Junio | 11

UNIVERSIDAD POLITECNICA DE VALENCIA

Proyecto Final de Carrera

Desarrollo de Aplicación Móvil para la Visualización del Transporte Público en Tiempo Real sobre plataforma iPhone/iPad

Autor:	Daniel Soro Coicaud
Director:	Roberto Agustín Vivó Hernando
Tutor Empresa:	Pedro Jorquera Hervás
Empresa:	Okode, S.L.
Código PFC.:	DSIC-160

okode

Okode · Edificio 9B · Ciudad Politécnica de la Innovación
Camino de Vera s/n, 46022 Valencia, www.okode.com

Tabla de Contenidos

Antecedentes

El desarrollo de la aplicación móvil para la visualización de transporte público en tiempo real para plataformas iPhone/iPad forma parte del proyecto final de carrera llevado a cabo por el alumno de Ingeniería Informática, Daniel Soro Coicaud.

La aplicación que se describe en este documento se corresponde con el proyecto DSIC-160, dirigido por el profesor del DSIC, Roberto Agustín Vivó Hernando, e implementado en la empresa Okode, durante su estancia en ésta en periodo de prácticas, bajo la supervisión y tutoría de su director Pedro Jorquera Hervás.

Okode es una consultora tecnológica experta en desarrollo de soluciones para Internet y dispositivos móviles. Entre las actividades de esta empresa destacan el desarrollo de aplicaciones móviles para iPhone, iPad y Android, portales web 2.0, integración de sistemas de información geográfica basados en Google Maps y aplicaciones de marketing interactivo para redes sociales (Facebook).

La idea de este proyecto surge en base a la creación previa de una aplicación web con el mismo fin, creada por Okode. Este proyecto, llamado inicialmente "*Mobicity*" y posteriormente renombrado como "*Mobitransit*", permitía la visualización de los transportes públicos de la ciudad de Helsinki (Finlandia) en tiempo real en los principales navegadores web: Internet Explorer, Mozilla Firefox, Safari… independientemente del sistema operativo y sin necesidad de instalar ninguna aplicación o plugin adicional. Sobre un mapa de Google Maps, se mostraban marcadores representativos de los vehículos moviéndose por sus calles y mostrando su orientación en todo momento.

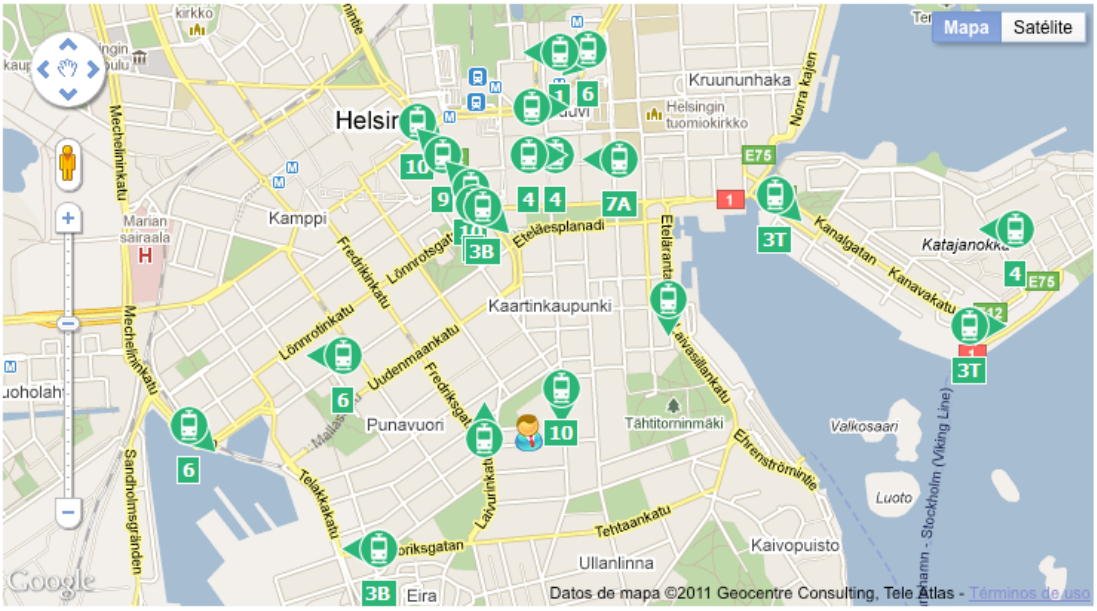

Ilustración 1 - Aplicación web Mobitransit original

La aplicación de "*Mobitransit Helsinki*" se hizo posible en parte, gracias al proyecto *Reittiopas* (http://developer.reittiopas.fi/pages/en/home.php) desarrollado por la empresa finlandesa HSL, principal responsable del transporte público en la región de Helsinki. Se trata de un servicio de consulta que ofrece de forma libre, información sobre transportes públicos de Helsinki bajo demanda. En todo momento, puede obtenerse la geolocalización, orientación e información de línea de cualquier vehículo registrado en esta aplicación. Además, desde la página principal de HSL (http://www.hsl.fi/EN/Pages/default.aspx), pueden consultarse también otros servicios relacionados con el transporte público, como tiempos de espera desde una determinada parada, avisos e incidencias de tráfico…

Tomando como objetivo llevar el proyecto Mobitransit al siguiente nivel, se decidió iniciar la creación de la aplicación nativa para dispositivos móviles. De entre las diversas posibilidades sobre las cuales implementar este proyecto, se estableció la plataforma de Apple, para hacer posible visualizar Mobitransit en los dispositivos iPhone y iPad.

Ilustración 2 - Dispositivo móvil iPhone 4

El iPhone es un teléfono inteligente multimedia con conexión a Internet, pantalla táctil capacitiva (con soporte multitáctil) y una interfaz de hardware minimalista de la compañía Apple Inc. Ya que carece de un teclado físico, contiene uno virtual en la pantalla táctil en orientación tanto vertical como apaisado.

El iPhone 3GS dispone de una cámara de fotos de 3 megapíxeles y un reproductor de música además de software para enviar y recibir mensajes de texto y mensajes de voz. También ofrece servicios de Internet como leer correo electrónico, cargar páginas web y conectividad por Wi-Fi. La primera generación de teléfonos eran GSM cuatribanda con la tecnología EDGE; la segunda generación ya incluye UMTS con HSDPA.

El nuevo iPhone 4 contiene el procesador Apple A4, 512 MiB de memoria RAM, un panel de alta resolución, 2 cámaras, una de 5 megapíxeles con opción de grabar en HD 720p y la otra VGA.

Ilustración 3 - Dispositivo tablet iPad

El iPad es un dispositivo electrónico tipo tablet desarrollado por Apple Inc. Se sitúa en una categoría entre un "teléfono inteligente" y un portátil, enfocado más al acceso que a la creación de contenido.

Las funciones son similares al resto de dispositivos portátiles de Apple, aunque la pantalla es más grande y su hardware más potente. Funciona a

través de una NUI (Interfaz natural de usuario), rediseñada para aprovechar el mayor tamaño del dispositivo y la capacidad de utilizar software para lectura de libros electrónicos, navegación web y correo electrónico, además de permitir el acceso al usuario a otras actividades multimedia.

Posee una pantalla con retroiluminación LED y capacidades multitáctiles de 9,7 pulgadas (24,638 cm), de 16 a 64 gigabytes (GB) de espacio en memoria flash, Bluetooth, y un conector dock de 30 pines. Existen dos modelos: uno con conectividad a redes inalámbricas Wi-Fi 802.11n y otro con capacidades adicionales de GPS y soporte a redes 3G (puede conectarse a redes de telefonía celular HSDPA).

Ambos dispositivos disponen de un kit de desarrollo de software (SDK) que permite a cualquier programador crear aplicaciones para éstos a través de la herramienta Xcode mediante el lenguaje de programación Objective-C.

Una vez implementada la aplicación para sendos dispositivos, se daría a conocer al mundo registrándola como aplicación de la App Store.

Ilustración 4 - Logotipo de App Store

App Store es un servicio para iPhone, iPod, iPad y Mac OS X creado por Apple Inc. que permite a los usuarios buscar y descargar aplicaciones informáticas desarrolladas para estos sistemas. El kit de desarrollo de software (SDK) permite, además de crear estas aplicaciones a desarrolladores particulares, registrar dichas aplicaciones en la plataforma de Apple, permitiendo compartirlas con todos los usuarios del App Store de forma gratuita o mediante compra.

Así pues, la presente documentación de proyecto final de carrera, describirá los objetivos, herramientas, arquitectura y solución establecida para la integración y ampliación del proyecto “Mobitransit” como aplicación nativa para los dispositivos iOS y su posterior distribución a través del servicio App Store.

Objetivos del Proyecto

Objetivo general del proyecto

El objetivo que persigue esta propuesta de Proyecto Final de Carrera es llevar a cabo la implementación software de una aplicación que permita visualizar en tiempo real, la ubicación exacta y movimiento de los transportes públicos (autobuses, tranvías…) en la ciudad de Helsinki, representándolas gráficamente en los dispositivos iPhone/iPad sobre la aplicación de Google Maps que integra la SDK de ambos dispositivos. El usuario final podrá saber en todo momento dónde se encuentran y qué dirección siguen cada uno de los transportes públicos de su ciudad, así como la información de las paradas correspondientes a estos vehículos, todo ello en una sencilla, atractiva e intuitiva interfaz que facilitará el uso de la aplicación, convirtiéndola en una herramienta de consulta de uso diario.

Objetivos particulares del proyecto

El proyecto abarca la resolución de distintas fases fundamentales para alcanzar su objetivo principal, así como algunas ampliaciones establecidas para mejorar y hacer más llamativa la aplicación para el usuario final.

Como requisitos básicos de implementación se estableció que el sistema final implementado, debía ser capaz de:

- **Adquirir la geolocalización de los transportes públicos** a partir de la señal de GPS que tengan integrados.
- **Servir y representar la información de geolocalización en tiempo real** sobre los dispositivos móviles mediante la librería de Google Maps.
- **Añadir información adicional** relacionada con los medios de transporte así como paradas, rutas de los vehículos, tiempos de espera, etc…
- **Permitir la geolocalización del usuario** en la propia aplicación para poder orientarse con respecto a las señales que esté visualizando.
- **Crear estadísticas de uso** a través de la librería de Google Analytics para realizar estudios de mercado posteriores.

Y a modo de ampliación, se agregaron los siguientes puntos:

- **Permitir el filtrado de visualización de transportes, paradas y rutas** para que el usuario final pudiera establecer qué elementos quería que se mostraran en el mapa, y guardar estas propiedades de filtrado automáticamente para un futuro uso de la aplicación.

- **Mostrar las alertas e incidencias** publicadas por el servicio de transporte público de la ciudad con el fin de informar al usuario del estado del tráfico de la ciudad. (Integrado únicamente en iPhone)

- **Compartir en redes sociales (Facebook y Twitter)** información relacionada con la aplicación a modo de mensaje textual editable por el usuario. (Integrado únicamente en iPhone)

- **Servicios de SMS**, aprovechando los servicios ofrecidos por la ciudad de Helsinki, el usuario podrá comprar un ticket de tranvía/autobús enviando un SMS con su iPhone o solicitar un taxi a su posición actual por la misma vía. (Integrado únicamente en iPhone)

Alcance del proyecto

La solución planteada está basada en una aplicación nativa, interactiva y multitáctil cuyo alcance cubre los siguientes aspectos:

- Muestra la ubicación exacta de los transportes públicos de la ciudad de Helsinki sobre un mapa de Google Maps y actualiza su posición de forma inmediata así como su orientación. De cada vehículo se puede obtener información sobre la línea a la que pertenece, así como su matrícula, pulsando sobre él en cualquier momento.
- Permite la visualización de las paradas de cada línea mediante marcadores ubicados en el mapa. Pulsando sobre estos marcadores, se puede obtener información directa del nombre de la parada y de las líneas que hacen uso de la misma. En segunda instancia, puede consultarse también, los tiempos aproximados de llegada de los vehículos a cada parada.
- Mediante líneas de colores sobre el mapa, puede visualizarse el recorrido por el que circulan los transportes públicos.
- Puede mostrar la ubicación del usuario en el mapa haciendo uso de la geolocalización por GSM o WiFi del dispositivo.
- Filtrado de objetos a visualizar en el mapa. Mediante un listado de líneas, el usuario podrá ocultar/mostrar los vehículos, paradas o trayectos de las líneas que desee.
- Obtención de avisos relacionados con las incidencias de tráfico que puedan ocurrir en el servicio de transporte público de Helsinki. Integrado únicamente en los dispositivos tipo iPhone.
- Con un dispositivo iPhone, se podrán comprar tickets de autobús/tranvía enviando un simple SMS o también, solicitar un taxi a la ubicación actual del usuario (mediante geolocalización o escribiendo a mano la dirección). Este servicio únicamente será efectivo en la ciudad de Helsinki.
- Se podrá iniciar una sesión con la cuenta del usuario en las redes sociales de Facebook y Twitter con la aplicación para compartir en las redes sociales la ubicación del usuario o el uso de ciertos transportes mediante un dispositivo tipo iPhone.
- La aplicación será descargable de forma gratuita desde App Store. Una vez descargada e instalada en el dispositivo del cliente se iniciará como cualquier otra aplicación de móvil pulsando sobre su icono en la pantalla de inicio. Presenta las diferentes vistas como el mapa principal, listado de líneas, información de servicios y configuración.

- La aplicación podrá ser utilizada por cualquier usuario anónimo que disponga de un dispositivo iPhone / iPad con acceso al App Store de Apple.
- La aplicación se podrá utilizar en cualquier lugar y en cualquier momento ya que se trata de una aplicación para móvil. Esto hace de la aplicación una práctica herramienta para determinar qué transporte conviene coger en cada momento.
- Cada vez que el usuario realice una operación significante, se enviará un evento a Google Analytics para el procesado estadístico.

Cobertura de requerimientos

Entorno tecnológico

La solución desarrollada cumple los siguientes requisitos tecnológicos:

- Escalabilidad del sistema: La solución resultante de este Proyecto facilita la cobertura de las necesidades futuras del servicio y su evolución conforme a los cambios que se produzcan en el área de transporte público de la ciudad de Helsinki (por ejemplo, inserción de nuevas líneas, paradas, vehículos…).
- Desacoplamiento entre capas: La solución propuesta presenta una clara separación entre los diferentes dominios o capas en que se estructura permitiendo de forma natural la evolución del sistema así como la reutilización de componentes. Concretamente se han determinado tres capas diferenciadas sobre las que podrán realizarse cambios sin afectar excesivamente al resto de capas: aplicación nativa móvil, información generada previamente (como archivos de configuración) y finalmente sistemas de información existentes, servicios web y agentes.
- Monitorización: El sistema propuesto permite su monitorización mediante el uso de Google Analytics, lo que permitirá estudiar el uso de la aplicación por parte de los diferentes usuarios que la instalarán en sus dispositivos.
- Modularización: La lógica de aplicaciones se ha desarrollado aplicando criterios de modularidad a fin de favorecer la reutilización de código y mantenibilidad de la misma.
- Rendimiento: El diseño de la solución ha sido realizado valorando la optimización del rendimiento como un factor clave de éxito. Es por ello que la aplicación nativa no dispone al ser instalada del todas las líneas, vehículos, paradas y rutas sino que la descarga de las mismas se producirá sobre demanda a través de Internet (con conexiones WiFi / 3G / GPRS) lo que hará necesario que los paquetes de datos transmitidos estén optimizados para minimizar el consumo de ancho de banda.

- Multiidioma: La aplicación está diseñada e implementada de forma tal que se soporte multiidioma parametrizando su inicio con el *locale* actual del dispositivo. Las cadenas de texto asociadas a los idiomas se externalizan en un módulo independiente que permite ser extendido para soportar más idiomas. Se han establecido los literales de cadena en su totalidad para el idioma Castellano e Inglés, y parcialmente para Finlandés, Francés e Italiano (dejando en inglés aquellos textos que no han sido traducidos) aunque el componente está listo para soportar otros idiomas en el momento en que se realicen las traducciones correspondientes de los mensajes de la aplicación.

- Disponibilidad 24x365: La aplicación desarrollada permite la operativa de negocio en condiciones de 24x365.

Entorno de desarrollo

Aplicaciones de organización y seguimiento

Subversion

Ilustración 5 - Logotipo de Subversion

Sistema de control de versiones del repositorio de código. Permite almacenar en un repositorio específico, recursos tanto de código fuente como binarios, descargarlos desde cualquier cliente y almacenar sus cambios. En todo momento se puede consultar el historial de modificaciones que ha sufrido un proyecto, para comprobar los avances realizados o volver a versiones previas para arreglar posibles fallos.

Jira

Ilustración 6 - Logotipo de Jira

Aplicación web para el seguimiento de errores, de incidentes y gestión operativa de proyectos. Con esta aplicación se pueden planificar los hitos del proyecto, describiendo sus objetivos y tareas relacionada, comprobar el estado de las incidencias pendientes y registrar su resolución.

Confluence

Ilustración 7 - Logotipo de Confluence

Aplicación web para la gestión de contenidos y plataforma de colaboración. Permite crear y compartir contenido, centralizando la gestión de conocimiento para que la información sea accesible siempre.

Aplicaciones de desarrollo

Xcode

Ilustración 8 - Logotipo de Xcode

Es el entorno de desarrollo integrado de Apple Inc. Incluye la colección de compiladores del proyecto GNU (GCC) y puede compilar código C, C++, Objective-C, Objective-C++, Java y AppleScript mediante una amplia gama de modelos de programación como Cocoa, Carbón y Java.

La aplicación cliente del proyecto (tanto en su versión para iPad como para iPhone) ha sido implementada en su totalidad con Xcode, haciendo uso del lenguaje de programación Objective-C y de los recursos de interfaz genéricos (botones, barras de navegación, menús…) establecidos por los desarrolladores de

Apple. Permite realizar proyectos de carácter "universal" en la que se estructuran tres secciones de código y recursos: uno para los dispositivos de tipo iPhone/iPod, otro para los de tipo iPad y un último de recursos y código compartido, permitiendo así que una misma aplicación pueda ejecutarse en distintos dispositivos haciendo cada uno uso de sus propios recursos o compartiendo aquellos que ambos pueden utilizar.

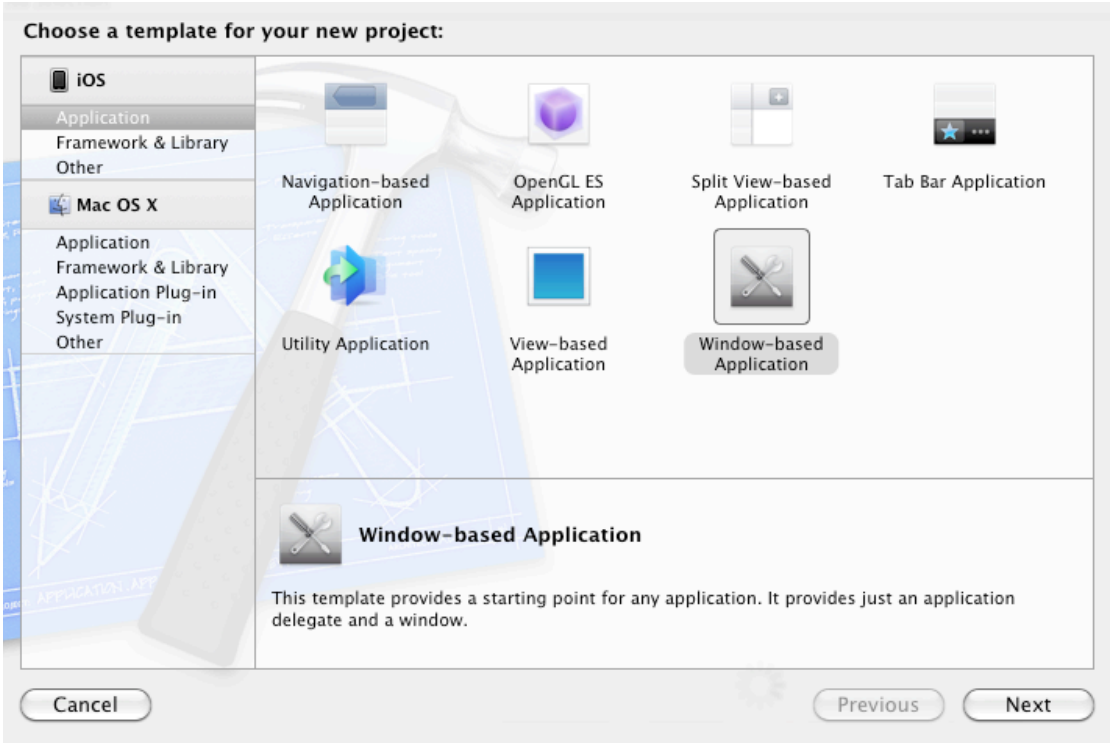

Ilustración 9 – Creación de nueva aplicación basada en ventana para dispositivo "Universal" en Xcode

Para el diseño gráfico de la aplicación se hará uso también de la utilidad llamada "Interface Builder", integrada en Xcode desde su versión 4.0. Mediante esta herramienta se puede crear de manera rápida y fácil una interfaz de usuario con los componentes básicos prediseñados por Apple y enlazarlos con la funcionalidad implementada en el proyecto de Xcode.

Finalmente, entre las herramientas y utilidades de Xcode, encontramos también el "iOS Simulator", que nos permitirá probar nuestra aplicación en local, sin necesidad de instalarla en ningún dispositivo físico.

Ilustración 10 - iOS Simulator, adaptado a iPhone (izquierda) y a iPad (derecha)

Eclipse

Ilustración 11 - Logotipo de eclipse

Eclipse es un entorno de desarrollo integrado (IDE, Integrated Development Environment) que facilita enormemente la tareas de edición, compilación y ejecución de programas durante su fase de desarrollo. Aunque Eclipse pretende ser un entorno versátil soportando varios lenguajes de programación, es con el lenguaje Java con el que mejor se integra y con el que ha ganado su popularidad.

Eclipse será nuestra principal herramienta para la creación, configuración y mantenimiento de los servicios web y agentes del apartado del servidor del proyecto.

Servicios de auxiliares

ActiveMQ

Ilustración 12 - Logotipo de ActiveMQ

Servicio intermediario que implementa el servicio de tiempo real Java. Mediante el protocolo Stomp, permite enviar mensajes a discreción a los agentes registrados al servicio implementado.

Google Analytics

Ilustración 13 - Logotipo de Google Analytics

Es un servicio gratuito de estadísticas de sitios web y aplicaciones nativas con funcionalidades basadas en web. Ofrece información agrupada según los intereses de tres tipos distintos de personas involucradas en el funcionamiento de una página: ejecutivos, técnicos de marketing y webmasters.

