Universidad Zaragoza
1542

Proyecto Fin de Carrera

Ingeniería Informática

BUSCOBUS : BUScador del Camino Optimo en BUS

Autor:
Elisa Cauhé Martín

Director:
Eduardo Mena Nieto
Área de Lenguajes y Sistemas Informáticos
Departamento de Informática e Ingeniería de Sistemas

Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Mayo de 2013

BUSCOBUS: Buscador del Camino Óptimo en BUS
Resumen

BUSCOBUS, Buscador del Camino Óptimo en BUS, es un PFC que consiste en una aplicación para dispositivo móvil con GPS y conexión a Internet que permite calcular en tiempo real la mejor combinación de autobuses urbanos para desplazarse en la ciudad de Zaragoza.

En la actualidad, los avances en Sistemas de Información Geográfica, el auge de Servicios Basados en Localización, así como el desarrollo tecnológicos en torno a dispositivos móviles cada vez más cercanos a la sociedad, han favorecido la búsqueda de la optimización en viajes y desplazamientos. En el caso concreto de una ciudad existen muchos factores que puede influir en la elección del medio de transporte a utilizar: localización de paradas y cobertura en diferentes zonas de la ciudad, horarios y frecuencias de cada línea o los transbordos son características propias de cada red de transporte adaptado a la geografía e idiosincracia de cada ciudad siendo estáticas en el tiempo. Sin embargo, existen factores dinámicos como puede ser el estado del tráfico, las horas punta, las condiciones meteorológicas o los desvíos temporales, que afectan directamente al funcionamiento del transporte.

Esta aplicación tiene como objetivo ayudar a los usuarios de la red de autobús con varias combinaciones de líneas posibles, a elegir la mejor de ellas en tiempo real para desplazarse en la ciudad. Para ello el sistema se servirá de información dinámica sobre las estimaciones de tiempo de llegada que cada línea tiene en cada parada de la ciudad, además de los datos estáticos asociado a la red de transporte urbano.

Este sistema de información está compuesto principalmente por tres partes diferenciadas:

- En primer lugar se ha desarrollado un núcleo de cálculo basado en Inteligencia Artificial, en el que mediante un algoritmo de búsqueda, se calcula la mejor combinación de autobuses en tiempo real. La red de autobuses es considerada como un grafo dirigido en el que los nodos son las paradas de autobús y los enlaces dirigidos son las conexiones origen-destino para cada par de paradas.

- En segundo lugar se ha creado un sistema de almacenamiento para los datos estáticos de la red de autobuses de Zaragoza, con información asociada a las líneas de autobús e información de cada parada como su nombre y coordenadas y orden que cada parada tiene en cada línea.

- Finalmente se ha creado un Interfaz Gráfico de Usuario basado en una aplicación móvil que permite solicitar los datos de entrada (origen-destino) y visualizar los resultados, tanto de forma gráfica como textual.

BUSCOBUS está orientada a ser utilizada por cualquier persona, de forma fácil y rápida, sin la necesidad de tener conocimientos avanzados de este tipo de dispositivos. Debido a la evolución de las tecnologías y a las nuevas demandas de la sociedad, este proyecto se ha readaptado en el diseño y en la implementación, a lo largo de su fase de desarrollo.

Caminante, no hay camino,
se hace camino al andar.

Índice

Índice de contenido

Índice de tablas

Índice de tablas

Sección 1

Introducción

Este Proyecto Fin de Carrera (PFC) titulado “BUSCOBUS, BUScador de Camino Óptimo en BUS”, recoge el trabajo realizado por la alumna Elisa Cauhé, estudiante de Ingeniería Informática impartida en la Escuela de Ingeniería y Arquitectura de la Universidad de Zaragoza.

El trabajo realizado consiste en la creación de un aplicación para dispositivos móviles con GPS y conexión a Internet para obtener en tiempo real la mejor combinación de líneas de autobús urbano en la ciudad de Zaragoza. Para su creación ha sido necesario crear una base de datos con la información estática de la red de transporte (paradas y líneas), un algoritmo de búsqueda basado en grafo partiendo de los datos almacenados, y un Interfaz Gráfico de Usuario (GUI) tanto para recoger los datos de entrada como visualizar los resultados de forma gráfica y textual. Este proyecto se diferencia de otros servicios existentes en que la búsqueda del camino óptimo tiene en cuenta la las estimaciones en tiempo real de llegada de los autobuses a cada parada (información dinámica), además de la información estática almacenada.

BUSCOBUS posee un carácter innovador, ya que a fecha de presentación de este proyecto no existe una aplicación que ofrezca este servicio, además de tener interés ciudadano, ya que se trata de una aplicación accesible por cualquier persona que posea un dispositivo con GPS y conexión a Internet. Sin embargo, como primera aproximación y ajustándonos a los objetivos y alcance de un proyecto final de carrera, se ha desarrollado un prototipo básico para un dispositivo móvil concreto. La solución adoptada minimiza las modificaciones a realizar en el caso de ser adaptado a otras plataformas en un futuro. En este aspecto, ha jugado un papel esencial el diseño de una solución que dependa lo menos posible de la tecnología utilizada, especialmente en el contexto actual donde los dispositivos móviles cambian constantemente.

1.1 Contexto y motivación del proyecto

El proyecto nace de la necesidad de optimizar el tiempo empleado por los usuarios de autobús urbano. La ubicuidad que proporcionan los dispositivos móviles así como el auge de sistemas Web con información sobre medios de transporte en tiempo real favorecen el uso de este tipo de aplicaciones. Por esta razón, este proyecto se presenta no sólo como una recopilación del trabajo realizado sino también como un posible servicio para ciudadanos en el marco de proyectos asociados a ciudades inteligentes o proyectos de análisis de movilidad.

En los últimos años hemos presenciado el auge de los Sistemas de Información Geográfica (GIS) y Servicios Basados en Localización (LBS) gracias a la liberación de la información GPS. De ello se ha beneficiado computación ubicua o también conocida como inteligencia ambiental, que tiene como objetivo insertar dispositivos inteligentes en el entorno y en aparatos de uso diario para que las personas puedan interactuar con ellos de manera natural y desinhibida, en todo tipo de situaciones y circunstancias. Dispositivos móviles como teléfonos móviles, smartphones, tablets, PDAs (Personal Digital Assistant) o navegadores GPS forman parte de nuestro día a día. Por todo ello empresas e instituciones aprovechan esta situación tecnológica para adaptar diferentes aplicaciones a las características de estos pequeños dispositivos mejorando así sus procesos de negocio o servicios.

Además del carácter innovador del proyecto en cuanto a tecnologías y dispositivos, y el potencial comercial que posee, resulta especialmente interesante como recopilación de conocimientos de una carrera universitaria con contenidos tan variados como es la Ingeniería Informática. Desde la algoritmia, estructura de datos, teoría de grafos, redes móviles, pasando por la inteligencia artificial, el diseño de bases de datos, la ingeniería del software hasta llegar a los sistemas de información, el diseño de aplicaciones móviles y el análisis de interacción hombre-máquina, han sido abordados en este PFC.

1.2 Objetivos

Esta aplicación tiene como objetivo ayudar a los usuarios de la red de autobús con varias combinaciones de líneas posibles, a elegir la mejor de ellas en tiempo real para desplazarse en la ciudad. Para ello el sistema se servirá de información dinámica sobre las estimaciones de tiempo de llegada que cada línea tiene en cada poste de la ciudad, además de los datos estáticos asociado a la red de transporte urbano.

Como objetivos específicos el Servicio Basado en Localización dispone de:

- Un sistema de almacenamiento de datos, donde se almacenan los datos estáticos de la red de autobús urbana.
- Un algoritmo de búsqueda en grafo, ya que los datos de la red urbana constituyen un grafo de nodos (paradas), líneas y enlaces (enlaces de líneas entre paradas).
- Un Sistema de Interfaz de Usuario que permita recoger los datos de entrada (origen-destino) y devuelva los datos de forma gráfica y textual, bajo un diseño amigable y una configuración con estudiada usabilidad.

Estos objetivos aportan originalidad al proyecto ya que no existe hasta la fecha de presentación un servicio que combine la información estática y dinámica en la búsqueda del camino óptimo en autobús.

1.3 Análisis del problema

En el estudio del tráfico de una ciudad y la previsión de caminos óptimos existen numerosos factores o variables que dotan al sistema de cálculo de rutas un carácter altamente complejo. Ejemplo de estas variables son los atascos y retrasos, los cortes y desvíos, la propia estructura de la ciudad (saturación de población, cascos antiguos,

barrios...), las características de cada línea (frecuencia, horarios) y las necesidades de los usuarios.

Zaragoza dispone de numerosas marquesinas pero solo algunas de ellas poseen estaciones de información de tiempo real para los usuarios y su funcionamiento está limitado a cierta franja de horarios. Desde abril de 2008, TUZSA ofrece a los usuarios el servicio "¿Cuando llega mi bus?" tanto a través de su página Web[1], como la posibilidad de consultarlo desde una parada cualquiera enviando el número de poste por mensaje de texto.

Este proyecto pretende ayudar a los usuarios en la elección de la mejor ruta de autobús en tiempo real teniendo en cuenta tanto los datos estáticos de la red de autobuses urbanos (paradas, enlaces, coordenadas) como los tiempos estimados de llegada a las paradas que TUZSA ofrece. A pesar de ser una aplicación de interés para diferentes públicos (turistas, personas con movilidad reducida, nuevos habitantes en la ciudad...) se considera el tipo de usuario objeto de análisis para este proyecto aquél que utiliza diariamente el autobús, que dispone de varias combinaciones posibles y que busca optimizar al máximo su tiempo. Es este tipo de usuario, acostumbrado a coger todos los días las mismas líneas, quien es susceptible de no tomar la mejor ruta en tiempo real. Un ejemplo sería un estudiante que vive en el barrio de las Fuentes y quiere llegar todos los días a las 9:00 al Campus Río Ebro. La línea 44, cubre este recorrido de forma directa, pero dadas las frecuencias de esta ruta, en ocasiones puede ser más rápido tomar una línea con mayor frecuencia al centro y hacer un transbordo a la línea 23.

1.4 Estudios preliminares

El hecho de realizar el Proyecto Fin de Carrera en un entorno de dispositivos móviles requiere una fase específica dentro de la planificación del proyecto para estudiar las diferentes tecnologías que existen, tanto a nivel hardware como software, para conocer qué posibilidades ofrecen así como analizar a los usuarios a los que va dirigido y los contextos de uso de estos dispositivos.

El primer planteamiento del proyecto, antes de estudiar aplicaciones o servicios similares existentes, estaba basado en el uso de un simulador de flota de autobús con GPS y una base de datos estáticos ficticia, ya que no se disponía del servicio Web de estimaciones de llegada de TUZSA. Varios PFC relacionados con el posicionamiento GPS [12] y con simuladores y gestión del tráfico [13] han sido consultados, con el fin de obtener algunas indicaciones para abordar el problema objeto de estudio.

Tras una primera toma de contacto con el entorno, utilizando y probando diferentes dispositivos, explorando modos de interfaz y analizando el mercado, enseguida ha surgido el planteamiento de cómo se van a integrar los núcleos básicos (núcleo de inteligencia artificial, base de datos estáticos y GUI) en el sistema para poder realizar un estudio más exhaustivo de las funcionalidades, recursos computacionales y de almacenamiento y para poder plantear, en la fase de análisis y diseño, la forma de integración de todas ellas. En esta fase inicial y sin entrar en el diseño, se contemplaban inicialmente dos planteamientos:

1 http://www.tuzsa.es/tuzsa_frm_esquemaparadas.php

- Integración en el dispositivo: este planteamiento busca que el sistema sea independiente de servicios externos. Esta solución contempla cargar la base de datos de datos estáticos en el dispositivo, así como realizar el cálculo del algoritmo en el dispositivo.

- Basado en un servicio Web, cliente-servidor: este enfoque busca liberar al dispositivo móvil de la carga de procesador y memoria para el cálculo, de almacenamiento y de conexión a Internet, dejando del lado del cliente exclusivamente el Interfaz Gráfico de Usuario.

Partiendo de estas posibilidades y teniendo en cuenta la tecnología disponible, las tendencias del mercado y las necesidades y escenarios de uso del servicio, se ha optado por un diseño capaz de ser abordado desde un punto de vista de PFC. El Anexo B recoge una mayor descripción del estudio realizado en la fase de análisis previo.

1.5 Trabajo realizado

El trabajo realizado por la autora parte del planteamiento de una idea innovadora por parte del director, lo que supone comenzar un proyecto desde cero. El hecho de no disponer de ningún otro sistema con el que integrarlo, ya sea infraestructura o servicio, supone libertad a la hora de realizar las diferentes tareas en el análisis, diseño e implementación del proyecto pero también conlleva una carga considerable de trabajo, siendo necesario rechazar planteamientos del proyecto no fundamentales para la consecución del mismo.

Este PFC comenzó en Febrero de 2008 llegando a estar en septiembre de 2008 con un grado de consecución del 95% a falta de terminar de implementar el GUI para PDA. La incorporación al mundo laboral en el Instituto de Biocomputación y Física de Sistemas Complejos (BIFI) de la Universidad de Zaragoza en septiembre de 2008 supuso un retraso en la presentación de este proyecto, y el desfase en tiempo que ha habido ha propiciado (casi obligado) a adaptarlo a la tecnología actual.

Para el desarrollo del nuevo Interfaz Gráfico de Usuario en el dispositivo móvil se ha recuperado parte del GUI desarrollada para PDA. Este desarrollo ha sido integrado en una aplicación nativa para móvil (iPhone), cuyo código ha sido reutilizado de una aplicación móvil de estudio de movilidad desarrollada en el BIFI por Gonzalo Ruiz y Alfredo Ferrer, siendo adaptado para este caso particular.

Este proyecto fue presentado como posible colaboración a la empresa TUZSA en colaboración con el BIFI, con una propuesta de participación en la convocatoria de subvenciones para *Proyectos de Investigación Fundamental orientada a la transmisión de conocimientos a la empresa (TRACE 2009)*, del entonces Ministerio de Ciencia e Innovación. Lamentablemente, aunque resultaba interesante la colaboración, no fue factible en aquel momento por haberse realizado una gran inversión previamente en la Web. Desde la dirección General de Ciencia y Tecnología del Ayuntamiento de Zaragoza también mostraron interés por este proyecto, aunque la fuerte apuesta económica del Ayuntamiento en la EXPO2008 y la previsible situación económica que se acercaba no les permitía hacer inversión en un sistema así.

Sección 2

Análisis, y Diseño de la solución

Previamente al análisis y diseño de una solución ha sido necesario contar con una fase de estudio previo con el objetivo de entrar en contacto con la tecnología a utilizar, y estudiar servicios similares en el mercado. Este análisis previo queda recogido en el Anexo A.

