Universidad Autónoma Metropolitana Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación
Reporte de Proyecto Terminal

Aplicación para dispositivos con Android que encuentre la mejor ruta entre dos estaciones del Metro

Hernández Muñoz Víctor Manuel.
206359116

Trimestre 13-P
Agosto de 2013

Asesores:

Dr. Francisco Javier Zaragoza Martínez
Profesor Titular
Departamento de Sistemas

M.C. Arturo Zúñiga López
Profesor Asociado
Departamento de Electrónica

CONTENIDO

RESUMEN

La aplicación consiste en proporcionar la mejor ruta, de una estación origen a una estación destino, a través del sistema de transporte colectivo metro, en función de distintas métricas, estas son:

1.- El menor kilometraje.

2.- El menor tiempo.

3.- El menor número de estaciones a recorrer.

La aplicación funciona en dos modalidades, la primera consiste en proporcionar estación origen y estación destino, además de la métrica y otros parámetros opcionales (posteriormente las comentaré). La segunda es en realidad casi igual a la segunda, sólo con la diferencia de que la aplicación calcula la estación origen, la cual es la estación más cercana al dispositivo.

Los parámetros opcionales deciden el flujo de la aplicación ya que estos son:

1.- Se le da la opción al usuario de introducir estaciones por las cual dese forzosamente pasar en su recorrido.

2.- Se le da la opción al usuario de introducir estaciones por las cual no dese pasar en su recorrido.

En función de los parámetros anteriores y la métrica seleccionada, la aplicación es capaz de arrojar la mejor ruta. La solución se presenta de dos maneras:

1.- Se presenta el conjunto de estaciones que representan la mejor ruta, en una lista con imágenes.

2.- Se presenta el conjunto de estaciones que representan la mejor ruta, a través de google maps (es opcional).

OBJETIVOS

Objetivo general:

Implementar una aplicación para un dispositivo con sistema operativo Android, la cual permita obtener la mejor ruta entre una estación origen y una estación destino del Sistema de Transporte Colectivo Metro.

Objetivos específicos:

- Diseñar e implementar un módulo que permita hacer la solicitud para encontrar la mejor ruta.

- Diseñar e implementar un módulo que permita obtener la ubicación actual del dispositivo.

- Diseñar e implementar un módulo que permita obtener la estación del Sistema de Transporte Colectivo Metro más cercana al dispositivo.

- Diseñar e implementar un módulo que permita obtener la mejor ruta.

- Diseñar e implementar un módulo que permita obtener la solución sobre una interfaz gráfica de usuario.

- Diseñar e implementar un módulo que permita obtener la solución sobre Google Maps.

- Diseñar e implementar las interfaces necesarias que permitan al usuario utilizar la aplicación.

DESARROLLO

Primero que nada presento el modelo del conjunto de módulos con los que cuenta la aplicación.

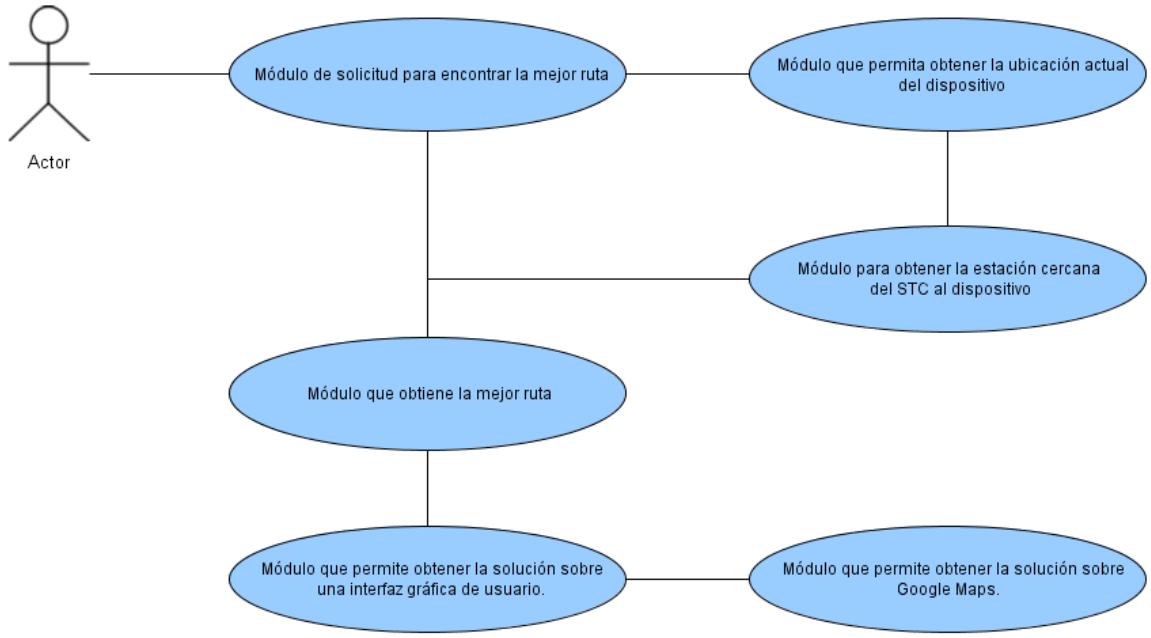

Figura 1. Diagrama de módulos de la aplicación.

A continuación se describen los módulos:

- **Módulo de solicitud para encontrar la mejor ruta.** Este módulo se encargará de obtener los parámetros y métricas introducidos por el usuario y de ofrecer las dos modalidades en que puede funcionar la aplicación:

1.- Solicitud de la mejor ruta a través del STC de modo normal. En este modo el usuario proporcionará la estación origen, la estación destino y la métrica elegida para obtener la mejor ruta en cuanto a: el mínimo número de estaciones a recorrer, el mínimo kilometraje a recorrer, o el mínimo tiempo estimado que cueste dicho trayecto. Además, si el usuario desea podrá introducir estaciones por las cuales desee o no pasar en su recorrido.

2.- Solicitud de la mejor ruta a través del STC mediante la ubicación actual del dispositivo. Es similar a la modalidad anterior sólo que con la excepción de que en este modo el usuario no introducirá la estación origen ya que esta se obtendría automáticamente. Para este modo el dispositivo necesita acceso a Internet y un sistema de posicionamiento global.

- **Módulo que permite obtener la ubicación actual del dispositivo.** Este módulo se encargará de obtener la ubicación actual del dispositivo, para esto se hará uso de los métodos disponibles para la programación en dispositivos móviles con Android.

- **Módulo para obtención de la mejor ruta.** En este módulo es el encargado de evaluar los parámetros y métricas introducidos por el usuario, con el objetivo de encontrar la mejor ruta en el STC. Para el caso en que se haya escogido la opción de forzar el paso por alguna(s) estación(es), se sacaran las permutaciones de las estaciones por las que se desea pasar, con el objetivo de elegir la mejor permutación (en cuanto a la métrica solicitada), para así formar la mejor ruta posible.

- **Diseñar e implementar un módulo que permita obtener la solución (solución offline).** En este módulo se mostrará en una interfaz gráfica de usuario una imagen de la red del STC metro y en un costado se enlistarán la secuencia de estaciones a seguir de origen a destino, donde también se mostrarán el kilometraje que cuesta llegar de la estación origen a la estación destino así como el número de estaciones a recorrer y un tiempo estimado en que se realizaría dicho trayecto, todo esto cambiará en función de la métrica elegida por el usuario.

- **Módulo que permita decidir cuál es la estación del STC más cercana a la ubicación actual del dispositivo.** Este módulo se encargará de decidir cuál es la estación del STC más cercana al dispositivo tras haber elegido la opción de obtener la mejor ruta de origen-destino respecto a la ubicación del dispositivo.

- **Diseñar e implementar un módulo que permita obtener la solución obtenida sobre los mapas de Google (solución online).** En este módulo se mostrará de manera gráfica la solución obtenida, dibujada o representada sobre los mapas de Google Maps.

- **Diseñar e implementar las interfaces necesarias que permitan al usuario utilizar la aplicación.** En este módulo se realizaran las interfaces graficas de usuario que requiere la aplicación.

Como se sabe cada módulo se compone de varias clases, en Android el diseño se basa en **Actividad** y xmls. A continuación se muestra el conjunto de actividades que compone la aplicación.

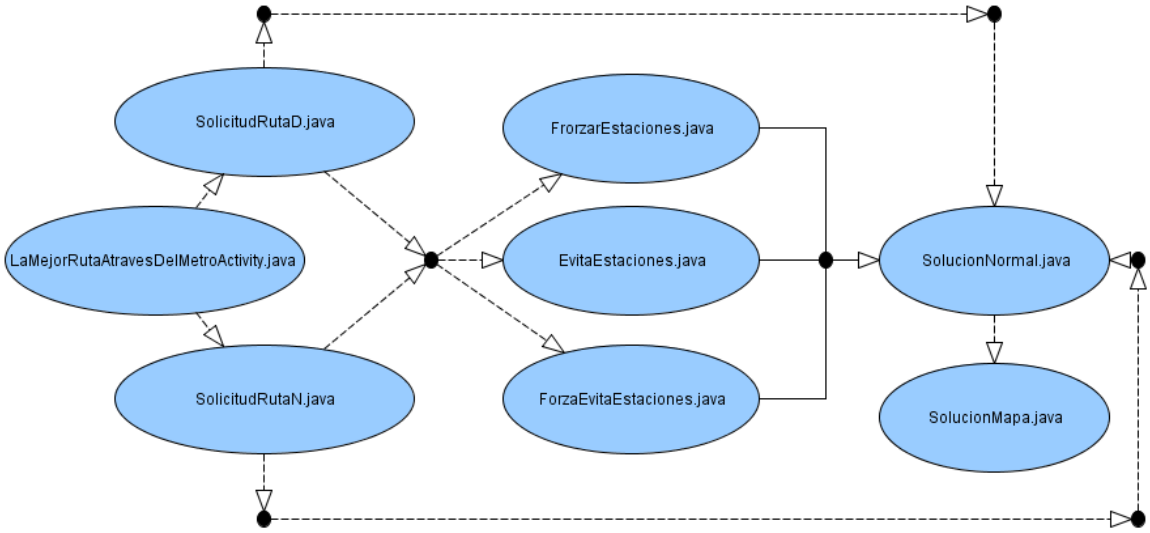

Figura 2. Diagrama de actividades de la aplicación.

Cada Actividad corresponde a una pantalla de la aplicación, las flechas con líneas punteadas indican que el paso por esas actividades no es obligatorio, debido a esto vemos que la aplicación puede tomar varios rumbos entre las actividades, por lo anterior a continuación explicaré actividad por actividad como funciona la aplicación siguiendo un flujo en particular, esto con el objetivo de explicar a detalle cómo funciona la aplicación siguiendo un flujo, además las clases SolicitudRutaD y SolicitudRutaN son similares así mismo las clases ForzaEstaciones, EvitaEstaciones y ForzaEvitaEstaciones, al final de explicar el flujo explicaré por aparte cada una de las clases faltantes, con la intención de que se tenga claro el funcionamiento de toda la aplicación.

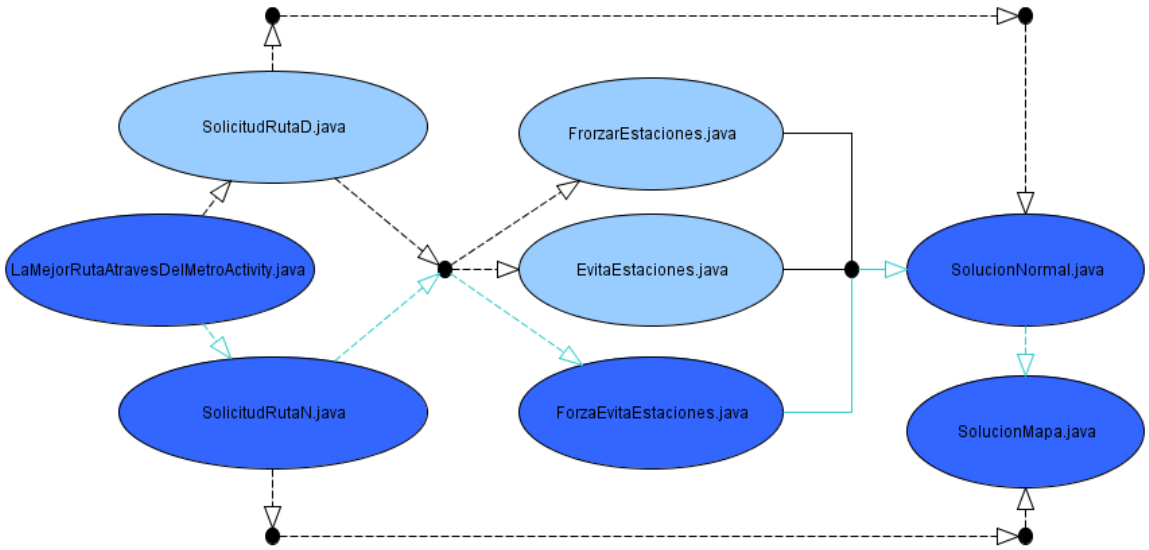

Figura 3. Diagrama de un Flujo en particular de la aplicación.

Descripción de las actividades a detalle.

Actividad uam.vhm.pt.LaMejorRutaAtravesDelMetroActivity.java

Esta actividad es la actividad inicial de la aplicación, es como si fuera el index de una aplicación web, el xml correspondiente (*activity_main.xml*), básicamente contiene la descripción de 3 **TextView** y dos Buttons. Es importante señalar que en los archivos xml se definen los componentes de diseño (la vista) de una aplicación Android, cada componente contiene varias propiedades entre estas las más importante es el id ya que por medio de esta propiedad se hace referencia desde las clases java a dicho componente.

Regresando con la descripción de los componentes de esta actividad, hay dos botones el botón con id=btn1 lo que hace simplemente es re direccionar a otra actividad (*uam.vhm.pt.SolicitudRutaN.java*) mediante un **Intent** , el botón con id=btn2 hace lo mismo que el botón anterior sólo que la redirección es a la actividad *uam.vhm.pt.SolicitudRutaD.java.* Lo anterior describe básicamente la funcionalidad de la actividad.

Actividad uam.vhm.pt.SolicitudRutaN.java

Comenzaré describiendo el xml correspondiente (*rutanxml.xml*), este contiene los siguientes controles: TextView, AutoCompleteTextView, Spinner, CheckBox, y Button.

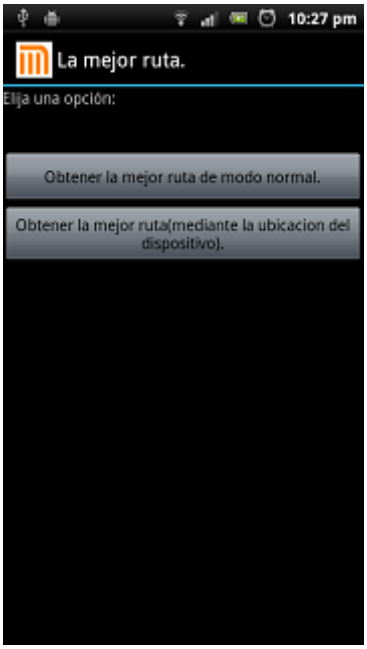

Figura 4. Pantalla correspondiente.

Pasando al código controlador (la actividad), La red del metro está descrita por un xml donde los nodos tienen la siguiente estructura:

```
<estacion nombre="Juanacatlán" latitud="19.413436518489483" longitud="-99.18229579925537" numero="3">
  <vecino nombre="Tacubaya(linea 1)" numero="2" distancia="1308"/>
  <vecino nombre="Chapultepec" numero="4" distancia="1123"/>
</estacion>
```

Como se ve cada nodo tiene las propiedades: nombre, latitud, longitud y numero, además de tener nodos hijos que son de tipo vecino, los cuales tienen como atributos nombre, numero y distancia (la cual señala la distancia [m] del vecino al nodo padre de tipo estacion), es importante mencionar como se obtuvo la latitud y longitud de cada estación. Simplemente se utilizó una fución javascript, la cual se coloca en la barra de direcciones del navegador web cuando se está posicionado en algún punto en particular en la página de google maps.

La función es:

```
javascript:void(prompt('',gApplication.getMap().getCenter()));
```

A continuación se ilustra el uso de la función.

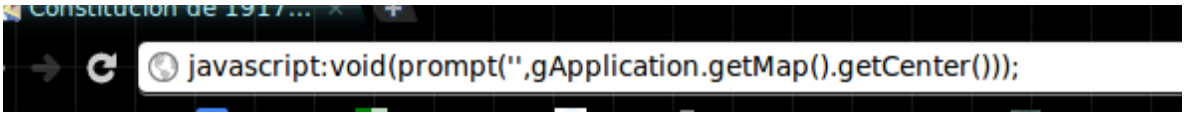

Figura 4. Función javascrip.

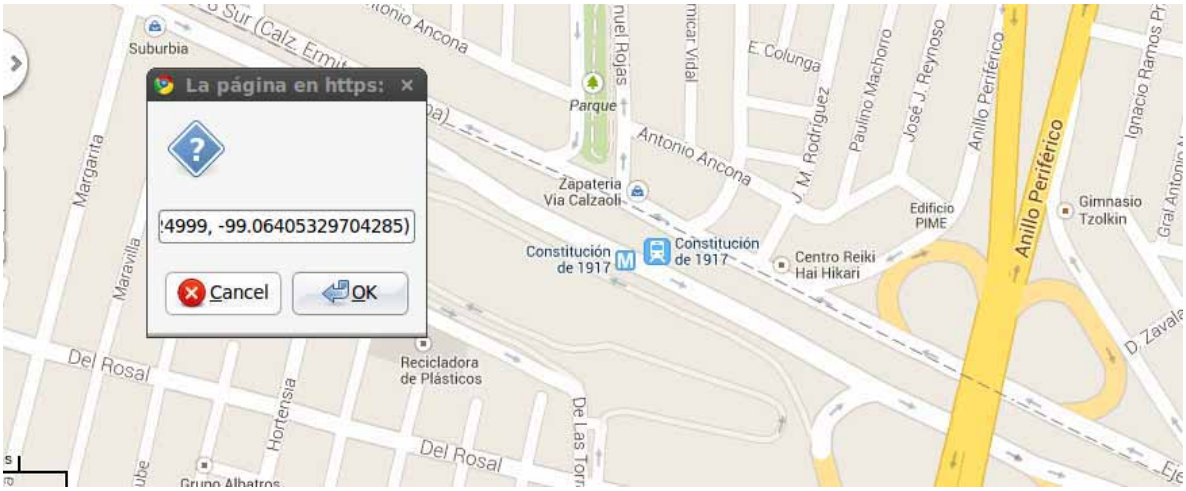

Figura 5. Resultado de aplicar la función para la obtención de coordenadas.

Ya descrito el formato del xml, ¿ahora qué?, se tiene que pasar a de serializarlo (pasarlo a objetos de determinado tipo) para esto se hizo la clase uam.pt.vhm.algoritmos.Parser.java, el método encargado de la de serealización es parse(), este método va analizando los nodos y atributos del xml, donde a la vez se van comparando el elemento xml en curso con strings para ver que nodo o que atributo se tiene en contexto, así mismo se van emcapsulando los atributos y los nodos a un objeto ya sea de tipo Estacion o de tipo Vecino, según sea el caso. Para encapsular los atributos, se tienen los beans *uam.vhm.pt.bean.Estacion.java*

y *uam.vhm.pt.bean.Vecino.java*. De esta manera se pasa el xml a objetos java de tipo Estación y tipo Vecino. Posteriormente ya en la clase SolicitudRutaN.java con la de serealización se obtiene un ArrayList<Estacion> el cual contiene las estaciones de toda la red del STC. Lo siguiente es referenciar los controles xml, algo importante que mencionar es que el componente **AutoCompleteTextView** necesita un insumo para poder realizar el auto completado, este insumo es el ArrayList<Estacion> (obtenido del parseo), se tienen dos AutoCompleteTextView uno para insertar la estación origen y uno para la estación destino. Otro control que deseo mencionar es el **Spinner**, este es el encargado de mostrar la métricas que se tienen para hacer la búsqueda de la mejor ruta, estas métricas se referencian de **Resource** de tipo <string-array/>, el cual contiene las métricas del menor t, menor km y menor número de estaciones. También se tiene dos checkBox(ForzaEstaciones y EvitaEstacionesn) por medio de los cuales el usuario decide forzar, evitar o ambas el paso por estaciones, estos son opcionales, aunque el comportamiento de estos define el comportamiento de la aplicación. A continuación describo los casos posibles en función de los CehckBox, ya que estos deciden la actividad que enseguida se debe llamar.

independientemente mente de la actividad que se mande a llamar en función del check o no check de los checkbox, en todos los casos posibles se pasa de la actividad en curso (*uam.vhm.pt.SolicitudRutaN.java*) a otra actividad haciendo uso de un Intent, a través de los intent no sólo podemos re dirigirnos a otra actividad si no también podemos enviar datos de la actividad en curso a la actividad destino, esto lo menciono ya que en independientemente mente del actividad que al que se re direccione se van a enviar los siguientes datos:

- El ArrayList<Estacion> (que contiene el conjunto de estaciones de la red del metro).
- La estación origen seleccionada (en forma de objeto de tipo Estacion).
- La estación destino seleccionada (en forma de objeto de tipo Estacion).
- La métrica seleccionada (en forma de String).

La estaciones origen y destinos provenientes de los AutocompleteTextView son de tipo *uam.vhm.pt.bean.Estacion.java* ya que el componente se llenó con el ArrayList<Estacion> (que contiene todas las estaciones de la red del metro). Por tanto al seleccionar una estación escrita en el componente, esta se obtiene como un objeto de tipo Estacion.

Si se checa el CheckBox ForzaEstaciones y no se checa el CheckBox EvitaEstacionesn, se manda a llamar a la actividad FrorzarEstaciones.java donde el xml correspondiente es forza_estaciones.xml El cual sólo cuenta con un botón y un AutoCompleteTextView por medio del cual se van dando las estaciones por las que se desea forzosamente pasar, en este caso se puede insertar más de una estación ya que cada vez que se va escogiendo una estación se va almacenando en un ArrayList<Estacion> el cual finalmente contendrá el array de estaciones por las que el usuario no desea pasar en su recorrido, cuando se oprime el botón enviar por medio de un Intent se envían los datos antes mencionados a la actividad *uam.vhm.pt.ForzaEstaciones.java*

Actividad uam.vhm.pt.ForzaEstaciones.java

El xml correspondiente a esta actividad es *forzar_estaciones.xml*, básicamente cuenta con dos TextView un Button y un AutoCompleteTextView.

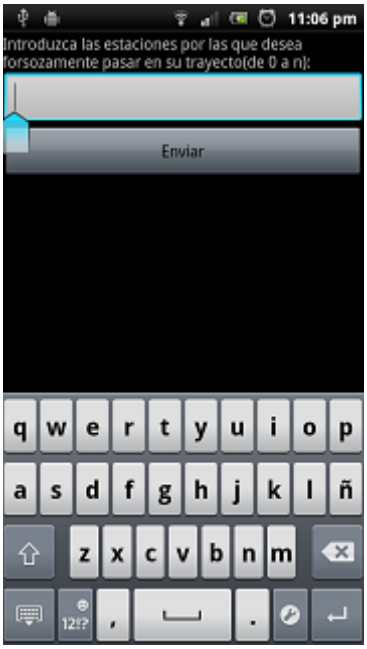

Figura 6. Pantalla correspondiente.

Ya en la actividad lo primero que se hace es recibir los datos provenientes de la otra actividad (la estación origen, la destino, la red del metro en forma de ArrayList y la métrica seleccionada por el usuario), ya después se hace referencia al AutoCompleteTextView este será el encargado de recibir y almacenar en un arraylist<Estacion>, el conjunto de estaciones por las que el usuario desee forzosamente pasar.

Finalmente lo que resta mencionar de esta actividad en la funcionalidad tras oprimir el botón enviar (método onclick()), primero se comienza validando la el tamaño y que este sea distinto de null del ArrayList<Estacion> que contiene el conjunto de estaciones por las que se desea forzosamente pasar, esto como validación, si se pasó la anterior validación, se crea un objeto de tipo SolucionForzaEstaciones, (*uam.vhm.pt.algoritmos.SolucionForzaEstaciones.java*) donde el constructor lleva los siguientes parámetros:

- El ArrayList<Estacion> (que contiene el conjunto de estaciones de la red del metro).
- La estación origen seleccionada (en forma de objeto de tipo Estacion).
- La estación destino seleccionada (en forma de objeto de tipo Estacion).
- La métrica seleccionada (en forma de String).
- Contexto de la actividad (en forma **Context**).

Creado el objeto de la clase SolucionForzaEstaciones.java se ocupa el método getMejorRutaPorPermutacion(ArraList<Estacion>, int), como vemos el primer parámetro es el arrayList que contiene las estacones por las cuales el usuario no desea pasar, el segundo parámetro es un int que significa una bandera, en este caso será 0.

El poner la bandera en 0 indica que las matrices de distancia y recorrido (resultantes del algoritmo de Floyd) provienen cada una de un archivo, donde estas fueron calculadas previamente, fuera de la aplicación y puestas en el directorio assets, esto se hizo por qué se pensó que era menos costoso para el dispositivo leer un archivo (que ya contiene las matrices resultado, a las cuales ya se les aplicó Floyd) que hacer los pasos siguientes, para **cada cálculo** de la mejor ruta:

1.- Deserialización del xml.
2.- Formar las matrices de adyacencia y distancias, para poder aplicar Floyd.
3.- Aplicar Floyd y obtener las matrices de distancias y recorridos.

Lo anterior sólo es útil para el caso en que el usuario no decida introducir estaciones a evitar (esto ocurre cuando el flujo sigue la aplicación pasa por las actividades EvitaEstacione.java, ForzaEvitaEstaciones.java), ya que en este caso cambia el plano del metro y por tanto se tienen que formar las matrices de adyacencia y distancias, después aplicar Floyd a estas, y finalmente sacar la mejor ruta.

Ya explicado el resultante de pasar como valor 0 (más adelante explicaré el funcionamiento de poner el valor en 1) en el método getMejorRutaPorPermutacion, este método devuelve un objeto de tipo Solucion (*uam.vhm.pt.bean.Solucion.java*), que es el bean que representa la solución en la aplicación (objeto que contiene la mejor Ruta).

El método lo que hace es retornan la mejor ruta en función de la mejor permutación de las estaciones por las que no se desea pasar.

Tenido el objeto solución comparamos si este es null, si es quiere decir que ocurrió algún error al intentar obtener la solución en el método getMejorRutaPorPermutacion, si es distinto de null entonces mediante un Intent enviamos el objeto Solucion a la clase o actividad *uam.vhm.pt.SolucionNormal.java*.