Se pueden obtener informes como el seguimiento de usuarios exclusivos, el rendimiento del segmento de usuarios, los resultados de la campaña de marketing, el marketing de motores de búsqueda, las pruebas de versión de anuncios, el rendimiento del contenido, el análisis de navegación, los objetivos y proceso de redireccionamiento o los parámetros de diseño web. También permite registrar eventos personalizados y estructurados por categoría y acción realizada añadiéndolos a los informes estadísticos estableciéndoles valores numéricos, igualmente personalizados.

Diseño e implementación

Diseño e implementación del servidor

Gestión de la configuración

Archivos de configuración Se exponen tres archivos de configuración inicial en formato XML, comprimidos en un único paquete. Estos archivos proporcionan información básica del servicio de transporte público y de la ciudad: ubicación en el mapa, área geográfica que ocupa, tipos de transporte que posee la ciudad, líneas existentes, paradas y rutas relacionadas con cada línea. La aplicación establece además un procedimiento de comunicación que minimiza la descarga de este recurso desde los dispositivos de forma que si no hubiera cambios en la configuración inicial no será necesario volver a descargar la información.

Servicios

Servicios de consulta

- Servicio de obtención de vehículos Bajo demanda se puede obtener, en todo momento, los vehículos registrados que están recorriendo la ciudad, su tipo, ubicación, orientación, matrícula, línea a la que pertenecen. Este servicio se consulta tras establecer la configuración inicial, para crear los marcadores que serán representados sobre el mapa.

- Obtención de tiempos de parada Otro servicio de consulta que resultará útil para nuestra aplicación, será el de obtención de tiempos de llegada. A partir de un identificador de parada, podremos conseguir los tiempos de llegada de los autobuses a dicha parada con respecto a la hora actual.

- Obtención de avisos de tráfico Finalmente, y aprovechando los recursos que poseemos, también se solicitará información sobre las incidencias de tráfico registradas por el servicio de transporte público de la ciudad, para informar al usuario de cualquier contratiempo que pudiera surgir.

Servicios de suscripción

- Servicio de información en tiempo real Una vez establecida la configuración inicial y los marcadores de vehículos sobre el mapa, el dispositivo se registrará a un servicio de mensajería (ActiveMQ haciendo uso del protocolo Stomp) para la actualización constante de la información de los vehículos que se hayan modificado, sin necesidad de consultar a cada vez bajo demanda, los posibles cambios que se hayan podido producir o no.

En el siguiente diagrama pueden apreciarse las partes que tienen lugar en el lado del servidor para el tratamiento y servicio de datos a enviar al cliente.

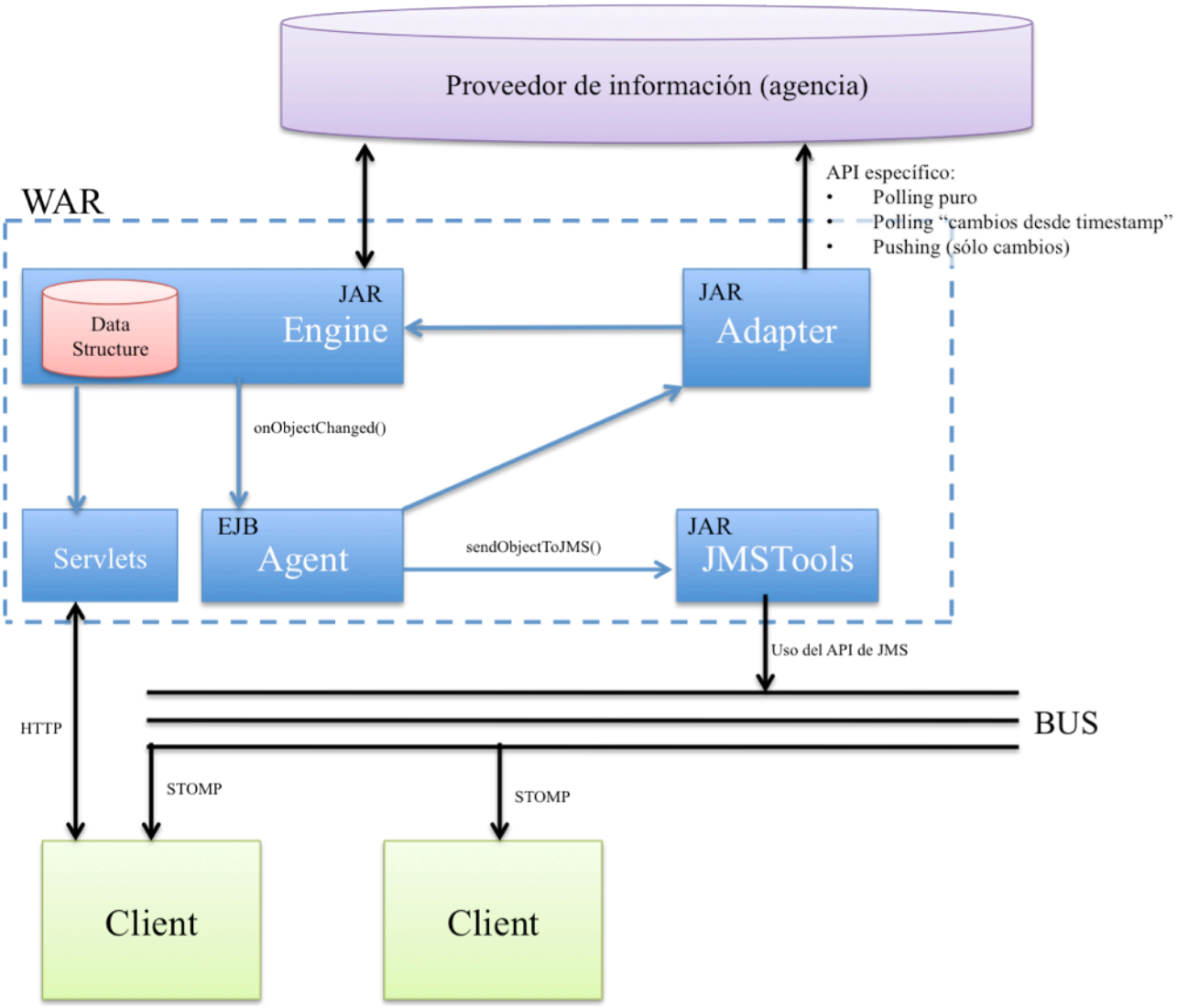

Ilustración 14 - Arquitectura del servidor

Proveedor de información (agencia)

Estos son los servicios externos que se consultan para obtener la información necesaria de los transportes públicos. En el caso de Helsinki, el principal proveedor de información son los servicios disponibles en el proyecto de HSL.

Para poder hacer uso de las funcionalidades de este API, se precisa un registro previo de usuario. El uso de este servicio y el registro al mismo es completamente gratuito y abierto.

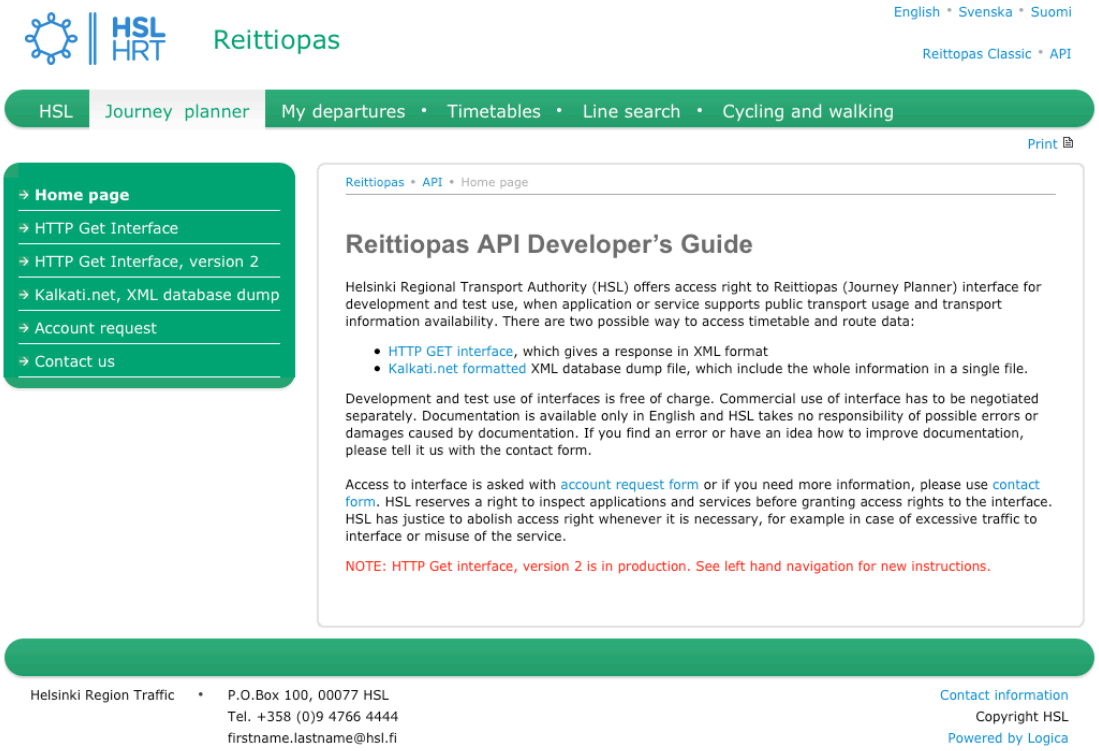

Ilustración 15 - Página principal del proyecto Reittiopas de HSL

Servicios expuestos y utilizados:

- Kalkati.net, XML database dump:

De este apartado podemos obtener toda la información estática relacionada con la base de datos del servicio de transporte público de Helsinki: líneas, rutas y tablas temporales. Esta información puede obtenerse en cualquier momento descargando el *Dump-file*. Este fichero es un XML que contiene, de forma estructurada, los datos que se utilizarán para la inicialización de la aplicación del cliente. Haciendo uso del esquema XSD (descargable desde la misma página), se procesará toda la información del XML en el servidor y se crearán los ficheros de configuración inicial adaptados a las necesidades del cliente.

Esta operación de procesado del XML y conversión a ficheros personalizados es una tarea costosa, pero teniendo en cuenta que la información que representa es completamente estática, y los cambios que pueda sufrir serán poco habituales (inserción de nuevas paradas, líneas…), se podrá establecer un periodo de regeneración de este recurso lo suficientemente largo como para no estresar al servidor.

- HTTP Get interface (version 2):

Desde aquí se realizarán consultas de carácter dinámico: Tiempos de llegada, información de paradas… De forma directa, el cliente móvil consultará el servicio de tiempos de llegada a una parada específica bajo demanda.

Ingresando un identificador de parada (y rellenando los campos de usuario y contraseña con los de nuestro registro), obtendremos como respuesta, un texto plano con la información relativa a dicha parada:

```
650|Eläintarha|Nordenskiöldinkatu|Helsinki
1156|3T|Katajanokan termin.|1003T 2
1208|3T|Katajanokan termin.|1003T 2
1220|3T|Katajanokan termin.|1003T 2
1232|3T|Katajanokan termin.|1003T 2
1244|3T|Katajanokan termin.|1003T 2
1256|3T|Katajanokan termin.|1003T 2
```

Ilustración 16 - Información dinámica de parada (texto plano)

Cuando el cliente consulte la información adicional de una parada, recurrirá directamente a este sencillo servicio. El dispositivo obtendrá esta información textual, la procesará y la mostrará en su interfaz.

Otro servicio consultado de forma directa desde el cliente es el de los avisos de incidencias de tráfico. Esta información se puede obtener de forma directa bajo demanda. Accediendo al servicio, se obtiene un XML con las notificaciones relativas al transporte público que se encuentran activas en el momento de realizar la consulta.

```
<DISRUPTIONS time="2011-06-25T14:01:11" valid="1" cancelled="0">
- <DISRUPTION id="9705" type="2" source="2">
    <VALIDITY status="1" from="2011-06-25T13:28:00" to="2011-06-25T14:21:00"/>
  - <INFO>
    - <TEXT lang="en">
        Regional traffic, line 106 from Helsinki, 13:21 cancelled. Cause: technical problems.
      </TEXT>
    </INFO>
  - <TARGETS>
      <LINE id="106" direction="1" linetype="5">106</LINE>
    </TARGETS>
  </DISRUPTION>
</DISRUPTIONS>
```

Ilustración 17 - Información dinámica de incidencias de tráfico (formato XML)

- API Push de Mattersoft:

Finalmente, el último servicio al que se realizarán consultas para el funcionamiento de nuestra aplicación, será el de obtención de información vehículos. Este servicio se basa en un API de Push que provee de los

datos enviados por los GPS de los transportes públicos de la ciudad de Helsinki.

Esta información será consultada de forma continua por nuestro servidor y transmitida al adaptador para procesar correctamente la información.

Adapter (JAR)

Aplicación de Java que realiza consultas de manera continua a los servicios del proveedor. Su principal tarea es recoger dicha información y tratarla para establecerle un formato estándar, debido a que cada proveedor de información utilizará un formato distinto. Este componente se encargará de adaptar los formatos de cada agencia al formato establecido para el proyecto y únicamente transmitirá los datos añadidos o actualizados desde la última consulta realizada.

Para el proyecto de la ciudad de Helsinki, el adaptador realiza consultas constantemente al API Push de Mattersoft de obtención de información vehículos.

Este servicio devuelve, en texto plano, la información de los vehículos registrados en base al filtrado establecido (área geográfica establecidas por latitud y longitud inicial y final), y cada consulta realizada al servicio devuelve información actualizada sobre los mismos:

```
RHKL00239;1009;24.937137;60.201705;256;1;1173431;1173433;1624
RHKL00075;1004;24.948538;60.169137;274;1;1020455;1020449;1640
RHKL00082;1003T;24.920308;60.1896;220;1;0;1080413;1722
RHKL00085;1004;24.872433;60.194498;107;2;0;1301456;1651
RHKL00237;1005;24.942342;60.17046;89;1;1030426;1090401;1530
EFENG1061000436;1011;24.968007;60.186137;35;1;1100102;1100105;1640
RHKL00072;1010;24.936477;60.171235;306;2;1070414;1070425;1609
RHKL00221;1006;24.974802;60.206993;293;2;1230407;1230406;1646
RHKL00220;1009;24.945475;60.194567;220;2;1220402;1220428;1642
RHKL00223;1003T;24.930755;60.168685;145;2;1130439;1040412;1632
RHKL00222;1006;24.94262;60.167133;146;2;1020464;1040445;1622
RHKL00224;1003T;24.970393;60.164347;65;1;1080413;1080415;1646
RHKL00227;1003T;24.921423;60.185612;326;1;1140436;1140438;1622
RHKL00226;1005;24.956308;60.161472;180;1;0;1090415;1650
```

Ilustración 18 - Información dinámica de vehículos registrados (texto plano)

En el ejemplo (Ilustración 18) podemos ver un listado de vehículos que sigue el siguiente formato:

Identificador_vehículo; línea; latitud; longitud; orientación; dirección; parada_anterior; parada_actual; salida; velocidad.

El adaptador leerá esta información, la procesará como “objetos vehículo” y enviará la información al ENGINE, quien se encargará de salvaguardar los nuevos vehículos registrados y los cambios producidos en los ya existentes.

Engine (JAR)

Este componente será la estructura de datos principal del servidor. En él se irán almacenando los datos obtenidos desde el adaptador y transmitiendo al agente los datos que se vayan modificando en la estructura.

Todos los mensajes recibidos desde el adaptador se incluirán en la memoria del servidor, componiendo así una base de datos actualizada en todo momento. De los "objetos vehículo" almacenados se podrá obtener la siguiente información:

```
    public enum Type { TRAM, SUBWAY, RAIL, BUS, FERRY,
CABLE_CAR, GONDOLA, FUNICULAR, TAXI }
    public enum Orientation { NORTH, NORTHEAST, EAST,
SOUTHEAST, SOUTH, SOUTHWEST, WEST, NORTHWEST, NONE }

    private String id;
    private float latitude;
    private float longitude;
    private String route;
    private Orientation orientation;
    private long lastupdate;
    private Type type;
```

Código fuente 1 - Atributos del modelo Vehicle.java

En el caso de que algún proveedor de información no disponga de la orientación del vehículo, ésta se calculará a partir de la geolocalización del vehículo, realizando la diferencia entre latitud/longitud previa y actual.

El componente Engine, almacenará toda la información que reciba y proporcionará interfaces y servicios de consulta y envío de eventos sobre su estructura para el cliente.

Servlets

Serán los servicios de consulta bajo demanda a los que recurrirá el cliente para la inicialización de la aplicación. Actualmente, para la aplicación de Helsinki, se hará uso de dos servlets: HelsinkiData y CurrentMarkers.

- **HelsinkiData.java:** Este servicio proporciona al cliente, en un fichero comprimido, un XML generado a partir de la unión de los XML de configuración inicial:
  - HelsinkiConfig.xml: Parámetros de configuración relacionados con la ciudad de Helsinki. Estos parámetros se establecen manualmente e indican la geolocalización de la ciudad, los tipos de transportes que va a mostrar, y algunos parámetros auxiliares relacionados con el nivel de zoom admisible en el cliente.
  - HelsinkiLines.xml: Archivo generado a raíz del procesado de información estática obtenida desde el Dumb-file de HSL. Este

fichero muestra la información de líneas de transporte: identificador, nombre, tipo de línea, coordenadas de su trayectoria…

- HelsinkiStops.xml: Archivo generado a raíz del procesado de información estática obtenida desde el Dumb-file de HSL. El fichero ofrece un listado de las paradas existentes en la ciudad indicando su geolocalización, las líneas que hacen uso de éstas, su nombre, identificador y tipo.

- **CurrentMarkers.java:** Este servicio proporciona la información de todos los vehículos registrados en el Engine en formato XML. Se utilizará para crear los marcadores de todos los vehículos en el cliente una vez terminada la configuración inicial.

Ambos servlets proporcionan la información necesaria para la inicialización del cliente y para acceder a ellos basta con acceder a las siguientes direcciones:

Servlet 1: http://www.mobitransit.com/helsinki/services/data.gz

Servlet 2: http://www.mobitransit.com/helsinki/services/markers.gz

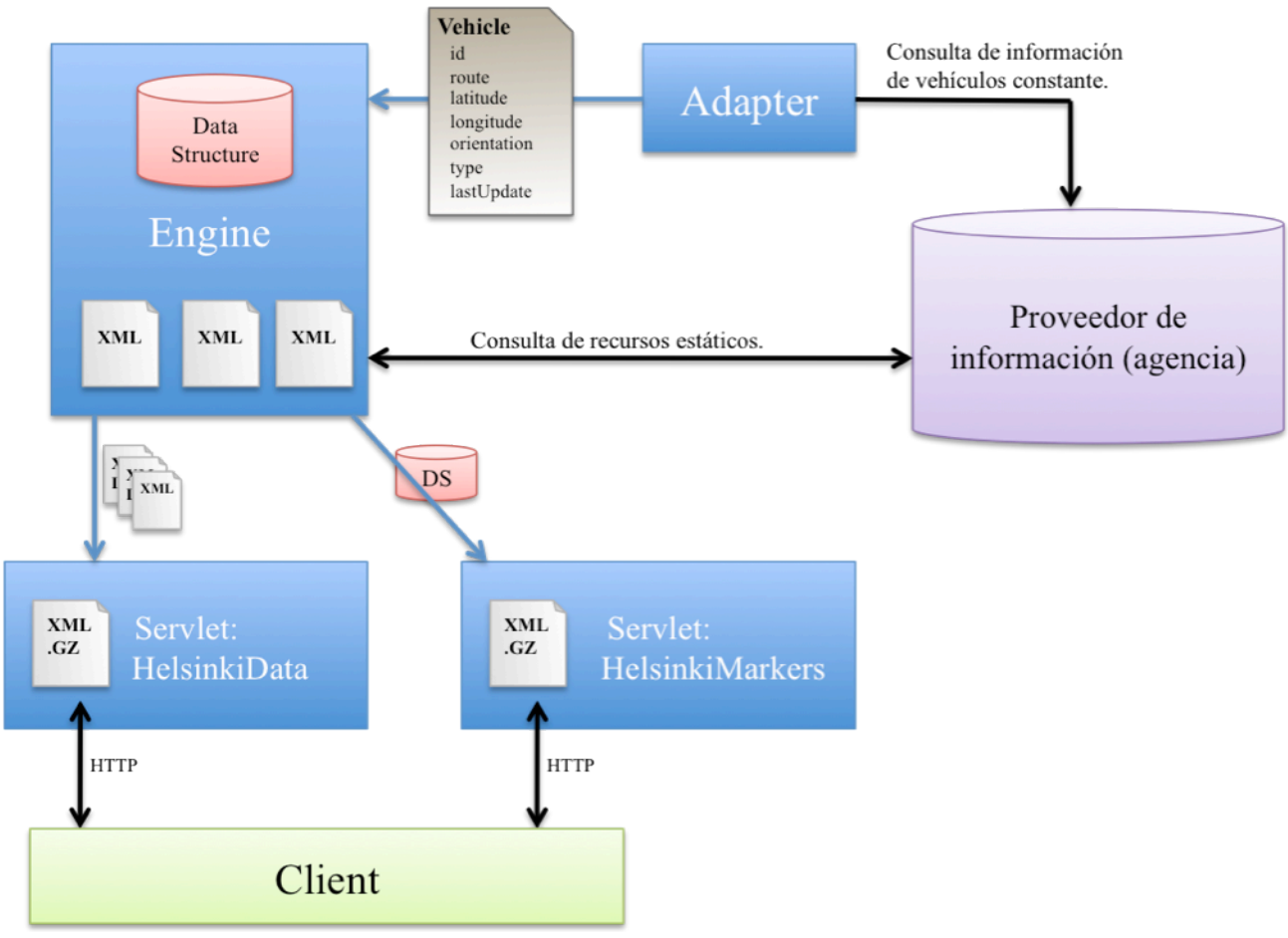

Ilustración 19 - Flujo de datos de información estática

Agent (EJB)

El agente ubicado en el servidor, se mantendrá escuchando la estructura de datos del Engine. En cuanto se produzca un cambio en alguno de los vehículos almacenados, transmitirá esta información al servicio de mensajería para enviarla al cliente.

JMSTools (JAR)

Las herramientas de JMS proporcionarán el servicio de tiempo real que necesitamos para transmitir la información desde el servidor hasta el cliente. Basándonos en ActiveMQ, se creará un componente que procesará los eventos recibidos desde el agente y los enviará a todos los dispositivos suscritos a un canal reservado para esta aplicación. El protocolo de envío al cliente será STOMP, y toda la información del evento se encontrará en la cabecera del mensaje como pares clave-valor.

