ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Aproximación a la planificación de rutas en el transporte público
(Towards public transit trip planning)

Para acceder al Titulo de

Graduado en
Ingeniería de Tecnologías de Telecomunicación

Autor: Gabriele Montagnini

Octubre - 2015

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

CALIFICACIÓN DEL TRABAJO FIN DE GRADO

Realizado por: Gabriele Montagnini
Director del TFG: Jorge Lanza Calderón
Título: "Aproximación a la planificación de rutas en el transporte público"
Title: "Towards public transit trip planning "

Presentado a examen el día: 23 de octubre 2015

para acceder al Título de

GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Alicia Casanueva López
Secretario (Apellidos, Nombre): Jorge Lanza Calderón
Vocal (Apellidos, Nombre): Alberto Eloy Garcia Gutiérrez

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado Nº
(a asignar por Secretaría)

Agradecimientos

En primer lugar quería dar las gracias a mi tutor del trabajo de fin de grado, Jorge Lanza Calderón, por ofrecerme la oportunidad de trabajar en este interesante proyecto y, sobre todo, por esforzarse al máximo durante todo el curso para que consiguiéramos terminar el proyecto a tiempo.

Además, quiero agradecer a Alberto Eloy Garcia su disponibilidad por resolverme las dudas sobre teoría de grafos que me quedaban y por su ayuda cuando Jorge no estaba disponible. En general, quiero dar las gracias al Departamento de Telemática por el buen ambiente respirado en el laboratorio durante el tiempo pasado trabajando allí y la disponibilidad de todos los profesores del departamento que me han ayudado durante la carrera.

Gracias a todos los compañeros y amigos encontrados durante esos años de universidad, muchas horas han pasado estudiando codo a codo para los exámenes y otras tantas riendo y bromeando sobre ello. Seguro que voy a echar de menos estos momentos y la libertad de ser un estudiante universitario.

Muchas gracias a mi familia, primero a mis padres por ofrecerme la posibilidad de empezar una carrera en el extranjero. Gracias por el apoyo infinito que me han demostrado durante todos estos años, sé que no es fácil tener a un hijo tan lejos, aunque siempre lo hayan hecho todo considerando primero mi felicidad y gracias por conseguir estar cerca aunque no lo estén físicamente. Gracias también a mi hermana Claudia por el apoyo incondicional durante la carrera, por saber escuchar, entenderme en los momentos difíciles y por defenderme siempre cuando las cosas no iban en la dirección que yo quería.

Por último, quiero dar las gracias a Lucía por estar a mi lado todos estos años de la carrera. Sin ella hubiera sido imposible llegar hasta aquí consiguiendo todos los retos presentes durante el camino. Gracias por cuidarme siempre, por confiar en mí y ayudarme en los momentos difíciles.

A todos, muchas gracias.

Índice general

Índice de figuras

Índice de tablas

Acrónimos

API: Application Programming Interfaces

ASP: Active Server Pages

BFS: Breadth First Search

GIS: Geografic Information System

GPS: Global Positioning System

GTFS: Google Transit Feed Specification

HTML: HyperText Markup Language

HTTP: HyperText Transfer Protocol

JP: Journey planner

JSON: JavaScript Object Notation

JSP: JavaServer Pages

KSP: K-Shortest Path

MST: Minimum Spanning Tree

NPM: Node Package Manager

OTP: Open Trip Planner

REST: Representational State Transfer

RPC: Remote Procedure Call

SOAP: Simple Object Access Protocol

SQL: Structured Query Language

TSP: Travel Salesman Problem

TUS: Transporte Urbano Santander

UDDI: Universal, Description, Discovery and Integration

WSDL: Web Service Definition Language

XML: eXtensible Markup Language

Resumen ejecutivo

El transporte supone el traslado de personas o bienes de un lugar a otro. Los lugares de en los que se produce suelen estar alejados de aquellos en los que se consume; las personas que residen en una localidad, trabajan, estudian o disfrutan de su tiempo libre en otros emplazamientos. En un mundo cada vez más globalizado, el transporte es uno de los sectores que más aporta a la generación de riqueza de una región pues abre un gran abanico de posibilidades en todos los sectores productivos, industria, turismo, educación, ocio, etc.

Centrándose en el transporte de personas, el autobús, el tren o el metro se presentan como medios de locomoción que facilitan un transporte de garantías y tranquilo en el que los usuarios pueden disfrutar del trayecto, sin la tensión que genera los desplazamientos en el vehículo personal. Adicionalmente, el transporte público se considera el medio más ecológico para desplazarse, pues al compartir con otros usuarios un mismo vehículo se solidariza la emisión de gases, haciendo más sostenible el entorno.

El transporte público, compartido y solidario compensa el tiempo añadido que un traslado conlleva en estos medios esos intangibles; pero aporta otro elemento cada vez más valorado, el aprovechamiento del tiempo. No obstante, las estadísticas indican que el transporte público se usa principalmente, y casi de forma exclusiva, cuando no se tiene otra alternativa. Se hace por tanto necesario buscar opciones que permitan cambiar esa tendencia. Si el disfrute del tiempo para uno mismo o los suyos no es suficiente, habrá que tratar de analizar mejorar la eficiencia en el desplazamiento. Cada ciudadano debería conocer las alternativas para cubrir sus necesidades de desplazamiento al trabajo, estudios u ocio y compararlas con el uso del coche y descubrir los ahorros ya no solo en tiempo, sino también dinero y la mejora de salud con la reducción de estrés.

Los sistemas expertos de planificación de rutas están haciendo posible que la tendencia comience lentamente a cambiar y los ciudadanos identifiquen los diferentes medios de transporte público como una alternativa viable y sencilla de usar en su día a día, o durante los viajes de placer a entornos no conocidos. Este trabajo tiene como objetivo desarrollar una solución de planificación de rutas que permita la aplicación de diferentes factores y necesidades de los usuarios (coste, comodidad, tráfico, etc.) para determinar la mejor ruta para llegar de un punto a otro de una ciudad, y de esta forma en un futuro poder establecer una comparativa entre diferentes medios de transporte y/o la combinación de ellos. Para ello se plantea el empleo de la teoría de grafos, empleada en la planificación de redes de comunicación entre otras, como elemento base para determinar la ruta más adecuada para un desplazamiento. La solución que se obtenga se evaluará de forma experimental en el entorno real del transporte público de la ciudad de Santander.

Executive summary

Transportation assumes the movement of people and goods from one place to another. The places where these movements take place are usually distant from those where consumption occurs; people who live in one place, work, study or enjoy their leisure time in other locations. In a world increasingly globalized, transportation is one of the sectors that contributes to the economic success of a region since it provides a wide variety of opportunities in all productive areas, industry, tourism, education, leisure, etc.

Focusing on transportation of people, bus, train or metro, are presented as a means of transport which provide transportation with guarantees and serenity where passengers can enjoy their journey without the stress caused by driving their cars. In addition, public transport is considered the most ecological mean of transportation, since sharing one vehicle between users, reduces the global emission of gases, making a more sustainable environment.

Public transport, shared and solidary, compensate the increase of time in a transfer in the terms of intangibles means; but it contributes to an element each time more valuable, making the most of our time. Nevertheless, statistics show that public transport is mainly used when there is no other alternative. It is therefore necessary to search for other options to change this trend. If the enjoyment of oneÂ´s own is not enough, it should be necessary to analyze improvements in the efficiency of journeys. Each citizen ought to know alternatives to meet his needs in his journeys to work, school or leisure and compare them with the use of car and discover the savings, not only in time but in money and well-being due to the reduction of stress.

Expert systems in trip planning are making possible a slow change in this trend and that citizens identify the different means of public transport as a viable alternative and easy to use in their day-to-day, or during pleasure journeys to unknown environments. The objective of this project is to develop a solution to trip route planning which allows the implementation of different factors and user necessities (costs, commodity, traffic, etc.) to determine the best route to get from one point of the city to the other, and in the future be able to establish a comparison among different means of transport and/or a combination of them. For such, the graph theory is considered, used in the planning of network communication, as a basic element to determine the adequate route for a journey. The results obtained will be experimentally evaluated in the environment of public transport of the city of Santander.

Capítulo 1

Introducción

Hoy en día, el transporte público es un servicio básico para cualquier centro urbano que permite el traslado de personas o bienes de un lugar a otro. Existen diferentes tipos de transporte público, entre los que destacan el autobús, el metro y el tren, los cuales permiten conectar diferentes lugares entre sí, cada uno con sus diferencias a la hora de tratar distancias, tiempo y coste monetario.

Según las recientes estadísticas [1], el número de viajeros en los transportes públicos urbanos ha disminuido un 2 % en el año 2013. Si se analiza el gráfico de la Figura 1.1 se puede observar cómo la tendencia del uso del servicio de transporte urbano es negativa, considerando los datos anteriores al 2013. Pese a que el transporte público cada vez tiene menos viajeros que confíen en él como medio de transporte de primera elección y solamente se utilice como alternativa, el servicio ofrece ventajas en muchas ocasiones, minusvaloradas. Por ejemplo, la posibilidad de disfrutar del trayecto en sí, evitando el estrés que suponen otros medios de transporte privados, para aquellas persona que residen en una localidad diferente y que deben desplazarse para ir a emplazamientos de trabajo, estudio u ocio. Adicionalmente, el transporte público se considera el medio más ecológico para desplazarse, dado que al compartir con otros usuarios un mismo vehículo se solidariza la emisión de gases, haciendo más sostenible el entorno. Y, por si no fuera suficiente, otro dato importante a destacar son las oportunidades de trabajo que ofrece un sistema de transporte a una comunidad y la buena valoración, por parte de la sociedad, creada por un servicio eficiente, además de ser uno de los sectores que más contribuye a la generación de riqueza de una región.

Se hace necesario, por lo tanto, buscar opciones que permitan cambiar esa tendencia, intentando mejorar la eficiencia y la calidad de los servicios de desplazamiento. Las telecomunicaciones juegan un papel relevante a la hora de incentivar el interés para el uso de medios de transporte público. Permiten dar a conocer a los usuarios las alternativas disponibles y, asimismo, compararlas con los medios de transporte privado. Si se tiene en cuenta las subidas de precio del carburante en los últimos años, resulta obvio observar cómo el desplazamiento a través de medios de transporte privado no resulta tan rentable, en mayor medida si se usa a diario y para sólo una persona. El transporte público puede ser una alternativa válida y ahorradora no sólo en lo que concierne al coste, sino también en términos de tiempo y de salud.

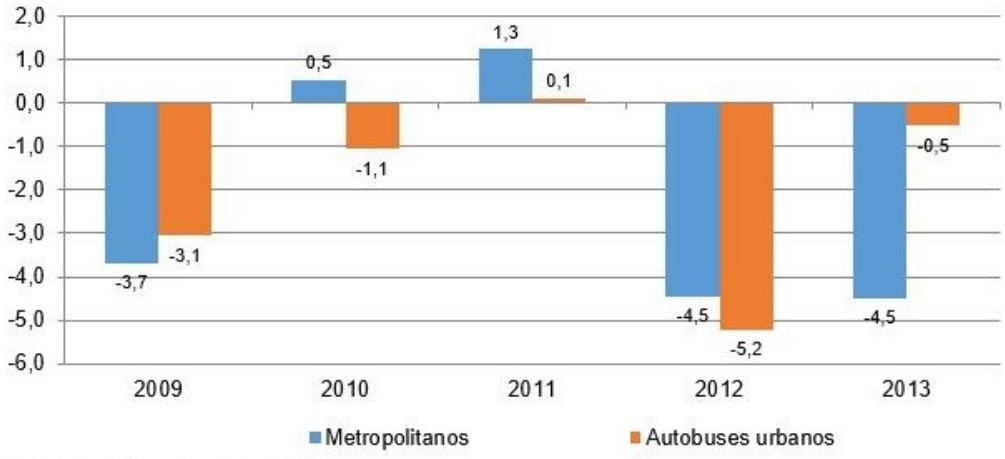


Figura 1.1: Variación anual, en porcentaje, de los viajeros en el tranporte urbano [1]

Los sistemas expertos de planificación de rutas están haciendo posible que la tendencia comience lentamente a cambiar y los ciudadanos identifiquen los diferentes medios de transporte público como una alternativa viable y sencilla de usar en su día a día, o durante los viajes de placer a entornos no conocidos. La tecnología actual con respecto a los dispositivos disponibles por el usuario medio (GPS, *smartphones*, ordenadores portátiles y *tablets*) permite ampliar el conocimiento del entorno de los usuarios y, por lo tanto, hacer una mejor planificación de los viajes. En efecto, cualquier dispositivo con conexión a internet permitiría el uso de un planificador de rutas, una aplicación compleja pero totalmente gratuita y que ha hecho que la manera de planificar un viaje haya cambiado radicalmente en los últimos años.

Un planificador de ruta, hoy en día, permite, no sólo identificar la manera más rápida de llegar a un destino, sino también indica las opciones a poder elegir, las alternativas y toda la información potencialmente útil para poder disfrutar del viaje. Es posible planificar, paso a paso, cada movimiento que tengamos que hacer para llegar al destino que queramos, lo cual resulta interesante sobre todo si no conocemos el entorno urbano en el cual estamos. Además, la interfaz para planificar el viaje, la mayoría de las veces resulta atractiva y sencilla lo que permite el uso del servicio a las personas con menos experiencia con las nuevas tecnologías o a las personas novatas en el campo de los sistema de planificación de ruta. Estos sistemas expertos están en constante evolución, sobre todo debido a nuevas filosofías de compartir la información, como puede ser el *Big Data*, lo que permite ampliar los datos a disposición y, asimismo, mejorar las soluciones planteadas.

Adicionalmente, las necesidades de movilidad de la gente han ido aumentando con la expansión de las ciudades, causando una descentralización de dichos menesteres. El ejemplo más claro son las grandes ciudades como Madrid o Barcelona y los respectivos atascos que se producen a medida que nos acercamos al centro de la ciudad. Desplazarse con el propio medio de transporte privado de esta forma resulta estresante, contaminante para el ambiente y sobre todo peligroso para la persona en sí y los demás. Una cómoda y posible alternativa es el uso de medios de transporte público para evitar los problemas descritos anteriormente.

Y la idea es crear un sistema experto lo más completo y flexible para que permita analizar información de nueva generación con el fin de incentivar el uso del transporte urbano en la actualidad.

1.1. Objetivos

La principal finalidad de este proyecto es desarrollar una aplicación real que permita planificar u obtener la ruta óptima en función de diversos criterios y/o parámetros para desplazarse de un punto a otro de una ciudad utilizando para ello el servicio de transporte urbano. Para conseguir el objetivo, se procederá a modelar la estructura entera de una plataforma, implementado un servidor que permita recibir peticiones por parte de clientes y que, una vez obtenida la información, procese todos los datos y ejecute los cálculos oportunos para poder devolver la información de ruta al usuario. Dichos cálculos se pueden desarrollar a través de patrones de comportamientos predecibles que permiten planificar la estimación de tiempos y trayectos de los medios de transporte.

Para poder modelar y planificar un sistema de este tipo se aprovechará de las técnicas utilizadas en la teoría de grafos, empleada en la planificación de redes de comunicación entre otras, como elemento base para organizar la estructura de la aplicación y para poder implementar el sistema en un entorno de prueba real. En concreto, se evaluará de forma experimental en la red del Servicio Municipal de Transportes Urbanos de Santander (TUS). Además, se deberá proporcionar un cliente para poder interactuar con el sistema.

En un futuro, el objetivo es poder establecer una comparativa entre diferentes medios de transporte para que el usuario pueda escoger la opción que considere más oportuna, basando su decisión sobre datos reales y no empezando desde el desconocimiento del sistema de transporte que pueda ofrecer su ciudad.

1.2. Estructura de la memoria

Tras introducir el contexto del proyecto, a continuación se describe cómo se estructura la memoria para abordar con éxito los objetivos planteados.

En el capítulo 2 se realiza un estudio del estado del arte en este campo, es decir, se analizarán las diferentes opciones que existen en el mercado y se hará una breve descripción de aquellos planificadores de rutas más relevantes y presentes a día de hoy, ahondando en sus elementos y funcionalidades. Gracias a este estudio se podrán inferir una serie de requisitos y funcionalidades que todo planificador debiera tener y a los que se dará respuesta con el sistema experto que se diseñe y desarrolle en este proyecto.

Seguidamente en el capítulo 3 se presenta una breve introducción sobre la teoría de grafos, considerándola como la base para el desarrollo del planificador. En este capítulo se detallan los conocimientos necesarios sobre los diferentes tipos de grafos y las distintas formas de poder representar de forma abstracta dichos grafos, destacando además los principales algoritmos para la búsqueda del camino más corto a implementar. Aplicando estos algoritmos adaptados a la red de transporte será posible desarrollar el planificador.

