**Trabajo ISFPP Programación Orientada a Objetos**

Alumno: Agustín Juárez

**Introducción**

Este trabajo se trata acerca de la simulación de un sistema de Líneas de subtes.

Dicho trabajo le brinda al usuario varios trayectos para llegar a su destino: el camino más rápido, el que tiene menos trasbordos entre estaciones de diferentes líneas y el que tiene menos congestión de gente.

El proyecto usa como ejemplo el sistema de subte de la ciudad de Buenos Aires, pero se construyó para que se pueda utilizar cualquier sistema de subte que exista en la actualidad. Las Líneas que la constituyen se denominarán A, B, C, D, E, H.

**Planteo del problema**

La problemática que se buscó solucionar con este programa era el de trazar la trayectoria que el usuario desee de una estación origen a una estación destino a través de una simulación de la red de Subtes de la ciudad de Buenos Aires, brindándole el mejor camino para llegar hasta ello.

**Análisis de las estructuras seleccionadas**

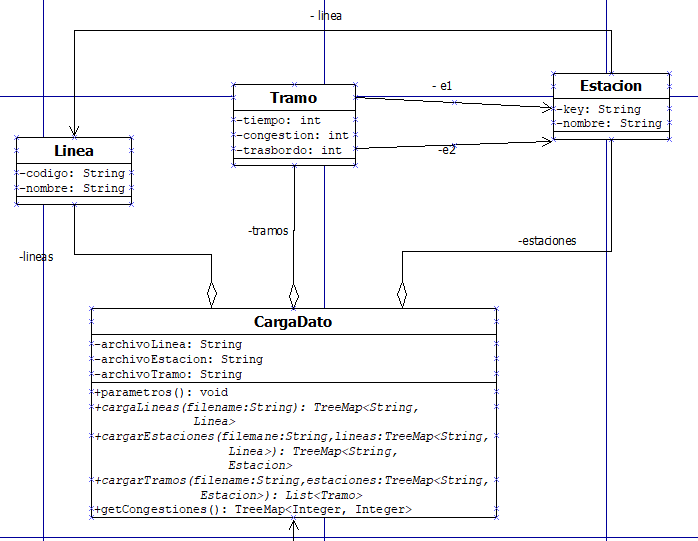
Para simular la red de líneas de los subtes se utiliza la estructura de un Grafo compuesto por sus vértices (estaciones), relacionados cada uno de ellos con arcos (tramo de una estación a otra), el cual contiene los valores con los cuales se trabaja en las soluciones de los problemas hablados anteriormente.

Como bien se dijo, se utiliza la estructura de un Grafo para la simulación. Los vértices de este están compuestos por las Estaciones y los pesos de los arcos son la variable que se requiera evaluar a la hora de elegir la opción que el usuario desea obtener.

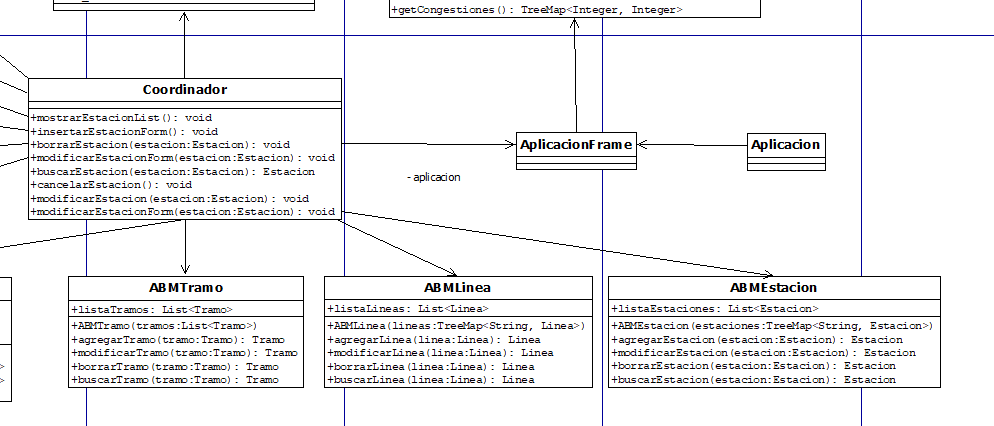
Por otro lado, se utiliza la estructura de los Mapas. El funcionamiento del Mapa es el de guardar datos, los cuales están asociadas a una clave. Se utiliza para guardar la información de las estaciones, las líneas y también para los valores de los coeficientes de la congestión.