Este es el último eslabón de la cadena del servidor en cuanto a servicios de información dinámica se refiere. Así pues, teniendo los apartados anteriores claros, podemos entender la actualización de información dinámica como el flujo de datos que muestra el siguiente diagrama:

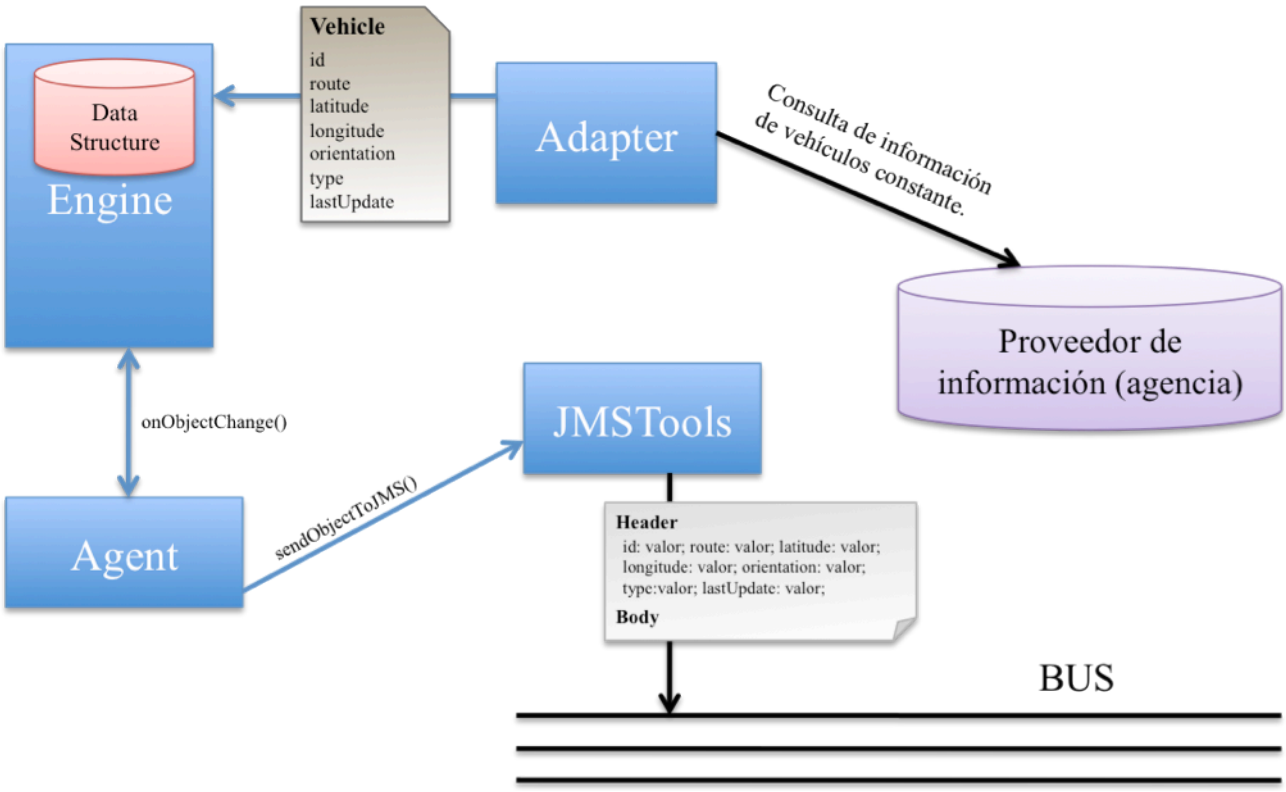

Ilustración 20 - Flujo de datos de información dinámica

Diseño e implementación de la aplicación móvil

En este apartado se describirá la arquitectura básica seguida para la implementación de la aplicación del cliente. Una manera práctica de representar y comprender la estructura que sigue la aplicación es haciendo uso del patrón MVC (Modelo-Vista-Controlador).

La arquitectura MVC separa los datos (Modelo), la interfaz de usuario (Vista) y la lógica de control (Controlador) en tres componentes distintos que se detallarán a continuación.

Modelo

Esta es la representación específica de la información con la cual opera el sistema. En el caso de este proyecto, el modelo se corresponderá con todos aquellos datos relativos al servicio de transportes de la ciudad de Helsinki: líneas, paradas, vehículos, rutas…

El siguiente diagrama de clases representa la información almacenada por la aplicación como modelo de datos durante su ejecución.

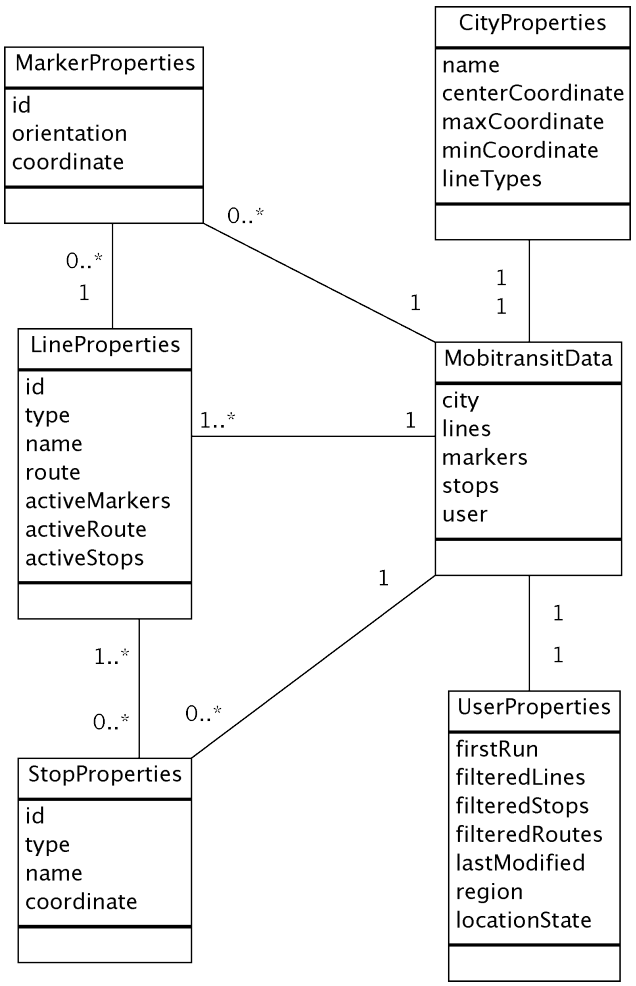

Ilustración 21 - Diagrama de clases del modelo de datos

El diagrama anterior (Ilustración 21) representa las siguientes clases:

- **MobitransitData:** Clase principal que almacenará de forma estructurada la información del programa. Posee los siguientes atributos:
    - city: Información relativa a la ciudad que se visualiza en el mapa.
    - lines: Vector que referencia a las líneas de transporte público existentes.
    - markers: Vector que referencia a los vehículos registrados en la aplicación.
    - stops: Vector que referencia a las paradas.
    - user: Información relativa a propiedades de usuario

- **CityProperties:** Clase que contiene las propiedades de la ciudad, su nombre, geolocalización y área que abarca en el mapa.
- **LineProperties:** Clase representante de las propiedades de una línea de transporte público. La clase presenta los siguientes atributos:
    - id: Identificador único de la línea
    - type: Tipo de vehículos de esta línea (tranvía, metro, autobús…)
    - name: Nombre de la línea.
    - route: Vector de coordenadas que representa el trayecto de una línea sobre el mapa.
    - activeMarkers: Booleano de filtrado de marcadores. Indica si deben mostrarse o no los vehículos de la línea.
    - activeRoute: Booleano de filtrado de trayectoria. Indica si debe mostrarse o no la trayectoria de la línea.
    - activeStops: Booleano de filtrado de paradas. Indica si deben mostrarse o no las paradas relacionadas con la línea.
- **MarkerProperties:** Clase representante de las propiedades de un vehículo. Todo vehículo debe tener una única línea asociada. La clase presenta los siguientes atributos:
    - id: Identificador único del vehículo.
    - orientation: Orientación actual del vehículo.
    - coordinate: Coordenadas actuales del vehículo (latitud/longitud).

- **StopProperties:** Clase representante de las propiedades de una parada. Toda parada debe tener asociada, al menos, una línea que haga uso de ella. La clase presenta los siguientes atributos:
    - id: Identificador único de parada.
    - type: Tipo de vehículos que utilizan la parada (tranvías, metros, autobuses…)
    - name: Nombre de la parada.
    - coordinate: Coordenadas geográficas de la parada.
- **UserProperties:** Clase que almacenará la información del usuario. La clase presenta los siguientes atributos:
    - firstRun: Booleano que indica si es la primera vez que se hace uso de la aplicación.
    - filteredLines: Vector que indica el estado del filtrado de vehículos en la aplicación.
    - filteredStops: Vector que indica el estado del filtrado de paradas en la aplicación.
    - filteredRoutes: Vector que indica el estado de filtrado de rutas en la aplicación.
    - lastModified: Fecha en la que se realizó la última descarga de datos.
    - region: coordenadas y nivel de zoom del área de mapa visualizada por el usuario.
    - locationState: Booleano que indica si el usuario tiene activada o no la función de geolocalización del dispositivo.

Prácticamente, toda la información de este modelo de datos será proporcionada al inicializar la aplicación bajo demanda al servidor (servlets de inicialización). Posteriormente, la información de los vehículos y de filtrado se actualizará en función de las acciones del usuario y del agente de comunicaciones.

Vista

La componente de vista presenta la información del modelo en un formato adecuado para la visualización e interacción por parte del usuario. Los elementos de la vista acaban formando lo que se denomina interfaz de usuario.

La interfaz de este proyecto varía en función del dispositivo. Existe una interfaz específica para iPhone y otra para iPad. Sin embargo, la vista principal de la aplicación, así como el listado de filtrado de líneas, paradas y

rutas presentan prácticamente el mismo aspecto y se detallarán en este apartado. El resto de vistas referentes a configuración, información y servicios adicionales, se describirán con detalle más adelante, en el apartado de “Interfaz y funcionalidad”.

Para describir adecuadamente las vistas y elementos de vista que componen la funcionalidad principal del proyecto, dividiremos este apartado en seis partes fundamentales: el mapa, las opciones de filtrado, los vehículos, las rutas, paradas y el usuario.

❖ El mapa

La vista principal de la aplicación, tanto en iPhone como en iPad, muestran inicialmente, un mapa de Google Maps en el dispositivo a tamaño completo.

Mediante la información del modelo referente a las propiedades de la ciudad, se establecerán como parámetros, la ubicación en la que debe centrarse el mapa en la primera ejecución, así como la región delimitada en éste, impidiendo que el usuario visualice zonas del mapa donde no se va a mostrar ninguna información sobre el transporte público.

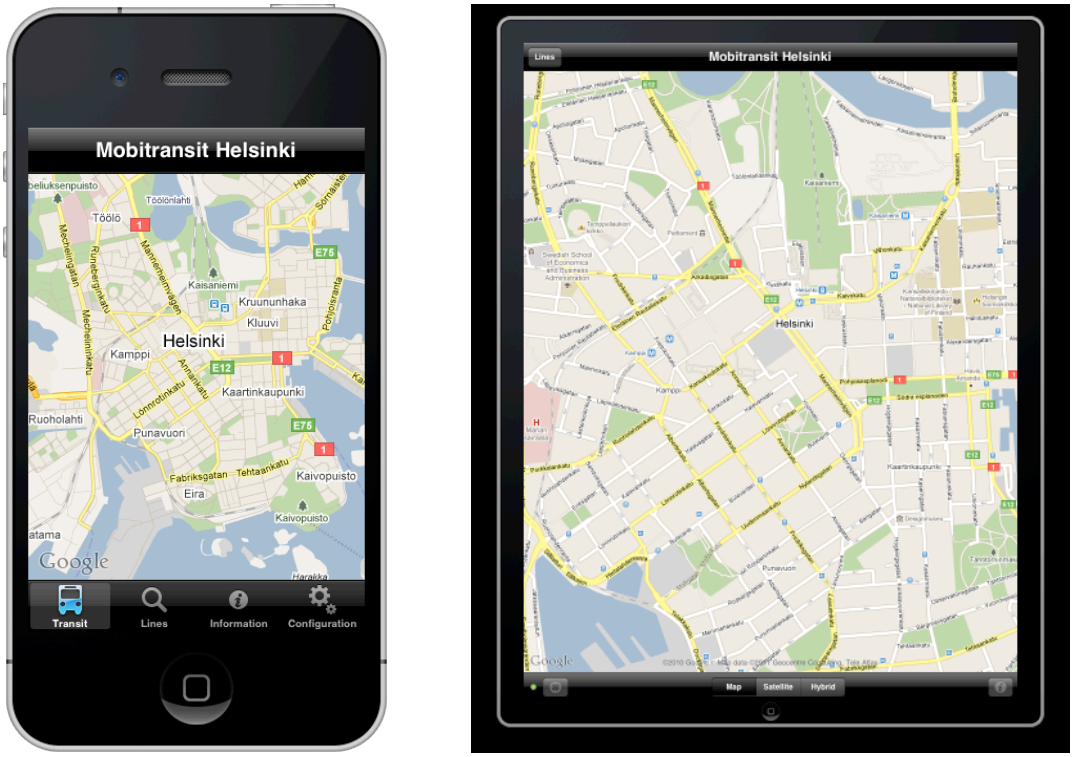

Ilustración 22 - Visualización de Google Maps en iPhone e iPad centrado en Helsinki

El API de Google Maps permite realizar opciones de desplazamiento y zoom sobre el mapa realizando distintos gestos sobre la pantalla del dispositivo. Esto hace que la vista del mapa sea completamente dinámica, dando así al usuario final, la posibilidad de visualizar el área geográfica que desee. También permite tres modos de visualización: Mapa normal, satélite e híbrido.

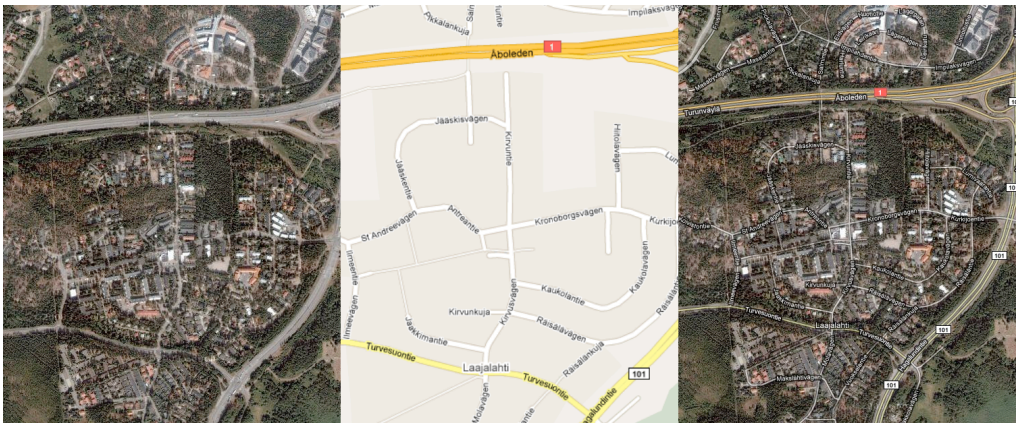

Ilustración 23 - Modos de visualización de Google Maps: Satélite, Normal e Híbrido

Sobre este mapa se visualizará el resto de información existente en el modelo de datos en función de las opciones de filtrado establecidas por el usuario, es decir, las rutas, los vehículos y las paradas.

❖ Las opciones de filtrado

Las opciones de filtrado no son más que un listado que muestra las líneas existentes en la ciudad (información obtenida desde el modelo de datos). Este listado se divide en secciones que representan los tipos de transporte (autobuses, metros, tranvías…) y cada sección contiene las líneas existentes para esos transportes.

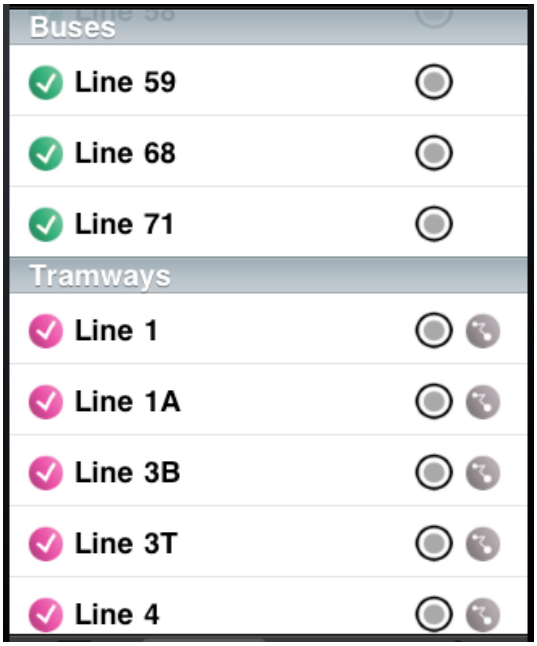

Ilustración 24 - Listado de líneas y opciones de filtrado

Cada celda del listado corresponde a una línea y posee tres opciones de filtrado distintas. Por cada línea, el usuario puede decidir si desea o no visualizar los vehículos pertenecientes a esa línea, la ruta que recorre dicha línea y/o las paradas relacionadas con la línea.

El filtrado de vehículos de una determinada línea viene representado por el primer icono de la celda con forma de “check”. Si este icono es de color gris, querrá decir que la visualización de vehículos para esta línea está desactivada. Si el icono se muestra de otro color, significará que la visualización de vehículos sobre el mapa estará activa.

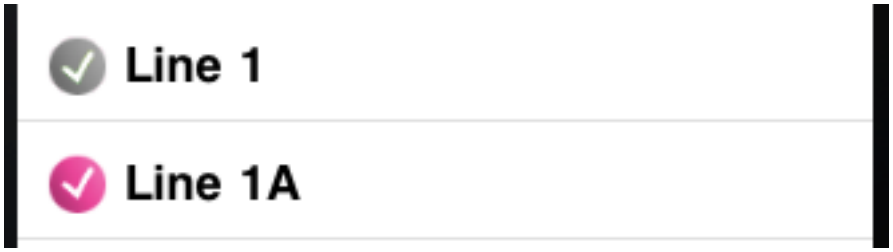

Ilustración 25 - Filtrado de vehículos: “Line 1” tiene inhabilitada la visualización de vehículos, mientras que “Line 1A" la tiene habilitada

El filtrado de paradas de una línea, viene representada por el segundo icono de la celda con forma de diana. Si este icono es de color gris, las paradas de dicha línea no se visualizarán en el mapa, de lo contrario, si que se mostrarán.

Ilustración 26 - Filtrado de paradas: “Line 1” tiene inhabilitada la visualización de paradas, mientras que “Line 1A" la tiene habilitada

Y finalmente, el filtrado de rutas de una línea viene representado por el último icono de la celda y sigue la misma lógica. Si la visualización de la ruta esta inactiva, el icono se mostrará de color gris. Cualquier otro color indicará que la ruta de la línea se estará mostrando en el mapa.

Ilustración 27 - Filtrado de rutas: “Line 1” tiene inhabilitada la visualización de su ruta, mientras que “Line 1A" la tiene habilitada

❖ Los vehículos

La representación de vehículos pertenecientes al servicio de transporte público, se realizará mediante marcadores dinámicos ubicados en el mapa de la vista principal. Estos marcadores identifican, mediante su color y su imagen decorativa, el tipo de vehículo al que pertenecen. Además, incorporan un pequeño elemento triangular que indica en todo momento su orientación actual.

Ilustración 28 - Marcadores de vehículos: autobuses y tranvías con distintas orientaciones

Además, dado que en el modelo de datos se establece que, todo vehículo debe ir asociado a una línea, el marcador incorporará además un indicador de la línea a la que pertenece en todo momento.

Ilustración 29 - Marcadores de vehículos: autobús perteneciente a la línea 51 y tranvía de la línea 1A

Los vehículos se mostrarán sobre el mapa haciendo uso de las coordenadas registradas en el modelo de datos, y ubicarán su flecha de orientación en función de su valor de orientación.

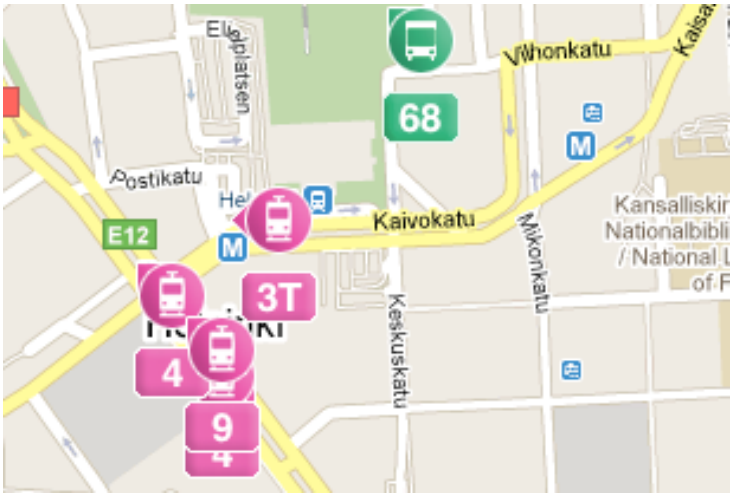

Ilustración 30 - Visualización de marcadores sobre el mapa

Finalmente, existirá un diálogo asociado al marcador que mostrará información adicional sobre el vehículo a petición del usuario. En el caso de Helsinki, se ha añadido la matrícula y el nombre completo de la línea.

Ilustración 31 - Diálogo de información adicional

❖ Las rutas

Las rutas indican las trayectorias que deben seguir los vehículos de cada línea por la ciudad. Estas trayectorias se representan sobre el mapa como líneas virtuales semitransparentes de distintos colores que subrayan una sucesión de calles.

