ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"Prototipo de Sistema de Administración y Planificación Automática de Rutas Óptimas para Expresos Escolares de Instituciones Educativas"

INFORME DE GRADUACIÓN

Previa a la obtención del Título de:

INGENIERO EN COMPUTACIÓN ESPECIALIZACIÓN SISTEMAS TECNOLÓGICOS

Presentada por:

Elizabeth Alexandra Cueva Montero

GUAYAQUIL – ECUADOR

AÑO

2011

AGRADECIMIENTO

A mi familia, por su apoyo y soporte.
A Carmen, mi directora por su invaluable ayuda.
A todas las personas que me ayudaron en la realización de este trabajo.

DEDICATORIA

A mis padres y familiares.
Al lector.

TRIBUNAL DE SUSTENTACIÓN

_____
Ing. Jorge Aragundi R.
SUB-DECANO DE LA FIEC
PRESIDENTE

_____
Msc. Carmen Vaca R.
DIRECTORA DE TESIS

_____
Msc. Vanessa Cedeño M.
VOCAL PRINCIPAL

DECLARACIÓN EXPRESA

"La responsabilidad del contenido de este Proyecto de Grado, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

(Reglamento de exámenes y títulos profesionales de la ESPOL)

Elizabeth Alexandra Cueva Montero

RESUMEN

Este trabajo tiene como objetivo crear una aplicación web prototipo que permita generar rutas de expreso escolar óptimas. Se considerará como rutas óptimas a rutas de menor distancia.

En el capítulo 1 se describe y justifica el problema a resolver, objetivos que se plantean alcanzar y se presenta el alcance del proyecto.

En el capítulo 2 se realiza el análisis de los conceptos teóricos en los cuales se basará el desarrollo del aplicativo. Además, se analiza el problema del camino más corto, el cual está presente en el desarrollo de sistemas de planeación de rutas. Como parte de los fundamentos teóricos, se exponen métodos para la representación de direcciones domiciliarias como puntos geográficos, es decir en unidades de latitud y longitud, esta información geocodificada es primordial para el funcionamiento del aplicativo. Finalmente se realiza el análisis de diferentes algoritmos que permiten obtener rutas con recorridos de menor distancia.

En el capítulo 3 se definen los módulos a implementarse, y se presentan los diagramas de clases, casos de uso y el diseño de la base. El diseño del

sistema incluye las pruebas a realizarse, en este capítulo se detallan las pruebas que se realizarán y en el capítulo 5 aparecerán los resultados y el análisis de las mismas.

En el capítulo 4 se presenta la implementación del diseño especificado en el capítulo 3, se detalla la lógica y funcionamiento de las clases y procesos más importantes que se implementarán en el aplicativo.

En el capítulo 5 se muestras los resultados de las pruebas diseñadas en el capítulo 3, las cuales permitirán determinar el grado de certeza de las rutas generadas y obtener un estimado de los tiempos de respuesta de los algoritmos utilizados.

ÍNDICE GENERAL

INDICE DE FIGURAS

INTRODUCCIÓN

En la actualidad se ha popularizado el uso de aplicaciones web para la ubicación de direcciones domiciliarias; gigantes de la computación como Google y Yahoo proveen API's para manejo de mapas de forma gratuita. Estas API's pueden ser utilizadas para mostrar rutas a seguir para ir de un punto a otro, una funcionalidad muy útil para cierto tipo de aplicaciones que requieren dibujar rutas por ejemplo: rutas escolares, rutas de entrega de comida, rutas entrega de correo, etc.

Existe, sin embargo, una limitación a los API's mencionados anteriormente: la información de generación de rutas no está disponible para muchos sectores geográficos como ocurre con nuestro país. El desarrollo de un prototipo que permita utilizar los mapas provistos por Google en conjunto con una base de datos de direcciones geocodificadas para generar rutas en un sector geográfico de la ciudad de Guayaquil permitiría tener un modelo para explotar el gran potencial de estas aplicaciones. En este documento se propone la creación de un prototipo para un sistema de generación de rutas escolares considerando puntos geográficos ubicados en el centro de la ciudad de Guayaquil.

Para un sistema de generación y manejo de rutas de expreso escolar, la visualización de las rutas es una parte importante de este tipo de aplicación, pero también es fundamental el uso de algoritmos que permitan obtener rutas optimizadas. En la implementación del prototipo propuesto se utiliza una implementación del algoritmo TSP que permite obtener las rutas de menor distancia. El resultado final del proyecto de graduación es una aplicación web que permite generar rutas escolares óptimas, que pueden ser visualizadas utilizando el API's de Google Maps.

Esto será de mucha utilidad para instituciones educativas que prestan el servicio de transporte a sus alumnos, facilitando el manejo y administración de este servicio que normalmente requiere de mucha inversión de tiempo.

CAPÍTULO 1

1. PLANTEAMIENTO DEL PROBLEMA

Introducción

La generación de rutas de transporte es un problema inherente a empresas de diferente índole. Los ejemplos son diversos: distribución de comidas, rutas para estudiantes, recolección de basura, entrega de correos, entre otros. Al generar las rutas se requiere no solamente incluir todos los puntos a visitar, sino también minimizar los costos de las rutas generadas.

Una vez que los puntos a visitar han sido agrupados por ruta resulta útil contar con una herramienta que permita visualizar los resultados. Han aparecido API's como el disponible para Google Maps que pueden ser utilizados para generar esta información geográfica de forma visual. Sin

embargo, en Ecuador no existe información geográfica completa y disponible para estos API's. Una herramienta de generación de rutas que permita visualizar la información en mapas sería de gran interés para las empresas que requieren este tipo de servicio.

En este proyecto se propone la construcción de un prototipo que permita generar rutas escolares para los miembros de una Institución Educativa. Dado que sería muy complejo obtener la información geográfica requerida para todo el país, se ha escogido un área geográfica limitada para la implementación.

1.1 Descripción del Proyecto

La herramienta prototipo a diseñar e implementar automatiza la planificación de rutas de expreso escolar de manera óptima y permita la administración de tareas propias de este tipo de servicio. Se detalla a continuación algunas de las principales características:

- Acceso Vía Web: el prototipo podrá ser accedido a través de un navegador web. Se necesitará de un servidor para alojar él prototipo a desarrollar.
- Aplicación web: que permita visualizar los datos geográficos utilizando funciones del API's de Google Maps.

- Contendrá un repositorio de datos de intersecciones de calles: se creará un repositorio con intersecciones de calles de la siguiente área geográfica de la ciudad de Guayaquil: Malecón Simón Bolívar y Colón - Av. Quito y Colón, al sur; Malecón Simón Bolívar y General Vernaza - Julián Coronel y Av. Quito, al norte.
- Planificación automática de rutas escolares de manera óptima: para la generación de rutas, primero se hará uso de las operaciones de Google Maps para obtener la geocodificación de las intersecciones de calles para luego proceder a la implementación del proceso de generación de rutas óptimas aplicando algoritmos que se explicarán en secciones posteriores para la obtención de rutas de recorridos de menor distancia.
- Mapas de Rutas: visualización gráfica de las rutas asignadas y áreas de servicio, para lo cual se recurrirá de igual manera a servicios provistos por Google Maps.
- Administración y mantenimiento de tareas propias a este tipo de servicio: entre las tareas de administración y mantenimiento tenemos: datos de alumnos, datos de conductores-vehículos, etc.

- Reportes de administración y mantenimiento de tareas: entre los reportes que se muestren tendremos los siguientes: listado de alumnos por rutas, listado de rutas-conductores, listado de direcciones.

1.2 Justificación del Proyecto

Muchas instituciones educativas de primer y segundo nivel de enseñanza de la ciudad de Guayaquil prestan el servicio de transporte a sus alumnos y la planificación de las rutas de expreso que se requiere implementar se vuelve una tarea que demanda mucho tiempo y esfuerzo dado el gran volumen de alumnos, el mismo que aumenta por año. El proceso de planificación de rutas en dichas instituciones se lleva a cabo siguiendo procesos anticuados de planificación manual.

Un sistema que permita la planificación automática de rutas de expreso escolar de forma óptima, sería de mucha utilidad y ayuda para las instituciones educativas que brindan este tipo de servicio, debido a que el tiempo y esfuerzo demandado en la planificación y administración de este servicio disminuiría sustancialmente y a la vez representaría para la institución un incremento en su productividad y en el servicio que otorga a sus alumnos.

Este servicio de transporte en muchas ocasiones también es dado al personal que labora en estas instituciones educativas, lo cual también representaría una disminución en el tiempo y esfuerzo adicional dedicado a la planificación y administración de rutas de transporte para dicho personal.

Como se mencionó anteriormente, no se cuenta con información geográfica completa para nuestro país y de manera específica para la ciudad de Guayaquil. Por lo tanto, se ha optado por el diseño e implementación de un prototipo que permite generar y visualizar rutas escolares óptimas en un área limitada.

1.3 Objetivos

1.3.1 Objetivo General

Diseñar e implementar un prototipo de aplicación web que permita automatizar y optimizar el proceso de planificación de rutas de expreso escolar de instituciones educativas.

1.3.2 Objetivos Específicos

Los objetivos específicos del prototipo a diseñar e implementar son los siguientes:

- Crear un repositorio de información de intersecciones de calles *geocodificadas*, que permitan aplicar algoritmos de cálculo de distancias entre puntos.
- Crear una aplicación web que permita mantener la información y visualizar los datos geográficos utilizando el API's de Google Maps.
- Automatizar y optimizar la asignación de rutas de expreso, aplicando algoritmos para la obtención de rutas de recorridos de menor distancia.
- Implementar reportes varios con la información de las rutas escolares.
- Visualizar los mapas de rutas asignadas y áreas de servicio.

1.4 Alcance del Proyecto

El desarrollo del prototipo contempla la implementación de una herramienta con acceso vía web que implementa cada una de las características detalladas en la descripción del prototipo.

Debido al costo para obtener la información geocodificada de todas las intersecciones de calles de la ciudad de Guayaquil se determinó que para darle viabilidad al proyecto propuesto se trabajaría con un área que abarque calles céntricas de la ciudad. Se eligió el sector céntrico debido a que es el sector de la ciudad de Guayaquil que tiene datos geográficos bien definidos en Google Maps, los cuales son importantes para la generación y visualización de las rutas generadas.

El prototipo que se diseñará e implementará incluirá:

- Aplicativo web para la generación automática y óptima de rutas de transporte escolar, que permitirá también la administración y mantenimiento de tareas propias de este tipo de servicio.
- Un repositorio de intersecciones de calles que cubrirá la siguiente área geográfica, delimitada por las calles: Malecón

Simón Bolívar y Colón - Av. Quito y Colón, al sur; Malecón Simón Bolívar y General Vernaza - Julián Coronel y Av. Quito, al norte.

- La generación de rutas considerará el criterio de menor distancia, sin considerar el sentido de las calles, puesto que estos datos no se tienen disponibles en formato digital.
- Las rutas generadas serán visualizadas usando funciones del API's Google Maps.
- La capacidad de las unidades de transporte serán consideradas como de valor fijo para la generación de las rutas.

Todas las decisiones que se tomaron para definir el alcance fueron necesarias realizarlas, porque sino no hubiera sido factible desarrollar el prototipo.

CAPÍTULO 2

2. FUNDAMENTOS TEÓRICOS

Introducción

En este capítulo se plantea los fundamentos teóricos en los cuales se basa el diseño e implementación del prototipo de rutas óptimas. Se definen conceptos que son utilizados en el desarrollo de este tipo de aplicativos orientados a generación de rutas y mapas de rutas escolares. Para el caso de geocodificación de direcciones domiciliarias y rutas mínimas se consideran dos alternativas, de las cuales se ha escogido la más óptima en costo, tiempo de implementación y tiempo de respuesta del aplicativo.

2.1 Sistemas de Planeación de Rutas

En toda institución cuya ubicación geográfica es relativamente apartada de los lugares donde se concentra la mayor cantidad de población, se vuelve imprescindible en un momento dado contar con unidades de transporte para dar servicio a empleados o a clientes.

Una institución educativa de secundaria es un claro ejemplo en el cual, se vuelve imprescindible manejar información geográfica de los estudiantes y profesores para distribuir a los mismos en las diferentes rutas de transporte escolar planeadas.

El trabajo de distribuir a personal, clientes, estudiantes en una serie de rutas se vuelve complicado cuando el número de los mismos aumenta y no se tiene un sistema que automatice el proceso de diseño de rutas. Así pues una distribución manual puede llevar a resultados poco eficientes con rutas extremadamente largas y poco equilibradas.

Existen múltiples implementaciones de sistemas de planeación de rutas construidos alrededor del mundo, a continuación detallamos algunos sistemas que han sido desarrollados:

- Software de planificación de ruteo de buses escolares, desarrollado por la compañía Versa Trans [1]. Este software provee los siguientes servicios: generación de mapas, mantenimiento de datos del estudiante, ruteo y planificación; y manejo de herramientas de reportes.

- Prototipo de Ruteo y planificación de bus escolar usando GIS (Sistema de Información geográfica), desarrollado como parte de una tesis de maestría para el colegio Sujatha de la ciudad de Hyderabad, India.

  Este prototipo permite la administración de transportación escolar a través del diseño de rutas escolares cortas y rápidas y también permite ubicar paradas de bus, lo cual ayuda a seleccionar las paradas para recoger a los estudiantes y personal de acuerdo a su concentración en las diferentes áreas [2].

- Manejador de operaciones de transportación (TOM), desarrollado por Gecko Microsolutions, Inc. Este sistema presta los siguientes servicios: ruteo usando GIS (Sistema de Información geográfica), visualización de mapas, manejo de

conductas de estudiantes en el bus, manejo de otras actividades de transportación de los alumnos [3].

En la mayoría de los paquetes de software para planificación de rutas escolares se puede encontrar la implementación de algoritmos relacionados a la generación de rutas, por ejemplo, encontrar el camino más corto entre dos puntos, generar una ruta para recorrer varios puntos de forma eficiente, etc. En las siguientes secciones se analizarán algunos de estos conceptos.

2.1.1 Problema del Camino más Corto

Los seres humanos por naturaleza minimizan esfuerzo, sobre todo cuando se trata de moverse. Cuando tienen la oportunidad, siempre tratarán de elegir el camino más corto para ir de un lugar a otro. Este comportamiento puede ser fácilmente observado en los peatones. Cuando sea posible, un peatón camina atravesando un césped, en zigzag por los coches en un estacionamiento, o cruza una calle lateral entre las intersecciones si la ruta seleccionada permite llegar más rápido a un destino.

El transporte, como una actividad económica, reproduce este proceso de minimización, en particular tratando de reducir al mínimo la distancia entre localidades. Estrategias para obtener rutas con tiempos más cortos y costos más bajos son constantemente analizadas por individuos y empresas. Para un individuo, a menudo es sólo una cuestión de conveniencia, pero para una sociedad es de importancia estratégica como un costo monetario directo. En tales circunstancias, no es sorprendente que numerosos métodos hayan sido desarrollados para hacer frente a la compleja cuestión de selección de rutas. Uno de esos clásicos es el problema del vendedor viajero, cuando el camino más corto tiene que ser seleccionado de un conjunto de numerosas combinaciones de posibles trayectorias.

El problema del camino más corto se lo puede clasificar en: el camino más corto desde un origen a un destino y el camino más corto desde un origen a muchos destinos. Para el prototipo propuesto en este documento se analizará e implementará una solución que use el camino más corto desde un origen a muchos destinos.

