INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Trabajo Terminal II

"SISTEMA DE NAVEGACIÓN GUIADA BASADO EN TÉCNICAS DE ALGORITMOS EVOLUTIVOS"

Que para obtener el título de

"Ingeniero en Telemática"

Presenta:

Santillán Moncayo Oriana

Zamudio Alonso Karina

Asesores:

M. en C. Blanca Alicia Rico Jiménez

M. en C. Carlos Hernández Nava

Dr. Carlos A. Coello Coello

Agosto 2010

INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Trabajo Terminal II

"SISTEMA DE NAVEGACIÓN GUIADA BASADO EN TÉCNICAS DE ALGORITMOS EVOLUTIVOS"

Que para obtener el título de

"Ingeniero en Telemática"

Presenta:

Santillán Moncayo Oriana

Zamudio Alonso Karina

Asesores:

M. en C. Blanca Alicia Rico Jiménez

M. en C. Carlos Hernández Nava

Dr. Carlos A. Coello Coello

Presidente del Jurado

Profesor titular

M. en C. Noé Torres Cruz

M. en C. Cynthia E. Enríquez Ortíz

Gracias a todos nuestros asesores por su confianza, apoyo y compromiso desde el principio hasta la finalización de éste proyecto. A mi Madre y hermanos por que son la razón de mis expectativas y de la motivación para terminar ésta etapa de mi vida. A mi pareja por su apoyo incondicional, sus consejos para continuar en la elaboración de éste proyecto y que en momentos de oscuridad siempre estas ahi. A mis amigos por su amistad que en 5 años a madurado por todos los buenos y malos momentos que hemos vivido en esta carrera.
Karina Zamudio Alonso

Quiero dar las gracias a cada uno de las personas que me han demostrado su apoyo y cariño incondicional, Quiero agradecer a nuestros asesores por su dedicación, solidaridad, apoyo, confianza y amistad a lo largo de este proyecto. A mis padres y hermanos, por que son el motor que rige mi existencia, por su apoyo, amor, confianza, paciencia y entrega que me han brindado. A mis amigos que son mi segunda familia y en cada uno de ellos he encontrado a un hermano a lo largo de todo este trayecto. A esa persona especial, que a pesar de todo me demostro su apoyo en los momentos difíciles y sigue presente en mis pensamientos. A cada uno que directa o indirectamente han influido en mi vida, gracias.
Oriana Santillán Moncayo

Índice

Índice de figuras

Índice de tablas

Resumen

Sistema de navegación guida basado en técnicas de algoritmos evolutivos

Palabras Clave: Aplicación Web, algoritmos evolutivos, utilización de mapas

Resumen: En este documento describimos el análisis, diseño e implementación a detalle de nuestro sistema, que lleva por nombre *Sistema de navegación guida basado en técnicas de algoritmos evolutivos* que está orientado a vehiculos que realicen entregas de pedidos, donde, como primer paso, el cliente efectua una solicitud de compra por medio de un Aplicación Web donde se le solicitan los datos de este y los productos a llevar, el cliente va a tener la facilidad de localizar el destino de entrega en un mapa, esto ayuda al análisis y determinación de la ruta, por que se obtienen las coordenadas del sitio donde se tiene que entregar el pedido, posteriormente una vez obtenida toda la información necesaria se genera la orden de pedido, una vez recabadas todas las peticiones del día, se crea y ordena un itinerario para los transportistas en base a la distancia entre la distribuidora y el destino. Sobre esto se crea una permutación que produce el minimo costo de la ruta, esta permutación es producto de un algoritmo evolutivo que hace uso de parámetros como la distancia, obteniendo como resultado las coordenadas de la ruta, por otra parte, el dispositivo móvil se podrá conectarse a internet para poder acceder a un sitio web donde de desplegará la ruta en un mapa de google maps.

Introducción

A lo largo de éste documento se describe de forma extensa los conceptos básicos que investigamos con el fin de poder establecer una idea de cómo atacar el problema y con ello definir los requerimientos del sistema, definición del software a utilizar, establecer a nuestros actores principales que interactuarán con el sistema y fijar las funciones que cada uno podrá realizar.

Posteriormente, se muestra la sección de diseño, donde nos apoyamos por medio de diagramas para modelar nuestro sistema definiendo la arquitectura del mismo y especificando las funciones de cada una de las partes que la conformen ya que más adelante nos sirve de guía al momento de realizar la implementación. Después, en la parte de implementación, se describe de manera breve y clara cada función que se realiza. Y finalmente se muestran los resultados que se obtuvieron a partir de la ejecución de las partes antes mencionadas, así como las conclusiones del presente proyecto.

Objetivos

Objetivo general

Realizar un sistema para el proceso de entrega de pedidos para flotillas de transportistas, generando itinerarios de entrega y proporcionando rutas a través de algoritmos evolutivos, donde las rutas se visualizarán por medio de un dispositivo móvil que podrá acceder a internet para poder visualizar la ruta.

Objetivos Particulares.

- Introducirnos en la teoría de algoritmos evolutivos.
- Estudiar, analizar y determinar el número de parámetros para la obtención de rutas.
- Estudiar, diseñar e implementar un modelo matemático que relacione los parámetros determinados.
- Investigar diferentes modelos ya creados para tener de referencia y nos auxilie para comprobar su funcionalidad con respecto al nuestro.
- Investigar y evaluar un algoritmo evolutivo que responda de forma eficiente al modelo matemático.
- Implementar el algoritmo evolutivo en base al modelo matemático obtenido mencionado anteriormente.
- Obtener rutas mediante la implementación del algoritmo evolutivo.

- Crear una aplicación Web para administrar el registro de los pedidos.
- Crear un módulo dentro de la aplicación Web para la creación del itinerario de entrega de los pedidos.
- Incrustar el API *Application Programming Interface* de Google Maps en la aplicación Web para que el usuario indique la ubicación de donde se deberá entregar el pedido.
- Investigar la forma en la que se enviará la ruta hacia el dispositivo móvil.
- Investigar el tema de bases de datos espaciales para la manipulación de los datos y así implementarlos en el sistema.
- Relacionar la información contenida en la base de datos espacial con las rutas obtenidas.
- Evaluar el software en la que se realizará el algoritmo y la base de datos espacial.
- Realizar pruebas y determinar resultados deseados.

Parte I

Panorama General

CAPÍTULO 1

Antecedentes

En éste capitulo, mostramos algunos de los proyectos o prototipos que se han realizado en UPIITA, y de igual forma manejaron temas semenjantes relacionados a la determinación de rutas. Exponiendo asu vez las diferencias de estos proyectos con nuestra propuesta. En la Tabla 1.1, una comparación de los diferentes trabajos realizados en UPIITA:

Además se puede apreciar las distintas características encontradas de algunos trabajos terminales con respecto al nuestro. Una de las primeras diferencias es que éste ofrece al usuario varias opciones diferentes como lo es la administración de itinerarios, implementación de algoritmo evolutivo para la determinación de la ruta y navegación para visualización de la ruta.

CARACTERÍSTICAS	LIVE	GUÍA MÓVIL	**SISTEMA PROPUEST**
Dispositivo para consulta	Dispositivo móvil instalado en el vehículo	Laptop	**Dispositivo móvil**
GPS	Sí	Sí	**Sí**
Algoritmo evolutivo	No	No	**Sí**
Monitoreo de transito	Si	Si	**No**
Consideración del medio de transporte	Si	Si	**Si**
Rutas	Si	Si	Si
Cobertura	Zona Zacatenco	Delegación Gustavo A. Madero	**Zona Acueducto, Zaca Lindavista**
Arquitectura cliente - servidor	Si	Si	**Si**

Tabla 1.1: Comparación con trabajos terminales realizados en upiita

CAPÍTULO 2

Planteamiento del problema

Algunos problemas con la operación de almacenes se debe a que cuentan con un servicio ineficiente en el manejo y control de rutas de distribución. Ya que el encargado de almacén no realiza un itinerario de tal forma que cubra los pedidos del día basándose en un criterio de distribución que facilite las entregas de estos, de manera se aproveche las distancias entre un punto y otro. Para no dar recorridos redundantes o innecesarios que implican gastos y que en muchos casos la mala distribución del rol de entrega genere clientes insatisfechos debido a que no se cumplieron las fechas de entrega así como el horario. Dada la problemática anterior, una mejora para este problema es diseñar un sistema que facilite y sistematice la creación del itinerario de entregas por día basado en un criterio de ordenamiento, que además, te permita crear las rutas que se deben tomar. De está manera la ruta que que se genere para cubrir la entrega de todos los pedidos sea la más corta y así evitar retrasos y costos para la empresa distribuidora.

Para la determinación de la ruta se requiere fijar parámetros como; la ruta más corta y diseñar un modelo matemático por lo que nos permite denotar la complejidad que implica, ya que obteniendo un buen diseño del modelo matemático nuestros valores serán más confiables con respecto a otros modelos. La elección del algoritmo evolutivo va de la mano con el modelo matemático, sin embargo, éste se deberá elegir de tal forma que se adapte completamente al modelo.

CAPÍTULO 3

Propuesta de solución

El sistema cuenta con dos módulos principales, como se muestra en la fig. 3.1 donde el primero, se enfoca al dispositivo móvil que traerá consigo el conductor del vehículo y por último se contará con un segundo módulo; que contendrá al servidor y la aplicación Web, donde se obtendrá la información del pedido y del proveedor, además de generar la ruta por medio de un algoritmo evolutivo que se visualizará en el dispositivo móvil.

Esquema general donde se involucran los dos módulos principales

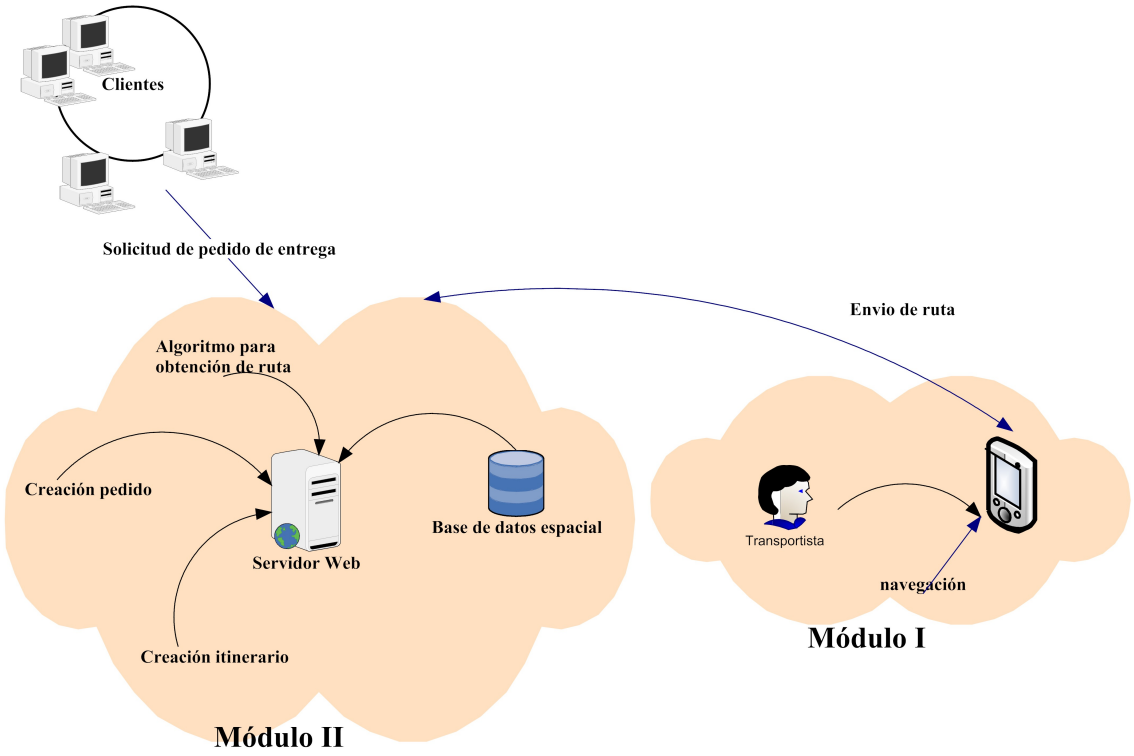


Figura 3.1: Módulos principales del sistema

3.1. Módulo I

Éste se encargará de todo lo referente a la aplicación del dispositivo móvil, donde se visualizará la ruta desplegada en un mapa. La visualización de los mapas se realizará con un servicio web a la cuál podrá tener acceso el dispositivo móvil, por lo que se utilizó (Dispositivo Móvil: Nokia N82, con sistema operativo Symbian). Las tareas para realizar este módulo serán:

- Estudiar sobre la creación de aplicaciones para el sistema operativo Symbian.
- Estudiar funciones, servicios y aplicaciones de Google Maps para la integración del sistema a tráves de KML.
- Estudiar diseño y alcances de servicios web y su interacción con sistemas terceros.

3.2. Módulo II

En éste modulo se realizará la obtención de las rutas mediante un algoritmo evolutivo ofreciendo mayores ventajas, así como la aplicación Web para la obtención de la información del pedido.

- Introducirnos a la teoría de algoritmos evolutivos.
- Estudiar, analizar y determinar los parámetros a tomar en cuenta dentro de las rutas.
- Estudiar, diseñar e implementar un modelo matemático que relacione todos los parámetros obtenidos anteriormente.
- Evaluar diferentes algoritmos evolutivos para determinar cúal se adecua mejor para resolver el problema.
- Implementar el algoritmo tomando como base el modelo matemático obtenido, en un lenguaje de programación.
- Estudiar la estructura de las bases de datos espaciales.
- Enlazar el algoritmo evolutivo con la información de la base de datos espacial de la delegación Gustavo A. Madero.
- Crear una aplicación Web para administrar la información del registro de los pedidos y gestione los itinerarios de los vehículos repartidores.
- Estudiar sistemas de información geográfica para la visualización de datos geográficos.

Parte II

Marco Teórico

CAPÍTULO 4

Marco Teórico

4.1. Breve descripción de los siguientes conceptos

4.1.1. Modelos del proceso del sotfware

A continuación mostraremos tres tipos de modelos donde cada uno representa una perspectiva particular, y proporciona sólo información parcial sobre ese proceso. Los modelos de procesos son:

- El modelo en cascada: Este modelo se muestra en la Figura. 4.1, se llama de esta forma debido a la cascada de una fase a otra, las principales etapas de este modelo se transforman en actividades fundamentales de desarrollo:

    1. Análisis y definición de requerimientos. Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Por lo que se definen en detalle y sirven como una especificación del sistema.
    2. Diseño del sistema y del software: El proceso de diseño divide los requerimientos en sistemas hardware o software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.

3. Implementación y prueba de unidades: durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programa. La prueba de unidades implica verificar que cada una cumpla su especificación.
4. Integración y prueba del sistema: Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema se entrega al cliente.
5. Funcionamiento y mantenimiento: Por lo general, ésta es la fase más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubran nuevos requerimientos.

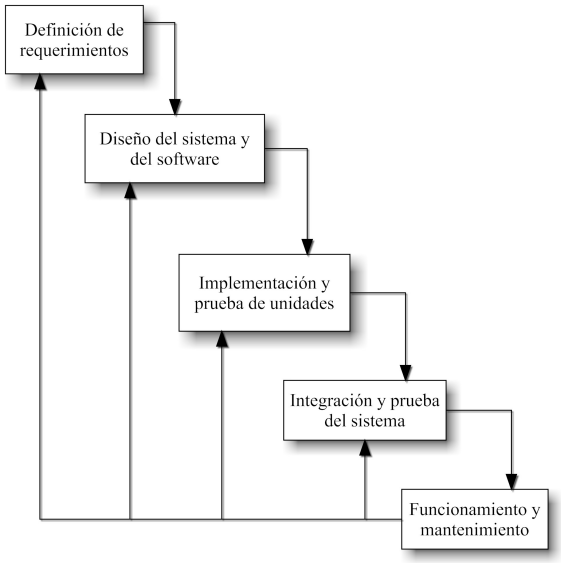


Figura 4.1: El ciclo de vida del software

el resultado de cada fase es uno o más documentos aprobados. La siguiente fase no debe empezar hasta que la fase previa haya finalizado.

Existen dos tipos de desarrollo evolutivo:

- Desarrollo exploratorio, donde el objetivo del proceso es trabajar con el cliente para explorar sus requerimientos y entregar un sistema final. El desarrollo empieza con las partes

- Desarrollo evolutivo: Se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado. Las actividades de especificación, desarrollo y validación se entrelazan en vez de separarse, con una rápida retroalimentación entre estas:

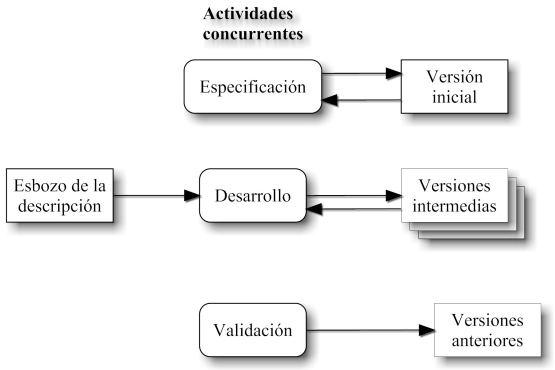


Figura 4.2: Desarrollo evolutivo

del sistema que se comprenden mejor. El sistema evoluciona agregando nuevos atributos propuestos por el cliente.

- Prototipos desechables: donde el objetivo del proceso de desarrollo evolutivo es comprender los requerimientos del cliente y entonces desarrollar una definición mejorada de los requerimientos para el sistema. El prototipo se centra en experimentar con los requerimientos del cliente que no se comprenden del todo.

1. Análisis de componentes. Se buscan estos componentes para implementar esta especificacion, donde éstos solo proporcionan parte de la funcionalidad requerida.

2. Modificación de requerimientos. Los requerimientos se analizan utilizando información acerca de los componentes que se han descubierto, entonces éstos se modifican para reflejar los componentes disponibles, si las modificaciones no son posibles, la actividad de análisis de componentes se puede llevar a cabo nuevamente para buscar soluciones alternativas.

3. Diseño del sistema con reutilización. Se diseña o se reutiliza un marco de trabajo para el sistema. Si los componentes reutilizables no están disponibles, se puede tener que diseñar un nuevo software.

- Ingeniería de software basada en componentes. Este enfoque basado en la reutilización se compone de una gran base de componentes software reutilizables y de algunos marcos de trabajo de integración para estos. Algunas veces estos componentes son sistemas por sí mismos que se pueden utilizar para proporcionar una funcionalidad especifica, como dar formato al texto o efectuar cálculos numéricos. En la Figura. 4.3. Aunque la etapa de especificación de requerimientos y la de validación son comparables con otros procesos, las etapas intermedias en el proceso orientado a la reutilización son diferentes. Estas etapas son:

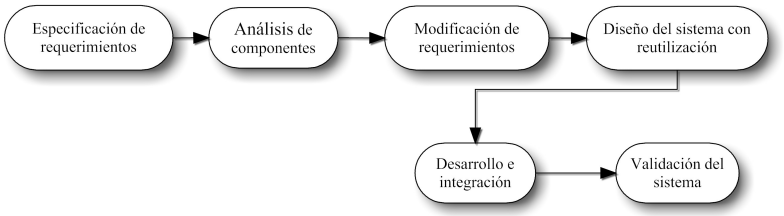


Figura 4.3: Ingeniería del software basada en componentes

4. Desarrollo e integración. El software que no se puede adquirir externamente se desarrolla, y los componentes y los sistemas se integran. En este modelo, la integración de sistemas es parte del proceso de desarrollo, más que una actividad separada.

4.1.2. Sistema de gestión de base de datos

Los Sistemas de gestión de base de datos son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan. Se compone de un lenguaje de definición de datos, de un lenguaje de manipulación de datos y de un lenguaje de consulta. En los textos que tratan este tema, o temas relacionados, se mencionan los términos SGBD y DBMS, siendo ambos equivalentes, y acrónimos, respectivamente, de Sistema Gestor de Bases de Datos y DataBase Management System.

- PostgreSQL: PostGreSQL es un sistema de gestión de bases de datos objeto-relacional es una derivación libre (OpenSource). Soporta distintos tipos de datos: además del soporte para los tipos base, también soporta datos de tipo fecha, monetarios, elementos gráficos, datos sobre redes (MAC, IP), cadenas de bits, etc. También permite la creación de tipos propios.

  Incorpora funciones de diversa índole:

- Manejo de fechas, geométricas, orientadas a operaciones con redes, etc.
- Permite la declaración de funciones propias, así como la definición de disparadores.
- Soporta el uso de índices, reglas y vistas.
- Permite la gestión de diferentes usuarios, como también los permisos asignados a cada uno de ellos.
- PostgreSQL está considerado como la base de datos de código abierto más avanzada del mundo. PostgreSQL proporciona un gran número de características que normalmente sólo se encontraban en las bases de datos comerciales tales como DB2 u Oracle.

Características:
Velocidad de respuesta: La velocidad de respuesta que ofrece este gestor con bases de datos relativamente pequeñas puede parecer un poco deficiente, aunque esta misma velocidad la mantiene al gestionar bases de datos realmente grandes.
DBMS Objeto-Relacional: PostgreSQL aproxima los datos a un modelo objeto-relacional, y es capaz de manejar complejas rutinas y reglas Altamente Extensible: PostgreSQL soporta operadores, funcionales métodos de acceso y tipos de datos definidos por el usuario.
Soporte SQL Comprensivo: PostgreSQL soporta la especificación SQL99 e incluye características avanzadas tales como las uniones (joins) SQL92.
Integridad Referencial: PostgreSQL soporta integridad referencial, la cual es utilizada para garantizar la validez de los datos de la base de datos.

API Flexible: La flexibilidad del API de PostgreSQL ha permitido a los vendedores proporcionar soporte al desarrollo fácilmente para el RDBMS PostgreSQL. Estas interfaces incluyen Object Pascal, Python, Perl, PHP, ODBC, Java/JDBC, Ruby, TCL, C/C++, y Pike.

Lenguajes Procedurales: PostgreSQL tiene soporte para lenguajes procedurales internos, incluyendo un lenguaje nativo denominado PL/pgSQL. Este lenguaje es comparable al lenguaje procedural de Oracle, PL/SQL. Otra ventaja de PostgreSQL es su habilidad para usar Perl, Python, o TCL como lenguaje procedural embebido.

Cliente/Servidor: PostgreSQL usa una arquitectura proceso-por-usuario cliente/servidor. Hay un proceso maestro que se ramifica para proporcionar conexiones adicionales para cada cliente que intente conectar a PostgreSQL.

4.2. Sistemas de bases de datos espaciales

4.2.1. Definición y clasificación de los datos espaciales

Los datos espaciales son datos que pueden ser visualizados, manipulados y analizados por sus atributos, que denotan una localización sobre la superficie del globo terraqueo. Estos atributos espaciales son normalmente dados en forma de coordenadas, que dan la posición y la estructura de una característica representada geográficamente. Los datos espaciales tienen dos propiedades importantes:

- se referencia a un espacio geográfico, es decir, son datos registrados y aceptados sobre un sistema de coordenadas geográfico para representar áreas de la superficie terrestre. Por lo tanto, si se tienen diversas fuentes, se puede hacer referencias cruzadas e integrarlas espacialmente.

- son representados en una variedad de escalas y cuando son almacenados en escalas relativamente pequeñas, representan grandes áreas de la superficie terrestre y deben ser generalizados y simbolizados.

La unidad básica de los datos espaciales en la representación vectorial es llamada “objeto geográfico”, el cual se identifica como una característica del mundo real o fenómeno representado por un punto, línea o polígono. En las bases de datos espaciales, los datos vectoriales pueden ser almacenados como parte de la base topográfica, cuya función es proveer la referencia espacial a sistemas para su visualización, manipulación y análisis.

Los datos espaciales incluyen un componente que describe la localización espacial de un objeto geográfico y un componente de atributos para describir sus propiedades y toponimia. El componente espacial puede ser representado mediante las siguientes formas:

- Representación raster.- Este modelo usualmente se aplica cuando se tienen mallas de puntos sobre campos continuos, como la elevación, temperatura y concentraciones químicas; y cuando se utilizan mallas de celdas, representan imágenes; se basa en almacenar matrices de valores.

Si las celdas se categorizan por números, uno o más atributos pueden ser asignados a la celda. La unidad de área en las mallas de celdas se llaman pixeles.

El modelo raster fue diseñado para el análisis, modelado y procesamiento de imágenes.

- Representación vectorial.- Este modelo se basa en la representación arco-nodo y es utilizado para representar áreas, líneas y puntos. Los arcos pueden formar conjuntos de líneas. Además, mediante arcos se forman polígonos para representar áreas. Un caso especial del modelado de datos vectorial es el modelado de datos puntual, que es una colección de puntos independientes que, establecidas las coordenadas, representan lugares puntuales que se distribuyen irregularmente.

4.2.2. Estructura de los datos espaciales

El concepto de un dato espacial tipo *Geometry*

El campo de la geometría, se entiende comúnmente como la rama de las matemáticas que estudia las propiedades y relaciones de puntos, líneas, ángulos, superficies y sólidos en cualquier dimensión. En el contexto del procesamiento de datos espaciales, la palabra *geometry* supone un nuevo significado cuando el OGC *(Open Geospatial Consortium)* formalizó su uso en la publicación OpenGIS Simple Feature Specification for SQL.

En este comunicado el OGC propuso una jerarquía de tipos de datos, llamada *geometry object model*, la cual permite que características espaciales puedan ser representadas en una base de datos. En el modelo, la palabra geometry es usada para representar una característica espacial como un "objeto"que tiene al menos un atributo *geometric type* en la base de datos.

La clase raíz del modelo jerárquico *geometry object model*, es la clase *geometry*, de la cual no se pueden crear instancias. La superclase *geometry* tiene cuatro subclases llamadas: *point*, *curve*, *surface* y *geometry collection*. Se pueden crear instancias de estas subclases las cuales contienen un conjunto de métodos, que son utilizados para verificar sus respectivas propiedades geométricas, y soportar el análisis espacial. Existen varios tipos *geometry* en el modelo, como se muestra en la **Figura. 4.4** , en la cual, los nombres de los objetos se mantienen en inglés para seguir la especificación. Estos tipos son primitivas gráficas utilizadas para construir la geometría de objetos geográficos que usan una o más primitivas. Un poste de luz, un segmento de carretera, una parcela son ejemplos de estas primitivas gráficas. Estos objetos son llamados *simple geometries*. Un grupo de islas que es tratado

como una sola entidad geográfica, e.g. Nueva Zelanda y Fiji, son construidas con múltiples primitivas gráficas. Estos objetos son llamados *complex geometries*.

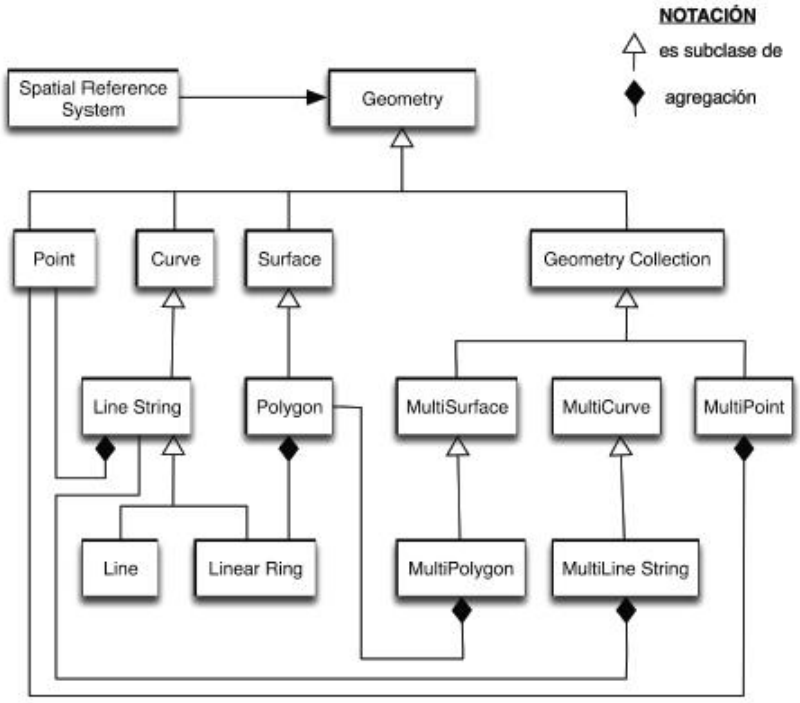


Figura 4.4: El modelo de la OGC (Geometry object model)

Las primitivas gráficas son espacialmente referenciadas a un sistema de coordenadas, la posición y orientación de un objeto siempre están relacionados a un espacio geográfico particular. Esta referencia espacial al sistema de coordenadas ayuda a unir los objetos geográficos a otros, no importando las primitivas gráficas con las cuales fue construido. Además de su localización, un objeto tipo geometry tiene otros atributos que describen sus características, tales como el nombre, clasificación y dimension. En una base de datos espacial, los objetos que tienen los mismos atributos forman una capa (layer, en inglés) o también llamada feature class; que consecuentemente son almacenadas en la misma tabla.

4.2.3. Sistemas de bases de datos espaciales

Los sistemas de bases de datos espaciales son definidos como una clase de sistemas de base de datos que cumplen con las siguientes características:

- Un sistema de bases de datos espaciales es un sistema de base de datos,
- ofrece tipos de datos espaciales (SDT, por *spatial data types*), un modelo de datos y un lenguaje de consulta,
- soporta tipos de datos espaciales en su desarrollo, dando un indexado espacial y algoritmos eficientes para la operación reunión espacial (*spatial join*).