2.1 Análisis

Para abordar un proyecto como BUSCOBUS, que no se basa en ningún otro sistema existente, es necesario dedicar esfuerzos al análisis del problema a resolver, recogiendo los requisitos del sistema y analizando los diferentes casos de uso para poder plantear un buen diseño que cumpla con las especificaciones.

2.1.1 Captura de requisitos

La captura de requisitos se ha centrado en el análisis de las diferentes necesidades del usuario y de las limitaciones que se impone al sistema para conseguir lo que se considera un correcto funcionamiento del mismo. Puesto que se trata de un sistema innovador, que no esta integrado en ninguna otro sistema, ni tiene un cliente asociado que demande ciertas funcionalidades, los requisitos han sido definidos teniendo en cuenta un usuario tipo de rutas de autobús, que utiliza BUSCOBUS en su rutina diaria para elegir la mejor combinación en tiempo real.

Para realizar este análisis se determinan diferentes tipos de requisitos: Básicos, los imprescindibles para la consecución del objetivo de este proyecto y de valor, para dar mayor calidad al sistema. También se especifica si se ha desarrollado o no para esta versión o se plantea como un requisito a cumplir en versiones futuras (Ver Tabla 1).

(*) Leyenda: **R**: Requisito, **F**: Funcional; **NF**: No Funcional; **U**: Usabilidad, **B**: Básico, **V**: Valor, **I**: Implementado, **M**: Modificación futura

Requisito (*)	Descripción	Tipo	Estado
R1	La aplicación se ejecuta en un entorno móvil.	NF-B	I
R2	El sistema almacena los datos estáticos en una	F-B	I

	base de datos.		
R3	Eficacia y eficiencia: el algoritmo de búsqueda encuentra siempre una solución y es la mejor.	F-B	I
R4	Los resultados obtenidos muestran el estado en tiempo real.	F-B	I
R5	El sistema no recarga el dispositivo móvil, ni en procesamiento ni almacenamiento, ni en uso de la red.	F-B	I
R6	El sistema recibe la información del sensor de GPS del dispositivo móvil.	F-B	I
R7	El sistema guarda las preferencias y estadísticas de uso de los usuarios.	F-V	M
R8	La aplicación no depende de software comercial de terceros.	F-B	I
R9	No es necesaria la encriptación de los datos.	F-B	I
R10	La combinación óptima muestra información precisa de líneas, paradas y tiempos de transbordo.	F-B	I
R11	La aplicación permite introducir parámetros al usuario.	NF-U-B	I
R12	El usuario puede optar por ver los resultados de forma gráfica o en modo texto.	NF-U-B	I
R13	El usuario interactua desde un entorno de ventanas sencillo e intuitivo.	NF-U-B	I
R14	La aplicación de interfaz esta adaptada a las características de navegación y usabilidad de pequeños dispositivos móviles.	NF-U-V	I
R16	Los caminos se ajustan a prioridades del usuario: número de transbordos (menor rato a pie, etc).	F-U-V	M

Tabla 1: Requisitos del sistema

2.1.2 Casos de uso

Esta sección pretende recoger los casos de uso necesarios para describir el funcionamiento de BUSCOBUS y posteriormente, junto con los requisitos poder realizar un correcto diseño del sistema.

Se puede considerar un caso de uso estándar que recoge el uso prioritario de esta aplicación, ya que tiene una funcionalidad única. Como se ha mencionado anteriormente, el usuario tipo objetivo de nuestro análisis ha sido un usuario que usa el autobús diariamente en su rutina y que busca en esta aplicación la optimización de su tiempo.

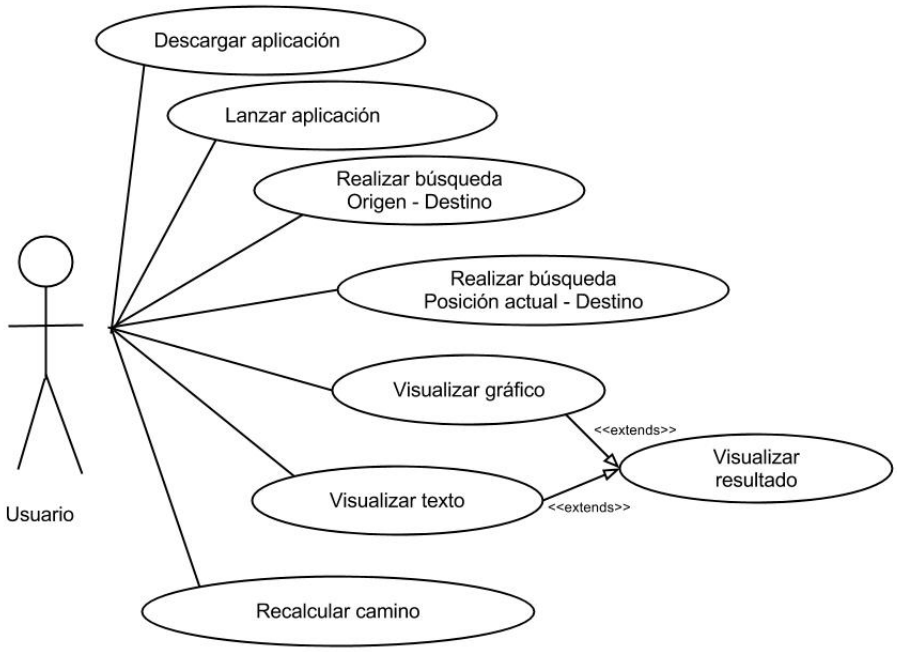

Figura 1: Caso de uso general

A continuación se detalla el caso de uso de la Figura 1:

- Al tratarse de una aplicación nativa, el usuario debe descargar la aplicación de la tienda virtual de aplicaciones correspondiente (Apple Store para iPhone, Google Apps Market Place para Android) e instalarla en su dispositivo.

- Cada día para ir al trabajo o a clase, el usuario lanza la aplicación a través del icono BUSCOBUS instalado en la pantalla principal de su pantalla de navegación.

- El usuario introduce los datos del destino al que quiere ir. No quiere perder tiempo introduciendo direcciones postales: si va al campus Rio Ebro, con poner EINA es suficiente.

- El usuario visualiza el mapa con la señalización de cada parada que va a intervenir en su camino. Esta visualización se hace de dos formas, modo gráfico y modo texto. En modo gráfico el usuario ve posicionadas las paradas

de la ruta, diferenciando en la trayectoria las diferentes líneas, en el caso de existir transbordo.
- El usuario puede recalcular el camino a lo largo de la ruta para ver nuevas rutas óptimas.

Tras analizar los requisitos del sistema y los casos de uso, se ha planteado el diseño de la solución.

2.1.3 Arquitectura y diseño de la solución

Inicialmente fueron considerados tres núcleos fundamentales de la arquitectura: el sistema debía estar compuesto al menos de una base de datos donde almacenar los datos estáticos de los elementos que intervienen en el sistema (paradas y líneas), un núcleo de Inteligencia Artificial que calculase el camino óptimo, y un Interfaz Gráfico de Usuario (GUI) que permitiese realizar la interacción hombre-máquina de una forma sencilla, basada en pantallas gráficas.

De acuerdo a los requisitos realizados y teniendo en cuenta los casos de estudio del sistema, a continuación se analizan las dos aproximaciones al problema hechas en la fase previa como posibles diseños a nuestra solución

- En primer lugar, una arquitectura basada en una aplicación o programa a instalar en el dispositivo, que minimice la dependencia de datos y servicios externos. Para ello tanto el núcleo de Inteligencia Artificial como la base de datos estáticos y por supuesto el Interfaz Gráfico de Usuario deben incluirse dentro del dispositivo. Exigiendo todavía mayor independencia, esta solución también contempla la posibilidad de incluir los mapas en el propio dispositivo. Este tipo de arquitecturas es el utilizado por los navegadores de automóviles tipo TomTom o Garmin. Aún así, no termina de ser completamente independiente, porque es necesario el acceso al servicio de datos de TUZSA en tiempo real, y al servicio de Google Maps para obtener las coordenadas y mostrar los resultados.

- En segundo lugar, una arquitectura distribuida basada en cliente-servidor, donde se libere al dispositivo móvil de la sobrecarga de cálculo y almacenamiento que pueda tener. El GUI puede realizarse a través de navegador Web para liberar al dispositivo de librerías gráficas de interfaz. Esta arquitectura también supone mayor flexibilidad para realizar actualizaciones, ya que la parte del cliente, que es la más dependiente de la tecnología y sus avances, es mucho más ligera.

Como se observa en la Figura 2, BUSCOBUS se ha diseñado como un servicio de información basado en una arquitectura distribuida cliente servidor, que consta de los siguientes elementos: por una parte, una aplicación móvil para instalar en un dispositivo con GPS y conexión a Internet, que actúa de Interfaz Gráfico de Usuario y que capta la geolocalización del dispositivo y, por otra parte, un servidor al que se conecta y que dispone de los núcleos de cálculo y almacenamiento, es decir, la Inteligencia Artificial y la Base de Datos. Además desde el lado del servidor también

se realizan las conexiones a los servicios externos de Google Maps y TUZSA para obtener la información de las paradas en tiempo real y hacer la conversión de coordenadas.

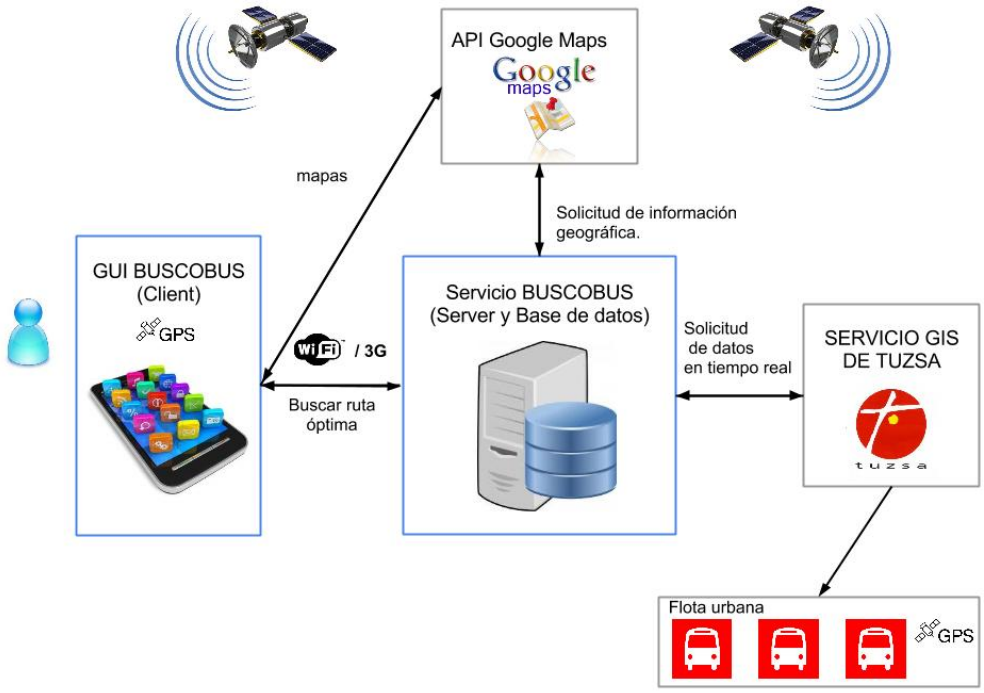

Figura 2: Arquitectura de la solución planteada para el sistema BUSCOBUS

Para llegar a esta solución como arquitectura se han tenido en cuenta diversos factores como la carga del sistema, el tipo de dispositivo a utilizar, la disponibilidad o las características de bases de datos ligeras para integrar en los dispositivos. La principal ventaja que aporta esta solución frente a una aplicación nativa con los núcleos pesados del sistema (la Inteligencia Artificial y la Base de Datos) integrados en un mismo dispositivo, es que la parte del sistema en el lado del usuario se reduce al interfaz de interacción para entrada de datos y visualización. Y se plantea como ventaja, porque la parte del usuario es la parte del sistema que más sufre los avances de la tecnología y más dependiente es de las diferentes plataformas de los dispositivos móviles. Un cambio en la base de datos (por ejemplo, para una parada temporalmente suprimida o desviada) supondría tener que actualizar la aplicación nativa para cada uno de las plataformas. Cualquier tipo de actualización, mejora u optimización en la base de datos o en la Inteligencia Artificial es mucho más rápida con esta arquitectura, y lo que es también destacable, se hace de forma transparente al usuario. Esta decisión ha sido condicionante para el diseño de todo el proyecto realizado.

En el Anexo C puede encontrarse más información sobre los posibles diseños, la arquitectura de la solución y las razones para su elección.

2.2 Diseño

A partir de los requisitos especificados en el análisis, el objetivo es realizar un diseño de la solución que se ajuste a las especificaciones, que sea modular y fácilmente reconfigurable. Los distintos componentes del sistema son analizados a continuación, describiendo las funciones y tareas que cumplen y las relaciones entre ellos suponiendo un caso de uso concreto.

2.2.1 Diagrama de componentes

El diseño de la solución se ha basado en la modularidad del sistema partiendo de los núcleos iniciales identificados previamente. Además de estos núcleos, representados por los módulos *Inteligencia Artificial, DB Manager* y *Aplicación Nativa*, se han definido otros nuevos para poder integrar los servicios externos y poder gestionar toda la información entre componentes.

Para el ilustrar el funcionamiento de BUSCOBUS y las relaciones de los diferentes módulos, en la Figura 3 se muestra un esquema de componentes con el flujo de información entre ellos y a continuación una descripción de cada uno.

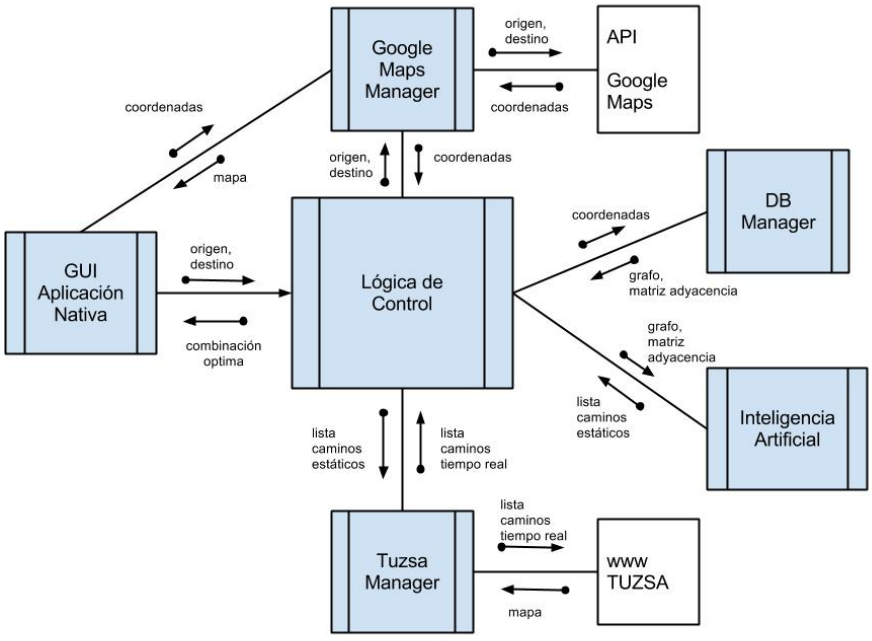

Figura 3: Diagrama de componentes UML

- *Aplicación Nativa*: BUSCOBUS se presenta al usuario como una aplicación nativa para iPhone, que recoge la solicitud de origen y destino del usuario. Los datos son recogidos por el módulo *Lógica de Control* en el lado del servidor. Se comunica con el módulo *Google Maps Manager* para cargar el mapa y una vez recibe los resultados del módulo *Lógica de Control* visualiza el camino sobre él.

