Capítulo IV. Implementación de Navin

Este capítulo describirá los aspectos técnicos de la implementación de Navin, un Servicio de Orientación para la Universidad de las Américas Puebla.

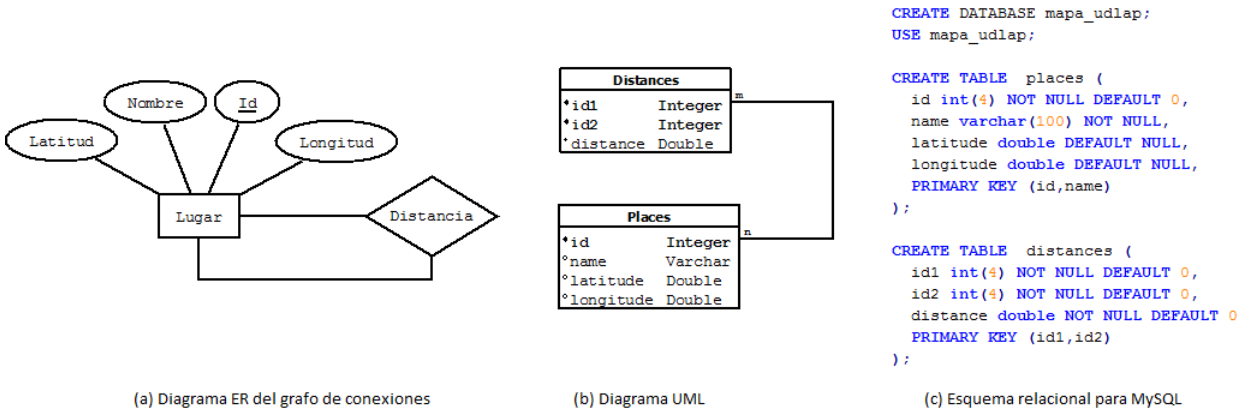
Para poder implementar este sistema, se desarrollaron diferentes herramientas, todas ellas en lenguaje Java.

En las secciones posteriores se mostrará el diseño de base de datos, la librería **PlaceOJBDB** que usa la base de datos, la técnica para construir grafos usando Google Maps, el uso de Axis2 para construir servicios web y la integración del servicio web del portal de noticias de la UDLAP. Finalmente se presentarán las extensiones necearías para desarrollar las aplicaciones móviles.

4.1 Base de datos de sitios de interés

La base de datos fue construida a partir de la descripción de los sitios de interés, estos están definidos, para este caso, con un id, un nombre y sus coordenadas (una latitud y una longitud). Cada sitio puede relacionarse con uno o muchos sitios por medio de su distancia ya que no todos los sitios se conectan por no haber un camino entre ellos o por ser físicamente imposible. Estas conexiones formarán las rutas posibles que más adelante se calcularán.

En la figura 4.1 se pueden observar los esquemas utilizados para modelar la base de datos de sitios de interés.



(a) Diagrama ER del grafo de conexiones (b) Diagrama UML (c) Esquema relacional para MySQL

Figura 4.1. Esquemas de base de datos

La figura 4.1.a muestra el esquema entidad – relación, la figura 4.1.b el diagrama en UML y finalmente la figura 4.1.c muestra el esquema relacional diseñado para construir la basa de datos en MySQL.

4.2 Librería PlaceOJBDB

Este paquete de clases escritas en Java forma la librería base del sistema. Esta librería modela los sitios de interés y provee las interfaces para poder actualizar, borrar o recuperar un sitio de interés fácilmente en una base de datos. La librería fue construida en base a las prácticas de *Domain-Driven Design* [31] y de la arquitectura MVC. La librería incluye las siguientes clases:

- **Place**. Representa a un sitio de interés con atributos necesario para definirlo, como su latitud, longitud, nombre, etc. tiene *Setters* y *Getters* para cada atributo.
- **IRepository**. Es una interfaz genérica para construir repositorios. De ella extienden los métodos que todo repositorio debería implementar.
- **PlaceRepository**. Representa una capa de acceso a los datos de los sitios de interés en la base de datos. Esta capa trata de ocultar el concepto base de datos y representando

la clase como si fuera una colección de objetos **Place** separando la capa de persistencia del dominio.