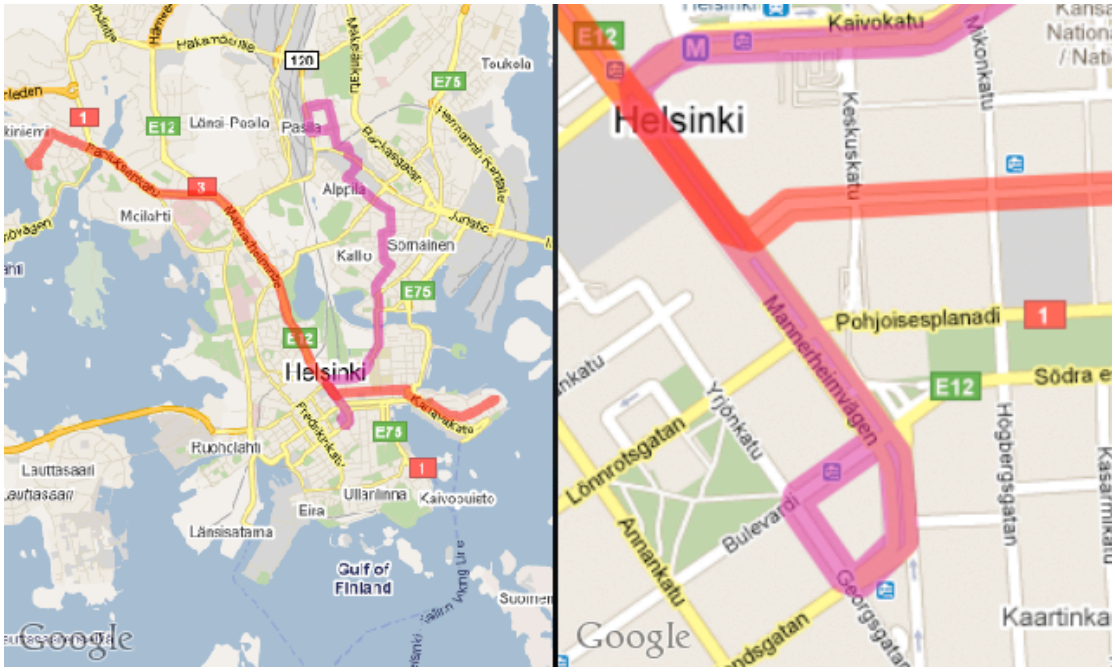

Ilustración 32 - Visualización de rutas de las líneas 4 y 9 a distintos niveles de zoom

❖ Las paradas

Las paradas de cada línea se mostrarán sobre el mapa según su atributo "coordinate" del modelo de datos. El icono que representa una parada se identifica como una diana con un círculo concéntrico de color. Este color indica el tipo de vehículos que hace uso de esta parada.

Ilustración 33 - Identificadores de parada

Al igual que los marcadores de vehículo, las paradas incorporarán un diálogo de información adicional. En el caso de Helsinki, se muestran, el nombre de la parada y las líneas que hacen uso de esa parada.

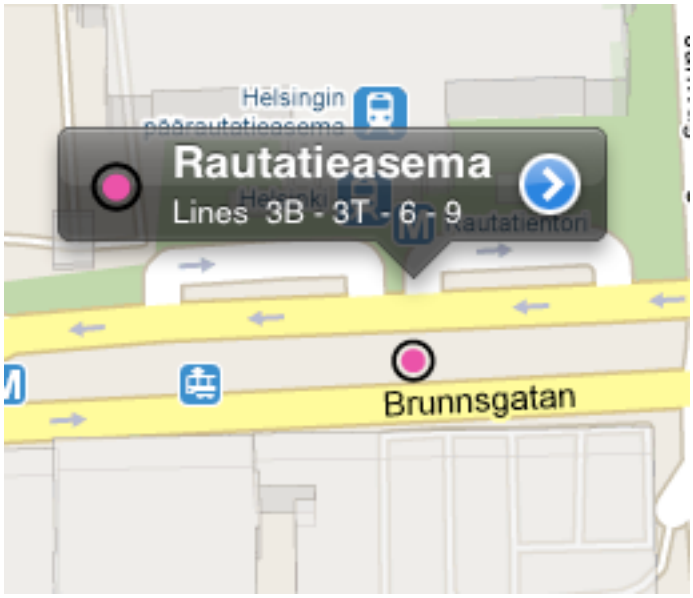

Ilustración 34 - Diálogo de información adicional de parada

Además, al cuadro de diálogo se añadirá un botón (botón azul situado a la derecha del diálogo en la ilustración 34) para consultar los tiempos de llegada de los vehículos a la parada. Los tiempos de parada se mostrarán en un listado ofreciendo información sobre la hora de llegada, el número de línea correspondiente y el nombre del destino del vehículo.

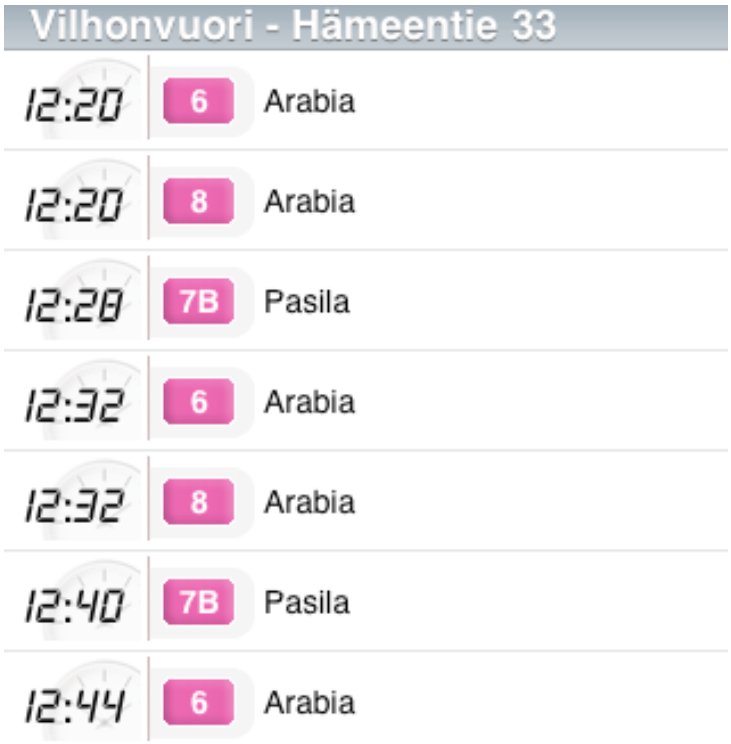

Ilustración 35 - Tiempos de llegada a una parada

- El usuario

La aplicación podrá también visualizar un marcador que represente la posición del dispositivo en el mapa. Así el usuario podrá saber dónde se encuentra en todo momento. Este marcador será el predefinido en el API de Google Maps, y pulsando sobre él, se mostrará un diálogo adicional en el que se podrá leer: “Ubicación actual”

Ilustración 36 - Geolocalización de usuario

Debido a las restricciones de área establecidas en la aplicación, la geolocalización del usuario solo podrá activarse si el dispositivo se encuentra dentro de los límites establecidos para la ciudad.

Controlador

El controlador será el componente encargado de gestionar los eventos producidos por el usuario, establecer las conexiones con el servidor y de poblar y mantener actualizado el modelo de datos de la aplicación. Podemos dividir las tareas del controlador en tres apartados: Inicialización de la aplicación, actualización dinámica de datos y gestión de eventos del usuario.

❖ Inicialización de la aplicación

Al iniciar la aplicación, el controlador deberá realizar dos operaciones fundamentales para la correcta carga y visualización de la misma.

En primer lugar, deberá consultar del servidor el fichero de configuración inicial, es decir, la información estática de la aplicación, accediendo al servlet HelsinkiData. De este fichero obtendrá los datos de la ciudad, sus líneas, paradas y rutas, y almacenará toda esta información en el modelo de datos.

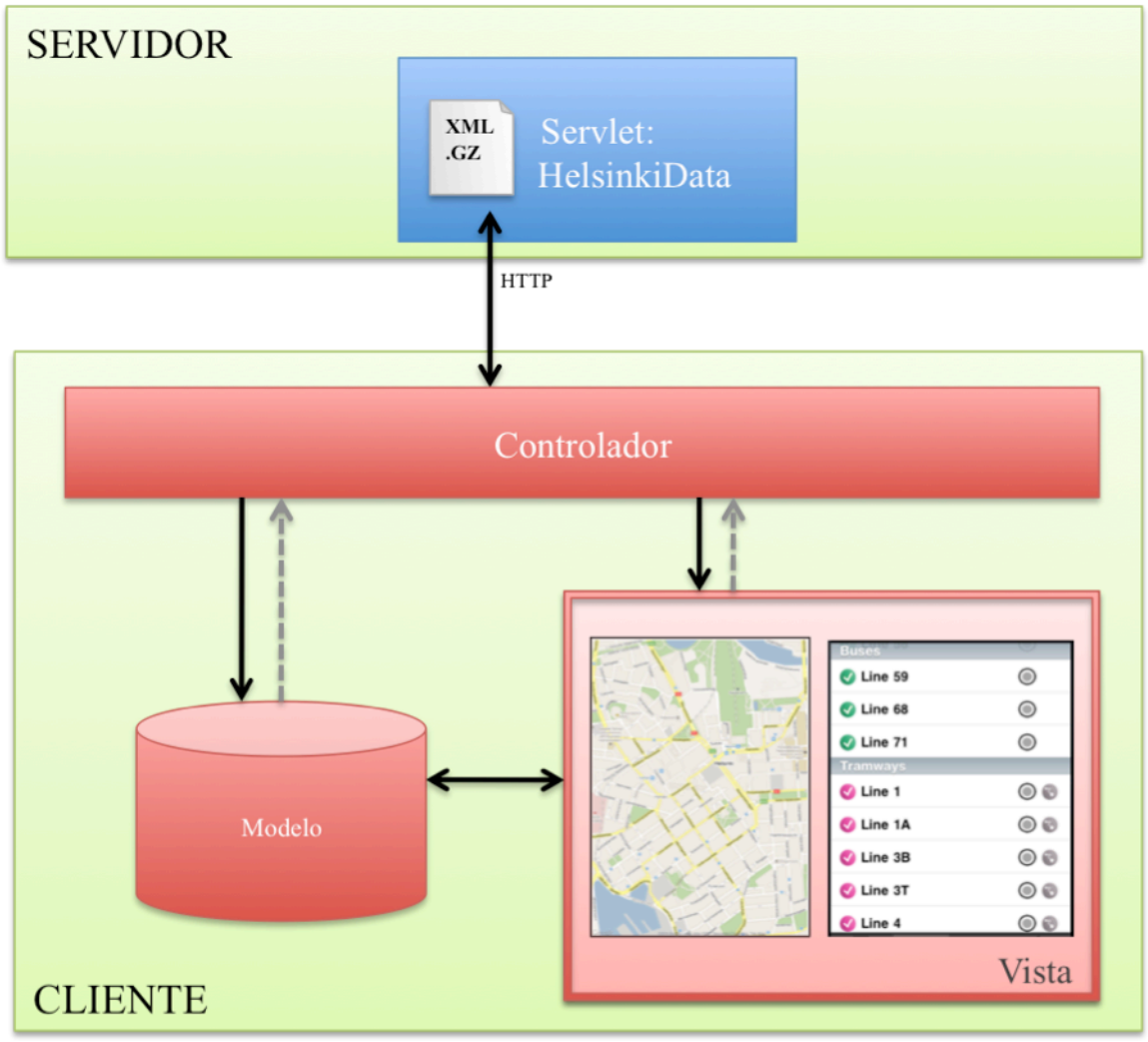

Ilustración 37 - Inicialización de información estática

Si la aplicación se está ejecutando por primera vez, el controlador siempre se descargará el fichero completo y, posteriormente, realizará una copia de éste en el dispositivo, memorizando además la fecha de descarga en las propiedades de usuario.

La descarga de este fichero se realizará en función de si se han producido cambios en él o no desde la última descarga. En caso de que no se hayan modificado los ficheros de configuración desde entonces, el controlador no se descargará el fichero y cargará todos los datos de configuración con el fichero que ya tiene almacenado en memoria. En caso contrario, el controlador obtendrá del servidor un fichero con las últimas modificaciones. Descargará el fichero, actualizará el modelo de datos y lo almacenará conjuntamente con su fecha de descarga.

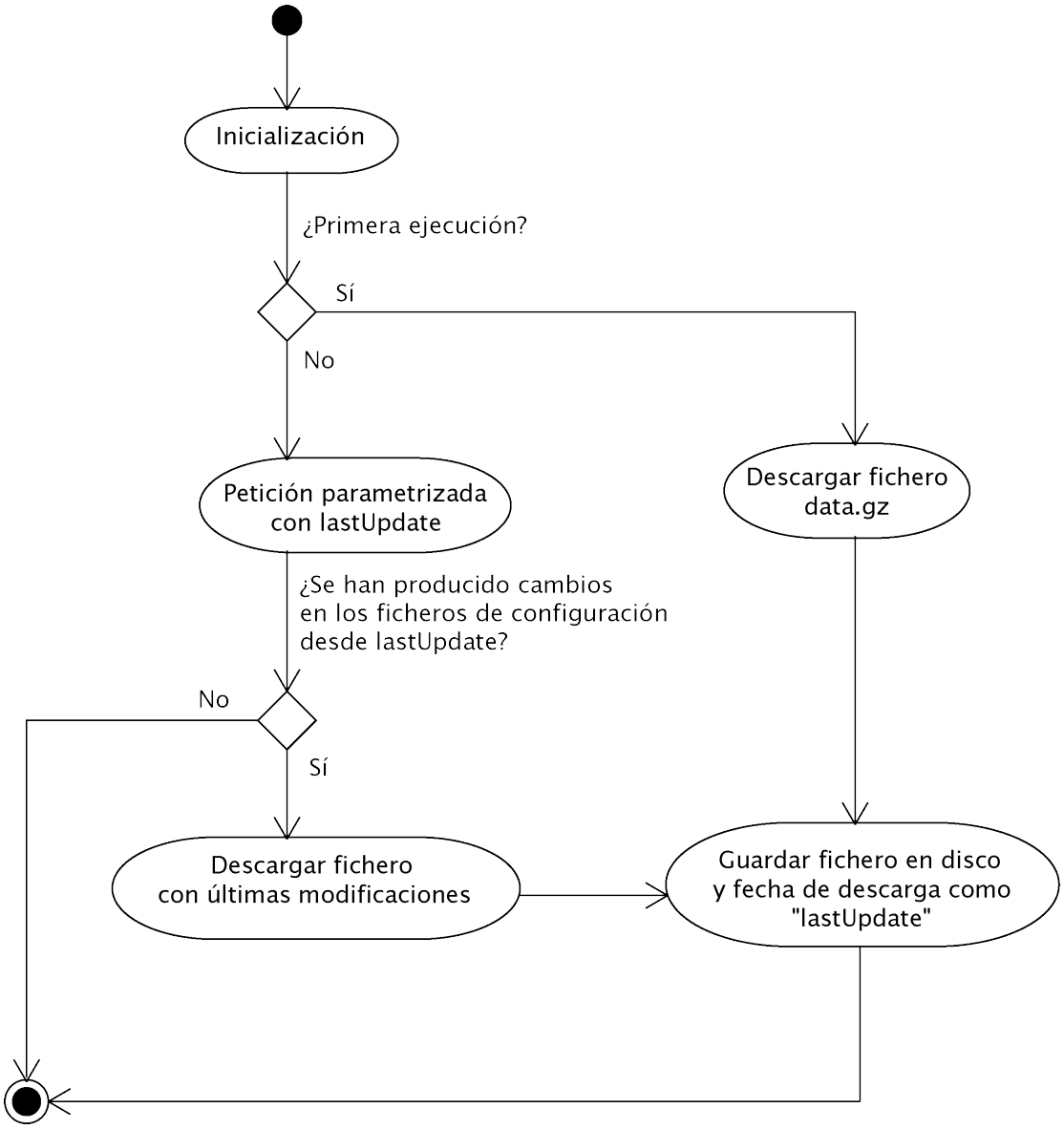

Ilustración 38 - Diagrama de actividad sobre la descarga de información estática

Con la información de este fichero, se poblará el modelo de datos con las propiedades de la ciudad, las líneas que manejará la aplicación, así como sus rutas y paradas. Una vez establecido el modelo, el controlador preparará la vista de mapa con los parámetros de la ciudad y el listado de líneas con el resto de información.

Finalmente, se consultarán las propiedades de usuario de la última vez que utilizó la aplicación para asignar a la vista de mapa la ubicación que consultó por última vez y, en el listado de líneas, activar/desactivar las opciones de filtrado que utilizó en dicha ocasión. Estas propiedades de usuario se encuentran almacenadas en el dispositivo y se actualizan cada vez que el usuario sale de la aplicación.

Habiendo cargado con éxito la información estática relativa a la ciudad y las líneas, el controlador deberá solicitar la información de todos los vehículos. Para ello, deberá establecer una nueva conexión con el servidor, solicitando el servicio del servlet HelsinkiMarkers.

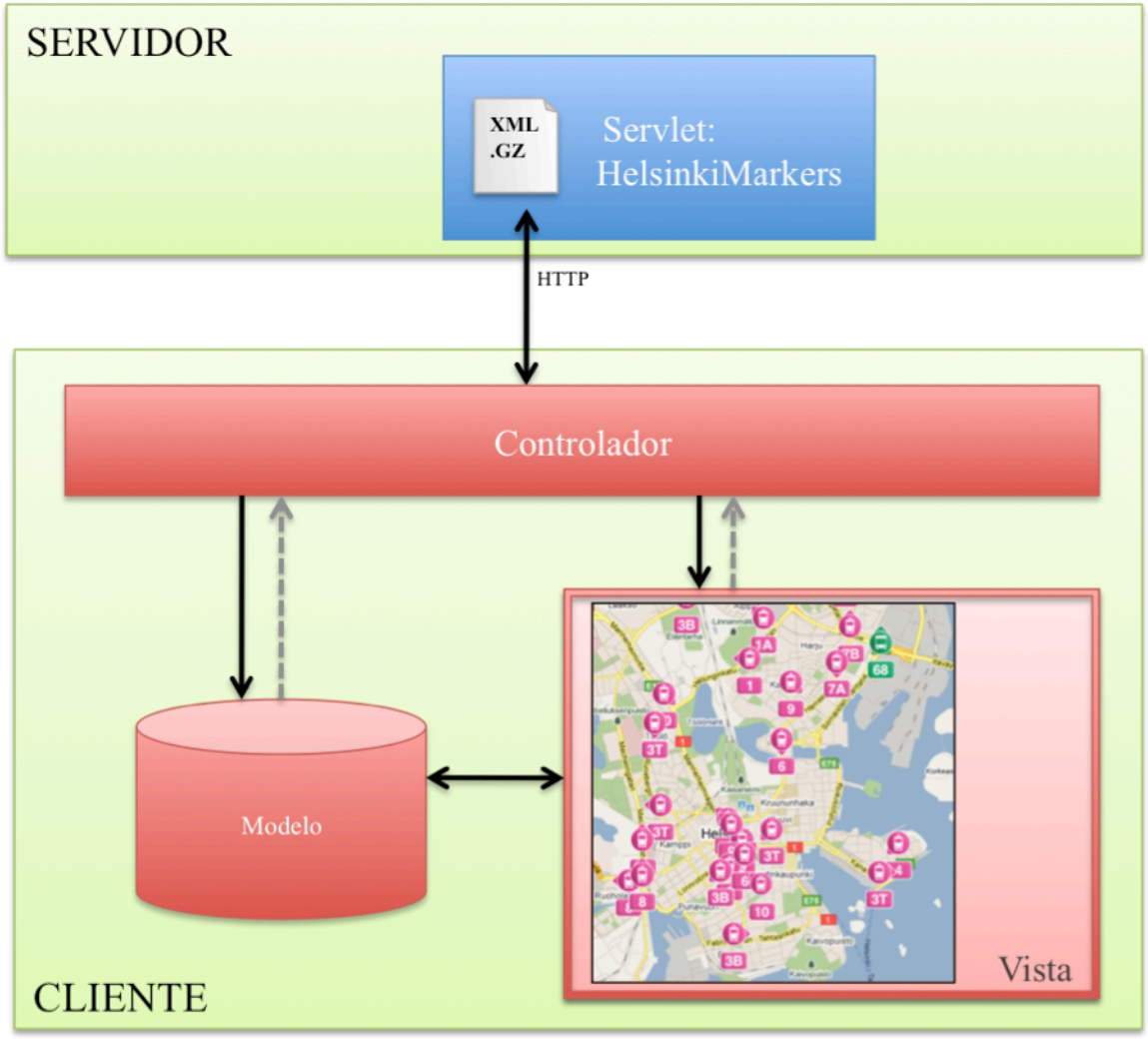

Ilustración 39 - Inicialización de vehículos

Mediante este servicio, el controlador obtendrá un XML comprimido con toda la información de todos los vehículos registrados. Descomprimirá el fichero, utilizará los datos que posee para terminar de poblar el modelo de datos con la información de vehículos que faltaba, y creará un marcador para cada uno de ellos y los añadirá a la vista de mapa.

El controlador deberá tener presente las relaciones existentes entre vehículos y líneas, por lo que también deberá ocultar aquellos marcadores cuyas líneas estén desactivadas en las opciones de filtrado.

❖ Actualización dinámica de datos

La aplicación ya tiene su modelo de datos poblado y su componentes de vista inicializados. La siguiente tarea del controlador, será suscribirse al servicio de mensajería del servidor para que éste pueda enviarle los mensajes de actualización de vehículos, que es precisamente lo que necesitamos para representar el movimiento de los vehículos a través de la vista de mapa.