- *Lógica de Control*: El módulo L*ógica de Control* es el módulo encargado de gestionar el resto de componentes por lo que se puede considerar el componente principal del sistema. Sirve de conexión entre la *Aplicación Nativa*, el resto de componentes en el servidor (DB Manager, Inteligencia Artificial) y los servicios externos. Recibe los datos de entrada del módulo Aplicación Nativa, bien como cadenas de caracteres ("direccionOrigen", "direccionDestino") o bien en forma de coordenadas y cadena de caracteres (coordenadasOrigen, "direcciónDestino"). A través de una conexión a la API de Google Maps a través del módulo *Google Maps Manager*, los convierte en coordenadas y los pasa al módulo *DB Manager*, del que recibe la estructura del grafo. Este grafo se facilita al módulo *Inteligencia Artificial* para calcular el camino óptimo, que devuelve un listado de los caminos más rápidos según los datos estáticos. Nuevamente se conecta con la base de datos para obtener información de los postes que intervienen en la ruta. Para cada uno de estos caminos se accede al módulo *TUZSA Manager* que conecta con la página de TUZSA mediante conexión http para obtener los tiempos y reordenar el listado. Esta es la forma que se obtiene el listado de los caminos más rápidos en tiempo real. Estos datos son devueltos al módulo *Aplicación Nativa* para ser visualizados sobre el mapa.

- *DB Manager*: Es el módulo de acceso a la base de datos para realizar las consultas y obtener los datos de paradas, líneas y enlaces. Una vez accede a la base de datos, los almacena en una estructura de grafo, basada en nodos y enlaces, de construye la matriz de adyacencia y esta matriz, junto con el grafo, servirán luego de entrada al módulo *Inteligencia Artificial*.

- *Inteligencia Artificial*: El módulo de *Inteligencia Artificial* recibe como entradas el grafo y su matriz de adyacencia. Una vez realizado el cálculo de la ruta óptima mediante el algoritmo de búsqueda A*, se devuelve un listado de los n mejores resultados.

- *TUZSA Manager*: Accede a los datos facilitados por TUZSA con el fin de obtener las estimaciones de llegada de los autobuses a su parada correspondiente. *TUZSA Manager* recibe un listado de las mejores rutas de autobús del módulo *Lógica de Control*. Para cada ruta calcula los tiempos de llegada a cada una de sus paradas y también calcula los tiempos a las paradas que tienen influencia en los transbordos, para hacer una estimación del tiempo real. Devuelve de nuevo los resultados como lista de rutas al módulo *Lógica de Control.*

- *GoogleMaps Manager*: Este módulo es el encargado de realizar la a geocodificación a través de la API de Google Maps. Este módulo recibe como entrada del módulo *Lógica de Control* las direcciones devolviendo coordenadas GPS.

Una vez identificados los componentes y la relación que existe entre ellos, en la siguiente sección se describe la secuencia de la interacción entre componentes.

2.2.2 Interacción de componentes

La Figura 4 muestra la secuencia de relaciones entre los distintos componentes del sistema de BUSCOBUS. Se toma como ejemplo un caso estándar de uso por un usuario.

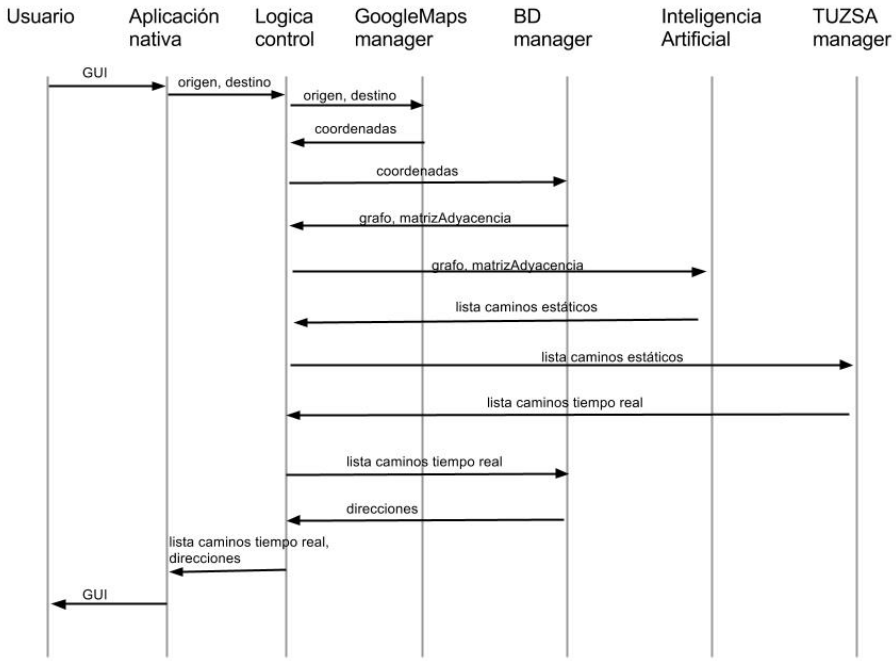

Figura 4: Diagrama de secuencia para un caso de BUSCOBUS

El usuario accede a su aplicación móvil e introduce como parámetros de entrada su origen y destino, dos cadenas de caracteres de dos localizaciones de la ciudad (calle y número por ejemplo). Estas cadenas son enviadas al servidor y recibidas como input por el módulo de *Lógica de Control*. A través de la API de Google Maps, el módulo *Google Maps Manager* las convierte a coordenadas GPS que se pasan como

parámetros de entrada al módulo *BD Manager.* Aquí son introducidos, junto a los datos estáticos de la BD (postes, líneas) en una estructura de datos basada en listas de nodos (postes), paradas (poste+línea), y una matriz de adyacencia, que identificamos como grafo y matriz.

El grafo y su matriz de adyacencia son las que se utilizan en el módulo *Inteligencia Artificial* para la búsqueda del camino óptimo devolviendo una lista de *n* de caminos, Este listado de caminos se basa en la información estática por lo que es necesario obtener los datos en tiempo real de cada uno de los postes.

Es el módulo *TUZSA Manager* el encargado de conectar con la página de Web de TUZSA para obtener las estimaciones de tiempos de cada ruta. Para cada ruta, consulta la estimación de llegada a la parada inicial de la ruta (línea, poste) y de los nodos transbordo (si los hay). Haciendo una conexión a la Web y parseando el código fuente, devuelve el resultado. Este método, a pesar de ser poco ortodoxo, es el único posible ya que TUZSA no dispone de un servicio Web para ofrecer estos datos a otros servicios. Nótese que no sólo intervienen las estimaciones de las paradas de la ruta, sino que, si hay transbordo, también es necesario estimar los tiempos en las paradas de la nueva línea, anteriores a la parada de transbordo en cuestión.

Los *m* mejores nuevos caminos (*m*<*n*) son devueltos a la lógica de control, obteniendo información relevante de los postes implicados en cada uno de ellos (dirección y coordenadas) a través del módulo *DB Manager*. Esta información es necesaria para su visualización y como información complementaria para el usuario.

El listado de caminos junto con la información complementaria, es facilitada al módulo *Aplicación Nativa* que, a través de la API de Google Maps, visualiza los datos en un mapa en el interfaz de usuario.

Sección 3

Implementación y fase de pruebas

Una vez realizado el análisis y diseño de la solución de los elementos que componen el sistema, esta sección pretende dar a conocer el proceso de implementación de las distintas soluciones, describiendo los lenguajes y tecnologías empleados.

3.1 Lógica de control

El módulo descrito como *Lógica de Control* se integra en el servidor de la aplicación y puede considerarse el núcleo de la misma. A través de *servlets* se comunica con el cliente para recibir los parámetros de entrada. Gestiona y secuencia el acceso al resto de los módulos y devuelve los resultados al cliente en formato JSON. La Figura 8 muestra de forma esquematiza la implementación del módulo de *Lógica de Control.*

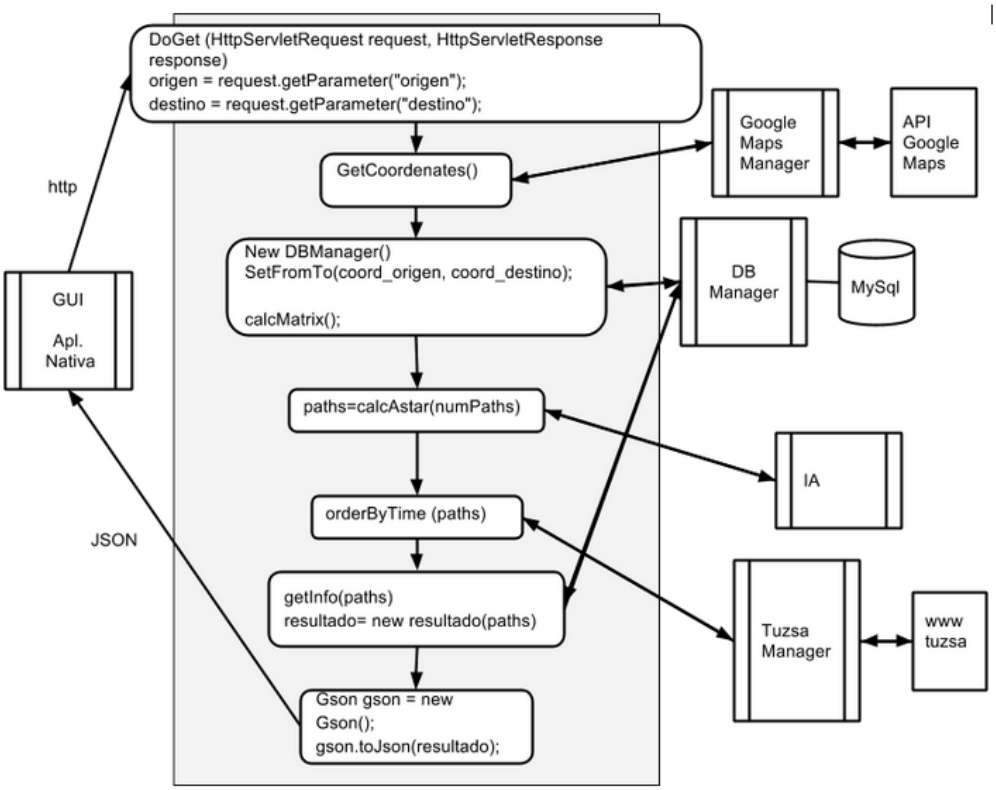

Figura 5: Esquema de implementación del módulo Lógica de Control

A continuación se explica el funcionamiento básico del módulo. Suponemos un usuario que quiere realizar una búsqueda. Su petición es recogida por el módulo Lógica de Control y los datos origen y destino son pasados al módulo Google Maps Manager del que se reciben las coordenadas. A través de DB Manager se construye el grafo a partir de los datos estáticos de la base de datos y se añaden los nodos origen y destino al grafo. Los transbordos no son siempre en la misma parada ya que a veces es necesario desplazarse entre paradas cercanas. Por ello ha sido necesario añadir nuevos enlaces entre paradas cuando se va caminando.

El siguiente esquema de la Figura 6 muestra cómo se añaden origen y destino al grafo como nuevos nodos, con enlaces a pie a las paradas dentro de un cierto radio de alcance. También se muestra cómo se crean enlaces entre paradas cercanas entre sí. La línea negra discontinua muestra un enlace a pie en la base de datos, es decir, se considera que dos nodos está lo suficientemente cerca para ir caminando. Es cierto que es factible ir caminando a todos los puntos de la ciudad pero se ha considerado necesario limitar a 150 metros la distancia origen-parada y destino parada, y a 100 metros la distancia entre paradas para considerarlas como transbordo, lo que supone 2 minutos a una velocidad de 3 km/s (velocidad condicionada por los semáforos y que en la núcleos urbanos no se sigue una línea recta entre dos puntos). Estas distancias así como la velocidad pueden ser configurables.

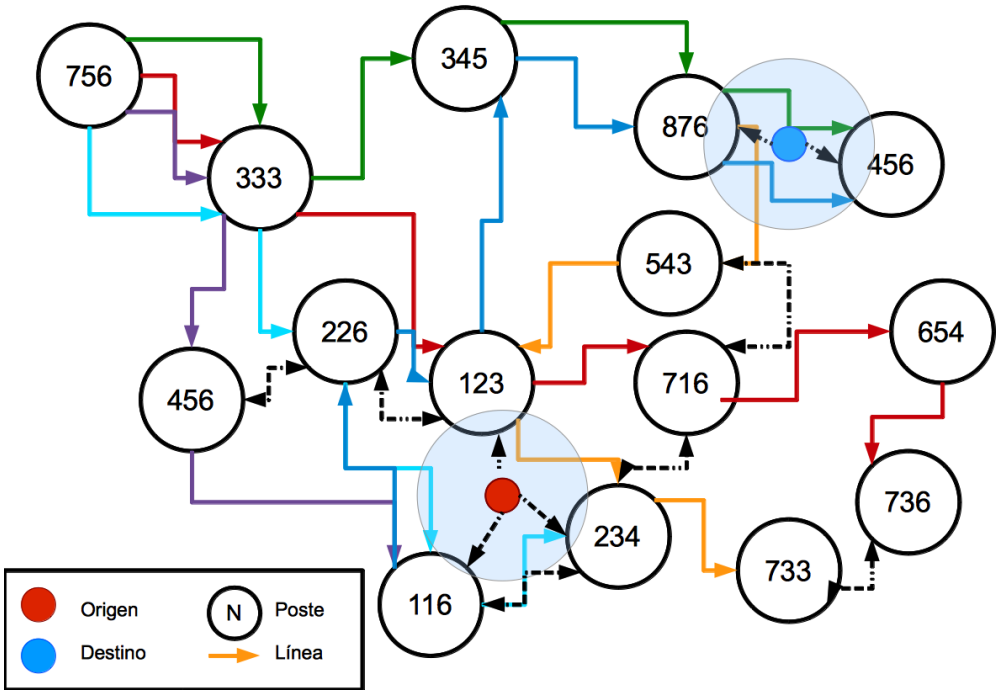

Figura 6: Ejemplo de rutas de autobús con origen y destino en grafo

Una vez cargado el grafo con el origen y destino, es utilizado por el el módulo *Inteligencia Artificial*, que mediante el algoritmo A* de devuelve un listado de cierto número de caminos estáticos que son los que después se ordenan por tiempo dependiendo de los tiempos dinámicos de TUZSA. Nótese que no sólo intervienen las estimaciones de las paradas de la ruta resultado, sino que si hay transbordo también es necesario estimar los tiempos futuros de las estimaciones de TUZSA por para el momento en el que se estima el transbordo. Un ejemplo: Se quiere ir de la EINA al Hospital Miguel Servet y aunque la línea 42 une ambos puntos, la combinación 20-40 puede resultar más rápida. Para calcular el tiempo que tarda toda la ruta no sólo se debe consultar las paradas origen y transbordo, porque a lo que se llega a dicho transbordo (en este caso Gran Vía) es probable que ese autobús ya se haya ido. Hay entonces que consultar los tiempos de las paradas anteriores, es decir, buscar hacia atrás en línea 40, para buscar qué posible autobuses van a llegar cuando se alcance la parada de trasbordo.