- **PlaceFactory**. Representa a un fabrica de objetos **Place** vacios. Tiene la responsabilidad de crear objetos nuevos que no estén en la base de datos.
- **DatabaseManager**. Representa una capa de acceso a datos que tiene directa relación con base de datos convirtiéndola en una capa de infraestructura. Tiene la responsabilidad de comunicarse con la base de datos por medio de lenguaje SQL.

El diagrama de clases de la librería puede verse en la figura 4.2:

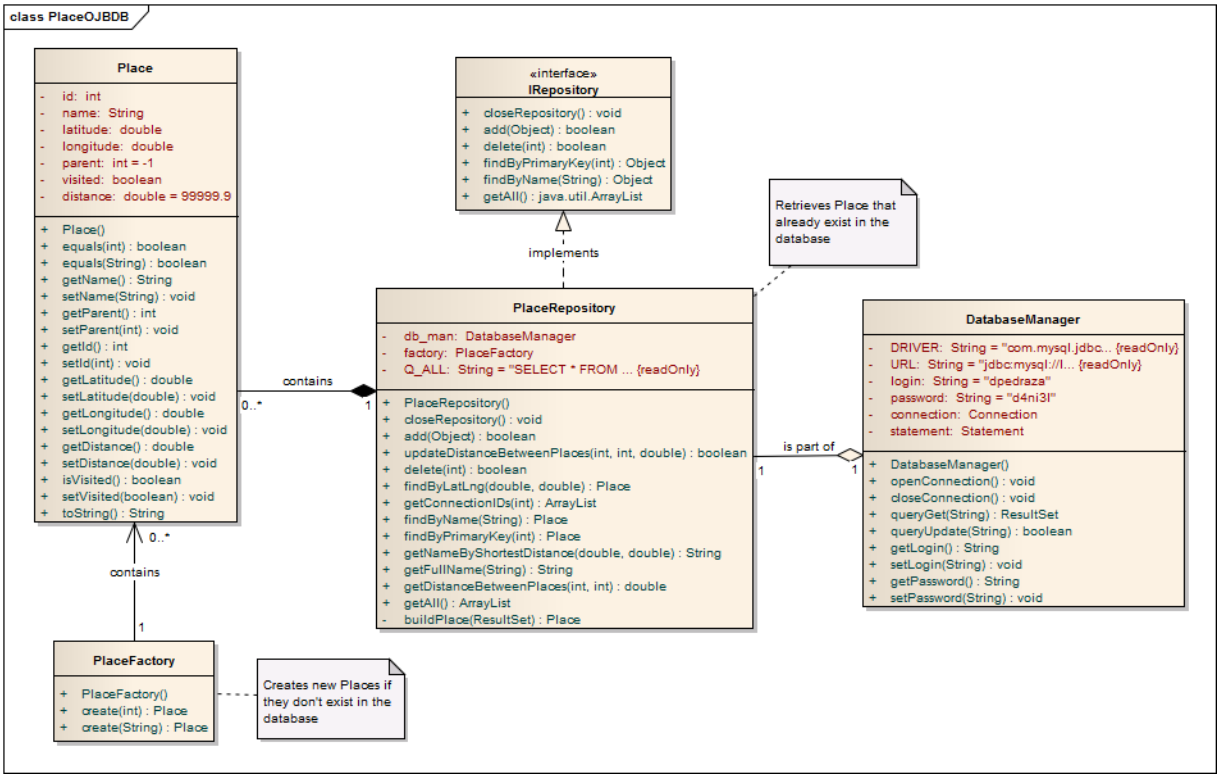


Figura 4.2. Diagrama de clases de la librería PlaceOJBDB

4.3 Modulo de construcción de grafos

Para poder construir la base de datos que usará Navin, se desarrollo un módulo especializado, utilizando la versión 2 del API de Google Maps para crear una aplicación web que permite

actualizar las coordenadas y los nombres de los sitios de interés, además de proveer métodos para conectar estos sitios de interés con otros y obtener su distancia sin tener que medirla manualmente. Este módulo de Navin construye un grafo del lugar con los sitios de interés y conexiones entre ellos, en este caso la UDLAP, para poderlo utilizar más adelante en la construcción de la ruta con el algoritmo de Dijkstra.