También esta implementada la estructura de lista, ya que el método shortestPathList retorna como resultado una lista posicional de las estaciones que comprender el resultado de la opción elegida por el usuario.

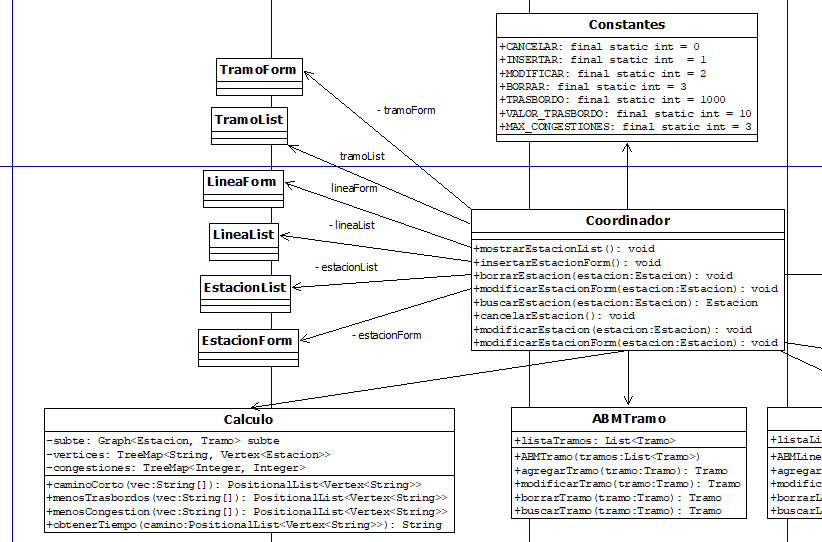
**Diagrama de clases**



*Diagrama de clases UML Parte 1*



*Diagrama de clases UML Parte 2*



*Diagrama de clases UML Parte 3*

**Implementación de la solución**

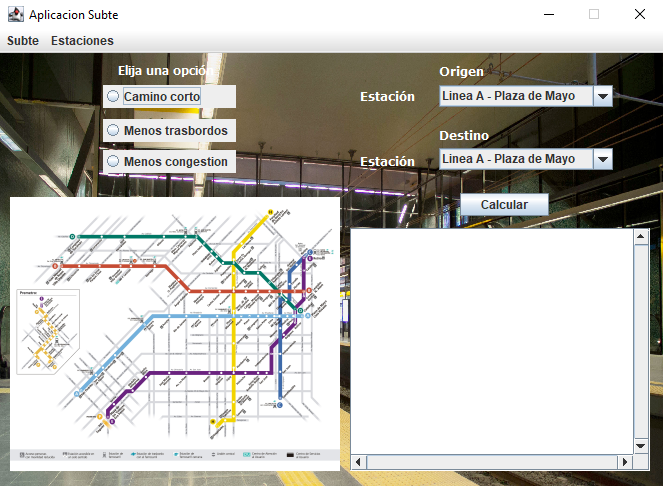
Para la solución del problema se creó un programa que satisfaga la necesidad del usuario para elegir el camino que desee. Los datos que se utilizan comienzan guardados en archivos. El archivo de “estaciones.txt” contiene el nombre, la línea a la cual pertenece y el código correspondiente que lo identifica. El archivo líneas contiene el código de la línea y la letra asociada a cada una. El archivo tramo contiene los extremos (estaciones) de cada tramo que conforma el Grafo. El peso de cada tramo estará constituido dependiendo de la opción elegida por el usuario.

Cuando ya se cargan todos los datos de los archivos en los TreeMap, se pasan estos mapas a la clase que contiene la lógica del programa, Calculo. Esta clase contiene los métodos que construyen los grafos con sus correspondientes pesos en los arcos dependiendo de la opción que elija el usuario. Los métodos que contiene son caminoCorto, que calcula la trayectoria con menos tiempo de la estación origen a la estación destino. El método menosTrasbordos construye la trayectoria que contenga menos combinaciones de líneas entre la estación origen y la estación destino. Y, por último, la menosCongestion, donde calcula el camino con menos congestión de gente entre las estaciones nombradas anteriormente.

