UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

"APLICACIÓN DEL ALGORITMO DE DIJKSTRA EN UN SISTEMA WEB DE GENERACIÓN DE RUTAS PARA CIUDAD UNIVERSITARIA"

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA

JOSÉ LUIS ITURBIDE LÓPEZ

DIRECTOR DE TESIS: ING. CARLOS ALBERTO ROMÁN ZAMITIZ

CIUDAD UNIVERSITARÍA, MÉXICO DF. 2006

A mi pequeña Michelle
A mi linda esposa Giovanna
A mis padres Paula y Luis
A Tere, Raúl e Israel
A mi Universidad a la que tanto debo

ÍNDICE

CAPÍTULO 1. INTRODUCCIÓN

Propósito del trabajo

Nuestra Universidad, además de importante y reconocida en el mundo, es una de las más grandes en tamaño de instalaciones no sólo de América sino del mundo.

El campus central de la UNAM se localiza en la Ciudad Universitaria, al sur de la Ciudad de México. Este campus forma parte importante de la historia de México, alberga la mayoría de las facultades y escuelas de la universidad, un amplio número de institutos, centros de investigación, dependencias y recintos de difusión cultural, y las principales oficinas de gobierno y administración de toda la institución. Se tienen instalaciones deportivas que incluyen una alberca olímpica, un estadio olímpico y varios centros de prácticas para la recreación de los estudiantes.

Ciudad Universitaria tiene una extensión de más de 3 km^2. Eso es más de lo que miden muchas ciudades importantes en Europa e incluso lo que miden algunos de los países más pequeños como el Vaticano (0.2 millas cuadradas) o Mónaco (0.7 millas cuadradas).[1]

Fig. La Biblioteca Central, el edificio más representativo de la UNAM.

El presente trabajo se refiere a un sistema de información que cubre la necesidad de conocer la ruta de recorrido mas corta entre dos instalaciones de Ciudad Universitaria (CU). Actualmente se tienen mapas de localización de algunas instalaciones de CU, pero no hay alguna que contenga todos ni tampoco que informe los puntos que hay que recorrer para llegar al destino.

La solución a esta necesidad es un sistema de información Web que integra un conjunto de conocimientos matemáticos, de programación y de desarrollo de sistemas. Específicamente haciendo uso del algoritmo de Dijkstra de teoría de grafos que es muy conocido en el medio académico para resolver problemas de redes.

Los algoritmos de solución a problemas de teoría de grafos son muy útiles y los campos de aplicación son muchos: transporte, redes de comunicación, flujos de trabajo, mensajería, logística, etc. En todos ellos se busca optimizar el esfuerzo de como realizar un recorrido haciendo el menor tiempo, recorriendo la menor distancia o tocando el número máximo de puntos de interés.

[1] Wikipedia, http://es.wikipedia.org/wiki/UNAM

En resumen el propósito del presente trabajo es el de proporcionar a la comunidad universitaria y a el público en general un sistema útil y que aplique los conocimientos de la carrera de Ingeniero en Computación.

Organización del texto

El texto esta organizado de forma similar a un proyecto real de sistemas de información, se plantea la necesidad y se desarrollan las fases de análisis, diseño, construcción y pruebas del sistema. Estas fases se adaptan a los requisitos de contenido que exige el trabajo de titulación.

Para una lectura mas fluida se recomienda contar con conocimientos en Notación Matemática notación de algoritmos, conocimientos de Análisis y Diseño Orientado a Objetos, Patrones de Diseño y Java; conocimientos básicos en Procesos de Desarrollo de Software ya que se mencionan algunos términos de estos temas.

Descripción del contenido del trabajo.

El texto esta organizado en 7 capítulos y 3 anexos.

En el capítulo 2, ***Definición del Problema***, Se enuncia el objetivo del presente trabajo que es el resolver el problema de la falta de información detallada de cómo trasladarse de un punto a otro dentro del campus de Ciudad Universitaria. Se propone resolver el problema con el desarrollo de un sistema de información que aplique el algoritmo de Dijkstra como forma de solución a la búsqueda de la ruta mas corta entre 2 puntos.

En el capítulo 3, de ***Análisis***, se listan los requisitos que debe cubrir el sistema. Ya que no hay la participación de un usuario o cliente que establezca los requisitos por ser un trabajo académico, enuncio los requerimientos de la funcionalidad básica que un sistema de rutas debe cubrir por cuenta propia. Así mismo se delimita el alcance del sistema mencionando que hará y que no hará.

Se hace mención del proceso de desarrollo a seguir, es decir los pasos se realizarán fase por fase para la construcción del sistema.

También se realiza una breve explicación de la teoría grafos y una descripción del pseudo código del algoritmo de Dijkstra. Se hace una introducción mas detallada de la Teoría de Grafos y de algoritmos sobre grafos en el Anexo “Teoría de grafos” al final del texto.

En el capítulo 4, ***Diseño***, En este capítulo se elabora un diseño de cada componente que conforma el sistema. El diseño de los componentes es orientado a objetos y se expresa con modelos de UML. Aunque se menciona el término clase java, el diseño se elabora tratando de ser independiente de la tecnología a usar para su construcción.

El diseño de la base de datos se hace con tradicional diseño relacional de bases de datos.

En el capítulo 5, ***Recursos tecnológicos elegidos***, se realiza una selección en base a criterios de la herramienta de construcción para cada componente y capa del sistema, de entre todas las opciones tecnológicas que se dispone. Este ejercicio es el parecido al que se realiza en una estimación de esfuerzos donde se evalúa, de entre otros, la tecnología más rentable para el desarrollo de un proyecto.

En el capítulo 6, ***Construcción***, se parte del hecho de que el sistema esta construido, funciona y a manera de manual de usuario se describe la función de cada componente construido, con sus controles, campos de captura y una imagen ilustrativa. En este capítulo no se transcribe el código codificado, pero todo el código fuente del sistema se incluirá en el CD de esta tesis.

Los archivos de configuración y el código fuente del proceso de solución de búsqueda de la Ruta si se transcribe y explica.

En el capítulo 7, ***Resultados y conclusiones***, se realizan las pruebas propuestas en la sección de análisis para la comprobación de que el sistema arroja resultados correctos. En estas pruebas se centran en los resultados que arroja el sistema y no en los detalles del funcionamiento del sistema relacionados a la usabilidad o validaciones.

Se describen los datos de prueba usados para la carga inicial de datos, se analizan los resultados y se enuncian las conclusiones.

Finalmente se mencionan los posibles cambios sobre el sistema para ampliar su funcionalidad y realizar mejoras.

En el anexo ***Teoría de Grafos***. Se proporciona una introducción a los grafos, los elementos que componen un grafo, se describe la notación matemática usada para describir un grafo y los principales algoritmos usados para dar solución a problemas sobre estos.

En el anexo de ***Referencias Bibliográficas*** se listan los recursos consultados. Gracias a la era del Internet es posible de obtener rápidamente información actualizada y de gran contenido. La mayor parte del material consultado se obtuvo de Internet y en menor proporción de libros. Esto presenta una ventaja ya que puede invertirse poco tiempo y recursos para obtener información útil pero también esta facilidad exige una responsabilidad en que el producto sea algo nuevo y no una copia sin aportación.

En el anexo ***Glosario***, se listan algunas definiciones para aclarar el significado de términos usados en el texto.

CAPÍTULO 2. DEFINICIÓN DEL PROBLEMA

Diariamente una infinidad de personas recorren Ciudad Universitaria por diferentes motivos, en mayor parte por estudiantes, profesores, trabajadores y personas cuya actividad depende de las actividades diarias que en ella se realizan; y en menor medida por personas que acuden por primera vez: estudiantes de otras instituciones, personas que acuden por eventos artísticos y culturales, actividades de recreación y deportivas, etc.

Para muchos de nosotros que tuvimos la necesidad de visitar las instalaciones, por que afortunadamente fuimos aceptados como universitarios, fue una expedición la primera vez que la visitábamos. Después de cierto tiempo, de ir y regresar al fin encontrábamos con el edificio que buscábamos[1]. Las personas mas precavidas hacen el recorrido un día antes y se olvidan de estos problemas.

Aquí surge la pregunta: ¿Habrá forma de que pueda saber con seguridad cómo llegar a una instalación dentro de CU sin necesidad de tener que hacer el recorrido antes o visitarla con mucho tiempo de anticipación?

Ejemplo real:

En la sala Netzhualcoyotl se presentan eventos culturales a los que acuden personas de las más diversas características. ¿Cómo llego a la sala?

Hay diferentes opciones, algunos ya las conocemos, pero quien no, se preguntará:

- Si vengo en automóvil, ¿Cual es el recorrido que debo hacer?
- Si vengo a pie ¿Donde es más recomendable iniciar el recorrido?
- De todos los caminos posibles ¿Cual es el más corto, si tengo que hacer el recorrido varias veces?
- En esta era de los sistemas Web ¿Hay un lugar en la Web de la UNAM donde pueda obtener esta información, para saber como llegar o al menos acercarme los mas posible?
- Si existe, ¿Habrá forma de obtener esta información junto con un croquis para una mejor referencia?

Revisando las cuestiones anteriores, es claro que existe la necesidad de contar con un medio que proporcione esta información.

[1] No les ocurrió que cuando quisimos registrarnos por recién ingreso a la hora de preguntar por la Facultad de Ingeniería, alguien conocedor le pregunto para darnos la información correcta, ¿Al anexo o al Principal?

Medios formales de consulta existentes

Guía-Roji

La Guía-Roji cuenta con un detalle general de la ubicación grafica del campus y algunos de los edificios más importantes, pero no cuenta con la información detallada para encontrar todas las instalaciones. Es un buen material de consulta pero no es suficiente si se desea mas detalle.

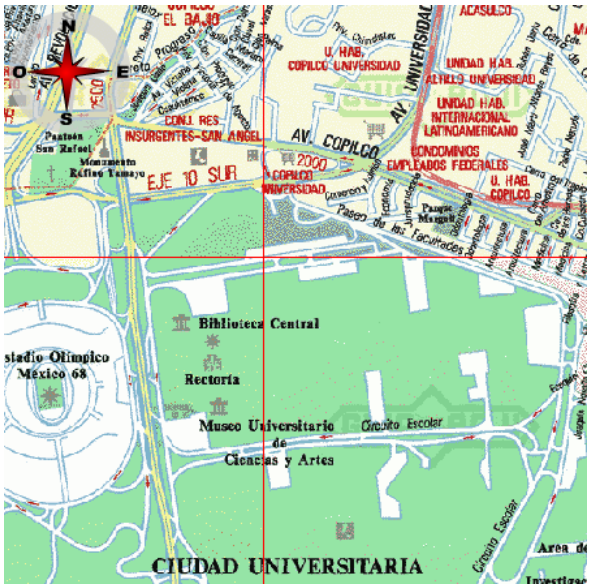


Fig. Mapa de CU en Guía-Roji

Sitio Web de la UNAM

En el sitio Web de la universidad existe un sistema de consulta para instalaciones, pero no nos dice como llegar a estas instalaciones, se tiene que imprimir los cuadrantes de los mapas para tener una buena referencia.

La UNAM dispone de un servicio de transporte universitario que cuenta con 5 rutas que cubren la mayoría de los puntos más importantes y que se encuentran a lo largo de los 2 circuitos de la Universidad. Dentro del sitio Web de la UNAM existe una guía de la red del transporte universitario. Esta información es buena como referencia pero si nuestro destino no esta dentro de estas rutas, entonces no será de mucha ayuda.

Fig. Mapa de CU en Sistema de Localización de instalaciones

Definición del problema

Se carece de un medio de consulta que proporcione la ruta mas corta de recorrido desde y hacia cualquier instalación dentro de Ciudad Universitaria. Un medio que muestre la ruta a seguir de manera gráfica y tomando en cuenta que se puede llegar a pie o desde automóvil.

Propuesta

Se propone el desarrollo de un sistema que provea esta información. Un sistema que:

- Proporcione la ruta mas corta de recorrido desde y hacia cualquier instalación dentro de Ciudad Universitaria y que esta instalación pueda ser tan detallada como una cancha, puesto de auxilio, una parada de camión,
- Muestre la información grafica y el detalle del recorrido con datos de tiempo y distancia de recorrido
- Y como condición, tal como lo describe el titulo del trabajo, que aplique el uso del Algoritmo de localización de la ruta mas corta diseñado por Edsger Dijkstra, que es uno de los algoritmos más eficientes para este tipo de problemas.

Resultados esperados

Al final del desarrollo del presente trabajo, se espera tener un sistema disponible a cualquier persona por medio del Web, donde de manera sencilla pueda obtener información de que ruta a seguir para trasladarse de un lugar a otro dentro del campus de CU.

El sistema tendrá la posibilidad de ampliarlo, optimizarlo, o basarse en el para otra aplicación o simplemente como caso de estudio con fines académicos.

CAPÍTULO 3. ANÁLISIS

A partir de este punto se dará un enfoque al trabajo como el que daría un consultor de sistemas al presentársele una necesidad de información: Conocer los requerimientos, analizarlos, elaborar una propuesta y un diseño de la solución.

Pero antes de entrar en materia del análisis del problema expongo la metodología del desarrollo de sistemas que utilizaré. Esto para establecer que el desarrollo tendrá un orden, un inicio y un alcance delimitado.

Método de desarrollo

La experiencia, la aplicación de conocimientos y el seguimiento de un método de desarrollo son importantes para que un proyecto que cumpla en contenido y tiempo. Las "buenas prácticas" no solo son aplicables a la programación sino a todo el proceso de desarrollo.

Un proceso de desarrollo de programas tiene como objetivo la formalización de las actividades relacionadas con la elaboración de sistemas informáticos. La formalización de un proceso de desarrollo tiende a dotar a las empresas de un conjunto de mecanismos que, cuando se aplican sistemáticamente, permiten obtener de manera repetitiva y fiable sistemas de programas de calidad constante. Por naturaleza, la descripción del proceso es general porque no es posible definir autoritariamente un estándar único.

Un método de desarrollo comprende:

1. **Un proceso** que describe las etapas a seguir en el desarrollo del sistema:
   Para este trabajo se usarán los elementos básicos del Proceso Unificado de Desarrollo (RUP).
   Idealmente el proceso de desarrollo se resume a las fases de Recopilación de requisitos, Análisis, Diseño y Construcción. Pero con RUP podemos hacer que el proceso sea iterativo e incremental, es decir aplicar estas fases anteriores varias veces para entregar productos pequeños funcionales y completos de manera cíclica.
   RUP hace uso de las técnicas de toma de requisitos como casos de uso y análisis y diseño orientado a objetos. RUP es un proceso probado y de mucha efectividad que no detallare aquí.
2. **Elementos de modelado** que son los módulos conceptuales básicos;
   En este trabajo se modelará haciendo uso de los elementos que proporciona RUP.
   Se generarán modelos de componentes, dominio, diagramas de secuencia, clases.
3. **Una notación** cuyo objetivo es asegurar la coherencia visual de los elementos de modelado;
   Se usará UML que es parte de RUP.
4. **Experiencia**,
   Haciendo uso de la experiencia actual, se orientará el desarrollo para obtener un mejor acercamiento al resultado esperado eligiendo la mejor opción.

Proceso de desarrollo

Este es el proceso de desarrollo a seguir para el presente trabajo.

Fase	Actividad	Objetivo	Entradas	Producto
Análisis	Levantamiento de requisitos	▪ Saber que debe hacer el sistema ▪ Comprender los términos utilizados.	▪ Objetivos del presente trabajo	▪ Listado de requisitos funcionales y no funcionales. ▪ Modelo de dominio, con diccionario de datos
	Generación de casos de uso	▪ Encontrar las reglas del sistema, entradas, salidas, validaciones. ▪ Conocen las operaciones que debe cumplir el sistema.	▪ Listado de requisitos	▪ Listado de casos de uso numerados, con actores, pasos detallados, casos exitoso y no exitosos, por cada requerimiento. ▪ Diagrama de casos de uso
	Definición del alcance del sistema	▪ Declarar que va y que no va a hacer el sistema.	▪ Listado de requisitos, ▪ Listado de casos de uso	▪ Texto que enuncia el Alcance del sistema
Diseño	Elaborar el Modelo de componentes	▪ Se proponen los componentes del sistema para tener una visión general de la solución. ▪ Se ubican componentes del sistema en módulos, no todos pueden estar contenidos en un solo lugar.	▪ Listado de requisitos funcionales y no funcionales ▪ Listado de casos de uso	▪ Diagrama de componentes del sistema
	Definir elementos a diseñar por componente	▪ Dependiendo del tipo de componente, definir ¿Qué? y ¿Cómo? se diseña	▪ Listado de requisitos funcionales y no funcionales ▪ Listado de casos de uso	▪ Matriz de seguimiento
	Diseño de clases	▪ Se diseñan los elementos estáticos del sistema	▪ Modelo de dominio ▪ Listado de casos de uso	▪ Diagrama de clases
	Diagrama de secuencia	▪ Diseñar el comportamiento dinámico del	▪ Listado de casos de uso	▪ Diagrama de secuencia por cada operación de cada

		sistema		requerimiento
	Diseño de la base de datos	▪ Diseñar el repositorio de datos del sistema	▪ Diagrama de Clases	▪ Diseño físico de la base de datos
	Matriz de seguimiento para la fase de análisis	▪ Identificar por cada requerimiento los elementos diseñados para llevarlo a cabo.	▪ Diagrama de Clases ▪ Diagramas de secuencia	▪ Una matriz que relaciona los elementos diseñados por cada requerimiento
Implementación[1]	Construcción de cada componente diseñado	Finalmente es la construcción del elemento diseñado.	▪ Todos los elementos diseñados	▪ Es el código fuente del componente diseñado ▪ Archivos de configuración ▪ Procedimiento de instalación
	Actualización de la matriz de seguimiento para los componentes construidos	▪ Para llevar un control de cada elemento construido se registra en la matriz de seguimiento	▪ Todos los elementos construidos	▪ Matriz de seguimiento actualizada.
Pruebas	Elaborar matriz de pruebas	▪ Generar un guión con las pruebas de la funcionalidad del sistema	▪ Casos de uso, en especial con casos exitosos y no exitosos ▪ Sistema terminado	▪ Matriz de pruebas, listado con todas las pruebas por cada caso de uso.
	Aplicación de pruebas	▪ Probar que el sistema cumple todos los requerimientos enunciados	▪ Matriz de pruebas	▪ Matriz de pruebas ejecutada al 100% y sin observaciones

RUP se basa en el desarrollo iterativo e incremental. El proceso plantea que no todo lo que se necesita se obtiene en el primer ciclo de trabajo, sino que se obtiene en varias iteraciones tantas como sean necesarias. En las primeras se obtendrá una buena parte de lo requerido para empezar y con cada iteración se irá afinando o cambiando el producto elaborado hasta terminarlo.

Es importante hacer notar que los productos obtenidos de cada fase, se describen sin detallar los incrementos de información obtenidos en cada iteración que se realizó.

[1] RUP Nombra a la actividad de codificación del sistema como Implementación y no como Construcción. Esto por que en la fase de Diseño se modelan por completo y con detalle todos los aspectos del componente a crear. Por lo que la codificación se resume a "realizar" un diseño que funciona independientemente del lenguaje de programación.

Elementos de modelado

Se usarán los elementos de modelados propuestos por RUP, artefactos y diagramas de componentes, no se detallan estos elementos en este trabajo.

Notación

Se usará UML como lenguaje de modelado, no se detalla ni se dan una introducción de UML.

Contextualización y Levantamiento de requisitos

Modelo de dominio - Definición de algunos términos

Con el fin de hacer más precisos los requerimientos se definen algunos términos que usan mas adelante.

Punto: Cualquier lugar que se pueda identificar y ubicar.

Tipo de Punto: Es un calificador del tipo de instalación que representa el punto: Un edificio, biblioteca, comedor, sala, cancha, etc. En adelante instalación y punto se usaran indistintamente.

Tramo: Tramo de recorrido que enlaza a dos puntos

Costo: El valor en tiempo o distancia asociado a un tramo que debe recorrerse desde el punto inicial al punto final.

Punto origen: Punto desde donde se inicia un recorrido

Punto destino: Punto al que se desea llegar.

Ruta: La sucesión de puntos que deben tocarse para llegar del punto origen al punto destino.

Red: El conjunto total de puntos y tramos disponible para recorrerse. Para que haya una red debe haber un conjunto de puntos unidos por tramos.

Estas primeras definiciones constituyen el Modelo de Dominio inicial.

El Modelo de Dominio dentro de RUP es un *artefacto* que sirve para identificar los elementos conceptuales del sistema, son términos que están presentes en todo el desarrollo y son importantes por que ayudan a definir las clases del Modelo de Clases y las entidades del Modelo de Base de Datos. Es inicial pues, en adelante se irán descubriendo otros y puede ser que los que se han encontrado se reorganicen para estar contenidos unos dentro de otros, convertirse en reglas etc. Es decir no son todos ni definitivos.

El modelo de dominio es equivalente al Modelo Conceptual para Bases de datos y es equivalente a un subconjunto del Diccionario de Datos. El presente trabajo usa el proceso RUP y por tanto debemos usar los términos que propone este proceso.

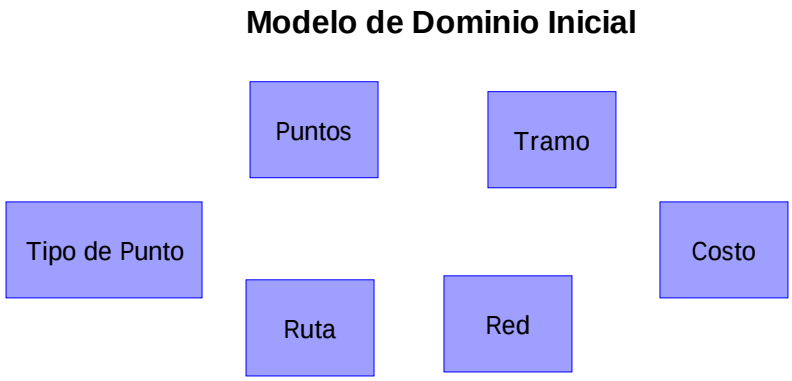


Fig. Modelo de Dominio Inicial

Requerimientos del sistema.

Los requerimientos del sistema pueden dividirse en dos grandes grupos: Los funcionales que aportan un valor significativo e indispensable para el usuario y los no funcionales que se refieren a apariencia, desempeño, usabilidad y comportamiento, estos son necesarios pero no indispensables para llevar a cabo una tarea.

Además de los requerimientos hay condiciones importantes que debe cumplir el desarrollo, estos se anotan en la sección de los requerimientos no funcionales con la nota "Condición indispensable"

Requerimientos funcionales

Requerimientos funcionales del sistema	
R1	Dados un punto de origen y un punto destino, el sistema debe encontrar y mostrar la ruta mas corta entre estos puntos. Para encontrar esta solución se usará el algoritmo de Dijkstra. La variable de análisis es la menor distancia de recorrido. Esta información deberá presentarse en texto y de forma gráfica. ▪ La información de texto que deberá mostrarse es la siguiente: • Un listado de los nombres de los puntos que hay que recorrer para llegar al destino • El tiempo total y distancia para hacer el recorrido ▪ La solución grafica mostrará un mapa completo de Ciudad Universitaria y sobre este se mostrará la solución con una línea uniendo cada uno de los puntos de la solución. Los puntos deben ubicarse con una marca y debe distinguirse los puntos de origen, destino de los intermedios ▪ Ambas soluciones deben poder imprimirse En caso de no haber solución mostrar el mensaje: "*No hay tramos de solución. Los tramos puede no estar conectados o tener datos incorrectos*." Esta situación puede darse cuando hay puntos pero no hay tramos que los unan.

R2	La selección de los puntos origen y destino podrá hacerse mediante una búsqueda previa por tipo de instalación, nombre de instalación.
R3	El sistema podrá registrar varias redes de puntos, por ejemplo red de puntos que solo puede recorrer un peatón, red de puntos que solo puede recorrer un automóvil. Un punto puede pertenecer a una o mas redes.
R4	Se tendrá la posibilidad de agregar, modificar y eliminar puntos, tramos de recorrido y tipos de redes mediante un módulo o programa de administración • Se darán de alta, baja y modificarán de manera gráfica los puntos de una Red. • Se darán de alta, baja y modificarán los valores de catálogos de Red, y Tipos de Punto. Solamente un usuario administrador tendrá la facultad para hacer estas tareas. Los datos de un punto son: Nombre, Tipo de instalación y para su ubicación en un croquis: un par de coordenadas. Dos puntos definen un tramo de recorrido, el sistema solo analizará tramos de recorrido previamente definidos. Los datos de un tramo son: el punto origen, el punto destino, tiempo de recorrido, distancia, estos dos últimos valores dependen del tipo de red, (Por ejemplo, no se recorre en el mismo tiempo un tramo en automóvil que uno a pie). De aquí se ve que un tramo pertenece a un tipo de red
R5	La solución que proporcione el sistema considerará que es diferente de ir de un punto A al B y de ir del B al A. Esto es importante por que algunos tramos de la ruta solamente pueden recorrerse en un solo sentido como el caso de los automóviles que solo pueden recorrer algunas calles en un solo sentido.
R6	Para hacer mas fácil el mantenimiento de puntos y tramos, estos deberán poderse dar de alta gráficamente y de una forma muy ágil ya que la captura requiere de mucho esfuerzo inicial.
R7	También es necesario poder consultar el avance en el alta de puntos de cada red, por lo que es necesario poder ver de manera gráfica los puntos existentes por red en el sistema de mantenimiento

Requerimientos no funcionales

Requerimientos No funcionales del sistema	
R10	**Condición indispensable**: La solución de la ruta debe obtenerse con el algoritmo de Dijkstra.
R11	**Condición indispensable**: El sistema de consulta debe ser un sistema Web para que pueda ser accedido desde cualquier punto del Internet.
R12	La solución gráfica mostrará todo el mapa de Ciudad Universitaria para tener una referencia completa. El mapa deberá contener un detalle aceptable para poder ubicar los edificios en el gráfico.
R13	El diseño del sistema debe poder permitir hacer flexible el sistema, para agregar más funcionalidad e información.
R14	La pantalla de consulta tendrá instrucciones simples y claras para el usuario de cómo usar el sistema

Análisis de requerimientos

Para poder continuar y antes de proponer una solución, veamos que entradas necesita el Algoritmo de Dijkstra para generar una ruta de solución.

Introducción a las Redes o Grafos

¿Qué es un grafo?

Si disponemos de una serie de puntos y un conjunto de tramos que los unen, de cada punto saldrán varios caminos, por lo que para ir de un punto a otro se podrán tomar diversas rutas. Cada camino tendrá un coste asociado (el tiempo o distancia de recorrido). Gracias a la representación por redes podremos elegir el camino más corto que conecta dos puntos.

Para representar a los puntos y tramos y referirnos a ellos se utilizan nodos y arcos de conexión.

Nodo (Punto): Se denomina nodo a un edificio, instalación o entronque, gráficamente se representa por un punto.

Arco (Tramo): Es una línea que une un nodo con otro. Todos los arcos tienen un costo asociado.

Costo: El costo es el tiempo, distancia de recorrido o cualquier otro valor de interés que se asociará a cada arco.

Red: Es un conjunto de Nodos y Arcos que la forman.

Gráficamente un Red se expresa como un par de conjuntos (N,A) donde N es un conjunto no vacío de elementos llamados nodos y A es un conjunto formado por pares no ordenados de elementos de N, llamados arcos. Se denota por G = (N,A).

Esta definición da lugar a una representación gráfica, en donde cada nodo es un punto del plano, y cada arco es una línea que une a sus dos nodos.

Notación Matemática de una Red:

G = {1,2,3,4,5,6,7;1-2,1-3,1-4,2-5,4-6,3-4,3-6,4-7,5-7}

Notación Gráfica de una Red:

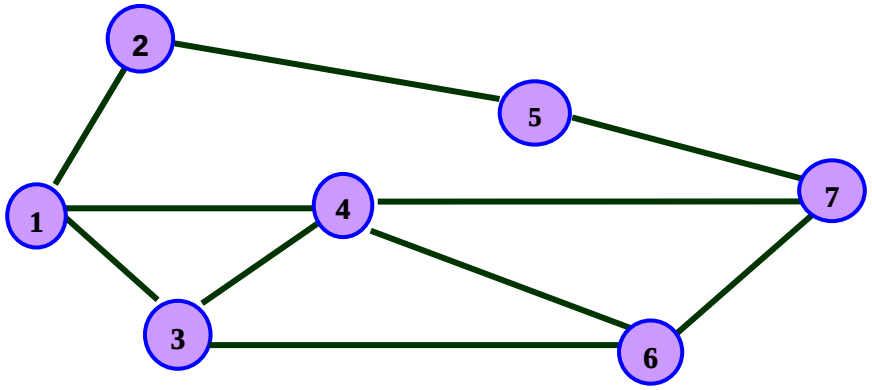


Fig. Notación Gráfica de una Red

¿Qué es un grafo dirigido y cargado?

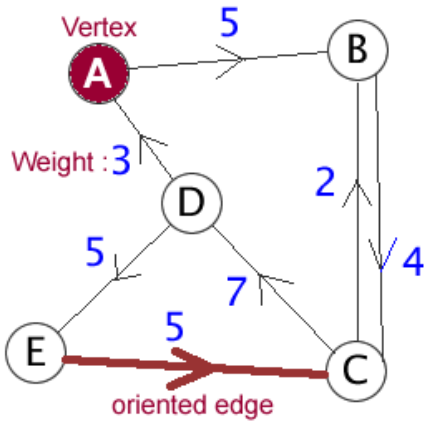


Fig. Ejemplo de un Grafo dirigido y cargado

En un grafo dirigido cada arco tiene una dirección representada por una flecha.

En un grafo cargado, en cada arco se asigna un peso. El peso puede entenderse como la distancia entre los 2 vértices.

Las estructuras que pueden ser representadas como grafos son ubicuos y muchos problemas de interés práctico pueden ser formulados como preguntas a cerca de ciertos gráficos, uno de los mejores ejemplos es la red de trenes.

- Cada estación del tren es un vértice.
- Los rieles entre 2 estaciones son un arco
- La distancia entre 2 estaciones del tren son equivalentes al peso.

Los problemas típicos que pueden ser resueltos con grafos son:

- El problema de la ruta mas corta
- El problema de inspección
- El problema de viaje del vendedor.

El algoritmo de Dijkstra se usa para encontrar la ruta mas corta desde un solo vértice a todos los demás en de un grafo dirigido y cargado. Todos los pesos no deben ser negativos.

El Algoritmo de Dijkstra

Dada una Red a cuyos arcos se han asociado una serie de pesos o costos, se define el camino de costo mínimo de un nodo *u* a otro *v*, como el camino donde la suma de los pesos de los arcos que lo forman es la mas baja entre todos los caminos posibles de *u* a *v*.

El algoritmo va construyendo sucesivamente los caminos de costo mínimo desde un nodo inicial hasta cada uno de los nodos de la Red y se utilizan los caminos conseguidos como parte de los nuevos caminos.

El algoritmo de Dijkstra necesita de los siguientes elementos:

R = R es una Red, **R=(N,A)** donde N es el conjunto de nodos y A es el conjunto de arcos.

S = Es el conjunto de nodos cuyos caminos más cortos desde el origen ya han sido determinados

N-S = Es el resto de los demás nodos

d(i) = Es un arreglo de las mejores estimaciones de los caminos más cortos a cada nodo. Es la distancia o costo acumulativo desde el origen hacia el nodo i

pi(i) = Es un arreglo de predecesores para cada nodo i

c(i,j) = Costo del arco, el costo de ir desde el nodo i hacia el nodo j

El modo básico de funcionamiento es:

```
1. Inicializar d(i) y pi(i)
     a. Para todo i de 1 al # de nodos
           i.   d(i)= INFINITO   /* un numero muy grande */
          ii.   p(i)=0
     b. Identificar el Nodo origen:
           i.   d(1)=0
          ii.   p(1)=1
2. Poner el conjunto S como VACIO
     /* N-S es una tabla auxiliar con la información de todos los
      nodos que existen en G*/
     a. N-S = Red. Vértices
     /* S es de la misma estructura que N-S, otra tabla auxiliar
     vacía */
     b. S = VACIO
3. Mientras existan nodos en la tabla N-S (i.e. existen nodos sin
   determinar su camino mínimo del origen)
     a. Ordenar los nodos en N-S y analizar de acuerdo a sus menores
        distancia desde el origen
     b. Añadir u, el nodo mas cercano en N-S a S
           i.   S = S+{u}, agregar el nodo u a la tabla S
     c. Relajar todos los nodos que todavía están en N-S adyacentes a u
```

```
                i.  Recalcular la distancia a todos los nodos adyacentes a u
                    que aún están en N-S
```

El Proceso de Relajación, actualiza los costos de todos los nodos, **v**, conectados a un nodo **u**, si podemos mejorar la mejor estimación del camino mas corto hacia v, incluyendo **(u,v)** en la ruta hacia **v**

Elementos para la generación de la ruta

De lo anterior puede verse que para generar una ruta es necesario contar con un **inventario de puntos y sus tramos de conexión** con otros puntos que será dado de alta en el sistema a través de pantallas de captura.

El sistema generará solo una ruta a seguir, que será siempre la de menor costo. **La variable de análisis que se usará es la menor distancia de recorrido.**

Los elementos mínimos necesarios para poder generar una ruta son los siguientes.

- Dos puntos, uno de origen y destino.
- Los tramos de recorrido, que se forman por la interconexión entre todos los puntos que se analizan para generar la ruta.

Premisas

- Los puntos dados de alta en el sistema se limitaran a Ciudad Universitaria y puntos perimetrales.
- El sistema debe ser un sistema Web

Datos del modelo de dominio

Datos de un punto

Los datos de un punto son:

- Número de instalación, que permitirá ubicarlo identificarlo de otras instalaciones.
- Nombre de instalación, Por ejemplo “Facultad de Ingeniería”, “3er Paradero”, “Comedor de Ingeniería”, etc.
- Tipo de instalación, “Facultad”, “Parada de camión”, “Estacionamiento”, etc.

Datos de apoyo o adicionales

- Para poder generar una solución gráfica es necesario que el punto esté referenciado a un mapa de CU, con un par coordenado X,Y.

Datos de un tramo

Los datos de un tramo de conexión entre 2 puntos son:

- Punto inicial, punto inicial del tramo.

- Punto final, el punto final del tramo.
- Nombre del tramo
- Costo o variable de análisis - Tiempo de recorrido
- Costo o variable de análisis - Distancia de recorrido
- Tipo de Red al que pertenece el tramo. Es importante clasificar tramos que solo pueden ser recorridos por automóviles o peatones

Datos de la Red

Los datos de la red son:

- Nombre de la red
- Velocidad promedio de la red

Por ejemplo: Red de puntos por los que puede pasar un Automóvil, un peatón, un ciclista, etc.

Datos del tipo de punto o instalación

Los datos del tramo

- Nombre del tipo de instalación, por ejemplo: Un comedor, una biblioteca, en sí cualquier grupo de puntos que pueda ser clasificado.

Modelo de Dominio Final

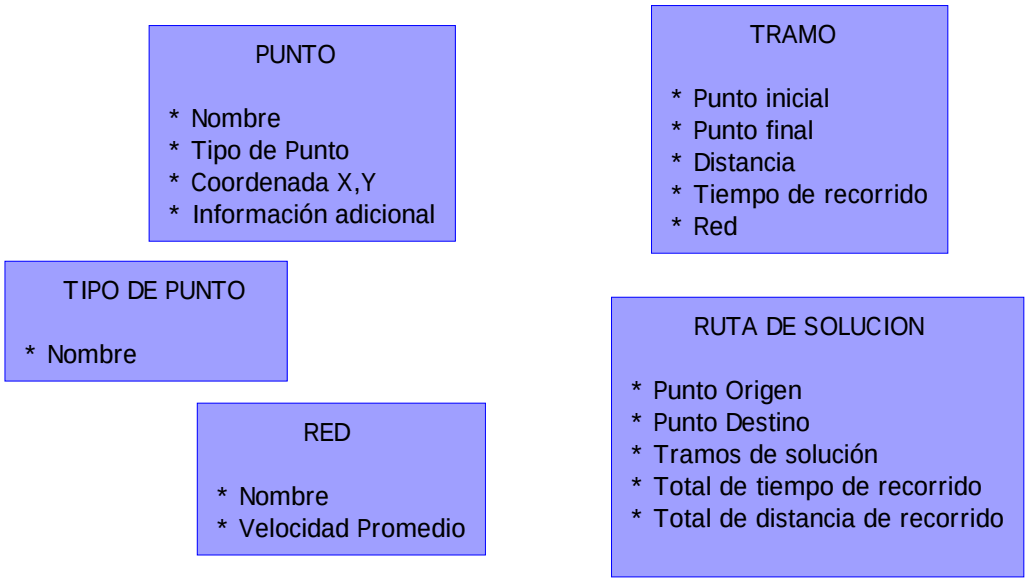


Fig. Modelo de Dominio Final

Operaciones identificadas

1. Solución de la ruta,

La operación más importante es la de encontrar la ruta de solución:

Dada una red, un punto origen y un punto destino al aplicar el método de búsqueda se obtiene la ruta de solución:

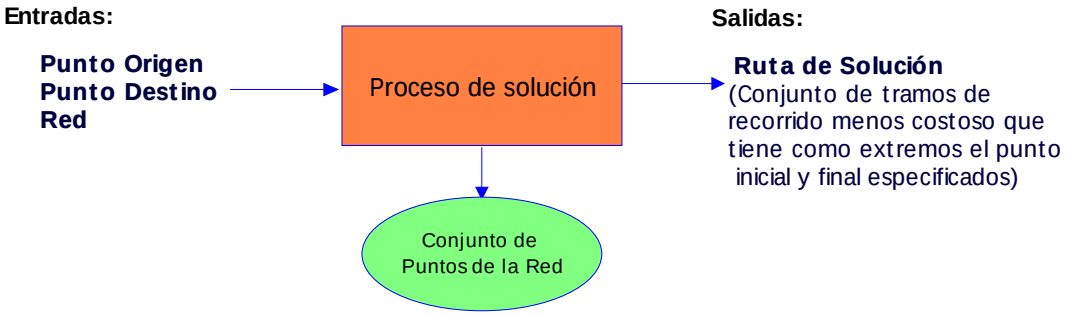


Fig. Proceso de Solución de Obtención de la Ruta

Las demás operaciones son de apoyo,

2. Mantenimiento a la red puntos, registro de nuevos puntos, redes, tramos.

3. Consulta de datos de entrada

4. Formateo de la salida de datos de la ruta generada

Propuesta de la solución

Elementos a desarrollar para generar la solución de la ruta en el sistema.

1. Proceso de solución de la ruta

- El proceso encontrará la solución aplicando el algoritmo de Dijkstra.
- El proceso de solución recibirá como datos de entrada: Un punto origen, uno destino y la red de consulta. El proceso de solución buscará los puntos disponibles en el conjunto de puntos de la red especificada, encontrará la mejor ruta en base al menor costo y devolverá la información de la ruta de solución.

2. Base de datos de puntos y tramos

- Se deberá construir una base de datos de tramos y puntos donde se almacene esta información
- Tablas para generar la ruta: Tabla de Puntos, Tramos, Redes, Tipos de Instalación.
- Tablas con información para mostrar gráficamente la ruta.

- El mantenimiento a esta base de datos deberá hacerse desde un sistema de administración restringido a un usuario y distinto al sistema de consulta para evitar la posibilidad de modificar la información.

3. Pantallas de mantenimiento

- Se crearán pantalla de alta y modificación de puntos y tramos, en esta pantalla se darán de alta los puntos, sus atributos y se darán de alta también los tramos que forman al tener conexión con otros puntos.
- Para hacer mas fácil el mantenimiento de puntos y tramos, estos deberán poderse dar de alta gráficamente y de una forma muy ágil, ya que la captura de esta requiere de mucho esfuerzo inicial.
  Se propone en una misma pantalla poder dar de alta los puntos y los tramos.
- Para tener una forma de verificar el avance del inventario de puntos, la totalidad de puntos debe poderse consultar gráficamente desde el sistema de mantenimiento. Se propone poder visualizar por cada punto sus tramos inmediatos
- El sistema de mantenimiento no debe ser accesible por un usuario diferente al administrador. Se propone separar el sistema del sistema de consulta y acceder a el solo con una clave de usuario
- Para poder mostrar la solución grafica es necesario contar con un mapa de ciudad Universitaria completo.
  - Debe mostrar a detalle los edificios de las instalaciones de Ciudad Universitaria.
  - Debe obtenerse de alguna fuente que otorgue permisos para su uso no comercial.
  - Este gráfico debe ser el mismo tanto en el sistema de consulta como el de administración. A un punto debe asignarse una coordenada para ubicarlo en el mapa y debe ser la misma coordenada debe poder ubicarse en los 2 módulos de consulta y mantenimiento.

4. Pantallas de consulta

- Pantalla inicial de generación de ruta con elección del tipo de red
- Pantalla de resultado de solución de la ruta.
- Gráfico de la ruta
- Pantalla de impresión de la ruta.

5. Carga inicial de datos para pruebas

Se debe alimentar la base de datos con datos validos para poder hacer pruebas. Estos datos serán los mínimos suficientes para hacer pruebas.

Propuestas para realizar pruebas

- Se darán de alta con el fin de mostrar el funcionamiento del sistema 2 tipos de redes de puntos: Red de Peatón y Red de Automóvil.
- El sistema permitirá registrar un número ilimitado de puntos pero para el presente trabajo solo se registraran un conjunto de puntos significativos de prueba para cada red. Así mismo para que pueda identificarse en el sistema la diferencia de usar una u otra ruta se registraran puntos suficientes para generar 2 opciones de recorrido que demuestren el funcionamiento del sistema.