Para construir el grafo se deben colocar marcadores haciendo clic en el mapa y el modulo dibujará líneas que los unen como se ven en la figura 4.3. Estas líneas se interpretan como las conexiones entre ellos.

Figura 4.3. Grafo generado con Google Maps

Se escogió la versión 2 del API de Google Maps para desarrollar este modulo porque provee un método llamado *distanceFrom* que devuelve la distancia en metros que hay desde un punto a otro en mapa de Google Maps [9]. Sin esta función esa distancia se tendría que medir manualmente o implementar la fórmula Haversine, una fórmula que usa los conceptos de las razones trigonométricas y teoremas de senos y cosenos aplicados a las esferas. Usar esta fórmula implica ignorar las elevaciones y deformaciones de la esfera terrestre, haciendo el cálculo de la distancia con menor precisión [32].

En general este módulo podría extenderse en el futuro para permitir que se den de alta los sitios de interés de cualquier parte del mundo, ya que no solo se puede construir un grafo para una zona específica como la UDLAP, sino gracias a que está construido sobre la plataforma de Google Maps, la cual abarca todo el globo terráqueo con mapas digitales actualizados, es posible generalizarlo a cualquier zona del mundo.

El modulo de construcción de grafos de Navin usa la librería **PlaceOJBDB**, un Servlet **GMLLServlet** y una página web que usa el API de Google Maps.

El diagrama de clases correspondiente a este módulo puede verse en la figura 4.4.

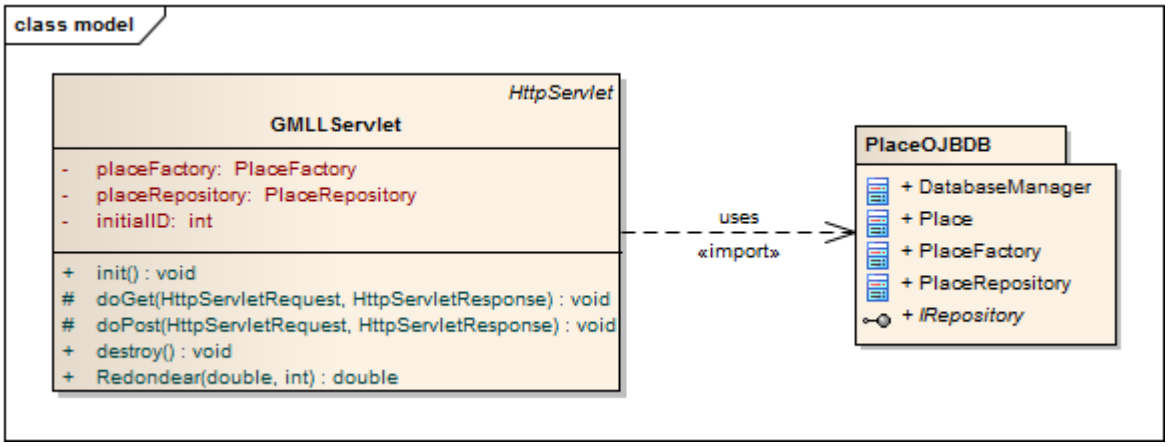


Figura 4.4. Diagrama de clases del modulo de construcción del grafo con Google Maps

Las acciones del clic en el mapa son escuchadas por un *listener* llamado *click* con parámetros *overlay:GOverlay, latlng:GLatLng* y *overlaylatlng:GLatLng*.

Este *listener* realizará las siguientes tareas [33]:

- Colocar un marcador (*GMarker*) en el mapa si se hizo clic en algún lugar sin *overlay* u *overlaylatlng*, es decir en marcadores o líneas. En esta acción también se obtendrán las coordenadas (latitud y longitud) del marcador.
- Cambiar un marcador por otro de otro color si se hizo clic en un marcador para indicar que se va a poder conectar siguiente marcador que se le dé un clic.

- Dibujar una línea (*GPolyline*) entre los marcadores conectados. Al mismo tiempo se recuperará la distancia que hay entre los marcadores conectados con el método *distanceFrom* de uno de los marcadores.

El código fuente de estos eventos puede verse más a detalle en el Apéndice C.