Los sistemas que implementan soluciones para encontrar el camino más corto entre un origen y varios destinos requieren la aplicación de conceptos que se definen en la siguiente lista de términos:

- Grafo: es un conjunto, no vacío, de objetos llamados nodos (vértices) y una selección de pares de nodos, llamados enlaces (aristas) que pueden ser direccionados o no, los grafos direccionados son llamados dígrafos. Típicamente, un grafo se representa mediante una serie de nodos conectados por enlaces [4].

- Vértices o nodos: En teoría de grafos, un nodo es la unidad fundamental de la que están formados los grafos [4]. Como ejemplos de nodos se nombran los siguientes: intersecciones de calles, poblaciones, ciudades, empalmes eléctricos, fases principales de un proyecto [5].

- Enlaces o aristas: son las uniones entre los nodos. Los enlaces denotan relaciones entre los nodos (vecindad, herencia, orden, etc.) [4]. Como ejemplos de enlaces se nombran los siguientes: segmentos de calles, carreteras

secundarias, tiempo de desplazamiento en aviones, componentes de circuito, tareas de un proyecto.

- Costos: número específico atribuido a cada enlace de un grafo, también llamado valuación o ponderación. Como ejemplos de costos se nombran los siguientes: número de habitantes, % de cierta población por zona, % de uso de electricidad por zona, longitud de una carretera. En la figura 2.1 se muestra un grafo con costos asignados a sus enlaces [5].

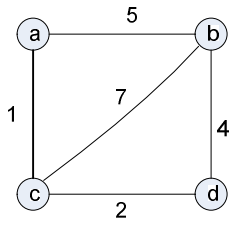


Figura 2.1: Grafo con costos

- Grafos ponderados: un grafo o dígrafo ponderado es un grafo o dígrafo con costos asignados a sus enlaces. Un interés en los grafos es motivado por numerosas aplicaciones de la vida real, como por ejemplo, encontrar la ruta más corta entre dos puntos en una red de

transporte o comunicación o el problema del vendedor viajante [5].

- Ruta: una ruta del nodo A al nodo B del grafo G puede ser definida como una secuencia de nodos adyacentes (conectados por enlaces) que empieza con A y termina con B. Si todos los enlaces de una ruta son distintos, se dice que la ruta es simple [5].

- Longitud de una ruta: es el número total de nodos en una secuencia de nodos que definen la ruta -1, el cual es el número de enlaces en la ruta. Por ejemplo, a, c, b, f es una ruta simple de longitud 3 de a a f como se muestra en el grafo de la figura 2.2, mientras a, c, e, c, b, f es una ruta (no simple) de longitud 5 de a a f [5].

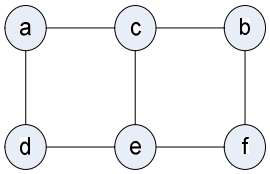


Figura 2.2: Grafo no direccionado

- Ruta dirigida: es una secuencia de nodos en el cual todos los pares de nodos consecutivos están conectados por enlaces dirigidos del nodo origen al nodo destino. Por ejemplo, a, c, e, f es un ruta dirigida de a a f en el grafo de la figura 2.3 [5].

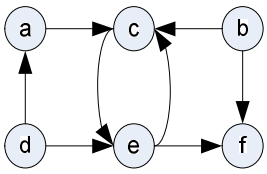


Figura 2.3: Grafo dirigido

- Grafo conectado: si para todos los pares de nodos **u**,**v** hay una ruta de **u** a **v**. Por ejemplo el grafo de la figura 2.2 es conectado, mientras el grafo de la figura 2.4 no es conectado porque no hay ruta, por ejemplo, de a a f.

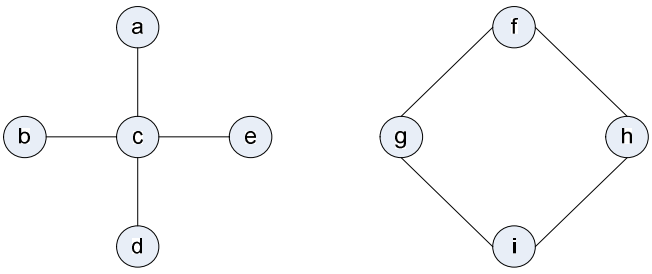


Figura 2.4: Grafo no conectado

- Ciclos: es una simple ruta de longitud positiva que inicia y termina en el mismo nodo. Po ejemplo, f, h, i, g, f es un ciclo en el grafo de la figura 2.4. Un grafo que no tiene ciclos se dice ser a cíclico [5].

- Búsqueda exhaustiva: Muchos problemas importantes requieren encontrar un elemento con una propiedad especial en un dominio que crece exponencialmente. Típicamente, tales problemas surgen en situaciones que implican explícitamente o implícitamente modelos combinatoriales que usan permutaciones o combinaciones de elementos de un conjunto dado. Muchos de tales problemas son problemas de optimización que requieren encontrar un elemento que maximice o minimice alguna deseada característica tales como una longitud de ruta o costo asignado.

  La búsqueda exhaustiva es simplemente un enfoque de fuerza-bruta de los problemas combinatoriales. Lo cual sugiere generar todos los elementos del dominio del problema, seleccionando aquellos que satisfacen las restricciones del problema, y entonces encontrando un

elemento deseado (por ejemplo, el que optimice algunas funciones objetivo). Note que aunque la idea de búsqueda exhaustiva es bastante sencilla, su implementación típicamente requiere mucho tiempo para ser ejecutada.

- Circuito Hamiltoniano: es un ciclo que pasa a través de todos los nodos del grafo exactamente una vez.

En teoría de grafos, el problema del camino más corto es el problema de encontrar un camino entre dos nodos, de tal manera que la suma de los costos de sus enlaces sea mínima; siendo los enlaces el segmento que une los nodos y el costo de cada enlace es el número que representa tal enlace. Un ejemplo podría ser encontrar la forma más rápida de llegar de un lugar a otro, en este caso, los nodos representan los lugares y los enlaces representan los segmentos de la carretera y se ponderan por el tiempo necesario para realizar el viaje de un punto a otro [6].

En un sistema de transporte de rutas, las direcciones domiciliarias constituyen los nodos del grafo y las calles

ubicadas entre una dirección y la siguiente más cercana constituyen los enlaces. Para generar información geográfica que pueda ser representada en un mapa es necesario como primer paso traducir las direcciones domiciliarias a un par latitud, longitud. Este proceso se conoce como Geocodificación de direcciones y se explica en la siguiente sección.

2.2 Geocodificación de Direcciones Domiciliarias

La geocodificación de una dirección domiciliaria consiste en tomar la información dada usando zonas y nombres de calles y representarla como un punto geográfico expresado en unidades de latitud y longitud. A continuación se definen dos métodos que existen para geocodificar direcciones domiciliarias.

2.2.1 Dispositivos GPS

Los dispositivos GPS son aparatos parecidos a un reproductor de mp4, pero cargados con cartografía digital que permite descifrar los datos del satélite y dibujar los mapas, marinos o terrestres. De esta manera en la pantalla dibuja un mapa con el sitio exacto del objeto indicando calles, rutas, ríos y accidentes

geográficos del terreno a todo color [7]. La figura 5 muestra el modelo de un dispositivo GPS.

Figura 2.5: Modelo de dispositivo GPS

Estos dispositivos hacen uso del Sistema de Posicionamiento Global, más conocido con las siglas *GPS. GPS* es un sistema global de navegación por satélite que permite determinar, en cualquier lugar del mundo, la posición de un objeto, persona, vehículo, etc., con una precisión hasta de centímetros, usando GPS diferencial, aunque lo habitual son unos pocos metros.

Para obtener la posición GPS de un punto específico se requiere estar ubicado en el lugar físicamente para de esta manera tener la lectura de los datos geográficos del punto. La lectura de los diferentes puntos de interés, pueden ser guardados en el dispositivo GPS para luego poder ser exportados a una PC y ser utilizados para hacer cálculos mediante un software específico. La figura 2.6 muestra la

aplicación que se le puede dar a datos de posicionamiento geográfico obtenidos a través de un dispositivo GPS.

Figura 2.6: Aplicación que muestra puntos geográficos obtenidos por un GPS.

2.2.2 Uso del API de Google Maps

Google Maps provee un API que permite incluir mapas en una página web utilizando el lenguaje de programación JavaScript. Es posible añadir contenido al mapa a través de una variedad de servicios, lo que permite crear aplicaciones robustas de un sitio web.

Google Maps permite usar el API de forma gratuita, siempre y cuando el sitio web que lo utilice ofrezca servicios sin costo para los consumidores. Google Maps ofrece una serie de

utilidades para la manipulación de mapas (desplazamiento en el mapa, zoom, etc.), mapas del mundo entero, fotos satelitales, la ruta más corta entre diferentes ubicaciones y muchas características interesantes. Google Maps es fácilmente integrable con cualquier sitio web.

Para hacer uso del API es necesario completar el registro, luego de lo cual se obtiene una clave necesaria para utilizar los scripts que mostrarán los mapas en las páginas web. La guía para desarrolladores y las referencias del API están publicadas en el web y pueden ser obtenidas de forma gratuita[1].

En el anexo A se muestran 2 ejemplos del uso del API: cómo incluir un mapa de Google en una página web y cómo insertar marcadores a un mapa.

Para geocodificar una dirección domiciliaria utilizando el API de Google Maps se procede de la siguiente forma:

[1]http://code.google.com/apis/maps/

- Se agrega un mapa en una página web, centrado en el área de interés.
- Se agrega el control que permite desplazarse en el mapa.
- Se agrega código JavaScript para que el usuario pueda hacer clic en el mapa y el API responda a esta acción devolviendo los datos de latitud y longitud del punto donde se hizo clic.
- Los datos de latitud y longitud del punto de interés pueden ser mostrados en un área de texto o directamente en pequeñas pantallas emergentes que ofrece el API.

En el anexo B se muestra el código usado para realizar la geocodificación de un punto geográfico de acuerdo con los pasos señalados en la lista anterior.

Para el desarrollo del prototipo propuesto se utilizará el último método expuesto para geocodificación de direcciones domiciliarias, de ese modo no es necesaria la adquisición de un dispositivo con GPS incluido.

Otro problema fundamental en la generación de rutas es el establecimiento de los nodos a visitar: cuáles serán los nodos intermedios y en qué orden serán visitados. En la sección siguiente se exponen conceptos relacionados al Problema del viajante: un modelo analizado ampliamente para la solución de problemas de generación de rutas.

2.3 Problema del Viajante (Traveling Salesman Problem)

2.3.1 Introducción

El problema del viajante consiste en encontrar una ruta de un número dado de nodos, visitando cada nodo exactamente una vez y retornando al nodo de partida donde la longitud de esta ruta es minimizada.

La primera instancia del problema del viajante fue propuesto por Euler en 1759 cuyo problema fue mover un caballo a todas las posiciones sobre un tablero de ajedrez exactamente una vez.

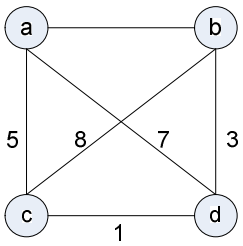
El problema del viajante ganó fama en un libro escrito por un vendedor Alemán en 1832 sobre cómo ser un vendedor viajante

exitoso. En este libro se menciona el TSP, aunque no por ese nombre, y se sugiere que es posible cubrir muchas localizaciones sin visitar cualquier localización dos veces lo cual es el aspecto más importante de la programación de una ruta.

El problema del vendedor viajante puede ser declarado matemáticamente como sigue:

Dado un grafo de costos G = (V, E) donde el costo C_{ij} sobre el enlace entre los nodos i y j es un valor no negativo, encontrar la ruta de todos los nodos que tenga el mínimo costo total.

El problema puede ser convenientemente modelado por un grafo ponderado, con los nodos del grafo representando las ciudades y los costos de los enlaces especificando las distancias. Entonces el problema puede ser declarado como el problema de encontrar el circuito Hamiltoniano más corto del grafo. La figura 2.7 presenta una pequeña instancia del problema y sus soluciones por este método [5].



Tour	Longitud	
a -> b -> c -> d -> a	l = 2 + 8 + 1 + 7 = 18	
a -> b -> d -> c -> a	l = 2 + 3 + 1 + 5 = 11	óptimo
a -> c -> b -> d -> a	l = 5 + 8 + 3 + 7 = 23	
a -> c -> d -> b -> a	l = 5 + 1 + 3 + 2 = 11	óptimo
a -> d -> b -> c -> a	l = 7 + 3 + 8 + 5 = 23	
a -> d -> c -> b -> a	l = 7 + 1 + 8 + 2 = 18	

Figura 2.7: Una solución a una pequeña instancia del problema del viajante por búsqueda exhaustiva.

Actualmente sólo el método que explora todas las combinaciones es el que permite resolver óptimamente el problema del viajante de cualquier tamaño, para resolverlo enumera cada posible ruta y busca por las rutas con costo mínimo. Cada posible ruta es una permutación de 123 … n, donde n es el número de nodos, así por lo tanto el número de rutas es n!. Cuando n aumenta, se vuelve imposible encontrar el costo de todos los nodos en tiempo polinomial. Muchos diferentes métodos de optimización han sido usados para tratar de resolver el TSP [8]. Para el prototipo propuesto se realizarán pruebas utilizando dos métodos de implementación del TSP.

2.3.2 Aplicaciones

El problema del viajante tiene diferentes aplicaciones en el mundo real, haciendo de ello un problema muy interesante de resolver. Por ejemplo, algunas instancias del problema de ruteo de vehículos puede ser modelado como un problema del viajante (bus escolar, atención de llamadas de emergencia, servicio de correo expreso). Para este ejemplo el problema es encontrar cual cliente debe ser visitado por cual vehículo y el número mínimo de vehículos necesitados para visitar a cada cliente. Hay diferentes variaciones de este problema incluyendo encontrar el tiempo mínimo para visitar a todos los clientes.

El problema de cableado de computadoras puede también ser modelado como un TSP. Para este problema se tiene muchos módulos cada uno con un número de pines. Se necesita conectar un subconjunto de esos pines con cables de tal forma que ningún pin tiene más de dos cables conectados a él y la longitud del cable se reduce al mínimo.

La programación de trabajos sobre una máquina dado el tiempo que se requiere para cada trabajo y el tiempo que toma

preparar la máquina para cada trabajo es también TSP. Se trata de minimizar el tiempo total para procesar cada trabajo.

Un robot debe ejecutar muchos diferentes operaciones para completar un proceso. En esta aplicación, opuesto a la programación de trabajos sobre una máquina, se tienen restricciones precedentes. Esto es un ejemplo de un problema que no puede ser modelado por un TSP pero los métodos usados para resolver el TSP podrían ser adaptados para resolver este problema.

Entre otras aplicaciones se tiene por ejemplo: el secuenciamiento de genes, ordenamiento de observaciones en telescopio (NASA), diseño de chips, tour mundial.

2.3.2.1 Métodos para Resolver el TSP

Homaifar [9] afirma: “un enfoque que encontraría la solución de cualquier TSP es la aplicación de enumeración exhaustiva y evaluación. El procedimiento consiste en generar todas las posibles rutas y evaluar las longitudes correspondientes. La ruta con la longitud

más pequeña es seleccionada como la mejor, la cual es garantizada a ser óptima”

El modelo de enumeración exhaustiva es factible para un número pequeño de ciudades, por ejemplo para n=6 existen 720 combinaciones posibles. Sin embargo, dado que el número total de rutas posibles es n!, esta estrategia se vuelve imposible de implementar para un escenario de 25 ciudades, en el cual se requeriría generar $1,55 * 10^{23}$combinaciones. Esto podría tomar millones de años de tiempo de procesamiento considerando que se genera y evalúa una ruta por nanosegundo. Por lo tanto, la solución se vuelve impráctica.