  Se hará un levantamiento de datos de puntos y tiempos de recorrido de la siguiente ruta: De el Edificio Principal de Ingeniería al edificio Anexo de Ingeniería, en un sentido y en otro.

- La tarea de registrar para los puntos de prueba los datos distancias y tiempo de recorrido se limitará a lo siguiente:
  - Para la red de Peatón, el promedio de 2 mediciones de la ruta de recorrido.
  - Para la red de Automóvil, el promedio de 2 mediciones de la ruta de recorrido.

  La velocidad de recorrido que se asignará inicialmente a cada red es el valor obtenido por esta toma de valores.

¿Qué no va a hacer el sistema?

- Las velocidades de recorrido son variables para una misma red puede ser en diferente hora por las variaciones del tipo de terreno, tráfico de personas o vehículos, hora del día, etc. Esta capacidad de generar una ruta con diferencias de tiempo recorrido dependiendo de condiciones variables para un mismo tramo queda fuera del alcance del sistema.
- La variable de análisis o costo será la menor distancia. Para una misma red se considera la velocidad de recorrido constante.

  Otras variables de análisis como el tiempo de recorrido, importancia turística, importancia *de* seguridad no serán variable de decisión pero se podrán agregar haciendo el levantamiento de información para cada punto.
- El sistema solo generará una ruta por cada para de puntos origen y destino. Con el Algoritmo de Dijkstra es posible generar todas las rutas de solución de un punto a otro, pero para el alcance del sistema solo mostrará la solución de la ruta con el recorrido mas corto.

Requerimientos Funcionales como Casos de uso

La información anterior es suficiente para enunciar los casos de uso del sistema.

La redacción de estos casos asume que el sistema va a ser de tipo Web, por lo que se usan términos como botón, pagina, lista desplegable, etc. Con esto el análisis cubre el requerimiento R11.

El requerimiento de hacer que el diseño del sistema sea flexible el sistema, para poder agregar más funcionalidad e información se cubre diseñando el sistema orientado a objetos y programando por capas, esto se detalla mas adelante en la fase de diseño.

Caso de uso 1. Consulta de la Ruta

	Caso de uso 1	**Consulta de la ruta**
	Requerimientos cubiertos	**R1, R2, R5, R10, R11, R12, R14**
	Descripción	El caso de uso tiene como objetivo generar la ruta de solución, se proporcionan los puntos origen, destino y la red y el sistema calcula la ruta. El sistema primero debe presentar una lista de puntos disponibles para hacer la búsqueda. Inicialmente no puede mostrar todos los puntos disponibles por que la red de puntos es muy grande, por eso antes de inicial el proceso de solución deben buscarse los puntos.
	Actores	**Usuario, Sistema**

	Precondiciones	Datos de puntos y tramos registrados para proporcionar la solución
	Pasos	1. El Usuario accede a la pantalla inicial de selección de red, punto origen y destino. 2. El Sistema presenta una lista de redes disponibles, un campo de búsqueda por nombre y tipo de punto tanto para el punto origen como para el punto destino 3. El Usuario elige la red, y los criterios de búsqueda de los puntos que desea mostrar y elige Continuar 4. El Sistema muestra una lista de puntos disponibles para el punto origen y una lista de puntos disponibles para el punto destino, para los criterios de búsqueda y red elegida. Los datos a mostrar por cada punto son: ▪ Tipo de punto, ▪ Nombre del punto Los datos estarán ordenados por nombre del punto 5. El Usuario elige el punto origen y el punto destino e indica al sistema que inicie la búsqueda. 6. El Sistema genera la ruta de solución en base al punto origen, el punto destino y la red elegida. La información de salida es la siguiente: 6.1 Un croquis de C.U. que muestra los puntos y tramos de la ruta generada, indicando con colores los puntos origen y destino. Si el croquis no cabe en la pantalla, debe poder recorrerse con barras de desplazamiento. 6.2 Un listado con los tramos a recorrer para llegar desde el punto origen al destino Por cada tramo se debe presentar la siguiente información: Nombre del punto inicial del tramo Nombre del punto final del tramo Distancia del tramo en metros Tiempo de recorrido del tramo en segundos 6.3 Un total del tiempo y distancia de recorrido al final del listado. Como adicional debe permitirse la impresión de la información. Debe tenerse una opción para volver a hacer una consulta.
	Validaciones	En 4.- El Sistema valida que se haya elegido algún criterio de búsqueda de puntos, en caso de que el Usuario no haya elegido alguno, el sistema le avisa así: "Debe elegir al menos un criterio de búsqueda para los puntos origen y destino" En 5.- El sistema valida que se haya elegido un punto origen y un punto destino para poder continuar. Si no eligió un punto origen el sistema muestra el mensaje "Debe elegir un punto origen" Si no eligió un punto destino el sistema muestra el mensaje "Debe elegir un punto destino"
	Caminos alternos	En 6. Si no tramos que unan los puntos origen y destino se deberá mostrar el mensaje: "No hay datos suficientes para generar la solución"
	Poscondiciones	----------

	Comentarios adicionales	El sistema debe proporcionar la solución de una ruta diferenciando el sentido del recorrido, la solución de un punto **A** a un **B** puede ser diferente de B a A. Por ejemplo los automóviles solo pueden puede recorrer ciertos tramos en un solo sentido. En 6. El algoritmo usado para obtener la solución es el de Dijkstra. En la pantalla se visualizarán instrucciones de cómo usar el sistema

Caso de uso 2. Mantener el Catálogo de Redes

Las operaciones Creación, Lectura, Actualización y Borrado (Operaciones CRUD por sus siglas en inglés) de datos de catálogos no se describen a detalle en los Casos de Uso. El objetivo de los Casos de Uso es detallar para entender como debe funcionar algún proceso. El mantenimiento de catálogos es una tarea conocida y común, por lo que no se detallan estas operaciones para los catálogos de Red y Tipo de Punto

	Caso de uso 2	**Mantener el Catálogo de Redes**
	Requerimientos cubiertos	**R3, R4**
	Descripción	Registrar, Actualizar, Borrar una Red
	Actores	**Administrador, Sistema**
	Precondiciones	El administrador esta registrado en el sistema
	Pasos	1. El Administrador accede a la pantalla de mantenimiento de Redes 2. El Sistema muestra el listado actual de todas las redes registradas en el sistema Los datos de que se muestran en pantalla son: • **Clave de la Red** • **Nombre de la Red** • **Velocidad promedio de recorrido** Las opciones desde esta pantalla son Alta, Eliminar, Modificar Cancelar 3. El Administrador elige alguna opción de modificación de catálogo 3.1 Si el Administrador elige Alta 3.1.1 El Sistema muestra una forma de captura con los siguientes datos • **Nombre de la Red** • **Velocidad promedio de recorrido** La clave del registro la asigna internamente el sistema 3.1.2 El Usuario captura los datos del catálogo y elige la opción de guardar 3.1.3 El Sistema valida los datos capturados, da de alta el nuevo registro y confirma la operación al Administrador 3.2 Si el Administrador elige Eliminar 3.2.1 El Sistema pide la confirmación del usuario. 3.2.2 El Administrador confirma el borrado del registro 3.2.3 El Sistema elimina el registro si no tiene información dependiente y confirma la operación. Si el registro tiene información

		dependiente, el sistema despliega un mensaje de error al usuario "El registro tiene información asociada, debe primero eliminar la información dependiente" 3.3 Si el Administrador elige Modificar 3.3.1 El Sistema muestra una forma de captura con los datos del registro elegido 3.3.2 El Administrador modifica la información del registro y da clic en guardar 3.3.3 El Sistema valida la información capturada y guarda los cambios. A continuación confirma la operación al Administrador.
	Validaciones	Para tipos de dato cadena se permiten todos los caracteres Para los datos de tipo numérico se validan que los datos sean caracteres numéricos
	Caminos alternos	----------
	Poscondiciones	----------
	Comentarios adicionales	----------

Caso de uso 3. Mantener el Catálogo de Tipos de Punto

	Caso de uso 3	**Mantener el Catálogo de Tipos de Punto**
	Requerimientos cubiertos	**R3, R4, R6**
	Descripción	Registrar, Actualizar, Borrar un Tipo de Punto o Tipo Instalación
	Actores	**Administrador, Sistema**
	Precondiciones	El administrador esta registrado en el sistema
	Pasos	1. El Administrador se accede a la pantalla de mantenimiento de Tipos de Punto 2. El Sistema muestra el listado actual de todas las redes registradas en el sistema Los datos de que se muestran en pantalla • **Clave del Tipo de Punto** • **Nombre del Tipo de Punto** Las opciones desde esta pantalla son Alta, Eliminar, Modificar, Cancelar 3. El Administrador elige alguna opción de modificación de catálogo 3.1 Si el Administrador elige Alta 3.1.1 El Sistema muestra una forma de captura con los siguientes datos • **Nombre del Tipo de Punto** La clave del registro la asigna internamente el sistema 3.1.2 El Usuario captura los datos del catálogo y elige la opción de guardar 3.1.3 El Sistema valida los datos capturados, da de alta el nuevo registro y confirma la operación al Administrador 3.2 Si el Administrador elige Eliminar 3.2.1 El Sistema pide la confirmación del usuario.

		3.2.2 El Administrador confirma el borrado del registro 3.2.3 El Sistema elimina el registro si no tiene información dependiente y confirma la operación. Si el registro tiene información dependiente, el sistema despliega un mensaje de error al usuario "El registro tiene información asociada, debe primero eliminar la información dependiente" 3.3 Si el Administrador elige Modificar 3.3.1 El Sistema muestra una forma de captura con los datos del registro elegido 3.3.2 El Administrador modifica la información del registro y da clic en guardar 3.3.3 El Sistema valida la información capturada y guarda los cambios. A continuación confirma la operación al Administrador.
	Validaciones	Para tipos de dato cadena se permiten todos los caracteres Para los datos de tipo numérico se validan que los datos sean caracteres numéricos
	Caminos alternos	----------
	Poscondiciones	----------
	Comentarios adicionales	----------

Caso de uso 4. Mantener el Catálogo de Puntos de forma gráfica

Los catálogos de Puntos y tramos podrían darse de alta de la misma forma que los anteriores, pero el mantenimiento seria muy complicado si se tiene que ubicar los puntos en un gráfico de manera manual, por esta razón el mantenimiento de estos catálogos se hará de manera gráfica.

	Caso de uso 4	**Mantener el Catálogo de Puntos de forma gráfica**
	Requerimientos cubiertos	**R3, R4, R6, R7**
	Descripción	Registrar, Actualizar, Borrar un Punto
	Actores	**Administrador, Sistema**
	Precondiciones	El administrador esta registrado en el sistema
	Pasos	1. El Administrador se accede a la pantalla gráfica de mantenimiento de Puntos 2. El Sistema muestra dos secciones: una donde aparece el grafico de CU y otra con los campos para la captura o modificación de un punto. Los datos del punto que se muestran en pantalla • **Clave del Punto** • **Nombre del Punto** • **Tipo de Punto** • **Coordenada X en gráfico** • **Coordenada Y en gráfico** • **Comentarios**

		Las opciones desde esta pantalla son Alta, Eliminación, Modificación de los datos del punto y la opción Cancelar 3. El Administrador selecciona la red de puntos sobre la que desea trabajar. 4. El Sistema recuerda la red elegida y solo mostrará o dará de alta puntos esta red. 5. El Administrador elige alguna opción de modificación de catálogo 5.1 Si el administrador da clic en Consultar Red 5.1.1 Se despliegan todos los puntos en el gráfico de esa red 5.2 Si el usuario da clic en un punto existente 5.2.1 El sistema muestra la información del punto en los campos de captura del punto 5.3 Si el Administrador da clic sobre el croquis donde no hay un punto y elige Alta 5.3.1 El Sistema muestra una forma de captura con los siguientes datos ▪ **Nombre del Punto** ▪ **Tipo de Punto** ▪ **Coordenada X del gráfico donde se dio clic** ▪ **Coordenada Y del gráfico donde se dio clic** ▪ **Comentarios** La clave del registro la asigna internamente el sistema 5.3.2 El Usuario captura los datos del punto y elige la opción de guardar 5.3.3 El Sistema valida los datos capturados, da de alta el nuevo registro y confirma la operación al Administrador 5.4 Si el Administrador da clic a un punto existente sobre el croquis y elige Eliminar 5.4.1 El Sistema pide la confirmación del usuario. 5.4.2 El Administrador confirma el borrado del punto 5.4.3 El Sistema elimina el punto si no tiene información dependiente y confirma la operación. Si el registro tiene información dependiente, el sistema despliega un mensaje de error al usuario "El registro tiene información asociada, debe primero eliminar la información dependiente" 5.5 Si el Administrador da clic sobre algún punto y elige Modificar 5.5.1 El Sistema muestra una pantalla de captura con los datos del punto elegido 5.5.2 El Administrador modifica la información del punto y da clic en guardar 5.5.3 El Sistema valida la información capturada y guarda los cambios. A continuación confirma la operación al Administrador.
	Validaciones	Para tipos de dato cadena se permiten todos los caracteres. Para los datos de tipo numérico se validan que los datos sean caracteres numéricos.
	Caminos alternos	----------
	Poscondiciones	----------
	Comentarios adicionales	----------

Caso de uso 5. Mantener el Catálogo de Tramos de forma gráfica

	Caso de uso 5	Mantener el Catálogo de Tramos de manera gráfica
	Requerimientos cubiertos	**R3, R4, R6, R7**
	Descripción	Registrar, Actualizar, Borrar un Tramo. Esta tarea se hará sobre la misma pantalla gráfica de mantenimiento de puntos para hace mas fácil esta operación.
	Actores	**Administrador, Sistema**
	Precondiciones	El administrador esta registrado en el sistema. Para poder realizar alguna operación sobre algún tramo es necesario haber elegido al menos un punto
	Pasos	1. El Administrador se accede a la pantalla gráfica de mantenimiento de Puntos 2. El Sistema muestra dos secciones: una donde aparece el grafico de CU y otra con los campos para la captura o modificación de un punto. El Sistema muestra una tercera sección con los datos de los tramos asociados al punto. Si no se ha elegido algún punto entonces no deben aparecer los tramos. Los datos para el tramo que se muestran en pantalla son: • **Clave del Tramo** • **Clave de la Red** • **Punto Inicial del tramo** • **Punto final del tramo** • **Distancia en metros** • **Tiempo de recorrido en segundos** Las opciones desde esta pantalla son Alta, Eliminar, Modificar un Tramo 3. Operaciones de modificación de un tramo 3.1 Si el Administrador requiere hacer un Alta de tramo 3.1.1 El Administrador elige consultar puntos para que se muestren todos los puntos de la red 3.1.2 El Sistema muestra gráficamente los puntos de la red elegida 3.1.3 El Administrador da clic en dos puntos consecutivos que no tengan tramo 3.1.4 El Sistema recuerda los puntos elegidos 3.1.5 El administrador da clic a la opción Alta de Tramo 3.1.6 El Sistema muestra una pantalla de captura con los siguientes datos ▪ **Clave del Tramo** ▪ **Clave de la Red elegida,** (No debe poder modificarse) ▪ **Punto Inicial del tramo que se dio el primer clic** ▪ **Punto final del tramo en que se dio el segundo clic** ▪ **Distancia en metros** ▪ **Tiempo de recorrido en segundos** ▪ **Un checkbox para preguntar si el tramo debe de darse de alta en ambos sentidos** La clave del tramo la asigna internamente el sistema 3.1.7 El Usuario captura los datos del Tramo y elige la opción de guardar

		3.1.8 El Sistema valida los datos capturados, da de alta el nuevo tramo y confirma la operación al Administrador y regresa a la pantalla anterior 3.2 Si el Administrador requiere Eliminar un tramo 3.2.1 El Administrador elige sobre el croquis el tramo a eliminar y da clic a la opción “Eliminar tramo” 3.2.2 El Sistema pide la confirmación del usuario. 3.2.3 El Administrador confirma el la eliminación del tramo 3.2.4 El Sistema elimina el tramo si no tiene información dependiente y confirma la operación. Si el registro tiene información dependiente, el sistema despliega un mensaje de error al usuario “El tramo tiene información asociada, debe primero eliminar la información dependiente” 3.3 Si el Administrador elige Modificar 3.3.1 El Administrador elige sobre el croquis un tramo y da clic en “Modificar Tramo” 3.3.2 El Sistema muestra una pantalla de captura con los datos del tramo elegido 3.3.3 El Administrador modifica la información del tramo y da clic en guardar 3.3.4 El Sistema valida la información capturada y guarda los cambios. A continuación confirma la operación al Administrador. Y regresa a la pantalla anterior
	Validaciones	Para tipos de dato cadena se permiten todos los caracteres Para los datos de tipo numérico se validan que los datos sean caracteres numéricos
	Caminos alternos	----------
	Poscondiciones	----------
	Comentarios adicionales	----------

Diagrama de casos de uso

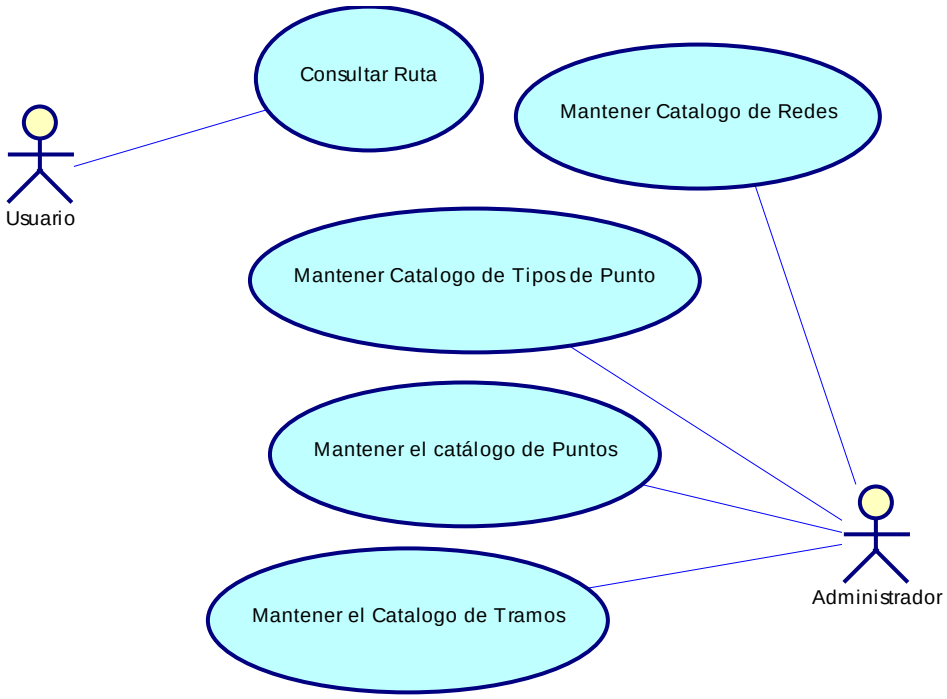


Fig. Casos de Uso del Sistema

Alcance del sistema

Los siguientes puntos se obtuvieron anteriormente, es importante siempre mencionarlos para saber que no va a hacer el sistema.

1. Los puntos dados de alta en el sistema se limitaran a Ciudad Universitaria y puntos perimetrales.
2. Se darán de alta con el fin de mostrar el funcionamiento del sistema 3 tipos de redes de puntos: Red de Peatón, Red de Automóvil y Red de transporte UNAM.
3. El sistema permitirá registrar un número ilimitado de puntos pero para el presente trabajo solo se registraran un conjunto de puntos significativos de prueba para cada red. Así mismo para que pueda identificarse en el sistema la diferencia de usar una u otra ruta se registraran puntos suficientes para generar 2 opciones de recorrido que demuestren el funcionamiento del sistema.

   La ruta de recorrido de muestra es del edificio Principal de Ingeniería al edificio Anexo de Ingeniería en un sentido y en otro.
4. Queda fuera del alcance del sistema la carga de puntos de toda una red, ya que es una tarea larga y que debe hacer un equipo especifico de personas.
   La tarea de registrar los datos de distancia y tiempo de prueba se limitara a lo siguiente:
   - Para la red de Peatón, el promedio de 2 mediciones de la ruta de recorrido.
   - Para la red de Automóvil, el promedio de 2 mediciones de la ruta de recorrido.
   - Para la red de Transporte UNAM, el promedio de 2 mediciones de la ruta de recorrido.

La velocidad de recorrido que se asignará inicialmente a cada red es el valor obtenido por esta toma de valores.

5. Las velocidades de recorrido son variables para una misma red puede ser en diferente hora por las variables del tipo de terreno, tráfico de personas o vehículos, hora del día, etc. Esta capacidad de generar una ruta con diferencias de tiempo recorrido dependiendo de condiciones variables para un mismo tramo queda fuera del alcance del sistema.
6. La variable de análisis o costo será la menor distancia. Por ello se considerará que para una misma red, el tiempo de recorrido es constante.
7. Ya que el objetivo del sistema es mostrar la aplicación del algoritmo de Dijkstra de generación de la ruta mas corta entre un punto y otro, el sistema solo generará una ruta para un mismo punto origen, destino y red, no generará rutas alternativas de recorrido para los mismos datos.

   Con el Algoritmo de Dijkstra es posible generar todas las rutas de solución de un punto a otro, pero para el alcance del sistema solo

CAPÍTULO 4. DISEÑO

El primer alcance del sistema es que pueda obtener la solución de encontrar la ruta mas corta. Sin embargo el sistema puede dar mas productos, para dar la solución a la ruta menos costosa en tiempo, combinación de modos de transporte, llegar a un punto especificando pasando obligatoriamente por ciertos puntos, comparar otras rutas, etc. Para incrementar las capacidades del sistema, debe preverse que habrá cambios para agregar más funcionalidad por ello es necesario hacer un buen diseño.

Se debe construir un sistema flexible y que pueda ampliarse. La metodología de orientación a objetos hace posible la modularización del sistema por la clara agrupación de responsabilidades en componentes independientes.

Un buen diseño con patrones de diseño permitirá que el sistema sea modular y que pueda crecer sin necesidad de hacer cambios que hagan que se recodifique el sistema. Este es un problema común cuando no se plantea un buen diseño de sistemas.

Componentes del sistema

En el capítulo anterior se vio que los componentes del sistema que deben considerarse son:

1. El Proceso o la lógica de solución de la ruta. Este debe estar dentro de un módulo de Consulta separado de la lógica de mantenimiento.

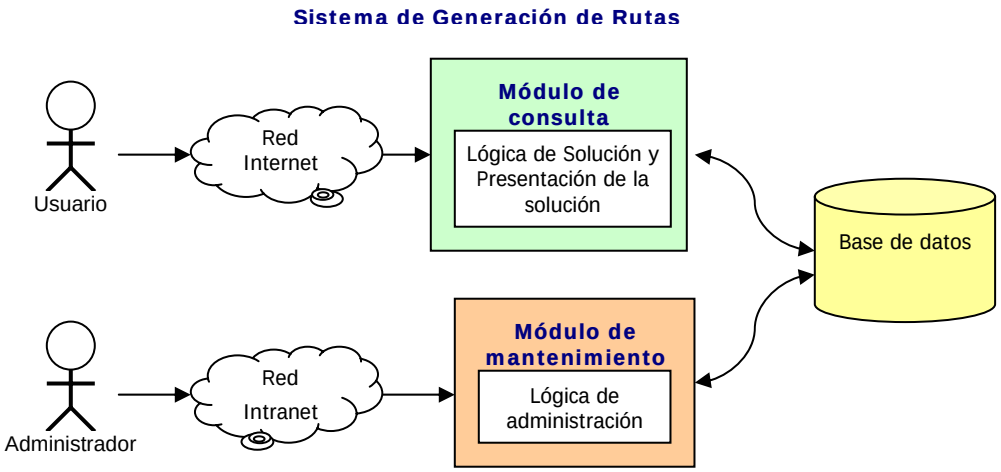
2. Pantallas de mantenimiento, La lógica de mantenimiento de catálogos debe estar contenida en un módulo de mantenimiento que no pueda ser accedido por un usuario normal.

3. Pantallas de consulta, La lógica de consulta y presentación de la solución deberá estar contenida en un módulo que sea accesible desde Internet

4. Base de datos de puntos y tramos. Como en todos los sistemas de información debe haber una base de datos donde almacenar la información a consultar.

Diagrama de componentes

El siguiente diagrama muestra estos componentes



Diseñemos estos componentes.

Diseño del componente 1. Lógica de Solución de la Ruta

Algoritmo de solución

Para dar solución al problema de la ruta debe modelarse una red de puntos y tramos como conjunto de solución y modelar también el proceso de solución que tome un punto origen y uno destino de este conjunto que estén unidos entre sí mediante la sucesión consecutiva de tramos.

Recordando el algoritmo:

El algoritmo de Dijkstra necesita de los siguientes elementos:

R = R es una Red, R=(N,A) donde N es el conjunto de nodos y A es el conjunto de arcos.
S = Es el conjunto de nodos cuyos caminos más cortos desde el origen ya han sido determinados
N-S = Es el resto de los demás nodos
d(i) = Es un arreglo de las mejores estimaciones de los caminos más cortos a cada nodo. Es la distancia o costo acumulativo desde el origen hacia el nodo i
pi(i) = Es un arreglo de predecesores para cada nodo i
c(i,j) = Costo del arco, el costo de ir desde el nodo i hacia el nodo j

El modo básico de funcionamiento es:

1. **Inicializar d(i) y pi(i)**
   - a. **Para todo i de 1 al # de nodos**
     - i. **d(i)= INFINITO** /* un numero muy grande */
     - ii. **p(i)=0**
   - b. **Identificar el Nodo origen:**
     - i. **d(1)=0**
     - ii. **p(1)=1**
2. **Poner el conjunto S como VACIO**
   - a. **N-S = Red. Vértices** /* N-S es una tabla auxiliar con la información de todos los nodos que existen en G*/
   - b. **S = VACIO** /* S es de la misma estructura que N-S, otra tabla auxiliar vacía */
3. **Mientras existan nodos en la tabla N-S** (i.e. existen nodos sin determinar su camino mínimo del origen)
   - a. **Ordenar los nodos en N-S y analizar de acuerdo a sus menores distancia desde el origen**
   - b. **Añadir u, el nodo mas cercano en N-S a S**
     - i. **S = S+{u}, agregar el nodo u a la tabla S**
   - c. **Relajar todos los nodos que todavía están en N-S adyacentes a u**
     - i. **Recalcular la distancia a todos los nodos adyacentes a u que aún están en N-S**

El Proceso de Relajación, actualiza los costos de todos los nodos, **v**, conectados a un nodo **u**, si podemos mejorar la mejor estimación del camino mas corto hacia v, incluyendo **(u,v)** en la ruta hacia **v**

El algoritmo de solución puede programarse desde cero, u optar por usar alguna de las implementaciones libres que hay disponibles en Internet sin fines comerciales.

El apoyarse sobre código ya construido hace más rápido el desarrollo y aprovechamos el beneficio de la reutilización que plantea la orientación a objetos.

He optado por basar la solución en una implementación básica en java que desarrollo Jean Michael Garnier[1], a continuación se describen las clases de esta implementación.

Implementación en java

Estas son las clases que definen las estructuras de datos necesarias para soportar el algoritmo:

- *shortestPathMap*, es el mapa resultado de la ruta de distancias mas cortas desde un vértice después de ejecutar el algoritmo.
- *predecessorsMap*, Es el mapa de precedencias de la ruta mas corta desde un vértice después de ejecutar el algoritmo.
- *determindedVerticesSet*, Es el conjunto de vértices cuyas rutas mas cortas desde el vértice origen ya se han determinado.
- *remainingVerticesQueue*, la lista de vértices remanentes esta implementada como una cola de prioridades.

Ilustración del algoritmo de solución:

```
runAlgorihtm(Object sourceVertex, Object destinationVertex) {

    // Initializacion, vacia todas las estructuras
    shortestPathMap.clear();
    predecessorsMap.clear();
    determinedVerticesSet.clear();
    remainingVerticesQueue.clear();

    // Agrega un vértice origen con una distancia de 0
    shortestPathMap.put(sourceVertex, new Integer(0));

    // Inserta un vértice origen con una distancia de 0
    remainingVerticesQueue.insert(sourceVertex, 0);

    // Mientras la cola de prioridad no este vacía
    while ( !remainingVerticesQueue.isEmpty() ) {
            // Ordena los vértices en los vértices remanentes de a cuerdo con
            // su distancia desde el origen y selecciona uno de los mas
            // cercanos, quita el elemento con la prioridad mas baja en la cola
        Object closest = remainingVerticesQueue.dequeueLowestPriorityElement();

        // si el destino se alcanza, se detiene la ejecución
        if ( closest.equals(destinationVertex) ) {
```

[1] Jean Michel Garnier es un desarrollador J2EE francés que al momento del inicio de este trabajo comentaba que desarrollo el código mientras encontraba empleo. A la fecha gracias a esta publicación pudo a darse a conocer y hoy labora en Inglaterra. Su sitio personal ha crecido considerablemente en artículos publicados por el mismo.

```
                break;
            }

            //  Agrega el vértice mas cercano al conjunto de vértices determinados
                 determinedVerticesSet.add(closest);

            // Relajación
            relax(closest);
        }
    }
```

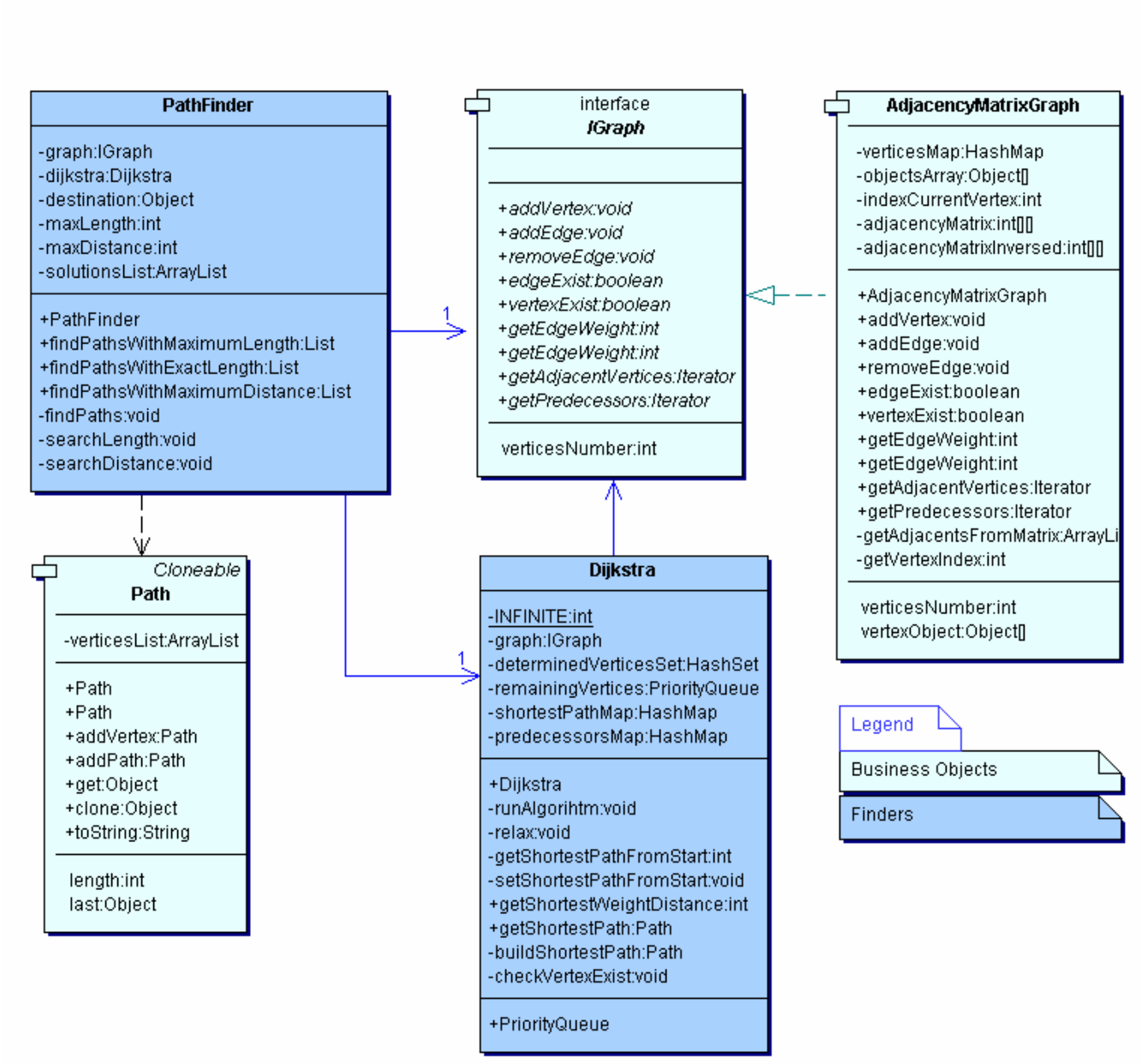
La cola de prioridad, (PriorityQueue)

Para encontrar el elemento más cercano se usa una *cola de prioridad.*

El método de relajación, (relax method)

```
    /**
     * El proceso de relajación actualiza el costo de todos los vértices V
     * conectados al vértice U, si mejoramos la estimación de la ruta mas
     * corta a V incluyendo (U, V) en la ruta a V.
     * @param vertex cuyos vértices adyacentes deben ser relajados.
     */

    private void relax(Object vertex) {
        // Itera en los vértices adyacentes del vértice que es relajado.

        Iterator adjacentVertices = graph.getAdjacentVertices(vertex);

        while (adjacentVertices.hasNext()) {
            Object adjVertex = adjacentVertices.next();

            // No relaja los elementos que ya han sido determinados
            if (!determinedVerticesSet.contains(adjVertex)) {
                // distancia = la distancia mas corta desde
                 // el origen + distance(vertex, adjacent vertex)

                int distance = getShortestPathFromSource(vertex) +
                               graph.getEdgeWeight(vertex, adjVertex);

                // Hemos encontrado la ruta mas corta
                if (getShortestPathFromSource(adjVertex) > distance) {
                    // actualiza el mapa resultante de la ruta mas corta
                    setShortestPathFromStart(adjVertex, distance);

                    // actualiza el mapa resultado de predecesores
                    predecessorsMap.put(adjVertex, vertex);

                    // re-balancea los vértices remanentes de acuerdo a la nueva
                    // distancia mas corta encontrada.
                    remainingVerticesQueue.insert(adjVertex, distance);
                }
            }
        } // fin del while
```

Diagrama de Clases de la implementación



Clases de la implementación

IGraph, Es la interfase que especifica un grafo orientado y cargado. Todos los algoritmos de búsqueda y clases cliente usan esta interfase para interactuar con grafos. Si la implementación cambia, no se necesitará ningún cambio en las clases cliente.

AdjacencyMatrixGraph Es una implementación del grafo usando una matriz de adyacencia.

Path Es la generalización del concepto de ruta. Es una lista de objetos. [O1--O2--O3--04]

PathFinder Contiene todos los métodos de búsqueda.

Dijkstra Es una implementación del algoritmo de Dijkstra para la búsqueda de la distancia mas corta en un gráfo con peso.

Dijkstra.PriorityQueue Es una clase interna de Dijkstra usada por el algoritmo de Dijkstra para escoger el vértice más cercano (El que tiene la prioridad mas baja).

Dijkstra.PriorityQueue.QueueElement Es un elemento de la cola. Contiene un objeto y su prioridad.

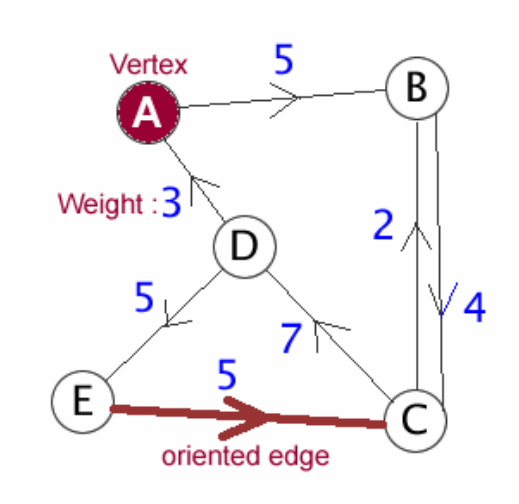
Implementación de IGraph

Realizando un diseño simple, La implementación de IGraph es lo mas simple como ha sido posible: Una matriz de adyacencia AM, con AM[inicio][destino] = peso. Si AM[inicio][destino] = 0, la ruta no existe. Esta implementación probablemente no es la más rápida. Más aún, si la gráfica tiene un número muy grande de vértices y pocos arcos, desperdiciará memoria. El escoger una implementación del grafo dependiendo del número de vértices y arcos tendrá un mejor desempeño. Otras implementaciones posibles son una lista y arco de adyacencia.

El diseño de la clase del grafo es lo más genérica posible, el contenido del grafo son objetos y el peso de un arco es un números entero.

Caso de prueba usando esta implementación

Supongamos que tenemos el siguiente grafo con vértices A, B, C, D, E.



En base a este grafo queremos responder a 3 preguntas:

¿Cual es la ruta mas corta de A a D?
¿Cuál es la ruta mas corta de C a E?
¿Cuál es el número de diferentes rutas de A a C con una distancia de menos de 20?

Usando JUnit definimos primero las pruebas, para saber que tenemos que obtener.

```
public class LetterGraphFinderTest extends TestCase {

    private LetterGraphFinder finder;

    public LetterGraphFinderTest(String s) {
        super(s);
    }

    protected void setUp() {
        // Configura el grafo antes de cada prueba
        LetterGraphBuilder builder = new LetterGraphBuilder(5);
        builder.addPath('A', 'B', 5);
        builder.addPath('B', 'C', 4);
        builder.addPath('C', 'B', 2);
        builder.addPath('C', 'D', 7);
        builder.addPath('E', 'C', 5);
        builder.addPath('D', 'A', 3);
        builder.addPath('D', 'E', 5);

        // Instancia el buscador
        finder = new LetterGraphFinder(builder.getLetterGraph());
    }

    protected void tearDown() {
    }

    public void testFindShortestPath() {
        assertEquals("A-B-C-D", finder.findShortestPath('A', 'D'));
        assertEquals("C-D-E", finder.findShortestPath('C', 'E'));
    }

    public void testFindPathsWithMaximumDistance() {
        ArrayList solutions = (ArrayList) finder.findPathsWithMaximumDistance('A',
'C', 20);



        // Itera en la solución, incrementa el contador por cada vez que se
encuentra una solución correcta
        int numberOfCorrectSolutions = 0;
        Iterator iter = solutions.iterator();

        while (iter.hasNext()) {
            String sol = iter.next().toString();
            if (sol.equals("A-B-C") || sol.equals("A-B-C-B-C")) {
                numberOfCorrectSolutions++;
            }
        }

        // ¿Cuantas soluciones buscar?
        assertEquals(2, numberOfCorrectSolutions);
    }
}
```

Clase LetterGraph:

```
public class LetterGraph {
    /**
     * La implementación del grafo se delega a la clase Builder
     */

    private IGraph graph;
    /**
     *
     * @param graph
     * @param numberOfLetters
     */
    public LetterGraph(IGraph graph) {
        this.graph = graph;
    }

    /**
     * Agrega un Nuevo tramo al grafo.
     * @param String nodo inicial de 1 caracter
     * @param String nodo final de 1 caracter
     * @param weight
     */
    protected void addPath(String start, String end, int weight) {
    // Los parametros no se verifican por que la clase AdjencyMatrixGraph lo hace
        graph.addEdge(start, end, weight);
    }

    /**
     * @return retorna el grafo
     */

    protected IGraph getGraph() {
        return graph;
    }
}
```

La clase Builder

```
public class LetterGraphBuilder {
    /**
     * Escogemos implementer el grafo con una matriz de adjacencia
     */
    private AdjacencyMatrixGraph graph;

    /**
     * El objeto que vamos a construir
     */
    private LetterGraph letterGraph;

    /**
     *
     * @param numberVertices
     */
```

```
    public LetterGraphBuilder(int numberVertices) {
        // There is no controls
        graph = new AdjacencyMatrixGraph(numberVertices);

        letterGraph = new LetterGraph(graph);

        // Agrega los vertices "A", "B", ... "Z"
        for (int i = 0; i < numberVertices; i++) {
            graph.addVertex( String.valueOf( (char) ('A' + i)) );
        }
    }

    /**
     * Agrega un tramo al grafo, envolviendo el metodo LetterGraph addPath
     * @param inicia el vertice
     * @param fin del vertice
     * @param weight
     */

    public void addPath(char start, char end, int weight) {
        letterGraph.addPath(String.valueOf(start), String.valueOf(end), weight);
    }

    /**
     *
     * @return retorna el grafo que se ha construido
     */
    public LetterGraph getLetterGraph() {
        return letterGraph;
    }
}
```

La clase Finder

```
public class LetterGraphFinder {
    // De hecho, la clase LetterGraphFinder es un wrapper alrededor de PathFinder
    private PathFinder finder;

   /**
     * @param letterGraph el grafo que vamos a usar!
     */

    public LetterGraphFinder(LetterGraph letterGraph) {
        finder = new PathFinder(letterGraph.getGraph());
    }

    /**
     * @param letra inicial
     * @param letra destino
     * @return la ruta mas corta en formato String (A-B-C) o una cadena vacia si no
hay ruta */

    public String findShortestPath(char start, char destination) {
        String s = String.valueOf(start);
        String d = String.valueOf(destination);
```

```
        return finder.getShortestPath(s, d).toString();
    }

    /**
     * Explorar el grafo desde un vertice inicial a un vertice final y encontrar
todas las rutas que tiene con una distancia total <= maxDistance. Si el destino se
alcanza no importa hasta parar la condicion
     * @param letra inicial
     * @param letra final
     * @param maxDistance la distancia es el total de pesos entre 2 vertices.
     * @return
     */

    public List findPathsWithMaximumDistance(char start, char destination, int
maxDistance) {
        String s = String.valueOf(start);
        String d = String.valueOf(destination);

        return finder.findPathsWithMaximumDistance(s, d, maxDistance);
    }
}
```

La salida de estas pruebas, dan respuesta a las preguntas iniciales:

¿Cual es la ruta mas corta de A a D?