Los datos que se obtienen del mapa al hacer estas acciones son:

- Latitud y longitud de la posición de los marcadores.
- La distancia en metros que hay en las conexiones de los marcadores.

Al entregar estos datos al Servelt **GMLLServlet,** éste utilizará la librería **PlaceOJBDB**, definida en la sección anterior, para dar de alta los sitios de interés en la base de datos.

El proceso de alta de datos es el siguiente:

- Crear objetos **Place** vacios con **PlaceFactory**.
- Los objetos **Place** recientemente creados son completados con la información recuperada de Google Maps mediante sus *Getters.*
- Se actualiza la base de datos con **PlaceRepository** con los objetos creados anteriormente.

Si hay conexión entre puntos de interés, el Servlet también actualizará la base de datos, buscando primero si existe el punto en la base de datos y luego actualizando las distancias entre ellos.

4.4 Servicio web para rutas

El servicio web desarrollado sobre la plataforma Axis2 de Apache está compuesto de la librería **PlaceOJBDB** y de dos clases más que implementan el Algoritmo de Dijkstra y propiamente el servicio web. La figura 4.5 muestra el diagrama de clases de la librería OJBDB y las relaciones que hay entre las clases que constituyen el servicio web:

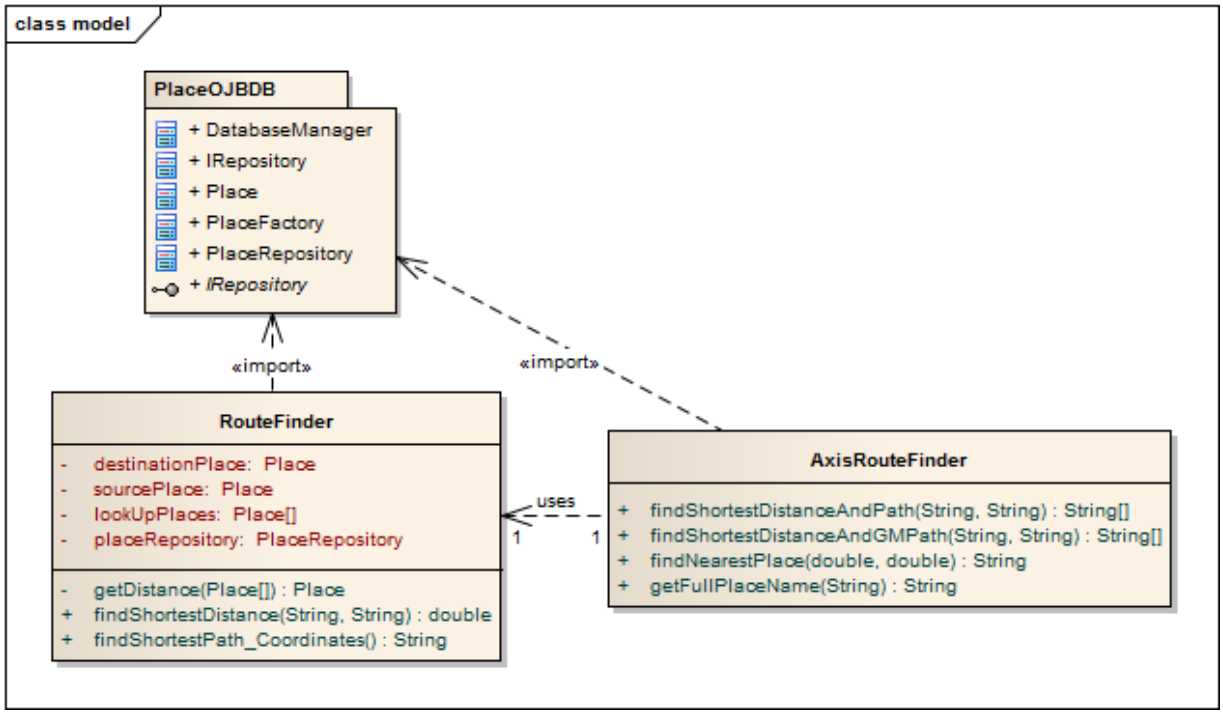


Figura 4.5. Diagrama de clases del servicio web

La descripción de las clases que constituyen el servicio web es la siguiente:

- **RouteFinder** es la clase que implementa el pseudocódigo del algoritmo de Dijkstra definido anteriormente por el Dr. Ricardo Pérez-Águila. Esta clase, principalmente, recupera la distancia mínima entre dos sitios, cuales quiera, definidos por sus nombres y reconstruye la ruta a seguir en el formato para dibujar el camino en un Google Static Map:

  path=rgba:0xff0000ff,weight:3|<latitude$_1$,longitude$_1$>|<latitude$_2$,longitude$_2$>|...<latitude$_n$,longitude$_n$>&markers=<latitude$_{destination}$,longitude$_{destination}$,blued>|<latitude$_{source}$,longitude$_{source}$,orangeo>

- **AxisRouteFinder** es la clase que proporciona los servicios web para las aplicaciones que quieran consumirlos. Los métodos que proporciona son los siguientes:

- **findShortestDistanceAndPath**. Recupera la distancia y la ruta más corta con formato de Google Static Map: *path=rgba:0xff0000ff,weight:3|<latitude$_1$,longitude$_1$>|<latitude$_2$,longitude$_2$>|…<latitude$_n$,longitude$_n$>&markers=<latitude$_{destination}$,longitude$_{destination}$,blued>|<latitude$_{source}$,longitude$_{source}$,orangeo>*. Este método recupera la distancia y la ruta más corta entre dos sitios cuales quiera por sus nombres.
- **findShortestDistanceAndGMPath**. Al igual que el anterior, recupera la distancia y la ruta más corta entre dos sitios cuales quiera por sus nombres. Este método es usado para recuperar la ruta con clases *google.maps.LatLng* para dibujarlas en un mapa de Google Maps, más adelante se hablará sobre esto.
- **findNearestPlace**. Recupera el nombre del sitio más cercano dadas una latitud y una longitud. Es usado para recuperar el nombre correcto de un sitio y buscar por nombre los sitios de interés.
- **getFullPlaceName**. Recupera el nombre correcto de un sitio dado un nombre cercano o parecido. Es usado para buscar por nombre los sitios de interés.

4.5 Aplicación web de noticias y eventos de la UDLAP para Java ME

Esta aplicación web hace uso del servicio web de noticias y eventos de la UDLAP. Este servicio web del Portal de Noticias y Eventos que fue desarrollado previamente por la Dirección General de Tecnologías de Información de la UDLAP, fue creado para la administración de noticias y eventos de toda la UDLAP y provee un número definido de noticias más recientes y eventos dada una fecha definida por el usuario. La aplicación web construida, consume dicho servicio web y filtra las noticias y eventos basándose en la

localización que el usuario precise. Si no se precisa ubicación, la aplicación no filtrará nada y enviará 10 noticias y los eventos de la fecha asignada. El diagrama de clases de la aplicación web puede verse con más detalle en el apéndice B.

La forma de acceder a esta aplicación web es mediante la URL http://srvudlaacad02.udla.fundacion.mx:8080/PhoneNewsWebApplication/NewsServlet?year=<año>&month=<mes>&day=<día>&source=<lugar>, donde <año>, <mes>, <día> y <lugar> son los parámetros para la fecha de búsqueda de eventos y el lugar donde se originaron las noticias y eventos de esa fecha.

La aplicación web de noticias y eventos de la UDLAP tiene una función de *proxy* para aplicaciones Java ME ya que no se pueden usar directamente un cliente del servicio web del Portal de noticias y eventos de la UDLAP dadas las limitaciones de soporte actual para ciertos tipos de datos como el tipo de dato *xsd:dateTime* (véase tabla de tipos de datos soportados de Java ME en apéndice B [34] para más información) el cual es usado para el filtrado de eventos en el servicio web del Portal de noticias y eventos de la UDLAP. Entonces, la aplicación web de noticias y eventos de la UDLAP consume el servicio web Portal de noticias y eventos de la UDLAP y envía las noticias y eventos pedidos por la aplicación Java ME en una cadena de caracteres que pueda recuperar correctamente la aplicación Java ME y así consumir indirectamente el servicio web Portal de noticias y eventos de la UDLAP aun cuando no tenga el soporte para el tipo de dato *xsd:dateTime*.