Para el desarrollo del sistema cliente servidor, se ha utilizado TOMCAT como servidor. El legunaje de programación utilizado ha sido Java ya que desde el principio se ha buscado un lenguaje multiplataforma con el objetivo de tener un sistema portable. Las librerías utilizadas para el uso de Servlets y de JSON han sido:

com.google.gson; javax.servlet.ServletException; *javax.servlet.http.HttpServlet*; *javax.servlet.http.HttpServletRequest; javax.servlet.http.HttpServletResponse*;

3.2 Base de datos

Los elementos que intervienen en la red de autobuses de transporte urbano de Zaragoza son bastante numerosos. Por un lado, en el momento de presentar este pFC, existe una flota de autobuses con más de 300 vehículos, con multitud de líneas de autobús: 33 líneas regulares, 3 lanzaderas, 2 circulares y 7 líneas búho. Por otro lado, cada línea tiene sus propias frecuencias, dependiendo de franjas horarios y con horarios de salida y llegada diferentes.

Para este proyecto se ha simplificado la información, teniendo sólo en cuenta las líneas regulares. Las paradas de autobús (número de poste y nombre) han sido obtenidas de la página Web de TUZSA mediante un pequeño programa creado para parsear el código HTML. Una vez más TUZSA no dispone de un servicio Web que permita hacer este tipo de consultas. La información relativa a la geolocalización de las mismas, fue añadido *a posteriori ,* a través de un contacto personal que de forma extraoficial y por el interés que tenía este proyecto para él, facilitó los datos para su almacenamiento.

En la actualidad mucha de la información de la base de datos ha tenido que ser actualizada ya que Zaragoza ha sufrido numerosos cambios en los últimos años por las tareas de acondicionamiento para el tranvía.

El almacenamiento de esta información ha sido necesario para alcanzar los objetivos del proyecto si en un principio ante de disponer de ellos se había pensado trabajar con una base de datos artificial. La Figura 7 muestra el modelo Entidad-Relación que fue implementado después en la base de datos.

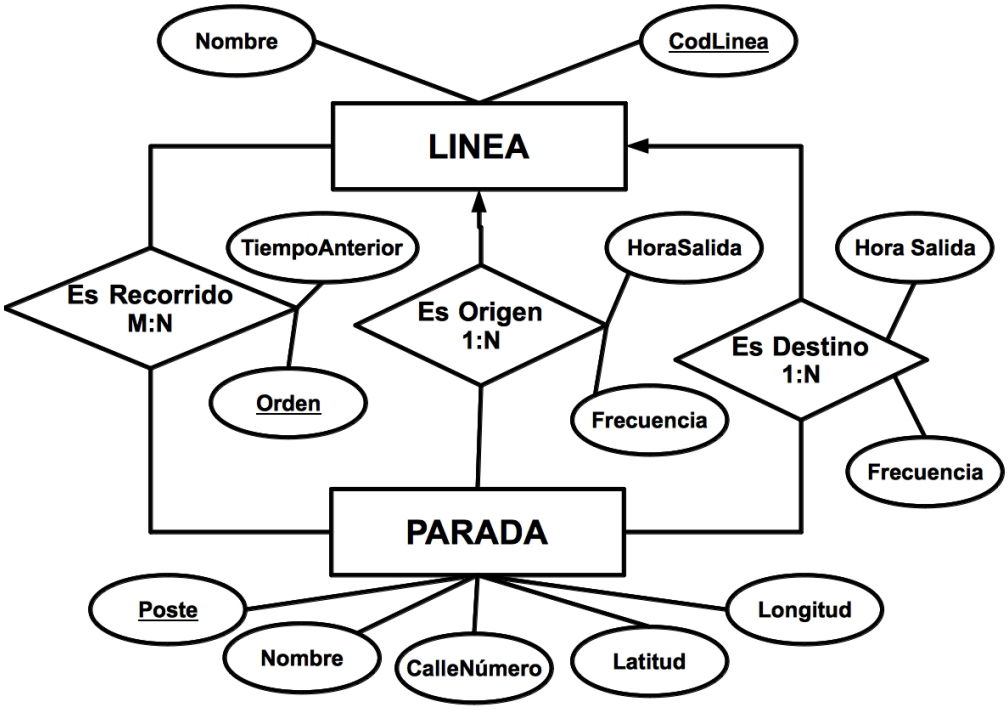

Figura 7: Modelo Entidad-Relación de BUSCOBUS

Respecto al software utilizado, como gestor de base de datos se ha utilizado MySQL por ser código abierto y se ha hecho uso de las siguientes librerías en el código fuente.

java.sql.Connection; java.sql.DriverManager; java.sql.ResultSet; java.sql.Statement; java.sql.SQLException;

3.3 Inteligencia artificial

La inteligencia artificial es el valor añadido que posee este proyecto frente a otros servicios existentes que ofrecen información sobre medios de transporte a los usuarios, ya que permite realizar una búsqueda óptima de las combinaciones de autobús teniendo en cuenta datos dinámicos en tiempo real.

El módulo *Inteligencia Artificial*, esta basado en la búsqueda del camino más rápido en grafo y para ello se ha implementado el algoritmo A* para ser usado con la estructura de datos planteada. Las razones para basar la búsqueda en este algoritmo en el algoritmo A* son las siguientes:

- Es un algoritmo completo: en caso de existir solución, la encuentra.
- Este algoritmo combina la búsqueda en anchura y la búsqueda en profundidad, basando su función de evaluación:

$$f(n) = g(n) + h'(n)$$

- La solución es siempre la de menor coste siempre que la función $h'(n)$ sea heurística admisible, que no sobreestime el coste real de alcanzar el objetivo. En este caso se ha tomado como heurística el camino en línea recta a una velocidad de 80km/hora, ya que nunca será superado.

Para la búsqueda en A* se han creado dos estructuras de datos auxiliares, como una lista de nodos abiertos y cerrados, donde los nodos se añaden por orden de $f(n)$ de cada nodo.

Ha sido necesario realizar diversas modificaciones al algoritmo original, ya que la red de autobuses no es un grafo con una única arista entre dos nodos: dos paradas de autobús pueden estar conectadas por varias líneas.

También ha sido necesario establecer algunas comprobaciones a las rutas ya que algunas, aunque viables, deben ser descartadas por carecer de sentido. Para ello se han añadido penalizaciones para sobrestimar el coste de algunos caminos por ser excesivamente incómodos o que asumen mucho riesgo de sufrir cambios y resultar en un camino lento. A continuación se detallan las comprobaciones que se han tenido en cuenta. Para ellos se utiliza la siguiente nomenclatura L1>L1>L1>L2>L2. Esto quiere decir que para una ruta de 5 paradas un usuario se sube en la línea 1 y en la 4ª parada ha hecho transbordo a la L2. Un 0 indica que la el tramo se hace caminando, cuando no coincide en la misma parada. Estos son algunos ejemplos:

- Si se hace un transbordo innecesario. Ejemplos: 23> 20 vs. 23>23; 40>20> 35 vs. 40>40>35> .
- Si existen caminos circulares. Ejemplos: 20>23>20.
- Si se realizan más de 3 transbordos. Un ejemplo: 20>23>45>42.
- Si se realiza un transbordo a pie, pudiendo hacerlo en bus. 20>0>23 vs. 20>20>23.

Este módulo ha sido implementado en Java por ser más sencillo a la hora de plantear el algoritmo con clases y por ser multiplataforma. No se han encontrado deficiencias en el tiempo de cálculo por estar el cuello de botella en el acceso a la página de TUZSA.

En el Anexo D se muestra el código desarrollado.

3.4 Aplicación Nativa

A partir del análisis de requisitos que están relacionados con el dispositivo móvil tanto a nivel funcional como no funcional y de usabilidad, se ha creado un diseño del módulo *Aplicación Nativa* incluyendo diseño de GUI. (Véase Anexo D). En esta sección se recoge el resultado de implementar dicho diseño para cumplir los requisitos especificados.

La aplicación consta de un interfaz de comunicación con el usuario sencillo que le permite introducir los datos de la búsqueda. Aunque en un principio se había pensado que el usuario introdujese origen y destino en la pantalla de inicio, en el momento de la implementación se ha optado por evitar este campo de entrada ya que el uso común de la aplicación va a ser con la posición actual del usuario tomada del navegador GPS. Véase figura 8.

Figura 8: GUI de BUSCOBUS. Pantallas de inicio, resultado modo mapa y modo texto.

La pantalla de visualización del resultado muestra en diferente color cada una de las líneas y permite obtener información relativa a cada parada. En la parte inferior se visualizan las lineas implicadas en la ruta óptima. Dispone de dos botones, uno para recalcular y otro para visualizar el resultado en modo texto.

Como plataforma cliente para prototipos con el usuario se ha elegido iPhone por ser uno de los modelos más extendidos junto con Android y disponer de uno para su desarrollo, así como un equipo iMac para poder desarrollar en el IDE de Apple para iOS. Esta elección fue condicionante a la hora de realizar la implementación ya que otros dispositivos se basan en otros OS, por lo que el desarrollo hubiese sido diferente. En un futuro se estudiará desarrollarlo también para Android.
.

3.5 Fase de pruebas

Esta sección describe el proceso de testing o fase de pruebas que se ha llevado a cabo para la verificación y validación de BUSCOBUS, tanto para cumplir los objetivos dentro del marco del PFC, como satisfacer las necesidades del usuario desde el

punto de vista de calidad de servicio. A continuación se especifican los objetivos que se ha pretendido conseguir:

- Comprobar el correcto funcionamiento de la aplicación, verificando que se adecua al caso genérico de uso.
- Validar los resultados obtenidos por la optimización del servicio,
- Minimizar posibles riesgos que afecten a la calidad del servicio o den lugar a fallos en el caso de uso.

La fase de pruebas ha comenzado con anterioridad a la implementación de todo el sistema, ya que está compuesto de módulos bastante independientes que permiten comprobar el cumplimiento de los requisitos planteados previamente, validar los resultados obtenidos, porque al tratarse de nuevas tecnologías los resultados podrían derivar en un replanteamiento del diseño de la solución. En las siguientes subsecciones se detallan los diferentes tipos de prueba aplicados.

3.5.1 Pruebas de caja blanca

Las pruebas de caja blanca fueron realizadas con la finalidad de validar el flujo de ejecución del sistema, especialmente en el entorno del algoritmo de búsqueda en grafo. Estas pruebas sirvieron para corregir diversos problemas organizacionales y estructurales que hacían que el algoritmo de búsqueda no funcionase correctamente. Para ello se utilizó el debugger del entorno de desarrollo Eclipse. Una ventaja para este tipo de pruebas fue que la red de autobuses a muy pequeña escala es fácilmente intuible por cualquier persona por lo que diferentes inputs con valores extremos fueron probados, para buscar situaciones no contempladas por el algoritmo.

El módulo *Lógica de Control* también fue sometido a un testeo más exhaustivo, ya que al ser el módulo encargado de acceder de forma secuencial cada uno de los módulos, algunos problemas en el flujo de ejecución podían afectar al resto de la aplicación.

Los resultados de las pruebas de caja blanca han sido algunos fallos en la implementación del algoritmo A* que no devolvía los resultados, ejecutando la búsqueda indefinidamente. Los errores han ido corrigiéndose conforme han surgido, al igual que en el *módulo de Control.*

3.5.2 Pruebas de caja negra

Las pruebas de caja negra se han realizado con el fin de encontrar los siguientes tipos de errores:

- Ineficiencia en el algoritmo de optimización
- Estado de no terminación del algoritmo de optimización, cuando el algoritmo A* siempre encuentra la solución óptima.
- Errores en el acceso a los servicios externos
- Errores en el interfaz

Estas pruebas fueron realizadas tanto de forma unitaria como pruebas de integración entre los diferentes componentes del sistema. Puesto que la implementación de cada módulo se hizo de forma secuencial, cada nuevo módulo a incorporar debía cumplir las pruebas de integración.

A continuación se describen las principales pruebas realizadas. En este caso también fue el módulo de Inteligencia Artificial el que fue testeado en mayor profundidad.

Pruebas unitarias:

- Módulo *Inteligencia Artificial*: Se realizaron para comprobar que los resultados del módulo de Inteligencia Artificial eran coherentes.
- Módulo *GUI*: Se realizaron para buscar problemas de formato en la visualización en dispositivo móvil.

Como resultado de pruebas unitarias se encontraron resultados poco coherentes con la lógica de transporte de una ciudad, como coger varias líneas de autobús cuando solo había que cruzar la calle al otro lado. Este hecho derivó en la incorporación de penalizaciones para ajustar los caminos razonables. En el caso del interfaz, al tratarse de una aplicación tan sencilla con pocas funcionalidades y muy concretas, no se han encontrado incidencias significativas.

Pruebas de integración:

- Integración de Módulo *Lógica de Control* con los módulos de acceso a servicios externos: Se han realizado para buscar posibles fallos en los datos devueltos por los módulos *TUZSA Manager* y *Google Maps Manager.*
- Integración de Módulo de Lógica de Control con el Módulo GUI. En esta ocasión las pruebas se han centrado en posibles fallos de visualización de las rutas.

Las pruebas de integración han tenido como resultado algunos errores encontrados al buscar coordenadas. Un ejemplo de este tipo de errores ha sido que algunas de las localizaciones de Zaragoza no son encontradas en Google Maps, ya que las sitúa en Puebla de Zaragoza en México.