R: A-B-C-D

¿Cuál es la ruta mas corta de C a E?

R: C-D-E

¿Cuál es el número de diferentes rutas de A a C con una distancia de menos de 20?

R: A-B-C, A-B-C-B-C

Adaptación de las clases de esta implementación.

Como puede verse del diagrama de clases, el costo en AdjacencyMatrix es un valor entero.

La variable de análisis del sistema es la distancia mas corta, se mide en metros y puede ser tener decimales. La clase de implementación maneja solo valores enteros por lo que es necesario hacer una refactorización en la variable para convertirla a un tipo float o double.

Para esta clase se hace un cambio de tipo en AdjacencyMatrix a double para soportar valores decimales.

Variable de análisis del algoritmo

El costo se obtendrá de la distancia de cada tramo.

Aplicación del algoritmo

1. Se consultan los puntos disponibles para la red elegida desde la BD y se agrega las claves de los puntos como cadenas a la ***AdjacencyMatrix*** con el método ***addVertex***()
2. Se consultan los tramos que pertenecen a la red seleccionada y se agregan las claves de los puntos que identifican el tramo a la ***AdjacencyMatrix*** con el método ***addEdge***()
3. Esta matriz se pasa a ***PathFinder*** y se aplica el método ***getShortestPath***() pasando como parámetro la clave del punto inicial y final.
4. ***getShortestPath*** retorna una cadena con las claves de los identificadores de cada punto. Mediante otra consulta se obtienen los datos de los puntos y tramos de la solución consultando por clave de punto.

Diseño del componente 2. Módulo de Mantenimiento

Elementos de diseño del Módulo de Mantenimiento

Descripción del Módulo de Mantenimiento

El módulo de mantenimiento está separado del módulo de consulta, y debe tener acceso solo con un usuario administrador y una contraseña.

Este módulo cuenta con las siguientes pantallas:

1. Pantalla de acceso al sistema.
2. Pantalla principal, de acceso a las demás pantallas de mantenimiento.
3. Pantalla de mantenimiento de Redes y pantallas de utilidad.
   La descripción del funcionamiento de esta pantalla se encuentra en el caso de uso 2
4. Pantalla de mantenimiento de Tipo de Punto
   La descripción del funcionamiento de esta pantalla se encuentra en el caso de uso 3
5. Pantalla de Mantenimiento Gráfico de Puntos y tramos

La descripción del funcionamiento de esta pantalla se encuentra en el caso de uso 4 y 5

Pantallas

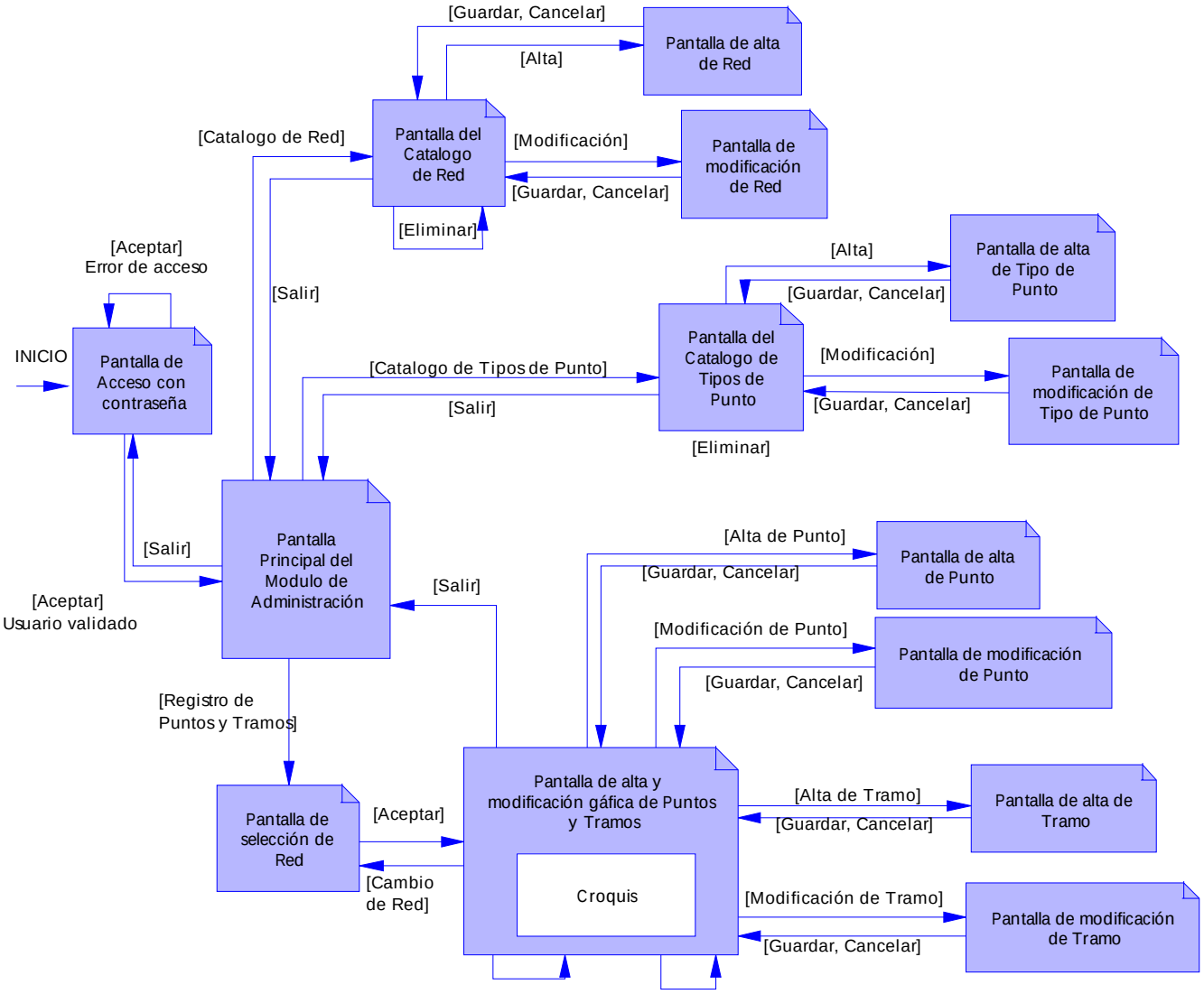


Fig. Mapa de Navegación del Módulo de Mantenimiento

Clases del modelo en el Módulo de Mantenimiento

No importando el lenguaje de programación, las siguientes clases deben ser codificadas para guardar, modificar y eliminar la información de la red.

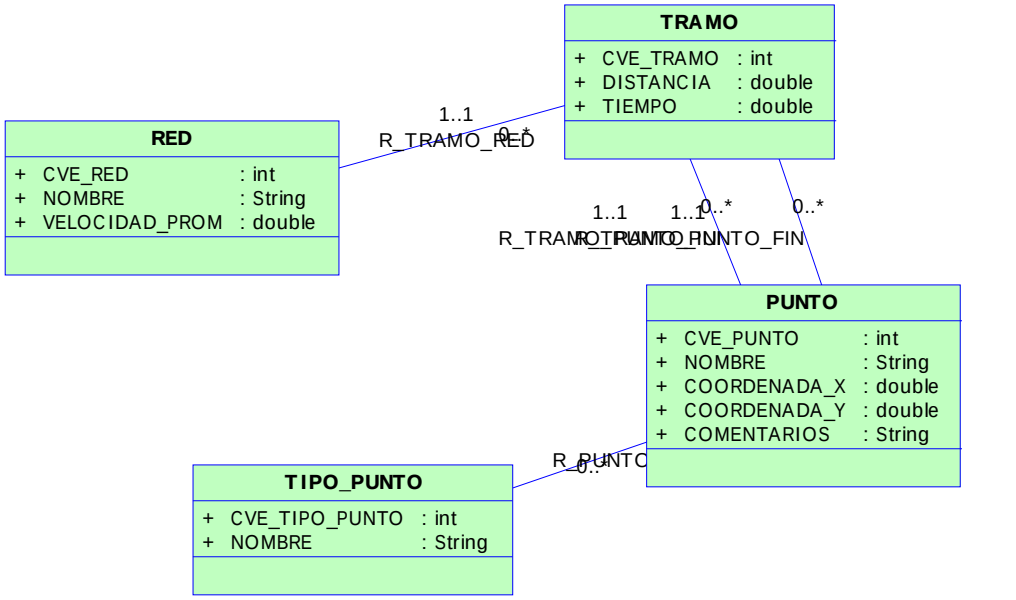


Fig. Clases del modelo que deben usarse en el Módulo de Mantenimiento

Imagen de Ciudad Universitaria

Esta imagen se usara como mapa de Ciudad Universitaria.

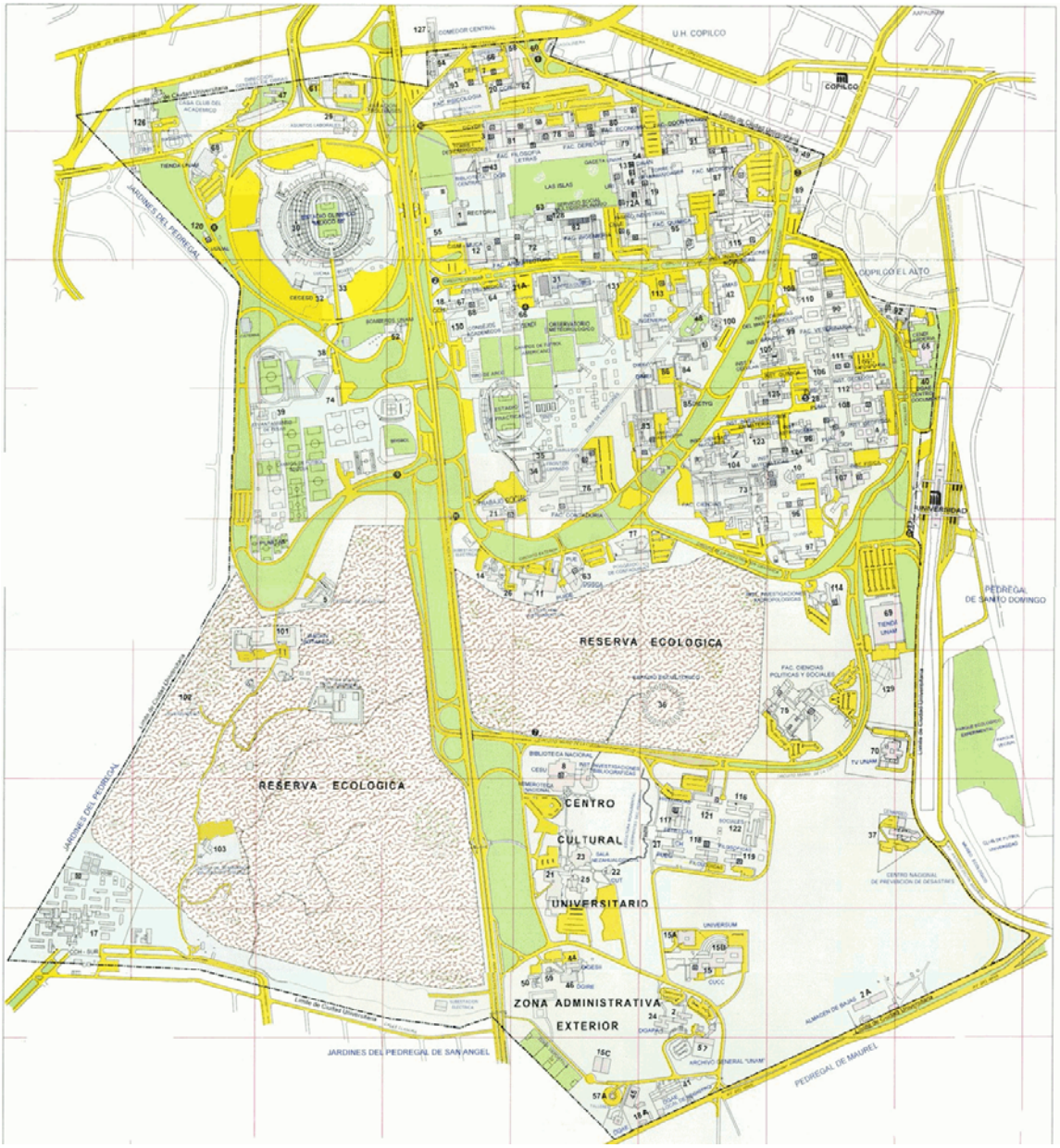


Fig. Imagen del mapa de Ciudad Universitaria que se usará

Fuente: Imagen tomada de la pagina “Servicio de Localización de Edificios en el campus de CU” en http://www.mapa.unam.mx/

Dimensiones: 1400 x 1500 píxeles

Formato: jpg

Elemento gráfico

La herramienta principal de este módulo es el alta y modificación gráfica de los puntos y tramos de la red.

Estas son las especificaciones:

- La ventana de alta gráfica tendrá tres tablas (grids) para consulta de redes, puntos y tramos disponibles. Y un área grafica donde se carga la imagen del mapa de CU.
- El formato de la imagen es jpg para lograr una mayor resolución y para que sea soportado por el navegador Web. Esta imagen se obtuvo de la imagen actual de un sistema de la página principal de la UNAM. Las dimensiones son: 1400x1500 píxeles

  El mapa de cubre toda Ciudad Universitaria, para visualizarla en el área gráfica de la ventana debe tener barras de desplazamiento
- Los valores de la coordenada son referidos desde la esquina superior izquierda, desde

  0 a 1400 para el eje X y

  0 a 1500 para el eje Y

  Esta misma orientación y valores se usaran en el mapa gráfico en la aplicación Web.
- Al abrir la ventana se cargan automáticamente todas las redes disponibles. Al seleccionar una red se muestran en el segundo grid todos los puntos disponibles para esta red, al elegir un punto se muestran en el tercer grid todos los tramos asociados al punto. El punto y los tramos asociados deben mostrarse también gráficamente en el mapa.
- En el mapa se distingue un punto inicial, final e intermedio con colores diferentes. Los tramos se distinguirán con una línea.

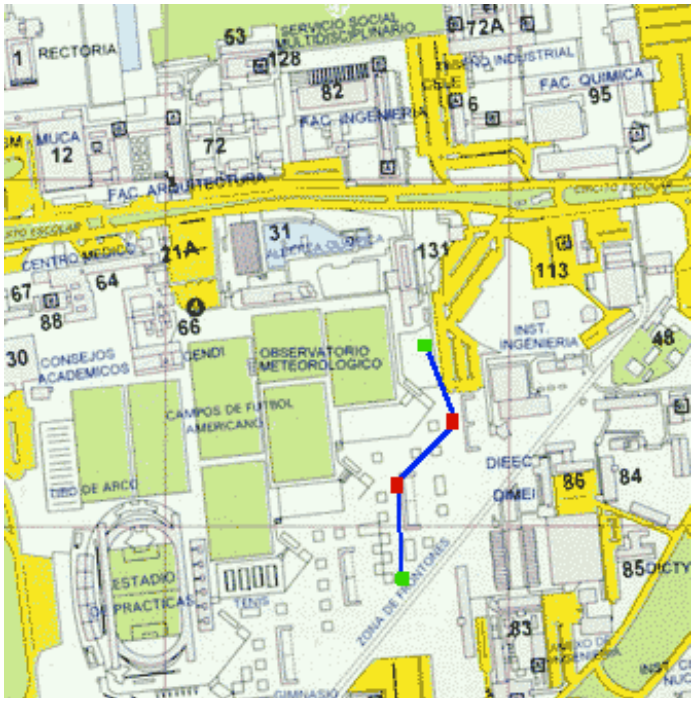


Fig. Los puntos y tramos se distinguirán con los colores mostrados.

- Para hacer el alta de un punto el usuario dará clic sobre el grafico, a continuación aparece una ventana mostrando las coordenadas del punto elegido y los campos disponibles para la captura del nombre, descripción y coordenadas del punto.

- Para hacer el alta de un tramo debe elegirse primero el punto inicial con un clic sobre el gráfico y un segundo clic para el punto final. El sistema recuerda esta sucesión de clics y guarda los datos de los puntos, a continuación se da clic a un botón “Alta de tramo” y el sistema despliega una ventana con los datos de estos puntos inicial y final.
    - El sistema calcula la distancia aproximada real calculando la distancia en píxeles entre los 2 puntos y multiplicando esta distancia con por el factor de la escala del gráfico.
- Para hacer una modificación de los datos de un punto, se da clic sobre el y a continuación se elige el botón “Modificar”.
- Para hacer una modificación de los datos de un tramo, se da clic sobre el tramo y a continuación se da clic al botón “Modificar”
- Para hacer una borrado de un punto, se da clic sobre el y a continuación se elige el botón “Eliminar”, el sistema debe pedir la confirmación del borrado con un mensaje.
- Para hacer una borrado de un tramo, se da clic sobre el y a continuación se elige el botón “Eliminar”, el sistema debe pedir la confirmación del borrado con un mensaje.

Consideraciones importantes acerca del trazado de la ruta.

- Inicialmente en el sistema se registrarán puntos importantes, si entre dos puntos importantes no se tiene puntos intermedios que den forma al recorrido, lo que sucederá es que al momento del trazar una ruta, esta puede salir muy “cuadrada”, en ciertas situaciones parecerá que una línea sobre el grafico pasa sobre lugares a los que no se tiene acceso.

  Para resolver esta situación se creara un tipo de punto que se llame “Conexión” que ayudará a dar forma a la ruta de recorrido pero solamente para eso. Estos puntos de conexión no se incluirán en el listado de puntos disponibles Origen y Destino de la pantalla de consulta, sólo ayudarán a dar forma al recorrido.

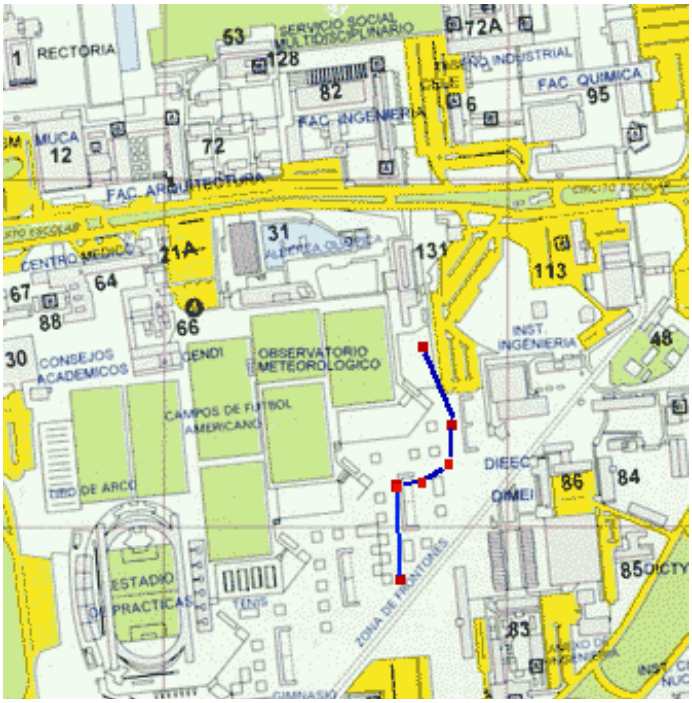


Fig. Los tramos de conexión darán una mejor aproximación de la ruta a recorrer evitando que la traza sea cuadrada o que pase sobre lugares imposibles como en el grafico anterior.

Diseño del Componente 3. Módulo de Consulta

Pantallas, navegación del sistema.

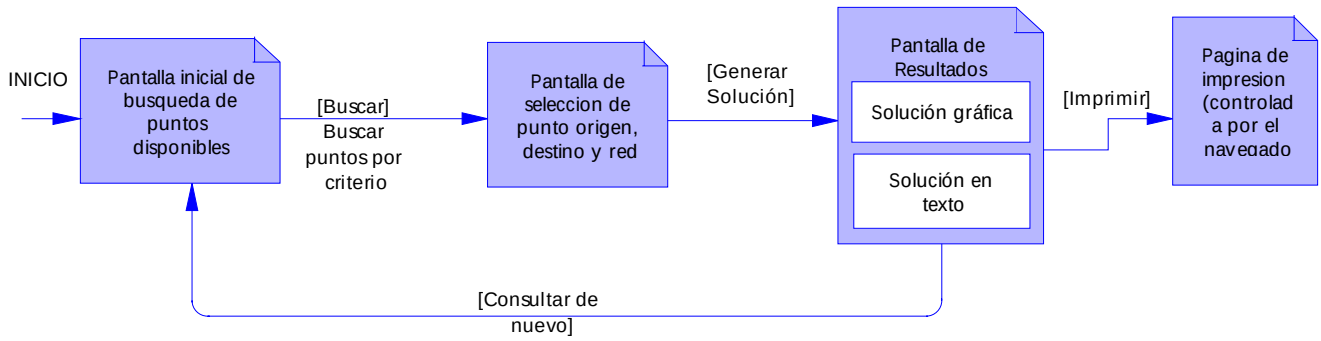


Fig. Mapa de Navegación del Módulo de Consulta

La lógica de navegación muestra que solo es necesario 3 pantallas para realizar la consulta.

Con el detalle descrito en el caso de uso 1, puede verse de manera general que la lógica de consulta es la siguiente:

1. El usuario entra a la pantalla inicial de consulta en P1
2. Establece criterios para la búsqueda de los puntos inicial y final su ruta en P1
3. Selecciona la red y los puntos inicial y final en P2 y lanza la consulta
4. El sistema despliega los resultados en P3, puede imprimir los resultados y el navegador despliega una ventana de impresión P4. Si el usuario desea otra consulta regresa a P1.

Modelo de capas del sistema

El sistema se diseñara en capas para hacer más flexible su mantenimiento.

Capa de presentación	Es la parte visible del sistema. Se compone de todas las clases y paginas que serán accesibles para el usuario. El objeto gráfico para la presentación del resultado esta en esta capa.
Capa de Control	La capa de control se encarga de la secuencia del sistema. Transfiere los datos de entrada y salida entre las capas de Presentación y Modelo.
Capa de Modelo	La capa de modelo tiene todas las clases para llevar a cabo la funcionalidad del sistema. Las clases de esta capa son invocadas haciéndoles llegar los datos de entrada por medio de clases Data Transfer Object (DTOs).
Capa de Datos	La capa de datos oculta los detalles del tipo de base de datos usada y se encarga de extraer y modificar los datos del modelo, en este caso el inventario de redes, tramos y puntos. Concretamente clases Data Access Object (DAOs).
	Base de datos

Diagrama general de clases del Módulo de Consulta

El siguiente es el diagrama general de clases, considerando el modelo de capas descrito.

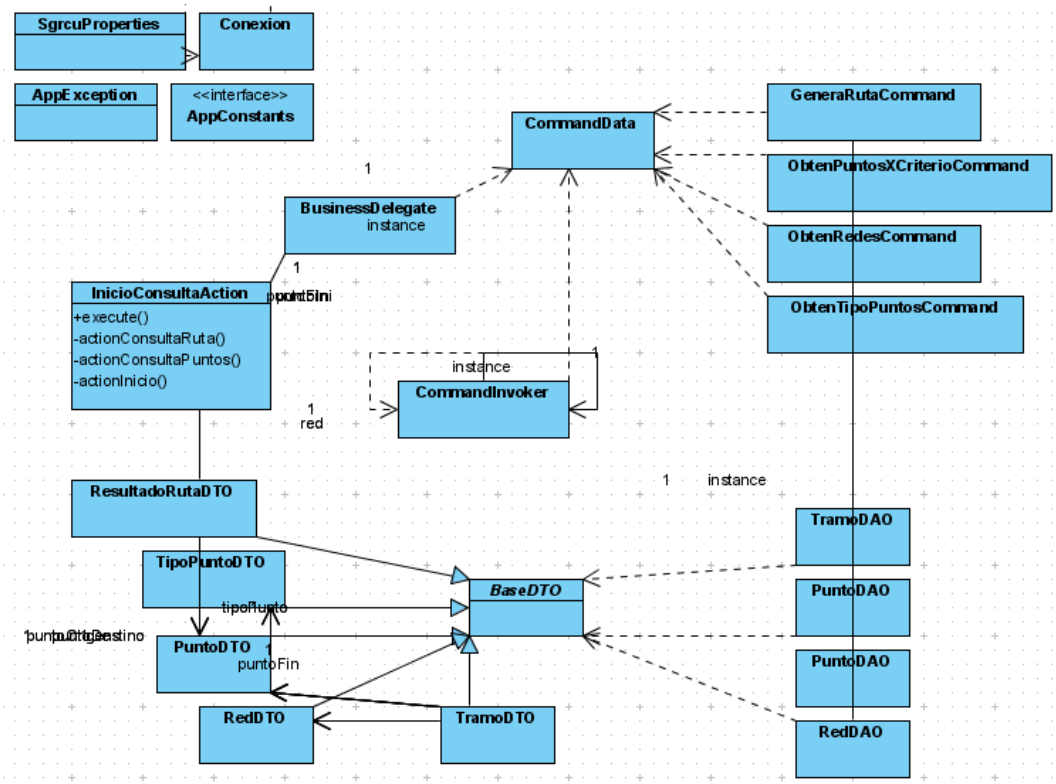


Fig. Diagrama General de Clases

Capa de Presentación

Siguiendo el patrón Modelo Vista Controlador (MVC), la capa de presentación es independiente de la capa de modelo donde residen las operaciones y el acceso a datos.

Los elementos de la capa de presentación pueden ser paginas jsp, una combinación de Struts con jsp o Java Server Faces. En el diagrama no se muestra ninguna clase específica para permitir que el diseño sea independiente de alguna tecnología.

Si se trabaja con Struts, hay que agregar al diagrama las clases de manera específica, las clases Action, Form y ActionMessage que dan soporte al framework y que trabajan estrechamente con las capas de presentación y control.

En cualquier caso, las clases a usar para intercambiar datos entre las 4 capas: presentación, control, modelo y acceso a datos serán las clases DTO de transferencia de datos.

Capa de Control y Modelo

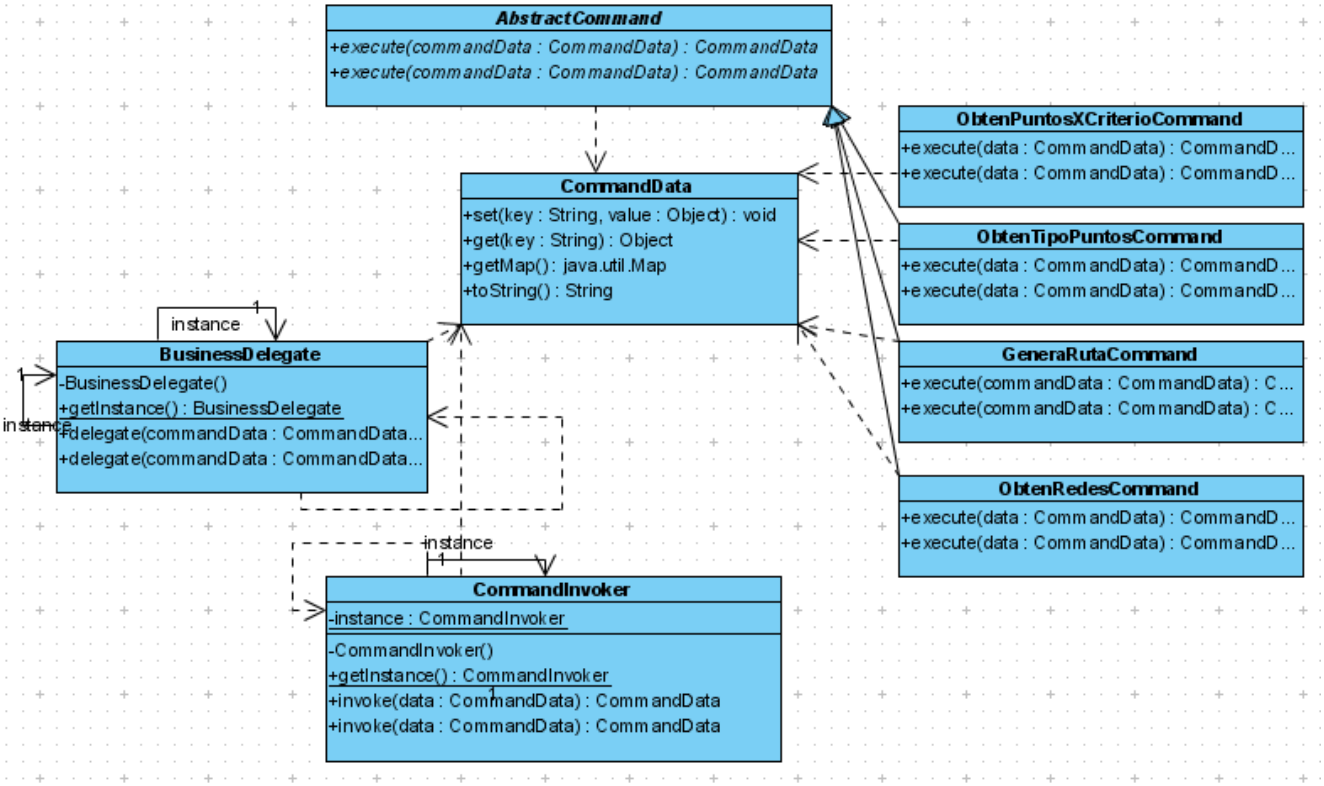


Fig. Clases de la Capa de Control y Modelo

Para la capa de control se usa el Patrón de Diseño *Command*, donde cada transacción tiene asociada una clase Command.

Cada clase command es invocada por una única clase de negocio.

Descripción de las clases:

BusinessDelegate – Esta clase es la única clase de negocio visible desde la capa de presentación. Para hacer la invocación a un comando se usa una instancia de la clase CommandData que tiene los datos necesarios para realizar una transacción y se envía como parámetro a BusinessDelegate, esta a su vez usa CommandInvoker para ejecutar el comando deseado.

CommandInvoker – Esta clase es de control, extrae el identificador del comando a ejecutar de la instancia CommandData.

AbstractCommand - Es una clase Abstracta de la que extienden todas las clases que efectúan una tarea. Todos los comandos deben implementar el método execute, ya que este método es el que se invoca por CommandInvoker.

CommandData – Es la clase para intercambio de datos entre la capa presentación y de control. En esta clase se establece el identificador del comando a ejecutar y los datos de entrada cuando se invoca un comando; y se crea una nueva instancia para devolver los datos de resultado a la capa de presentación.

ObtenPuntosXCriterioCommand – Obtiene los puntos disponibles origen y destino, los datos de entrada son el tipo de red, tipo de instalación y una cadena filtro para el nombre del punto.

ObtenTipoPuntosCommand – Obtiene el catálogo de tipos de punto.

GeneraRutaCommand – Devuelve una instancia de ResultadorutaDTO (dentro de CommandData) con los datos de solución de la ruta. Los datos de entrada son la red, punto origen y punto destino, dentro de CommandData.

ObtenRedesCommand – Obtiene el catálogo de tipos de redes.

Capa de Datos

Clases DAO

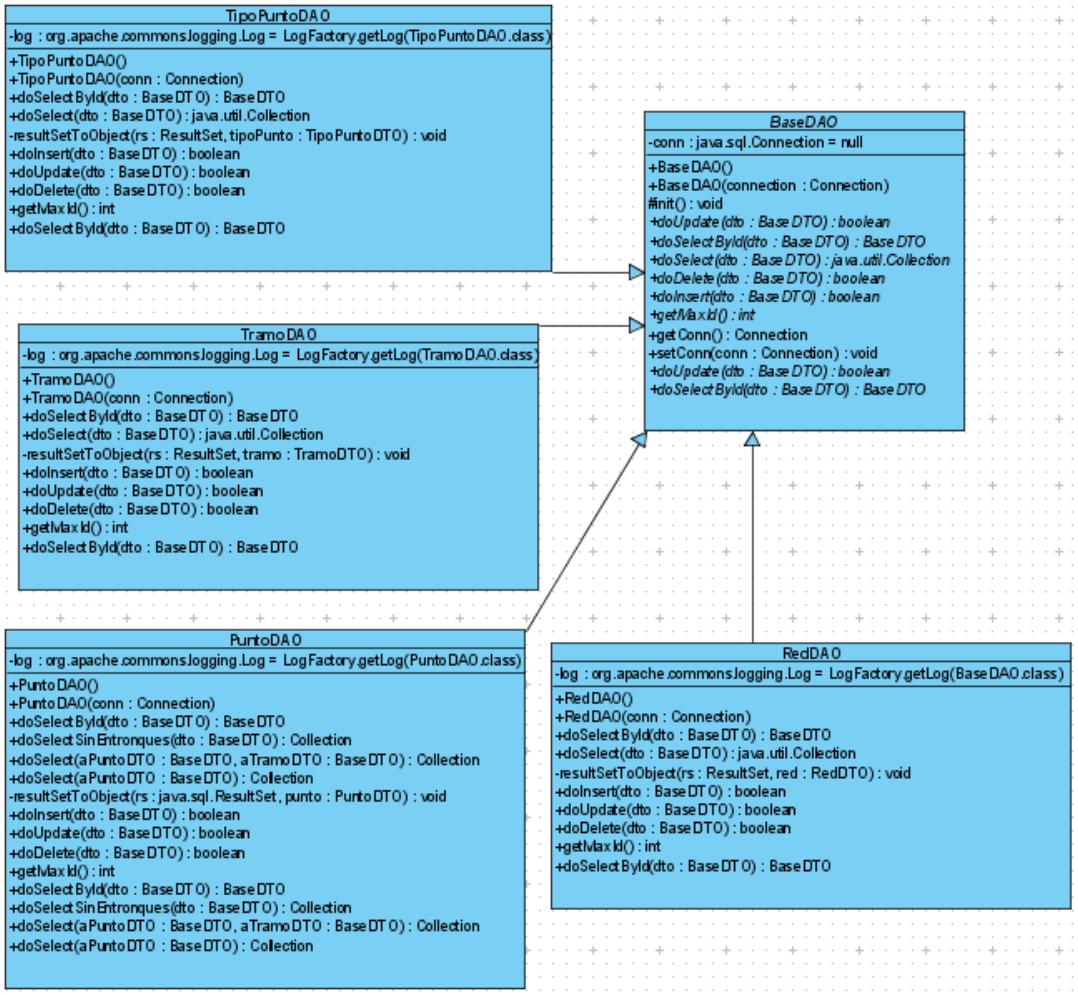


Fig. Clases DAO de la Capa de Datos

Descripción de las clases.

BaseDAO: Clase Abstracta que contiene funcionalidad común de inicialización, obtención cerrado de la conexión y define 4 métodos básicos a implementar de select, update, insert y delete.

PuntoDAO: Clase de acceso a los datos del Punto.

RedDAO: Clase de acceso a los datos de una Red.

TipoPuntoDAO: Clase de acceso a datos del Tipo de Punto.

TramoDAO: Clase de acceso a datos del Tramo

Clases del Modelo en el Módulo de Mantenimiento

Todas las clases del modelo son necesarias desde el Módulo de Mantenimiento. Desde este modelo no hay ninguna operación de actualización de datos, estas se hacen desde el Módulo de Mantenimiento, por lo que los métodos para modificación de datos no deben ser accesibles.

Estas son las clases que representan al modelo de dominio.

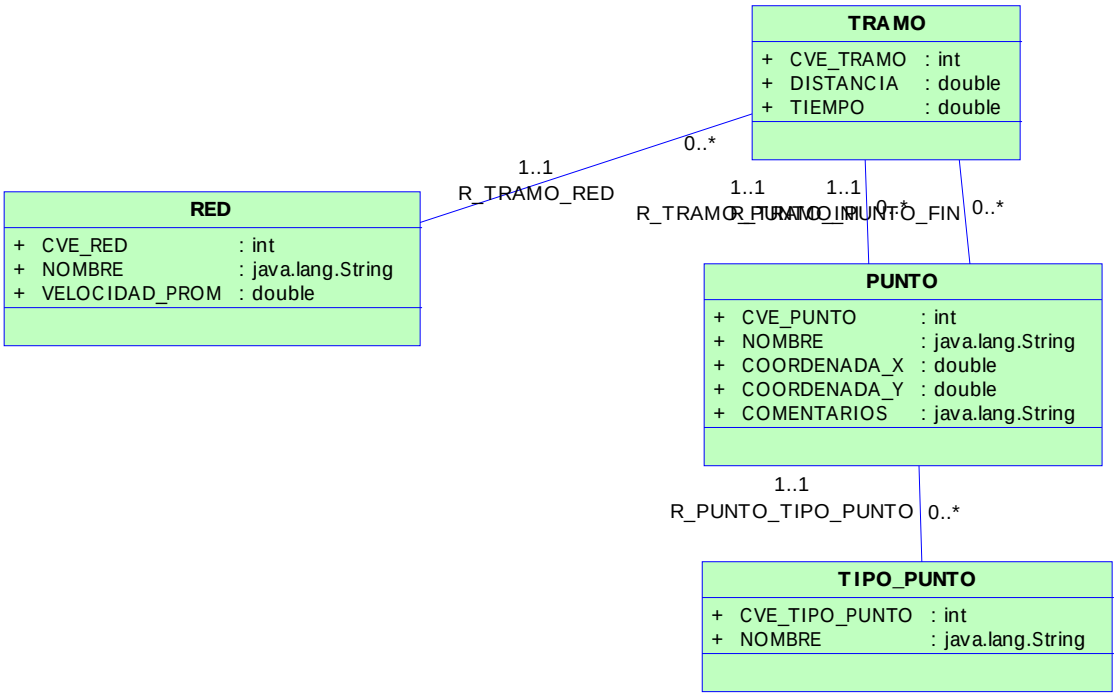


Fig. Clases del modelo que deben usarse en el Módulo de Mantenimiento

Las clases del modelo de dominio serán construidas como Clases DTO de transferencia de datos, es decir tendrán el comportamiento de un Bean y se les agrega el sufijo DTO.

Las operaciones del sistema no se realizaran en estas clases, las realizan las clases command que son modeladas con el patrón Command.

Clases de Transferencia de Datos

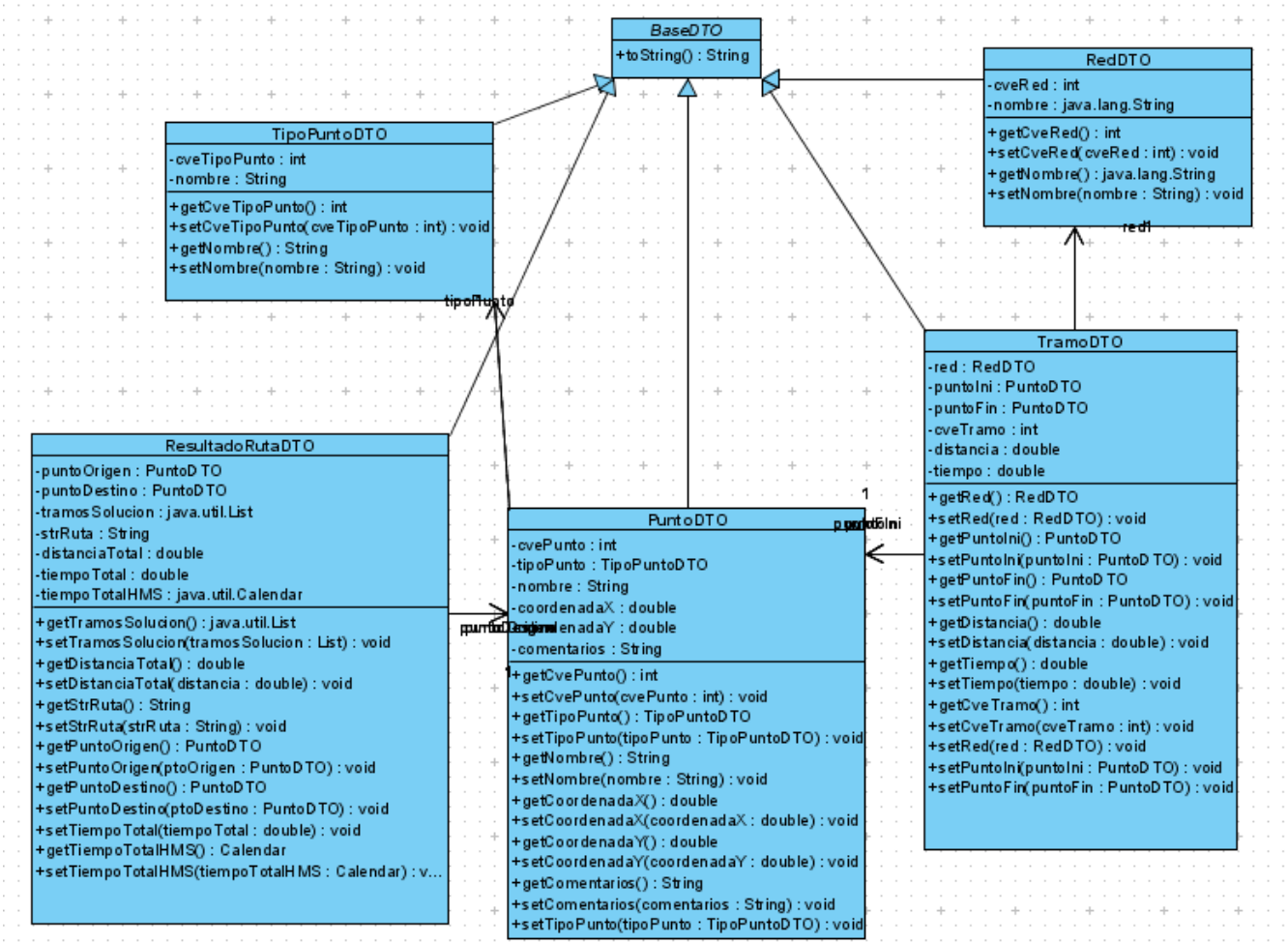


Fig. Clases DTO de la transferencia de datos entre capas

Las clases DTO (Data Transfer Object) usadas para la transferencia de datos entre las capas de Datos, Negocio y Presentación, representan un registro en la base de datos.