La cadena de caracteres enviada por la aplicación web de noticias y eventos de la UDLAP contiene los siguientes delimitadores para construir objetos *StringItem* o *ImageItem* en una aplicación Java ME:

- "&%". Representa el final de un *StringItem* o *ImageItem*

- "#$". Representa el fin de un *label* de un *StringItem*.
- "http://". Representa el URL de una imagen de un *ImageItem*.

4.6 Disponibilidad de Navin – UDLAP

Navin es una aplicación desarrollada para tres versiones de *clientes ligeros*:

- Para teléfonos celulares con plataforma Java ME y que soporten el API de localización JSR-179.
- Para teléfonos celulares con plataforma Java ME que no soporten dicho API.
- Para iPhone/iPod touch como una aplicación web.

Para instalar el software en teléfonos celulares con Java ME es necesaria la descargar del archivo *jar* de la aplicación desde la URL http://srvudlaacad02.udla.fundacion.mx:8080/iPhoneWebApp/downloads/NavinUDLAP.jar (para la versión con función de localización) y http://srvudlaacad02.udla.fundacion.mx:8080/iPhoneWebApp/downloads/NavinUDLAPlite.jar (para la versión sin función de localización). Para la versión de iPhone / iPod touch, no es necesaria la instalación del software ya que es una aplicación web que se encuentra en la siguiente URL: http://srvudlaacad02.udla.fundacion.mx:8080/iPhoneWebApp. El usuario debe ingresar la URL en el navegador web Safari de iPhone/iPod touch para poder utilizar el software.

Los servicios web y aplicaciones web se encuentran instalados en el servidor institucional para pruebas académicas de la UDLAP: srvudlaacad02.udla.fundacion.mx. Dadas las restricciones de seguridad del servidor de la UDLAP, Navin solo puede ser usado dentro del campus de universidad, usando la red WiFi de la misma para usar los servicios web de Navin.

4.6.1 Navin versión 1.0iw

Es una aplicación web especialmente diseñada para iPhone/iPod touch accesible solo en el campus de la UDLAP utilizando la red WiFi institucional en la siguiente URL mencionada anteriormente: http://srvudlaacad02.udla.fundacion.mx:8080/iPhoneWebApp.

Como es una aplicación web, no se necesita ninguna instalación adicional y es accesible por medio del explorador web Safari. Está construida con la ayuda iWebKit para simular las interfaces nativas de iPhone / iPod touch, con el fin de hacer más *amigable* y accesible la aplicación.

La aplicación web consta de dos partes:

1) **Mapas y rutas**. El usuario puede precisar el lugar donde se encuentra y el lugar a donde quiere ir por medio un mapa de Google Maps en JavaScript, el cuál es una versión especial para dispositivos móviles como iPhone/iPod touch y teléfonos con Android [35]. En la figura 4.6 puede apreciarse como se ve el mapa dentro de la versión de Navin para iPhone / iPod touch, en el se muestran los marcadores de origen (icono de color naranja) y destino (icono de color azul):

Figura 4.6. Google Maps en Navin

Este mapa tiene la peculiaridad de usar los gestos naturales de la interfaz de entrada de iPhone/iPod touch para recorrer, alejar y agrandar objetos. Esta interfaz tiene la capacidad de poner marcadores cuando se toca el mapa, uno para definir el lugar de origen y otro para definir el lugar de destino.

También puede usar la función "Mi ubicación" que permite obtener la ubicación del dispositivo por medio de su GPS (como el caso del iPhone 3G o 3GS) o por medio de la red WiFi (como el caso de iPod touch). Las clases que muestran la localización y rastreo del dispositivo ocultan los detalles de la implementación y es totalmente transparente [22]. Estas clases forman parte de la implementación de la especificación del API de Geolocalización de la W3C, son las mismas clases que actualmente se usan en exploradores web como Mozilla Firefox [23].

Los dispositivos iPhone/iPod touch que quieran utilizar esta función deben contar con la versión del sistema operativo iPhone OS 3.0 o posterior y tener habilitada la función de Localización.