En los capítulos 4 y 5 se realiza la descripción del trabajo de diseño e implementación desarrollado en base a los análisis realizados en los capítulos anteriores. Así, en el capítulo 4 se presenta el diseño de la solución y se acomete la implementación funcional del sistema, definiendo razonadamente la metodología de recolección de los datos de la red de transporte del TUS y la generación del grafo sobre la que se aplicarán los diferentes algoritmos. En el capítulo 5 se incluyen aquellas mejoras al sistema que permitan ofrecer un servicio mejor adaptado a una red de transporte público. Se detallan la modificaciones necesarias sobre el procedimiento inicial para poder obtener mejores estimaciones. Al finalizar cada etapa del desarrollo se incluyen también los procesos de validación de los procedimientos implementados.

Por último, en el capítulo 6 se recopilan las principales conclusiones obtenidas y se proponen nuevas potenciales líneas de investigación surgidas del trabajo realizado.

Capítulo 2

Estado del arte

En la última década, la planificación de transporte urbano ha cambiado sustancialmente de tendencia. Bajo los preceptos de transporte sostenible o movilidad sustentable, la planificación de transporte propende hacia la promoción de modos más convenientes en términos ambientales (emisiones), sociales (equidad) y económicos (uso racional de los recursos). Para fomentar el uso del servicio de transporte público, los planificadores de rutas ejercen un papel relevante puesto que su uso se ha hecho cada vez más común. La planificación de rutas se ha convertido en una aplicación *software* real y completa, denominada también sistema experto, que maneja una multitud de variables, muchas de ellas en tiempo real como la información sobre tráfico, posibles accidentes, cortes de carreteras, etc. En los diferentes casos, el sistema experto iría adaptando la salida a las características prefijadas por el usuario a través de su perfil, preparando la ruta con anterioridad gracias al estudio de patrones predecibles para definir cuándo sería el momento ideal de efectuar un cambio de línea, por ejemplo, ofreciendo alternativas a la ruta principal. En este capítulo se hará un extenso análisis de los diferentes tipos de journey planners o planificadores de ruta, para entender su funcionamiento, los requisitos que conlleva la implementación de un sistema y las diferencias entre los tipos que están disponibles.

2.1. Planificador de rutas

Comúnmente, cuando se habla de un planificador de rutas o *journey planner*, se hace referencia a un algoritmo de búsqueda que permita averiguar la manera óptima de llegar desde un origen hasta un destino determinado, proporcionando información coherente sobre el camino que vayamos a recorrer. Además, existen varios modelos de estimación de rutas, entre los cuales permiten una planificación anticipada del estado de las carreteras, en concreto se hace referencia al modelo estático. Este modelo consiste en una estimación aproximada del trayecto que se va a recorrer o del tráfico presente en el camino determinado, principalmente basándose en datos históricos sobre la circulación. Al contrario, el modelo de estado dinámico permite una estimación usando información en tiempo real y dependiente de factores a tener en cuenta como pueden ser la franja horaria, el tipo de tramo o las condiciones meteorológicas actuales. Desde este tipo de modelos se exige un procesamiento de los datos de tráfico para

poder establecer patrones de comportamiento que permitan intuir o calcular la evolución del estado actual de las carreteras a los planificadores. La combinación de estos dos tipos de modelos de estimación descritos anteriormente, se aplican a varios tipos de planificadores de rutas como pueden ser los GPS, usados habitualmente para viajar en coche, o sistemas diferentes que mantengan una estructura por detrás, como los servicios de transporte urbano.

Existen diferentes tipos de *journey planner* [3] y, en nuestro caso especifico, se ha analizado el funcionamiento de programas de planificación de ruta que trabajen a través de medios de transporte público. Para entender mejor el tema a tratar, se procederá a explicar a continuación algunos ejemplos de sistemas funcionales que se pueden encontrar en el mercado y que son utilizados por la mayoría de las personas.

2.1.1. Google Transit

Esta herramienta es, probablemente, el planificador de rutas más utilizado por el usuario medio, debido sobretodo a la gran popularidad de la cual disfruta Google Maps. En este sentido, Google Maps se apoya en el proyecto Google Transit [4] para ofrecer un planificador de ruta para sistemas urbanos de grandes ciudades o directamente entre medios de transporte público de empresas privadas. Integra información sobre paradas de autobuses, rutas, horarios y tarifas para planear un viaje de manera rápida y sencilla. Además, permite ordenar los diferentes resultados de rutas a través de distintas opciones como el medio de transporte elegido, o si se quiere tener en consideración no sólo la mejor ruta, sino también el camino con menos transbordos o, por ejemplo, con menos recorrido para hacer a pie.

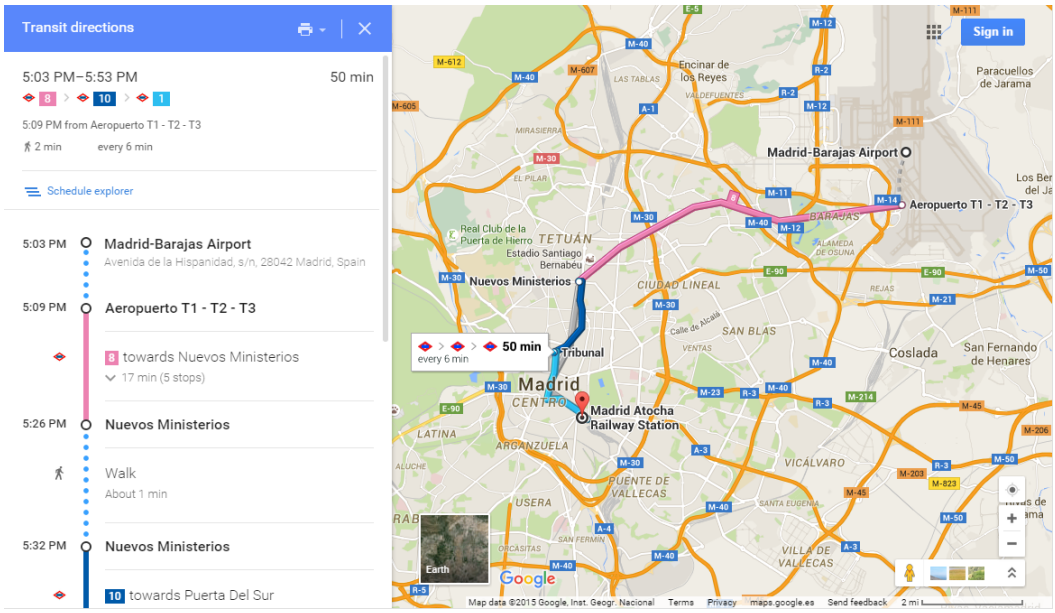


Figura 2.1: Ejemplo de búsqueda en Google Transit

Google Transit permite planificar viajes en transporte público gracias al intercambio de información con el sistema de transporte de una ciudad. En la Figura 2.1 se puede ver un ejemplo de como la aplicación planifica el trayecto y la información final proporcionada al

usuario. Para ello se utiliza un formato común de intercambio de información llamado GTFS (*Google Transit Feed Specification*) del cuál se va a hablar a continuación.

2.1.1.1. GTFS

Este estándar [5] permite que las empresas de transporte público compartan sus datos de transporte de una manera que sea fácil, manejable e intuitiva. Este proyecto tiene como objetivo recopilar toda la información posible sobre transportes para ofrecer un servicio completo y gratuito a la comunidad. Para poder manejar la misma información, el estándar establece unas normas a cumplir, es decir, todos los campos obligatorio que tiene que contener una *feed* con la respectiva información a proporcionar. En este caso, GTFS sólo incluye la información sobre la estructura de la red o aquellas operaciones previamente planificadas, como la ruta de paradas por la cual pasa una determinada línea de autobús (horarios transporte publico, posición geográfica, etc.). Sin embargo, es posible añadir información de tiempo real gracias a la especificación GTFS-Realtime. Este tipo de datos, denominados también emphlive feeds, permiten enviar información actual y dinámica sobre la estructura de la red y sus componentes. En general, existen tres tipos de *feeds* empleados que son: información sobre actualizaciones de rutas, alertas sobre el servicio (Cuadro 2.1) y la posición de los vehículos circulantes por la estructura (retrasos, alertas, desvíos, traslados, información sobre vehículos, etc.).

Cuadro 2.1: Ejemplo de GTFS-Realtime sobre información de alertas

```
header {
        gtfs_realtime_version: "1.0"
        incrementality: FULL_DATASET
        timestamp: 1284457468
}
entity {
        id: "0"
        alert {
                active_period {
                        start: 1284457468
                        end: 1284468072
                }
                informed_entity {
                        route_id: "219"
                }
                informed_entity {
                        stop_id: "16230"
                }
                cause: CONSTRUCTION
                effect: DETOUR
                url {
                        translation {
                                text: "http://www.sometransitagency/alerts"
                                language: "es"
                        }
                }
                header_text {
                        translation {
                                text: "La parada en Elm street está cerrada, detenerse
                                    temporalmente en Oak street"
                                language: "es"
                        }
                }
                description_text {
                        translation {
                                text: "Debido a una construcción en Elm street, la parada
                                    está cerrada. La parada temporal se puede encontrar a 300
                                     metros al norte de Oak street."
                                language: "es"
                        }
                }
        }
}
```

Para poder enviar los *feeds* creados a Google se ha de seguir una serie de pasos los cuales pueden dificultar el proceso de cumplimiento de los estándares prefijados por la compañía. Antes de ponerse en contacto con el equipo de Google Transit para poder publicar la información, se deben tener preparados todos los *feeds* de acuerdo a las especificaciones recomendadas y guardarlos en un formato establecido después de haber pasado unas pruebas de validación e inspección. Adicionalmente, la complejidad del conjunto de datos a entregar es elevada y no trivial a la hora de obtener dicha información por la estructura del sistema.

Una vez publicada, la información pasada a Google Transit en formato *feed* resulta disponible online [6] para aquel que quiera averiguar determinada información o hacer pruebas con un determinado tipo de dato de alguna ciudad. Entre las ciudades más importantes que participan en este proyecto, caben destacar la ciudad americana de Portland como pionera en el sector junto con San Francisco. En España, además de la ciudad de Madrid, se puede destacar a Vitoria como una de las primeras ciudades en aprovechar de este servicio.

2.1.2. Open Trip Planner

Este planificador [8] tiene como característica fundamental su naturaleza *Open Source*. El código del programa es público y está disponible para ser modificado por otros programadores ajenos al proyecto en sí. En la actualidad se han visto muchos proyectos llevados a cabo con este sistema y, con el pasar de los años, la fiabilidad de proyectos desarrollados de esta manera ha aumentado exponencialmente, otorgando un manera diferente de desarrollo del software a la comunidad de programadores.

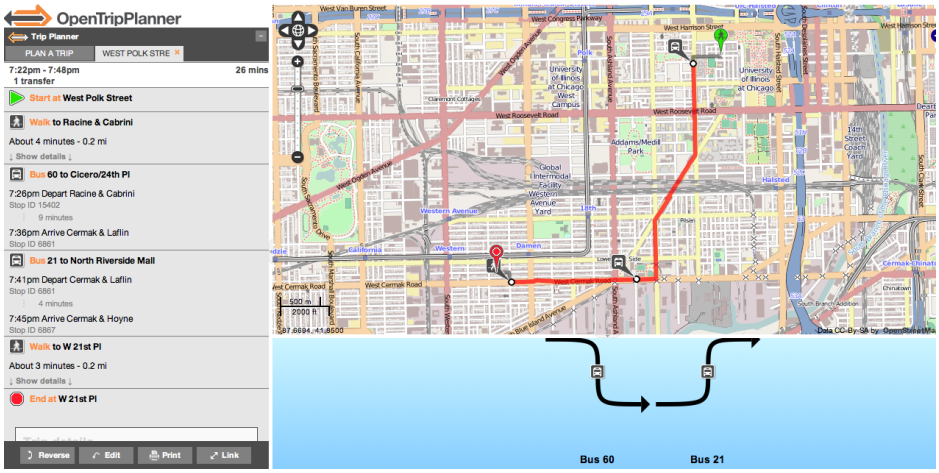


Figura 2.2: Ejemplo de búsqueda en Open Trip Planner [9]

Open Trip Planner ofrece todas las herramientas para crear un sistema completo de planificación de rutas por medio de todos los transportes públicos disponible en un entorno. Los puntos a destacar de este proyecto son, sin duda, su sencillez a la hora de implementarse en una localización de gran complejidad y tamaño, como puede ser la gestión del transporte público de una ciudad. Adicionalmente, Open Trip Planner permite combinarse con otros

sistemas de transporte cómo puede ser el alquiler de bicicletas y otorga la posibilidad de alcanzar un nivel de personalización de los servicios bastante elevado. Este sistema usa el estándar GTFS de Google que hemos presentado anteriormente.

A la hora de intercambiar información con la base de datos, Open Trip Planner no sólo puede pasar información sobre la estructura de la red de transporte, sino también información de tiempo real para alertas, actualizaciones de rutas y la posición exacta para situar los medios de transporte que circulen por la red. En la Figura 2.2 se puede observar un ejemplo de búsqueda con este planificador.

2.1.3. TomTom

Tomamos TomTom [10] como ejemplo de planificador de ruta o *route planner* que usa el sistema de posicionamiento global (GPS) para establecer la posición de origen de la ruta, así como el seguimiento en tiempo real del usuario. Si bien en los casos anteriormente señalados la ruta también se puede establecer dinámicamente en función de la posición, en esos casos su operativa está más orientada a una planificación estática que interconecte dos puntos en un momento dado. En el caso de TomTom, la planificación de la ruta se suele considerar empleando un único medio de transporte (usualmente el coche), cuya posición está determinada mediante dispositivos GPS.

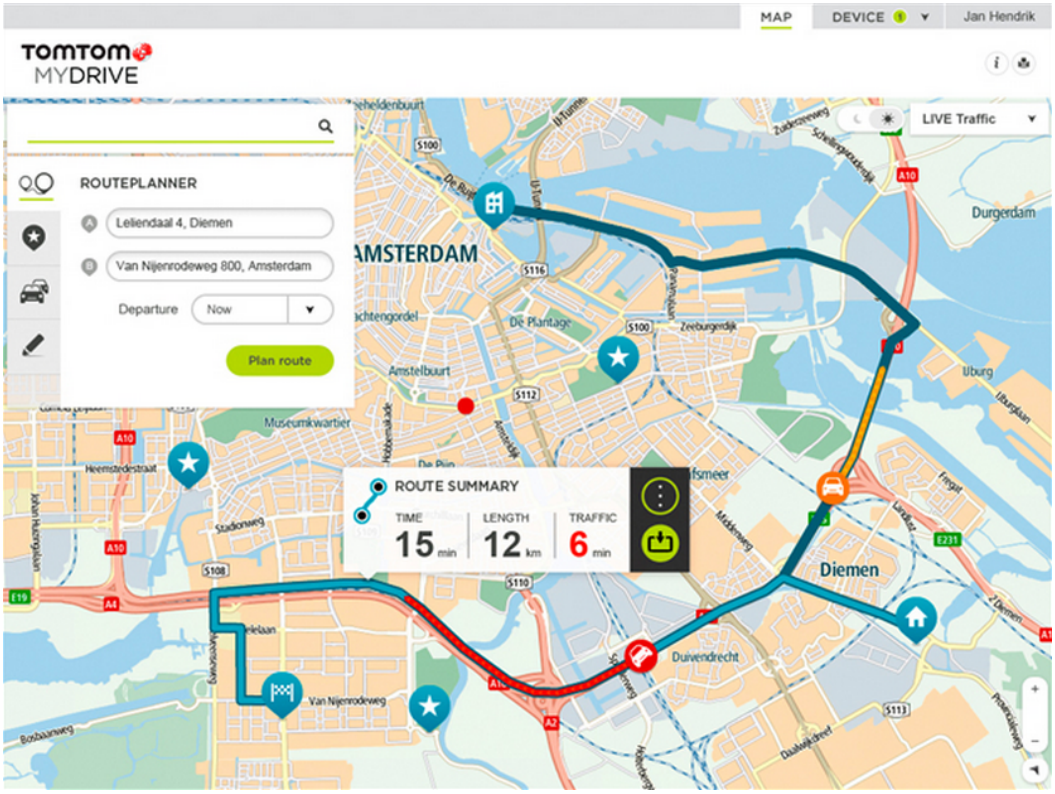


Figura 2.3: Ejemplo de búsqueda en el servicio TomTom MyDrive

Los terminales de TomTom permiten establecer la ruta más rápida o más corta hacia un destino determinado y son capaces de recalcular variaciones de esta de forma rápida en caso de que el desplazamiento del usuario así lo requiera. Su principal característica, como se ha explicado anteriormente, se basa en el uso del GPS para establecer el punto de partida, lo que permite al sistema actualizar la información de ubicación del dispositivo en tiempo real, proporcionando información detallada y constante. Adicionalmente, a diferencia de los anteriores, el modelo de planificación de ruta empleado por los dispositivos TomTom pueden operar con mapas almacenados y si no se consideran parámetros adicionales como el tráfico, la meteorología o incidencia, no requieren de establecer una conexión a Internet.