Actividad uam.vhm.pt.SolucionNormal.java

El xml acompañado de esta actividad *solucion_normal.xml*, esta cuenta con dos TextView, un Button, y un **ListView** Este control de Android nos permite desplegar una serie de elementos, digamos que es una lista en forma de vista.

Figura 7. Pantalla correspondiente.

Descrito el xml, y siguiendo con la clase, como su nombre lo indica, esta es la encargada de mostrar el conjunto de estaciones que se dan como solución, lo primero que hacemos es recibir el objeto Solucion, proveniente de la actividad ya sea (*ForzaEstaciones.java*, *uam.vhm.pt.EvitaEstaciones.java o uam.vhm.pt.ForzaEvita-Estaciones.java*), si recordamos todo esto depende del comportamiento de los ChekBox, que se encuentran en las actividades de solicitud de la mejor ruta (*uam.vhm.pt.SolicitudRutaN.java y uam.vhm.pt.SolicitudRutaD.java* y *uam.vhm.pt.SolicitudRutaD.java*). Como sabemos el objeto solución contiene:

- Un ArrayList<Estacion> el cual describe el conjunto de estaciones que conforman la solución.
- Un String que describe la métrica seleccionada por el usuario.
- Un double que describe el costo del recorrido.

Continuando describiendo la clase, se valida que el objeto solución no sea null, de ser así se crea una instancia de la clase ***uam.vhm.pt.EstacionesAdapter.java***, al constructor pasamos el contexto de la actividad y el objeto y el ArrayList que contiene el conjunto de estaciones que conforman la solución. Esta clase extiende o hereda de BaseAdapter por lo tanto tenemos que sobrescribir los métodos de generados los cuales son:

- int getCount() -> este sólo retorna la longitud del ArrayList<Estacion>.
- Object getItem(int pos) -> este método devuelve el elemento en la pasión = pos del Arraylist<Estacion>, en forma Object.
- Long getItemId(int pos) -> este retorna 0.
- View getView()-> este es el método más importante de la clase ya que por medio de este se dibuja la lista, este método va devolviendo cada imagen en modo de View, la

imagen se referencia en función del arrayList<Estaacion> a través de la posición de cada elemento, las imágenes están como resources, hay restricción al nombrarlas por eso me di a la tarea de construir un método el cual calcule el nombre del recurso (de la imagen), el método se llama getNombreIgane este método recibe un objeto de tipo Estacion y haciendo los siguientes replace se calcula el nombre de la imagen que se tiene como recurso:

- estacionActual.getNombre().trim().replace("(", "").replace(")", "").replace(" ", "").replace("á", "a").replace("é", "e").replace("í", "i").replace("ó", "o").replace("ú", "u").replace("Á", "A").replace("É", "E").replace("Í", "I").replace("Ó", "O").replace("Ú", "U").replace("ñ", "n").replace("/", "").replace("-", "").toLowerCase();

Sólo se permite para nombrar los recursos o elementos en el directorio res con minúsculas de la a-z números del 0-9 y utilizar _.

Ya descrita la clase que se encarga de dibujar los elementos en el ListView, volvemos a la actividad SolucionNormal, enseguida hacemos un setAdapter(referencia de la clase EstaconesAdapter), este método es el encargado de que añadir automáticamente un ListView que rellena toda la pantalla de la ListActivity.

Lo siguiente es obtener el costo del recorrido en función de la métrica seleccionada, esto se hace a través del método getCostoUnidad(String métrica, double costo), este método obtiene el costo final, ya que el algoritmo de Floyd arroja el costo en sí, en función de una métrica, pero este costo a un se tiene que procesar ya que si la métrica fue el menor km el costo es arrojado en metros, en cuanto a la métrica de menor tiempo el costo es arrojado en segundos, entonces mediante el método getCostoUnidad se obtiene el costo ya formateado con sus unidades convenientes y el dato numérico en función de estas, esto se obtiene en forma de String. Finalmente lo último que comentaré de esta actividad es lo que ocasiona el evento onClick del botón que se encuentra al inicio del ListView, tras oprimir dicho botón se pasa el objeto Solucion a la actividad *uam.vhm.pt. SolucionMapa.java* así mismo se redirigirnos a esta.

Antes de explicar cómo funciona la actividad *uam.vhm.pt.SolucionMapa.java* si nos fijamos como su nombre lo dice esta es la clase que nos perite ver la solución a través de google maps, para usar google maps se tienen que hacer una serie de pasos, en el referencia http://www.sgoliver.net/blog/?p=3244 se explica detalladamente los pasos a seguir para el uso de la biblioteca.

Actividad uam.vhm.pt.SolucionMapa.java

El xml correspondiente a esta actividad es *solucion_mapa.xml*, este sólo trae el control correspondiente para el uso de mapas en la clase SolucionMapa.

Figura 8. Pantalla correspondiente.

Ya configurada la biblioteca, podemos hacer uso de esta, bien la actividad SolucionMapa extiende de la clase *android.support.v4.app.FragmentActivity* que en realidad es el actividad que definió google para el uso de mapas, con esta api todas las acciones se hacen en función del objeto GoogleMap así que lo primero que hacemos es construir este objeto, lo hacemos de la siguiente manera:

```
GoogleMap mapa = ((SupportMapFragment) getSupportFragmentManager()
                        .findFragmentById(R.id.map)).getMap();
```

Creado el objeto mapa, la mayoría de las acciones básicas se pueden hacer con el objeto mapa. Lo siguiente es recibir los datos correspondientes de la actividad anterior (el objeto de tipo Solucion), checamos que este no sea null, de ser así comenzaremos checamos que el arrayList<Estacion> (el conjunto solución de estaciones) tenga elementos y no se null, de ser así comenzamos a pintar la solución, llamamos al método void drawSolution (ArrayList<Estacion>), lo primero que hacemos en este método es crear un objeto PolylineOptions, con este objeto podremos dibujar líneas sobre los mapas de google enseguida establecemos algunas propiedades del objeto (el color y el ancho de la línea), después empezamos a recorrer el ArrayList de estaciones, donde por cada estación vamos añadiendo un marker esto haciendo uso del objeto mapa (el que al inicio creamos) mediante el método addMarker(sobre el objeto mapa), al cual le pasamos un objeto de tipo MarkerOptions, por medio de este establecemos la posición (donde ira el marker) y el título, obviamente la posición la obtenemos mediante los atributos latitud y longitud de cada elemento de tipo estación y al igual que el nombre, que será el de la estación en curso. A continuación también vamos añadiendo las líneas que van de estación en estación mediante el método add al cual le pasamos la latitud y longitud de la estación en curso y mediante el método mapa.addPolyline(PolylineOptions) vamos añadiendo las líneas al objeto mapa para que sean dibujadas. Finalmente centramos el mapa al inicio del recorrido aplicando un zoom

adecuado con el uso de la clase CameraPosition y el método animateCamera de la siguiente manera mapa.animateCamera(CameraPosition). Con lo anterior básicamente queda dibujada la solución sobre google maps.

Al crear un proyecto Android, en la raíz del proyecto entre otras cosas se crea un fichero llamado AndroidManifest.xml donde básicamente ahí se declaran las distintas actividades con las que cuenta la aplicación, permisos, servicios, etc. También se generan los requisitos del sistema, SDK requerida y le damos un ID a nuestra aplicación. Esto lo menciona ya que para usar el api de google maps necesitamos incluir unas líneas en el **AndroidManifest.xml** de la aplicación.

Para el uso del api de google maps, se tienen que agregar permisos en el archivo AndroidManiest así como el api_key que se generó previamente.

Integracion de la api_key

```
...
<application>
...
    <meta-data android:name="com.google.android.maps.v2.API_KEY"
               android:value="api_key"/>
...
</application>
```

Integración de permisos

Estos permisos que permiten que el dispositivo tenga acceso a internet, a los servicios de google y al almacenamiento externo del dispositivo (utilizado para el caché de los mapas).

```
<permission
      android:name=" paquete.donde.usamos.gMaps.permission.MAPS_RECEIVE"
      android:protectionLevel="signature"/>

<uses-permission android:name="paquete.donde.usamos.gMaps.permission.MAPS_RECEIVE"/>

<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
```

Con esto termina el flujo ilustrado en la figura 3. A continuación como ya había mencionado antes explicaré las actividades faltantes.

Actividad uam.vhm.pt.SolicitudRutaD.java

El xml correspondiente (*rutadxml.xml*) contiene los siguientes controles: un TextView, un AutoCompleteTextView, un Spinner, dos CheckBox, y un Button.

Figura 9. Pantalla correspondiente.

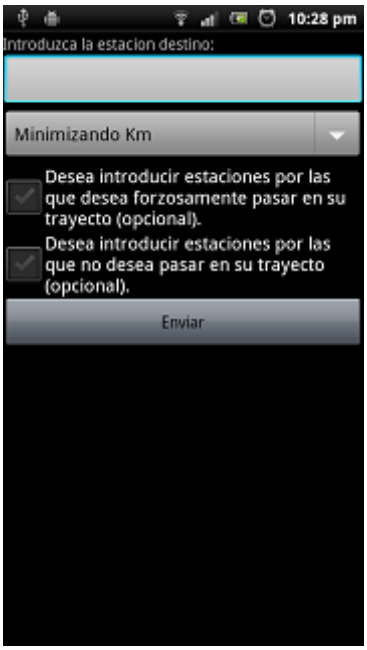

Figura 10. Pantalla correspondiente.

Si notamos en realidad este xml es muy parecido al xml *rutanxml.xml* correspondiente a la clase *uam.vhm.pt.SolicitudRutaN.java*, igualmente la clase *uam.vhm.pt.SolicitudRutaD.java* es muy parecida a la clase *uam.vhm.pt.SolicitudRutaN.java*, en realidad sólo hay dos diferencias, la primera es que en la clase SolicitudRutaD sólo se utiliza un TextView, un AutoCompleteTextView, y un Spinner, a diferencia de la clase SolicitudRutaN donde se utilizan dos controles de los anteriores mencionados, la segunda diferencia y además la más importante es que se añade la funcionalidad de la búsqueda de la estación origen, que es la estación más cercana al dispositivo.

Comenzamos añadiendo un PregressDialog, este nos servirá para hacerle saber al usuario que debe esperar mientras se calcula la estación más cercana al dispositivo. Posteriormente tras de arrancar e ProgressDialog (mediante el método ProgressDialog.show()), hacemos uso de la clase LocationManager y LocationListener. Mediante LocationManager hacemos la instancia para utilización de servicios, en este caso el de localización geográfica, pasándole (en el constructor) un String (`Context.LOCATION_SERVICE`), mientras que LocationListener es la interface en la que se definen las acciones a realizar tras recibir cada actualización de la posición geográfica.

Creadas las instancias anteriores, mando a llamar a método tiempo de espera este básicamente lanza un hilo paralelamente a la aplicación, este hilo inicia esperando 10 segundos mientras se trata de obtener la estación más cercana al dispositivo.

Tras lanzar el hilo de espera se revisa si el proveedor gps está habilitado, de ser así llamamos al método requestLocationUpdates, (pasándole como proveedor gps) que es el método que permite obtener la posición actualizada. Los parámetros que lleva este método son:

- Nombre del proveedor de localización al que nos queremos suscribir.
- Tiempo mínimo entre actualizaciones, en milisegundos.
- Distancia mínima entre actualizaciones, en metros.
- Instancia de un objeto LocationListener, que tendremos que implementar previamente para definir las acciones a realizar al recibir cada nueva actualización de la posición.

Para la utilización de LoaltionListener creé una subclase dentro de la clase SoliciudD.java la cual implementa la interfaz LocationListener, a continuación mencionare los métodos que se implementan:

- void OnLocationChanged(Location location) -> Lanzado cada vez que se recibe una actualización de la posición.

- void OnProviderDisabled(String provider) -> Lanzado cuando el proveedor se deshabilita.
- void onProviderEnabled(String provider) -> Lanzado cuando el proveedor se habilita.

- void onStatusChanged(String provider, int status, Bundle extras) -> Lanzado cada vez que el proveedor cambia su estado, que puede variar entre OUT_OF_SERVICE,TEMPORARILY_UNAVAILABLE, AVAILABLE.

El único método que utiliza la aplicación es onLocationChanged, como su nombre lo indica, este método se llama tras darse un cambio de posición del dispositivo, aunque, en realidad yo forcé el método para que al encontrar la primera localización ya no se actualice más, cada que ocurre lo anterior asigno a una variable de tipo estación (que en realidad es la estación origen) el resultado del método updateLocation(Location location).

Dentro del método updateLocation lo que se hace es crear una instancia de la clase *uam.vhm.pt.Localizacion.java*, a la cual le pasamos el contexto de la aplicación, y el array de estaciones proveniente de la deserialización del xml (el plano de la red del metro).

Clase uam.vhm.pt.Localizacion.java

El método más importante de esta clase es:

```
Estacion getEstacionMasCercana(double LatB, double LngB)
```

Este método es el encargado de buscar la estación más cercana al dispositivo en función de la longitud y la latitud entrante, como vemos el método retorna un objeto de tipo Estacion, como mencione anterior mente el resultado de este método se asigna a una variable de tipo Estación (ya dentro del método updateLocation), así mismo se finaliza la ejecución del ProgressDialog usando el método ProgressDialog dismmis(). Este es el caso en el que se encontró la estación vía gps, si no se encontró vía gps se intenta vía wifi, (todo lo anterior ocurre durante los 10 segundos de espera del hilo lanzado), regresando al método tiempo de espera, finalizados los 10 segundos se checa si el proveedor fue gps si fue así se manda a llamar un proceso el cual, comienza a hacer ahora la búsqueda de la estación más cercana pero ahora mediante el proveedor `LocationManager.NETWORK_PROVIDER`, en este caso se comprueba si el dispositivo tiene habilitado la localización vía redes celulares, igualmente si tiene conexión a internet, de no ser así mediante un AlertDialog se invita al usuario a habilitarlos, en este caso igualmente se lanza un hilo de 10 segundos en lo que se calcula la estación por este método, aunque este método la mayoría de los casos es más rápido pero más impreciso, si se logró obtener la estación más cercana, se termina este proceso y se libera la actividad para que el usuario introduzca la estación destino, así como los demás parámetros para la búsqueda de la mejor ruta, si no se logró obtener la estación vía redes celulares entonces se regresa al usuario a la actividad anterior (*uam.vhm.pt.LaMejorRuta-AtravesDelMetroActivity.java*).

Modificaciones en el AndroidManifest.xml

Para usar la geo localización en Android necesitamos agregar en el AndroidManifest.xml una serie de permisos, estos son:

```
<!-- Permisos para poder usar wifi -->
<uses-permission android:name="android.permission.INTERNET" />

<!-- Permisos para localización por gps y wifi -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

<!-- Permisos para comprobar si hay internet -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Actividad uam.vhm.pt.EvitaEstaciones.java

El xml correspondiente a esta actividad es *evitar_estaciones.xml*, es idéntico al xml *forzar_estaciones.xml* ya que cuentan ambos con los mismos componentes.

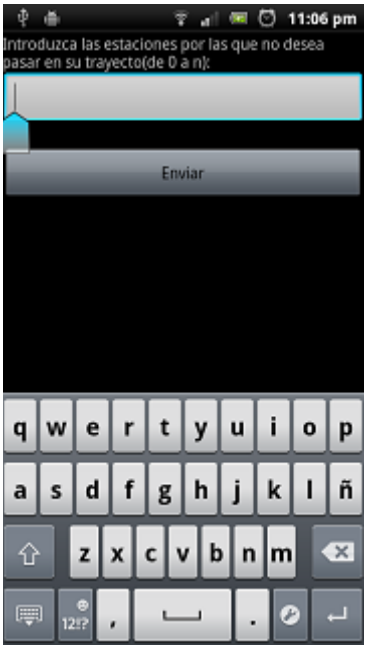

Figura 11. Pantalla correspondiente.

Análogo a la actividad anterior se reciben los datos ya sea de la actividad *uam.vhm.pt.SolicitudRutaN.java* o *uam.vhm.pt.SolicitudRutaD.java*, estos recordemos que son: la estación origen, la estación destino, la métrica elegida y el ArrayList que contiene el conjunto de estaciones de la red del metro. Igualmente se van almacenando en un ArrayList<Estacion> las estaciones que va ingresando el usuario donde este array representa las estaciones por las que el usuario no desea pasar, así mismo se tiene un botón en cual, donde tras oprimir este primero se valida si el array (que contiene las estaciones a evitar) esta vació, en caso de no estar vacío creamos una instancia de la clase *uam.vhm.pt.algoritmos.SolucionEvitaEstaciones.java*

cuyo constructor lleva los siguientes parámetros:

- La estación origen seleccionada (en forma de objeto de tipo Estacion).
- La estación destino seleccionada (en forma de objeto de tipo Estacion).
- La métrica seleccionada (en forma de String).

El método que se encarga de la obtención de la mejor ruta en función de los parámetros del constructor, es getMejorRuta(arrayList<Estacion> estacionesAEvitar), este método finalmente devuelve la mejor ruta en forma de un objeto de tipo Solucion (*uam.vhm.pt.bean.Solucion.java*), el método getMejorRuta.

Actividad uam.vhm.pt.ForzaEvitaEstaciones.java

Esta actividad es una combinación de la actividad *uam.vhm.pt.ForzaEstaciones.-java* y la actividad *uam.vhm.pt.EvitaEstaciones.java* igualmente el xml, contiene dos TextView dos AutoCompleteTextView y un Button, donde un conjunto de controles es para introducir las estaciones por las que no se desea pasar, el otro conjunto es para introducir las estaciones or las que se desea forzosamente pasar, el xml correspondiente es *forza_evita_estaciones.xml*.

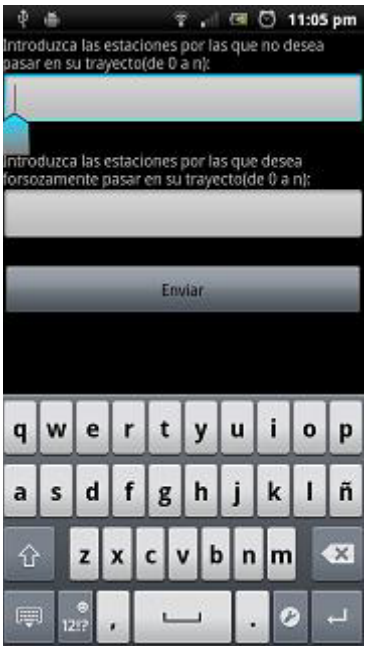

Figura 12. Pantalla correspondiente.

Lo único que comentar es los métodos que se llaman tras detectarse el click del botón enviar, como sabemos el método onClick se laza, lo primero que ocurre es que se crea una instancia de la clase SolucionEvitaEstaciones, ya había hablado de esta clase, como sabemos el constructor lleva estación origen, estación destino, array e estaciones (la red del metro) y la métrica, después mandamos a llamar al método cambiaCostosEnLaRedDelMetro a este

método le pasamos el array de estaciones por las que no se quiere pasar, este método lo que hace es cambiar los costos de los vecinos de las estaciones por las que no se quiere pasar, el valor que les pone es infinito esto para que al correr el algoritmo del Floyd descarte el paso por alguna de aquellas estaciones y así se logre evitarlas, (ojo, con esto se tiene una nueva red del metro, por lo tanto se tendrán que volver a generar las matrices de distancias y adyacencia, después se tendrá que aplicar Floyd y finalmente obtener la mejor ruta en función de lo anterior), enseguida se comprueba si esta tiene más de un elemento y no es null de ser así se crea un objeto de tipo SolucionForzaEstaciones este lleva los mismos parámetros que la clase SolucionevitaEstaciones más el contexto de la actividad, de esta clase llamamos el método getMejorRutaPorPermutacion(Estacion estacionesAForzar, int 1), este método nos retorna la solución en función de la mejor permutación (la de menor costo), el pasar como parámetro un 1 indica que las matrices de recorridos y distancias se calcularan desde cero. Esto quiere decir que:

1.- Se obtendrá en forma de ArrayList<Estacion> la red del metro o el conjunto de estaciones del metro.
2.- Se calcularan o formaran las matrices de adyacencia y distancias.
3.- Se aplicara el algoritmo de Floyd.

Finalmente ya tenido el objeto solución, se pasa mediante un Intent y se re direcciona a la actividad ***uam.vhm.pt.SolucionNormal.java***, la cual como ya vimos pinta la solución.

RECURSOS

Para la realización del proyecto se cuenta con todos los recursos necesarios, estos se enlistan enseguida:

- Una Notebook HP Mini 110-1016LA, con un disco duro de 160 GB, memoria RAM de 1G y un procesador Intel ATOM de 1.6 Ghz.,

- Un Smarthphone Xperia Play R800, con sistema operativo Android 2.3.4, con memoria interna de 158 MB, procesador a 1 Ghz ARM 11.

- En cuanto a las herramientas para desarrollo se usará software libre (el IDE Eclipse que cuente con el plugin para desarrollo en Android y el SDK de Android).

CONCLUSIÓN

Al programar la aplicación noté ciertas cuestiones que quiero comentar. Por ejemplo hay grandes diferencias entre la localización vía gps y la vía red. Las ventajas de usar gps es que la posición obtenida mediante este medio es más precisa que obtenerla vía red, además para el uso del gps no se necesita internet ni uso de datos, pero lamentablemente el gps no funciona en interiores, el dispositivo debe de estar en exteriores a campo abierto, además de que la batería del dispositivo se descarga más rápido, las ventajas de usar localización por red es que este método es más rápido, no consume tanta batería pero es menos exacto y se necesita red ya sea plan de datos o internet.

Otra cosa que quiero mencionar es que en la aplicación fue benéfico el uso de un xml para la representación de la red del metro, debido a que en ocasiones (según del flujo que tome la aplicación) se puede evitar las siguientes tareas:

1.- La construcción de las matrices de adyacencia y distancias que entran al Floyd.

2.- El cálculo de Floyd.

Lo anterior se puede evitar debido a que se tienen archivos ya con el resultado de Floyd (calculados fuera de la aplicación), los cuales sólo se leen y se pasan a la matriz respectiva. La siguiente figura muestra en los flujos de la aplicación donde ya no se calcula Floyd, estos son el flujo de flechas verdes, flechas azules y flechas rojas.

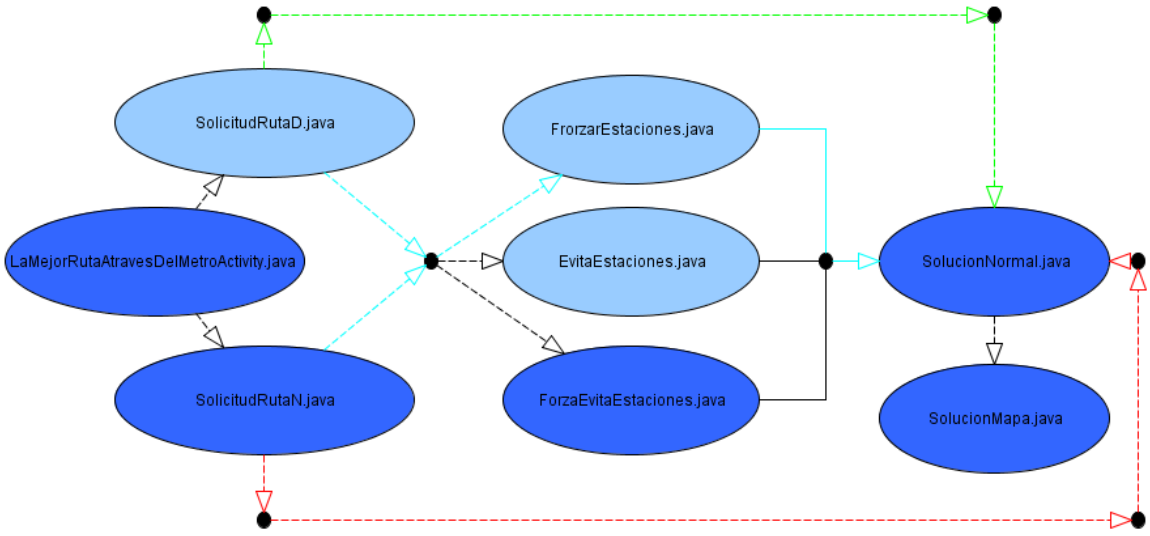

Figura 13. Flujos de la aplicación que no aplican Floyd.

Si notamos sólo se aplica Floyd en el caso en que el usuario decida introducir estaciones por las cuales no desea pasar, esto debido a que se tiene que cambiar la red del metro (cambio de costos en algunas estaciones), y por lo tanto se tiene que aplicar Floyd. Por lo anterior insisto en que fue buena idea codificar el grafo médiate un xml, otra idea que tenía era codificar el grafo en una bd, pero creo que esto no convenía ya que cada vez que se calculara la mejor ruta se tendría que sacra las matrices que consume Floyd y calcular Floyd. Aunque quizá si se agregan más transportes públicos como el metrobús, el suburbano, la ecobici, el pumabus etc, ya no convenga utilizar xml.

También fue una gran idea usar el algoritmo de Floyd ya que al principio tenía en mente utilizar el algoritmo de Dijkstra, lo cual no hubiera sido atinado, al menos para este problema no hubiese sido eficiente ya que este no saca los mejores costos de todos los nodos vs todos, cosa que si hace el algoritmo de Floyd.

GLOSARIO

Actividad es la clase java es la base de cualquier aplicación Android con interfaz de usuario descrita por un archivo xml.

TextView es un componente nativo de Android el cual es básicamente una label o etiqueta para introducir texto.

Intent sirve para invocar componentes, en android entendemos por componentes las actividades, servicios, código ejecutándose en segundo plano, etc.

AutoCompleteTextView, es un simplemente un TextView con autocompletado.

Spinner, se trata de una lista despegable.

Resource, en android en un archivo (como un archivo de música, o archivo que define el layout de una ventana) o un valor (como puede ser el título de un botón). Estos archivos son ligados a la aplicación de tal forma que si es necesario cambiarlos por alguna razón no sea necesario recompilar la aplicación y sólo cambiar la referencia o contenido de estos. Algunos de los tipos más comunes de recursos en android son Strings, bitmaps, colors y layouts. Los resources se encuentran en el directorio res de un proyecto android.

Context, Es la referencia que se tiene para una actividad en Android, permite la utilización de componentes, recursos y todos los elementos que se tienen disponibles en una actividad, sin necesidad de heredar de Activity.

ListView Este control de Android nos permite desplegar una serie de elementos, digamos que es una lista en forma de vista.

ProgressDialog Es un elemento en Android para hacer entender al usuario que debe esperar mientras la aplicación hace una tarea en segundo plano, ya sea un cálculo, una autentificación, una descarga etc.

AlertDialog Ventana de confirmación (Aceptar/Cancelar), en Android.

APÉNDICES

A. Paquete uam.vhm.pt (clases controladoras)

EstacionesAdapter.java