Descripción de las clases:

BaseDTO: Clase abstracta de transferencia de datos, tiene un solo método *toString* para imprimir el contenido de los valores del bean.

RedDTO: Clase para transferir entre capas los datos de una Red.

TipoPuntoDTO: Clase para transferir entre capas los datos de un Tipo de Punto.

TramoDTO: Clase para transferir entre capas los datos de un Tramo.

PuntoDTO: Clase para transferir entre capas los datos de un Punto.

ResultadoRutaDTO: Clase para transferir entre capas los datos del resultado de la Ruta. Esta es una de las clases más importantes por que almacena los datos del resultado del proceso de búsqueda de la solución.

Atributos

PuntoDTO puntoOrigen: Es una copia de los datos del punto Origen de la ruta.

PuntoDTO puntoDestino: Es una copia de los datos del punto Destino de la ruta.

List tramosSolucion: Es una lista de objetos de tipo TramoDTO, que contiene cada uno de los tramos de la ruta de solución.

String strRuta: Es la representación en cadena de la ruta, se usa para enviarlo al objeto gráfico y que dibuje la ruta.

double distanciaTotal: Es la suma de la distancia total en metros del recorrido.

double tiempoTotal: Es la suma del tiempo total en segundos del recorrido.

Clases de utilidad

Las siguientes clases son de utilería como ayuda a las demás clases.

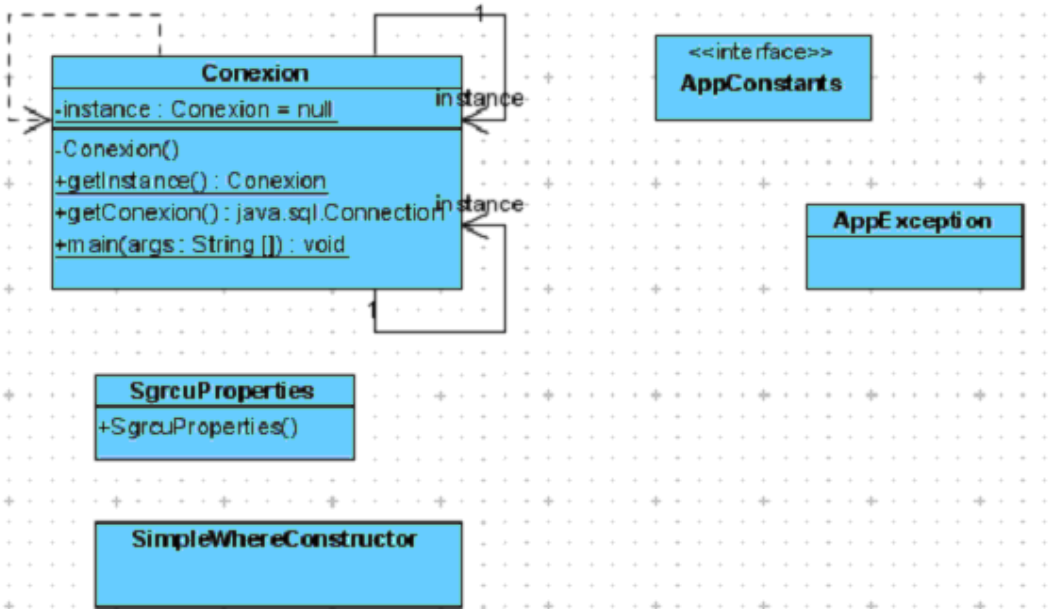


Fig. Clases de utilería

Descripción de las clases.

Conexion: Clase que devuelve un objeto Connection a las clases DAO.

AppConstants: Clase donde se almacenan las constantes de la aplicación, tales como identificadores para guardar datos en la clase CommandData y en el objeto Session.

AppException: Clase personalizada para el manejo de de excepciones.

SgrcuProperties: Clase que recupera valores del archivo de configuración del sistema.

SimpleWhereConstructor: Clase utilizada por las clases DAO para armar una sentencia SQL con una cláusula where.

Diseño del Objeto Gráfico

El objeto gráfico que muestra la ruta de solución, debe poderse ver en el navegador en la tercera pantalla del Módulo de Consulta.

El trazo de la ruta se hará de manera similar que en el Módulo de Mantenimiento, una serie de puntos distinguiendo los puntos iniciales y finales unidos por líneas. Los puntos de conexión no deben aparecer, solo los puntos importantes.

La imagen y las coordenadas son las mismas que se usan en el Módulo de Mantenimiento.

El objeto esta incrustado en la página de solución. A este componente gráfico, se le hace llegar los datos de la solución por medio de la siguiente cadena:

$X1,Y1,NombrePunto_1; X2,Y2,NombrePunto_1; \ldots ;Xn,Yn,NombrePunto_n$

Donde:

X1 – Es la coordenada X del punto referida a la esquina superior izquierda registrada.

Y1 – Es la coordenada Y del punto registrada

NombrePunto – Es el nombre del punto.

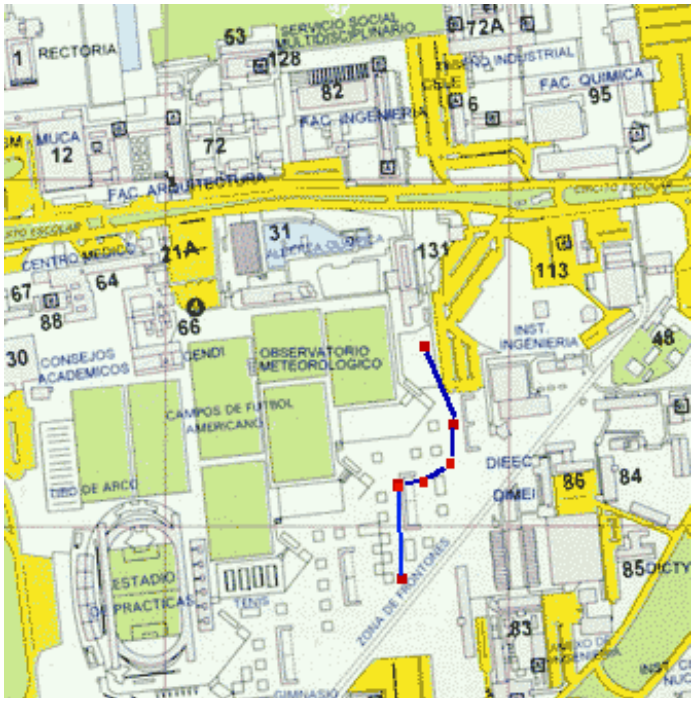


Fig. Apariencia de una ruta de solución en el objeto gráfico visible en la página Web

Clases de la Solución Gráfica

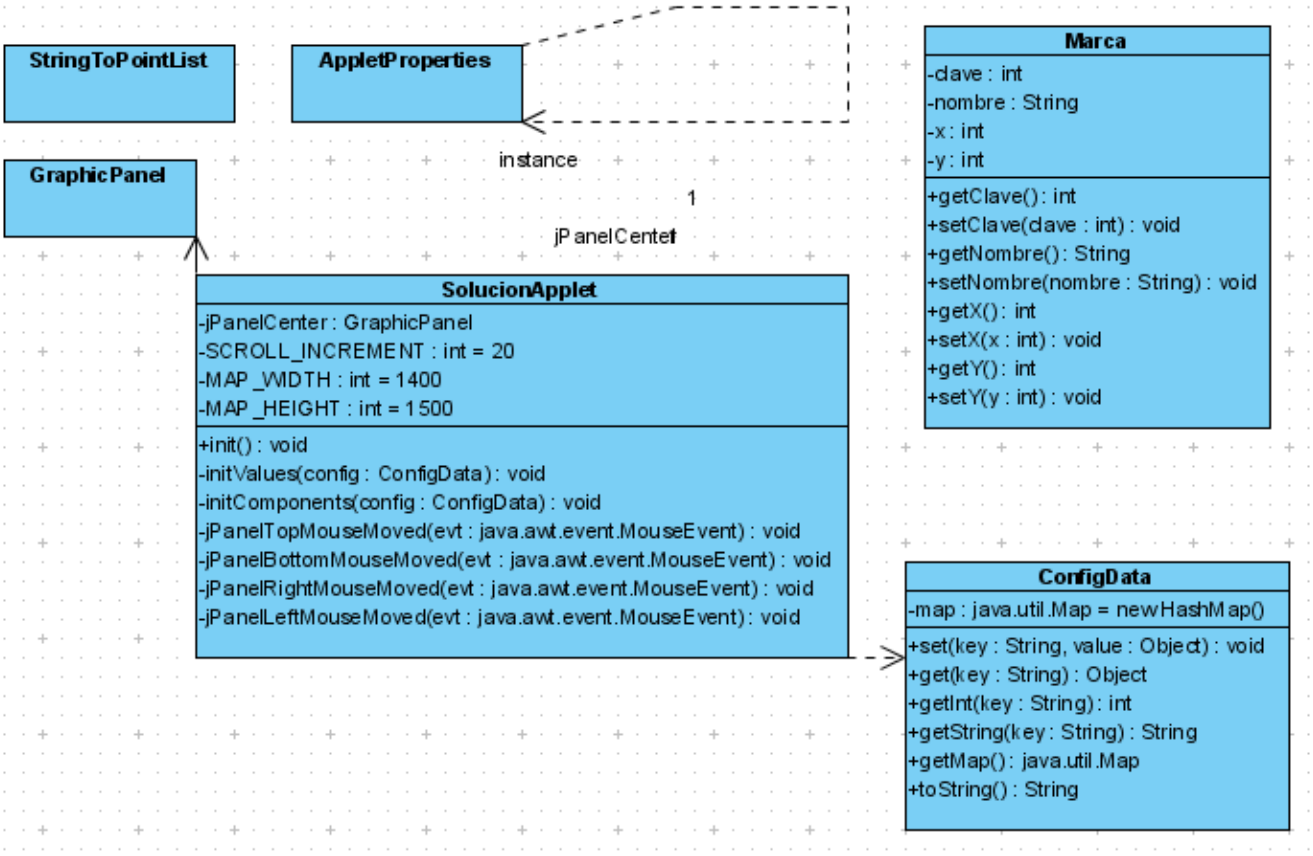


Fig. Clases de utilería

Descripción de las clases

SolucionApplet: Applet para mostrar el mapa de CU.

Marca: Clase da utilería para guardar los datos de la marca a dibujar sobre el mapa, coordenada x, y y nombre.

ConfigData: Clase de utilería para guardar los datos de configuración del applet.

StringToPointList: Clase de utilería para convertir la cadena de puntos a un listado de clases Marca.

AppletProperties: Clase para recuperar los valores desde un archivo de configuración.

Diagramas de Secuencia

Diagrama de secuencia de la solución de la ruta

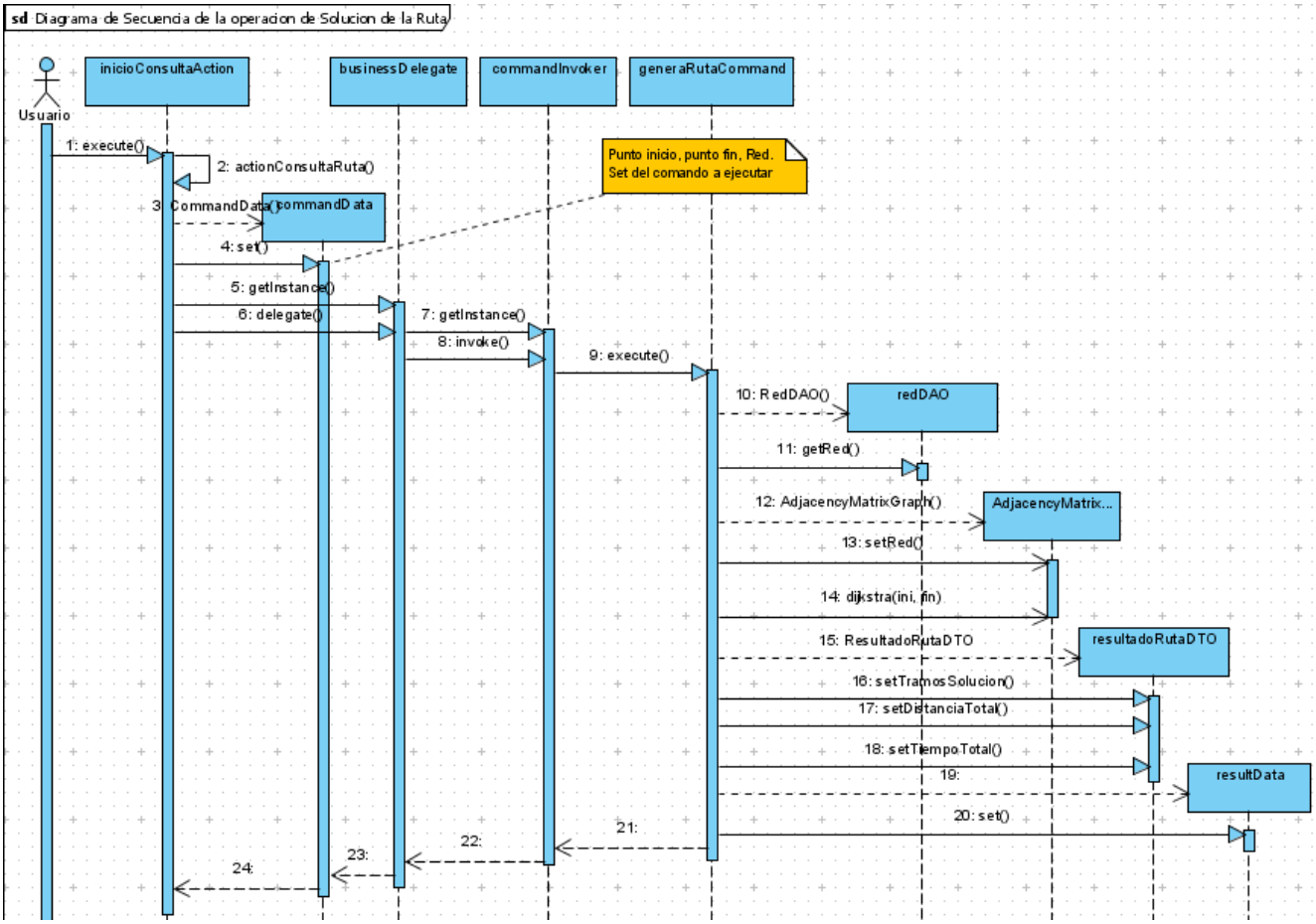
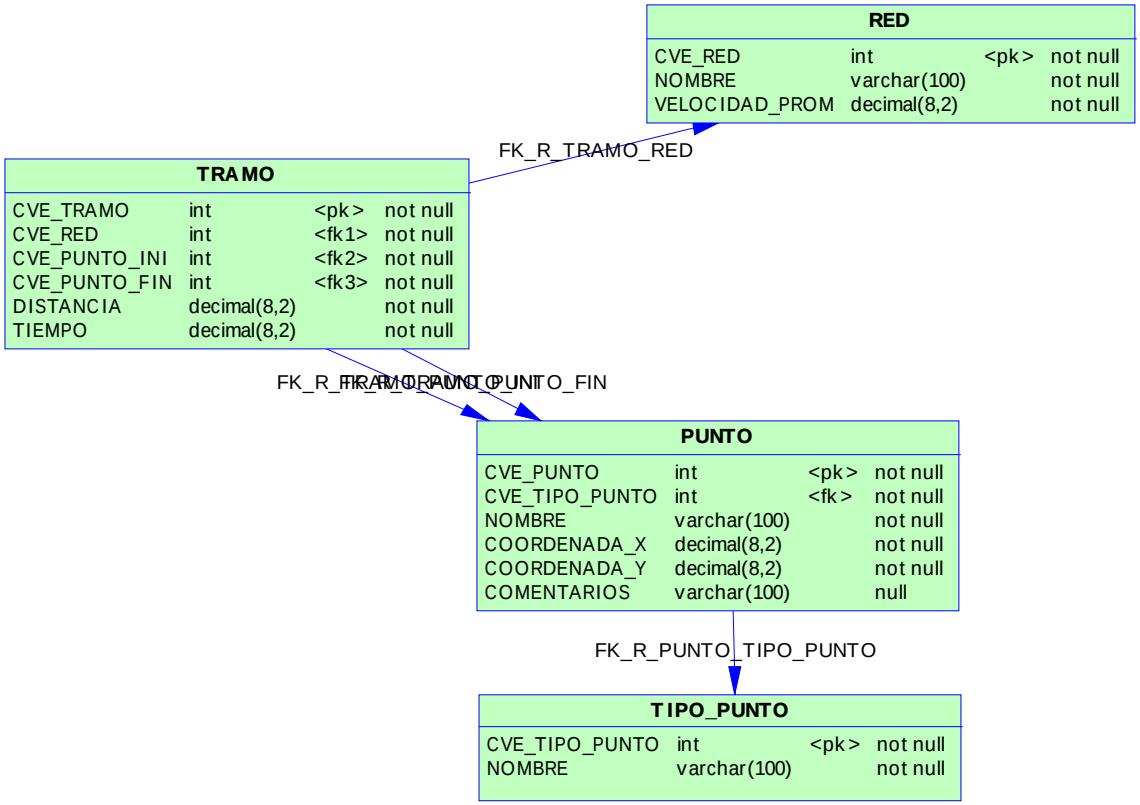


Fig. Diagrama de secuencia de la operación central del sistema: La Solución de la Ruta

Diseño del Componente 4. Base de Datos

El modelo físico de datos relaciona uno a uno las clases del modelo de Dominio, excepto en los casos donde una clase "tiene" a otra clase de otro tipo. Por ejemplo la clase ***Punto*** tiene una instancia de ***Tipo de Punto***. En el caso del modelo físico la entidad ***Punto*** tiene solo la clave correspondiente de la tabla ***Tipo de Punto***

Diagrama físico



Diccionario de datos

1.1 Tabla PUNTOS

Nombre	Descripción	Tipo	M	Pk	Fk
CVE_PUNTO	Clave del punto	int	X	X	
CVE_TIPO_PUNTO	Clave del tipo de punto	int	X		X
NOMBRE	Nombre del punto	varchar(100)	X		
COORDENADA_X	Coordenada en X	decimal(8,2)	X		
COORDENADA_Y	Coordenada en Y	decimal(8,2)	X		
COMENTARIOS	Campos para comentarios adicionales	varchar(100)			

1.2 Tabla RED

Nombre	Descripción	Tipo	M	Pk	Fk
CVE_RED	Clave de la red	int	X	X	
NOMBRE	Nombre del tipo de red	varchar(100)	X		
VELOCIDAD_PROM	Velocidad promedio de recorrido en m/s	decimal(8,2)	X		

1.3 Tabla TIPO_PUNTO

Nombre	Descripción	Tipo	M	Pk	Fk
CVE_TIPO_PUNTO	Clave del tipo de punto	int	X	X	
NOMBRE	Nombre del tipo de punto	varchar(100)	X		

1.4 Tabla TRAMO

Nombre	Descripción	Tipo	M	Pk	Fk
CVE_TRAMO	Clave del tramo	int	X	X	
CVE_RED	Clave de la red	int	X		X
CVE_PUNTO_INI	Clave del punto inicial	int	X		X
CVE_PUNTO_FIN	Clave del punto final	int	X		X
DISTANCIA	Distancia de ir del punto inicial al final en m.	decimal(8,2)	X		
TIEMPO	Tiempo de recorrido en segundos	decimal(8,2)	X		

CAPÍTULO 5. RECURSOS TECNOLÓGICOS ELEGIDOS

En este capítulo se fundamenta la elección de recursos tecnológicos a usar para la construcción del sistema.

Los recursos tecnológicos son el software de desarrollo, software donde se instalará el sistema, código ya construido sobre el que se apoya el desarrollo, cualquier recurso informático que se requiera para el desarrollo del sistema y finalmente, aunque no se mencionan, los IDEs en los que se programará el sistema.

En adelante en lugar del término *software* de desarrollo se usara el término *herramienta* de desarrollo por ser más apropiada

Hay diversas herramientas para la programación y soporte de los componentes del sistema, en una situación real, cuando un cliente contrata a una consultoría en sistemas, muchos de estos recursos ya han sido establecidos por necesidades específicas del desarrollo; en otras ocasiones puede proponerse al cliente el uso de alguna herramienta cuando se sabe que se obtendrá un ahorro en los recursos a invertir.

Para el presente trabajo no hay ningún condicionamiento en el uso de algún software, por lo que se elegirá para la construcción del sistema las herramientas que mas nos convengan usando algunos criterios para la elección.

Antes de esta elección se listan de manera general los componentes a construir que se obtuvieron en los capítulos anteriores.

Componentes a desarrollar

Como se vió en el capítulo anterior los componentes a construir son los siguientes:

- **Módulo de Consulta**, Este módulo de consulta tiene 2 partes importantes
  - **Componente Gráfico** que muestra la solución
  - **Lógica de solución** (Modelo), y Lógica de consulta y Presentación del resultado (Vista)
- **El Módulo de Administración del sistema**

También participan estos componentes de soporte

- **Servidor Web** para desplegar la aplicación Web de consulta
- Servidor de **Base de Datos** para alojar la información del sistema

El siguiente diagrama muestra estos componentes

Diagrama de componentes del sistema

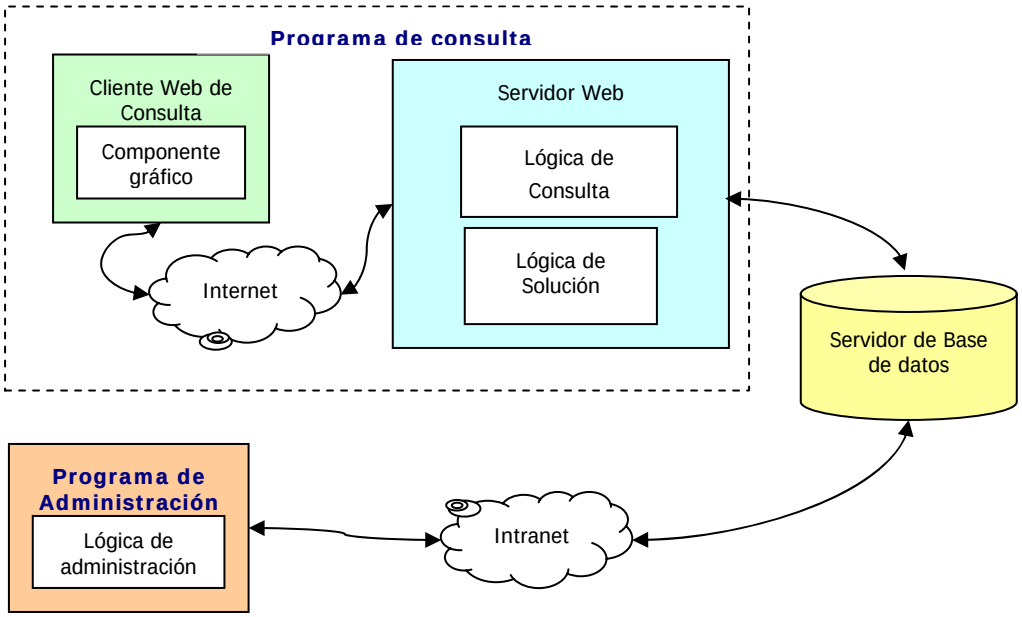


Fig. Diagrama de componentes del sistema

Por lo tanto necesitamos

- Un lenguaje de programación para el módulo Web de consulta y para el Componente Gráfico.
- Un lenguaje de programación para el módulo de administración.
- Una Base de Datos.
- Un Servidor de aplicaciones Web.

Criterios de elección de las herramientas de construcción

Son muy sencillos para no complicar la elección y tienen que ver con los criterios que un consultor de sistemas siempre debe considerar y cuidar en cualquier desarrollo de sistemas que emprenda.

En cualquier lugar y momento los recursos disponibles para llevar a cabo un trabajo son limitados y siempre como ingenieros debemos buscar optimizar el uso de los recursos disponibles. Las limitantes que siempre rigen el desarrollo de un proyecto son Tiempo, Dinero, Esfuerzo y como condición la satisfacción del los requerimientos. Para este trabajo se busca minimizar el Esfuerzo de construcción y cumplimiento de los requerimientos al 100%.

Los criterios son:

1. El desarrollo en un **menor tiempo**: Elegir la herramienta mas adecuada por que permite un desarrollo más rápido o por que ya se poseen los conocimientos sobre esta y la curva de aprendizaje sería mayor si se utilizara otra.
2. El desarrollo con **optimización de esfuerzo**: Reutilizar código ya elaborado, (¿Para que reinventar la rueda?). Integrar las herramientas elegidas con otras. No se debe perder tiempo por el uso de una herramienta nueva, que dice ser mágica.
3. El desarrollo con un **menor costo**: Usar en lo posible software que no requiera de inversión de compra de licencias de desarrollo ni de explotación del producto final. Aquí el *open source* es una excelente opción por su libre uso para fines no comerciales, eficacia y confiabilidad.
4. **Cubrir al 100% los requerimientos del sistema**: No se debe dejar fuera ningún requerimiento por la utilización de una herramienta específica que no permita alguna funcionalidad.
5. El último criterio es basarse en el uso de Java por que ha demostrado ser una excelente herramienta de desarrollo libre que tiene como características: ser orientado a objetos, confiable, escalable, flexible, de fácil aprendizaje y con una amplia comunidad de desarrollo. Esto último es importante ya que ante cualquier duda de cómo resolver algún problema se puede recurrir a la ayuda publicada en los foros de Internet. Además existen muchos *frameworks* java, que facilitan y hacen más rápido el desarrollo.

   Aun con estas cualidades debe tenerse presente que no debe usarse java donde otras herramientas sean mejores, por eso este último criterio es recomendable pero no decisivo.

Herramientas propuestas

A continuación se listan las herramientas propuestas por cada componente, se describen a detalle y al final se hace una elección

Componente	Opción 1	Opción 2	Opción 3	¿Por que estas opciones?
Módulo de Consulta • **Lógica de solución** (Modelo), • **Lógica de consulta y Presentación del resultado** (Vista)	J2EE y Struts	J2EE y Java Server Faces	--------	Java es una excelente opción para desarrollo de sistemas Web
Componente Gráfico	Applet Java Swing	Flash	--------	Se propone una opción java pero por ser un elemento en la capa de cliente, la solución puede ser flash que es muy versátil
Módulo de Mantenimiento	Java Swing	Visual Basic	Python	Se propone java swing, pero en el manejo de gráficos Visual Basic es muy bueno y también Python
Base de datos	MySQL	PostGresSQL	--------	Son las 2 Bases de datos open source mas populares
Servidor de aplicaciones Web	TomCat	JBoss	--------	Son los mejores servidores open source o de uso libre personal

Elección

Evaluación de las características vs. los criterios establecidos.

Criterios / Componente	Herramienta	1. Menor Tiempo	2. Optimización de esfuerzo	3. Menor costo	4. 100% de requerimientos	5. (deseable) En lo posible compatible con java
Base de datos	MySQL	**Si,** Se conoce la herramienta	**Si,** Apto, tiene administradores y hay clientes SQL	**Si,** Es open source	**Si,** Es apta para sistemas pequeños	**100%** Tiene drivers JDBC y ODBC
	PostGress	**Si, pero** Pero no se tiene experiencia, habría una curva de aprendizaje	**Si,** Apto, tiene administradores y hay clientes SQL	**Si,** Es open source	**Si,** Es apta para sistemas pequeños y grandes	**100%** Tiene drivers JDBC y ODBC
Servidor Web	TomCat	**Si,** Se conoce la herramienta	**Si,** La administración es fácil y hay documentación Esta incorporado en la mayoría de los IDES	**Si,** Es open source	**Si,** Si 100% J2EE	**100%** Compatible
	JBoss	**Si, pero** Pero no se tiene mucha experiencia, habría una curva de aprendizaje 50%	**Si, pero** La administración no es difícil y hay documentación No esta incorporado en la mayoría de los IDES	**Si,** Es open source	**Si,** Si 100% J2EE	**100%** Compatible
Programación del sistema Web	Struts	**Si,** Se conoce la herramienta	**Si,** Esta soportado en la mayoría de los IDEs	**Si,** Es open source	**Si,** 100 %	**100%** java
	JSF	**Si, pero** Pero no se tiene experiencia, habría una curva de aprendizaje	**Si,** Esta soportado en la mayoría de los IDEs	**Si,** Es open source	**Si,** 100 % Es mejor que Struts	**100%** java
Programación del Objeto Gráfico	Applet Swing	**Si,** Es un objeto pequeño que no	**Si,** Se puede programar en	**Si,** Es open source	**Si, pero** Pero puede ser lento al	**100%** java

		lleva tiempo	cualquier IDE		desplegars e al cliente	
	Flash	**Si, pero** Pero no se tiene experiencia, habría una curva de aprendizaje	**Si, pero** Pero hay que adquirir el IDE	**No,** Hay que adquirir el IDE y licencias	**Si**	**No es java** pero se integra al 100% la pagina Web
Lenguaje de programación del Módulo de Mantenimiento	Java Swing	**No,** Es un componente grande y la programación con gráficos lleva mas tiempo No se tiene experiencia, habría que invertir tiempo en la curva de aprendizaje.	**Si,** Se integra sin problemas	**Si,** Es open source	**Si, pero** Puede haber algún requerimie nto gráfico que no se cumpla al 100%	**100%** java
	Visual Basic	**Si** El desarrollo es muy rápido	**Si, pero** Seria una solución aislada, no usaría código java hecho y hay que usar ODBC	**No** Hay que adquirir el IDE y licencias	**Si**	**No**
	Python	**Si, pero** No se tiene experiencia. No se conocen los recursos gráficos al 100%	**Si, pero** Seria una solución aislada, no usaría código java preconstruido. No se conoce toda la documentación. No hay IDE de desarrollo	**Si** Es open source	**Si, pero** puede haber algún requerimie nto que no se cumpla	**No** **No es java pero al ser aislada no habría consecuen cias**

Herramientas a usar para el desarrollo del sistema

Componente	Herramienta a usar.	Comentarios
Base de datos	**MySQL**	PostGresSQL y MySQL son buenos, pero como se menciona anteriormente MySQL es más adecuado para sistemas pequeños y ya se conoce. Se usará la versión 4.1
Servidor Web	**Tomcat**	TomCat y JBoss son muy buenas opciones pero como se describió, JBoss Necesita de Tomcat para desplegar paginas. Además ya se conoce Tomcat. Por lo tanto se trabajara con Tomcat. Se usara la versión 5.5 que implementa las especificaciones 2.4 de Servlets, 1.2 de jsp y 1.4 de J2EE
Lenguaje de programación del sistema Web	**J2EE Struts**	Aunque JSF tiene más ventajas por la lógica de programación, Struts es una herramienta en la que tengo experiencia. Por el tiempo de desarrollo del trabajo se usará Struts Se usará la versión 1.2
Lenguaje de programación del objeto gráfico	**Java Swing**	Aquí Flash y un Applet Java Swing son buenas opciones pero sí la capa de negocio es Java hay mas ventajas de usar el componente grafico sea Java. Nos ahorramos la utilización de otro IDE. Se usará un Applet Java Swing Se usara la versión 1.4 de JDK
Lenguaje de programación del sistema de administración	**Visual Basic**	He aquí el dilema, hay varios pros y contras de las herramientas propuestas que hay que analizar: El problema a resolver en este módulo es dar cumplimiento a los requerimientos gráficos que se conoce: dar de alta un punto, ver los disponibles, trazar un tramo. **Visual Basic** es apto para la programación de componentes gráficos. Además el desarrollo es más rápido. Pero hay que programar los componentes de manera aislada. Cuando el sistema este terminado y haya un cambio habría 2 mantenimientos: El de Java en el Módulo de Consulta y el de VB para el Módulo de Mantenimiento Usar **Java** nos da la ventaja de que reutilizamos las clases del modelo y nos ahorramos un IDE. Pero aunque java es potente, no se tiene la certeza de que los requerimientos gráficos se cumplan al 100% Por otro lado Python es una herramienta muy atractiva, pero aun no se tiene la seguridad de que sea apto para el manejo de gráficos al nivel deseado. Se descarta por esta incertidumbre. Decisión:

		Como se tiene la certidumbre que en el manejo de gráficos no se va a tener problema se usara **Visual Basic** a costa de hacer 2 mantenimientos. Se reduce el riesgo de tener que cambiar la herramienta a mitad del desarrollo por que no se puede lograr alguna funcionalidad. El modelo de programación a eventos de Visual Basic puede prestarse a crear código spaghetti. Para evitar esto se cuidará no caer en los errores de programación de Visual Basic.

Modelo de Componentes con las herramientas elegidas

Finalmente se muestran las herramientas a usar para la construcción de cada uno de los componentes del sistema.

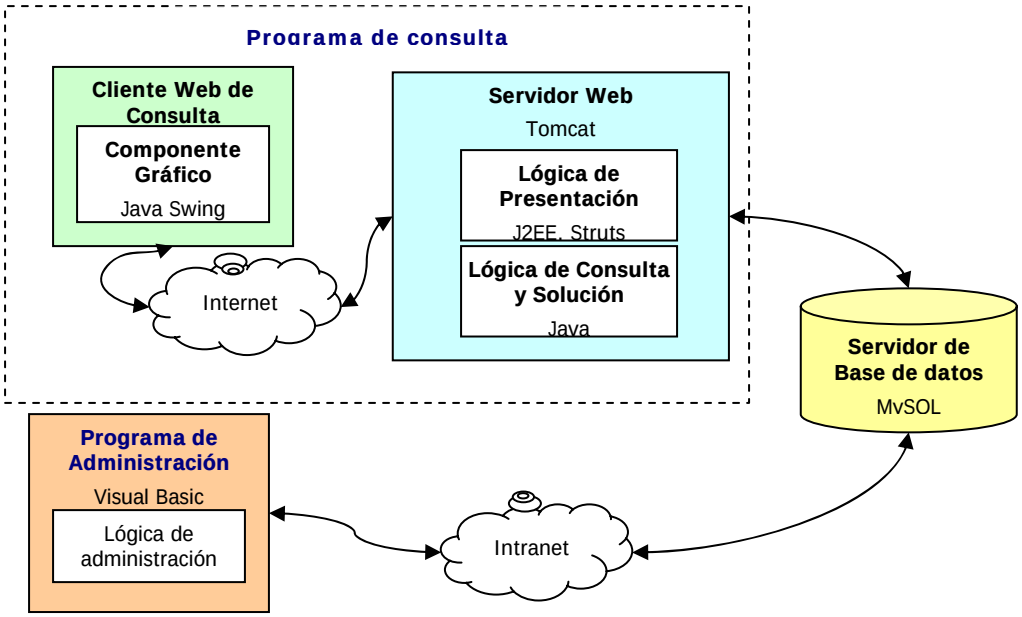


Fig. Herramientas a usar en la construcción del sistema

Definidas las herramientas se inicia la construcción del sistema.

CAPÍTULO 6. CONSTRUCCIÓN DEL SISTEMA

El producto de este trabajo, es software, no es práctico trasladar el código terminado a esta sección. La única forma de comprobar su construcción es observar directamente el código funcionando y hacer pruebas durante su ejecución; por esta razón a continuación se detalla el sistema construido mencionando el número de componentes, su función, los requerimientos y casos de uso que cubre, tal como se hace en una documentación del sistema. Al final del capítulo se mostrará una matriz de seguimiento de los componentes desarrollados vs. requerimientos iniciales. Los códigos de la aplicación se anexan al CD del presente trabajo

No obstante hay partes importantes del sistema que deben revisarse con detalle, en estos casos se ilustrará el código para comentarlo.

Los componentes del sistema, con la respectiva tecnología usada para su desarrollo, son:

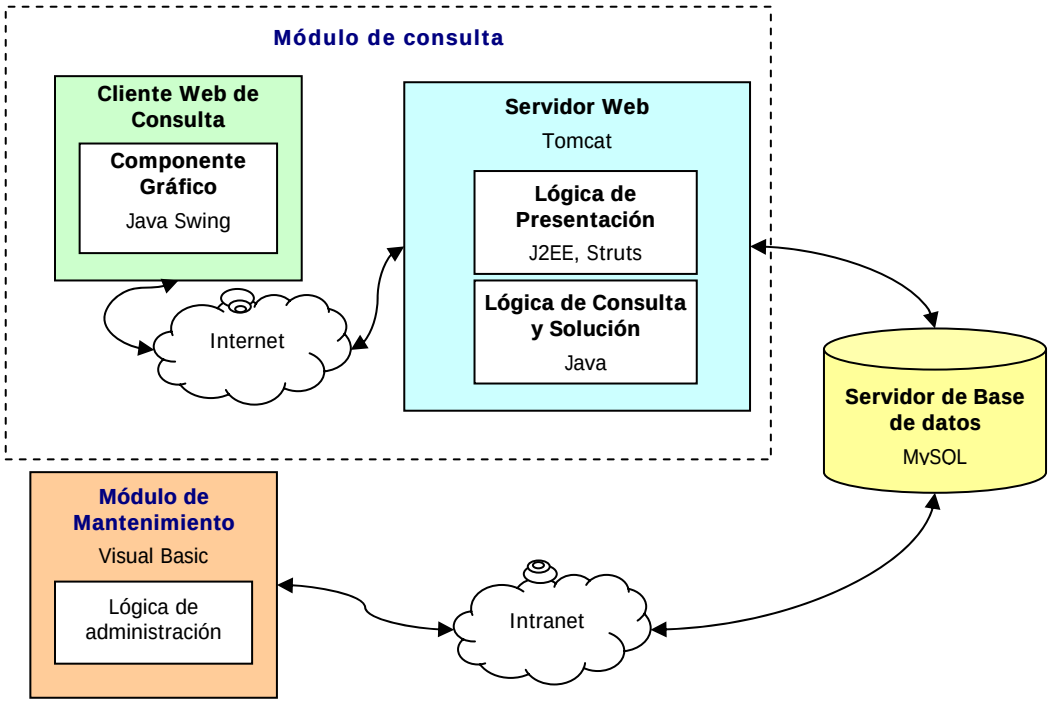


Fig. Componentes del Sistema

Módulo de Mantenimiento

El Módulo de Mantenimiento es una aplicación Visual Basic 6.

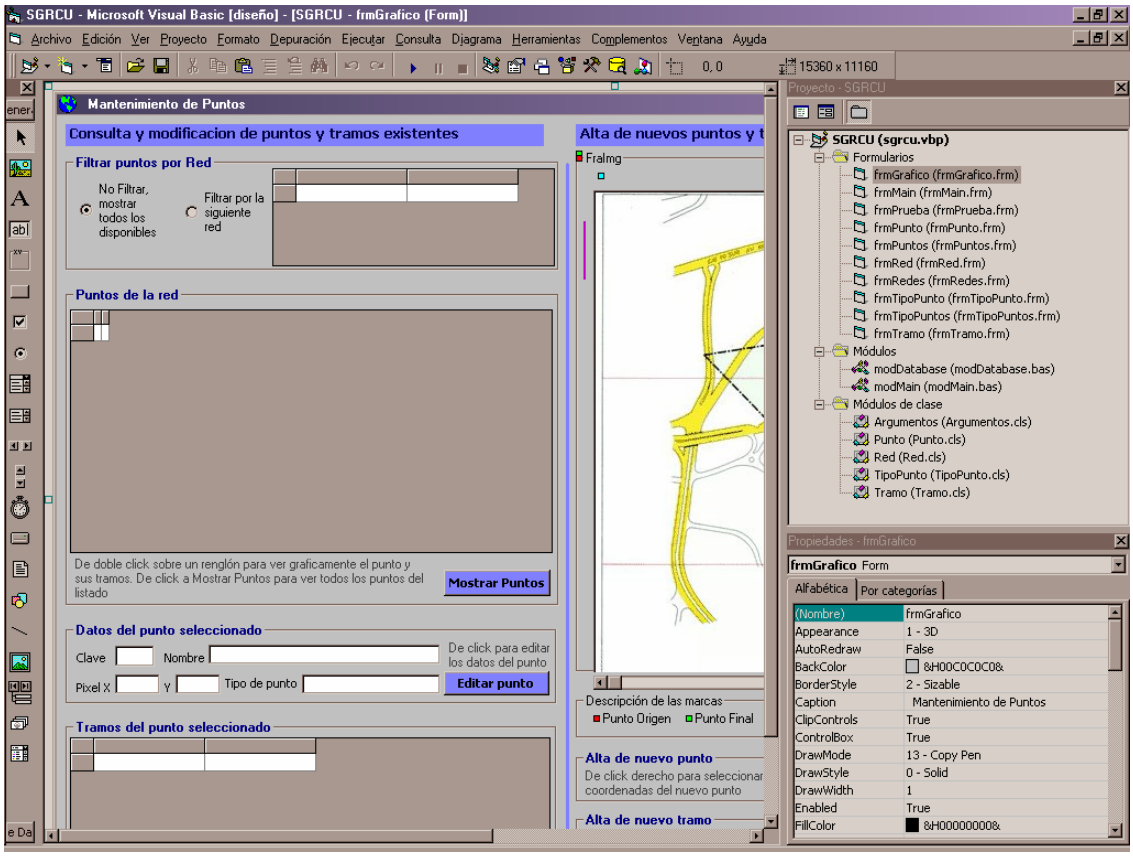


Fig. Desarrollo del Módulo de Mantenimiento en el editor Visual Basic 6.0

Componentes del Módulo de Mantenimiento

Componente	Tipo	Descripción
sgrcu.vbp	Proyecto VB	Es el archivo de proyecto Visual Basic.
frmGrafico.frm	Formulario	Es la pantalla de mantenimiento de puntos y tramos de forma gráfica.
frmMain.frm	Formulario	Es la pantalla principal, desde aquí se tiene acceso a las pantallas de mantenimiento de catálogos de Punto, Red, Tipo de Punto y Tramo.
frmPunto.frm	Formulario	Pantalla de captura de los datos de un Punto.
frmPuntos.frm	Formulario	Pantalla con el listado de todos los puntos registrados.
frmRed.frm	Formulario	Pantalla de captura de datos de una Red.
frmRedes.frm	Formulario	Pantalla con el listado de redes registradas.
frmTipoPunto.frm	Formulario	Pantalla de captura de datos de un Tipo de Punto.
frmTipoPuntos.frm	Formulario	Pantalla con el listado de Tipos de Punto registrados.
frmTramo.frm	Formulario	Pantalla de captura de datos de un Tramo, es accesible solo desde

		la pantalla gráfica.
frmLogin.frm	Formulario	Pantalla de acceso al módulo.
modDatabase.mod	Módulo VB	Módulo VB con las funciones de conexión y desconexión a la base de datos.
modMain.mod	Módulo VB	Módulo VB con las variables generales de la aplicación.
Red.cls	Módulo de Clase	Clase en VB con la funcionalidad de DAO y DTO para los datos de la Red.
Punto.cls	Módulo de Clase	Clase en VB con la funcionalidad de DAO y DTO para los datos del Punto.
TipoPunto.cls	Módulo de Clase	Clase en VB con la funcionalidad de DAO y DTO para los datos del Tipo de Punto.
Tramo.cls	Módulo de Clase	Clase en VB con la funcionalidad de DAO y DTO para los datos del Tramo.
cu.jpg	Imagen jpg	Imagen del mapa de CU(1400x1500) que se carga desde frmGrafico.