La primera característica enfatiza el hecho que estos sistemas son completamente sistemas de bases de datos, capaces de soportar todos los estándares de modelado y consultas, pero con la funcionalidad adicional de efectuar tareas específicas a datos espaciales. Esta habilidad de manipular y relacionar datos espaciales y no espaciales distingue a sistemas de bases de datos espaciales de otros sistemas que usan datos con referencias geográficas, tales como sistemas cartográficos, diseño asistido por computadora y sistemas de procesamiento de imágenes para el sensado remoto.

Los tipos de datos espaciales, segunda característica, se refieren al uso de puntos, líneas y polígonos como forma de representación geométrica de objetos o entidades en el espacio.

Los sistemas de bases de datos modernos se caracterizan porque pueden manipular grandes cantidades de datos, estos sistemas deben conservar un acceso eficiente mediante técnicas de indexado. Esta característica también es necesaria en los sistemas de bases de datos espaciales. Sin embargo, los datos en bases de datos espaciales son representados y referenciados en dos y tres dimensiones, así los datos en una base de datos espacial deben ser indexados espacialmente. Lo que permite que los datos sean accedidos y analizados por localización (utilizando las coordenadas geográficas) y por las relaciones topológicas que definen la posición de objetos en el espacio relativo a otro (adyacencia, contenido y conectividad). Juntos, la localización y la topología permiten varios métodos de traslape y combinación espacial de objetos en las capas de mapas, este tipo de consultas se llaman reuniones espaciales, y son usadas para propósitos de análisis.

4.2.4. Relación entre bases de datos espaciales y SIG

En ambientes de procesamiento de datos espaciales existe una división entre sistemas de bases de datos espaciales y SIG (véase **Tabla 4.1**). Tomando la ventaja de los avances en las tecnologías computacionales, los sistemas de bases de datos son desarrollados para almacenar y gestionar grandes cantidades de información referenciada geográficamente basada en su localización. Estos sistemas indexan los datos espacialmente para crear consultas eficientes, proveer mecanismos y procedimientos requeridos para proteger los datos de una destrucción física y perdida o degradación de la integridad lógica. Sin embargo, los sistemas de bases de datos espaciales son generalmente débiles en su funcionalidad para la edición de datos, en el análisis de datos y generación de mapas y otros productos información cartográfica. Estos procesos y funciones son manejados de mejor forma por SIG, los cuales son primordialmente diseñados para reunir y usar los datos espaciales para esos propósitos.

Sistemas	Tareas Primarias
Sistemas de Información Geográfica	▪ Reunir y editar datos ▪ Análisis de datos ▪ Generación de mapas y productos de información catográfica
Sistemas de bases de datos espaciales	▪ Almacenamiento y gestión espaciales de datos ▪ Indexado espacial ▪ Seguridad e integridad de los datos ▪ Consultas de datos espaciales

Tabla 4.1: Relación entre bases de datos espaciales y SIG

4.2.5. Visualizador de mapas

Para la realización de un Visualizador de Mapas puede establecerse con Interfaz de Programación de Aplicaciones directa de mapas que incluya el script de las coordenadas. O bien un Sistema de Información Geográfica(SIG).

A continuación fundamentaremos ambas tecnologías:

- SIG Es un sistema computacional, que consiste en una base de datos que almacena información espacial y descriptiva de un entorno geográfico como parte del Mundo real; permitiendo la entrada, mantenimiento, análisis, transformación, manipulación y presentación de datos espaciales, de algún punto geográfico en particular, se dice:

  - Un SIG sirve para distinguir algunas necesidades importantes cuando las descripciones geográficas juegan un papel muy importante en las observaciones.
  - Un SIG se utiliza para resolver problemas de planificación y gestión relacionados con elementos geográficos, haciendo uso de información espacial.

  Las partes de un SIG consiste de 4 componentes interrelacionados:

  - Hardware: Es la parte física y es representado por algún tipo de plataforma.
  - Software: Realiza la manipularon de los datos y las operaciones a través del usuario.
  - Datos espaciales: Elemento base para llevar acabo todas las operaciones deseadas en un SIG
  - Personal: Personas encargadas del diseño, implantación y uso del SIG.

- Interfaz de Programación de Aplicaciones “Google Maps”
  Google Maps es el nombre de un servicio gratuito de Google. Es un servidor de aplicaciones de mapas en Web. Ofrece imágenes de mapas desplazables, así como fotos satelitales del mundo entero e incluso la ruta entre diferentes ubicaciones.

  La API es un conjunto de sentencias en javaScript que permiten hacer el uso de Google Maps en una aplicación propia con todas las siguientes características:

  - La capacidad de hacer acercamientos o alejamientos para mostrar el mapa.

- El usuario puede controlar el mapa con el mouse o las teclas de dirección para moverse a la ubicación que se desee.
- Para permitir un movimiento más rápido, las teclas + y - pueden ser usadas para controlar el nivel de zoom.

4.2.6. XML (*Extensible Markup Language*)

XML es un lenguaje que garantiza el intercambio de cualquier tipo de información, sin que ocasione problemas de tipo contenido o de tipo presentación. Este garantiza que los datos estructurados sean uniformes e independientes de aplicaciones o fabricantes, lo que está originando una nueva generación de aplicaciones en la Web.

Como su nombre lo indica, es un lenguaje de marcas que ofrece un formato para la descripción de datos estructurados, el cual conserva todas las propiedades importantes de SGML. Es decir, XML es un metalenguaje, dado que con él se puede definir nuestro propio lenguaje de presentación, a diferencia de HTML, que se centra en la representacón de la información. XML se centra en la información de sí misma. La particularidad más importante de XML es que no posee etiquetas prefijadas con anterioridad, ya que es el propio diseñador el que las crea a su antojo, dependiendo del contenido del documento.

XML se basa en tecnología desarrollada a partir de estándares probados y optimizada para la Web. La iniciativa XML consta de un conjunto de estándares relacionados entre sí:

- XML (Extensible Markup Language). Es una recomendación, que significa que el estándar es estable y que los desarrolladores de Web y de herramientas pueden adoptarlo plenamente.
- Namespaces. En XML es una recomendación que describe la sintaxis y la compatibilidad de los espacios de nombres para los intérpretes de XML.
- DOM (Document Object Model). Es una recomendación que ofrece un estándar para el acceso mediante programación a los datos estructurados (a través de scripts), de modo que los desarrolladores puedan interactuar de forma coherente con los datos basados en XML y computarlos.
- XSL (Extensible Stylesheet Language). XLS es la cara de presentación del XML. Este debe representar de forma independiente a la plataforma utilizada la información existente en los documentos XML.

4.2.7. KML(*Keyhole Markup Language*)

Es un lenguaje de marcado basado en XML para representar datos geográficos en tres dimensiones. Fue desarrollado para ser manejado con Keyhole LT, precursor de Google Earth. Su gramática contiene muchas similitudes con la de GML. Los ficheros KML a menudo suelen distribuirse comprimidos como ficheros KMZ.

Un fichero KML especifica una característica (un lugar, una imagen o un polígono) para Google Maps. Contiene título, una descripción básica del lugar, sus coordenadas(latitud y longitud) y alguna otra información.

KML es un formato de archivo que se utiliza para mostrar información geográfica en navegadores terrestres como Google Earth, Google Maps y Google Maps para móviles. KML utiliza una estructura basada en etiquetas con atributos y elementos anidados y está basado en el estándar XML. Un archivo kml puede contener desde un objeto con sus atributos, hasta tantos objetos como sean necesarios.

4.2.8. CDC (*Connected Device Configuration*)

Las configuraciones (CDC y CDLC[1]) definen el entorno mínimo necesario para la ejecución de aplicaciones java en un grupo amplio de dispositivos móviles (JVM/KVM+paquetes básicos). CDC están enfocados básicamente para PDAs de última generación o smartphones avanzados. Esta requiere una máquina virtual java compatible con J2SE 1.4.2 y los siguientes paquetes:

java.lang	javax.microedition.io	java.security.cer
java.lang.ref	java.util	
java.lang.reflect	java.util.zip	
java.math	java.util.jar	
java.text	java.net	
java.io	java.security	

CLDC (*Connected Limited Device Configuration*)

[1] *Connected Limited Device Configuration*

CLDC es la base para que los perfiles (como MIDP [2]) funcionen, proveyendo las apis básicas y la máquina virtual KVM [3]). CLDC está diseñada para equipos microprocesadores RISC o CISC de 16 a 32 bits y con una memoria mínima de 160 KB para la pila de la tecnología Java. Los dispositivos cuentan con potencia de cálculo, memoria, batería y acceso a red limitados.

Incluye una máquina virtual java reducida(KVM) y los siguientes paquetes:

java.lang	javax.microedition.io
java.lang.ref	java.util
java.lang.io	

4.2.9. MIDP (*Mobile Information Device profile*)

Los perfiles definen paquetes adicionales para soportar funcionalidades básicas imprescindibles de cada familia de dispositivos. Mobile Information Device Profile (MIDP) es el perfil más común con CLDC. MIDP 2.0 incorpora apis de interfaz de usuario, de ciclo de vida del programa, almacenamiento persistente, juegos, trabajo en red y multimedia. Según la especificación de la tecnología inalámbrica de Java todo dispositivo que soporte MIDP 2.0 debe incluir mínimamente las siguientes características:

- Debe permitir archivos Java (JAR [4]) de más de 64 KB. y archivos descriptores de aplicaciones (JAD) mayores a 5 KB.
- Se debe permitir a cada MIDlet la utilización de 30 KB de almacenamiento persistente y se recomienda que las MIDlets incluyan información acerca del almacenamiento mínimo con el que trabajan correctamente.
- El espacio de memoria libre para una aplicación ejecutándose (Heap o del montón) debe ser por lo menos de 256 KB.
- Soporte para pantallas de 125 x 125 pixeles, con una profundidad de color de 12 bits.

[2] *Mobile Information Device Profile*
[3] *Kilobyte Virtual Machine*
[4] *Java ARchive*

- Se deben incluir la capacidad de que el dispositivo reaccione a eventos de tiempo (una alarma a determinada hora, los llamados ticklers o despertadores).
- Soporte para imágenes en formato JPEG y PNG.
- Acceso a contenidos multimedia por el protocolo HTTP 1.1.

4.2.10. KVM (*Kilobyte Virtual Machine*)

Como ya se explicó anteriormente la máquina virtual es la base de la plataforma Java. En el caso de la plataforma J2ME, debido a las capacidades limitadas de almacenamiento, memoria, procesamiento y pantalla de los dispositivos; no se puede integrar una máquina virtual Java (JVM) del las dimensiones de J2SE o J2EE. Por esto se ha creado una nueva máquina virtual: KVM.

KVM es una máquina virtual Java compacta y portable específicamente diseñada para ser la base de desarrollo en dispositivos pequeños y de recursos limitados. Actualmente CLDC trabaja sobre KVM. Además KVM está diseñada para mantener los aspectos centrales del lenguaje Java ejecuándose en unos cuantos kilobytes de memoria (de ahí su nombre). Aunque KVM deriva de la máquina virtual J2SE (JVM), algunas características de esta última han sido eliminadas para soportar CLDC, debido a que resultan demasiado costosas de implementar o su presencia supone problemas de seguridad, resultando una KVM con las siguientes limitantes:

- Soporte de punto flotante.- KVM no soporta números de punto flotante, esto debido a que la mayoría de los dispositivos en los que se implementa no lo soportan tampoco.
- Finalización.- Las apis CLDC no incluyen el método object.finalize, así que no se pueden hacer operaciones de limpieza final antes de que el recolector de basura tome los objetos.
- Manejo de errores.- CLDC sólo define tres clases error: `java.lang.Error`, `java.lang.OutOfMemoryErr` y `java.lang.VirtualMachineError`. Todo tipo de errores que no sean en tiempo de ejecución se manejan de modo dependiente del dispositivo, esto incluye la finalización de una aplicación o reinicio del dispositivo.
- Interfaz Nativa Java (JNI).- No se implementa JNI (posibilidad de incluir código en C dentro de clases Java), primeramente por motivos de seguridad, aunque también es considerado excesivo dadas las limitantes de memoria del dispositivo al que se dirige.

4.3. Descripción de los modelos de diseños Web

Para crear nuestra aplicación web necesitamos conocer las diferentes formas existentes para conseguirlo, pero cumpliendo con los factores de una buena presentación, soporte y una mayor funcionalidad, por eso conoceremos el Modelo Vista Controlador y el modelo anterior a este.

4.3.1. Modelo 1

Si bien es cierto las aplicaciones Web en un inicio fueron simplemente CGIs (*Common Gateway Interface*) corriendo en un servidor Web con escasos recursos, pronto la gran demanda de las paginas dinámicas, hicieron que se idearan y construyeran aplicaciones Web mas complejas lo cual naturalmente incremento la carga en el servidor, anexado a las consultas, también cada vez mas complejas, a la base de datos, hicieron que se pensara por primera vez en la separación de ambos (presentación y acceso a datos).

El modelo 1 es aquel que separa la presentación del acceso a datos, en una aplicación Web en Java (JSP para este caso), se esquematizaría de la siguiente manera:

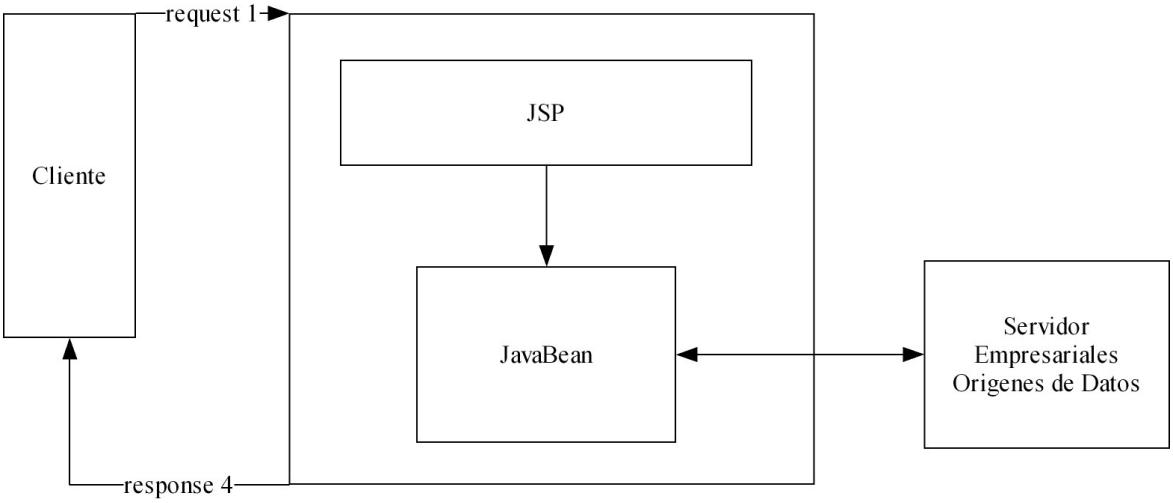


Figura 4.5: Esquema de Modelo 1

El Browser representa la carátula de la aplicación, mientras que las paginas JSP se encargan de recoger las peticiones del usuario y generan a la vez la respuesta que se le envía al mismo, lógica intermedia si se le puede llamar de algún modo, se puede notar que el acceso a datos ya esta manejado por un JavaBean.

4.3.2. Modelo 2 *MVC*

Modelo Vista Controlador (*MVC*) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

- Modelo: Esta es la representación específica de la información con la cual el sistema opera. La lógica de datos asegura la integridad de estos y permite todo derivar nuevos datos.
- Vista: Este presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.
- Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

Esta demás decir que en este modelo si ocurre una completa separación de la lógica, los datos y la presentación, hasta aquí el MVC, no viene a ser mas que un modelo lógico o arquitectura de diseño Web.

En la siguiente Figura. 4.6 se esquematiza la estructura del Modelo Vista Controlador:

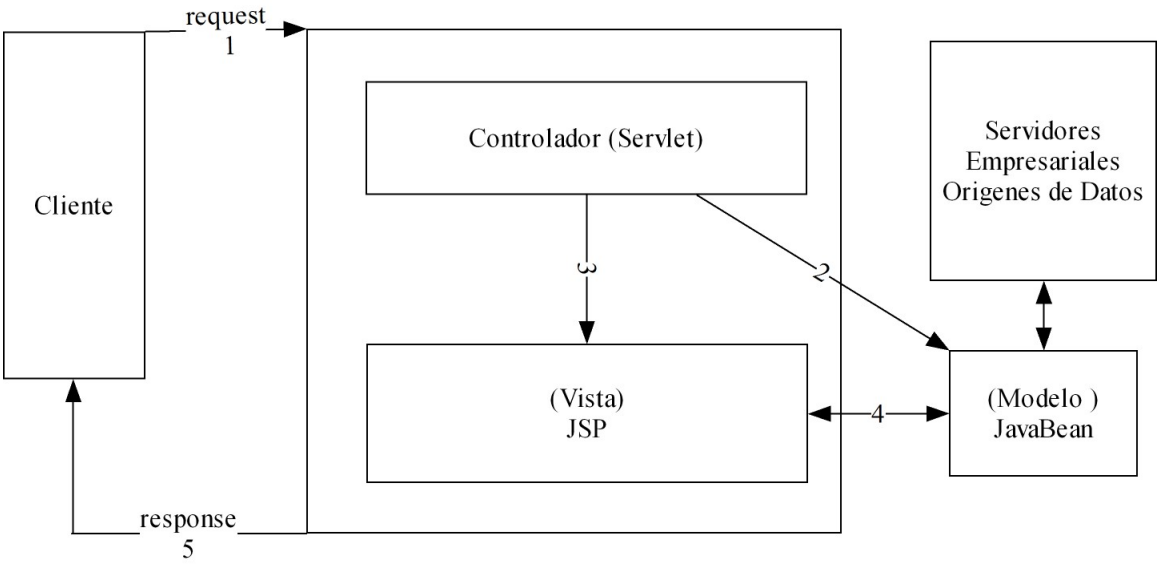


Figura 4.6: Esquema de Modelo 2

Para finalizar, en esta parte se ha mencionado los conceptos de las herramientas de software que se han investigado hasta el momento, por lo que nos resta mencionar la parte de algoritmos evolutivos, que en la siguiente página se presenta.

4.4. Introducción a la teoría de algoritmos genéticos

En la siguiente información se trata de explicar brevemente los conceptos que permitan entender el porque de un algoritmo evolutivo. Así como el encontrar un modelo de ruteo existente que pueda ayudarnos a resolver nuestras necesidades.

4.4.1. Conceptos Básicos

Los problemas cuya complejidad está acotada por polinomios son los denominados problemas *P*. Más detalladamente, podemos decir que un problema pertenece a la clase si puede ser resuelto en tiempo polinomial en una computadora determinista.

El término determinista significa que sin importar lo que haga el algoritmo, sólo hay una cosa que puede hacer a continuacióon (es decir, el paso siguiente se determina por los pasos anteriores).

Un problema pertenece a la clase NP si puede ser resuelto en tiempo polinomial pero usando una computadora no determinista.Cuando una computadora no determinista es confrontada con varias opciones, tiene el poder de "adivinar"la correcta (en caso de que ésta exista). Una computadora no determinista no hace nunca elecciones incorrectas que la hagan regresar a un estado previo. Obviamente, las computadoras no deterministas no existen en el mundo real. El no determinismo es una herramienta imaginaria que hace que los problemas difíciles parezcan triviales. Su mayor valía radica en el hecho de que existe forma de convertir un algoritmo no determinista a uno determinista, aunque a un costo computacional que suele ser muy elevado.

4.4.2. Heurística

Existen problemas que no pueden resolverse usando un algoritmo que requiera tiempo polinomial. De hecho, en muchas aplicaciones prácticas, no podemos siquiera decir si existe una solución eficiente. Asimismo, hay muchos problemas para los cuales el mejor algoritmo que se conoce requiere tiempo exponencial. Cuando enfrentamos espacios de búsqueda tan grandes , y que además los algoritmos más eficientes que existen para resolver el problema requieren tiempo exponencial, resulta obvio que las técnicas clásicas de búsqueda y optimización son insuficientes. Es entonces cuando recurrimos a las *"heurísticas"*. La palabra *heurística* se deriva del griego heuriskein, que significa *encontrar* o *descubrir*.

Las heurísticas fueron un área predominante en los orígenes de la Inteligencia Artificial. Actualmente, el término suele usarse como un adjetivo, refiriéndose a cualquier técnica que mejore el desempeño en promedio de la solucin de un problema, aunque no mejore necesariamente el desempeño en el peor caso. Una definición más precisa y adecuada sería la siguiente:

Una heurística es una técnica que busca soluciones buenas (es decir, casi óptimas) a un costo computacional razonable, aunque sin garantizar factibilidad u optimalidad de las mismas. En algunos casos, ni siquiera puede determinar qué tan cerca del óptimo se encuentra una solución factible en particular.

4.4.3. Algoritmos Evolutivos

El término computación evolutiva o algoritmos evolutivos, realmente engloba una serie de técnicas inspiradas biológicamente (en los principios de la teoría Neo-Darwiniana de la evolución natural). En términos generales, para simular el proceso evolutivo en una computadora se requiere:

- Codificar las estructuras que se replicarán.
- Operaciones que afecten a los individuos.
- Una función de aptitud.
- Un mecanismo de selección.

Aunque hoy en día es cada vez más difícil distinguir las diferencias entre los distintos tipos de algoritmos evolutivos existentes, por razones sobre todo históricas, suele hablarse de tres paradigmas principales:

- Programación Evolutiva
- Estrategias Evolutivas
- Algoritmos Genéticos.

Cada uno de estos paradigmas se originó de manera independiente y con motivaciones muy distintas.

4.4.4. Programación Evolutiva

Lawrence J. Fogel propuso en los 1960s una técnica denominada "programación evolutiva", en la cual la inteligencia se ve como un comportamiento adaptativo.

La programación evolutiva enfatiza los nexos de comportamiento entre padres e hijos, en vez de buscar emular operadores genéticos específicos (como en el caso de los algoritmos gennéticos).

El algoritmo básico de la programación evolutiva es el siguiente:

- Generar aleatoriamente una población inicial.
- Se aplica mutación.
- Se calcula la aptitud de cada hijo y se usa un proceso de seleccion mediante torneo(normalmente estocástico) para determinar cuáles serán las soluciones que se retendrán.

4.4.5. Estrategias Evolutivas

Las estrategias evolutivas fueron desarrolladas en 1964 en Alemania para resolver problemas hidrodinámicos de alto grado de complejidad por un grupo de estudiantes de ingeniería encabezado por Ingo Rechenberg.

El algoritmo usa un solo padre y con él se genera un solo hijo. Este hijo se mantenía si era mejor que el padre, o de lo contrario se eliminaba (a este tipo de selección se le llama extintiva, porque los peores individuos obtienen una probabilidad de ser seleccionado de cero).

4.4.6. Estrategias Evolutivas vs Programación Evolutiva

La Programación Evolutiva usa normalmente selección estocástica, mientras que las estrategias evolutivas usan selección determinística. Ambas técnicas operan a nivel fenotípico (es decir, no requieren codificación de las variables del problema). La programación evolutiva es una abstracción de la evolución al nivel de las especies, por lo que no se requiere el uso de un operador de recombinación (diferentes especies no se pueden cruzar entre sí). En contraste, las estrategias evolutivas son una abstracción de la evolución al nivel de un individuo, por lo que la recombinación es posible.

4.4.7. Algoritmos Genéticos

Los algoritmos genéticos fueron desarrollados por John H. Holland a principios de los 1960s. Estos algoritmos usan selección probabilística al igual que la Programación Evolutiva, y en contraposición a la selección determinística de las Estrategias Evolutivas. El algoritmo genético enfatiza la importancia de la cruza sexual (operador principal) sobre el de la mutación (operador secundario), y usa selección probabilística.

El algoritmo básico es el siguiente:

- Generar (aleatoriamente) una población inicial.
- Calcular aptitud de cada individuo.
- Seleccionar (probabilísticamente) en base a aptitud.
- Aplicar operadores genéticos (cruza y mutación) para generar la siguiente población.
- Ciclar hasta que cierta condición se satisfaga.

Los algoritmos genéticos usan la representación binaria para codificar las soluciones a un problema, por lo cual se evoluciona el genotipo y no el fenotipo como en la Programación Evolutiva o las Estrategias Evolutivas.

El operador principal en los algoritmos genéticos es la cruza, y la mutación es un operador secundario. En la Programación Evolutiva, no hay cruza y en las Estrategias Evolutivas es un operador secundario. Ha sido demostrado que los algoritmos evolutivos requieren de elitismo (o sea, retener intacto al mejor individuo de cada generación) para poder converger al óptimo.

El proceso de un algoritmo genético básico es el que se muestra en la Figura 4.7. El proceso en el que cada vez que se iteran los pasos: Evaluación - Selección - Operadores Genéticos es llamado una generación y se itera creando nuevas generaciones hasta que el criterio de terminación es cumplido. La representación de los individuos (soluciones) se hace mediante cadenas de genes llamadas cromosomas en donde se define claramente la solución, un aspecto muy importante en algoritmos genéticos es definir como se van a representar las soluciones en estos cromosomas ya que esto es lo que va a permitir un proceso de evaluación mas sencillo y también permitirá conseguir mejores resultados. A continuación se describen brevemente cada uno de los pasos mostrados en la Figura 4.7.

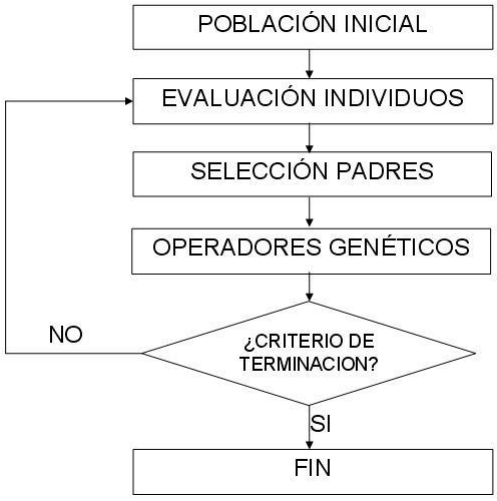


Figura 4.7: Proceso de un algoritmo genético básico

- Población Inicial: En este paso se crean un número determinado de individuos aleatoriamente o también se pueden crear de una manera estructurada con la que se va a obtener una mejor población inicial. En muchos de los problemas en los que se usan algoritmos genŽeticos se usa un tamaño de población de 100 individuos, sin embargo esta cantidad ser diferente.

- Evaluación de los Individuos: Aquí es donde se mide que tan buenos son los individuos que hay en la población. En este paso es importante que la función de evaluación tenga en cuenta todos los aspectos que harían bueno o malo a un individuo.

- Operadores Genéticos: Después del proceso de selección, se aplican generalmente dos operadores genéticos los cuales son: cruce y mutación. El cruce nos va a permitir crear posibles mejores nuevos individuos que sean producto de los buenos padres seleccionados en el paso anterior, se define una probabilidad de cruce para establecer si en cierta generación se realiza el cruce. La mutación es aplicada generalmente a cada individuo después del cruce, también se define una probabilidad de mutación.

- Criterio de Terminación: El criterio de terminación generalmente es un número dado de iteraciones(generaciones),o en algunos casos cierto valor de aptitud que alcance algún individuo.

4.4.8. Diferencias y ventajas de las técnicas evolutivas con respecto a las tradicionales

Existen varias diferencias que vale la pena destacar entre los algoritmos evolutivos y las técnicas tradicionales de búsqueda y optimización:

- Las técnicas evolutivas usan una población de soluciones potenciales en vez de un solo individuo, lo cual las hace menos sensibles a quedar atrapadas en mńimos/máximos locales.
- Las técnicas evolutivas no necesitan conocimiento específico sobre el problema que intentan resolver.
- Las técnicas evolutivas usan operadores probabilísticos,mientras las técnicas tradicionales utilizan operadores determinísticos.
- Aunque las técnicas evolutivas son estocásticas, el hecho de que usen operadores probabilísticos no significa que operen de manera análoga a una simple búsqueda aleatoria.

Es importante destacar las diversas ventajas que presenta el uso de técnicas evolutivas para resolver problemas de búsqueda y optimización:

- Simplicidad Conceptual.
- Amplia aplicabilidad.
- Superiores a las técnicas tradicionales en muchos problemas del mundo real.
- Tienen el potencial para incorporar conocimiento sobre el dominio y para hibridizarse con otras técnicas de búsqueda/optimización.
- Pueden explotar fácilmente las arquitecturas en paralelo.
- Son robustas a los cambios dinámicos.
- Generalmente pueden auto-adaptar sus parámetros.
- Capaces de resolver problemas para los cuales no se conoce solución alguna.

4.4.9. Problema del agente viajero *Travelling Salesman Problem - (TSP)*

En éste modelo el objetivo es encontrar el viaje más corto pasando por un grupo de ciudades, visitando cada una de ellas solamente una vez y regresar al punto donde se inicio el viaje.

A lo largo de los años el problema del agente viajero ha ocupado la mente de nu- merosos investigadores. Los motivos son varios. En primer lugar, el TSP es un prob- lema muy sencillo de enunciar, pero muy difícil de resolver. En segundo lugar, el TSP es aplicable a una gran variedad de problemas de planificación. Finalmente, se ha convertido en una especie de problema test, es decir los nuevos métodos de opti- mización combinatoria son a menudo aplicados al TSP con objeto de tener una idea de sus potencialidades.

4.4.10. *Vehicle Route Problem* (VRP)

Los problemas de rutas de vehículos (VRP) en realidad son un amplio conjunto de variantes y personalizaciones de problemas. Desde los que son más sencillos hasta algunos que hoy en día siguen siendo materia de investigación.