Debido al tiempo que se necesita para procesar todas las rutas, se hace necesario encontrar un algoritmo que nos dé una solución en un período de tiempo más corto. Anteriormente se dijo que el problema del viajante es un NP (no polinomial)-duro, lo cual significa que no hay un algoritmo conocido para resolver este problema en tiempo polinomial. De esta manera probablemente se

tiene que sacrificar optimización en orden a lograr una buena respuesta en un tiempo más corto. Muchos algoritmos han sido tratados para el problema del viajante. Se explorarán unos pocos de estos en esta sección.

Algoritmos Greedy

El algoritmo Greedy da soluciones factibles, sin embargo no son siempre buenas. El enfoque Greedy sugiere construir una solución a través de una secuencia de pasos, cada expansión una solución parcial construida hasta el momento, hasta que una solución completa del problema es alcanzada. En cada paso, y este es el punto central de esta técnica, la elección hecha debe ser:

- Factible, por ejemplo, tiene que satisfacer las restricciones del problema.
- Localmente óptima, por ejemplo, tiene que ser la mejor elección local entre todas las elecciones factibles disponibles sobre ese paso.

- Irrevocable, por ejemplo, una vez hecho, no puede ser cambiado en pasos subsecuentes del algoritmo.

Estos requerimientos explican el nombre de la técnica: en cada paso, se sugiere una selección “greedy”, escoger la mejor alternativa disponible con la confianza que una secuencia de elecciones óptimas locales producirán una solución óptima a todo el problema. Evitando la discusión filosófica de si la codicia es buena o mala, desde la perspectiva algorítmica, la pregunta es si una estrategia greedy funciona o no. En realidad, existen problemas para los cuales una secuencia de elección óptima local produce una solución óptima para todas las instancias del problema en cuestión. Sin embargo, hay otros para los cuales este no es el caso; para este tipo de problemas, un algoritmo greedy puede ser de valor si se busca una solución aproximada.

Existen varios algoritmos Greedy propuestos para resolver TSP, a continuación se explican dos de ellos: el

algoritmo del vecino más próximo y el algoritmo del árbol de expansión mínimo.

El más intuitivo algoritmo greedy para el TSP es basado sobre la heurística del vecino más próximo: iniciando de una ciudad cualquiera, continúa a la más cercana ciudad no visitada y continúa hasta que todas las ciudades hayan sido visitadas, en este punto retorna a la primera ciudad. Hay frecuentemente un alto precio que pagar para realizar selecciones greedy en el inicio del proceso. Un simple ejemplo para ilustrar el caso es dado en la figura 2.8, donde se inicia en la ciudad A, esta heurística greddy construye un tour A – B – C – D – A, con el costo total de 2+3+23+5=33, pero los costos del tour A – C– B – D – A es sólo 4+3+7+5=19.

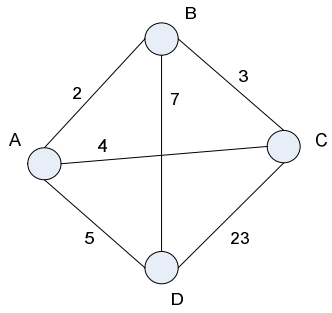


Figura 2.8: un ejemplo TSP con cuatro ciudades. Note que los costos para cada enlace no corresponden a la distancia entre las ciudades dibujadas.

Un árbol de expansión mínimo ([11] y [12]) es un juego de n – 1 enlaces (donde nuevamente n es el número de ciudades) que conecta todas las ciudades así que la suma de todos los enlaces usados es minimizada. En el grafo se puede crear una ruta por el tratamiento de los enlaces en el árbol de expansión como un enlace bidireccional. Entonces se inicia con una ciudad que está solamente conectada a otra ciudad (esto es conocido como una ciudad hoja) y se continúa al siguiente enlace de ciudades no visitadas. Si hay un enlace no utilizado se retorna al enlace previo. Se continúa haciendo esto hasta que se retorna a la ciudad de inicio. Esto permite tener un límite superior para la ruta del viajante. Se puede notar, sin embargo, que se visitarán algunas ciudades más de una vez. Se puede arreglar esto si cuando se necesita visitar de retorno a una ciudad que ya ha sido visitada, en cambio se va a la próximo ciudad no visitada: cuando todas las ciudades han sido visitadas se va directamente de retorno a la ciudad de inicio.

2.4 Algoritmos de Rutas Mínimas

Para encontrar rutas mínimas es necesario calcular el camino más corto desde un nodo de origen a un nodo destino. Existen dos algoritmos importantes para el camino más corto entre dos nodos: Dijkstra y A-star, ambos se describen brevemente en esta sección.

2.4.1 Dijkstra

El algoritmo de Dijkstra, concebido por el holandés Edsger Dijkstra, científico de computación en 1959, es un algoritmo de búsqueda de grafos que resuelve el problema de ruta más corta para un grafo con los costos de enlace no negativo.

Se da un grafo G donde cada enlace es asignado a un costo no negativo y también se tiene un nodo de inicio s y un nodo destino g. Se requiere la ruta de menos costo de s a g.

Procedimiento de la ruta más corta de Dijkstra.

1. Inicializar Q con el conjunto de par simple (s, 0).
2. S $\leftarrow$ 0.
3. **Mientras** Q no es vacío y S no contiene una asignación a g:

4. Remover un par (n, w) de Q con costo w mínimo (sobre todos los elementos de Q).
5. **Si** S no contiene un costo para n, por ejemplo, si S no contiene cualquier par de la forma (n, w'), entonces:
6. Añada (n, w) a S.
7. Para cada enlace (n, m) con costo w' en el grafo G añada el par (m, w + w') a Q.
8. Si S contiene una asignación a g entonces termina con éxito (una ruta ha sido encontrada) sino termina con fracaso (no hay ruta).

Note que si removiendo (n, w) de Q resulta en añadiendo (m, w') entonces porque los costos de enlace son no negativos se tiene que w' ≥ w. Esto implica que los costos removidos de Q monótonamente incrementan. Esto implica que cuando un par (n, w) es añadido a S, w es el costo de la ruta más corta a n.

Luego de haber encontrado la ruta más corta, se puede leer la ruta más corto desde el origen al destino por iteración:

S := secuencia vacía

u := destino

Mientras previous[u] es definido:

inserte u en el inicio de S

u := previous[u]

La secuencia S es la lista de nodos que constituyen una de las rutas más cortas del destino al origen, o una secuencia vacía si no existe camino.

2.4.2 A-star

A * (pronunciado como “A star”) es uno de los mejores algoritmos de búsqueda de grafos que encuentra el camino de menor costo de un nodo inicial a un nodo final (de uno o más objetivos posibles).

El algoritmo fue descrito por primera vez en 1968 por Peter Hart, NilsNilsson, y Bertram Raphael [10]. En su documento, es llamado algoritmo A. Desde que se usó este algoritmo produjo un comportamiento óptimo para una heurística dada, por lo que se lo ha denominado A*.

El siguiente procedimiento difiere de la ruta de menor costo de Dijkstra solamente en la prioridad usada en el paso 4.

Procedimiento A*

1. Inicialice Q a ser el conjunto conteniendo el par simple (s, 0).
2. S $\leftarrow$ 0.
3. **Mientras** Q no es vacío y S no contiene una asignación a g:
4. Remover un par (n, w) de Q minimizando w + h(n) (sobre todos los elementos de Q).
5. **Si** S no contiene un costo para n, por ejemplo, si S no contiene cualquier par de la forma (n, w'), entonces:
6. Añada (n, w) a S.
7. Para cada enlace (n, m) con costo w' en el grafo G añada el par (m, w + w') a Q.
8. Si S contiene una asignación a g entonces termina con éxito (una ruta ha sido encontrada) sino termina con fracaso (no hay ruta).

Si h es monótono entonces la suma w + h(n) de el ítem removido de la cola monótonamente incrementa. Esto implica que cuando se remueve (n, w) de la cola, el costo w es el menor costo posible para n. La monotonicidad sólo da exactitud del algoritmo.

Tomando la distancia a la meta en cuenta A* puede ser bastamente más eficiente que la ruta más corta de Dijkstra.

El algoritmo A* dirige la búsqueda hacia la meta deseada preferentemente que explorando los nodos basado simplemente sobre la distancia del vértice inicial. La idea clave es definir una función heurística que estime cuán lejos un vértice dado n esta del vértice destino. Preferentemente que tratando de minimizar la distancia del vértice inicial, se debe entonces correr el algoritmo de ruta más corta de Dijkstra con la noción de distancia definida como la suma de la distancia real y la función heurística. Eso es, cuando seleccionamos un vértice a quitar de la cola, preferentemente que seleccionando la distancia mínima actual, el algoritmo selecciona el vértice que minimiza esta suma.

Idealmente, una función heurística satisface dos propiedades:

- Una función heurística es admisible si ello nunca sobrestima la distancia al vértice destino. Por ejemplo, cuando la función heurística es aplicada al vértice destino por sí mismo, ello debe retornar cero. La admisibilidad asegura que cuando el vértice destino es primero encontrado por la búsqueda, la ruta usada es una ruta óptima. Para el prototipo a implementar, se utilizará como heurística admisible la distancia Euclideana (distancia en línea recta) entre el vértice dado y el vértice destino. Esta heurística ayudará a dirigir la búsqueda, que es también claramente admisible.
- Una función heurística es monotónica si la combinación de distancia+heurística del vértice inicial nunca decrementa a lo largo de cualquier ruta. Esto es actualmente una propiedad más fuerte que la admisibilidad: si una función es monotónica, ello debe ser también admisible. La mayoría de las funciones admisibles son también monotónicas. Monotonicidad corresponde al

requerimiento de Dijkstra que el grafo contenga enlaces de costo no negativo.

CAPÍTULO 3

3. DISEÑO DEL PROTOTIPO

Introducción

En los capítulos anteriores se describió la importancia de contar con una herramienta que permita generar rutas de transporte de manera automática y óptima y a la vez poder visualizarlas de manera gráfica, así como cuales eran los alcances dentro de este proyecto de tesis. En este capítulo se detallará la estrategia de diseño a aplicar para implementar el aplicativo web para generación de rutas con las características previamente definidas.

3.1 Arquitectura del Sistema

Plataforma de Desarrollo

Las herramientas a ser utilizadas para desarrollar la aplicación web son las siguientes:

- Visual JSF.
- Java Server Pages (JSP).
- Netbeans.
- Base de Datos MySql.
- JavaScript.

Visual JSF:

Entre los diferentes frameworks de desarrollo de aplicaciones web que existen se eligió uno que ofreciera componentes ricos ya desarrollados para de esta manera disminuir el tiempo de desarrollo sin sacrificar la usabilidad. Visual JSF es propuesto por la Sun MicroSystems como un modelo de componentes que cumple con las características señaladas. En la figura 3.1 se muestran ejemplos de algunos de los componentes.

Figura 3.1: Algunos componentes disponibles en Visual JSF.

Visual JSF es un framework de desarrollo basado en el modelo MVC (Modelo Vista Controlador) de libre distribución que simplifica el desarrollo de interfaces del usuario. JSF usa Java Server Pages para el despliegue de las páginas web. Entre las características de JSF tenemos:

- Un conjunto de APIs para representar componentes de una interfaz de usuario y administrar su estado, manejar eventos, validar entrada, definir un esquema de navegación de las páginas y dar soporte para internacionalización y accesibilidad.
- Un conjunto por defecto de componentes para la interfaz de usuario.
- Dos bibliotecas de etiquetas personalizadas para JavaServer Pages que permiten expresar una interfaz JavaServer Faces dentro de una página JSP.

- Un modelo de eventos en el lado del servidor.
- Administración de estados.
- Beans administrados.

Java Server Pages (JSP):

Es una tecnología orientada a crear páginas web dinámicas con programación Java de lado del servidor. El motor de JSP está basado en los servlets de Java - programas en Java destinados a ejecutarse en el servidor.

Netbeans:

Para facilitar el desarrollo del proyecto se utilizará Netbeans como entorno de desarrollo. Netbeans es una herramienta para desarrollar que permite compilar, depurar y ejecutar programas. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso desarrollado por la Sun MicroSystems.

Base de Datos MySql:

Dado que el proyecto a implementar requiere almacenar información variada sobre: intersecciones de calles geocodificadas, rutas generadas, datos de alumno/personal, etc.; fue necesario utilizar un repositorio que almacene los datos de manera confiable y eficiente. MySql cumple con los requerimientos anteriormente mencionados y además no representa un costo para el proyecto porque es un producto de software libre.

JavaScript:

Para implementar las páginas web que mostrarán las rutas generadas en un mapa geográfico, se utilizará el lenguaje de programación Javascript, desarrollado por Netscape, como la herramienta que permita ejecutar funciones del API's de Google Maps del lado del cliente.

Usuarios del Sistema:

A continuación se definen los usuarios con los que el aplicativo web a implementar interactuará:

- Usuario Administrador.- quien tendrá acceso a todas las funciones como: ingreso de personal, generación de rutas, mantenimiento de rutas, reportes y mantenimientos de las diferentes tareas propias de este tipo de servicio que se implementarán en el aplicativo.

- Usuario de consulta.- quien tendrá sólo acceso a la visualización de las rutas que se tengan generadas en el sistema.

En la figura 3.2 se muestra un diagrama con la relación entre cada uno de sus componentes. El usuario administrador trabaja directamente con el aplicativo web, que a su vez almacena la información recibida en la base de datos para luego ser utilizada en la generación de rutas óptimas. El usuario de consulta también trabaja directamente con el aplicativo web, pero sólo tiene acceso a la consulta y visualización de rutas generadas.

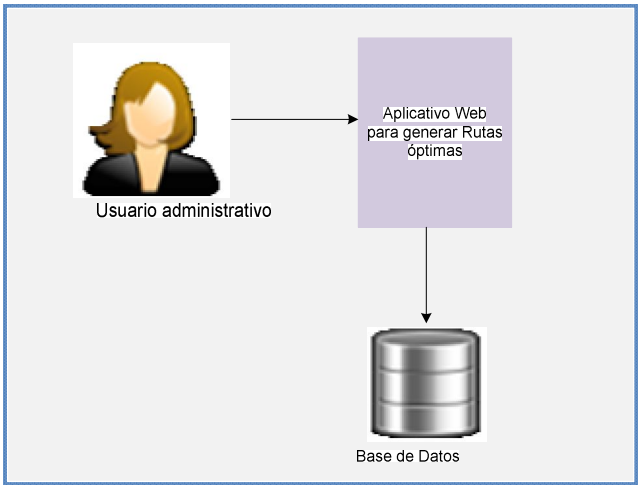


Figura 3.2: Arquitectura del Prototipo.

Para implementar cada una de las funciones del aplicativo web y administrar los datos requeridos de una manera organizada, adecuada y eficiente se ha procedido a crear módulos que a la vez permiten obtener un aplicativo de fácil mantenimiento. Los módulos a ser utilizados en la implementación del aplicativo web son:

- Módulo de geocodificación de direcciones.
- Módulo de ingreso de alumnas.
- Módulo de asignación de direcciones.
- Módulo de generación de rutas.
- Módulo de mantenimiento de rutas.
- Módulo de mantenimiento de datos.
- Módulo de reportes.

A continuación se resume la tarea de cada módulo y en secciones posteriores se presentan los detalles de diseño de cada uno de ellos.