```
package uam.vhm.pt;


import java.util.ArrayList;


import uam.vhm.pt.bean.Estacion;
import android.app.Activity;
import android.content.Context;
//import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
//import android.widget.TextView;


public class EstacionesAdapter extends BaseAdapter{


protected Activity activity;
protected ArrayList<Estacion> listaDeEstaciones;
public static String TAG = "MSGLOG";


public EstacionesAdapter(Activity activity, ArrayList<Estacion> items)
{
this.activity = activity;
this.listaDeEstaciones = items;
}


@Override
public int getCount() {
return listaDeEstaciones.size();
}


@Override
```

```
public Object getItem(int position) {
return listaDeEstaciones.get(position);
}

@Override
public long getItemId(int position) {
return 0;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
// TODO Auto-generated method stub

View vi = convertView;

if(convertView == null) {
LayoutInflater inflater = (LayoutInflater) activity.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
vi = inflater.inflate(R.layout.listado_estaciones, null);
}

Estacion estacionActual = listaDeEstaciones.get(position);

if(estacionActual != null)
{
ImageView image = (ImageView) vi.findViewById(R.id.imagen);
int imageResource = activity.getResources().getIdentifier("drawable/"+getNombreImagen(estacionActual), null, activity.getPackageName());
//Log.d("MSGLOG","imageResource="+imageResource);
image.setImageDrawable(activity.getResources().getDrawable(imageResource));

/*TextView nombre = (TextView) vi.findViewById(R.id.nombre);
nombre.setText(estacionActual.getNombre());*/
}

return vi;
}

//Este método recompone el nombre de la estacion para encontrar la imagen o recurso correspondiente de la carpeta res/drawable
public String getNombreImagen(Estacion estacionActual)
{
```

```
if(estacionActual != null)

{

String estacion = estacionActual.getNombre().trim().replace("(", "").replace(")", "").replace(" ", "").replace("á", "a").replace("é", "e").replace("í", "i").replace("ó", "o").replace("ú", "u").replace("Á", "A").replace("É", "E").replace("Í", "I").replace("Ó", "O").replace("Ú", "U").replace("ñ", "n").replace("/", "").replace("-", "").toLowerCase();

return estacion;

}


return null;

}


}
```

EvitaEstaciones.java

```
package uam.vhm.pt;


import java.util.ArrayList;


import uam.vhm.pt.bean.Estacion;

import android.app.Activity;

import android.content.Context;

//import android.util.Log;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.BaseAdapter;

import android.widget.ImageView;

//import android.widget.TextView;


public class EstacionesAdapter extends BaseAdapter{


protected Activity activity;

protected ArrayList<Estacion> listaDeEstaciones;

public static String TAG = "MSGLOG";


public EstacionesAdapter(Activity activity, ArrayList<Estacion> items)

{

this.activity = activity;

this.listaDeEstaciones = items;

}
```

```
@Override
public int getCount() {
return listaDeEstaciones.size();
}

@Override
public Object getItem(int position) {
return listaDeEstaciones.get(position);
}

@Override
public long getItemId(int position) {
return 0;
}

@Override
public View getView(int position, View convertView, ViewGroup parent) {
// TODO Auto-generated method stub

View vi = convertView;

if(convertView == null) {
  LayoutInflater inflater = (LayoutInflater) activity.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
  vi = inflater.inflate(R.layout.listado_estaciones, null);
}

Estacion estacionActual = listaDeEstaciones.get(position);

if(estacionActual != null)
{
                                        ImageView image = (ImageView) vi.findViewById(R.id.imagen);
                                        int imageResource = activity.getResources().getIdentifier("drawable/"+getNombreImagen(estacionActual),
null, activity.getPackageName());
                                        //Log.d("MSGLOG","imageResource="+imageResource);
                                        image.setImageDrawable(activity.getResources().getDrawable(imageResource));

                                        /*TextView nombre = (TextView) vi.findViewById(R.id.nombre);
                                        nombre.setText(estacionActual.getNombre());*/
}
```

```
return vi;

}


//Este método recompone el nombre de la estacion para encontrar la imagen o recurso correspondiente de la carpeta res/drawable

public String getNombreImagen(Estacion estacionActual)

{

if(estacionActual != null)

{

                                                String estacion = estacionActual.getNombre().trim().replace("(", "").replace(")", "").replace(" ", "").replace("á", "a").replace("é", "e").replace("í", "i").replace("ó", "o").replace("ú", "u").replace("Á", "A").replace("É", "E").replace("Í", "I").replace("Ó", "O").replace("Ú", "U").replace("ñ", "n").replace("/", "").replace("-", "").toLowerCase();

                                                return estacion;

}


return null;

}


}
```

EvitaEstaciones.java

```
package uam.vhm.pt;


import java.util.ArrayList;


import uam.vhm.pt.R;

import uam.vhm.pt.algoritmos.SolucionEvitaEsaciones;

import uam.vhm.pt.bean.Estacion;

import uam.vhm.pt.bean.Solucion;

import android.app.Activity;

import android.content.Intent;

import android.os.Bundle;

import android.text.Editable;

import android.text.TextWatcher;

import android.util.Log;

import android.view.View;

import android.widget.AdapterView;

import android.widget.ArrayAdapter;

import android.widget.AutoCompleteTextView;

import android.widget.Button;

import android.widget.TextView;
```

```
import android.widget.Toast;

import android.widget.AdapterView.OnItemClickListener;


public class EvitaEstaciones extends Activity implements TextWatcher {

/** Called when the activity is first created. */


AutoCompleteTextView autoCompleteEvitarEst1;


ArrayList<Estacion> estEvita;

ArrayList<Estacion> estaciones = null;

Estacion estacionOrg = null;

Estacion estacionDes = null;

String metrica = null;


@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.evita_estaciones);


estaciones = new ArrayList<Estacion>();


//Recivimos los datos provenientes de otra actividad

estacionOrg = getIntent().getParcelableExtra("estOrig");

estacionDes = getIntent().getParcelableExtra("estDest");

estaciones = getIntent().getParcelableArrayListExtra("estaciones");

metrica = getIntent().getStringExtra("metrica");



//Referenciamos los controles de autocompletado Y atrapamos el evento que ocurre al seleccionar un elememto del autocomplete\\

estEvita = new ArrayList<Estacion>();

referenciaAutoComplete(autoCompleteEvitarEst1, R.id.CompleteEvitarEst1).setOnItemClickListener(new OnItemClickListener(){

@Override

public void onItemClick(AdapterView<?> parent, View view, int position, long rowId) {

estEvita.add((Estacion)parent.getItemAtPosition(position));

TextView numEst = (TextView)findViewById(R.id.TextEvitaNumEst);

numEst.setText(getResources().getString(R.string.EstEnCola)+estEvita.size());

}

});
```

```
final Button BtnEnviarEstAEvitar = (Button)findViewById(R.id.BtnEnviarEstAEvitar);
BtnEnviarEstAEvitar.setOnClickListener(new View.OnClickListener(){

@Override
public void onClick(View arg0) {
// TODO Auto-generated method stub
if( (estEvita != null) && (estEvita.size() > 0))
{
SolucionEvitaEsaciones se = new SolucionEvitaEsaciones(estacionOrg, estacionDes, estaciones, metrica);
Solucion sol = null;
sol = se.getMejorRuta(estEvita);

//Envialos la solucion para que sea mostrada al usuario
if(sol != null && !Double.isInfinite(sol.getCosto()))
{
                                        Intent it = new Intent(getApplicationContext(), SolucionNormal.class);
                                        it.putExtra("solucion", sol);
                                        startActivity(it);
}

else
{
                                        Toast toast = Toast.makeText(getApplicationContext(), "No es posible calcular la ruta, con los parametros tenidos.", Toast.LENGTH_LONG);
                                        toast.show();

                                        Log.d("MSGLOG","El objecto solucion es null, mÃ©todo onCreate(), clase: "+this.getClass().getName());
}
}
}
});
}


                                                                        //MÃ©todos para el autocomplete\\
public AutoCompleteTextView referenciaAutoComplete(AutoCompleteTextView autoComplete, int idResource)
{
autoComplete = (AutoCompleteTextView)findViewById(idResource);
autoComplete.addTextChangedListener((TextWatcher) this);
```

```
autoComplete.setAdapter(new ArrayAdapter<Estacion>(this, android.R.layout.simple_dropdown_item_1line, estaciones));
return autoComplete;
}

//MÃ©todos de la interfaz TextWatcher (para uso del autocompletado)
@Override
public void afterTextChanged(Editable s) {
// TODO Auto-generated method stub
}

@Override
public void beforeTextChanged(CharSequence s, int start, int count,
int after) {
// TODO Auto-generated method stub
}

@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {
// TODO Auto-generated method stub
//selection.setText(edit.getText());
}

}
```

ForzaEvitaEstaciones.java

```
package uam.vhm.pt;

import java.util.ArrayList;

import uam.vhm.pt.R;
import uam.vhm.pt.algoritmos.SolucionEvitaEsaciones;
import uam.vhm.pt.algoritmos.SolucionForzaEstaciones;
import uam.vhm.pt.bean.Estacion;
import uam.vhm.pt.bean.Solucion;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
```

```
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;

public class ForzaEvitaEstaciones extends Activity implements TextWatcher {
/** Called when the activity is first created. */

private ArrayList<Estacion> estaciones = null;
ArrayList<Estacion> estEvita;
ArrayList<Estacion> estForz;;
Estacion estacionOrg = null;
Estacion estacionDes = null;
String metrica = null;

AutoCompleteTextView autoCompleteForzarEst1;
AutoCompleteTextView autoCompleteEvitarEst1;

@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.forza_evita_estaciones);

estacionOrg = getIntent().getParcelableExtra("estOrig");
estacionDes = getIntent().getParcelableExtra("estDest");
estaciones = getIntent().getParcelableArrayListExtra("estaciones");
metrica = getIntent().getStringExtra("metrica");

//Referenciamos los controles de autocompletado Y atrapamos el evento que ocurre al seleccionar un elememto del autocomplete\\
estEvita = new ArrayList<Estacion>();
referenciaAutoComplete(autoCompleteEvitarEst1, R.id.CompleteForzarEvitarEst1).setOnItemClickListener(new OnItemClickListener(){
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long rowId) {
```

```
estEvita.add((Estacion)parent.getItemAtPosition(position));
TextView numEst = (TextView)findViewById(R.id.TextEvitaForzarNum1Est);
numEst.setText(getResources().getString(R.string.EstEnCola)+estEvita.size());
}
});

//Referenciamos los controles de autocompletado Y atrapamos el evento que ocurre al seleccionar un elememto del autocomplete\\
estForz = new ArrayList<Estacion>();
referenciaAutoComplete(autoCompleteForzarEst1, R.id.CompleteForzarEvitarEst4).setOnItemClickListener(new OnItemClickListener(){
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long rowId) {
estForz.add((Estacion)parent.getItemAtPosition(position));
TextView numEst = (TextView)findViewById(R.id.TextEvitaForzarNum2Est);
numEst.setText(getResources().getString(R.string.EstEnCola)+estForz.size());
}
});

final Button BtnEnviarEstForzarEvitar = (Button)findViewById(R.id.BtnEnviarEstForzarEvitar);
BtnEnviarEstForzarEvitar.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View v)
{
if( (estEvita != null) && (estEvita.size() > 0) && (estForz != null) && (estForz.size() > 0))
{
SolucionEvitaEsaciones se = new SolucionEvitaEsaciones(estacionOrg, estacionDes, estaciones, metrica);
ArrayList<Estacion> redDelMetro = null;
redDelMetro = se.cambiaCostosEnLaRedDelMetro(estEvita);

/*for(int i=0;i<redDelMetro.size();i++)
{
Log.d("MSGLOG","---"+redDelMetro.get(i)+"---");
for(int j=0;j<redDelMetro.get(i).getVecinos().size();j++)
{
                                        Log.d("MSGLOG","\t"+redDelMetro.get(i).getVecinos().get(j));

}
}*/
```

```
if( (redDelMetro != null) && (redDelMetro.size() > 0) )
{
//Obtenemos la red del metro (con los costos cambiados en algunas estaciones, para asi evitar el paso por dichas estaciones)
SolucionForzaEstaciones sf = new SolucionForzaEstaciones(estacionOrg, estacionDes, redDelMetro, metrica, getApplicationContext());
Solucion sol = sf.getMajorRutaPorPermutacion(estForz, 1);

//Envialos la solucion para que sea mostrada al usuario
if(sol != null)
{
                                        Intent it = new Intent(getApplicationContext(), SolucionNormal.class);
                                        it.putExtra("solucion", sol);
                                        startActivity(it);
}
}
}
}
});
}

                                                //MÃ©todos para el autocomplete\\

public AutoCompleteTextView referenciaAutoComplete(AutoCompleteTextView autoComplete, int idResource)
{
autoComplete = (AutoCompleteTextView)findViewById(idResource);
autoComplete.addTextChangedListener((TextWatcher) this);
autoComplete.setAdapter(new ArrayAdapter<Estacion>(this, android.R.layout.simple_dropdown_item_1line, estaciones));
return autoComplete;
}

//MÃ©todos de la interfaz TextWatcher (para uso del autocompletado)
@Override
public void afterTextChanged(Editable s) {
// TODO Auto-generated method stub
}

@Override
public void beforeTextChanged(CharSequence s, int start, int count,
int after) {
// TODO Auto-generated method stub
```

```
}

@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {
// TODO Auto-generated method stub
//selection.setText(edit.getText());
}

}
```

FrorzarEstaciones.java

```
package uam.vhm.pt;

import java.util.ArrayList;

import uam.vhm.pt.R;
import uam.vhm.pt.algoritmos.SolucionForzaEstaciones;
import uam.vhm.pt.bean.Estacion;
import uam.vhm.pt.bean.Solucion;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.TextView;

public class FrorzarEstaciones extends Activity implements TextWatcher {
/** Called when the activity is first created. */

AutoCompleteTextView autoCompleteForzarEst1;
ArrayList<Estacion> estForz;
ArrayList<Estacion> estaciones = null;
Estacion estacionOrg = null;
```

```
Estacion estacionDes = null;
String metrica = null;

private static final String TAG = "MSGLOG";

@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.frorzar_estaciones);

estaciones = new ArrayList<Estacion>();

//Recivimos los datos provenientes de otra actividad
estacionOrg = getIntent().getParcelableExtra("estOrig");
estacionDes = getIntent().getParcelableExtra("estDest");
estaciones = getIntent().getParcelableArrayListExtra("estaciones");
metrica = getIntent().getStringExtra("metrica");

//Referenciamos los controles de autocompletado Y atrapamos el evento que ocurre al seleccionar un elememto del autocomplete\\
estForz = new ArrayList<Estacion>();
referenciaAutoComplete(autoCompleteForzarEst1, R.id.CompleteForzarEst1).setOnItemClickListener(new OnItemClickListener(){
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long rowId) {
estForz.add((Estacion)parent.getItemAtPosition(position));
TextView numEst = (TextView)findViewById(R.id.TextForzarNumEst);
numEst.setText(getResources().getString(R.string.EstEnCola)+estForz.size());
}
});

final Button BtnEnviarEstAForzar = (Button)findViewById(R.id.BtnEnviarEstAForzar);
BtnEnviarEstAForzar.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View v)
{
// TODO Auto-generated method stub
if( (estForz != null) && (estForz.size() > 0) )
{
//Clase que tiene un mÃ©todo (getMajorRutaPorPermutacion()) que obtiene la solucion en funcion de la metrca y la mejor permutacion
```

```
SolucionForzaEstaciones sf = new SolucionForzaEstaciones(estacionOrg, estacionDes, estaciones, metrica, getApplicationContext());

Solucion sol = sf.getMajorRutaPorPermutacion(estForz, 0);


//Envialos la solucion para que sea mostrada al usuario

if(sol != null)

{

Intent it = new Intent(getApplicationContext(), SolucionNormal.class);

it.putExtra("solucion", sol);

startActivity(it);

}


else

Log.d(TAG,"El objecto solucion es null, clase: "+this.getClass().getName());


}

}

});


}



                                                    //MÃ©todos para el autocomplete\\

public AutoCompleteTextView referenciaAutoComplete(AutoCompleteTextView autoComplete, int idResource)

{

autoComplete = (AutoCompleteTextView)findViewById(idResource);

autoComplete.addTextChangedListener((TextWatcher) this);

autoComplete.setAdapter(new ArrayAdapter<Estacion>(this, android.R.layout.simple_dropdown_item_1line, estaciones));

return autoComplete;

}


//MÃ©todos de la interfaz TextWatcher (para uso del autocompletado)

@Override

public void afterTextChanged(Editable s) {

// TODO Auto-generated method stub

}


@Override

public void beforeTextChanged(CharSequence s, int start, int count,

int after) {
```

```
// TODO Auto-generated method stub
}

@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {
// TODO Auto-generated method stub
//selection.setText(edit.getText());
}

}
```

LaMejorRutaAtravesDelMetroActivity.java

```
package uam.vhm.pt;

import uam.vhm.pt.R;
import uam.vhm.pt.SolicitudRutaD;
import uam.vhm.pt.SolicitudRutaN;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import com.actionbarsherlock.app.SherlockActivity;
import android.content.Context;

public class LaMejorRutaAtravesDelMetroActivity extends SherlockActivity {

protected Context context;

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

context = getApplicationContext();
getSupportActionBar().setTitle(R.string.app_name_ab);
getSupportActionBar().setHomeButtonEnabled(true);
getSupportActionBar().setDisplayShowHomeEnabled(true);
getSupportActionBar().setIcon(R.drawable.metro_logo);

//referencia a los botones del xml
```

```
final Button btnBoton1 = (Button)findViewById(R.id.btn1);
final Button btnBoton2 = (Button)findViewById(R.id.btn2);

//eventos de de click
btnBoton1.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Intent i1 = new Intent(LaMejorRutaAtravesDelMetroActivity.this, SolicitudRutaN.class);
startActivity(i1);
}
});

btnBoton2.setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View v) {
Intent i1 = new Intent(LaMejorRutaAtravesDelMetroActivity.this, SolicitudRutaD.class);
startActivity(i1);
}
});
}

@Override
public boolean onCreateOptionsMenu(com.actionbarsherlock.view.Menu menu) {
com.actionbarsherlock.view.MenuInflater inflater = getSupportMenuInflater();
inflater.inflate(R.menu.main, menu);
return super.onCreateOptionsMenu(menu);
}
}
```

Localizacion.java

```
package uam.vhm.pt;

import java.util.ArrayList;

import android.content.Context;
import android.widget.Toast;
import uam.vhm.pt.bean.Estacion;

public class Localizacion
{
```

```
Context context;
private ArrayList<Estacion> allEstaciones = null;

public Localizacion(Context context, ArrayList<Estacion> allEstaciones)
{
this.context = context;
this.allEstaciones = allEstaciones;
}

//MÃ©todo que obtiene la estación mas cercana al dispositivo
public Estacion getEstacionMasCercana(double LatB, double LngB)
{
//Obtenemos la distancia de la primer estacon de la red del metro, esto para obtener un dato con que empezar a comparar
double dist = 0.0;
Estacion estacion = new Estacion();

//calculo inicial de la diatancia
dist = Math.sin(deg2rad(allEstaciones.get(0).getLatitud()))*Math.sin(deg2rad(LatB)) +
Math.cos(deg2rad(allEstaciones.get(0).getLatitud()))*Math.cos(deg2rad(LatB))*Math.cos(deg2rad(allEstaciones.get(0).getLatitud())-deg2rad(LngB));

dist = Math.acos(dist);
dist = this.rad2deg(dist);
dist = dist*60*1.1515;
dist = dist*1.609344;//a km

//buscamos la estación ma cercana
for(int i=1;i<allEstaciones.size();i++)
{
double aux = Math.sin(deg2rad(allEstaciones.get(i).getLatitud()))*Math.sin(deg2rad(LatB)) +
Math.cos(deg2rad(allEstaciones.get(i).getLatitud()))*Math.cos(deg2rad(LatB))*Math.cos(deg2rad(allEstaciones.get(i).getLongitud())-deg2rad(LngB));

aux = Math.acos(aux);
aux = this.rad2deg(aux);
aux = aux*60*1.1515;
aux = aux*1.609344;//a km

//vamos comprando distancias
if(dist > aux)
```

```
{
dist = aux;
estacion = allEstaciones.get(i);
}
}

//motramos los resultdos mediante un toast
Toast toast = Toast.makeText(context, estacion.toString()+" a una distancia de "+dist+" km.", Toast.LENGTH_SHORT);
toast.show();

return estacion;
}

//conversion de degrados a radianes
private double deg2rad(double deg)
{
return (deg * Math.PI / 180.0);
}

//conversion de radianes a degrados
private double rad2deg(double rad)
{
return (rad * 180.0 / Math.PI);
}

}
```

SolicitudRutaD.java

```
package uam.vhm.pt;

import java.util.ArrayList;
import org.xmlpull.v1.XmlPullParser;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.location.Location;
import android.location.LocationListener;
```

```
import android.location.LocationManager;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
import android.os.Bundle;
import android.os.Handler;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.util.Xml;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.Spinner;
import android.widget.Toast;
import uam.vhm.pt.algoritmos.Parser;
import uam.vhm.pt.algoritmos.SolucionNormal;
import uam.vhm.pt.bean.Estacion;
import uam.vhm.pt.bean.Solucion;
//import uam.vhm.pt.datos.Datos;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.AdapterView.OnItemSelectedListener;

public class SolicitudRutaD extends Activity implements TextWatcher {
/** Called when the activity is first created. */
private static final String TAG = "MSGLOG";
CheckBox checkForzaE;
CheckBox checkEvitaE;

Handler handler;

Spinner sp;

AutoCompleteTextView autoCompleteDest;

private boolean forzaEst = false;
```

```
private boolean evitaEst = false;
private String metrica = "";
private String estacionDest = "";

private Estacion estOrig = null;
private Estacion estDest = null;
private ArrayList<Estacion> estaciones = null;

LocationManager myLocManager;
LocationListener myLocListener;
private ProgressDialog dialog;

final Handler handle = new Handler();

@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.rutadxml);

//Creamos y configuramos un ProgressDialog para evitar que el usuario interactue con la interfaz
//en lo que se calcula la estación mas cercana.
dialog = new ProgressDialog(this);
dialog.setTitle("Estación mas cercana\nal dispositivo.");
dialog.setMessage("Calculando...");
dialog.setCancelable(true);
dialog.show();

//traemos la rred del metro, deserializando el xml
estaciones = new ArrayList<Estacion>();
XmlPullParser parser = Xml.newPullParser();
parser  = getResources().getXml(R.xml.red_del_metro);
Parser p = new Parser(parser);
estaciones = p.parse();

//referenciamos el servicio de geolocalizacion
myLocManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
myLocListener = new MyLocationListener();

//Lanzamos el hilo de espera de 1 minuto
```

```
tiempoEspera(LocationManager.GPS_PROVIDER);

Toast toast = Toast.makeText(getApplicationContext(), "Intento gps", Toast.LENGTH_SHORT);
toast.show();

//Primero intentamos por gps
if(myLocManager.isProviderEnabled(LocationManager.GPS_PROVIDER))
myLocManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 15000, 15, myLocListener);

checkForzaE = (CheckBox)findViewById(R.id.ForzaEstacionesd);
checkEvitaE = (CheckBox)findViewById(R.id.EvitaEstacionesd);

autoCompleteDest = (AutoCompleteTextView)findViewById(R.id.CompleteEstDestinod);
autoCompleteDest.addTextChangedListener(this);

autoCompleteDest.setAdapter(new ArrayAdapter<Estacion>(this, android.R.layout.simple_dropdown_item_1line, estaciones));
//atrapamos el evento que ocurre al seleccionar un elememto del autocomplete
autoCompleteDest.setOnItemClickListener(new OnItemClickListener() {

@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long rowId) {
estDest = (Estacion)parent.getItemAtPosition(position);
}
});

sp = (Spinner)findViewById(R.id.Metricasd);

ArrayAdapter<?> adaptador = ArrayAdapter.createFromResource(this, R.array.metricas, android.R.layout.simple_spinner_item);
adaptador.setDropDownViewResource(android.R.layout.simple_spinner_item);
sp.setAdapter(adaptador);

obtenMetrica(sp);

checaF(checkForzaE);
checaE(checkEvitaE);

final Button btnEnviard = (Button)findViewById(R.id.BtnEnviard);
btnEnviard.setOnClickListener(new View.OnClickListener() {
```

```
@Override
public void onClick(View v)
{
// TODO Auto-generated method stub

Toast met = Toast.makeText(getApplicationContext(),metrica, Toast.LENGTH_SHORT);
met.show();

if(checaEditEstacion(autoCompleteDest))
{
if(forzaEst && !evitaEst)
{
Intent it = new Intent(SolicitudRutaD.this, FrorzarEstaciones.class);
it.putExtra("estOrig", estOrig);
it.putExtra("estDest", estDest);
it.putParcelableArrayListExtra("estaciones", estaciones);
it.putExtra("metrica", metrica);
startActivity(it);
}

else if(evitaEst && !forzaEst)
{
Intent it = new Intent(SolicitudRutaD.this, EvitaEstaciones.class);
it.putExtra("estOrig", estOrig);
it.putExtra("estDest", estDest);
it.putParcelableArrayListExtra("estaciones", estaciones);
it.putExtra("metrica", metrica);
startActivity(it);
}

else if(forzaEst && evitaEst)
{
Intent it = new Intent(SolicitudRutaD.this, ForzaEvitaEstaciones.class);
it.putExtra("estOrig", estOrig);
it.putExtra("estDest", estDest);
it.putParcelableArrayListExtra("estaciones", estaciones);
it.putExtra("metrica", metrica);
startActivity(it);
}
```

```
else
{
SolucionNormal s = new SolucionNormal(estaciones, estOrig, estDest, metrica, getApplicationContext());

Solucion solucion = s.getSolucion();
if(solucion != null)
{
Intent it = new Intent(SolicitudRutaD.this, uam.vhm.pt.SolucionNormal.class);
it.putExtra("solucion", solucion);
startActivity(it);
}

else
Log.d(TAG,"El objeto solucion es null calse: "+getClass().getName());
}
}

else
{
Toast msg = Toast.makeText(getApplicationContext(),"Aun no se han ingresado " +
"todos los campos obligatorios.", Toast.LENGTH_SHORT);
msg.show();
}
}

});
}

public boolean checaEditEstacion(AutoCompleteTextView eText)
{
estacionDest = eText.getText().toString().toLowerCase();

if(estacionDest.length() == 0 || estacionDest.equals(null))
{
return false;
}

else
```

```
{
return true;
}
}

public void checaF(CheckBox cBox)
{
cBox.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener()
{
@Override
public void onCheckedChanged(CompoundButton arg0, boolean isChecked) {
// TODO Auto-generated method stub
if (isChecked)
{
forzaEst = true;
}

else
{
forzaEst = false;
}
}
});
}



public void checaE(CheckBox cBox)
{
cBox.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener()
{
@Override
public void onCheckedChanged(CompoundButton arg0, boolean isChecked) {
// TODO Auto-generated method stub
if (isChecked)
{
evitaEst = true;
}

else
```

```
{
evitaEst = false;
}
}
});
}

public void obtenMetrica(Spinner sp)
{
sp.setOnItemSelectedListener(new OnItemSelectedListener(){

@Override
public void onItemSelected(AdapterView<?> arg0, View arg1,
int arg2, long arg3) {
// TODO Auto-generated method stub
metrica = arg0.getItemAtPosition(arg2).toString();
}

@Override
public void onNothingSelected(AdapterView<?> arg0) {
// TODO Auto-generated method stub
}

});
}

@Override
public void afterTextChanged(Editable arg0) {
// TODO Auto-generated method stub

}

@Override
public void beforeTextChanged(CharSequence s, int start, int count,
int after) {
// TODO Auto-generated method stub

}
```

```
@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {
// TODO Auto-generated method stub

}

//                                          *******          Hilo de tiempo de espera *******     //

//Aquí NO puedo alterar la GUI
protected void tiempoEspera(final String proveedor)
{
//Creamos un hilo que se dormira por 10 segundos, si pasados 10 segundos no se ha obtenido
//la estación mas cercana paramos el intento de obtención de la estación mas cercana al dispositivo.