Aceptación:
Para validar de forma genérica el correcto funcionamiento del sistema, las pruebas de aceptación se han realizado mediante el simulador del xCode, IDE de Apple, si bien, si este producto sale finalmente al mercado se definirá como versión Beta para poder realizar una fase de global pruebas más exhaustiva.

3.5.3 Resultados

Como resumen a los resultados objetivos, se destacan dos que son los que más han hecho replantear el diseño e implementación adoptados: el tiempo de ejecución de la búsqueda y los resultados de rutas y caminos.

- Tiempos de ejecución: Diversas configuraciones previas del algoritmo A* tenían un tiempo de ejecución de varias decenas de segundos, en ocasiones

minutos. Tras diversas optimizaciones del código se ha rebajado los tiempos de ejecución a pocos segundos, siendo el coste del algoritmo en torno a 1 segundo en el peor de los casos, por lo que el restante depende de la conexión a TUZSA, lo que supone un cuello de botella dada la velocidad ofertada por el servicio de TUZSA.

- Resultados de los caminos óptimos: La propia validación de resultados también ha sido objeto de muchas pruebas. Prueba de ello ha sido la incorporación sucesiva de distintas penalizaciones para omitir combinaciones de líneas poco lógicas, y los ajustes de las penalizaciones y márgenes de error asignados cuando influyen transbordos y rutas a pie.

El análisis de estos resultados tiene como consecuencia el planteamiento de mejoras y en algunos casos de rediseños que se plantean en la siguiente sección, contemplados como futuras líneas de trabajo.

Apartado 4

Conclusiones

BUSCOBUS se ha desarrollado con el objetivo de ayudar a los usuarios de autobús que posean un dispositivo móvil con GPS y acceso a Internet, a elegir entre posibles combinaciones de autobús en tiempo real, basándose en la información estática de la infraestructura de la red de transporte, como en la dinámica, dada por las estimaciones de autobús de TUZSA en cada parada.

Una vez planteado el contexto del problema, los objetivos de este PFC, y la información sobre el análisis, diseño, implementación y fase de pruebas de la aplicación, podemos concluir que:

- BUSCOBUS consiste en una herramienta que permite a usuarios determinar en tiempo real la mejor combinación de autobuses a tomar en la ciudad de Zaragoza. Para ello utiliza como interfaz de usuario una aplicación nativa para iPhone, y como servicio Web con una base de datos, un módulo de inteligencia artificial y una lógica de control, con accesos a los servicios de Google Maps y TUZSA.

- El entorno en el que se ha trabajado es atractivo, visual, útil y cercano a la sociedad, pero está basado en tecnologías dinámicas en un periodo de cambios constantes.

- La solución adoptada en cuanto a arquitectura y tecnologías utilizadas no es la más eficiente, pero sí la más sencilla y viable desde el punto de vista de la autora para un primer prototipo de producto. Futuras líneas de trabajo intentará mejorar tanto las deficiencias en el diseño como añadir nuevas funcionalidades.

Esta sección tiene como objetivos detallar la gestión del proyecto que se ha llevado a cabo, analizar las dificultades encontradas y planteamientos de nuevas líneas de trabajo y valoración personal sobre la realización de este PFC.

4.1 Planificación

La planificación inicial de este proyecto se realizó al comenzar el proyecto en febrero de 2008. Esta planificación contemplaba una primera fase de contacto con las tecnologías disponibles en torno a los dispositivos móviles, evaluando las posibilidades que ofrecía, sus limitaciones y su compatibilidad con el resto de elementos. Después se realizó una fase de análisis, con la toma de requisitos y los casos de uso para posibles usuarios de BUSCOBUS. El objetivo fundamental era definir con claridad las funcionalidades básicas que el sistema debía cumplir para ser un producto con interés y viabilidad y que se ajustase al alcance y dedicación que se supone a un proyecto fin de carrera.

Tras el análisis de los requisitos y funcionalidades, se procedió al diseño de la solución, fuertemente condicionado por la primera fase de contacto inicial. Tras varios planteamientos y continuas valoraciones se apostó por una arquitectura sencilla basada en un servidor Web, minimizando la parte del cliente, tanto para liberarlo de carga, como para eliminar complejidad.

La implementación fue completamente modular, comenzando por la algoritmia de la Inteligencia Artificial, pasando por la creación de la base de datos y el acceso a servicios externos y finalizando con el servicio Web y el componente GUI. La fase de testing se centró en la verificación y validación de resultados del módulo de Inteligencia Artificial y en la experiencia de usuario del interfaz en la aplicación nativa.

El hecho de no trabajar desde el comienzo con la base de datos tiene su explicación en que en un principio se había planteado trabajar con datos ficticios, una red básica de paradas y líneas de autobús y un simulador que proporcionase los datos de sus posiciones. Durante los meses de desarrollo, más concretamente en abril de 2008, TUZSA lanzó el servicio "*¿Cómo ir? a*", que ofrecía información sobre las paradas y líneas de autobús. Poco después y de forma extraoficial a través de contacto personal, fue posible incorporar las posiciones GPS de cada poste, por lo que la base de datos se convertía en una realidad.

La Figura 9 muestra la planificación de las tareas realizadas.

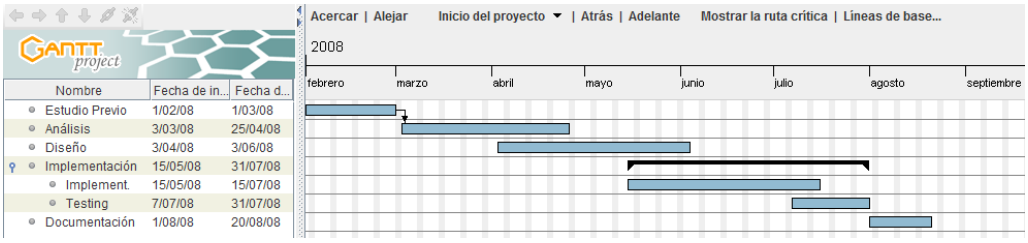

Figura 9: Diagrama de Gantt de la planificación estimada

Figura 10: Diagrama de Gantt de la ejecución real

La planificación inicial (véase Figura 9) difiere de la ejecución real del proyecto (véase Figura 10) en que, debido a cambios en los servicios que TUZSA ofrecía, se tuvo que volver a realizar un diseño de la solución, para prescindir del simulador. Además, la fase final del desarrollo, en concreto el componente GUI, fue pospuesto en el tiempo por motivos personales, lo que fue nuevamente condicionado por el avance de la tecnología: si en un comienzo se contemplaba como dispositivo móvil un PDA, en la actualidad este tipo de dispositivos prácticamente ha desaparecido dando paso a los

smartphones o teléfonos inteligentes con su propia API de desarrollo que ha propiciado el desarrollo y uso de aplicaciones nativas.

Si además de la planificación analizamos los esfuerzos realizados, éstos difieren ligeramente de los tiempos estimados en la planificación, condicionados por tratarse de un entorno de trabajo completamente nuevo, y por los constantes cambios en la tecnología y los servicios.

Figura 11: Gráfica de esfuerzos realizados en la realización del proyecto.

El hecho de que la fase final de implementación coincidiera con el periodo estival también contribuyó a una dedicación menor al proyecto, por lo que, aunque se alarga bastante en el tiempo, el esfuerzo realizado no es tan notablemente mayor.

4.2 Incidencias

A lo largo del desarrollo de este PFC ha habido numerosas incidencias que afrontar, problemas que resolver y retos que superar.

Se ha encontrado bastante dificultad para trabajar en un entorno completamente nuevo como son las tecnologías móviles. Cuando comenzó este proyecto en febrero de 2008, las tecnologías móviles estaban experimentando continuas transformaciones. Desde hacía varios años que habían visto la luz PDAs, algunas de

ellas, muy pocas, con teléfono integrado. El primer iPhone fue puesto a la venta en EEUU en junio de 2007, en Europa (Reino Unido y Alemania) en noviembre de 2007 y en la primavera de 2008 en España. En marzo de 2008 se liberalizó el SDK para iOS (versión Beta) y en verano fue oficialmente lanzada. Fue entonces cuando comenzó a desarrollarse aplicaciones nativas para iPhone, hecho que se vería fuertemente incrementado con la aparición del iPad en enero de 2010 el incremento. Similar evolución en cuanto a dispositivos y entornos de desarrollo han seguido otros fabricantes, llegando en la actualidad a cubrir más del 90% de mercado de los teléfonos móviles 2 plataformas: iOS y Android.

Cuando se planteó la realización de este proyecto los únicos dispositivos móviles que satisfacían los requisitos para ser utilizados eran las PDAs. En concreto se trabajó con el modelo de PDA Pocket PC, Fujitsu Siemens, con GPS integrado y con Windows Mobile 6 como sistema operativo. Se trataba de un dispositivo de alta gama para la tecnología existente en aquel momento.

Para poder ejecutar aplicaciones en el dispositivo, era necesario instalar una JVM (Java Virtual Machine). En este casó se instaló IBM WebSphere Everyplace Micro Environment for J2ME. Se instalaron también diversos navegadores Web, entre los cuales destacaba Opera Mobile por ser el que mejor se adaptaba a la mayoría de las páginas Web. Sólo hacer funcionar un pequeño programa en la PDA llevó una carga de trabajo considerable para lo simple que pueda parecer, condicionado en parte por la escasa documentación existente para algunas tecnologías más aún cuando cada dispositivo y cada librería se desfasaba (que no actualizaba) en cuestión de meses.

El entorno de tecnología cambiante, no sólo en los dispositivos móviles sino también en los servicios que aparecen, y el auge de las aplicaciones móviles han marcado el rumbo de este proyecto. Como incidencias cabe destacar la aparición del servicio de TUZSA "*¿Cómo ir? "*, que aunque fue muy beneficioso para el proyecto, supuso un cambio en el diseño a los pocos meses de empezar el proyecto.

Además de los factores relacionados con la tecnología y el entorno, también la propia definición del proyecto ha propiciado ciertas incidencias, especialmente en la parte algorítmica por trabajar con un grafo tan complejo. El hecho de ser un proyecto sin un diseño establecido ni condicionante alguno también ha supuesto en ocasiones líneas de desarrollo que luego han sido abandonadas por considerarse innecesarias.

4.3 Análisis de riesgos

A pesar de que pensamos que es un proyecto muy interesante desde un punto de vista tanto funcional como innovador, BUSCOBUS presenta diferentes riesgos asociados al contexto tecnológico en el que se enmarca. A continuación se detallan algunos de ellos que sin duda condicionan el desarrollo de este proyecto:

- La tecnología disponible y su continua transformación han condicionado muchas de las decisiones de este proyecto, en una etapa marcada por el *boom* tecnológico en cuanto a generaciones de dispositivos móviles y sus funcionalidades se refiere.

- No solo la tecnología y las funcionalidades, también la nomenclatura en la tecnología, los servicios, las plataformas o las metodologías sufre constantes cambios, sumergidos en la moda que a veces es implantada por las compañías dominantes. Esto supone también un desfase en la documentación.

- El mercado diversificado de los dispositivos móviles no permite trabajar con estándares y condiciona con sus éxitos y fracasos al resto de fabricantes. Cualquier actualización necesaria en una aplicación en el lado del cliente supone realizar varias modificaciones para las distintas plataformas.

- Finalmente, es necesario nombrar la saturación que sufre el mercado de aplicaciones para dispositivos móviles: Apple Store superaba, en marzo de 2013, las 800.000 aplicaciones. Muchas de estas aplicaciones, algunas de ellas de dudoso interés, se convierten en un fenómeno viral en pocos meses. El sector del marketing y posicionamiento de servicios en este sector está todavía poco desarrollado con respecto a su homólogo en la Web.

4.4 Ampliación y líneas futuras

Son muchas las posibilidades que ofrece este PFC en cuanto a futuros desarrollos tanto a nivel de optimización del sistema desarrollado como de nuevas funcionalidades. A continuación, se describen algunas de ellas que no se han implementado por no considerarse dentro del alcance de este proyecto.

Optimización de la aplicación:

- El almacenamiento de los datos del grafo en la matriz de adyacencia puede implementarse con otra estructura de datos que no sea una matriz. La razón es que el índice de ocupación de la matriz es muy bajo (la mayor parte de las paradas no están conectadas con el resto), por lo que una estructura de listas ocuparía menos espacio en memoria.

- Los datos estáticos de tiempos entre dos paradas en cada línea podría actualizarse según las estadísticas, permitiendo incluso hacer distinciones entre diferentes horarios y épocas, para obtener un resultado más preciso.

- Para gestionar una posible sobrecarga del sistema podría plantearse un diseño basado en Cloud computing (arquitectura distribuida con gestión remota) que replicase la infraestructura del servidor con un nodo frontal actuando de balanceador de carga hacia el resto de servidores. Se podría establecer un servicio de monitorización para detectar y prevenir estadísticamente (horas punta) las sobrecargas del sistema y hacer el despliegue de máquinas virtuales con suficiente tiempo de antelación a las sobrecargas.

Nuevas funcionalidades:

- Modo navegador, con visualización en perspectiva mediante Google Street View[2], con seguimiento de instrucciones desde el origen al destino.
- Incorporación de líneas especiales: búho bus (noches), refuerzos (Valdespartera, Interpeñas, cementerio) e incorporación de conexiones con otros medios de transporte: tranvía, red de cercanías o la red Bizi de Zaragoza.
- Registro de los usuarios para añadir más valor a las estadísticas y ofrecer más opciones de configuración.
- Adaptación a otras ciudades, haciendo ligeras modificaciones y añadiendo una nueva base de datos.

4.5 Valoración personal

A nivel académico y profesional este proyecto ha supuesto un reto por tratar un entorno hasta entonces desconocido y ser a la vez una aplicación clara, cercana y atractiva. BUSCOBUS podría definirse como un proyecto muy completo, que si bien no profundiza en exceso en ningún ámbito, sí hace un repaso por muchas de las principales líneas académicas que se siguen en la carrera de Ingeniería Informática. Otro punto a valorar es que se ha partido de cero, ya que no es continuación de ningún trabajo previo, ni debe ser integrado con ningún otro módulo, lo que da mucha libertad a la hora del diseño, pero a veces también hace difícil encaminar el trabajo y ceñirse a objetivos concretos, siendo uno mismo el que pone los límites.

En el ámbito personal, se puede resaltar que el proyecto fue elegido por su interés y por su novedad, al margen de su complejidad y de los temores a las tecnologías desconocidas, buscando la motivación como arma de trabajo. Los primeros meses de proyecto fueron duros debido a la novedad del entorno, sensación que fue calmándose con el paso del tiempo. Unos meses más tarde, ya incorporada al mundo laboral y tras no recibir respuesta exitosa de las entidades interesadas, comenzó una época de menor motivación, que en parte ha condicionado el retraso de su presentación.

2 http://maps.google.com/intl/en/help/maps/streetview/#utm_campaign=en&utm_medium=van&utm_source=en-van-na-us-gns-svn

Bibliografía

(1) Lázaro Issi Camy. JAVASCRIPT. ANAYA, Julio 2002.