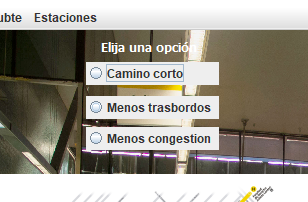
Todos los métodos mencionados anteriormente utilizan el algoritmo de Dijkstra, que, dado un grafo, calcula el camino entre vértices con la trayectoria entre los tramos que contengan el menor peso. Dicho grafo entregado al algoritmo de Dijkstra son construidos dentro de cada uno de los métodos anteriores.

**Manual de funcionamiento**

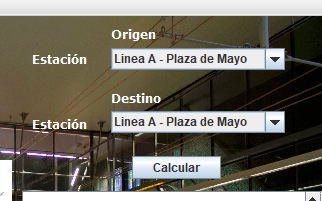
Al iniciar el programa se muestra una pantalla que contiene todos los elementos que necesitara el usuario para saber el recorrido que desee.



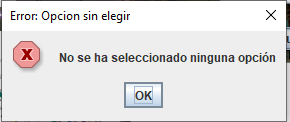
Lo primero que se verá, será una lista de las opciones de recorrido que podrá elegir el usuario. Cada opción se encuentra en la parte superior izquierda de la aplicación.



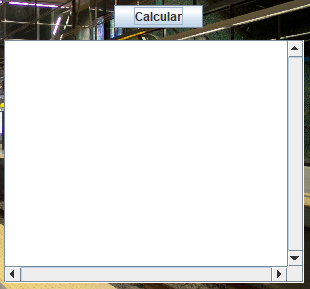
Luego de marcar la opción, el usuario deberá elegir las estaciones correspondientes: la estación origen y la de destino.



NOTA: si el usuario no elige una opción de las anteriores dichas, se le aparecerá un cartel de error recordándole que debe seleccionar uno. Si no, no se hará ninguna operación.

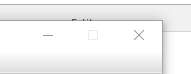


Seguido de apretar en el botón ‘Calcular’, se muestra la respuesta debajo del botón, que se ubica una pantalla en blanco, donde mostrará una concatenación de los nombres de las estaciones con los respectivos tiempos de demora que se ven involucrados en la solución de la trayectoria que se debería hacer.



El usuario podrá volver a hacer cualquier otra consulta que necesite.

Si desea cerrar el programa, tendrá que hacer clíck en la esquina superior derecha, en donde se encontrara el botón de cierre de la aplicación.



Otra manera de salir de la aplicación, es a través del menú ubicado en la parte superior. Al hacer clíck en Subte y luego en Salir (Subte ---> Salir), se cerrará el programa.

**Errores detectados**

1. Cuando se va a la lista de estaciones y se quiere modificar una estacion de la lista, no aparecera el formulario para modificarlo y la fila que se queria modificar, en la seccion de Linea, cambiara al nombre de referencia guardada en la memoria.

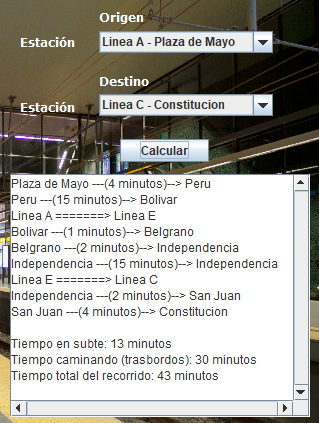


1. Al querer borrar una fila que contiene una estación, y se aprieta en el botón que contiene el icono de X, se eliminará automáticamente y no saldrá el formulario para asegurarse de que si realmente se quiere eliminar la fila de la estación.
2. Se tiene que clickear dos veces en los RadioButton para seleccionar otra opción cuando ya hay una opción seleccionada.