- **Módulo de Geocodificación de Direcciones -** En este módulo se realizará la geocodificación de las intersecciones de las calles del área geográfica indicada como alcance en la implementación del prototipo. Para lograr obtener los datos de las intersecciones de calles en términos de latitud y longitud se utilizará las funciones de geocodificación que nos facilita el API de Google Maps. La obtención de los datos geocodificados de las intersecciones de las calles del área geográfica a considerar se realizará una sola vez, para de esta manera contar con el repositorio de datos geográficos necesarios para la obtención de rutas de transporte escolares óptimos.

- **Módulo de Ingreso de Alumnas** - En este módulo se realizará el mantenimiento de alumnas y personal a quienes se necesite generar rutas de expreso. Aquí se permite el ingreso de información básica, entre las cuales tenemos el ingreso de la información domiciliaria. La asignación de la dirección domiciliaria geocodificada se realizará cuando se ingresa la dirección domiciliaria.

- **Módulo de Generación de Rutas -** Este módulo tendrá las funciones que detallamos a continuación:

    - **Ordenamiento -** Este módulo se encargará de colocar el orden de visita de los puntos geográficos a ser visitados. Este módulo es utilizado tanto para el ordenamiento inicial en la generación de rutas como para el ordenamiento de cada una de las rutas generadas. El primer ordenamiento va a permitir obtener rutas de distancias mínimas es decir rutas balanceadas. El segundo ordenamiento nos permite de igual manera hacer que cada una de las rutas generada sea óptima.

    - **Determinación de Menor Distancia -** Este módulo se encargará de encontrar cuáles son las intersecciones de calles a ser visitados de tal manera que se visite cada uno de los puntos que conforman la ruta en el orden asignado por el módulo de ordenamiento, es decir que encuentra el camino más corto para visitar todos los puntos de la ruta.

    - **Visualización de la Ruta -** Este módulo nos permite visualizar cada una de las rutas generadas. Para lo cual

hace uso de los puntos devueltos por la función que calcula la ruta de menor distancia. Este módulo emplea las funciones del API de Google Maps para mostrar la ruta de manera gráfica.

En la sección de implementación se presenta en detalle las estrategias utilizadas para encontrar rutas eficientes usando las direcciones almacenadas en la base de datos. De acuerdo al alcance del trabajo a realizar se considera como rutas eficientes a las rutas de menor distancia, sin tomar en cuenta el sentido de las calles.

En la figura 3.3 se detallan las funciones a implementar en el módulo de generación de rutas.

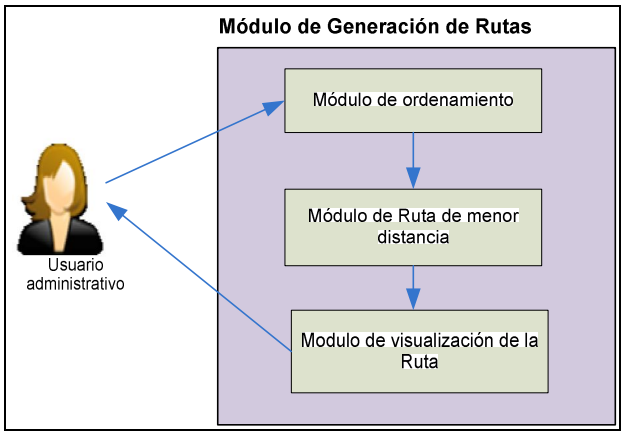


Figura 3.3: Funciones del módulo de generación de Rutas.

- **Módulo de Mantenimiento de Rutas –** En este módulo se podrá realizar cambios en cada una de las rutas de acuerdo con algún requerimiento específico, como por ejemplo cambiar de ruta a una estudiante, lo cual modifica la ruta obtenida en la generación automática.

- **Módulo de Mantenimiento de Datos** – En este módulo se realizará el mantenimiento de diferentes tareas propias de este tipo de servicio, a continuación detallamos cuales son:

    - Mantenimiento Propietario – Bus
    - Mantenimiento de Unidades de Expreso
    - Mantenimiento de Conductores.

  Detallamos a continuación cada uno de estos mantenimientos:

    - **Mantenimiento Propietario** – **Bus** - En este módulo se podrá hacer el mantenimiento de los datos de propietario de cada una de las unidades de expreso.

    - **Mantenimiento de Unidades de Expreso** - En este módulo se podrá hacer el mantenimiento de los datos de

cada una de las unidades de expreso que se utilizarán para los recorridos de las rutas de transporte escolar.

- **Mantenimiento de Conductores** - En este módulo se podrá hacer el ingreso y mantenimiento de los datos de cada uno de los choferes de las unidades de transporte.

- **Módulo de Reportes** – Este módulo entregará al usuario información acerca de: listado de direcciones, listado de rutas – choferes y listado de alumnos/personal por rutas. Esta información será de mucha utilidad para el usuario ya que le permitirá tener la información de manera condensada y poder tenerla impresa para ser consultada en cualquier momento.

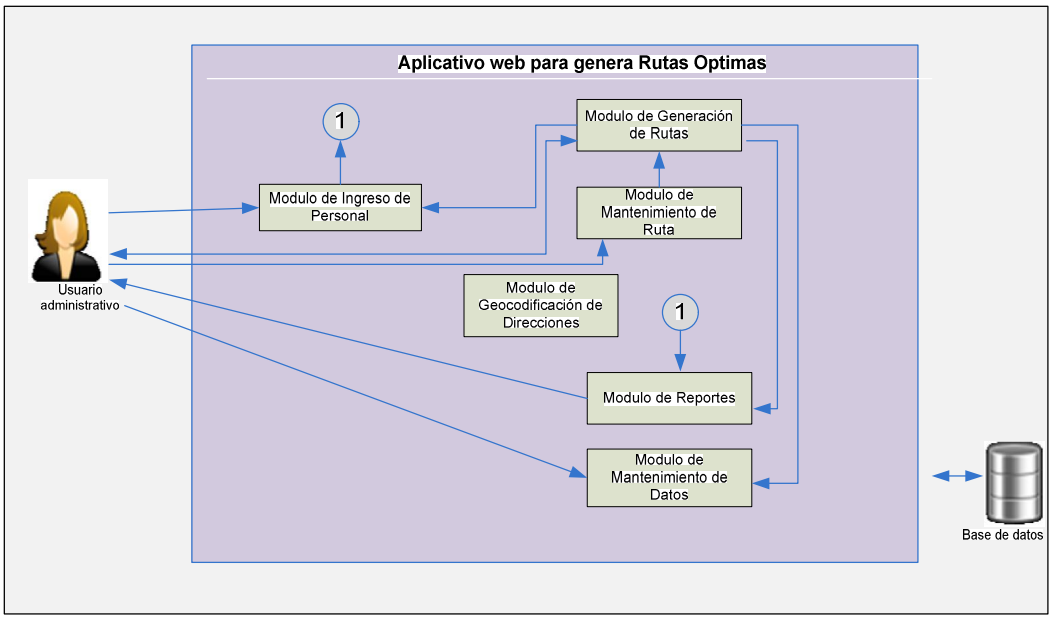


Figura 3.4: Detalle de la arquitectura del prototipo.

3.2 Diagrama de Clases

En esta sección se detallarán las clases más relevantes del aplicativo web a implementar, estas clases pertenecen a las funciones del módulo de generación de rutas. Estas funciones son: el módulo de ordenamiento y módulo de visualización de la ruta, los cuales fueron descritos en secciones anteriores.

En el módulo de ordenamiento se hace uso de la implementación del algoritmo TSP y en el módulo de visualización de la ruta se hace uso de la implementación del algoritmo A-star. A continuación se especifica cada una de las clases a implementar.

3.2.1 Diagrama de Clases de la Función de Ordenamiento.

La función de ordenamiento hará uso del algoritmo TSP descrito en la sección de fundamentos teóricos, el método utilizado en la implementación es la técnica Greedy con heurística del vecino más próximo. A continuación se describe la clase que participa en esta función:

- **Clase Graph:** esta clase permitirá obtener el orden de visita de cada uno de los puntos que conforman una ruta aplicando la técnica Greedy con la heurística ya señalada. La clase implementa métodos que permiten asignar y obtener el costo de todos los enlaces posibles con uno de los puntos de la ruta a ordenar.

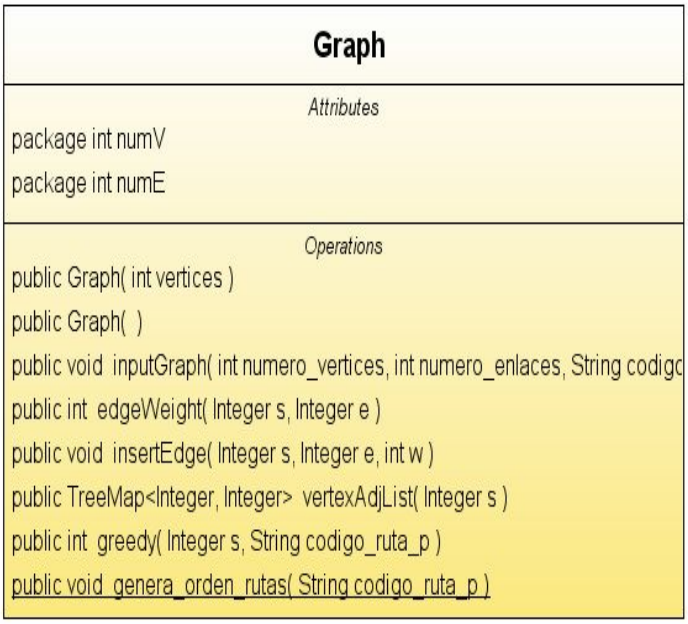


Figura 3.5: Clase de la función de ordenamiento.

3.2.2 Diagrama de Clases de Función de Determinación de Menor Distancia.

En el proceso de generación de rutas de transportes escolares existen dos tipos de puntos: aquellos que representan la dirección de una alumna y aquellos que representan las intersecciones que se atravesarán para ir de cada una a otra. El módulo de generación de rutas determina para cada ruta los puntos que representan direcciones de alumnas y el orden en el cual serán visitados.

La función de determinación de menor distancia debe determinar cuáles serán las intersecciones a utilizar en cada una de las rutas para ir de un domicilio a otro. Para este fin se correrá el algoritmo A-Star entre puntos consecutivos de cada una de las rutas a visualizar.

Las intersecciones que se determinen para visitar cada uno de los puntos de cada una de las rutas serán almacenadas para ser utilizadas por la función de visualización de la ruta. A continuación de describen las clases más relevantes de la función de determinación de menor distancia.

- **Clase AStarAlgorithm**

  Esta clase implementa el método para encontrar el camino más corte entre dos puntos. Entre los parámetros que recibe este método están el nodo inicial y el nodo final.

  El proceso empieza con el nodo inicial y busca los nodos vecinos de este nodo, para seleccionar el siguiente nodo. En la construcción del camino más corto se selecciona siempre aquel nodo que tiene el costo más bajo. Para el

costo se considera tanto la distancia en línea recta del punto inicial al nodo como del nodo a la meta. Este proceso de encontrar el camino más corto continúa de un punto a otro hasta llegar al nodo final.

- **Clase Mapa**

  Esta clase implementa métodos para la obtención del costo de un punto a otro y la selección de los vecinos de un determinado nodo.

- **Clase HeuristicNode**

  Esta clase hereda de la clase Node. Implementa métodos de comparación, estos métodos de comparación son utilizados para el ordenamiento de elementos de este tipo que son considerados en estructuras que requieren ordenamiento y que son utilizados por la clase AStarAlgorithm.

- **Clase Node**

  Esta clase maneja atributos de un nodo tales como: el costo, coordenadas, también implementa métodos de comparación para ordenamiento.

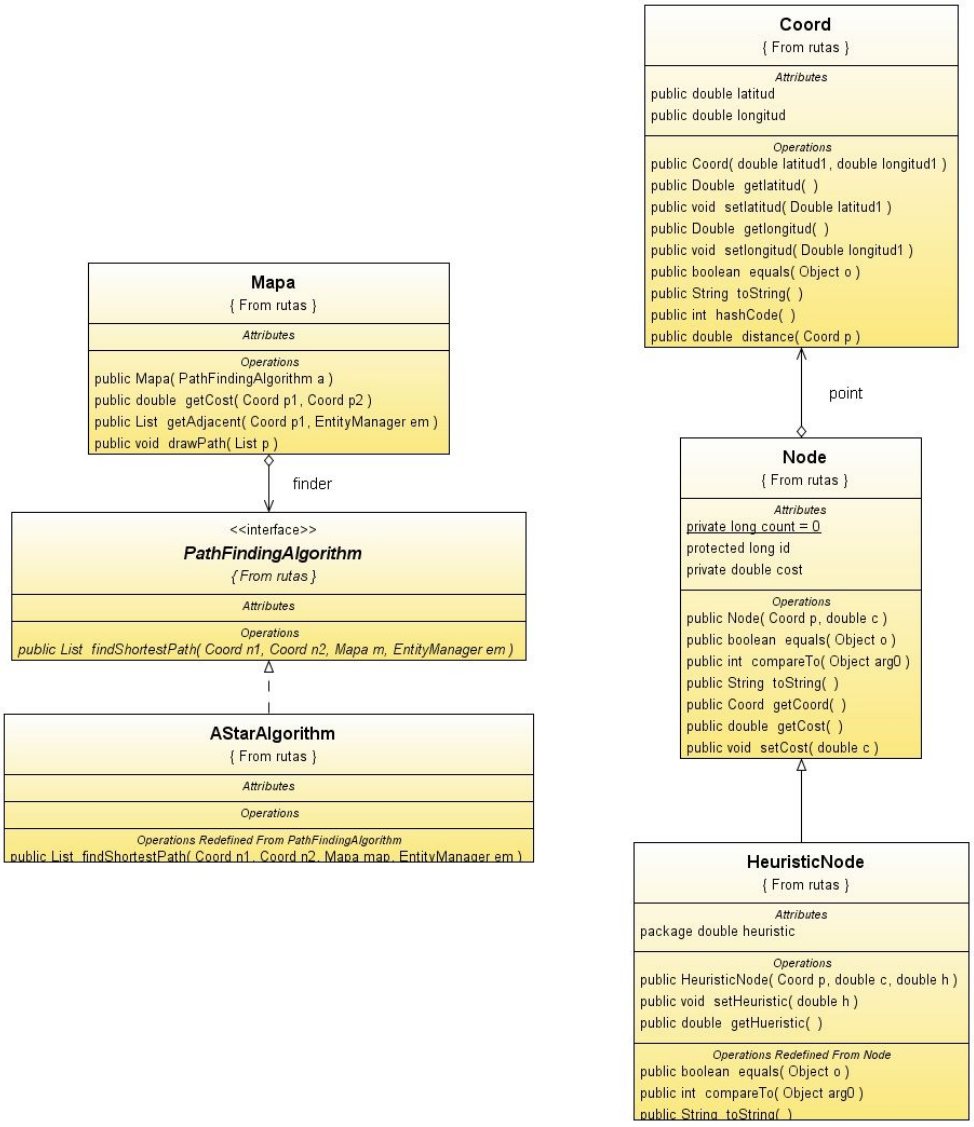


Figura 3.6: Diagrama de Clases del módulo de visualización de la ruta.

3.3 Casos de Uso

3.3.1 Asignación de Dirección Geocodificada

Esta opción permite al usuario del aplicativo poder asignar la dirección domiciliaria geocodificada de la persona que está ingresando en el módulo de ingreso de alumna.