Thread t = new Thread()
{
public void run()
{
try
{                                          //Esperamos un momento
Thread.sleep(10000);
}

catch(InterruptedException e)
{
e.printStackTrace();
}

//Terminado el minuto de espera
if(proveedor.equals("gps"))
{
handle.post(proceso);
}

if(proveedor.equals("network"))
{
dialog.dismiss();
}
}
```

```
};
t.start();
}

//Aquí puedo alterar la GUI
final Runnable  proceso = new Runnable()
{
public void run()
{
//Si no se encontro la estación (via gps), lo intentamos con wifi
if(estOrig == null)
{
//paramos la localización via gps
myLocManager.removeUpdates(myLocListener);

Toast toast = Toast.makeText(getApplicationContext(), "El tiempo de cálculo se termino, " +
"no se pudo obtener la estación mas cercana al dispo.", Toast.LENGTH_SHORT);
toast.show();

toast = Toast.makeText(getApplicationContext(), "Intento wifi", Toast.LENGTH_SHORT);
toast.show();

//Comprobamos conexion a internet
if(existeConexionInternet())
{
//Nuevamente lanzamos el hilo de espera de 1 minuto
tiempoEspera(LocationManager.NETWORK_PROVIDER);

//ahora intentamos buscar la estacion mar cercana via Wffi(primero checamos que este
// activada la localización por redes inalambricas)
if(myLocManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER))
{
myLocManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER, 15000, 15, myLocListener);
}

//si no esta activa la localizaciÃ³n por redes inalambricas invitamos al uauario a activarla
else
{
dialog.dismiss();
```

```
Log.i("LogsAndroid", "activa redes inalambricas");
activaGPSoRedesInalambricas();
}
}

//Si no hubo conexión invitamos al usuario a activarla
else
{
dialog.dismiss();
Log.i("LogsAndroid", "activa internet");
activaWifi();
}
}

}
};

//                                        *******        Geolocalización        *******    //

public boolean existeConexionInternet() {

Log.i("LogsAndroid", "******* existeConexionInternet");

ConnectivityManager cm = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);

NetworkInfo netInfo = cm.getActiveNetworkInfo();

if (netInfo != null && netInfo.isConnectedOrConnecting()) {

Log.i("LogsAndroid", "******* true");

return true;
}

Log.i("LogsAndroid", "******* false");

return false;
}
```

```
//Método que invita al usuario para que active la utilización de satÃ©lites GPS o bien la utilizaciÃ³n redes inalÃ¡mbricas
private void activaWifi()
{
AlertDialog.Builder msg = new AlertDialog.Builder(this);
msg.setMessage("Desea activar internet?.")
.setTitle("Activaión de coexión a internet.")
.setCancelable(true)
.setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {

@Override
public void onClick(DialogInterface dialog, int which) {
// TODO Auto-generated method stub

//Dirigimos al usuario al menu para activaciÃ³n de localización via gps o wiffi
Intent settingsIntent = new Intent(android.provider.Settings.ACTION_WIFI_SETTINGS);
settingsIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED);
startActivity(settingsIntent);

finish();//para que finalice la actividad actual => que te regrese a la actividad anterior
}
})
.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {

@Override
public void onClick(DialogInterface dialog, int which) {
// TODO Auto-generated method stub

//Terminamos la actividad y mostramos un msg de la necesidad de activar alguna de las opciones
Toast toast = Toast.makeText(getApplicationContext(), "Sin la activación de alguna de las opciones" +
" anteriores no se puede calcular la estación mas cercana al dispositivo.", Toast.LENGTH_SHORT);
toast.show();
finish();//para que finalice la actividad actual => que te regrese a la actividad anterior
}
});

AlertDialog alert = msg.create();
alert.show();
}
```

```
//MÃ©todo que invita al usuario para que active la utilizaciÃ³n de satÃ©lites GPS o bien la utilizaciÃ³n redes inalÃ¡mbricas
private void activaGPSoRedesInalambricas()
{
AlertDialog.Builder msg = new AlertDialog.Builder(this);
msg.setMessage("Para encontrar la estaciÃ³n mas cercana al dispositivo se necesita tener activada la" +
" utilización de satélites GPS o bien la utilización redes inalámbricas, desea activar el alguna de estas?.")
.setTitle("Activaión de GPS o redes\ninalámbricas.")
.setCancelable(true)
.setPositiveButton("Aceptar", new DialogInterface.OnClickListener() {

@Override
public void onClick(DialogInterface dialog, int which) {
// TODO Auto-generated method stub

//Dirigimos al usuario al menu para activaciÃ³n de localizaciÃ³n via gps o wiffi
Intent settingsIntent = new Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS);
settingsIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK | Intent.FLAG_ACTIVITY_RESET_TASK_IF_NEEDED);
startActivity(settingsIntent);

finish();//para que finalice la actividad actual => que te regrese a la actividad anterior
}
})
.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {

@Override
public void onClick(DialogInterface dialog, int which) {
// TODO Auto-generated method stub

//Terminamos la actividad y mostramos un msg de la necesidad de activar alguna de las opciones
Toast toast = Toast.makeText(getApplicationContext(), "Sin la activación de alguna de las opciones" +
" anteriores no se puede calcular la estación mas cercana al dispositivo.", Toast.LENGTH_SHORT);
toast.show();
finish();//para que finalice la actividad actual => que te regrese a la actividad anterior
}
});

AlertDialog alert = msg.create();
alert.show();
```

```
}

@Override
protected void onPause() {
// TODO Auto-generated method stub
super.onPause();
Toast toast = Toast.makeText(getApplicationContext(), "call onPause()", Toast.LENGTH_SHORT);
toast.show();
}

@Override
protected void onStop() {
// TODO Auto-generated method stub
super.onStop();
Toast toast = Toast.makeText(getApplicationContext(), "call onStop()", Toast.LENGTH_SHORT);
toast.show();
myLocManager.removeUpdates(myLocListener);
this.finish();
}

@Override
protected void onDestroy() {
// TODO Auto-generated method stub
super.onDestroy();
Toast toast = Toast.makeText(getApplicationContext(), "call onDestroy()", Toast.LENGTH_SHORT);
toast.show();
myLocManager.removeUpdates(myLocListener);
this.finish();
}

private Estacion updateLocation(Location loc)
{
Localizacion localiza = new Localizacion(this.getApplicationContext(), estaciones);
return localiza.getEstacionMasCercana(loc.getLatitude(), loc.getLongitude());
}



private class MyLocationListener implements LocationListener
```

```
{

@Override
public void onLocationChanged(Location location) {
// TODO Auto-generated method stub
estOrig = updateLocation(location);
dialog.dismiss();
}

@Override
public void onProviderDisabled(String provider) {
// TODO Auto-generated method stub
}

@Override
public void onProviderEnabled(String provider) {
// TODO Auto-generated method stub
}

@Override
public void onStatusChanged(String provider, int status, Bundle extras) {
// TODO Auto-generated method stub
}

}

}
```

SolicitudRutaN.java

```
package uam.vhm.pt;

import java.util.ArrayList;
import org.xmlpull.v1.XmlPullParser;
import uam.vhm.pt.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.Editable;
import android.text.TextWatcher;
//import android.util.Log;
```

```
import android.util.Log;
import android.util.Xml;
import android.view.View;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.AutoCompleteTextView;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.Spinner;
import android.widget.Toast;
import uam.vhm.pt.algoritmos.Parser;
import uam.vhm.pt.algoritmos.SolucionNormal;
import uam.vhm.pt.bean.Estacion;
import uam.vhm.pt.bean.Solucion;

public class SolicitudRutaN extends Activity implements TextWatcher {
/** Called when the activity is first created. */
private static final String TAG = "MSGLOG";
CheckBox checkForzaE;
CheckBox checkEvitaE;

Spinner sp;

AutoCompleteTextView autoCompleteOrig;
AutoCompleteTextView autoCompleteDest;

private boolean forzaEst = false;
private boolean evitaEst = false;
private String metrica = "";
private Estacion estOrig = null;
private Estacion estDest = null;
private ArrayList<Estacion> estaciones = null;

@Override
public void onCreate(Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState);
setContentView(R.layout.rutanxml);

//deserialización del xml
estaciones = new ArrayList<Estacion>();
XmlPullParser parser = Xml.newPullParser();
parser  = getResources().getXml(R.xml.red_del_metro);
Parser p = new Parser(parser);
estaciones = p.parse();

checkForzaE = (CheckBox)findViewById(R.id.ForzaEstacionesn);
checkEvitaE = (CheckBox)findViewById(R.id.EvitaEstacionesn);

autoCompleteOrig = (AutoCompleteTextView)findViewById(R.id.CompleteEstOrigenn);
autoCompleteOrig.addTextChangedListener(this);
autoCompleteOrig.setAdapter(new ArrayAdapter<Estacion>(this, android.R.layout.simple_dropdown_item_1line, estaciones));

autoCompleteDest = (AutoCompleteTextView)findViewById(R.id.CompleteEstDestinon);
autoCompleteDest.addTextChangedListener(this);
autoCompleteDest.setAdapter(new ArrayAdapter<Estacion>(this, android.R.layout.simple_dropdown_item_1line, estaciones));


//atrapamos el evento que ocurre al seleccionar un elememto del autocomplete
autoCompleteOrig.setOnItemClickListener(new OnItemClickListener() {

@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long rowId) {
estOrig = (Estacion)parent.getItemAtPosition(position);
}
});

//atrapamos el evento que ocurre al seleccionar un elememto del autocomplete
autoCompleteDest.setOnItemClickListener(new OnItemClickListener() {

@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long rowId) {
estDest = (Estacion)parent.getItemAtPosition(position);
}
});
```

```
//referenciamos y llenamos el spinner
sp = (Spinner)findViewById(R.id.Metricasn);

ArrayAdapter<?> adaptador = ArrayAdapter.createFromResource(this, R.array.metricas, android.R.layout.simple_spinner_item);
adaptador.setDropDownViewResource(android.R.layout.simple_spinner_item);
sp.setAdapter(adaptador);

obtenMetrica(sp);

checaF(checkForzaE);
checaE(checkEvitaE);

final Button btnEnviarn = (Button)findViewById(R.id.BtnEnviarn);
btnEnviarn.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View v)
{
// TODO Auto-generated method stub

Toast met = Toast.makeText(getApplicationContext(),metrica, Toast.LENGTH_SHORT);
met.show();

//decidimos que actividad llamar
if( (estOrig != null) && (estDest != null))
{
if(forzaEst && !evitaEst)
{
Intent it = new Intent(SolicitudRutaN.this, FrorzarEstaciones.class);
it.putExtra("estOrig", estOrig);
it.putExtra("estDest", estDest);
it.putParcelableArrayListExtra("estaciones", estaciones);
it.putExtra("metrica", metrica);
startActivity(it);
}

else if(evitaEst && !forzaEst)
{
```

```
Intent it = new Intent(SolicitudRutaN.this, EvitaEstaciones.class);
it.putExtra("estOrig", estOrig);
it.putExtra("estDest", estDest);
it.putParcelableArrayListExtra("estaciones", estaciones);
it.putExtra("metrica", metrica);
startActivity(it);
}

else if(forzaEst && evitaEst)
{
Intent it = new Intent(SolicitudRutaN.this, ForzaEvitaEstaciones.class);
it.putExtra("estOrig", estOrig);
it.putExtra("estDest", estDest);
it.putParcelableArrayListExtra("estaciones", estaciones);
it.putExtra("metrica", metrica);
startActivity(it);
}

else
{
SolucionNormal s = new SolucionNormal(estaciones, estOrig, estDest, metrica, getApplicationContext());

Solucion solucion = s.getSolucion();
if(solucion != null)
{
Intent it = new Intent(SolicitudRutaN.this, uam.vhm.pt.SolucionNormal.class);
it.putExtra("solucion", solucion);
startActivity(it);
}

else
Log.d(TAG,"El objeto solucion es null calse: "+getClass().getName());
}
}

else
{
Toast msg = Toast.makeText(getApplicationContext(),"Aun no se han ingresado " +
"todos los campos obligatorios.", Toast.LENGTH_SHORT);
```

```
msg.show();
}
}
});
}

public void checaF(CheckBox cBox)
{
cBox.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener()
{
@Override
public void onCheckedChanged(CompoundButton arg0, boolean isChecked) {
// TODO Auto-generated method stub
if (isChecked)
{
forzaEst = true;
}

else
{
forzaEst = false;
}
}
});
}

public void checaE(CheckBox cBox)
{
cBox.setOnCheckedChangeListener(new CheckBox.OnCheckedChangeListener()
{
@Override
public void onCheckedChanged(CompoundButton arg0, boolean isChecked) {
// TODO Auto-generated method stub
if (isChecked)
{
evitaEst = true;
}
```

```
else
{
evitaEst = false;
}
}
});
}

public void obtenMetrica(Spinner sp)
{
sp.setOnItemSelectedListener(new OnItemSelectedListener(){

@Override
public void onItemSelected(AdapterView<?> arg0, View arg1,
int arg2, long arg3) {
// TODO Auto-generated method stub
metrica = arg0.getItemAtPosition(arg2).toString();
}

@Override
public void onNothingSelected(AdapterView<?> arg0) {
// TODO Auto-generated method stub
}

});
}

//MÃ©todos de la interfaz TextWatcher
@Override
public void afterTextChanged(Editable s) {
// TODO Auto-generated method stub
}

@Override
public void beforeTextChanged(CharSequence s, int start, int count,
int after) {
// TODO Auto-generated method stub
}
```

```
@Override
public void onTextChanged(CharSequence s, int start, int before, int count) {
// TODO Auto-generated method stub
//selection.setText(edit.getText());
}

}
```

SolucionMapa.java

```
package uam.vhm.pt;

import java.util.ArrayList;

import com.google.android.gms.maps.CameraUpdate;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.model.CameraPosition;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.PolylineOptions;
import com.google.android.gms.maps.SupportMapFragment;
import android.annotation.SuppressLint;
import android.graphics.Color;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import uam.vhm.pt.bean.Solucion;
import uam.vhm.pt.bean.Estacion;

public class SolucionMapa extends android.support.v4.app.FragmentActivity {

private GoogleMap mapa = null;

@SuppressLint("NewApi")
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.solucion_mapa);

mapa = ((SupportMapFragment) getSupportFragmentManager()
```

```
.findFragmentById(R.id.map)).getMap();

//Recivimos los datos provenientes de otra actividad
Solucion sol = getIntent().getParcelableExtra("solucion");

if(sol != null)
{
ArrayList<Estacion> estaciones = sol.getSolucion();

if(estaciones != null && estaciones.size() > 0)
drawSolution(estaciones);



else
Log.d("MSGLOG","El estaciones es null o no tiene elementos metodo onCreate, clase:"+getClass().getName());
}

else
Log.d("MSGLOG","El objeto sol es null metodo onCreate, clase:"+getClass().getName());
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.solucion_mapa, menu);
return true;
}

private void drawSolution(ArrayList<Estacion> estaciones)
{
//Creamos un objecto PolylineOptions para dibujar el recorrido con lineas
PolylineOptions lineas = new PolylineOptions();
lineas.width(8);
lineas.color(Color.RED);

for(int i=0;i<estaciones.size();i++)
{
//añadimos un marker por cada estacion
mapa.addMarker(new MarkerOptions()
```

```
.position(new LatLng(estaciones.get(i).getLatitud(), estaciones.get(i).getLongitud()))
.title(estaciones.get(i).getNombre()));

//.title(estaciones.get(i).getNombre())
//.icon(BitmapDescriptorFactory.fromResource(R.drawable.aculco)));

//Dibujo con Lineas
lineas.add(new LatLng(estaciones.get(i).getLatitud(), estaciones.get(i).getLongitud()));
mapa.addPolyline(lineas);
}

CameraPosition camPos = new CameraPosition.Builder()
.target(new LatLng(estaciones.get(0).getLatitud(), estaciones.get(0).getLongitud()))
.zoom(13)         //Establecemos el zoom en 19
//.bearing(45)      //Establecemos la orientaciÃ³n con el noreste arriba
//.tilt(70)         //Bajamos el punto de vista de la cÃ¡mara 70 grados
.build();

CameraUpdate camUpd3 =
CameraUpdateFactory.newCameraPosition(camPos);

mapa.animateCamera(camUpd3);
//mapa.moveCamera(camUpd3);

}

}
```

SolucionNormal.java

```
package uam.vhm.pt;

import uam.vhm.pt.R;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
import android.app.ListActivity;
import android.content.Intent;
```

```
import android.content.res.Resources;

import uam.vhm.pt.bean.Solucion;


public class SolucionNormal extends ListActivity{

/** Called when the activity is first created. */


Solucion sol;


@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.solucion_normal);


//Recivimos los datos provenientes de otra actividad

sol = getIntent().getParcelableExtra("solucion");


if(sol != null)

{

//Mostramos al usuario la solucion

EstacionesAdapter adaptador = new EstacionesAdapter(this, sol.getSolucion());

setListAdapter(adaptador);


String costoUnidad = getCostoUnidad(sol.getMetrica(), sol.getCosto());


if(costoUnidad != null)

{

Toast t = Toast.makeText(getApplicationContext(), costoUnidad, Toast.LENGTH_LONG);

t.show();

}


else

Log.d("MSGLOG","costoUnidad es null, calse: "+getClass().getName());


}


else

Log.d("MSGLOG","sol es null, calse: "+getClass().getName());


TextView txtSol = (TextView)findViewById(R.id.TxtMensaje3);
```

```
txtSol.setText("");

final Button btnSolMap = (Button)findViewById(R.id.SolMap3);
btnSolMap.setOnClickListener(new View.OnClickListener() {

@Override
public void onClick(View v)
{
// TODO Auto-generated method stub
Intent i1 = new Intent(SolucionNormal.this, SolucionMapa.class);
i1.putExtra("solucion", sol);
startActivity(i1);
}
});
}

//Obtiene el costo junto con unidad en funciÃ³n de la metrica seleccionada
private String getCostoUnidad(String metrica, double costo)
{
//Obtenemos el costo de la metrica
Integer divisor = getDivisor(metrica, costo);
//Obtenemos las unidades.
String unidad = getUnidad(metrica);

/*Log.d("MSGLOG","costo="+costo);
Log.d("MSGLOG","divisor="+divisor);
Log.d("MSGLOG","unidad="+unidad);*/

if(divisor > 0 && unidad != null)
return (costo/divisor)+" "+unidad;

else
{
Log.d("MSGLOG","El divisor de la unidad: "+unidad+" es: "+divisor+" calse: "+getClass().getName());
return null;
}
}

//Obtiene el divisor de la metrica seleccionada
```

```
private Integer getDivisor(String metrica, double costo)
{
//Accedemos al objeto 'Recursos' desde la Activity
Resources res = this.getResources();

String uniMetricas [] = res.getStringArray(R.array.unidades_metricas);
for (String uM: uniMetricas)
{
if(uM.contains(metrica))
{
uM = uM.replaceAll(metrica, "").replaceAll(" ", "").replaceAll("=>", "");

return Integer.parseInt(uM);
}
}

return 0;
}

//Obtiene la unidad de la metrica seleccionada
private String getUnidad(String metrica)
{
if(metrica.contains("Km"))
return " km.";

else if(metrica.contains("Tiempo"))
return " hr.";

else if(metrica.contains("Numero De Estaciones"))
return " estaciones.";

else
return null;
}
}
```

B. Paquete uam.vhm.pt.algoritmos (algoritmos)

Camino.java

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/

package uam.vhm.pt.algoritmos;

import java.util.ArrayList;

/**
*
* @author victor
*/
public class Camino {

private int [][] matrizRecorrido;
private ArrayList<Integer> camino;

public Camino(int [][] matrizRecorrido)
{
this.matrizRecorrido = matrizRecorrido;
}

//Se obtiene el camino recursivamente
private void camino(int origen, int destino)
{
int medio = this.matrizRecorrido[origen-1][destino-1];

if(medio == destino)
camino.add(medio);

else
{
camino(origen, medio);
camino(medio, destino);
}
}
```

```
//MÃ©todo que retorna en n ArrayList<Integer> la mejor ruta entre origen-destino
public ArrayList<Integer> getCamino(int origen, int destino)
{
if((matrizRecorrido.length > 0) && (matrizRecorrido != null))
{
camino = new ArrayList<Integer>();
camino.add(origen);

if(origen != destino)
camino(origen, destino);

if(camino.size() > 0)
return camino;
}

return null;
}

}
```

Floyd.java

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/

package uam.vhm.pt.algoritmos;

import uam.vhm.pt.bean.ResultadoFloyd;

/**
*
* @author victor
*/
public class Floyd {

private Double matDeAdyacencia[][] = null;
private Integer matDeRecorrido[][] = null;
private ResultadoFloyd floyd;
```

```
//para aplicar floyd sobre datos primitivos

private double matDeAd[][] = null;

private int matDeRec[][] = null;


//Para objetos

public Floyd(Double matDeAdyacencia[][], Integer matDeRecorrido[][])

{

this.matDeAdyacencia = matDeAdyacencia;

this.matDeRecorrido = matDeRecorrido;

this.floyd = new ResultadoFloyd();

}


//Para datos primitivos

public Floyd(double matDeAdyacencia[][], int matDeRecorrido[][])

{

this.matDeAd = matDeAdyacencia;

this.matDeRec = matDeRecorrido;

this.floyd = new ResultadoFloyd();

}



//Algorimo de Floyd

public ResultadoFloyd AplicaFloyd()

{

for (int k = 0; k < matDeAdyacencia.length; k++){

for (int i = 0; i < matDeAdyacencia.length; i++){

for (int j = 0; j < matDeAdyacencia.length; j++){


if ((matDeAdyacencia[i][k] != Double.MAX_VALUE)&&(matDeAdyacencia[k][j] != Double.MAX_VALUE)&&(Math.min(matDeAdyacencia[i][j],
matDeAdyacencia[i][k] + matDeAdyacencia[k][j]) != matDeAdyacencia[i][j])){


matDeAdyacencia[i][j] = matDeAdyacencia[i][k] + matDeAdyacencia[k][j];

matDeRecorrido[i][j] = matDeRecorrido[i][k];

}

}

}

}


//Seteamos las marices de distancias y resultados y devolveos en forma de objeto de tipo ResultadoFloyd
```

```
floyd.setMatDistancias(matDeAdyacencia);

floyd.setMatRecorridos(matDeRecorrido);


return floyd;

}



//Algorimo de Floyd

public ResultadoFloyd AplicaFloydSobrePrimitivos()

{

for (int k = 0; k < matDeAd.length; k++){

for (int i = 0; i < matDeAd.length; i++){

for (int j = 0; j < matDeAd.length; j++){


if ((matDeAd[i][k] != Double.MAX_VALUE)&&(matDeAd[k][j] != Double.MAX_VALUE)&&(Math.min(matDeAd[i][j], matDeAd[i][k] + matDeAd[k][j])
!= matDeAd[i][j])){


matDeAd[i][j] = matDeAd[i][k] + matDeAd[k][j];

matDeRec[i][j] = matDeRec[i][k];

}

}

}

}


//Seteamos las marices de distancias y resultados y devolveos en forma de objeto de tipo ResultadoFloyd

floyd.setMatDist(matDeAd);

floyd.setMatRec(matDeRec);


return floyd;

}


}
```

GeneraMatricesDeResultado.java

```
package uam.vhm.pt.algoritmos;


import java.util.ArrayList;


import uam.vhm.pt.bean.Estacion;

import uam.vhm.pt.bean.ResultadoFloyd;
```

```
import uam.vhm.pt.bean.Vecino;
import android.util.Log;

public class GeneraMatricesDeResultado {

                                        private ArrayList<Estacion> est = null;
                                        private String metrica = null;
  private static final String TAG = "MSGLOG";

  public GeneraMatricesDeResultado(ArrayList<Estacion> estaciones, String metrica)
  {
                                            this.est = estaciones;
                                            this.metrica  = metrica;
  }

  //metodo que construye las matrices de adyacencia y distancias, posterior menete aplica floy
  // y devuelve el resultado en la forma del bean ResultadoFloyd
                                        public ResultadoFloyd generaMatrices()
  {
                                            double matAdy [][] = null;
                                          int matRec [][] = null;

    if(est.size() > 0)
    {
      matAdy = new double [est.size()][est.size()];
      matRec = new int [est.size()][est.size()];

      Log.d(TAG,"matAdy["+est.size()+"]["+est.size()+"]");
      Log.d(TAG,"matAdy["+est.size()+"]["+est.size()+"]");

      //Inicializamos la matriz de distancias(adyacencias) y la de recorridos
      for(int i=0;i<est.size();i++)
      {
        for(int j=0;j<est.size();j++)
        {
          if(i == j)
          {
            matAdy[i][j] = 0.0;
            matRec[i][j] = 0;
```

```
            }


            else
            {
               matAdy[i][j] = Double.POSITIVE_INFINITY;
               matRec[i][j] = j+1;
            }
         }
      }


      //Formamos la matriz de adyacendia o distancias
      for(int i=0;i<est.size();i++)
      {
         ArrayList<Vecino> v = est.get(i).getVecinos();


         if(v.size() > 0)
         {
            for(int j=0;j<v.size();j++)
            {
               //generamos los elementos correspondientes a cada metrica
               if(metrica.equals(("Minimizando Km")))
                  matAdy[est.get(i).getNumero()-1][v.get(j).getNumero()-1] = v.get(j).getDistancia();


               else if(metrica.equals("Minimizando Numero De Estaciones"))
                  matAdy[est.get(i).getNumero()-1][v.get(j).getNumero()-1] = 1.0;


               else if(metrica.equals("Minimizando Tiempo"))
                  matAdy[est.get(i).getNumero()-1][v.get(j).getNumero()-1] = (v.get(j).getDistancia()/1000)/35.5;


               else{
                                                   Log.d(TAG,"No se pudioerón generar las matrices de resultado ya que no se conoce la metrica en curso =>
"+metrica+", método generaMatricesDeResultado(), clase: "+getClass().getName());
                  return null;
               }
            }
         }
      }