En ellos en general, se trata de averiguar las rutas de una flota de transporte para dar servicio a unos clientes. Este tipo de problemas pertenece a los problemas de optimización combinatoria. En la literatura científica los primeros autores en 1959, estudiaban la aplicación real en la distribución de gasolina para estaciones de carburante.

La función objetivo depende de la topología y características del problema. Lo más habitual es intentar: minimizar el coste total de operación, minimizar el tiempo total de transporte, minimizar la distancia total recorrida, minimizar el tiempo de espera, maximizar el beneficio, maximizar el servicio al cliente, minimizar la utilización de vehículos, equilibrar la utilización de los recursos, etc. Existen una gran variedad de tipos de problemas de ruteo de vehículos. A continuación se enumeran los más conocidos:

- *Vehicle Routing Problem with Time windows(VRPTW)*

  En éste modelo un número de vehículos están localizados en un almacén central y tienen que servir a un grupo de clientes que están dispersos geográficamente. Cada vehículo tiene una capacidad dada, así como el cliente tiene una demanda y éste tiene que ser atendido en un tiempo determinado.

- *Pick-up and Delivery Problem with Time Windows(PDPTW)* En éste modelo un número de vehículos tiene que atender a un número de peticiones para transporte. Cada vehículo tiene una capacidad. Cada petición específica el tamaño de la carga que será transportada, la locación donde debe ser adquirida mejora el tiempo y la posición de entrega mejora el tiempo de entrega.

Parte III

Diseño

CAPÍTULO 5

Diseño

En esta parte se incluye todos los diagramas que consideramos necesario para poder realizar nuestro proyecto. Estos diagramas son los siguientes:

- Diagrama de paquetes
- Diagrama de clases
- Diagrama de estados
- Diagrama de secuencias
- Diagrama de objetos

5.1. Diagrama de paquetes

Nuestro proyecto lo hemos dividido en tres módulos, Servidor Web, Algoritmo Evolutivo y Teléfono celular, cada módulo estará comunicado de forma bidireccional, donde compartirán información para que cada uno realice sus funciones principales. A continuación se muestra el diagrama correspondiente, denotando los flujos de comunicación que existirá entre ellos.

- Diagrama de paquetes.

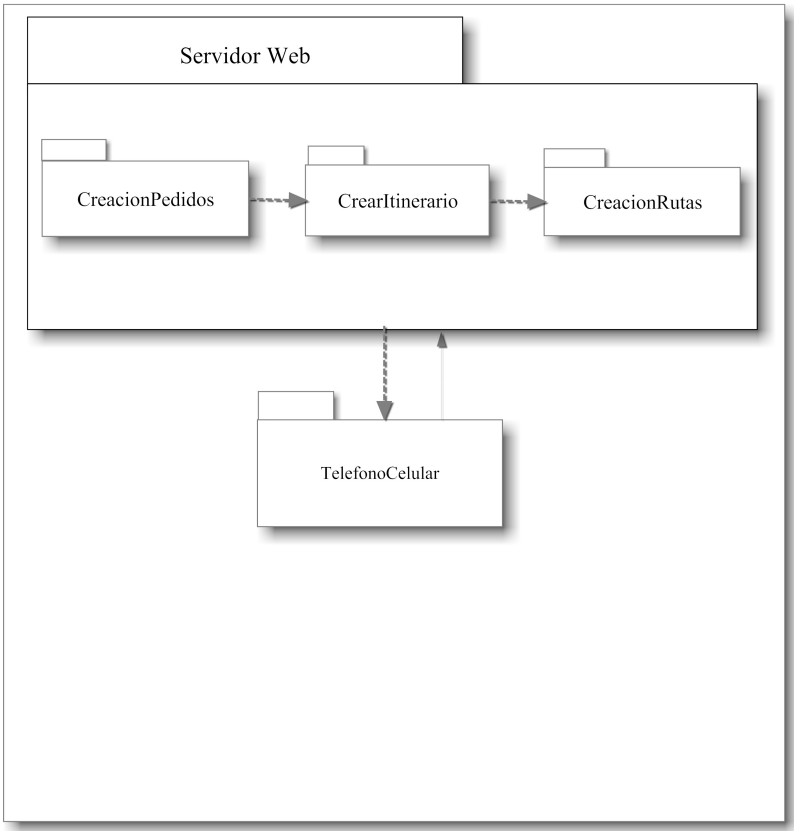


Figura 5.1: Diagrama de paquetes del Sistema

5.2. Diseño de casos de uso

Éste análisis nos ayudó a conocer los procedimientos y el ambiente externo, o sea los factores externos que participan en los procesos.

- Diagrama general: En la Fig. 5.2 se muestra el diagrama de caso de uso que corresponde a nuestro sistema.

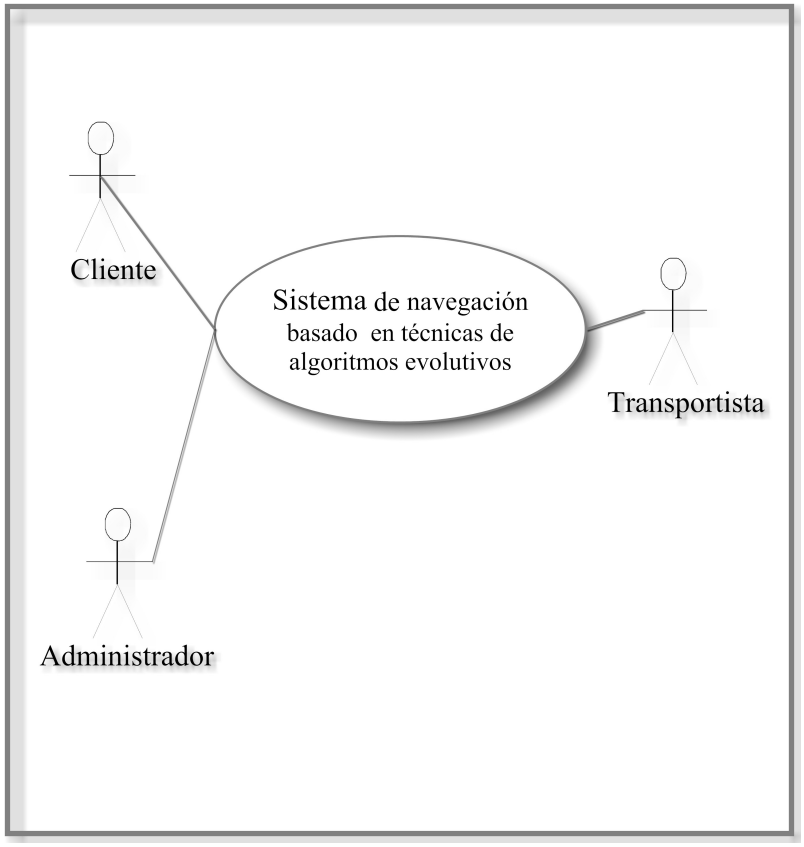


Figura 5.2: Diagrama de caso de uso de sistema general

Descripción de caso de uso Diagrama General

Nivel: 0

Actor Principal:

Cliente, Administrador, Teléfono celular

Personal involucrado e intereses:

Cliente: A quien se le desea satisfacer.

Administrador: Está al tanto del correcto funcionamiento del sistema.

Teléfono celular: Hará uso de las rutas obtenidas.

Precondiciones:

- El usuario deberá registrarse en el Sistema
- El transportista deberá contar con un teléfono celular con conexión a Internet

Escenario principal de éxito:

- Se genera itinerario y rutas de pedidos.

Requisitos especiales:

- Exista batería del móvil.
- Exista cobertura de redes celulares.

Frecuencia:

- Cuando se ingresa por primera vez al sistema.

Tabla 5.1: Descripción de caso de uso Escenario principal

- Diagrama de caso de uso de Cliente: En la Figura. 5.3 se muestra el diagrama de caso de uso que corresponde a las actividades que puede realizar el cliente en nuestro sistema.

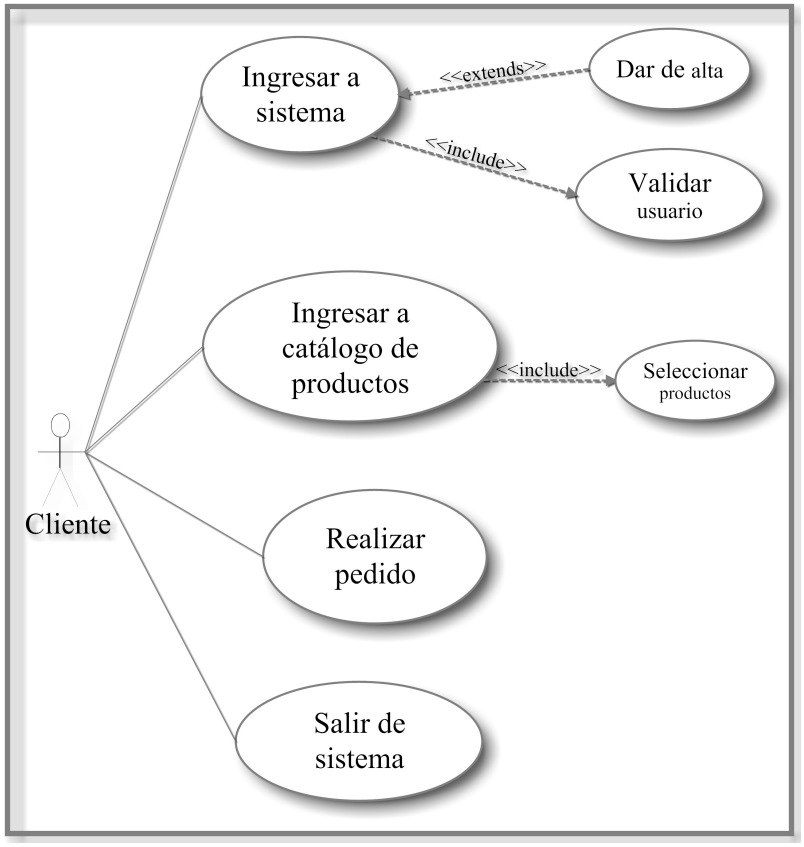


Figura 5.3: Diagrama de caso de uso de Cliente

Descripción de caso de uso Cliente

Nivel: 0

Actor Principal:

Cliente

Personal involucrado e intereses:

Cliente: quien hará uso del sistema.

Precondiciones:

- El usuario deberá registrarse en el Sistema.
- Ingreso de toda la información que le solicite el sistema.

Escenario principal de éxito:

- Ingresa al sistema exitosamente.
- Realiza el pedido.

Requisitos especiales:

- Conexión a Internet.

Frecuencia:

- Cuando se ingresa por primera vez al sistema.

Tabla 5.2: Descripción de caso de uso Cliente

- Diagrama de caso de uso de Administrador: En la Figura. 5.4 se muestra el diagrama de caso de uso que corresponde a las actividades que puede realizar el administrador, como su nombre lo indica, administrar los usuarios y los pedidos, para evitar duplicidad de datos y abusos por parte de los clientes .

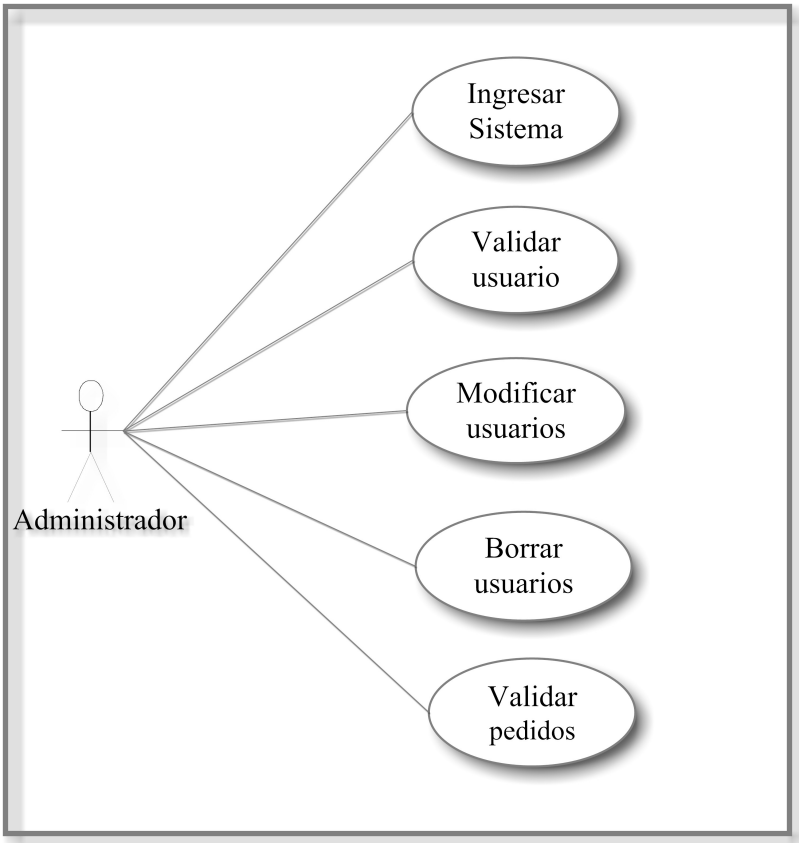


Figura 5.4: Diagrama de caso de uso de Cliente

Descripción de caso de uso Administrador

Nivel: 0

Actor Principal:

Administrador

Personal involucrado e intereses:

Administrador: quien podrá realizar tareas de validación de usuarios y de pedidos del sistema.

Precondiciones:

- Haberse registrado en el sistema como administrador.
- Tener privilegios de administrador.

Escenario principal de éxito:

- Ingresa al sistema exitosamente.
- Validar usuarios.
- Validar pedidos.
- Validar productos.

Requisitos especiales:

- Conexión a Internet.
- Tener privilegios de administrador.

Frecuencia:

- Cada vez que se ingrese al sistema.

Tabla 5.3: Descripción de caso de uso Administrador

- Diagrama de caso de uso de Sistema: En la Figura. 5.5 se muestra el diagrama de caso de uso de las principales actividades que puede realizar el sistema.

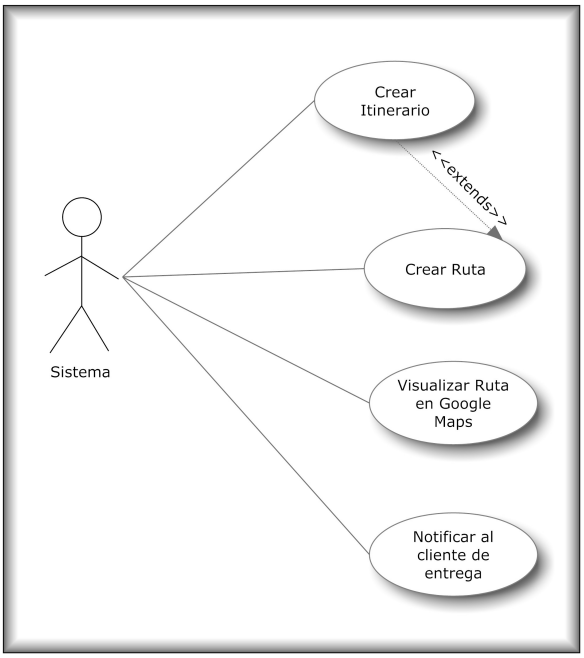


Figura 5.5: Diagrama de caso de uso de Sistema

Descripción de caso de uso Sistema

Nivel: 0

Actor Principal:

Administrador, Cliente y Transportista

Personal involucrado e intereses:

Administrador: quien podrá realizar la tarea de invocar la aplicación que genera la ruta.

Transportista: quien podrá visualizar la ruta mediante un mapa. Cliente: quien podrá consultar el estatus de su compra.

Precondiciones:

- Haber finalizado la lista de pedidos del día

Escenario principal de éxito:

- Ingresa al sistema exitosamente.
- Generar correctamente la ruta.
- Generar correctamente itinerario

Requisitos especiales:

- Conexión a Internet.

Frecuencia:

- Al finalizar los pedidos del día

Tabla 5.4: Descripción de caso de uso Sistema

- Diagrama de caso de uso de Generador de rutas: En la Figura. 5.6 se muestra el diagrama de caso de uso de las principales actividades que puede realizar el sistema.

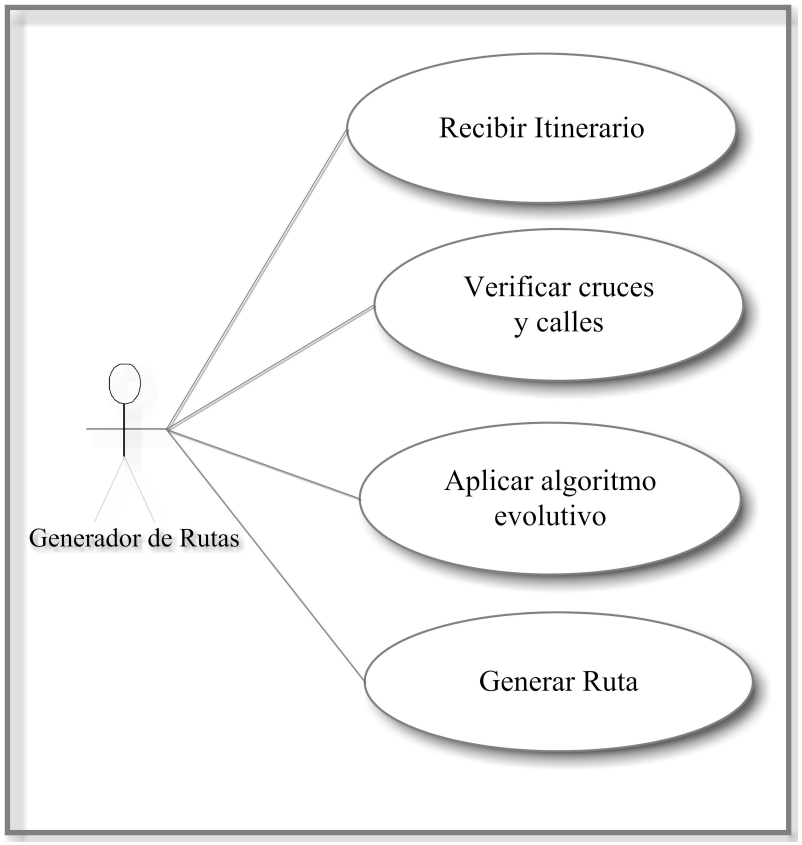


Figura 5.6: Diagrama de caso de uso de Generador de rutas

Descripción de caso de uso Generador de rutas

Nivel: 0

Actor Principal:

Administrador, Algoritmo Evolutivo

Personal involucrado e intereses:

Administrador: quien podrá realizar las tareas de cargar las aplicaciones de creación de itinerario y ruta.

Algoritmo evolutivo: quién es el encargado de generar el itinerario y la ruta.

Precondiciones:

- Haber obtenido toda la información necesaria de las calles,sentidos, cruces y retornos.
- Recibir todos los pedidos del día.

Escenario principal de éxito:

- Ingresa al sistema exitosamente.
- Generar ruta correctamente.

Requisitos especiales:

- Tener privilegios de administrador.

Frecuencia:

- Cada vez que se desea generar ruta.

Tabla 5.5: Descripción de caso de uso Generador rutas

Descripción de caso de uso Celular

Nivel: 0

Actor Principal:

Dispositivo Móvil

Personal involucrado e intereses:

Transportista: quien podrá realizar visualizar la ruta por medio de su dispositivo móvil.

Dispositivo móvil: quién es el encargado de desplegar el mapa con la ruta.

Precondiciones:

- Conexión internet

Escenario principal de éxito:

- Visualización de la ruta correctamente.

Requisitos especiales:

- Batería dispositivo móvil.

Frecuencia:

- Cada vez que se desea visualizar la ruta.

Tabla 5.6: Descripción de caso de uso Celular

- Diagrama de caso de uso de Teléfono celular: En la Figura. 5.7 se muestra el diagrama de caso de uso de las principales actividades que puede realizar el sistema.

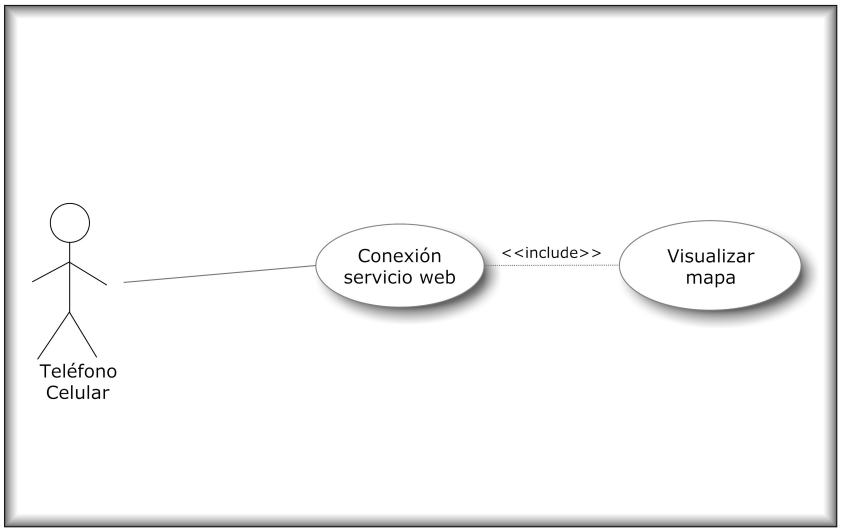


Figura 5.7: Diagrama de caso de uso de Celular

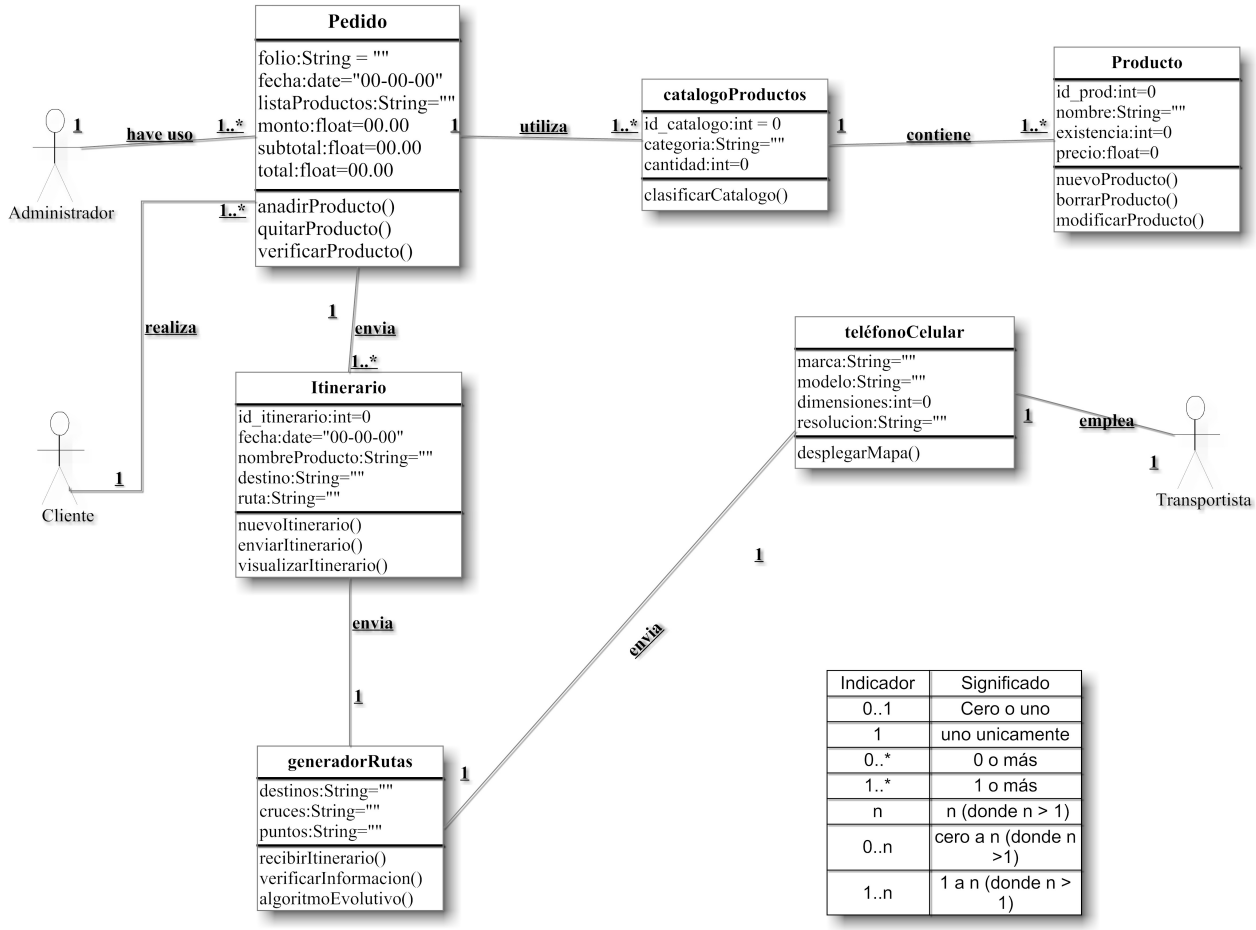
5.3. Diagrama de clases

El diagrama de Clase muestra los bloques de construcción de cualquier sistema orientado a objetos. Los diagramas de clases describen la vista estática del modelo o parte del modelo, describiendo que atributos y comportamientos tienen en lugar de detallar los métodos para realizar operaciones. Los diagramas de clases son más útiles para ilustrar relaciones entre clases e interfaces. Las generalizaciones, agregaciones, y asociaciones son todas valiosas al reflejar herencias, composición o uso, y conexiones respectivamente.

- Cliente: contiene los atributos y métodos de inserción, modificación y eliminación de clientes. Su propósito es poder identificar al usuario de tipo Cliente para que pueda ingresar al sistema, su cardinalidad va de uno a varios Clientes.
- Pedido: clase que permite al usuario realizar pedidos donde añadirá productos, su cardinalidad es de 1 a * ya que un Cliente puede realizar varios pedidos cada vez que ingresa al sistema.
- Catálogo de productos: aquí es donde se ordenaran los productos por categoría, lo que le permitirá al cliente tener un mayor control sobre sus peticiones.
- Producto: en esta clase contiene el registro de los productos que están disponibles para su venta, así como algunos atributos de los productos, como nombre, existencia, precio, etc.
- Itinerario: contiene el itinerario de entregas del día, este itinerario se generará uno por día para poder determinar la ruta que incluya todos los pedidos del mismo día.
- Reporte: esta clase se generará una vez que el itinerario se crea, con el fin de informar al transportista sobre los destinos a los que debe entregar.
- Generador de rutas: considerada una clase de suma importancia, para garantizar las entregas de la mejor forma, esta clase recibirá el itinerario, donde posteriormente analizará la información recibida y determinará la ruta completa.
- Teléfono celular: podrá visualizar el mapa teniendo conexión a internet.

En la Fig. 5.8 se muestra el diagrama general del sistema

- Diagrama de clases del sistema general



Indicador	Significado
0..1	Cero o uno
1	uno unicamente
0..*	0 o más
1..*	1 o más
n	n (donde n > 1)
0..n	cero a n (donde n >1)
1..n	1 a n (donde n > 1)

Figura 5.8: Diagrama de clases del sistema general

5.3.1. Agregación

Las agregaciones se usan para describir elementos que están compuestos de componentes más pequeños. Donde destacamos la importancia que tiene cada componente, ya que el Sistema esta ligado a todos los módulos por lo que no se puede descartar alguno de ellos, ya que implicaría una falla en el Sistema.

El Sistema General esta conformado por 2 módulos que son indispensables para su funcionamiento ya que entre ellos se comunican, en la Fig. 5.9 estos módulos se describen a continuación:

- Diagrama de clases de agregación

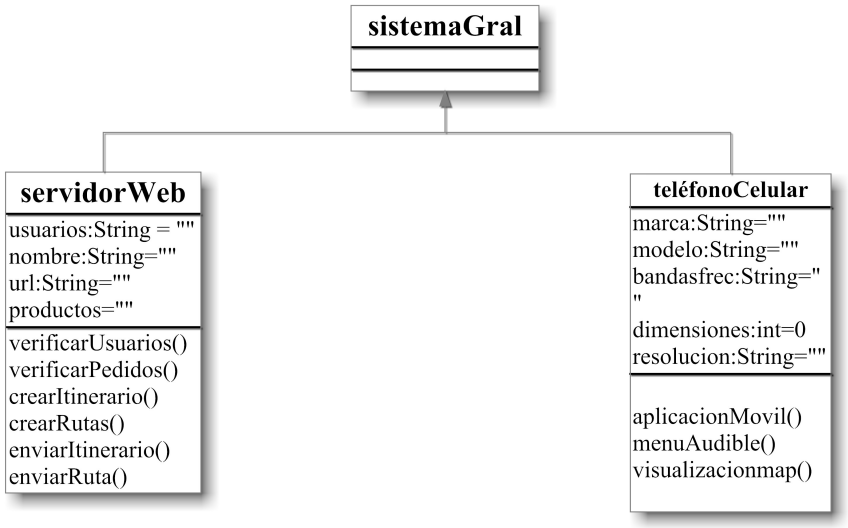


Figura 5.9: Diagrama de clases de agregación

- ServidorWeb: se encarga de procesar las solicitudes de los clientes, y los pedidos, además de recolectar información para los siguientes módulos.

- Teléfono celular: Desplegar la ruta a recorrer e interactuar con el transportista.

5.3.2. Generalizaciones

Usarlo para indicar herencia. Dibujada desde un clasificador especifico a un clasificador general, la implicación general es que el origen hereda las características del destino. En la Fig. 5.10 siguiente diagrama muestra una clase padre generalizando tres clases hijo. Implícitamente, un objeto instanci-

ado de la clase Administrador, Cliente y Transportistas tendrá atributos de la clase Usuario, ya que puede existir de dos tipos, con diferentes privilegios para acceder al sistema.