La asignación de dirección se realiza cuando el usuario necesita ingresar o hacer mantenimiento de los datos de una persona, para lo cual el usuario debe ingresar en la página de ingreso de personal, ingresar o cambiar los datos básico de la persona, asignar la dirección domiciliaria usando el mapa geográfico que se muestra para el efecto.

3.3.2 Mantenimiento de Datos

El usuario tendrá varias interfaces para el mantenimiento de datos para rutas, unidades de expreso, propietarios de buses, etc.

3.3.3 Generación de Rutas

Esta opción nos permitirá generar rutas de transporte para todas aquellas personas que han sido ingresadas al aplicativo y requieran el servicio de transporte. El usuario accederá a una página en la cual se mostrará el número de personas a quienes se les generará ruta de transporte. Finalmente el usuario generará las rutas presionando el botón etiquetado como generar rutas.

3.3.4 Visualización de Rutas

Esta opción permitirá visualizar cada una de las rutas generadas de manera gráfica. Para lo cual el usuario accede a una página que le permite seleccionar que rutas desea visualizar. Al seleccionar una ruta se muestra en esta página las personas que pertenecen a esta ruta. Se procede a mostrar en un mapa la ruta seleccionada una vez que el usuario presiona el botón etiquetado como mostrar ruta.

3.3.5 Reportes varios

Esta opción permitirá al usuario obtener reportes de mantenimiento de tareas propias de este tipo de servicio, tales como: reporte de direcciones, reporte de personal por rutas, reporte de rutas-conductores. El usuario podrá acceder a cada uno de los reportes indicados en los respectivos módulos.

3.4 Diseño de la Base de Datos

La estructura utilizada en la base de datos se diseñó de tal manera que los datos geocodificados y de rutas sean almacenados de manera ordenada.

Las tablas que se presentan en la base de datos representan a algún ente dentro del contexto de una clase como una alumna, una ruta y las relaciones entre estos. A continuación se listan las tablas y se explica qué representan dentro de dicho contexto:

Tabla Alumna – tabla que contendrá datos de alumnas a usar el servicio de transporte, manejará campos que se relacionan con la generación de rutas.

Tabla Direccion – tabla que contendrá los nombres de las intersecciones de calles identificados con un id. Esta tabla se relaciona con la tabla Interseccion.

Tabla Interseccion – tabla que contendrá los datos de latitud y longitud de cada una de las intersecciones de calle del área definida como alcance del trabajo a realizar. Esta tabla se relacionada con la tabla Direccion.

Tabla Interseccion_ruta– tabla que contendrá las intersecciones de distancias mínimas de una determinada ruta. Esta tabla contiene un campo que guarda el orden de visita de las intersecciones.

Tabla intersección_vecino – tabla que contiene las intersecciones de calles vecinas de cada uno de las intersecciones de calles del área predefinida como alcance, esta tabla se relaciona con la tabla Interseccion.

Tabla Ruta – tabla que guarda la cabecera de datos de las rutas que se generen, esta tabla se relaciona con la tabla de detalle de rutas.

Tabla Ruta_det – tabla que guarda el detalle para cada una de las rutas que se generen. En esta tabla se especifica las personas que intervienen en cada ruta generada, así como el campo que permite identificar el orden de visita de las personas que conforman cada una de las rutas.

Tabla Conductor– contiene datos del conductor para el manejo de tareas de administración, esta tabla se relaciona con las tablas: Ruta, Ruta_temp y expreso.

Tabla Usuario – contiene información de los usuarios que pueden acceder al aplicativo, contendrá un campo que identifique el tipo de usuario.

Tabla Parámetros – define valores constantes que se manejan en el aplicativo, como por ejemplo los datos geográficos del punto de llegada y partida que, para nuestro caso, es la dirección de la institución educativa.

Tabla Propietario_expreso – tabla que contiene datos de los propietarios de expreso. Esta tabla se relaciona con las siguientes tablas: Expreso, Pais, Provincia y Ciudad.

Tabla Expreso – Tabla que contiene datos de los diferentes expresos con los que cuenta la institución para la generación de rutas.

Tabla Pais – Tabla que contiene la codificación de los países que el usuario puede elegir al momento de ingreso/mantenimiento de Propietarios de expreso

Tabla Ciudad –Tabla que contiene la codificación de las ciudades que el usuario puede elegir al momento de ingreso/mantenimiento de Propietarios de expreso.

Tabla Provincia - Tabla que contiene la codificación de los países que el usuario puede elegir al momento de ingreso/mantenimiento de Propietarios de expreso ingresado al aplicativo.

En la figura 3.7 se presenta cómo se relacionan las tablas para permitir el manejo adecuado de los datos que intervienen en el aplicativo de generación de rutas óptimas. La tabla alumna se relaciona con la dirección domiciliaria de la persona y con el detalle de la ruta a la cual la persona pertenece.

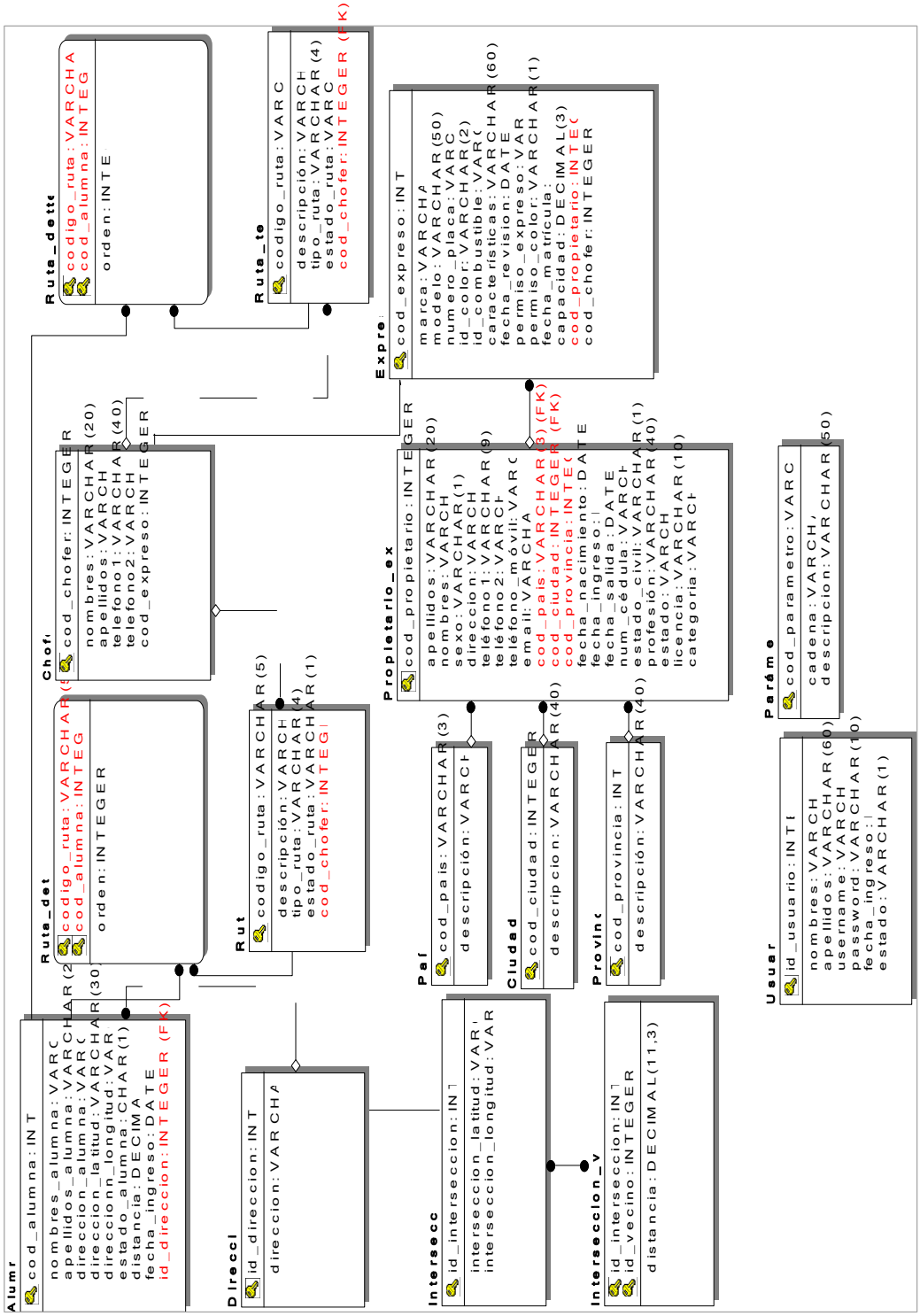


Figura 3.7: Modelo de la base de datos

El modelo descrito en los párrafos anteriores es utilizado por los módulos del sistema y permite implementar todas las características detalladas en el capítulo de planteamiento del problema. A continuación se definen las pruebas a realizarse que nos permitirá medir el tiempo de respuesta de la generación de rutas de transporte, así como el grado de certeza de la asignación de las personas a cada una de las rutas.

3.5 Diseño de Pruebas

3.5.1 Asignación Correcta de Nodos a Rutas

Para verificar el grado de certeza de la generación de rutas se realizará una prueba llevando a cabo los siguientes pasos:

1.) Se generan las rutas, se visualizan

2) Luego se agrega un nodo nuevo cuya ubicación sea obvia en una de las rutas obtenidas.

3) Se generan rutas nuevamente para verificar que el nodo haya sido asignado de forma correcta.

3.5.2 Medición de Tiempo Usando TSP

Es importante determinar el orden de tiempo requerido para generar rutas según el número de nodos a utilizar. Para esta prueba se correrá el algoritmo de generación de rutas usando la técnica Greedy en 3 escenarios que difieren solamente en el número de nodos. Al final de cada corrida se determinará el tiempo empleado en la selección y ordenamiento de nodos.

3.5.3. Medición de Tiempo Usando A-Star

Se medirá el tiempo para determinar las intersecciones que se deben visitar para ir de un domicilio a otro dentro de una ruta.

CAPÍTULO 4

4. IMPLEMENTACIÓN DEL PROTOTIPO

Introducción

En los dos primeros capítulos se mostró el planteamiento del problema a resolver, así como los fundamentos teóricos en los cuales apoyarse para poder diseñar e implementar el problema propuesto de la manera más adecuada. En este capítulo se explican detalles de implementación del sistema.

4.1 Generación Automática de Intersecciones

Para implementar el sistema de generación de rutas era imprescindible contar con los datos de latitud y longitud de cada una

de las intersecciones de calles del área predefinida como alcance del prototipo. Para generar esta información se recurrió a las funciones del API's de Google Maps según lo especificado en el capítulo 2.

La tarea de obtener la información de intersecciones de calles se realiza una sola vez y el aplicativo debe contar con esta información antes de su utilización. A continuación se detalla el proceso que se siguió para obtener los datos geocodificados:

- El usuario accede a una página web la cual mostrará un mapa de Google Maps centrado en el área definida como alcance del prototipo implementado.
- El usuario debe seleccionar la intersección con la cual desea iniciar la tarea para obtener los datos geocodificados.
- Se tiene una opción que permite ir avanzando a la siguiente intersección en línea recta. Existe la opción que permite indicar la distancia que se desea avanzar para alcanzar la siguiente intersección.
- En cualquier momento se puede parar y seleccionar la opción de obtener los datos geocodificados de todos los puntos marcados en el mapa.

- Existe la opción que permite corregir el punto siguiente que se marca como intersección si este se está marcando de manera desviada de su posición correcta.
- Existe la opción que permite avanzar en diferentes sentidos, los cuales son: este, oeste, norte y sur.

Los datos geográficos obtenidos para todos los puntos del mapa del área predefinida son almacenados luego en la base de datos.

En la figura 4.1, se muestra la interfaz para la generación automática de intersecciones.

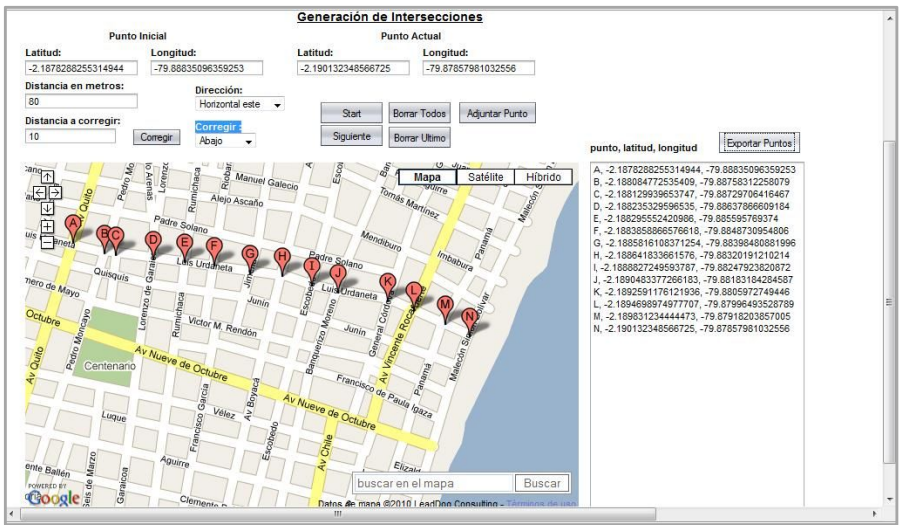


Figura 4.1: Interfaz gráfica del módulo de generación automática de intersecciones de calles.

4.2 Algoritmo para Determinar Orden de Visita de los Nodos

De acuerdo con lo especificado en el capítulo de diseño, TSP será el algoritmo a utilizarse para determinar en qué orden se visitará cada uno de los nodos que conforman las rutas. La técnica que el algoritmo TSP utiliza es la técnica Greedy con la heurística del vecino más próximo.

La implementación del algoritmo utiliza una estructura de datos adecuada para almacenar los enlaces entre los diversos puntos y sus respectivos costos.

El Costo de un punto a otro está dado por la distancia que existe entre ellos en línea recta. Los nodos que son considerados a ordenar son todos los nodos que pertenecen al grupo que se especifica como parámetro del método que implementa el ordenamiento.

El proceso de ordenamiento empieza en el punto de llegada/partida que en nuestro caso sería la dirección geográfica de la institución educativa. A partir de ahí se busca el siguiente punto a visitar seleccionando aquel que represente el de menor costo. Este proceso descrito se continúa hasta que se evalúen todos los nodos del grupo.

El algoritmo de ordenamiento es aplicado dos veces, la primera vez que se aplica es para el ordenamiento general de todos los nodos a ser considerados en la generación de rutas. La segunda vez que se aplica el algoritmo es cuando se tienen los nodos ya agrupados por ruta.

Utilizar el proceso de ordenamiento en dos etapas fue una decisión que se tomó durante la implementación del prototipo al observar que, utilizando el algoritmo de ordenación una sola vez, los nodos eran agrupados correctamente pero luego el orden a seguir dentro de cada grupo no era el óptimo.

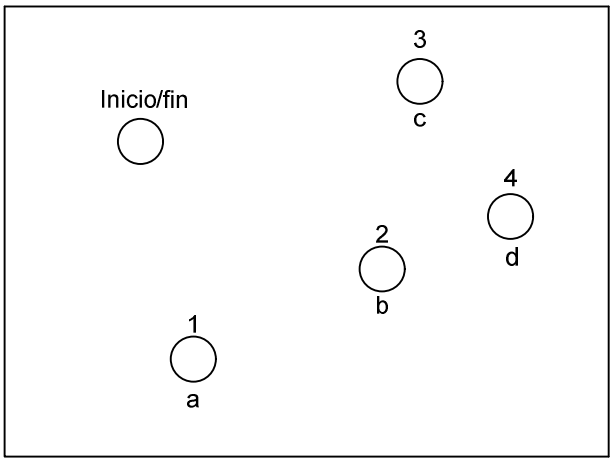


Figura 4.2: Ordenamiento de los nodos usando TSP.