(2) Biggs, N.L.: Matemática Discreta. Vicens Vives, Barcelona 1994

(3) W. Kocay, D. Kreher: "Graphs, Algorithms and Optimization". Chapman & Hall/CRC, 2005

(4) Hernández, G. : "Grafos. Teoría y algoritmos". Facultad de Informática. UPM. 2003

(5) Artificial Intelligence. A Modern approach. S. Russel, P. Norvig. Pearson. Third Edition, 2010.

(6) Rumbaugh J, Jacobson I, Booch G (2007). El Lenguaje Unificado de Modelado. Manual de Referencia, 2ª edición. Pearson Educación (Addison Wesley), Madrid.

(7) [1] R.Elmasri, S.Navathe: “Fundamentos de Sistemas de Bases de Datos”. Pearson Educación Madrid 2007, 5ªed. ISBN: 9788478290857.

(8) [2] A. Silverschatz, H.F. Korth, S. Sudarshan: “Fundamentos de Bases de Datos”. McGraw-Hill/Interam.de España 2006, 5ªed. ISBN: 9788448146443.

(9) *Diseño de interfaces de usuarios : estrategias para una interacción persona-computadora efectiva*. Ben Shneiderman, Catherine Plaisant. Pearson Educación, 2006

(10) Nielsen, Jakob. *Usabilidad : diseño de sitios web*. Traducción de Santiago Fraguas . Madrid [etc.] : Prentice Hall

(11) Ralph Stair and George Reynolds, Fundamentals of Information Systems, 6th ed.

(12) Macián Senz, Eva. *Desarrollo de un sistema de geolocalización web mediante el uso de terminales móviles con tecnología GPS.* Proyecto Final de Carrera. Universidad de Zaragoza, 2008.

(13) Moreno Gonzalo, Yolanda.: *Sistema de Control y Simulación de Tráfico*. Proyecto Final de Carrera. Universidad de Zaragoza, 2001.

Links

(14) http://dev.mysql.com/doc/

(15) http://www.sqlite.org/

(16) http://www.java.com/

(17) http://www.w3schools.com/html/

(18) http://courses.coreservlets.com/Course-Materials/csajsp2.html

(19) http://www.tuzsa.es/

(20) http://www.tomtom.com/es_es/

(21) http://www.garmin.com/en-US

(22) www.ibm.com/Websphere

(23) www.j2mepolish.org/

(24) http://jcp.org/en/home/index

(25) https://developer.apple.com/devcenter/ios/index.action

Anexos

Anexo A

Estudio Previo

Esta sección recoge la labor que se llevó a cabo durante el primer mes trabajo. Los principales objetivos de este periodo fueron:

- Entrar en contacto con el dispositivo móvil, una PDA Pocket PC con GPS integrado, un dispositivo muy avanzado en la época en la que se planteó el proyecto. Este primer contacto se realizó con el fin de analizar las diferentes funcionalidades que tenía y las limitaciones que prestaba. Estudiar qué máquina virtual instalar, qué clientes ligeros de bases de datos existían y cuáles se adaptaban mejor a una posible integración de todos componentes, y qué navegadores Web disponían de versión para móvil y cuáles eran los que mejores resultados ofrecían.

- Conocer el entorno de los Sistemas de Información Geográfica: familiarizarse con la nomenclatura, herramientas disponibles para análisis geográficos, funcionamiento del GPS y sus limitaciones.

- Investigar sobre proyectos similares en el mercado, no tanto para analizar nichos de mercado como para analizar la forma en que se habían planteado los diferentes problemas a resolver: pathfinders para excursiones en la naturaleza o navegadores tipo tom-tom para coches.

- Recoger características de usabilidad de aplicaciones similares y poder evaluar qué características son imprescindibles en un contexto móvil.

Esta información ha sido posteriormente tenida en cuenta, tanto en el análisis, el diseño y como en la implementación.

A.1 Sistemas de Información Geográfica

Un Sistema de Información Geográfica (GIS) es un conjunto integrado de componentes hardware, software y datos geográficos, con el fin de capturar, almacenar, editar, compartir y analizar datos geográficamente referenciados para resolver problemas de previsión medioambiental o planificación. De forma más general, los Sistemas de Información Geográfica pueden considerarse herramientas que permiten crear consultas sobre información espacial a través de mapas y datos estructurados.

Los GIS tiene multitud de aplicaciones, especialmente en diferentes campos de investigación como la arqueología, al gestión de recursos, la planificación urbana, la

geografía histórica, la logística o el impacto ambiental. La geocodificiación, uno de los usos más extendidos de los GIS, es la asignación de coordenadas geográficas (latitud-longitud) a puntos de mapa, proceso muy utilizado para traducir direcciones postales en coordenadas y viceversa, proceso conocido como georreferenciación.

Existen numerosas aplicaciones de software para consultar y tratar información geográfica, algunas de ellas propias de empresas como ESRI, MapInfo, Bentley Systems o Autodesk. El manejo de este tipo de software es realizado por personal especializado con experiencia y conocimiento en Sistemas de Información Geográfica como cartografía o geografía.

Para el uso de público general Google Earth es la herramienta más extendida. Otros servicio más sencillo que Google Earth es Google Maps, un servicio gratuito de mapas a través de la Web que permite desplazarse, visualizar fotos satelitales o establecer rutas o incluso posicionarse a pie de calle con Google Street View. En junio de 2005 Google lanzó su API de Google Maps, con el fin de que los usuarios fueran libres de modificar y customizar la interfaz. Con la contraseña de desarrollador, la API es de uso libre para cualquier sitio Web.

A.2 Proyectos y servicios relacionados

Al estar enfocado este proyecto para tener aplicación en Zaragoza, vamos a realizar el estudio de productos y servicios en la ciudad de Zaragoza, haciendo después un pequeño análisis de otros proyectos de ciudades de mayor envergadura como pueden ser Madrid o Barcelona.

Cuando comenzó este proyecto en 2008, prácticamente no había ningún servicio relacionado con este PFC, al menos realizado con rutas de autobús. Unos meses después comenzaron a aparecer servicios de información en tiempo real, como es el caso del servicio ¿Cuando llega mi bus? De TUZSA.

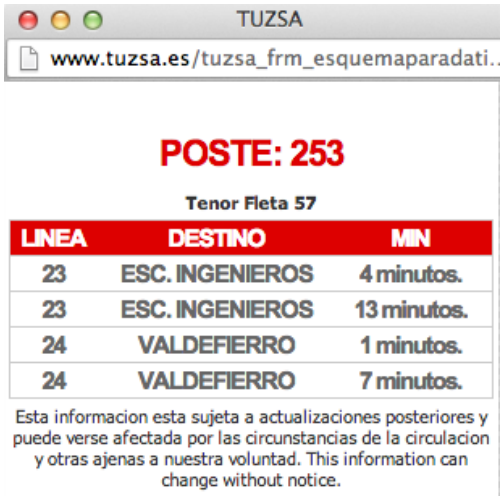

Figura 12: Estimación de tiempos de llegada de las líneas del poste 253

La Figura 12 muestra la estimación de tiempos en la parada o poste 253, para las líneas 23 y 24.

Este servicio se basa en un flota de autobuses equipados con GPS que envían su geolocalización a la central de TUZSA para su monitorización siendo esos datos son los que se muestran en las paradas. Su aplicación Web realmente no tenía mayor interés que mostrar para un poste dado la misma información que si estuvieses en una parada que tiene visor de estimación de tiempos. Poco después ofrecían la posibilidad de recibirlo por mensaje de texto, enviando el código de la marquesina previamente.

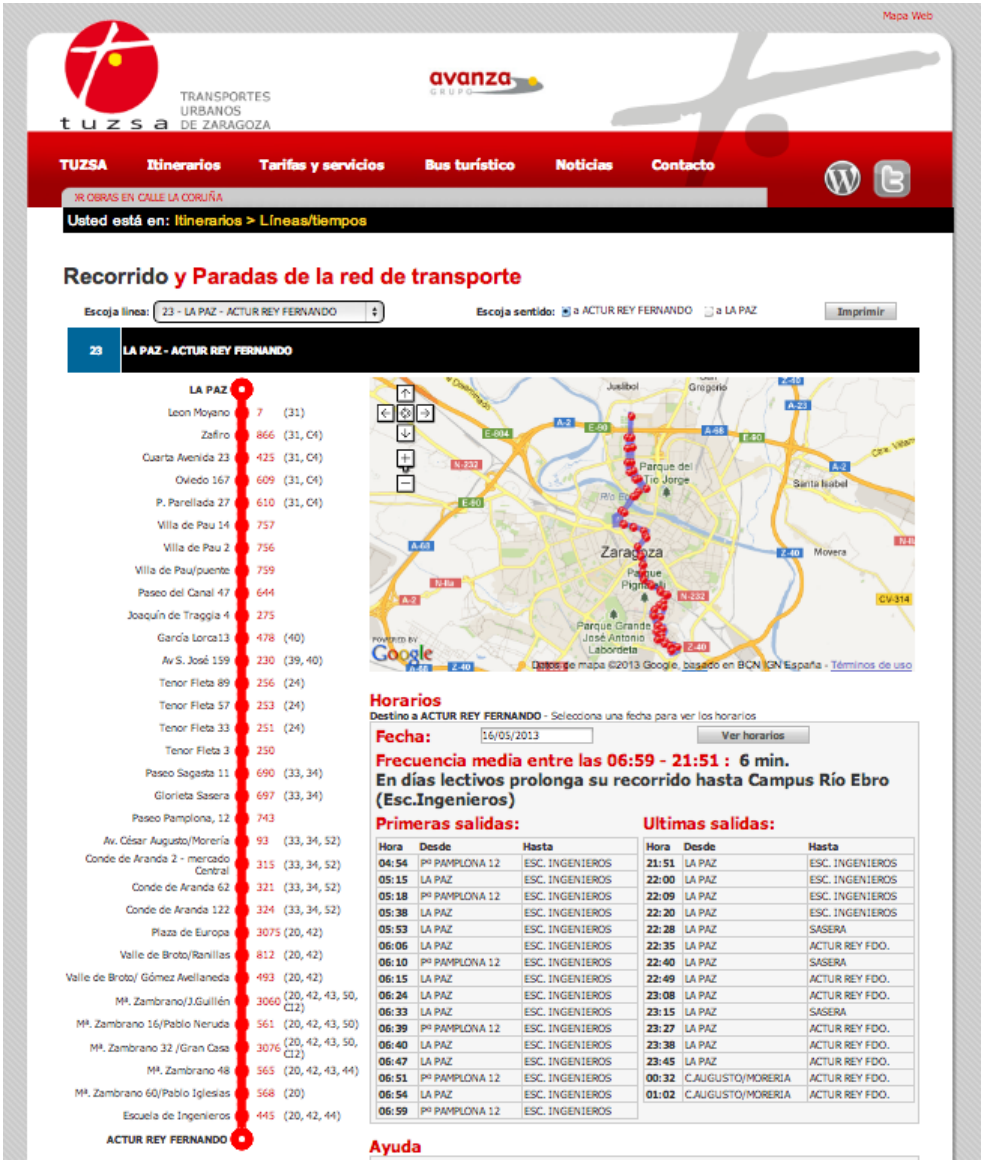

Figura 13: Servicio "¿Cuanto tarda mi bus?" de TUZSA.

Por otro lado, en TUZSA también dispone en su página Web del sistema “¿Como ir ?” que muestra la combinación optima de rutas de autobús para dos direcciones postales dadas, optimizando por tiempo o por número de transbordos. Se puntualiza “direcciones postales dadas” porque es bastante sensible a direcciones poco específicas como “Escuela de Ingenieros”, Parque Grande o incluso Plaza San Francisco si no le ponemos el número. Una aplicación con estas restricciones es difícilmente portable a un entorno móvil.

Figura 14: Servicio ¿Cómo ir? de TUZSA

Si se analizan los dos servicios, este PFC es una integración de estos los sistemas que ofrece TUZSA. Es la suma de los datos dinámicos en tiempo real que ofrece “¿Cuándo llega mi bus?” con los datos estáticos que sirven para calcular la ruta óptima en “¿Cómo ir?”.

En la primavera de 2008, un PFC de Ingeniería de Telecomunicaciones de la Universidad de Zaragoza [12] recogía los datos de un dispositivo WiFi con GPS y lo mostraba sobre un mapa. Este PFC ser realizó en la empresa Iternova con la que también se estudió una posible colaboración.

Otro PFC, en este caso de Ingeniería Informática realizaba el análisis de flujo de tráfico a través de un simulador [13].

En un ámbito más extenso podemos encontrar otras aplicaciones que trabajan con medios de transporte y que ofrecen información útil a los usuarios como es el caso de la aplicación de Madrid y Barcelona para metro, ampliamente utilizada.

Anexo B

Tecnología utilizada

BUSCOBUS ha sido un fuerte proyecto condicionado por la tecnología hasta el punto de ser determinante en el diseño de la solución y

B.1 Tecnología Software

La siguiente tabla recoge todo el software utilizado desde el comienzo del PFC, tanto para entorno de desarrollo en sobremesa como en entorno móvil.

	Entorno Sobremesa	**Entorno Móvil**
Sistema Operativo	Windows XP Macintosh Snow Leopard and Lion.	Windows Mobile iOS
Entorno de desarrollo	Eclipse, con el IDE de Java, Java Development Toolkit (JDT) + Java Platform, Enterprise Edition (Java EE)	Java Platform, Micro Edition (Java ME or J2ME) + IBM WebSphere Everyplace Micro Environment XCode + Objective C
Base de datos	MySQL MySQLAdmin	SQLite
Web Services	Tomcat Applets, Servlets JSON, Javascript HTML, CSS, API GoogleMaps	HTML, CSS,
Navegador	Chrome Firefox	Opera Mini Explorer
Documentación	GoogleDocs AdobePDF Reader OpenOffice Microsoft Office Photoshop Ilustrator	

Tabla 2: Tecnología software utilizada

B.2 Tecnología Hardware

En el entorno de trabajo, el servidor ha quedado instalado en el equipo personal de la autora, un iMac, 2,5 GHz Intel Core i5, 4GB 1333MHz DD3, pudiendo ser alojado en un servidor comercial en un futuro.

Para la realización de este proyecto se ha trabajado inicialmente con una PDA Pocket PC, Fujitsu Siemens LOOX N560, con procesador Intel 624 Mhz y con GPS integrado. Posteriormente con iPhone 4.

Anexo C

Análisis y Diseño de la solución

Este apartado tiene como finalidad presentar los diferentes planteamientos que se realizaron en cuanto a arquitectura y diseño de la solución.