En este sentido, este producto se suele usar habitualmente hoy en día para planificar una ruta de viaje en coche. Además, los últimos modelos de este dispositivo proporcionan información variada sobre la ruta, la cual puede ser de utilidad para el conductor, como pueden ser la ubicación de radares en una autopista o la información sobre cortes repentinos de alguna carretera. En la Figura 2.3 se puede apreciar la labor del servicio ofrecido por la compañía.

2.1.4. Here Transit

Here Transit [7] es una aplicación gratuita para móviles que permite planificar un viaje a través del transporte urbano de una ciudad. Hace parte de un conjunto de otras aplicaciones (Here Maps, Here Drives, Here City Lens) de propiedad de Nokia, y actualmente se puede disfrutar de sus servicios de planificación de rutas como funcionalidad de Here Maps. La aplicación se centra en ofrecer la posibilidad de planificar un viaje, no sólo a través de los autobuses, sino también mediante otros servicios de transporte público como metros, trenes, tranvías, ferries y proporciona información sobre trayectos a píe, todo en una aplicación. Además, permite poder seleccionar de antemano la hora a la cual queramos empezar nuestro viaje para así poder coordinar los tiempos en referencia al establecido.

La aplicación, gracias a varias colaboraciones con otros servicios, permite obtener información en tiempo real sobre el trafico y, en conjunto, utiliza también datos históricos para planificar de antemano la mejor ruta o encontrar soluciones en caso de concurrir con eventos imprevistos. Dicho esto, la oferta relativa a las ciudades soportadas resulta todavía limitada, sobe todo si se compara con servicios más conocidos explicados en los apartados anteriores.

2.2. Requerimientos de un planificador de rutas

Para entender la complejidad de un *journey planner*, a continuación se procede a describir cómo típicamente funciona un sistema de este tipo, haciendo hincapié en los elementos que necesita un planificador de rutas para medios de transporte público.

El principal objetivo de un *journey planner* para transporte público es proporcionar información sobre las posibles formas de llegar a un destino a través de rutas en las que se consideran únicamente medios de transporte públicos o desplazamientos a pie. Normalmente, esta información suele estar disponible gracias a aplicaciones web dedicadas a ello o a aplicaciones móviles.

A diferencia de otros sistemas, como por ejemplo la búsqueda de la mejor ruta para ir en coche de un lugar a otro, la elección de una ruta para los servicios de transporte público se enfrenta a varias restricciones y a diferentes parámetros que tiene que cumplir. Por ejemplo, es imprescindible que la aplicación actualice de manera dinámica la información sobre los horarios del servicio de transporte, para informar al usuario a qué hora sería posible usar determinado servicio, medio de transporte o línea. Otros factores externos, como la velocidad de marcha del medio de transporte, o las condiciones de la carretera sobre la cual se viaja, pueden influir sobre el tiempo empleado para ir a un destino determinado. A su vez, la elección de un diferente servicio de transporte público, sea el metro, el autobús o el tren, conlleva diferentes criterios de elección. En este sentido, se han analizado los requerimientos de un *journey planner* basado en el transporte por medio de autobuses, en particular caso a la red del Servicio Municipal de Transportes Urbanos de Santander (TUS).

2.2.1. Algoritmo de búsqueda

Normalmente, los planificadores de ruta para redes de gran tamaño usan la teoría de grafos para poder modelar el sistema, representando las paradas del servicio de transporte a través de nodos conectados entre sí. En el capítulo siguiente se va a profundizar más sobre el tema a modo de introducción a la teoría de grafos. El enlace o arco que conecta dos o más nodos representa el recorrido que un hipotético medio de transporte tendría que seguir para alcanzar el siguiente nodo o, dicho de otra manera, la siguiente parada. Para ello, se atribuyen diferentes pesos, como por ejemplo la distancia, el tiempo que tarda un autobús en recorrer el enlace o el coste monetario de la ruta por cada parada. Para encontrar la ruta de menor coste, dependiendo del peso que se haya elegido, se utilizan algoritmos de búsqueda sobre grafos.

Como se ha escrito anteriormente, se han de tener en cuenta otros criterios a la hora de trabajar con una red de servicios de transporte público. Por ejemplo, los horarios de servicio en el cual están disponibles las diferentes líneas de autobús puesto que puede haber diferencia entre calendarios de verano e invierno, o entre días festivos y laborales. Esta información adicional hará que la configuración del grafo varíe y por tanto la opción óptima que se muestre al usuario.

Sin embargo, a la hora de utilizar cualquier *journey planner*, debemos tener en cuenta que debe cumplir adecuadamente con su principal objetivo: buscar de forma óptima la mejor ruta para llegar a un destino previamente definido. Sea cual sea el tipo de peso o coste elegido, la respuesta de un planificador de ruta se encarga de informar al usuario final cuál es el trayecto más corto según tiempo, distancia u otros criterios. Además, en ocasiones, proporcionará información sobre el coste total del camino en cuestión. Para lograr el objetivo, todo planificador se basa en un algoritmo de búsqueda sobre grafos para determinar cuál es el camino óptimo. Entre muchos algoritmos desarrollados sobre este tema, se pueden destacar el algoritmo de Dijkstra, el algoritmo de Bellman-Ford [15] y el algoritmo de Floyd-Warshall [15] entre otros.

2.2.2. Información en tiempo real

Normalmente, la respuesta a una solicitud debe ser entregada en tiempo real, o lo antes posible una vez analizado el trayecto y habiendo realizado los cálculos oportunos. Dicho en otras palabras, cualquier *journey planner* tiene que ocuparse de optimizar la memoria para representar la estructura de la red y para permitir ejecutar una búsqueda rápida del camino más corto a elegir entre muchos. La mayoría de los *journey planners* soportan el uso de una base de datos de rápida búsqueda, a partir de la cual recoger el estado de la estructura de la red el cual habrá sido previamente almacenado en tablas relacionadas.

Se deberá tener en cuenta no sólo la respuesta rápida, sino también el número de peticiones que pueden llegar, a la vez o en distinto momento, al servidor del sistema. Una petición se suele generar gracias a una interfaz web específica, la cual puede ser utilizada por diferentes usuarios finales o por diferentes dispositivos. El objetivo en este sentido se centra en poder responder a todas las peticiones recibidas con un retardo aceptable, sea debido a la cantidad de información procesada o al número de peticiones simultáneas recibidas.

En ese tipo de sistemas, además, se debe tener en cuenta la información que varía en tiempo real como lo es el tráfico, la meteorología y en definitiva todos aquellos elementos que componen un modelo dinámico. El conjunto de estos parámetros y su combinación permiten definir el peso asignado a los arcos que interconectan cada parda. Estas características permiten definir la velocidad de desplazamiento del medio de transporte a través de estimaciones, un dato que resulta muy relevante para estimar cuánto tiempo se necesitaría para realizar el recorrido.

2.2.3. Detección posible transbordo

A la hora de definir una ruta en transporte público es difícil encontrarse con la situación en el desplazamiento entre el origen y el destino pueda hacerse empleando el mismo vehículo o línea, si se consideran todos los elementos para determinar la opción más adecuada. En este sentido, dependiendo de los deseos del usuario se puede optar por la opción más rápida,más simple, más cómoda, etc. En cualquiera de los casos, habrá que realizar lo que se denomina un transbordo entre medios de transporte o líneas de un mismo medio durante el recorrido. Sin embargo, la detección de estos cambios e identificar las interconexiones entre líneas no es un proceso sencillo, pues no sólo hay que considerar la estructura de la propia red de transporte, sino también la localización de las paradas y posibles desplazamientos a pie. Asimismo, otro factor que toma importancia en este aspecto es el poder estimar el tiempo de llegada y salida de un autobús de una parada determinada. Por lo tanto, se puede observar cómo se añaden elementos y variables externas que hacen que el procesar la información y elegir el camino más corto no sea tarea fácil.

2.2.4. Interfaz usuario

Una característica fundamental del sistema de planificación ideal debe ser cómo se proporciona información al usuario final. Comúnmente, un planificador de ruta ofrece la posibilidad de establecer el punto final del recorrido y el punto de partida, aunque los últimos sistemas

integran la posibilidad de identificar la posición del solicitante o del dispositivo, estableciendo la ubicación de inicio del trayecto automáticamente. Una vez identificada la ruta, las aplicaciones suelen dibujar sobre un mapa, ya sea del país entero o de una ciudad en concreto, el camino a seguir para llegar al destino, incrementando o no el nivel de detalle sobre el mapa a segunda de la longitud del trayecto normalmente. Se suele, también, presentar alternativas a la ruta principal dependiendo de los criterios elegidos para establecer la ruta, ya sea la más rápida o la que menos recorrido tenga, recordando cómo los dos parámetros no tengan porque coincidir.

Adicionalmente a la ruta dibujada sobre el mapa, se suele presentar toda la información con respecto a trayectos realizadosa pie, en coche o en medio de transporte público, evidenciando aquello eventos relevantes como la necesidad de cambiar de línea de transporte, las paradas restantes para llegar al destino, el tiempo estimado de llegada, etc.

Capítulo 3

Teoría de grafos

La teoría de grafos ha alcanzado un notable desarrollo en las últimas décadas gracias a que numerosos problemas de la vida cotidiana pueden ser modelados por medio de grafos y planteados como problemas de optimización. Este campo se ocupa de estudiar las características y propiedades que surgen de las diferentes relaciones que se establecen entre los elementos de un conjunto dado de objetos, es decir, de la definición de un grafo. A continuación, se procederá a explicar de manera breve y general cómo se construye un grafo, haciendo hincapié en la importancia que la teoría de grafos tiene en este proyecto.

3.1. ¿Qué es un grafo?

Un grafo es un término matemático que se usa para definir a un conjunto de puntos unidos entre sí por segmentos y sobre el cual se pueden representar procesos o relaciones funcionales de todo tipo. Un grafo $G(N, E)$ se define como un par, ordenado o no, de nodos N y una colección de enlaces E o arcos, donde cada enlace consta de una pareja de nodos de N. Un ejemplo donde se puede apreciar la utilidad de un grafo es la representación de las relaciones topólogicas entre elementos. La teoría de grafos, en este caso en concreto, permite asociar un grafo a una red de transporte o, por ejemplo, a la estructura de interconexión de carreteras entre ciudades. De esta forma se eleva el problema a un nivel de abstracción superior, facilitando su computación y modelado para razonarlo como un problema de optimización.

En los apartados sucesivos, se procederá a analizar los elementos de un grafo y la manera en la que se organizan las relaciones, resultado de la cual surgen diferentes tipos de grafos.

3.1.1. Nodos y arcos

Como se ha descrito anteriormente, un grafo se puede definir con un conjunto de nodos conectados entre sí mediante arcos. La relación entre nodos y arcos se puede asociar fácilmente a objetos geográficos de la vida real. De esta forma, los nodos pueden representar objetos estáticos como estaciones, paradas, cruces de carreteras o lugares de interés a destacar, también podrán ser dinámicos y en este caso lo arcos, relaciones y demás se podrían

generar dinámicamente; mientras que los arcos, de manera similar, concatenan los anteriores elementos para representar las que pueden ser carreteras, líneas de autobuses, metro o tren, canales u otras. En definitiva, estos enlaces representan el mecanismo de transporte sobre el cual se mueve el problema a analizar, sean personas, coches o información digital, por ejemplo.

3.1.2. Tipos de grafos

En el ámbito de la teoría de grafos existe una gran variedad a la hora de configurar un grafo y sus características. Obviamente, cada una se adapta de manera diferente según el objetivo que se quiera conseguir. Seguidamente se realiza una enumeración de las tipologías de grafos vinculadas al presente proyecto:

- **Grafo "weighted":** Se puede traducir como grafo con pesos asignados. En algunas ocasiones resulta interesante y necesario asignar algún tipo de peso a los enlaces E, por ejemplo, coste, tiempo, distancia o capacidades, dependiendo del problema en cuestión.

- **Dirigidos y no dirigidos:** Un grafo dirigido, suponiendo que u es el origen y v es el destino, se define como un enlace unidireccional donde $(u, v) \neq (v, u)$, como se observa en la Figura 3.1(a).

(a) Dirigido (b) No dirigido

Figura 3.1: Grafos según su direccionalidad

  Por el contrario, en grafos no dirigidos no es necesario establecer un criterio de ordenación, puesto que el enlace (u, v) implica que existe el enlace (v, u), siendo este bidireccional, como se demuestra en la Figura 3.1(b).

- **Densos o poco densos:** Este tipo de grafo clasifica en función del número de enlaces que tiene el grafo. Se asume por lo tanto que un grafo denso será aquel que el número de enlaces se aproxima al número máximo de enlaces que pueda tener, es decir, $(N^2-N)/2$, mientras que un grafo con solo algún enlace se le puede renombrar poco denso o *sparse*.

- **Conexo o inconexo:** Un grafo inconexo se define cuando no existe un enlace entre todos los pares de nodos, y por tanto no existe un camino que pase por todos los nodos del grafo, como se puede ver en la Figura 3.2.

  De otro modo, un grafo G se dice conexo si, para cualquier par de vértices u y v en G, existe al menos una trayectoria (una sucesión de vértices adyacentes que no repita vértices) de u a v, como por ejemplo el grafo incluido en la Figura 3.3(a). De igual forma, un grafo puede ser fuertemente conexo si para cualesquiera dos vértices u y v existe un camino dirigido de ida y de regreso, como el de la Figura 3.3(b).

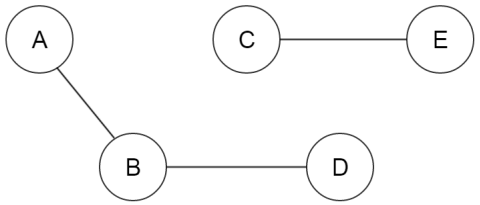
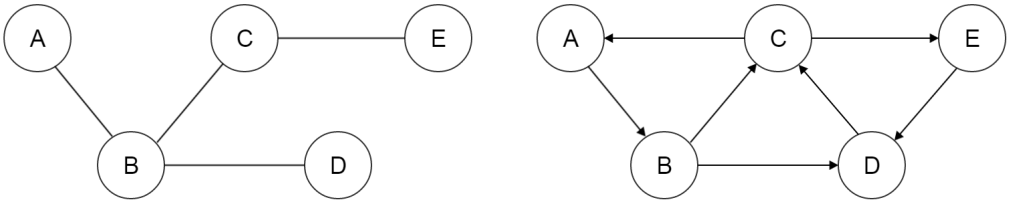


Figura 3.2: Grafo inconexo



(a) Conexo (b) Fuertemente Conexo

Figura 3.3: Grafos conexos

- **Árbol:** Cuando un grafo es conexo, no dirigido y conectando a todos los nodos no contiene ciclos, se le denomina árbol. Es decir, un grafo con N nodos tendrá un árbol con $N-1$ enlaces, siempre y cuando cumpla con las características anteriores. Este tipo de grafo se suele utilizar en protocolos de enrutamiento para la búsqueda del Minimum Spanning Tree (MST) [15].

3.1.3. Representación de grafos

Para poder representar de manera computacional un modelo de grafo abstracto se ha de crear primero una estructura de datos para poder procesar la información. Para ello, existen dos tipo de filosofías a implementar que explican a continuación:

- **Matriz de adyacencia:** se define una matriz A de dimensión NxN, y por tanto con tamaño igual a N^2, para cualquier tipo de grafo. En la matriz A las filas representan el número del nodo origen u y las columnas los nodos destino v. Cada elemento de la matriz de adyacencia indica la existencia de enlace entre dos nodos cualesquiera. Así el elemento (u, v) de una matriz A será 1 cuando el nodo u sea un nodo adyacente al nodo correspondiente v. Adicionalmente, el valor del elemento (u, v) podrá tener un valor distinto de uno, es decir, el peso de dicho enlace para grafos con pesos asignados. En caso contrario, el valor es 0 y, según un estudio reciente [16], se consideraría que dos nodos son el mismo de alguna forma, lo que permitiría aumentar la eficiencia computacionalmente. Gracias a su estructura permite efectuar una búsqueda rápida de un enlace concreto. En la Figura 3.4 se muestra un ejemplo de matriz de adyacencia.
- **Lista de adyacencia:** En una lista de adyacencia se genera un array por cada nodo N del grafo, donde cada array contiene los nodos adyacentes del nodo al cual se

M	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

Figura 3.4: Ejemplo de una matriz de adyacencia

hace referencia. Cada posición del array contiene un puntero a cada uno de los nodos adyacentes, como se puede ver en la Figura 4.5. Se suele implementar como una lista enlazada en programación, lo cual permite optimizar la memoria necesaria para almacenar el grafo entero.