- Diagrama de clases de generalización

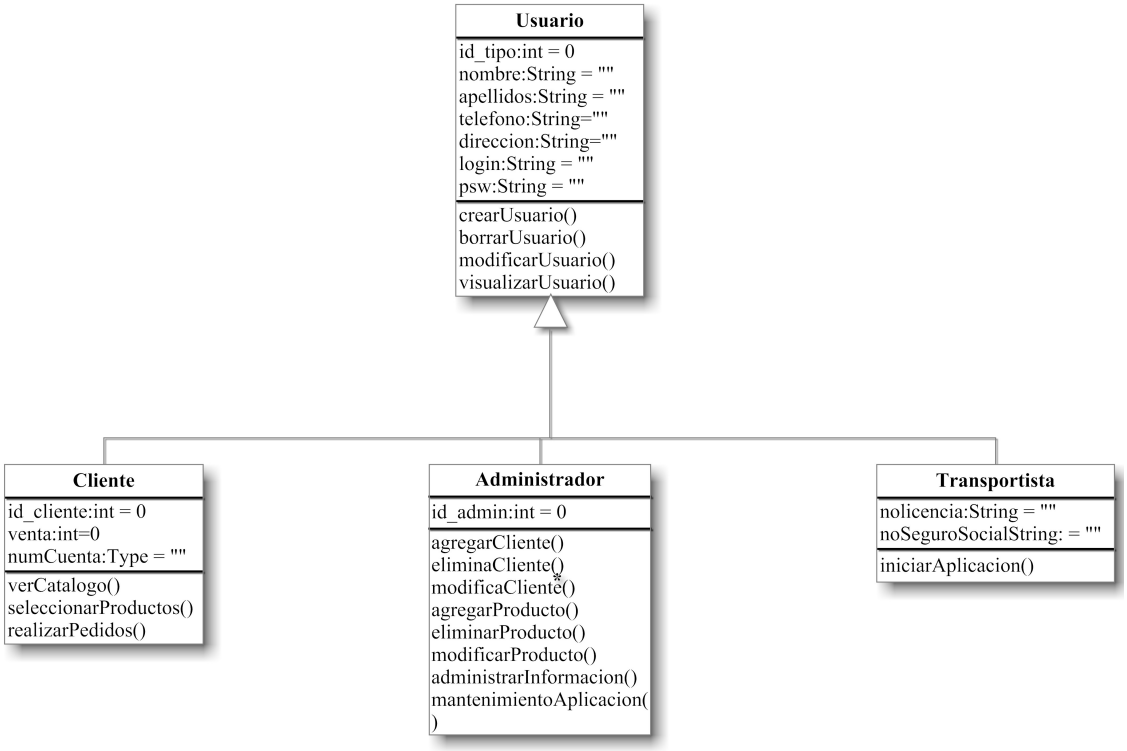


Figura 5.10: Diagrama de clases de generalización

5.4. Diagrama de objetos

Este diagrama nos sirve para modelar las instancias de elementos contenidos en los diagramas de clases. Además de mostrar un conjunto de objetos y sus relaciones en un momento concreto

A continuación en la Fig. 5.11, observamos las instancias principales de los objetos del sistema, y representamos las relaciones con otros objetos de diferentes clases.

- Diagrama de clases de objetos

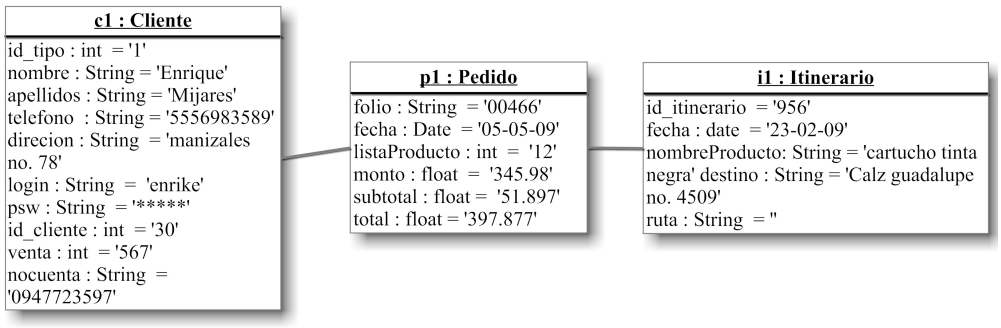


Figura 5.11: Diagrama de objetos

5.5. Diagrama de secuencias

En esta sección se muestra más a detalle las funciones de Cliente y de la creación de un pedido, con el propósito de dar un panorama general de la secuencia de las actividades que pueden realizar cada uno de los mencionados anteriormente.

- Cliente: En la Fig. 5.12, el primer paso para el Cliente es entrar al sistema para poder visualizar el catálogo de productos, seleccionar los productos de su elección para finalmente crear el pedido introduciendo los datos que el sistema le requerirá.

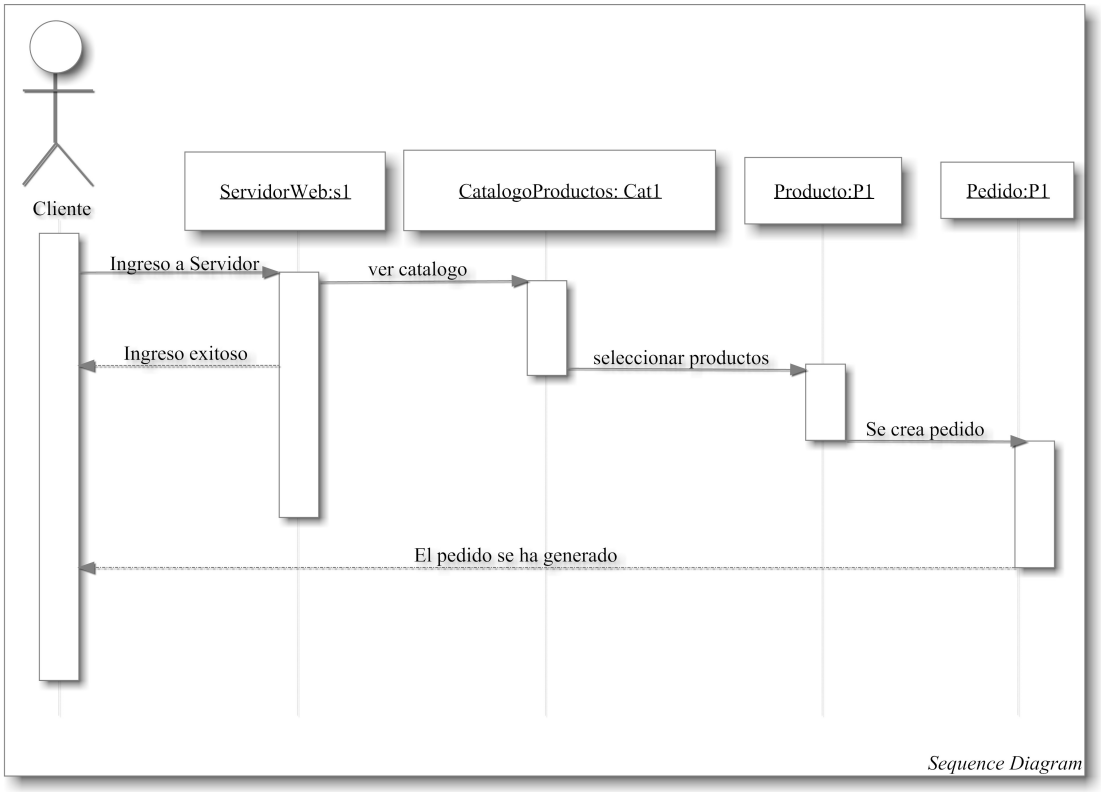


Figura 5.12: Diagrama de secuencias del cliente

- Pedido: En el diagrama anterior se indica la creación del pedido, en esta parte se (Fig. 5.13) se muestra que cada pedido formará parte de un itinerario, además, el módulo de la ruta recibirá los parámetros que éste necesite para obtener la ruta para que finalmente pueda ser visualizada en un mapa

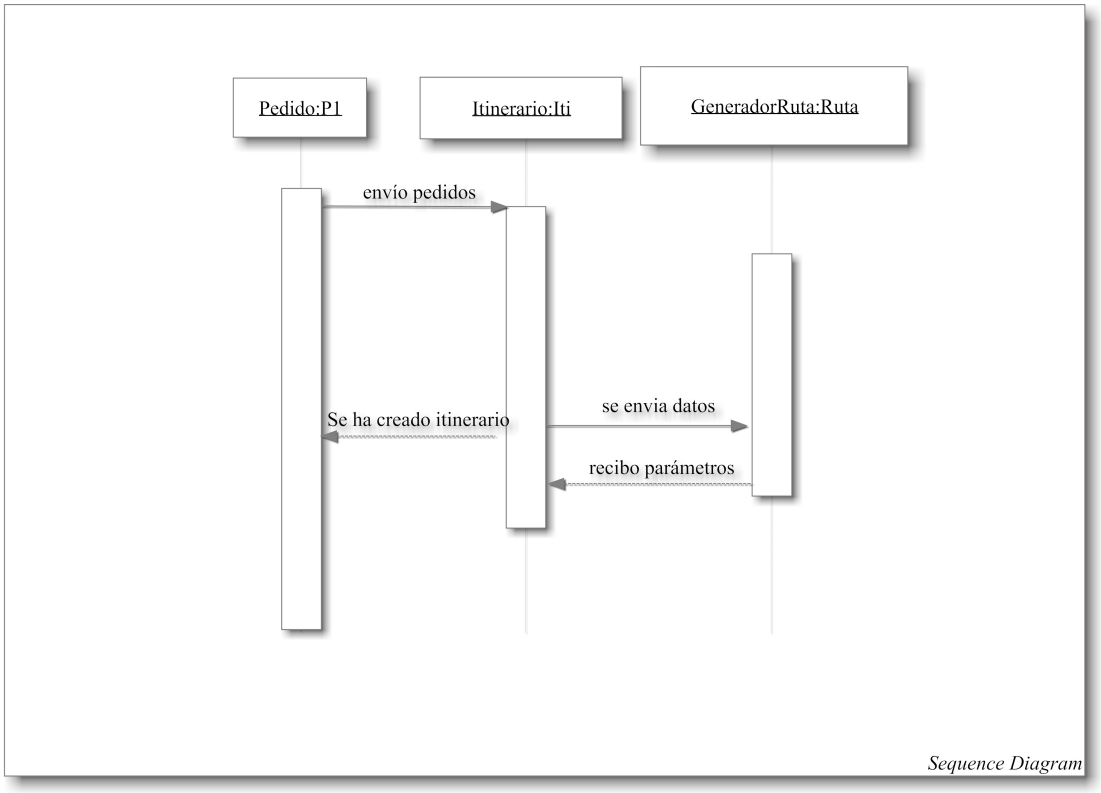


Figura 5.13: Diagrama de secuencias de pedido

- En el diagrama de secuencia de Módulo (Fig. 5.14), se ve la interacción del Dispositivo Móvil con el transportista, ya que el dispositivo móvil le muestra la ruta que debe de seguir. El transportista podra visualizar la ruta desplegada en el mapa.

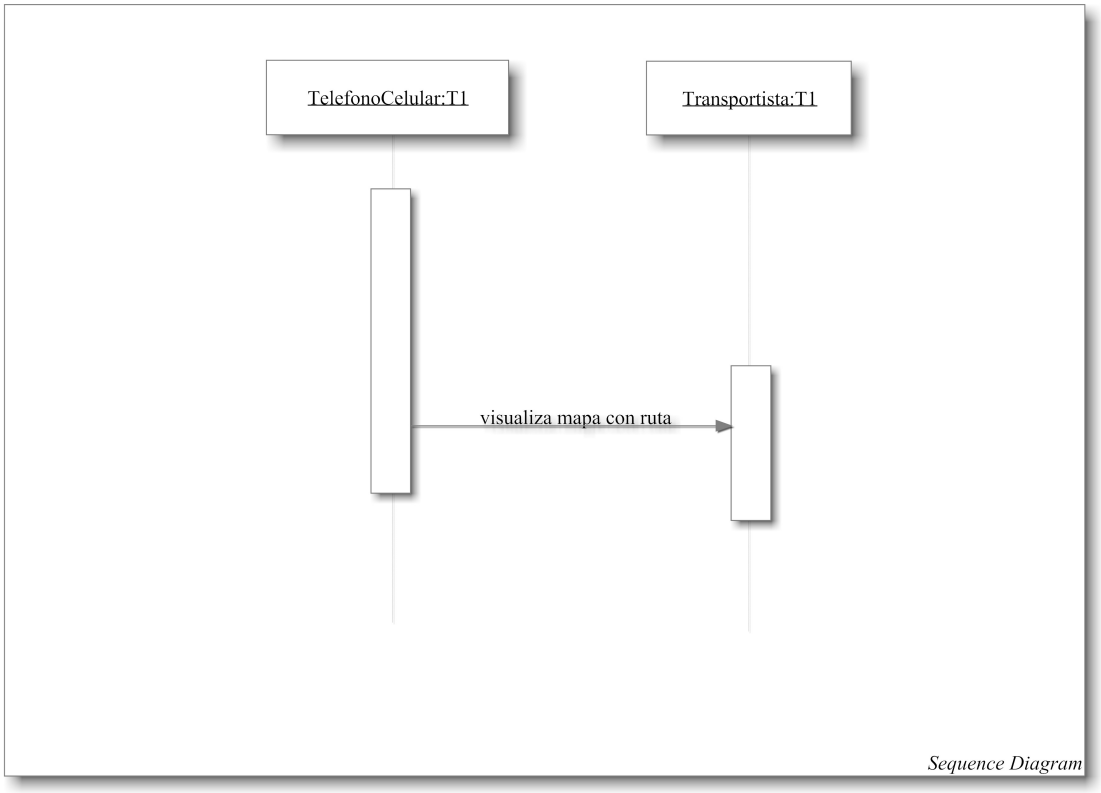


Figura 5.14: Diagrama de secuencias de módulo

5.6. Modelo de Base de Datos

5.6.1. Modelo Entidad-Relación

Para el modelo entidad-relación se crearon seis entidades donde se describe los atributos que nos interesa de cada uno de ellos. Los usuarios pueden ser tanto administradores como clientes, y tienen datos que son iguales para ambos, pero con relación al cliente se necesita saber un identificador que relacione la compra que hizo un número de cuenta y un identificador de cliente. Se contará con una tabla que contenga la información acerca de las ventas realizadas, a su vez tendremos una tabla para los productos que podrán ser seleccionados por el usuario, estos productos pertenecen a un catalogo determinado. Lo antes mencionado se muestra en la Fig. 5.15.

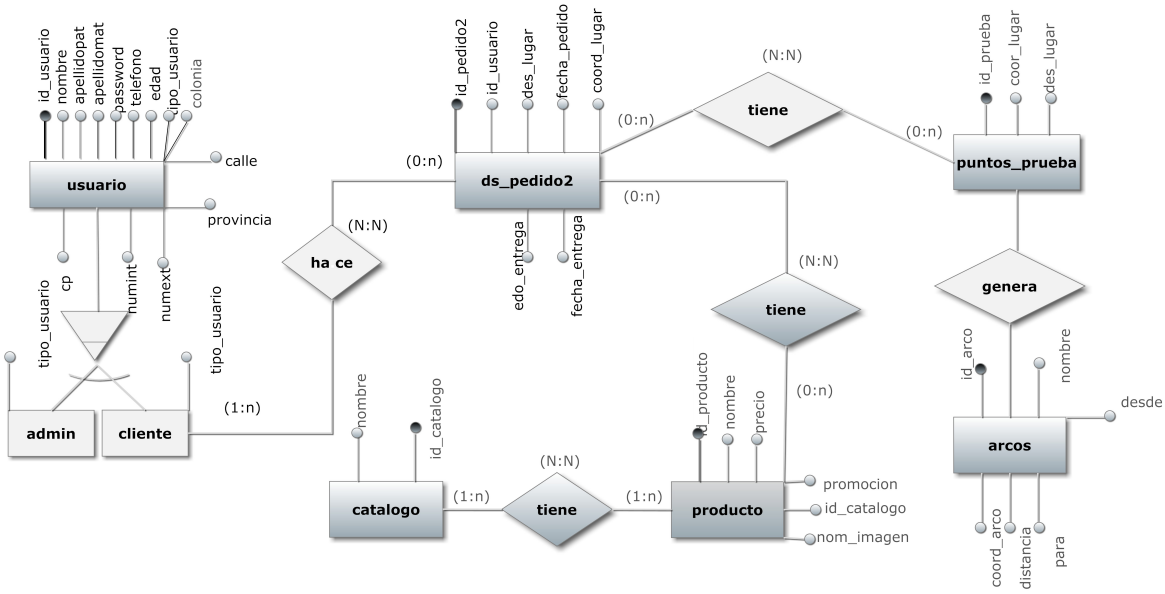


Figura 5.15: Modelo entidad - relación

5.6.2. Modelo relacional

La siguiente figura 5.16 presenta el modelo relacional con respecto al diagrama entidad – relacion antes mencionado.

Modelo relacional

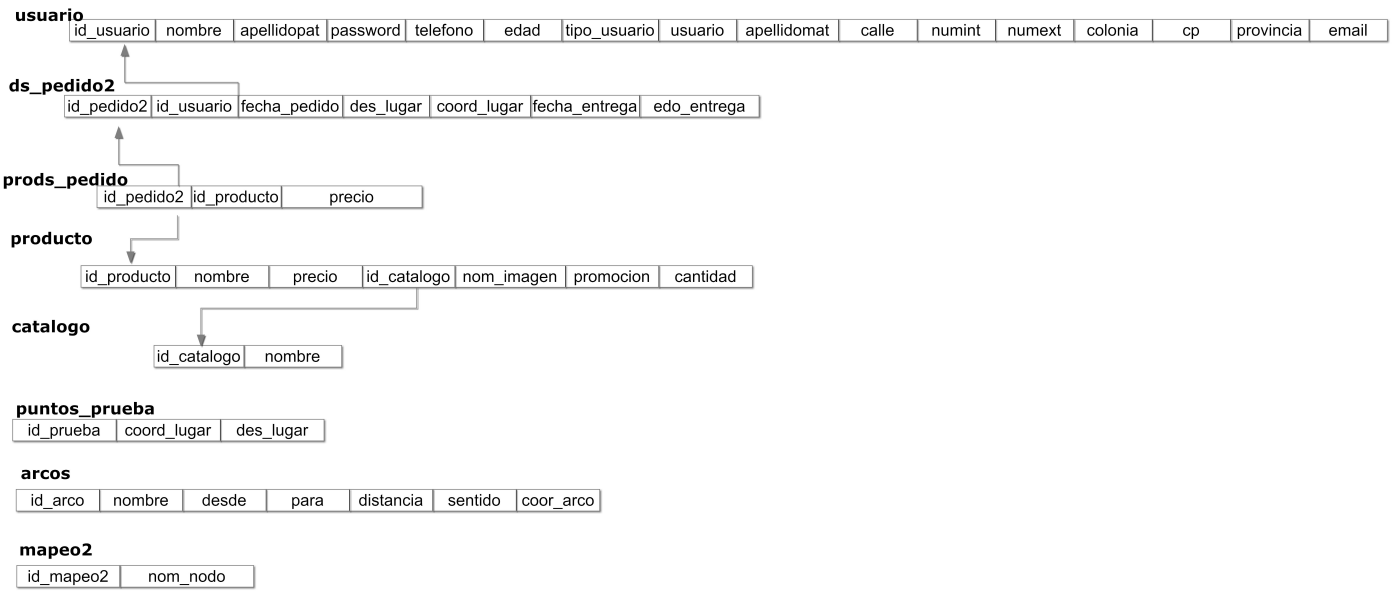


Figura 5.16: Modelo relacional de la base de datos

5.7. Herramientas utilizadas

En la presente sección se explicarán las herramientas que se utilizaron para el desarrollo del proyecto dando a conocer la importancia de éstas y sus características

5.8. Manejador de base de datos

Se ha elegido utilizar PostgreSQL por razones que nosotros consideramos sustentadas por la información de su desempeño mostrándose una tabla (Tabla 5.7) comparativa de los manejadores de bases de datos que se mencionarón anteriormente y que a continuación se presenta.

SGBD	Sistema Operativo	Características esenciales
PostgreSql	Linux, Mac, Sun, Win	Soporta distintos tipos de datos. Velocidad lenta pero de los gestores más completos.
MySql	Linux, Mac, Sun, Win	Múltiples motores de almacenamiento. Agrupación de transacciones, reuniendo múltiples transacciones de varias conexiones para incrementar el número de transacciones por segundo.
Microsoft SQL Server 2005 E.E	Win	Una base de datos robusta para crear aplicaciones dinámicas. Fuerte soporte de XML

Tabla 5.7: Comparativa de manejadores de bases de datos

Se tienen la plataforma y el manejador elegidos, ahorapara la elección del controlador para la gestión de conexión entre la base de datos y el desarrollo de entorno será el que utiliza JAVA que es JDBC. La arquitectura de JDBC está basada en un conjunto de interfaces y clases que permiten que cualquier programa escrito en Java acceda a una base de datos, crear y ejecutar sentencias SQL, recuperar y modificar datos de una base de datos.

PostGis

La opción que tomamos para tener una base de datos espacial es PostGIS, que es una extensión al sistema de bases de datos objeto relacional PostgreSQL. PostGIS ha sido certificado en 2006 por el Open Geospatial Consortium (OGC) lo que garantiza la interoperabilidad con otros sistemas también interoperables. PostGIS almacena la información geográfica en una columna del tipo GEOMETRY, que es diferente del homónimo "GEOMETRYütilizado por PostgreSQL, donde se pueden almacenar la geometría en formato WKB (Well-Known Binary), aunque hasta la versión 1.0 se utilizaba la forma WKT (Well-Known Text). También PostGIS soporta SDT como lo son points, line strings, polygons, Multi points, Multi line strings, Multi polygons y geometry collections. Además de datos espaciales vectoriales como puntos, líneas, polígonos y superficies, es posible definir datos espaciales raster como un tipo de dato abstracto (ADT, en inglés) en bases de datos objeto-relacionales. Como resultado, los datos espaciales vectoriales y raster pueden ser almacenados y gestionados dentro de un mismo sistema de bases de datos espacial.

En la siguiente Figura 5.17 se muestra un esquema de los diferentes datos que soporta PostGIS.

Tipos *Geometry* de PostGIS

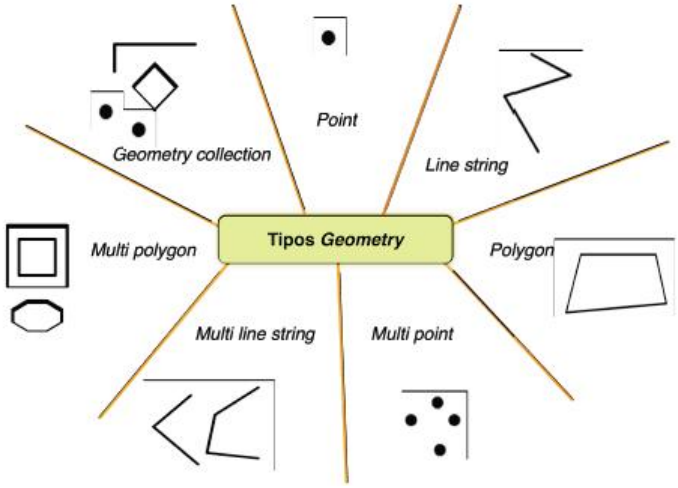


Figura 5.17: Tipos *Geometry* de PostGIS

5.8.1. Quantum Gis

Quantum GIS (QGIS) lo seleccionamos para poder hacer la visualización de los datos geográficos obtenidos. Éste es un Sistema de Información Geográfica (SIG) de código libre para plataformas GNU/Linux, Unix, Mac OS y Microsoft Windows. Era uno de los primeros ocho proyectos de la Fundación OSGeo y en 2008 oficialmente graduó de la fase de incubación. Permite manejar formatos raster y vectoriales, así como bases de datos. Algunas de sus características son:

- Soporte para la extensión espacial de PostgreSQL, PostGIS.
- Manejo de archivos vectoriales Shapefile, ArcInfo coverages, Mapinfo, GRASS GIS, etc.
- Soporte para un importante número de tipos de archivos raster (GRASS GIS, GeoTIFF, TIFF, JPG, etc.)

QGIS es un entorno multiplataforma (Linux, Windows, Mac) de aplicaciones de código abierto con muchas características comunes y las funciones de los SIG. Las características principales incluyen:

1. Ver vectores de recubrimiento y datos de trama en diferentes formatos y proyecciones, sin conversión a un formato interno o común.
    - Habilitada espacialmente tablas de PostgreSQL con PostGIS y SpatiaLite
    - La mayoría de los formatos vectoriales con el apoyo de la OGR * biblioteca, incluyendo shapefiles de ESRI, MapInfo, SDTS y GML
    - Formatos de mapa de bits con el apoyo de la GDAL * biblioteca, tales como modelos de elevación digital, fotografías aéreas o imágenes Landsat
    - En línea de datos espaciales sirvió como OGC compatibles con WMS o la CMA
2. Crear mapas de forma interactiva y explorar los datos espaciales con una interfaz de usuario gráfica. Algunas de sus herramientas de la interfaz gráfica son las siguientes:
    - información general del panel
    - marcadores espaciales

- identificar y seleccionar las características
- editar y ver atributos de búsqueda
- función de etiquetado
- diagrama vectorial de superposición
- cambio del vector y raster simbología
- anadir una capa de retícula
- guardar y restaurar proyectos

3. Crear, editar y exportar datos espaciales usando:

   - herramientas para la digitalización de GRASS y formatos shapefile
   - el plugin georeferencer
   - herramientas GPS para importación y exportación GPX formato, convertir formatos de GPS a GPX, o hacia abajo o subir directamente a una unidad GPS

4. Realizar análisis espacial utilizando el plugin para fTools Shapefiles o el plugin integrado de césped, incluyendo:

   - Mapa del álgebra
   - análisis del terreno
   - Modelación hidrológica, de análisis y muchos otros

5.8.2. Patrón de diseño a utilizar

STRUTS

Struts es un framework de los mas conocidos y de fácil construcción, hacen de MVC una opción más que aceptable y recomendable para el desarrollo de cualquier aplicación Web, bajo el lenguaje de programación Java.

Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón MVC bajo la plataforma J2EE (Java 2, Enterprise Edition). Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

Algunas de las características más importantes de Struts Framework son las siguientes:

- Una arquitectura general basada en los principios de diseño de Modelo-Vista-Controlador(MVC) en el que todas las peticiones son procesadas por el controlador que realiza todo el control y despacha las peticiones a los componentes de la aplicación apropiados, basándose en los indentificadores lógicos que reducen el acoplamiento entre capas.

- Capacidades de manejo de formularios, como el JavaBean ActionForm que representa el estado del lado del servidor de los campos de entrada de un formulario, y un marco de trabajo de validación que externaliza la configuración de un conjunto de chequeos de exactitud que se aplican a los valores de los campos de entrada, además implementa estos chequeos tanto en el lado del cliente como en el lado del servidor.

- El marco de trabajo Tiles para el control de la distribución de componentes, que soporta la creación de sofisticadas plantillas que se pueden reutilizar entre varias páginas, esto permite una fácil modificación del aspecto y comportamiento general de toda la aplicación.

- Un conjunto de etiquetas JSP personalizadas que simplifican el proceso de crear las etiquetas HTML de la aplicación para la capa "Vista", y que trabaja conjuntamente con las capacidades de manejo de formularios y la arquitectura general del controlador.

5.8.3. NetBeans

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

También está disponible NetBeans Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones.

Una de las grandes ventajas de tiene NetBeans es que un entorno multiplataforma, podrías utilzarlo en el sistema operativo que tu elijas. Las principipales herramientas que se usaron de NetBeans

fueron:

- Java Web
- Java SE
- Java ME

5.8.4. LaTeX

LaTeX fue una herramienta utilizada para la creación del documento de Tesis, ya que ofrece una mejor calidad de la presentación del documento y tiene multiples funcionalidades a comparación de otros editores de texto, como lo son la numeración automática de las imágenes e índices, entre muchas otras más, pero dada la importancia de éste documento en especial y el nivel académico se decidio presentar un documento con nivel que se merece. Éste es un sistema de composición de textos, orientado especialmente a la creación de libros, documentos científicos y técnicos que contengan fórmulas matemáticas. LaTeX está formado por un gran conjunto de macros de TeX, escrito por Leslie Lamport en 1984, con la intención de facilitar el uso del lenguaje de composición tipogrfica. Está formado mayoritariamente por órdenes (macros) construidas a partir de comandos de TeX un lenguaje de bajo nivel, en el sentido de que sus acciones últimas son muy elementales pero con la ventaja añadida de poder aumentar las capacidades de LaTeX utilizando comandos propios del TeX. Esto es lo que convierte a LaTeX en una herramienta práctica y útil pues, a su facilidad de uso, se une toda la potencia de TeX. Estas características hicieron que LaTeX se extendiese rápidamente entre un amplio sector científico y técnico, hasta el punto de convertirse en uso obligado en comunicaciones y congresos, y requerido por determinadas revistas a la hora de entregar artículos académicos.

Parte IV

Desarrollo y Puesta en Marcha

CAPÍTULO 6

Desarrollo

6.1. Requerimientos iniciales

Antes de llevar a cabo la implementación de todo el proceso del algoritmo evolutivo, se tuvo que hacer una serie de pasos necesarios para recabar la abundante información de tipo geográfica utilizando diferentes herramientas como lo son Google Maps; para la recolección de datos, bases de datos espaciales (Postgis) y Sistemas de Información Geográfica (Quantum GIS); para la manipulación y visualización de los datos geográficos referenciados en un mapa.

