]" />
          path<br>

</form>
```

Figura 11: Código y visualización de las capas temáticas
Fuente: (Autores)

La función ChgLayers() incluida en el mscross.js realiza la visualización de las capas que desee según como vaya presionando las casillas de verificación.

```
function chgLayers()
{
  var list = "SARDINIA ";
  var objForm = document.forms[0];
  for(i=0; i<document.forms[0].length; i++)
  {
    if( objForm.elements["layer[" + i + "]"].checked )
    {
         list = list + objForm.elements["layer[" + i + "]"].value + " ";
    }
  }
  myMap1.setLayers( list );
  myMap1.redraw();
}
```

Figura 12: Función chgLayers()

Fuente: (Mariño y Moncayo, 2008)

Con el código introducido y marcadas las casillas de verificación Manzanas y vías_cuenca para obtener un mapa de referencia, se obtendrá el siguiente resultado:

Figura 13: Mapa de referencia. Capas Manzanas y Vias_Cuenca
Fuente: (Autores)

3.6.1. Coordenadas 'X' y 'Y' de georreferenciación del mapa

Al posicionarse en algún lugar del mapa es importante obtener la georreferenciación del mismo mostrando las coordenadas geográficas de longitud y latitud en 'X' y 'Y'. A través de la librería mscross.js que permite implementar esto mediante la función denominada ***ShowCoordinates,*** se consigue sacar el mayor provecho de las ventajas que presenta MapServer al visualizar los mapas de forma georreferenciada.

```
this.ShowCoordinates=function(event)
{
        var myX,myY;
        if (browser.isNS)
        {
             xc = event.layerX;
             yc = event.layerY;
        }
        else
        {
            xc = window.event.offsetX;
            yc = window.event.offsetY;
        }
        //  Posición del ratón
```

```
        //  Visualiza en pantalla.
        myX = _ext_Xmin + xc * _pixel_w;
        myY = _ext_Ymax - yc * _pixel_h;
        loc //  valores 'x' y 'y' retornados
}
```

En la función original hemos cambiado la forma de retorno de la función, pues recibimos los valores 'x' y 'y' separándolos con los caracteres '@@' esto con el propósito de manejar los datos en las consultas que necesitamos posteriormente al implementar la librería PgRouting.

Para visualizar en pantalla los valores de las coordenadas creamos una etiqueta donde se crea el "map_tag" y se llama a la función enviando como parámetro el evento que se ejecuta cada vez que se mueve el mouse.

```
<div id="map_tag"
onMouseMove="myMap1.ShowCoordinates(event)"></div>
```

3.7. La Librería PgRouting

La aplicación que desarrollamos tiene como objetivo principal proporcionar la funcionalidad de *routing* (ruteo) al gestor de base de datos PostgreSQL con la extensión PostGIS.

Para ello utilizamos la librería pgRouting que es parte del conjunto de librerías denominadas *PostLBS*, que proporcionan herramientas básicas de Localización de Servicios (LBS) en software libre de código abierto (OSS). Sus herramientas son similares a las que podríamos encontrar en un software propietario de GIS.

Para el desarrollo del prototipo de nuestra tesis hemos tomado como guía el tutorial **Pgrouting usando el algoritmo A* (camino más corto con heurística) y MapServer** el mismo que tiene su base de conocimientos y aplicaciones publicados en la dirección de internet: http://pgrouting.postlbs.org/.

Lo que intentamos hacer es que saliendo de un nodo determinado, podamos visitar todos los nodos sin repetir ninguno y hacerlo por una ruta de mínimo costo, con o sin las consideraciones de sentido propia de las vías.

Para ello hay que tomar en cuenta las siguientes consideraciones importantes:

- Creación de la red cuyos elementos son nodos conectados por arcos.
    - Nodos = Clientes y Agencias
    - Arcos = calles o vías
- Consideraciones de sentidos de vías por arco determinado.

3.7.1. Entorno de instalación de pgRouting

Para poder hacer uso de la librería pgRouting, previamente tenemos instalado en el sistema las herramientas:

- Paquete de MapServer con Servidor Web Apache → MS4W, v2.2.3
- PostgreSQL 8.2.4 con la extensión PostGIS 1.1

La instalación y configuración de las mismas se realizo de forma sencilla siguiendo los pasos indicados anteriormente.

Las versiones de las herramientas aquí citadas son las requeridas para un correcto funcionamiento de esta librería y las que utilizamos en el desarrollo de nuestro prototipo.

3.7.2, Instalación de pgRouting

Para la instalación de pgRouting, en primer lugar, necesitamos descargarnos el archivo **pgRouting 1.0.0a-win32-installer** desde el sitio web oficial: http://pgrouting.postlbs.org

Abrimos el paquete de instalación. La instalación recorre casi automáticamente. Se recomienda instalar en un directorio personalizado, por ejemplo: **C:\Programme\PostgreSQL\8.2 **

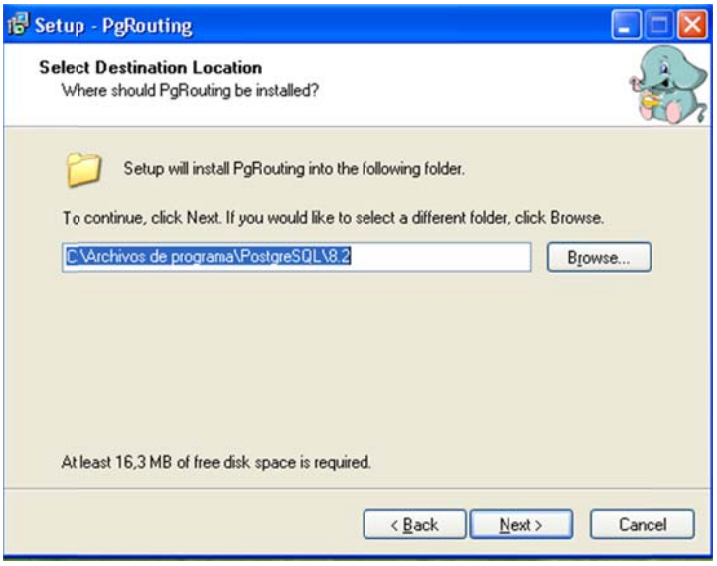


Figura 14. Instalación de pgRouting en Windows XP
Fuente (Autores)

3.8. Creación de la Base de Datos

La interfaz que implementamos en el proyecto necesita interactuar con una base de datos que permite almacenar y estructurar adecuadamente la información para las diferentes peticiones y consultas que se necesiten.

En nuestro caso tenemos la base de datos **cuenca,** que debe tener como *template* (plantilla) el: **template_postgis** que se obtiene al instalar la extensión PostGis.

Esto con el propósito de darle a nuestra base datos todas las características de georreferenciación y análisis espacial que necesitamos para llevar a cabo nuestro proyecto.

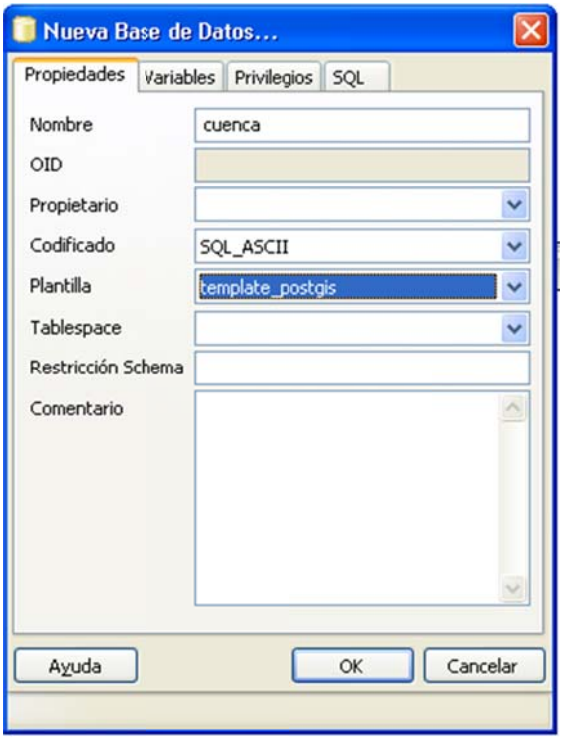


Figura 15. Creando Base de Datos con PgAdmin III
Fuente (Autores)

3.8.1. Importación de datos a PostgreSQL/PostGIS

Las capas recolectadas poseen información referente a algunas características de la ciudad que son básicas para nuestro proyecto. Es necesario importar cada una de las capas utilizadas en la generación del Mapa de Cuenca a la base de datos, convirtiendo cada uno de los archivos .shp a scripts con los cuales podamos crear las tablas que necesitamos en nuestra base de datos. (Ortega y Flores, 2009).

Por ejemplo, para el archivo **vías_cuenca.shp,** utilizamos la siguiente línea de comando en el **command prompt** de PostgreSQL 8.2.

```
shp2pgsql -s 24877 c: \vias_cuenca.shp vías_cuenca>
C:\vias_cuenca.sql
```

Tome en cuenta que para las capas en el sistema de coordenadas PSAD56 Zona 17S, el ESPG correspondiente es 24877, para el sistema de coordenadas WGS 1984 UTM Zone 17S el ESPG correspondiente es 32717

Figura 16: Creación de scripts para importar shapes a tablas en PostgreSQL
Fuente: (Autores)

3.8.2. Creación de las tablas de la base de datos

Ahora que tenemos estos archivos convertidos en scripts, debemos ejecutar cada uno de los script creados, para ello utilizamos el pgAdmin III, seleccionando previamente nuestra base de datos **cuenca.**

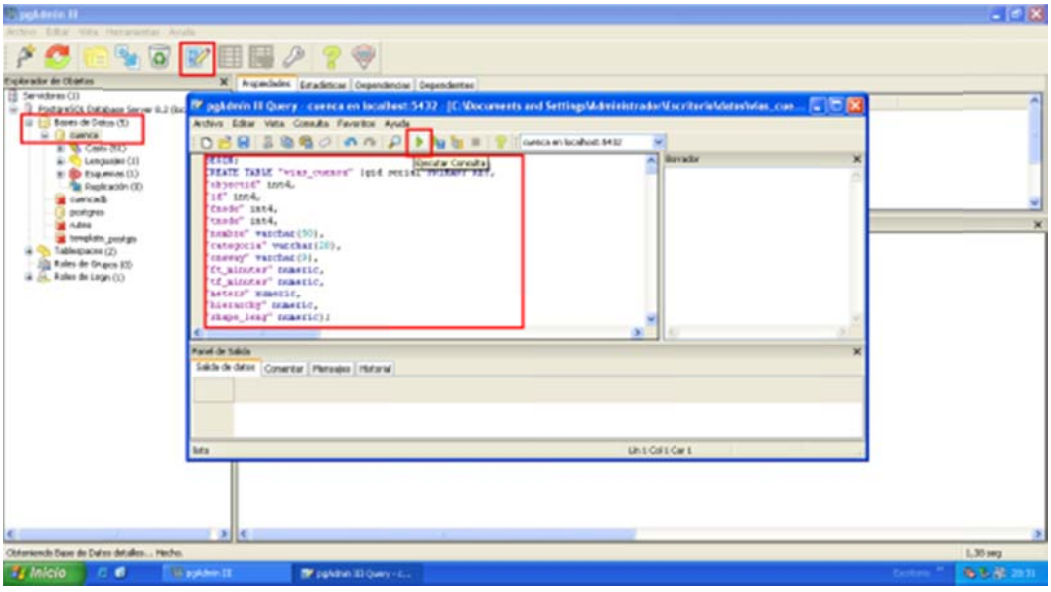
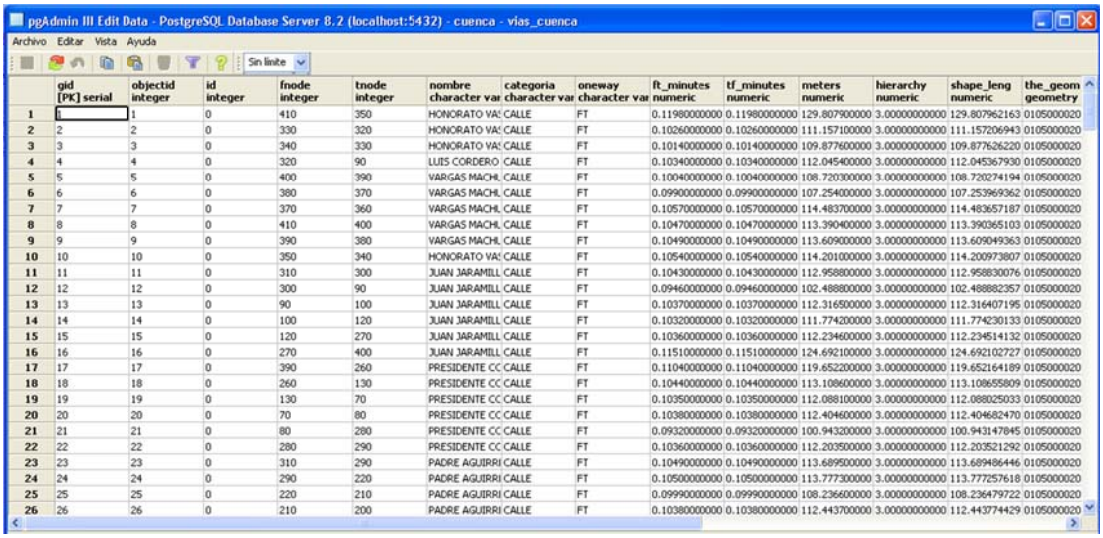


Figura 17: Creación de tabla “vías_cuenca” en PostgreSQL
Fuente: (Autores)

Para verificar que hemos creado correctamente las tablas podemos hacer una simple consulta de los datos contenidos en estas, por ejemplo, para la tabla vías_cuenca hacemos:

```
SELECT *
FROM vías_cuenca
```

Consultando los datos de la tabla vías_cuenca, se tienen los siguientes resultados:



	gid [PK] serial	objectid integer	id integer	fnode integer	tnode integer	nombre character var	categoria character var	oneway character var	ft_minutes numeric	tf_minutes numeric	meters numeric	hierarchy numeric	shape_leng numeric	the_geom geometry
1	1	1	0	410	350	HONORATO VA	CALLE	FT	0.11980000000	0.11980000000	129.807900000	3.00000000000	129.807962163	0105000020
2	2	2	0	330	320	HONORATO VA	CALLE	FT	0.10260000000	0.10260000000	111.157100000	3.00000000000	111.157206943	0105000020
3	3	3	0	340	330	HONORATO VA	CALLE	FT	0.10140000000	0.10140000000	109.877600000	3.00000000000	109.877626220	0105000020
4	4	4	0	320	90	LUIS CORDERO	CALLE	FT	0.10340000000	0.10340000000	112.045400000	3.00000000000	112.045367930	0105000020
5	5	5	0	400	390	VARGAS MACHI	CALLE	FT	0.10040000000	0.10040000000	108.720300000	3.00000000000	108.720274194	0105000020
6	6	6	0	380	370	VARGAS MACHI	CALLE	FT	0.09900000000	0.09900000000	107.254000000	3.00000000000	107.253969362	0105000020
7	7	7	0	370	360	VARGAS MACHI	CALLE	FT	0.10570000000	0.10570000000	114.483700000	3.00000000000	114.483657187	0105000020
8	8	8	0	410	400	VARGAS MACHI	CALLE	FT	0.10470000000	0.10470000000	113.390400000	3.00000000000	113.390365103	0105000020
9	9	9	0	390	380	VARGAS MACHI	CALLE	FT	0.10490000000	0.10490000000	113.609000000	3.00000000000	113.609049363	0105000020
10	10	10	0	350	340	HONORATO VA	CALLE	FT	0.10540000000	0.10540000000	114.201000000	3.00000000000	114.200973807	0105000020
11	11	11	0	310	300	JUAN JARAMILL	CALLE	FT	0.10430000000	0.10430000000	112.958800000	3.00000000000	112.958830076	0105000020
12	12	12	0	300	90	JUAN JARAMILL	CALLE	FT	0.09460000000	0.09460000000	102.488800000	3.00000000000	102.488882357	0105000020
13	13	13	0	90	100	JUAN JARAMILL	CALLE	FT	0.10370000000	0.10370000000	112.316500000	3.00000000000	112.316407195	0105000020
14	14	14	0	100	120	JUAN JARAMILL	CALLE	FT	0.10320000000	0.10320000000	111.774200000	3.00000000000	111.774230133	0105000020
15	15	15	0	120	270	JUAN JARAMILL	CALLE	FT	0.10360000000	0.10360000000	112.234600000	3.00000000000	112.234514132	0105000020
16	16	16	0	270	400	JUAN JARAMILL	CALLE	FT	0.11510000000	0.11510000000	124.692100000	3.00000000000	124.692102727	0105000020
17	17	17	0	390	260	PRESIDENTE CC	CALLE	FT	0.11040000000	0.11040000000	119.652200000	3.00000000000	119.652164189	0105000020
18	18	18	0	260	130	PRESIDENTE CC	CALLE	FT	0.10440000000	0.10440000000	113.108600000	3.00000000000	113.108655809	0105000020
19	19	19	0	130	70	PRESIDENTE CC	CALLE	FT	0.10350000000	0.10350000000	112.088100000	3.00000000000	112.088025033	0105000020
20	20	20	0	70	80	PRESIDENTE CC	CALLE	FT	0.10380000000	0.10380000000	112.404600000	3.00000000000	112.404682470	0105000020
21	21	21	0	80	280	PRESIDENTE CC	CALLE	FT	0.09320000000	0.09320000000	100.943200000	3.00000000000	100.943147845	0105000020
22	22	22	0	280	290	PRESIDENTE CC	CALLE	FT	0.10360000000	0.10360000000	112.203500000	3.00000000000	112.203521292	0105000020
23	23	23	0	310	290	PADRE AGUIRRI	CALLE	FT	0.10490000000	0.10490000000	113.689500000	3.00000000000	113.689486446	0105000020
24	24	24	0	290	220	PADRE AGUIRRI	CALLE	FT	0.10500000000	0.10500000000	113.777300000	3.00000000000	113.777257618	0105000020
25	25	25	0	220	210	PADRE AGUIRRI	CALLE	FT	0.09990000000	0.09990000000	108.236600000	3.00000000000	108.236479722	0105000020
26	26	26	0	210	200	PADRE AGUIRRI	CALLE	FT	0.10380000000	0.10380000000	112.443700000	3.00000000000	112.443774429	0105000020

Figura 18: Estructura y datos originales de la tabla “vías_cuenca”
Fuente (Autores)

3.9. Implementación de archivos para manejar consultas tipo AJAX en la base de datos

Para poder acceder a la información almacenada en la base de datos a través de consultas tipo AJAX para realizar las diferentes operaciones de mantenimiento de los datos de forma más rápidas y con mejor respuesta a las acciones del usuario, es necesario estructurarlos en un formato como XML que permita la lectura fácil de los mismos desde la interfaz desarrollada.

Para esto y gracias a la colaboración del Ing. Diego Pacheco hemos implementado a nuestro proyecto dos archivos de su autoría denominados xml.php y dbconsultas.js. (El código completo de dichos archivos se puede ver en el **Anexo 2**).

El archivo **xml.php** se trata de un código PHP que estructura los datos de registros y campos de los *querys* (consultas) en formato XML en filas y columnas. La parte del código que hace esto se muestra a continuación:

```
$xml .= '<dato>'; // creando la estructura
    $xml .= "<filas>$filas</filas>";
    $xml .= "<columnas>$cols</columnas>";
$xml .= '</dato>';
```

El archivo **dbconsultas,js** hace el llamado a función que sirve para leer un xml dinámicamente sin necesidad de saber el número de líneas o campos que posea, el resultado devuelto se almacena en una variable para obtener los datos necesarios. Aquí mostramos el código referente a esta función