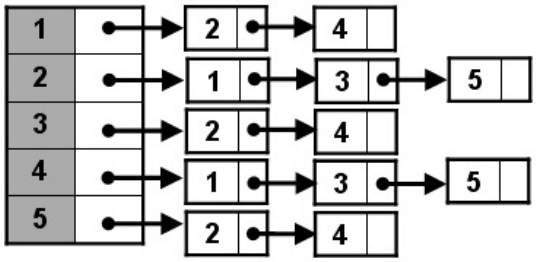


Figura 3.5: Ejemplo de una lista de adyacencia.

Si se comparan las dos formas de representar un grafo, se pueden destacar ventajas e inconvenientes de cada filosofía. En general, se suelen representar más a menudo los grafos a través de una lista de adyacencia porque resulta más eficiente a la hora almacenar los datos y permite recorrer el grafo de manera más rápida. Sin embargo, implementar una solución por medio de la matriz de adyacencia puede resultar conveniente sobretodo cuando se trabaja mediante un grafo denso, es decir, con muchos enlaces.

3.2. Algoritmos de búsqueda del camino más corto

Una de las problemáticas más frecuentes es el deber encontrar el camino mas corto en un grafo dado es una de las problemáticas más frecuentes en el ámbito de las telecomunicaciones, en particular el reto de hallar el camino con menos saltos entre un vértice origen y un destino.

Existen, a su vez, otras problemáticas relacionadas con el camino más corto, y por esto cabe destacar el llamado problema del viajante o TSP (*Travelling Salesman Problem*) [15].

El reto de este problema se basa en recorrer todos los nodos de un grafo de la forma más eficiente, con la obligación de visitar todos los nodos y de volver al punto de origen. Es uno de los problemas de optimización más estudiados, también por su aplicación y modelado en el entrono real. Lo que busca es la manera de optimizar computacionalmente el cálculo de la mejor ruta, comparando entre ellas soluciones heurísticas, exactas y aproximadas.

Estos algoritmos y sus respectivas soluciones fueron desarrollados mayoritariamente en los años cincuenta y, por esta razón, se ha trabajado en este campo centrando sus objetivos en dirección a aumentar la eficiencia computacional de los diferentes algoritmos.

A continuación, con el objetivo de mejorar la comprensión de este trabajo, se procederá con una pequeña explicación de algunos de los algoritmos que se han tenido en cuenta en la elaboración del mismo:

3.2.1. Algoritmo de Dijkstra

El algoritmo de Dijkstra [15] es, probablemente, el algoritmo más conocido que trata de hallar el camino más corto de un grafo entre dos nodos definidos. Fue propuesto por Edsger Dijkstra a finales de los años cincuenta y aún a día de hoy constituye la base de la teoría de grafos, siendo uno de los algoritmos más sencillos y que primero se imparten en el ámbito académico. Permite resolver el problema de búsqueda del trayecto con menor coste desde una fuente S hasta un nodo destino D, lo que a su vez permite determinar el camino más corto hacia el resto de nodos. Este algoritmo se desarrolla sobre grafos con pesos en los arcos y que implementan costes no negativos.

3.2.1.1. Funcionamiento

El algoritmo de Dijkstra permite descubrir el trayecto óptimo paso a paso, mientras se va analizando el grafo. Se procede a continuación a explicar el funcionamiento del algoritmo y los pasos a seguir para poder recorrer un grafo y obtener el camino más corto entre dos nodos:

1. Crear e inicializar el array distancia d. Asignar una distancia de 0 al nodo fuente S y una infinita (o un valor de un coste muy alto) para el resto de los nodos del grafo.
2. Seleccionar el nodo fuente S como nodo actual y crear un array Q con todos los nodos del grafo, representando todos los nodos que todavía no se han visitado.
3. Para el nodo actual, averiguar todos los nodos adyacentes que aún no se han visitado, o lo que es lo mismo, que están presentes en el array Q, y analizar el coste potencial del enlace. Comparar el nuevo coste del enlace con el asignado actualmente en el array distancias d para elegir el de menor valor.
4. Una vez analizados todos los nodos adyacentes del nodo actual, marcar el nodo como visitado, es decir, eliminar el nodo del array Q, que contiene la lista de nodos aún no visitados.
5. Cambiar el nodo actual por el nodo adyacente de menor coste y repetir el paso 3.

6. Si el nodo destino D aparece como visitado o si el array Q no contiene ningún elemento, se termina el algoritmo.

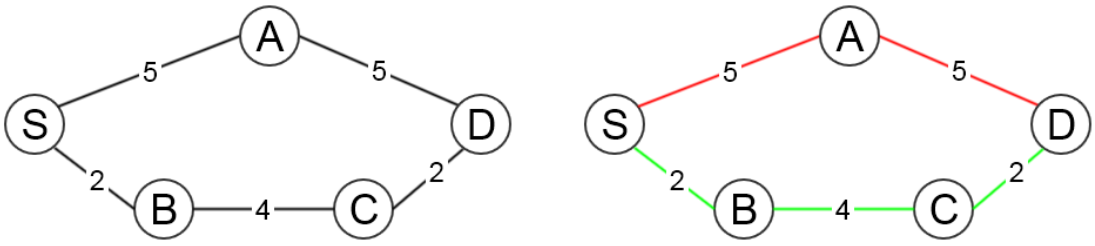
Al final de la ejecución del algoritmo, devuelve como resultado el camino de menor coste, según el criterio que se haya utilizado, para llegar desde el origen S al destino D. Adicionalmente, el algoritmo puede devolver el coste total del camino, lo que se consigue mediante la suma del coste de cada enlace del recorrido.

En el Procedimiento 1 se representa el pseudo-código del algoritmo de Dijkstra.

Procedimiento 1 Algoritmo de Dijkstra

Require: $d(S) = 0$, $d(v) = \text{ínf}$, $Q = N$
Ensure: array d, array $prev$
1: **while** $Q \neq \{0\}$ **do**
2: $u \leftarrow min\{d(v)\}$ del array Q
3: Borrar u del array Q
4: **for all** v en Q adyacentes a u **do**
5: **if** $d(v) > d(u) + c(u, v)$ **then**
6: $d(v) = d(u) + c(u, v)$
7: $prev(v) = u$
8: **end if**
9: **end for**
10: **end while**

Si se analiza, por ejemplo, el grafo de la Figura 3.6(a), el objetivo es, como se ha comentado anteriormente, encontrar el camino de coste mínimo entre el nodo origen S y el nodo destino D. Aplicando el algoritmo de Dijkstra sobre dicho grafo (Figura 3.6(b)) se puede ver cómo el algoritmo devuelve al camino más corto, en este caso $S \rightarrow B \rightarrow C \rightarrow D$ con un coste total igual a 8, aunque la ruta alternativa $S \rightarrow A \rightarrow D$ tenga menos saltos para llegar al destino pero un coste total del camino superior.



(a) Grafo original (b) Algoritmo de Dijkstra

Figura 3.6: Aplicación del algoritmo de Dijkstra sobre un grafo

Dicho esto, puede que el algoritmo no devuelva una solución de ruta mínima en el caso particular de que el grafo no sea conexo. En efecto, si se busca el trayecto de coste mínimo entre dos nodos, pero no existe un camino para llegar del origen al destino, el algoritmo no devuelve una solución.

3.2.2. Algoritmo BFS

El algoritmo Breadth-First Search (BFS) [15] es uno de los algoritmos más sencillos que existen para la búsqueda del camino más corto. El algoritmo funciona tanto para grafos dirigidos como no dirigidos. El BFS también permite encontrar el camino más corto entre dos nodos determinados, aunque no es esta su única función. Sin embargo, resulta tener una elevada eficiencia computacional, o, expresado con otras palabras, recorre de manera rápida el grafo comparado con otros algoritmos. El BFS se puede definir como un caso particular del algoritmo de Dijkstra, donde los costes asignados a los arcos sean todos iguales a uno. En efecto, el algoritmo original BFS no trabaja por medio de grafos con asignación de coste, sino que para conseguirlo habría que modificar parte del algoritmo y empelar una cola de prioridad.

3.2.2.1. Funcionamiento

Se trata de un conocido algoritmo de exploración de grafos, que comienza en un nodo *root* o raíz para después recorrer uno por uno todos sus "hijos". Se puede decir que explora la red "en anchura" puesto que primero ha de revisar todos los nodos de un nivel antes de avanzar al siguiente nivel. Con nivel se hace referencia a todos aquellos nodos adyacentes que estén a una distancia de N saltos desde el origen. Esta particularidad hace que el BFS descubra paulatinamente los nodos de un nivel N antes de descubrir los nodos de nivel $N+1$, evitando recorrer de forma repetitiva los nodos y los enlaces ya procesados.

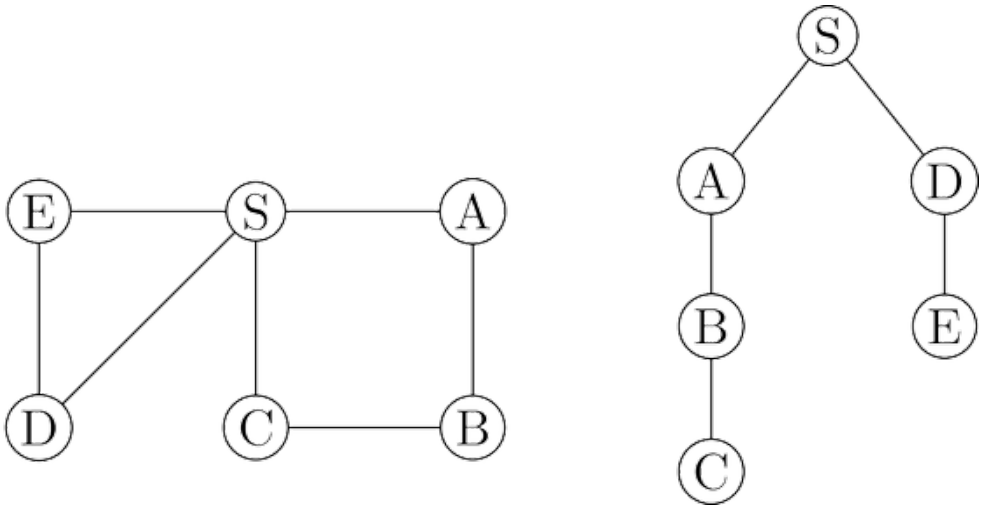
La ejecución del algoritmo permite generar un árbol a partir del grafo original, con un nodo raíz asignado a cada nodo del grafo, a excepción del nodo de origen S.

1. Inicializar el grafo creando un array Q que conste de el nodo origen o raíz.
2. Comprobar si el primer elemento del array Q es el nodo destino. De ser así, se termina el algoritmo; de otro modo, ir al siguiente paso.
3. Eliminar el primer elemento del array Q. Agregar los nodos adyacentes, llamados también nodos hijos, del primer nodo al final del array, si existen.
4. Si el array Q está vacio, se termina el algoritmo. En caso contrario, volver al paso 2.

En la Figura 3.7, podemos ver un ejemplo de la función del algoritmo BFS, es decir, cómo se descubre secuencialmente el camino, en diferentes niveles generando un árbol del grafo original.

3.2.3. Algoritmo de Yen

El algoritmo de Yen [14,], mejor conocido por K-Shortest Path (KSP), fue creado por el mismo Jin Yen en 1971. Este algoritmo, como dice su propio nombre, permite hallar los K caminos de menor coste entre dos vértices determinados de un grafo, evitando que estos contengan bucles. Suele recomendarse su uso sobre grafos densos, es decir, para un valor de K elevado p. ej. $(N-1)!$, puesto que presume de resolver el algoritmo con el mismo esfuerzo independientemente del valor de K.



(a) Grafo original (b) Árbol generado

Figura 3.7: Generación del árbol final como parte del algoritmo BFS

3.2.3.1. Funcionamiento

El mismo algoritmo asume que para encontrar el primer camino de menor coste se use un algoritmo para ello, como puede ser el algoritmo de Dijkstra o el BFS. En las iteraciones sucesivas se procederá a realizar algunas modificaciones al camino previo para generar las rutas alternativas y ordenarlas por orden de coste. El algoritmo devolverá un array A^k con los K caminos explorados y en orden creciente, es decir, ordenado por el resultado de coste mínimo de cada trayecto analizado. Se explicarán, a continuación, los pasos a seguir para ejecutar el algoritmo:

1. Hallar el camino de menor coste para determinar A^1, es decir, la primera iteración del algoritmo con el camino de coste mínimo, a través de un algoritmo de búsqueda del camino más corto.

2. Modificar el camino previo y eliminar el enlace de menor coste para generar una ruta diferente entre el nodo fuente S y el nodo destino D. Almacenar el nuevo trayecto generado en un array temporal B.

3. Repetir el paso 2 para cada enlace del camino más breve hallado anteriormente.

4. Ordenar todos los caminos almacenados en orden ascendente y guardarlos en el array A. Repetir para *K-1* veces.

En conclusión, como se ha explicado con anterioridad, la importancia del algoritmo de Yen se basa en la eficiencia computacional directamente proporcional al valor de K, lo que atribuye al algoritmo una mayor eficiencia en comparación con algoritmos similares.

3.3. Aplicación a una red de transporte

La teoría de grafos tiene especial aplicación en problemas de la vida real. En este trabajo se intenta aprovechar este aspecto para modelar un sistema de planificación de rutas sobre una estructura de red.

La aplicación de la teoría de grafos a un problema responde principalmente a un estudio descriptivo y analítico de las redes para conocer su estructura y su desarrollo, destacando de manera concreta las propiedades topológicas con respecto a sus conexiones y a los puntos de encuentro en común. Bajo esta perspectiva, consideramos que se trata de un buen mecanismo de abstracción, sino el mejor, para evidenciar las relaciones funcionales e interacciones entre infraestructuras de transporte y el entorno social y cultural del territorio sobre el cual se emplazan.

Lo primero que se puede observar es la compenetración entre el grafo y la red. En este caso, se puede modelar fácilmente una red de transporte, convirtiendo las paradas de los autobuses en un nodo o vértice del grafo, mientras que el arco que conecta los dos nodos representaría el camino a recorrer para llegar de una parada a otra. Por esta razón, cobra sentido el uso de un grafo de coste, es decir, un grafo al cual podamos asignarle un peso o coste a cada arco del grafo. En nuestro caso, el peso podría representar la distancia entre una parada u otra, o simplemente la distancia en minutos que se tarda en recorrer el enlace, basándonos en alguna estimación.

Para que sea factible y viable la realización de un estructura de red de transporte real sobre un grafo, se va a utilizar un grafo dirigido para trabajar con las líneas de autobuses del servicio de transporte. En efecto, un enlace podría ser recorrido de manera bidireccional por un autobús, sin embargo, resulta evidente que cada línea tiene un sentido de dirección y, de esta manera, podemos distinguir entre una línea que vaya por el mismo trayecto pero con diferente sentido.

Para poder hacer computable el grafo, se ha de describir la red subyacente mediante una matriz de adyacencia o lista de adyacencia como se ha descrito anteriormente. Por las características de la red de transporte definida, y siendo un grafo dirigido, no será posible encontrar más de una sola entrada con los mismos valores. No obstante, sí será posible encontrar dos enlaces invertidos, es decir, una entrada con un nodo origen y destino o viceversa, o expresado con otras palabras otra entrada donde los roles sean intercambiados y donde el destino sea el origen.

Capítulo 4

Diseño e implementación del sistema

Tras haber descrito en los capítulos previos las características de un planificador de rutas y haber profundizado en los conceptos teóricos en teoría de grafos sobre los que apoyarse para comprender y desarrollar su operativa, seguidamente se propone un modelo de un planificador que proporcione la suficiente flexibilidad de configuración para permitir considerar múltiples parámetros, a fijar bien por parte del usuario o del gestor del sistema, y así evaluar la ruta óptima entre dos puntos teniendo en cuenta diferentes perfiles. Este planificador se puede considerar la base de un sistema experto en la gestión de rutas a través de múltiples medios de transporte.

Teniendo en cuenta los requerimientos y características identificadas para un planificador, la arquitectura global del sistema debe incluir los siguiente elementos: un motor de ejecución del algoritmo de búsqueda de rutas, una interfaz de interacción a través del cual configurar el sistema, enviar solicitudes, y obtener y analizar las respuestas de la ejecución del algoritmo, un recolector de información de las redes de transporte público subyacentes, la cual, en caso de ser estática, se almacenará localmente en una base de datos para acelerar los procesos de toma de decisión.

En la Figura 4.1 se puede observa la interacción entre los componentes lógicos definidos. Esta arquitectura sigue un paradigma de descentralización de los servicios y un acceso remoto a la plataforma como forma de promover la movilidad y el acceso ubicuo a la información, aspecto este ligado al uso del transporte público. El modelo de comunicación entre los diferentes elementos de dicha arquitectura seguirá las premisas de un modelo cliente-servidor, siendo el servidor quien aloje toda la lógica de proceso de forma que se libere al cliente de toda la capacidad de cómputo necesaria y se permita el acceso a las rutas desde cualquier dispositivo.