Al usar esta función, el mapa dibujará un círculo que muestra la precisión del API y un marcador en el centro, el área donde se encuentra el usuario es este círculo. La precisión puede variar de unos cuantos metros hasta kilómetros. El código fuente de esta función puede verse más a detalle en el Apéndice C. La figura 4.7 muestra la respuesta de la función localización "Mi ubicación".

Figura 4.7. "Mi ubicación" en Navin

Si no se puede ubicar en el mapa, el usuario también puede usar los nombres de las ubicaciones o usar una combinación entre su punto de ubicación actual y el nombre del lugar a donde quiere ir. Estos datos serán procesados por un Servlet que consumirá el servicio web de rutas.

El resultado es la ruta dibujada en un mapa de Google Maps después del procesamiento con el servicio web para rutas junto con la distancia a recorrer y la información de eventos y noticias de los puntos de origen y destino.

El mapa resultante tendrá la posibilidad de hacer acercamientos, como el mapa de entrada, además de tener las siguientes características:

- Una línea en color rojo semitransparente *google.maps.Polyline* que representa la ruta a seguir por el usuario.
- Dos marcadores iguales a los que se usaron para definir los puntos origen y destino en el mapa anterior.
- 2 ventanas con información sobre el punto origen y el punto destino (noticias y eventos) y otra ventana para la información de la distancia de la ruta. Estas se abren al tocar los marcadores y la línea.

En la figura 4.8 se puede apreciar la ruta construida en Google Maps con las ventanas de información de los puntos de origen y destino.

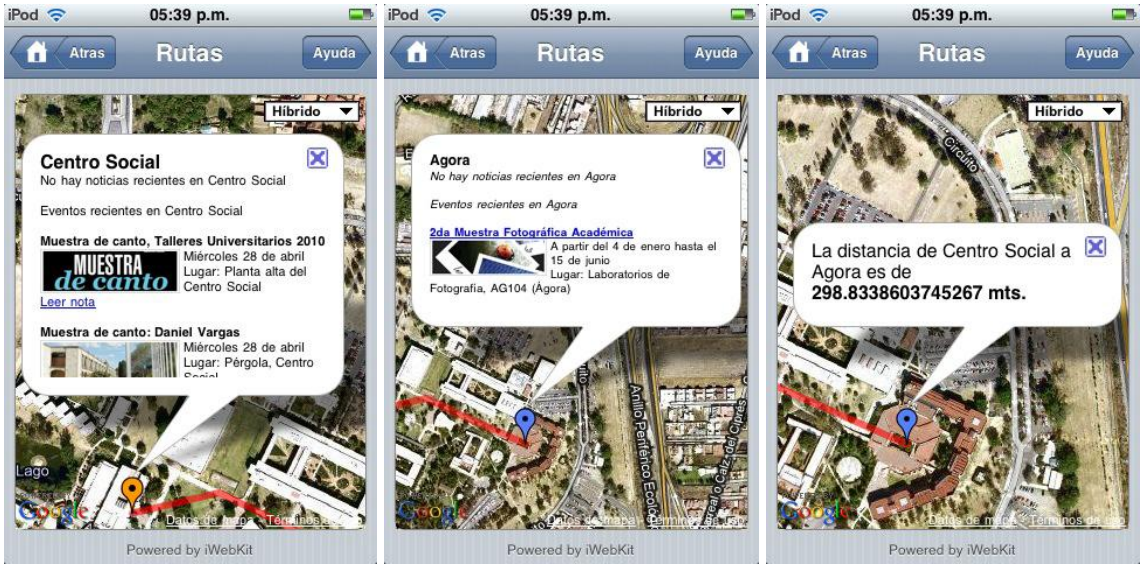


Figura 4.8. Resultado de la ruta en Navin

2) **Noticias y eventos**. En esta opción, se provee también un mapa para ubicar el lugar en donde se encuentra el usuario y la opción "Mi ubicación". A diferencia de la opción de Mapas y Rutas, el usuario solo puede poner un marcador y definir la fecha de búsqueda para los eventos. En esta opción se presenta una barra de búsqueda en la parte superior que tiene la función de buscar las noticias y eventos del lugar que se especifique en la barra. En la figura 4.9 se puede apreciar la pantalla de la opción de Noticias.