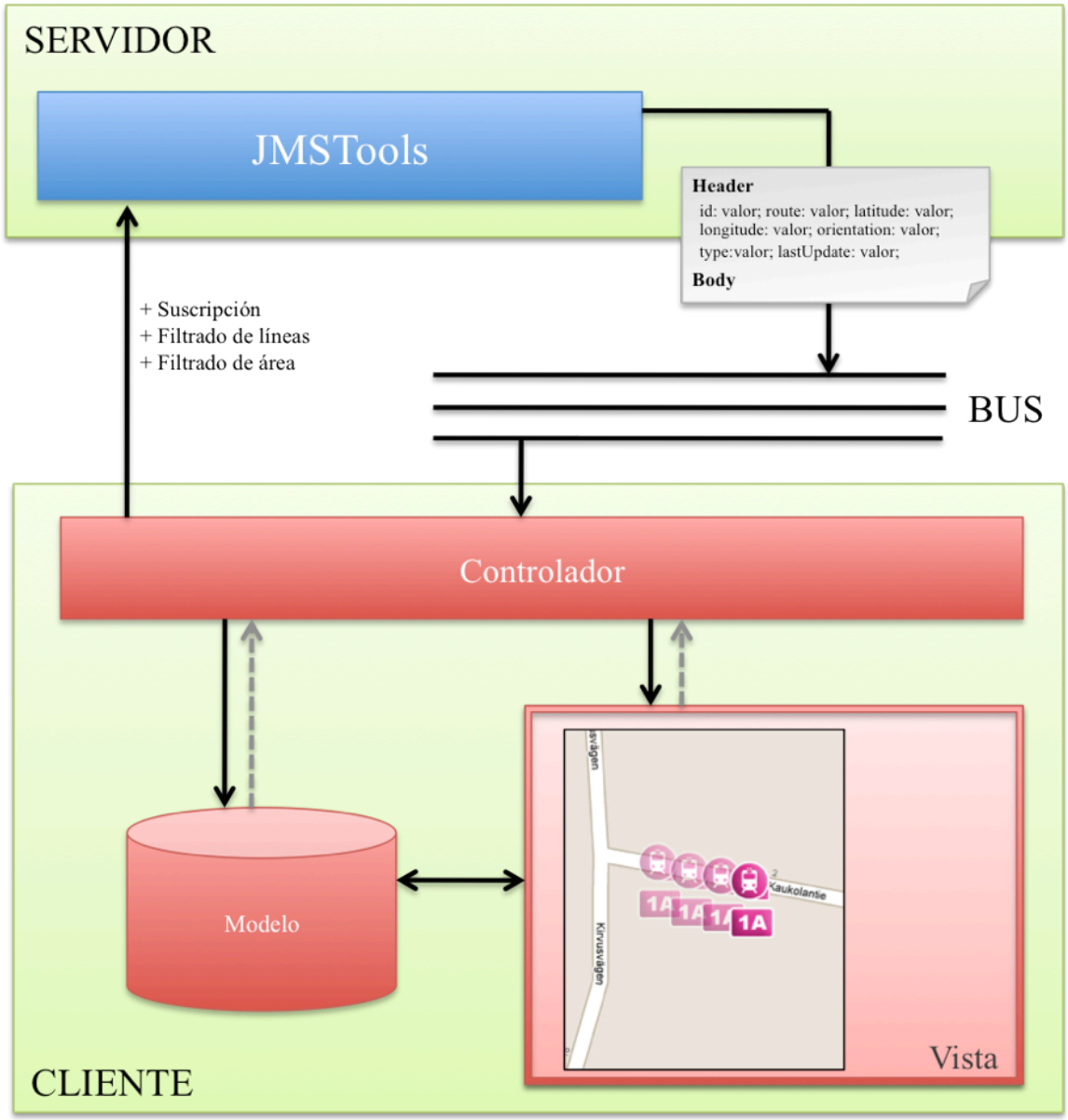

Ilustración 40 - Proceso de suscripción, obtención de eventos y actualización

Inicialmente, el controlador deberá suscribirse a un "topic" del servicio de ActiveMQ alojado en nuestro servidor. Esta suscripción permite, además de obtener los eventos gestionados por el agente, filtrar la llegada de dichos eventos en base a una serie de parámetros. Estos parámetros son:

- Área geográfica: Enviar únicamente aquellos eventos de objetos cuya latitud/longitud estén dentro de un área establecida.
- Líneas activas: Enviar únicamente aquellos eventos de objetos que pertenezcan a unas líneas establecidas.

El controlador, al realizar la suscripción, enviará como parámetros, el área del mapa que se está visualizando en el dispositivo (mediante cuatro valores: latitud máxima, latitud mínima, longitud máxima y longitud mínima), así como el filtrado de líneas que tiene establecido en su modelo de datos (el vector "activeMarkers"). Así, el servidor no enviará eventos de vehículos que estén fuera de la pantalla del usuario o de líneas que él mismo haya desactivado.
Con este filtrado, se minimiza el envío de eventos al cliente, haciendo más ágil la aplicación y estresando menos al servidor.

Una vez suscrito al canal y establecido el filtrado adecuado, el controlador comenzará a recibir eventos de actualización vehículos del servidor. Su tarea consistirá en tratar debidamente esta información, y actualizar los datos de dicho vehículo tanto en el modelo de datos como en la vista de mapa. Los eventos de actualización se irán sucediendo constantemente, y el controlador repetirá continuamente esta operación, manteniendo un modelo de datos siempre actualizado y mostrando en la vista de mapa el movimiento de los vehículos.

❖ Gestión de eventos del usuario

La tercera tarea del controlador se mantendrá presente durante toda la ejecución de la aplicación. Esta tarea basa en gestionar los eventos que pueda crear el usuario (hacer click sobre un botón, desplazamiento del área geográfica, modificaciones de filtrado…) y transmitir al resto de componentes las funciones a realizar en cada caso.

Existen varias situaciones provocadas por el usuario a tener presente:

Modificar los parámetros de filtrado de mensajes:

Ante un desplazamiento en la vista de mapa, realizar un zoom o un cambio en las vista de opciones de filtrado, el controlador deberá advertir al servicio de comunicaciones que los parámetros de filtrado de mensajes se ha modificado para que puedan llegar los mensajes correspondientes a la nueva zona geográfica visualizada o a las nuevas líneas filtradas. Una vez actualizados estos parámetros, el servidor continuará enviando eventos, pero conformes a la última acción establecida por el usuario.

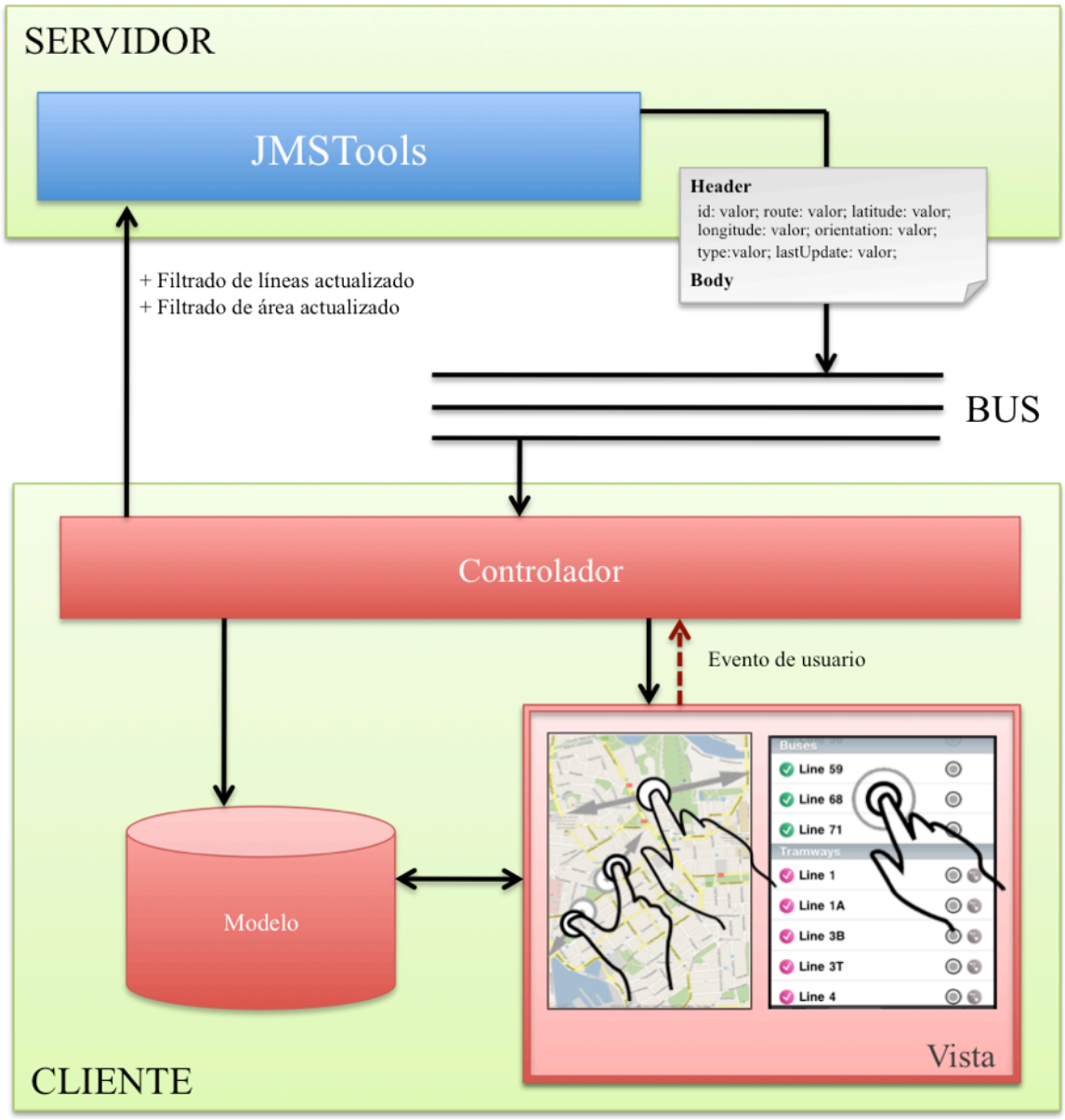

Ilustración 41 -Actualización de filtrado de mensajes por una acción del usuario

Solicitud de tiempos de llegada a una parada:

Cuando un usuario quiera visualizar los tiempos de llegada relativos a una parada específica (pulsando el botón azul ubicado en el diálogo emergente de los marcadores de parada), el controlador deberá solicitar esta información al proveedor de información. Haciendo uso del identificador de parada, podemos obtener los tiempos de llegada de la misma en base a la hora actual como un texto plano. El controlador deberá tratar esta información adecuadamente y mostrar en la vista de detalle de tiempos de llegada los datos que obtenga.

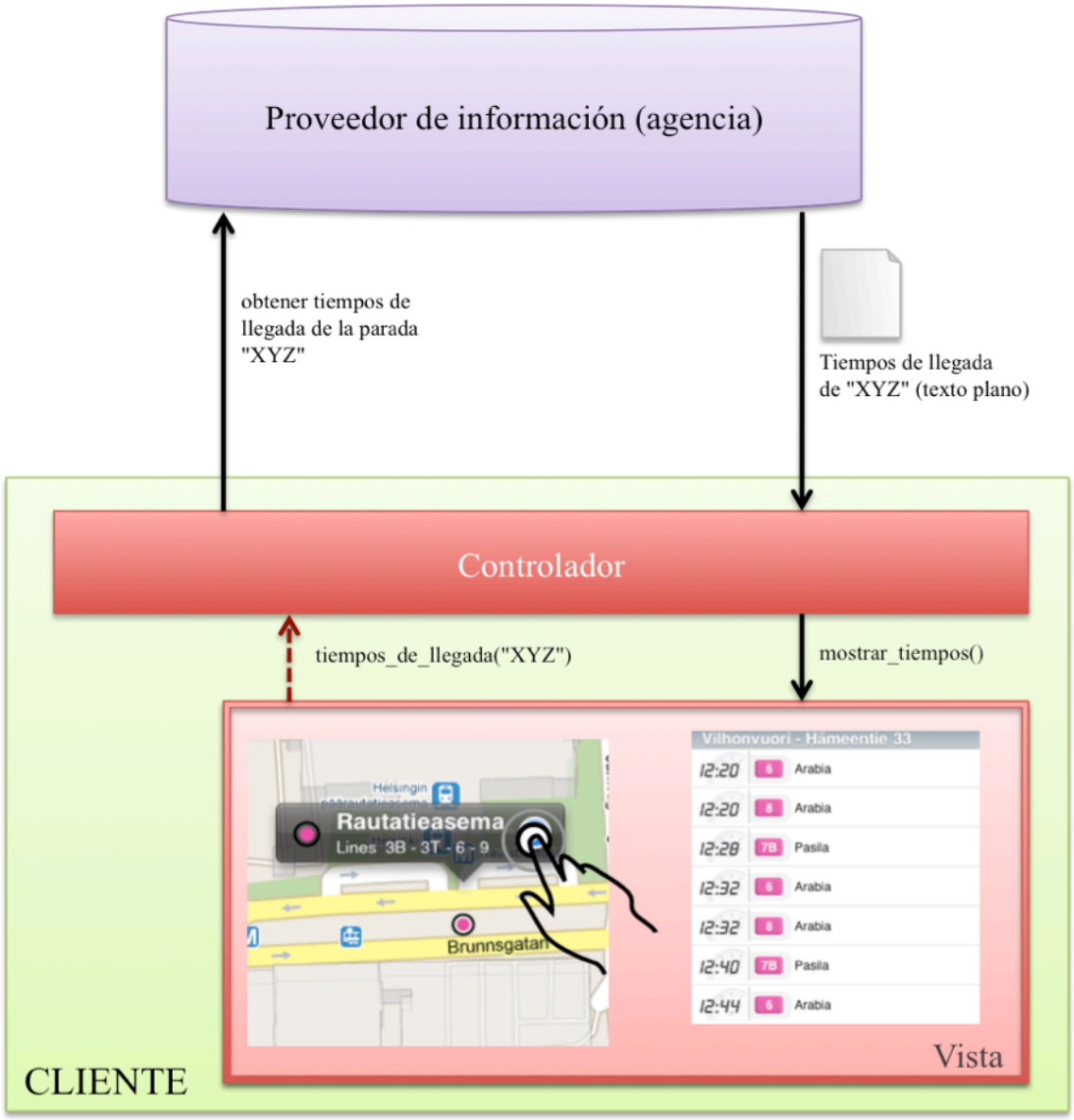

Ilustración 42 - Obtención de tiempos de llegada de una parada

Opciones de filtrado visual:

El usuario puede, desde el listado de líneas, ocultar y mostrar en todo momento las líneas, paradas relacionadas con una línea y rutas de línea que le plazca. Ante estas acciones, el controlador deberá advertir a la vista de mapa qué elementos deben mostrarse o no en función de dicho filtrado.

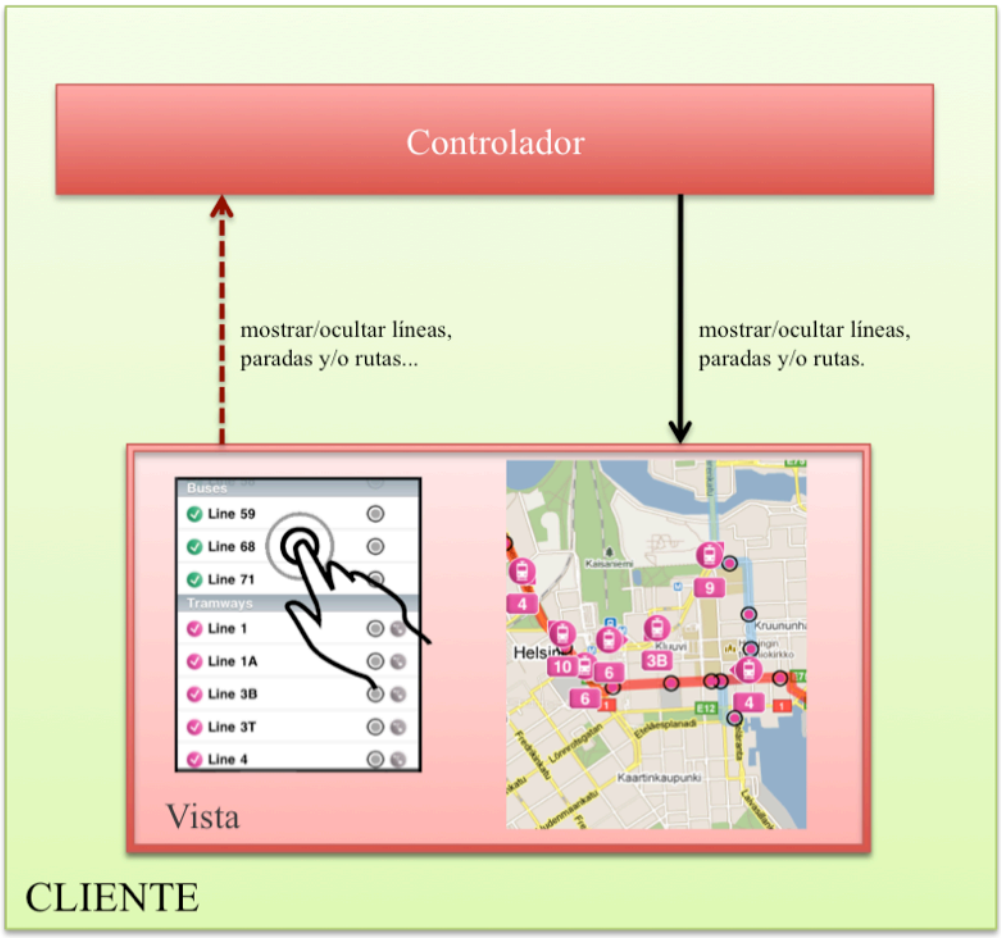

Ilustración 43 - Mostrando/Ocultando líneas, paradas y/o rutas

Restricciones de área establecida:

Al iniciar la aplicación, entre los parámetros de configuración de la ciudad se encuentran las coordenadas que delimitan su área geográfica. La vista de mapa mostrará únicamente las zonas contenidas en este área y añadirá una capa negra que ocultará las regiones que estén más allá de estas fronteras.

El controlador deberá tener presente esta restricción, y reubicar el área de visualización del mapa si el usuario realiza un desplazamiento hacia las afueras de estos límites o un zoom excesivamente alejado que dé, como resultado, un área de visualización de mapa demasiado pequeña en el que predomina la visualización de la capa negra.

Con ello se consigue que el punto de interés de la aplicación se centre en las zonas donde se podrá visualizar información del transporte público y el usuario no se aleje a zonas donde no obtendrá ninguna información adicional que no pueda obtener en cualquier otra aplicación de mapas.

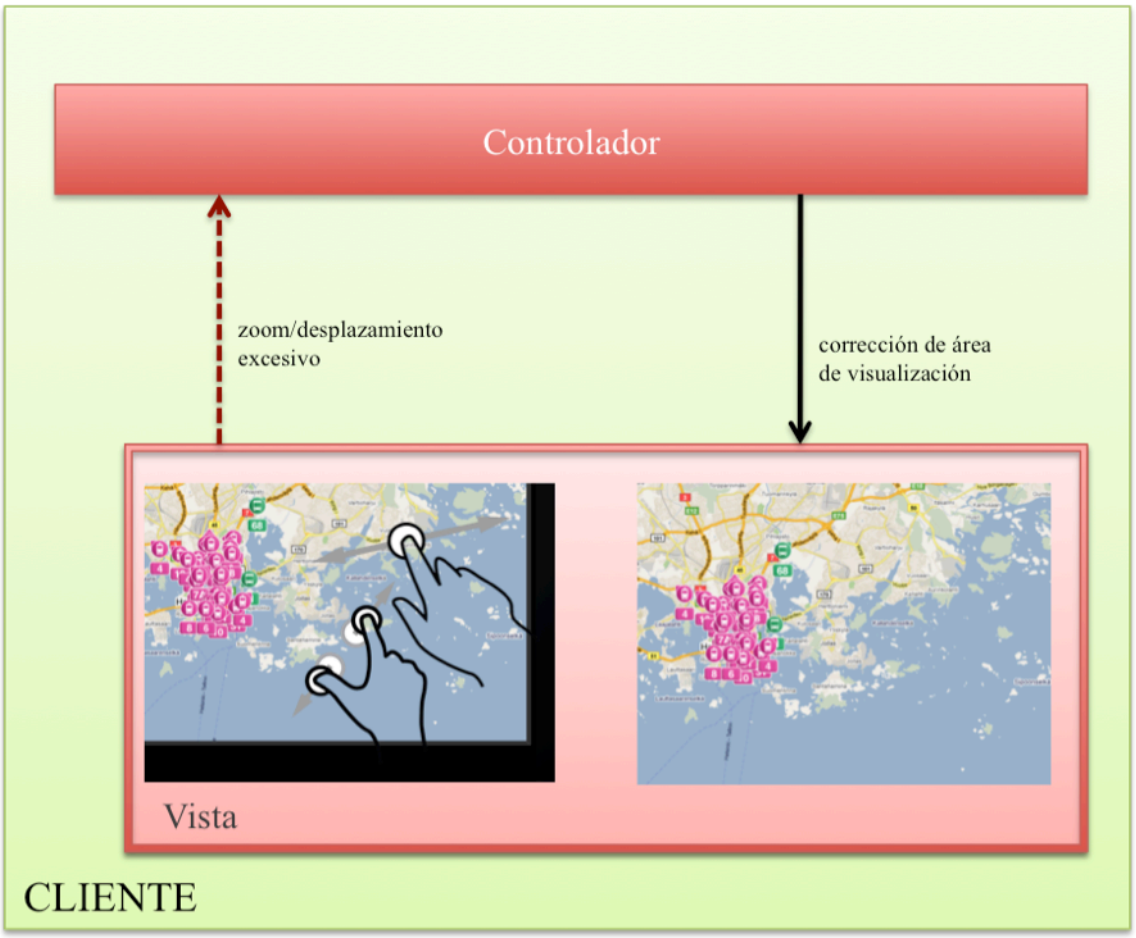

Ilustración 44 - Restricción de área geográfica

Cambio de tipo de mapa:

El usuario podrá establecer el tipo de mapa que desea visualizar pudiendo elegir entre la vista normal, de satélite o híbrida.

Estos modos de visualización podrán establecerse mediante unos botones ubicados en un lugar determinado de la aplicación (existe un lugar específico para la aplicación de iPhone y otro distinto para la de iPad que se detallará en el apartado de “Interfaz y funcionalidad”). El controlador se mantendrá a la escucha de los eventos producidos por estos botones y cambiará el tipo de mapa consecuentemente.