A continuación se afronta el proceso de desarrollo y validación del sistema propuesto aplicado al entorno del Servicio Municipal de Transportes Urbanos de Santander (TUS). Este proceso se puede dividir en varias etapas que comprenden implementar el algoritmo de búsqueda del camino más corto sobre un grafo e incorporar las interfaces para la ejecución remota de las instrucciones de control y gestión, recolección de los datos disponibles de la

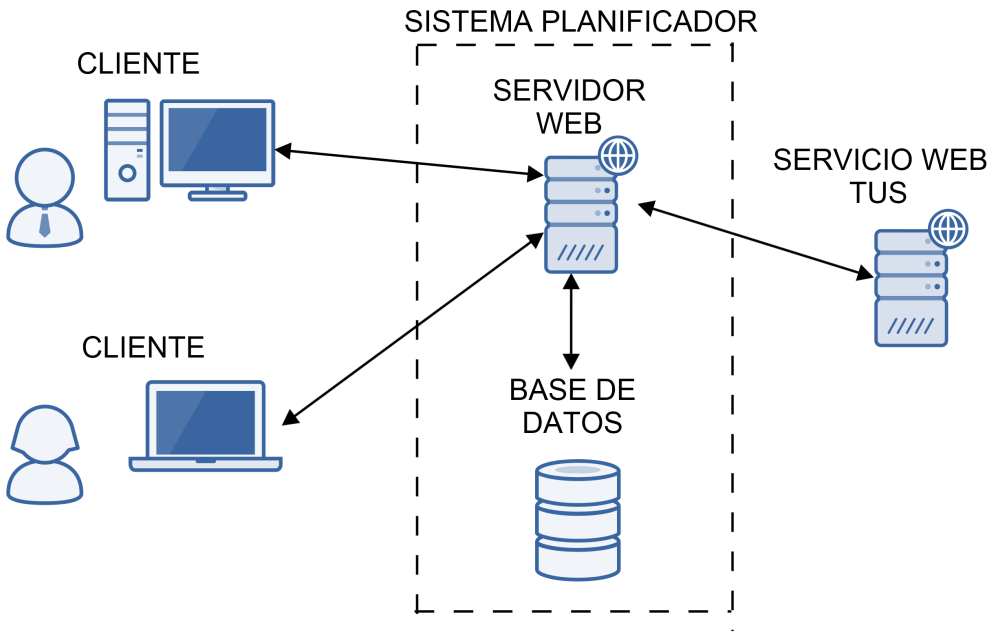


Figura 4.1: Arquitectura del sistema de planificación de rutas

red transporte (en este caso de TUS) mediante *Application Programming Interfaces* (API) públicas y, para finalizar, la adaptación y mejora del algoritmo para ofrecer soluciones inteligentes basadas en parámetros variables en tiempo real de la propia red como externas a ella.

4.1. Plataforma software Node.js

Existen multitud de alternativas a la hora de implementar soluciones de acceso remoto a información como la que se requiere en este proyecto. Desde soluciones basadas en código nativo como C/C++ pasando por Java, .NET, Python, etc., cada una tiene sus fortalezas y debilidades, tanto desde el punto de vista de curva de aprendizaje, de posibilidades de personalización, de robustez y de comunidad de soporte. Para este proyecto y dado que, en un inicio se requiere la implementación tanto de un entorno de servidor como de cliente, se ha decidido usar un lenguaje de programación empleado usualmente para la ejecución de funcionalidades en clientes web y, para el que recientemente se ha desarrollado una plataforma para la ejecución de aplicaciones de forma independiente al navegador. Este lenguaje de programación al que se hace referencia es *JavaScript* [17] y la plataforma *Node.js*.

Node.js [18] es un software de programación que permite crear aplicaciones en el lado del servidor de una forma fácil y rápida. Presenta una arquitectura orientada a eventos, totalmente asíncrona, lo que permite ejecutar simultáneamente más de una tarea y atender más eventos a la vez gracias a la máquina virtual *V8 JavaScript* de Google, la cual implementa para ejecutar los scripts. Este entorno es un proyecto desarrollado por el programador Ryan

Dahl, quien propuso su idea por primera vez en 2009. Su forma innovadora de afrontar y gestionar la implementación de aplicaciones y servidores ha despertado el interés de algunas de las compañías mas importantes del sector tecnológico, las cuales han decidido apoyar su proyecto e invertir en ello.

Comúnmente, se intenta describir a este sistema de desarrollo como *JavaScript* ejecutándose en el lado del servidor. *JavaScript* ha sido, por tradición, uno de los lenguajes más utilizados en el lado del cliente, debido a que permite comunicarse con el servidor de manera sencilla, complementándose de forma óptima con el lenguaje HTML (*HyperText Markup Language*). Sin embargo, el planteamiento cambia a la hora de ejecutar funciones y operaciones en el lado del servidor. *Node.js* aprovecha los puntos fuertes del lenguaje *JavaScript* y su popularidad entre los desarrolladores para destacar en un entorno donde otros lenguajes, como pueden ser *PHP*, *ASP*, *JSP* o *Ruby*, tienen más importancia y están más implementados.

Sin duda alguna, resulta importante destacar la naturaleza asíncrona del lenguaje utilizado a la hora de realizar la comparación con lenguajes que se aplican a soluciones que se ejecutan en el lado del servidor. La programación asíncrona permite no bloquear la ejecución de un programa y atender a más eventos a la vez, justo al contrario de lo que sucede mediante programación síncrona. Este factor aporta numerosas ventajas a la hora de ejecutar la labor de un servidor, como atender peticiones HTTP (*HyperText Transfer Protocol*). En este caso en concreto, el servidor no tendría la necesidad de esperar para atender cada petición de forma individualizada y tampoco iniciaría otro proceso por cada petición nueva, sino que atendería simultáneamente todas las peticiones que se reciban. Señalar que para facilitar la gestión y ejecución de las tareas las funciones asociadas a estas llamadas asíncronas es recomendable que sean de ejecución corta y eficiente.

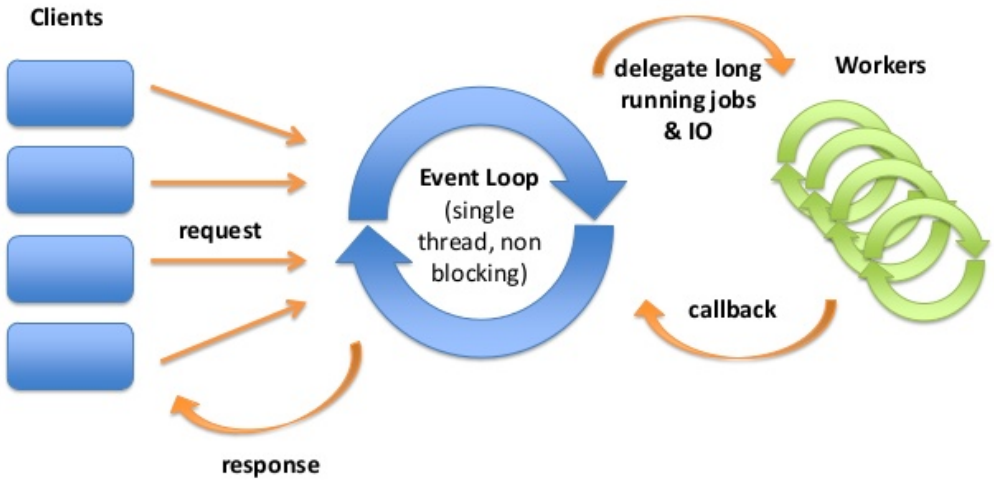


Figura 4.2: Entorno orientado a eventos de *Node.js* [19]

Como se puede observar en la Figura 4.2, las solicitudes vienen gestionadas por un único hilo, mientras que las operaciones de entradas y salidas se procesan en paralelo de manera no bloqueante. Considerando el entorno en el cual se va a desarrollar el sistema, es decir la implementación de un *journey planner* para el transporte público de una ciudad, puede suponer una gran ventaja. La programación orientada a eventos permite aumentar la eficiencia de forma drástica y, para una aplicación Web, este parámetro puede resultar fundamental puesto que permite reducir el tiempo de respuesta del servidor al cliente. Esta ha sido, entre otras, una de las principales razones para usar este lenguaje de programación frente a otros.

4.1.1. *Callbacks*

Los entornos de programación de aplicaciones basados en una gestión asíncrona de las funciones operan mediante lo que se conocen como *callbacks* o funciones de retorno. Estas funciones *callbacks* se ejecutan una vez llamada la tarea asíncrona y gestionan el resultado de dicha operación. De este modo, el programa no se bloquea, como sucede en el caso de emplear operaciones síncronas, sino que las diferentes tareas se ejecutan en paralelo sin interrumpir el flujo de la aplicación. Aunque usualmente las aplicaciones se realizan empleando código que se ejecuta de manera secuencial, considerando sólo ciertas tareas para su ejecución asíncrona o paralela, como se ha comentado, *Node.js* supone un cambio total a esta filosofía. En este sentido, resulta indispensable, a la hora de programar en *Node.js*, organizar el código de forma adecuada para que la anidación de *callbacks* y la gestión de las variables de los objetos permita obtener el resultado deseado.

4.1.2. Node Package Manager (NPM)

Una característica interesante de *Node.js* es su integración con el software *Node Package Manager* (NPM). Este servicio permite al programador descargar paquetes, más conocidos como librerías, creados, publicados y compartidos por otros programadores, los cuales permiten añadir funcionalidades de forma sencilla e intuitiva. Esta filosofía está muy vinculada con el paradigma de código abierto. En este trabajo, se ha aprovechado esta funcionalidad para facilitar la implementación del algoritmo de búsqueda y hacer así que el código resulte más robusto, ordenado, legible y fácil de entender.

4.2. Implementación del algoritmo de Dijkstra

Para poder determinar la mejor opción para desplazarse de una parada a otra, se ha decidido emplear el algoritmo de Dijkstra, cuya descripción se ha realizado en el capítulo 3. De entre todos los planteados, las características que presenta este algoritmo han hecho que se selecciones como la alternativa para cubrir las exigencias del sistema de planificación de rutas óptimas bajo desarrollo. Así, la facilidad de implementación y su sencillez a la hora de entender el proceso de búsqueda del camino más corto han desempeñado un papel clave para decantarse a favor de este algoritmo. Del mismo modo, objetivamente, la velocidad de computación a la cual el algoritmo procesa la información para buscar el camino óptimo es

suficientemente elevada para poder desempeñar sin problemas las tareas asignadas a este algoritmo.

Haciendo uso de la facilidades que incorpora *Node.js* y aprovechando *NPM* se realizó una búsqueda de soluciones que implementaran de forma eficiente el algoritmo de Dijkstra. De entre las varias opciones disponibles se seleccionó la librería *dijkstra-edsger* [20] por su simplicidad a la hora de describir el grafo y de las interfaces de interacción. Emplea como entrada una lista de adyacencia, lo que nos permite reducir la cantidad de memoria necesaria, algo importante dado el elevado número de nodos, en este caso, paradas que forman parte de la red. Requiere que la lista de adyacencia se asocie a un *array JavaScript* de dos dimensiones, especificando por cada enlace a crear el nodo origen, el nodo destino y el coste del enlace como se muestra en la Tabla 4.1. Una vez construida la lista de adyacencia, automáticamente se determina el grafo sobre el cual se va a realizar la búsqueda (Figura 4.3).

Tabla 4.1: Ejemplo de lista de adyacencia

Nodo origen	Nodo destino	Coste
1	2	3
1	3	2
2	3	4
2	5	3
3	4	2
4	5	1

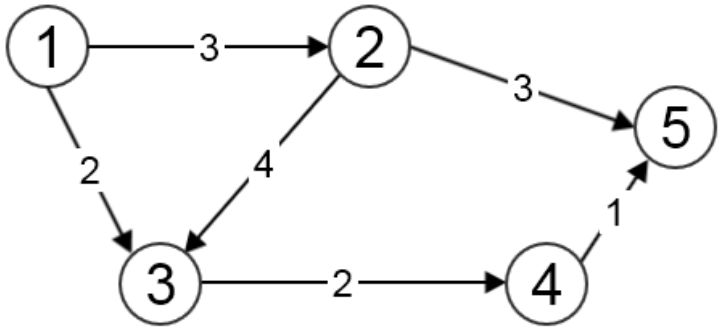


Figura 4.3: Grafo obtenido a partir de la lista de adyaciencia de la Tabla 4.1

El siguiente paso será encontrar el camino más corto para llegar de una parada a otra empleando el algoritmo de Dijkstra. Para poder ejecutar la función de Dijkstra de la librería seleccionada, se ha de establecer el nodo fuente S y el nodo destino D, además de pasar a la función nuestra lista de adyacencia anteriormente definida. Considerando la lista de adyacencia de la Tabla 4.1 y suponiendo se quiera recorrer de la forma más breve el camino entre el nodo *1* y el nodo *5*, el resultado sería el mostrado en el grafo de la Figura 4.4. La ejecución de la función de la librería *dijkstra-edsger* para estos parámetros retorna un objeto que aloja un *array* con la lista ordenada de nodos a atravesar, así como el coste total del recorrido.

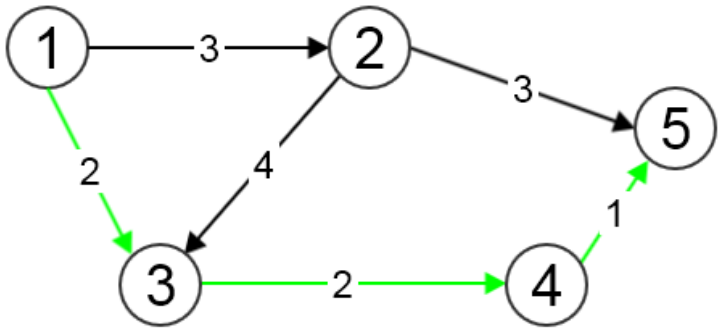


Figura 4.4: Camino más corto encontrado por el algoritmo de Dijkstra

4.3. Generación del grafo de la red transporte

Una vez se ha evaluado el correcto funcionamiento de la implementación del algoritmo de Dijkstra mediante un grafo de prueba, el siguiente paso es acometer su validación empleando un grafo más complejo y basado en los datos reales de la red del TUS. En este sentido, primeramente se tendrá que generar el nuevo grafo a partir de la información disponible de la localización de las paradas y las líneas de autobús que circulan por ellas. Posteriormente, este grafo junto con los nodos origen y destino se introducirá en el procedimiento para la obtención de la ruta más adecuada, pero esta vez teniendo en cuenta no solo el coste entre paradas sino también que autobuses paran en cada una, de forma que se establezcan intersecciones y potenciales puntos de transbordo.

El TUS dispone de una interfaz basada en servicios web a partir de la cual se tiene acceso en tiempo real a la información del sistema de transporte. Esta información incluye de una parte datos que se pueden considerar estáticos como son las líneas, la localización de las paradas y la asignación de paradas a cada línea. De otra, habilita el acceso a estimaciones de tiempo de llegada de los próximos autobuses a cada parada y la posición actual de cada uno de los autobuses en servicio. En la Tabla 4.2 se enumeran las diferentes funciones proporcionadas y una pequeña descripción de cada una.

4.3.1. Acceso a la información

Un servicio web o *web service* define un entorno que permite intercambiar información entre aplicaciones mediante conexiones basadas en HTTP. Existen varios modelos o protocolos para la implementación de servicios web, como por ejemplo, XML-RPC (*eXtensible Markup Language* - Remote Procedure Call), SOAP (*Simple Object Access Protocol*) o REST (*Representational State Transfer*). Los servicios web del TUS están implementados por medio de SOAP.

SOAP es un protocolo que establece la comunicación para el intercambio de datos y, adicionalmente, engloba una arquitectura compuesta por varios elementos independientes, como pueden ser WSDL (*Web Service Definition Language*) y UDDI (*Universal Descrip-*

Tabla 4.2: Funcionalidades empleadas del servicio de información del TUS

Estructura	**Estimaciones**
• **GetLineas:** Devuelve las líneas del sistema. • **GetNodosMapSublinea:** Devuelve el nombre, el número y la posición de las paradas de la línea solicitada. • **GetRutasSublinea:** Devuelve la ruta y las distancias entre paradas de la línea solicitada.	• **GetPasoParada:** Devuelve información sobre la línea y la distancia o el tiempo que tardan los autobuses en llegar a la parada solicitada.

tion, Discovery and Integration), para ejecutar el proceso y ofrecer un soporte completo a cualquier tipo de servicio. Los mensajes SOAP se envían en formato XML y, normalmente, sobre protocolo HTTP, donde se encapsula el mensaje. Estos mensajes se componen de tres elementos principales: un *Envelope* el cual describe la estructura del mensaje, un *Header* como segmento opcional para introducir detalles sobre la seguridad o el protocolo de routing y, finalmente, un *Body* donde se transporta el contenido del mensaje. La descripción de las funcionalidades y elementos que incorpora el servicio web SOAP se realiza mediante WSDL, donde se define el formato de los datos y llamadas y los requisitos a cumplir para interactuar con el servicio web.