      //Aplicamos el algoritmo de floyd, obtenemos la matriz de distancias y la de recorridos
```

```
        Floyd floyd = new Floyd(matAdy, matRec);
        ResultadoFloyd resultado = floyd.AplicaFloydSobrePrimitivos();

        //matAdy = resultado.getMatDist();
        //matRec = resultado.getMatRec();

        return resultado;
      }

      else
      {
                                          Log.d(TAG,"No se pudioerón generar las matrices de resultado ya que est.size() < 0, método generaMatricesDeResultado(), clase: "+getClass().getName());
        return null;
      }
   }


}
```

LeeMatriz.java

```
package uam.vhm.pt.algoritmos;

/**
*
* @author victor
*/

import java.io.*;
import java.util.ArrayList;

import android.content.Context;
import android.widget.Toast;

/*
* Esta clase sirve para obtener los archivos que contienen las matrices de resultado
* las matrices se pueden obtener como un double[][] o un int[][]
* la intencion es que la matriz double[][] corresponda a la matriz de distancias
* y la matriz de int[][] corresponda a la matriz de recorridos
```

```
* el constructor recive el contexto de la actividad y el numbre del archivo a convertir
*/
public class LeeMatriz {

private String rutaArchivo = null;
private Context contexto = null;

public LeeMatriz(Context contexto, String rutaArchivo)
{
this.contexto = contexto;
this.rutaArchivo = rutaArchivo;
}

//MÃ©todo que pasa la matriz que esta en el archivo a un arrayList<String>
private ArrayList<String> fixheroToArrayList()
{
try{

//Lectura del archivo
InputStream in = contexto.getAssets().open(rutaArchivo);

if (in != null)
{
InputStreamReader input = new InputStreamReader(in);
BufferedReader buffreader = new BufferedReader(input);

String strLinea;

ArrayList<String> lineas = new ArrayList<String>();

// Leer el archivo linea por linea
while ((strLinea = buffreader.readLine()) != null)   {
//Obtenemos cada elemnto del archivo exceto los espacios en blanco
lineas.add(strLinea);
}
// Cerramos el archivo
input.close();

return lineas;
```

```
}

else
{
Toast.makeText(contexto.getApplicationContext(), "Ocurrio un error: en la lectura del archivo => InputStream in = null", Toast.LENGTH_SHORT).show();
}
}
catch (Exception e){
Toast.makeText(contexto.getApplicationContext(), "Ocurrio un error: "+e.getMessage(), Toast.LENGTH_SHORT).show();
}

return null;
}

//MÃ©todo que devuelve la matriz contenida en el archivo en forma de matriz de doubles
public double[][] obtenMatrizDoubles()
{

ArrayList<String> lineas = fixheroToArrayList();

if(lineas != null)
{
double matrizDeDistancias [][] = new double[lineas.size()][lineas.size()];

for(int i=0;i<lineas.size();i++)
{
String [] elementos =  lineas.get(i).split(" ");
for(int j=0;j<elementos.length;j++)
{
matrizDeDistancias[i][j] = Double.parseDouble(elementos[j]);
}
}

return matrizDeDistancias;
}

else
return null;
}
```

```
//MÃ©todo que devuelve la matriz contenida en el archivo en forma de matriz de enteros

public int[][] obtenMatrizEnteros()

{

ArrayList<String> lineas = fixheroToArrayList();


if(lineas != null)

{

int DeRecorridos [][] = new int[lineas.size()][lineas.size()];


for(int i=0;i<lineas.size();i++)

{

String [] elementos =  lineas.get(i).split(" ");

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

{

DeRecorridos[i][j] = Integer.parseInt(elementos[j]);

}

}


return DeRecorridos;

}


else

return null;

}


}package uam.vhm.pt.algoritmos;


/**

*

* @author victor

*/


import java.io.*;

import java.util.ArrayList;


import android.content.Context;

import android.widget.Toast;
```

```
/*
* Esta clase sirve para obtener los archivos que contienen las matrices de resultado
* las matrices se pueden obtener como un double[][] o un int[][]
* la intencion es que la matriz double[][] corresponda a la matriz de distancias
* y la matriz de int[][] corresponda a la matriz de recorridos
* el constructor recive el contexto de la actividad y el numbre del archivo a convertir
*/
public class LeeMatriz {

private String rutaArchivo = null;
private Context contexto = null;

public LeeMatriz(Context contexto, String rutaArchivo)
{
this.contexto = contexto;
this.rutaArchivo = rutaArchivo;
}

//MÃ©todo que pasa la matriz que esta en el archivo a un arrayList<String>
private ArrayList<String> fixheroToArrayList()
{
try{

//Lectura del archivo
InputStream in = contexto.getAssets().open(rutaArchivo);

if (in != null)
{
InputStreamReader input = new InputStreamReader(in);
BufferedReader buffreader = new BufferedReader(input);

String strLinea;

ArrayList<String> lineas = new ArrayList<String>();

// Leer el archivo linea por linea
while ((strLinea = buffreader.readLine()) != null)   {
//Obtenemos cada elemnto del archivo exceto los espacios en blanco
lineas.add(strLinea);
```

```
}
// Cerramos el archivo
input.close();

return lineas;
}

else
{
Toast.makeText(contexto.getApplicationContext(), "Ocurrio un error: en la lectura del archivo => InputStream in = null", Toast.LENGTH_SHORT).show();
}
}
catch (Exception e){
Toast.makeText(contexto.getApplicationContext(), "Ocurrio un error: "+e.getMessage(), Toast.LENGTH_SHORT).show();
}

return null;
}

//MÃ©todo que devuelve la matriz contenida en el archivo en forma de matriz de doubles
public double[][] obtenMatrizDoubles()
{

ArrayList<String> lineas = fixheroToArrayList();

if(lineas != null)
{
double matrizDeDistancias [][] = new double[lineas.size()][lineas.size()];

for(int i=0;i<lineas.size();i++)
{
String [] elementos =  lineas.get(i).split(" ");
for(int j=0;j<elementos.length;j++)
{
matrizDeDistancias[i][j] = Double.parseDouble(elementos[j]);
}
}

return matrizDeDistancias;
```

```
}

else
return null;
}

//MÃ©todo que devuelve la matriz contenida en el archivo en forma de matriz de enteros
public int[][] obtenMatrizEnteros()
{
ArrayList<String> lineas = fixheroToArrayList();

if(lineas != null)
{
int DeRecorridos [][] = new int[lineas.size()][lineas.size()];

for(int i=0;i<lineas.size();i++)
{
String [] elementos =  lineas.get(i).split(" ");
for(int j=0;j<elementos.length;j++)
{
DeRecorridos[i][j] = Integer.parseInt(elementos[j]);
}
}

return DeRecorridos;
}

else
return null;
}

}
```

Parser.java

```
package uam.vhm.pt.algoritmos;

import java.util.ArrayList;
import org.xmlpull.v1.XmlPullParser;

import uam.vhm.pt.bean.Estacion;
```

```
import uam.vhm.pt.bean.Vecino;

//import android.util.Log;


public class Parser {


private XmlPullParser parser;

//  private static final String TAG = "TratamientoXML";


public Parser(XmlPullParser parser)

{

this.parser  = parser;

//                                                      Log.d(TAG, "entro al constructor");

}


//MÃ©todo que deserializa el xml

public ArrayList<Estacion> parse()

{

ArrayList<Estacion> estaciones = null;


try

{

int evento = parser.getEventType();


Estacion estacionActual = null;

Vecino vecinoActual = null;


//mientras no se llegue al fin del xml

while (evento != XmlPullParser.END_DOCUMENT)

{

String etiqueta = null;


switch (evento)

{

//al inicio del xml

case XmlPullParser.START_DOCUMENT:


estaciones = new ArrayList<Estacion>();

break;
```

```
//al inicio de cada tag
case XmlPullParser.START_TAG:

etiqueta = parser.getName();

//cuanso se eencontro un nodo estacion
if (etiqueta.equals("estacion"))
{
estacionActual = new Estacion();

//obtenemos todos los atributos del nodo estacion en curso
for(int i=0;i<parser.getAttributeCount();i++)
{
if(parser.getAttributeName(i).equals("nombre"))
estacionActual.setNombre(parser.getAttributeValue(i));

if(parser.getAttributeName(i).equals("latitud"))
estacionActual.setLatitud(Double.parseDouble(parser.getAttributeValue(i)));

if(parser.getAttributeName(i).equals("longitud"))
estacionActual.setLongitud(Double.parseDouble(parser.getAttributeValue(i)));

if(parser.getAttributeName(i).equals("numero"))
estacionActual.setNumero(Integer.parseInt(parser.getAttributeValue(i)));

}
}

else if (estacionActual != null)
{
//sis e trata de un nodo vecino
if (etiqueta.equals("vecino"))
{
vecinoActual = new Vecino();

//obtenemos todos los atributos del nodo vecino en curso
for(int i=0;i<parser.getAttributeCount();i++)
{
if(parser.getAttributeName(i).equals("nombre"))
```

```
vecinoActual.setNombre(parser.getAttributeValue(i));

if(parser.getAttributeName(i).equals("numero"))
vecinoActual.setNumero(Integer.parseInt(parser.getAttributeValue(i)));

if(parser.getAttributeName(i).equals("distancia"))
vecinoActual.setDistancia(Double.parseDouble(parser.getAttributeValue(i)));

}

}
}

break;

//al encontra el fin de cada tag vamos llenando los arrayList
case XmlPullParser.END_TAG:

etiqueta = parser.getName();

if(etiqueta.equals("estacion") && estacionActual != null)
{
estaciones.add(estacionActual);
}

if(etiqueta.equals("vecino") && estacionActual != null)
{
estacionActual.addVecino(vecinoActual);
}

break;
}

evento = parser.next();
}
}
catch (Exception ex)
{
throw new RuntimeException(ex);
```

```
}

return estaciones;
}

}
```

Permutaciones.java

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/

package uam.vhm.pt.algoritmos;

import java.util.ArrayList;

import uam.vhm.pt.bean.Estacion;

/**
*
* @author victor
*/

public class Permutaciones
{
ArrayList<String> permutaciones = null;
private ArrayList<Estacion> estaciones = null;

public Permutaciones(ArrayList<Estacion> estaciones)
{
this.estaciones = estaciones;
}

//Se obtienen todas las permutaciones posibles de los elementos
private void generaPermutaciones(String a, ArrayList<Estacion> estaciones)
{
if (estaciones.size()==1)
permutaciones.add(a+estaciones.get(0).getNumero());
```

```
for (int i=0;i<estaciones.size();i++)
{
Estacion est = estaciones.remove(i);
generaPermutaciones(a+est.getNumero()+",", estaciones);
estaciones.add(i,est);
}
}

public ArrayList<String> getPermutaciones()
{
if(this.estaciones.size() > 0)
{
permutaciones = new ArrayList<String>();
generaPermutaciones("", estaciones);
}

return permutaciones;
}
}/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/

package uam.vhm.pt.algoritmos;

import java.util.ArrayList;

import uam.vhm.pt.bean.Estacion;

/**
*
* @author victor
*/

public class Permutaciones
{
ArrayList<String> permutaciones = null;
private ArrayList<Estacion> estaciones = null;
```

```
public Permutaciones(ArrayList<Estacion> estaciones)
{
this.estaciones = estaciones;
}

//Se obtienen todas las permutaciones posibles de los elementos
private void generaPermutaciones(String a, ArrayList<Estacion> estaciones)
{
if (estaciones.size()==1)
permutaciones.add(a+estaciones.get(0).getNumero());

for (int i=0;i<estaciones.size();i++)
{
Estacion est = estaciones.remove(i);
generaPermutaciones(a+est.getNumero()+",", estaciones);
estaciones.add(i,est);
}
}

public ArrayList<String> getPermutaciones()
{
if(this.estaciones.size() > 0)
{
permutaciones = new ArrayList<String>();
generaPermutaciones("", estaciones);
}

return permutaciones;
}
}
```

RelacionMetricaArchivo.java

```
package uam.vhm.pt.algoritmos;

import uam.vhm.pt.R;
import android.content.Context;
import android.content.res.Resources;

public class RelacionMetricaArchivo {
```

```
private Context conexto = null;

private String archivoDistancia = null;

private String archivoRecorrido = null;


public RelacionMetricaArchivo(Context conexto)

{

this.conexto = conexto;


if(this.conexto != null && this.conexto instanceof Context)

{

Resources res = this.conexto.getResources();

archivoDistancia = res.getString(R.string.archivoDistancia);

archivoRecorrido = res.getString(R.string.archivoRecorrido);

}

}


//Metodos que forma el nombre del archivo(que contiene una matriz de resultado) en función de la

//metrica entrante.

public String getArchivoDistanciasDandoMetrica(String metrica)

{

if(archivoDistancia != null && archivoDistancia.length() > 0 && metrica != null && metrica.length() > 0)

{

return getNombreDelArchivo(archivoDistancia+""+metrica);

}


return null;

}


public String getArchivoRecorridosDandoMetrica(String metrica)

{

if(archivoRecorrido != null && archivoRecorrido.length() > 0 && metrica != null && metrica.length() > 0)

{

return getNombreDelArchivo(archivoRecorrido+metrica);

}


return null;

}


//Metodo que simplemente quita los espacios en blanco de la cadena entrante
```

```
private String getNombreDelArchivo(String archivo)
{
if(archivo != null && archivo.length() > 0)
{
return archivo.replaceAll(" ", "");
}

return null;
}

}
```

SolucionEvitaEsaciones.java

```
package uam.vhm.pt.algoritmos;

import java.util.ArrayList;

import uam.vhm.pt.bean.Estacion;
import uam.vhm.pt.bean.ResultadoFloyd;
import uam.vhm.pt.bean.Solucion;
import uam.vhm.pt.bean.Vecino;
import android.util.Log;

public class SolucionEvitaEsaciones {

private Estacion estacionOrg = null;
private Estacion estacionDes = null;
private ArrayList<Estacion> estaciones = null;
private String metrica = null;
private static final String TAG = "MSGLOG";

double matAdy [][] = null;
int matRec [][] = null;


public SolucionEvitaEsaciones(Estacion estOrig, Estacion estDest, ArrayList<Estacion> estaciones, String metrica)
{
this.estacionOrg = estOrig;
this.estacionDes = estDest;
this.estaciones = estaciones;
```

```
this.metrica  = metrica;
}

//metodo que obytiene la mejor ruta, contemplando estaciones a evitar
public Solucion getMejorRuta(ArrayList<Estacion> evitaEst)
{
cambiaCostodeVecinos(evitaEst);

//Generamos las matrices resultado
GeneraMatricesDeResultado gen = new GeneraMatricesDeResultado(estaciones, metrica);
ResultadoFloyd rF = gen.generaMatrices();

if(rF != null)
{
matAdy = rF.getMatDist();
matRec = rF.getMatRec();

if( (matRec != null) && (matRec.length > 0) && (matAdy != null) && (matAdy.length > 0) )
{
ArrayList<Integer> camino = null;
camino = new ArrayList<Integer>();
Camino c = new Camino(matRec);
camino = c.getCamino(estacionOrg.getNumero(), estacionDes.getNumero());

double costo = 0.0;

if(camino != null & camino.size() > 0)
{
//Obtenemos el costo del camino
costo  += matAdy[estacionOrg.getNumero()-1][estacionDes.getNumero()-1];

//guardamos el camino
ArrayList<Estacion> listaDeEsaciones = new ArrayList<Estacion>();
for(int i=0;i<camino.size();i++)
{
listaDeEsaciones.add(estaciones.get(camino.get(i).intValue()-1));
}

//Seteamos la solución
```

```
Solucion sol = new Solucion(listaDeEsaciones, metrica, costo);
return sol;
}

else
{
Log.d(TAG,"el array camino es null o no tiene elementos, método getMejorRuta(), clase: "+getClass().getName());
return null;
}
}

else
{
Log.d(TAG,"Alguna o la  matrcies de resultado son null o no tienen elementos, mÃ©todo getMejorRuta(), clase: "+getClass().getName());
return null;
}
}

Log.d(TAG,"No se pudioerón generar las matrices de resultado, método getMejorRuta(), clase: "+getClass().getName());
return null;
}

//Metodo que setea a infinito los costos de los vecinos a las estaciones por las que no se desea pasar
private void cambiaCostodeVecinos(ArrayList<Estacion> evitaEst)
{
//cambiamos el costo de los vecios de las estaciones por las que no se desea pasar
for(int i=0;i<evitaEst.size();i++)
{
//Obtengo los vecinos de la n-esima estacion por la que no se desa pasar
ArrayList<Vecino> vecinos = evitaEst.get(i).getVecinos();

for(int j=0;j<vecinos.size();j++)
{
//Obtenemos la estacion del vecino en curso
Estacion estAct = estaciones.get(vecinos.get(j).getNumero()-1);

//Obtenemos los vecinos de la estaciÃ³n en curso(los vecinos del vecino en curso)
ArrayList<Vecino> vecinosAct = estAct.getVecinos();
```

```
//removevemos la estaciÃ³n para sustituirla mas adelante
estaciones.remove(vecinos.get(j).getNumero()-1);

//Obtenemos los vecinos de cada vecino
for(int z=0;z<vecinosAct.size();z++)
{
//Obtenemos solo el vecino que nos interesa
if(vecinosAct.get(z).getNumero() == evitaEst.get(i).getNumero())
{
//sustituimos el vecino por el que tiene costo infinito
Vecino sustituto = vecinosAct.get(z);
sustituto.setDistancia(Double.POSITIVE_INFINITY);

vecinosAct.remove(z);
vecinosAct.add(z, sustituto);
}



}
//seteamos los vecinos en la estacion y esta la sustituimos en el ArrayList estaciones
estAct.setVecinos(vecinosAct);
estaciones.add(vecinos.get(j).getNumero()-1, estAct);
}
}

//cambiamos el costo de los vecios de las estaciones por las que no se desea pasar
for(int i=0;i<estaciones.size();i++)
{
ArrayList<Vecino> v = estaciones.get(i).getVecinos();

for(int j=0;j<v.size();j++)
{
if(v.get(j).getDistancia() == Double.POSITIVE_INFINITY)
{
Log.d(TAG,"EstaciÃ³n: "+estaciones.get(i).getNombre());
Log.d(TAG,"Vacino: "+v.get(j).getNombre());
Log.d(TAG," ");
}
}
```

```
}
}

public ArrayList<Estacion> cambiaCostosEnLaRedDelMetro(ArrayList<Estacion> evitaEst)
{
if(evitaEst != null && evitaEst.size() > 0)
{
cambiaCostodeVecinos(evitaEst);

return estaciones;
}

else
{
return null;
}
}
}
```

SolucionForzaEstaciones.java

```
package uam.vhm.pt.algoritmos;

import java.util.ArrayList;

import android.content.Context;
import android.util.Log;
import uam.vhm.pt.bean.*;
//import android.util.Log;

import uam.vhm.pt.bean.Estacion;

public class SolucionForzaEstaciones {

private Estacion estacionOrg = null;
private Estacion estacionDes = null;
private ArrayList<Estacion> estaciones = null;
private String metrica = null;
private Context contexto = null;
private static final String TAG = "MSGLOG";
```

```
ArrayList<String> permutaciones = null;
double [][] matDist;
int [][] matRec;

public SolucionForzaEstaciones(Estacion estOrig, Estacion estDest, ArrayList<Estacion> estaciones, String metrica, Context context)
{
this.estacionOrg = estOrig;
this.estacionDes = estDest;
this.estaciones = estaciones;
this.metrica  = metrica;
this.contexto = context;
}

// Método que decide cual es la mejor permutación (la de menor costo) y a partir de esta retorna la mejor
// solución considerando la permutación con mnor costo de ArrayList<Estacion>.
public Solucion getMajorRutaPorPermutacion(ArrayList<Estacion> estForz, int evitaForza)
{
if( (estForz != null) && (estForz.size() > 0) )
{
double mejorCosto = Double.POSITIVE_INFINITY;
int pos = 0;

//se obtienen tosas la permutaciones
Permutaciones per = new Permutaciones(estForz);
permutaciones = per.getPermutaciones();

//Obtenemos la matriz de distancias en funcion de la metrica tenida

////Caso EvitaForzaEstaciones
if(evitaForza == 1)
{
GeneraMatricesDeResultado gen = new GeneraMatricesDeResultado(estaciones, metrica);
ResultadoFloyd rFloyd = gen.generaMatrices();

matDist = rFloyd.getMatDist();
matRec = rFloyd.getMatRec();
}
```

```
////Caso EvitaForzaEstaciones
else
{
RelacionMetricaArchivo rel = new RelacionMetricaArchivo(contexto);
String arhivoDist = rel.getArchivoDistanciasDandoMetrica(metrica);
String arhivoRec = rel.getArchivoRecorridosDandoMetrica(metrica);

LeeMatriz ficheroDist = new LeeMatriz(contexto, arhivoDist);
matDist = ficheroDist.obtenMatrizDoubles();

LeeMatriz ficheroRec = new LeeMatriz(contexto, arhivoRec);
matRec = ficheroRec.obtenMatrizEnteros();
}

//Obtenemos la mejor permutación (la de menor costo)
if( (matDist != null) && (matDist.length > 0) )
{
double cost = 0.0;

if(permutaciones.size() > 1)
{
for(int i=0;i<permutaciones.size();i++)
{
String [] p = permutaciones.get(i).split(",");

cost = 0.0;

for(int j=0;j<p.length;j++)
{
if(j == 0)
{
cost += matDist[estacionOrg.getNumero()-1][estaciones.get(Integer.parseInt(p[j])-1).getNumero()-1];

cost += matDist[estaciones.get(Integer.parseInt(p[j])-1).getNumero()-1][estaciones.get(Integer.parseInt(p[j+1])-1).getNumero()-1];
}

else if( j == (p.length-1) )
{
```

```
cost += matDist[estaciones.get(Integer.parseInt(p[p.length-1])-1).getNumero()-1][estacionDes.getNumero()-1];
}

else
{
cost += matDist[estaciones.get(Integer.parseInt(p[j])-1).getNumero()-1][estaciones.get(Integer.parseInt(p[j+1])-1).getNumero()-1];
}

}

if(cost < mejorCosto)
{
//Log.d(TAG,"Entro cuando i="+i);
mejorCosto = cost;
pos = i;
}
}
}

else//cuando solo se selecciono un estacion para forzar el paso por esta
{
mejorCosto = 0;
pos = 0;
}
}

return getSolucion(mejorCosto, pos);
}

else
{
Log.d(TAG, "matDist = null o sin elementos. método getMajorRutaPorPermutacion clase "+this.getClass());
return null;
}
}

//Método que llena el ArrayList<Estacion> de solución en función de la mejor permutación
private Solucion getSolucion(double mejorCosto, int pos)
{
```

```
ArrayList<Estacion> solucion = null;

if( (permutaciones != null) && (permutaciones.size() > 0) && (matDist != null) && (matDist.length > 0) )
{
solucion = new ArrayList<Estacion>();
String [] per = permutaciones.get(pos).split(",");

solucion.add(estacionOrg);

for(int j=0;j<per.length;j++)
solucion.add(estaciones.get(Integer.parseInt(per[j])-1));

solucion.add(estacionDes);

if( (solucion != null) && (solucion.size() > 0) )
{
if( (matRec != null) && (matRec.length > 0) )
{
ArrayList<Integer> camino = null;
camino = new ArrayList<Integer>();
Camino c = new Camino(matRec);
ArrayList<Estacion> listaDeEsaciones = new ArrayList<Estacion>();
double costo = 0.0;

//vamos obteneiendo las distancas mas cortas y los costos correspondientesa estas
for(int i=0;i<solucion.size()-1;i++)
{
camino = c.getCamino(solucion.get(i).getNumero(), solucion.get(i+1).getNumero());
costo  += matDist[solucion.get(i).getNumero()-1][solucion.get(i+1).getNumero()-1];
//Log.d(TAG, solucion.get(i).getNombre()+"-"+solucion.get(i+1).getNombre()+" => costo = "+costo);

for(int j=0;j<camino.size();j++)
{
//Para eliminar repetidos ya que por in saacando ruta de origen-destino en ocaciones por las permutaciones
//unas estaciones actuan primero como destino y luego como origen.
if(j > 0) || (i == 0) )
{
listaDeEsaciones.add(estaciones.get(camino.get(j).intValue()-1));
Log.d(TAG,"-> "+estaciones.get(camino.get(j).intValue()-1).getNombre());
```

```
}

}

}


Solucion s = new Solucion(listaDeEsaciones, metrica, costo);


return s;

}


else

{

Log.d(TAG, "matRec = null o sin elementos. mÃ©todo getSolucion clase: "+this.getClass().getName());

return null;

}

}


else

{

Log.d(TAG, "o matDist = null o sin elementos o no se obtuvo el arhivoRec o el ArrayList solucion no se obtuvo correctamente, mÃ©todo getSolucion clase
"+this.getClass().getName());

return null;

}

}


return null;

}


}
```

SolucionNormal.java

```
package uam.vhm.pt.algoritmos;


import java.util.ArrayList;

import android.content.Context;

import android.util.Log;

import uam.vhm.pt.bean.Estacion;

import uam.vhm.pt.bean.Solucion;


public class SolucionNormal {
```

```
private Estacion estacionOrg = null;

private Estacion estacionDes = null;

private ArrayList<Estacion> estaciones = null;

private String metrica = null;

private Context contexto = null;


ArrayList<Integer> camino = null;

double costo = 0.0;

private static final String TAG = "MSGLOG";


public SolucionNormal(ArrayList<Estacion> estaciones, Estacion estacionOrg, Estacion estacionDes, String metrica, Context contexto)

{

this.estaciones = estaciones;

this.estacionOrg = estacionOrg;

this.estacionDes = estacionDes;

this.metrica = metrica;

this.contexto = contexto;

}


//MÃ©todo que obtiene la sol dado un origen y un destino, y devuelve la solucion en forma de un objeto de tipo Solucion

public Solucion getSolucion()

{

if( (estaciones != null) && (estaciones.size() > 0) && (estacionOrg != null) && (estacionDes != null) && (metrica != null) && (metrica.length() > 0) )

{

//Obtenemos las matrices de distancias y recorridos

RelacionMetricaArchivo rel = new RelacionMetricaArchivo(this.contexto);

String arhivoDist = rel.getArchivoDistanciasDandoMetrica(metrica);

String arhivoRec = rel.getArchivoRecorridosDandoMetrica(metrica);


LeeMatriz ficheroDist = new LeeMatriz(this.contexto, arhivoDist);

LeeMatriz ficheroRec = new LeeMatriz(this.contexto, arhivoRec);


double [][] matDist = ficheroDist.obtenMatrizDoubles();

int [][] matRec = ficheroRec.obtenMatrizEnteros();


//Obtenemos la mejor ruta en funcion de la metrica tenida

camino = new ArrayList<Integer>();

Camino c = new Camino(matRec);

camino = c.getCamino(estacionOrg.getNumero(), estacionDes.getNumero());
```

```
//guardamos el camino
ArrayList<Estacion> listaDeEsaciones = new ArrayList<Estacion>();
for(int i=0;i<camino.size();i++)
{
listaDeEsaciones.add(estaciones.get(camino.get(i).intValue()-1));
}

//Obtenemos el costo del camino
costo = matDist[estacionOrg.getNumero()-1][estacionDes.getNumero()-1];

//Retornamos la solucion
if( (listaDeEsaciones != null) && (listaDeEsaciones.size() > 0) && (costo > 0.0) )
{
Solucion sol = new Solucion(listaDeEsaciones, metrica, costo);
return sol;
}

Log.d(TAG,"listaDeEsaciones es null o no tiene elementos o costo es 0.0 calse: "+getClass().getName());
return null;
}

else
{
Log.d(TAG,"Algun elemento es null o algun ArrayLust no tiene elementos, (estaciones, estacionOrg, estacionDes, metrica) clase: "+getClass().getName());
return null;
}
}
}
```