C.1 Evolución en la Arquitectura y diseño

Tras realizar la fase de estudio previo, y siguiendo la tendencia de los dispositivos móviles y navegadores del momento, se planteó un primer diseño basado en una arquitectura del sistema con todos los elementos del sistema integrados en una aplicación nativa para PDA. El siguiente gráfico muestra un esquema del planteamiento:

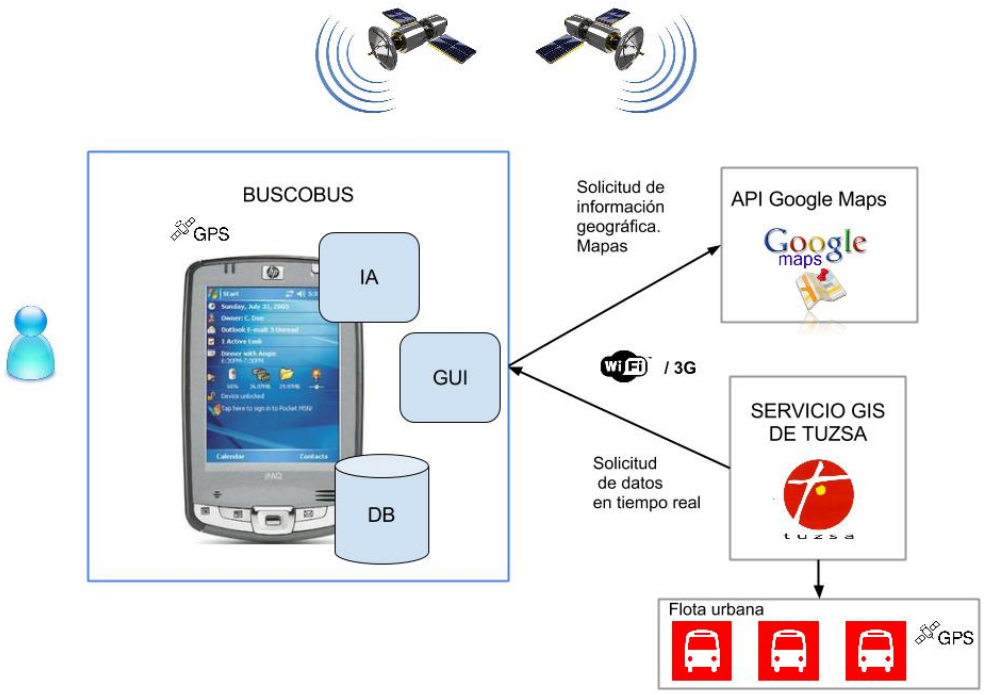

Figura 15: Primer diseño de solución

En esta arquitectura se consideran los siguientes aspectos, tanto positivos como negativos.

Aspectos positivos:

- Su funcionamiento no depende de que otros usuarios estén accediendo al servicio y puedan saturar el servidor, teniendo en cuenta que se realiza un cálculo de búsqueda en grafo.
- La base de datos ligera para PDA se adapta bien a las necesidades de BUSCOBUS ya que se trata de una base de datos muy ligera.

Aspectos negativos:

- Para un diseño en PDA, suponía consumir demasiados recursos del sistema: red, procesador y memoria.
- A pesar de estar integrado, sigue teniendo dependencia de servicios externos: GoogleMaps y la página de TUZSA.
- Dificultad en la creación de interfaces amigables *basados en las librerías de J2ME.*
- Entorno de desarrollo complejo para tener que desarrollar un algoritmo de búsqueda.
- Aplicaciones nativas para PDA con tendencia al desuso. Tendencia a la integración de servicios. Inicios del SaaS, tecnología Cloud Computing.
- Si todo el sistema se aloja en el dispositivo, cualquier actualización en tecnologías o software (muy frecuentes en estos aparatos) supone una readaptación de todo el desarrollo.
- Desarrollo a medida para cada dispositivo.

El segundo diseño planteado tiene el mismo planteamiento que el diseño final pero con uso de una PDA. Esto sugiere algunos cambios con respecto al diseño con un smartphone: la utilización de applets en el lado del cliente con el fin de tener acceso acceso al GPS integrado, imposible realizar a través de una simple conexión al servicio.

Este diseño también presenta una serie de ventajas y desventajas que enumeramos a continuación.

Aspectos positivos:

- Simplicidad en el desarrollo.
- Arquitectura modular permite no tener que modificar todo el sistema cuando hay actualizaciones.
- No satura los recursos de la PDA.
- Mayor cantidad y mejor calidad en la documentación.
- Sin instalaciones. Accesible a través de un navegador.
- Al ser accesible desde navegador, el interfaz se basa en una página Web adaptada a móvil.

Aspectos negativos:

- Sobrecarga del sistema: su funcionamiento puede depender de que un número elevado de usuarios estén accediendo al servicio y puedan saturar el servidor, teniendo en cuenta que se realiza un cálculo de búsqueda en grafo.
- Dificultad en la adaptación de la aplicación Web a cada versión de navegador.
- Tendencia actual a las aplicaciones nativas, aunque muchas de ellas con conexión a servidor.

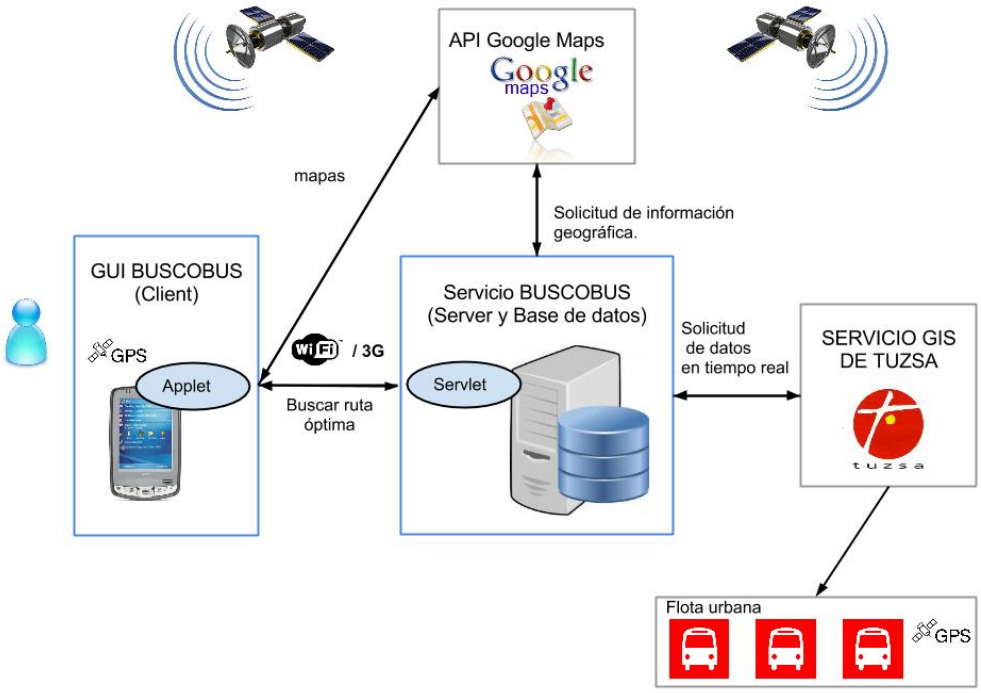

Figura 16: Segundo diseño de solución

En la Figura 16 se muestra el diseño final de la arquitectura de BUSCOBUS. Este diseño está basado en un un servicio de información que consta de los siguientes elementos: una aplicación móvil para instalar en un dispositivo móvil o smartphone con GPS y conexión a Internet, y un servidor al que se conecta y que dispone de la lógica de control para acceder a la Base de datos, lanzar el algoritmo de búsqueda de la *Inteligencia Artificial* y conectar con los servicios externos de Google Maps y TUZSA. Es la evolución del sistema anterior adaptado a los nuevos dispositivos, lo que permite prescindir de los applets y servlets ya que la aplicación nativa es la encargada de la lectura de las coordenadas GPS del dispositivo.

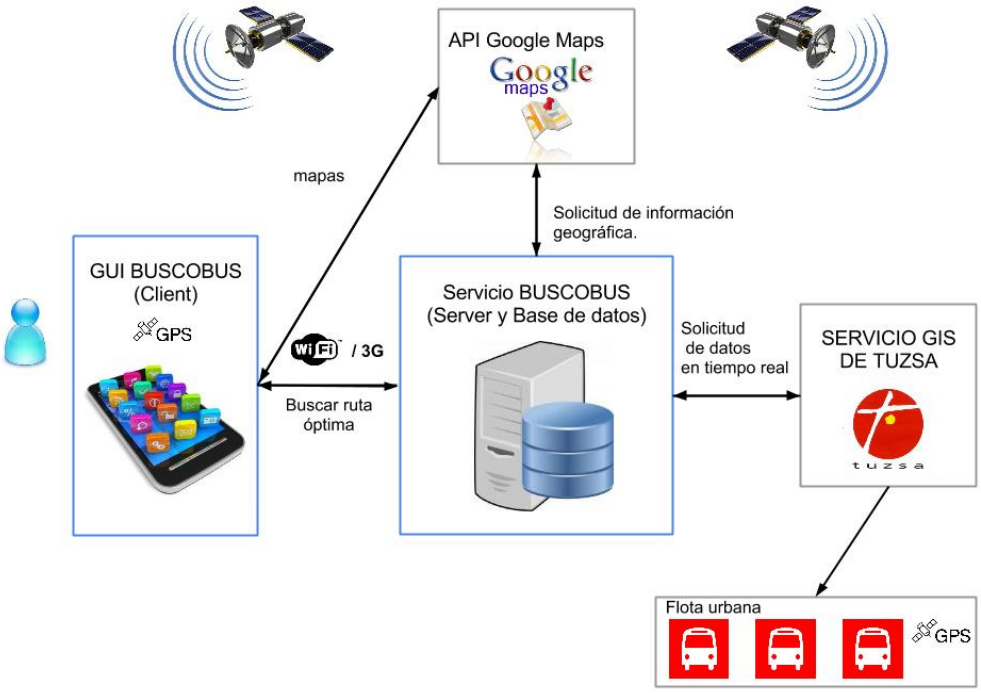

Figura 17: Diseño final de la solución

Aspectos positivos:

- Simplicidad en el desarrollo.
- Arquitectura modular permite no tener que modificar todo el sistema cuando hay actualizaciones.
- No satura los recursos de la PDA.
- Herramientas disponibles para desarrollar interfaces amigables.
- Mayor cantidad y mejor calidad en la documentación.
- Tendencia actual del mercado

Aspectos negativos:

- Sobrecarga del sistema: su funcionamiento puede depender de que un número elevado de usuarios estén accediendo al servicio y puedan saturar el servidor, teniendo en cuenta que se realiza un cálculo de búsqueda en grafo.
- Saturación de mercado
- Actualizaciones de software constantes

C.2 Análisis y diseño de GUI

El Interfaz Gráfico de Usuario (GUI) es un programa que actúa de interfaz de usuario utilizando un conjunto de imágenes y objetos gráfico para mostrar información y poder ejecutar acciones disponibles a través de la interfaz. Su principal objetivo es proporcionar un entorno visual sencillo, simple y fácil de manejar. Para el diseño de un buen GUI debe tenerse en cuenta tanto el equipo o sistema para el que se va a desarrollar, como las necesidades de los usuarios a los que se va a dirigir.

C.2.1 Interfaz de usuario

Se describe el interfaz de usuario como medio con el que un usuario puede comunicarse con una equipo o sistema. Este concepto recoge tanto elementos como menús, ventanas, teclado o ratón o incluso sonidos. El interfaz de usuario tiene por lo tanto la finalidad de ofrecer la interacción hombre máquina.

En el caso de las aplicaciones móviles el interfaz de usuario cobra todavía mayor relevancia ya que introduce nuevos parámetros o situaciones que condicionan el uso del dispositivo. A continuación se detallan algunos de ellos que tienen implicación directa en el diseño de un interfaz de usuario apropiado:

Entorno del dispositivo:

- El tamaño de la pantalla es mucho menor que un entorno Web.
- La conexión a Internet puede ser limitada.
- La capacidad de almacenamiento y procesamiento puede ser limitado.
- El dispositivo produce sonidos con frecuencia (llamadas, alertas, mensajes).
- El software del sistema operativo se actualiza con relativa frecuencia.

Entorno de usuario

- El usuario puede estar en movimiento.
- El usuario puede estar a la luz del día, lo que dificulta la lectura de la pantalla.
- El usuario puede cambiar de dispositivo móvil con frecuencia (aproximadamente cada 2-3 años).
- El usuario puede usar la aplicación en un contexto de urgencia (necesita optimizar en tiempo).

La evolución en el diseño de GUI en los último años ha evolucionado considerablemente. En el caso del diseño Web móvil, ha pasado de desarrollarse para cada tipo de navegador, para cada tipo de dispositivo (pc, tablet, móvil), a basarse en un diseño único, fluido y adaptable al tamaño de la pantalla del dispositivo. Esta nueva tendencia es conocida como *Responsive Web Design*.

En el caso de las aplicaciones móviles el diseño debe ser simplificado al máximo. No sólo es necesario que se visualice bien en una pantalla pequeña, también es necesario que sea una aplicación ligera y con funcionalidades básicas. Muchas de los

servicios Web que disponen además de aplicación para dispositivo móvil limitan mucho la funcionalidades y acortan la navegación.

C.2.2 Navegación

Teniendo en cuenta el análisis de la sección anterior y tras estudiar aplicaciones similares para smartphone se han definido las siguientes funcionalidades para la aplicación BUSCOBUS.

Pantalla de Inicio:

- Búsqueda
- Historial: Muestra las últimas 5 búsquedas

Pantalla mapa:

- Recalcular
- Texto

Otras opciones han sido analizadas para ofrecer más funcionalidades, como la configuración de la aplicación con preferencias de usuario, el historial o la versión navegador (perspectiva), que serían accesibles desde la pantalla principal y desde la pantalla de resultados respectivamente.