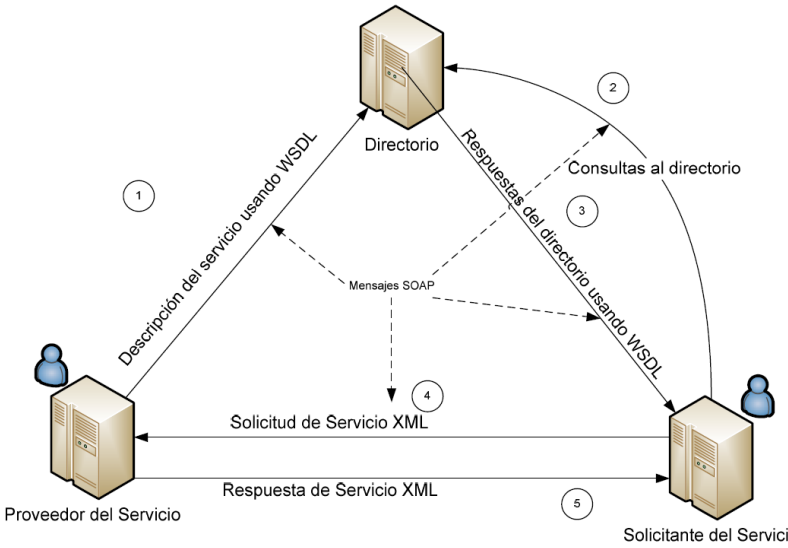


Figura 4.5: Funcionamiento básico de un servicio web basado en SOAP [22]

En la Figura 4.5 se muestra un resumen del funcionamiento básico de un servicio web basado en SOAP y las interacciones entre los diferentes elementos del sistema y los protocolos empleados para ello. Así, existirá un directorio de servicios donde a través de UDDI se

describen y publicitan las características específicas de los servicios. Localizado el servicio deseado, mediante su descripción WSDL se define la forma en la que operar contra el servicio para realizar las solicitudes que la aplicación requiera, estableciendo una comunicación directa entre el cliente y el proveedor del servicio mediante SOAP .

En una primera fase, para conocer el funcionamiento de los servicios web del TUS y los modelos de datos que intercambian, se ha empleado el programa SoapUI [21]. Se trata de un programa *freeware* que facilita la interacción con servicios web SOAP o REST. Proporcionándole la descripción WSDL, genera un modelo de mensajes asociados a las funcionalidades exportadas por el servicio web. En la Figura 4.6 se muestra un ejemplo de una solicitud realizada a los servicios web del TUS para la obtención de las paradas asignadas a una determinada línea.

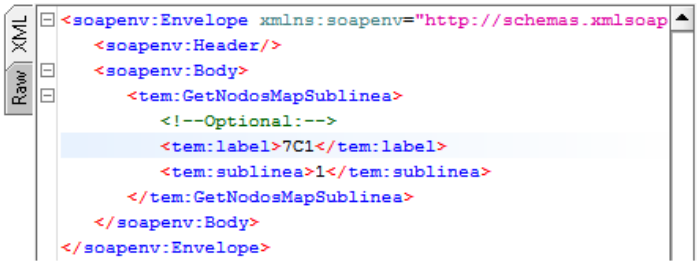


Figura 4.6: Mensaje SOAP para la solicitud de las paradas de una línea

El uso de SoapUI ha permitido comprender mejor la metodología de operación del servicio web del TUS y estimar los tiempos de respuesta, el número de solicitudes necesarias para obtener la información mínima para poder realizar el planificador y el volumen de información devuelta. Tras este análisis, se puede afirmar que la latencia del servicio es mínima, con tiempos de respuesta son prácticamente inmediatos. No obstante, se ha detectado que el número de solicitudes y la información intercambiada es elevada. Por esta razón, para evitar un acceso redundante e innecesario a la plataforma del TUS, se ha establecido que la información que se considera estática (estructura de la red) se solicite la primera vez que se inicia el programa y se aloje en una base de datos local. Periódicamente en momentos de baja carga, se podrá analizar si ha habido cambios (nuevas líneas o paradas, modificaciones en los recorridos, etc.) y se incorporarán dichos cambios a la versión local. De esta forma se logra acelerar los tiempos de ejecución a la hora de generar el grafo.

La implementación del acceso al servicio web mediante *Node.js* se ha realizado apoyándose en la librería *soap* [23]. Gracias a esta librería, se gestiona de forma sencilla el proceso de creación de los mensajes y el envío de las peticiones o llamadas a los procedimientos remotos. En el Cuadro 4.1 se muestra un ejemplo del código empleado para realizar la solicitud de las paradas asociadas a una línea, que anteriormente se mostró mediante SoapUI (ver la Figura 4.6).

Los datos recibidos del servicio web se encuentran en formato XML. Para facilitar el procesamiento posterior en el programa y poder usarlos con las interfaces de la librería de ejecución del algoritmo de Dijkstra, la información recibida se encapsula en formato JSON (*JavaScript Object Notation*).

Cuadro 4.1: Código para enviar una petición SOAP como la de la Figura 4.6

```
soap.createClient(urlEstructura, function(err, client) {
        if (err) throw err;
        client.Estructura.EstructuraSoap.GetNodosMapSublinea({label:'7C1',
             sublinea:'1'}, function(err, response) {
                if (err) throw err;
                //Do something with response
        });
});
```

4.3.2. Replica en base de datos local

Tal como se ha introducido, la información relativa a la estructura de la red puede que sufra pequeñas variaciones en el tiempo y sin embargo supone un gran volumen de datos. Es por esto que para mejorar la operativa de la solución en desarrollo, una vez obtenida toda la información del servicio web, se procede a guardarla en una base de datos local para su manejo posterior a la hora de generar las rutas. Como motor para almacenar los datos, se ha utilizado *MySQL*, un sistema para gestionar una base de datos relacional y multiusuario. La gestión de la base de datos desde *Node.js* se realiza mediante la librería *mysql* [24], que permite establecer conexiones a la base de datos *MySQL* y realizar peticiones o *queries* en SQL (Structured Query Language) para crear tablas, insertar o borrar datos, etc.

Para poder manejar de forma eficiente la enorme cantidad de información recibida por el servicio web de la estructura de la red, se han creado tres tablas distintas en la base de datos: *Lineas*, *Paradas* y *Red*. En la Figura 4.7 se muestran las relaciones entre las tablas definidas.

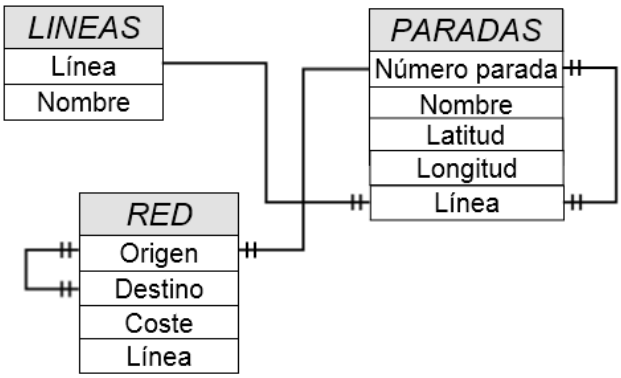


Figura 4.7: Relaciones entre tablas de la base de datos

La tabla *Lineas* contiene todas las líneas disponibles en el TUS con el nombre asociado a cada una. Esta tabla, a la cual corresponde la Tabla 4.3, permite presentar de manera ordenada y legible la información al usuario final. El disponer del listado de líneas y su identificador es un requisito fundamental para posteriormente poder hacer las peticiones necesarias al servicio web para obtener las paradas asociadas y así crear y rellenar la tabla *Paradas* que se analiza a continuación.

Tabla 4.3: Líneas de autobuses del TUS

Línea	Nombre	Línea	Nombre
1	PCTCAN - VALDENOJA	E30	SE ESTACIONES-FERIAS
2	CORBÁN - CONSUELO BERGÉS	E31	S.E. EL ALISAL - SARDINERO
3	OJAIZ - PIQUÍO	E1	VALDECILLA - GREGORIO MARAÑÓN
4	BARRIO PESQUERO - PIQUÍO	E2	S.E. INTERMODAL
11	VALDECILA - C/ALTA	E3	SE MIRANDA - INSTITUTOS
12	CARREFOUR - CANALEJAS	E4	SE SAN MARTÍN - INSTITUTOS
13	LLUJA - CUETO	E5	SE C/ALTA - INSTITUTOS
14	ESTACIONES - AV.VALDECILLA	E9	SE ESTACIONES - CEMENTERIO
15	ESTACIONES - EL FARO	5C1	MIRANDA/PLZ. ITALIA C1
16	PLAZA DE LOS REMEDIOS	5C2	MIRANDA/PLZ. ITALIA C2
17	CIRIEGO - CORBÁN - ESTACIONES	6C1	COMPLEJO C1
18	PUERTOCHICO - MONTE	6C2	COMPLEJO C2
19	ESTACIONES - RICARDO L.ARANDA	7C1	LUIS QUINTANILLA
20	ESTACIONES - BARRIO LA TORRE	7C2	JOAQUÍN BUSTAMANTE
21	CENTRO DE SALUD - PUERTOCHICO	N1	CORBÁN - G. ATECA por Valdecilla
23	ESTACIONES - CAMARREAL	N2	CORBÁN - COMPLEJO por Gral. Dávila
E30	SE ESTACIONES - FERIAS	N3	PEÑACASTILLO -PLAZA DE ITALIA

La tabla *Paradas* contiene todas las paradas de la red del TUS. El servicio web no dispone de una funcionalidad para recoger todas las paradas, sino que únicamente habilita su obtención para una determinada línea. La tabla *Paradas* incluye además del identificador de cada parada asignado por TUS, el nombre asociado y la posición de cada parada en coordenadas geográficas (latitud y longitud). El servicio web retorna las coordenadas en formato ETRS89 (x e y), pero la mayoría de los sistemas de información geográfica (GIS, *Geographic Information System*) requieren que éstas se encuentren en formato WGS84. La conversión entre formatos no resulta trivial, pues además de la conversión matemática hay que realizar correcciones basadas en tablas proporcionadas por los países u organismos oficiales [25]. En este sentido, el Ministerio de Fomento del Gobierno de España a través del Instituto Geográfico Nacional proporciona un servicio web de transformación de coordenadas [26]. Haciendo uso de este servicio se han transformado todas las coordenadas proporcionadas por el TUS y se han almacenado en la base de datos.

Dado que cada parada se asocia a una línea, las tablas *Paradas* y *Lineas* deberían estar vinculadas. Para poder relacionar ambas tablas, se ha utilizado una *FOREIGN KEY* como restricción entre los números de las líneas. Este método se ha usado para prevenir el hecho de que alguna fila sea eliminada accidentalmente y evita, por ejemplo, que por error se inserte un valor de línea equivocado y que no corresponda a ninguna línea de la tabla *Lineas*. Sin embargo, en la tabla *Paradas* se puede encontrar la misma parada repetida en más ocasiones, puesto que una parada puede pertenecer a varias líneas diferentes. Por este motivo, se ha establecido también una *UNIQUE KEY* en la tabla *Paradas* entre el número de la parada y la línea.

En la Tabla 4.4 se muestra un extracto de la tabla *Paradas* de la base de datos definida.

Finalmente, con el nombre *Red*, se define la tabla que corresponde a la lista de adyacencia que se ha utilizado para crear el grafo de la red de transporte. Como se ha explicado en el capítulo 3, la lista necesita un nodo origen, un nodo destino y el coste del enlace asignado a

Tabla 4.4: Paradas del TUS

Número de Parada	Nombre	Latitud	Longitud	Línea
169	VALDENOJA, 25	43.48090891	-3.79274671	1
28	LOS AGUSTINOS	43.47844172	-3.79142027	1
30	PIQUÍO	43.47370168	-3.78469989	1
449	VALDECILLA SUR	43.45462144	-3.82747004	12
13	AYUNTAMIENTO	43.46148618	-3.81003906	12
149	CATEDRAL	43.46142961	-3.80721161	12
19	AVENIDA REINA VICTORIA	43.46453008	-3.78274644	7C2
67	LOS CASTROS, 20	43.47338179	-3.7880273	7C2
70	ESCUELA DE CAMINOS	43.4712674	-3.79950676	7C2

cada pareja de nodos. En nuestro caso, los identificadores de los nodos se corresponden con los identificadores de cada parada, mientras que el coste se define como la distancia entre paradas, que ponderándolo por la velocidad del autobús nos dará el tiempo que se tarda en recorrer el trayecto entre dos paradas.

Dado que las paradas van a ser los nodos de los arcos, la información alojada en la tabla *Paradas* se referenciará en la *Red*. Para ello se ha empleado una *FOREIGN KEY*. Esta tiene la función de evitar que en la lista de adyacencia se definan nodos que no pertenecen a la red, es decir, que son paradas que no existen y que por tanto no aparecen en la tabla *Paradas*. Además se ha definido como *UNIQUE KEY* a la dupla origen-destino de forma que se asegure que no se insertará un mismo arco múltiples veces. Puesto que estamos trabajando con un grafo dirigido que evidencia el sentido de cada enlace, la *UNIQUE KEY* permite evitar crear enlaces innecesarios y, de esta manera se evita sobrecargar el grafo, lo que tendría una repercusión sobre la velocidad de búsqueda del algoritmo.

La tabla *Red* se completa con una columna adicional que se corresponde con las líneas autobuses que pasan por las paradas de destino. De esta manera se puede comparar con las líneas que pasan por la parada anterior e identificar las líneas en común, es decir, las que pasan por el arco. Esta columna refleja una restricción a tener en cuenta a la hora de realizar la búsqueda del recorrido óptimo y que determinará la necesidad de realizar transbordos, etc.

En la Tabla 4.5 se incluye un extracto de la tabla *Red* de la base de datos definida, donde se muestran diversos arcos de la red de transporte y las líneas asociadas a ellos.

4.3.3. Interfaz del planificador

Tras definir las estructuras de datos que permitirán generar el grafo de la red de transporte público de la ciudad de Santander, resta ahora evaluar el correcto funcionamiento del sistema y exportar su funcionalidad para el acceso a través de sistemas remotos.

Para ello se ha definido una interfaz web que, por un lado permita acceder a la función de cálculo de la ruta más adecuada y, por otro permita visualizar las opciones del planificador y el resultado final al usuario. Como interfaz de acceso al servicio se ha optado por definir

Tabla 4.5: Arcos del grafo de la red de transporte de TUS

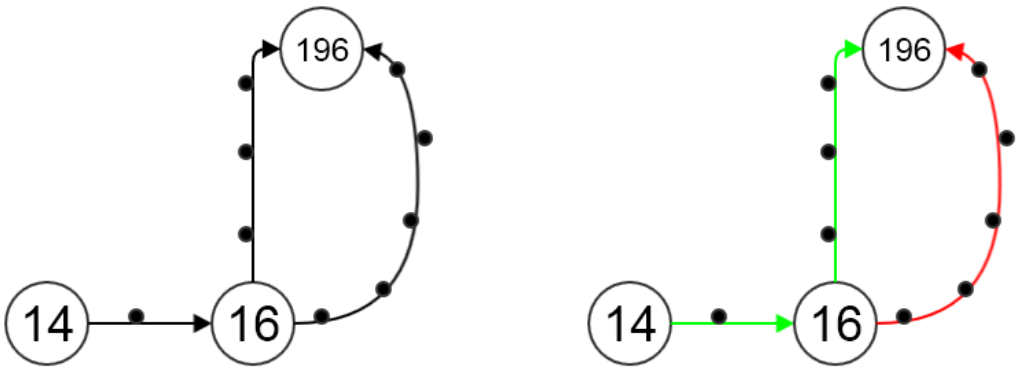
Origen	Destino	Coste	Línea
164	213	2	20, 6C1
76	39	2	20, E30, 6C1
42	356	2	20, E30
39	16	2	21
107	293	3	23
261	195	5	E30
16	69	5	E2
135	136	2	E3, 5C2, N2
140	141	2	E3

un servicio web REST, definido en el Cuadro 4.2 que exporte un recurso para el cálculo de la ruta a partir de una parada origen y un destino.

Cuadro 4.2: Definición del servicio REST

```
host:port/autobuses?origen=NOMBRE+PARADA+ORIGEN&destino=NOMBRE+PARADA+DESTINO
```

Para implementar este servicio y la interacción de la interfaz visual del cliente, se ha hecho uso de la fácil compenetración entre *Node.js* y *JavaScript* ejecutado en un cliente web. Así, se ha procedido a conectar mediante *sockets* web el cliente con el servidor. En el entorno del cliente, se han utilizado las funcionalidades proporcionadas por *JQuery*, mientras que en el lado del servidor se ha empleado el paquete *io.socket*.