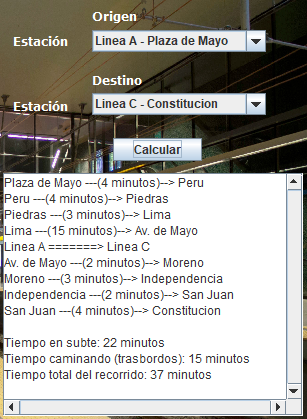
**Lotes de prueba**

Ejemplo 1: Origen: Plaza de Mayo – Línea A

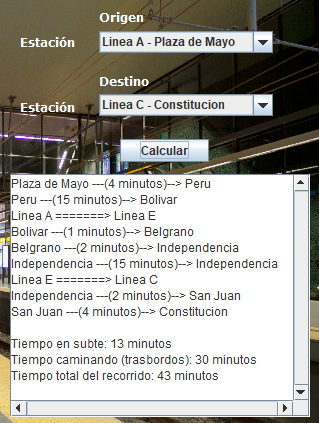
Destino: Constitución – Línea C



Camino corto



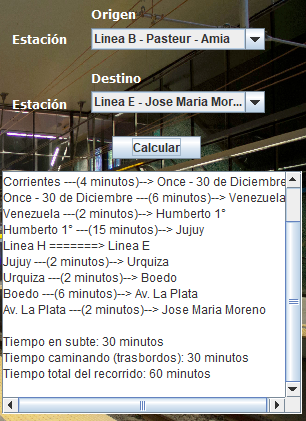
Menos trasbordos



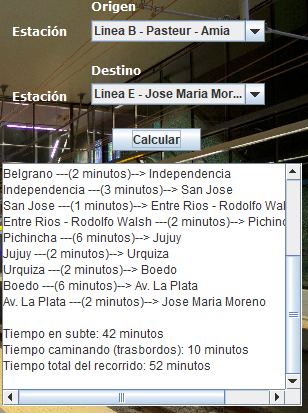
Menos congestión

Ejemplo 2: Origen: Pasteour – Amia – Línea B

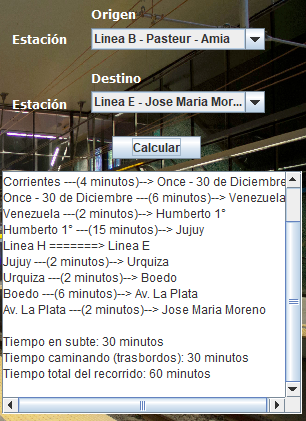
Destino: José María Moreno – Línea E



Camino Corto



Menos trasbordos



Menos congestión

**Posibles mejoras y extensiones**

Se tiene para futuro que las soluciones aparezcan en una ventana emergente y no en un panel de texto debajo del botón ‘Calcular’.

Otra posible mejora es usar una base de datos para la carga de los datos y para las altas, bajas y modificaciones de los mismos, para dejar de hacer uso de los archivos.

**Conclusión**

Al principio no sabía por qué parte empezar, ya que se nos pedía varias cosas a implementar, pero a medida que iba avanzando se armaba el trabajo con sus correspondientes implementaciones.

Lo que más costo al principio fue hacer uso de archivos, ya que era la primera vez que lo usaba. Otras complicaciones aparecieron en los métodos para construir los caminos con el uso de grafos y mapas.

A medida que avancé con el proyecto, hice entregas al profesor Gustavo Samec. Con las devoluciones que me entregaba, pude ir corrigiendo los errores que encontraba con indicaciones que me ofrecía.

En cuanto a las estructuras utilizadas para hacer la aplicación, se utilizaron Maps para cargar las estaciones, los nombres de las líneas y el valor de las congestiones que la aplicación necesita para su funcionamiento. Los tramos se crean después de cargar las estaciones y son guardados en un ArrayList, con sus correspondientes estaciones relacionadas. Gracias al uso de los Maps y Listas, facilitó el armado del grafo para su futuro uso. El grafo permitió el modelado de la red de subte con los componentes que lo conforman, usando sus vértices como estaciones y los arcos como el recorrido que se tiene que hacer para llegar de un destino a otro con sus 3 pesos distintos (tiempo, trasbordos y congestión), en donde el usuario debe elegir cual usar en la aplicación. Al ser 3 pesos en los arcos, el programa no aborda el problema con una única solución. Ya eligiendo una de las opciones, se utilizaría el algoritmo de Dijkstra para armar el camino correspondiente y, consecuentemente, poder mostrar el resultado al usuario.

El uso de la estructura Map no solo facilito el armado del grafo, sino que también se utilizaba para la validación de las estaciones que el usuario elije, porque al estar creado el Map con las claves de las estaciones como Key, es muy fácil verificar si una estación está o no en la red de subte.