```
function writeList()
{
  var labels = xmlDocm.getElementsByTagName('dato');
  var ol = document.createElement('OL');
  filas=labels[0].childNodes[0].firstChild.nodeValue;
  cols=labels[0].childNodes[1].firstChild.nodeValue;
  datam='';
  for (i=1; i <= filas; i++) //barro todas las líneas y columnas del xml
   {
        for (j=0; j < cols; j++)
        {
            datam = datam + labels[i].childNodes[j].firstChild.nodeValue +
'@'; //@ sera usado como separadador de campos
        }
       datam = datam + '@*@'; //sera utilizado como separador de registros
     var li = document.createElement('LI');
     var labelId = document.createTextNode('(' + labels[i].getAttribute('id') +
')');
     li.appendChild(labelId);
     ol.appendChild(li);
   }
   retornoie=datam;
}
```

Este archivo es incluido en los PHP-Mapscripts como una librería, a través de la siguiente línea de código:

```
<script src="dbconsultas.js" type="text/javascript"></script>
```

Una vez que se tiene almacenada en una variable la estructura XML de los datos (mediante la llamada al archivo xml.php), se tiene que invocar a la función leerxml() enviándole como parámetro la variable. A continuación se muestra un fragmento de código de una función donde podemos ver todo lo descrito:

```
function dibujar_ruta()
{
var datos1,tbl1,cod;
cod="";
sql1="xml.php?sql=SELECT@*@FROM@shortest_path(\'select@gid@as@id,source::integer,target::integer,length::double@precision@as@cost@from@vias\'," +
inicio + "," + fin + ",false,false)";
datos1=leerxml(sql1);
tmp= datos1.split("@*@");
        for (i=0;i<=tmp.length-2;i++)
        {
                tmp1= tmp[i].split("@");
                cod=cod + trim(tmp1[1]) + ",";
        }
cod=cod.substring(1,cod.length-1);
```

3.10. Generar Scripts de ruteo en PostgreSQL

Hasta el momento hemos creado las tablas iniciales con los datos que necesitamos dentro de nuestra base de datos, así como también los diferentes archivos y configuraciones que requerimos para interactuar con dichos datos. Sin embargo, la base de datos aún no es capaz de calcular rutas. Para conseguirlo necesitamos ejecutar el contenido de los scripts:

- routing.sql
- routing_postgis.sql

Los mismos que se encuentran en la ruta donde instalamos la librería pgRouting en: **C:\Programme\PostgreSQL\8.2\share\contrib**

Para ello utilizamos el pgAdmin III, seleccionamos nuestra base de datos y hacemos clic en el botón ***SQL,*** copiamos el contenido de los scripts y los ejecutamos, con ello se generan las funciones necesarias para la generación de las rutas.

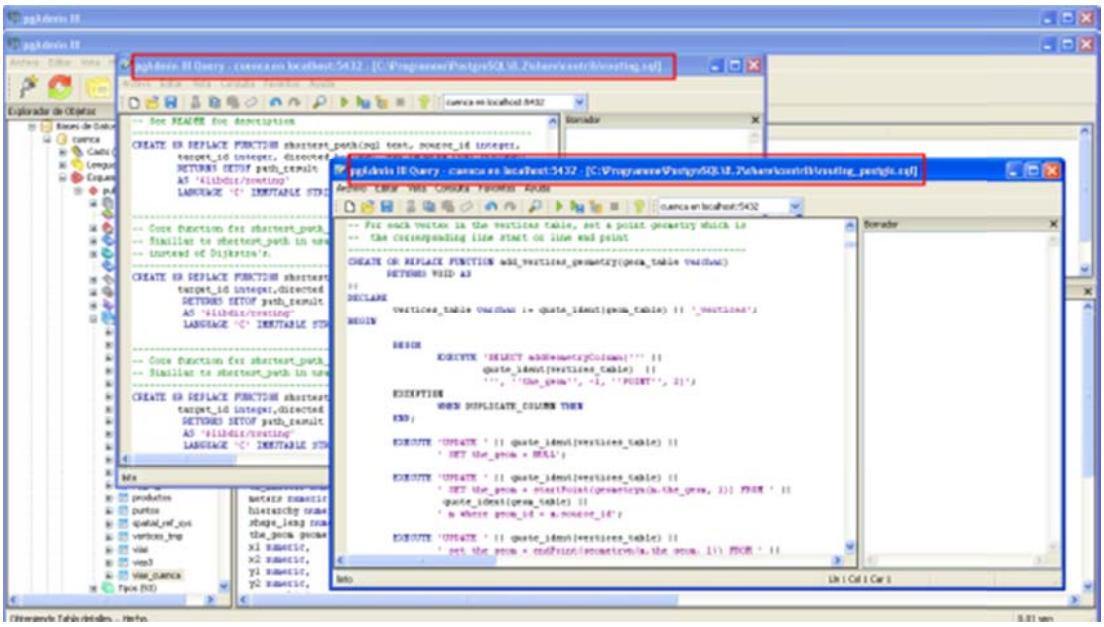


Figura 19: Scripts ejecutados en PostgreSQL
Fuente: (Autores)

3.11. Estructura de la Tabla para las diferentes funciones de pgRouting

Ahora nuestra base de datos está preparada para calcular las rutas. Sin embargo, eso no significa que funcione aún. A fin de lograr que las diferentes funciones de pgRouting se ejecuten se necesita de una tabla con una estructura especial.

Además de los campos gid y la geometría (the_geom) también necesitamos los valores que proyectan las coordenadas "x", "y" y crear las columnas para almacenar estos valores, para ello utilizando el pgAdmin III editamos nuestra tabla **vías_cuenca** creando las siguientes columnas:

- "X1", Tipo de dato: Numérico
- "X2", Tipo de dato: Numérico
- "Y1", Tipo de dato: Numérico
- "Y2", Tipo de dato: Numérico

También necesitamos las columnas de longitud, fuente y objetivo para realizar las tareas propias de pgRouting:

- "*length*", Tipo de dato: Numérico

- "*source*", Tipo entero grande, *bigint*
- "*target*", Tipo entero grande, *bigint*

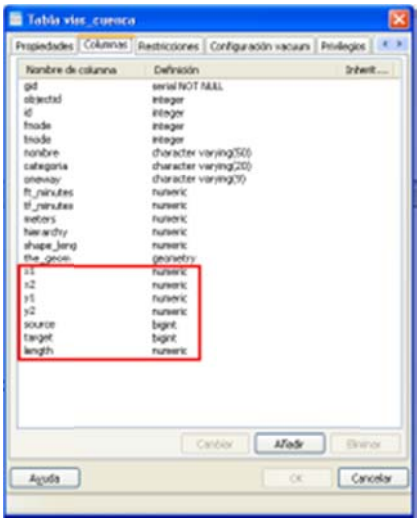


Figura. 20: Columnas adicionadas a la tabla vias_cuenca
Fuente: (Autores)

3.11.1. Creación de coordenadas de puntos inicio y fin (x,y)

Después de haber creado esas columnas, tenemos que importar los valores de las coordenadas de x1, y1, x2, y2. Para ello escribimos un php-script al que lo llamamos **"x_y_create.php",** el código de este archivo es el siguiente:

```
<?php

$host = "localhost";
$port = "5432";
$dbname = "cuenca";
$user = "postgres";
$password = "postgres";

$con_string = "host=$host port=$port dbname=$dbname user=$user
password=$password";
$con = pg_connect ($con_string);

//proyeccion coordenas x1,y1

 $id_check = "SELECT max(gid)as oid from vias_cuenca";
 $res_id_check = pg_query($con,$id_check);
   $count = pg_result($res_id_check,"oid");
   echo "Anzahl der Eintraegege in der DB: ".$count;
   echo "<br>";
for ($x=1;$x<=$count;$x++)
```

```
{
 $start = "SELECT astext(StartPoint(the_geom))as startpoint from vias_cuenca
where gid='$x'";
 $res_start= pg_query($con,$start);
   $start_ergebnis = pg_result($res_start,"startpoint");
   echo "<b>Geometrie $x</b></br>";
  echo "Anfangspunkte (x1,y1): ".$start_ergebnis;
  // echo "<br>";
  $array_01=array("POINT(",")");
  $array_02=array("","");
    for($r=0;$r<sizeof($array_01);$r++)
   {
    $start_ergebnis=str_replace($array_01[$r],$array_02[$r],$start_ergebnis);
    }
$explode=explode(" ",$start_ergebnis);
$x1=$explode[0];
$y1=$explode[1];
  echo "<br>";


//proyeccion coordenas x2,y2

$end = "SELECT astext(EndPoint(the_geom))as endpoint from vias_cuenca
where gid='$x'";
 $res_end= pg_query($con,$end);
   $end_ergebnis = pg_result($res_end,"endpoint");
   echo "Endpunkte (x2,y2): ".$end_ergebnis;
   echo "<br>";
   echo "--------------";
   echo "<br>";
    $array_01=array("POINT(",")");
  $array_02=array("","");
    for($r=0;$r<sizeof($array_01);$r++)
   {
    $end_ergebnis=str_replace($array_01[$r],$array_02[$r],$end_ergebnis);
    }
$explode=explode(" ",$end_ergebnis);
$x2=$explode[0];
$y2=$explode[1];

  //Valores escritos en las columnas

   $werte_in_tabelle_schreiben="UPDATE vias_cuenca SET
x1='$x1',y1='$y1',x2='$x2',y2='$y2' where gid='$x'";
 $res = pg_query($werte_in_tabelle_schreiben);
  }
?>
```

El funcionamiento del script es sencillo. Se crea una conexión con la base de datos PostgreSQL/PostGIS. Entonces se lee inicialmente el número de

cada una de las geometrías contenidas en la tabla y se escribe los valores de los vértices en un bucle.

A continuación, para activar el script se necesita guardarlo e invocarlo desde la carpeta htdocs del MapServer con Servidor Web Apache digitando la url: http://localhost/cuenca/x_y_create.php.

La ejecución del .php podría tomar algún tiempo hasta que todas las coordenadas sean escritas como registros en la base de datos.

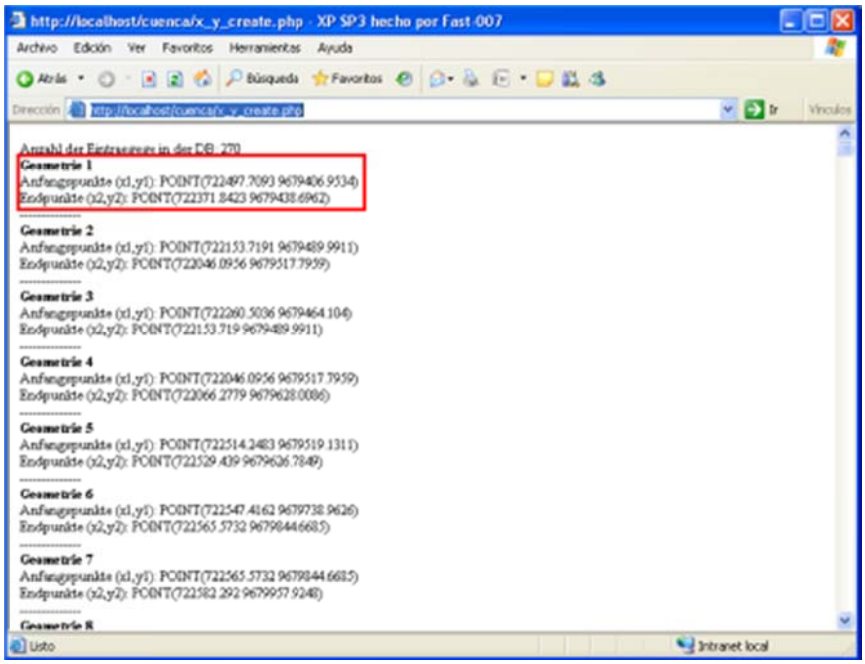


Figura. 21: Creación de las coordenadas del punto de inicio y fin (x,y)
Fuente: (Autores)

Para que todos los datos sean importados, y evitar que se pierdan algunos de ellos se debe realizar lo siguiente:

- Cambiar los valores en el archivo de configuración PHP, el **php.ini** que lo podemos encontrar en la siguiente dirección: C:\ms4w\Apache\cgi-bin. En caso de no encontrarse en esta dirección, buscarlo con el Windows.
- Abrir el archivo y cambiar el valor de máximo tiempo de ejecución que se encuentra en la línea de código 255, por algún valor mayor, por ejemplo cambiarlo por: max_execution_time=300.

3.11.2. Cálculo del campo "*length*"

Una vez que las geometrías están en la base de datos, tenemos que conseguir los valores para el campo *length* (longitud) de todas las calles. Esto se podría utilizar para hacer comparaciones de distancias de las calles. Para esto, utilizando pgAdmin III ejecutamos el siguiente comando SQL en nuestra base de datos:

UPDATE vias_cuenca set length=length(the_geom);

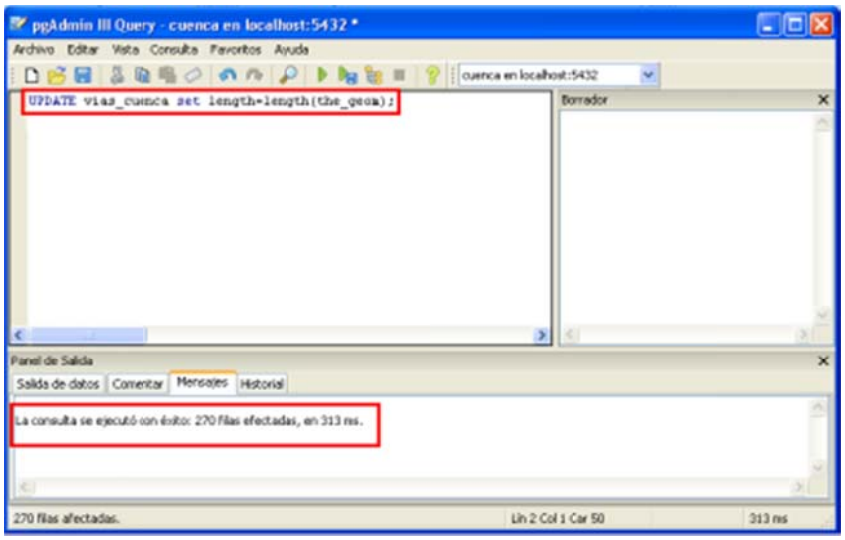


Figura. 22: Cálculo del campo "length" (longitud)
Fuente: (Autores)

Verificar que en el resultado de la consulta, se reciba el total de filas afectadas.

3.11.3. Cálculo del campo "*source" y" target"* a través de la función *assign_vertex_id*

Con la finalidad de poder crear los vértices de las intersecciones de las vías y calcular los valores para los campos "*source*" y "*target*" vamos a utilizar la función predefinida en la librería pgRputing llamada **assign_vertex_id**, Para ello ejecutamos el siguiente comando SQL en nuestra base de datos:

SELECT assign_vertex_id('vias_cuenca', 5);

Al ejecutar la consulta, debería darnos un error porque la función espera que los nombres de las columnas sean "*source_id*" y "*target_id*" en lugar de *source*" y "*target*". Por supuesto que podría modificarse la función pero es más fácil sólo renombrar las columnas e invocar a la función después, esto

Una vez renombradas las columnas ejecutamos nuevamente la consulta. El proceso toma un tiempo hasta que todas las entradas sean verificadas

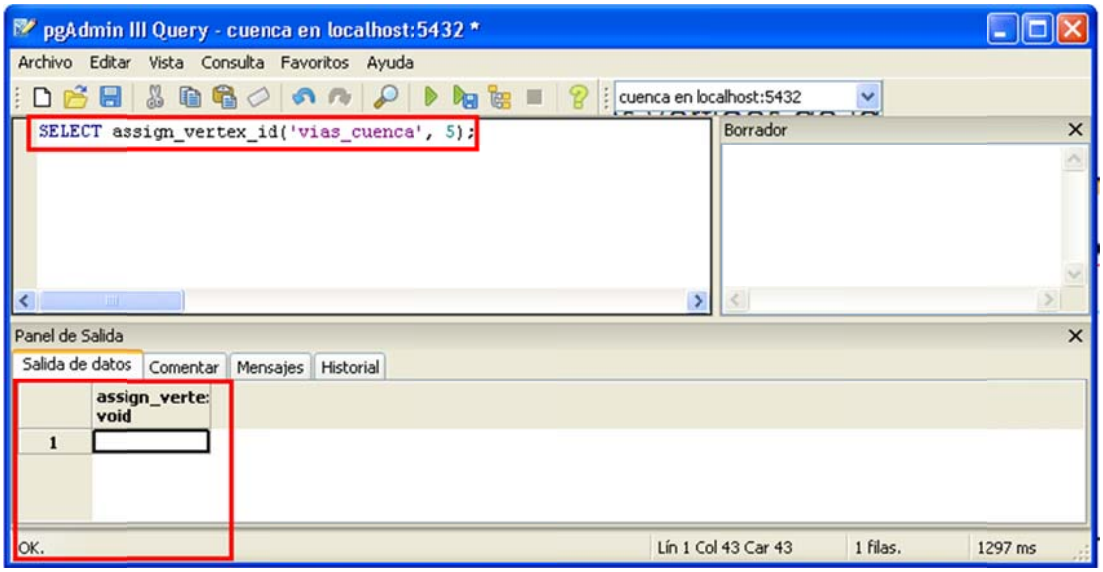


Figura. 23: Cálculo de los campos “source” y “target”
Fuente: (Autores)

Verificar que en la Salida de datos de la consulta tengamos un “OK” por respuesta, caso contrario sigua las indicaciones anotadas mas adelante bajo el tema **Problemas encontrados con la proyección en las coordenadas**. Finalmente volver a renombrar los campos *source_id* por "source" y "*target_id*" por "target" para dejar las columnas como se encontraban inicialmente.

3.11.3.1. Proyección de las coordenadas en metros

El número que se envía de parámetro en esta función puede ser variable. Un numero entero, como en nuestro caso el valor de 5, representa la proyección de las coordenadas en metros que permite asignar un mismo *vertexid (*vértice de intersección*)* a cada nodo.

Para una proyección más cercana la vértice de intersección bastara modificar el valor asignado por uno mucho menor que represente su equivalente en centímetros como puede ser el valor de 0,05.

3.11.3.2. Problemas encontrados con la proyección en las coordenadas

Al momento de ejecutar el SELECT, aun después de renombrar los campos source" y "target", tuvimos algunos inconvenientes pues se generaba un error con la proyección de las coordenadas de nuestra tabla vías_cuenca, con el campo llave principal **gid,** algo similar a lo siguiente:

ERROR: relation with OID 43540 does not exist Estado SQL:42P01 Contexto:SQL statement "SELECT id, the_geom FROM vertices_tmp WH"

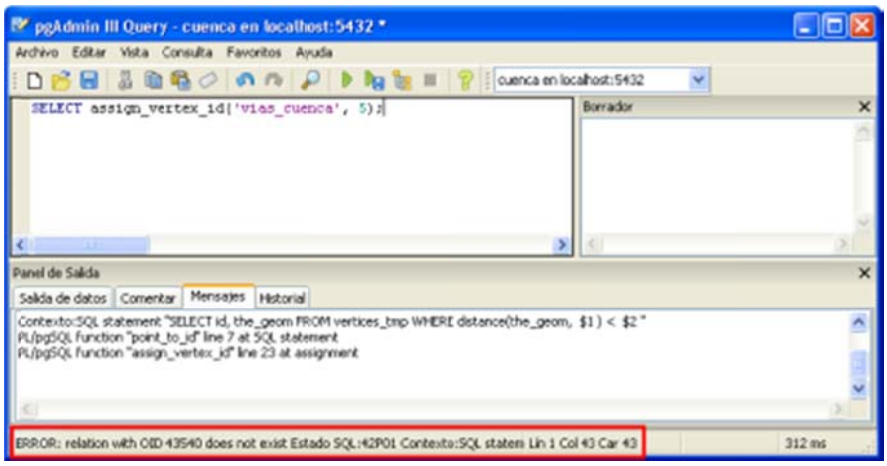


Figura. 24: Error al ejecutar función assign_vertex_id
Fuente: (Autores)

Además, en las columnas correspondientes no se ingresaban los datos de los respectivos códigos que necesitamos para ejecutar la función de ruteo.

Para solucionarlo, gracias a la colaboración del Ing. Diego Pacheco, quien nos asesoro sobre el tema, nos indico que teníamos que hacer unos cambios en el código de la función.

Para ello ingresando en el pgAdmin III y en la lista de funciones de nuestra base de datos modificamos la función assign_vertex_id y en la parte del código que hace referencia al addGeometryColumn cambiamos el valor

por 32717 que corresponde al sistema de coordenadas WGS84, los pasos se muestran en la siguiente figura:

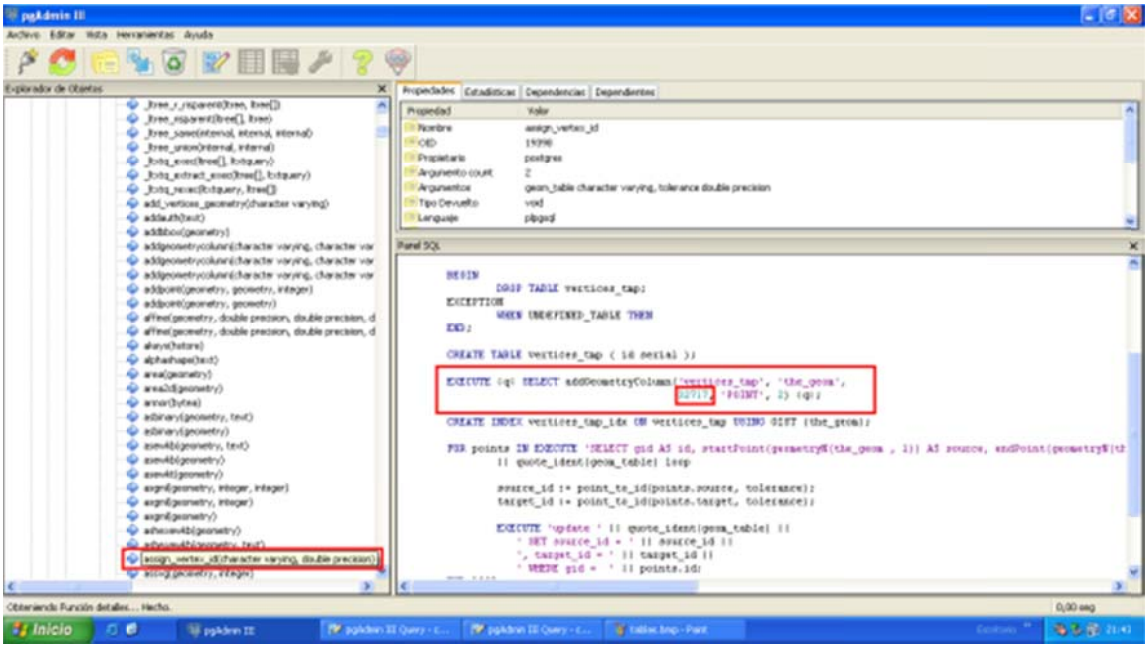


Figura. 25: Editando coordenadas en función assign_vertex_id
Fuente: (Autores)

Cabe indicar que posterior a estos cambios fue necesario además desconectar la instancia de conexión de la base de datos, luego volver a conectarla y ejecutar nuevamente el comando SELECT para que el proceso se ejecute de la forma esperada. Verificando esta vez que en la Salida de datos de la consulta tengamos un "OK" por respuesta

3.12. Ruta entre dos puntos creada con pgRouting

Llevado a cabo todos los pasos descritos anteriormente y solventando de la manera sugerida algunos inconvenientes que se pudieran presentar al proyectar las coordenadas especialmente con la función assign_vertex_id, hemos conseguido "habilitar" la funcionalidad de *routing* (ruteo) a nuestro gestor de base de datos PostGres/PostGis.

Para probar la correcta funcionalidad de la librería pgRouting, fue necesario, en primer lugar poder generar una ruta entre dos puntos

cualesquiera, ya que a partir de esto se hizo posible la generación de una ruta entre varios puntos con las consideraciones necesarias para generar la ruta óptima en un tipo de recorrido a pie o vehículo.

Para conseguir esto, lo que hicimos fue crear un archivo PHP/Mapscript denominado **index.php** adaptado a nuestro *geodata*, el mismo que utiliza el archivo **cuencaesf.map** creado anteriormente para poder visualizar las rutas y diferentes elementos de los mapas correctamente.

El archivo **index.php** es muy importante ya que nos sirvió como base para nuestros desarrollos ya que después de agregar e implementar nuevas funciones en el mismo pudimos llevar a cabo la creación de todos los otros archivos que requería nuestra aplicación. El código fuente del mismo lo podemos ver en el **Anexo 4**.

3.12.1. Interacción del usuario ubicando puntos en el mapa

Lo que necesitamos es proporcionarle al usuario la posibilidad de interactuar con el mapa ubicando los puntos en las calles que desee.

Básicamente, para implementar esta parte del proyecto lo que hacemos es que cuando el usuario hace clic en el botón para ubicar puntos  llamamos a una función que obtiene las coordenadas x e y del punto mediante la función **ShowCoordinates1** incluida dentro del mscross.js.

Con esto se hace la consulta a la base de datos para devolver los datos de ese punto como nombre, id, origen, destino y otros necesarios para construir la ruta. Para mostrar visualmente los puntos ubicados en el mapa usamos una función propia de mscross denominada **addPointOverlay** que recibe como parámetros los datos anteriormente obtenidos en la consulta y coloca una imagen que podemos cambiarle indicando el path adecuado.

La parte de código que corresponde a lo indicado y la ubicación de puntos en la interfaz, se muestran a continuación:

```
function vercor()
{
var cords;
var datos;
var sql;
cords=myMap1.ShowCoordinates1(event);
lonlat=cords.split("@@");

// Consulta devuelta en estructura xml del archivo xml.php
sql="xml.php?sql=select@name,source,target,gid@from@vias@where@distan
ce(vias.the_geom,setsrid(makepoint(" + trim(lonlat[0]) + "," + trim(lonlat[1]) +
"),23030))<0.0001@order@by@distance(vias.the_geom,setsrid(makepoint(" +
trim(lonlat[0])+ "," + trim(lonlat[1]) + "),23030))" + " limit 1";

//Llamada a función leerxml() del archivo dbconsultas.js
datos=leerxml(sql);
tmp= new Array();
tmp=datos.split("@@");
tmp1=tmp[0].split("@");
//saco el source y target para construir la ruta
if (puntos==0)
{
inicio=trim(tmp1[1]);
ginicio=trim(tmp1[3]);
}
if (puntos==1)
{
fin=trim(tmp1[2]);
gfin=trim(tmp1[3]);
}
//*********************
addPoint(event,myMap1,lonlat[0],lonlat[1],tmp1[0],inicio,fin);
}
function addPoint(event, map, x, y, calle,source,target)
{
        if (puntos==0)
        {
          Point = new pointOverlay( new mslcon('img/inicio.png', '', 13, 39),
null, 'Información', x, y,
                    new Array('Lat:', 'Lon:', 'Calle','Source','Target'), new Array(x,
y,calle,source,target) );
          map.addPointOverlay(Point);
        }
        if (puntos==1)
        {
          Point = new pointOverlay( new mslcon('img/fin.png', '', 13, 39), null,
'Información', x, y,
                    new Array('Lat:', 'Lon:', 'Calle','Source','Target'), new Array(x,
y,calle,source,target) );
          map.addPointOverlay(Point);
        }
```

```
puntos+=1;
if (puntos==2)
        {
        dibujar_ruta();
        }
}
```

Figura 26: Puntos ubicados en el mapa por el usuario
Fuente: (Autores)

3.12.2. Ruta dibujada entre dos puntos

Una vez que se termina de elegir dos puntos de la aplicación, se asocia al botón dibujar ruta el llamado de la función **dibujar_ruta()** que a su vez invoca a la función **shortest_path** dentro de una consulta la misma que identifica la ruta apropiada y la dibuja como capa denominada "*path*" en el mapfile, mostrando la ruta. El código de esta parte y la visualización en la aplicación se muestran a continuación:

```
function dibujar_ruta()
{
var datos1,tbl1,cod;
cod="";
sql1="xml.php?sql=SELECT@*@FROM@shortest_path(\'select@gid@as@id,sour
ce::integer,target::integer,length::double@precision@as@cost@from@vias\'," 
+ inicio + "," + fin + ",false,false)";
datos1=leerxml(sql1);
tmp= datos1.split("@*@");
        for (i=0;i<=tmp.length-2;i++)
        {
                tmp1= tmp[i].split("@");
```

```
                cod=cod + trim(tmp1[1]) + ",";
        }
cod=cod.substring(1,cod.length-1);

//llamo a la función que dibuja la capa de la ruta para visualizarla
dibujar_capa(cod);
}
function dibujar_capa(codig)
{
        var list = "SARDINIA ";
        var objForm = document.forms[0];

        for(i=0; i<document.forms[0].length-1; i++)
        {
         if( objForm.elements["layer[" + i + "]"].checked )
                {
              list = list + objForm.elements["layer[" + i + "]"].value + " ";
              }
        }
         codig = codig + ',' + ginicio + ',' + gfin;
         var fecha1= "codigos=" + codig;
         myMap1.setArgs(fecha1);
         myMap1.setLayers( list + 'path');
      myMap1.redraw();
}
```

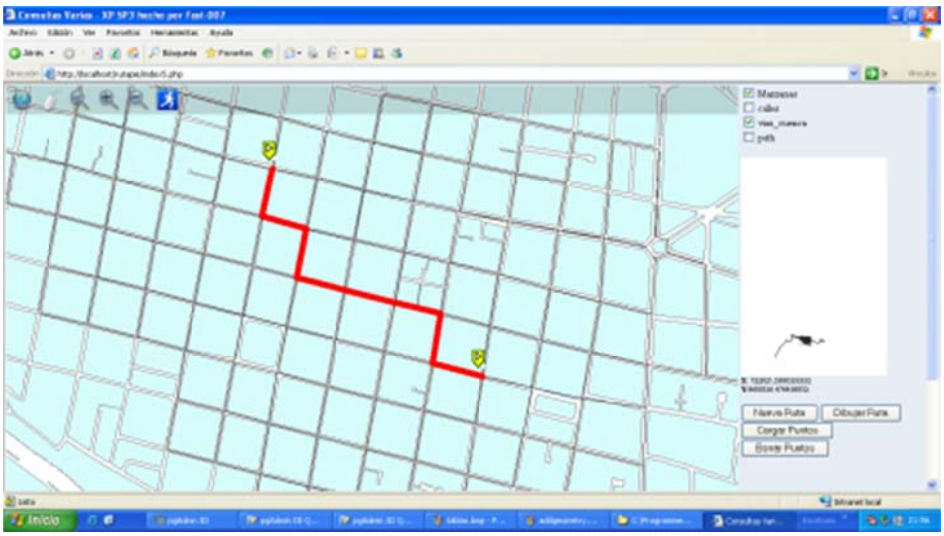


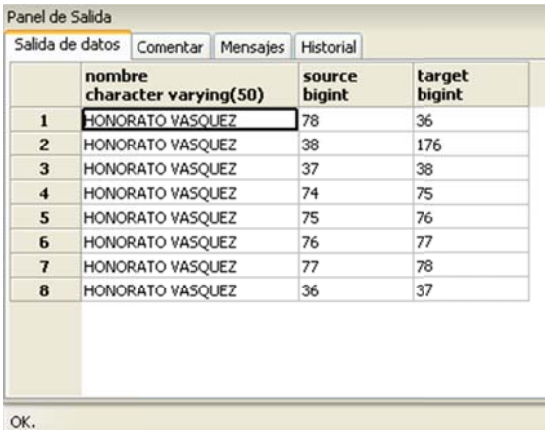
Figura 27: Ruta entre dos puntos creada con pgRouting
Fuente: (Autores)

3.12.3. Referencia a La funcion shortest_path

La función shortest_path, posee el algoritmo para identificar el enrutamiento mas corto y la misma fue definida en el archivo "routing_postgis.sql" que

hemos escrito anteriormente en la base de datos. Fundamentalmente es la definición del inicio y fin de los puntos. Por ejemplo, ejecutando la siguiente consulta, obtenemos los códigos de dichos puntos:

```
SELECT nombre, source, target
FROM vias_cuenca WHERE nombre like '%HONORATO VASQUEZ%'
```



	nombre character varying(50)	source bigint	target bigint
1	HONORATO VASQUEZ	78	36
2	HONORATO VASQUEZ	38	176
3	HONORATO VASQUEZ	37	38
4	HONORATO VASQUEZ	74	75
5	HONORATO VASQUEZ	75	76
6	HONORATO VASQUEZ	76	77
7	HONORATO VASQUEZ	77	78
8	HONORATO VASQUEZ	36	37

Figura 28: Valores de “source” y “target” de la calle HONORATO VASQUEZ
Fuente: (Autores)

El código de la función **shortest_path** se muestra a continuación:

```
CREATE OR REPLACE FUNCTION
shortest_path_astar2_as_geometry_internal_id(
      varchar,int4, int4, float8, float8, float8, float8)
      RETURNS SETOF GEOMS AS
$$
DECLARE
       geom_table ALIAS FOR $1;
          sourceid ALIAS FOR $2;
          targetid ALIAS FOR $3;
          ll_x ALIAS FOR $4;
          ll_y ALIAS FOR $5;
          ur_x ALIAS FOR $6;
          ur_y ALIAS FOR $7;
          rec record;
          r record;
           path_result record;
           v_id integer;
          e_id integer;
          geom geoms;
           srid integer;
BEGIN
          FOR path_result IN EXECUTE 'SELECT gid,the_geom FROM ' | |
          'shortest_path_astar2_as_geometry_internal_id_directed(''' | |
```

```
        quote_ident(geom_table) || '', ' || quote_literal(sourceid) || ', ' ||
          quote_literal(targetid) || ', ' || ll_x || ', ' || ll_y || ', ' ||
          ur_x || ', ' || ur_y || ', false, false)'
    LOOP
        geom.gid     := path_result.gid;
        geom.the_geom := path_result.the_geom;
        RETURN NEXT geom;
    END LOOP;
    RETURN;
END;
```

3.13. Ruta generada entre varios puntos

Hasta esta parte del proyecto se ha conseguido generar la ruta entre dos puntos, los mismos que se almacenaron solo en memoria y se pierden al salir de la aplicación, el siguiente paso es poder ubicar varios puntos en el mapa que tengamos registrados en la base de datos los que finalmente representaran clientes y agencias en la gestión de cobros y generar la ruta óptima u hoja de rutas entre los mismos.

3.13.1. Puntos ubicados en el mapa guardados como registros en la Base de Datos

Al ubicar varios puntos en el mapa es necesario almacenarlos en alguna tabla en nuestra base de datos para luego poder acceder a ellos. Para ello creamos una tabla que posea los campos necesarios para ejecutar la función shortest_path como son las coordenadas x,y, nombre, source, target, calle, gid, num que mencionamos anteriormente.

El script para generar la tabla es el siguiente:

```
-- Table: puntos
-- DROP TABLE puntos;
CREATE TABLE puntos
(
  x numeric,
  y numeric,
  source bigint,
  target bigint,
  calle character varying(200),
  gid real NOT NULL,
```

```
  num integer
)
WITHOUT OIDS;
ALTER TABLE puntos OWNER TO postgres;
```

3.13.2. Modificación de la función addPoint()

Para que esta función añada los puntos a la tabla en la base de datos se requiere realizar un INSERT de la siguiente manera:

```
sql="xml.php?sql=INSERT@INTO@puntos(x,y,source,target,calle,gid)@VALUES
@(" + x + "," +  y + "," + source + "," + target + "," + "'" + calle + "'" + "," + gid + ")";
datos=leerxml(sql);
```

3.13.3. Modificación de la función dibujar_ruta()

Con todos los puntos almacenados, es posible obtener la información de los mismos con una sentencia SELECT de la siguiente manera:

```
sql="xml.php?sql=SELECT@calle,gid,source,target@from@puntos";

datos1=leerxml(sql);
```

Con esta información se pueden cargar simultáneamente los puntos en el mapa. Pero para que se genere la ruta entre todos es necesario "recorrerlos" por así decirlo, para ello almacenamos en unos arreglos la información de los campos target y gid y los mandamos a dibujar dentro de un bucle con el código de la función chgLayer(), las partes de código aquí descritas y el resultado en la interfaz se muestran a continuación:

```
//definición de los arreglos a ser utilizados

tempgid = new Array();
temptarget = new Array();

//Select que obtiene la información de los puntos almacenados
```

```
sql="xml.php?sql=select@calle,gid,source,target@from@puntos";
        datos1=leerxml(sql);
        tmp= datos1.split("@*@");

//Bucle para almacenar en arreglos datos de la consulta
for (i=0;i<=tmp.length-2;i++)
        {
                tmp1= tmp[i].split("@");
                temptarget[i] = trim(tmp1[3]);
                tempgid[i] = trim(tmp1[1]);
        }

//Bucle para dibujar la ruta a través de todos los puntos utilizando la función chgLayer(),
for(cont=0;cont<puntos;cont++)
        {

        sql1="xml.php?sql=SELECT@*@FROM@shortest_path(\'select@gid@as@
id,source::integer,target::integer,length::double
        @precision@as@cost@from@vias_cuenca\'," + temptarget[aux] + "," +
temptarget[aux+=1] + ",false,false)";
                datos1=leerxml(sql1);
                tmp= datos1.split("@*@");
         if (band == 0)
         {
                for (i=0;i<=tmp.length-2;i++)
                {
                        tmp1= tmp[i].split("@");
                        cod=cod + trim(tmp1[1]) + ",";
                }
                cod= cod.split(",-1,");
                var list = "SARDINIA ";
                var objForm = document.forms[0];
                 for(i=0; i<document.forms[0].length-1; i++)
                 {
                   if( objForm.elements["layer[" + i + "]"].checked )
                           {
                        list = list + objForm.elements["layer[" + i + "]"].value + " ";
                        }
                 }
                 cod = cod + tempgid[aux = aux - 1] + ',' + tempgid[aux+=1];
                  fecha1= "codigos=" + cod;
                  myMap1.setArgs(fecha1);
                  myMap1.setLayers( list + 'path');
                band+=1;
         }
```

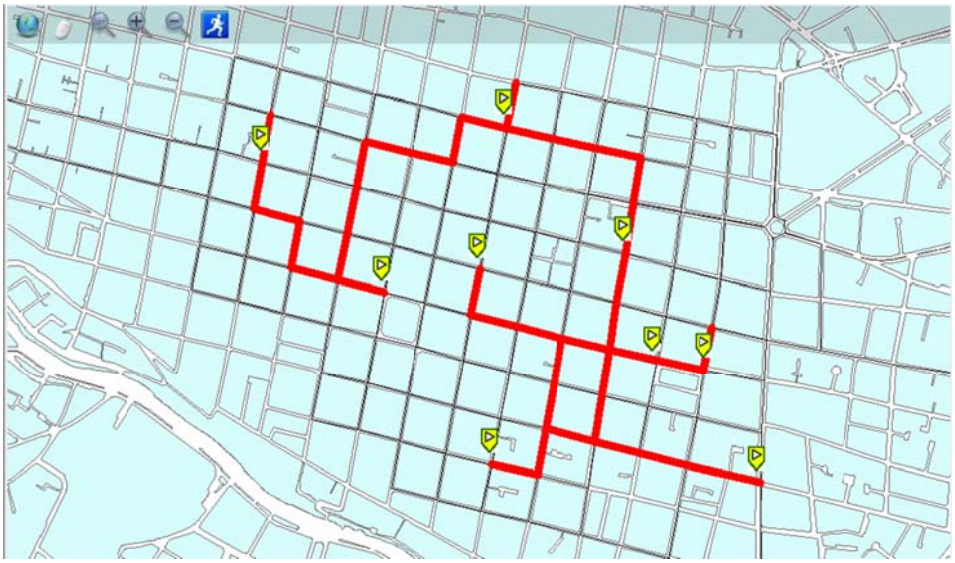


Figura 29: Ruta generada entre varios puntos
Fuente: (Autores)

3.14. Georreferenciación de los puntos como entidades clientes y agencias

En esta fase de la implementación del prototipo, los puntos que el usuario ubica en el mapa son almacenados en la tabla de puntos, contienen los datos necesarios que posee la capa de vías_cuenca y están georreferenciados.

Sin embargo, necesitamos que estos puntos representen a las entidades clientes y agencias, para la gestión de cobros y entrega de pedidos. Para ello necesitamos crear unas nuevas tablas para almacenar la información de clientes, agencias y crear los respectivos archivos PHP/MapScript que permiten implementar estas entidades en el prototipo.

3.14.1. Entidad Clientes

3.14.1.1. Creación y obtención de datos de la tabla: Clientes

El script para crear esta tabla y su estructura es el siguiente:

```
CREATE TABLE clientes
(
  ci character(15),
  x numeric,
  y numeric,
```

```
  source bigint,
  target bigint,
  calle character(200),
  gid real NOT NULL,
  estado "char" DEFAULT 'P'::"char",
  apellido_paterno character(40),
  apellido_materno character(40),
  nombres character(50),
  telefono_domicilio character(20)
)
WITHOUT OIDS;
ALTER TABLE clientes OWNER TO postgres;
```

Los datos descriptivos del cliente como: gid, ci, nombres, apellidos, teléfono_domicilio, se obtienen de un archivo de texto plano (CSV) de bases de datos que fue creado por nosotros y que contienen datos ficticios de registros de clientes, para utilizarlo con nuestro prototipo hemos seleccionado 100 registros únicamente.

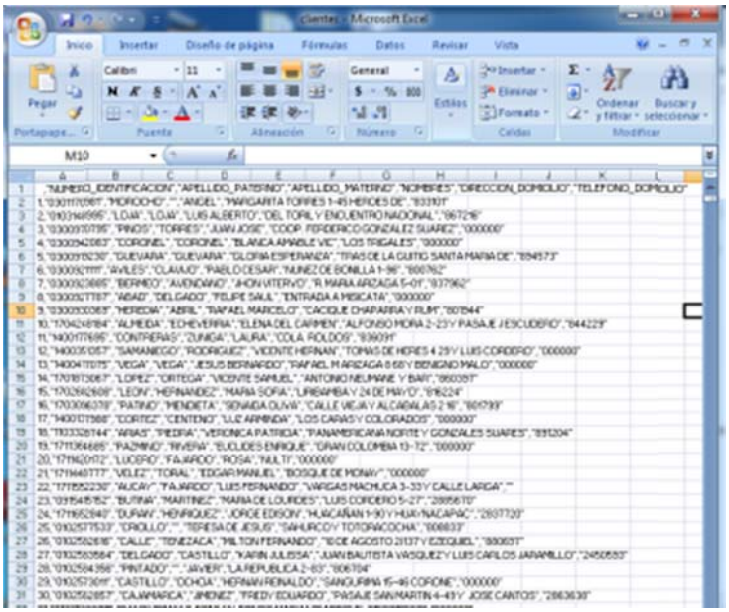


Figura 30: Archivo plano CSV de datos de Clientes
Fuente: (Autores)

Esto se hace con la finalidad de que la aplicación únicamente realice consultas a este archivo que ya se encuentra creado y almacenado, para obtener una copia de los datos dentro de nuestra base de datos, Para realizar esto debemos ingresamos al pgAdmin III, abrir nuestra base de datos cuenca y ejecutar el siguiente comando:

```
COPY
clientes(GID,CI,APELLIDO_PATERNO,APELLIDO_MATERNO,NOMBRES,TELEFON
O_DOMICILIO) FROM 'c:/clie.csv' DELIMITERS ';' WITH CSV
```

No olvidar colocar la ruta correcta donde se encuentre el archivo CSV para completar de forma correcta la importación de los datos.

Ahora con los registros ingresados en la tabla lo que tenemos que hacer es georreferenciar a estos clientes en el mapa a través de ingresar su dirección domiciliaria y otros campos necesarios.

3.14.1.2. El archivo Ingreso_cli.php

El archivo creado es muy similar al index.php que hemos utilizado, pero este realiza la actualización de los registros de los clientes en los campos utilizados para la Georreferenciación, con el siguiente código:

```
//Actualización de clientes
        sql="xml.php?sql=UPDATE@clientes@SET@gid="+gid+",@x=" + x + ",@y="+
y + ",@source=" + source + ",@target="+ target + ",@calle="+ "'" +
trim(direccion) + "'" + "@@WHERE@ci=" + cl;
        datos=leerxml(sql);
```

Además, se ha añadido algunas opciones nuevas a las funciones addPoint() pues hemos introducido nuevas consultas en la base de datos que permite que los puntos ubicados en el mapa muestren un cuadro informativo con Nombre del cliente, y la dirección exacta de las calles que forman la dirección del cliente, la parte del código se muestra continuación:

```
function addPoint(event, map, x, y, calle,source,target,gid,cl)
{
        var datos;
        var direccion;
   var apellido_paterno,apellido_materno,nombres,telefono_domicilio;
        var objForm = document.forms[1];
        myMap1.removeOverlayPoints();
        calle=calle.replace('Ñ','NI');
        sql="xml.php?sql=select@nombre@from@vias_cuenca@where@source
@="+source+"@and@gid@not@in@(select@gid@from@vias_cuenca@where@so
urce@="+source+"@and@target@=@"+target+")";
datos=leerxml(sql);
tmp= datos.split("@*@");
for (i=0;i<=tmp.length-2;i++)
```

```
        {
                tmp1= tmp[i].split("@");
                direccion=trim(tmp1[0]);
  }
sql="xml.php?sql=select@nombre@from@vias_cuenca@where@target@="+tar
get+"@and@gid@not@in@(select@gid@from@vias_cuenca@where@source@=
@"+source+"@and@target@=@"+target+")";
datos=leerxml(sql);
        tmp= datos.split("@*@");
        for (i=0;i<=tmp.length-2;i++)
        {
                tmp1= tmp[i].split("@");
                direccion=direccion + " y " + trim(tmp1[0]);
  }
        direccion=calle + " entre "+direccion;
        direccion=direccion.replace('Ñ','NI');
```

El resultado que se consigue en la interfaz es el siguiente:

Figura 31: Cuadro informativo de la entidad Clientes ubicado en el mapa
Fuente: (Autores)

3.14.2. Entidad Agencias

3.14.2.1. Creación de la tabla: Agencias

El script para crear esta tabla y su estructura es el siguiente:

```
CREATE TABLE agencias
(
  age_id numeric,
  x numeric,
  y numeric,
  source bigint,
```

```
  target bigint,
  calle character varying(200),
  gid real NOT NULL,
  nombre character(100)
)
WITHOUT OIDS;
ALTER TABLE agencias OWNER TO postgres;
```

Los datos para esta tabla serán ingresados cuando el usuario ubique los puntos sobre el mapa.

3.14.2.2. El archivo ingreso_age.php

Este archivo permite ingresar nuevas agencias que son ubicadas en el mapa, insertando los datos en la tabla de agencias. Con los datos ingresados es posible hacer una consulta para visualizar todas las agencias en el mapa. El código para insertar los datos es el siguiente:

```
//inserta datos en la tabla
sql="xml.php?sql=INSERT@INTO@agencias(age_id,x,y,source,target,calle,gid,
nombre)@VALUES@(" + age_id + ", " + x + "," +  y + "," + source + "," + target +
"," + "'" + trim(direccion) + "'" +  "," + gid + "," + "'" + trim(age) + "'" + ")";
datos=leerxml(sql);
```

Al igual que el archivo ingreso_cli.php se ha modificado la función addPoint() para que en el mapa se muestren un cuadro informativo sobre la el nombre y la dirección exacta de la agencia seleccionada. El resultado que se consigue en la interfaz es el siguiente:

Figura 32: Cuadro informativo de la entidad Agencias ubicado en el mapa
Fuente: (Autores)

3.14.3. El archivo listado.php

Este archivo realiza una consulta de todos los datos de las tablas clientes y agencias, para verificar si existen nuevos ingresos, el resultado de la consulta muestra una tabla con los datos obtenidos. Se crearon dos archivos de listado para cada tabla, el código es el mismo, solo cambia el SELECT utilizado.

Para listar los Clientes: $sql="select * from clientes where x is not null";
Para llistar las Agencias: $sql="select * from agencias";

3.15. Creación de la tabla: Distancias

Adicionalmente, requerimos crear una tabla de distancias que nos permita almacenar los valores calculados de las distancias entre los puntos ubicados en el mapa para poder compararlos entre si, determinar el menor recorrido y utilizar esto al momento de generar la ruta óptima.

El script para crear esta tabla y su estructura es el siguiente:

```
CREATE TABLE distancias
(
  distancia1 numeric,
  distancia2 numeric,
  source integer,
  target integer,
  gid integer
)
WITHOUT OIDS;
ALTER TABLE distancias OWNER TO postgres;
```

3.16. Función ruta_optima_clientes()

La ruta que se genera a través de todos los puntos no discrimina en cuanto a que puntos de menor distancia deben ser recorridos inicialmente hasta completar toda el enrutamiento.

Con el propósito de lograr que la ruta tenga un determinado orden pasando por los puntos de menor distancia partiendo desde una agencia hacia los clientes, hemos implementado la función denominada

ruta_optima_clientes(), la cual hace uso de la función shortest_path calculando las menores distancias y guardándolos dentro de la tabla de distancias que hemos creado.

La función luego de comparar las distancias, inserta en forma ordenada los valores de los campos de Georreferenciación: x, y, source, target, calle, gid, num en la tabla de puntos con la cual se dibuja la ruta siendo esta la optima para el recorrido. El código de la función es el siguiente:

```
CREATE OR REPLACE FUNCTION ruta_optima_clientes()
RETURNS SETOF geoms AS
$BODY$
DECLARE
        r record;
        path_result record;
        inicio integer;
        fin integer;
        distancia_1 numeric;
        distancia_2 numeric;
        num_clientes integer;
        cont integer;
        geom geoms;
        est char;
        gid1 real;
        gid2 real;
        gidmenor real;
        num_puntos integer
BEGIN
        est:='P';
        cont:=0;
        delete from puntos;
        FOR path_result IN EXECUTE 'SELECT  x, y, source, target, calle, gid
FROM agencias'
        LOOP
        cont:=cont+1;
        inicio     := path_result.source;
INSERT INTO puntos(x, y, source, target, calle, gid, num) VALUES
(path_result.x, path_result.y, path_result.source, path_result.target,
path_result.calle, path_result.gid, cont);
        END LOOP;
        ----Obtengo el total de clientes----
        select count(*) into num_clientes from clientes where  x is not null and
estado='P'; //
        WHILE num_clientes != 0 LOOP
        distancia_2:=999999;
        FOR i IN 1..num_clientes LOOP
        FOR path_result IN EXECUTE 'select  x, y, source, target, calle, gid from
clientes where x is not null and estado = ' | |  quote_literal(est)
```

```
        LOOP
        if i=1 then
        fin:=path_result.target;
        SELECT sum(cost) into distancia_1
        FROM shortest_path('select gid as
id,source::integer,target::integer,length::double precision as cost from
vias_cuenca',inicio,fin,false,false);
        gid1:=path_result.gid;
        end if;
        fin:=path_result.target;
        SELECT sum(cost) into distancia_2
        FROM shortest_path('select gid as
id,source::integer,target::integer,length::double precision as cost from
vias_cuenca',inicio,fin,false,false);
        gid2:=path_result.gid;
        if distancia_1 >= distancia_2 then
        if i = num_clientes then
        INSERT INTO distancias(distancia1, distancia2, source, target, gid)
        VALUES (distancia_1, distancia_2, inicio, fin, path_result.gid);
        cont:=cont+1;
        gidmenor:=gid2;
        FOR r IN EXECUTE 'select  x, y, source, target, calle, gid from clientes
where gid = ' || quote_literal(gidmenor)
        LOOP
        select count(*) into num_puntos from puntos where gid=gidmenor;
if num_puntos=0 then
        INSERT INTO puntos(x, y, source, target, calle, gid, num)
        VALUES (r.x, r.y, r.source, r.target, r.calle, r.gid, cont);
end if;
        inicio:=r.target;
        END LOOP;
        UPDATE clientes SET estado='I' WHERE gid=gidmenor;
        else
        INSERT INTO distancias(distancia1, distancia2, source, target, gid)
        VALUES (distancia_1, distancia_2, inicio, fin, path_result.gid);
        distancia_1:=distancia_2;
        gidmenor:=gid2;
        gid1:=gidmenor;
        INSERT INTO distancias(distancia1, distancia2, source, target, gid)
        VALUES (distancia_1, distancia_2, inicio, fin, path_result.gid);
end if;
else
if i = num_clientes then
        INSERT INTO distancias(distancia1, distancia2, source, target, gid)
        VALUES (distancia_1, distancia_2, inicio, fin, path_result.gid);
        cont:=cont+1;
        gidmenor:=gid1;
        FOR r IN EXECUTE 'select  x, y, source, target, calle, gid from clientes
where gid = ' || quote_literal(gidmenor)
        LOOP
        select count(*) into num_puntos from puntos where gid=gidmenor;
```

```
        if num_puntos=0 then
        INSERT INTO puntos(x, y, source, target, calle, gid, num)
VALUES (r.x, r.y, r.source, r.target, r.calle, r.gid, cont);
end if;
        inicio:=r.target;
        END LOOP;
        UPDATE clientes SET estado='I' WHERE gid=gidmenor;
else
        INSERT INTO distancias(distancia1, distancia2, source, target, gid)
        VALUES (distancia_1, distancia_2, inicio, fin, path_result.gid);
        gidmenor:=gid1;
        INSERT INTO distancias(distancia1, distancia2, source, target, gid)
        VALUES (distancia_1, distancia_2, inicio, fin, path_result.gid);
        end if;
end if;
geom.gid     := path_result.gid;
RETURN NEXT geom;
        END LOOP;
END LOOP;
--Para control
        select count(*) into num_clientes  from clientes where  x is not null  and
estado='P';
END LOOP;
        UPDATE clientes SET estado='P' WHERE x is not null and estado='I';
RETURN;
END;
$BODY$
LANGUAGE 'plpgsql' VOLATILE STRICT;
ALTER FUNCTION ruta_optima_clientes() OWNER TO postgres;
```

3.17. Generación de la hoja de rutas

Llegado a esta etapa del desarrollo e implementación del prototipo necesitamos generar la hoja de rutas que deberá tener en cuenta el cobrador para realizar su trabajo de entrega de paquetería y gestión de cobros. La hoja de rutas es la interface más importante del sistema pues en ella se resume todo la información de clientes y agencias, y permite generar la ruta de recorrido.

3.17. 1. El archivo hoja.php

Este archivo es el que interactúa con todas las tablas y funciones creadas. Permite cargar todos los clientes y agencias ubicados sobre el mapa consultando a través de su ID, nombre o descripción y generando la ruta

óptima, para ello hace el llamado a la función ruta_optima_clientes que permite dibujar la ruta, el código es el que se muestran a continuación:

```
function dibujar_ruta()
{
        var aux=0;
        var datos1,tbl1,cod,tmp1,tmp2, band;
        cod="";
        tmp1="";
        tmp2="";
        band=0;
// Llamado a la funcion ruta_optima_clientes()";

sql="xml.php?sql=select@gid@from@ruta_optima_clientes()";

        datos1=leerxml(sql);
// La función anterior se encarga de insertar en la tabla de puntos el orden a seguir para generarse la ruta optima

sql="xml.php?sql=select@calle,gid,source,target@from@puntos";
        datos1=leerxml(sql);
        tmp= datos1.split("@*@");n
```

El recorrido de los puntos para generar la ruta se realiza reutilizando el mismo código que se indico anteriormente en el punto 3.15.3. Modificación de la función dibujar_ruta() de esta tesis, en la parte referente al Bucle para dibujar la ruta a través de todos los puntos utilizando la función chgLayer(), que se puede consultar. El resultado sería el siguiente:

Figura 33: Hoja de rutas
Fuente: (Autores)

3.17.2. Implementación de la función que permita respetar el sentido de las vías

Como podemos ver en la Hoja de rutas, el recorrido generado hasta ahora es un recorrido a pie, pues este es indiferente al sentido geográfico Norte, sur, este u oeste que pueden tener las calles y avenidas.

Las redes o vías de transporte son caminos transitables que respetan el sentido de las vías y que pueden ser llevados a cabo a través de un vehículo, ya sea un automóvil, motocicleta, bicicleta, etc. Para ello fue necesario implementar la siguiente función:

```
CREATE OR REPLACE FUNCTION sentido_vias(sql text, source_id integer,
target_id integer, directed boolean, has_reverse_cost boolean)
  RETURNS SETOF path_result AS
$BODY$
DECLARE
        r record;
        resultado path_result; ---utilizado para cargar la ruta
        path_result record;
      cont integer;
      num_registros integer;
      registro_rutas   record;
        geom geoms;
      est   char;
        nodo_inicial integer;
        nodo_pivote integer;
      nodofinal integer;
      costo_minimo numeric;
      gid_minimo numeric;
BEGIN
        cont:=1;
        est := 'l';
        num_registros:=0;
      delete from rutas;
      select distinct tnode into nodofinal from vias_cuenca where
target=target_id;
        ------sentencia para ingresar las posibles rutas---------
        FOR path_result IN EXECUTE 'select gid,fnode,tnode,meters,oneway
from vias_cuenca where source =' || source_id
      LOOP
                        INSERT INTO rutas(gid, fnode, tnode, costo, estado,
orden, oneway,gid_pred)
                        VALUES (path_result.gid, path_result.fnode,
path_result.tnode, path_result.meters, 'l',0,path_result.oneway,0);
        END LOOP;
```

```
        FOR path_result IN EXECUTE 'SELECT gid, fnode, tnode, costo FROM
rutas where estado = ' | |
                                quote_literal(est) LOOP
                IF  nodofinal = path_result.tnode then
                        --update rutas set costo = nodofinal;
                        update rutas set estado = 'E' where tnode = nodofinal;
                        ---------obtengo el costo_minimo al punto destino--------
--
                        select min(costo) into costo_minimo from rutas where
estado = 'E';
                        select s.gid into gid_minimo from rutas s where s.costo =
costo_minimo;
                        ---------elimino las rutas con el estado encontrado que
tienen un costo mayor----------
                        update rutas set estado = 'B' where gid not in
(gid_minimo) and estado='E';
                        ------elimino las rutas con el estado ingresado que son
mayores o iguales a la ruta encontrada-----
                        update rutas set estado = 'B' where costo >=
costo_minimo  and gid not in (gid_minimo) and estado in ('I','R');
                ELSE
                        select count(*) into num_registros from rutas where
estado = 'I';
                        --update rutas set costo = num_registros;
                END IF;
        END LOOP;
        WHILE num_registros != 0 LOOP
                FOR path_result IN EXECUTE 'SELECT gid, fnode, tnode, costo
FROM rutas where estado = ' | |
                                        quote_literal(est) | | ' order by costo' LOOP
                        ----ingreso las nuevas rutas con el estado de ingresado
                        FOR r IN EXECUTE 'select gid,fnode,tnode,meters,oneway
from vias_cuenca where fnode =' | | path_result.tnode
                        LOOP
                                        INSERT INTO rutas(gid, fnode, tnode, costo,
estado, orden, oneway,gid_pred)
                                        VALUES (r.gid, r.fnode, r.tnode, r.meters +
path_result.costo, 'I', 0 ,r.oneway,path_result.gid);
                        END LOOP;
                        ---para comprobar si es en dos sentidos
                        FOR r IN EXECUTE 'select gid,fnode,tnode,meters,oneway
from vias_cuenca where tnode =' || path_result.fnode ||' and oneway= ' ||
'''BI'''
                        LOOP
                                        INSERT INTO rutas(gid, fnode, tnode, costo,
estado, orden, oneway,gid_pred)
                                        VALUES (r.gid, r.fnode, r.tnode, r.meters +
path_result.costo, 'I', 0 ,r.oneway, path_result.gid);