4.3 Determinar Distancias Mínimas entre Nodos

De acuerdo con lo indicado en el diseño, para el módulo de visualización de rutas generadas se hace uso de la implementación del algoritmo A Star que permite obtener la distancia mínima entre dos nodos. En el diagrama de clases mostrado en el capítulo de diseño se muestra la clase AStarAlgorithm la cual implementa el método de distancias mínimas. Para determinar la distancia mínima entre dos puntos:

- Se visitan todos los nodos vecinos de un punto y se determina aquel cuyo costo sea menor.
- Se agrega este punto a la ruta y se verifica si se ha llegado a la meta.
- Si aún no se llega a la meta se continúa con el nodo siguiente.
- En el algoritmo se utilizan estructuras de datos adecuadas para el manejo de los nodos y sus respectivos costos. Las siguientes son las estructuras que se crean:

    - PriorityQueue queue – cola de prioridad para los costos.
    - HashMap nodes - estructura de tipo clave, valor para almacenar cada uno de los nodos a evaluar y determinar el de menor costo.

- HashMap previous – estructura de tipo clave, valor para almacenar el nodo previo de un nodo en el proceso de determinar la distancia mínima de un punto a otro.

- Para obtener los nodos que conforman la distancia o ruta mínima del punto inicio al punto meta se crea una lista en el que se añadirá como primer elemento el nodo meta y de reverso obtendremos el nodo previo empezando en el nodo meta, estos nodos que se obtiene se almacenan en la lista creada que contendrá la ruta más corta de un punto a otro.

En la figura 4.3, se ilustra cómo se determina la distancia o ruta mínima entre dos puntos.

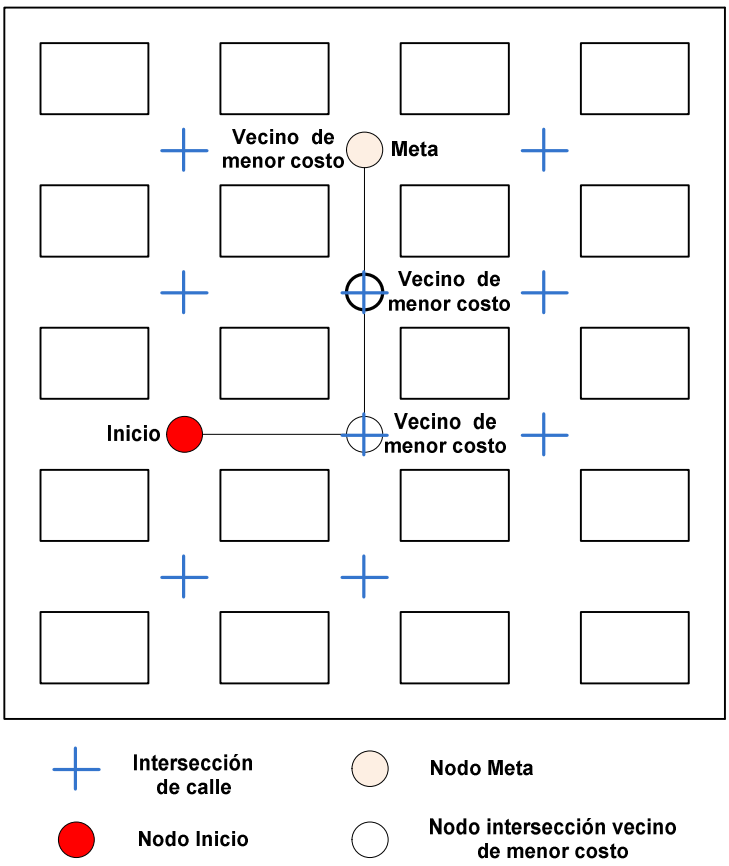


Figura 4.3: Distancia mínima de un punto a otro.

Lo anteriormente descrito corresponde a determinar la distancia mínima de un punto a otro. Para poder determinar la distancia mínima de una determinada ruta, se procede a aplicar la implementación de distancia mínima de un punto a otro varia veces hasta que se consideren todos los puntos que conforman la ruta, el orden en que van tomando los puntos es de acuerdo con el ordenamiento obtenido a través de la implementación del algoritmo de TSP en sentido descendente.

4.4 Visualización de la Ruta Generada en Google Maps

A través del módulo de generación de rutas se obtienen las diferentes rutas requeridas, en este módulo primeramente se ordenan todos los puntos que serán considerados en la generación para luego proceder a generar grupos de acuerdo con la capacidad de bus que ha sido definido como un valor constante. Una vez que se obtienen estos grupos, se ordenan nuevamente para que el orden corresponda a los puntos de esa ruta. Luego de esto se procede a obtener las distancias mínimas para ir de un punto a otro dentro de una ruta. Las intersecciones que conforman las distancias mínimas de un punto a otro son almacenadas en la base. Al almacenar esta información se registra un campo que indica el orden de visita de esa intersección.

Al acceder a la interfaz de consulta de rutas se leerá de la base todas las intersecciones que conforman la distancia mínima de una ruta las cuales han sido previamente almacenadas en la base, al leerlos se los mantiene ordenados por el orden registrado en un campo de la tabla.

Una vez que los puntos se han cargado en una colección en memoria, se recorre la misma para dibujar fragmentos de línea que en su conjunto representarán la ruta.

A continuación se muestra un fragmento de código que permite dibujar una línea de un punto a otro.

```
map = new GMap2(document.getElementById("map_canvas"));
var points = [];
var point1 = new GLatLng(punto1_latitud, punto1_longitud);
points.push(point1);
var point2 = new GLatLng(punto2_latitud, punto2_longitud);
points.push(point2);
map.addOverlay(new GPolyline(points));
```

En la figura 4.4 se muestra la visualización de una ruta de transporte escolar.

Figura 4.4: Visualización de ruta generada usando funciones del API's de Google Maps.

4.5 Mantenimiento de Datos

4.5.1 Mantenimiento y Administración de Datos.

En esta sección se detalla los diferentes módulos de mantenimiento que se implementaron:

- Módulo de ingreso de alumnas.
- Módulo de mantenimiento de rutas.

- Módulo de mantenimiento propietario-bus.
- Módulo de mantenimiento de unidades de expreso.
- Módulo de mantenimiento de conductores.

- **Módulo de Ingreso de Alumnas**

  Este módulo como se especificó en el diseño permite el mantenimiento de los datos de las alumnas que requieren se les genere rutas de transporte.

  El usuario podrá acceder a una página que permitirá realizar el ingreso de datos básicos de la alumna.

  Para el ingreso de la dirección domiciliaria de la alumna el usuario debe hacer clic en el mapa que se muestra en la página para señalar la dirección. La información que se guarda como dirección domiciliaria de la alumna es la dirección geocodificada del punto seleccionado. Además en el momento de escoger el punto geocodificado el sistema utiliza un algoritmo para determinar la intersección más cercana y la guarda junto con la dirección. El id de la intersección es obtenido por la función asignación de dirección.

En la figura 4.5 se muestra la interfaz de ingreso de alumna, aquí se puede observar el punto señalado como dirección domiciliaria de la alumna, su respectiva información geográfica y la dirección domiciliaria que se asigna a la alumna, los cuales se han señalado en la figura como puntos 1 y 2.

Figura 4.5: Interfaz del módulo de Ingreso de Personal

- **Módulo de Mantenimiento de Rutas**

  En este módulo el usuario tiene la posibilidad de visualizar cada una de las rutas que se hayan generado y realizar cambios manuales a una ruta específica.

  Los cambios que el usuario puede realizar son:

- Mover una alumna de una ruta a otra.
- Cambiar el orden de visita de los puntos para una determinada ruta.

Mover una Alumna de una Ruta a Otra: cuando el usuario mueve a una determinada alumna de una ruta a otra, de manera automática se realiza el reordenamiento considerando a la nueva alumna colocada en la ruta.

Cambiar el Orden de Visita de los Puntos para una Determinada Ruta: en esta opción el usuario puede colocar de acuerdo a algún requerimiento específico el orden en el cual requiere que las alumnas sean recogidas. Si una determinada ruta es modificada, la ruta original es almacenada en tablas temporales para tener la posibilidad de poder visualizar la ruta óptima que el sistema generó y que fue modificada. La opción de visualizar ruta original se mostrará activa si la ruta ha tenido modificaciones.

Figura 4.6: Interfaz del módulo de mantenimiento de rutas.

- **Módulo de Ingreso/Mantenimiento Propietario-Bus**

  En este módulo se realiza el mantenimiento de los datos de propietarios-bus, estos datos son importantes para definir el propietario de cada una de las unidades con las que se cuenta en la institución.

- **Módulo de Ingreso/Mantenimiento de Unidades de Expreso**

  Este módulo permite el mantenimiento de datos de las unidades de expreso escolar, a la vez que permite relacionar cada unidad en estado activo a un propietario de bus.

- **Módulo de Ingreso/Mantenimiento de Conductores**

  Este módulo permite el mantenimiento de los datos de conductores, cada uno de los conductores se relaciona con una unidad de expreso, las unidades de expreso están relacionadas como ya se mencionó con su respectivo propietario. El código de conductor es requerido para ser almacenado en la ruta a la cual es asignado.

4.5.2 Reportes de Administración y Mantenimiento de Datos.

En esta sección se detalla los diferentes reportes de administración y mantenimientos de datos implementados. Los reportes que se desarrollaron fueron:

- Reporte de direcciones.
- Reporte de rutas – conductores.
- Reporte de alumnas por ruta.

Se detalla a continuación cada uno de los reportes desarrollados:

- **Reporte de Direcciones**.- lista las direcciones domiciliarias de cada una de las alumnas ingresadas en el sistema y que tengan estado activo.
- **Reporte de Rutas – Conductores.-** lista todas las rutas que se han generado con el respectivo conductor asignado a cada una de las rutas.
- **Reporte de Alumnas por Ruta.-** lista los alumnos asignados a cada una de las rutas que se han generado.

CAPÍTULO 5

5. PRUEBAS Y ANÁLISIS DE RESULTADOS

Introducción

En capítulos anteriores se definió de manera progresiva el desarrollo del prototipo para la generación de rutas óptimas. En este capítulo se describe y analiza las diferentes pruebas que se realizaron, las cuales permiten verificar el grado de certeza y efectividad del aplicativo. A continuación se describe cada una de las pruebas realizadas.

5.1 Asignación Correcta de Nodos a Rutas Generadas

Como se describe en la sección 3.5.1, los pasos para llevar a cabo esta prueba son los siguientes:

a) Generar y visualizar rutas.

b) Añadir punto.

c) Volver a generar y visualizar rutas.

Para la implementación de esta prueba se ha considerado un set de datos de 22 puntos geográficos que corresponden a las direcciones domiciliarias de las alumnas, las cuales a su vez se relacionan con los datos geográficos de la intersección de calle más próxima a la dirección domiciliaria de la alumna. Estos puntos son mostrados en la figura 5.1.

Para formar este set de datos se escogieron puntos geográficos distribuidos en el área definida en el alcance.

La prueba comienza con la generación y visualización de las rutas usando el set de datos propuesto. Una vez completada la generación, se observa que se han generado 3 diferentes rutas. La opción de visualización grafica las rutas generadas, como se muestra en la figura 5.2, 5.3 y 5.4.

Hay que notar que los puntos han sido agrupados correctamente puesto que todos los puntos más cercanos han sido asignados a una misma ruta.

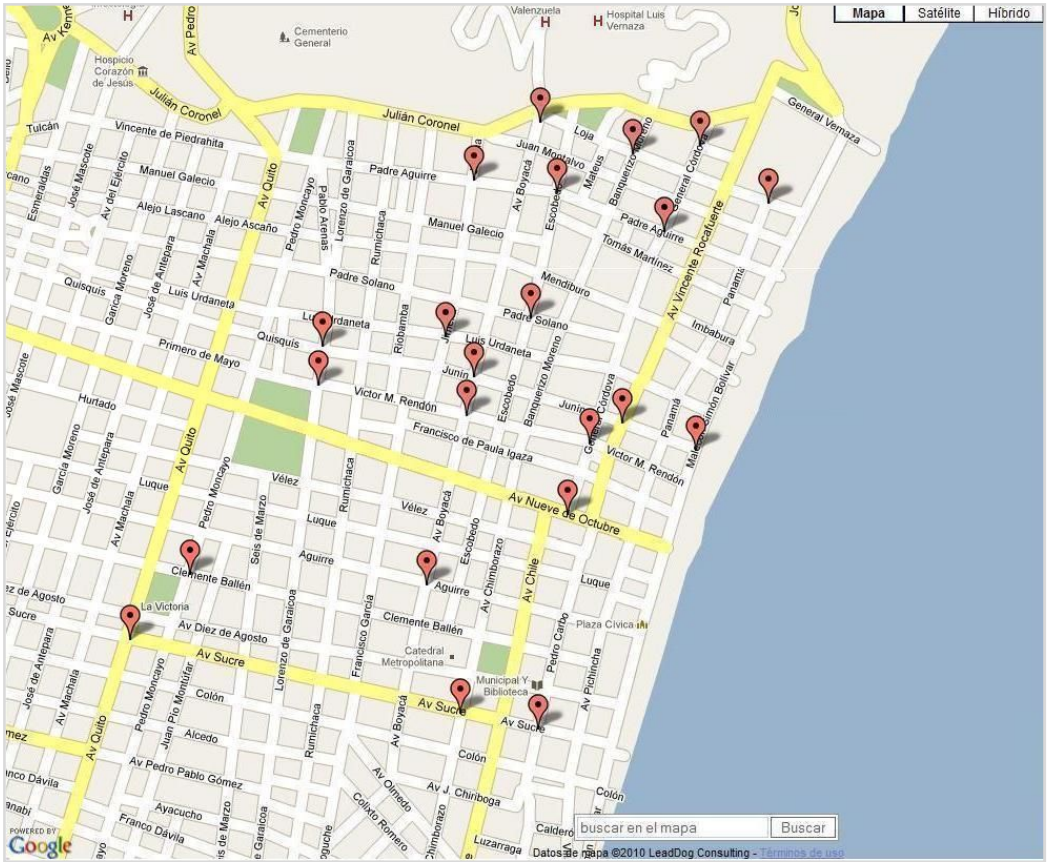


Figura 5.1: Puntos del set de datos considerado para prueba asignación correcta de nodos a rutas.