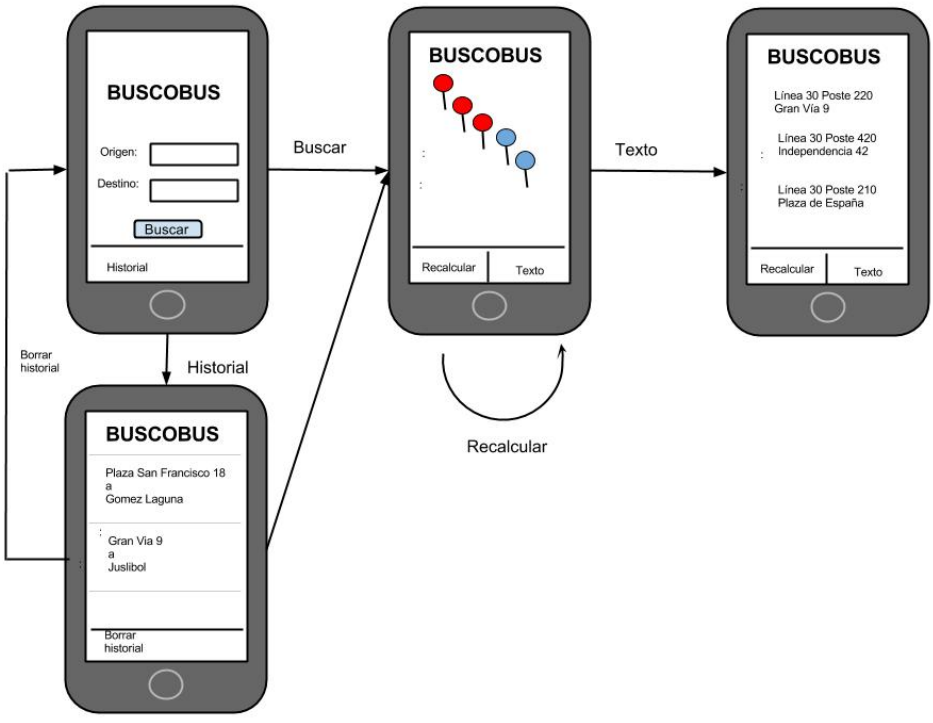

Figura 18: Diseño de navegación en la Aplicación BUSCOBUS

Anexo D

Implementación

Como anexo a la Sección 3, se complementa la información relativa a la implementación con una descripción de la base de datos y con el código fuente del algoritmo de búsqueda en grafo A*

D. 1 Tablas de la base datos

```
+-----------------------------------+
Tabla Linea
+-----------------------------------+
CodLinea              int (5)
nombre                varchar (4)
frecuencia            decimal (10,0)
paradaOrigen          varchar (50)
paradaDestin          varchar (50)
+-----------------------------------+

+-----------------------------------+
Tabla Parada
+-----------------------------------+
poste                 int(5)
nombre                varchar (50)
calle                 varchar (50)
numero                int (7)
longitud              decimal (10,7)
latitud               decimal (10,7)

+-----------------------------------+
Tabla Ida
+-----------------------------------+
orden                 int (11)
codLinea              int (5)
poste                 int (5)
tpoAnterior           time

+-----------------------------------+
Tabla Vuelta
+-----------------------------------+
orden                 int (11)
codLinea              int (5)
poste                 int (5)
tpoAnterior           time
+-----------------------------------+
```

D.2 Código Fuente Algoritmo A*

```
package graph;

import java.util.ArrayList;
public class AStar{

// ************** Variables ***********
protected Graph graph;
protected Open open;
protected Closed closed;
protected Sucessors sucessors;
protected Node start;
protected Node end;
protected boolean modeStatic;
protected AdjacenceMatrix matrix;
protected int numPath;
protected PathList pathList;
protected ArrayList endPath;
protected Map map;
protected int PENAL1 =5;
protected int PENAL2 =20;
protected int PENAL3 =100;
protected int INI =-2; /* Id del nodo origen*/

private float f;
private float g;
private float p;



// ************** Public methods ***********
public AStar(Graph graph, AdjacenceMatrix matrix){//,Open open,Closed closed) {
        this.graph= graph;
        this.matrix = matrix;
        this.open = new Open();
        this.closed = new Closed();
        this.sucessors =  new Sucessors();
        this.map= new Map();
        this.endPath = new ArrayList();/
        this.pathList = new PathList();
}

public void setModeStatic (boolean mode){
        this.modeStatic=mode;
}


public void setNumPath (int num){
        this.numPath=num;
}

public PathList findPath(Node start,Node end) throws Exception{
        this.start=start;
```

```
        this.end=end;
        boolean goalReached = false;
        this.modeStatic=false;

        init();
        initNodes();
         int iter=0;

        while((!open.isEmpty()&& !goalReached) && iter<70) {
                iter++;
                open.data();
                Node best = new Node();
                best = open.smallestCostNode();
                open.removeSmallest();
                closed.add(best);
                if(best.id==end.id) {
                        goalReached = addPath(best);
                        if (goalReached)
                                break;
                }
                else { // Goal not reached
                        exploreBest(best);
                }
        }//while

        if(goalReached) {
                for (int i=0; i<endPath.size();i++){
                        Path path = new Path();
                        Node current = (Node) endPath.get(i);
                        while(current.getId()!=start.getId()) {
                                path.insert(current);
                                current = closed.getParent(current);
                        }
                        path.insert(current);
                        pathList.add(path);
                }
                return pathList;
        } else {
                return null;
        }
}

private void init(){
        open.clear();
        closed.clear();
        sucessors.clear();
        endPath.clear();
}


private void initNodes()
{/*Introducimos start en abiertos*/
        start.h=(map.estimate(start,end));
        start.data("START");
        node node = new Node(start.id, start.getLine());
        node.setLat(start.lat);
        node.setLon(start.lon);
        node.setH(start.h);
```

```
        node.linesList = start.linesList;
        node.setStopWalkingList(start.getStopWalkingList());
        node.addLineBefore(0);
        open.add(node, node.getG(), null, node.getG());
}

private boolean addPath(Node best){
        Node last = new Node();
        last=best;
        boolean exist=false;
        for(int i=0; i<endPath.size();i++){
                if (((Node
        endPath.get(i)).getLinesBefore().equals(last.getLinesBefore()))
                                exist=true;
                 }
                 if (!exist)
                        endPath.add(endPath.size(), last);

                if (endPath.size()==numPath)
                        return true;
                else
                        return false;
}
private void exploreBest(Node best){
        sucessors.clear();
        sucessors.add (matrix , best);
        for (int i=0; i < sucessors.size();i++){
        // Establecemos los datos para cada poste sucesor
        Node next = new Node();
        next = (Node)Graph.getNodeByIndex(sucessors.getSuc(i).getId());
        if (next.getId()== start.getId())
                break;

        next.setH(map.estimate(next, end));
        next.setF(next.getG()+next.getH());
        next.setParent(best);
        next.setStopWalkingList(sucessors.getSuc(i).getStopWalkingList());
        next.setLinesBefore(best.getLinesBefore());
        if( !next.contains(0)) next.stopWalkingList.size()>0){
                next.linesList.add(0);
                // suponemos que si estan seguidas estan cerca
        }
        next.data("NEXT");

        if        (next.contains(best.line)||(!next.contains(best.line)        &&
best.line==0)){ //Lineas enlazadas
                for (int j=0; j<next.linesList.size();j++){
        // Establecemos los datos para cada parada (poste-linea) enlazada

                        if (best.line==0 && !next.contains(0) && best.getId()!=-
2){
                                break;
                        }
                        /*Actualizamos coste*/

                        float                        gCost                        =
matrix.get(graph.getNodeIndexById(best.id),sucessors.getSuc(i).getId()) ;
                        /*Casos admisibles para tomar la linea del sucesor*/
```

```
                        if (((Integer)next.linesList.get(j)==best.getLine())//&&
                        best.getLine()!=0) //sigue la linea
                        /*|| (best.line==0 &&next.linesList.get(j)!=0)// va a pie
a coger un bus*/

                        ||((Integer)next.linesList.get(j)!=best.line
                        &&best.contains((Integer)next.linesList.get(j)))//nueva
                                                        linea en el poste
                        || (Integer)next.linesList.get(j)==0 && gCost>0 &&
                        best.getLine()!=0) //sig. parada transbordo, pero no en
                                        el sentido contrario
                                {
                                        Node node = new Node(next.id,
        (Integer)next.getLineFromList(j)); //node(id, linea)
                                        if (gCost<0){
                                                if (best.line!=0){
                                                        break;
                                                }
                                                else{
                                                        gCost=-gCost;
                                                        node.setLine(0);
                                                        node.setWalking(true);
                                                }
                                        }

                                        node.setParent(best);
                                        node.linesList=next.linesList;
                                        node.setLat(next.getLat());
                                        node.setLon(next.getLon());
                                        node.setH(next.getH());
                                node.setLinesBefore(best.getLinesBeforeClone());

                                        g = best.getG() + gCost; //Actualizamos
                                                                camino recorrido
                                        p = testTransbordo(node, end);
                                        p += testUtility(node, end);
                                        node.setG(g);
                                        f = g + p+ node.getH();
                                        node.setF(f);
                                        int index = closed.containsNode(node);
                                                if (f<100){
                                                        if(index == -1) { //no esta
                                                                        en  cerrados
                        //ahora hay que mirar que ademas sea de la misma linea
                                                        if(!
open.update(node,f,best,g)) {
        open.add(node,f,best,g);
}
                                                }
                                                else{

                                                        if((g                       <
closed.getCostg(index)  &&  node.getLine()!=node.getParent().getLine())  ||  //si
transbordo que valga la pena

        (g>=closed.getCostg(index)&&
node.getLine()==node.getParent().getLine())) {
//si es la misma linea y es menor actualiza
```

```
        closed.remove(node);
        open.add(node,f,best,g);
                                                                    }
                                                                }
                                                            }
                                                        }//if
                                                    }//if
                                                }
                                            }//for
         }


private float testTransbordo(Node node1, Node end)
            {
        //Añadimos la nueva linea utilizada
        if (node1.getLine()!=node1.getLineParent()){
                node1.linesBefore.add(node1.getLine());
        }

        float penal = 0.0F;
        //TRANSBORDO BUS -> BUS, BUS ->PIE
        if(node1.getLine() != node1.getLineParent() && node1.getLineParent() !=
        0 && node1.getIdParent()!=INI)
        {
                penal = penalty(node1, end);
                if(node1.getLine()!=0)
                {
                        g += 5;
                }
                 else
                 {
                     if (node1.getLine()==0 && map.estimateWalk(node1,end)>5)
                        {
                        g++;
                     }
                 }
               }

        //TRANSBORDO PIE->BUS
                        if(node1.getLine()      !=      node1.getLineParent()      &&
node1.getLineParent()== 0 && node1.getIdParent()!=INI)//si no es al principio
                {
                penal = penaltyPieBus(node1, end);
        }


                 //TRANSBORDO PIE->PIE
        if(node1.getParent().getLine()  ==  0  &&  node1.getLine()  ==  0  &&
        node1.getParent().getId()!=INI) //si no es al principio
        {
                node1.g = node1.g + 2;
                penal += 3;
        }
        return penal;
}

public float testUtility(Node node, Node end){
```

```
        //Es el nodo final y ultimo transbordo innecesario?
        float penal=0;
        int befLastLine;
        if (node.linesBefore.size()>=3){
                //lastLine = node.linesBefore.get(node.linesBefore.size()-2);
                                                        befLastLine          =
(Integer)node.linesBefore.get(node.linesBefore.size()-3);

                if(node.getId()    ==    end.getId()    &&    node.getLine()    !=
node.getLineParent()     &&     end.contains(node.getParent().getLine())     &&
node.getParent().getLine() != 0){
                        penal= PENAL3+(float)2.2222222;
                    }

//comprobamos si el camino que se ha hecho hasta este momento se
//podia haber ahorrado un transbordo (hacer 20->30 si se llega con 20)

                if(node.getLine() != node.getLineParent() && node.getLine() > 0
                && node.getParent().contains(befLastLine)){
                        node.data("Camino estupido1");
                        penal= PENAL3+(float)1.11111111;
                }
        }
        if(node.linesBefore.size() >= 2){
                befLastLine                                                     =
(Integer)node.linesBefore.get(node.linesBefore.size() - 2);
                if(node.getLine() != node.getLineParent() && node.getLine() > 0
                && node.getParent().contains(befLastLine)){
                        node.data("Camino estupido2");
                        penal= PENAL3+(float)1.11111111;
                                                        }
                        if(node.getId() == end.getId() && node.getLine() !=
                        befLastLine  &&  node.contains(befLastLine)&&  befLastLine!
=0)
                {
                         penal = PENAL3+(float)3.3333333;
                }
         }
        return penal;
}


public float penalty (Node node, Node end){

        float p = 0;

        if (node.linesBefore.contains(node.getLine())&& node.getLine() != 0){
                p = p + PENAL3;
        }

        //por hacer mas de 3 transbordos
        if  (node.linesBefore.size()>3  ||  node.linesBefore.size()  ==  3  &&  !
        node.linesBefore.contains(0)){
                p = p + PENAL3*node.linesBefore.size();
        }

        //para evitar que escoja un nodo final como transbordo, excepto a pie
```

```
        if(node.getId()==end.getId()&&
        node.contains(node.getParent().getLine())&&
        node.getParent().getLine()!=0){
                p = p + PENAL3;
        }
                return p;
        }

public float penaltyPieBus (Node node, Node end){
        float p = 0;
        if  (node.linesBefore.contains(node.getLine())&&(node.getLineParent()  !=
0 || node.getId() != end.getId())){
                p = p + PENAL3;
        }

        //por hacer mas de 3 transbordos
        if  (node.linesBefore.size()  >  3   ||node.linesBefore.size()  ==  3  &&  !
node.linesBefore.contains(0)){
                p = p + PENAL3*node.linesBefore.size();
        }

        //hacer transbordo a pie si se podia hacer en bus
        float
link=matrix.get(graph.getNodeIndexById(node.getParent().id),graph.getNodeIndexB
yId(node.id));
        if  (node.contains(node.getParent().getLine())&&node.getLine()==0  &&  link
> 0){
                p = p + PENAL3;
        }
        return p;

}
```

Anexo E

Glosario

- **A***: Algoritmo de Búsqueda en grafo.
- **API**: Application Program Interface, interfaz de comunicación entre componentes
- software.
- **Applet:** Componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador
- **Geocodificación:** Proceso de asignar coordenadas geográficas (latitud-longitud) a puntos del mapa (direcciones, puntos de interés, etc.).
- **GIS**: Sistema de Información Geográfica
- **GPS**: Global Positioning System: sistema de posicionamiento global, basado en las coordenadas de latitiud y longitud, es un sistema global de navegación por satélite que permite localizar un objeto en cualquier lugar del planeta
- **HTML**: HyperText Markup Language («lenguaje de marcado hipertextual»): Lenguaje de marcado predominante para la elaboración de páginas web
- **HTTP**: Hypertext Transfer Protocol (Protocolo de transferencia de hipertexto): Protocolo usado en cada transacción de la World Wide Web
- **IDE:** Entorno de Desarrollo Integrado, es el entorno de trabajo compuesto por un conjunto de herramienta de programación
- **iOS**: Sistema Operativo desarrollado por Apple para el dispositivo móvil iPhone
- **JSON**: JavaScript object notation
- **KDE**: Kit Development Environment
- **GUI**: Graphical User Interface, o Interfaz Gráfico de Usuario es el entorno de un servicio que está directamente en contacto con el usuario y que se basa en un interfaz sencillo para favorecer la comunicación
- **xCode**: IDE de la compañía americana Apple para el desarrollo de aplicaciones en iOS para iPhone