A continuación se explicará el orden del proceso para toda esta recopilación.

6.1.1. Creación de base de datos

La creación de la base de datos es fundamental para el sistema, ya que en ella, está contenida toda la información que permitirá atender las peticiones de los usuarios, realizadas a través de la aplicación web.

Algunos campos de la base de datos mencionada en la sección 5.6.2 son datos geográficos, por lo que no pudieron ser creados de la manera estándar en PostgreSQL, por lo que se decidió crear una base de datos espacial utilizando PostGis. Para este proceso se tuvo que seleccionar la opción de template= template-postgis durante la creación de la base de datos, como lo muestra siguiente

Figura 6.1.

Creación de Base de datos espacial

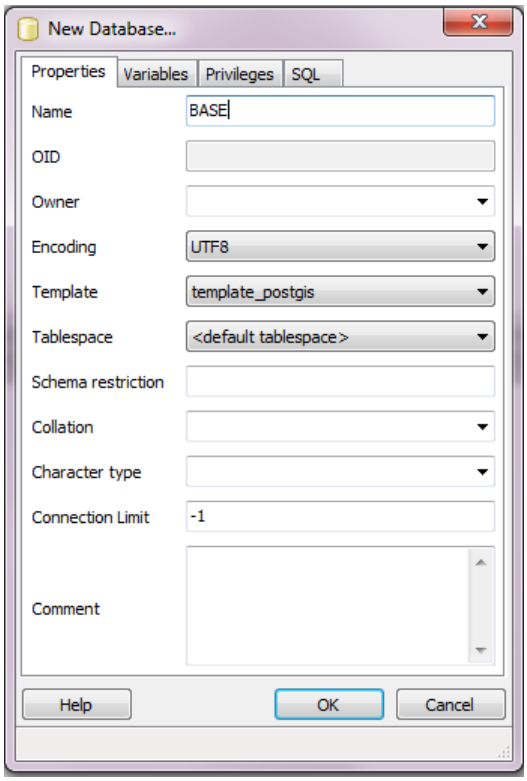


Figura 6.1: Creación de Base de datos espacial

Una vez creada la base de datos es necesaria crear las tablas correspondientes y sus respectivos campos. Para crear una tabla con datos espaciales realizamos dos pasos:

1. Creamos una tabla no espacial CREATE TABLE arcos(idarco serial NOT NULL,arco character varying)

2. Añadimos una columna(campo) espacial a la tabla usando la función AddGeometryColumn de OpenGIS.

```
1 SELECT AddGeometryColumn(<bdname>,<tablename>,<columname>, <srid>, <
     type>,<dimension>);
2 SELECT AddGeometryColumn('', 'arcos','coorarco',32614,'LINESTRING'
     ,2);
```

Cabe aclarar el significado de cada uno de los campos.

- bdname= Nombre base de datos
- tablename= Nombre de la tabla
- columname= nombre de la columna
- srid= Valor entero que identifica el sistema de referencia espacial, para México es 32614. Este valor es de suma importancia, al principio se hicieron varias pruebas hasta encontrar el valor inidicado para el área de México. Una de las razones por las cuáles era necesaria tener el srid de México es por que sino era de está forma no podíamos hacer otro tipo de consultas que son importantes como lo son: select astext, select ST_AsKML,select askml.
- type= Tipo de objeto multiespacial POINT, LINESTRING, POLYGON, MULTYPOINT, GEOMETRYCOLLECTION, GEOMETRY

6.1.2. Obtención de datos

Una vez delimitado el área de estudio del proyecto, se procedió a reunir la información necesaria. Para llevar a cabo el algoritmo evolutivo se requiere una matriz de pesos y nodos de todos los puntos del área, por lo que se obtuvo está información con los datos de la ubicación en coordenadas geográficas y distancias entre puntos. Para realizar dichas tareas se utilizó la herramienta Google Maps que da la oportunidad de sacar las coordenadas geográficas, distancia entre puntos y visualización de calles con la opción Street View, esto para poder verificar el sentido de las calles y cruces entre éstas.

Cabe aclarar que muchas de las uniones entre nodos eran líneas rectas, pero en el proyecto se consideran lo que son retornos y calles las cuales no son lineas rectas, esto afectaba en el momento en que se tenía que generar y dibujar la ruta, ya que no podíamos ir de un punto a otro en línea recta si esta no era su forma. La solución para este problema fue dividir la forma en pequeños segmentos de línea recta, para que unidos nos pudieran dar la forma deseada, a esta unión de nuevos puntos le llamamos arcos y la cantidad de esos puntos dependía de la forma y longitud de la calle o retorno.

- Medición de distancias en Google Maps

Una de las opciones que nos ofrece Google Maps, como se mencionó anteriormente, es la posibilidad de sacar la distancia entre dos puntos, para cada nodo se saco su distancia y cabe mencionar que esta recolección de datos fue manual. Como se muestra en la Figura 6.2.

Distancia en Google Maps

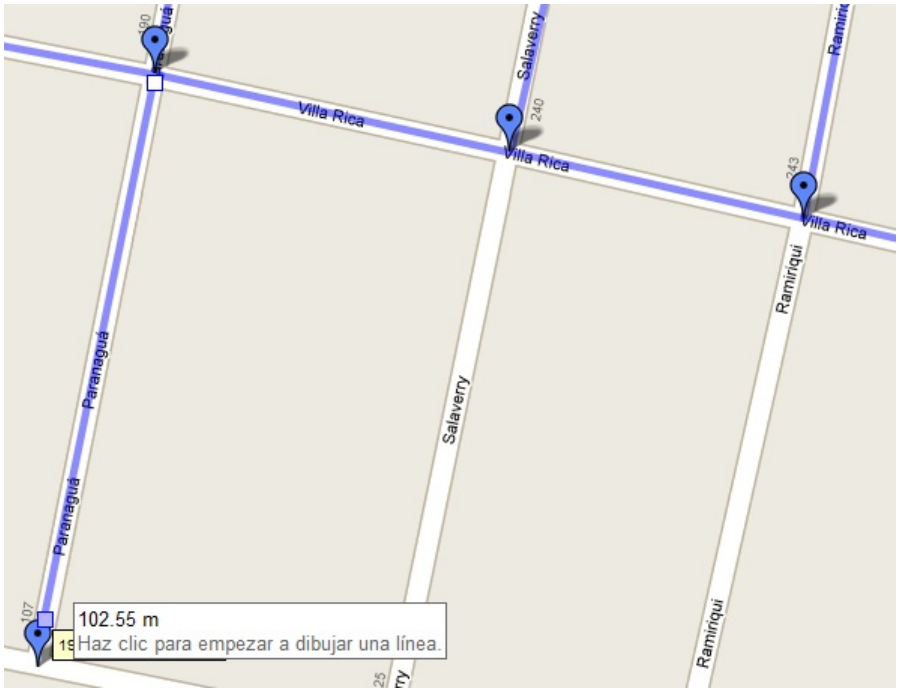


Figura 6.2: Distancia en Google Maps

- Plugin GPS Location Google Maps

  Otra de las opciones que nos ofrece Google Maps, es la posibilidad de sacar las coordenadas ¡latitud, longitud¿. Para cada nodo se saco sus coordenadas de manera manual, como se muestra en la Figura 6.3.

Coordenadas geográficas en Google Maps

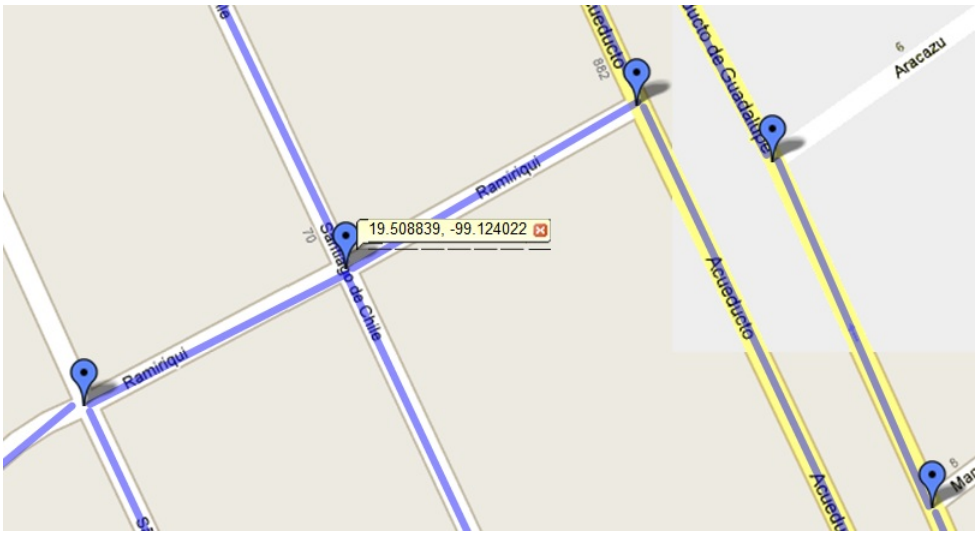


Figura 6.3: Coordenadas geográficas en Google Maps

- Verificación de sentidos de calle con Street View

  Con la nueva característica de Google Maps y Google Earth de visualizar los mapas con imágenes esféricas a nivel de calle, podemos verificar los sentidos de las calles, lo cual fué un parámetro importante para la elaboración del algoritmo evolutivo, como se muestra en la Figura 6.4. Por cuestiones de tiempo no fue posible verificarlas en sitio.

Street View de Google Maps

Figura 6.4: Street View de Google Maps

6.1.3. Conversión de coordenadas

Como mencionamos antes, las coordenadas obtenidas en Google Maps estaban en formato ¡latitud, longitud¿, pero para visualizarlo en un sistema de información geográfica como Quantum Gis, se realizó una conversión de coordenadas a formato UTM, ya que este es uno de los sistemas de coordenadas proyectadas que soporta Quantum Gis.

Para realizar dicha conversión se hizo uso de la API (Interfaz de Programación de Aplicaciones) de Java Jscience (JSR- 275), una vez realizada la conversión se almaceno la información dentro de la base de datos. En las líneas siguientes se muestra parte del código para llevar acabo la conversión de coordenadas.

```
1     import org.jscience.geography.coordinates.*;
2     CoordinatesConverter<LatLong, UTM> latLongToUTM = LatLong.CRS.
          getConverterTo(UTM.CRS);
3     LatLong latLong = LatLong.valueOf(19.511461, -99.139187, DEGREE_ANGLE
          );
4     UTM utm = latLongToUTM.convert(latLong);
```

6.1.4. Almacenamiento de información

Para el almacenamiento de la información se realizaron las consultas necesarias para la inserción de dichos datos, cabe mencionar que en el inicio el proceso se realizó de manera manual para cada uno de los nodos, posteriormente se explicará la solución para que esta inserción fuera automática.

A continuación se muestra un ejemplo de consulta para inserción de puntos geográficos:

```
1 INSERT INTO tabla (name,geom)
2 VALUES (ST_GeomFromText('name','POINT(coordenadas_utm)', -1));
```

6.1.5. Aplicaciones para insertar nodos, arcos y valores para matriz de peso

Dado el volumen de información a insertar en la base de datos, y el número de queries utilizado en los diferentes nodos y arcos, fue necesario generar aplicaciones que ayudarán al rápido almace-

namiento de dicha información.

Una de las primeras soluciones que se realizaron, fue diseñar una aplicación que pudiera leer archivos de textos; en la cual se ponían todos los nodos recolectados con el siguiente formato:

una vez recolectada la información se guardaba en un documento txt, y se procedía a llamar el archivo para que fuera leído, se analizara las coordenadas y las convertía a formato UTM, además insertaba los nodos en la base de datos con la conversión correspondiente. A continuación se pondrá parte del código que se programo resaltando las funciones más importantes.

```
1          archivo = new File (''ruta/archivo.txt");
2          fr = new FileReader (archivo);
3          float lattt=Float.valueOf(lat.trim()).floatValue();
4          float longg2=Float.valueOf(longg.trim()).floatValue();
5          CoordinatesConverter<LatLong,UTM> latLongToUTM = LatLong.CRS.
               getConverterTo(UTM.CRS);
6          LatLong latLong = LatLong.valueOf(lattt, longg2, DEGREE\_ANGLE);
7          UTM utm = latLongToUTM.convert(latLong);
8          ConsultasDB cb=new ConsultasDB();
9          cb.insertarPedidos(punto, coordUtm);
```

Durante el proceso se tuvo que crear una segunda aplicación como ayuda para almacenar la abundante información, principalmente para insertar arcos, un arco no solo consta de un par de puntos, sino de segmentos unidos por líneas rectas para que nos dan la forma deseada curva, por lo que la recolección era más tardada y se tuvo que buscar la forma de agilizar el proceso. A continuación se explicará como funciona dicha aplicación.

En la siguiente Figura 6.5 se muestra la página inicial cuando se ha cargado la aplicación, donde hemos insertado un script en la página para poder tomar las funciones de Google Maps. Se puede apreciar que hay un marcador, con este iremos seleccionando los diferentes puntos que formaran el arco y conforme se vayan añadiendo estos puntos se hará la conversión a coordenadas UTM cuando damos click en convertir coordenadas.

Pantalla inicial Aplicación

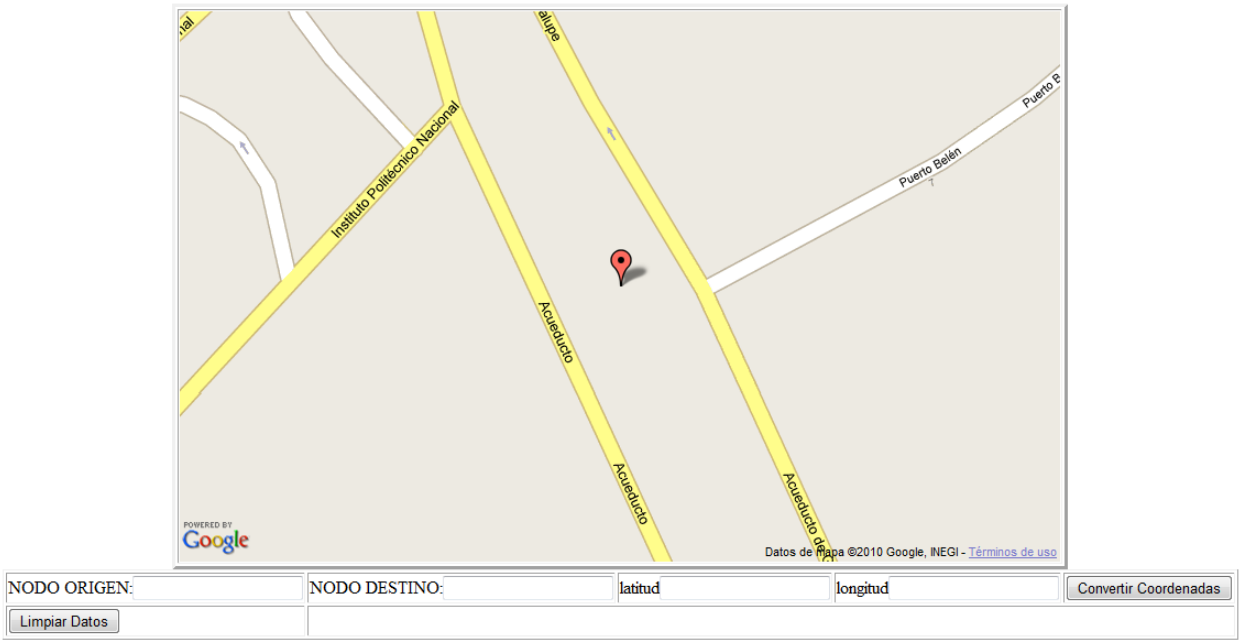


Figura 6.5: Pantalla inicial Aplicación

En la Figura 6.6 se muestran las casillas que se van completando de acuerdo a los puntos que vayamos añadiendo, con información como la distancia total, que corresponde a la suma de todos los puntos y las coordenadas convertidas a UTM. Los únicos datos que se tienen que indicar es el nombre del punto inicial, final del arco y el sentido de la calle, que es '1' si es a la derecha y '0' a la izquierda.

Pantalla inicial Aplicación

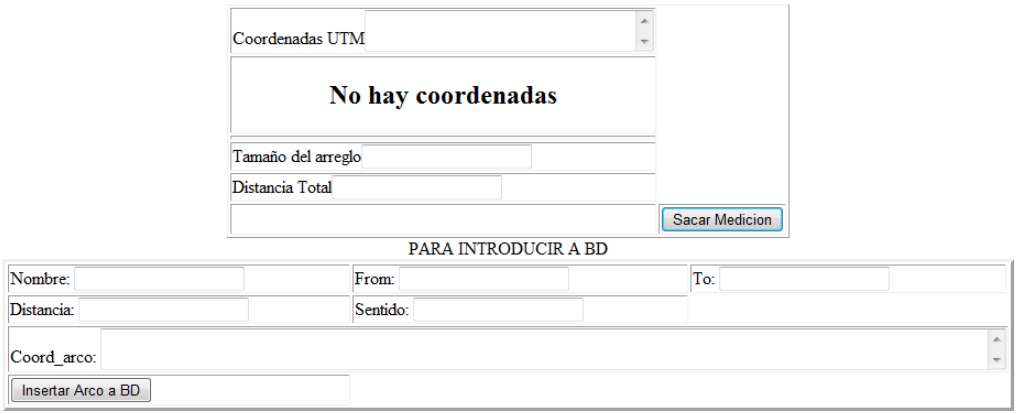


Figura 6.6: Pantalla inicial Aplicación

Después de haber marcado todos los puntos en las casillas correspondientes se muestra la información completa como se aprecia en la Figura 6.7

Aplicación con datos completos

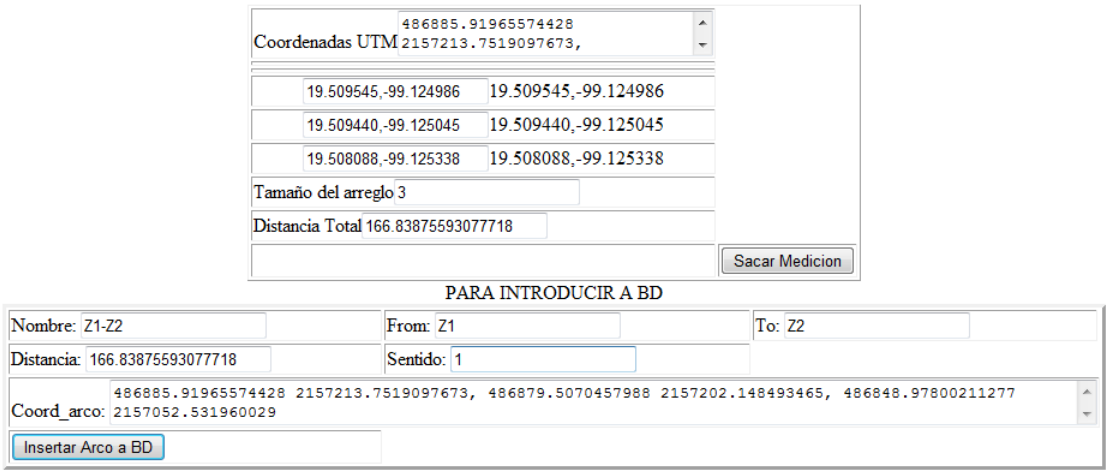


Figura 6.7: Aplicación con datos completos

6.1.6. Visualización de la información geográfica

Una vez recaudada y almacenada la información fue necesario verla reflejada en un mapa, para que pudiéramos constatar que los datos eran correctos. Los requisitos previos para hacer este paso eran tener instalado Quantum GIS y Postgis, haber cargado la base de datos de la delegación Gustavo A. Madero y tener almacenado ya en la base de datos los nodos y arcos del área de estudio en Postgis. El siguiente paso es cargar esta base de datos a Quantum Gis; para esto se hace una conexión entre Quantum Gis y Postgis, dando la información necesaria para establecer la conexión con los datos de: nombre, servidor, bd, puerto, nombre de usuario y contraseña, en la Figura 6.8, se muestra la interfaz para establecer la conexión.

Conexión Quantum Gis - PostGis

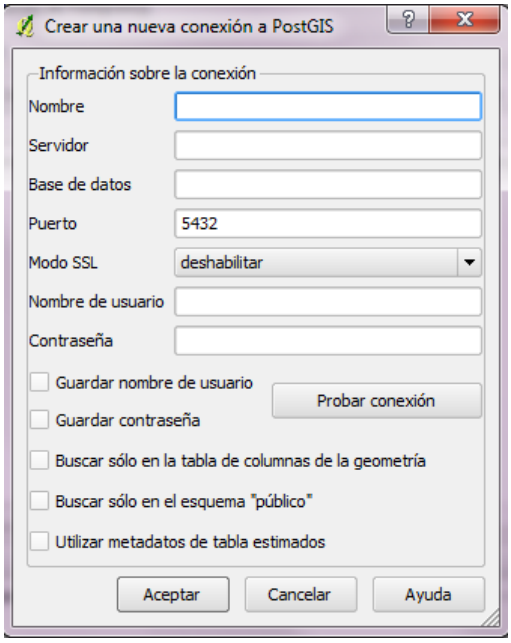


Figura 6.8: Conexión Quantum Gis - PostGis

Una vez establecida la conexión, Quantum Gis detecta los datos geográficos que contiene la base de datos, por lo que estos se pueden seleccionar y cargar. En la Figura 6.9 se puede observar todos los nodos insertados para hacer la matriz, los arcos, la unión entre estos y los nodos de los pedidos.

Datos Geográficos en Quantum Gis

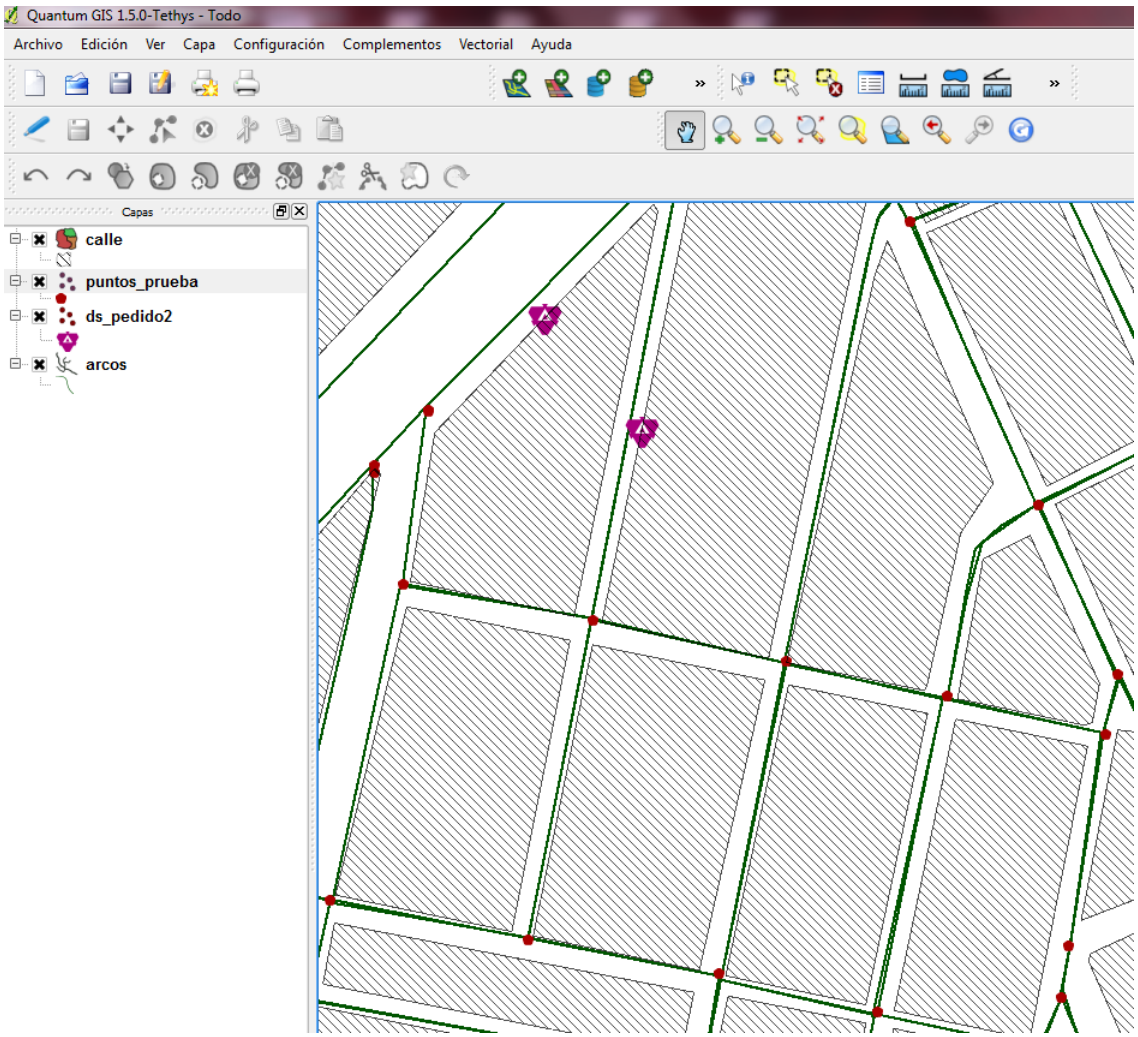


Figura 6.9: Datos Geográficos en Quantum Gis

6.2. Aplicación Web Administrativa

Está aplicación es la encargada de llevar el control tanto de usuarios y clientes, donde en cada uno de los casos se realizan tareas diferentes y específicas de acuerdo a sus permisos y funciones. A continuación se explicará cuaĺes son las partes que la conforman y que pasos se tuvieron que ejecutar para llevarlo a cabo.

Pantalla inicial

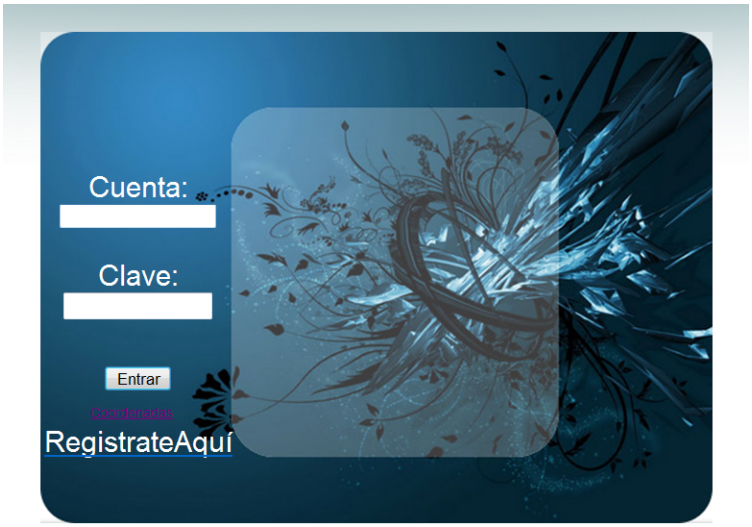


Figura 6.10: Pantalla inicial

En la imagen anterior ejemplifica el ingreso al sistema proporcionando tu cuenta y clave, si no se cuenta con ello, se puede generar usuarios ingresando todos los datos necesarios para el registro como se visualiza en la siguiente Figura 6.11.

Registro usuario

Figura 6.11: Registro usuario

Clientes

Una vez obtenida la cuenta, el cliente puede acceder a su sesión y comprar los productos que prefiera. Hay diferentes catálogos de productos, cada vez que el cliente decida comprar alguno puede añadirlo a su carrito de compras como se observa en la Figuras 6.12, 6.13, 6.14.