Cuidados seguidos en la programación con Visual Basic.

Para evitar caer en las malas prácticas de la programación con Visual Basic se hizo lo siguiente:

- Se crearon clases para los objetos del modelo de negocio: Punto, Red, TipoPunto, Tramo. Estas clases por si solas funcionan como beans o DTOs para transferir los datos entre ventanas o subrutinas. Y así mismo funcionan como DAOs por que tienen métodos para ejecutar las operaciones de alta, baja, cambio y selección.
- Por tanto en los botones de la aplicación no se dejo lógica de negocio dispersa, los botones hacen llamadas de select, insert, delete, update a las clases.
- La obtención de la conexión es mediante un único método del módulo modDatabase.
- La configuración de valores de la aplicación se hace mediante un único módulo modMain
- No se usan variables automáticas
- No se usan Data Access Objects que bloqueen las tablas, no se usan grids para actualización de datos.

Valores generales usados en el sistema

La configuración general del módulo de mantenimiento se encuentra en el **modMain.mod.** Ahí se debe ubicar nuevos valores para el sistema.

```
Public G_RUTA As String
Public G_OFFSET_X, G_OFFSET_Y As Integer
Public G_FACTOR_ESCALA_PIXEL_X, G_FACTOR_ESCALA_PIXEL_Y As Double

Public Sub main_inicializar()
    'Ruta del proyecto
    G_RUTA = "C:\HOME\Proyecto\codigos\VersionVB\"

    'Llama al procedimiento para inicializar los parametros de la BD
    db_SubInicio

    'Mitad del ancho de un punto grafico
    G_OFFSET_X = 3
    G_OFFSET_Y = 3

    'Pixeles vs scala
    G_FACTOR_ESCALA_PIXEL_X = 2.4
    G_FACTOR_ESCALA_PIXEL_Y = 2.3
End Sub
```

Pantallas del Módulo

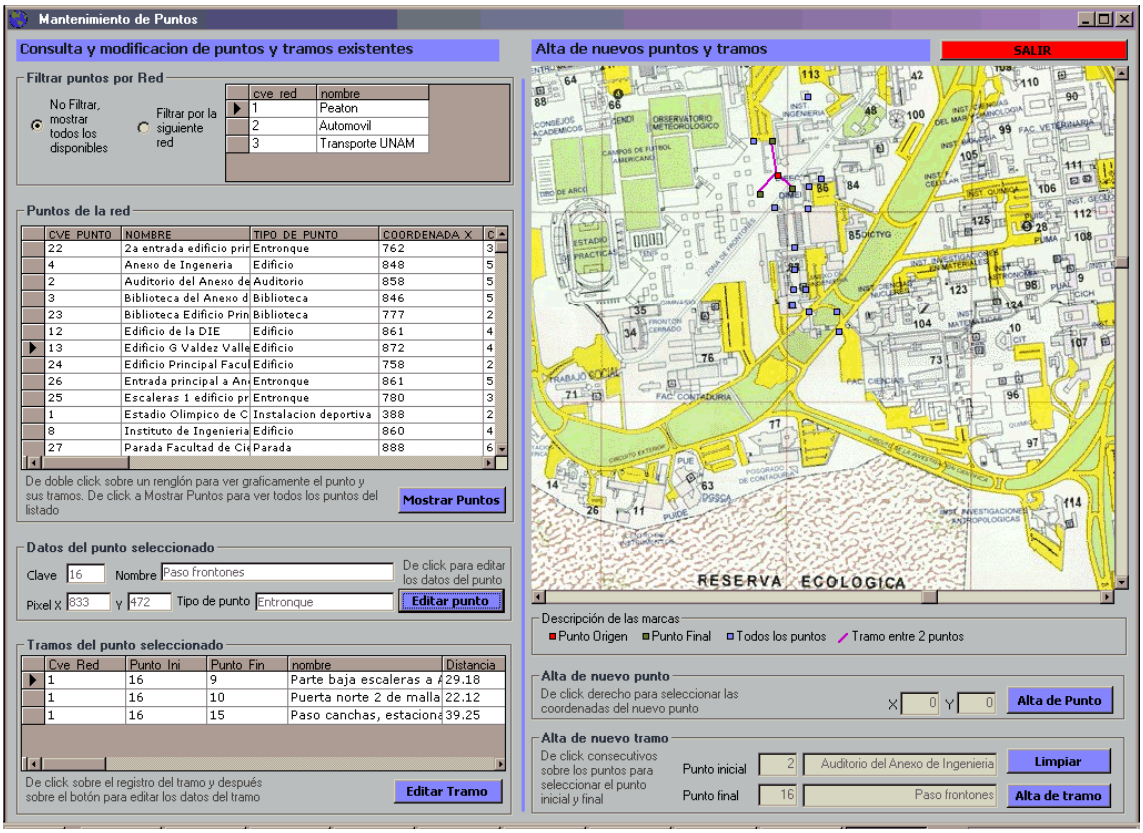
Pantalla de acceso al Módulo de Mantenimiento

Descripción	Es la pantalla de acceso al Módulo de Mantenimiento
Campos	***Usuario*** – Clave de usuario que tiene permiso al módulo ***Contraseña*** – Contraseña
Controles	Botón ***Aceptar*** – Valida la el usuario y contraseña capturados y permite el acceso a la pantalla principal. Botón ***Limpiar*** – Limpia los valores capturados Botón ***Cancelar*** – Termina el programa
Pantallas asociadas	Botón Aceptar – Pantalla Principal (frmMain.frm)
Requerimientos / Casos de uso cubiertos	R4 “Solo un usuario administrador tendrá acceso al sistema”
Notas	---

Pantalla principal

Descripción	Es la pantalla principal de acceso a los catálogos de Red, Tipo de Punto, Puntos, Tramos
Campos	---
Controles	Botón ***Catálogo de Red*** – Despliega la pantalla del listado de Redes registradas (frmRedes.frm) Botón ***Catálogo de Tipos de Puntos*** – Despliega la pantalla del listado de Tipos de Punto registrados (frmTipoPuntos.frm) Botón ***Catálogo de Puntos*** – Despliega la pantalla del listado de Puntos registrados (frmPuntos.frm) Botón ***Mantenimiento gráfico de Puntos y Tramos*** – Despliega la pantalla de alta, baja, modificación y borrado gráfico de tramos y puntos. Botón ***Salir*** – Termina el programa.
Pantallas asociadas	frmRedes.frm, frmTipoPuntos.frm, frmPuntos.frm
Requerimientos / Casos de uso cubiertos	Ninguno, solo es una pantalla de soporte.
Notas	---

Pantalla de Mantenimiento Gráfico de Puntos y Tramos



Descripción	Es la pantalla gráfica de alta y modificación de Puntos y Tramos.
Campos	Campos de ***Datos del punto seleccionado*** – Son campos no editables que muestran los datos del punto seleccionado en el datagrid de Puntos de la Red. Campos de ***Alta de nuevo punto*** – Campos no editables X y Y de las coordenadas en píxeles donde se ha hecho clic sobre el mapa. Campos de ***Alta de nuevo tramo*** – Campos no editables que muestran los datos de los puntos existentes sobre el mapa a los que se ha hecho clic consecutivamente.
Controles	Botón ***Mostrar Puntos***– Visualiza los puntos listados en el datagrid en el mapa de CU Botón ***Editar Punto*** – Despliega la pantalla de captura de los datos del Punto para el punto seleccionado en el datagrid de “Puntos de la Red” Botón ***Editar Tramo*** – Despliega la pantalla de captura de datos del Tramo para el tramo seleccionado en el datagrid de “Tramos del Punto seleccionado” Botón ***Alta de Punto*** – Despliega la ventana de alta de Punto (frmPunto), a esta ventana se le pasan las coordenadas del ultimo clic realizado sobre el mapa de CU. Botón ***Alta de Tramo*** – Despliega la ventana de alta de Tramo (frmTramo), a esta ventana se le pasan las coordenadas de los dos últimos clics realizados sobre dos puntos existentes en el mapa de CU y la red actual seleccionada. Botón ***Limpiar*** – Limpia los valores de los campos Punto Inicial y Punto Final. Botón ***Salir*** – Sale de la pantalla y regresa a la pantalla Principal. RadioButton ***Filtrar puntos por Red*** – Este radiobutton filtra los puntos visualizados en el datagrid “Puntos de la Red” de acuerdo a los puntos asociados a tramos que pertenecen a la red elegida.

	Datagrid ***Puntos de la Red*** – Lista todos los puntos sin asociar y asociados a algún tramo que pertenecen a la red elegida en el RadioButton *Filtrar puntos por Red*. Datagrid ***Tramos del Punto seleccionado*** – Lista los tramos asociados al punto elegido en el datagrid *Puntos de la Red* ***Control Shape*** – Contiene la imagen del mapa de CU de 1400x1500 píxeles. Dentro del Shape hay 4 objetos diferentes que ayudan a indicar los puntos y tramos Label ***pIni*** – Es un control Label de color rojo que indica el punto al que se ha dado clic y es el inicial de los tramos que aparezcan conectados a el. Label ***pfin***() – Es un arreglo de tipo Label de color verde que indica los puntos finales de los tramos asociados al punto origen, Label ***pOtro***() – Es un arreglos de tipo Label de color azul que indica los demás puntos que se han dado de alta y que no están relacionados con el punto inicial seleccionado. Line ***linea***() – Es un arreglo de tipo Line que sirve para conectar el punto origen seleccionado (pIni) y los puntos finales (pFin) de sus tramos. Scrollbars - Los 2 scrollbars se usan para poder desplazar la imagen.
Pantallas asociadas	frmTramo.frm, frmPunto.frm
Requerimientos / Casos de uso cubiertos	Caso de uso 4 y 5, Requerimientos R3, R4, R6, R7
Notas	Las instrucciones de alta de un punto y un tramo están anotadas sobre la pantalla.

Imágenes ejemplo de la operación

Para dar de alta un nuevo punto, se da clic sobre el mapa para que el programa detecte las coordenadas del punto donde se hizo clic, después se da clic en el botón Alta de Punto y se capturan los datos.

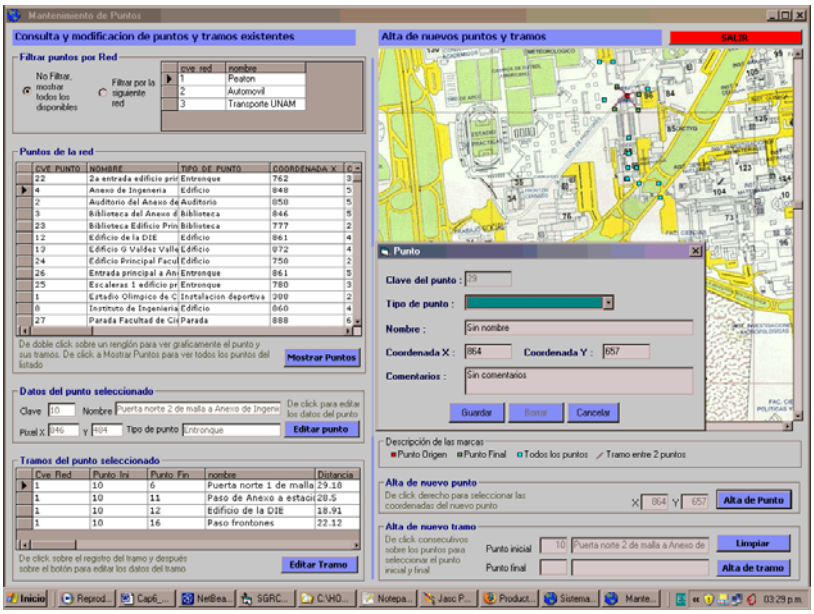


Fig. Alta de un nuevo Punto

Para dar de alta un nuevo tramo se da clic consecutivamente en dos punto existentes que no tengan tramo, el sistema identifica los puntos, el primer clic a un punto es el punto origen el segundo clic indica el punto final. A continuación se da clic al botón de Alta de Tramo. El sistema despliega la ventana de captura de datos del tramo con los puntos elegidos, el sistema calcula la distancia de recorrido haciendo una regla de tres entre los 2 puntos y calcula el tiempo de recorrido de acuerdo a la velocidad promedio registrada para la red.

Opcionalmente y por default puede darse de alta el tramo de regreso si aplica, dando clic al checkbox "Mismos datos para el tramo de retorno". Por ejemplo en el caso de los automóviles que circulan por el circuito de CU esta opción no aplica por que sobre los sentidos están separados por camellones.

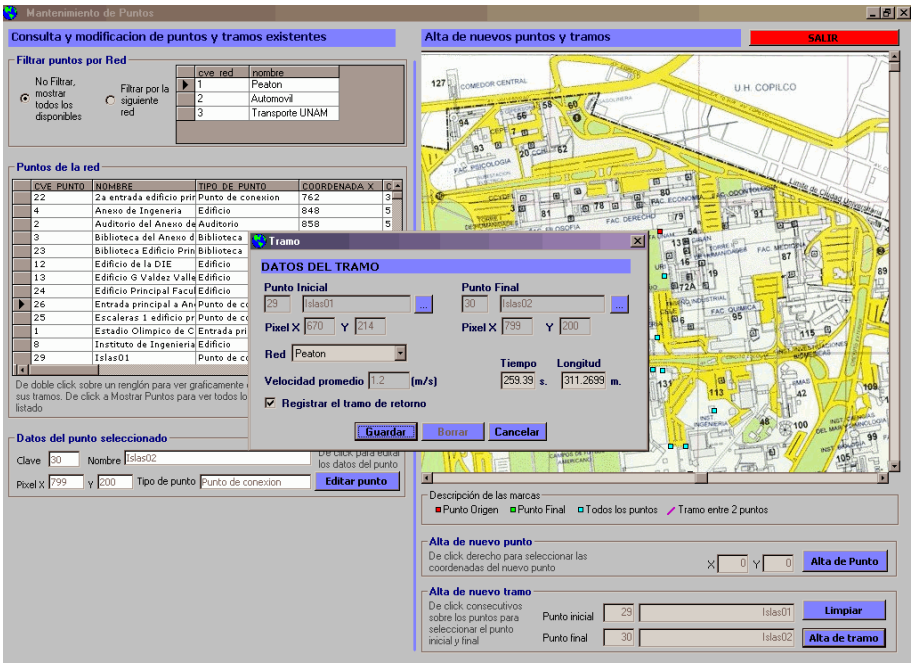
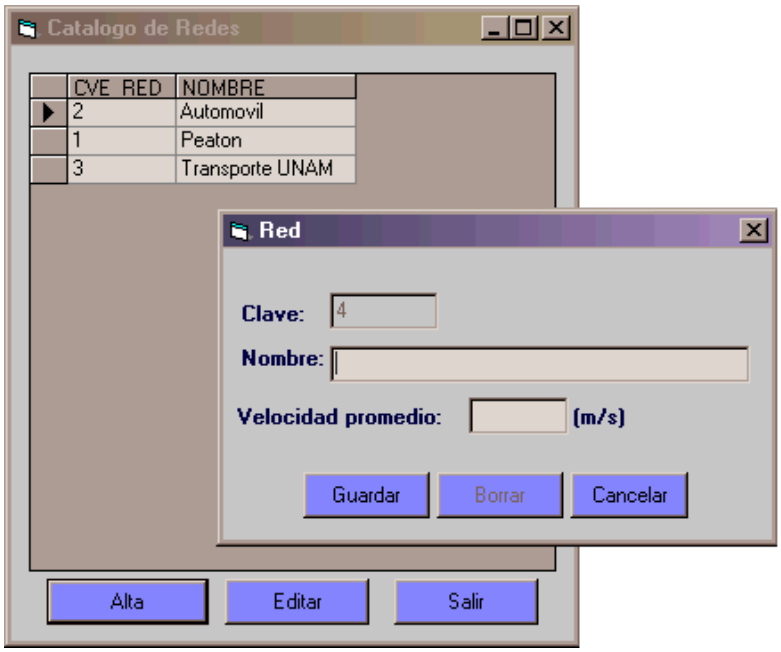


Fig. Alta de un nuevo tramo

Pantallas de Mantenimiento de Redes

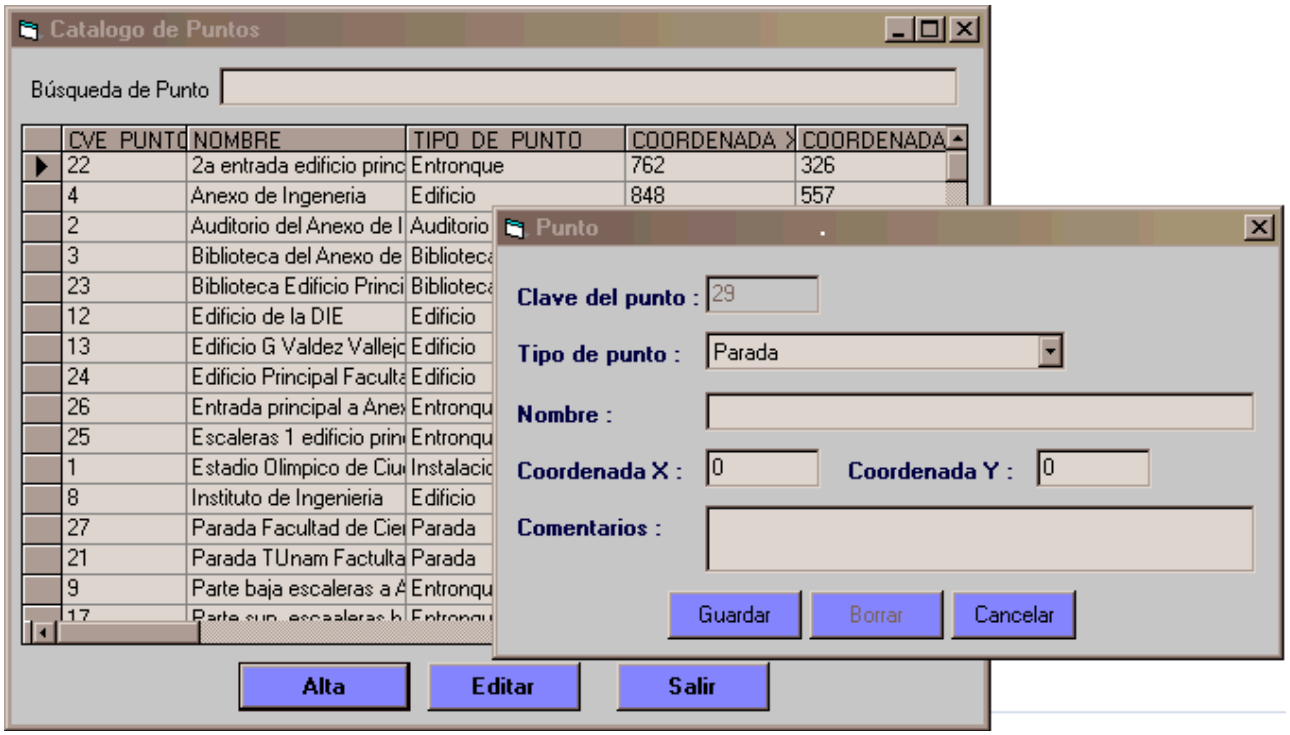


Descripción	Estas dos pantallas permiten el alta, baja modificación y eliminación de una red
Campos	Pantalla ***Catálogo de Redes*** (frmRedes) DataGrid **Redes** – No editable, al seleccionar un registro y dar clic en Alta o en Editar se despliega la ventana de captura de los datos de la Red (frmRed) Pantalla ***Red*** (frmRed) Clave: Se muestra la clave de red asignada por el sistema Nombre: Aquí se captura el nombre de la Red Velocidad Promedio (m/s): Se captura la velocidad promedio de recorrido, para hacer una estimación del tiempo de recorrido de un tramo en base a su longitud.
Controles	Pantalla ***Catálogo de Redes*** (frmRedes) Botón **Alta** – Despliega la ventana de captura de los datos de la Red (frmRed) para la captura de una nueva Red. Botón **Editar** - Despliega la ventana de captura de los datos de la Red (frmRed), para la modificación del registro elegido en el datagrid. Botón **Salir** – Cierra la pantalla Pantalla ***Red*** (frmRed) Botón ***Guardar*** – Guarda los datos de la red nueva o modificada Botón ***Borrar*** – Elimina la red existente Botón ***Cancelar*** – Cancela la operación.
Pantallas asociadas	frmRed.frm
Requerimientos / Casos de uso cubiertos	Caso de uso 2, Requerimientos R3 y R4
Notas	---

Pantallas de Mantenimiento de Tipos de Punto

Descripción	Estas dos pantallas permiten el alta, baja modificación y eliminación de un Tipo de Punto
Campos	Pantalla ***Catálogo de Tipos de Punto*** (frmTipoPuntos) DataGrid **Tipos de Punto** – No editable, al seleccionar un registro y dar clic en Alta o en Editar se despliega la ventana de captura de los datos del Tipo de Punto (frmTipoPunto) Pantalla ***Tipo de Punto*** (frmTipoPunto) Clave: Se muestra la clave del tipo de Punto asignada por el sistema Nombre: Aquí se captura el nombre del Tipo de Punto Velocidad Promedio (m/s): Se captura la velocidad promedio de recorrido, para hacer una estimación del tiempo de recorrido de un tramo en base a su longitud.
Controles	Pantalla ***Catálogo de Tipos de Punto*** (frmTipoPuntos) Botón **Alta** – Despliega la ventana de captura de los datos del Tipo de Punto (frmTipoPunto) para la captura de un nuevo Tipo de Punto. Botón **Editar** - Despliega la ventana de captura de los datos del Tipo de Punto (frmTipoPunto), para la modificación del registro elegido en el datagrid. Botón **Salir** – Cierra la pantalla Pantalla ***Tipo de Punto*** (frmTipoPunto) Botón ***Guardar*** – Guarda los datos del Tipo de Punto nuevo o modificado Botón ***Borrar*** – Elimina el Tipo de Punto existente y si no tiene información dependiente. Botón ***Cancelar*** – Cancela la operación.
Pantallas asociadas	frmTipoPunto.frm
Requerimientos / Casos de uso cubiertos	Caso de uso 3, Requerimientos R3, R4 y R6
Notas	---

Pantallas de Mantenimiento de Puntos



Descripción	Estas dos pantallas permiten el alta, baja modificación y eliminación de un Punto. La manipulación por estas pantallas de la información de un punto es menos amigable ya que hay que dar manualmente las coordenadas del punto en las cajas de texto. Debe usarse de preferencia para listar los puntos y hacer búsquedas por nombre.
Campos	Pantalla ***Catálogo de Puntos*** (frmPuntos) Campo Búsqueda: Se usa para encontrar por nombre el nombre de los Puntos DataGrid **Puntos** – No editable, al seleccionar un registro y dar clic en Alta o en Editar se despliega la ventana de captura de los datos del Punto (frmPunto) Pantalla ***Punto*** (frmPunto) Clave: Se muestra la clave del Punto asignada por el sistema Tipo de punto: Se elige el tipo de punto al que corresponde el punto Nombre: Aquí se captura el nombre del Punto Coordenada X: Es la coordenada X en píxeles en la imagen del mapa de CU. Coordenada Y: Es la coordenada Y en píxeles en la imagen del mapa de CU. Comentarios: Se anota información adicional del punto, esta no aparece en la ruta de solución pero se crea para un uso general.
Controles	Pantalla ***Catálogo de Puntos*** (frmPuntos) Botón **Alta** – Despliega la ventana de captura de los datos del Punto (frmPunto) para la captura de un nuevo Punto. Botón **Editar** - Despliega la ventana de captura de los datos del Punto (frmPunto), para la modificación del registro elegido en el datagrid. Botón **Salir** – Cierra la pantalla

	Pantalla ***Punto*** (frmPunto) Botón ***Guardar*** – Guarda los datos del Punto nuevo o modificado Botón ***Borrar*** – Elimina el Punto existente y si no tiene información dependiente. Botón ***Cancelar*** – Cancela la operación.
Pantallas asociadas	frmPunto.frm
Requerimientos / Casos de uso cubiertos	Caso de uso 4, da soporte al requerimiento R4
Notas	---

Pantalla de Alta y Modificación de Tramos

Descripción	Esta pantalla es accesible solo desde la pantalla grafica de Puntos y Tramos, sirve para dar de alta y modificar los datos del tramo.
Campos	***Punto inicial***: Datos del punto: Se muestran campos no editables de la clave y el nombre del punto que es inicial Píxel X, Y: Campos no editables de las coordenadas X, Y del píxel que tiene el punto en el mapa. ***Punto final***: Datos del punto: Se muestran campos no editables de la clave y el nombre del punto que es inicial Píxel X, Y: Campos no editables de las coordenadas X, Y del píxel que tiene el punto en el mapa. Campo ***Red***: Catálogo de las redes disponibles a la que pertenece el tramo Campo ***Velocidad Promedio***, No editable, es el la velocidad promedio asociada a la red elegida. Cuando se cambia la red, se cambia este valor y se recalcula el tiempo de recorrido de la longitud que es fija. Campo ***Tiempo***: Tiempo de recorrido del tramo en segundos. Campo ***Longitud***: Longitud del tramo en metros. Checkbox "***Registrar el tramo de retorno***" Con este checkbox se puede dar de alta el tramo de retorno con los mismos datos, solo con los puntos inicial y final invertidos.
Controles	Botón ***Guardar*** – Guarda los datos del tramo nuevo o modificado Botón ***Borrar*** – Elimina el tramo existente Botón ***Cancelar*** – Cancela la operación.
Pantallas asociadas	frmRed.frm
Requerimientos / Casos de uso cubiertos	Caso de uso 5, Requerimientos R3, R4, R6, R7
Notas	---

Código importante en esta pantalla

La función ***funCalculaDistancia*** calcula el valor de la distancia aproximada real en base a los píxeles y la escala del gráfico. El valor calculado se coloca en el campo de distancia y es sugerido, puede sobrescribirse.

```
Private Function funCalculaDistancia(ByVal pX1 As Integer, ByVal pY1
As Integer, ByVal pX2 As Integer, ByVal pY2 As Integer) As Double
    'Calcula la distancia aproximada real en base a la distancia entre
pixeles
     'Se usa la formula básica de calculo de distancias entre puntos.
Los factores
     'de conversion G_FACTOR_ESCALA_PIXEL_X son los valores de
relación entre
     'la escala del mapa y los pixeles.
    'Hay que recordar que los pixeles no son cuadrados, son
rectangulos con
     'una relación de 0.8 por ello hay 2 factores
    Dim x1x2, y1y2 As Double
    Dim distancia As Double

    x1x2 = (pX1 - pX2) * G_FACTOR_ESCALA_PIXEL_X
    x1x2 = x1x2 * x1x2
    y1y2 = (pY1 - pY2) * G_FACTOR_ESCALA_PIXEL_Y
    y1y2 = y1y2 * y1y2
    distancia = Sqr(x1x2 + y1y2)

    funCalculaDistancia = distancia
End Function
```

Módulo de Consulta

El módulo de consulta es una aplicación con componentes Web de J2EE, Consta de 2 proyectos:

Proyecto java ***sgrcuWebApp*** - Aplicación Web de Consulta, empaquetado en sgrcu.war

- Se compone de clases java, librerías de Struts y de utilidad.
- Las paginas son jsp que usan imágenes, hojas de estilos y scripts de validación

Proyecto java ***sgrcuApplet*** - Applet del objeto gráfico, empaquetado en sgrcu-applet.jar

- Clases java y la imagen del mapa de cu main.jpg

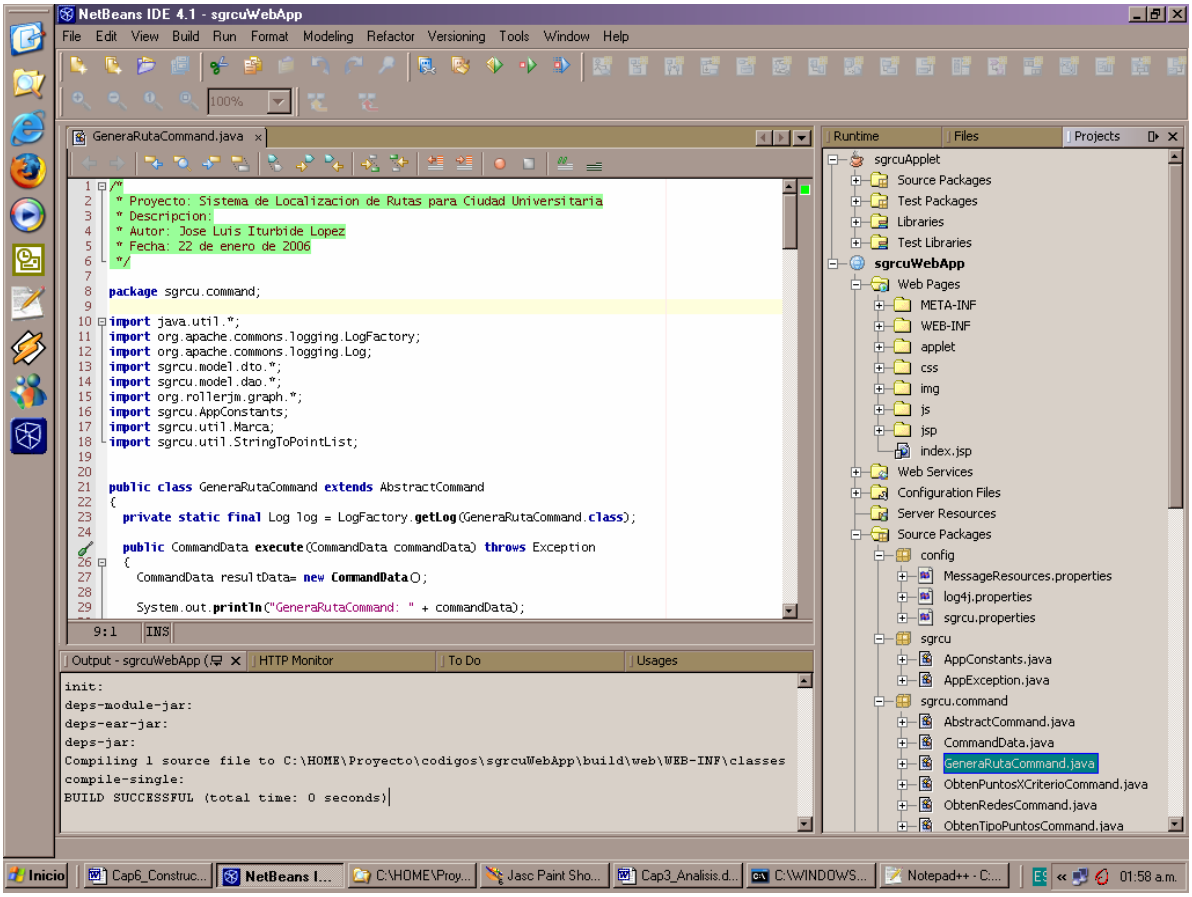


Fig. Vista del desarrollo de la aplicación de consulta en el IDE NetBeans 4.1

Componentes del Módulo de Consulta

Aplicación Web de Consulta

Clases java y archivos de configuración

Componente	Descripción
config	package
log4j.properties	Archivo de configuración para el log
MessageResources.properties	Archivo de mensajes para struts
sgrcu.properties	Archivo de configuración de la aplicación, aquí se colocan los valores de la base de datos
sgrcu	Package
AppConstants.java	Valores constantes de la aplicación
AppExeption.java	Clase personalizada para el manejote de excepciones.
command	Package
AbstractCommand.java	Es una clase Abstracta de la que extienden todas las clases que efectúan una tarea. Todos los comandos deben implementar el método execute, ya que este método es el que se invoca por CommandInvoker.
CommandData.java	Es la clase para intercambio de datos entre la capa presentación y de control. En esta clase se establece el identificador del comando a ejecutar y los datos de entrada cuando se invoca un comando; y se crea una nueva instancia para devolver los datos de resultado a la capa de presentación.
GeneraRutaCommand.java	Devuelve una instancia de ResultadorutaDTO (dentro de CommandData) con los datos de solución de la ruta. Los datos de entrada son la red, punto origen y punto destino, dentro de CommandData.
ObtenPuntosXCriterioCommand.java	Obtiene los puntos disponibles origen y destino, los datos de entrada son el tipo de red, tipo de instalación y una cadena filtro para el nombre del punto.
ObtenRedesCommand.java	Obtiene el catálogo de tipos de redes
ObtenTipoPuntosCommand.java	Obtiene el catálogo de tipos de punto
control	package
AppContext.java	Clase para guardar variables que desean recuperarse entre llamadas
BusinessDelegate.java	Esta clase es la única clase de negocio visible desde la capa de presentación. Para hacer la invocación a un comando se usa una instancia de la clase CommandData que tiene los datos necesarios para realizar una transacción y se envía como parámetro a BusinessDelegate, esta a su vez usa CommandInvoker para ejecutar el comando deseado.
CommandInvoker.java	Esta clase es de control, extrae el identificador del comando a ejecutar de la instancia CommandData
model	package
dao	package
BaseDAO.java	Clase Abstracta que contiene funcionalidad común de inicialización, obtención y cerrado de la conexión y define 4 métodos básicos a implementar de select, update, insert y delete.
PuntoDAO.java	Clase de acceso a los datos del Punto.
RedDAO.java	Clase de acceso a los datos de una Red.
TipoPuntoDAO.java	Clase de acceso a datos del Tipo de Punto.
TramoDAO.java	Clase de acceso a datos del Tramo
dto	package
BaseDTO.java	Clase abstracta de transferencia de datos, tiene un solo método toString para imprimir el contenido de los valores del bean.
PuntoDTO.java	Clase para guardar temporalmente los datos de un Punto.
RedDTO.java	Clase para guardar temporalmente los datos de una Red.
ResultadoRutaDTO.java	Clase para guardar temporalmente los datos del resultado de la Ruta. Esta es una de las clases más importantes por que almacena los datos del resultado del proceso de búsqueda de la solución.

TipoPuntoDTO.java	Clase para guardar temporalmente los datos de un Tipo de Punto.
TramoDTO.java	Clase para guardar temporalmente los datos de un Tramo.
service	package
Conexion.java	Clase que devuelve un objeto Connection a las clases DAO.
util	package
ConfigProperties.java	Clase que recupera valores del archivo de configuración del sistema.
AppletProperties.java	Clase que recupera valores del archivo de configuración del applet.
Marca.java	Clase da utilería para guardar los datos de la marca a dibujar sobre el mapa, coordenada x, y y nombre.
SimpleWhereConstructor.java	Clase utilizada por las clases DAO para armar una sentencia sql con un where.
StringToPointList.java	Clase de utilería para convertir la cadena de puntos a un listado de clases Marca.
Utils	Funciones varias de utilería
Web	package
action	package
BaseAction.java	Clase Abtracta que extiende de Action para información común
InicioConsultaAction.java	Clase Action del Framework de struts que hace el llamado a todas las operaciones de consulta y generación de la ruta
forms ConsultaForm	Contiene los parámetros de punto origen, destino, listas de catálogos de red y tipos de punto de la página de consulta.