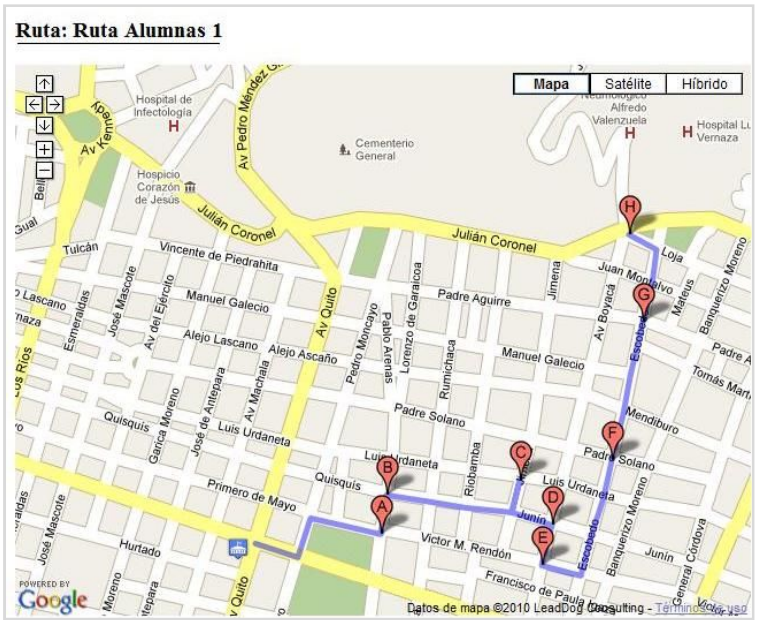


Figura 5.2: Mapa de ruta de alumnas número 1

Figura 5.3: Mapa de ruta de alumnas número 2.

Figura 5.4: Mapa de ruta de alumnas número 3.

La siguiente parte consiste en añadir un punto cercano a las rutas generadas y constatar que este es ubicado en la ruta adecuada. La figura 5.5 muestra el set original y el nuevo punto.

Como parte final se genera rutas nuevamente y se visualizan las 3 rutas resultantes. Se observa que el punto con latitud/longitud (-2.1923316451757047, -79.8864197731018) ha sido incluido en la

ruta 3 que es la ruta correcta que le corresponde ser asignado. Las 3 nuevas rutas son mostradas en las figuras 5.6, 5,7 y 5.8.

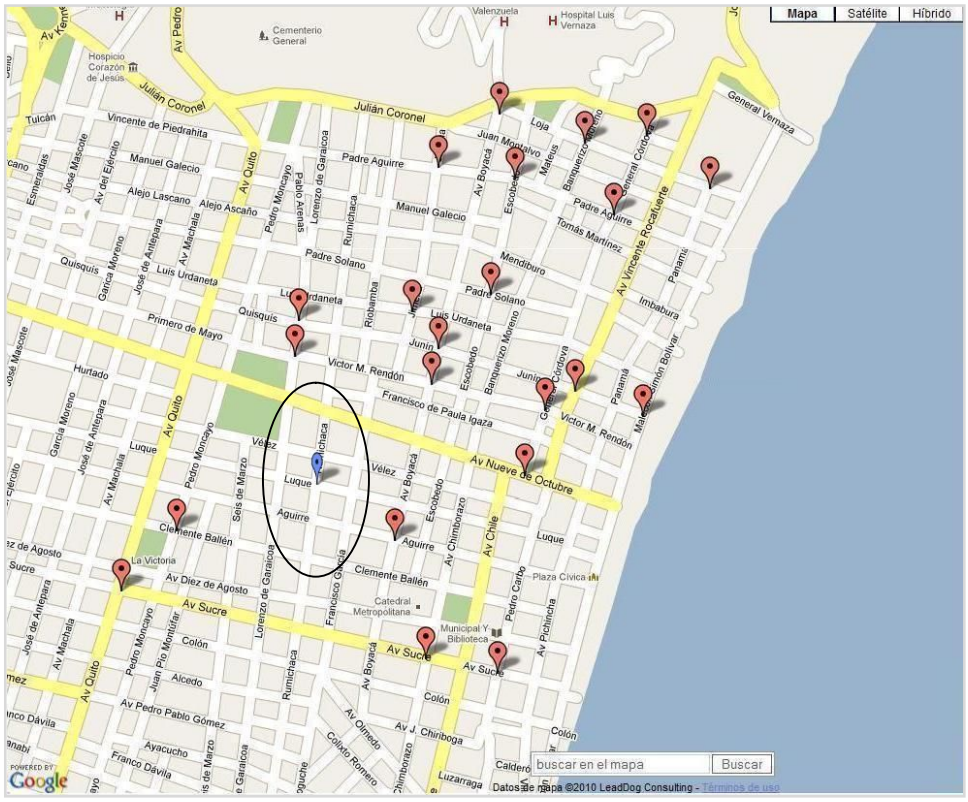


Figura 5.5: Set de datos original más nuevo punto para nueva generación de rutas.

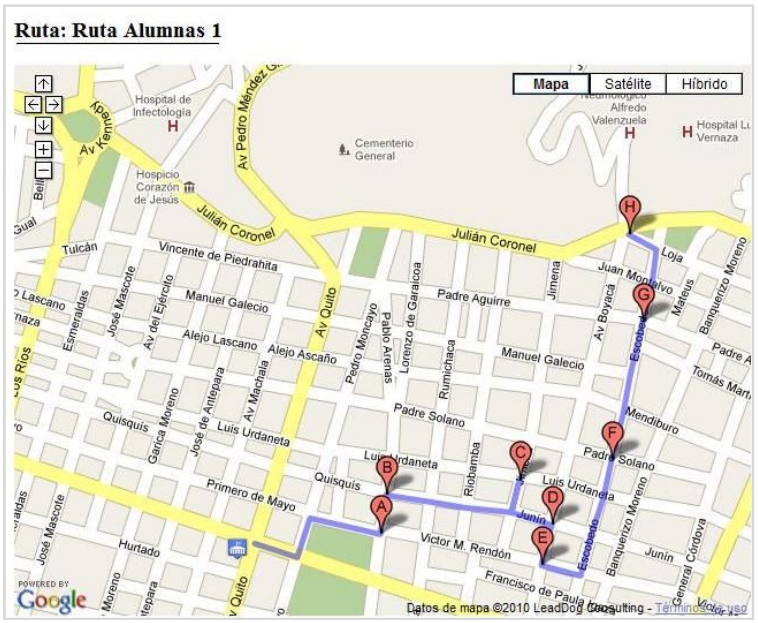


Figura 5.6: Muestra ruta 1 de alumnas luego de añadir punto al set de datos.

Figura 5.7: Muestra ruta 2 de alumnas luego de añadir punto al set de datos.

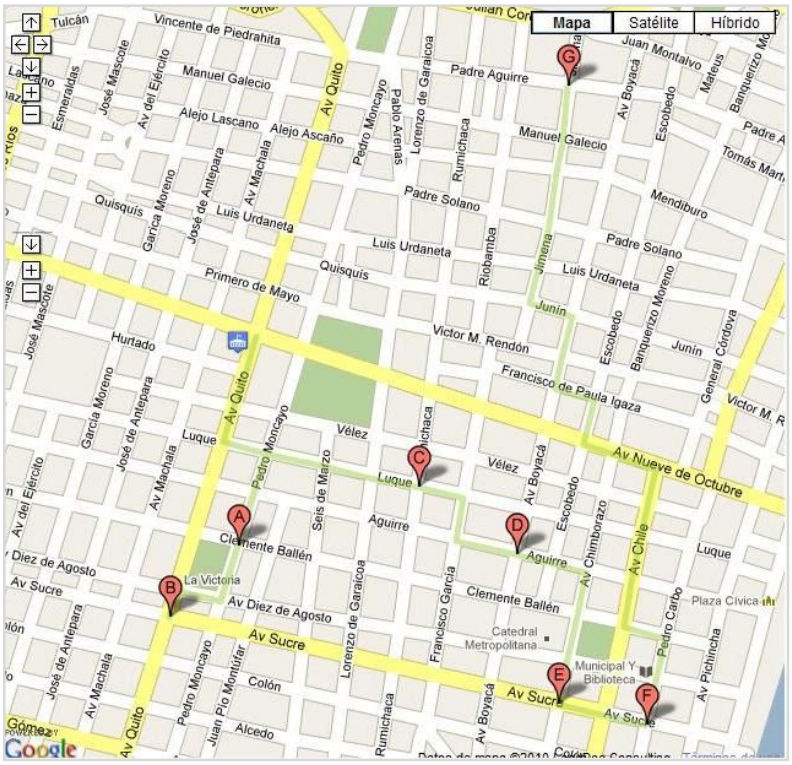


Figura 5.8: Muestra ruta 3 de alumnas luego de añadir punto al set de datos.

Como conclusión de esta prueba realizada se puede decir que la agrupación de los puntos es realizado de acuerdo a lo que se esperaba, es decir considerando la cercanía de los mismos. Cuando se añade un punto, las rutas son reagrupados y el nuevo punto es colocado en la ruta más cercana al punto.

5.2 Medición de Tiempo Usando TSP

Para la medición del tiempo de repuesta del algoritmo de ordenamiento de nodos TSP que nos permite generar rutas óptimas, se ha procedido a realizar pruebas variando de manera progresiva el número de nodos. En la tabla 5.1 se describen los resultados obtenidos y en la figura 5.9 se muestra la relación de tiempo vs número de nodos. Como se puede observar el tiempo de respuesta aumenta rápidamente para un conjunto pequeño de nodos, pero a partir de los 10 nodos la curva se suaviza.

Número de Puntos	Tiempo (Segs.)
5	0,43
10	2,39
30	3,28

Tabla 5.1: Resultados obtenidos de pruebas de tiempo TSP con diferente número de puntos.

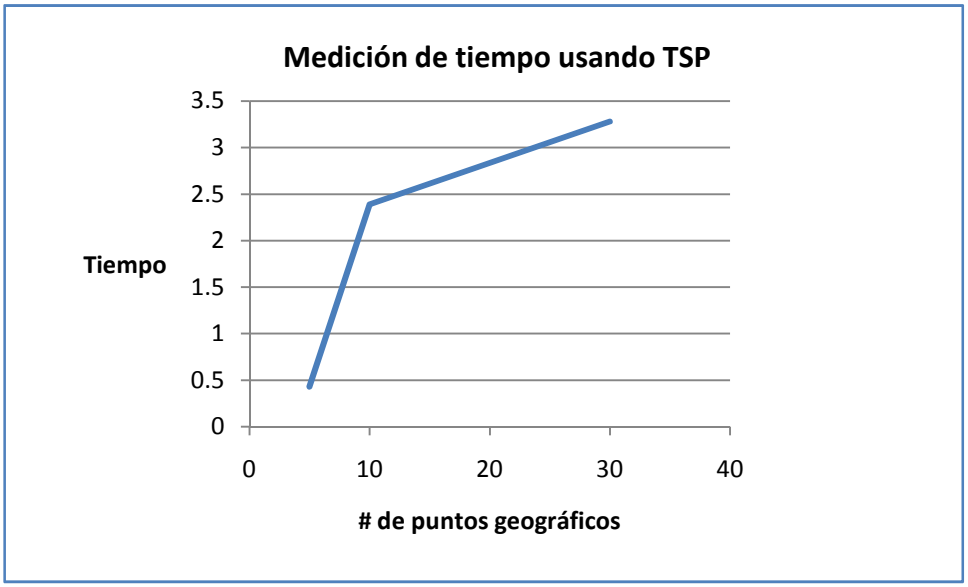


Figura 5.9: Relación número de nodos Vs. tiempo para el algoritmo TSP.

5.3 Medición de Tiempo Usando A-Star

Para medir el tiempo de respuesta del algoritmo de rutas mínimas A-star, se ha procedido de la misma manera que para el algoritmo TSP, se han realizado pruebas aumentando el número de nodos de manera progresiva.

En la tabla 5.2 se muestra los resultados obtenidos para el tiempo de respuesta para diferente número de nodos. También se muestra un gráfico que define la relación de número de nodos vs. tiempo. Como se puede observar en la tabla el tiempo de respuesta del algoritmo

de distancias mínimas aumenta rápidamente para un conjunto pequeño de nodos, la curva tiende a suavizarse a partir de los 10 nodos.

Número de Puntos	Tiempo (Segs.)
5	0,38
10	1,31
30	1,81

Tabla 5.2: Resultados obtenidos de pruebas de tiempo A-Star con diferente número de puntos.

Figura 5.10: Relación número de nodos Vs. tiempo para el algoritmo A-Star.

CONCLUSIONES Y RECOMENDACIONES

Las conclusiones son:

1. Los objetivos propuestos se cumplieron. El API de Google Maps facilitó la creación del repositorio de datos de direcciones domiciliarias geocodificados para el área geográfica definida en el alcance, y utilizando implementaciones previamente propuestas para algoritmos de rutas mínimas fue posible construir un aplicativo web que permite obtener rutas de expreso escolar de óptimas.

2. El algoritmo TSP utilizado para el agrupamiento de puntos en rutas presentó resultados en los cuales se observa que un punto geográfico que se agrega a un set de datos de direcciones geocodificadas es colocado en la ruta que contiene los puntos más cercanos al mismo.

3. Los tiempos de respuesta del algoritmo TSP aumentaron rápidamente para un conjunto pequeño de puntos, luego a partir de un conjunto de 10 puntos el tiempo creció pero con una pendiente menor, por lo que la curva que muestra la relación de tiempo vs. el número de nodos en las pruebas que se realizaron en el capítulo 5 se presenta más suavizada a partir de este número de puntos.

4. El algoritmo A-Star permitió obtener las distancias mínimas de un punto a otro, este algoritmo se aplica para cada punto de la ruta de acuerdo con el orden de visita, los cuales son utilizados luego para graficar estos segmentos de líneas que en su conjunto representan la ruta. Los resultados obtenidos para este algoritmo si representan la distancia mínima de un punto a otro.

5. Los tiempos de respuesta obtenidos en las pruebas realizadas para el algoritmo A-Star aumentaron rápidamente para un conjunto pequeño de nodos, para luego incrementar de manera más lenta a partir de un conjunto de 10 de puntos. El incremento más lento de tiempo a partir de 10 puntos se debe a que como se cuenta con un mayor número de puntos las rutas generadas corresponden a puntos que se encuentran más próximos entre sí. El aumento rápido de tiempo entre la prueba de 5 y 10 puntos se debe a que para el caso de 5 puntos se genera 1 ruta y corresponden a puntos próximos por lo cual su tiempo de respuesta es de 0,38 segs.; en cambio que para la prueba de 10 puntos se generan 2 rutas, la primera ruta corresponde a puntos próximos en cambio la segunda ruta corresponde a puntos que están más distantes.

6. En conclusión el tiempo requerido para la generación de rutas, el cual depende de los tiempos de respuesta de los dos algoritmos mencionados sería como valor promedio 0,25 seg por punto. Con este valor se podría considerar cual sería el tiempo de respuesta de la generación de rutas para un conjunto de puntos mayor. Es importante considerar que el proceso de generación de rutas no se realiza siempre, para el caso de una institución educativa es realizado al inicio de un período lectivo y cada vez que nuevas rutas se requieran generar.

7. La elección de un área geográfica definida para implementar el prototipo fue de gran importancia pues no habría sido posible crear un repositorio de datos de direcciones geocodificadas para toda la ciudad de Guayaquil con los recursos de que se disponían.

8. Para implementar una aplicación se requiere de mucha más información geográfica tales como: sentido de las calles e información domiciliaria geocodificada de un área más amplia. La obtención de esta información geocodificada para esta área más amplia representaría una inversión de tiempo significativo. En el caso de tener un área geográfica de alcance mayor, los algoritmos de generación de rutas considerados en el presente trabajo podrían ser

aplicados ya que los tiempos de respuestas de acuerdo a lo mencionado son aceptables para una aplicación.

Anexos

ANEXO A

Primer ejemplo: Como Incluir un Mapa de Google en tu Página web.

Para insertar un mapa en un sitio web haciendo uso de la API de Google Maps, lo primero que se debe realizar es solicitar una API Key, en la cual se debe especificar en qué URL vamos a utilizar el mapa.

Es recomendable solicitar una API Key para la dirección **http://localhost** con esta se hacen los ajustes necesarios y una vez que el código esté listo cambiar la API Key por la del sitio en Internet para publicar la página.