Catálogo de productos

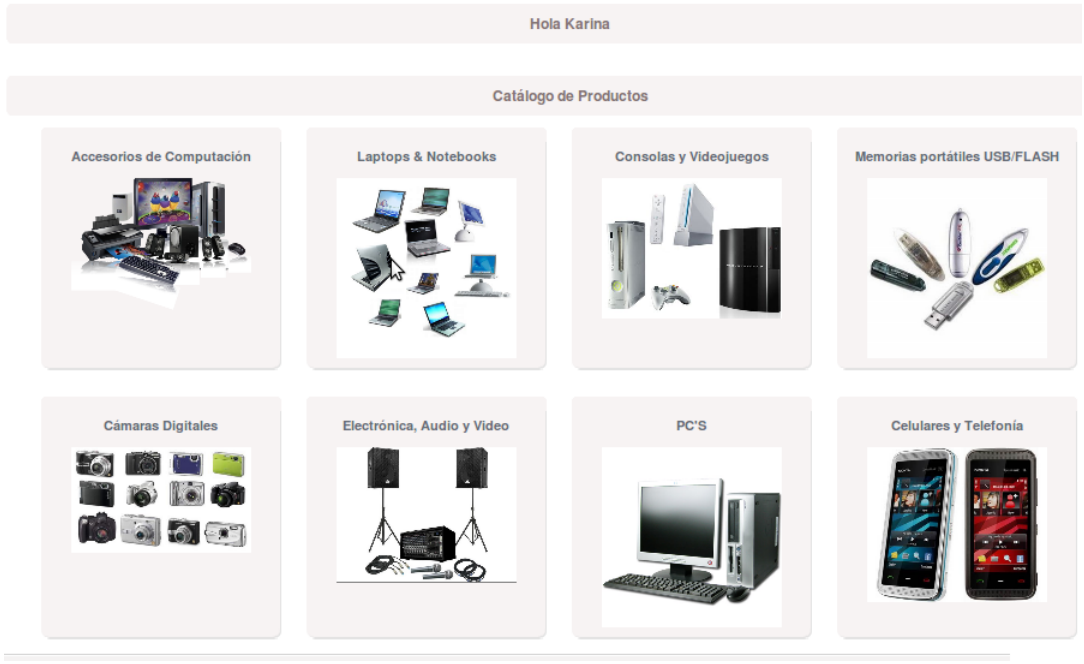


Figura 6.12: Catálogo de productos

Selección de productos

Figura 6.13: Selección de productos

Carrito de compra

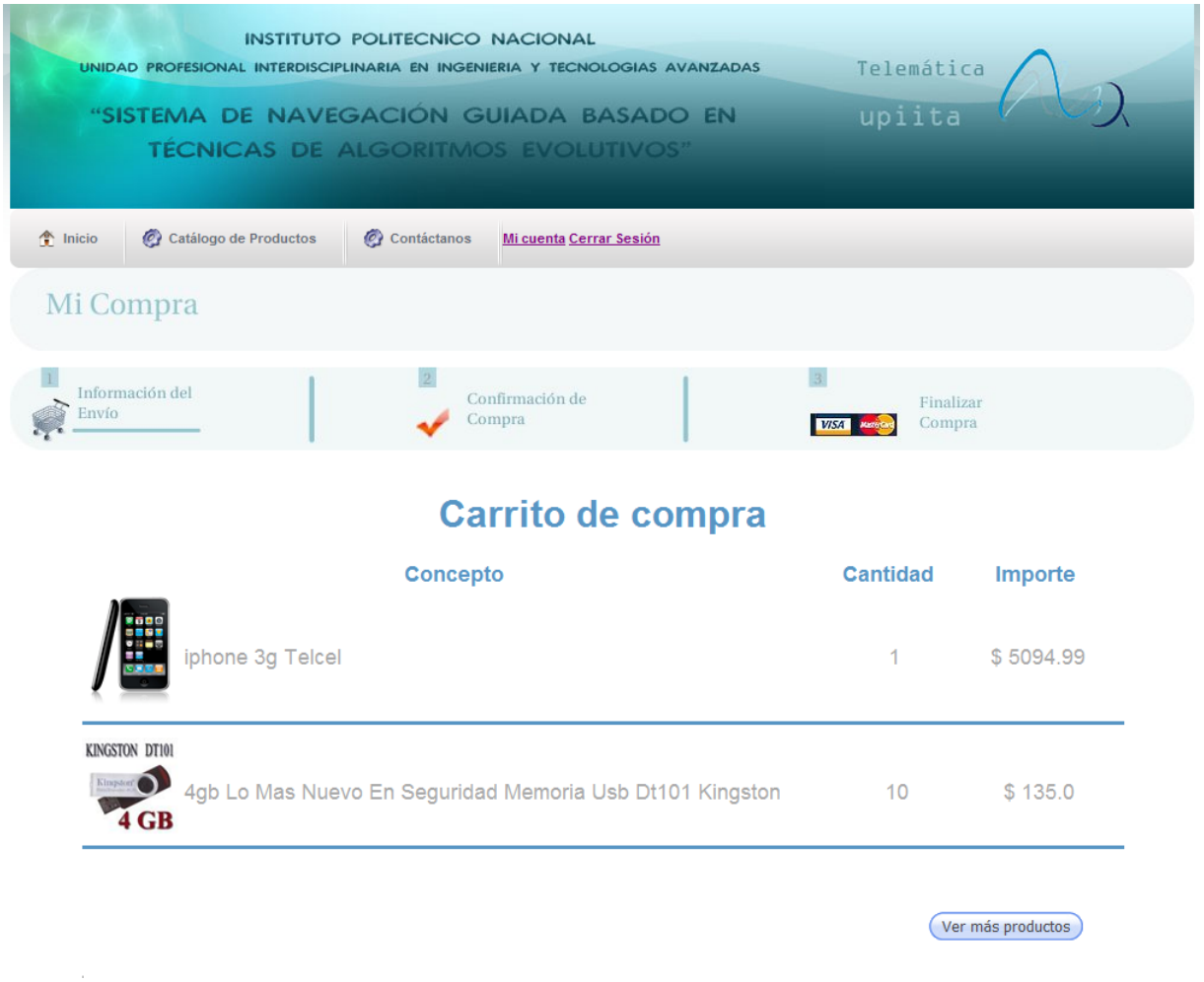


Figura 6.14: Carrito de compra

Cuando el cliente ha seleccionado todos sus productos, es momento de que proporcione la información para la generación de rutas, que es el destino. Al cliente se le habilita la opción de especificar la ubicación del sitio de entrega por medio de un mapa de Google Maps e inmediatamente queda capturada la posición con sus coordenadas geográficas. Como lo muestra la Figura 6.15

Ubicación de entrega de pedido

Figura 6.15: Ubicación de entrega de pedido

El último paso para el pedido es ingresar la información de la forma de pago, cabe aclarar que este paso es simulado ya que no se dio validación a la cuestión de seguridad para pagos en línea. La Figura 6.16 ejemplifica el último paso del pedido.

Finalización de pedido

Figura 6.16: Finalización de pedido

Administrador

Ya hemos analizado las funciones que puede realizar el cliente, ahora, corresponde analizar la parte del administrador. Este tiene funciones como altas, bajas, modificaciones y consultas de usuarios y productos, así como la parte fundamental del proyecto que es la generacón de la ruta y ver los pedidos del día. En la siguiente Figura 6.17 se observa la pantalla inicial del administrador con los menús a los que puede tener acceso.

Pantalla inicial administrador

Figura 6.17: Pantalla inicial administrador

Una de las primeras opciones que tiene el administrador es es el registro de productos para los diferents catálogos, con los campos requeridos para su correcto registro como lo muestra la Figura 6.18.

Registro de productos

Figura 6.18: Registro de productos

A su vez, el administrador puede eliminar estos productos o modificar sus atributos como lo son: el catálogo al que pertenece, su precio, nombre de producto, fotografía y dar de alta alguna promoción, si es que la tiene. En las Figuras 6.19 y 6.20 se muestra las opciones mencionadas. El administrador puede hacer éstas mismas altas, bajas, consultas y modificaciones para los clientes.

Eliminar productos

Figura 6.19: Eliminarproductos

Cuando finalizó la jornada de pedidos del día, el administrador tiene la opción de ver todos los pedidos hechos como se muestra en la Figura 6.21, para después proceder a crear la ruta con la opción de crear itinerario, y este es el punto de partida para dar paso al análisis del algoritmo evolutivo que se verá en el siguiente capítulo.

Modificar productos

Figura 6.20: Modificar productos

Crear itinerario

Figura 6.21: Crear itinerario

CAPÍTULO 7

Generación de itinerarios

7.1. ¿Qué información se necesita?

El tema pricipal de nuestro proyecto fue entender el problema de como hacer una representación de los datos en nuestro caso las calles, que de tal manera nos facilitará la obtención de las rutas. Por ello, como primer paso fue definir lo que para nosotros serían la información a recolectar:

- **Nodos:** Cruces de calles.
    - Para los nodos se consideró su respectiva posición en coordenadas geográficas en formato latitud, longitud y UTM(*Sistema de Coordenadas Universal Transversal de Mercator*) y un nombre identificador.
- **Arcos:** Conexión entre nodos.
    - Los arcos son nuestra representación de la calles, por lo que consideramos el nodo origen y nodo destino, la distancia entre ellos, sentido de la calle y en caso de que el tipo de arco sea irregular (refiriendose si es curvo el cual requiera más de dos puntos para tener una semejanza a la calle de un mapa real).
    - Se considerarón los retornos, en la mayoría de las calle y/o avenidad principales se tomó en

cuenta los retornos permitidos, auxiliandonos con la aplicación de *Google Street View*, de los cuales también se considerarón las caracteristicas que arriba se mencionan.

Teniendo una idea de cómo vamos a visualizar los datos de forma digital, los resultados que obtuvimos fue la Fig. 7.1, en la cual muestra los nodos a considerar para la generación de nuestra matriz de incidencias.

Figura 7.1: Nodos a considerar en nuestro proyecto

A continuación en la Fig. 7.2, se muestran los arcos que son alrededor de más de 500 arcos que indican sentidos de calles y retornos dando como resultado un grafo dirigido, donde cada nodo esta conectado con pesos positivos.

Figura 7.2: Arcos a considerar en nuestro proyecto

7.2. ¿Cómo plantear el problema?

El TSP (*Problema del agente viajero*), es una aplicación conceptualmente sencilla, en donde el viajero debe visitar cada ciudad dentro de un área delimitada exactamente una vez y regresar al punto de partida. Dado el costo del viaje entre todas las ciudades, ¿cómo se debe planear su itinerario para obtener el minimo costo total de la ruta completa?. El espacio de búsqueda para el TSP, consiste en un grupo de permutaciones de *n* ciudades. Cada permutación de *n* ciudades trata una posible solución. Por lo tanto **la solución consiste en una permutación que produce el minimo costo de la ruta** y con ello el espacio de búsqueda es n!. El TSP se volvió popular hace unos años debido a que áreas de programación lineal tratarón de resolver problemas combinatorios. El cual fue probado ser un problema de clase NP-hard que significa que los algoritmos para resolverlo requieren tiempo exponencial. A raiz de esto, el TSP se convirtió en un tema de interés para la comunidad GA(*Algoritmos Genéticos*), del cual se han generado diversos algoritmos, estos algoritmos producen soluciones casi optimas a tráves de poblaciones de soluciones potenciales bajo un esquema de selección a favor de su aptitud. La solución que estamos proponiendo consiste en obtener un espacio considerablemente grande de posibles soluciones, sin embargo, partiendo de un caso particular, empleamos un algoritmo que nos ayudó a obtener la distancia de todos los nodos, apoyandose en los siguientes parametros.

- Matriz de adyacencias: muestra la distancia entre un vértice y los demas. El valor de infinito denota que no hay una ruta directa.

- Matriz de recorridos:Eventualmente muestra el siguiente vértice que se necesita en el recorrido para encontrar la ruta más corta hacia el siguiente vértice.

Posteriormente se extraen de la base de datos todos los pedidos que se recibieron durante el día, por medio de la página web y con ello tendriamos la información suficiente para el procesamiento y obtener una solución.

7.3. Algoritmo de Floyd

El objetivo de éste algoritmo es encontrar el camino más corto entre todos los pares de nodos de un grafo. Dado un grafo g, lo que queremos es hallar el camino más corto para ir desde el nodo i hasta el nodo j dentro de todas las posibilidades que se puedan presentar. Lo único que necesitamos para resolver este problema es la matriz de pesos del grafo (D), es decir, una matriz que recoge el coste de ir de un nodo a otro del grafo sin pasar por nodos intermedios. Cuando no hay camino directo entre dos nodos, la celda correspondiente tendrá valor infinito.

7.3.1. Funcionamiento

Veamos un ejemplo de cómo funciona el algoritmo para el siguiente grafo:

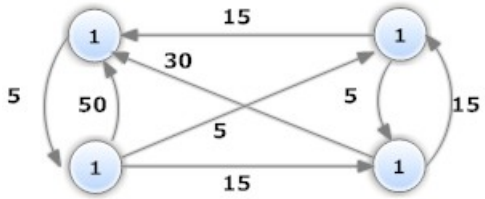


Figura 7.3: Grafo a resolver

La matriz de pesos del grafo es:

D	1	2	3	4
1	0	5	∝	∝
2	50	0	15	5
3	30	∝	0	15
4	15	∝	5	0

Figura 7.4: Matriz de pesos

Como se explicó antes, esta matriz representa el coste de ir de un nodo a otro del grafo sin pasar por nodos intermedios. En cada iteración del algoritmo se añade un nodo a través del cual se pueden establecer caminos para ir de un nodo a otro, así, al final de la *k-ésima* iteración, D[i][j] indica el menor coste de cualquier camino entre el nodo i y el nodo j que pase por nodos con número menor o igual que k.

A continuación se muestran los cambios que sufre la matriz en las sucesivas iteraciones que los producen y como va quedando por faces:

k=0; i=2; j=1

D(2,1) < D(2,0) + D(0,1)? → ∝ < 30 + 5? → **D(2,1) = 35**

D	1	2	3	4
1	0	5	∝	∝
2	50	0	15	5
3	30	35	0	15
4	15	∝	5	0

k=0; i=3; j=1

D(3,1) < D(3,0) + D(0,1)? → ∝ < 15 + 5? → **D(3,1) = 20**

D	1	2	3	4
1	0	5	∝	∝
2	50	0	15	5
3	30	35	0	15
4	15	20	5	0

k=1; i=0; j=2

D(0,2) < D(0,1) + D(1,2)? → ∝ < 5 + 15? → **D(0,2)=20**

D	1	2	3	4
1	0	5	20	∝
2	50	0	15	5
3	30	35	0	15
4	15	20	5	0

k=1; i=0; j=3

D(0,3) < D(0,1) + D(1,3)? → ∝ < 5 + 5? → **D(0,3)=10**

D	1	2	3	4
1	0	5	20	10
2	50	0	15	5
3	30	35	0	15
4	15	20	5	0

Figura 7.5: Parte I de las iteraciones

Como no podemos representar el infinito, por ejemplo, cuando el valor de una celda de la matriz de pesos es infinito, lo representaremos con el valor `Double.MAX_VALUE`, ya que es poco probable que se presenten pesos de esta magnitud y en dicho caso se podría adaptar esta circunstancia a la solución.

k=2; i=1; j=0

D(1,0) < D(1,2) + D(2,0)? → 50 < 15 + 30? → **D(1,0) = 45**

D	1	2	3	4
1	0	5	20	10
2	45	0	15	5
3	30	35	0	15
4	15	20	5	0

k=3; i=0; j=2

D(0,2) < D(0,3) + D(3,2)? → 20 < 10 + 5? → **D(0,2) = 15**

D	1	2	3	4
1	0	5	20	10
2	45	0	15	5
3	30	35	0	15
4	15	20	5	0

k=3; i=1; j=0

D(1,0) < D(1,3) + D(3,0)? → 45 < 5 + 15? → **D(1,0)=20**

D	1	2	3	4
1	0	5	20	10
2	20	0	15	5
3	30	35	0	15
4	15	20	5	0

k=3; i=1; j=2

D(1,2) < D(1,3) + D(3,2)? → 15 < 5 + 5? → **D(1,2)=10**

D	1	2	3	4
1	0	5	20	10
2	20	0	10	5
3	30	35	0	15
4	15	20	5	0

Figura 7.6: Parte II de las iteraciones

7.4. Algoritmos Genéticos

7.4.1. ¿Cómo funciona un algoritmo genético?

Los algoritmos genéticos consisten en generar una serie de posibles soluciones al azar y en ir creando sucesivas generaciones (soluciones hijas, mezcla de otras soluciones posibles anteriores), dando más importancia y más peso a las mejores soluciones.

7.4.2. ¿Cuál es el objetivo de utilizar un algoritmo genético?

El objetivo de los AG es buscar dentro de un espacio de hipótesis candidatas la mejor de ellas. En los AG la mejor hipótesis es aquella que optimiza a una métrica predefinida para el problema dado, es decir, la que más se aproxima a dicho valor numérico una vez evaluada por la función de evaluación.

7.4.3. Conceptos y funciones

En la fig. 7.7 muestra la estructura de un algoritmo genético:

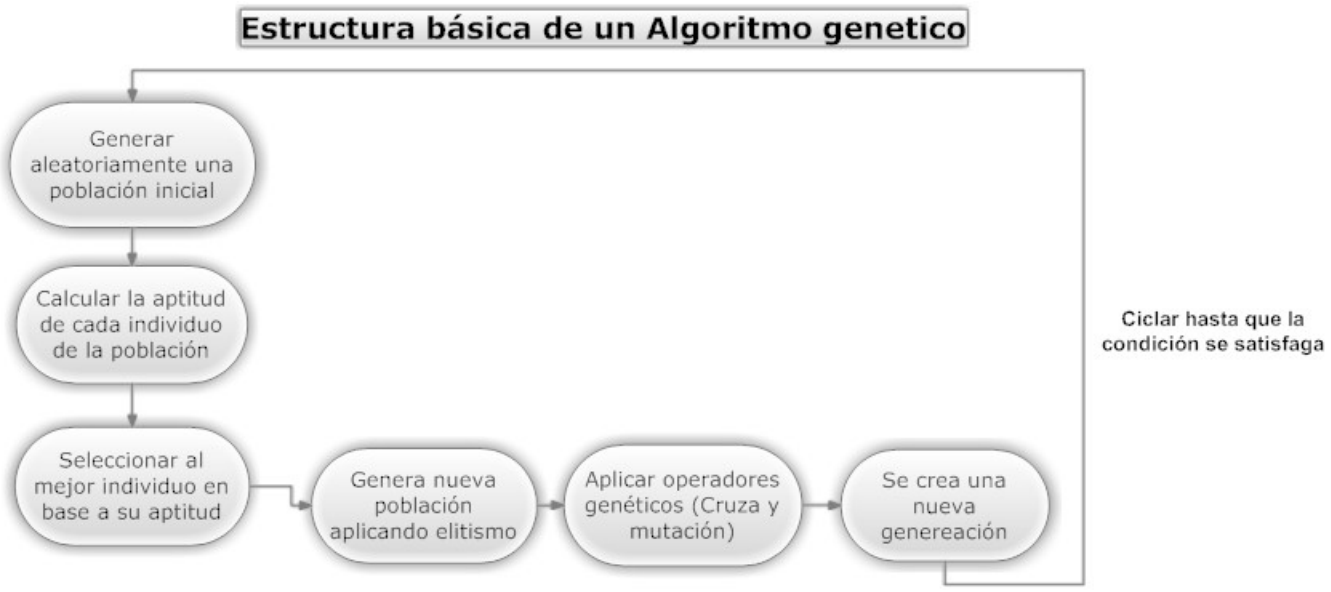


Figura 7.7: Estructura básica de un algoritmo genético

- **Generar (aleatoriamente) una población inicial:**Es el punto de partida para el algoritmo genético, donde se generan nuestros individuos de forma aleatoria sin que se repitan entre sí.

- **Calcular aptitud de cada individuo:**Una vez generada la población inicial se debe obtener la aptitud que consiste en la suma total de dicho recorrido tomando de referencia lo obtenido en el algoritmo de Floyd.

- **Seleccionar (probabilísticamente) en base a aptitud:** Mediante un método de selección, se determina el individuo que pasará a la siguiente generación.

- **Generar nueva población:**Son todos aquellos individuos que fueron seleccionados en el punto anterior.

- **Aplicar operadores genéticos (cruza y mutación) para generar la siguiente población:**A la nueva generación se aplican estos operadores con el fin de reproducirse para resultar una nueva generacion.

- **Ciclar hasta que cierta condición se satisfaga:** Continuar con los pasos anteriores hasta que un inviduo logre sobrepasar todas aquellas generaciones creadas dado que su aptitud siempre fue la más fuerte.

Por otra parte, para seguir continuando con este tema es necesario dar una breve explicación sobre los conceptos que estaremos manejando a lo largo de éste capitulo, que se enfoca únicamente al algoritmo genético. Éstos son los siguientes:

- **Cromosoma:** Se le denomina a una estructura de datos que contiene una cadena de parámetros de diseño o genes. Esta estructura de datos puede almacenarse, por ejemplo, como una cadena de bits o un arreglo de enteros.

- **Gene:** Es una subsección de un cromosoma que representa el valor de un solo parámetro.

- **Individuo:** Se refiere a un solo miembro de la población de soluciones potenciales fig. 7.10, a un problema. Cada individuo contiene un cromosoma que representa una solución posible al problema a resolverse.

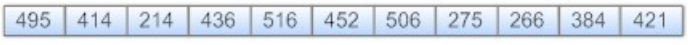


Figura 7.8: Individuo de una posible solución

- **Aptitud:** Es el valor que se asigna a cada individuo indicando su importancia con respecto a los demas inviduos de la generación.

- **Generación:** Es una iteración de la medida de aptitud y a la creación de una nueva población por medio de operadores de reproducción.

- **Operadores de reproducción:** Es todo aquel mecanismo que influencia la forma en que se pasa la información genética de padres a hijos. Los operadores de reproducción que utilizaremos son:

  Cruza: Forma un nuevo cromosoma combinando partes de cada uno de sus cromosomas padres.

  Mutación: Se denomina mutación a un operador que forma un nuevo cromosoma a través de alteraciones (usualmente pequeñas) de los valores de los genes de un solo cromosoma padre.

- **Selección:** Es la selección de los que se cruzarán en la siguiente generación (donde se escogen a los "mejores").

- **Elitismo:** Asegura que la aptitud máxima de la población nunca se reducirá de una generación a la siguiente.

7.5. Implementación de algoritmo genético

7.5.1. Representación de datos

Nosotros representamos nuestros datos de forma entera, es decir, una vez que tenemos todos los pedidos recolectados, hacemos un mapeo de los datos de los nodos que estan involucrados para asi poder trabajar con ello para las demas funciones, quedando de la siguiente manera:

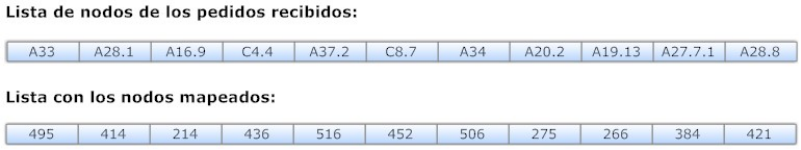


Figura 7.9: Mapeo de datos de entrada

7.5.2. Proceso de selección

Selección Mediante Torneo

Se baraja la población, es decir tomamos a cada individuo y desordenamos sus genes dando como resultado otro individuo, y después se hace competir a los cromosomas que la integran en grupos de tamaño predefinido (normalmente en parejas) en un torneo del que resultarán ganadores aquéllos que tengan valores de aptitud más altos (que para nuestro caso será aquel que tenga la distancia mínima). La selección que se implementó fue mediante torneo y a continuación el algoritmo de la versión determinística:

1. Barajar los individuos de la población.
2. Escoger un numero p de individuos (tipicamente 2).
3. Compararlos con base en su aptitud.
4. El ganador del "torneo" es el individuo más apto.
5. Debe barajarse la población un total de p veces para seleccionar N padres (donde N es el tamaño de la población).

Análisis de la seleccion mediante torneo

La version deterministica garantiza que el mejor individuo ser seleccionado "p" veces (tam torneo).

- Complejidad:
    1. Cada competencia requiere la selección aleatoria de un número constante de individuos de la población. Esta comparacion puede realizarse en O(1).
    2. Se requieren "n" competencias de este tipo para completar una generación.
    3. Por lo tanto, el algoritmo es O(n).
- La técnica es eficiente y fácil de implementar.
- No requiere escalamiento de la funcion de aptitud (usa comparaciones directas).

- Puede introducir una presión de selección muy alta (en la version determinística) porque a los individuos menos aptos no se les da oportunidad de sobrevivir.

7.5.3. Operador de Cruza

En los sistemas biológicos, la cruza es un proceso complejo que ocurre entre parejas de cromosomas. En computación evolutiva se simula la cruza intercambiando segmentos de cadenas lineales de longitud fija (los cromosomas). Normalmente la cruza se maneja dentro de la implementación del algoritmo genético como un porcentaje que indica con qué frecuencia se efectuará. Esto significa que no todas las parejas de cromosomas se cruzarán, sino que habrán algunas que pasarán intactas a la siguiente generación, en donde el individuo más apto a lo largo de las distintas generaciones no se cruza con nadie, y se mantiene intacto hasta que surge otro individuo mejor que él, que lo desplazará dcha técnica se denomina elitismo.

Cruza para Permutaciones

La representación de permutaciones se usa frecuentemente en problemas de optimización combinatoria, como el del viajero y consiste básicamente en usar cadenas de enteros para representar una permutación como se muestra a continuación:

495	414	214	436	516	452	506	275	266	384	421

Figura 7.10: Individuo generado apartir de una permutación

Estudiaremos para nuestro caso la técnica **Partially Mapped Crossover** (*PMX*) El algoritmo es el siguiente:

1. Elegir aleatoriamente dos puntos de cruza.
2. Intercambiar estos 2 segmentos en los hijos que se generan (como la cruza de 2 puntos convencional).
3. El resto de las cadenas que conforman los hijos se obtienen haciendo mapeos entre los 2 padres:
4. Si un valor no está contenido en el segmento intercambiado, permanece igual.
5. Si está contenido en el segmento intercambiado, entonces se sustituye por el valor que tenga dicho segmento en el otro padre.

A continuación se muestra un ejemplo:

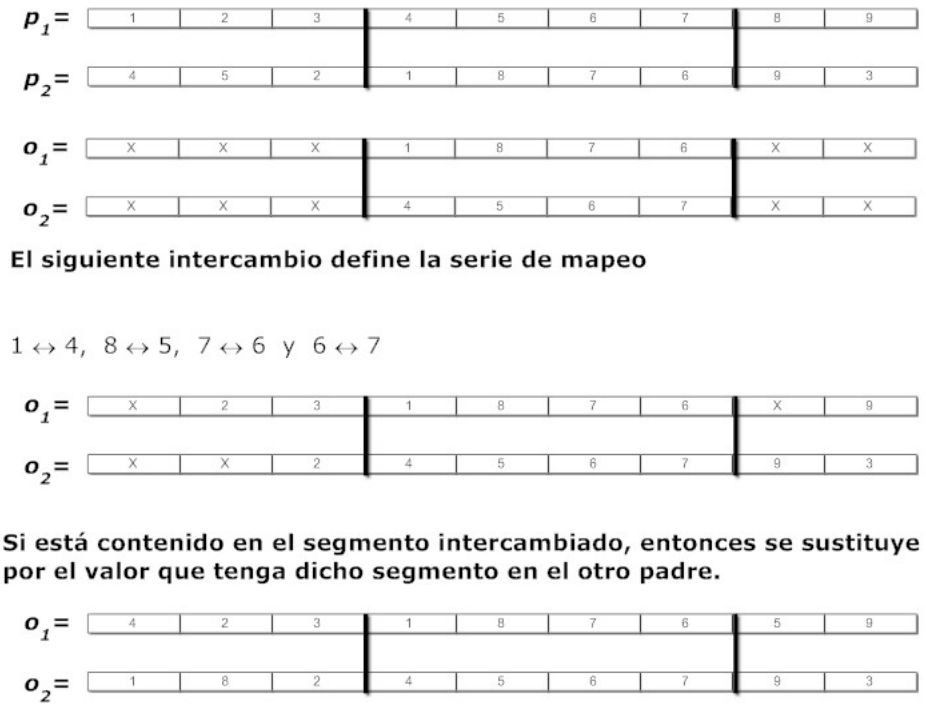


Figura 7.11: Ejemplo de cruza para permutaciones

7.5.4. Operador de Mutación

La mutación se considera como un operador secundario en los algoritmos genéticos. Es decir, su uso es menos frecuente que el de la cruza. En la práctica, se suelen recomendar porcentajes de mutación de entre 0.001 y 0.01 para la representación binaria. El papel que juega la mutación en el proceso evolutivo, así como su comparación con la cruza, sigue siendo tema frecuente de investigación y debate en la comunidad de computación evolutiva.

Mutación para Permutaciones

Comenzaremos nuestra revisión de técnicas enfocándonos a la representación de permutaciones que se suele utilizar en problemas de optimización combinatoria (*como el del viajero*).

Mutación por inserción

Se selecciona un valor en forma aleatoria y se le inserta en una posición arbitraria.

Mutación por desplazamiento

Es una generalización de la mutación por inserción en la que en vez de mover un solo valor, se cambian de lugar varios a la vez.

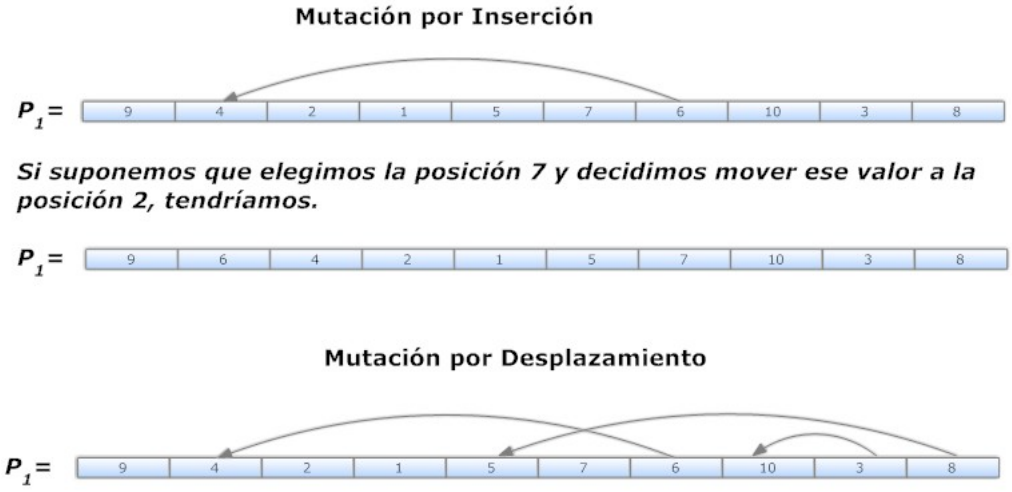


Figura 7.12: Ejemplo de mutación para permutaciones

7.6. Resultados obtenidos

El resultado de todo el algoritmo evolutivo nos dió el orden en que se deben entregar los pedidos y con ello se obtuvo una ruta a seguir con ayuda del algoritmo de Floyd. En el siguiente ejemplo denotaremos los resultados que se obtuvieron con la implementación de la solución propuesta. Cuando generamos los pedidos, éstos se guardan en la forma en como fueron llegando, es decir, cada vez que se registraron pedidos se almacenan en el orden en que la base de datos los recibe. A continuación se muestra una cadena que representa los pedidos que se tienen para generar el itinerario:

Ruta obtenida

Prueba 1

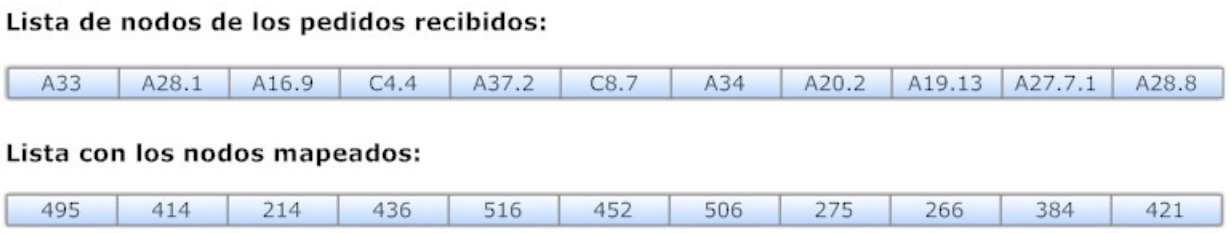


Figura 7.13: Mapeo de los datos de entrada - Pedidos registrados

En la Figura. 7.13 se muestra la cadena mapeada, la cual al hacer el cálculo inicial de su aptitud obtenemos un valor 14,370.03 m, tomándolo como valor de comparación despues de generar el AG.