Archivos Web

Componente	Descripción
ROOT	
index.jpg	Pagina inicial de presentación
css estilos.css	Directorio para las hojas de estilo
img	Directorio para las imágenes
js valida.js	Directorio para los archivos javascript de validación
applet sgrcu-applet.jar	Archivo donde se deja el jar del applet, esta a este nivel por facilidad.
jsp	Archivo de las paginas jsp
InicioConsulta.jsp	Pagina de entrada de parámetros de consulta: punto inicial, final y red.
ResultadoConsulta.jsp	Pagina que muestra el resultado de la ruta generada
error.jsp	Pagina de apoyo para mostrar errores en la aplicación
WEB-INF	Directorio de recursos privados
struts-html.tld, validation.xml, struts-logic.tld, validator-rules.xml struts-nested.tld, Web.xml, struts-bean.tld, struts-tiles.tld struts-config.xml, tiles-defs.xml	Archivos de configuración de Struts y archivo web.xml
lib dijkstra-1.0.1.jar mysql-connector-java-3.1.6-bin.jar struts.jar, standard-1.1.2.jar jstl-1.1.2.jar, jakarta-oro.jar commons-beanutils.jar commons-digester.jar commons-lang-2.1.0.jar commons-validator.jar	Directorio para librerías usadas por la aplicación. Librerías de Struts, de las clases Dijkstra, del driver jdbc de Mysql y de utilería.
classes	Clases compiladas

Archivo de configuración de la aplicación sgrcu.properties

```
# Archivo de configuración de la aplicación
db.name=sgrcu
db.user=sgrcu
db.password=sgrcu
db.url=jdbc:mysql://localhost:3306/sgrcu
db.driver=com.mysql.jdbc.Driver
```

Código de la clase Command GeneraRutaCommand de solución a la ruta

Se remarcan con negritas los comentarios sobre el código de solución de la Ruta

```
/*
 * Proyecto: Sistema de Localización de Rutas para Ciudad Universitaria
 * Descripción: Clase que genera la solución de la ruta
 * Autor: José Luis Iturbide López
 * Fecha: 22 de enero de 2006
 */

package sgrcu.command;

import java.util.*;
import org.apache.commons.logging.LogFactory;
import org.apache.commons.logging.Log;
import sgrcu.model.dto.*;
import sgrcu.model.dao.*;
import org.rollerjm.graph.*;
import sgrcu.AppConstants;
import sgrcu.util.Marca;
import sgrcu.util.StringToPointList;


public class GeneraRutaCommand extends AbstractCommand
{
  private static final Log log = LogFactory.getLog(GeneraRutaCommand.class);

  public CommandData execute(CommandData commandData) throws Exception
  {
    CommandData resultData= new CommandData();

    System.out.println("GeneraRutaCommand: " + commandData);

    // Inicio
    double distanciaTotal=0;
    double tiempoTotal=0;
```

```
    // Recupera los parámetros de Punto Origen, Punto Destino y Red
    int argPuntoIni =
((Integer)commandData.get(AppConstants.PARAM_PUNTO_INICIO)).intValue();
    int argPuntoFin =
((Integer)commandData.get(AppConstants.PARAM_PUNTO_FIN)).intValue();
    int argRed =
((Integer)commandData.get(AppConstants.PARAM_RED)).intValue();

    String strCvePuntoIni=String.valueOf(argPuntoIni);
    String strCvePuntoFin=String.valueOf(argPuntoFin);

    String result="";
    try
      {
        // Inicializa los DAOs de donde se recuperaran los datos
        TramoDAO tramoDAO = new TramoDAO();
        PuntoDAO puntoDAO = new PuntoDAO();

        PuntoDTO puntoDTO = new PuntoDTO();
        puntoDTO.setCvePunto(argPuntoIni);
        PuntoDTO puntoOrigen = (PuntoDTO) puntoDAO.doSelectById(puntoDTO);
        puntoDTO.setCvePunto(argPuntoFin);
        PuntoDTO puntoDestino = (PuntoDTO) puntoDAO.doSelectById(puntoDTO);

        //1. OBTENCIÓN DE LOS PUNTOS DE LA RED
        //Se consultan todos los tramos de la red especificada en el argumento
        TramoDTO tramoDTO = new TramoDTO();
        RedDTO redDTO = new RedDTO();
        redDTO.setCveRed(argRed);
        tramoDTO.setRed(redDTO);
        Collection alTramos = tramoDAO.doSelect(tramoDTO);

        //Obtiene el número de vértices únicos
        HashSet hsPuntosUnicos = new HashSet();
        Iterator iterPunto = alTramos.iterator();
        TramoDTO tmpTramo;
        PuntoDTO tmpPunto;
        while(iterPunto.hasNext())
        {
          tmpTramo = (TramoDTO) iterPunto.next();
          tmpPunto = tmpTramo.getPuntoIni();
          hsPuntosUnicos.add(tmpPunto);
          tmpPunto = tmpTramo.getPuntoFin();
          hsPuntosUnicos.add(tmpPunto);
```

```
        }

        //2. CONFIGURACIÓN DE LA RED
        //Agrega los puntos a la matriz de adyacencia (graph)
        AdjacencyMatrixGraph graph = new
AdjacencyMatrixGraph(hsPuntosUnicos.size());

        iterPunto = hsPuntosUnicos.iterator();
        while(iterPunto.hasNext())
        {
          tmpPunto = (PuntoDTO) iterPunto.next();
          graph.addVertex(String.valueOf(tmpPunto.getCvePunto()));
        }

        //Agrega los tramos a la matriz
        Iterator iterTramo = alTramos.iterator();
        while(iterTramo.hasNext())
        {
          tmpTramo = (TramoDTO) iterTramo.next();
          if(((int) tmpTramo.getDistancia()) > 0)
          {
          graph.addEdge(String.valueOf( tmpTramo.getPuntoIni().getCvePunto()
),
                        String.valueOf( tmpTramo.getPuntoFin().getCvePunto()
),
                        (int) tmpTramo.getDistancia());
          }
        }
        //Inicializa el objeto de solución con el grafo creado
        Dijkstra dijkstra = new Dijkstra(graph);

        //3. RECUPERA LOS DATOS DETALLADOS DE LA SOLUCIÓN
        //Obtiene la ruta mas corta desde el objeto de solución pasandole
        // los puntos origen y destino
        Path path = dijkstra.getShortestPath(strCvePuntoIni, strCvePuntoFin);
        // Cadena de solución del tipo 1-2-3-4-5-6
        result = path.toString();
        // Recupera la distancia total de la ruta
        int distancia = path.getLength();

        String key1=null;
        String key2=null;
        List tramosSolucion= new ArrayList();
        Collection alTramosQry;

        // Estos DAOs se usaran para recuperar los datos de los tramos de la
```

```
        //  solución
        /* Llenar la solución *************************/
        TramoDTO tramoIterDTO = new TramoDTO();
        RedDTO redIterDTO = new RedDTO();
        PuntoDTO pIniIterDTO = new PuntoDTO();
        PuntoDTO pFinIterDTO = new PuntoDTO();
        TramoDTO resTramoDTO = null;

        // Estas 2 clases se usaran para generar la cadena de puntos que
        //  se enviara al applet
        StringToPointList p = new StringToPointList();
        Marca m = null;

        // Por cada par de puntos de la cadena de la solución se recupera
        //  el tramo correspondiente
        for(int k=1; k < path.getLength() ; k++)
        {
            key1 = (String) path.get(k);
            key2 = (String) path.get(k+1);

            tramoIterDTO = new TramoDTO();
            redIterDTO = new RedDTO();
            pIniIterDTO = new PuntoDTO();
            pFinIterDTO = new PuntoDTO();

            //Consulta el tramo
            pIniIterDTO.setCvePunto(Integer.parseInt(key1));
            pFinIterDTO.setCvePunto(Integer.parseInt(key2));
            tramoIterDTO.setPuntoIni(pIniIterDTO);
            tramoIterDTO.setPuntoFin(pFinIterDTO);
            redIterDTO.setCveRed(argRed);
            tramoIterDTO.setRed(redIterDTO);

            alTramosQry = tramoDAO.doSelect(tramoIterDTO);
            resTramoDTO = (TramoDTO) ((List)alTramosQry).get(0);
            tramosSolucion.add(resTramoDTO);

            // Va sumando la distancia y tiempo total
            distanciaTotal += resTramoDTO.getDistancia();
            tiempoTotal += resTramoDTO.getTiempo();

            //Crea una nueva Marca por cada  punto para usarse en el applet
            m = new Marca();
            m.setClave(resTramoDTO.getPuntoIni().getCvePunto());
            m.setNombre(resTramoDTO.getPuntoIni().getNombre());
```

```
            m.setX((int)resTramoDTO.getPuntoIni().getCoordenadaX());
            m.setY((int)resTramoDTO.getPuntoIni().getCoordenadaY());
            p.addMarca(m);

            //Si es el último tramo agrega la Marca del punto final
            if(k+1 == path.getLength())
            {
                m = new Marca();
                m.setClave(resTramoDTO.getPuntoFin().getCvePunto());
                m.setNombre(resTramoDTO.getPuntoFin().getNombre());
                m.setX((int)resTramoDTO.getPuntoFin().getCoordenadaX());
                m.setY((int)resTramoDTO.getPuntoFin().getCoordenadaY());
                p.addMarca(m);
            }
        }

        // Calcula el tiempo de recorrido en horas, minutos y segundos a
        //  partir de la suma total de segundos
        Calendar cal = Calendar.getInstance();
        int hora = 0; int minuto = 0; int segundo = 0;
        double tmpTiempoTotal = tiempoTotal;
        hora = (int) tmpTiempoTotal / 3600;
        tmpTiempoTotal = (int)tmpTiempoTotal- hora*3600;
        minuto = (int) tmpTiempoTotal / 60;
        tmpTiempoTotal = tmpTiempoTotal- minuto*60;
        segundo = (int)tmpTiempoTotal;

        cal.set(Calendar.HOUR, (int)hora);
        cal.set(Calendar.MINUTE, (int)minuto);
        cal.set(Calendar.SECOND, (int)segundo);


        /* Llenar los datos de la solución en el DTO */
        ResultadoRutaDTO datosRuta = new ResultadoRutaDTO();
        datosRuta.setPuntoOrigen(puntoOrigen);
        datosRuta.setPuntoDestino(puntoDestino);
        datosRuta.setTramosSolucion(tramosSolucion);
        datosRuta.setDistanciaTotal(distanciaTotal);
        datosRuta.setTiempoTotal(tiempoTotal);
        datosRuta.setStrRuta(p.toString());
        datosRuta.setTiempoTotalHMS(cal);

    }
    catch(Exception e)
    {
```

```
        e.printStackTrace();
      }
    // Retorna los resultados a la capa de presentación
    resultData.set(AppConstants.KEY_RUTA_GENERADA, datosRuta);
    return resultData;
  }
}
```

Archivo de configuración Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Web-app                                          xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/Web-app_2_4.xsd" version="2.4">

  <display-name>Sistema SGRCU</display-name>

  <!-- Standard Action Servlet Configuration (with debugging) -->
  <servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
    <init-param>
      <param-name>config</param-name>
      <param-value>/WEB-INF/struts-config.xml</param-value>
    </init-param>
    <init-param>
      <param-name>debug</param-name>
      <param-value>2</param-value>
    </init-param>
    <init-param>
      <param-name>detail</param-name>
      <param-value>2</param-value>
    </init-param>
    <init-param>
      <param-name>application</param-name>
      <param-value>/config/MessageResources</param-value>
    </init-param>
    <load-on-startup>2</load-on-startup>
  </servlet>


  <!-- Standard Action Servlet Mapping -->
  <servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.do</url-pattern>
  </servlet-mapping>
```

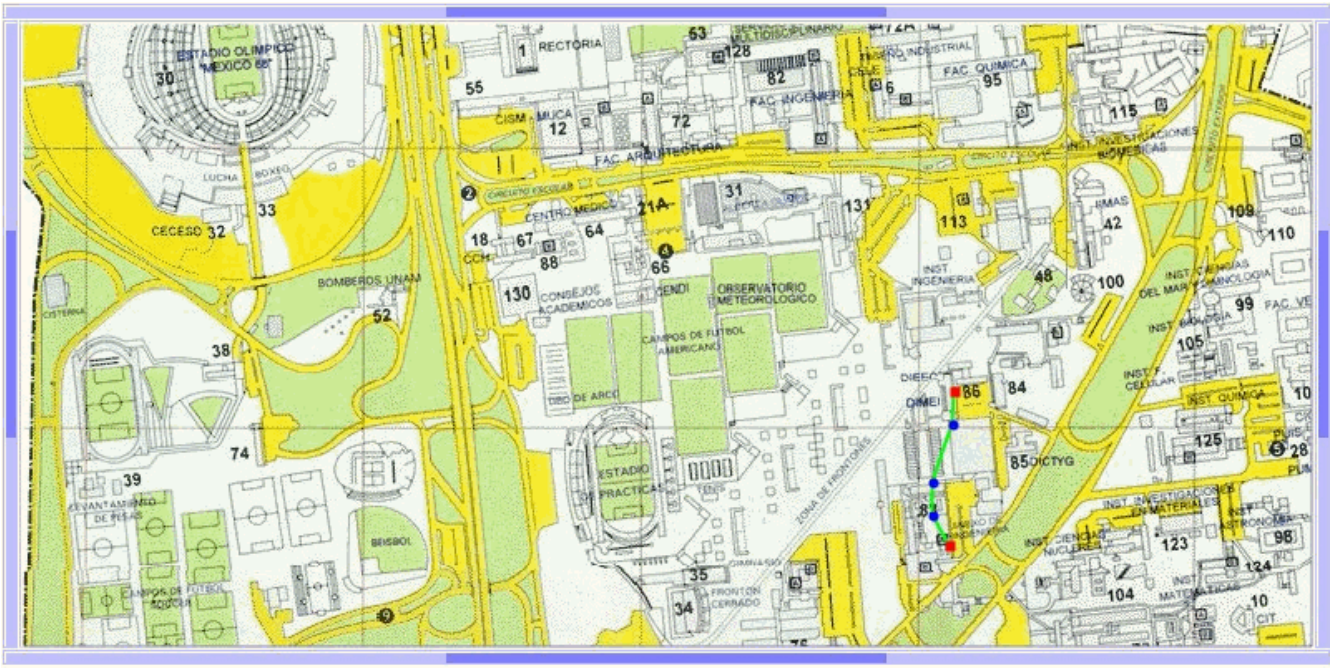
```
    <session-config>
        <session-timeout>
            30
        </session-timeout>
    </session-config>

    <welcome-file-list>
        <welcome-file>
            index.jsp
        </welcome-file>
    </welcome-file-list>

</Web-app>
```

Archivo de configuración struts-config.xml

```
<?xml version="1.0" encoding="ISO-8859-1" ?>

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_2.dtd">

<struts-config>
<!-- ============================================== Form Bean Definitions -->

    <form-beans>
        <form-bean
            name="consultaForm"
            type="sgrcu.Web.forms.ConsultaForm" />
    </form-beans>

<!-- ========================================== Global Exception Definitions -->
    <global-exceptions>
    </global-exceptions>

<!-- ========================================== Global Forward Definitions -->
    <global-forwards>
    </global-forwards>

<!-- ========================================== Action Mapping Definitions -->
    <action-mappings>
            <!-- Default "Welcome" action -->
            <!-- Forwards to Welcome.jsp -->
        <action
            path="/InicioConsulta"
            type="sgrcu.Web.action.InicioConsultaAction"
```

```
            name="consultaForm"
            scope="request"
            validate="false"
            input="/jsp/InicioConsulta.jsp">
            <forward name="FORWARD_SUCCESS" path="/jsp/InicioConsulta.jsp"/>
            <forward name="FORWARD_EXECUTE" path="/jsp/ResultadoConsulta.jsp"/>
            <forward name="FORWARD_ERROR" path="/jsp/error.jsp"/>
        </action>
    </action-mappings>


<!-- ========================================== Controller Configuration -->
    <controller
       processorClass="org.apache.struts.tiles.TilesRequestProcessor"/>

<!-- ======================================== Message Resources Definitions -->
    <message-resources parameter="config.MessageResources" />

<!-- ============================================== Plug Ins Configuration -->
    <plug-in className="org.apache.struts.tiles.TilesPlugin" >

      <!-- Path to XML definition file -->
      <set-property property="definitions-config"
                      value="/WEB-INF/tiles-defs.xml" />
      <!-- Set Module-awareness to true -->
      <set-property property="moduleAware" value="true" />
    </plug-in>

  <!-- ================================================== Validator plugin -->
  <plug-in className="org.apache.struts.validator.ValidatorPlugIn">
    <set-property
        property="pathnames"
        value="/WEB-INF/validator-rules.xml,/WEB-INF/validation.xml"/>
  </plug-in>

</struts-config>
```

Applet



Descripción del applet

La mini-aplicación que muestra un mapa navegable de CU es un Applet Swing.

Este Applet usa un objeto Panel para cargar la imagen del mapa y se dibuja sobre el, los puntos y tramos de la ruta. Cada vez que se desplaza la parte visible a los lados o abajo y arriba, los puntos y tramos deben volverse a dibujar. Por tanto los datos de las coordenadas y tipo de punto se deben guardar dentro del applet al inicializarse.

Cuando se muestra la página de resultado al usuario, la pagina pasa un parámetro de tipo cadena al applet con la secuencia de datos de cada uno de los puntos, nombre, coordenada x, y y nombre del punto. El applet al recibir la cadena la decodifica para saber cuantos puntos deben pintarse.

Por la forma en que funcionan los applets este debe empaquetarse en un archivo jar y dejarlo por facilidad en un subdirectorio debajo de la página que lo llama.

Código en el navegador que llama al applet.

```
<applet code="sgrcu.Web.applet.SolucionApplet.class"
  codebase="applet/"
  archive="sgrcu-applet.jar"
  alt="No pudo cargarse el applet"
  align="center" width="800" height="400">
    <param name="paramCadPuntos" value="${resultadoRutaDTO.strRuta}">
</applet>
```

resultadoRutaDTO.strRuta – Es una variable de la instancia de la clase ResultadoRutaDTO en el request.

Clases java y archivos de configuración del applet

Componente	Descripción
config	
applet.properties	Archivo de configuración para el applet
sgrcu	
util	
ConfigData.java	Clase de utilería para guardar los datos de configuración del applet.
AppletProperties.java	Clase que recupera valores del archivo de configuración del applet.
Marca.java	Clase da utilería para guardar los datos de la marca a dibujar sobre el mapa, coordenada x, y y nombre.
StringToPointList.java	Clase de utilería para convertir la cadena de puntos a un listado de clases Marca.
Web	
applet	
GraphicPanel.java	Clase que extiende de java.awt.GraphicPanel que es el contenedor de la imagen del mapa.
SolucionApplet.java	Clase Applet que contiene todos los objetos que muestran la solución Gráfica.
images	
main.jpg	Imagen jpg del mapa de CU, esta a este nivel por que el applet lo busca debajo de su package.

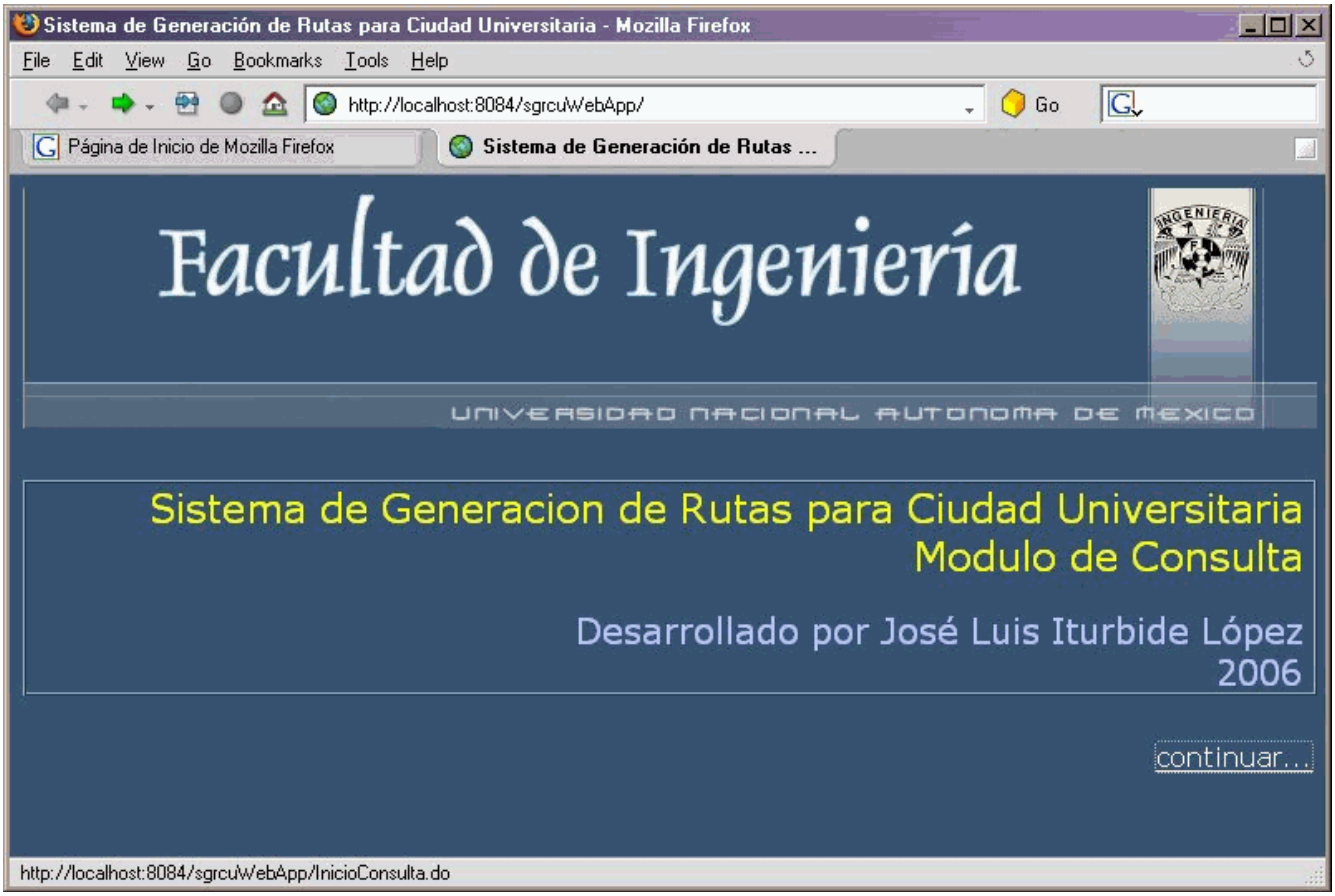
Archivo de configuración applet.properties

El applet al cargarse configura sus colores y tamaño.
Este archivo se encuentra dentro del jar del applet.

```
# Configuración de los valores del applet
applet.initial.width=600
applet.initial.height=400
applet.map.width=1400
applet.map.height=1500
applet.scroll.increment=20
applet.border.width=10
applet.border.height=10

#Datos del gráfico
applet.image.byteswidth=1024
applet.image.bytesheight=768
applet.image.file=images/main.jpg
applet.image.shapewidth=4
applet.image.penwidth=2
#Color de la pluma
applet.image.endcolormark.r=255
applet.image.endcolormark.g=0
applet.image.endcolormark.b=0
applet.image.middlecolormark.r=0
applet.image.middlecolormark.g=0
applet.image.middlecolormark.b=255
applet.image.linecolormark.r=0
applet.image.linecolormark.g=255
applet.image.linecolormark.b=0
#Color de borde del applet
applet.backcolor.r=192
applet.backcolor.g=192
applet.backcolor.b=255
applet.scrollcolor.r=128
applet.scrollcolor.g=128
applet.scrollcolor.b=255
```

Pantallas del Módulo de Consulta

Página inicial

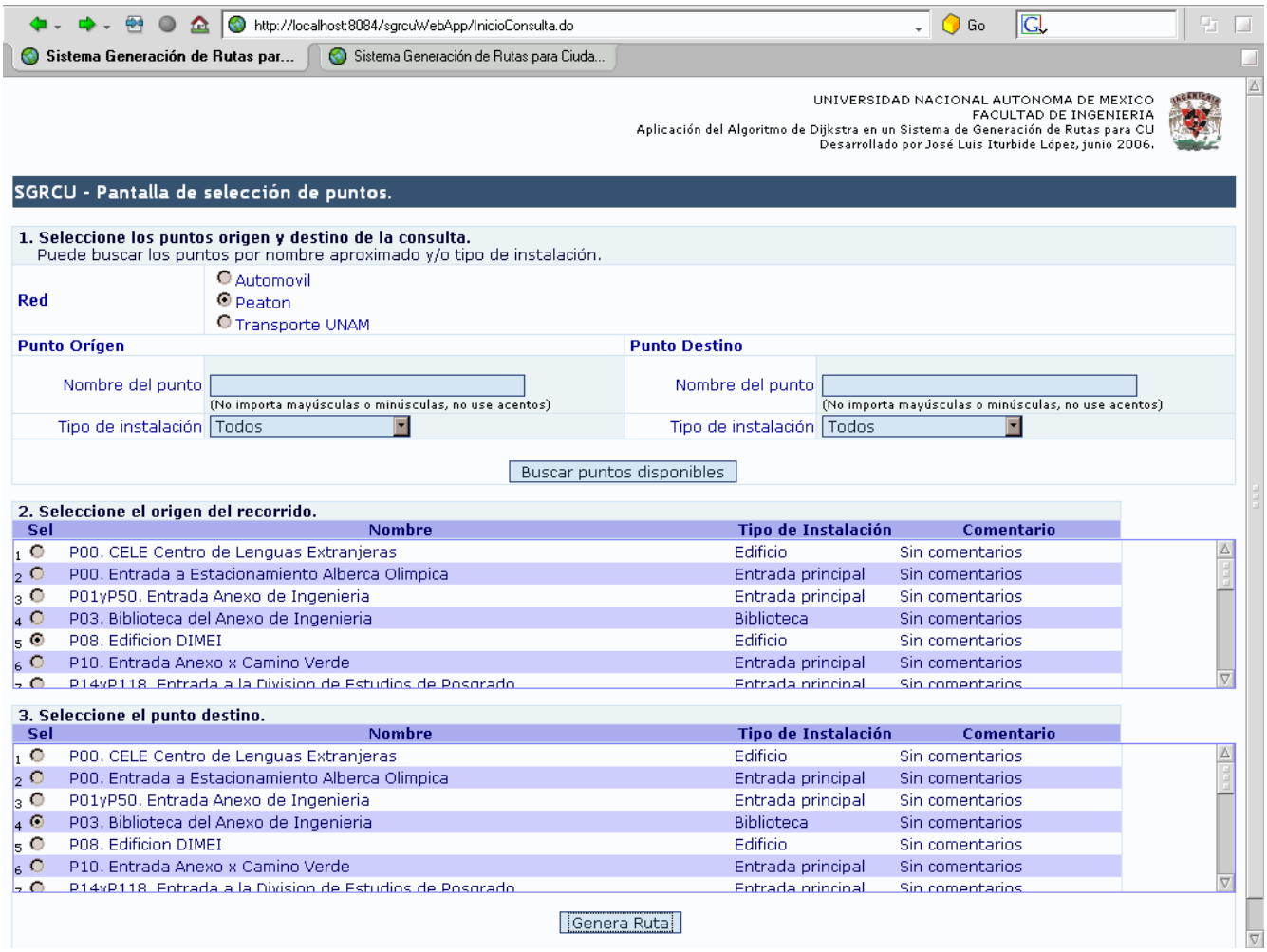


Página de búsqueda de Puntos Origen y Destino por tipo o nombre

Fig. Página de búsqueda de Puntos Origen y Destino por Tipo o Nombre

Descripción	Es la pantalla de selección de punto origen y destino para generar la ruta de recorrido. Hay dos pasos, 1. Primero se buscan los puntos disponibles por nombre o tipo de punto para la red elegida. 2. Después se seleccionan los puntos origen y destino de los listados desplegados.
Nombre	InicioConsulta.jsp
Controles	Radiobutton **Red**: Se elige la red sobre la cual se buscará una ruta de recorrido. Campos **Punto Origen** y **Punto Destino:** **Tipo de Instalación:** Se selecciona un tipo de instalación (tipo de punto) Edificio, Auditorio, si se desea filtrar los puntos disponibles por este criterio. Por default se listan todos. **Nombre del punto:** Se captura un texto que se aproxime al nombre del punto. Ambos criterios son inclusivos.
Botones	Botón ***Buscar*** – Recupera dos listados de puntos disponibles para usarse como Origen y Destino en base a los criterios de selección.
Pantallas asociadas	ResultadoConsulta.jsp
Requerimientos / Casos de uso cubiertos	Requerimiento R2, Caso de uso 1
Notas	Esta página se recarga mostrando las listas de puntos disponibles

Página de selección de Puntos Origen y Destino



Descripción	Es la pantalla de selección de punto origen y destino para generar la ruta de recorrido. Hay dos pasos, 1. Primero se buscan los puntos disponibles por nombre o tipo de punto para la red elegida. 2. Después se seleccionan los puntos origen y destino de los listados desplegados.
Nombre	InicioConsulta.jsp
Controles	Misma descripción de controles que la anterior, Adicionalmente… Lista de puntos de **origen del recorrido** Se muestra un radiobutton por cada punto, se muestran los datos Nombre, Tipo de Instalación y comentario Lista de puntos de **punto destino** Se muestra un radiobutton por cada punto, se muestran los datos Nombre, Tipo de Instalación y comentario
Botones	Botón ***Buscar puntos disponibles*** la funcionalidad es la misma,

	Botón ***Generar Ruta*** – Inicia el proceso de búsqueda de la ruta con la red y el Punto Origen y Destino elegidos.
Pantallas asociadas	ResultadoConsulta.jsp
Requerimientos / Casos de uso cubiertos	Requerimiento R2, Caso de uso 1
Notas	La siguiente pagina es la del Resultado de la consulta

Página de la Solución de la Ruta

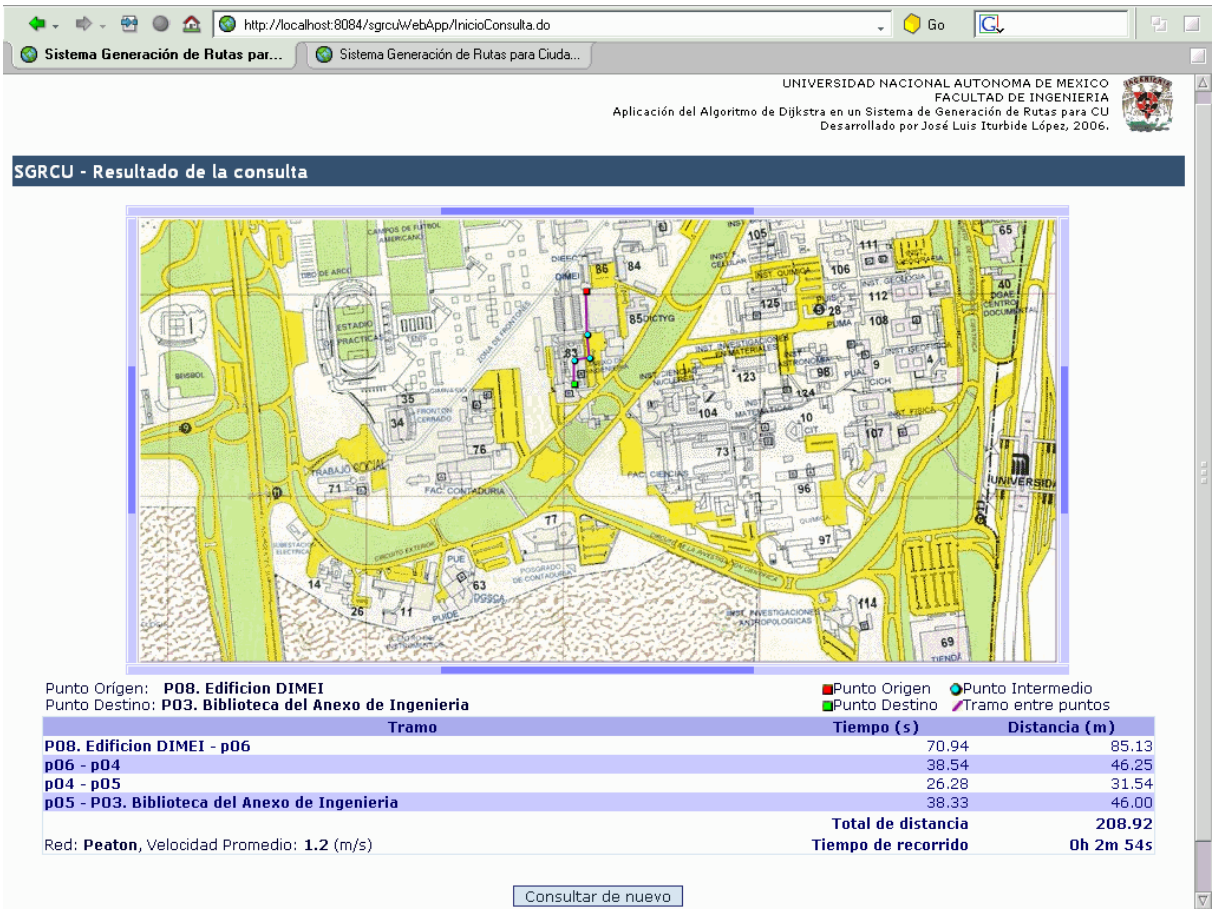


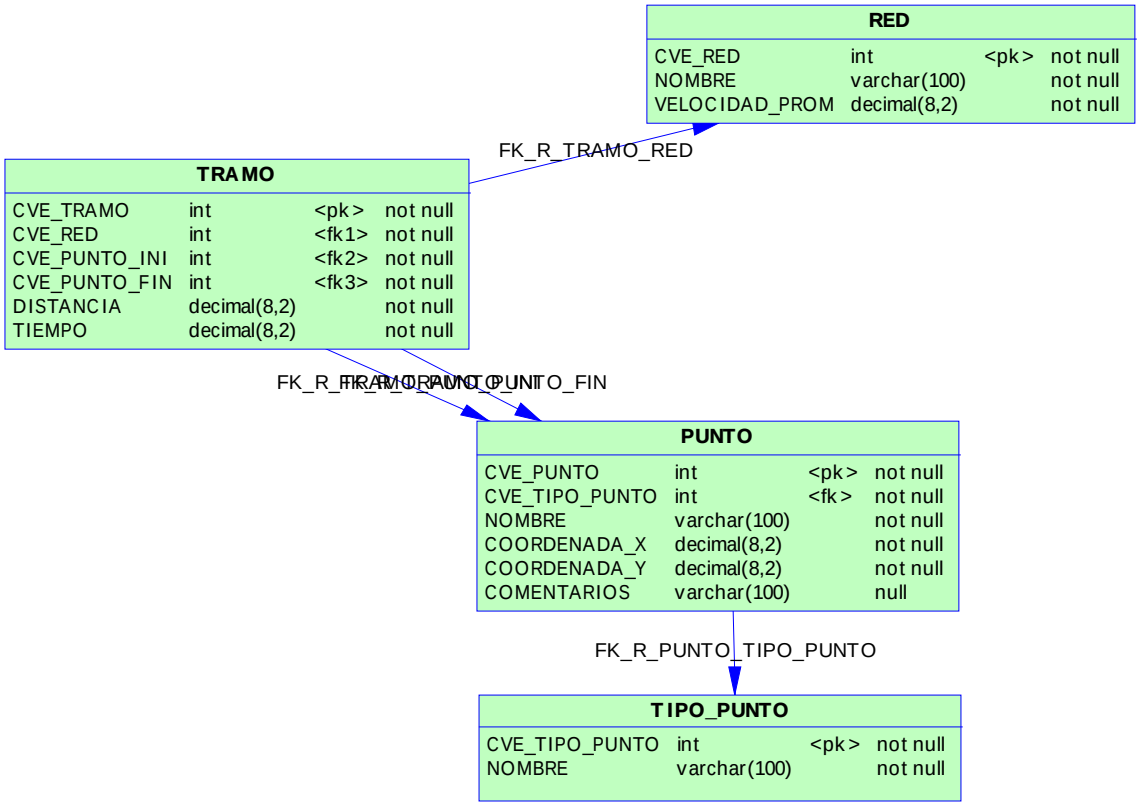
Fig. Página de la Solución de la Ruta

Descripción	Es la pantalla que muestra la solución de la ruta. Se muestra el mapa con el trazo de la solución, una serie de puntos intermedios unidos por líneas que representan los tramos. El punto origen y destino se diferencian de los demás puntos. También se muestra el listado de tramos de la solución con el nombre del tramo, tiempo de recorrido en segundos y distancia en metros. Al final se muestra la red elegida, la velocidad promedio, el tiempo y distancia total del recorrido
Nombre	InicioConsulta.jsp
Controles	Applet del mapa de CU. Hay cuatro líneas alrededor del applet para desplazar el área visible del mapa.
Botones	Botón ***Otra consulta*** regresa a la pantalla inicial.
Pantallas relacionadas	
Requerimientos / Casos de uso cubiertos	Requerimiento R2, Caso de uso 1
Notas	Se puede desplazar la imagen usando las barras laterales azules.

Base de Datos

Modelo físico

El modelo físico de la base de datos es el siguiente



Diccionario de datos

Tabla PUNTOS

Nombre	Descripción	Tipo	M	Pk	Fk
CVE_PUNTO	Clave del punto	int	X	X	
CVE_TIPO_PUNTO	Clave del tipo de punto	int	X		X
NOMBRE	Nombre del punto	varchar(100)	X		
COORDENADA_X	Coordenada en X	decimal(8,2)	X		
COORDENADA_Y	Coordenada en Y	decimal(8,2)	X		
COMENTARIOS	Campos para comentarios adicionales	varchar(100)			

Tabla RED

Nombre	Descripción	Tipo	M	Pk	Fk
CVE_RED	Clave de la red	int	X	X	
NOMBRE	Nombre del tipo de red	varchar(100)	X		
VELOCIDAD_PROM	Velocidad promedio de recorrido en m/s	decimal(8,2)	X		

Tabla TIPO_PUNTO

Nombre	Descripción	Tipo	M	Pk	Fk
CVE_TIPO_PUNTO	Clave del tipo de punto	int	X	X	
NOMBRE	Nombre del tipo de punto	varchar(100)	X		

Tabla TRAMO

Nombre	Descripción	Tipo	M	Pk	Fk
CVE_TRAMO	Clave del tramo	int	X	X	
CVE_RED	Clave de la red	int	X		X
CVE_PUNTO_INI	Clave del punto inicial	int	X		X
CVE_PUNTO_FIN	Clave del punto final	int	X		X
DISTANCIA	Distancia de ir del punto inicial al final en m.	decimal(8,2)	X		
TIEMPO	Tiempo de recorrido en segundos	decimal(8,2)	X		

Propiedades del manejador de bases de datos

El manejador de la base de datos es MySQL tiene las siguientes características:

Propiedad	Valor
Versión	MySQL Server 4.1 para Windows
Puerto	3306
Usuario administrador	root / mysql123
Nombre de base de datos	sgrcu
Tablas	Punto, Tramo, Tipo_Punto, Red
Usuario dueño de la BD	sgrcu / sgrcu
Usuarios	Dueño de la base de datos: **sgrcu** /sgrcu Tiene todos los permisos sobre la base de datos sgrcu El usuario sgrcu debe poder conectarse desde la máquina donde esta instalada la base de datos. Para consulta: **consulta** / consulta123, Permisos de select, Este usuario debe conectarse solo desde el servidor Web Para modificación: **mantenimiento** / mantenimiento123, Permisos de select, insert, update. delete, Este usuario debe conectarse sólo desde la máquina donde se dará mantenimiento al sistema.
Url jdbc	jdbc:mysql://IP_HOST:3306/sgrcu Nota: HOST es la ip del host donde se instala MySQL
Cadena ODBC	SGRCU_MYSQL
Componentes adicionales	▪ Programa de administración del DBMS MySQL Administrador 1.0 ▪ Driver ODBC para MySQL `MyODBC-3.51.11-1-win.msi` ▪ Driver jdbc para conexión java versión **jar**: mysql-connector-java-3.1.6-bin.jar **driver**: com.mysql.jdbc.Driver
Notas	Debe instalarse en este orden: MySQL Server 4.1, MySQL Administrador 1.0, mysql-connector-java-3.1.6, MySQL `MyODBC-3.51.11-1-win.msi`
Usuario de conexión a la bd	Desde el Módulo de Mantenimiento: mantenimiento / mantenimiento123 Desde el Módulo de Consulta: consulta / consulta123

Imágenes de la base de datos en MySQL

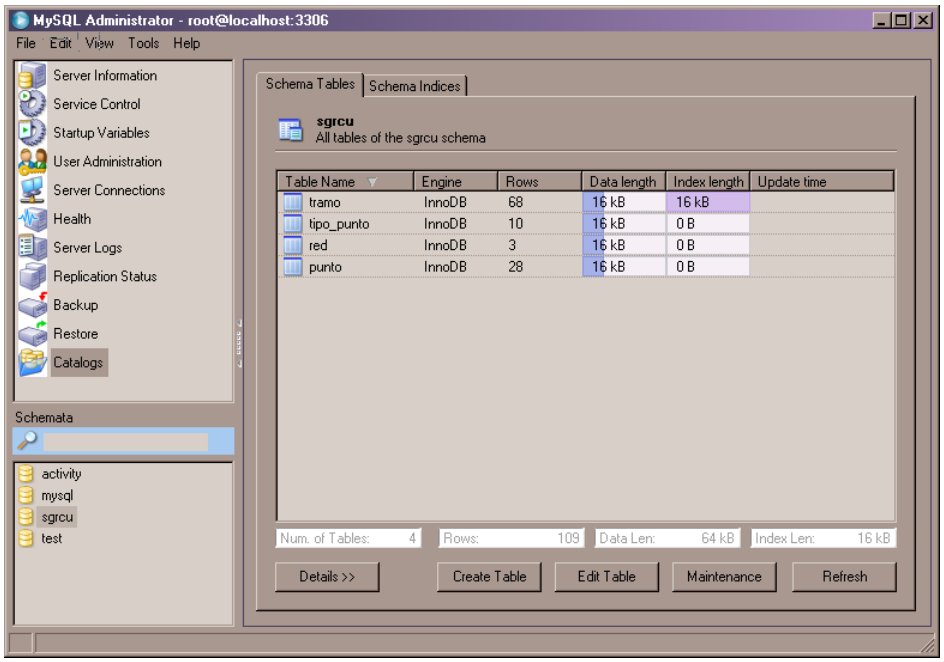


Fig. Vista de la base de datos sgrcu desde MySQL Adminstrator 1.0

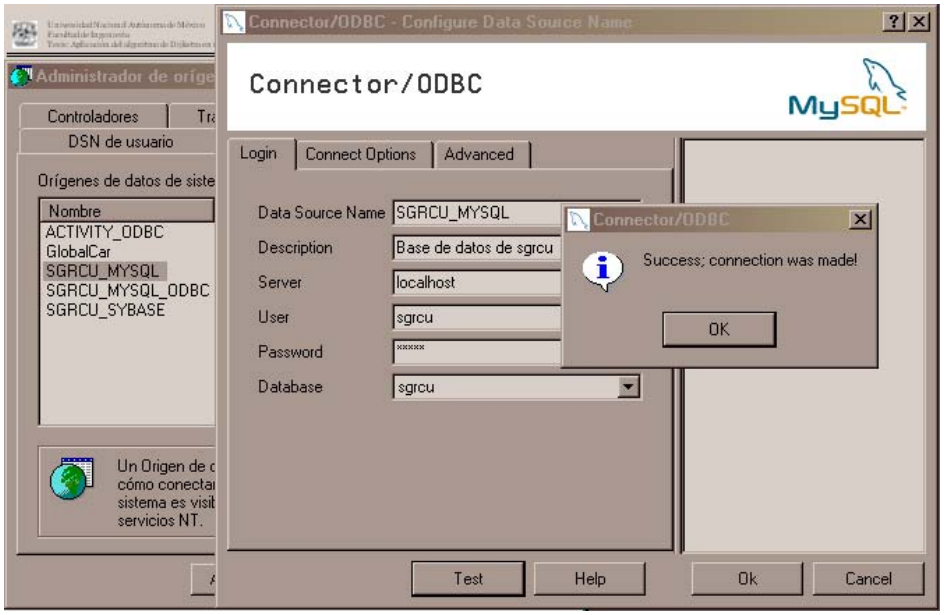


Fig. Configuración del controlador de ODBC

Script de creación de la base de datos

```
/*==============================================================*/
/* Database name:  sgrcu                                        */
/* DBMS name:      MySQL 4.1                                    */
/*==============================================================*/

drop index IX_TRAMO2 on TRAMO;
drop table if exists PUNTO;
drop table if exists RED;
drop table if exists TIPO_PUNTO;
drop table if exists TRAMO;

/*==============================================================*/
/* Table: PUNTO                                                 */
/*==============================================================*/
create table PUNTO
(
   CVE_PUNTO                      int                            not null,
   CVE_TIPO_PUNTO                 int                            not null,
   NOMBRE                         varchar(100)                   not null,
   COORDENADA_X                   decimal(8,2)                   not null,
   COORDENADA_Y                   decimal(8,2)                   not null,
   COMENTARIOS                    varchar(100),
   primary key (CVE_PUNTO)
);

/*==============================================================*/
/* Table: RED                                                   */
/*==============================================================*/
create table RED
(
   CVE_RED                        int                            not null,
   NOMBRE                         varchar(100)                   not null,
   VELOCIDAD_PROM                 decimal(8,2)                   not null,
   primary key (CVE_RED)
);

/*==============================================================*/
/* Table: TIPO_PUNTO                                            */
/*==============================================================*/
create table TIPO_PUNTO
(
   CVE_TIPO_PUNTO                 int                            not null,
   NOMBRE                         varchar(100)                   not null,
   primary key (CVE_TIPO_PUNTO)
);

/*==============================================================*/
/* Table: TRAMO                                                 */
/*==============================================================*/
create table TRAMO
(
   CVE_TRAMO                      int                            not null,
   CVE_RED                        int                            not null,
   CVE_PUNTO_INI                  int                            not null,
   CVE_PUNTO_FIN                  int                            not null,
   DISTANCIA                      decimal(8,2)                   not null,
   TIEMPO                         decimal(8,2)                   not null,
   primary key (CVE_TRAMO)
);