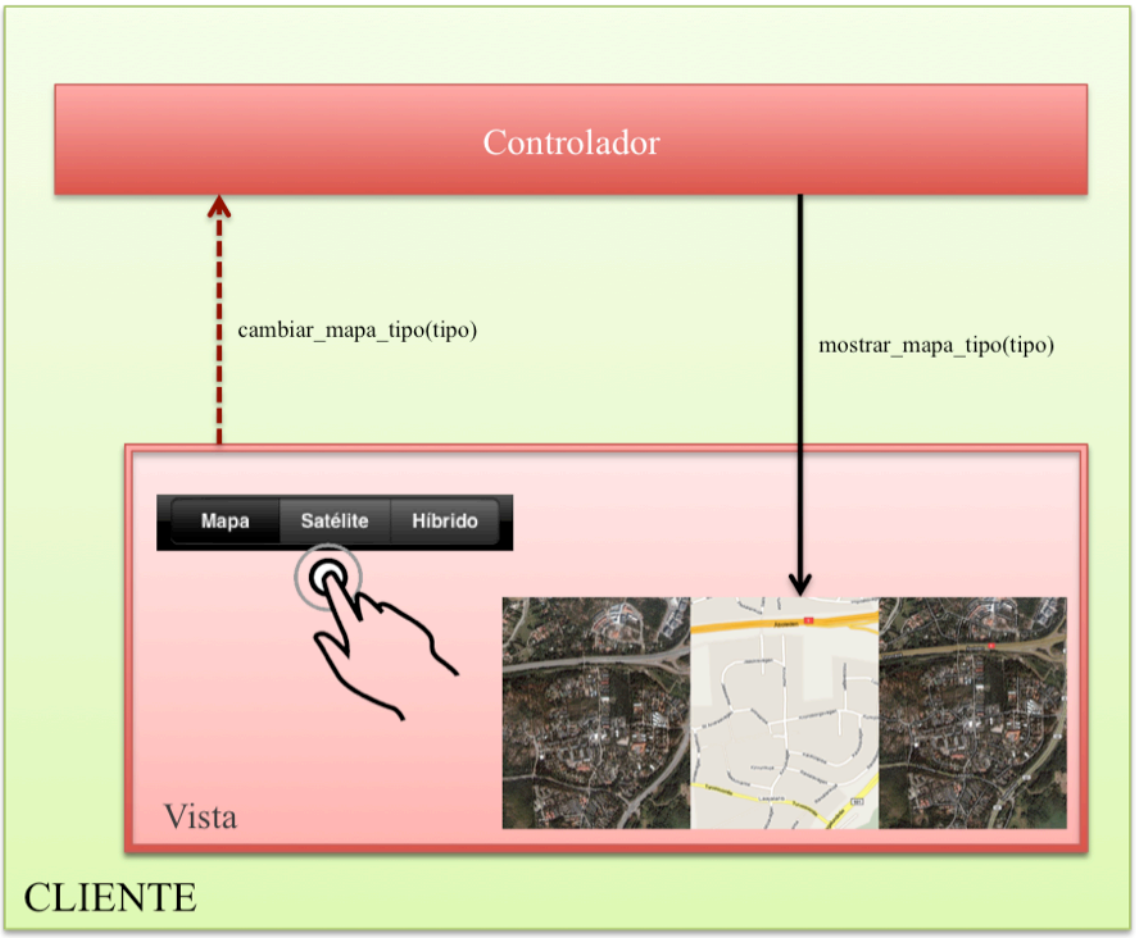

Ilustración 45 - Cambiando el tipo de mapa

Seguimiento de vehículo:

Pulsando sobre cualquier vehículo representado en el mapa, además de visualizarse el diálogo de información adicional sobre éste, el controlador iniciará la funcionalidad de seguimiento de vehículo.

Esta funcionalidad se basa en centrar en el mapa el vehículo que ha sido seleccionado, incluso cuando se produzca un cambio de posición por su parte. Así conforme se mueva el vehículo, el mapa se moverá con él, manteniendo su icono siempre en el centro de la pantalla.

El seguimiento de vehículo se mantendrá activo mientras el usuario no pulse cualquier otro sitio, si lo hace, se cerrará el diálogo de información adicional y el mapa ya no seguirá centrándose en el marcador.

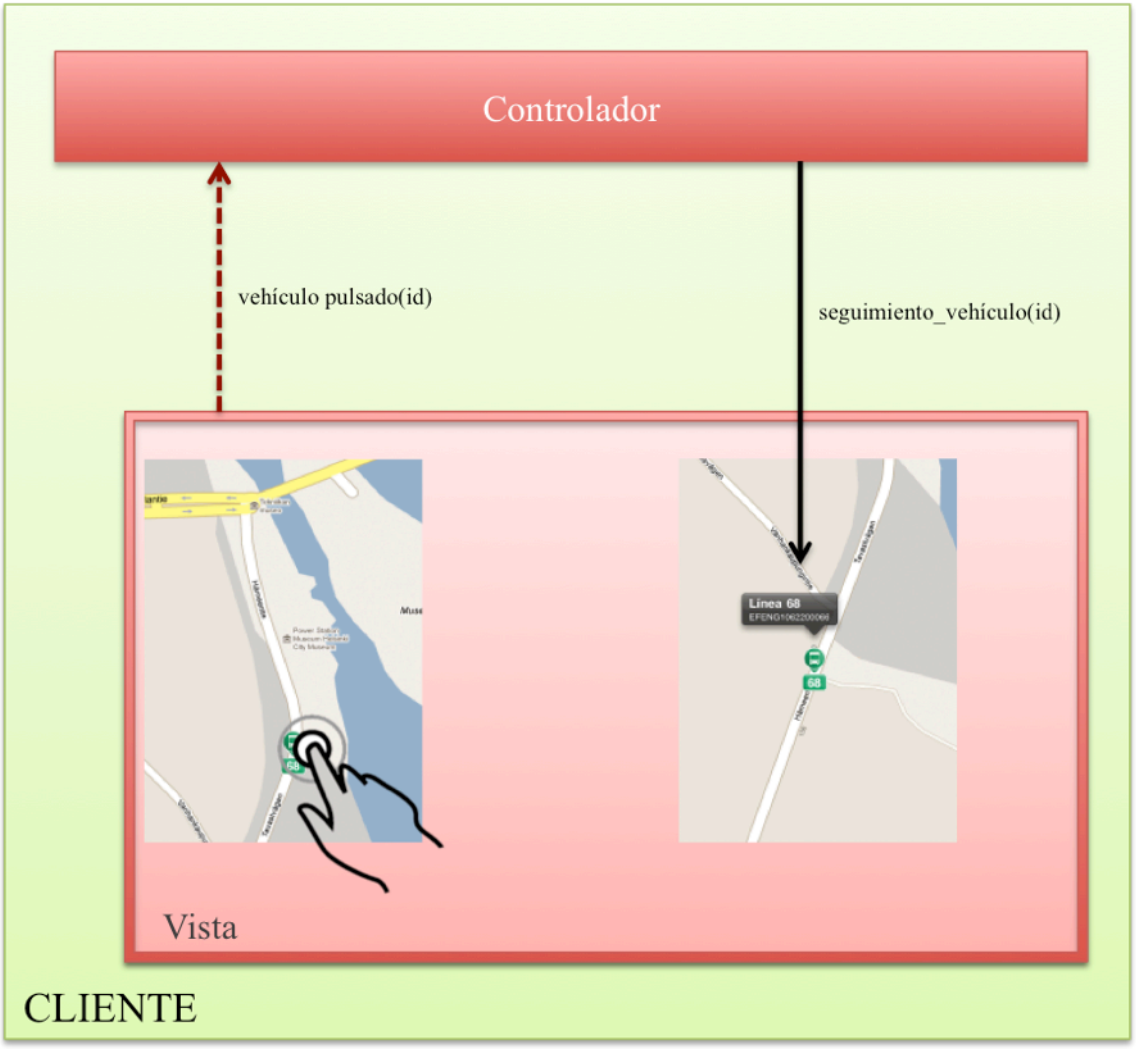

Ilustración 46 – Seguimiento de vehículo

Salir de la aplicación:

El usuario podrá salir en cualquier momento de la aplicación. Ante este hecho, el controlador recibirá una señal de "exit()" y realizará un guardado de la configuración de usuario antes de salir.

El controlador analizará los parámetros de filtrado establecidos antes del cierre, así como el área geográfica que se estaba visualizando y su nivel de zoom. Almacenando esta información, se consigue que, cuando se vuelva a ejecutar la aplicación, el usuario se encuentre con la misma vista que tenía la última vez que utilizó la aplicación.

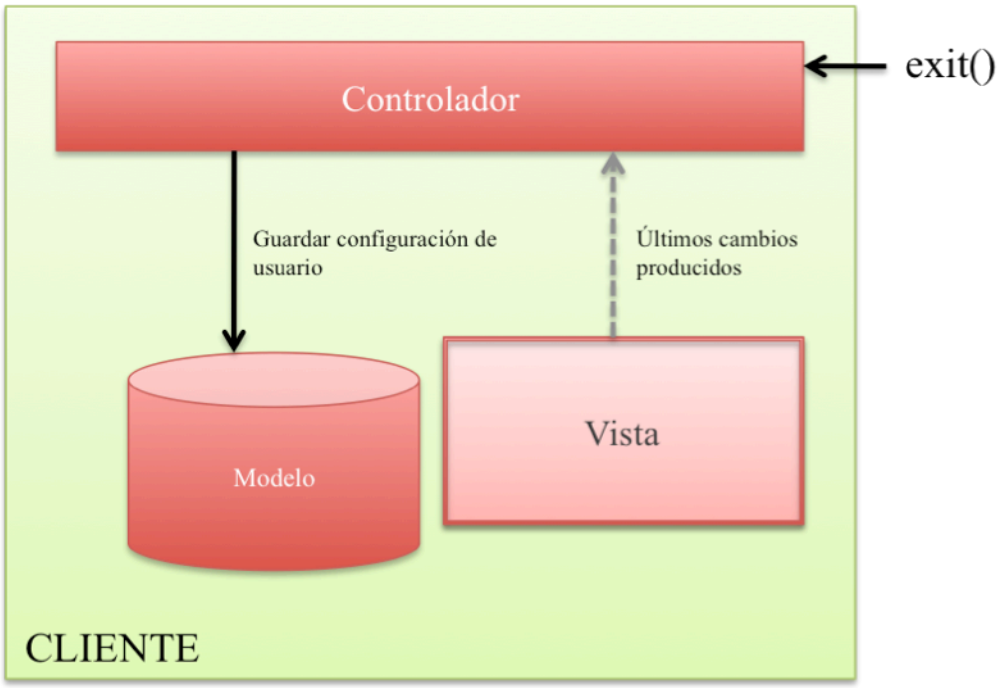

Ilustración 47 - Comportamiento ante salida de la aplicación

Interfaz de usuario de la aplicación móvil

Las mayoría de las interfaces de usuario de aplicaciones para iOS se basan en vistas y controladores. La ventana principal de un programa nativo de iPhone/iPad, no es más que una vista a pantalla completa, ligada a un controlador que captura los eventos producidos por el usuario mediante “toques” sobre la pantalla e indica cómo tratar estos eventos. A menudo, las aplicaciones de iOS recurren a elementos de navegación que permiten la transición entre vistas a modo de menús. Los principales controladores de navegación existentes con esta finalidad, y se usarán para este proyecto, son los siguientes:

❖ UINavigationController:

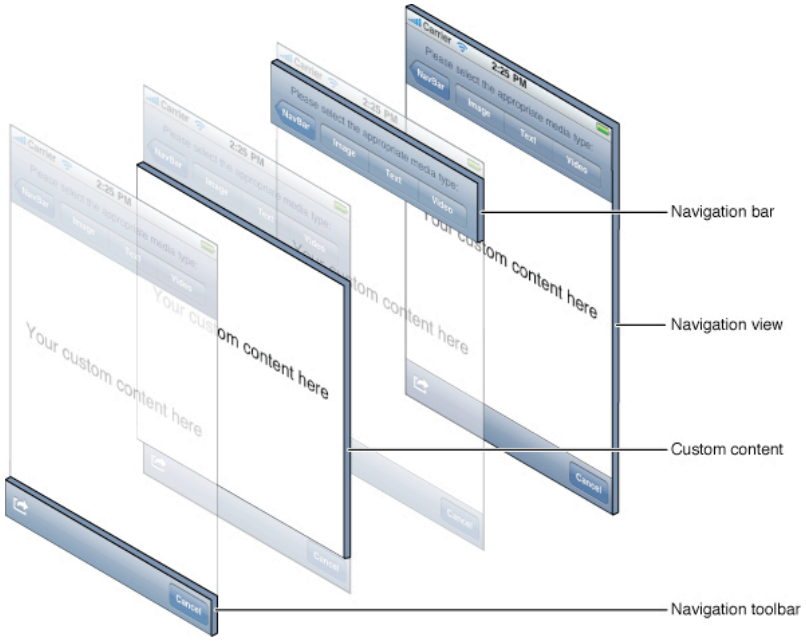

Ilustración 48 - Estructuración del UINavigationController

El UINavigationController permite la transición entre vistas (o controladores de vista) añadiendo de forma secuencial las nuevas vistas en una pila. Para hacer uso de este controlador, típicamente, se inserta una barra de navegación en la parte superior de la pantalla. Cuando se quiere visualizar una nueva vista, se coloca ésta sobre la vista actual con una pequeña animación de barrido, y aparece, en la parte izquierda de la barra de navegación, un botón con el nombre de la página anterior que acaba de ser ocultada. Pulsando sobre este botón, la vista actual desaparecerá, mostrando de nuevo la vista previa. Esta operación de paginación puede repetirse tantas veces como se desee, ocultando y mostrando vistas de la aplicación como si de una pila se tratara.

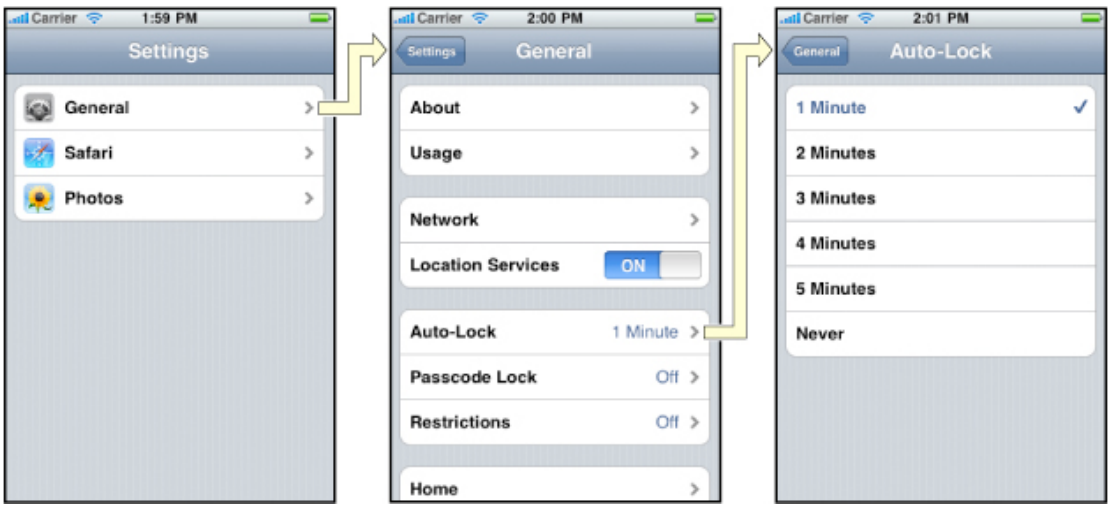

Ilustración 49 - Ejemplo de funcionamiento del UINavigationController

❖ UITabBarController

Ilustración 50 - Estructuración del UITabBarController

El UITabBarController permite la navegación entre controladores de vista a modo de pestañas. En una barra de navegación situada en la parte inferior de la pantalla, se añaden los iconos correspondientes a cada apartado en el que se dividirán las escenas. Cada escena tiene sus controladores de vista precargados, por lo que no goza del dinamismo del UINavigationController. Cuando el usuario pulsa sobre alguno de los iconos de la barra de navegación, se muestra de forma inmediata, el controlador de vista asociado con la escena que lo identifica.

Ilustración 51 - Ejemplo de funcionamiento del UITabBarController

❖ UISplitViewController

Este controlador de navegación únicamente está disponible para el dispositivo iPad. Permite dividir la pantalla principal en dos secciones que interactúan entre ellas y se muestran de forma diferente en función de la orientación del dispositivo. Típicamente, la primera sección suele estar destinada a ser un menú o listado de acciones que afectará a la segunda sección, que será la vista principal. Si el iPad está en posición horizontal, el listado de acciones se mostrará en la parte izquierda, y la vista principal en la derecha. Si se coloca el dispositivo en posición vertical, en primera instancia, únicamente se mostrará la vista principal, pero en su barra de navegación, aparecerá un nuevo botón. Pulsando sobre este botón, se mostrará el listado de acciones en una ventana desplegable.

Ilustración 52 - Ejemplo de funcionamiento del UISplitViewController

Cada una de las secciones del UISplitViewController posee su propio UINavigationController, por lo que además, cada sección puede incorporar sus propias vistas de navegación interna.

Vistos y comprendidos los principales controladores de navegación que estructurarán la aplicación del cliente, ahora pasaremos a definir la interfaz establecida para cada dispositivo.

Mobitransit iPhone

❖ Precarga

La versión de iPhone de la aplicación cliente se inicia con una pantalla de precarga que oculta la aplicación hasta que se haya inicializado debidamente.

Ilustración 53 - Pantalla de precarga

Durante el periodo de precarga, el controlador principal de la aplicación comenzará la tarea de inicialización de la aplicación:

1. Comprobará el establecimiento de conexión con el servidor.
2. Solicitará el fichero de configuración inicial, en caso de que no esté actualizado, hará uso del que ya conserve en el disco.
3. Poblará el modelo de datos con la información del fichero de configuración.
4. Solicitará el fichero de vehículos actuales.
5. Ingresará la información de los vehículos en el modelo.
6. Creará los recursos de vista en función de la información obtenida.

Una vez cargada la información básica para el arranque de la aplicación (operación que dura en torno a unos 4 segundos como mucho), se eliminará la pantalla de precarga y se mostrará la vista principal de la aplicación.

❖ Vista principal

Tras mostrar la vista principal, el controlador se suscribirá al servicio de mensajería que actualizará la información de los vehículos y consultará los avisos e incidencias de tráfico de forma inmediata y sin que el usuario se cerciore de ello.

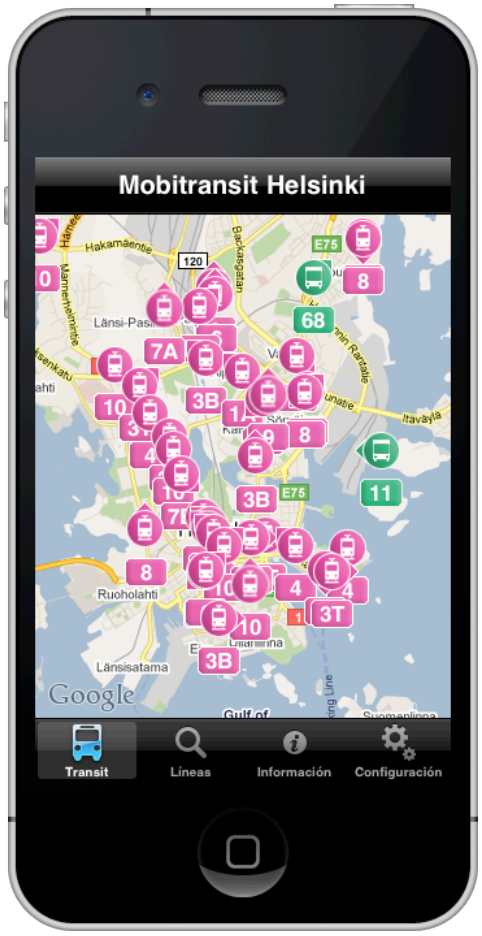

Ilustración 54 - Vista principal

La aplicación para iPhone basa su navegación en un UITabBarController, dividiéndola en cuatro apartados: “Transit”, “Líneas”, “Información” y “Configuración”. La vista principal se corresponde al apartado “Transit” y muestra la funcionalidad primaria de la aplicación en ésta.

En la ilustración 54, podemos observar el mapa de Google Maps como motivo principal, poblado por los marcadores que representan a los vehículos del servicio de transporte de la ciudad. Si el controlador ya ha conseguido suscribirse al servicio de tiempo real, los vehículos se moverán de forma automática por el mapa de forma continuada.

La vista principal incorpora las funcionalidades descritas en apartados anteriores. Si seleccionamos uno de los vehículos, se abrirá un cuadro de diálogo indicando el nombre completo de la línea a la que pertenece y su número de matrícula. Además, mientras el cuadro de diálogo se mantenga abierto, la vista permanecerá centrada en ese vehículo.

El usuario podrá desplazarse y hacer zoom sobre el mapa con sencillos gestos sobre la pantalla, al igual que en la aplicación nativa de Google Maps.

También, en función de las opciones de filtrado de visualización, en la vista principal podrán verse las trayectorias de las líneas y sus paradas sobre el mapa. Esta vista incorpora además, un controlador de navegación y su correspondiente barra en la parte superior de la pantalla. Este elemento de navegación, permitirá visualizar la información de tiempos de llegada de las paradas en una nueva página e indexar el retorno a la vista principal mediante un botón de navegación.

Ilustración 55 - Navegación para visualizar tiempos de llegada

En el caso en que una parada tenga establecida dos listados de tiempos de llegada (uno para trayectos de ida y otro de trayectos de vuelta), el menú mostrará además dos pestañas para visualizar uno u otro.

❖ Vista de listado de líneas

Ilustración 56 - Vista de listado de líneas

El apartado del listado de líneas se corresponde al segundo icono del navegador de pestañas: "Líneas". En él encontramos el listado de líneas, separadas por tipo de vehículo (autobuses y tranvías en el proyecto de Helsinki), y cada celda de línea posee las opciones de filtrado descritas en el subapartado "Vista" del apartado "Diseño e implementación del cliente".

Pulsando sobre los botones correspondientes, el usuario podrá activar y desactivar la visualización de elementos de la pantalla principal: vehículos, rutas y paradas.

El controlador principal se encargará de las operaciones filtrado visual en la vista principal, así como de actualizar el filtrado en la suscripción de mensajería si se modifica la visualización de vehículos.

❖ Vista de información

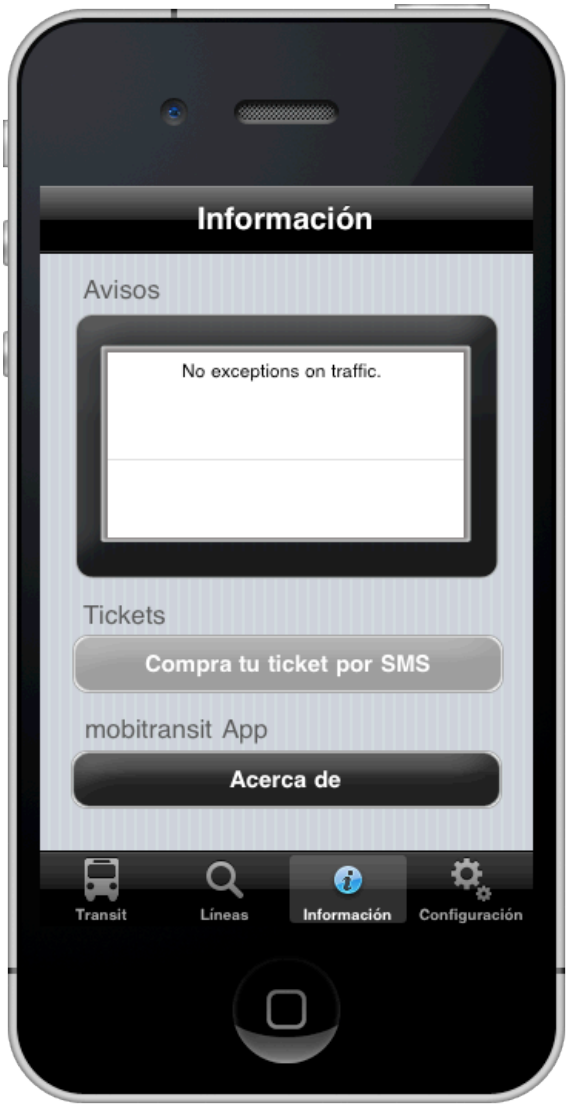

Ilustración 57 - Vista de información

La vista de información incorpora dos funcionalidades adicionales al proyecto:

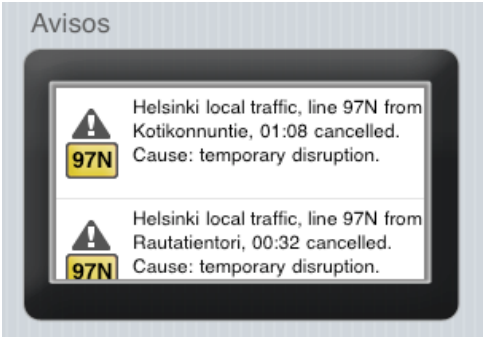

Ilustración 58 - Avisos de tráfico

1. Mostrar los avisos e incidencias de tráfico.

Al iniciarse la aplicación y mostrarse la pantalla principal, el controlador habrá solicitado del proveedor de información, el XML con las incidencias de tráfico que servicio de transportes esté generando en ese momento. Esta información se mostrará en el primer cuadro informativo que vemos en la ilustración 58 bajo el título de “Avisos”.