Figura 4.9. Noticias y eventos en Navin

Los datos de la ubicación del usuario y la fecha de búsqueda serán procesados por otro Servlet que consumirá el servicio web del Portal de noticias y eventos de la UDLAP y filtrará las noticias y eventos en base a la ubicación del usuario.

El resultado que se presentará es un conjunto de noticias y eventos relacionados con la posición del usuario o todos los eventos y noticias si no se definió ninguna ubicación

de búsqueda. En la figura 4.10 se puede observar un ejemplo de la ejecución de la aplicación al recuperar las noticias y eventos sin definir una ubicación específica.

Figura 4.10. Resultado de búsqueda de eventos y noticias basados en localización

Esta aplicación web, como ya se mencionó, usa Servlets y JSP para mostrar los resultados en el formato deseado para su apropiada disposición en la pantalla de iPhone / iPod touch.

El diagrama de clases y otros ejemplos de ejecución de la aplicación se encuentran en los apéndices B y D respectivamente.

4.6.2 Navin versión 1.0cs y 1.0bs

Estas aplicaciones fueron construidas con el lenguaje de programación Java ME y son básicamente la misma, la diferencia radica en que la versión *cs* no puede utilizar el API de localización de Java ME, siendo una versión para dispositivos más limitados.

La instalación de este cliente ligero se hace por medio un archivo *jar* que se copia al dispositivo móvil y este se instalará al accederlo.

Esta versión de Navin no cuenta con localización por red WiFi y solo puede usarse la localización por GPS por medio del API de localización de Java ME (JSR-179).

Al igual que en la versión para iPhone/iPod touch hay dos opciones de uso:

1) **Mapas y rutas**. Se podrá consultar la ruta más corta y la distancia a recorrer entre dos puntos, uno llamado origen y el otro destino, en la UDLAP. Utilizando la función de "Mi ubicación" el sistema podrá detectar el lugar en donde se encuentra el usuario sin tener que teclear el nombre directamente, esta función puede tardar hasta dos minutos en encontrar la posición del usuario y es necesario conectarse a internet para usar esta función. En la figura 4.11 se muestra pantalla donde recupera la localización del usuario.

Figura 4.11. Pantalla de localización del usuario

A diferencia de la aplicación para iPhone/iPod touch, no hay la opción de usar un mapa para ubicarse y poner marcadores para determinar las ubicaciones origen y destino.

El resultado de la ruta se mostrará de forma similar a la aplicación iPhone/iPod touch en pantalla completa y se podrá hacer *scroll* en todo el mapa. En la figura 4.12 puede apreciarse cómo se despliega la ruta y la información relacionada con la misma.

Figura 4.12. Resultado de la ruta

La forma de recuperar el mapa de Google Mapas se hace de la misma forma que en la iPhone/iPod touch.

2) **Noticias y eventos**. El sistema también puede recuperar las noticias más noticas y eventos de la UDLAP tal como la versión iPhone/iPod touch. La diferencia con el sistema anterior es que Navin para Java ME no puede utilizar un cliente del servicio web del Portal de Noticias y Eventos de la UDLAP por las limitaciones de la plataforma como se mencionó antes, así que se usó la Aplicación web de noticias y eventos de la UDLAP para Java ME descrita en el capítulo anterior**.** En la figura 4.13 puede observarse la ejecución de la aplicación cuando recupera las noticias y eventos sin definir una ubicación específica.

Figura 4.13. Recuperación de noticias y eventos en Navin para Java ME

Además de estas opciones también se puede configurar la precisión del GPS y el uso de la red WiFi en vez del plan de datos ofrecido por el proveedor de servicio de telefonía celular, con el fin de hacer más personal la aplicación.

El diagrama de clases y otros ejemplos de ejecución de la aplicación se encuentran en los apéndices B y D respectivamente.

4.7 Resumen

En este capítulo se presentaron las implementaciones de las diferentes herramientas, servicios y clientes que forman el Sistema Navin. Se presentaron los diagramas de clase de cada una de las implementaciones y su explicación correspondiente, así como las funciones, métodos y atributos más importantes que integran a cada implementación.

Finalmente se presentaron las implementaciones de los clientes finales de Navin y todas las funciones disponibles para cada uno, y se explicó cómo se utilizan las funciones de localización en cada uno de ellos.