C. Paquete uam.vhm.pt.bean (clases de persistencia)

Estacion.java

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/
```

```
package uam.vhm.pt.bean;

import java.util.ArrayList;
import android.os.Parcel;
import android.os.Parcelable;

/**
*
* @author victor
*/
public class Estacion implements Parcelable{

private String nombre;
private Double latitud;
private Double longitud;
private int numero;
private ArrayList<Vecino> vecinos;

public Estacion(String nombre, Double latitud, Double longitud, int numero, ArrayList<Vecino> vecinos) {
this.nombre = nombre;
this.latitud = latitud;
this.longitud = longitud;
this.numero = numero;
this.vecinos = vecinos;
}

public Estacion()
{
this.vecinos = new ArrayList<Vecino>();
}

//////////////////////////////////////////////////////////////
public Estacion(Parcel source)
{
nombre = source.readString();
latitud = source.readDouble();
longitud = source.readDouble();
```

```
numero = source.readInt();
this.vecinos = new ArrayList<Vecino>();
source.readTypedList(vecinos, Vecino.CREATOR);
}

//Para poder crear una lista de objetos tipo Estacion y pasarlos por entre actividades
public static final Parcelable.Creator<Estacion> CREATOR = new Parcelable.Creator<Estacion>() {
public Estacion createFromParcel(Parcel in) {
return new Estacion(in);
}

public Estacion[] newArray(int size) {
return new Estacion[size];
}
};

public int describeContents() {
return 0;
}

public void writeToParcel(Parcel dest, int flags) {
dest.writeString(nombre);
dest.writeDouble(latitud);
dest.writeDouble(longitud);
dest.writeInt(numero);
dest.writeTypedList(vecinos);
}
//////////////////////////////////////////////////////////////////
```

```
public Double getLatitud() {
return latitud;
}

public void setLatitud(Double latitud) {
this.latitud = latitud;
}
```

```
public Double getLongitud() {
return longitud;
}

public void setLongitud(Double longitud) {
this.longitud = longitud;
}

public String getNombre() {
return nombre;
}

public void setNombre(String nombre) {
this.nombre = nombre;
}

public int getNumero() {
return numero;
}

public void setNumero(int numero) {
this.numero = numero;
}

public ArrayList<Vecino> getVecinos()
{
return vecinos;
}

public void addVecino(Vecino vecino)
{
vecinos.add(vecino);
}

public void setVecinos(ArrayList<Vecino> vecinos)
{
this.vecinos = vecinos;
}
```

```
@Override
public String toString() {
return nombre;
}

}
```

LineaMetro.java

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/

package uam.vhm.pt.bean;

import java.util.ArrayList;

/**
*
* @author victor
*/
public class LineaMetro {

private String nombre;
private ArrayList<Estacion> estaciones;

public LineaMetro(String nombre) {
this.nombre = nombre;
}

public ArrayList<Estacion> getEstaciones() {
return estaciones;
}

public void addEstaciones(Estacion estaciones) {
this.estaciones.add(estaciones);
}

public void setEstaciones(ArrayList<Estacion> estaciones) {
this.estaciones = estaciones;
```

```
}

public String getNombre() {
return nombre;
}

public void setNombre(String nombre) {
this.nombre = nombre;
}

}
```

ResultadoFloyd.java

```
/*
* To change this template, choose Tools | Templates
* and open the template in the editor.
*/

package uam.vhm.pt.bean;

/**
*
* @author victor
*/

//Clase bean para representacion del resultado del algoritmo de Floyd
public class ResultadoFloyd {

private Double matDistancias[][];
private Integer matRecorridos[][];

//para datos primitivos
private double matDist[][];
private int matRec[][];

public ResultadoFloyd() {
}

public Double[][] getMatDistancias() {
return matDistancias;
```

```
}

public void setMatDistancias(Double[][] matDistancias) {
this.matDistancias = matDistancias;
}

public Integer[][] getMatRecorridos() {
return matRecorridos;
}

public void setMatRecorridos(Integer[][] matRecorridos) {
this.matRecorridos = matRecorridos;
}


//Para tipos primitivos
public double[][] getMatDist() {
return matDist;
}

public void setMatDist(double[][] matDist) {
this.matDist = matDist;
}

public int[][] getMatRec() {
return matRec;
}

public void setMatRec(int[][] matRec) {
this.matRec = matRec;
}

}
```

Solucion.java

```
package uam.vhm.pt.bean;

import java.util.ArrayList;
import android.os.Parcel;
import android.os.Parcelable;
```

```
public class Solucion implements Parcelable{

private ArrayList<Estacion> solucion = null;
private String metrica = null;
private double costo = 0.0;

public Solucion()
{

}

public Solucion(ArrayList<Estacion> solucion, String metrica, double costo)
{
this.solucion = solucion;
this.metrica = metrica;
this.costo = costo;
}

public Solucion(Parcel source)
{
solucion = new ArrayList<Estacion>();
source.readTypedList(solucion, Estacion.CREATOR);
metrica = source.readString();
costo = source.readDouble();
}

//Para poder crear una lista de objetos tipo Solucion y pasarlos por entre actividades
public static final Parcelable.Creator<Solucion> CREATOR = new Parcelable.Creator<Solucion>() {
public Solucion createFromParcel(Parcel in) {
return new Solucion(in);
}

public Solucion[] newArray(int size) {
return new Solucion[size];
}
};

@Override
```

```
public int describeContents() {
// TODO Auto-generated method stub
return 0;
}

@Override
public void writeToParcel(Parcel arg0, int arg1) {
// TODO Auto-generated method stub
arg0.writeTypedList(solucion);
arg0.writeString(metrica);
arg0.writeDouble(costo);
}
public ArrayList<Estacion> getSolucion() {
return solucion;
}

public void setSolucion(ArrayList<Estacion> solucion) {
this.solucion = solucion;
}

public String getMetrica() {
return metrica;
}

public void setMetrica(String metrica) {
this.metrica = metrica;
}

public double getCosto() {
return costo;
}

public void setCosto(double costo) {
this.costo = costo;
}
}
```

Vecino.java

```
/*
 * To change this template, choose Tools | Templates
```

```
 * and open the template in the editor.
 */

package uam.vhm.pt.bean;

import android.os.Parcel;
import android.os.Parcelable;

/**
 *
 * @author victor
 */
public class Vecino implements Parcelable{

    private String nombre;
    private int numero;
    private Double distancia;

    public Vecino()
    {

    }

    public Vecino(String nombre, int numero, Double distancia) {
        this.nombre = nombre;
        this.numero = numero;
        this.distancia = distancia;
    }

    ////////////////////////////////////////////////////////////////
    public Vecino(Parcel source)
    {
                                                nombre = source.readString();
                                                numero = source.readInt();
                                                distancia = source.readDouble();
    }

    //Para poder crear una lista de objetos tipo Vecino y pasarlos por entre actividades
    public static final Parcelable.Creator<Vecino> CREATOR = new Parcelable.Creator<Vecino>() {
```

```
    public Vecino createFromParcel(Parcel in) {
        return new Vecino(in);
    }

    public Vecino[] newArray(int size) {
        return new Vecino[size];
    }
};

public int describeContents() {
 return 0;
}

public void writeToParcel(Parcel dest, int flags) {
                                                dest.writeString(nombre);
                                                dest.writeInt(numero);
                                                dest.writeDouble(distancia);
}
//////////////////////////////////////////////////////////////////

public Double getDistancia() {
    return distancia;
}

public void setDistancia(Double distancia) {
    this.distancia = distancia;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public int getNumero() {
    return numero;
}
```

```
    public void setNumero(int numero) {
        this.numero = numero;
    }

    @Override
    public String toString() {
        return "nombre = "+nombre+", numero = "+numero+", distancia = "+distancia;
    }

}
```

D. Directorio res/layout (clases de vista)

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">

<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/msg"
/>

<TextView
android:id="@+id/text1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>

<TextView
android:id="@+id/text2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
```

```
<Button
android:id="@+id/btn1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/petNormal"
/>

<Button
android:id="@+id/btn2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/petDispo"
/>

</LinearLayout>
```

evita_estaciones.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<TextView
android:id="@+id/TextEvitarEst"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/TextViewEvitaPasos"
/>

<AutoCompleteTextView
android:id="@+id/CompleteEvitarEst1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:completionThreshold="3"
android:textColor="@color/black"
/>

<TextView
android:id="@+id/TextEvitaNumEst"
```

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>

<Button
android:id="@+id/BtnEnviarEstAEvitar"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/BtnEnviar" />

</LinearLayout>
```

forza_evita_estaciones.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<TextView
android:id="@+id/TextForzarEvitarEst1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/TextViewEvitaPasos"
/>

<AutoCompleteTextView
android:id="@+id/CompleteForzarEvitarEst1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:completionThreshold="3"
android:textColor="@color/black"
/>

<TextView
android:id="@+id/TextEvitaForzarNum1Est"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
```

```
<TextView
android:id="@+id/TextForzarEvitarEst2"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/TextViewForzarPasos"
/>

<AutoCompleteTextView
android:id="@+id/CompleteForzarEvitarEst4"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:completionThreshold="3"
android:textColor="@color/black"
/>

<TextView
android:id="@+id/TextEvitaForzarNum2Est"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>

<Button android:id="@+id/BtnEnviarEstForzarEvitar"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/BtnEnviar"
/>


</LinearLayout>
```

forzar_estaciones.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

<TextView
android:id="@+id/TextForzarEst"
```

```
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/TextViewForzarPasos"
/>

<AutoCompleteTextView
android:id="@+id/CompleteForzarEst1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:completionThreshold="3"
android:textColor="@android:color/black"
/>

<Button android:id="@+id/BtnEnviarEstAForzar"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/BtnEnviar"
/>

<TextView
android:id="@+id/TextForzarNumEst"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>

</LinearLayout>
```

listado_estaciones.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#FFF">

<!--<TextView
android:id="@+id/nombre"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textColor="#FFF"
android:text="aaaa"
```

/>-->

<ImageView

android:id="@+id/imagen"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:layout_alignParentLeft="true"

android:contentDescription="@string/pathImage"

android:layout_centerInParent="true"/>

</RelativeLayout>

rutadxml.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout_width="match_parent"

android:layout_height="match_parent">

<TextView

android:id="@+id/TextEstDetinod"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

android:text="@string/EstDestino"

/>

<AutoCompleteTextView

android:id="@+id/CompleteEstDestinod"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

android:completionThreshold="3"

android:textColor="#000"

/>

<Spinner

android:id="@+id/Metricasd"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

android:prompt="@string/msgMetricas"

```
/>

<CheckBox
android:id="@+id/ForzaEstacionesd"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/opcionForza"
/>

<CheckBox
android:id="@+id/EvitaEstacionesd"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/opcionEvita"
/>

<Button android:id="@+id/BtnEnviard"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/BtnEnviar"
/>

</LinearLayout>
```

rutanxml.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">

<TextView
android:id="@+id/TextEstOrigenn"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/EstOrigen"
/>

<AutoCompleteTextView
```

```
android:id="@+id/CompleteEstOrigenn"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:completionThreshold="3"
android:textColor="#000"/>

<TextView
android:id="@+id/TextEstDestinon"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/EstDestino"
/>

<AutoCompleteTextView
android:id="@+id/CompleteEstDestinon"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:completionThreshold="3"
android:textColor="#000"/>

<Spinner
android:id="@+id/Metricasn"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:prompt="@string/msgMetricas"
/>

<CheckBox
android:id="@+id/ForzaEstacionesn"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/opcionForza"
/>

<CheckBox
android:id="@+id/EvitaEstacionesn"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="@string/opcionEvita"
```

/>

<Button android:id="@+id/BtnEnviarn"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

android:text="@string/BtnEnviar"

/>

</LinearLayout>

solucion_mapa.xml

<?xml version="1.0" encoding="utf-8"?>

<fragment xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/map"

android:layout_width="match_parent"

android:layout_height="match_parent"

class="com.google.android.gms.maps.SupportMapFragment" />

solucion_normal.xml

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="fill_parent"

android:layout_height="fill_parent"

android:orientation="vertical"

android:background="#FFF"

>

<TextView

android:id="@+id/TxtMensaje3"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

/>

<TextView

android:id="@+id/nose3"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

android:text="@string/solMapLbl"

```
/>

<Button android:id="@+id/SolMap3"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

android:text="@string/solMap"

/>

<ListView

android:id="@android:id/list"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

/>

</LinearLayout>
```

E. Otros

AdroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="uam.vhm.pt"
android:versionCode="1"
android:versionName="1.0" >

<uses-sdk
android:minSdkVersion="8"
android:targetSdkVersion="17" />

<!-- Permisos para usar mapas y wifi -->
<uses-permission android:name="android.permission.INTERNET" />

<!-- Permisos para loc por wifi -->
<!-- <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/> -->


<!-- Permisos para loc por gps y wifi -->
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

<!-- Permisos para comprovar si hay internet -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />


<!-- google maps V2 -->

<permission
android:name="uam.vhm.pt.permission.MAPS_RECEIVE"
android:protectionLevel="signature"/>

<uses-permission android:name="uam.vhm.pt.permission.MAPS_RECEIVE"/>

<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

<uses-feature
android:glEsVersion="0x00020000"
android:required="true"/>
```

```
<!--  -->

<application
android:allowBackup="true"
android:icon="@drawable/ic_launcher"
android:label="@string/app_name"
android:theme="@style/Theme.Sherlock" >

<!-- Para google maps V2 -->
<meta-data
android:name="com.google.android.maps.v2.API_KEY"
android:value="API_KEY"/>
<!--  -->


<activity
android:name="uam.vhm.pt.LaMejorRutaAtravesDelMetroActivity"
android:label="@string/app_name" >
<intent-filter>
<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name="uam.vhm.pt.SolicitudRutaD" >
</activity> <!-- Forma 1 de poner una activity -->
<activity android:name="uam.vhm.pt.SolicitudRutaN" /> <!-- Forma 2 de poner una activity -->
<activity android:name="uam.vhm.pt.FrorzarEstaciones" >
</activity> <!-- Forma 1 de poner una activity -->
<activity android:name="uam.vhm.pt.EvitaEstaciones" >
</activity> <!-- Forma 1 de poner una activity -->
<activity android:name="uam.vhm.pt.ForzaEvitaEstaciones" >
</activity> <!-- Forma 1 de poner una activity -->
<activity android:name="uam.vhm.pt.SolucionNormal" >
</activity> <!-- Forma 1 de poner una activity -->
<activity android:name="uam.vhm.pt.EjMapActivity" >
</activity> <!-- Forma 1 de poner una activity -->
<activity android:name="uam.vhm.pt.Prueba" >
</activity>
<activity
android:name="uam.vhm.pt.SolucionMapa"
android:label="@string/title_activity_solucion_mapa" >
</activity>
</application>

</manifest>
```