(a) Grafo generado (b) Solución del grafo original

Figura 4.8: Grafo de prueba generado sobre la red de transporte TUS

Para evaluar el correcto funcionamiento del sistema y analizar que los tiempos de respuesta son aceptables para un uso cotidiano, considerando únicamente como parámetro de decisión la distancia del recorrido, se ha procedido a realizar la petición de ruta entre las paradas de 14 y 196. Como se observa en la Figura 4.8(a) existen dos posibles recorridos. Como usuario del TUS y conocedor de la red de paradas, intuitivamente se conoce el óptimo.

No obstante, un turista o visitante de la ciudad desconoce estos aspectos y por tanto tendría que hacer uso del planificador. La ejecución del planificador con la configuración definida reporta, como era de esperar, el camino más corto con una distancia de aproximadamente tres kilómetros, tal como se puede ver en la Figura 4.8(b). El tiempo de respuesta ha sido de aproximadamente dos segundos, lo que se considera más que adecuado para este tipo de soluciones.

La página a través de la cual el usuario accede al planificador es sencilla y de fácil comprensión, tal como se muestra en la Figura 4.9. El usuario tiene la posibilidad de escoger una línea para poder visualizar las paradas disponibles, y, a su vez, puede elegir qué parada establecer como punto de origen y qué parada como punto de destino. Sucesivamente, el servidor procesará la información recibida, calculando el trayecto de menor coste a través del algoritmo de Dijkstra y presentando los resultados obtenidos.

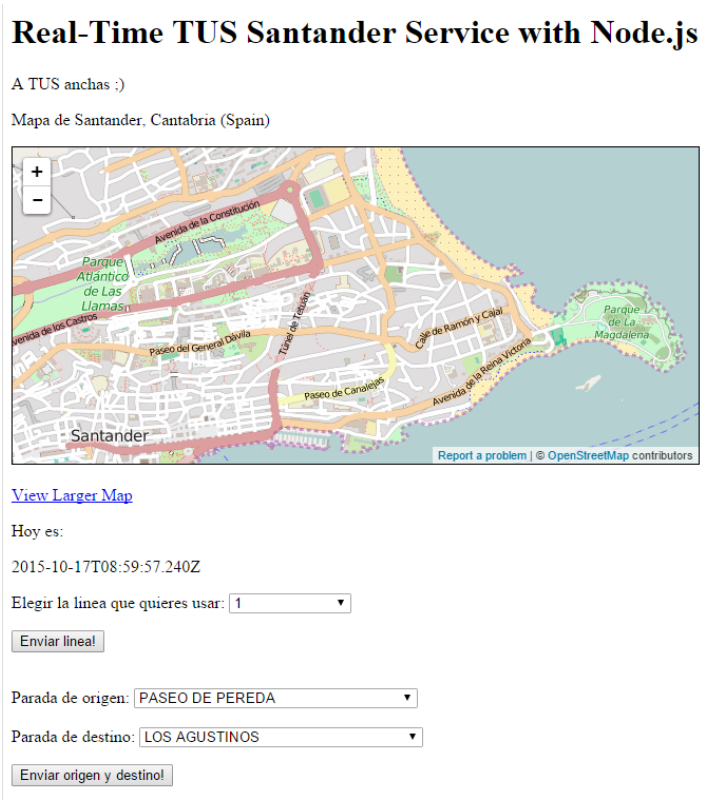


Figura 4.9: Interfaz de usuario del planificador de rutas

Hasta el momento no se han considerado más parámetros de coste que la propia distancia entre paradas. En el siguiente capítulo se incluirán una serie de mejoras sobre el procedimiento básico descrito de forma que se puedan ofrecer opciones de trasbordo entre líneas o andando, estimaciones de duración del trayecto, etc.

Capítulo 5

Mejoras sobre el planificador básico

A la hora de aplicar la teoría de grafos y más concretamente el algoritmo de Dijkstra a una red de transporte público de una ciudad, surgen diferentes necesidades que hacen que el modelo tradicional no sea suficiente para ofrecer la ruta óptima para desplazarse de un punto a otro. Esto es así puesto que el grafo no es estático y debe adaptarse en tiempo de ejecución a los medios de transporte disponibles, así como a las rutas definidas para cada uno de esos medios. En este sentido, dos nodos de un grafo pueden estar conectados pero en realidad el trayecto final se debe recorrer en diferente medios.

Adicionalmente, situaciones como la congestión del tráfico hacen que el coste entre dos paradas no se pueda considerar como una constante (la distancia entre ellas) sino que esa distancia, traducida a tiempo, será variable dependiendo del momento en el que se realice la solicitud. Es por esto que se hace necesario la inclusión de modificaciones del grafo inicial en tiempo real, que supondrán una mejora en el cálculo de la ruta mínima.

A continuación se acomete la descripción de un conjunto de mejoras del planificador desarrollado en las líneas anteriormente identificadas.

5.1. Detección de un trasbordo

Uno de las cuestiones principales a la hora de analizar las soluciones de búsqueda del camino más corto y adaptarlas a la estructura de la red de transporte es determinar los puntos de trasbordo, puesto que para recorrer la distancia entre dos puntos se pueden tener que emplear varios autobuses. En este sentido, el grafo conexo representando una red de transporte, lo es pero puede incluir ciertas restricciones a la hora de considerar una ruta entre los puntos. Además el número de transbordos es una variable adicional a tener en cuenta a la hora de determinar el camino más corto, ya que en ciertas redes de transporte, como es el caso de la del TUS, únicamente se puede realizar un trasbordo.

Para poder solucionar esta problemática, se ha elaborado un modificación sobre el algoritmo básico empleado. Esta adaptación consiste principalmente en recalcular la ruta más corta a medida que se avanza en el camino, es decir, una vez obtenida la ruta más breve y conocer la siguiente parada, ésta se tomará como parada inicial y se recalculará la ruta para

llegar al destino teniendo en cuenta las líneas disponibles y los nuevos costes de los enlaces.

Antes de profundizar en el análisis del algoritmo es preciso comentar algunas características o premisas que se han supuesto. Por un lado el proceso se inicia considerando que el usuario se montará en el primer autobús que llegue a la parada origen y con ello se fija la línea empleada. En el resto del proceso se calculará la ruta, teniendo en cuenta que se considerarán transbordos siempre que el tiempo para llegar al destino se reduzca por el hecho de tomar otra línea. Esta solución puede incurrir en un número elevado de transbordos que exceda el máximo de la red de transporte. Es por esto que para analizar otras opciones de recorrido, del mismo modo que se toma como punto de partida inicial el primer autobús que llegue a la parada origen, la ejecución del proceso global se podría repetir considerando otras líneas. Con esto se ofrecerá al usuario otras opciones que pueden resultar en un menor tiempo de viaje o en un menor número de transbordos, pero que incurren en un mayor tiempo de espera en la parada inicial. Esta última opción se describe más adelante.

Entre las modificaciones incluidas sobre el procedimiento básico descrito en el capítulo 4 se incluye el hecho de incorporar costes de enlace en tiempo real, para lo que se solicita a través del servicio web del TUS los tiempos de paso de los autobuses por cada una de las paradas. Mediante el programa *SoapUI* se realizaron pruebas sobre el servicio web del TUS para conocer qué funciones y cómo se deben de llamar para obtener la información temporal en cada parada. En este sentido, la función *GetPasoParada*, pasando como parámetro el identificador de parada, retorna el tiempo, en minutos, que tardan en llegar a esa parada los diferentes autobuses asociados a las líneas. Esta información se solicita en cada iteración del algoritmo y resulta importante para determinar si es conveniente o no hacer un transbordo a otra línea.

No obstante, no es suficiente con conocer cuándo llega un autobús a una parada, tomando como referencia el momento en el se que realiza la solicitud de ruta. Dado que el usuario hipotéticamente irá desplazándose de parada en parada, habrá que tener en cuenta una estimación del tiempo que se tarda en llegar a dicha parada para conocer realmente cuándo llegarán los autobuses a esa parada para el viajante y determinar si compensa realizar el trasbordo o no. Este tiempo se obtiene a partir del coste del subtrayecto al que se le suma un tiempo adicional equivalente al tiempo que un autobús consume cada vez que se detiene en una parada (aproximadamente 1 minuto). Gracias a esta modificación se consigue evitar que el algoritmo escoja una línea la cual no podrá tomar el usuario por no estar aún en la parada.

Teniendo en cuenta todo lo anterior el procedimiento consta de las siguientes acciones:

1. Solicitar al servicio web del TUS los tiempos que tardan los autobuses de cada línea en llegar a la parada actual.
2. Escoger el autobús de la línea que llegue primero a la parada y sumar su tiempo al arco en el grafo.
3. Ejecutar el algoritmo de Dijkstra, pasando como elementos la parada actual a modo de nodo de origen, la parada destino y el grafo con las modificaciones hechas considerando la información obtenida en el paso *2*.
4. Una vez que el algoritmo de Dijkstra devuelva el camino más corto, recorrer el camino analizando cada parada.

a. Mientras la parada siguiente tenga únicamente un nodo adyacente conectado para el arco de salida, considerar la siguiente parada del camino.
b. Si la parada siguiente del camino tiene más de una salida o nodo adyacente, volver al paso *1*.
c. Si la parada siguiente resulta ser la parada de destino, se termina el algoritmo.

5. Calcular el coste total del trayecto.
6. Retornar el resultado al usuario.

Para poder proporcionar al usuario final toda la información relativa a transbordos, el tiempo que se tarda en recorrer el camino y las paradas por las cuales se pasará, el algoritmo almacena la información a través de diferentes variables las cuales nos proporcionarán, al finalizar el algoritmo, todos los datos conseguidos.

Para poder entender mejor la ejecución de cada iteración del algoritmo en cuestión, se procede seguidamente a explicar paso a paso el procedimiento para detectar el trasbordo apoyandose en diversos diagramas. En la primera iteración del algoritmo habrá que solicitar los tiempos de llegadas de los autobuses para la parada de origen. Como criterio de elección básico se va a escoger la línea de autobús que llegue antes a la parada de origen. Se puede ver un ejemplo de un grafo de prueba en la Figura 5.1, donde se comienza en el nodo origen S y termina el recorrido en el nodo de destino D. Se va a tomar la línea Roja siendo la primera en llegar al origen S, y seguidamente se activa el algoritmo de Dijkstra.

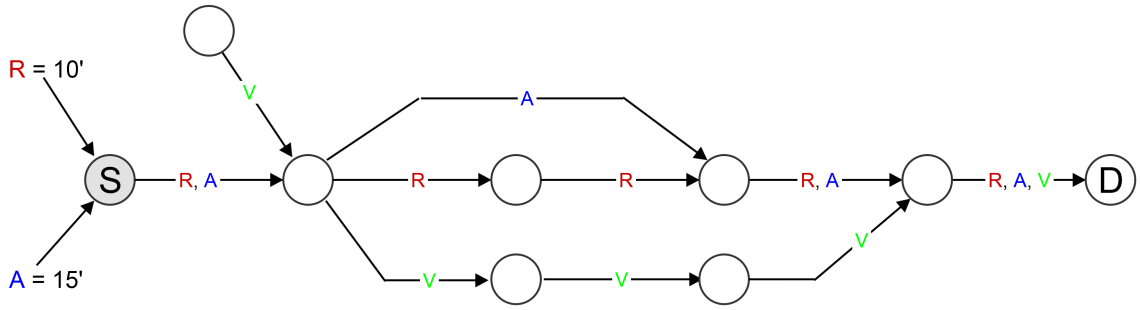


Figura 5.1: Primera iteración del algoritmo para detectar un transbordo

Una vez ejecutado el algoritmo de Dijkstra, se conoce cual es la siguiente parada y se obtiene un potencial camino de coste mínimo. En la segunda iteración el proceso resulta idéntico, solo que, en este caso, para ir a la siguiente parada se hace mediante un autobús. Esto implica que, a la hora de volver a ejecutar el algoritmo de Dijkstra para recalcular el camino más corto, el tiempo a sumar al arco del grafo sobre cuál continúa nuestra línea será cero. Sin embargo, esto no significa que este sea el recorrido más rápido. Por ejemplo, en la Figura 5.2, si se supone que el trayecto por la línea Roja resulta mucho más largo que el camino de la línea Azul, puede que convenga hacer un transbordo y cambiar a la línea Azul aunque tengamos que esperar la llegada de dicho autobús en la actual parada S.

En las siguientes paradas de la línea Azul, como se puede ver en la Figura 5.3 de abajo, solo se tiene un nodo adyacente al cual podemos ir y, como se ha explicado anteriormente en

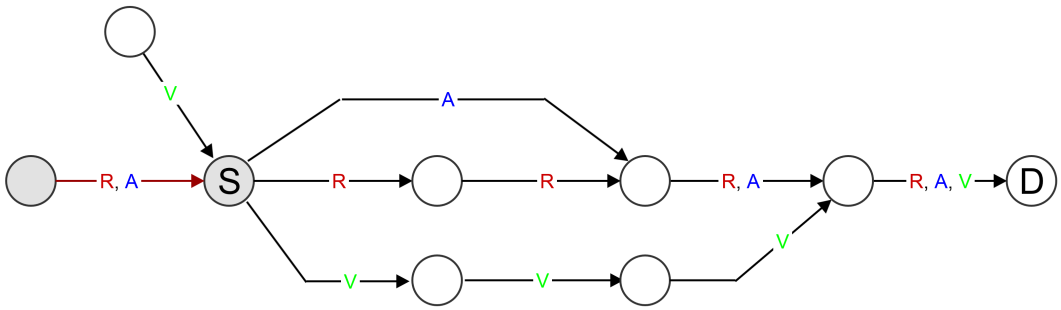


Figura 5.2: Segunda iteración del algoritmo para detectar un transbordo

el algoritmo, directamente se pasa a la siguiente parada del trayecto y, en el grafo de ejemplo, se llega directamente al destino D por la línea Azul, como se demuestra en la Figura 5.4.

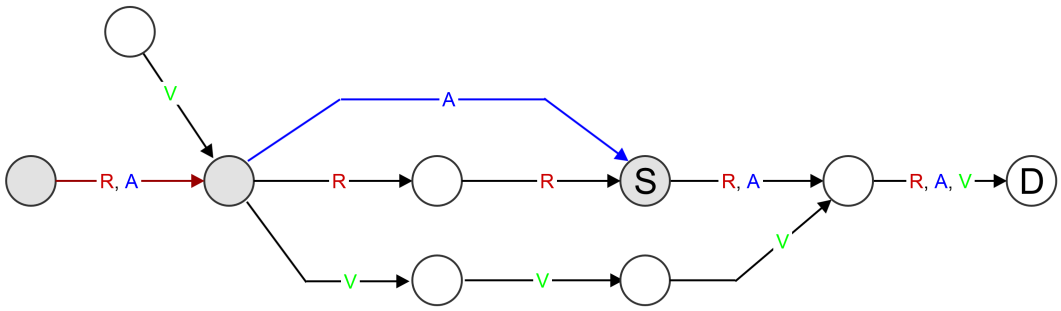


Figura 5.3: Tercera iteración del algoritmo para detectar un transbordo

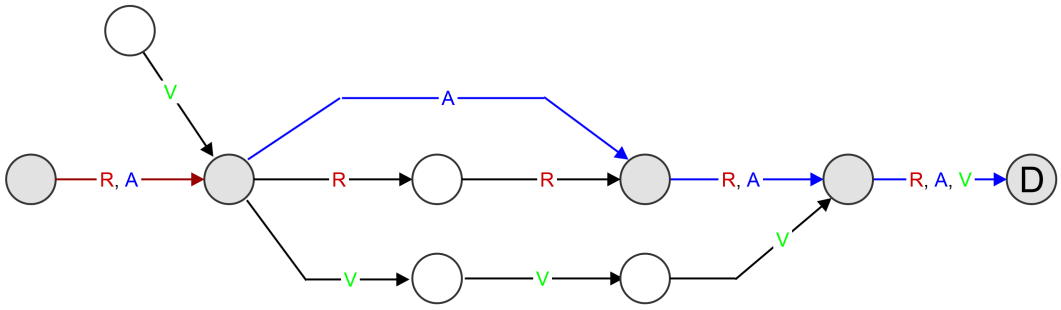


Figura 5.4: Ultima iteración del algoritmo para detectar un transbordo

El pseudo-código del nuevo algoritmo adaptado a la filosofía asíncrona de llamadas a funciones remotas de *Node.js* se muestra en el Procedimiento 2.