Ilustración 59 - Vista de ticketing

2. Compra de tickets por SMS

Si el dispositivo permite enviar SMS, el usuario podrá acceder a la pantalla de ticketing.

Como puede apreciarse en la ilustración 59, la vista de ticketing permite:

Comprar un ticket Mobile, equivalente a un bono de transporte de un viaje. Al pulsar sobre este botón se iniciará el controlador de SMS por defecto del dispositivo, indicando la frase a enviar y el destinatario de forma predeterminada.

Llamar al teléfono de información del servicio de ticketing. Al pulsar esta opción, se establecerá una llamada telefónica con el número de atención al cliente de HKL/HST.

Solicitar un taxi. Mediante el controlador de SMS, se rellenarán de forma automática el número del destinatario y, si el usuario está geolocalizado, los datos relativos a su posición para solicitar un taxi a la compañía de taxis de la ciudad de Helsinki.

En la pantalla también se añaden dos enlaces a las páginas de Internet con la información de los términos de uso de estos servicios. Pulsando sobre ellos, la aplicación finalizará y se abrirá el navegador integrado del dispositivo con la página web solicitada.

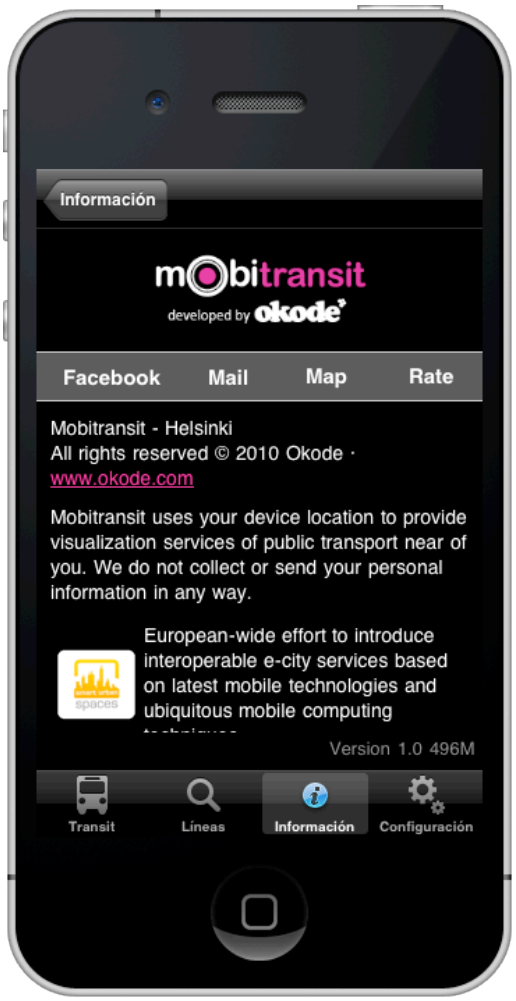

Ilustración 60 - Vista de "Acerca de…"

3. Acerca de mobitransit App

Esta nueva vista muestra la habitual página de información relativa a la aplicación y a la empresa desarrolladora, condiciones de uso, librerías de terceros utilizadas…

Además incorpora cuatro enlaces a modo de información adicional:

Facebook: Abre el navegador del dispositivo con la página de Facebook oficial dirigida a la aplicación mobitransit.

Mail: Abre el gestor de correo del dispositivo con la dirección de correo del servicio técnico de Okode como destinatario.

Map: Abre la aplicación de mapas del dispositivo con la geolocalización de la empresa Okode.

Rate: Abre la página de mobitransit en el App Store para que el usuario pueda opinar y/o puntuar la aplicación públicamente.

❖ Vista de Configuración

Ilustración 61 - Vista de configuración

El cuarto icono del navegador de pestañas se corresponde con el panel de configuración general de la aplicación.

El apartado de “Conexión” muestra un indicador lumínico del estado del servidor. La luz verde indica que la conexión es correcta y que el servicio de actualización de vehículos en tiempo real debería estar funcionando sin problemas. El indicador en rojo informa de que el dispositivo no se ha podido suscribir al servicio de mensajería o que la conexión al bus se ha cortado. En este estado, los vehículos no se moverán por el mapa.

El segundo apartado “Propiedades del mapa”, permite al usuario cambiar el tipo de visualización del mapa entre: vista normal, vista de satélite o híbrido. Por otra parte, también incorpora un interruptor para habilitar/deshabilitar el proceso de geolocalización que muestra la ubicación del usuario en el mapa.

Y por último, el apartado de “Redes sociales”, incorpora dos interruptores que dan pie a una de las funcionalidad adicionales agregadas a la aplicación: compartir eventos en redes sociales. El usuario podrá iniciar sesión en dos conocidas redes sociales: Facebook y Twitter.

Al pulsar alguno de los interruptores, se abrirá la ventana de inicio de sesión correspondiente a la red social seleccionada.

Ilustración 62 - Inicio de sesión en Facebook (izquierda) y Twitter (derecha)

Al rellenar los campos de usuario y contraseña, se establecerá la sesión con la red social seleccionada. Si la sesión se establece con éxito, la aplicación registrará los datos necesarios para mantener la sesión abierta el los siguientes usos de la aplicación.

Durante el transcurso de la aplicación, si el usuario ha iniciado sesión en Twitter o Facebook, podrá crear un “tweet” (en el caso de la primera red social) o una publicación en su muro (en el caso de la segunda) con la información relativa a:

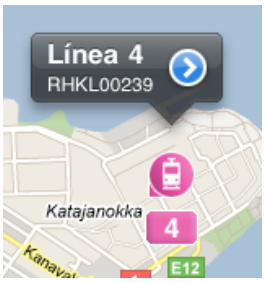

Ilustración 63 - Botón de compartir

Vehículos seleccionados

Al abrir el diálogo de información adicional de cualquier vehículo, aparecerá un nuevo botón azul en la parte derecha del diálogo (al igual que en la ilustración 63).

Pulsando sobre este botón, se mostrará la vista de publicación de mensaje en redes sociales.

En esta vista, aparecerá un cuadro textual, editable por el usuario, que por defecto mostrará el mensaje:

"Estoy viajando en el xxx yyy de la línea zzz. ¡Sigue mi trayecto en tiempo real con Mobitransit! http://www.mobitransit.com"

Donde los campos xxx, yyy y zzz se refieren al tipo de vehículo, identificador del vehículo y nombre/número de la línea, respectivamente.

Si el usuario pulsa sobre el botón "Publicar en Facebook, este mensaje será impreso en su muro.

Si pulsa "publicar en Twitter" generará un nuevo "tweet" en su cuenta con este texto.

Ilustración 64 - Vista de publicación de mensajes en redes sociales

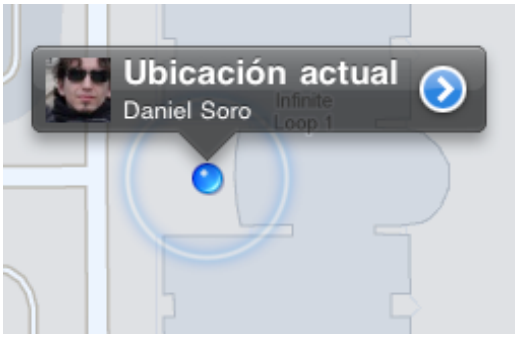

Ilustración 65 - Posición de usuario conectado a Facebook

Posición del usuario

Si el usuario ha iniciado sesión en Facebook, la aplicación consultará su imagen miniatura de su perfil y su nombre de usuario y, al abrir el diálogo auxiliar del icono que indica la geolocalización del dispositivo, aparecerán las siguientes novedades:

En la parte izquierda del cuadro de diálogo, se mostrará la imagen miniatura de Facebook del usuario.

En el mismo diálogo, a modo de subtítulo, aparecerá el nombre de usuario que tenga registrado en Facebook.

Y finalmente, se añadirá, en la parte derecha del diálogo el botón que mostrará la vista de publicación de mensajes en redes sociales, al igual que con los vehículos seleccionados, usando como mensaje predeterminado:

"Estoy en (xxx,yyy). ¡Viendo el transporte público en tiempo real con Mobitransit! http://www.mobitransit.com"

Siendo “xxx” e “yyy” las coordenadas (latitud y longitud) correspondientes a la geolocalización del dispositivo.

❖ Vistas auxiliares

Dado que la aplicación precisa de una conexión a Internet continuada durante su ejecución, se han añadido dos vistas adicionales que se superpondrán a cualquier vista que esté visualizando el usuario cuando se produzca alguno de estos casos:

1. Error en el servidor: Cuando el dispositivo no consiga conectar con el servidor y descargar la información de inicialización del programa.

   Si se produce este error, la aplicación no podrá continuar hasta que la conexión con el servidor vuelva a restablecerse. En este punto, se le sugiere al usuario que vuelva a intentar a utilizar la aplicación en otro momento.

2. Error de conexión: Cuando el dispositivo pierda temporal o permanentemente la conexión a Internet.

   Esta vista desaparecerá en cuanto el dispositivo recupere de nuevo la conexión y la aplicación seguirá su curso habitual.

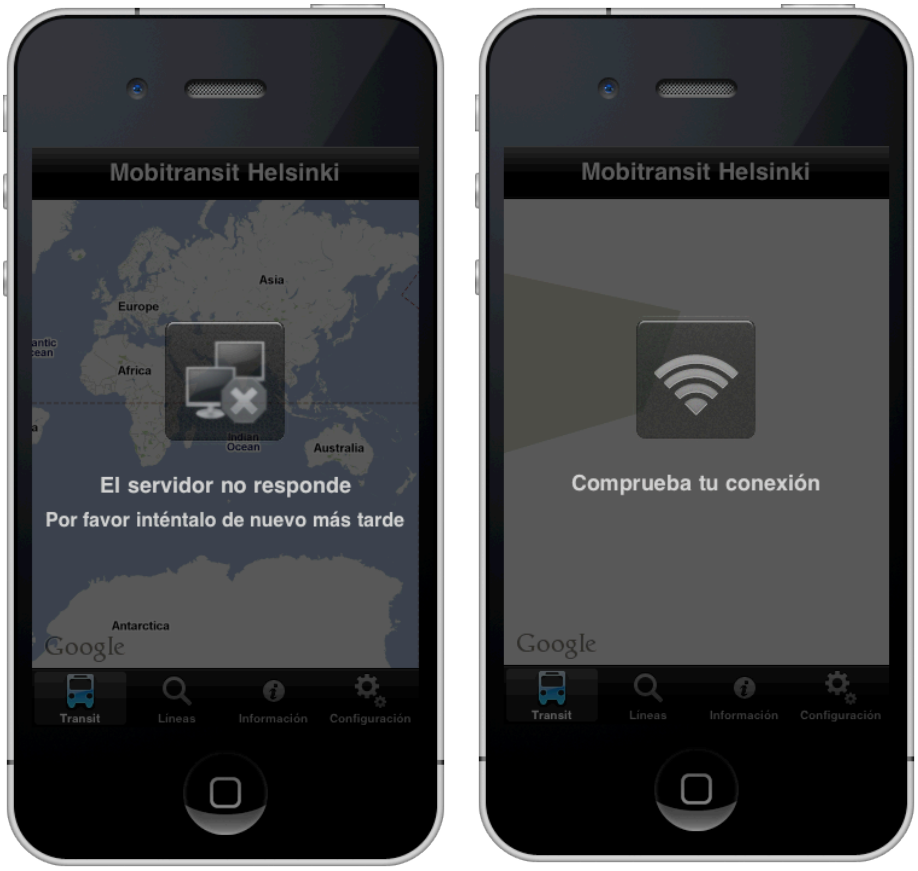

Ilustración 66 - “Error en el servidor” (izquierda) y “Error de conexión” (derecha)

Mobitransit iPad

❖ Precarga

La versión de iPad de la aplicación cliente, al igual que en la versión para iPhone, se inicia con una pantalla de precarga.

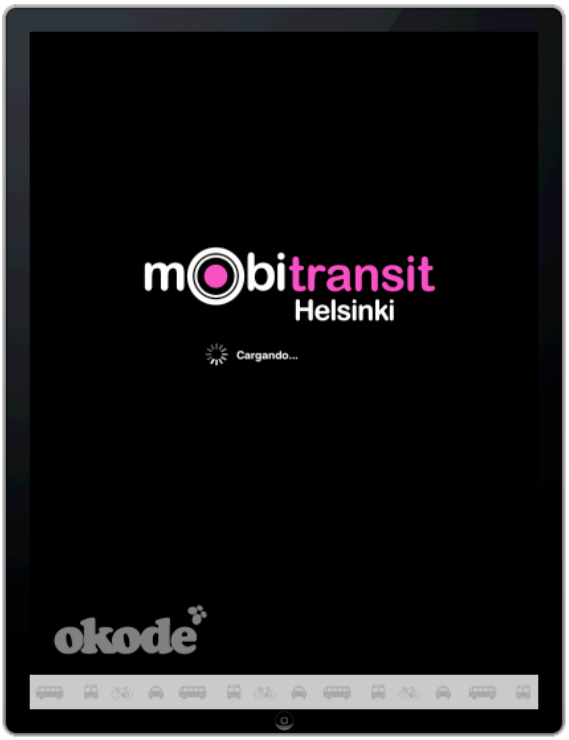

Ilustración 67 - Pantalla de precarga

Durante el periodo de precarga, el controlador principal realiza las mismas operaciones de inicialización que en la versión de iPhone:

1. Establecimiento de conexión con el servidor.
2. Obtención del fichero de configuración inicial.
3. Población del modelo de datos.
4. Obtención de vehículos.
5. Actualización del modelo de datos.
6. Creación de recursos de vista a partir del modelo.

Una vez cargada la información básica para el arranque de la aplicación (operación que dura en torno a unos 4 segundos como mucho), se eliminará la pantalla de precarga y se mostrará la vista principal de la aplicación.

❖ Vista principal

En cuanto finaliza el proceso de precarga, el controlador se suscribirá al servicio de mensajería y obtendrá las incidencias de tráfico, mientras en la pantalla del dispositivo se muestra por primera vez la vista principal de la aplicación.

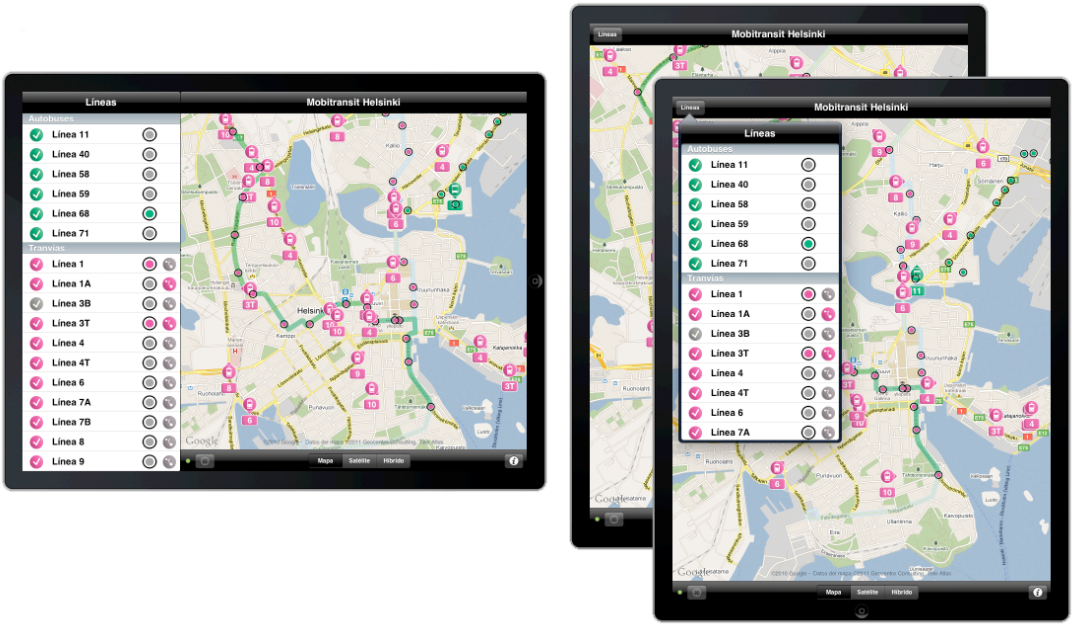

Ilustración 68 - Vista principal

La interfaz diseñada para la aplicación de iPad basa su navegación en un UISplitViewController, donde el apartado del listado de acciones se corresponde al listado de filtrado de líneas y, en el apartado de vista principal, se muestra el mapa donde se visualizarán los elementos del servicio de transporte público. El componente de navegación UISplitViewController, permite que, estando el dispositivo en posición horizontal, se muestre el mapa en el lado derecho de la pantalla y el listado de líneas en el izquierdo en todo momento. Al colocarlo en posición vertical, se oculta el listado de líneas, ocupando toda la pantalla con el mapa y agregando un nuevo botón en la barra de navegación que, al pulsarlo, mostrará en un desplegable, el listado de líneas.

A parte de integrar el listado de líneas y el mapa en la misma vista, gracias a su tamaño, la interfaz de iPad presenta otras diferencias de visualización para algunas funcionalidades. Por ejemplo, el listado de tiempos de llegada relativo a la información adicional de una parada, no se mostrará en una nueva vista, sino que se abrirá en un nuevo diálogo, sin ocultar por completo la interfaz del mapa como en la versión de iPhone.

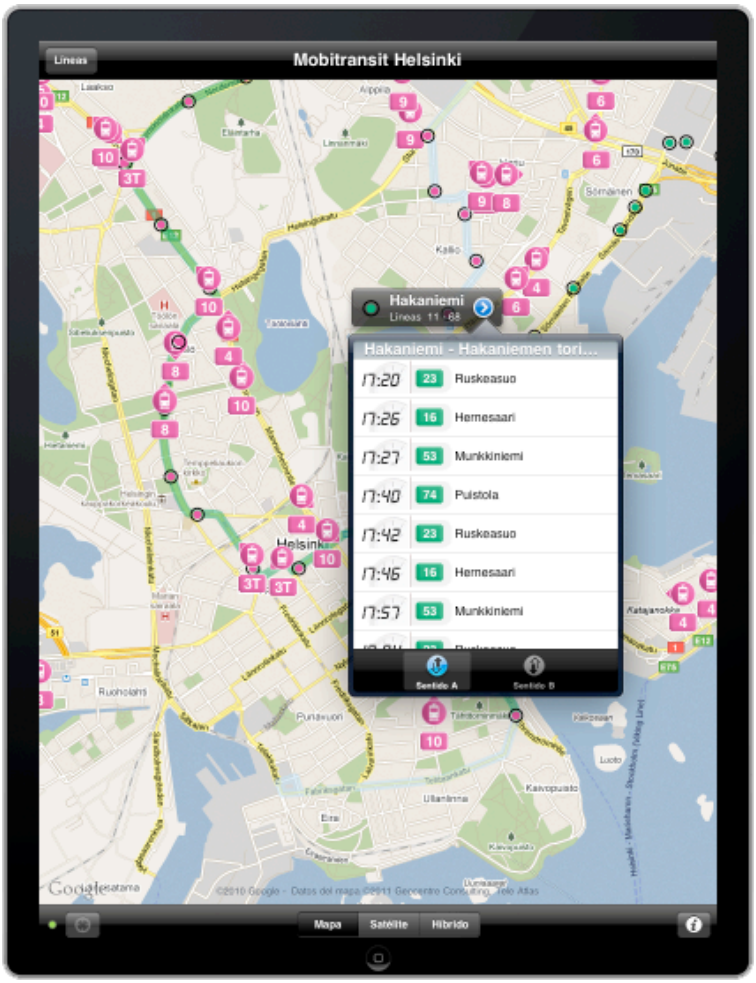

Ilustración 69 - Desplegable de tiempos de llegada

El resto de opciones de la aplicación se encuentran situada en una barra de herramientas situada en la parte inferior de la vista principal.

Ilustración 70 - Barra de herramientas

En esta barra (Ilustración 70), podemos encontrar, recorriéndola de izquierda a derecha:

1. Indicador de estado del servidor: La luz verde indica que la suscripción y conexión al bus es correcta, mientras que el indicador rojo significa que se ha producido algún fallo.

2. Activación/Desactivación de geolocalización: Permite habilitar o deshabilitar la representación de la posición del dispositivo en el mapa mediante su geolocalización.

3. Tipo de mapa: Permite cambiar el tipo de visualización del mapa a normal, satélite o híbrido.

4. Botón de información: Muestra la vista de “Acerca de…” sobre la página principal.

Ilustración 71 - Vista “Acerca de…”

En la vista “Acerca de…” se muestra información legal sobre la aplicación y la empresa desarrolladora. La barra de herramientas oculta sus anteriores iconos y muestra los botones para que el usuario pueda acceder de forma directa a la página web de la empresa, la página de Facebook de la aplicación, el estor de correo con la dirección de soporte técnico de Okode como destinatario, la aplicación de mapas con la dirección de la empresa o la página de Mobitransit del App Store.

Así pues, la versión de iPad de la aplicación, integra toda su funcionalidad en la vista principal, convirtiéndola en una herramienta sencilla y práctica, de inicialización inmediata, que permitirá a los usuarios saber, echándole un rápido vistazo a la pantalla, la ubicación exacta de los transportes públicos en su ciudad.