XML que representa el grafo del metro

```
<linea nombre="RedDelMetro">


 <!--Linea 1 del metro-->
 <estacion nombre="Observatorio" latitud="19.402325646816475" longitud="-99.2010498046875" numero="1">
   <vecino nombre="Tacubaya(linea 1)" numero="2" distancia="1412"/>
 </estacion>
 <estacion nombre="Tacubaya(linea 1)" latitud="19.401900625723 85" longitud="-99.18884038925171" numero="2">
   <vecino nombre="Observatorio" numero="1" distancia="1412"/>
   <vecino nombre="Juanacatlán" numero="3" distancia="1308"/>
   <vecino nombre="Tacubaya(linea 7)" numero="109" distancia="100"/>
   <vecino nombre="Tacubaya(linea 9)" numero="133" distancia="100"/>
 </estacion>
 <estacion nombre="Juanacatlán" latitud="19.413436518489483" longitud="-99.18229579925537" numero="3">
   <vecino nombre="Tacubaya(linea 1)" numero="2" distancia="1308"/>
   <vecino nombre="Chapultepec" numero="4" distancia="1123"/>
 </estacion>
 <estacion nombre="Chapultepec" latitud="19.420823107700716" longitud="-99.17620182037354" numero="4">
   <vecino nombre="Juanacatlán" numero="3" distancia="1123"/>
   <vecino nombre="Sevilla" numero="5" distancia="651"/>
 </estacion>
 <estacion nombre="Sevilla" latitud="19.422280148671245" longitud="-99.1705584526062" numero="5">
   <vecino nombre="Chapultepec" numero="4" distancia="651"/>
   <vecino nombre="Insurgentes" numero="6" distancia="795"/>
 </estacion>
 <estacion nombre="Insurgentes" latitud="19.423635994506" longitud="-99.16279077529907" numero="6">
   <vecino nombre="Sevilla" numero="5" distancia="795"/>
   <vecino nombre="Cuauhtémoc" numero="7" distancia="943"/>
 </estacion>
 <estacion nombre="Cuauhtémoc" latitud="19.425902457801868" longitud="-99.15472269058228" numero="7">
```

```
  <vecino nombre="Insurgentes" numero="6" distancia="943"/>
  <vecino nombre="Balderas(linea 1)" numero="8" distancia="559"/>
</estacion>
<estacion nombre="Balderas(linea 1)" latitud="19.427784407761706" longitud="-99.1490364074707" numero="8">
  <vecino nombre="Cuauhtémoc" numero="7" distancia="559"/>
  <vecino nombre="Salto del Agua(linea 1)" numero="9" distancia="608"/>
  <vecino nombre="Balderas(linea 3)" numero="53" distancia="100"/>
</estacion>
<estacion nombre="Salto del Agua(linea 1)" latitud="19.426752373386552" longitud="-99.14214849472046" numero="9">
  <vecino nombre="Balderas(linea 1)" numero="8" distancia="608"/>
  <vecino nombre="Isabel la Católica" numero="10" distancia="595"/>
  <vecino nombre="Salto del Agua(linea 8)" numero="117" distancia="100"/>
</estacion>
<estacion nombre="Isabel la Católica" latitud="19.426489304752145" longitud="-99.13747072219849" numero="10">
  <vecino nombre="Salto del Agua(linea 1)" numero="9" distancia="595"/>
  <vecino nombre="Pino Suárez(linea 1)" numero="11" distancia="532"/>
</estacion>
<estacion nombre="Pino Suárez(linea 1)" latitud="19.426367888315664" longitud="-99.13279294967651" numero="11">
  <vecino nombre="Isabel la Católica" numero="10" distancia="532"/>
  <vecino nombre="Merced" numero="12" distancia="895"/>
  <vecino nombre="Pino Suárez(linea 2)" numero="34" distancia="100"/>
</estacion>
<estacion nombre="Merced" latitud="19.425538206904474" longitud="-99.12463903427124" numero="12">
  <vecino nombre="Pino Suárez(linea 1)" numero="11" distancia="895"/>
  <vecino nombre="Candelaria(linea 1)" numero="13" distancia="848"/>
</estacion>
<estacion nombre="Candelaria(linea 1)" latitud="19.428836671354425" longitud="-99.11912441253662" numero="13">
  <vecino nombre="Merced" numero="12" distancia="848"/>
  <vecino nombre="San Lázaro(linea 1)" numero="14" distancia="1016"/>
  <vecino nombre="Candelaria(linea 4)" numero="72" distancia="100"/>
</estacion>
<estacion nombre="San Lázaro(linea 1)" latitud="19.430293640460384" longitud="-99.11491870880127" numero="14">
  <vecino nombre="Candelaria(linea 1)" numero="13" distancia="1016"/>
  <vecino nombre="Moctezuma" numero="15" distancia="628"/>
  <vecino nombre="San Lázaro(linea B)" numero="169" distancia="100"/>
</estacion>
<estacion nombre="Moctezuma" latitud="19.426995205594082" longitud="-99.11062717437744" numero="15">
  <vecino nombre="San Lázaro(linea 1)" numero="14" distancia="628"/>
  <vecino nombre="Balbuena" numero="16" distancia="853"/>
</estacion>
<estacion nombre="Balbuena" latitud="19.423170556166323" longitud="-99.10219430923462" numero="16">
  <vecino nombre="Moctezuma" numero="15" distancia="853"/>
  <vecino nombre="Boulevard Puerto Aéreo" numero="17" distancia="745"/>
</estacion>
<estacion nombre="Boulevard Puerto Aéreo" latitud="19.41960889691277" longitud="-99.09597158432007" numero="17">
  <vecino nombre="Balbuena" numero="16" distancia="745"/>
  <vecino nombre="Gómez Farías" numero="18" distancia="761"/>
</estacion>
<estacion nombre="Gómez Farías" latitud="19.416431669094433" longitud="-99.09047842025757" numero="18">
  <vecino nombre="Boulevard Puerto Aéreo" numero="17" distancia="761"/>
  <vecino nombre="Zaragoza" numero="19" distancia="912"/>
</estacion>
<estacion nombre="Zaragoza" latitud="19.412323441700956" longitud="-99.08247470855713" numero="19">
  <vecino nombre="Gómez Farías" numero="18" distancia="912"/>
  <vecino nombre="Pantitlán(linea 1)" numero="20" distancia="1470"/>
</estacion>
<estacion nombre="Pantitlán(linea 1)" latitud="19.41535908772028" longitud="-99.07219648361206" numero="20">
  <vecino nombre="Zaragoza" numero="19" distancia="1470"/>
  <vecino nombre="Pantitlán(linea 5)" numero="76" distancia="100"/>
  <vecino nombre="Pantitlán(linea 9)" numero="144" distancia="100"/>
  <vecino nombre="Pantitlán(linea A)" numero="145" distancia="100"/>
</estacion>
<!--Fin Linea 1 del metro-->


<!--Linea 2 del metro-->
<estacion nombre="Cuatro Caminos" latitud="19.459774219933985" longitud="-99.21557664871216" numero="21">
  <vecino nombre="Panteones" numero="22" distancia="1789"/>
</estacion>
<estacion nombre="Panteones" latitud="19.459045869329 27" longitud="-99.20313119888306" numero="22">
  <vecino nombre="Cuatro Caminos" numero="21" distancia="1789"/>
  <vecino nombre="Tacuba(linea 2)" numero="23" distancia="1566"/>
</estacion>
<estacion nombre="Tacuba(linea 2)" latitud="19.458661460746864" longitud="-99.18712377548218" numero="23">
  <vecino nombre="Panteones" numero="22" distancia="1566"/>
  <vecino nombre="Cuitlahuac" numero="24" distancia="787"/>
  <vecino nombre="Tacuba(linea 7)" numero="104" distancia="100"/>
</estacion>
<estacion nombre="Cuitlahuac" latitud="19.457487997332713" longitud="-99.18208122253418" numero="24">
  <vecino nombre="Tacuba(linea 2)" numero="23" distancia="787"/>
```

```
  <vecino nombre="Popotla" numero="25" distancia="770"/>
</estacion>
<estacion nombre="Popotla" latitud="19.452126375443815" longitud="-99.17478561401367" numero="25">
  <vecino nombre="Cuitlahuac" numero="24" distancia="770"/>
  <vecino nombre="Colegio Militar" numero="26" distancia="612"/>
</estacion>
<estacion nombre="Colegio Militar" latitud="19.44890931724899" longitud="-99.17193174362183" numero="26">
  <vecino nombre="Popotla" numero="25" distancia="612"/>
  <vecino nombre="Normal" numero="27" distancia="666"/>
</estacion>
<estacion nombre="Normal" latitud="19.444680508508704" longitud="-99.16729688644409" numero="27">
  <vecino nombre="Colegio Militar" numero="26" distancia="666"/>
  <vecino nombre="San Cosme" numero="28" distancia="807"/>
</estacion>
<estacion nombre="San Cosme" latitud="19.44184775203063" longitud="-99.16120290756226" numero="28">
  <vecino nombre="Normal" numero="27" distancia="807"/>
  <vecino nombre="Revolución" numero="29" distancia="687"/>
</estacion>
<estacion nombre="Revolución" latitud="19.440046902565864" longitud="-99.15545225143433" numero="29">
  <vecino nombre="San Cosme" numero="28" distancia="687"/>
  <vecino nombre="Hidalgo(linea 2)" numero="30" distancia="737"/>
</estacion>
<estacion nombre="Hidalgo(linea 2)" latitud="19.437335473554956" longitud="-99.14710521697998" numero="30">
  <vecino nombre="Revolución" numero="29" distancia="737"/>
  <vecino nombre="Bellas Artes(linea 2)" numero="31" distancia="597"/>
  <vecino nombre="Hidalgo(linea 3)" numero="51" distancia="100"/>
</estacion>
<estacion nombre="Bellas Artes(linea 2)" latitud="19.436182090770412" longitud="-99.14193391799927" numero="31">
  <vecino nombre="Hidalgo(linea 2)" numero="30" distancia="597"/>
  <vecino nombre="Allende" numero="32" distancia="537"/>
  <vecino nombre="Bellas Artes(linea 8)" numero="115" distancia="100"/>
</estacion>
<estacion nombre="Allende" latitud="19.435554808974327" longitud="-99.13742780685425" numero="32">
  <vecino nombre="Bellas Artes(linea 2)" numero="31" distancia="537"/>
  <vecino nombre="Zócalo" numero="33" distancia="752"/>
</estacion>
<estacion nombre="Zócalo" latitud="19.43322777469157" longitud="-99.13298606872559" numero="33">
  <vecino nombre="Allende" numero="32" distancia="752"/>
  <vecino nombre="Pino Suárez(linea 2)" numero="34" distancia="895"/>
</estacion>
<estacion nombre="Pino Suárez(linea 2)" latitud="19.43620232562671" longitud="-99.14206266403198" numero="34">
  <vecino nombre="Zócalo" numero="33" distancia="895"/>
  <vecino nombre="San Antonio Abad" numero="35" distancia="967"/>
  <vecino nombre="Pino Suárez(linea 1)" numero="11" distancia="100"/>
</estacion>
<estacion nombre="San Antonio Abad" latitud="19.415986447496365" longitud="-99.13483142852783" numero="35">
  <vecino nombre="Pino Suárez(linea 2)" numero="34" distancia="967"/>
  <vecino nombre="Chabacano(linea 2)" numero="36" distancia="792"/>
</estacion>
<estacion nombre="Chabacano(linea 2)" latitud="19.408296063841117" longitud="-99.13609743118286" numero="36">
  <vecino nombre="San Antonio Abad" numero="35" distancia="792"/>
  <vecino nombre="Viaducto" numero="37" distancia="924"/>
  <vecino nombre="Chabacano(linea 8)" numero="120" distancia="100"/>
  <vecino nombre="Chabacano(linea 9)" numero="138" distancia="100"/>
</estacion>
<estacion nombre="Viaducto" latitud="19.400848187763163" longitud="-99.13691282272339" numero="37">
  <vecino nombre="Chabacano(linea 2)" numero="36" distancia="924"/>
  <vecino nombre="Xola" numero="38" distancia="640"/>
</estacion>
<estacion nombre="Xola" latitud="19.39518109710903" longitud="-99.13779258728027" numero="38">
  <vecino nombre="Viaducto" numero="37" distancia="640"/>
  <vecino nombre="Villa de Cortés" numero="39" distancia="848"/>
</estacion>
<estacion nombre="Villa de Cortés" latitud="19.387125392924805" longitud="-99.1389513015747" numero="39">
  <vecino nombre="Xola" numero="38" distancia="848"/>
  <vecino nombre="Nativitas" numero="40" distancia="900"/>
</estacion>
<estacion nombre="Nativitas" latitud="19.379494370856314" longitud="-99.14021730422974" numero="40">
  <vecino nombre="Villa de Cortés" numero="39" distancia="900"/>
  <vecino nombre="Portales" numero="41" distancia="1074"/>
</estacion>
<estacion nombre="Portales" latitud="19.369899421993985" longitud="-99.14169788360596" numero="41">
  <vecino nombre="Nativitas" numero="40" distancia="1074"/>
  <vecino nombre="Ermita(linea 2)" numero="42" distancia="898"/>
</estacion>
<estacion nombre="Ermita(linea 2)" latitud="19.361882955623688" longitud="-99.14300680160522" numero="42">
  <vecino nombre="Portales" numero="41" distancia="898"/>
  <vecino nombre="General Anaya" numero="43" distancia="988"/>
  <vecino nombre="Ermita(linea 12)" numero="189" distancia="100"/>
</estacion>
```

```
<estacion nombre="General Anaya" latitud="19.35319800559105" longitud="-99.14495944976807" numero="43">
  <vecino nombre="Ermita(linea 2)" numero="42" distancia="988"/>
  <vecino nombre="Tasqueña" numero="44" distancia="1480"/>
</estacion>
<estacion nombre="Tasqueña" latitud="19.344148160133138" longitud="-99.14268493652344" numero="44">
  <vecino nombre="General Anaya" numero="43" distancia="1480"/>
</estacion>
<!--Fin Linea 2 del metro-->


<!--Linea 3 del metro-->
<estacion nombre="Indios Verdes" latitud="19.495378480201317" longitud="-99.1196608543396" numero="45">
  <vecino nombre="Deportivo 18 de Marzo(linea 3)" numero="46" distancia="1316"/>
</estacion>
<estacion nombre="Deportivo 18 de Marzo(linea 3)" latitud="19.485244200634334" longitud="-99.12562608718872" numero="46">
  <vecino nombre="Indios Verdes" numero="45" distancia="1316"/>
  <vecino nombre="Potrero" numero="47" distancia="1116"/>
  <vecino nombre="Deportivo 18 de Marzo(linea 6)" numero="97" distancia="100"/>
</estacion>
<estacion nombre="Potrero" latitud="19.477435720287186" longitud="-99.13212776184082" numero="47">
  <vecino nombre="Deportivo 18 de Marzo(linea 3)" numero="46" distancia="1116"/>
  <vecino nombre="La Raza(linea 3)" numero="48" distancia="1256"/>
</estacion>
<estacion nombre="La Raza(linea 3)" latitud="19.47046643485585" longitud="-99.1366446018219" numero="48">
  <vecino nombre="Potrero" numero="47" distancia="1256"/>
  <vecino nombre="Tlatelolco" numero="49" distancia="1595"/>
  <vecino nombre="La Raza(linea 5)" numero="85" distancia="100"/>
</estacion>
<estacion nombre="Tlatelolco" latitud="19.455019650024536" longitud="-99.14272785186768" numero="49">
  <vecino nombre="La Raza(linea 3)" numero="48" distancia="1595"/>
  <vecino nombre="Guerrero(linea 3)" numero="50" distancia="1192"/>
</estacion>
<estacion nombre="Guerrero(linea 3)" latitud="19.445044716468892" longitud="-99.14532423019409" numero="50">
  <vecino nombre="Tlatelolco" numero="49" distancia="1192"/>
  <vecino nombre="Hidalgo(linea 3)" numero="51" distancia="852"/>
  <vecino nombre="Guerrero(linea B)" numero="174" distancia="100"/>
</estacion>
<estacion nombre="Hidalgo(linea 3)" latitud="19.437335473554956" longitud="-99.14710521697998" numero="51">
  <vecino nombre="Guerrero(linea 3)" numero="50" distancia="852"/>
  <vecino nombre="Juárez" numero="52" distancia="401"/>
  <vecino nombre="Hidalgo(linea 2)" numero="30" distancia="100"/>
</estacion>
<estacion nombre="Juárez" latitud="19.43305577518203" longitud="-99.1480815410614" numero="52">
  <vecino nombre="Hidalgo(linea 3)" numero="51" distancia="401"/>
  <vecino nombre="Balderas(linea 3)" numero="53" distancia="809"/>
</estacion>
<estacion nombre="Balderas(linea 3)" latitud="19.427784407761706" longitud="-99.1490364074707" numero="53">
  <vecino nombre="Juárez" numero="52" distancia="809"/>
  <vecino nombre="Niños Héroes" numero="54" distancia="815"/>
  <vecino nombre="Balderas(linea 1)" numero="8" distancia="100"/>
</estacion>
<estacion nombre="Niños Héroes" latitud="19.419507712270956" longitud="-99.15051698684692" numero="54">
  <vecino nombre="Balderas(linea 3)" numero="53" distancia="815"/>
  <vecino nombre="Hospital General" numero="55" distancia="709"/>
</estacion>
<estacion nombre="Hospital General" latitud="19.413557944587218" longitud="-99.15384292602539" numero="55">
  <vecino nombre="Niños Héroes" numero="54" distancia="709"/>
  <vecino nombre="Centro Médico(linea 3)" numero="56" distancia="803"/>
</estacion>
<estacion nombre="Centro Médico(linea 3)" latitud="19.406659280640284" longitud="-99.15521085262299" numero="56">
  <vecino nombre="Hospital General" numero="55" distancia="803"/>
  <vecino nombre="Etiopía / Plaza de la Transparencia" numero="57" distancia="1269"/>
  <vecino nombre="Centro Médico(linea 9)" numero="136" distancia="100"/>
</estacion>
<estacion nombre="Etiopía / Plaza de la Transparencia" latitud="19.395262056936556" longitud="-99.15628910064697" numero="57">
  <vecino nombre="Centro Médico(linea 3)" numero="56" distancia="1269"/>
  <vecino nombre="Eugenia" numero="58" distancia="1100"/>
</estacion>
<estacion nombre="Eugenia" latitud="19.385404901930855" longitud="-99.15749073028564" numero="58">
  <vecino nombre="Etiopía / Plaza de la Transparencia" numero="57" distancia="1100"/>
  <vecino nombre="División del Norte" numero="59" distancia="865"/>
</estacion>
<estacion nombre="División del Norte" latitud="19.379858724865993" longitud="-99.15910005569458" numero="59">
  <vecino nombre="Eugenia" numero="58" distancia="865"/>
  <vecino nombre="Zapata(linea 3)" numero="60" distancia="944"/>
</estacion>
<estacion nombre="Zapata(linea 3)" latitud="19.370749630150474 " longitud="-99.16497945785522" numero="60">
  <vecino nombre="División del Norte" numero="59" distancia="944"/>
  <vecino nombre="Coyoacán" numero="61" distancia="1303"/>
  <vecino nombre="Zapata(linea 12)" numero="192" distancia="100"/>
```

```
</estacion>
<estacion nombre="Coyoacán" latitud="19.361781735093853" longitud="-99.17051553726196" numero="61">
  <vecino nombre="Zapata(linea 3)" numero="60" distancia="1303"/>
  <vecino nombre="Viveros / Derechos Humanos" numero="62" distancia="1058"/>
</estacion>
<estacion nombre="Viveros / Derechos Humanos" latitud="19.35372437936431" longitud="-99.17624473571777" numero="62">
  <vecino nombre="Coyoacán" numero="61" distancia="1058"/>
  <vecino nombre="Miguel Angel de Quevedo" numero="63" distancia="974"/>
</estacion>
<estacion nombre="Miguel Angel de Quevedo" latitud="19.34680040397268" longitud="-99.18094396591187" numero="63">
  <vecino nombre="Viveros / Derechos Humanos" numero="62" distancia="974"/>
  <vecino nombre="Copilco" numero="64" distancia="1445"/>
</estacion>
<estacion nombre="Copilco" latitud="19.33584696577336" longitud="-99.17658805847168" numero="64">
  <vecino nombre="Miguel Angel de Quevedo" numero="63" distancia="1445"/>
  <vecino nombre="Universidad" numero="65" distancia="1456"/>
</estacion>
<estacion nombre="Universidad" latitud="19.324467569736512" longitud="-99.17399168014526" numero="65">
  <vecino nombre="Copilco" numero="64" distancia="1456"/>
</estacion>
<!--Fin Linea 3 del metro-->

<!--Linea 4 del metro-->
<estacion nombre="Martín Carrera(linea 4)" latitud="19.48531500119498" longitud="-99.10436153411865" numero="66">
  <vecino nombre="Talismán" numero="67" distancia="1279"/>
  <vecino nombre="Martín Carrera(linea 6)" numero="99" distancia="100"/>
</estacion>
<estacion nombre="Talismán" latitud="19.47407755340729" longitud="-99.10841703414917" numero="67">
  <vecino nombre="Martín Carrera(linea 4)" numero="66" distancia="1279"/>
  <vecino nombre="Bondojito" numero="68" distancia="1109"/>
</estacion>
<estacion nombre="Bondojito" latitud="19.464730963541342" longitud="-99.11191463470459" numero="68">
  <vecino nombre="Talismán" numero="67" distancia="1109"/>
  <vecino nombre="Consulado(linea 4)" numero="69" distancia="795"/>
</estacion>
<estacion nombre="Consulado(linea 4)" latitud="19.458054497972054" longitud="-99.11393165588379" numero="69">
  <vecino nombre="Bondojito" numero="68" distancia="795"/>
  <vecino nombre="Canal del Norte" numero="70" distancia="1034"/>
  <vecino nombre="Consulado(linea 5)" numero="82" distancia="100"/>
</estacion>
<estacion nombre="Canal del Norte" latitud="19.448757567645533" longitud="-99.11617934703827" numero="70">
  <vecino nombre="Consulado(linea 4)" numero="69" distancia="1034"/>
  <vecino nombre="Morelos(linea 4)" numero="71" distancia="1060"/>
</estacion>
<estacion nombre="Morelos(linea 4)" latitud="19.438964359846697" longitud="-99.1182553768158" numero="71">
  <vecino nombre="Canal del Norte" numero="70" distancia="1060"/>
  <vecino nombre="Candelaria(linea 4)" numero="72" distancia="1212"/>
  <vecino nombre="Morelos(linea B)" numero="170" distancia="100"/>
</estacion>
<estacion nombre="Candelaria(linea 4)" latitud="19.428836671354425" longitud="-99.11912441253662" numero="72">
  <vecino nombre="Morelos(linea 4)" numero="71" distancia="1212"/>
  <vecino nombre="Fray Servando" numero="73" distancia="783"/>
  <vecino nombre="Candelaria(linea 1)" numero="13" distancia="100"/>
</estacion>
<estacion nombre="Fray Servando" latitud="19.421824824771324" longitud="-99.12054598331451" numero="73">
  <vecino nombre="Candelaria(linea 4)" numero="72" distancia="783"/>
  <vecino nombre="Jamaica(linea 4)" numero="74" distancia="1183"/>
</estacion>
<estacion nombre="Jamaica(linea 4)" latitud="19.409227024370857" longitud="-99.12186026573181" numero="74">
  <vecino nombre="Fray Servando" numero="73" distancia="1183"/>
  <vecino nombre="Santa Anita(linea 4)" numero="75" distancia="908"/>
  <vecino nombre="Jamaica(linea 9)" numero="139" distancia="100"/>
</estacion>
<estacion nombre="Santa Anita(linea 4)" latitud="19.40270006923998" longitud="-99.12166714668274" numero="75">
  <vecino nombre="Jamaica(linea 4)" numero="74" distancia="908"/>
  <vecino nombre="Santa Anita(linea 8)" numero="122" distancia="100"/>
</estacion>
<!--Fin Linea 4 del metro-->


<!--Linea 5 del metro-->
<estacion nombre="Pantitlán(linea 5)" latitud="19.41535908772028" longitud="-99.07219648361206" numero="76">
  <vecino nombre="Hangares" numero="77" distancia="1794"/>
  <vecino nombre="Pantitlán(linea 1)" numero="20" distancia="100"/>
  <vecino nombre="Pantitlán(linea 9)" numero="144" distancia="100"/>
  <vecino nombre="Pantitlán(linea A)" numero="145" distancia="100"/>
</estacion>
<estacion nombre="Hangares" latitud="19.4241216678738" longitud="-99.08756017684937" numero="77">
  <vecino nombre="Pantitlán(linea 5)" numero="76" distancia="1794"/>
  <vecino nombre="Terminal Aérea" numero="78" distancia="1303"/>
```

```
</estacion>
<estacion nombre="Terminal Aérea" latitud="19.433389656416505" longitud="-99.08775329589844" numero="78">
  <vecino nombre="Hangares" numero="77" distancia="1303"/>
  <vecino nombre="Oceanía(linea 5)" numero="79" distancia="1324"/>
</estacion>
<estacion nombre="Oceanía(linea 5)" latitud="19.445823714084483" longitud="-99.08724904060364" numero="79">
  <vecino nombre="Terminal Aérea" numero="78" distancia="1324"/>
  <vecino nombre="Aragón" numero="80" distancia="1369"/>
  <vecino nombre="Oceanía(linea B)" numero="166" distancia="100"/>
</estacion>
<estacion nombre="Aragón" latitud="19.451003447283256" longitud="-99.09654021263123" numero="80">
  <vecino nombre="Oceanía(linea 5)" numero="79" distancia="1369"/>
  <vecino nombre="Eduardo Molina" numero="81" distancia="1010"/>
</estacion>
<estacion nombre="Eduardo Molina" latitud="19.45101356378788" longitud="-99.10543441772461" numero="81">
  <vecino nombre="Aragón" numero="80" distancia="1010"/>
  <vecino nombre="Consulado(linea 5)" numero="82" distancia="965"/>
</estacion>
<estacion nombre="Consulado(linea 5)" latitud="19.458054497972054" longitud="-99.11393165588379" numero="82">
  <vecino nombre="Eduardo Molina" numero="81" distancia="965"/>
  <vecino nombre="Valle Gómez" numero="83" distancia="829"/>
  <vecino nombre="Consulado(linea 4)" numero="69" distancia="100"/>
</estacion>
<estacion nombre="Valle Gómez" latitud="19.45876262098847" longitud="-99.11936044692993" numero="83">
  <vecino nombre="Consulado(linea 5)" numero="82" distancia="829"/>
  <vecino nombre="Misterios" numero="84" distancia="1119"/>
</estacion>
<estacion nombre="Misterios" latitud="19.463658701238554" longitud="-99.13079738616943" numero="84">
  <vecino nombre="Valle Gómez" numero="83" distancia="1119"/>
  <vecino nombre="La Raza(linea 5)" numero="85" distancia="1042"/>
</estacion>
<estacion nombre="La Raza(linea 5)" latitud="19.47046643485585" longitud="-99.1366446018219" numero="85">
  <vecino nombre="Misterios" numero="84" distancia="1042"/>
  <vecino nombre="Autobuses del Norte" numero="86" distancia="1125"/>
  <vecino nombre="La Raza(linea 3)" numero="48" distancia="100"/>
</estacion>
<estacion nombre="Autobuses del Norte" latitud="19.47905408910742" longitud="-99.1407322883606" numero="86">
  <vecino nombre="La Raza(linea 5)" numero="85" distancia="1125"/>
  <vecino nombre="Instituto del Petróleo(linea 5)" numero="87" distancia="1217"/>
</estacion>
<estacion nombre="Instituto del Petróleo(linea 5)" latitud="19.489704575494674" longitud="-99.14492726325989" numero="87">
  <vecino nombre="Autobuses del Norte" numero="86" distancia="1217"/>
  <vecino nombre="Politécnico" numero="88" distancia="1338"/>
  <vecino nombre="Instituto del Petróleo(linea 6)" numero="95" distancia="100"/>
</estacion>
<estacion nombre="Politécnico" latitud="19.500799351290954" longitud="-99.14952993392944" numero="88">
  <vecino nombre="Instituto del Petróleo(linea 5)" numero="87" distancia="1338"/>
</estacion>
<!--Fin Linea 5 del metro-->


<!--Linea 6 del metro-->
<estacion nombre="El Rosario(linea 6)" latitud="19.50437945388146" longitud="-99.19989109039307" numero="89">
  <vecino nombre="Tezozomoc" numero="90" distancia="1407"/>
  <vecino nombre="El Rosario(linea 7)" numero="100" distancia="100"/>
</estacion>
<estacion nombre="Tezozomoc" latitud="19.495075067871014" longitud="-99.19624328613281" numero="90">
  <vecino nombre="El Rosario(linea 6)" numero="89" distancia="1407"/>
  <vecino nombre="Azcapotzalco" numero="91" distancia="1123"/>
</estacion>
<estacion nombre="Azcapotzalco" latitud="19.491009287768183" longitud="-99.18637275695801" numero="91">
  <vecino nombre="Tezozomoc" numero="90" distancia="1123"/>
  <vecino nombre="Ferrería / Arena Ciudad de México" numero="92" distancia="1323"/>
</estacion>
<estacion nombre="Ferrería / Arena Ciudad de México" latitud="19.490989059751172" longitud="-99.17405605316162" numero="92">
  <vecino nombre="Azcapotzalco" numero="91" distancia="1323"/>
  <vecino nombre="Norte 45" numero="93" distancia="1222"/>
</estacion>
<estacion nombre="Norte 45" latitud="19.48860213599771" longitud="-99.16281223297119" numero="93">
  <vecino nombre="Ferrería / Arena Ciudad de México" numero="92" distancia="1222"/>
  <vecino nombre="Vallejo" numero="94" distancia="810"/>
</estacion>
<estacion nombre="Vallejo" latitud="19.490402446158296" longitud="-99.15560245513916" numero="94">
  <vecino nombre="Norte 45" numero="93" distancia="810"/>
  <vecino nombre="Instituto del Petróleo(linea 6)" numero="95" distancia="905"/>
</estacion>
<estacion nombre="Instituto del Petróleo(linea 6)" latitud="19.490837349543007" longitud="-99.14833903312683" numero="95">
  <vecino nombre="Vallejo" numero="94" distancia="905"/>
  <vecino nombre="Lindavista" numero="96" distancia="1408"/>
  <vecino nombre="Instituto del Petróleo(linea 5)" numero="87" distancia="100"/>
```

```
</estacion>
<estacion nombre="Lindavista" latitud="19.487772772896037" longitud="-99.13474559783936" numero="96">
  <vecino nombre="Instituto del Petróleo(linea 6)" numero="95" distancia="1408"/>
  <vecino nombre="Deportivo 18 de Marzo(linea 6)" numero="97" distancia="1225"/>
</estacion>
<estacion nombre="Deportivo 18 de Marzo(linea 6)" latitud="19.483848411832263" longitud="-99.12620544433594" numero="97">
  <vecino nombre="Lindavista" numero="96" distancia="1225"/>
  <vecino nombre="La Villa Basílica" numero="98" distancia="720"/>
  <vecino nombre="Deportivo 18 de Marzo(linea 3)" numero="46" distancia="100"/>
</estacion>
<estacion nombre="La Villa Basílica" latitud="19.481481612021312" longitud="-99.1179871559143" numero="98">
  <vecino nombre="Deportivo 18 de Marzo(linea 6)" numero="97" distancia="720"/>
  <vecino nombre="Martín Carrera(linea 6)" numero="99" distancia="1291"/>
</estacion>
<estacion nombre="Martín Carrera(linea 6)" latitud="19.482584099997272" longitud="-99.10704374313354" numero="99">
  <vecino nombre="La Villa Basílica" numero="98" distancia="1291"/>
  <vecino nombre="Martín Carrera(linea 4)" numero="66" distancia="100"/>
</estacion>
<!--Fin Linea 6 del metro-->


<!--Linea 7 del metro-->
<estacion nombre="El Rosario(linea 7)" latitud="19.50437945388146" longitud="-99.19989109039307" numero="100">
  <vecino nombre="Aquiles Sedrán" numero="101" distancia="1765"/>
  <vecino nombre="El Rosario(linea 6)" numero="89" distancia="100"/>
</estacion>
<estacion nombre="Aquiles Sedrán" latitud="19.490483358504385" longitud="-99.19491291046143" numero="101">
  <vecino nombre="El Rosario(linea 7)" numero="100" distancia="1765"/>
  <vecino nombre="Camarones" numero="102" distancia="1552"/>
</estacion>
<estacion nombre="Camarones" latitud="19.47921592510019" longitud="-99.19012784957886" numero="102">
  <vecino nombre="Aquiles Sedrán" numero="101" distancia="1552"/>
  <vecino nombre="Refineria" numero="103" distancia="1102"/>
</estacion>
<estacion nombre="Refineria" latitud="19.470071938037734" longitud="-99.1906213760376" numero="103">
  <vecino nombre="Camarones" numero="102" distancia="1102"/>
  <vecino nombre="Tacuba(linea 7)" numero="104" distancia="1445"/>
</estacion>
<estacion nombre="Tacuba(linea 7)" latitud="19.45906610133468" longitud="-99.18903350830078" numero="104">
  <vecino nombre="Refineria" numero="103" distancia="1445"/>
  <vecino nombre="San Joaquín" numero="105" distancia="1583"/>
  <vecino nombre="Tacuba(linea 2)" numero="23" distancia="100"/>
</estacion>
<estacion nombre="San Joaquín" latitud="19.445671961595487" longitud="-99.19195175170898" numero="105">
  <vecino nombre="Tacuba(linea 7)" numero="104" distancia="1583"/>
  <vecino nombre="Polanco" numero="106" distancia="1313"/>
</estacion>
<estacion nombre="Polanco" latitud="19.433996711447804" longitud="-99.19139385223389" numero="106">
  <vecino nombre="San Joaquín" numero="105" distancia="1313"/>
  <vecino nombre="Auditorio" numero="107" distancia="962"/>
</estacion>
<estacion nombre="Auditorio" latitud="19.425275136304027" longitud="-99.19171571731567" numero="107">
  <vecino nombre="Polanco" numero="106" distancia="962"/>
  <vecino nombre="Constituyentes" numero="108" distancia="1580"/>
</estacion>
<estacion nombre="Constituyentes" latitud="19.411837733077892" longitud="-99.19128656387329" numero="108">
  <vecino nombre="Auditorio" numero="107" distancia="1580"/>
  <vecino nombre="Tacubaya(linea 7)" numero="109" distancia="1155"/>
</estacion>
<estacion nombre="Tacubaya(linea 7)" latitud="19.40273042777708" longitud="-99.18699502944946" numero="109">
  <vecino nombre="Constituyentes" numero="108" distancia="1155"/>
  <vecino nombre="San Pedro de los Pinos" numero="110" distancia="1234"/>
  <vecino nombre="Tacubaya(linea 1)" numero="2" distancia="100"/>
  <vecino nombre="Tacubaya(linea 9)" numero="133" distancia="100"/>
</estacion>
<estacion nombre="San Pedro de los Pinos" latitud="19.391274737581114" longitud="-99.18607234954834" numero="110">
  <vecino nombre="Tacubaya(linea 7)" numero="109" distancia="1234"/>
  <vecino nombre="San Antonio" numero="111" distancia="756"/>
</estacion>
<estacion nombre="San Antonio" latitud="19.384473805105923" longitud="-99.18667316436768" numero="111">
  <vecino nombre="San Pedro de los Pinos" numero="110" distancia="756"/>
  <vecino nombre="Mixcoac(linea 7)" numero="112" distancia="938"/>
</estacion>
<estacion nombre="Mixcoac(linea 7)" latitud="19.376336601967292" longitud="-99.18802499771118" numero="112">
  <vecino nombre="San Antonio" numero="111" distancia="938"/>
  <vecino nombre="Barranca del Muerto" numero="113" distancia="1626"/>
  <vecino nombre="Mixcoac(linea 12)" numero="195" distancia="100"/>
</estacion>
<estacion nombre="Barranca del Muerto" latitud="19.361336364016044" longitud="-99.189612865448" numero="113">
  <vecino nombre="Mixcoac(linea 7)" numero="112" distancia="1626"/>
```

```
</estacion>
<!--Fin Linea 7 del metro-->


<!--Linea 8 del metro-->
<estacion nombre="Garibaldi(linea 8)" latitud="19.442657116069l9" longitud="-99.13925170898438" numero="114">
  <vecino nombre="Bellas Artes(linea 8)" numero="115" distancia="784"/>
  <vecino nombre="Garibaldi(linea B)" numero="173" distancia="100"/>
</estacion>
<estacion nombre="Bellas Artes(linea 8)" latitud="19.436404674051026" longitud="-99.14075374603271" numero="115">
  <vecino nombre="Garibaldi(linea 8)" numero="114" distancia="784"/>
  <vecino nombre="San Juan de Letrán" numero="116" distancia="606"/>
  <vecino nombre="Bellas Artes(linea 2)" numero="31" distancia="100"/>
</estacion>
<estacion nombre="San Juan de Letrán" latitud="19.431204239515083" longitud="-99.14150476455688" numero="116">
  <vecino nombre="Bellas Artes(linea 8)" numero="115" distancia="606"/>
  <vecino nombre="Salto del Agua(linea 8)" numero="117" distancia="442"/>
</estacion>
<estacion nombre="Salto del Agua(linea 8)" latitud="19.428330776247154" longitud="-99.14161205291748" numero="117">
  <vecino nombre="San Juan de Letrán" numero="116" distancia="442"/>
  <vecino nombre="Doctores" numero="118" distancia="714"/>
  <vecino nombre="Salto del Agua(linea 1)" numero="9" distancia="100"/>
</estacion>
<estacion nombre="Doctores" latitud="19.42134926289151" longitud="-99.14362907409668" numero="118">
  <vecino nombre="Salto del Agua(linea 8)" numero="117" distancia="714"/>
  <vecino nombre="Obrera" numero="119" distancia="911"/>
</estacion>
<estacion nombre="Obrera" latitud="19.413213903741404" longitud="-99.1442084312439" numero="119">
  <vecino nombre="Doctores" numero="118" distancia="911"/>
  <vecino nombre="Chabacano(linea 8)" numero="120" distancia="1293"/>
</estacion>
<estacion nombre="Chabacano(linea 8)" latitud="19.408255587175454" longitud="-99.13382291793823" numero="120">
  <vecino nombre="Obrera" numero="119" distancia="1293"/>
  <vecino nombre="La Viga" numero="121" distancia="993"/>
  <vecino nombre="Chabacano(linea 2)" numero="36" distancia="100"/>
  <vecino nombre="Chabacano(linea 9)" numero="138" distancia="100"/>
</estacion>
<estacion nombre="La Viga" latitud="19.406555558120445" longitud="-99.1262698173523" numero="121">
  <vecino nombre="Chabacano(linea 8)" numero="120" distancia="993"/>
  <vecino nombre="Santa Anita(linea 8)" numero="122" distancia="783"/>
</estacion>
<estacion nombre="Santa Anita(linea 8)" latitud="19.40270006923998" longitud="-99.12166714668274" numero="122">
  <vecino nombre="La Viga" numero="121" distancia="783"/>
  <vecino nombre="Coyuya" numero="123" distancia="1118"/>
  <vecino nombre="Santa Anita(linea 4)" numero="75" distancia="100"/>
</estacion>
<estacion nombre="Coyuya" latitud="19.39850041701016" longitud="-99.11352396011353" numero="123">
  <vecino nombre="Santa Anita(linea 8)" numero="122" distancia="1118"/>
  <vecino nombre="Iztacalco" numero="124" distancia="1143"/>
</estacion>
<estacion nombre="Iztacalco" latitud="19.38896731019128" longitud="-99.11218285560608" numero="124">
  <vecino nombre="Coyuya" numero="123" distancia="1143"/>
  <vecino nombre="Apatlaco" numero="125" distancia="1060"/>
</estacion>
<estacion nombre="Apatlaco" latitud="19.379307133062092" longitud="-99.10958111286163" numero="125">
  <vecino nombre="Iztacalco" numero="124" distancia="1060"/>
  <vecino nombre="Aculco" numero="126" distancia="684"/>
</estacion>
<estacion nombre="Aculco" latitud="19.373705081136634" longitud="-99.10736560821533" numero="126">
  <vecino nombre="Apatlaco" numero="125" distancia="684"/>
  <vecino nombre="Escuadrón 201" numero="127" distancia="939"/>
</estacion>
<estacion nombre="Escuadrón 201" latitud="19.36500051716813" longitud="-99.10916805267334" numero="127">
  <vecino nombre="Aculco" numero="126" distancia="939"/>
  <vecino nombre="Atlalilco(linea 8)" numero="128" distancia="1888"/>
</estacion>
<estacion nombre="Atlalilco(linea 8)" latitud="19.356194264384783" longitud="-99.10133600234985" numero="128">
  <vecino nombre="Escuadrón 201" numero="127" distancia="1888"/>
  <vecino nombre="Iztapalapa" numero="129" distancia="882"/>
  <vecino nombre="Atlalilco(linea 12)" numero="187" distancia="100"/>
</estacion>
<estacion nombre="Iztapalapa" latitud="19.357935308399323" longitud="-99.0935468673706" numero="129">
  <vecino nombre="Atlalilco(linea 8)" numero="128" distancia="882"/>
  <vecino nombre="Cerro de la Estrella" numero="130" distancia="867"/>
</estacion>
<estacion nombre="Cerro de la Estrella" latitud="19.3566396495065" longitud="-99.08547878265381" numero="130">
  <vecino nombre="Iztapalapa" numero="129" distancia="867"/>
  <vecino nombre="UAM-I" numero="131" distancia="1285"/>
</estacion>
<estacion nombre="UAM-I" latitud="19.350687584229092" longitud="-99.07477140426636" numero="131">
```

```
  <vecino nombre="Cerro de la Estrella" numero="130" distancia="1285"/>
  <vecino nombre="Constitución de 1917" numero="132" distancia="1287"/>
</estacion>
<estacion nombre="Constitución de 1917" latitud="19.345970317023518" longitud="-99.06389236450195" numero="132">
  <vecino nombre="UAM-I" numero="131" distancia="1287"/>
</estacion>
<!--Fin Linea 8 del metro-->


<!--Linea 9 del metro-->
<estacion nombre="Tacubaya(linea 9)" latitud="19.40273042777708" longitud="-99.18699502944946" numero="133">
  <vecino nombre="Patriotismo" numero="134" distancia="1283"/>
  <vecino nombre="Tacubaya(linea 1)" numero="2" distancia="100"/>
  <vecino nombre="Tacubaya(liena 7)" numero="109" distancia="100"/>
</estacion>
<estacion nombre="Patriotismo" latitud="19.40623174104757" longitud="-99.17894840240479" numero="134">
  <vecino nombre="Tacubaya(linea 9)" numero="133" distancia="1283"/>
  <vecino nombre="Chilpancingo" numero="135" distancia="1105"/>
</estacion>
<estacion nombre="Chilpancingo" latitud="19.406130548080082" longitud="-99.1684341430664" numero="135">
  <vecino nombre="Patriotismo" numero="134" distancia="1105"/>
  <vecino nombre="Centro Médico(linea 9)" numero="136" distancia="1302"/>
</estacion>
<estacion nombre="Centro Médico(linea 9)" latitud="19.406659280640284" longitud="-99.15521085262299" numero="136">
  <vecino nombre="Chilpancingo" numero="135" distancia="1302"/>
  <vecino nombre="Lázaro Cárdenas" numero="137" distancia="1209"/>
  <vecino nombre="Centro Médico(linea 3)" numero="56" distancia="100"/>
</estacion>
<estacion nombre="Lázaro Cárdenas" latitud="19.406960328554916" longitud="-99.14495944976807" numero="137">
  <vecino nombre="Centro Médico(linea 9)" numero="136" distancia="1209"/>
  <vecino nombre="Chabacano(linea 9)" numero="138" distancia="1150"/>
</estacion>
<estacion nombre="Chabacano(linea 9)" latitud="19.408296063841117" longitud="-99.13609743118286" numero="138">
  <vecino nombre="Lázaro Cárdenas" numero="137" distancia="1150"/>
  <vecino nombre="Jamaica(linea 9)" numero="139" distancia="1181"/>
  <vecino nombre="Chabacano(linea 2)" numero="36" distancia="100"/>
  <vecino nombre="Chabacano(linea 8)" numero="120" distancia="100"/>
</estacion>
<estacion nombre="Jamaica(linea 9)" latitud="19.40886273610262" longitud="-99.12214994430542" numero="139">
  <vecino nombre="Chabacano(linea 9)" numero="138" distancia="1181"/>
  <vecino nombre="Mixiuhca" numero="140" distancia="1092"/>
  <vecino nombre="Jamaica(linea 4)" numero="74" distancia="100"/>
</estacion>
<estacion nombre="Mixiuhca" latitud="19.40859963851249" longitud="-99.11300897598267" numero="140">
  <vecino nombre="Jamaica(linea 9)" numero="139" distancia="1092"/>
  <vecino nombre="Velódromo" numero="141" distancia="971"/>
</estacion>
<estacion nombre="Velódromo" latitud="19.40853892362356" longitud="-99.10333156585693" numero="141">
  <vecino nombre="Mixiuhca" numero="140" distancia="971"/>
  <vecino nombre="Ciudad Deportiva" numero="142" distancia="1260"/>
</estacion>
<estacion nombre="Ciudad Deportiva" latitud="19.408417493777666" longitud="-99.09120798110962" numero="142">
  <vecino nombre="Velódromo" numero="141" distancia="1260"/>
  <vecino nombre="Puebla" numero="143" distancia="950"/>
</estacion>
<estacion nombre="Puebla" latitud="19.40714247492176" longitud="-99.08247470855713" numero="143">
  <vecino nombre="Ciudad Deportiva" numero="142" distancia="950"/>
  <vecino nombre="Pantitlán(linea 9)" numero="144" distancia="1530"/>
</estacion>
<estacion nombre="Pantitlán(linea 9)" latitud="19.41535908772028" longitud="-99.07219648361206" numero="144">
  <vecino nombre="Puebla" numero="143" distancia="1530"/>
  <vecino nombre="Pantitlán(linea 1)" numero="20" distancia="100"/>
  <vecino nombre="Pantitlán(linea 5)" numero="76" distancia="100"/>
  <vecino nombre="Pantitlán(linea A)" numero="145" distancia="100"/>
</estacion>
<!--Fin Linea 9 del metro-->


<!--Linea A del metro-->
<estacion nombre="Pantitlán(linea A)" latitud="19.41495433819274" longitud="-99.07230377197266" numero="145">
  <vecino nombre="Agrícola Oriental" numero="146" distancia="1559"/>
  <vecino nombre="Pantitlán(linea 1)" numero="20" distancia="100"/>
  <vecino nombre="Pantitlán(linea 5)" numero="76" distancia="100"/>
  <vecino nombre="Pantitlán(linea 9)" numero="144" distancia="100"/>
</estacion>
<estacion nombre="Agrícola Oriental" latitud="19.405138853666905" longitud="-99.06985759735107" numero="146">
  <vecino nombre="Pantitlán(linea A)" numero="145" distancia="1559"/>
  <vecino nombre="Canal de San Juan" numero="147" distancia="1243"/>
</estacion>
<estacion nombre="Canal de San Juan" latitud="19.39866233332212" longitud="-99.05947208404541" numero="147">
```

```
  <vecino nombre="Agrícola Oriental" numero="146" distancia="1243"/>
  <vecino nombre="Tepalcates" numero="148" distancia="1606"/>
</estacion>
<estacion nombre="Tepalcates" latitud="19.391112813919065" longitud="-99.04629707336426" numero="148">
  <vecino nombre="Canal de San Juan" numero="147" distancia="1606"/>
  <vecino nombre="Guelatao" numero="149" distancia="1311"/>
</estacion>
<estacion nombre="Guelatao" latitud="19.38508104276951" longitud="-99.03567552566528" numero="149">
  <vecino nombre="Tepalcates" numero="148" distancia="1311"/>
  <vecino nombre="Peñón Viejo" numero="150" distancia="2356"/>
</estacion>
<estacion nombre="Peñón Viejo" latitud="19.37295610205349" longitud="-99.01747941970825" numero="150">
  <vecino nombre="Guelatao" numero="149" distancia="2356"/>
  <vecino nombre="Acatitla" numero="151" distancia="1529"/>
</estacion>
<estacion nombre="Acatitla" latitud="19.364413448446467" longitud="-99.00604248046875" numero="151">
  <vecino nombre="Peñón Viejo" numero="150" distancia="1529"/>
  <vecino nombre="Santa Marta" numero="152" distancia="1250"/>
</estacion>
<estacion nombre="Santa Marta" latitud="19.360222930999274" longitud="-98.99512052536011" numero="152">
  <vecino nombre="Acatitla" numero="151" distancia="1250"/>
  <vecino nombre="Los Reyes" numero="153" distancia="1933"/>
</estacion>
<estacion nombre="Los Reyes" latitud="19.358967779209884" longitud="-98.97686004638672" numero="153">
  <vecino nombre="Santa Marta" numero="152" distancia="1933"/>
  <vecino nombre="La Paz" numero="154" distancia="2106"/>
</estacion>
<estacion nombre="La Paz" latitud="19.350667338738496" longitud="-98.96115303039551" numero="154">
  <vecino nombre="Los Reyes" numero="153" distancia="2106"/>
</estacion>
<!--Fin Linea A del metro-->


<!--Linea B del metro-->
<estacion nombre="Ciudad Azteca" latitud="19.534574566242306" longitud="-99.0273928642273" numero="155">
  <vecino nombre="Plaza Aragón" numero="156" distancia="724"/>
</estacion>
<estacion nombre="Plaza Aragón" latitud="19.528588575750607" longitud="-99.03026819229126" numero="156">
  <vecino nombre="Ciudad Azteca" numero="155" distancia="724"/>
  <vecino nombre="Olímpica" numero="157" distancia="859"/>
</estacion>
<estacion nombre="Olímpica" latitud="19.521414195258373" longitud="-99.03347879648209" numero="157">
  <vecino nombre="Plaza Aragón" numero="156" distancia="859"/>
  <vecino nombre="Ecatepec / Tecnológico" numero="158" distancia="746"/>
</estacion>
<estacion nombre="Ecatepec / Tecnológico" latitud="19.515260861145272" longitud="-99.03599739074707" numero="158">
  <vecino nombre="Olímpica" numero="157" distancia="746"/>
  <vecino nombre="Muzquiz" numero="159" distancia="1635"/>
</estacion>
<estacion nombre="Muzquiz" latitud="19.50162864764207" longitud="-99.04204845428467" numero="159">
  <vecino nombre="Ecatepec / Tecnológico" numero="158" distancia="1635"/>
  <vecino nombre="Río de los Remedios" numero="160" distancia="1305"/>
</estacion>
<estacion nombre="Río de los Remedios" latitud="19.490948603709572" longitud="-99.04647946357727" numero="160">
  <vecino nombre="Muzquiz" numero="159" distancia="1305"/>
  <vecino nombre="Impulsora" numero="161" distancia="586"/>
</estacion>
<estacion nombre="Impulsora" latitud="19.48545660222343" longitud="-99.04890418052673" numero="161">
  <vecino nombre="Río de los Remedios" numero="160" distancia="586"/>
  <vecino nombre="Nezahualcóyotl" numero="162" distancia="1543"/>
</estacion>
<estacion nombre="Nezahualcóyotl" latitud="19.472964892403265" longitud="-99.05449390411377" numero="162">
  <vecino nombre="Impulsora" numero="161" distancia="1543"/>
  <vecino nombre="Villa de Aragón" numero="163" distancia="1485"/>
</estacion>
<estacion nombre="Villa de Aragón" latitud="19.46129160651945" longitud="-99.06129598617554" numero="163">
  <vecino nombre="Nezahualcóyotl" numero="162" distancia="1485"/>
  <vecino nombre="Bosque de Aragón" numero="164" distancia="934"/>
</estacion>
<estacion nombre="Bosque de Aragón" latitud="19.458094962227715" longitud="-99.06925678253174" numero="164">
  <vecino nombre="Villa de Aragón" numero="163" distancia="934"/>
  <vecino nombre="Deportivo Oceanía" numero="165" distancia="1315"/>
</estacion>
<estacion nombre="Deportivo Oceanía" latitud="19.450973097765647" longitud="-99.07942771911621" numero="165">
  <vecino nombre="Bosque de Aragón" numero="164" distancia="1315"/>
  <vecino nombre="Oceanía(linea B)" numero="166" distancia="1013"/>
</estacion>
<estacion nombre="Oceanía(linea B)" latitud="19.445823714084483" longitud="-99.08724904060364" numero="166">
  <vecino nombre="Deportivo Oceanía" numero="165" distancia="1013"/>
  <vecino nombre="Romero Rubio" numero="167" distancia="959"/>
```

```
  <vecino nombre="Oceanía(linea 5)" numero="79" distancia="100"/>
</estacion>
<estacion nombre="Romero Rubio" latitud="19.440805689888819" longitud="-99.09436225891113" numero="167">
  <vecino nombre="Oceanía(linea B)" numero="166" distancia="959"/>
  <vecino nombre="Ricardo Flores Magón" numero="168" distancia="1058"/>
</estacion>
<estacion nombre="Ricardo Flores Magón" latitud="19.43656655260893" longitud="-99.10364270210266" numero="168">
  <vecino nombre="Romero Rubio" numero="167" distancia="1058"/>
  <vecino nombre="San Lázaro(linea B)" numero="169" distancia="1057"/>
</estacion>
<estacion nombre="San Lázaro(linea B)" latitud="19.430293640460384" longitud="-99.11491870880127" numero="169">
  <vecino nombre="Ricardo Flores Magón" numero="168" distancia="1057"/>
  <vecino nombre="Morelos(linea B)" numero="170" distancia="1446"/>
  <vecino nombre="San Lázaro(linea 1)" numero="14" distancia="100"/>
</estacion>
<estacion nombre="Morelos(linea B)" latitud="19.438964359846697" longitud="-99.1182553768158" numero="170">
  <vecino nombre="San Lázaro(linea B)" numero="169" distancia="1446"/>
  <vecino nombre="Tepito" numero="171" distancia="648"/>
  <vecino nombre="Morelos(linea 4)" numero="71" distancia="100"/>
</estacion>
<estacion nombre="Tepito" latitud="19.442495243584336" longitud="-99.12345886230469" numero="171">
  <vecino nombre="Morelos(linea B)" numero="170" distancia="648"/>
  <vecino nombre="Lagunilla" numero="172" distancia="761"/>
</estacion>
<estacion nombre="Lagunilla" latitud="19.443587879724284" longitud="-99.13156986236572" numero="172">
  <vecino nombre="Tepito" numero="171" distancia="761"/>
  <vecino nombre="Garibaldi(linea B)" numero="173" distancia="624"/>
</estacion>
<estacion nombre="Garibaldi(linea B)" latitud="19.444377001250878" longitud="-99.13976669311523" numero="173">
  <vecino nombre="Lagunilla" numero="172" distancia="624"/>
  <vecino nombre="Guerrero(linea B)" numero="174" distancia="907"/>
  <vecino nombre="Garibaldi(linea 8)" numero="114" distancia="100"/>
</estacion>
<estacion nombre="Guerrero(linea B)" latitud="19.445044716468892" longitud="-99.14532423019409" numero="174">
  <vecino nombre="Garibaldi(linea B)" numero="173" distancia="907"/>
  <vecino nombre="Buenavista" numero="175" distancia="671"/>
  <vecino nombre="Guerrero(linea 3)" numero="50" distancia="100"/>
</estacion>
<estacion nombre="Buenavista" latitud="19.44650154013618" longitud="-99.15322065353394" numero="175">
  <vecino nombre="Guerrero(linea B)" numero="174" distancia="671"/>
</estacion>
<!--Fin Linea B del metro-->


<!--Linea 12 del metro-->
<estacion nombre="Tláhuac" latitud="19.286030941975667" longitud="-99.01413202285767" numero="176">
  <vecino nombre="Tlaltenco" numero="177" distancia="1460"/>
</estacion>

<estacion nombre="Tlaltenco" latitud="19.2944080815213" longitud="-99.02401059865952" numero="177">
  <vecino nombre="Tláhuac" numero="176" distancia="1460"/>
  <vecino nombre="Zapotitlán" numero="178" distancia="1070"/>
</estacion>

<estacion nombre="Zapotitlán" latitud="19.296496603486545" longitud="-99.03443902730942" numero="178">
  <vecino nombre="Tlaltenco" numero="177" distancia="1070"/>
  <vecino nombre="Nopalera" numero="179" distancia="1340"/>
</estacion>

<estacion nombre="Nopalera" latitud="19.299931847696435" longitud="-99.04577940702438" numero="179">
  <vecino nombre="Zapotitlán" numero="178" distancia="1340"/>
  <vecino nombre="Olivos" numero="180" distancia="1440"/>
</estacion>

<estacion nombre="Olivos" latitud="19.304204917429256" longitud="-99.0592360496521" numero="180">
  <vecino nombre="Nopalera" numero="179" distancia="1440"/>
  <vecino nombre="Tezonco" numero="181" distancia="621.45"/>
</estacion>

<estacion nombre="Tezonco" latitud="19.3063009084603" longitud="-99.06528979539871" numero="181">
  <vecino nombre="Olivos" numero="180" distancia="621.45"/>
  <vecino nombre="Periférico Oriente" numero="182" distancia="1500"/>
</estacion>

<estacion nombre="Periférico Oriente" latitud="19.317582852791187" longitud="-99.07438516616821" numero="182">
  <vecino nombre="Tezonco" numero="181" distancia="1500"/>
  <vecino nombre="Calle 11" numero="183" distancia="1310"/>
</estacion>

<estacion nombre="Calle 11" latitud="19.320453207536897" longitud="-99.08577114343643" numero="183">
```

```
  <vecino nombre="Periférico Oriente" numero="182" distancia="1310"/>
  <vecino nombre="Lomas Estrella" numero="184" distancia="849.88"/>
 </estacion>

 <estacion nombre="Lomas Estrella" latitud="19.32217691486308" longitud="-99.09560948610306" numero="184">
  <vecino nombre="Calle 11" numero="183" distancia="849.88"/>
  <vecino nombre="San Andrés Tomatitlan" numero="185" distancia="1240"/>
 </estacion>

 <estacion nombre="San Andrés Tomatitlan" latitud="19.32813255073551" longitud="-99.1043347120285" numero="185">
  <vecino nombre="Lomas Estrella" numero="184" distancia="1240"/>
  <vecino nombre="Culhuacán" numero="186" distancia="1120"/>
 </estacion>

 <estacion nombre="Culhuacán" latitud="19.336861860048803" longitud="-99.10887569189072" numero="186">
  <vecino nombre="San Andrés Tomatitlan" numero="185" distancia="1120"/>
  <vecino nombre="Atlalilco(linea 12)" numero="187" distancia="2150"/>
 </estacion>

 <estacion nombre="Atlalilco(linea 12)" latitud="19.356179080779647" longitud="-99.10130381584167" numero="187">
  <vecino nombre="Culhuacán" numero="186" distancia="2150"/>
  <vecino nombre="Mexicaltzingo" numero="188" distancia="994.52"/>
  <vecino nombre="Atlalilco(linea 8)" numero="128" distancia="100"/>
 </estacion>

 <estacion nombre="Mexicaltzingo" latitud="19.357674659097572" longitud="-99.12188440561295" numero="188">
  <vecino nombre="Atlalilco(linea 12)" numero="187" distancia="994.52"/>
  <vecino nombre="Ermita(linea 12)" numero="189" distancia="2580"/>
 </estacion>

 <estacion nombre="Ermita(linea 12)" latitud="19.36197152353576" longitud="-99.14293438196182" numero="189">
  <vecino nombre="Mexicaltzingo" numero="188" distancia="2580"/>
  <vecino nombre="Eje Central" numero="190" distancia="1040"/>
  <vecino nombre="Ermita(linea 2)" numero="42" distancia="100"/>
 </estacion>

 <estacion nombre="Eje Central" latitud="19.36134142505785" longitud="-99.15145576000214" numero="190">
  <vecino nombre="Ermita(linea 12)" numero="189" distancia="1040"/>
  <vecino nombre="Parque De Los Venados" numero="191" distancia="1270"/>
 </estacion>

 <estacion nombre="Parque De Los Venados" latitud="19.37077493389669" longitud="-99.15882647037506" numero="191">
  <vecino nombre="Eje Central" numero="190" distancia="1270"/>
  <vecino nombre="Zapata(linea 12)" numero="192" distancia="599.02"/>
 </estacion>

 <estacion nombre="Zapata(linea 12)" latitud="19.37072432640033" longitud="-99.16491240262985" numero="192">
  <vecino nombre="Parque De Los Venados" numero="191" distancia="599.02"/>
  <vecino nombre="Hospital 20 de Noviembre" numero="193" distancia="721.86"/>
  <vecino nombre="Zapata(linea 3)" numero="60" distancia="100"/>
 </estacion>

 <estacion nombre="Hospital 20 de Noviembre" latitud="19.372022403713373" longitud="-99.1709366440773" numero="193">
  <vecino nombre="Zapata(linea 12)" numero="192" distancia="721.86"/>
            <vecino nombre="Insurgentes Sur" numero="194" distancia="774.09"/>
 </estacion>

 <estacion nombre="Insurgentes Sur" latitud="19.373583625302963" longitud="-99.1787338256836" numero="194">
  <vecino nombre="Hospital 20 de Noviembre" numero="193" distancia="774.09"/>
  <vecino nombre="Mixcoac" numero="195" distancia="1040"/>
 </estacion>

 <estacion nombre="Mixcoac(linea 12)" latitud="19.376169602869975" longitud="-99.18781042098999" numero="195">
  <vecino nombre="Insurgentes Sur" numero="194" distancia="1040"/>
  <vecino nombre="Mixcoac(linea 7)" numero="112" distancia="100"/>
 </estacion>

 <!--Fin Linea 12 del metro-->


</linea>
```