Una vez ya mapeados los datos ahora si procedemos a introducir esa cadena de entrada a nuestro algoritmo genético. Nuestro tamaño de poblacion es de 200, lo que significa que generará alrededor de 200 diferentes soluciones posibles a evaluar, y tomar en cuenta para determinar un individuo "ganador". Las implementaciones de código de las funciones del AG, se muestran al final como un apartado indicando el nombre del método y su objetivo. Al ejecutar el AG, los resultados que obtuvimos fueron los siguientes: **prueba 1**

```
El mejor individuo de la generacion:
 516 421 436 452 384 266 214 275 414 495 506 6726.66
```

Recapitulando lo que obtuvimos en la entrada:

```
MAPEO
 495 414 214 436 516 452 506 275 266 384 421
```

```
Aptitudes
 1535.95 2004.29 1869.7700000000002 694.36 967.28 1142.9299999999998
    1555.5600000000002 1491.5700000000002 1138.39 1309.12 660.81
```

¿Que conclusiones podemos determinar con esta información? En nuestros datos de entrada la aptitud de nuestro individuo inicial es de 14,370.03 m, una vez que se ha pasado estos datos al AG, obtenemos que la aptitud del individuo "ganador" resulto de 6726.66, lo que significa que mejoro en aproximadamente un 40 %, la distancia total del recorrido, obteniendo el orden en que estos pedidos deben ser entregados:

```
MAPEO
 516 421 436 452 384 266 214 275 414 495 506
```

Lo cual el siguiente paso es decodificar esta cadena para ver de que nodos se trata, y ahora ayudarnos del algoritmo de Floyd para que nos devuelva el camino entre nodo y nodo para obtener la ruta completa.

Ruta desplegada en mapa

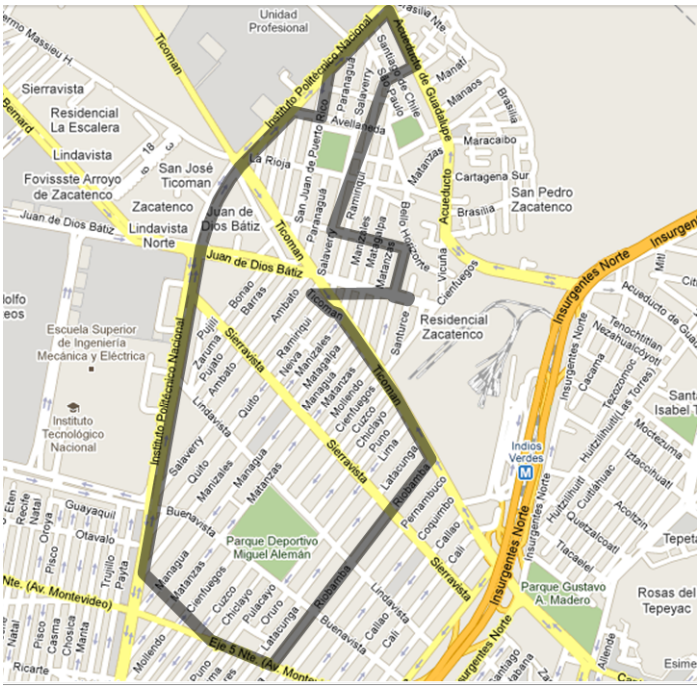


Figura 7.14: Ruta obtenida - prueba 1

Prueba 2

```
MAPEO
 211 511 419 256 448 330
Aptitudes
 5256.610000000001 573.42 1585.4400000000003 1488.01 662.1800000000001
    1526.0800000000002
```

```
El mejor individuo de la generacion:
 511 448 256 211 330 419 5972.189999999999
```

¿Que conclusiones podemos determinar con esta información? En nuestros datos de entrada la aptitud de nuestro individuo inicial es de 11091.74 m, una vez que se ha pasado estos datos al AG, obtenemos que la aptitud del individuo "ganador" resulto de 5972.189, lo que significa que mejoro en aproximadamente un 40 %, la distancia total del recorrido, obteniendo el orden en que estos pedidos deben ser entregados:

```
MAPEO
 211 511 419 256 448 330
```

Lo cual el siguiente paso es decodificar esta cadena para ver de que nodos se trata, y ahora ayudarnos del algoritmo de Floyd para que nos devuelva el camino entre nodo y nodo para obtener la ruta completa.

Ruta desplegada en mapa

Figura 7.15: Ruta obtenida - prueba 2

Prueba 3

```
MAPEO
 342 266 436 256 448 452
Aptitudes
 2808.44 1275.0700000000002 1577.4399999999998 1488.01 594.07 1193.07
```

```
 El mejor individuo de la generacion:
 452 342 256 266 448 436 5339.3
```

¿Que conclusiones podemos determinar con esta información? En nuestros datos de entrada la aptitud de nuestro individuo inicial es de 8936.02 m, una vez que se ha pasado estos datos al AG, obtenemos que la aptitud del individuo "ganador" resulto de 5339.3, lo que significa que mejoro en aproximadamente un 40 %, la distancia total del recorrido, obteniendo el orden en que estos pedidos deben ser entregados:

```
MAPEO
 342 266 436 256 448 452
```

Lo cual el siguiente paso es decodificar esta cadena para ver de que nodos se trata, y ahora ayudarnos del algoritmo de Floyd para que nos devuelva el camino entre nodo y nodo para obtener la ruta completa.

Ruta desplegada en Google Maps

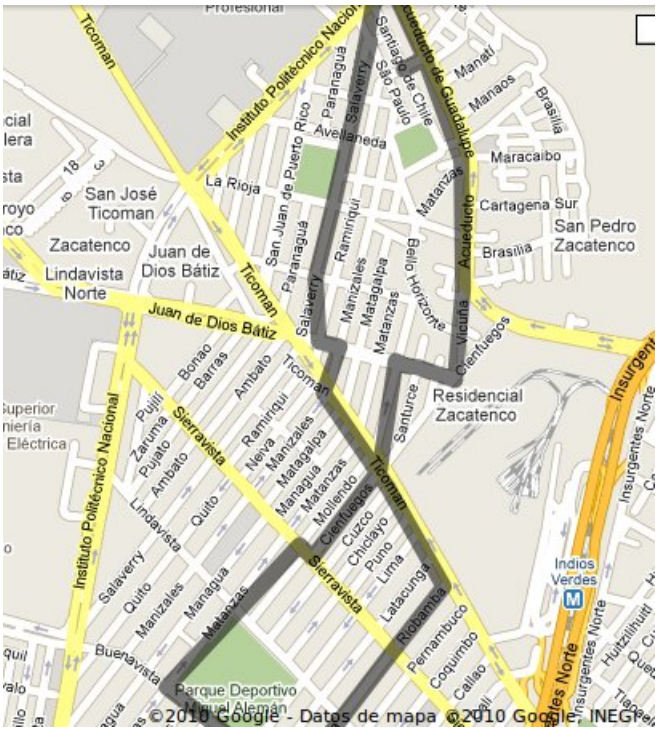


Figura 7.16: Ruta obtenida - prueba 3

Generación KML

Dado que por parte del genetico nos dio genercódigo solo entrega los puntos en coordenadas geográficas sin una interpretación por lo que a estos datos se referenciaron en un mapa para poder ver el contenido de la información. Una forma encontrada fue utilizar KML, que son archivos que nos permiten manipular datos geográficos. En la siguiente sección explicaremos la utilziación de KML La generación de la ruta por medio de KML, fue de gran ayuda para visualizarla en Google Maps, una vez obtenidos todos los puntos de la ruta pasaron a una función que es la encargada de generar dicho archivo KML, y posteriormente subirlo al servidor de Google Maps para que pueda ser visualizado públicamente. En la Figura 7.17 se observa la ruta obtenida y visualizada en Google Maps.

Archivo kml en Google Maps

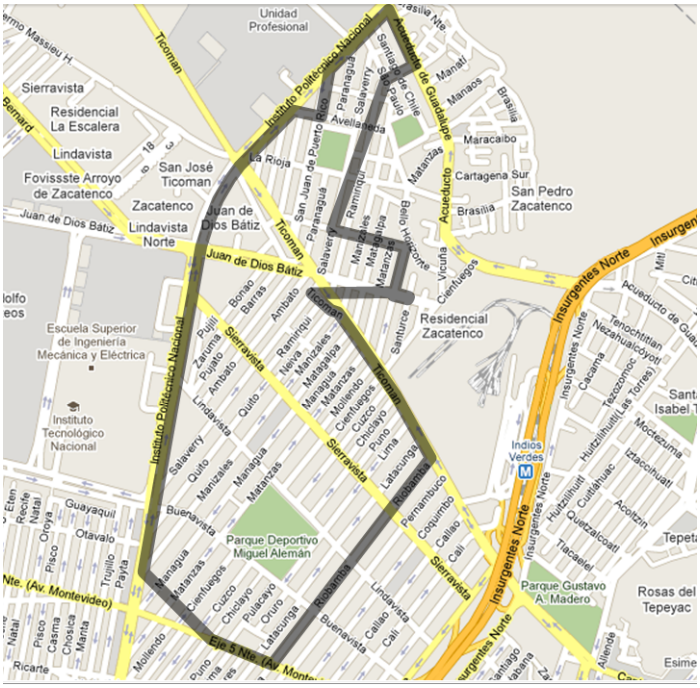


Figura 7.17: Kml en Google Maps

El siguiente código muestra la función que genera el archivo KML, recibiendo como parámetros el arreglo de coordenadas de la ruta obtenida.

```
1 public static void GeneraKml(String geo [] ){
2         BufferedWriter br = null;
```

```
3           try {
4               br = new BufferedWriter(new FileWriter("Final.kml"));
5               br.write("<?xml version=\"1.0\" encoding=\"UTF-8\"?>");
6               br.newLine();
7               br.write("<kml xmlns=\"http://www.opengis.net/kml/2.2\">");
8               br.newLine();
9               br.write("<Document>");
10              br.newLine();
11              br.write("<name>KmlFile</name>");
12              br.newLine();
13              br.write("<Style id=\"thickBlackLine\">");
14              br.newLine();
15              br.write("<LineStyle>");
16              br.newLine();
17              br.write("<color>87000000</color>");
18              br.newLine();
19              br.write("<width>10</width>");
20              br.newLine();
21              br.write(" </LineStyle>");
22              br.newLine();
23              br.write("</Style>");
24              br.newLine();
25              br.write("<Placemark>");
26              br.newLine();
27              br.write("<name>Relative</name>");
28              br.newLine();
29              br.write("<visibility>1</visibility>");
30              br.newLine();
31              br.write("<description>Tu ruta a seguir</description>");
32              br.newLine();
33              br.write("<styleUrl>#thickBlackLine</styleUrl>");
34              br.newLine();
35              br.write("<LineString>");
36              br.newLine();
37              br.write("<tessellate>1</tessellate>");
```

```
38              br.newLine();
39              br.write("<altitudeMode>relativeToGround</altitudeMode>");
40              br.newLine();
41                          br.write("<coordinates>");
42                          br.newLine();
43              for(int i=0; i< geo.length; i++){
44                                  br.write(""+geo[i]+",645  ");
45                                  br.newLine();
46                          }
47                          br.write("</coordinates> ");
48                          br.newLine();
49              br.write("</LineString>");
50              br.newLine();
51              br.write(" </Placemark>");
52              br.newLine();
53              br.write(" </Document>");
54              br.newLine();
55              br.write("</kml> ");
56          } catch (FileNotFoundException ex) {
57              ex.printStackTrace();
58          } catch (IOException ex) {
59              ex.printStackTrace();
60          } finally {
61              try { if (br != null) {br.close();}} catch (IOException ex) {
                    ex.printStackTrace(); }
62          }
63      }
```

Visualización de Ruta

Una vez obtenido el ordenamiento de los pedidos con ayuda del algoritmo evolutivo proporcionando una ruta a seguir, el paso siguiente es referenciar está ruta en un mapa para que pudiera ser entendida y utilizada por el transportista. Teniendo el KML con las coordenadas de la ruta, procedimos a importar el archivo a Google Maps para poder visualizar dicha información, además creamos

una página web importando este mapa para que el transportista pudiera acceder a la página mediante su dispositivo móvil con conexión a Internet. En la Figura 7.18 se muestra la página. El transportista podía visualizar esta página, solo con acceder desde su navegador a http://misrutas.webs.com

Página web con Ruta

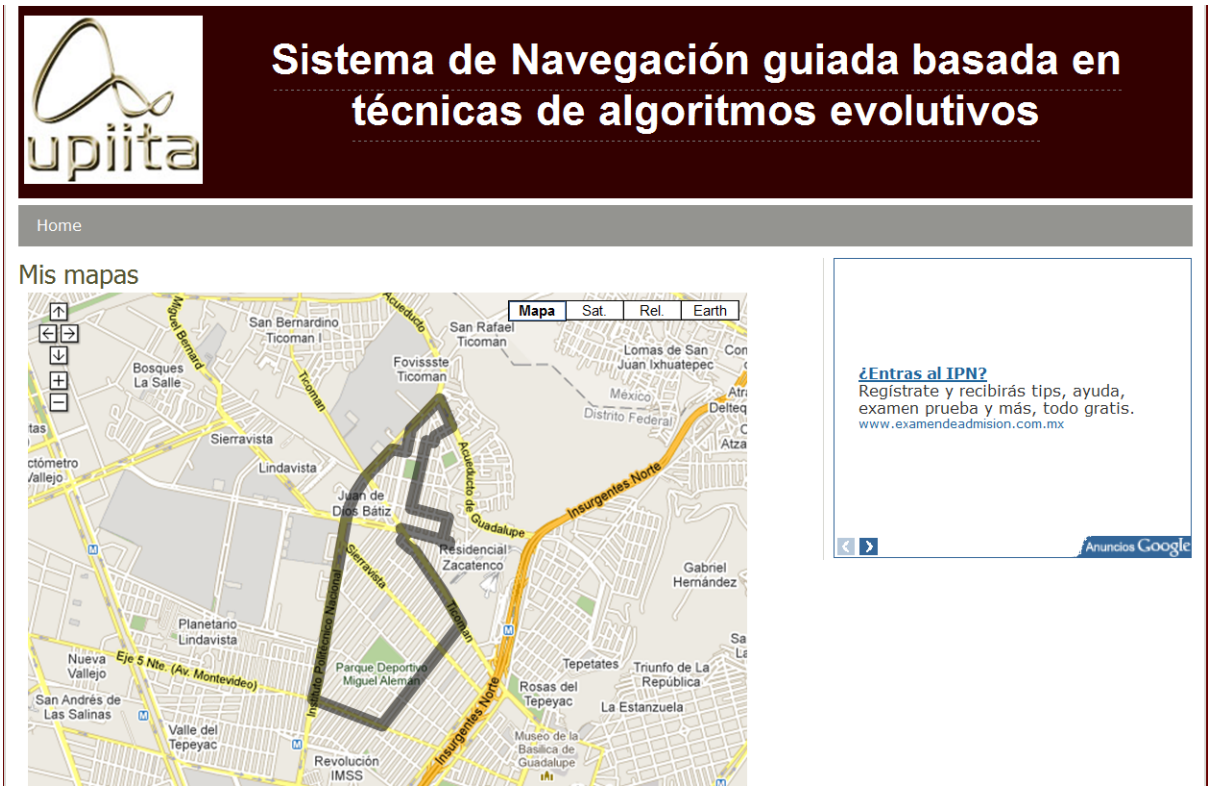


Figura 7.18: Página web con Ruta

7.7. Implementación en código JAVA

En la implementación ocupamos el lenguaje de programación java con un jdk 6.0, bajo el IDE(*Integrated Development Environment*), y partimos por crear una clase denominada `Individuo`, donde esta tiene como atributos un arreglo de tipo `int` y la aptitud como de tipo `double`, y con ella podremos crear una población con estas caracteristicas. Nombre de la clase `public class Individuo`:

```
1  public class Individuo {
2   private int  individuo[];
3   private double aptitud=0;
4
5       public void setIndividuo(int ind[]){
6          individuo=new int[ind.length];
7           for (int i=0;i<ind.length;i++)
8                individuo[i] = ind[i];
9       }
10
11      public void setHijo(int valInd,int pos){
12          individuo[pos] = valInd;
13      }
14
15      public void setAptitud(double apt){
16          aptitud=apt;
17      }
18
19      public int[] getIndividuo(){
20              return individuo;
21      }
22
23      public double getAptitud(){
24          return aptitud;
25      }
26  }
```

`public int[] creaPoblacionInicial(int mapEnt[], int numPob)`: Crea la población inicial que necesita el AG, para poder realizar las demas funciones

```
 public int[] creaPoblacionInicial(int mapEnt[], int numPob)
    {
        Random rng = new Random();
        int aux=mapEnt[0];
        int k=0,n=0;
                n = mapEnt.length;
                while (n > 1 ) {
                        k = rng.nextInt(n);
                        n--;
                        int temp2 = mapEnt[n];
                        mapEnt[n] = mapEnt[k];
                        mapEnt[k] = temp2;
                }
        return mapEnt;

    }
```

`public int numAleatorio(int tam)`: Este método como su nombre lo indica genera valores aleatorios dentro de los parametros que recibe.

```
public int numAleatorio(int tam)
    {
        int randomPosition=0;
        Random rgen = new Random();  //SE GENERA UN NUMERO ALEATORIO
        for (int i=0; i<tam; i++) {
            randomPosition = rgen.nextInt(tam);
        }
        return randomPosition;
    }
}
```

`public Individuo[] seleccionTorneo(Individuo poblacion[],int numPob):`
Realiza la selección por torneo que consiste en barajar al individuo en especifico y seleccionar a uno de ellos dependiendo de su valor de aptitud y asi generar una nueva generación.

```
1   public Individuo[] seleccionTorneo(Individuo poblacion[],int numPob)
2     {
3          Individuo[] newPoblacion=new Individuo[numPob];
4          int aux[];
5          int p1 =0,p2 =0;
6
7          GA obj=new GA();
8
9          for(int i=0;i<newPoblacion.length;i++)
10         {       while(p1==p2)
11                 {
12                     p1=obj.numAleatorio(poblacion.length);
13                     p2=obj.numAleatorio(poblacion.length);
14                 }
15                  System.out.println("\nNum aleatorio 1: "+ p1 + " aptitud
                        : " + poblacion[p1].getAptitud());
16                  System.out.println("\nNum aleatorio 2: "+ p2 + " aptitud
                         : " + poblacion[p2].getAptitud());
17                  if( poblacion[p1].getAptitud()<poblacion[p2].getAptitud
                        ())
18                  {
19                     System.out.println("\nSELECCIONAMOS: "+ p1 + " con
                           aptitud: " + poblacion[p1].getAptitud());
20                     newPoblacion[i]=new Individuo();
21                     newPoblacion[i].setIndividuo(poblacion[p1].
                           getIndividuo());
22                     newPoblacion[i].setAptitud(poblacion[p1].getAptitud()
                           );
23                  }
24                  else
25                  {
```

```
26                      System.out.println("\nSELECCIONAMOS: "+ p2 + " con
                            aptitud: " + poblacion[p2].getAptitud());
27                      newPoblacion[i]=new Individuo();
28                      newPoblacion[i].setIndividuo(poblacion[p2].
                            getIndividuo());
29                      newPoblacion[i].setAptitud(poblacion[p2].getAptitud()
                            );
30                  }
31
32              p1=0;
33              p2=0;
34
35           }
36
37          System.out.print("\n NUEVA  POBLACION");
38          for(int i=0;i<newPoblacion.length;i++)
39          {
40               System.out.print("\nIndividuo ");
41              for(int j=0;j<newPoblacion[i].getIndividuo().length;j++)
42               System.out.print(newPoblacion[i].getIndividuo()[j]+" ");
43               System.out.print(" aptitud "+newPoblacion[i].getAptitud());
44               System.out.print("\n");
45          }
46          return newPoblacion;
47
48      }
```

`public Individuo elitismo(Individuo newPoblacion[]):` Esta función se encargar de evaluar a cada individuo con el fin de determinar al mejor de la generación, con el fin de que logre sobrevivir durante todo el proceso del AG.

```
 public Individuo elitismo(Individuo newPoblacion[])
     {
         int aux=0;
         double val=0;
         int pos=0;
         Individuo indElit=new Individuo();
         val=newPoblacion[0].getAptitud();
         pos=0;
       for(int i=1;i<newPoblacion.length;i++)
       {
                if(val<newPoblacion[i].getAptitud())
                {
                     aux=i;
                     System.out.println("\nIF "+val+"\n");

                }
                else
                {
                    aux=i;
                    val=newPoblacion[i].getAptitud();
                    pos=i;
                    System.out.println("ELSE "+val+"\n");
                }
       }
System.out.println("POS "+pos+"\n");

       return newPoblacion[pos];


     }
```

`public int puntosCruza(Individuo mapEnt[])`: Éste método calcula los puntos de cruza que se necesitan para realizar el PMX, tomando en cuenta la matriz de entrada para que éstos puntos se encuentren en el rango de los datos que se están manejando.

```
public int puntosCruza(Individuo mapEnt[])
     {
        Random rgen = new Random();
        int tamCadena=mapEnt[0].getIndividuo().length;
        int pf=0;
        for (int i=0; i<tamCadena; i++) {
           pf = rgen.nextInt(tamCadena);
        }
        return pf;
     }
```

`public Individuo[] pmxCrossOver(Individuo mapEnt[],int numPob)`: Método que realiza el operador de cruza, como se menciona anteriormente se empleo la cruza PMX, la cual genera dos hijos a partir de la cruza de dos padres, éstos hijos pueden llegar a tener mas de un mapeo con el fin de que no se repitan los genes dentro del individuo. La cruza se va a estar evaluando dependiendo del parametro denominado probabilidad de cruza, en nuestro caso es de .6, lo cual evaluamos por medio de una funcion flip para que se lleve a cabo o no el operador de cruza, y asi creando una nueva generación de individuos.

```
 public Individuo[] pmxCrossOver(Individuo mapEnt[],int numPob)
     {
        for(m=0 ;m<numPob;m+=2)
        {
//*******************************INICIO DE LA CRUZA
   ***************************

                if(obj.flip(probCruza))
                {
                    System.out.println("SE PUEDE HACER LA CRUZA....");
```

```
10                          System.out.println("\n\n********************"+m+"
                               **********************");
11                          while(p1==p2)
12                          {
13                               p1=obj.puntosCruza(mapEnt);
14                               p2=obj.puntosCruza(mapEnt);
15                          }
16
17                          System.out.println("\nPUNTO DE CRUZA "+m+": "+p1)
                               ;
18                          System.out.println("PUNTO DE CRUZA "+m+": "+p2);
19
20                          if(p1>p2)
21                          {
22                              limInf=p2;
23                              limSup=p1;
24                          }
25                          else
26                          {
27                              limInf=p1;
28                              limSup=p2;
29                          }
30                          System.out.println("limInf: "+limInf);
31                          System.out.println("limSup: "+limSup);
32                           p1=0;
33                           p2=0;
34
35                        hijosCruza[m]=new Individuo();  //SE CREA EL HIJO 1
36                        int inCeros[]=new int[mapEnt[m].getIndividuo().
                             length];
37                        for(int l=0;l<inCeros.length;l++)
38                            inCeros[l]=-1;
39
40                        hijosCruza[m].setIndividuo(inCeros);
41                        XHijo1=inCeros;
```

```
42
43                      hijosCruza[m+1]=new Individuo();  //SE CREA EL HIJO
                            2
44                      hijosCruza[m+1].setIndividuo(inCeros);
45                      YHijo2=inCeros;
46                      int ex=0;
47
48                     for(int j=limInf;j<=limSup;j++)
49                     {
50                          hijosCruza[m].setHijo(mapEnt[m+1].getIndividuo()
                                [j],j);
51                          ex=hijosCruza[m].getIndividuo()[j];
52                          XHijo1[j]=ex;
53
54                          ex=0;
55                          hijosCruza[m+1].setHijo(mapEnt[m].getIndividuo()
                                [j],j);
56                          ex=hijosCruza[m+1].getIndividuo()[j];
57                          YHijo2[j]=ex;
58                     }
59
60                    System.out.print("\nPadre 1: \n");
61                     for(int j=0;j<mapEnt[m].getIndividuo().length;j++)
62                         System.out.print(mapEnt[m].getIndividuo()[j]+" ")
                            ;
63
64                     System.out.print("\nIndividuo: ");
65                           for(int j=0;j<hijosCruza[m].getIndividuo().
                                length;j++)
66                             System.out.print(hijosCruza[m].getIndividuo()
                                [j]+" ");
67
68  //INCISO B) DEL PASO 3 DE LA CRUZA
69                     for(int b=0;b<hijosCruza[m].getIndividuo().length;b
                        ++)
```

```
70                          {
71                              if(obj.buscarAPmx(hijosCruza, mapEnt[m].
                                    getIndividuo()[b],m))
72                              {
73                                      while((band==0) && hijosCruza[m].
                                            getIndividuo()[b]==-1)
74                                      {
75                                          hijosCruza[m].getIndividuo()[b]=
                                                mapEnt[m].getIndividuo()[b];
76                                          band++;
77                                      }
78                                      band=0;
79                              }
80                          }
81
82                          System.out.print("\nHIJO A EVALUAR: ");
83                          for(int j=0;j<hijosCruza[m].getIndividuo().length;j
                                ++)
84                             System.out.print(hijosCruza[m].getIndividuo()[j]+
                                    " ");
85
86  //*********SE HACE EL MAPEO DE LOS ELEMENTOS CONTENIDOS EN EL SEGMENTO
        INTERCAMBIADO*****************
87                          for(int y=0;y<hijosCruza[m].getIndividuo().length;y
                                ++)
88                          {
89                             if(hijosCruza[m].getIndividuo()[y]==-1)
90                             {
91                                 int valMapeo=obj.buscarSegmentoContenido(
                                       hijosCruza,mapEnt[m].getIndividuo()[y], m)
                                       ;
92                                 hijosCruza[m].getIndividuo()[y]=valMapeo;
93                             }
94                          }
```

```
95  //************************************EL HIJO
        1*************************************************

96

97                      int arrAux[]=new int[hijosCruza[m].getIndividuo().
                            length];

98

99

100                     for(int j=0;j<hijosCruza[m].getIndividuo().length;j
                            ++)

101                     {

102                         arrAux[j]=hijosCruza[m].getIndividuo()[j];

103                     }

104

105                     dist=obj2.getDistancias(arrAux, path);//DEVUELVE EL
                            ARREGLO CON LAS DISTANCIAS ENTRE NODO Y NODO

106

107                     for(int j=0;j<dist.length;j++)

108                        fitness=fitness+dist[j];//CALCULA LA APTITUD DEL
                               INDIVIDUO

109                    hijosCruza[m].setAptitud(fitness);//INSERTA LA
                          APTITUD DEL NUEVO INDIVIDUO

110                    fitness=0;

111

112                     System.out.print("\nHIJO 1: ");

113                     for(int j=0;j<hijosCruza[m].getIndividuo().length;j
                            ++)

114                     {

115                        System.out.print(hijosCruza[m].getIndividuo()[j]+
                               " ");

116                       // System.out.print(hijosCruza[m].getAptitud());

117                     }

118

119 //*********************************CREACION DE HIJO
        2*********************************************

120                band=0;
```

```
121                 System.out.print("\n");
122                 System.out.print("\nPadre 2: \n");
123                 for(int j=0;j<mapEnt[m+1].getIndividuo().length;j++)
124                     System.out.print(mapEnt[m+1].getIndividuo()[j]+" ");
125
126                 System.out.print("\nIndividuo: ");
127                  for(int j=0;j<hijosCruza[m+1].getIndividuo().length;
                         j++)
128                     System.out.print(hijosCruza[m+1].getIndividuo()[j
                            ]+" ");
129
130                 for(int b=0;b<hijosCruza[m+1].getIndividuo().length;b++)
131                 {
132                     if(obj.buscarAPmx(hijosCruza, mapEnt[m+1].
                            getIndividuo()[b],m+1))
133                     {
134                             while((band==0) && hijosCruza[m+1].
                                 getIndividuo()[b]==-1)
135                             {
136                                 hijosCruza[m+1].getIndividuo()[b]=mapEnt[
                                     m+1].getIndividuo()[b];
137                                 band++;
138                             }
139                             band=0;
140                     }
141                 }
142
143                  System.out.print("\nHIJO A EVALUAR: ");
144                  for(int j=0;j<hijosCruza[m+1].getIndividuo().length;j++)
145                     System.out.print(hijosCruza[m+1].getIndividuo()[j]+"
                            ");
146
147                 System.out.print("\n");
148 //*********SE HACE EL MAPEO DE LOS ELEMENTOS CONTENIDOS EN EL SEGMENTO
    INTERCAMBIADO*****************
```

```
149                     for(int y=0;y<hijosCruza[m+1].getIndividuo().length;y++)
150                     {
151                        if(hijosCruza[m+1].getIndividuo()[y]==-1)
152                        {
153                             int valMapeo=obj.buscarSegmentoContenido2(
                                    hijosCruza,mapEnt[m+1].getIndividuo()[y], m+1)
                                    ;
154                             hijosCruza[m+1].getIndividuo()[y]=valMapeo;
155                        }
156                     }
157 //************************************EL HIJO
    2*************************************************
158                     System.out.print("\nHIJO 2: ");
159                     for(int j=0;j<hijosCruza[m+1].getIndividuo().length;j++)
160                        System.out.print(hijosCruza[m+1].getIndividuo()[j]+"
                            ");
161                     System.out.print("\n");
162
163
164                     System.out.print("\n");
165                     System.out.println("
                        ****************************************");
166
167 //*************************************TERMINA CRUZA
    ***********************************************
168                  }
169                  else//SI NO SE CUMPLE FLIP
170                  {
171                       System.out.println("NO SE PUEDE HACER LA CRUZA....")
                            ;
172                     hijosCruza[m]=new Individuo();  //SE CREA EL HIJO 1
173                     hijosCruza[m+1]=new Individuo();  //SE CREA EL HIJO 2
174                     int inCeros[]=new int[mapEnt[m].getIndividuo().length
                        ];
175                     hijosCruza[m].setIndividuo(inCeros);
```

```
176                     hijosCruza[m+1].setIndividuo(inCeros);
177
178
179                     //LOS PADRES SE PASAN A LA SIG GENERACION
180                     for(int j=0;j<mapEnt[m].getIndividuo().length;j++)
181                         hijosCruza[m].setHijo(mapEnt[m].getIndividuo()[j
                            ],j);
182
183                     for(int j=0;j<mapEnt[m+1].getIndividuo().length;j++)
184                         hijosCruza[m+1].setHijo(mapEnt[m+1].getIndividuo
                            ()[j],j);
185                 }
186             }
187         System.out.print("\n NUEVA  POBLACION");
188          for(i=0;i<hijosCruza.length;i++)
189          {
190               System.out.print("\nIndividuo ");
191              for(int j=0;j<hijosCruza[i].getIndividuo().length;j++)
192               System.out.print(hijosCruza[i].getIndividuo()[j]+" ");
193               System.out.print(" aptitud "+hijosCruza[i].getAptitud());
194               System.out.print("\n");
195          }
196
197
198
199          return hijosCruza;
200                /*  hijosCruza[i]=new Individuo();  //SE CREA EL HIJO 1
201                 hijosCruza[i].getIndividuo()[j]=mapEnt[i+1].getIndividuo
                        ()[j];
202                 hijosCruza[i].getIndividuo()[j]=mapEnt[i-1].getIndividuo
                        ()[j];*/
203
204      }
```