                        END LOOP;
```

```
                    ---actualizo la ruta que ya se recorrio para no revisar
dos veces
                    update rutas set estado = 'R' where gid = path_result.gid;
                END LOOP;
                FOR r IN EXECUTE 'SELECT gid, fnode, tnode, costo, oneway
FROM rutas where estado = ' | |
                        quote_literal(est) LOOP
                    IF  nodofinal = r.tnode then
                        --update rutas set costo = nodofinal;
                        update rutas set estado = 'E' where tnode =
nodofinal;
                        ---------obtengo el costo_minimo al punto destino--
--------
                        select min(costo) into costo_minimo from rutas
where estado = 'E';
                        select s.gid into gid_minimo from rutas s where
s.costo = costo_minimo;
                        ---------elimino las rutas con el estado encontrado
que tienen un costo mayor----------
                        update rutas set estado = 'B' where gid not in
(gid_minimo) and estado='E';
                        ---------elimino las rutas con el estado ingresado
que son mayores o iguales a la ruta encontrada-----
                        update rutas set estado = 'B' where costo >=
costo_minimo  and gid not in (gid_minimo) and estado in ('I','R');
                        select count(*) into num_registros from rutas where
estado='E';
                        if num_registros >= 1 then
                            num_registros:=0;
                        end if;
                    END IF;
                    IF  nodofinal = r.fnode then
                        --update rutas set costo = nodofinal;
                        update rutas set estado = 'E' where fnode =
nodofinal;
                        ---------obtengo el costo_minimo al punto destino--
--------
                        select min(costo) into costo_minimo from rutas
where estado = 'E';
                        select s.gid into gid_minimo from rutas s where
s.costo = costo_minimo;
                        ---------elimino las rutas con el estado encontrado
que tienen un costo mayor----------
                        update rutas set estado = 'B' where gid not in
(gid_minimo) and estado='E';
                        ---------elimino las rutas con el estado ingresado
que son mayores o iguales a la ruta encontrada-----
                        update rutas set estado = 'B' where costo >=
costo_minimo  and gid not in (gid_minimo) and estado in ('I','R');
```

```
                                select count(*) into num_registros from rutas where
estado='E';
                                if num_registros >= 1 then
                                        num_registros:=0;
                                end if;
                        END IF;
                END LOOP;
                --select count(*) into num_registros from rutas where estado = 'I';

                --select count(*) into num_registros from rutas where estado='E';
                --if num_registros >= 1 then
        --                num_registros:=0;
        --      end if;
        END LOOP;
      select distinct gid_pred into nodo_pivote from rutas where estado = 'E';
        delete from rutas where estado='B';
        num_registros:=1;
        WHILE num_registros != 0 LOOP
        ------eligo el proximo nodo a buscar y despues lo actualizo----------
                select distinct gid_pred into nodo_inicial from rutas where
estado in ('R') and gid = nodo_pivote;
                update rutas set estado = 'E'   where estado in ('R') and gid =
nodo_pivote;

                nodo_pivote:=nodo_inicial;
                select distinct gid_pred into num_registros from rutas where
estado in ('R') and gid = nodo_inicial;
                --if num_registros = 1 then
                --end if;
        END LOOP;
        ---------sentencia para retornar la funcion con datos, se carga en r que
es de tipo path_result---------
      FOR path_result IN EXECUTE 'select gid,source,meters from vias_cuenca
where gid in (select gid from rutas where estado in (''E'') )'
    --  FOR path_result IN EXECUTE 'select gid,source,meters from vias_cuenca
where gid in (select gid from rutas)'
      LOOP
                resultado.vertex_id:=path_result.source;
                resultado.edge_id:=path_result.gid;
                resultado.cost:=path_result.meters;
                RETURN NEXT resultado;
        END LOOP;
        RETURN;
END;
$BODY$
 LANGUAGE 'plpgsql' VOLATILE STRICT;
ALTER FUNCTION sentido_vias(sql text, source_id integer, target_id integer,
directed boolean, has_reverse_cost boolean) OWNER TO postgres;
```

3.18. Interfaces para el sistema Gestión de Cobro y Entrega de Pedido

Una vez llevado a cabo la implementación de los diferentes archivos PHP/MapScript con todas las consideraciones descritas anteriormente, tenemos como resultado la creación de algunas interfaces tanto para clientes, agencias, hojas de rutas y consultas.

Es necesario recopilar a todos estos archivos y crearnos una estructura de carpetas dentro de nuestro servidor Apache para contener estos archivos. La estructura creada es la siguiente:

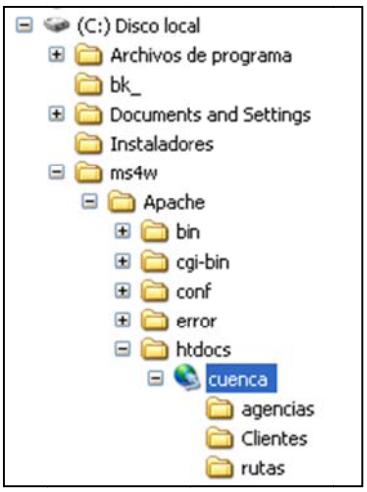


Figura 34: Estructura de carpeta creadas en el Servidor Apache
Fuente: (Autores)

Además, hemos adaptada las interfaces con nuevas pantallas e iconos que sean más representativos a los Clientes, Agencias y otras características que enriquecen el aspecto visual de la aplicación.

3.18.1. La Pantalla Principal de la aplicación

En vista de que tenemos varias interfaces creadas es necesario crearnos una página html principal para nuestra aplicación que sirva como un menú de opciones donde que permita una fácil navegación por las diferentes ventanas de la aplicación.

Para acceder a la página web coloque en su browser la siguiente dirección:

http://localhost/cuenca/index.html, tenga en cuanta que la dirección URL especifica la ubicación del archivo .html, note que este debe estar siempre bajo la carpeta de cuenca para un correcto funcionamiento.

Figura 35: Página Principal de la Aplicación
Fuente: (Autores)

3.18.2. La interfaz para Clientes

A través de la página principal de la aplicación, accediendo en la opción de Clientes se presenta un listado general clientes

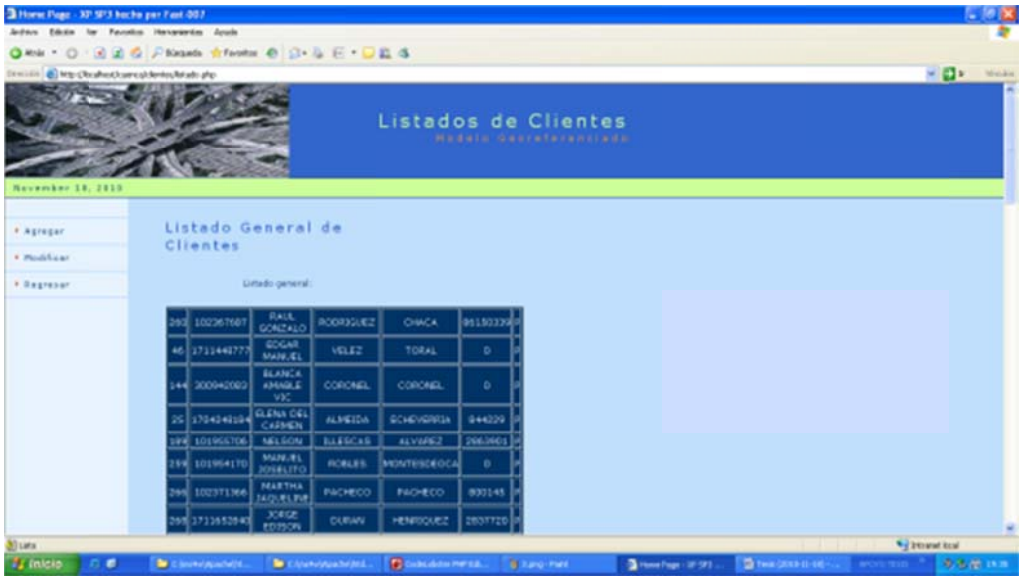


Figura 36: Listado General de Clientes
Fuente: (Autores)

En la opción Agregar / Modificar tenemos la interfaz para georreferenciar a los clientes

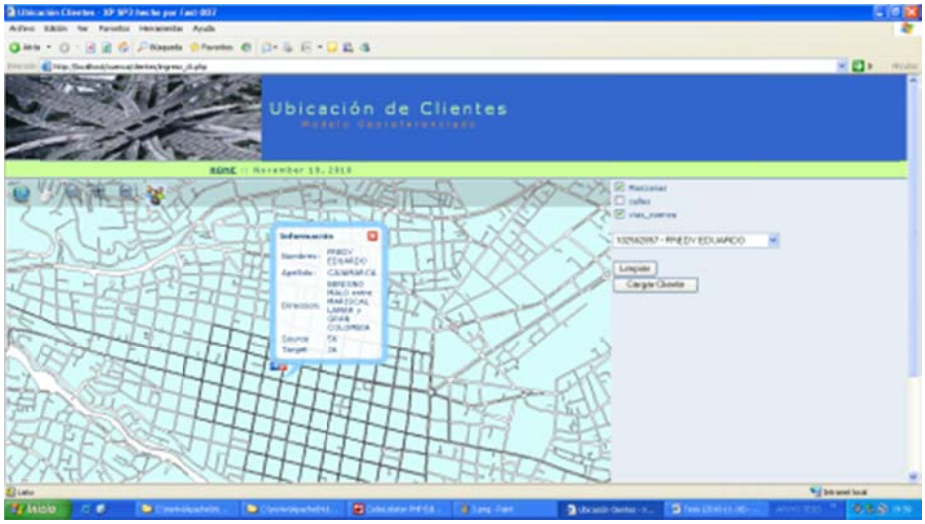


Figura 37: Interfaz para georreferenciar Clientes
Fuente: (Autores)

3.18.3. La interfaz para Agencias

A través de la página principal de la aplicación, accediendo en la opción de Agencias se presenta un listado general de agencias:

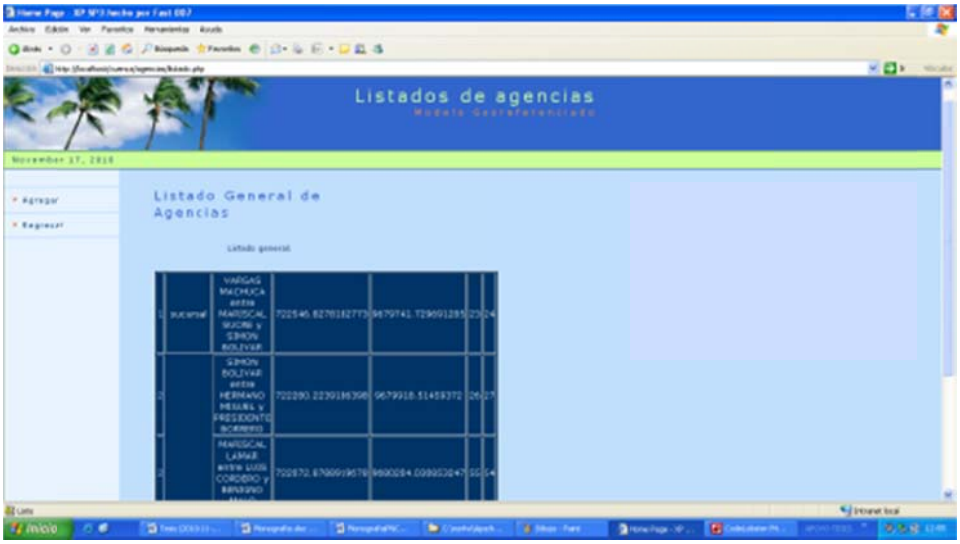


Figura 38: Listado General de Agencias
Fuente: (Autores)

En la opción Agregar tenemos la interfaz para ubicación de las Agencias:

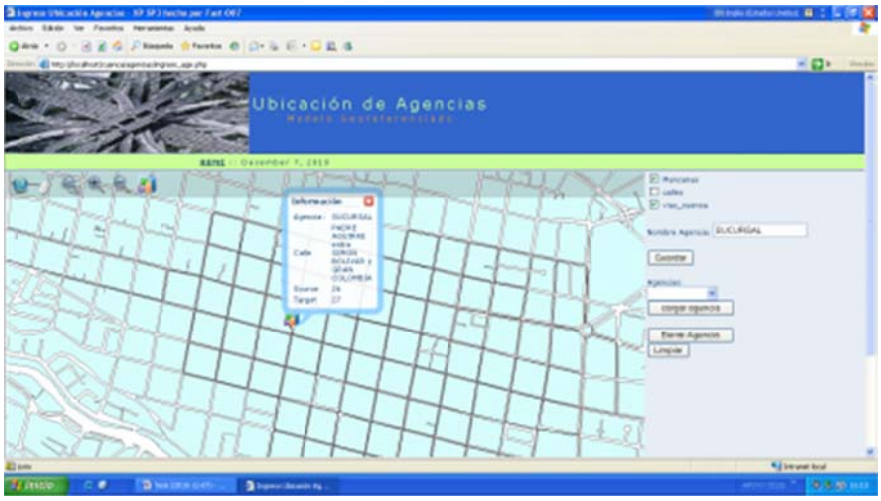


Figura 39: Interfaz para georreferenciar Agencias
Fuente: (Autores)

3.18.4. La interfaz para Hoja de rutas

A través de la página principal de la aplicación, accediendo en la opción de Hoja de ruta.

3.18.4.1. Recorrido a Pie en Hoja de Ruta

En la interface para Hoja de ruta a través del botón **Dibujar Ruta** se genera la ruta sin las consideraciones del sentido de las calles.

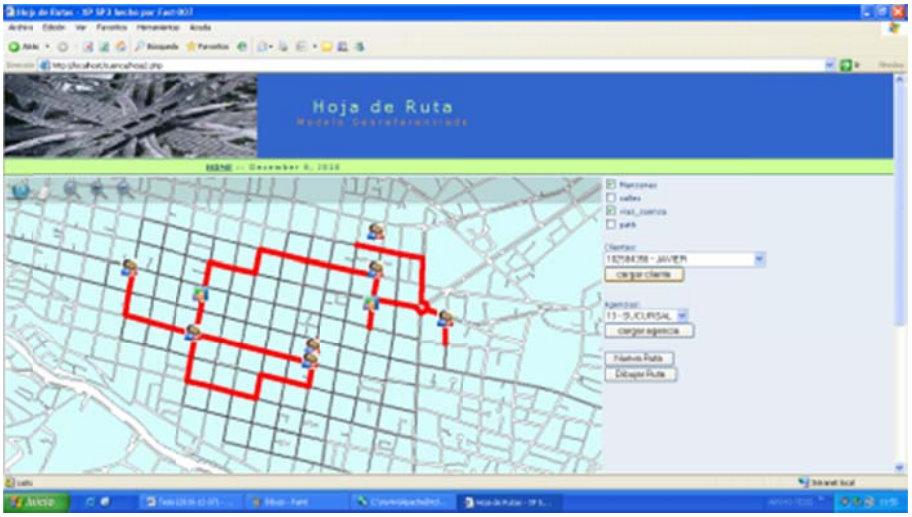


Figura 40: Hoja de Rutas recorrido a Pie
Fuente: (Autores)

3.18.4.2. Recorrido en Vehículo en Hoja de Ruta

En la interface para Hoja de ruta a través del link **Sentido de Vias** podemos obtener el recorrido en vehículo generando la ruta que respete el sentido de las calles, ya sea una vía o doble vía.

En el siguiente gráfico, mostramos a manera de ejemplo, el recorrido desde una agencia ubicada en la calle Padre Aguirre con dirección Norte y dos clientes, el cliente 1 ubicado en la misma calle y el cliente 2 en la calle Gran Colombia con direccion Este. Note que para dirigirse al cliente 2 desde la agencia se toman las calles PadreAguirre, Mariscal Lamar, General Torres y Gran Colombia respetando el sentido de circulación.

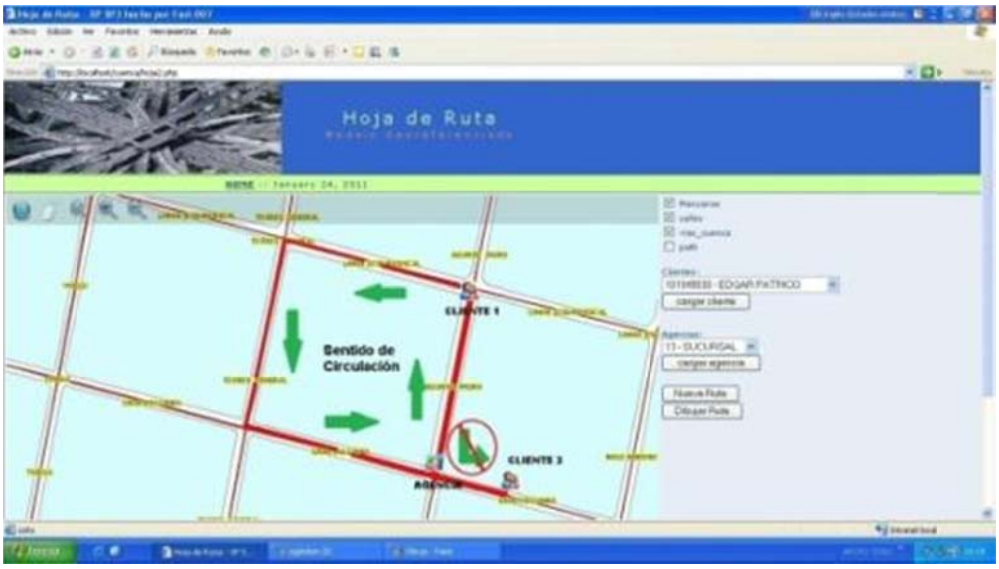


Figura 41: Hoja de Rutas recorrido en Vehículo
Fuente: (Autores)

3.19. Consultas realizadas

La realización de consultas son llevadas a cabo con el propósito de sacar un mayor provecho de nuestro modelo georreferenciado y proporcionar a los ejecutivos y los altos mandos de una empresa información relacionada con las ventas realizadas a los clientes.

Para ello, a más de la tabla de clientes que tenemos ya creada en nuestra base de datos fue necesario crear dos nuevas tablas: productos y compras.

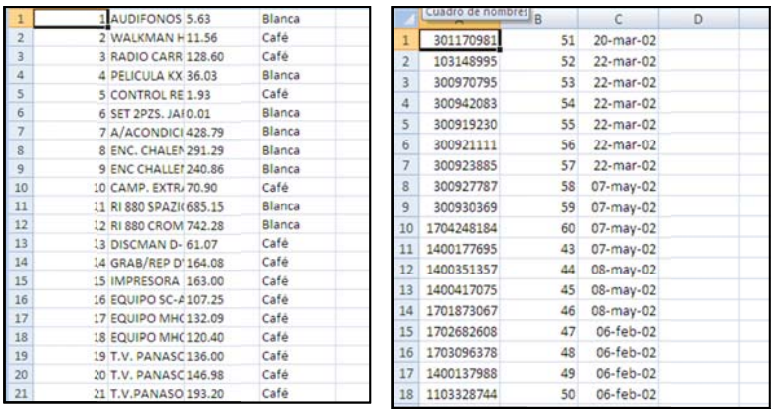
La tabla productos permitirá almacenar los productos y la línea del mismo, identificándolos a través de un ID único, el script para crear esta tabla es el siguiente:

```
CREATE TABLE productos
(
  pro_id character(10),
  nombre character(100),
  precio real,
  linea character(10)
)
WITHOUT OIDS;
ALTER TABLE productos OWNER TO postgres;
```

La tabla compras permite relacionar la cedula del cliente y el ID del producto, posee también la fecha en que fue efectuada la compra. El script para crear la tabla es:

```
CREATE TABLE compras
(
  ci character(15),
  pro_id character(10),
  fecha_compra date
)
WITHOUT OIDS;
ALTER TABLE compras OWNER TO postgres;
```

Los datos ficticios para llenar los registros de estas tablas fueron creados a partir de dos archivos de texto plano (CSV) de bases de datos que fueron creados por nosotros para aplicarlos exclusivamente con estas consultas.



1	1	AUDIFONOS	5.63	Blanca
2	2	WALKMAN H	11.56	Café
3	3	RADIO CARR	128.60	Café
4	4	PELICULA KX	36.03	Blanca
5	5	CONTROL RE	1.93	Café
6	6	SET 2PZS. JAI	0.01	Blanca
7	7	A/ACONDICI	428.79	Blanca
8	8	ENC. CHALEN	291.29	Blanca
9	9	ENC CHALLEI	240.86	Blanca
10	10	CAMP. EXTR/	70.90	Café
11	11	RI 880 SPAZI	685.15	Blanca
12	12	RI 880 CROM	742.28	Blanca
13	13	DISCMAN D-	61.07	Café
14	14	GRAB/REP D	164.08	Café
15	15	IMPRESORA	163.00	Café
16	16	EQUIPO SC-A	107.25	Café
17	17	EQUIPO MHC	132.09	Café
18	18	EQUIPO MHC	120.40	Café
19	19	T.V. PANASC	136.00	Café
20	20	T.V. PANASC	146.98	Café
21	21	T.V.PANASO	193.20	Café

	Cuadro de nombres	B	C	D
1	301170981	51	20-mar-02	
2	103148995	52	22-mar-02	
3	300970795	53	22-mar-02	
4	300942083	54	22-mar-02	
5	300919230	55	22-mar-02	
6	300921111	56	22-mar-02	
7	300923885	57	22-mar-02	
8	300927787	58	07-may-02	
9	300930369	59	07-may-02	
10	1704248184	60	07-may-02	
11	1400177695	43	07-may-02	
12	1400351357	44	08-may-02	
13	1400417075	45	08-may-02	
14	1701873067	46	08-may-02	
15	1702682608	47	06-feb-02	
16	1703096378	48	06-feb-02	
17	1400137988	49	06-feb-02	
18	1103328744	50	06-feb-02	

Figura 42: Archivos planos CSV de datos de Productos y Compras
Fuente: (Autores)

Para obtener una copia de los datos de estos archivos dentro de las tablas creadas en nuestra base de datos ingresamos al pgAdmin III, abrimos nuestra base de datos cuenca y ejecutamos el siguiente comando para cada tabla:

Para la tabla productos:

```
COPY
productos(pro_id,nombre,precio,linea) FROM 'c:/pro.csv' DELIMITERS ';' WITH
CSV
```

Para la tabla compras:

```
COPY
compras(ci,pro_id,fecha_compra) FROM 'c:/compras.csv' DELIMITERS ';' WITH
CSV
```

Como indicamos anteriormente, no olvide colocar la ruta correcta donde se encuentren los archivo CSV, en nuestro caso en la raíz del disco C:

Además fue necesario crear un nuevo archivo PHP/MapScript para manejar las consultas, denominado **consultas.php**. Es importante indicar que en el código de todas las consultas efectuadas en la clausula WHERE **where@x@is@not@null** se utiliza para poder seleccionar solo a los clientes georreferenciados en la tabla Clientes.

3.19.1. Consulta de los Clientes que han comprado un producto determinado

En ciertas circunstancias un jefe del departamento de ventas, quería conocer información descriptiva como nombre, dirección, etc de los clientes que compraron y se les entregaron cierto producto hace mucho tiempo atrás, con la información georreferenciada de ese cliente podría ubicar exactamente en el mapa de la ciudad la vivienda o lugar donde se fue entregado ese producto.

El código para la implementación de esta consulta, es el siguiente:

```
function cargar_clientes(pro_id)
{
        var datos,map;
        var x, y, calle,source,target;
        puntos=0;
        map=myMap1;
        myMap1.removeOverlayPoints();  // Rimuove tutti gli overlay

        sql="xml.php?sql=select@gid,x,y,source,target,calle,apellido_paterno,
nombres,telefono_domicilio@from@clientes@where@ci@in@(select@distinct@
ci@from@compras@c@where@c.ci@in@(select@ci@from@clientes@where@x@i
s@not@null)@and@c.pro_id@="+ pro_id+")";

        datos1=leerxml(sql);
        if (datos1=="")
        {
                alert('No existe coleccion de puntos ha ser cargados');
        }
        tmp= datos1.split("@*@");
                for (i=0;i<=tmp.length-2;i++)
        {
                tmp1= tmp[i].split("@");
                gid=trim(tmp1[0]);
                x=trim(tmp1[1]);
                y=trim(tmp1[2]);
                source=trim(tmp1[3]);
                target=trim(tmp1[4]);
                calle=trim(tmp1[5]);
                apellido_paterno=trim(tmp1[6]);
                //apellido_materno=trim(tmp1[7]);
                nombres=trim(tmp1[7]);
                telefono_domicilio=trim(tmp1[8]);
                puntos+=1;
                //alert("source: " + source );
                //objForm.elements[1].value= nombres + " " + apellido_paterno
+ " " + apellido_materno;
                Point = new pointOverlay( new mslcon('img/fin.png', '', 13, 39),
null, 'Información', x, y,
                    new                Array('Nombres:',                'Apellido:',
'Direccion:','Source','Target'),                new                Array(nombres,
apellido_paterno,calle,source,target) );
                map.addPointOverlay(Point);
        }
         puntos=i;
}
```

En esta función se envía como parámetro el ID del producto y la subconsulta se realiza entre las tablas clientes y compras.

El resultado en la interfaz seria el siguiente:

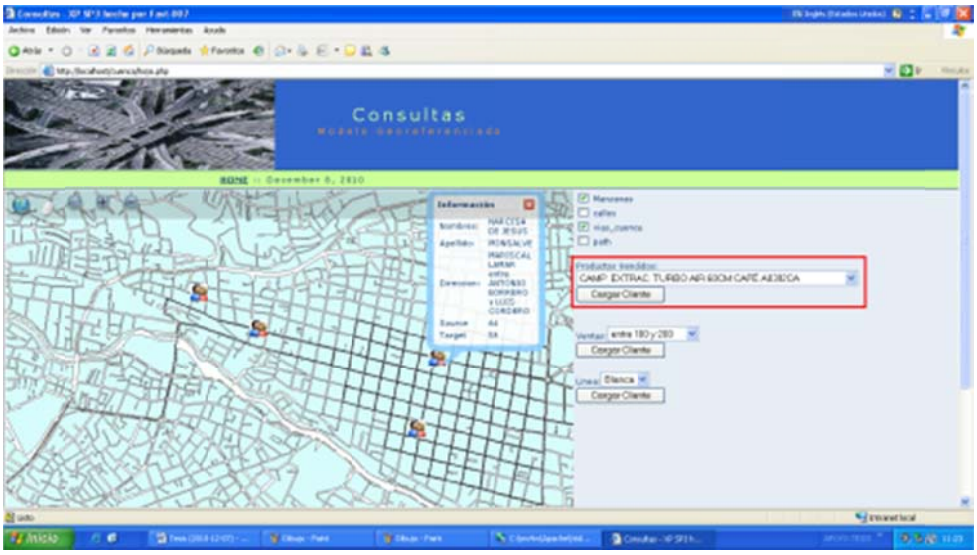


Figura 43. Consulta de Clientes por producto vendido
Fuente: (Autores)

3.19.2. Consulta parametrizada de Ventas realizadas a Clientes

Por otro lado un jefe del departamento de ventas, le gustaría saber donde se ubican en el mapa todos los clientes a los que se les ha vendido cierto monto de ventas, parametrizado por un rango de ventas, por ejemplo: Ventas menores entre $100 y $200 dólares y mayores entre $1000 y $2000 dólares.

El código para la implementación de esta consulta, es el siguiente:

```
function cargar_clientes_precio(op_id)
{
        var datos,map;
        var x, y, calle,source,target;
        var entre;
        entre="100@and@200";
        puntos=0;
        map=myMap1;
        myMap1.removeOverlayPoints();  // Rimuove tutti gli overlay
        if (op_id==1)
        {
        entre="100@and@200";
        }
        if (op_id==2)
        {
        entre="200@and@400";
        }
```

```
        if (op_id==3)
        {
        entre="400@and@600";
        }
        if (op_id==4)
        {
        entre="600@and@1000";
        }
        if (op_id==5)
        {
        entre="1000@and@2000";
        }

        sql="xml.php?sql=select@gid,x,y,source,target,calle,apellido_paterno,
nombres,telefono_domicilio@from@clientes@where@ci@in@(select@distinct@
ci@from@compras@c@where@c.ci@in@(select@ci@from@clientes@where@x@i
s@not@null)@and@c.pro_id@in@(select@pro_id@from@productos@where@pre
cio@between@"+entre+"))";

        datos1=leerxml(sql);
        if (datos1=="")
        {
                alert('No existe coleccion de puntos ha ser cargados');
        }
        tmp= datos1.split("@*@");
                for (i=0;i<=tmp.length-2;i++)
        {
                tmp1= tmp[i].split("@");
                gid=trim(tmp1[0]);
                x=trim(tmp1[1]);
                y=trim(tmp1[2]);
                source=trim(tmp1[3]);
                target=trim(tmp1[4]);
                calle=trim(tmp1[5]);
                apellido_paterno=trim(tmp1[6]);
                //apellido_materno=trim(tmp1[7]);
                nombres=trim(tmp1[7]);
                telefono_domicilio=trim(tmp1[8]);
                puntos+=1;
                //alert("source: " + source );
                //objForm.elements[1].value= nombres + " " + apellido_paterno
+ " " + apellido_materno;
                Point = new pointOverlay( new mslcon('img/fin.png', '', 13, 39),
null, 'Información', x, y,
                    new                Array('Nombres:',                'Apellido:',
'Direccion:','Source','Target'),                new                Array(nombres,
apellido_paterno,calle,source,target) );
                map.addPointOverlay(Point);
        }
         puntos=i;
}
```

En esta función se envía como parámetro un ID relacionado a los diferentes rangos de Ventas que podría seleccionar el usuario y la subconsulta se realiza entre las tablas clientes, compras y productos.

El resultado de la interfaz es el siguiente:

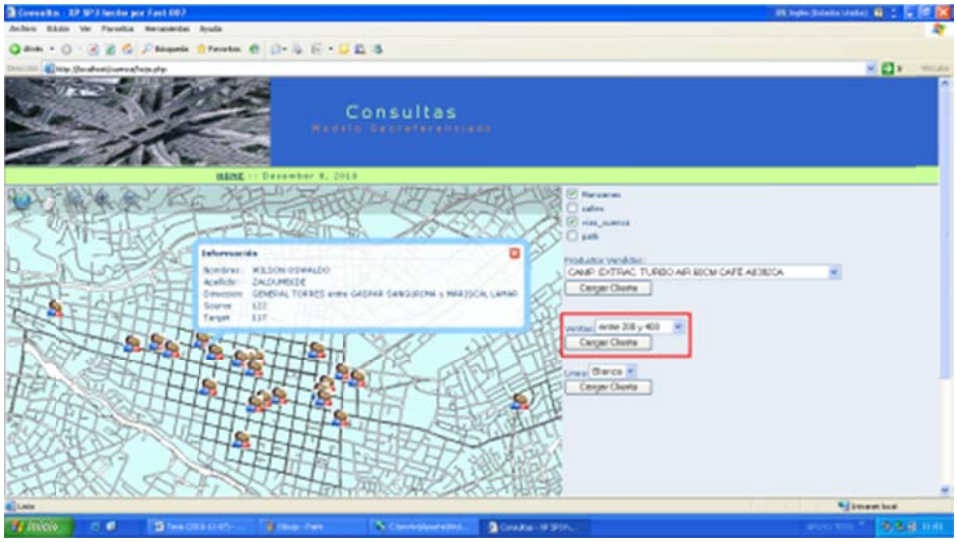


Figura 44: Consulta parametrizada de Ventas a Clientes
Fuente: (Autores)

3.19.3. Consulta de los Clientes por la Línea del producto comprado

Una consulta por Línea de producto vendida en un almacén de electrodomésticos por ejemplo es mucho más generalizada porque no se necesita saber si se trata específicamente de una cocina o una refrigeradora, sino de cualesquiera de las dos. Realizar una consulta de aquellos clientes que compraron cierta Linea de producto como podría ser Linea Blanca o Linea café y poder ubicar en el mapa el lugar, puede ser una necesidad oportuna para un jefe de departamento de ventas.

El código para la implementación de esta consulta, es el siguiente:

```
function cargar_clientes_linea(linea)
{
        var datos,map;
        var x, y, calle,source,target;
        var entre;
        puntos=0;
        map=myMap1;
```