Esta versión también incorpora las vistas auxiliares de iPhone para advertir al usuario de errores de conexión o de servidor.

Analíticas de uso

A medida que los usuarios utilicen la aplicación, internamente, se realizarán envíos de eventos significativos a un perfil creado de Google Analytics.

Estos eventos han sido establecidos con el fin de confeccionar informes estadísticos sobre el uso de la aplicación y su futuro estudio. Puede accederse en todo momento a esta información a través de la página de Google Analytics de Mobitransit, en el apartado de:

"/Contenido/ Seguimiento de eventos"

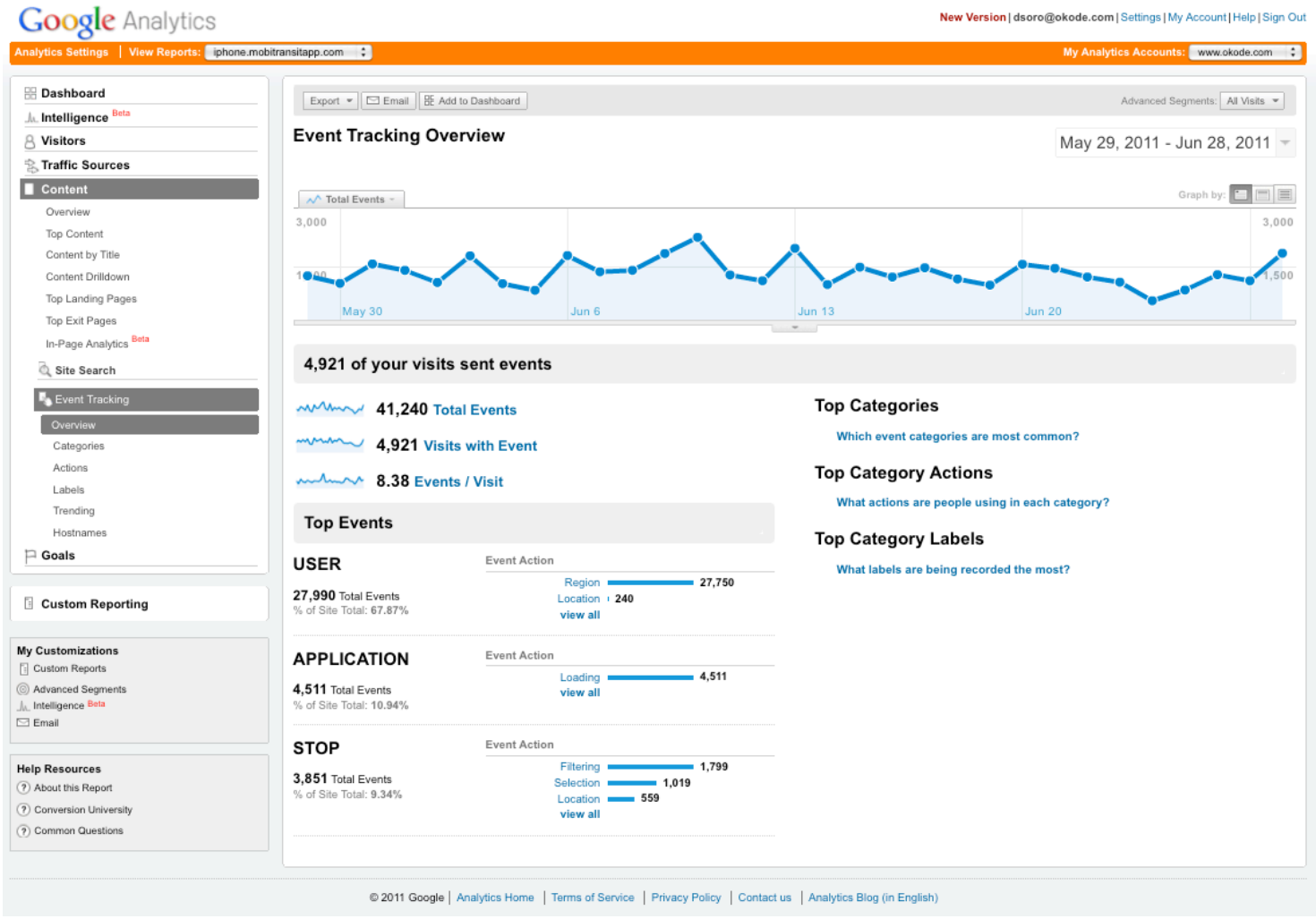

Ilustración 72 - Página de entrada de Google Analytics: Mobitransit

Los eventos se agrupan por Categoría y Acción, y adjuntan una etiqueta textual y un valor numérico entero. Para el proyecto actual se han registrado los siguientes eventos:

Categoría	Acción	Etiqueta	Valor	Descripción
APPLICATION	Loading	“elapsed time”	milisegundos	Registra el tiempo consumido para iniciar la aplicación por parte del usuario.
USER	Region	Zona de coordenadas	Incremental unitario	Registra la región correspondiente a las coordenadas del usuario geolocalizado.
STOP	Filtering	Identificador de parada	Incremental unitario	Registra el identificador de las paradas que se han activado en la visualización de elementos del mapa.
STOP	Selection	Identificador de parada	Incremental unitario	Registra las veces que se ha consultado la información adicional de una parada.
STOP	Location	Identificador de parada	metros	Indica los metros a los que se encuentra el usuario (si está geolocalizado) de la parada que está consultando.
STOP	Schedule	Identificador de parada	milisegundos	Registra el tiempo consumido al realizar la consulta de tiempos de llegada a una parada concreta.
LINE	Filtering	Identificador de línea	Incremental unitario	Registra el identificador de las líneas que se han activado en la visualización de sus vehículos en el mapa.
MARKER	Selection	Identificador de vehículo	Incremental unitario	Registra las veces que se ha consultado la información adicional de un vehículo.
MARKER	Location	Identificador de vehículo	metros	Indica los metros a los que se encuentra el usuario (si está geolocalizado) del vehículo que está consultando.
ROUTE	Filtering	Identificador de línea	Incremental unitario	Registra el identificador de las líneas que se han activado en la visualización de sus trayectos en el mapa.

El evento “USER-Region”, por ejemplo, permitirá hacer un estudio posterior sobre las regiones donde es más habitual el uso de la aplicación.

Cabe mencionar que, las regiones de coordenadas de Mobitransit no registran la posición del usuario, sino que, teniendo establecida una cuadrícula de coordenadas sobre el mapa de Helsinki, se registra la celda correspondiente a la geolocalización del usuario a modo de “zona de uso”.

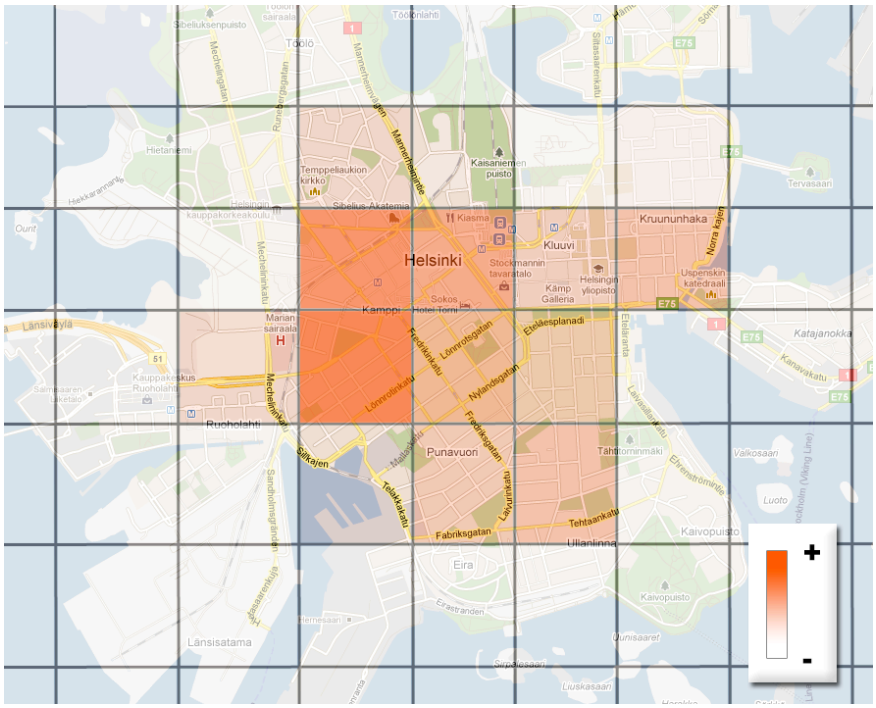

Ilustración 73 - Ejemplo estadístico de zonas de uso

A parte de estos eventos personalizados, Google Analytics registra de forma automática, otros factores estadísticos sobre el uso de la aplicación como:

Visitas diarias, promedio de tiempo uso de la aplicación, gráfico de visitas por ubicación, dispositivo utilizado, operadoras de telefonía móvil…

Publicación en App Store

Una vez finalizada la fase de implementación y pruebas, comienza la etapa de despliegue.

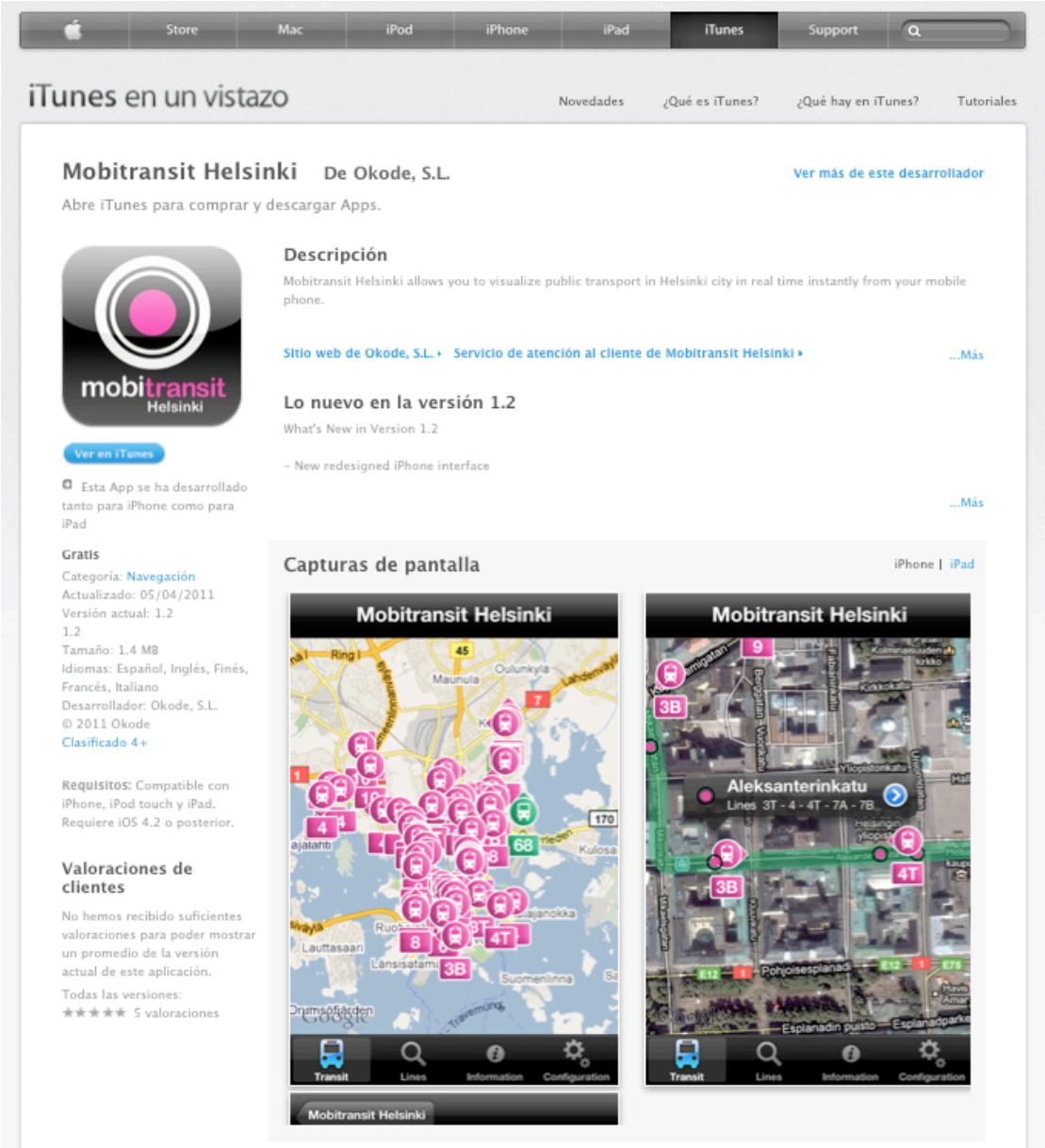

Ilustración 74 - Aplicación Mobitransit Helsinki en iTunes

La forma de subir la aplicación al App Store requiere de un registro previo de la misma en el servicio “iTunes Connect”. Al registro únicamente puede accederse mediante una licencia de distribución de Apple, que permite la creación y despliegue público de aplicaciones para las plataformas de iOS y Mac OS X.

Con las licencias de desarrollo y distribución de la empresa Okode, se pudo registrar la aplicación de “Mobitransit Helsinki” para su distribución gratuita en el App Store.

En primer lugar, se cumplimentó un formulario referente a la información de la aplicación: Nombre de la aplicación, identificador, descripción...

La aplicación final se registró con los siguientes datos:

Nombre de la aplicación:	Mobitransit Helsinki
Calificación:	4+
Categoría principal:	Navegación
Categoría secundaria:	Utilidades
Descripción:	Mobitransit Helsinki allows you to visualize public transport in Helsinki city in real time instantly from your mobile phone. Why is it so easy finding information on maps? Simply because all information is represented graphically on your screen: streets, buildings, shops; everything is visible and easy to find. We think that public transport vehicles should also be presented in the same natural way as any other static element. In fact they are moving on the map! Use Mobitransit Helsinki everyday for viewing buses and trams moving near of you in real time. You will arrive on time to your destination!
Palabras clave:	Helsinki, Finland, transport, real time, bus, metro, streetcar, map, route, transit
Copyright:	2011 Okode
Contact Email Address:	mobitransit@okode.com
Support URL:	http://www.mobitransit.com
Review Notes:	The application uses real time information about transit in the Helsinki city (Finland). You need access to Internet for viewing real time public transport moving in the application.

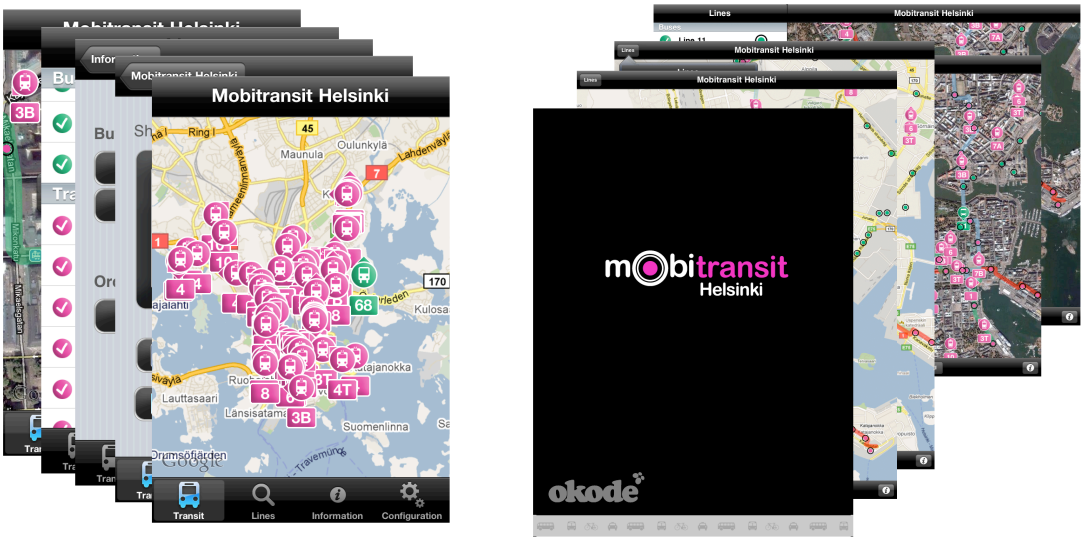

Ilustración 75 - Capturas de la aplicación en iPhone e iPad en iTunes

Ilustración 76 - Icono de la aplicación

Una vez cumplimentada esta información, mediante el propio Xcode, se envió el bundle de la aplicación con sus permisos de distribución a iTunes Connect, y pasó un periodo de pruebas hasta su aceptación final.

Resultados/Conclusiones

Una vez transcurrido el proceso de pruebas y selección de Apple, el proyecto de “Aplicación móvil para la visualización del transporte público en tiempo real sobre plataforma iPhone/iPad”, pasó a ser una aplicación de carácter público bajo el sobrenombre de “Mobitransit Helsinki”. La descarga de la aplicación se puede realizar de forma gratuita desde el App Store, haciendo uso de un dispositivo iPhone/iPad.

http://itunes.apple.com/en/app/mobitransit-helsinki/id412630444

Ilustración 77 - Identificativo de “Disponible en App Store”

Para dar a conocer la aplicación, se crearon, paralelamente al proyecto, varias páginas y anuncios para su promoción en la web:

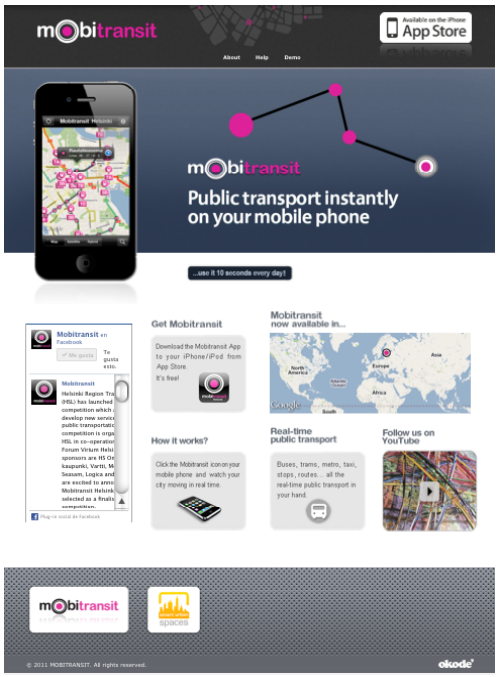

Ilustración 78 - Página oficial de la aplicación: www.mobitransit.com

Ilustración 79 - Página de Facebook

Ilustración 80 - Noticia de Mobitransit en la web de Okode

La aceptación por parte del público, ha sido muy positiva. Mobitransit Helsinki se ha convertido en una de las primeras aplicaciones de navegación de Finlandia, registrando, aún hoy después de varios meses de su lanzamiento, una media de 200 usos diarios.

Objetivos futuros del proyecto

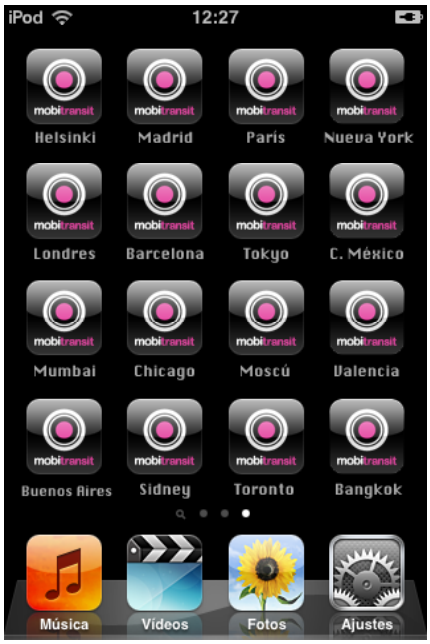

Ilustración 81 - Mobitransit en otras ciudades

Tras el éxito obtenido con Mobitransit Helsinki, el próximo paso, es el de expandir la aplicación a otras ciudades. Se realizará un nuevo estudio sobre la aplicación actual, tratando de generalizar lo máximo posible su parametrización, para poder hacer de ella, una herramienta adaptable a cualquier ciudad.

Teniendo presente, que gran parte de la información de la ciudad de Helsinki, nos ha sido concedida por HSL, y que no todas las ciudades dispondrán de un servicio de transporte público con estas facilidades, se está estudiando una solución posible a ello. La solución más factible consiste en extraer toda la información estática referente a los servicios de transporte público de "Google Transit".

El servicio de Google Transit posee ya más de 477 ciudades registradas, y permite el registro de nuevas ciudades por parte de usuarios. Las nuevas versiones de Mobitransit, adaptarán su modelo de datos al modelo establecido por Google Transit o a uno intermedio formalizado desde nuestros servidores.

En un futuro, cuando una nueva ciudad se quiera sumar a Mobitransit, para obtener su información de líneas paradas y rutas, bastará con direccionar el servicio de obtención de datos a la dirección donde la ciudad esté registrada en Google Transit.

La idea es, tener una aplicación mobitransit para todas las ciudades posibles, permitiendo a todo el mundo, conocer la localización exacta de los vehículos de transporte público de su ciudad, o de la ciudad que esté visitando, en todo momento, echando un simple vistazo a su dispositivo móvil, y facilitando su decisión a la hora de preguntarse "¿Qué transporte me convendría coger?"

Referencias y bibliografía

Referencias

Okode, Innovación y tecnologías de la comunicación
http://www.okode.com

HSL, Helsinki Region Transport
http://www.hsl.fi/EN/Pages/default.aspx

HSL, Reittiopas API
http://developer.reittiopas.fi/pages/en/home.php

Subversion, Tigris Open Source Software Engineering Tools
http://subversion.tigris.org

Jira, Confluence, Atlassian Software Development Tools
http://www.atlassian.com

Apple, iOS Dev Center
http://developer.apple.com/devcenter/ios/index.action

Apple, iTunes Connect
http://itunesconnect.apple.com

Eclipse, Eclipse Foundation
http://www.eclipse.org

ActiveMQ, from the Apache Foundation
http://activemq.apache.org

Google Analytics
http://www.google.com/intl/es/analytics

Mobitransit, Official webSite
http://www.mobitransit.com

Mobitransit, Facebook site
http://www.facebook.com/mobitransit

Mobitransit Helsinki, iTunes App Store
http://itunes.apple.com/en/app/mobitransit-helsinki/id412630444

Wikipedia, la enciclopedia libre
http://es.wikipedia.org

Bibliografía

Mark Dalrymple y Scott Knaster. Learn Objective-C on the Mac. Berkeley: Apress Inc, 2009.

Wei-Meng Lee. Beginning iPhone SDK Programming with Objective-C. Indiana: Wiley Publishing Inc. 2010