BIBLIOGRAFÍA

Aurora Rodríguez, David Toca, Manuel Sánchez (2013, Mayo). Componentes de UI de android. [En línea]. Disponible: http://www.androideity.com

Propiedad de Google (2013). API de android. [En línea]. Disponible:http://developer.android.com/index.html

Richar Cabazo (2012, Diciembre). Ciclo de vida de una actividad en android. [En línea]. Disponible:http://www.monocode.net/2012/04/04/ciclo-de-vida-de-activity/

María Medina (2013, Abril, 25). Ciclo de vida de una actividad en android. [En línea]. Disponible:http://telekita.wordpress.com/2012/02/03/ciclo-de-vida-de-una-activity/

Yésica Hernández (2012, Mayo). Interfaz gráfica en android. [En línea]. Disponible:http://www.slideshare.net/jezabelink/interfaces-increibles-en-android

Propiedad de SlashMobility (2013, Abril). Localización geográfica en android. [En línea]. Disponible:http://www.slideshare.net/slashmobility/desarrollo-de-apps-android-basadas-en-localizacin

Propiedad de Androidforums (2013, Junio). Manejo de array de recursos en android. [En línea]. Disponible:http://androidforums.com/developer-101/388332-getting-multi-dimensional-array-resources.html

Propiedad de Stackoverflow (2013). Consulta de dudas especificas del desarrollo en android. [En línea]. Disponible:http://stackoverflow.com/

Propiedad de javaya (2012, Noviembre).Uso de los Intents en android. [En línea]. Disponible:http://www.javaya.com.ar/androidya/detalleconcepto.php?codigo=140&inicio=0

Jack (2013, Enero, 27). AlerDialog en android. [En línea]. Disponible:http://android-helper.blogspot.mx/search/label/AlertDialog

Richard Shaome (2013, Febrero, 10). Uso de servicios en android. [En línea]. Disponible:http://androide.hijodeblog.com/2010/06/17/creacion-de-aplicaciones-android-parte-4-servicios/

Prpiedad de Wikipedia (2013, Marzo, 17). Paso de datos entre actividades usando Intents en Android. [En línea]. Disponible:http://www.wikidroid.es/index.php?title=Intents_-_Traspaso_de_datos_entre_Activity