```
        myMap1.removeOverlayPoints();  // Rimuove tutti gli overlay
        sql="xml.php?sql=select@gid,x,y,source,target,calle,apellido_paterno,
nombres,telefono_domicilio@from@clientes@where@ci@in@(select@distinct@
ci@from@compras@c@where@c.ci@in@(select@ci@from@clientes@where@x@i
s@not@null)@and@c.pro_id@in@(select@pro_id@from@productos@where@line
a@=@'"+linea+"'))";
        datos1=leerxml(sql);
        if (datos1=="")
        {
                alert('No existe coleccion de puntos ha ser cargados');
        }
        tmp= datos1.split("@*@");
                for (i=0;i<=tmp.length-2;i++)
        {
                tmp1= tmp[i].split("@");
                gid=trim(tmp1[0]);
                x=trim(tmp1[1]);
                y=trim(tmp1[2]);
                source=trim(tmp1[3]);
                target=trim(tmp1[4]);
                calle=trim(tmp1[5]);
                apellido_paterno=trim(tmp1[6]);
                //apellido_materno=trim(tmp1[7]);
                nombres=trim(tmp1[7]);
                telefono_domicilio=trim(tmp1[8]);
                puntos+=1;
                //alert("source: " + source );
                //objForm.elements[1].value= nombres + " " + apellido_paterno
+ " " + apellido_materno;
                Point = new pointOverlay( new mslcon('img/fin.png', '', 13, 39),
null, 'Información', x, y,new Array('Nombres:', 'Apellido:',
'Direccion:','Source','Target'), new Array(nombres,
apellido_paterno,calle,source,target) );
map.addPointOverlay(Point);
}
 puntos=i;}
```

En esta función se envía como parámetro la línea de los productos seleccionada por el usuario y la subconsulta se realiza de igual manera entre las tablas clientes, compras y productos.

El resultado de la interfaz es el siguiente:

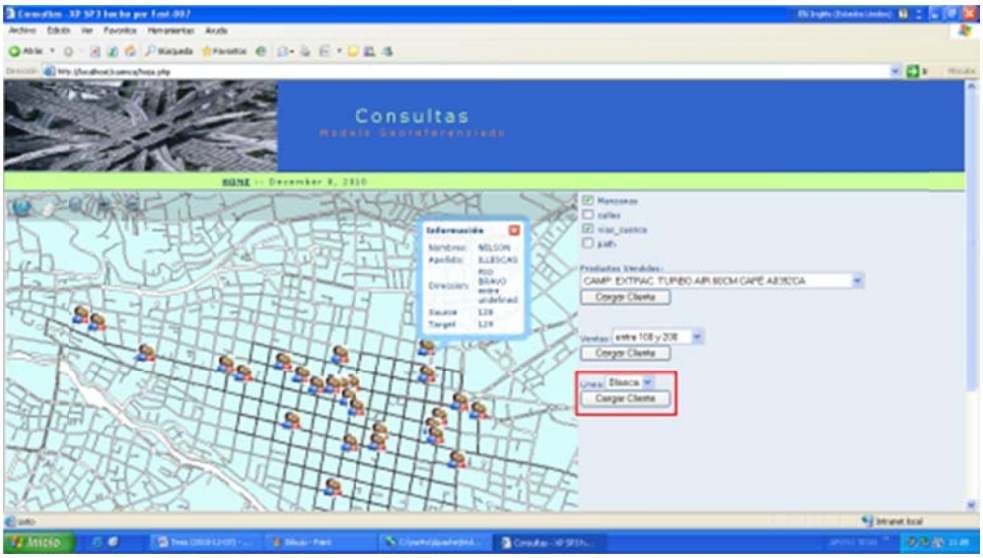


Figura 45: Consulta de Clientes por Línea de productos comprada
Fuente: (Autores)

CONCLUSIONES

El cliente ligero msCross y la implementación de la librería pgRouting nos proporcionan todos los servicios que necesitamos para proporcionarle la capacidad de ruteo a nuestro sistema PostgreSQL-PostGis,

Adicionalmente, el uso de herramientas de código abierto (Open Source) nos provee grandes ventajas, como facilidad de implementar nuevas funcionalidades a nuestros desarrollos a través de programación, facilidad para obtener las herramientas en la Web, licencias gratuitas entre otras.

La generación de una ruta, es un proceso que conlleva varios pasos, una vez que se tienen la información cartográfica en el sistema de coordenadas adecuado y debidamente publicado se necesita interactuar con el mismo y esto se logra mediante la programación del cliente mscross y el uso de las funciones Pgrouting que permitan generar la ruta de los puntos ubicados con lo cual el cobrador sabrá donde dirigirse con exactitud a realizar su ciclo de cobranza cuyo recorrido podrá realizarse a Pie o en Vehículo con las consideraciones respectivas en ambas casos y además estos objetos geográficos tienen asociados atributos que almacenan información descriptiva referente a los clientes, agencias que permiten a los cobradores realizar una correcta gestión de cobros.

CAPITULO 4

MANUAL DEL PROYECTO

INTRODUCCIÓN

Una vez desarrollado e implementado nuestro prototipo en un Servidor Web, es necesario y muy importante que los usuarios obtengan el mayor provecho posible de la aplicación conociendo las diferentes opciones y funcionalidades que este ofrece, Por esta razón, este capítulo va dirigido exclusivamente a la capacitación de los usuarios, detallando las herramientas y las operaciones que ellos pueden realizar.

4.1. Ingreso al Sistema

Para ingresar al Sistema, acceda a la página principal de la Universidad del Azuay mediante la siguiente dirección url: www.uazuay.edu.ec, una vez que nos encontremos en el sitio, presionamos el link Proyectos Geomáticos, el cual nos enviará a la página en donde se listan todos los Proyectos Geográficos desarrollados, escoja Modelo de un Sistema Georreferenciado de Gestión de Cobros y Entrega de Pedidos y debe visualizarse la siguiente interfaz:

Figura 46: Interfaz Principal de la Aplicación
Fuente: (Autores)

4.2. Herramientas de msCross

Para la implementación del sistema utilizamos las herramientas de msCross con las cuales el usuario puede interactuar con los mapas y manipular la información que se presenta, las operaciones que puede realizar son: colocación del mapa, acercamiento a una sección específica, zoom, paneo.

Este conjunto de controles básicos son claramente visibles y representadas en botones, se encuentran ubicados en la parte superior izquierda por encima de los mapas en todas las interfaces.

Figura 47: Ubicación de las herramientas msCross en la interface
Fuente: (Autores)

A continuación detallamos la función de cada uno de estos botones:

 → Le coloca al mapa en su tamaño original

 → Nos permite movernos a través del mapa

 → Hace un acercamiento al mapa de la zona señalada

 → Hace un acercamiento a todo el mapa

 → Hace un alejamiento del mapa

4.3. Visualización de capas temáticas

Al ingresar a las diferentes interfaces del sistema en el mapa que se carga, se mostrará siempre de manera predeterminada las capas de Manzanas y Vias_Cuenca para efectos de referenciación.

Usted puede marcar o desmarcar la casilla correspondiente a la capa temática de su preferencia. Dichas casillas las encontrará ubicadas en la parte superior derecha de la interface junto al mapa.

Figura 48: Ejemplo de Layer "calles "seleccionado
Fuente: Autores

4.4. Visualización de coordenadas geográficas

Cada vez que usted se deslice con el mouse sobre el área de cualquiera de los mapas de las interfaces se visualizarán las coordenadas X e Y en el

extremo inferior izquierdo. Tenga en cuenta que el sistema de coordenadas de todos los mapas visualizados es PSAD56.

Figura 49: Coordenadas ´X´ y ´Y´ visualizadas en la interface
Fuente:(Autores)

4.5. Interfaz Principal

Se pueden distinguir dos áreas específicas en esta pantalla: Área de información y Menú de opciones.

4.5.1. Área de información

Corresponde al área de la pantalla principal donde podrá encontrar información descriptiva del sistema y vínculos a diferentes páginas Web relacionados con nuestro proyecto.

Figura 50: Área informativa de la Interfaz Principal
Fuente:(Autores)

4.5.2. Menú de opciones

Corresponde al área de la pantalla principal con fondo de color celeste claro donde encontrará los diferentes menús disponibles (Clientes, Agencias, Hoja de Rutas y Consultas), a través de los cuales podrá acceder a las interfaces que conforman el sistema.

Figura 51: Menú de opciones de Interfaz Principal
Fuente:(Autores)

4.6. Interfaz Clientes

Para acceder a este módulo, haga clic en la opción Clientes del menú de opciones de la pantalla principal del sistema.

4.6.1. Listado y menú de Opciones para Clientes

Al acceder al modulo de clientes en la pantalla **Listado de Clientes** se muestra un listado general de clientes que es solo informativo. También encontrará un menú de opciones con las funciones que se indican a continuación

→ Georreferenciar a los clientes

→Modificar la Georreferenciación de los clientes

→ Regresar a la pantalla principal

4.6.2. Controles para Agregar / Modificar Clientes Georreferenciados

Para poder llevar a cabo la Georreferenciación de los clientes seleccione la opción **Agregar** del menú de opciones y en la pantalla **Ubicación de Clientes** siga los siguientes pasos:

- Seleccione un Cliente de la lista que muestra el control combo box siguiente:

Este se encuentra ubicado a lado derecho de la pantalla por debajo de las casillas de verificación de los layers,

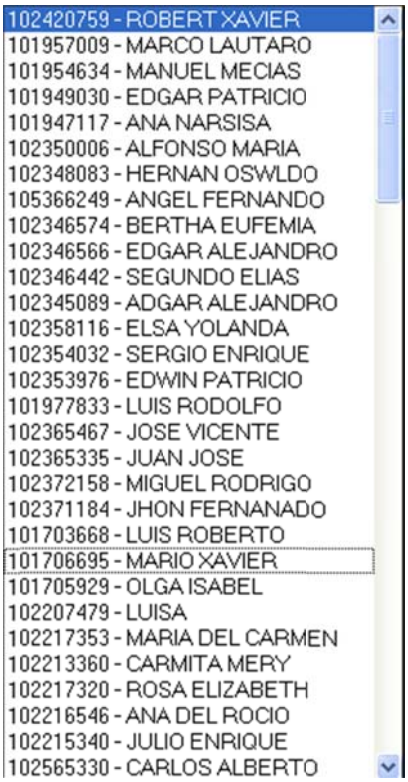


Figura 52: Listado de Clientes
Fuente:(Autores)

Note que la lista muestra los números de cédula y los nombres de los clientes.

- Haga clic en el botón  y señale el área del mapa que corresponde a la capa de **vías_cuenca** para hacer un acercamiento de las calles de la ciudad que vamos a utilizar para georreferenciar al cliente.

- Haga clic en el botón  que se encuentra ubicado ubicado en la parte superior izquierda por encima del mapa y forma parte del conjunto de controles del msCross y ubique el puntero del mouse sobre la calle de la ciudad que pertenece a la dirección del cliente.

- Haga clic sobre la calle ubicada, deberá aparecer el icono  que representa al cliente georreferenciado sobre el mapa. El sistema

actualizará en la base de datos los campos de Georreferenciación de los clientes recién ubicados sobre el mapa.

El botón 

le permite borrar del mapa los clientes ubicados y el botón 

carga sobre el mapa aquellos clientes que se encuentran georreferenciados.

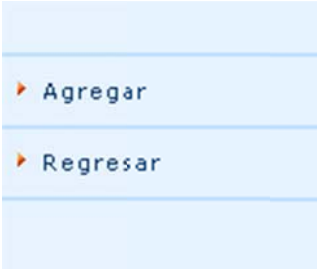
Nota: La interface para Modificar es igual a esta y se utilizará siguiendo los mismos pasos anotados.

4.7. Interfaz Agencias

Para acceder a este módulo, haga clic en la opción Agencias del menú de opciones de la pantalla principal del sistema.

4.7.1. Listado y menú de Opciones para Agencias

Al acceder al modulo de Agencias en la pantalla **Listado de Agencias** se muestra un listado general de las agencias que es solo informativo. También encontrará un menú de opciones con las funciones que se indican a continuación



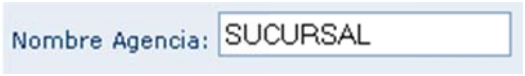
→ Georreferenciar a las Agencias

→ Regresar a la pantalla principal

4.7.2. Controles para Agregar Agencias Georreferenciados

Para poder llevar a cabo la Georreferenciación de las Agencias seleccione la opción **Agregar** del menú de opciones, en la pantalla **Ubicación de Agencias** siga los siguientes pasos:

- Ingrese el Nombre de la Agencia en el cuadro de texto, que se encuentra ubicado al lado derecho de la pantalla por debajo de las casillas de verificación de los capas.



- Haga clic en el botón  y señale el área del mapa que corresponde a la capa de **vías_cuenca** para hacer un acercamiento de las calles de la ciudad que vamos a utilizar para georreferenciar la Agencia.

- Haga clic en el botón  que se encuentra ubicado ubicado en la parte superior izquierda por encima del mapa y forma parte del conjunto de controles del msCross y ubique el puntero del mouse sobre la calle de la ciudad que pertenece a la dirección de la Agencia.

- Haga clic sobre la calle ubicada, deberá aparecer el icono  que representa a la Agencia georreferenciado sobre el mapa.

- Para guardar la ubicación de la Agencia recién georreferenciada en la base de datos, haga clic en el botón 

El sistema muestra una lista todas las Agencias georreferenciadas: en un control combo box 

como se muestra en la figura:

Figura 53: Listado de Agencias georreferenciadas
Fuente:(Autores)

Seleccione una Agencia y haga clic en el botón 

para visualizar la localización de la Agencia en el mapa.

Si desea borrar una Agencia haga clic en el botón 

esto eliminará de forma permanente de la base de datos la agencia por lo cual debe utilizar con cautela esta opción. Puede verificar que la agencia ha sido eliminada listando nuevamente las Agencias del control combo box.

El botón 

le permite borrar del mapa las agencias ubicadas.

4.8. Manejo de herramientas para la interfaz Hoja de Ruta

Seleccione la opción **Hoja de Ruta** del menú de opciones de la pantalla principal para ingresar a la interfaz que permite generar la Hoja de ruta óptima que deberá seguir el cobrador para realizar la entrega de paquetería y la gestión de cobro.

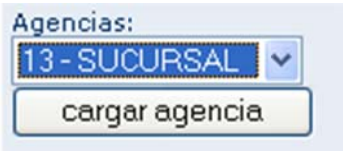
Para generar una hoja de ruta siga los siguientes pasos:

Si el recorrido de cobrador **es a Pie:**

- Seleccione un Cliente de la lista que muestra el control combo box y haga clic en el botón cargar cliente, aparecerá el icono  que representa al cliente.

Repita las veces que sea necesario hasta ubicar todos los clientes que necesite en el mapa.

- Seleccione una Agencia de la lista que muestra el control combo box y haga clic en el botón cargar agencia aparecerá el icono  que representa a la agencia.



Repita las veces que sea necesario hasta ubicar todas las agencias existentes.

- Haga clic en el botón  y señale el área del mapa que corresponde a la capa de **vías_cuenca** para hacer un acercamiento de las calles de la ciudad sobre las cuales se generará la ruta óptima.

- Haga clic en el botón 

para que el sistema genere la Ruta a Pie, tome en cuenta que la ruta no considerara el sentido de las calles.

Si el recorrido de cobrador **es en Vehículo**, haga clic en la url **Sentido de Vias** para que el sistema muestre la interfaz con la cual generar la ruta

óptima.que respete el sentido de circulación de las vías. Debera seguir los mismos pasos indicados anteriormente para dibujar la ruta.

El botón 

le permite borrar todos los puntos ubicados en el mapa y le permitirá generar una nueva hoja de ruta.

4.9. Manejo de herramientas para la interfaz de Consultas

Seleccione la opción **Consultas** del menú de opciones de la pantalla principal para ingresar a la interfaz que permita obtener las diferentes consultas realizadas

Para realizar la **Consulta de Clientes por producto vendido** siga estos pasos:

- Seleccione un Producto de la lista que muestra el control combo box

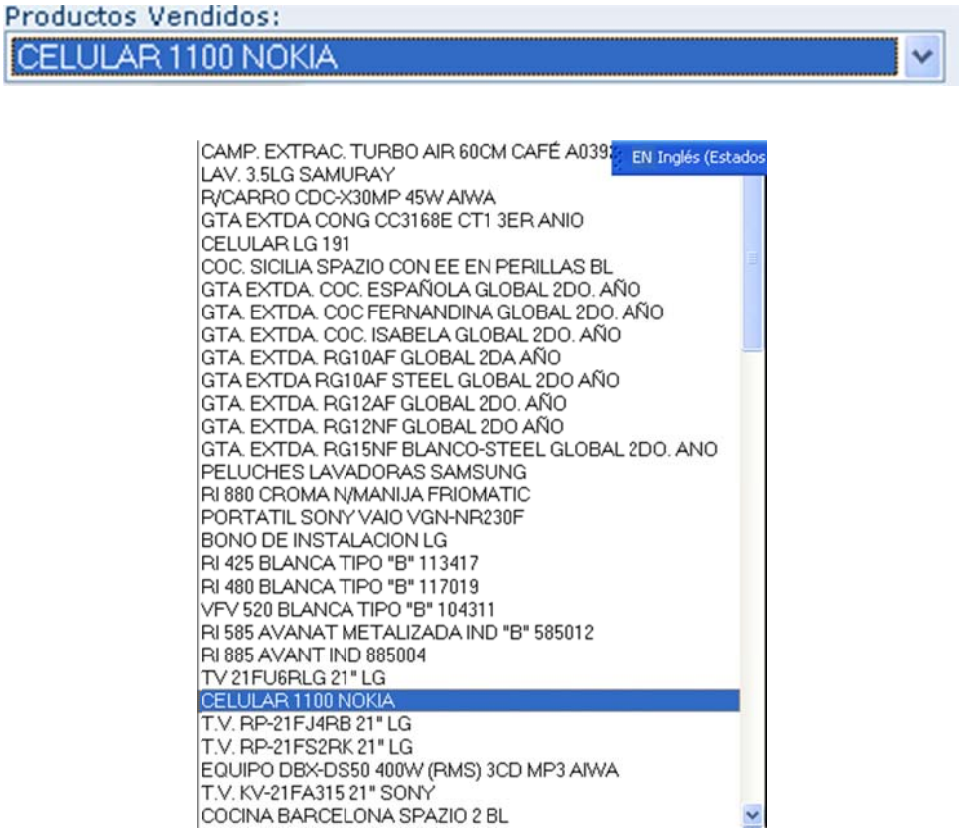


Figura 54: Listado de Productos para Consultar
Fuente:(Autores)

- Haga clic en el botón 

, aparecerá uno, varios o ningún icono  sobre el mapa dependiendo de aquellos clientes que hayan comprado el producto seleccionado.

Para realizar la **Consulta parametrizada de Ventas a Clientes** siga estos pasos:

- Seleccione un rango de Ventas de los disponible en la lista que muestra el control combo box

Figura 55: Rango de Ventas para Consultar
Fuente:(Autores)

- Haga clic en el botón 

, aparecerá uno, varios o ningún icono  sobre el mapa dependiendo de aquellos clientes a los que se les haya efectuado ventas en el rango seleccionado.

Para realizar la **Consulta de Clientes por Línea de productos comprada** siga estos pasos:

- Seleccione una línea de producto de la lista que muestra el control combo box

Figura 56: Líneas de productos para Consultar
Fuente:(Autores)

- Haga clic en el botón 

, aparecerá uno, varios o ningún icono  sobre el mapa dependiendo de aquellos clientes a los que se les haya efectuado ventas en el rango seleccionado.

4.10. Cuadro Informativo de Clientes, Agencias, Hoja de Ruta y Consultas

Con esto se podrá mostrar la información descriptiva de los Clientes y las Agencias como el nombre y la dirección exacta de las calles que la conforman entre otros datos, dicho **cuadro informativo** se puede acceder desde las diferentes interfaces del sistema.

Para ello haga clic sobre el icono que representa al Cliente  o a la Agencia  que necesita obtener información y el resultado en ambos casos es el siguiente:

Figura 57: Obtener cuadros informativos de Clientes y Agencias
Fuente:(Autores)

CONCLUSIONES.

Un sistema informático para ser catalogado como eficiente debe poder ofrecer a los usuarios toda su funcionalidad y prestaciones que posee a través de una interfaz sencilla, intuitiva y amigable.

A través de este manual pretendemos que el usuario logre familiarizarse con el uso de la aplicación, conociendo los diferentes controles y herramientas que posee. Además, en el manual se han indicado de forma sencilla y objetiva los pasos que el usuario deberá seguir para interactuar con la aplicación para que no representen problema alguno y de esta manera pueda conseguir los resultados esperados.

CONCLUSIONES Y RECOMENDACIONES

Se ha logrado llevar a cobo un prototipo SIG que permite publicar y georreferenciar las direcciones de clientes y agencias de un sector de la ciudad de Cuenca. Este prototipo posee, además de la generación de rutas optimas de recorrido para entrega de paquetería y seguimiento de los cobros, un valor agregado que consiste en la localización geográfica de clientes en función de diferentes requerimientos (productos adquiridos, líneas de productos, rango de precios, etc)

El software ArcGis posee las herramientas necesarias como los componentes metadata y geodatabase que permite gestionar eficazmente la información y lo utilizamos para desarrollar el prototipo.

La utilización de Mapserver como servidor de mapas, permitió de forma fácil y sencilla publicar la información cártografica a través del cliente ligero msCross.

La base de datos PostGreSQL con su extensión PostGis nos permitió el almacenacimiento de los datos y el manejar espacialmente los mismos.

Las herramientas de software libre que hemos descrito y utilizado en esta tesis deberían emplearse siempre en estos desarrollos pues ofrecen todos los beneficios que una herramienta de software propietario.

El cliente ligero msCross con la librería PgRouting y sumado a la programación JavaScript que fue llevado a cabo nos permitió obtener la funcionalidad de ruteo a nuestro prototipo.

La ruta generada a través de los diferentes puntos tiene una precisión de cercanía de unos 10 metros por lo que no es exacto al punto.

Se recomienda que tomando como base el desarrollo de este prototipo se puedan implementar otros desarrollos que permitan obtener mayor

beneficio como podría ser el poder obtener la ubicación precisa de los cobradores en la hoja de ruta en la calles de la ciudad de Cuenca a través de la utilización de navegadores GPS (Global Positioning System).

El estudio de herramientas msCross, la librería pgRouting se encuentran en etapas iniciales y no existe mucha información disponible en la Web sobre las mismas, por ello se recomienda realizar mayores desarrollos en las mismas ya que son completamente programables y de fácil acceso como proyecto de software libre para desarrollar aplicaciones SIG.

BIBLIOGRAFÍA

- CIEMAT, Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas, Madrid – España, Octubre 2000.
- The GIS Software Leader., http://www.esri.com
- OCHOA A. Paúl. Tutorial ArcGis. Septiembre 2008.
- MARTÍNEZ, Rafael, Universidad de Oslo, 2009.
- RAMSEY, Paúl. (2008). PostGIS Manual.
- GONZALES, Jaramillo, V. H. (2005). MapServer y su aplicación a SIG. Obtenido de http://geovisualiza.blogspot.com/
- ORTEGA Pamela., & FLORES Cristina, (2009). Publicación en la Internet del Mapa Turístico del Parque Nacional Cajas.
- POSADA, David, Guía básica del Lenguaje HTML, 2006. Obtenido de http://www.arrakis.es/~wenceslao/CursoWeb/1/guia.html#seccion2
- ALEJANDRO, PHP – Conceptos Básicos, 2008, Obtenido de http://blogdeprogramacion.blogspot.com/2007/12/php-conceptos-basicos.html
- CARRIÓN, Daniela, Manual de Javascript, Obtenido de http://www.uazuay.edu.ec/estudios/sistemas/lenguaje_iii/MAnualJavaScript/centro.htm
- W3C (Web, Consorcio), Guía Breve de Tecnologías XML, 2005. Obtenido de http://www.w3c.es/Divulgacion/GuiasBreves/TecnologiasXML
- CARRILLO, Germán, Comparación de clientes ligeros web para SIG, 2009, Obtenido de http://www.geotux.tuxfamily.org
- ESPINOZA M. Andrea & ESPINOZA M. Gabriela, (2010). Sistema de Redes Viales de la Ciudad de Cuenca y del Ecuador para la navegación con GPS.

- MARIÑO A. Lorena & MONCAYO T. Anita, (2008), Publicación en la Internet del Atlas de la Provincia del Azuay mediante un servidor de mapas, componente: Imagen de la Provincia

- CAU Pierluigi & MANCA Simone, msCross: un cliente WEB AJAX, 2006

- BARRIENTOS MARTÍNEZ Miguel Ángel. Network Analyst, el Análisis de Redes desde ArcGis. http://www.mediafire.com/?5dyqctodwyl

- PACHECO, D (Diciembre de 2008). Apuntes en Clase: Curso de Graduación. Cuenca.

- BEHNCKE Kai & THURKOW Florian, (Junio 2007), **pgRouting (with the A*-Algorithm)** and the UMN MapServer, Creative Commons, Obtenido de http://www.umn-mapserver-community.de

ANEXOS

Anexo 1. Edicion del archivo .map

El archivo .map generado nos presenta ciertos inconvenientes al momento de cargarlo en el navegador, por lo que debemos hacer ciertas correcciones, las cuales se detallarán a continuación:

1. En la parte donde se detallan las coordenadas del extent del mapa, deben ser cambiadas las comas por puntos.

2. Borrar o comentar las líneas que tengan ANTIALIAS FALSE, para comentarlas coloque el signo **#** al comienzo de la línea.

3. Reemplazar las etiquetas STYLE por CLASS, únicamente en los lugares en donde se definen las capas, dejar STYLE en donde se especifique el formato de la misma.

4. En la línea FONTSET coloque la dirección en donde se encuentra el archivo.txt, que tiene los fonts a usar.

5. Luego de la línea de SYMBOLSET que comento en el paso anteriormente detallado, haga las definiciones de los diferentes símbolos a utilizar para dar formato a cada capa. Por ejemplo:

```
SYMBOL
   NAME 'circle'
                TYPE ELLIPSE
                FILLED TRUE
   POINTS 5 5 END
END
```

   Permite utilizar un círculo para que los puntos de una capa de este tipo se visualicen con esta figura.

6. En cada una de las capas, configure el color del símbolo si utiliza alguno o simplemente el color con el que se visualizarán los objetos geográficos que contiene cada capa. Ejemplo:

```
CLASS
  STYLE
    SYMBOL 'circle' #indica que utilizara el símbolo circle
    COLOR 230 0 0 #indica el color de relleno del mismo
    OUTLINECOLOR 0 0 0 # define el color del contorno del círculo
        SIZE 1 #
END #end style
```

7. Para añadir etiquetas (labels) a las entidades graficas, utilize LABELCACHE ON, use las siguientes líneas:

```
LABELITEM "texto"
LABELCACHE ON
LABEL
    COLOR 0 0 0
    FONT sans
    TYPE TRUETYPE
    POSITION CC
    PARTIALS TRUE
    SIZE 7
    BUFFER 1
    OUTLINECOLOR 255 255 0
END
```

Anexo 2. Archivos xml.php y dbconsultas.js

xml.php

```
<?php
$sqlrec=str_replace("@", " ", $_GET["sql"]);
$sqlrec=str_replace("\ \", "", $sqlrec);
include('./Connections/cnnsrvmps.php');
$conexion=$dbconn;
if (!conexion)
{
  echo "Error en conexion a la Base de Datos";
  exit;
}
        $hoy=date("Y").'-'.date('m').'-'.date('d');
        $sql="$sqlrec";
        $resultado = pg_Exec($conexion,$sql);
      $filas=pg_NumRows($resultado);
      $cols=pg_NumFields($resultado);
        header('Content-type: text/xml');
        $xml .= '<?xml version="1.0" encoding="ISO-8859-1"?>';
      $xml.='<clases>';
              $xml .= '<dato>';
                  $xml .= "<filas>$filas</filas>";
                  $xml .= "<columnas>$cols</columnas>";
              $xml .= '</dato>';
              while($fila = pg_fetch_array($resultado, null, PGSQL_ASSOC)) {
              $xml .= '<dato>';
                $i=0;
                foreach ($fila as $valor_col)
        {
                        $xml .= "<campo$i>$valor_col </campo$i>";
                        $i++;
                }
              $xml .= '</dato>';
  }
  $xml.='</clases>';
echo $xml;
?>
```

dbconsultas.js

```
////////////**********************************/////////////////////////
// FUNCION QUE LEE UN XML DINAMICAMENTE SIN NECESIDAD DE SABER EL #
DE LINEAS O CAMPOS
// WWW.GEONETSI.COM
// ING. DIEGO PACHECO
////////////**********************************/////////////////////////
```

```
var xmlDocm;
var xmlDocie;
var retornoie;
var datam;

function leerxml(url)
{
  var mozilla = (typeof document.implementation != 'undefined')&& (typeof
document.implementation.createDocument != 'undefined');
  var ie = (typeof window.ActiveXObject != 'undefined');
  dir= url;
  if (mozilla)
   {
  xmlDocm = document.implementation.createDocument("", "", null);
  xmlDocm.async=false;
  xmlDocm.onload = writeList;
  xmlDocm.load(dir);
  }
  else
  {
          if (ie)
                  {
                  xmlDocie = new ActiveXObject("Microsoft.XMLDOM");
                  xmlDocie.async = false;
                  xmlDocie.load(dir);
                  if (xmlDocie.readyState == 4) {retornoie = writeList1(xmlDocie);}
                  }
          else {alert('Tu navegador no puede manejar este script')}
 }
if (retornoie==undefined)
{ retornoie= datam;}
return retornoie;
}
function writeList()
{
   var labels = xmlDocm.getElementsByTagName('dato');
   var ol = document.createElement('OL');
   filas=labels[0].childNodes[0].firstChild.nodeValue;
   cols=labels[0].childNodes[1].firstChild.nodeValue;
   datam='';
   for (i=1; i <= filas; i++) //barro todas las líneas y columnas del xml
    {
          for (j=0; j < cols; j++)
          {
                  datam = datam + labels[i].childNodes[j].firstChild.nodeValue +
'@'; //@ sera usado como separadador de campos
          }
         datam = datam + '@*@'; //sera utilizado como separador de registros
      var li = document.createElement('LI');
```

```
      var labelId = document.createTextNode('(' + labels[i].getAttribute('id') +
')');
      li.appendChild(labelId);
      ol.appendChild(li);
    }
    retornoie=datam;
}
function writeList1(xml)
{
////////////////FUNCION IMPLEMENTADA PARA FUNCIONAR EN IE 6 O
SUPERIOR/////////////////
xmlDocie=xml;
  var labels = xmlDocie.getElementsByTagName('dato');
  var ol = document.createElement('OL');
         filas=labels[0].childNodes[0].firstChild.nodeValue;
         cols=labels[0].childNodes[1].firstChild.nodeValue;
         dataie='';
    for (i=1; i <= filas; i++) //barro todas las líneas y columnas del xml
    {
          for (j=0; j < cols; j++)
          {
                dataie = dataie + labels[i].childNodes[j].firstChild.nodeValue +
'@'; //@ sera usado como separadador de campos
          }
          dataie = dataie + '@*@'; //sera utilizado como separador de registros
      var li = document.createElement('LI');
      var labelId = document.createTextNode('(' + labels[i].getAttribute('id') +
')');
      li.appendChild(labelId);
      ol.appendChild(li);
    }
         return dataie;
}

function sleep(milliseconds) {
  var start = new Date().getTime();
  for (var i = 0; i < 1e7; i++) {
    if ((new Date().getTime() - start) > milliseconds){
      break;
    }
  }
}
function formatear(str)
{
tabla="";
vector= str.split("@*@");
nreg=vector.length-1;
for (i=0; i<nreg;i++ ) //barro los registros
{
res= i%2;
         if (res==0)
```

```
        {tabla= tabla + '<tr>';}
        else
        {tabla= tabla + '<tr bgcolor=8cd4e2>';}
        tmpc=vector[i];
        vector1=tmpc.split("@");
        ncol=vector1.length-1;
        for (j=0;j<ncol;j++)
        {
        tabla= tabla + '<td>' + vector1[j] + '</td>';
        }
tabla= tabla + '</tr>';
}
return tabla;
}
```

Anexo 3. Archivo cuencaesf.map de visualización del mapa

```
MAP
 IMAGETYPE   JPEG
 EXTENT      3428000 5787000 3444000 5800000
 SIZE        600 600

 IMAGECOLOR  255 255 255
 SHAPEPATH   "C:/ms4w/Apache/htdocs/routing/data"
SYMBOLSET    "C:/ms4w/Apache/htdocs/routing/symbols.sym"
 FONTSET     "C:/ms4w/Apache/htdocs/routing/fonts/fonts.txt"


 WEB
 TEMPLATE "template.html"
  IMAGEPATH "C:/ms4w/Apache/htdocs/tmp/"
  IMAGEURL  "/tmp/"

 END

NAME "MS"
        STATUS ON
        SHAPEPATH "/your_data_directory/" #Make sure this points to the root
of the data folder (where all your shape or raster files are)
        SIZE 800 400
        IMAGECOLOR 255 255 255
        IMAGETYPE png
        EXTENT 713606.846 9675184.594 733223.215 9686001.686
        UNITS meters
        PROJECTION
                "init=epsg:24877"
        END #end projection

        DEBUG ON
        WEB
                TEMPLATE "/your_data_directory/00000-
00099/00015/wms/mapserv_template.html"
                IMAGEPATH "/wms/tmp"
                IMAGEURL "/tmp/"
                LOG "/wms/tmp/manzanas.log"

        END #end web

        LAYER
                NAME 'manzanas'
                GROUP 'manzanas'

         CONNECTIONTYPE postgis CONNECTION "host=localhost
dbname=cuenca user=postgres password=postgres port=5432"
                DATA "the_geom from manzanas"
```

```
        PROJECTION
            "init=epsg:24877"
        END #end projection

        TYPE polygon
        STATUS ON
        TOLERANCE 8 #default is 3 for raster, 0 for vector
        TEMPLATE "query.html"
            CLASS
                COLOR 215 252 252
                BACKGROUNDCOLOR 215 252 252 # not sure
about this one
                OUTLINECOLOR 110 110 110
            END #end CLASS
    END #end layer

LAYER
        NAME 'vias_cuenca'
        GROUP 'vias_cuenca'
 CONNECTIONTYPE postgis CONNECTION "host=localhost dbname=cuenca
user=postgres password=postgres port=5432"

        DATA "the_geom from vias_cuenca"
        PROJECTION
            "init=epsg:24877"
        END #end projection
                TYPE line
        STATUS ON
        TOLERANCE 8 #default is 3 for raster, 0 for vector
        #TOLERANCEUNITS meters #default is meters,
[pixels | feet | inches | kilometers | meters | miles | dd]

        TEMPLATE "query.html"
        # These classes are based on the first set of symbols from a
group renderer. Not optimal yet!

                CLASS
        NAME "path2"
        STYLE
            SYMBOL 'circle'
            COLOR 255 0 0
            SIZE 2
        END
END

    END #end layer

LAYER
        NAME 'calles'
        GROUP 'calles'
```

```
 CONNECTIONTYPE postgis CONNECTION "host=localhost dbname=cuenca
user=postgres password=postgres port=5432"

                DATA "the_geom from vias3"
                LABELITEM "nombre_via"
                LABELCACHE ON

                PROJECTION
                        "init=epsg:24877"
                END #end projection
                                TYPE LINE
                CLASS
                        STYLE
                                COLOR 0 0 0
                                OUTLINECOLOR 0 0 0
                        END

                        LABEL
                                COLOR 0 0 0
                                FONT sans
                                TYPE TRUETYPE
                                POSITION CC
                                PARTIALS TRUE
                                SIZE 7
                                BUFFER 1
                                OUTLINECOLOR 255 255 0
                                END
                        END
        END #end layer

LAYER
        NAME "path"
 CONNECTIONTYPE postgis CONNECTION "host=localhost dbname=cuenca
user=postgres password=postgres port=5432"
 CONNECTIONTYPE postgis
DATA "the_geom from (select the_geom from vias_cuenca where gid in
(%codigos%)) as foo using unique the_geom using SRID=32717"
        STATUS ON
        TYPE LINE
        CLASS
                NAME "path"
                STYLE
                        SYMBOL 'circle'
                        COLOR 255 0 0
                        SIZE 8
                END
        END
    END
END
```

Anexo 4. El archivo index.php (original que dibuja 2 rutas)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Consultas Varias</title>
<script src="dbconsultas.js" type="text/javascript"></script>
<script src="mscross.js" type="text/javascript"></script>
<style type="text/css">
<!--
body {
        margin-left: 0px;
        margin-top: 0px;
        background-color: #e6edf5;
}
}
.Estilo1 {
        color: #FFFFFF;
        font-weight: bold;
}
-->
</style>
</head>
<body>
<table>
<tr><td>
<div style="font-size:9px; postion:absolute; width: 1000px; height: 600px;
margin-left:0px, margin-top:0px" id="map_tag"
onMouseMove="myMap1.ShowCoordinates(event)"></div>
</td><td valign="top">

<form id="select_layers" name="select_layers">
                <input onClick="chgLayers()" type="checkbox"
value="manzanas" name="layer[0]" checked />
        Manzanas<br>
        <input onClick="chgLayers()" type="checkbox" value="calles"
name="layer[1]" />
        calles<br>
        <input onClick="chgLayers()" type="checkbox" value="vias_cuenca"
name="layer[2]" checked/>
        vias_cuenca<br>
        <input onClick="chgLayers()" type="checkbox" value="path"
name="layer[3]" />
        path<br>
</form>
<div style="font-size:8px; postion:absolute; width: 200px; height: 300px;
margin-rigtht:0px, margin-top:0px" id="ref_tag"></div>
<div id="coordinates2" style="font-size:10px;"></div>
```

```
            <input type="button" name="Calcular Ruta" value="Nueva
Ruta" onClick="nueva_busqueda();">
            <input type="button" name="Dibujar Ruta" value="Dibujar Ruta"
onClick="dibujar_ruta();">
          <input type="button" name="Cargar Puntos" value="Cargar
Puntos" onClick="cargar_puntos();">
          <input type="button" name="Borrar Puntos" value="Borrar Puntos"
onClick="borrar_puntos();">
</td></tr>
</table>

<script type="text/javascript">
var puntos;
var inicio;
var ginicio, gfin;
var fin;
puntos=0;
   myMap1 = new msMap(
document.getElementById('map_tag'),'standardUp');
   myMap1.setCgi( '/cgi-bin/mapserv.exe' );
   //myMap1.setFullExtent( -79.07, -78.92,-2.94, -2.85 );
myMap1.setFullExtent( 713606.8461, 733223.2157, 9675184.5942,
9686001.686);
   var tool = new msTool('Ruta a Pie', clicca, 'img/button_skull.png', vercor);
   myMap1.getToolbar(0).addMapTool(tool);

function clicca(e, map) { map.getTagMap().style.cursor = "crosshair"; }
   myMap1.setMapFile( '/ms4w/Apache/htdocs/rutapie/cuencaesf.map' );
   myMap1.setLayers( 'manzanas','calles','vias_cuenca','path' );
   myMap1.redraw();
myMap1.debug();
   myMap2 = new msMap( document.getElementById('ref_tag') );
   myMap2.setCgi( '/cgi-bin/mapserv.exe' );
   myMap2.setActionNone();
   //myMap2.setFullExtent( -79.07, -78.92,-2.94, -2.85 );
myMap2.setFullExtent( 713606.8461, 733223.2157, 9675184.5942,
9686001.686);
   myMap2.setMapFile( '/ms4w/Apache/htdocs/rutapie/cuencaesf.map' );
   myMap2.setLayers( 'vias_cuenca' );
   myMap1.setReferenceMap(myMap2);
   myMap2.redraw();
   chgLayers();

function chgLayers()
{
        var list = "SARDINIA ";
          var objForm = document.forms[0];
          for(i=0; i<document.forms[0].length; i++)
          {
            if( objForm.elements["layer[" + i + "]"].checked )
        {
```

```
        list = list + objForm.elements["layer[" + i + "]"].value + " ";
        }
        }
          myMap1.setLayers(list);
          myMap1.redraw();
        }

function nueva_busqueda()
{
          myMap1.removeOverlayPoints();  // Rimuove tutti gli overlay
          puntos=0;
          inicio="";
          fin="";
alert('Ingresar nuevos puntos para cálculo de ruta.');
}

function vercor()
{
var cords;
var datos;
var sql;
cords=myMap1.ShowCoordinates1(event);
lonlat=cords.split("@@");
sql="xml.php?sql=select@nombre,source,target,gid@from@vias_cuenca@wh
ere@distance(vias_cuenca.the_geom,setsrid(makepoint(" + trim(lonlat[0]) +
"," + trim(lonlat[1]) +
"),32717))<10@order@by@distance(vias_cuenca.the_geom,setsrid(makepoin
t(" + trim(lonlat[0])+ "," + trim(lonlat[1]) + "),32717))" + "@limit@1";

datos=leerxml(sql);
tmp= new Array();
tmp=datos.split("@@");
tmp1=tmp[0].split("@");
//saco el source y target para construir la ruta
if (puntos==0)
{
        inicio=trim(tmp1[1]);
        ginicio=trim(tmp1[3]);
}
if (puntos==1)
{
        fin=trim(tmp1[2]);
        gfin=trim(tmp1[3]);
}
addPoint(event,myMap1,lonlat[0],lonlat[1],tmp1[0],trim(tmp1[1]),trim(tmp1[2
]),trim(tmp1[3]));
}

function addPoint(event, map, x, y, calle,source,target,gid)
{
        var datos;
```

```
        sql="xml.php?sql=INSERT@INTO@puntos(x,y,source,target,calle,gid)@V
ALUES@(" + x + "," +  y + "," + source + "," + target + "," + "'" + calle + "'" + "," + gid
+ ")";
        datos=leerxml(sql);
        Point = new pointOverlay( new mslcon('img/inicio.png', '', 13, 39), null,
'Información', x, y,
                new Array('Lat:', 'Lon:', 'Calle','Source','Target'), new Array(x,
y,calle,source,target) );
        map.addPointOverlay(Point);
}

function cargar_puntos()
{
        var datos,map;
        var x, y, calle,source,target;
        map=myMap1;
        sql="xml.php?sql=select@gid,x,y,source,target,calle@from@puntos";
        datos=leerxml(sql);
        datos1=leerxml(sql);
        tmp= datos1.split("@*@");
        for (i=0;i<=tmp.length-2;i++)
        {
                tmp1= tmp[i].split("@");
                gid=trim(tmp1[0]);
                x=trim(tmp1[1]);
                y=trim(tmp1[2]);
                source=trim(tmp1[3]);
                target=trim(tmp1[4]);
                calle=trim(tmp1[5]);
                Point = new pointOverlay( new mslcon('img/inicio.png', '', 13,
39), null, 'Información', x, y,
                    new Array('Lat:', 'Lon:', 'Calle','Source','Target'), new Array(x,
y,calle,source,target) );
                map.addPointOverlay(Point);
        }
}

function borrar_puntos()
{
        var datos;
        sql="xml.php?sql=delete@from@puntos";
        datos=leerxml(sql);
        myMap1.removeOverlayPoints();  // Rimuove tutti gli overlay
        puntos=0;
        inicio="";
        fin="";
        alert('Todos los puntos han sido borrados'); }
function dibujar_ruta()
{
var datos1,tbl1,cod,sql,ban;
cod="";
```

```
puntos=0;
ban=0;
sql="xml.php?sql=select@calle,gid,source,target@from@puntos";
datos1=leerxml(sql);
tmp= datos1.split("@*@");
if (tmp.length != 1 && tmp.length != 1)
{
	for (i=0;i<=tmp.length-2;i++)
	{
		if (puntos==0)
		{
			tmp1= tmp[i].split("@");
			inicio=trim(tmp1[2]);
			ginicio=trim(tmp1[1]);
			puntos=1;
		}
		else if (puntos==1)
		{
			tmp1= tmp[i].split("@");
			fin=trim(tmp1[3]);
			gfin=trim(tmp1[1]);
			puntos=0;
			i=i-1;
			cod = cod + ',' + ginicio + ',' + gfin;

			if (ban==0)
			{
				ban=ban+1;
	sql1="xml.php?sql=SELECT@*@FROM@shortest_path(\'select@gid@as@
id,source::integer,target::integer,length::double@precision@as@cost@from@
vias_cuenca\'," + inicio + "," + fin + ",false,false)";
			}
			else if (ban==1)
			{
				sql1=sql1 + "union
SELECT@*@FROM@shortest_path(\'select@gid@as@id,source::integer,target::i
nteger,length::double@precision@as@cost@from@vias_cuenca\'," + inicio +
"," + fin + ",false,false)";
			}
		}
	}
	datos1=leerxml(sql1);
	tmp= datos1.split("@*@");
	for (i=0;i<=tmp.length-2;i++)
	{
		tmp1= tmp[i].split("@");
		if (tmp1[1]!=-1)
		{
			cod=cod + trim(tmp1[1]) + ",";
	}
	}
```

```
        cod=cod.substring(1,cod.length-1);
                tbl= formatear(datos1);
                var contenido = document.getElementById("datos");
                info= '<table style=" font:Arial, Helvetica, sans-serif; font-
size:9px">';
                info= info + '<tr bgcolor=#478bca
style="color:#ffffff;"><td>VERTEX_ID</td><td>SOURCE_ID</td><td>COST</td>
</tr>';
                info= info + tbl;
                info= info + '</table>';
              contenido.innerHTML=info;
        //cargo la capa para visualizarla
        dibujar_capa(cod);
}
}

function dibujar_capa(codig)
{
         var list = "SARDINIA ";
         var objForm = document.forms[0];
//dibuja las capas
         for(i=0; i<document.forms[0].length-1; i++)
         {
          if( objForm.elements["layer[" + i + "]"].checked )
                    {
                list = list + objForm.elements["layer[" + i + "]"].value + " ";
                }
         }
         // codig = codig + ',' + ginicio + ',' + gfin;
//fecha1 es la variable parametros de los codigos que deben dibujarse
          var fecha1= "codigos=" + codig;
          myMap1.setArgs(fecha1);
          myMap1.setLayers( list + 'path');
       myMap1.redraw();

}

function trim (myString)
{
return myString.replace(/^\s+/g,'').replace(/\s+$/g,'')
}
  </script>
            <div id="datos">DATOS DE ENTRADA</div>
</body>
</h>
```