/*==============================================================*/
/* Index: IX_TRAMO2                                             */
/*==============================================================*/
create unique index IX_TRAMO2 on TRAMO
(
   CVE_RED,
   CVE_PUNTO_INI,
   CVE_PUNTO_FIN
);
```

Matriz de Seguimiento

Esta es la actualización de la matriz de seguimiento para la etapa de construcción

Requerimiento	Análisis Caso de Uso	Diseño	Construcción
R1	CU1	Diseño del Componente Lógica de Solución de la ruta	▪ Pantalla Resultado de la Solución de la Ruta ▪ Clases AdjacencyMatrix y Dijkstra del framework
R2	CU1	Diseño de componente Lógica de Consulta	▪ Pantalla de búsqueda de Puntos Origen y Destino por tipo o por nombre ▪ Pantalla de selección de puntos Origen y Destino
R3	CU2, CU3, CU4, CU5	Diseño de componente Módulo de Mantenimiento	▪ Pantallas de Mantenimiento de Red
R4	CU2, CU3, CU4 CU5	Diseño de componente Módulo de Mantenimiento Diseño del Modelo Físico de la Base de Datos	▪ Pantalla de acceso al Módulo de Mantenimiento ▪ Pantalla de Mantenimiento Gráfico de Puntos y Tramos ▪ Pantallas de Mantenimiento de Tipos de Punto ▪ Pantallas de Mantenimiento de Puntos ▪ Pantallas de Mantenimiento de Tramos
R5	CU1	Clases del framework Dijkstra	▪ Clases AdjacencyMatrix y Dijkstra del framework ▪ Pantalla de Mantenimiento de Tramos
R6	CU3, CU4, C5	Diseño de componente Módulo de Mantenimiento	▪ Pantalla de Mantenimiento Gráfico de Puntos y Tramos
R7	CU4, C5	Diseño de componente Módulo de Mantenimiento	▪ Pantalla de Mantenimiento Gráfico de Puntos y Tramos
R10	CU1	Clases del framework Dijkstra	▪ Clases AdjacencyMatrix y Dijkstra del framework
R11	CU1	Diseño de componente Módulo de Consulta	▪ Todas las pantallas del Módulo de Consulta Web
R12	CU1	Diseño de componente Módulo de Mantenimiento	▪ Pantalla de Mantenimiento Gráfico de Puntos y Tramos ▪ Imagen de CU de 1400x1500 ▪ Applet SolucionApplet.java para mostrar el mapa.
R13		Diseño de componente Módulo de Consulta Diseño de componente Módulo de Mantenimiento	▪ El diseño de clases de control, DTO, DAO, Command, Struts, hace flexible al sistema ▪ El diseño de la aplicación Visual Basic también prevé esta condición
R14	CU1	Diseño de componente Módulo de Consulta	▪ Pantalla Resultado de la Solución de la Ruta

CAPÍTULO 7. RESULTADOS Y CONCLUSIONES

En este capítulo se describen las pruebas realizadas en el sistema para la comprobación de su funcionamiento y a partir de los resultados se concluye sobre el cumplimiento del objetivo formulado.

Carga inicial de datos de prueba

Para que el sistema sea completamente funcional es necesario dar de alta puntos suficientes para generar cualquier ruta dentro del campus; pero esta es una tarea ardua y debe ser realizada por un equipo específico de personas, lo cual queda fuera del alcance inicial del sistema. Por ello en el capítulo de análisis se propuso que para probar el funcionamiento del sistema se realizará una carga de datos que permita hacer una consulta de la ruta para ir del Edificio Anexo de Ingeniería al Edificio Principal.

La carga inicial se limita a puntos y tramos de está zona, estos deben ser suficientes para generar una solución de ruta mas corta de entre varias posibles alternativas. Se dan de alta los puntos en 2 redes diferentes: Puntos para la red de automóviles y puntos para la red de peatón.

Recopilación de datos en sitio

Para realizar pruebas con datos reales, se hace un levantamiento de datos en sitio.

Para efecto de mostrar el funcionamiento del sistema se hace un solo levantamiento de datos de tiempos de recorrido y distancias de los puntos y tramos existentes entre el Anexo de Ingeniería y el Edificio Principal. Este levantamiento de datos no sigue ninguna metodología formal, se realiza de forma directa y apoyándose en suposiciones y de otras herramientas como se explica más adelante. El levantamiento de datos con datos aceptables debe realizarlo personas experimentadas en toma de distancias y tiempos, tal vez gente de Transporte y Geográfia.

Errores de la toma de datos.

Los errores que puede generarse en el levantamiento de datos están identificados. El principal error está en la metodología de toma de datos.

- Para mediciones de tiempo de recorrido es: Velocidad no constante, Lectura de tiempo no exacto, errores generados por el lecturista.
- Para mediciones de distancia: Definición no exacta de los puntos de inicio y fin de la ruta, trazo de la ruta, generalmente recorremos un camino con una trayectoria sin “bordes” pero en el sistema son tramos rectos siempre.

Debe cuidarse en no generar estos errores en la carga formal de datos ya que si ocurren afectarán la calidad de la información proporcionada del sistema.

Dado lo anterior ***se conoce que los datos recopilados en la visita en sitio no son exactos pero son muy cercanos a los reales*** y ayudarán a realizar la prueba del sistema.

Plan de recopilación de datos

Punto Origen y Punto Destino.

- Se elige como punto de origen de la ruta para peatón y automóvil la puerta principal de entrada al Anexo de Ingeniería para la red de peatón y de automóvil por ser un punto común y como punto destino de la ruta la segunda entrada al Edificio de la Facultad de Ingeniería sobre el circuito.

Consideraciones en el levantamiento de datos

- Se considera que la velocidad de desplazamiento es constante, con esto se logra que para una misma distancia los tiempos de recorrido son los mismos en un sentido y en otro siguiendo el mismo camino. La velocidad de desplazamiento puede variar en cualquier momento por distracciones, por condiciones físicas del camino, por tráfico, etc.
- El tiempo de recorrido a pie y en automóvil se mide con un cronometro convencional.
- Para la red de automóvil el recorrido se hace a una velocidad de 30 km/h.
- La guía de recorrido se hace con una imagen de apoyo. En la imagen el punto marcado en rojo marca la entrada al Anexo de Ingeniería, el punto marcado en verde es la entrada al Edificio Principal de la Facultad de Ingeniería, los puntos azules indican entronques o puntos de apoyo para dar forma a la ruta, no son puntos origen o destino. Los puntos en amarillo son puntos importantes que estarán también en el catálogo de puntos disponibles iniciales o finales.
- Se numeraron los puntos arbitrariamente para una mejor identificación.

Símbolos usados en el mapa de apoyo
Punto de apoyo de conexión;
Punto inicial
Punto final
Punto de alguna instalación que es de paso.
Línea de conexión entre puntos, cada segmento define un tramo

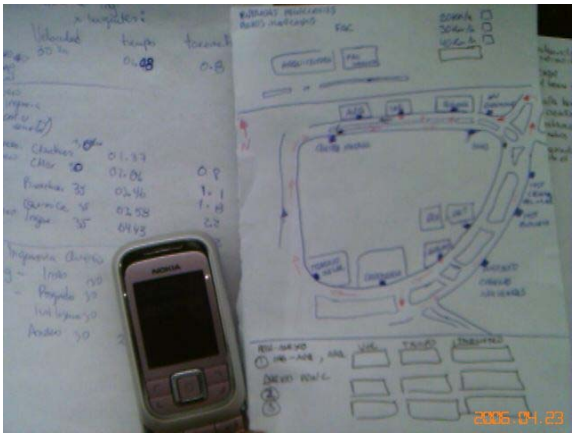


Fig. Preparación del itinerario del levantamiento de puntos

Fig. Ejecución de la toma de datos en Automóvil

Recorrido del levantamiento de datos para la red de peatón

Para la red de peatón estos valores aplican para un sentido u otro.

Recorrido 1. Del Anexo de Ingeniería al edificio Principal de la Facultad de Ingeniería por caminando por el área de frontones.

Recorrido 2. Del Anexo de Ingeniería al edificio Principal de la Facultad de Ingeniería por el circuito exterior oriente.

Recorrido 3. Del Anexo de Ingeniería al edificio Principal de la Facultad de Ingeniería sin pasar por el interior de los frontones

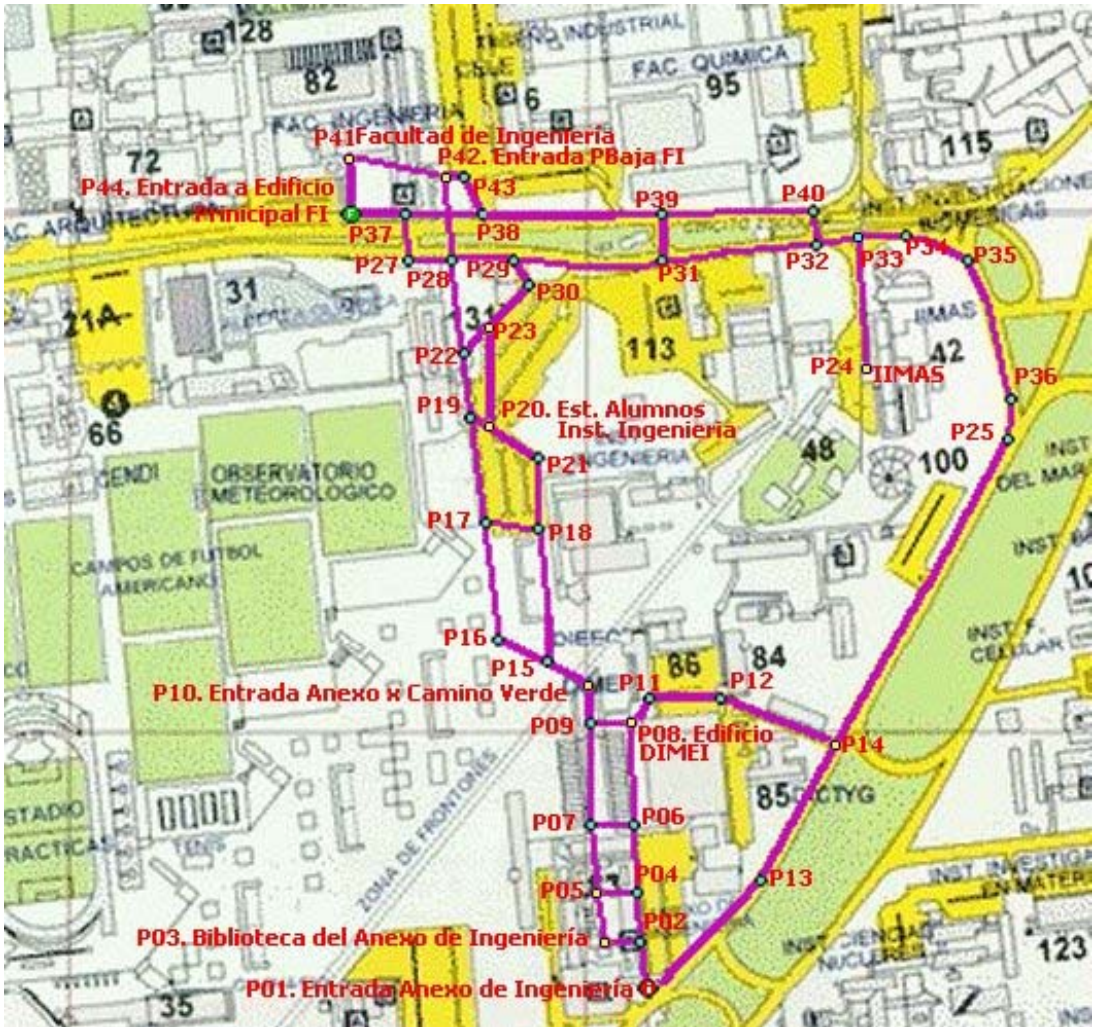


Fig. Tramos y Puntos considerados en el levantamiento de puntos de la red de peatón

Recorrido del levantamiento de datos para la red de automóvil

Los datos de tiempo de recorrido se recopilan haciendo varios recorridos.

Recorrido 1. Del Anexo de Ingeniería al edificio Principal de la Facultad de Ingeniería por el circuito exterior poniente

Recorrido 2. Del edificio Principal de la Facultad de Ingeniería al Anexo de Ingeniería por el circuito exterior oriente

Recorrido 3. Del Anexo de Ingeniería al edificio Principal de la Facultad de Ingeniería por el circuito exterior oriente.

Para la red de automóvil solo aplica un sentido, por lo que los datos recopilados deben darse de alta para un solo sentido

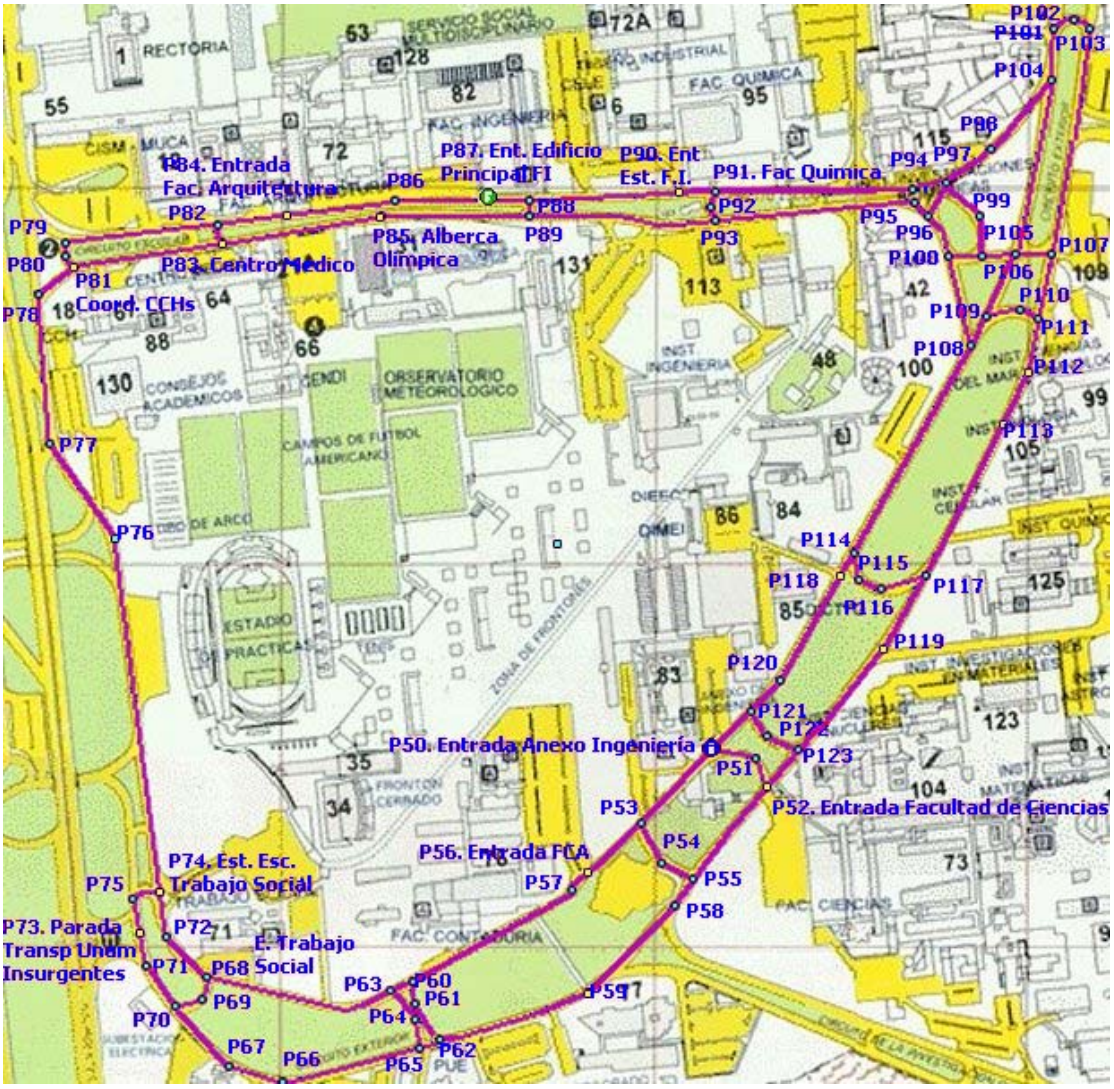


Fig. Tramos considerados en el levantamiento de puntos de la red de automóvil

Métodos indirectos para el cálculo de tiempo de recorrido y distancias.

El recorrido realizado nos ayuda a obtener algunos valores aunque no todos y también para comprobar si los valores supuestos distan de los reales.

Ante la imposibilidad de obtener las mediciones de distancias y una velocidad de recorrido exacta se trabaja con los siguientes valores:

Velocidad de desplazamiento.

Red	Velocidad a registrar	Comentario
Automóvil	8.33 m/s	Se toma como velocidad de recorrido 30km/h = 8.33 m/s
Peatón	1.20 m/s	Mediante las mediciones Tiempo vs. Distancia, se encontró que el promedio era de 1.233 m/s, se toma 1.20 m/s
Transporte UNAM	5.55 m/s	No se midió en estas pruebas, se asume que avanza a razón de 20 km/h = 5.55 m/s

Distancias entre puntos

Las distancias de recorrido se obtienen por métodos indirectos.

Las distancias se obtienen con ayuda de Google Earth[1]. Para comprobar que este software proporciona distancias reales se hace una prueba de medición de un costado de las Islas.

Tramo de las Islas	Medición directa	Medición con Google Eart	Variación %
Costado Norte de las Islas EsqNte01 – EsqNte02	310.35	311.00	0.20
Extremo Oriente EsqSur05 – EsqNte02	115.00	118.62	3.14
Islas Vertical MitadNte03 – MitadSur04	153.83	159	3.36

Se usara Google Earth como apoyo para obtener las distancias de los tramos de prueba ya que hay poca diferencia con respecto a las mediciones directas.

[1] Google Eart es un software que muestra una vista aérea de varias regiones del mundo. Puede calcular la distancia entre 2 puntos de un mapa haciendo uso de escalas.

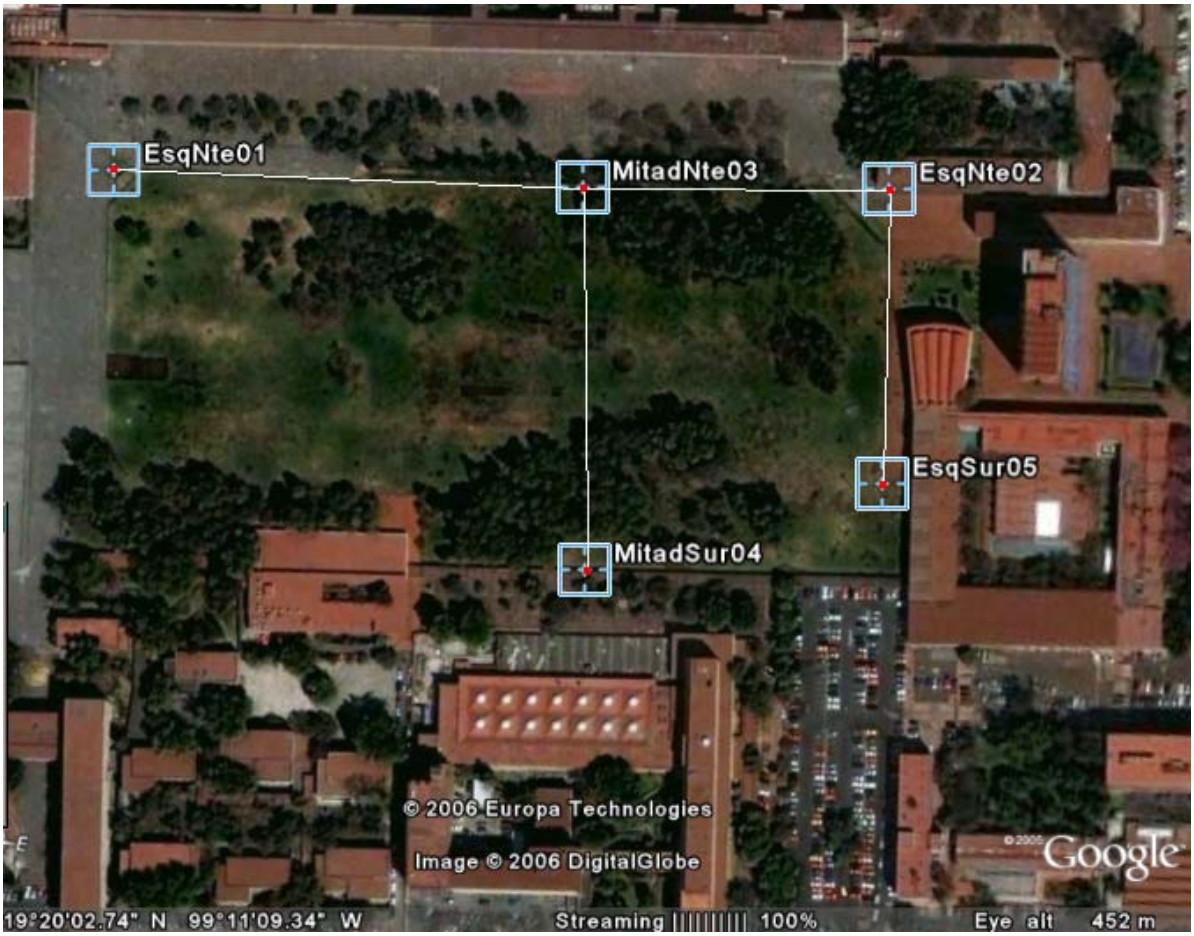


Fig. Tramos de las "Islas" para comparar Medición de Google Earth vs. Medición Directa

Escala

Para que el sistema de mantenimiento calcule automáticamente las distancias entre puntos cuando se da de alta un tramo, se obtiene el factor de escala del mapa = Píxeles vs. Distancia real.

El factor de escala es:

Píxeles vertical: **2.40**

Píxeles horizontal: **2.30**

Con estos valores es posible ahora dar de alta directamente en el sistema y que se calcule automáticamente la distancia entre los tramos. **Esta distancia es aproximada y sirve solo para efectos demostrativos del sistema.**

Resultado del levantamiento de datos de tramos y distancias de recorrido

*Datos de tramos para la red de peatón**

Punto inicial	Punto final	Distancia (m)	Tiempo (s)
P01yP50. Entrada Anexo de Ingenieria	p02	32.20	26.83
p02	P01yP50. Entrada Anexo de Ingenieria	32.20	26.83
p02	P03. Biblioteca del Anexo de Ingenieria	31.54	26.28
p02	p04	46.00	38.33
P03. Biblioteca del Anexo de Ingenieria	p02	31.54	26.28
P03. Biblioteca del Anexo de Ingenieria	p05	46.00	38.33
p04	p02	46.00	38.33
p04	p05	31.54	26.28
p04	p06	46.25	38.54
p05	P03. Biblioteca del Anexo de Ingenieria	46.00	38.33
p05	p04	31.54	26.28
p05	p07	46.56	38.80
p06	p04	46.25	38.54
p06	p07	33.91	28.26
p06	P08. Edificio DIMEI	85.13	70.94
p07	p05	46.56	38.80
p07	p06	33.91	28.26
p07	p09	83.35	69.46
P08. Edificio DIMEI	p06	85.13	70.94
P08. Edificio DIMEI	p09	41.38	34.48
P08. Edificio DIMEI	p11	19.74	16.45
P08. Edificio DIMEI	P47. Edificio Valdes Vallejo	57.55	47.96
p09	p07	83.35	69.46
p09	P08. Edificio DIMEI	41.38	34.48
p09	P10. Entrada Anexo x Camino Verde	26.30	21.92
P10. Entrada Anexo x Camino Verde	p09	26.30	21.92
P10. Entrada Anexo x Camino Verde	p15	31.13	25.94
P10. Entrada Anexo x Camino Verde	p26	263.04	219.20
P10. Entrada Anexo x Camino Verde	P47. Edificio Valdes Vallejo	63.79	53.16
p11	P08. Edificio DIMEI	19.74	16.45
p11	p12	58.01	48.34
p12	p11	58.01	48.34
p12	P46. Division de Ingenieria de Ciencias de la Tierra y Geodesica	57.70	48.08
p12	P48. Division de Estudios de Posgrado	30.00	25.00
p12	P50. Instituto Ingenieria Edificio 5	51.41	42.84
p13	P14yP118. Entrada a la Division de Estudios de Posgrado	124.07	103.39
P14yP118. Entrada a la Division de Estudios de Posgrado	p13	124.07	103.39
P14yP118. Entrada a la Division de Estudios de Posgrado	P50. Instituto Ingenieria Edificio 5	35.11	29.26
p15	P10. Entrada Anexo x Camino Verde	31.13	25.94
p15	p16	36.66	30.55
p15	p18	92.28	76.90
p16	p15	36.66	30.55
p16	p17	90.85	75.71
p17	p16	90.85	75.71
p17	p18	43.44	36.20
p17	p19	84.04	70.04
p17	P20. Est. Alumnos Instituto de Ingenieria	71.94	59.95
p17	p21	59.79	49.82
p18	p15	92.28	76.90
p18	p17	43.44	36.20

p18	p19	104.67	87.23
p18	P20. Est. Alumnos Instituto de Ingenieria	83.00	69.17
p18	p21	48.36	40.30
p19	p17	84.04	70.04
p19	p18	104.67	87.23
p19	P20. Est. Alumnos Instituto de Ingenieria	26.61	22.18
p19	p22	36.88	30.73
P20. Est. Alumnos Instituto de Ingenieria	p17	71.94	59.95
P20. Est. Alumnos Instituto de Ingenieria	p18	83.00	69.17
P20. Est. Alumnos Instituto de Ingenieria	p19	26.61	22.18
P20. Est. Alumnos Instituto de Ingenieria	p21	41.66	34.71
P20. Est. Alumnos Instituto de Ingenieria	P23. Comedor Facultad de Ingenieria	67.77	56.48
p21	p17	59.79	49.82
p21	p18	48.36	40.30
p21	P20. Est. Alumnos Instituto de Ingenieria	41.66	34.71
p22	p19	36.88	30.73
p22	P23. Comedor Facultad de Ingenieria	23.36	19.47
p22	p_x802y366	48.54	40.45
P23. Comedor Facultad de Ingenieria	P20. Est. Alumnos Instituto de Ingenieria	67.77	56.48
P23. Comedor Facultad de Ingenieria	p22	23.36	19.47
P23. Comedor Facultad de Ingenieria	p30	35.53	29.61
P24. IIMAS	p26	74.88	62.40
P24. IIMAS	P33. Entrada Est. IIMAS	92.50	77.08
p25y108	p36	30.28	25.24
p25y108	p114	247.65	206.37
p26	P10. Entrada Anexo x Camino Verde	263.04	219.20
p26	P24. IIMAS	74.88	62.40
p26	P49. Facultad de Quimica	268.91	224.09
p27y89	p29	76.94	64.11
p27y89	p37y88	33.00	27.50
p27y89	P85. Alberca Olimpica	141.60	118.00
p27y89	p_x787y366	32.56	27.13
p29	p27y89	76.94	64.11
p29	p30	27.70	23.09
p29	P31yP93. Ent Estacionamiento Instituto de Ingenieria	124.82	104.02
p29	p38	39.52	32.93
p30	P23. Comedor Facultad de Ingenieria	35.53	29.61
p30	p29	27.70	23.09
p30	p_x802y366	38.40	32.00
P31yP93. Ent Estacionamiento Instituto de Ingenieria	p29	124.82	104.02
P31yP93. Ent Estacionamiento Instituto de Ingenieria	p32	116.32	96.93
P31yP93. Ent Estacionamiento Instituto de Ingenieria	P39yP91. Parada TUNAM Fac. Quimica	39.17	32.64
P31yP93. Ent Estacionamiento Instituto de Ingenieria	P45. Instituto Ingeneria Centro de Registro Sismico	88.42	73.68
p32	P31yP93. Ent Estacionamiento Instituto de Ingenieria	116.32	96.93

***Nota**: En la captura se omiten los acentos para poder realizar búsquedas por aproximación

Después de insertar estos valores en el sistema, se usa este comando SQL para la consulta de los tramos de la red de Peatón

```
select p1.cve_punto, p1.nombre, p2.nombre, distancia, tiempo
 from tramo t, punto p1, punto p2
where cve_red = 1
  and t.cve_punto_ini = p1.cve_punto
  and t.cve_punto_fin = p2.cve_punto
order by p1.cve_punto
```

*Datos de tramos para la red de automóvil**

Punto inicial	Punto final	Distancia (m)	Tiempo (s)
P01yP50. Entrada Anexo de Ingenieria	p53	94.96	11.40
p13	p121	45.29	5.44
P14yP118. Entrada a la Division de Estudios de Posgrado	p13	124.07	14.89
p25y108	p114	247.65	29.73
p27y89	p29	76.94	9.24
p27y89	p37y88	27.60	3.31
p29	P31yP93. Ent Estacionamiento Instituto de Ingenieria	124.82	14.98
P31yP93. Ent Estacionamiento Instituto de Ingenieria	p32	116.32	13.96
p32	P33. Entrada Est. IIMAS	31.28	3.76
p32	p40	23.50	2.82
P33. Entrada Est. IIMAS	p34y95	45.66	5.48
p34y95	p35y96	32.53	3.90
p35y96	p100	57.50	6.90
p36	p25y108	20.84	2.50
p37y88	p27y89	27.60	3.31
p37y88	P44y87. Entrada a Edificio Principal Facultad de Ingenieria	48.06	5.77
p38	p37y88	62.57	7.51
P39yP91. Parada TUNAM Fac. Quimica	P31yP93. Ent Estacionamiento Instituto de Ingenieria	39.17	4.70
P39yP91. Parada TUNAM Fac. Quimica	P90. Entrada Est. Facultad de Ingenieria	31.54	3.79
p40	p_x899y338	38.40	4.61
P44y87. Entrada a Edificio Principal Facultad de Ingenieria	P86. Parada TUNAM Facultad de Arquitectura	144.66	17.37
p51	P01yP50. Entrada Anexo de Ingenieria	62.78	7.54
P51. Entrada a Facultad de Ciencias	p51	36.53	4.39
P51. Entrada a Facultad de Ciencias	p123	58.41	7.01
p53	p54	41.51	4.98
p53	P56. Entrada a Facultad de Contaduria y Administracion	77.52	9.31
p54	p55	40.43	4.85
p55	P51. Entrada a Facultad de Ciencias	117.95	14.16
P56. Entrada a Facultad de Contaduria y Administracion	p57	28.00	3.36
p57	p60	192.57	23.12
p58	p55	50.43	6.05
P59. Posgrado de Contaduria	p58	114.77	13.78
p60	p61	35.81	4.30
p60	p63	24.47	2.94
p61	P62. Parada TUNAM DGSCA	38.19	4.59
P62. Parada TUNAM DGSCA	P59. Posgrado de Contaduria	158.73	19.05
p63	p_x698y706	66.77	8.02
p64	p63	36.58	4.39
p65	P62. Parada TUNAM DGSCA	27.29	3.28
p65	p64	30.75	3.69
p66	p65	145.74	17.50
p67	p66	74.92	8.99
P68. Parada TUNAM Trabajo Social	p72	56.75	6.81
p70	p67	71.42	8.57
p72	P74. Entrada a Estacionamiento Trabajo Social	39.17	4.70
P73. Parada TUNAM Insurgentes	p70	92.31	11.08
P74. Entrada a Estacionamiento Trabajo Social	p75	28.89	3.47
P74. Entrada a Estacionamiento Trabajo Social	P80. Estadio de Practicas	250.66	30.09
p75	P73. Parada TUNAM Insurgentes	39.17	4.70
p76	p77	115.00	13.81
p77	p78	145.87	17.51
p78	P81. Coordinacion de CCHs	49.86	5.99

p79	P81. Coordinacion de CCHs	30.00	3.60
P80. Estadio de Practicas	p76	123.40	14.81
P81. Coordinacion de CCHs	p_x636y364	132.05	15.85
p82	P83. Centro Medico	23.50	2.82
p82	p_x633y355	33.68	4.04
P83. Centro Medico	P00. Entrada a Estacionamiento Alberca Olimpica	98.83	11.86
P86. Parada TUNAM Facultad de Arquitectura	p82	140.73	16.89
P85. Alberca Olimpica	p27y89	141.60	17.00
P00. Entrada a Estacionamiento Alberca Olimpica	P85. Alberca Olimpica	88.34	10.60
P90. Entrada Est. Facultad de Ingenieria	p38	105.70	12.69
p_x899y338	P39yP91. Parada TUNAM Fac. Quimica	74.40	8.93
P94. Instituto Investigaciones Biomedicas	p40	91.20	10.95
p97	p35y96	38.77	4.65
p97	P94. Instituto Investigaciones Biomedicas	43.44	5.22
p98	p97	59.81	7.18
p104	p98	99.80	11.98
p104	p105	186.22	22.36
p101	p104	57.95	6.96
p102	p101	19.34	2.32
p103	p102	18.16	2.18
p100	p36	68.24	8.19
p100	p106	26.80	3.22
p99	p97	53.15	6.38
p105	p107	43.75	5.25
p105	p109	71.73	8.61
p106	p99	36.80	4.42
p106	p105	38.47	4.62
p107	p103	231.33	27.77
p109	p25y108	35.05	4.21
p110	p109	30.23	3.63
p111	p107	79.51	9.55
p111	p110	29.79	3.58
P112. Instituto de Ciencias del Mar	p111	50.40	6.05
P113. Instituto Biologia	P112. Instituto de Ciencias del Mar	63.27	7.60
p114	P14yP118. Entrada a la Division de Estudios de Posgrado	26.66	3.20
p114	p115	32.22	3.87
p115	p116	25.70	3.09
p116	p117	46.96	5.64
p117	P119. Ent. Instituto de Quimica	42.67	5.12
P119. Entrada a Instituto de Investigacion de Materiales	p117	97.70	11.73
p121	P01yP50. Entrada Anexo de Ingenieria	72.30	8.68
p121	p122	33.62	4.04
p122	p123	35.51	4.26
p123	P119. Entrada a Instituto de Investigacion de Materiales	129.60	15.56
P119. Ent. Instituto de Quimica	P113. Instituto Biologia	129.71	15.57
p_x69	P68. Parada TUNAM Trabajo Social	142.28	17.08
p_x633	p79	125.83	15.11
p_x636	P83. Centro Medico	31.20	3.75

***Nota**: En la captura se omiten los acentos para poder realizar búsquedas por aproximación

Después de insertar estos valores en el sistema, se usa este comando SQL para la consulta de los tramos de la red de Automóvil