`public Individuo [] mutacion(Individuo hijosCruza[]):` Esta función emplea la mutación por desplazamiento, que practicamente consiste en elegir de valores dentro del rango *N/2* tomando como N el tamaño del individuo, para que sean ubicados en nuevas posiciones, recorriendo a los demas datos. Este operador tambien emplea un parametro denominado probabilidad de mutacion, la cual es muy pequeña para que por lo menos en una generación éste operador se cumpla.

```
1  public Individuo [] mutacion(Individuo hijosCruza[])
2     {
3          System.out.println("\n\n*****************INICIA****MUTACION
              ******************************");
4
5           for(int i=0;i<hijosCruza.length;i++)
6           {
7               if(obj.flip(probMutacion))
8               {
9                    System.out.println("\n\n*******************SE HACE LA
                        MUTACION*****************************");
10                   N=hijosCruza[i].getIndividuo().length;
11                   System.out.println("\nN "+N);
12                   numMut=obj.numAleatorio(N/2);
13                   if(numMut==0)numMut=1;
14                   System.out.println("\nNUM MUTACION: "+numMut);
15                   System.out.println("\n*****************HIJO A MUTAR
                        ****************************");
16                        for(int m=0;m<hijosCruza[i].getIndividuo().length
                             ;m++)
17                             System.out.print(" "+hijosCruza[i].
                                  getIndividuo()[m]);
18
19                   for(int mut=0;mut<numMut;mut++)
20                        {
21                         while(p1==p2)
22                        {
```

```
23                                  p1=obj.numAleatorio(hijosCruza[i].
                                        getIndividuo().length);
24                                  p2=obj.numAleatorio(hijosCruza[i].
                                        getIndividuo().length);
25                              }
26                              if(p1<p2)
27                              {
28                                  d1=p2;
29                                  d2=p1;
30                              }
31                              else
32                              {
33                                  d1=p1;
34                                  d2=p2;
35                              }
36
37                              System.out.println("\nPos1*: "+d1);
38                              System.out.println("Pos2*: "+d2);
39
40
41                              valor1=hijosCruza[i].getIndividuo()[d1];
42                              valor2=hijosCruza[i].getIndividuo()[d2];
43
44                              for(int j=hijosCruza[i].getIndividuo().length-1;j
                                    >=0;j--)
45                              {
46                                  if(j<d2)
47                                  {
48                                      arrAux[j]=hijosCruza[i].getIndividuo()[j
                                            ];
49                                  }
50                                  else if(j==d1)
51                                  {
52                                      aux=hijosCruza[i].getIndividuo()[d2];
```

```
53                                  arrAux[d2]=hijosCruza[i].getIndividuo()[
                                        d1];
54                              }
55                              else if(j==d2)
56                              {
57                                  arrAux[d2+1]=aux;
58                              }
59                              else if(j>d1)
60                              {
61                                  arrAux[j]=hijosCruza[i].getIndividuo()[j
                                        ];
62                              }
63                              else
64                              {
65                                  arrAux[j+1]=hijosCruza[i].getIndividuo()[
                                        j];
66                              }
67                          }
68
69                          for(int m=0;m<hijosCruza[i].getIndividuo().length
                                ;m++)
70                          {
71                             hijosCruza[i].setIndividuo(arrAux);
72                             System.out.print(" "+hijosCruza[i].
                                  getIndividuo()[m]);
73                          }
74
75
76                      //  System.out.println("\n*****************HIJO EN
                            MUTACION*****************************");
77                          for(int m=0;m<hijosCruza[i].getIndividuo().length
                                ;m++)
78                                  System.out.print(" "+hijosCruza[i].
                                        getIndividuo()[m]);
79
```

```
80                          for(int m=0;m<hijosCruza[i].getIndividuo().length;
                                m++)
81                              System.out.print(" "+hijosCruza[i].
                                    getIndividuo()[m]);
82                          System.out.println("\n
                                ***********************************************
                                ");
83                          p1=0;
84                          p2=0;
85                      }
86
87                  System.out.println("\n*****************HIJO FINAL
                        ******************************");
88                          for(int m=0;m<hijosCruza[i].getIndividuo().length
                                ;m++)
89                              System.out.print(" "+hijosCruza[i].
                                    getIndividuo()[m]);
90                    numMut=0;
91                    numMut2=0;
92
93                      arrAux2=new int[hijosCruza[i].getIndividuo().length
                            ];
94                      for(int j=0;j<hijosCruza[i].getIndividuo().length;j
                            ++)
95                      {
96                          arrAux2[j]=hijosCruza[i].getIndividuo()[j];
97                      }
98
99                      dist=obj2.getDistancias(arrAux2, path);//DEVUELVE EL
                            ARREGLO CON LAS DISTANCIAS ENTRE NODO Y NODO
100                     System.out.print("[ ");
101                     for(int j=0;j<dist.length;j++)
102                     {
103                       System.out.print(" "+dist[j]);
```

```
104                         fitness=fitness+dist[j];//CALCULA LA APTITUD DEL
                                INDIVIDUO
105                       }
106                       System.out.print(" ]");
107                      hijosCruza[i].setAptitud(fitness);//INSERTA LA
                            APTITUD DEL NUEVO INDIVIDUO
108                      fitness=0;
109
110                       arrAux2=null;
111          }
112
113          System.out.println("\n\n*****************NO**SE HACE LA MUTACION
                 ******************************");
114          System.out.println("\n*****************HIJO NO MUTADO
                 ******************************");
115                       for(int m=0;m<hijosCruza[i].getIndividuo().length
                             ;m++)
116                          System.out.print(" "+hijosCruza[i].
                                getIndividuo()[m]);
117          System.out.println("\n*****************HIJO FINAL NO MUTADO
                 ************************");
118                       for(int m=0;m<hijosCruza[i].getIndividuo().length
                             ;m++)
119                          System.out.print(" "+hijosCruza[i].
                                getIndividuo()[m]);
120                      arrAux2=new int[hijosCruza[i].getIndividuo().length
                             ];
121                      for(int j=0;j<hijosCruza[i].getIndividuo().length;j
                             ++)
122                      {
123                           arrAux2[j]=hijosCruza[i].getIndividuo()[j];
124                      }
125
126                      dist=obj2.getDistancias(arrAux2, path);//DEVUELVE EL
                            ARREGLO CON LAS DISTANCIAS ENTRE NODO Y NODO
```

```
127                         System.out.print("[ ");
128                         for(int j=0;j<dist.length;j++)
129                         {
130                           System.out.print(" "+dist[j]);
131                           fitness=fitness+dist[j];//CALCULA LA APTITUD DEL
                                  INDIVIDUO
132                         }
133                         System.out.print(" ]");
134                        hijosCruza[i].setAptitud(fitness);//INSERTA LA
                              APTITUD DEL NUEVO INDIVIDUO
135                        fitness=0;
136
137                          arrAux2=null;
138
139         }
140 fitness=0;
141         System.out.println("\n*****************FINAL
                ****************************");
142                for(int i=0;i<hijosCruza.length;i++)
143                {
144                   for(int m=0;m<hijosCruza[i].getIndividuo().length;m++)
145                        System.out.print(" "+hijosCruza[i].getIndividuo()
                              [m]);
146                        System.out.print(" "+hijosCruza[i].getAptitud());
147                        System.out.print("\n");
148                }
149         System.out.println("\n\n****************TERMINA******MUTACION
                ****************************");
150        return hijosCruza;
151     }
```

Parte V

Pruebas Adicionales

7.8. Diferentes Pruebas

A lo largo del desarrollo del proyecto se realizaron diferentes pruebas en las áreas relacionadas a este, una de estas áreas fue la creación de aplicaciones para los dispositivos móviles. Ante la necesidad de visualizar imagenes en el dispositivo, buscamos la manera de poder manipular, analizar y visualizar imagenes correspondientes a la Delegación Gustavo A. Madero, por lo que se uso diferentes programas que nos dan la oportunidad de descargar imagenes de Google Maps sin ningún costo, algunos de los programas utilizados fueron gMapMarker, Google Maps Images Downloader entre muchos otros. El utilizado fue gMapMarker dado que descargaba las imagenes con un nivel de zoom bastante alto (17), solo hay que indicarle las coordenadas de izquierda a derecha, arriba y abajo de la zona que deseamos obtener los mapas, y guarda las imagenes en una carpeta de salida, en la Figura 7.19 ejemplifica la configuración de la descarga

Configuración GMapMarker

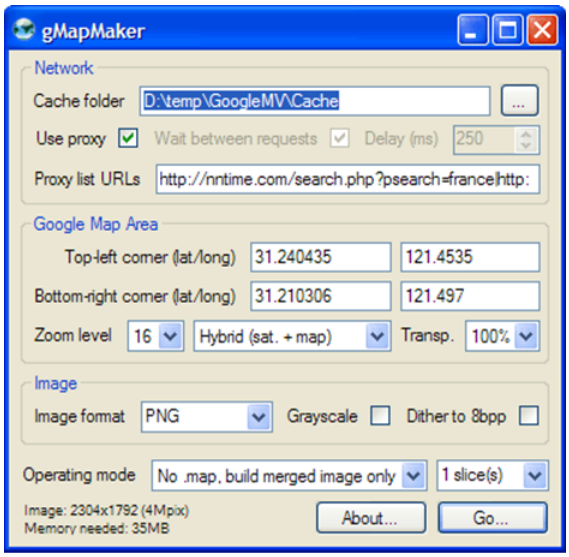


Figura 7.19: Imagenes descargadas

A continuación se muestra un ejemplo de las imagenes obtenidas en la la Figura 7.20

La siguiente tarea era manipular estas imagenes de tal manera que pudieran ser vistas en el dispositivo y unirlas como un rompecabezas de tal manera que se tuviera el mapa completo. En la siguiente parte de código se muestra como se podía manipular las imagenes.

Imagenes descargadas

Figura 7.20: Imagenes descargadas

```
1
2  import java.io.*;
3  import javax.microedition.lcdui.*;
4  import javax.microedition.midlet.*;
5  import javax.microedition.media.*;
6  import javax.microedition.media.Control.*;
7
8
9  public class Imagen extends Canvas implements CommandListener {
10 private MoverImagen midlet;
11 private Image im,im2;
12 private Command salir;
13 private int x, y;
14 public Imagen(MoverImagen mid){
15 salir = new Command(''Salir",Command.EXIT,1);
16 this.midlet = mid;
17 this.addCommand(salir);
18 this.setCommandListener(this);
19 try{
```

```
20  im = Image.createImage(''/1.png");
21  im2= Image.createImage(''/2.png");
22  }
23      catch (Exception e){
24         System.out.println(''Error al cargar archivo de imagen");
25       }
26       x = 0;
27       y = 0;
28    }
29   public void paint(Graphics g) {
30      g.setColor(255,255,255);
31      g.fillRect(0,0,getWidth(),getHeight());
32      g.setColor(0,0,0);
33      g.drawImage(im,x,y,Graphics.VCENTER|Graphics.LEFT);
34      g.drawImage(im2,x,y,Graphics.VCENTER|Graphics.RIGHT);
35      }
36     protected void keyPressed(int keyCode) {
37      switch(getGameAction(keyCode)){
38          case Canvas.DOWN: { if ((y+40)<getHeight ()) y = y+1;
39                        break; }
40           case Canvas.LEFT: { if (x> 0) x = x-1;
41                        break;}
42           case Canvas.UP: { if (y> 0) y = y-1;
43                         break;}
44           case Canvas.RIGHT:{ if ((x+40) <getWidth()) x = x+1;
45                        break;}
46       }
47        this.repaint();
48      }
49     public void commandAction(Command c, Displayable d){
50          if (c == salir){
51              midlet.salir();
52         }
53      }
54  }
```

El anterior código no muestra como salida la siguiente pantalla que se observa en la Figura 7.21, recordemos que NetBeans tiene un emulador propio J2ME donde se pueden correr las aplicaciones para móviles

Aplicación emulada Visualización Mapas

Figura 7.21: Aplicación emulada Visualización Mapas

Otras de las aplicaciones que se llevaron a cabo fue con sonido, asi podíamos escuchar algunos mensajes y grabaciones . Para esto se hizo uso de la librería de Java J2ME Media. A continuación ponemos el código de una aplicación donde fue utilizada.

```
1
2
3  import java.io.IOException;
4  import javax.microedition.lcdui.Display;
5  import javax.microedition.lcdui.*;
6  import java.io.*;
7  import javax.microedition.media.*;
8  import javax.microedition.midlet.*;
9  import javax.microedition.lcdui.*;
10
11
12
13     public class Hola extends MIDlet  implements CommandListener{
14
15     Display pantalla;
16     Form ventana1, ventana2, ventana3;
17     Command botonS,botonSig,botonC,atras;
18     StringItem cad1,cad2, cad3;
19     private Form coordenadas;
20
21     public Hola(){
22
23     pantalla= Display.getDisplay(this);
24     ventana1= new Form(''Inicio Repartidor");
25     ventana2= new Form(''Mapa Ruta");
26     ventana3= new Form(''Obtener Coordenadas");
27
28
29     botonS= new Command(''Salir", Command.EXIT, 1);
30     botonSig= new Command( ''Ruta",Command.OK,2);
31     botonC= new Command(''Ubicaci\'on", Command.OK,2);
```

```
32      atras= new Command(''Back", Command.BACK,1);
33
34      cad1= new StringItem('' ", ''Bienvenido Repartidor");
35      cad2= new StringItem('' ", ''La ruta a seguir es");
36      cad3= new StringItem(''", ''Tu ubicaci\'on es");
37
38      ventana1.append(cad1);
39      ventana1.addCommand(botonS);
40      ventana1.addCommand(botonSig);
41      ventana1.addCommand(botonC);
42
43      ventana2.append(cad2);
44      ventana2.addCommand(botonS);
45      ventana2.addCommand(atras);
46
47      ventana1.setCommandListener(this);
48      ventana2.setCommandListener(this);
49      ventana3.setCommandListener(this);
50
51
52
53      }
54
55      public void startApp() {
56
57      pantalla.setCurrent(ventana1);
58      }
59
60      public void pauseApp() {
61      }
62
63      public void destroyApp(boolean unconditional) {
64      this.notifyDestroyed();
65      }
66
```

```
67      public void commandAction(Command boton, Displayable d)
68
69      {
70        if(boton==botonS)
71        {
72           notify Destroyed();
73        }
74
75
76        if(boton==botonSig)
77        {
78        Audio("/Ruta.wav");
79        pantalla.setCurrent(ventana2);
80        }
81
82        if(boton==atras)
83        {
84        pantalla.setCurrent(ventana1);
85        }
86
87
88
89        if(boton==botonC)
90        {
91
92            if (coordenadas == null){
93          Audio("/Ubicacion.wav");
94          Display display = Display.getDisplay(this);
95          display.setCurrent(new LocationMainForm(this));
96          }
97        }
98
99      }
100
101
```

```
102
103   public void Audio (String menu){
104
105          try {
106                   InputStream in = null;
107                   in = getClass().getResourceAsStream(menu);
108                   Player sonido = Manager.createPlayer(in, ''audio/x-wav");
109                        try {
110                              sonido.start();}
111                              catch (MediaException me) { }
112                             }
113
114                             catch(Exception e){}
115                                 }
116                           }
```

Un último ejemplo de la aplicaciones realizadas, fue obtener las coordenadas geográficas usando el GPS del dispositivo móvil, esto se llevo a cabo gracias a la librería de Java J2ME Location, como lo muestra el código siguiente.

```
1
2
3  import javax.microedition.lcdui.*;
4  import javax.microedition.location.*;
5  import javax.microedition.midlet.MIDlet;
6
7
8  public class LocationMainForm extends Form implements CommandListener,
       LocationListener {
9
10     //Distancia total recorrida actualmente
11     private double totalCurrentDistance;
12     private MIDlet midlet;
13
14     //Coordenadas
15     private QualifiedCoordinates lastCoord;
16
17     //Controles de la interfaz
18     private Command   cmdStartClick;
19     private Command   cmdExitClick;
20
21     private TextField txtEstadoProveedor;
22     private TextField txtLongitud;
23     private TextField txtLatitud;
24     private TextField txtDistanciaRecorrida;
25     private TextField txtVelocidad;
26     private TextField txtCalle;
27
28
29     public LocationMainForm(MIDlet    midlet){
30         super(''Location API Ejemplo");
```

```
31
32          this.totalCurrentDistance = 0;
33          this.midlet               = midlet;
34
35          this.createUI();
36      }
37
38      /**
39       *  Contruimos y configuramos el interfaz
40       */
41      private void createUI(){
42          this.cmdStartClick = new Command(''Empezar", Command.OK,   1);
43          this.cmdExitClick  = new Command(''Salir",   Command.EXIT, 1);
44
45          this.txtCalle      = new TextField(''Calle:",           null, 50,
                 TextField.ANY);
46          this.txtEstadoProveedor     = new TextField(''Proveedor:",
                null, 50,  TextField.ANY);
47          this.txtDistanciaRecorrida  = new TextField(''Metros recorridos:"
                , null, 50,   TextField.ANY);
48          this.txtVelocidad  = new TextField(''Velocidad:", null, 20,
                TextField.ANY);
49          this.txtLongitud   = new TextField(''Longitud:",        null, 50,
                  TextField.ANY);
50          this.txtLatitud    = new TextField(''Latitud:",         null, 50,
                  TextField.ANY);
51
52          this.append(txtEstadoProveedor);
53          this.append(txtDistanciaRecorrida);
54          this.append(txtVelocidad);
55          this.append(txtCalle);
56          this.append(txtLongitud);
57          this.append(txtLatitud);
58
59          this.addCommand(cmdStartClick);
```

```
60          this.addCommand(cmdExitClick);
61          this.setCommandListener(this);
62      }
63
64      /**
65       * Configuramos e iniciamos el proveedor de localizaci\'on
66       */
67
68
69      private void cmdStartClick_click() throws LocationException   {
70          // Establecemos los criterios del proveedor de localizacion
                deseado.
71          Criteria criteria = new Criteria();
72          criteria.setCostAllowed(false);
73
74
75
76          criteria.setAddressInfoRequired(true);
77
78
79          criteria.setSpeedAndCourseRequired(true);
80
81          // Solicitamos el proveedor de localizaci\'on
82          LocationProvider provider = LocationProvider.getInstance(criteria
                );
83
84
85          if (provider  == null){
86
87              criteria.setAddressInfoRequired(false);
88
89
90              provider = LocationProvider.getInstance(criteria);
91          }
92
```

```
93
94          if (provider  != null){
95              try {
96
97                  Location location = provider.getLocation(-1);
98
99
100                 if (location != null){
101                     this.lastCoord    = location.getQualifiedCoordinates
                            ();
102
103
104                     provider.setLocationListener(this, 5, -1, -1);
105                 }
106
107                 // Inicializamos el UI con los datos del estado del
                        proveedor
108                 this.providerStateChanged(provider, provider.getState());
109             } catch (java.lang.InterruptedException ex){
110                 this.append(ex.getMessage());
111             }
112         }
113     }
114
115     /**
116      * Finalizamos la aplicaci\'on
117      */
118     private void cmdExitClick(){
119         this.midlet.notifyDestroyed();
120     }
121
122
123     public void commandAction(Command c, Displayable d) {
124         try {
125             if (c == this.cmdStartClick){
```

```
126                 this.cmdStartClick_click();
127             } else if (c == this.cmdExitClick){
128                 this.cmdExitClick();
129             }
130         } catch (Exception ex){
131             this.append(ex.getMessage());
132         }
133     }
134 
135 
136     public void locationUpdated(LocationProvider provider, Location
        location)  {
137         try {
138             QualifiedCoordinates coord     = location.
                getQualifiedCoordinates();
139             AddressInfo          addr      = null;
140 
141             if (coord != null){
142                 this.txtLongitud.setString(String.valueOf(coord.
                    getLongitude()));
143                 this.txtLatitud.setString(String.valueOf(coord.
                    getLatitude()));
144 
145                 addr = location.getAddressInfo();
146             }
147 
148             if (addr != null){
149                 this.txtCalle.setString(addr.getField(AddressInfo.STREET)
                    );
150             } else {
151                 this.txtCalle.setString(''INFO NO DISPONIBLE");
152             }
153 
154 
155             totalCurrentDistance += Math.abs(coord.distance(lastCoord));
```

```
156
157
158             this.txtDistanciaRecorrida.setString(String.valueOf(
                    totalCurrentDistance));
159
160             this.txtVelocidad.setString(String.valueOf(location.getSpeed
                    ()));
161             this.lastCoord = coord;
162         } catch (Exception ex){
163             this.append(ex.getMessage());
164             // Nada
165         }
166     }
167
168
169     public void providerStateChanged(LocationProvider provider, int
            newState) {
170         String stateDesc = null;
171
172         if (newState == LocationProvider.AVAILABLE){
173             stateDesc = "DISPONIBLE";
174         } else if (newState == LocationProvider.OUT_OF_SERVICE){
175             stateDesc = "FUERA DE SERVICIO";
176         } else if (newState == LocationProvider.TEMPORARILY_UNAVAILABLE){
177             stateDesc = "TEMPORALMENTE FUERA DE SERVICIO";
178         } else {
179             stateDesc = "DESCONOCIDO";
180         }
181
182         this.txtEstadoProveedor.setString(stateDesc);
183     }
184 }
```

EL código anterior nos da como resultado la emulación en J2ME con los diferentes campos del la ubicación en coordenadas Latitud y Longitud como lo muestra la Figura 7.22

Aplicación emulada Obtención de Coordenadas

Figura 7.22: Aplicación emulada Obtención de coordenadas

Parte VI

Conclusiones

CAPÍTULO 8

Conclusiones

Durante este proyecto, enfrentamos muchos problemas de tal manera que se tuvo que hacer una extensa investigación y con ello pudimos aprender mucho sobre diferentes temas, como lo fue desde la parte web, la parte de J2ME, Postgis, Quantum Gis que te da un panorama general de todas las herramientas que se pueden explotar y en la cual fue de gran utilidad para ciertas actividades que necesitabamos, pero consideramos que el área más complicada fue comprender la parte del algoritmo genetico, y gracias a toda la documentación recolectada te das cuenta de toda la gama de aplicaciones en la que esta área de estudio se encuentra involucrada.

La recolección de datos fue complicada al ingresar los nodos uno por uno de forma que se implementarón aplicaciones para facilitar la adquisición de éstos.

Dentro del algoritmo genético tuvo muchas complicaciones debido al saber cómo aplicar toda la teoría en una codificación que nos determine resultados, éstos resultados fuerno positivos en los casos expuestos se logra llegar a una distancia miníma para todo el recorrido, cumpliendo con el objetivo de encontrar la distancia mínima del recorrido total, por otra parte, los resultados tuvieron comportamientos sorprendentes ya que se visualizó la toma de retornos para llegar a un punto en especifico, llegar de un punto a atro utilizando vías alternas que llevaron a una buena generación de itinerarios.

De alguna u otra manera el implementar algoritmos genéticos da cabida a que más adelante se implemente para el mismo tema de rutas pero con un mayor número de parametros, por ejemplo,

número de semaforos, tiempo, tráfico, etc, así como otra aplicaciones de optimización, como en la señalizacion de routers.

Finalmente, gracias a la ayuda de muchas herramientas, aplicaciones, estudio e investigación se pudo llevar a cabo el proyecto presente.

Karina Zamudio Alonso

En el presente proyecto se tuvo que realizar una amplia investigación en diferentes áreas, donde se entendió la complejidad de las herramientas y teoría acerca de estas. Al principio se tenía en mente un proyecto bastante ambicioso, pero se fue aterrizando conforme la marcha de la implementación.
Uno de los primeros retos a los que nos enfrentamos fue recaudar toda la información geográfica necesaria del área de estudio de manera manual, y a pesar de que se utilizó una base de datos de la delegación Gustavo A. Madero, esto no minorizo el trabajo de recolección, ya que dicha base de datos solo nos ayudo para corroborar que la información recolectada era correcta.
Otro parte que fue una de las más importantes y difíciles fue llevar a cabo el algoritmo evolutivo, dado que era un tema totalmente desconocido y complejo, sus fundamentos teóricos son una parte complicada de abordar y llevarlo a la práctica. Pero ahora se puede concluir cuales son las ventajas de un algoritmo evolutivo, no es mejor por sí solo a diferencia de otros algoritmos más conocidos que sacan rutas, la diferencia radica en que los algoritmos evolutivos hacen uso de otros algoritmos como Floyd, Dijkstra, para poder realizar sus funciones que son hacer un ordenamiento optimizado para no hacer recorridos redundantes o innecesarios, una de las primeras condiciones que establece el algoritmo evolutivo es trabajar en un área de estudio grande, por que como vaya incrementando el área, los resultados se vuelven más eficientes a diferencia de otros algoritmos.
Por la parte de los dispositivos móviles, recordemos que a pesar de que la tecnología crece de manera impresionante incluso para estos dispositivos, seguimos teniendo limitantes de espacio en memoria y ejecución para algunas tareas, ya que no cuentan con el mismo poder de procesamiento que una computadora, pero no se duda que en un trabajo a futuro pueda realizarse aplicaciones más complejas.

Oriana Santillán Moncayo

Parte VII

Bibliografia

Bibliografía

[1] Ian Sommerfield : Ingeniería de Software. Pearson Addison Wesley, United Kingdom 2004.

[2] Alberk KW Yeung, G. Brent Hall.Spatial Database Systems, Springer.

[3] Martin Fowler. UML Distilled 3rd edition. Pearson Addison Wesley.

[4] Elmasri Navathe, Sistemas de bases de datos, Addison-Wesley Iberoamericana, Segunda edición.

[5] Dr. Carlos A. Coello Coello: Introducción a la Computación Evolutiva (Notas de Curso), CINVESTAV-IPN, Mayo, 2008.

[6] Zbigniew Michalewicz: Genetic Algorithms + Data Structures = Evolution Programs,Springer, Third, Revised and extended edition.

[7] Ellis Horowitz, Sartaj Sahni: Fundamentals of computer algorithms, Computer Science Press, Hardcover, September 1984.

[8] Patricia Jorge Cárdenas, Agustín Froufe Quintas, J2ME. Java 2 Micro Edition. Manual de usuario y tutorial

[9] `http://struts.apache.org/`

[10] `http://postgis.refractions.net/`

[11] http://maps.google.com.mx/

[12] http://www.postgresql.org/docs/8.1/

[13] http://netbeans.org/

[14] http://www.forum.nokia.com//

[15] http://code.google.com/

[16] http://java.sun.com/