A continuación se pone una muestra del código que permite colocar un mapa de Google en un sitio web. Es importante destacar que se debe remplazar el texto resaltado en el código: **COLOCAR_AQUI_NUESTRA_KEY** por el API Key, para que funcione correctamente.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8"/>
<title>EjemploApi Google Maps</title>
<script
src="http://maps.google.com/maps?file=api&v=2&key=COLOCAR_A
QUI_NUESTRA_KEY" type="text/javascript"></script>
<script type="text/javascript">
//<![CDATA[
function load() {
if (GBrowserIsCompatible()) {
var map = new GMap2(document.getElementById("map"));
map.setCenter(new GLatLng(-2.18619923032713,-
79.8956036567688), 16);
  }
}
//]]>
</script>
</head>
<body onload="load()" onunload="GUnload()" >
</body>
```

```
<div id="map" style="width: 640px; height:  480px"></div>
</html>
```

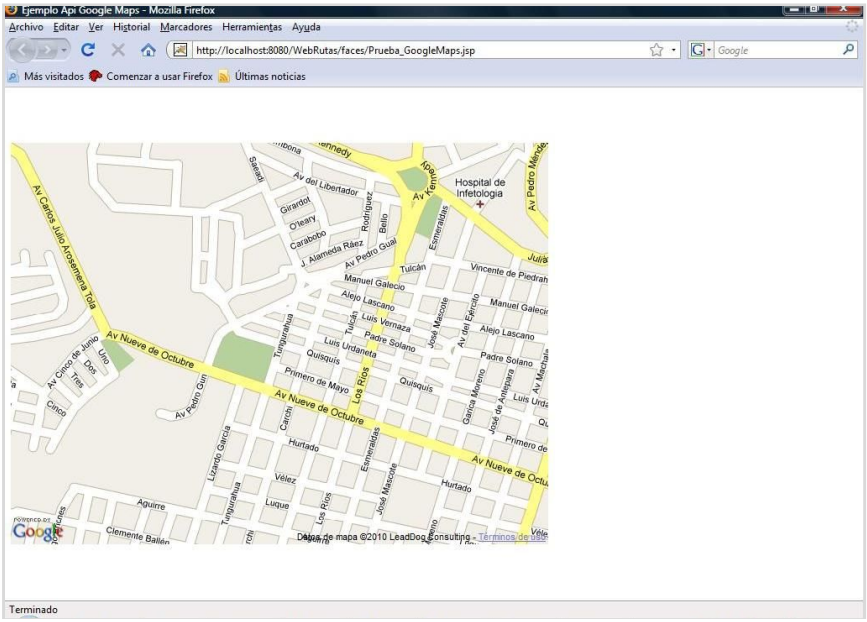


Figura A.1: Pagina web con mapa.

Segundo ejemplo: Cómo Insertar Marcadores a un Mapa

Los marcadores son muy útiles para resaltar ubicaciones. Con la función GMarker() se puede crear una marca en el punto que se pase como parámetro y la función addOverlay() permite adicionar la marca al mapa.

En el código siguiente se ha agregado una marca en la posición (-2.1897371650329718, -79.88893032073975. Este punto corresponde a la posición de llegada y partida de cada una de las rutas a generarse.

```
function load() {
if (GBrowserIsCompatible()) {
var map = new
GMap2(document.getElementById("map"));
map.setCenter(new GLatLng(-2.1897371650329718,
79.88893032073975),5);
```

```
map.addControl(new GMapTypeControl());
map.addControl(new GLargeMapControl());
map.addControl(new GScaleControl());
map.addControl(new GOverviewMapControl());
map.setMapType(G_HYBRID_TYPE);
var point = new GLatLng(-2.1897371650329718,-
79.88893032073975);
map.addOverlay(new GMarker(point));
  }
}
```

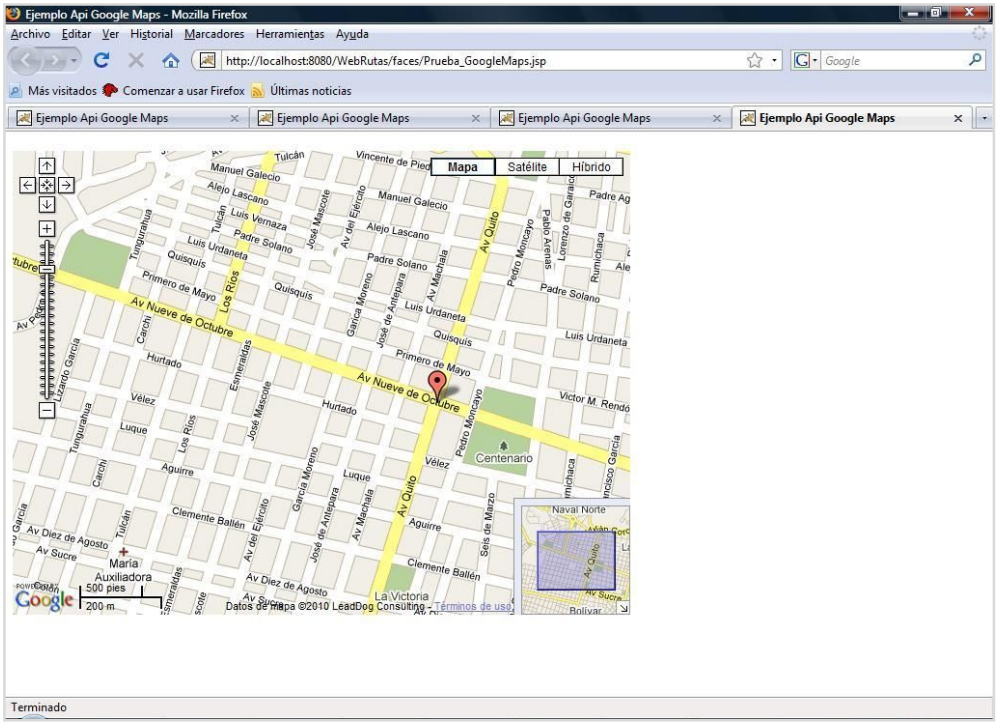


Figura A.2: Mapa que muestra un marcador.

En el código anterior se ha utilizado la función GLatLng() para declarar un punto en la ubicación en que se desea colocar el marcador, luego se crea el marcador con GMarker() y se la agrega al mapa haciendo uso de addOverlay.

ANEXO B

Como Geocodificar una Dirección Domiciliaria y Obtener los Datos de su Longitud Y Latitud.

A continuación se muestra el código que nos permite geocodificar una dirección domiciliaria para obtener de esta manera sus datos de latitud y longitud.

```
<?xml version="1.0" encoding="UTF-8"?>
<jsp:root version="2.1" xmlns:f="http://java.sun.com/jsf/core"
xmlns:h="http://java.sun.com/jsf/html"
xmlns:jsp="http://java.sun.com/JSP/Page"
xmlns:webuijsf="http://www.sun.com/webui/webuijsf">
<jsp:directive.pagecontentType="text/html;charset=UTF-8"
pageEncoding="UTF-8"/>
<f:view>
<webuijsf:page binding="#{Geocodificar_punto.page1}" id="page1">
<webuijsf:html binding="#{Geocodificar_punto.html1}" id="html1">
<webuijsf:head binding="#{Geocodificar_punto.head1}" id="head1">
<webuijsf:link binding="#{Geocodificar_punto.link1}" id="link1"
url="/resources/stylesheet.css"/>
<webuijsf:link id="link2"
url="http://www.google.com/uds/css/gsearch.css"/>
<webuijsf:link id="link3"
url="http://www.google.com/uds/solutions/localsearch/gmlocalsearch.c
ss"/>
<webuijsf:script id="script1" type="text/javascript"
url="http://maps.google.com/maps?file=api&amp;amp;v=2&amp;amp;key=ABQIA
AAA86Rn8UmqFraJ_1kM3GqUFxTwM0brOpm-
All5BF6PoaKBxRWWERSd-99GEJvFZcRxBRrJER-of0wQVg"/>
<webuijsf:script id="script2" type="text/javascript"
url="http://www.google.com/uds/api?file=uds.js&amp;amp;v=1.0"/>
<webuijsf:script id="script3" type="text/javascript"
url="http://www.google.com/uds/solutions/localsearch/gmlocalsearch.js
"/>
<webuijsf:script id="script4" type="text/javascript">
var map;
function initialize() {
alert("hola");
if (GBrowserIsCompatible()) {
map = new GMap2(document.getElementById("map_canvas"));
```

```
map.setCenter(new GLatLng(-2.18619923032713, -
79.8956036567688), 16);

map.addControl(new google.maps.LocalSearch(), new
GControlPosition(G_ANCHOR_BOTTOM_RIGHT, new
GSize(10,20)));
map.addControl(new GSmallMapControl());
map.addControl(new GMapTypeControl());

GEvent.addListener(map,"click", function(overlay,point) {
var marker = new GMarker(point)
map.addOverlay(marker);

document.getElementById("form1:longitud").setProps({value:point.x});
document.getElementById("form1:latitud").setProps({value:point.y});
  });
    }

 }

</webuijsf:script>
</webuijsf:head>
<webuijsf:body binding="#{Geocodificar_punto.body1}" id="body1"
onLoad="initialize();" onUnload="GUnload();" style="-rave-layout:
grid">
<webuijsf:form binding="#{Geocodificar_punto.form1}" id="form1">
<webuijsf:textField binding="#{Geocodificar_punto.latitud}" id="latitud"
style="position: absolute; left: 216px; top: 72px"/>
<webuijsf:textField binding="#{Geocodificar_punto.longitud}"
id="longitud" style="position: absolute; left: 408px; top: 72px"/>
<webuijsf:staticText binding="#{Geocodificar_punto.staticText1}"
id="staticText1" style="position: absolute; left: 216px; top: 48px"
text="Latitud"/>
<webuijsf:staticText binding="#{Geocodificar_punto.staticText2}"
id="staticText2" style="position: absolute; left: 408px; top: 48px"
text="Longitud"/>
<webuijsf:staticText binding="#{Geocodificar_punto.staticText3}"
id="staticText3"
style="font-size: 12px; font-weight: bold; left: 384px; top: 24px; position:
absolute" text="GeocodificarDireccionDomiciliaria"/>
</webuijsf:form>
<div id="map_canvas" style="height: 432px; left: 216px; top: 120px;
position: absolute; width: 622px"></div>
</webuijsf:body>
```

```
</webuijsf:html>
</webuijsf:page>
</f:view>
</jsp:root>
```

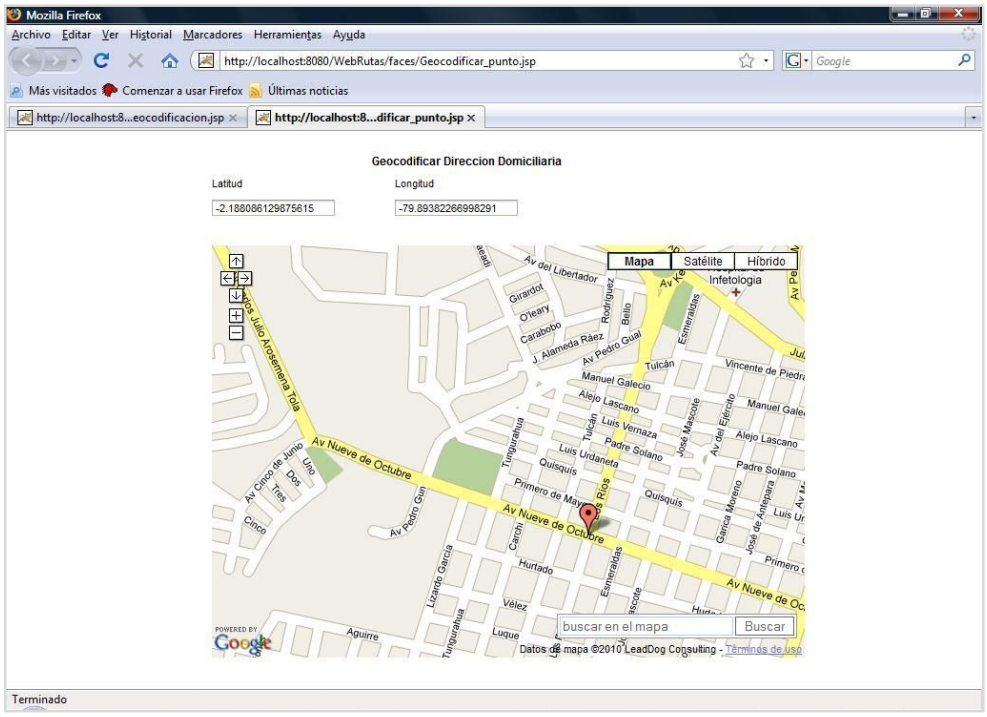


Figura B.1: Geocodificación de dirección domiciliaria.

En el código mostrado se ha utilizado el espacio de nombres GEvent.addListener que contiene funciones que permiten registrar manejadores de eventos tanto customizados y eventos DOM[2], y disparar eventos customizados. El evento utilizado es el evento clic el cual luego de disparado este evento crea un punto del lugar donde se hizo clic para luego mostrar un marcador donde se hizo clic. Finalmente se muestra se muestra los valores x(latitud) e y(longitud) del punto geográfico creado.

[2] DOM: DocumentObjectModel("modelo de objetos del documento" o "modelo de objetos para la representación de documentos").

BIBLIOGRAFÍA

[1] Tyler technologies, Productos Versatrans, <http://www.versatrans.com/products/>, [Consulta: Noviembre 2009].

[2] Mohammed Abdul Khader Nayati, School bus routing and scheduling Using GIS, <http://hig.diva-ortal.org/smash/record.jsf?pid=diva2:120117 > , [Consulta: Noviembre 2009] .

[3] Gecko Microsolutions, Transportation Operations Manager, <http://www.geckoms.com/school-transportation/bus-routing.php>, [Consulta: Diciembre 2009].

[4] WordPress MU, Teoria de Grafos, <http://blogs.utpl.edu.ec/ajbda/2010/05/18/capitulo-5-teoria-de-grafos/>, [Consulta: Enero 2010].

[5] Anany Levitin, Introduction to the Design & Analysis of Algorithms, Pearson Education, Agosto 2002.

[6] Pilar Bravo, Juan Carlos Ferrando y Ana Martínez, Complementos de Matemática Discreta Curso Práctico, Universidad Politécnica de Valencia - Servicio de Publicaciones, 1994.

[7] ALEGSA, Para qué sirve y cómo funciona un dispositivo GPS, <http://www.alegsa.com.ar/Notas/195.php>. [Consulta: Febrero 2010].

[8] Kylie Bryant y Arthur Benjamin, Genetic Algorithms and the Traveling Salesman Problem, <http://www.math.hmc.edu/seniorthesis/archives/2001/kbryant/kbryant-

2001-thesis.pdf>, [Consulta: Marzo 2010].

[9] Abdollah Homaifar, Shanguchuan Guan, and Gunar E. Liepins, Schema Analysis of the traveling salesman problem using genetic algorithms. Sociedad Matemática Americana, 1992.

[10] *Hart P. E., Nilsson N. J. and Raphael B, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Transactions on Systems Science and Cybernetics, 1968.*

[11] Gerard Reinelt, The Traveling Salesman: Computational Solutions for TSP Applications, Springer-Verlag, 1994.

[12] E.L. Lawler, J.K. Lenstra, A.H.G. RinnooyKan, and D.B.Shmoys, The Traveling Salesman. JohnWiley and Sons, 1986.