```
select p1.cve_punto, p1.nombre, p2.nombre, distancia, tiempo
 from tramo t, punto p1, punto p2
where cve_red = 2
  and t.cve_punto_ini = p1.cve_punto
  and t.cve_punto_fin = p2.cve_punto
order by p1.cve_punto
```

Carga del resto de los catálogos

Los siguientes valores los catálogos de Red y Tipo de Punto se cargan en el Módulo de Mantenimiento.

Datos del catálogo de redes

Id Interno	Nombre de la Red	Velocidad promedio (m/s)	Comentarios
1	Peatón	1.20	1.20 (m/s) Fue la velocidad promedio de recorrido para un peatón
2	Automóvil	8.33	8.33 (m/s) Es la conversión de 30 (km/h)
3	Transporte UNAM	5.55	5.55 (m/s) Es la conversión de 20 (km/h)

Datos del catálogo de Tipos de Punto

Estos son los datos iniciales del catálogo, se pueden agregar mas según se vayan necesitando.

Id Interno	Nombre del Tipo de Punto o Instalación	Descripción
1	Punto de conexión	Es el tipo de punto de apoyo para dar curvatura a los tramos de recorrido. Estos puntos solo serán visibles desde el sistema de mantenimiento, pero desde el sistema de consulta no serán elegibles.
2	Parada de transporte	Cualquier parada de transporte público o de la UNAM
3	Auditorio	Auditorio
4	Biblioteca	Biblioteca
5	Edificio	Cualquier edificio Facultad, Escuela, Instituto, etc.
6	Andador	Camino disponible solo para peatones
7	Comedor	Comedor
8	Estacionamiento	Estacionamiento
9	Instalación deportiva	Instalación deportiva de cualquier tipo.
10	Entrada principal	Entrada principal a instalación

Pruebas

Teniendo la red de datos de puntos y tramos registrada en el sistema puede realizarse una consulta.

Prueba 1. Consulta de ruta en la red de Peatón

Datos de entrada

Se ejecuta la siguiente prueba:

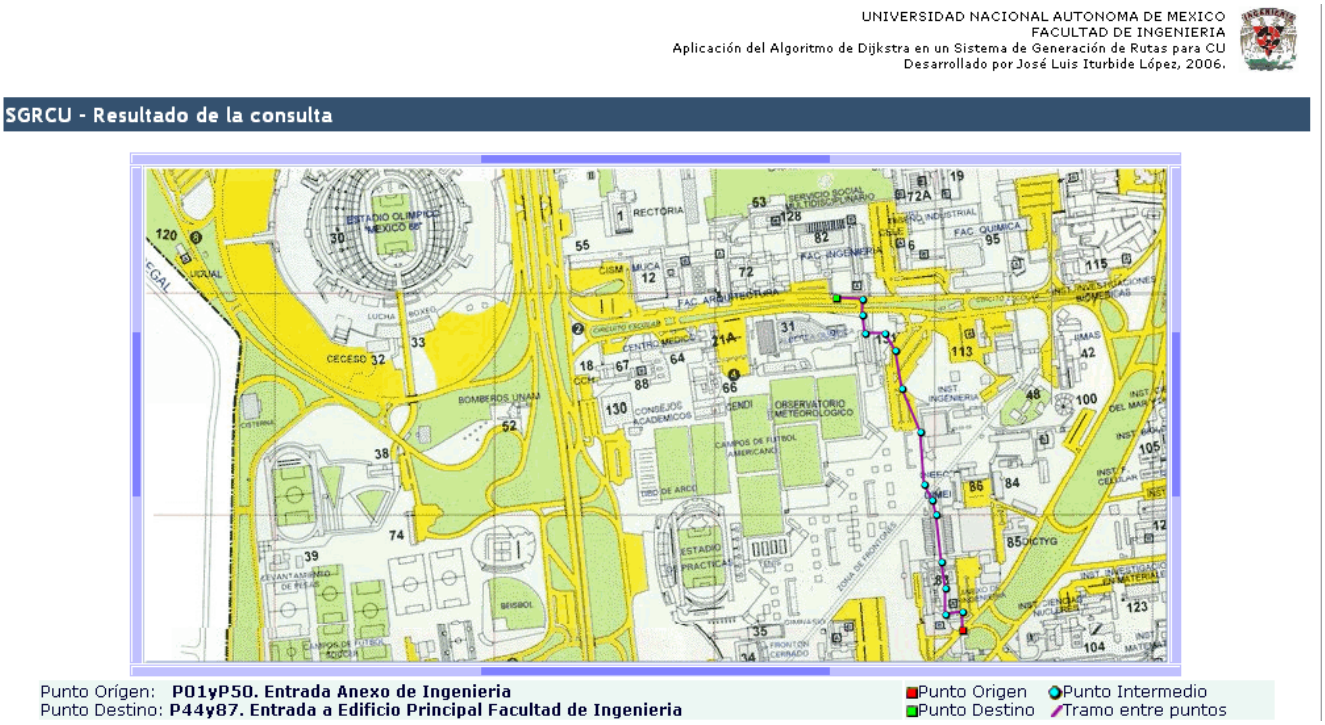
Red: **Peatón**

Punto Origen: **Entrada Anexo de Ingeniería**

Punto Destino: **1ª Entrada del edificio Principal Facultad de Ingeniería**

Estos datos se proporcionan en el módulo de consulta y se genera la ruta

Resultado del sistema



Tramo	Tiempo (s)	Distancia (m)
P01yP50. Entrada Anexo de Ingenieria - p02	26.83	32.20
p02 - P03. Biblioteca del Anexo de Ingenieria	26.28	31.54
P03. Biblioteca del Anexo de Ingenieria - p05	38.33	46.00
p05 - p07	38.80	46.56
p07 - p09	69.46	83.35
p09 - P10. Entrada Anexo x Camino Verde	21.92	26.30
P10. Entrada Anexo x Camino Verde - p15	25.94	31.13
p15 - p18	76.90	92.28
p18 - P20. Est. Alumnos Instituto de Ingenieria	69.17	83.00
P20. Est. Alumnos Instituto de Ingenieria - P23. Comedor Facultad de Ingenieria	56.48	67.77
P23. Comedor Facultad de Ingenieria - p_x802y366	29.61	35.53
p_x802y366 - p_x787y366	30.00	36.00
p_x787y366 - p27y89	27.13	32.56
p27y89 - p37y88	27.50	33.00
p37y88 - P44y87. Entrada a Edificio Principal Facultad de Ingenieria	38.05	45.66
	Total de distancia	**722.88**
Red: **Peaton**, Velocidad Promedio: **1.2** (m/s)	**Tiempo de recorrido**	**0h 10m 2s**

Consultar de nuevo

El sistema encuentra que la distancia mas corta para esta consulta es **722.88 mts**. y el tiempo de recorrido suponiendo que se tiene una velocidad promedio de **1.20 m/s**. es de **10** minutos **2** segundos.

Prueba 2. Consulta de ruta en la red de Automóvil

Datos de entrada

Red: **Automóvil**

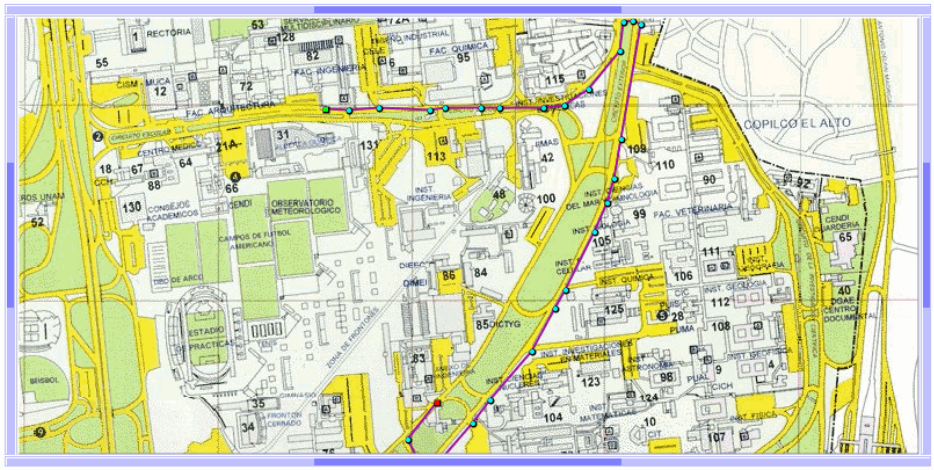
Punto Origen: **Entrada Anexo de Ingeniería**

Punto Destino: **1ª Entrada del edificio Principal Facultad de Ingeniería**

Resultado del sistema

UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
FACULTAD DE INGENIERIA
Aplicación del Algoritmo de Dijkstra en un Sistema de Generación de Rutas para CU
Desarrollado por José Luis Iturbide López, 2006.

SGRCU - Resultado de la consulta



Punto Orígen: **P01yP50. Entrada Anexo de Ingenieria**
Punto Destino: **P44y87. Entrada a Edificio Principal Facultad de Ingenieria**

Punto Origen · Punto Intermedio · Punto Destino · Tramo entre puntos

Tramo	Tiempo (s)	Distancia (m)
P01yP50. Entrada Anexo de Ingenieria - p53	11.40	94.96
p53 - p54	4.98	41.51
p54 - p55	4.85	40.43
p55 - P51. Entrada a Facultad de Ciencias	14.16	117.95
P51. Entrada a Facultad de Ciencias - p123	7.01	58.41
p123 - P119. Entrada a Instituto de Investigacion de Materiales	15.56	129.60
P119. Entrada a Instituto de Investigacion de Materiales - p117	11.73	97.70
p117 - P119. Ent. Instituto de Quimica	5.12	42.67
P119. Ent. Instituto de Quimica - P113. Instituto Biologia	15.57	129.71
P113. Instituto Biologia - P112. Instituto de Ciencias del Mar	7.60	63.27
P112. Instituto de Ciencias del Mar - p111	6.05	50.40
p111 - p107	9.55	79.51
p107 - p103	27.77	231.33
p103 - p102	2.18	18.16
p102 - p101	2.32	19.34
p101 - p104	6.96	57.95
p104 - p98	11.98	99.80
p98 - p97	7.18	59.81
p97 - P94. Instituto Investigaciones Biomedicas	5.22	43.44
P94. Instituto Investigaciones Biomedicas - p40	10.95	91.20
p40 - p_x899y338	4.61	38.40
p_x899y338 - P39yP91. Parada TUNAM Fac. Quimica	8.93	74.40
P39yP91. Parada TUNAM Fac. Quimica - P90. Entrada Est. Facultad de Ingenieria	3.79	31.54
P90. Entrada Est. Facultad de Ingenieria - p38	12.69	105.70
p38 - p37y88	7.51	62.57
p37y88 - P44y87. Entrada a Edificio Principal Facultad de Ingenieria	5.77	48.06
	Total de distancia	**1,927.82**
Red: **Automovil**, Velocidad Promedio: **8.33** (m/s)	**Tiempo de recorrido**	**0h 3m 51s**

Consultar de nuevo

El sistema encuentra que la distancia mas corta para esta consulta es **1927.82 mts**. y el tiempo de recorrido suponiendo que se tiene una velocidad promedio de **8.33 m/s**. es de **3** minutos **51** segundos.

Análisis de resultados

De las pruebas sobre el sistema

- En ambos resultados, si se realizan combinaciones de sumas de las distancias de los demás tramos disponibles, se comprueba que la ruta generada es la más corta.

De la funcionalidad del sistema

- Aun con las carencias en la calidad de datos, se puede ver que el sistema funciona y es útil.
- Fue importante que la variable de análisis haya sido distancia y no el tiempo de recorrido pues este último es muy variable. Independientemente de la calidad de datos de la velocidad o tiempo de recorrido, si las distancias son correctas, el sistema siempre proporcionará los valores correctos, y el tiempo de recorrido se convierte en un dato de referencia.
- El módulo de mas demanda en funcionalidad es el de mantenimiento, pues debe permitir que la captura y modificación de datos sea muy ágil debido a el gran volumen de datos que debe introducirse para que el sistema pueda proporcionar cualquier solución.
- Uno de los días de la recopilación de datos se hizo en día domingo. Para la red de peatón se encontró que el acceso del Anexo de Ingeniería sobre el camino verde estaba cerrado, para tomar la lectura se hizo una pausa y se continuó cuando se estaba del otro lado de la reja. Esta situación de acceso cerrado se repitió para el recorrido por automóvil a la altura de la Escuela de Trabajo Social. Estas situaciones plantean la necesidad de deshabilitar ciertos puntos en ciertos días para obligar a que el sistema proporcione la mejor solución siguiente.

El sistema podría ampliarse para que:

- Proporcione la solución de varias rutas, las mejores opciones.
- Proporcione la solución combinada de rutas, una parte en auto otra a pie. Por ejemplo cuando no traemos auto, usamos el transporte UNAM para acercarnos y el resto lo cubrimos a pie.
- Proporcione una ruta cubriendo ciertos puntos de interés.
- Se pueda deshabilitar un punto para que no se pueda pasar por este, dadas ciertas condiciones.
- La velocidad de desplazamiento, que es un dato aproximado pueda hacerse mas real asignando velocidades de acuerdo a la zona y horario en caso de automóvil.
- A cada punto pueda asociarse información adicional como fotos, links, comentarios. El diseño de la base soporta esa ampliación.
- Si se diseña una forma en que el alta de datos se realice por una comunidad interesada (al estilo wikipedia) podría acelerarse la tarea de carga de datos del sistema. Para ello habría que migrar el módulo de mantenimiento a Web para que fuera accesible desde el Internet.

Conclusiones

El sistema desarrollado combina un amplio conjunto de conocimientos, en gran medida de desarrollo de sistemas, que son parte de la carrera Ingeniería en Computación, conocimientos de matemáticas y de ciencias básicas.

Puede verse que ante una necesidad de información si se combina estos conocimientos con técnica y recursos puede crearse un sistema de información que resuelva el problema, aplicando soluciones sencillas y que existen desde hace mucho tiempo. La creatividad en este tipo de sistemas es esencial pues puede hacer que una solución útil sea atractiva si se hace uso de otros recursos.

El algoritmo de Dijkstra es uno de muchos algoritmos diseñados para la resolución de problemas de teoría de redes. Como este, existen muchos otros que resuelven de manera eficiente un problema específico. Dijkstra no es el más eficiente de los algoritmos para la búsqueda de rutas mas cortas en ciertos casos, pero es uno de los más didácticos y para este sistema es suficiente ya que demuestra sin ningún problema su objetivo.

Al principio del trabajo se formulo el objetivo de crear un medio de consulta que proporcione la ruta mas corta entre 2 puntos de Ciudad Universitaria. Hasta este punto se tiene el sistema terminado y una pequeña cantidad de datos registrados en el sistema, de la que se hablo oportunamente en los alcances.

Dados los resultados expresados en esta última sección puedo concluir que el objetivo esta cubierto pues se tiene un sistema que muestra en forma de texto y de manera gráfica el resultado de la ruta más corta entre dos puntos que pueden seleccionarse de un catálogo; este catálogo de puntos y tramos puede ampliarse y modificarse en cualquier momento.

Finalmente, es importante aplicar los conocimientos adquiridos a nuestro paso por la carrera en beneficio de nuestra Universidad. Como estudiantes contamos con conocimientos, tiempo y con la experiencia de nuestros profesores para guiar nuestras capacidades. La Universidad Nacional Autónoma de México siempre tendrá la necesidad de productos útiles para ella, así pues las bases están dadas, es solo cuestión de creatividad y de animarse a retribuirle un poco de lo mucho que nos ha dado.

ANEXO A - TEORÍA DE GRAFOS

Definiciones

En matemáticas y ciencias de la computación, la teoría de grafos estudia las propiedades de los grafos, que son colecciones de objetos llamados vértices (o nodos) conectados por líneas llamadas aristas (o arcos) que pueden tener orientación (dirección asignada). Típicamente, un grafo está diseñado por una serie de puntos (los vértices) conectados por líneas (las aristas).

Grafo

Un grafo es una pareja G = (V, A), donde V es un conjunto de puntos, llamados vértices, y A es un conjunto de pares de vértices, llamadas aristas. Para simplificar, notaremos la arista {a, b} como ab.

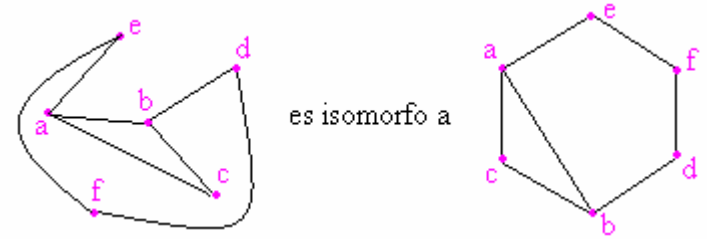


Fig. Equivalencia de un grafo

En la figura, V = { a, b, c, d, e, f }, y A = { ab, ac, ae, bc, bd, df, ef }.

En teoría de grafos, sólo queda lo esencial del dibujo: la forma de las aristas no son relevantes, sólo importa a qué vértices están unidas. La posición de los vértices tampoco importa, y se puede variar para obtener un grafo más claro. Generalmente, se considera que colocar los vértices en forma de polígono regular da grafos muy legibles.

Prácticamente cualquier red puede ser modelada con un grafo: una red de carreteras que conecta ciudades, una red eléctrica o un alcantarillado.

Aristas dirigidas y no dirigidas

En algunos casos es necesario asignar un sentido a las aristas, por ejemplo, si se quiere representar la red de las calles de una ciudad con sus inevitables direcciones únicas. El conjunto de aristas será ahora un subconjunto de todos los posibles pares ordenados de vértices, con (a, b) ≠ (b, a). Los grafos que contienen aristas dirigidas se denominan **grafos orientados o dirigidos**, como el siguiente:

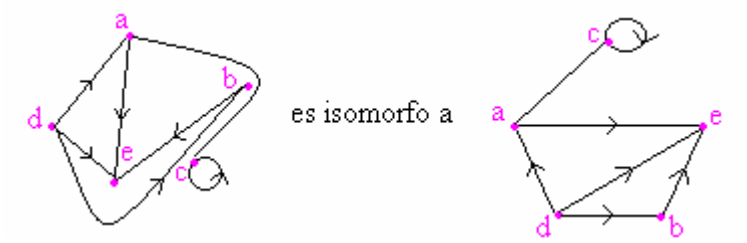


Fig. Grafo Orientado, Dirigido o Dirigido

Las aristas no orientadas se consideran bidirigidos para efectos prácticos (equivale a decir que existen dos aristas orientadas entre los nodos, cada una en un sentido).

En el grafo anterior se ha utilizado una arista que tiene sus dos extremos idénticos: es un lazo (o bucle), y aparece también una arista bidirigido, y corresponde a dos aristas orientadas.

Aquí V = { a, b, c, d, e }, y A = { **(a, c)**, (d, a), (a, e), (b, e), **(c, a)**, **(c, c)**, (d, b) }.

Según la terminología seguida en algunos problemas clásicos de Investigación de Operaciones (p.ej.: el *Problema del flujo máximo*), a un vértice del que sólo salen aristas se le denomina *fuente* (en el ejemplo anterior, el vértice *d*). Por el contrario, a aquellos en los que sólo entran aristas se les denomina *pozo* o *sumidero* (en el caso anterior, el vértice *e*).

Ciclos y caminos Hamiltonianos

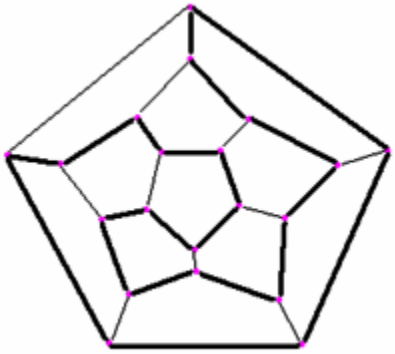


Fig. Ejemplo de un camino hamiltoniano

Un **ciclo** es un camino, es decir una sucesión de aristas adyacentes, donde no se recorre dos veces la misma arista, y donde se regresa al punto inicial. Un **ciclo hamiltoniano** tiene además que recorrer todos los vértices.

Por ejemplo, en un museo grande (al estilo del Louvre), lo idóneo sería recorrer todas las salas una sola vez, esto es buscar un ciclo hamiltoniano en el grafo que representa el museo (los vértices son las salas, y las aristas los corredores o puertas entre ellas).

Se habla también de **camino hamiltoniano** si no se impone regresar al punto de partida, como en un museo con una única puerta de entrada. Por ejemplo, un caballo puede recorrer todas las casillas de un tablero de ajedrez sin pasar dos veces por la misma: es un camino hamiltoniano.

Hoy en día, no se conocen métodos generales para hallar un ciclo hamiltoniano, siendo la búsqueda por fuerza bruta de todos los posibles caminos u otros métodos excesivamente costosos. Este problema entra en el conjunto de los NP-completos.

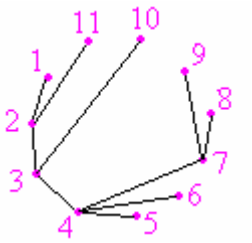


Fig. Ejemplo de árbol.

Árboles

Un grafo que no tiene ciclos y que conecta a todos los puntos, se llama un **árbol**. En un grafo con **n** vértices, los árboles tienen exactamente **n - 1** aristas, y hay $\mathbf{n^{n-2}}$ árboles posibles. Su importancia radica en que los árboles son grafos que conectan vértices utilizando el menor número posible de aristas. Un importante campo de aplicación de su estudio se encuentra en el análisis filogenético, el de la filiación de entidades que derivan unas de otras en un proceso evolutivo, que se aplica sobre todo a la averiguación del parentesco entre especies; aunque se ha usado también, por ejemplo, en el estudio del parentesco entre lenguas.

Grafos ponderados

En muchos casos, es preciso atribuir a cada arista un número específico, llamado *valuación*, *ponderación* o *coste* según el contexto, y se obtiene así un **grafo valuado**. Formalmente, es un grafo con una función v: $A \rightarrow \mathbf{R}_+$.

Por ejemplo, un representante comercial tiene que visitar *n* ciudades conectadas entre sí por carreteras; su interés previsible será minimizar la distancia recorrida (o el tiempo, si se pueden prever atascos). El grafo correspondiente tendrá como vértices las ciudades, como aristas las carreteras y la valuación será la distancia entre ellas.

Y, de momento, no se conocen métodos generales para hallar un ciclo de valuación mínima, pero sí para los caminos desde *a* hasta *b*, sin más condición.

Grafos planos

Cuando un grafo o multigrafo se puede dibujar en un plano sin que dos segmentos se corten, se dice que es plano.

Un juego muy conocido es el siguiente: Se dibujan tres casas y tres pozos. Todos los vecinos de las casas tienen el derecho de utilizar los tres pozos. Como no se llevan bien en absoluto, no quieren cruzarse jamás. ¿Es posible trazar los nueve caminos que juntan las tres casas con los tres pozos sin que haya cruces?

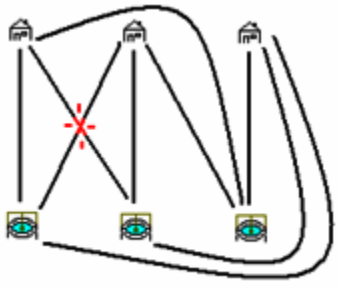


Fig. Grafo del juego de 3 casas y 3 pozos.

Un grafo es **plano** si se puede dibujar sin cruces de aristas.

Cualquier disposición de las casas, los pozos y los caminos implica la presencia de al menos un cruce.

Se nota K_n el **grafo completo** con *n* vértices, es decir en el cual cada par de vértices están conectados por una arista. $K_{n,p}$ es el grafo compuesto de un grupo de *n* vértices y otro de *p*, tal que cada vértice del primer grupo está conectado con cada uno del segundo, y no hay más aristas.

El juego anterior equivale a descubrir si el grafo $K_{3,3}$ es **planario** (se dice también **plano**), es decir, si se puede dibujar en un plano sin que haya cruces. Y la respuesta es no.

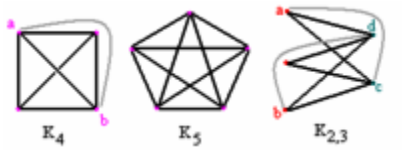


Fig. Grafos planos.

En la figura se nota que K_4 es plano (desviando la arista *ab* al exterior del cuadrado), que K_5 no lo es, y que $K_{3,2}$ lo es también (desvíos en gris).

Establecer qué grafos son planos no es obvio, y tiene que ver con la topología.

Diámetro

En un grafo, la distancia entre dos vértices es el menor número de aristas de un recorrido entre ellos. El diámetro, en una figura como en un grafo, es la mayor distancia entre dos puntos de la misma.

El diámetro de los Kn es 1, y el de los Kn,p es 2. Un diámetro infinito puede significar que el grafo tiene una infinidad de vértices o simplemente que no es conexo. También se puede considerar el diámetro promedio, como el promedio de las distancias entre dos vértices.

El mundo de Internet ha puesto de moda esa idea del diámetro: Si descartamos los sitios que no tienen enlaces, y escogemos dos páginas Web al azar: ¿En cuántos clics se puede pasar de la primera a la segunda? El resultado es el diámetro de la Red, vista como un grafo cuyos vértices son los sitios, y cuyas aristas son lógicamente los enlaces.

En el mundo real hay una analogía: tomando al azar dos seres humanos del mundo, ¿En cuántos saltos se puede pasar de uno a otro, con la condición de sólo saltar de una persona a otra cuando ellas se conocen personalmente? Con esta definición, se estima que el diámetro de la humanidad es de ocho solamente.

Aplicaciones

Gracias a la teoría de Grafos se pueden resolver diversos problemas como por ejemplo la síntesis de circuitos secuenciales, contadores o sistemas de apertura.

Los grafos se utilizan también para modelar trayectos como el de una línea de autobús a través de las calles de una ciudad, en el que podemos obtener caminos óptimos para el trayecto aplicando diversos algoritmos como puede ser el algoritmo de Floyd.

Para la administración de proyectos, utilizamos técnicas como PERT en las que se modelan los mismos utilizando grafos y optimizando los tiempos para concretar los mismos.

Notaciones Matemáticas

Definiciones

Un grafo puede ser dirigido o no dirigido.

Un **grafo dirigido** G es un par (V, A):
• V es un conjunto finito de **vértices**

• A es un conjunto de **aristas** que representa una relación binaria sobre V - A $\subseteq$ V x V

En un **grafo no dirigido** G = (V, A), el conjunto de aristas A consiste en pares no ordenados de vértices:

Una arista es un conjunto {u, v}, en que u, v $\in$ V y u $\neq$ v, y que representamos como (u, v) o (v, u)

Si (u, v) es una arista en un grafo dirigido
- (u, v) es incidente desde o sale del vértice u y es incidente a o entra al vértice v
- v es **adyacente** a u

Si (u, v) es una arista en un grafo no dirigido
- (u, v) es incidente sobre u y v
- v es adyacente a u y u es adyacente a v

Una **ruta** de longitud k desde un vértice u a un vértice u' en un grafo G = (V, A) es una secuencia v0, v1, ..., vk de vértices tal que
- u = v0
- u' = vk
- (vi-1, vi) $\in$ E para i = 1, 2, ..., k

Si hay una ruta p desde u a u', u' es **alcanzable** desde u vía p.

Representaciones.

Hay dos formas de representar un grafo G = (V, A):
- Una colección de listas de adyacencias, preferible para grafos ralos — |A| << $|V|^2$
- Una matriz de adyacencias, preferible para grafos densos — |A| $\approx$ $|V|^2$ — o cuando queremos saber rápidamente si hay una arista conectando dos vértices dados

Las **listas de adyacencias** de G = (V, A) son un arreglo α de |V| listas, una para cada vértice de V:

- Para cada u $\in$ V, la lista α [u] contiene (punteros a) todos los vértices v tales que hay una arista (u, v) $\in$ A — todos los vértices adyacentes a u en G, en orden arbitrario
- La suma de las longitudes de todas las listas es |A|, si G es dirigido, y 2|A| si G es no dirigido — la cantidad de memoria necesaria es siempre O(max(V, A)) = O(V + A).

La **matriz de adyacencias** de G = (V, A) supone que los vértices están numerados 1, 2, ..., |V| arbitrariamente, y consiste en una matriz A = (a_{ij}) de |V|x|V|:

a_{ij} = 1 si (i, j) $\in$ A
a_{ij} = 0 en otro caso

Esta representación requiere $\Theta(V^2)$ memoria, independiente del número de aristas en G.

Rutas Más Cortas

Definiciones.

Dado un grafo dirigido G = (V, A) y una función de longitud w : A $\rightarrow$ R, definimos

- La longitud de una ruta p = <v0, v1, ..., vk> es
- La longitud de la ruta más corta de u a v es
- Una ruta más corta de u a v es cualquier ruta p tal que w(p) = d(u, v).

Las longitudes pueden representar cualquier cantidad que se acumule linealmente a lo largo de una ruta y que queremos minimizar.

Puede haber aristas con longitudes negativas:

- Si G no contiene ciclos de longitud negativa alcanzables desde s, la longitud $\delta(s, v)$ está bien definida para todo v $\in$V, aun si su valor es negativo
- De lo contrario, ninguna ruta desde s a un vértice en el ciclo puede ser una ruta más corta—siempre es posible encontrar una ruta aún más corta que sigue la ruta propuesta y luego recorre el ciclo.

Algoritmos para Teoría de Grafos

Existen diferentes algoritmos especializados en la resolución de problemas de grafos:

- Algoritmo de Dijkstra (árbol mínimo/máximo - camino mínimo/máximo - camino crítico)
- Algoritmo de Bellman-Ford (camino mínimo - camino máximo)
- Algoritmo de Floyd-Warshall (camino mínimo entre todos los pares de nodos)
- Algoritmo de Kruskal (árbol de coste total mínimo/máximo)
- Algoritmo de Prim (árbol de coste total mínimo/máximo)
- Algoritmo de Ford-Fulkerson (flujo máximo)
- Problema del Transbordo-Transporte (coste mínimo)
- Problema de Asignación (coste mínimo)
- Problema del Viajante de Comercio (distancia total mínima)
- Problema de los m-Viajantes de Comercio (distancia total mínima)
- Algoritmo de Rutas (paso por nodos seleccionados a coste mínimo)
- Problema de los m-Rutas (distancia total mínima)
- Problema Rutas con Vehículos Capacitados (CVRP)

A continuación solo se explican los 2 algoritmos más importantes para la solución de la ruta mas corta:

Algoritmo de Dijkstra

Algoritmo de Dijkstra (ruta más corta - árbol mínimo - camino mínimo)

El problema de la ruta más corta se puede resolver utilizando programación lineal sin embargo, debido a que el método simplex es de complejidad exponencial, se prefiere utilizar algoritmos que aprovechen la estructura en red que se tiene para estos problemas.

Para ello, el algoritmo mantiene un conjunto S de nodos cuyos pesos finales de camino mínimo desde el nodo origen ya han sido determinados.

Algoritmo de Dijkstra (ruta más larga - árbol máximo - camino crítico)

El camino crítico estará formado por tareas críticas (nodos) cuya duración (coste del arco sucesor) determina la duración total de un proyecto. Si una tarea crítica se retrasa o su duración cambia durante la realización del proyecto, afectaría directamente a la duración total del proyecto y a su fecha de finalización.

Encontrar el camino crítico de la planificación de un proyecto es lo mismo que encontrar el camino más largo desde el nodo inicial (tarea inicial) al nodo final (última tarea); esto es, la mínima cantidad de tiempo necesaria para finalizar un proyecto.

El algoritmo de Dijkstra aunque fue diseñado para encontrar la ruta más corta se puede transformar fácilmente para encontrar la ruta más larga (camino crítico), cambiando simplemente su función objetivo. Del mismo modo, se encuentra el árbol máximo desde un nodo origen.

Descripción matemática

Aplicable sólo cuando las longitudes de *todas las aristas son no negativas* — $\omega(u, v) \geq 0$ para cada $(u, v) \in A$; determina las rutas más cortas desde un vértice $s \in V$ a todos los otros vértices del grafo:

- Mantiene un conjunto S de vértices para los cuales las longitudes de las rutas más cortas desde s ya han sido determinadas

  $\forall\ v \in S, d[v] = \delta(s, v)$
- Repetidamente,
  - selecciona el vértice $u \in V-S$ tal que $d[u]$ es mínimo,
  - ingresa u a S, y
  - reduce las aristas que salen de u.

La siguiente implementación

- mantiene una cola de prioridades Q que contiene todos los $u \in V-S$ priorizados según sus valores $d[u]$
- supone que G se representa mediante listas de adyacencias

DIJKSTRA(G, ω, s) :
INIT(G, s)
S = ∅
Q = V
while (Q ≠ ∅)
u = Xmin(Q)
S = S ∪ {u}
for (each $v \in \alpha[u]$)
REDUCE(u, v, ω)

INIT(G, s) :
for (each $v \in V$)
$d[v]$ = ¥
p[v] = NIL
$d[s]$ = o

REDUCE(u, v, w) :
if ($d[v] > d[u]$ + w(u,v))
$d[v] = d[u] + \omega(u,v)$
$\pi[v] = u$

- d[v] — la longitud de la ruta más corta de s a v encontrada hasta el momento
- p[v] — el vértice predecesor de v en esta ruta

¿Cuál es el desempeño de Dijkstra?

• DIJKSTRA realiza $|V|$ XMÍN y $|A|$ REDUCE
• Si Q es implementada como un *heap* binario, entonces cada una de las XMin y cada una de las actualizaciones de $d[v]$ en REDUCE toma tiempo log$V$
• DIJKSTRA toma tiempo O((V+A) logV).

Pseudo código

Dijkstra (G,s):

```
Inicializar
    for cada v perteneciente a V[G]
        do d[v] = infinito
            p[v] = nulo
    d[s] = 0
S =  vacio
Q = V[G]
mientras Q no vacio
```

```
do u = nodo v con min d[v]

S = S unión u 'se añade al conjunto de nodos finalizados
for cada v perteneciente Adyacente u
    Relajacion
        if d[v] > d[u] + w(u,v) then
            d[v] = d[u] + w(u,v)
            p(v) = u
```

Algoritmo de Bellman-Ford

Algoritmo de Bellman-Ford (camino mínimo)

Soluciona el problema de la ruta más corta o camino mínimo desde un nodo origen, de un modo más general que el Algoritmo de Dijkstra, ya que permite valores negativos en los arcos.

El algoritmo devuelve un valor booleano si encuentra un circuito o lazo de peso negativo. En caso contrario calcula y devuelve el camino mínimo con su coste.

Para cada vértice v perteneciente a V, se mantiene el atributo d[v] como cota superior o coste del camino mínimo desde el origen s al vértice v.

Algoritmo de Bellman-Ford (camino máximo)

El problema de la ruta más larga puede ser transformado en el de ruta más corta cambiando el signo de los costes de los arcos.

De manera alternativa se puede transformar también cambiando los procesos de inicialización y relajación. En este caso el problema es inconsistente para circuitos de peso positivo.

Descripción Matemática

El algoritmo de Bellman-Ford. Resuelve el problema de las rutas más cortas desde un vértice en el caso más general en el cual las longitudes de las aristas pueden ser negativas.

Devuelve un valor 0 (falso) o 1 (verdadero) que indica si hay o no un ciclo con longitud total negativa que sea alcanzable desde el vértice s:

• si lo hay, indica que no hay solución

• si no lo hay, produce las rutas más cortas y sus longitudes

BELLMAN-FORD(G, ω, s) :

INIT(G, s)

for (i = 1, i = $|V| - 1$, i++)

for (each $(u, v) \in E$)

REDUCE(u, v, ω)

for (each (u, v) $\in$ E)

if (d[v] > d[u] + ω(u, v)) return 0

return 1

BELLMAN-FORD corre en tiempo $O(V\ E)$: cada una de las $|V| - 1$ pasadas por las aristas toma tiempo $O(E)$.

Si G no contiene ciclos con longitud negativa alcanzables desde s:

- sea v alcanzable desde s, y sea $p = \langle v0, v1, ..., vk \rangle$ una ruta (simple) más corta de s a v, en que $v0 = s$, $vk = v$ y $k \leq |V| - 1$
- como $d[v0] = \delta(s, v0) = 0$ después de la inicialización, suponemos que $d[vi-1] = \delta(s, vi-1)$ después de la pasada $i-1$
- como la arista $(vi-1, vi)$ es reducida en la pasada i, $d[vi] = \delta(vi-1, vi)$ después de la pasada i

Si G contiene un ciclo $c = \langle v0, v1, ..., vk \rangle$, en que $v0 = vk$, con longitud alcanzable desde s,

- Supongamos que BELLMAN-FORD devuelve 1, es decir,
- $d[vi] \leq d[vi-1] + \omega(vi-1, vi)$, para i = 1, 2, ..., k
- Sumando estas desigualdades alrededor del ciclo c, tenemos
- Pero como entonces lo que contradice la conclusión de la hipótesis
- Luego, si G contiene un ciclo con longitud negativa alcanzable desde s, entonces BELLMAN-FORD devuelve 0

Pseudo código

```
Bellman-Ford (G,s):
    Inicializar
        for cada v perteneciente a V[G]
            do d[v] = infinito
                p[v] = nulo
        p[s] = 0
    for i=1 to V[G]-1
        do for cada arco (u,v) perteneciente a A[G]
            Relajación
            if d[v] > d[u] + w(u,v) then
                d[v] = d[u] + w(u,v)
                p(v) = u
    for cada arco (u,v) chequea lazo de peso negativo
        do if d[v] > d[u] + w(u,v) then
            return FALSO 'el algoritmo no converge
    return VERDADERO
```

```
Inicializar
    for cada v perteneciente a V[G]
        do d[v] = - infinito
           p[v] = nulo
    p[s] = 0

Relajacion:
        if d[v] < d[u] + w(u,v) then
            d[v] = d[u] + w(u,v)
            p(v) = u
```

Eficiencia de Algoritmos

- El algoritmo de ruta mas corta sin costos es $O(|A| + |V|)$.
- El costo del algoritmo de Dijkstra depende de cómo se implemente la lista de vértices:
  - Si se recorre la lista de vértices en secuencia cada vez que se busca el v.dist mínimo el algoritmo es $O(|A|+|V|^2)$
  - Si se use una cola de prioridad el algoritmo es $O(|A|\log |V|)$ o $O(|V|\log |V|)$.
- Si se permiten costos negativos el algoritmo es $O(|A|\cdot|V|)$

Biografía de Edsger Dijkstra

Fig. Edsger Wybe Dijkstra.

Edsger Wybe Dijkstra[1] nació en Rotterdam, (Holanda) en 1930. Sus padres eran ambos intelectuales y él recibió una excelente educación. Su padre era químico y su madre matemática. En 1942, cuando Dijkstra tenía 12 años, entró en Gymnasium Erasminium, una escuela para estudiantes especialmente brillantes, donde dio clases, fundamentalmente, de Griego, Latín, Francés, Alemán, Inglés, biología, matemáticas y química. En 1945, Dijkstra pensó estudiar Derecho y trabajar como representante de Holanda en las Naciones Unidas.

Sin embargo, debido a su facilidad para la química, las matemáticas y la física, entró en la Universidad de Leiden, donde decidió estudiar física teórica. Durante el verano de 1951, asistió a un curso de verano sobre programación en la Universidad de Cambridge. A su vuelta empezó a trabajar en el Centro Matemático en Amsterdam, en marzo de 1952, donde se incrementó su creciente interés en la programación. Cuando terminó la carrera se dedicó a problemas relacionados con la programación. Pero uno de los problemas con que se encontró es que ser programador no estaba oficialmente reconocido como una profesión. De hecho, cuando solicitó una licencia de matrimonio en 1957, tuvo que señalar que su profesión era físico teórico.

Dijkstra continuó trabajando en el Centro Matemático hasta que aceptó un trabajo como desarrollador en Burroughs Corporation, en los Estados Unidos, a principio de la década de los 70. En 1972 ganó el Premio Turing ACM, y ,en 1974, el AFIPS Harry Good Memorial. Dijkstra se trasladó a Austin, Texas a principio de los 80. En 1984, se le ofreció un puesto en Ciencias de la Computación en la Universidad de Texas, donde ha permanecido desde entonces. Es miembro honorario de la Academia Americana de Artes y Ciencias y de Real Academia Holandesa de Artes y Ciencias. Además es miembro distinguido de la Sociedad de Computación Británica. Finalmente es Doctor Honoris Causa en Ciencias por la Queen's University Belfast.

En 1956, Dijkstra anunció su algoritmo "***Algoritmo de caminos mínimos***", después de haber estado trabajando con el ARMAC, el ordenador que el Centro Matemático poseía. Más tarde propuso el

[1] Fuente: http://www.alumnos.unican.es/uc900/Algoritmo.htm

algoritmo del árbol generador minimal. A principios de la década de los 60, Dijkstra aplicó la idea de la exclusión mutua a las comunicaciones entre una computadora y su teclado. Su solución de exclusión mutua ha sido usada por muchos procesadores modernos y tarjetas de memoria desde 1964, cuando IBM la utilizó por primera vez en la arquitectura del IBM 360. El siguiente problema del que se ocupó Dijkstra fue el de los filósofos comensales. En este problema, cinco filósofos están sentados en una mesa circular con un plato de arroz delante y un palillo a cada lado, de manera que hay cinco palillos en total. El problema trata sobre el uso de recursos comunes sin que los procesos (los filósofos) lleguen a una situación de bloqueo mutuo, inanición y que los recursos sean usados de la manera más eficiente por todos los procesos. También ayudó a fomentar la disciplina en la programación: "GOTO se puede considerar dañino. Cuanto más sentencias GOTO haya en un programa, más difícil es entender el código fuente".

ANEXO B - GLOSARIO

Adjacency Matrix — En matemáticas y ciencia de la computación, la matriz de adyacencia para una gráfica finita G en vértices n es una matriz de n x n donde la entrada no diagonal a_{ij} es el número de arcos que unen al vértice i y j, y la entrada diagonal a_{ii} ya sea el doble del número de uniones en el vértice i o el número de uniones.

ADO — Es un API para acceso a datos de Microsoft

API — Interface de Programación de Aplicaciones, es un conjunto de funciones específicas proporcionadas por el fabricante para crear aplicaciones sin partir de 0.

Applet — Un componente java que se ejecuta generalmente en un Web browser, pero puede ser ejecutado en una variedad de otras aplicaciones o dispositivos que soporten el modelo de programación del applet.

Archivo properties — Archivo de propiedades que sirve a las aplicaciones java para guardar los valores de configuración de la aplicación.

Cola de prioridad — Una cola de prioridad es un tipo de dato abstracto que soporta las siguientes operaciones:

- Agregar un elemento a la cola con una prioridad asociada
- Remover el elemento de la cola que tiene la prioridad mas alta y regresarlo
- Opcionalmente, consultar el elemento con la prioridad mas alta sin removerlo

Component — Es una unidad de software a nivel aplicación soportada por un contenedor.

Framework — Es un conjunto de clases e interfaces que definen una arquitectura a seguir, y un conjunto de librerías de soporte para el desarrollo rápido de aplicaciones.

HTML — Hypertext Markup Language. Un lenguaje de marcas para los documentos con hipervínculos de Internet. HTML posibilita la incorporación de imagines, sonido, video, campos de captura y referencias a otros objetos con URL y formateo básico de texto.

HTTP — Hypertext Transfer Protocol. Es el protocolo de transferencia usado para la comunicación entre clientes y servidores. Los mensajes http consisten de peticiones del cliente al servidor y respuestas del servidor al cliente.

Java Server Pages — La tecnología Java Server Pages (JSP) permite a los diseñadores y desarrolladores de sitios Web crear rápidamente y mantener de manera sencilla las páginas Web dinámicas, ricas en información sobre las que se basan los sistemas de negocio. La tecnología JSP separa la interface de usuario de la generación de contenidos, permitiendo a los diseñadores cambiar el formato de la página sin alterar el contenido dinámico.

JDBC La tecnología JDBC (Java DataBase Connectivity) es un API que permite acceder desde el lenguaje de programación Java, a virtualmente cualquier fuente de datos tabulados. Proporciona conectividad cruzada DBMS a un amplio rango de bases de datos SQL, así como otras fuentes de datos tabulados, como hojas de cálculo o simples archivos.

J2EE JavaTM 2, Enterprise Edition. Es la versión empresarial de Java con soporte a diversas APIs para transacciones, localización de servicios, programación con componetes Web y de negocio EJBs

J2SE JavaTM 2, Standard Edition.

J2EE application Cualquier unidad instalable de funcionalidad J2EE. Este puede ser un módulo simple empaquetado en un archivo EAR con un descriptor de despliegue. Las aplicaciones J2EE son desarrolladas para se distribuidas a través de múltiples capas.

JAR JAR Java ARchive Es un formato de archive independiente de la plataforma que permite que muchos archivos sean agregados en un solo archive.

JSP JavaServer Pages. Una tecnología web extensible que usa plantillas de datos, elementos personalizados, lenguajes script, y objetos java del lado del servidor para regresar contenido dinámico al cliente. Típicamente la plantilla es HTML o XML.

ODBC ODBC son las siglas de Open DataBase Connectivity, un estándar de acceso a Bases de Datos desarrollado por Microsoft Corporation, el objetivo de ODBC es hacer posible el acceder a cualquier dato de cualquier aplicación, sin importar qué Sistema Gestor de Bases de Datos (DBMS por sus siglas en Ingles) almacene los datos, ODBC logra esto al insertar una capa intermedia llamada manejador de Bases de Datos, entre la aplicación y el DBMS, el propósito de esta capa es traducir las consultas de datos de la aplicación en comandos que el DBMS entienda

Open Source Código abierto (del inglés open source) es el término por el que se conoce al software distribuido y desarrollado en forma libre.

Este término empezó a utilizarse en 1998 por algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original en inglés del software libre (free software).

Free en inglés puede significar diferentes cosas: gratuidad y libertad. Por ello, por un lado, permite pensar en "software por el que no hay que pagar" (software gratuito) y, por otro, se adapta al significado que se pretendió originalmente (software que posee ciertas libertades).

El término para algunos no resultó apropiado como reemplazo para el ya tradicional free software, pues eliminaba la idea de libertad (incluso hay algunos que usan —en inglés— el término libre software para evitar la ambigüedad de free).

Desde el punto de vista de una "traducción estrictamente literal", el significado obvio de "código abierto" es que "se puede mirar el código fuente", por lo que puede ser interpretado como un término más débil y flexible que el del software libre.

Patrón de Diseño En ingeniería de software, un patrón de diseño en una solución general y repetible a un problema común de diseño de software.

	Un patrón de diseño no es un diseño terminado que puede ser transformado directamente en código, es una descripción o plantilla de cómo resolver un problema que puede ser usado en muchas situaciones. El diseño de patrones orientado a objetos, muestra la relación entre clases u objetos, sin especificar las clases finales de la aplicación o los objetos involucrados.
RUP	Rational Unified Process. Proceso de desarrollo de software creado por la OMG que propone el desarrollo de proyectos bajo 4 premisas: Iterativo, Incremental, basado en UML y en artefactos o entregables.
SQL	Structured Query Language. Es el lenguaje estandarizado de bases de datos relacionales para la definición de objetos, manipulación de datos y control de acceso a objetos de la base de datos.
WAR file	Es un archivo JAR que contiene un módulo Web.
Web Component	Es un componente que provee servicios en respuesta a una petición ya sea un servlet o un jsp.
XML	eXtensible Markup Language. Un lenguaje de marcas que permite definir los tags (markup) necesarios para definir los datos y los textos en un documento XML.

BIBLIOGRAFÍA

Libros

Análisis y Diseño Orientado a Objetos con UML y el Proceso Unificado
Stephen R. Schach
Mc Graw Hill, México DF, 2005

Struts in Action, Building Web applications with the lead java framework
Ted Husted, et alt
Manning, USA 2003

Core Servlets & Java Server Pages
Marty Hall
Sun Microsystems, USA 2004

Sitios en Internet

Teoría de Grafos
http://personales.upv.es/arodrigu/grafos/mindmap_grafos.htm
http://es.wikipedia.org/wiki/Teor%C3%ADa_de_grafos
http://personales.upv.es/arodrigu/grafos/funciones.htm

Edsger Dijkstra
http://www.alumnos.unican.es/uc900/Algoritmo.htm

Implementación de Algoritmo de Dijkstra, pagina de Jean Michel Garnier
http://rollerjm.free.fr/pro/graphs.html

Definiciones técnicas
http://es.wikipedia.org/wiki/Javapedia

Tecnología Java
http://www.iec.csic.es/CRIPTONOMICON/java/quesjava.html
http://peterpan.uc3m.es/docencia/p_s_ciclo/iu/practicas/p7/IntroSwing.pdf
http://www.javahispano.org/
http://java.sun.com/products/jsp/tomcat/

Java Server Faces
http://www.programacion.com/java/tutorial/jsf_intro/1/
http://Websphere.sys-con.com/read/46516.htm

Bases de datos
http://www.netpecos.org/docs/mysql_postgres/x15.html
http://www.mysql.com

Python
http://www.planetacodigo.com/wiki/guia:python
http://www.python.org/

JBoss
http://www.osmosislatina.com/jboss/

Información acerca de la UNAM
http://www.unam.mx
http://www.edumexico.org/em/apps/universidad.php?uniname=unam