Procedimiento 2 Algoritmo para detectar un transbordo

Require: función $getRoute(parada)$
Ensure: $finalPath$, $tempCost$

1: $getNextBusAtStop(parada, callback)$ → Llamada al Web Service que devuelve mediante un *callback* los tiempos de los autobuses de la parada actual. La función esperará a que se retorne el *callback*, es decir, espera a que el servicio web devuelva una respuesta
2: $getNodosAdyacentes(parada)$ → Devuelve los nodos adyacentes a *parada*.
3: $getInterseccionLineas(parada)$ → Devuelve las líneas en común que pasan por *parada* y cada nodo adyacente.
4: $getFirstBus()$ → Devuelve la línea que pasa primero en cada nodo adyacente y el tiempo que tarda en minutos.
5: $addDelay(parada)$ → Sumo al arco correspondiente en la lista de adyacencia el tiempo que tarda cada autobús en llegar.
6: $Dijkstra(parada, destino, grafo)$ → Ejecuto el algoritmo de *Dijkstra* entre la *parada* actual y el *destino* prefijado sobre el nuevo grafo modificado. La función devuelve la mejor ruta provisional.
7: **repeat**
8: **if** $next == destino$ **then**
9: Llamada función $fin(next)$, $exitWhile = true$ → se termina el algoritmo.
10: **else if** $getNodosAdyacentes(next).length == 1$ **then**
11: Añadimos *next* al camino final, $exitWhile = false$
12: **else**
13: $exitWhile = true$, $getRoute(next)$
14: **end if**
15: **until** $exitWhile \neq true$

5.1.1. Estimación de tiempo de ruta

Para determinar la duración de un trayecto se ha considerado que los autobuses se desplazan a una velocidad media de 30 Km/h y que en cada parada consumen un tiempo de aproximadamente 1 minuto. Con ello, partiendo de la distancia total del recorrido y el número de paradas totales, se estima el tiempo.

Sin embargo, esta velocidad y el tiempo por parada no siempre corresponden a la realidad en cada tramo. Como se explica en [29], donde se comparan las velocidades entre diferentes sistemas de transporte público como metro, trenes y autobuses, las estimaciones relativas a los autobuses que circulan en la ciudad de Madrid evidencian que el autobús es el medio de transporte más lento, con una velocidad media aproximada de 13,5 Km/h. Sabiendo que, razonadamente, el tráfico presente en una ciudad de las dimensiones de Madrid no es comparable al de una ciudad más pequeña como Santander, se han realizado algunas estimaciones para adaptar la velocidad media con el fin de que el planificador desarrollado proporcione unos resultados más fiables.

Como se puede observar en la Tabla 5.1, se han considerado 3 trayectos elegidos al azar, con el objetivo de analizar la velocidad media de los autobuses, tomando como referencia de comparación la estimación proveniente del servicio web del TUS. Si se sabe cuál es el

autobús en el que se viajará para llegar a la parada de destino, aunque la petición se haga en un instante de tiempo anterior a escoger cualquier parada en concreto, se puede averiguar cuánto tardaría dicho autobús en llegar a la parada final mediante de una petición al servicio web y obteniendo la información de estimación de llegada de autobuses a esa parada. De esta forma, comparando este tiempo con el calculado mediante el algoritmo implementado, se puede estimar qué velocidad media sería la más adecuada para cada tramo analizado.

Tabla 5.1: Comparativa entre velocidades y tiempos de las estimaciones

Trayecto	**Velocidad**	**Estimación planificador**	**Estimación TUS**
1	10 Km/h	25 minutos	25 minutos
	15 Km/h	29 minutos	36 minutos
2	20 Km/h	55 minutos	36 minutos
	40 Km/h	42 minutos	32 minutos
	50 Km/h	35 minutos	27 minutos
3	15 Km/h	24 minutos	24 minutos
	20 Km/h	23 minutos	26 minutos

Los resultados de la Tabla 5.1 varían en función de la velocidad y del trayecto a analizar; por esta razón se va a explicar cada trayecto por separado y se van a analizar los resultados obtenidos. Evidentemente, la velocidad media insertada como parámetro en el algoritmo solamente afectará al tiempo de estimación calculado por el algoritmo, mientras que la estimación de tiempo obtenida del servicio web solamente dependerá del instante en el cual se haga la petición. En el trayecto 1, por ejemplo, se observa que, para una velocidad de 10 Km/h, la estimación del algoritmo resulta idéntica al tiempo estimado por el servicio web. Si se aumenta la velocidad del autobús para el mismo trayecto hasta 15 Km/h se observa cómo el autobús, según la estimación del algoritmo, tardaría menos del tiempo estimado por el servicio web. Este factor puede ser debido al alto tráfico del tramo que ha sido analizado, por ejemplo, por la toma de valores en una hora de punta, y, para este caso, cobra sentido atribuir a los autobuses una velocidad media baja.

Sin embargo, si se analiza el trayecto 2 sucede lo contrario. A pesar de que se aumente la velocidad media en cada prueba, no se consigue alcanzar el tiempo de estimación del TUS. Esto puede ser debido a un alto número de paradas, lo que podría causar que la estimación del tiempo de espera en cada parada de 1 minuto resulte demasiado alto. Adicionalmente, el trayecto analizado resulta ser una carretera recta donde los autobuses pueden tomar una mayor velocidad lo cual influye sobre el tiempo que tarda un autobús en recorrer un trayecto.

Finalmente, en el trayecto número 3 se obtiene una velocidad óptima de 15 Km/h para recorrer el tramo, puesto que coincide con la estimación del servicio web. El trayecto en cuestión resulta similar al trayecto analizado en el primer escenario.

Como se puede entender, en una red de transporte público aparecen varios escenarios donde la velocidad media de los autobuses varía en función de variables externas como pueden ser el tráfico presente en el trayecto, las condiciones meteorológicas o la hora y el día en que se toman los valores de referencia. En este sentido se desconoce el modelo de estimación que emplea TUS en los tiempos que retorna el servicio web. No obstante, el modelo de planificador definido en este proyecto puede incorporar diferentes métricas como

históricos de tráfico en zonas de la ciudad, incidencias en un tramos, etc., información que el Ayuntamiento de Santander en el ámbito de la *Smart City* está empezando a publicar. Todo ello hace que sea posible adaptar los resultados finales en tiempo de ejecución considerando costes variables por arco, lo que supondría convertir el planificador desarrollado en un sistema experto.

5.1.2. Tiempo de espera en la parada origen

Durante el desarrollo del proyecto, a la hora de implementar el algoritmo para detectar un transbordo, se ha presentado el dilema que se va a exponer a continuación. Como se ha comentado anteriormente, la condición que utiliza el sistema para que funcione consiste en tomar como punto de partida el primer autobús que llegue a la parada de origen. No obstante, el autobús que llegue antes puede que no siempre sea la mejor opción a escoger para llegar al destino final. A modo de ejemplo, es posible que esperar algo más en la parada nos permita evitar hacer un transbordo y ahorrar tiempo de recorrido.

Para lograr el objetivo y tener la posibilidad de escoger más de una opción, se ha implementado en el sistema un método para ejecutar más veces el procedimiento con el fin de que calcule el mejor trayecto según diferentes líneas como punto de partida. Es decir, en la primera iteración se escoge el primer autobús que llegue a la parada de origen; en la segunda iteración se toma como referencia el segundo autobús que llegue a la parada, o lo que es lo mismo, el primer autobús que llegue después de descartar la línea de la primera iteración. Este proceso se puede repetir hasta que tengamos el mejor resultado, por ejemplo, comparando entre todos los costes finales para recorrer el camino y elegir el que menos tiempo tarde. Otra opción interesante sería escoger, como criterio de iteración, el trayecto con menos trasbordos.

Para comprobar la exactitud de los resultados, se han hecho varias pruebas sobre diferentes trayectos. Suponiendo el mismo trayecto para las dos pruebas realizadas, se ha observado como en el primer resultado devuelto por el algoritmo fue necesario cambiar tres veces de líneas (líneas 1, 12 y 20) para poder llegar al destino, con un tiempo estimado de 32 minutos. Esta decisión, evidentemente condicionada por el hecho de escoger como línea de partida la primera en llegar a la parada origen, en este caso la línea 1, no resulta óptima para un servicio público como el TUS de Santander porque solamente permite hacer un transbordo como máximo. Es decir, para este caso en concreto se debería pagar dos veces el billete para poder llegar al destino e esta forma.

Sin embargo, en la segunda prueba del algoritmo, fue escogida la línea 6C2 como primer autobús en llegar a la parada. En este caso, solo se necesitaba hace un cambio de autobús a la línea 2 para poder llegar al destino, lo cual permitía abaratar el coste de la anterior opción estando en el limite de transbordos consentidos. Además, el tiempo de llegada al destino resultó sensiblemente más rápido, en efecto 30 minutos, probablemente por el hecho de no tener que esperar a hacer otro transbordo.

Como se puede observar, el sistema resulta altamente configurable por el administrador del sistema, pudiendo añadir variables o criterios variados para obtener resultados y conclusiones diferentes.

5.2. Trayectos a pie

Otra mejora que se ha añadido a la versión final del sistema es la posibilidad de poder desplazarse a pie entre paradas cercanas. De esta forma, el abanico de opciones de trasbordo que se abre es muy superior y será posible considerar transbordos entre líneas que no comparten paradas pero sí recorridos muy próximos.

Para poder conseguir el objetivo, se han realizado unas pruebas con el fin de poder establecer un protocolo por el cual, un potencial usuario del servicio de transporte público, tenga la posibilidad de poder llegar andando de una parada a otra para luego poder volver a incorporarse al sistema de transporte efectuando un transbordo a otro autobús. En este caso en concreto, se ha sometido el sistema a diferentes pruebas y, en particular, un trayecto característico de la ciudad donde la posibilidad de desplazarse andando otorgara una considerable ventaja al usuario final.

Para poder introducir la nueva opción al sistema, ha sido necesario crear una hipotética línea de transporte, a la que se ha denominado *Walking*. Esta línea, como se puede observar en la Figura 5.5, permite conectar dos paradas entre sí que, de otro modo, no estarían conectadas. Tomando la sección de la red de transporte de la Figura 5.5, si se considera que el viajante quisiera ir a la parada número 69 desde la parada 171, el hecho de conectar dos paradas mediante un trayecto a pie permite al usuario obtener una ruta viable y sencilla. En otras ocasiones puede suponer un ahorro de tiempo considerable al no tener que recorrer trayectos innecesarios para llegar a una parada conjunta para dos líneas, cuando un trayecto a pie resulta más rápido. La velocidad asignada para los recorridos a pie es de 4 Km/h.

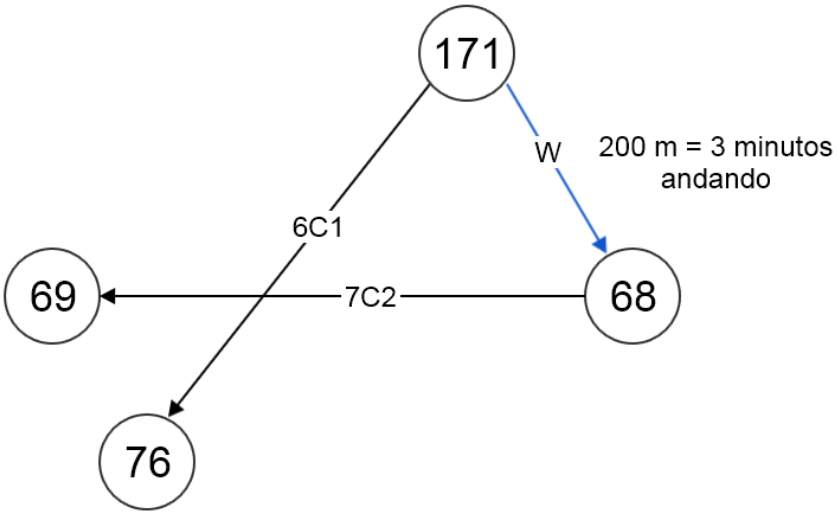


Figura 5.5: Representación de la introducción de la línea *Walking* (W) en el sistema

Además de incluir la nueva línea en la tabla *Red* de la base de datos, se debe modificar la información asociada a la lista de adyacencia del grafo de la red almacenada en la tabla *Red*, incluyendo la nueva línea *Walking* para la interconexión de dos paradas cuales quiera. En la Tabla 5.2 se hace referencia a la entrada añadida para que la solución sea factible y relacionada con la Figura 5.5.

Tabla 5.2: Entrada en la base de datos para la línea *Walking*

Origen	**Destino**	**Coste**	**Línea**
171	68	3	4, 7C2, W

Esto que en la vida real resulta relativamente sencillo si se conoce la ciudad y la estructura del transporte público y sus paradas, no es trivial a la hora de incluirlo en el planificador. En esta primera versión se han introducido diversas conexiones de manera manual, si bien dado que se conocen las coordenadas geográficas de las paradas se puede determinar fácilmente la distancia entre ellas y considerar un posible punto de trasbordo siempre que esta no exceda un valor configurable.

Capítulo 6

Conclusiones y líneas futuras

En el presente proyecto se ha abordado el diseño y desarrollo de una solución para la planificación de rutas en el ámbito de red de transporte de autobuses de la ciudad de Santander. El planificador realizado se basa en la aplicación de la teoría de grafos, con adaptaciones sobre algunos de los algoritmos, a una red con unas características muy variables. A continuación se recogen las principales conclusiones derivadas del trabajo y se plantean posibles líneas futuras que mejoren e incrementen las funcionalidades implementadas.

En este capítulo se van a resaltar los puntos a favor de la aplicación creada mediante este trabajo, planteando las posibles líneas futuras del proyecto para poder ampliar los conocimientos sobre el tema, mejorar las funciones ya implementadas y añadir nuevos aspectos para conseguir el objetivo deseado.

A continuación, se van a exponer los logros conseguidos durante el desarrollo de este proyecto.

6.1. Conclusiones

Tras realizar un estudio del estado del arte en la planificación de redes identificando soluciones disponibles y empleadas por la comunidad, se plantea el desarrollo del planificador objeto del proyecto aprovechando la teoría de grafos para modelar un sistema de transporte público. En esta línea, gracias el estudio realizado de la teoría de grafos aplicada al descubrimiento de rutas de coste mínimo se ha identificado la existencia de diferentes opciones y algoritmos que pueden aplicarse en este contexto para conseguir los objetivos propuestos y dar respuesta al conjunto de requerimientos planteados.

La opción seleccionada en este proyecto ha sido sustentar el planificador en el algoritmo de Dijkstra, uno de los algoritmos más conocidos y utilizados en la teoría de grafos para la búsqueda del trayecto más corto. El algoritmo de Dijkstra trabaja sobre un grafo estático en cuanto al coste que define sus arcos, mientras que en una red de transporte el coste de los arcos definidos para modelar los trayectos entre paradas depende de multitud de variables cuyo valor varía frecuentemente. Es por esto que si bien este algoritmo permite encontrar una ruta entre dos puntos, se han tenido que realizar diversas modificaciones, no al algoritmo

sino al procedimiento global para adaptar el resultado a una red de transporte, es decir, incluir la detección de trasbordos, estimación de tiempo, etc.

Tanto la información estática de configuración de la red como la variable sobre el estado de ésta se han obtenido de la plataforma del Servicio de Transportes Urbanos de Santander (TUS). El TUS ofrece un servicio web a partir del cual obtener la estructura de la red de transporte, es decir, líneas y sus paradas, e información dinámica sobre estimaciones de tiempo de llegada de los autobuses a una parada o la posición de éstos en el recorrido de la línea. Para evitar incurrir en solicitudes de información innecesarias y recurrentes, reducir la carga sobre el servicio web del TUS y acelerar el proceso de generar una respuesta ante una solicitud de ruta, el planificador desarrollado implementa un modelo mixto, por el cual la información de la estructura de la red se aloja en una base de datos local para acelerar los procesos de definición del grafo, mientras que la información variable y que puede suponer una modificación del coste de un arco específico se consulta en tiempo real.

Habiendo desarrollado y evaluado una solución básica del planificador (obtención de ruta más corta considerando únicamente la distancia entre paradas), para lo que se ha empleado *Node.js* y un modelo de gestión basado en un servicio REST, se ha ido incrementando el número de posibilidades del planificador. Para la detección del los transbordos o cambios de línea se ha diseñado un procedimiento iterativo de ejecución del algoritmo de Dijkstra sobre las paradas del recorrido. Así, tomando como parámetro adicional en cada iteración el tiempo estimado de llegada de los autobuses a una parada se ha podido definir conjunto de líneas más adecuado para obtener el trayecto óptimo. Adicionalmente, y tratando de considerar una de las situaciones más usuales en los desplazamientos por una ciudad, es decir, el realizar trayectos a pie, se ha demostrado la flexibilidad del planificador diseñado incluyendo y validando las mejoras resultantes de realizar transbordos a pie.

En este sentido, la inclusión de nuevas variables en el proceso de toma de decisiones se ha erigido en el artífice de lograr unas rutas más adecuadas para afrontar los desplazamientos entre paradas. Para facilitar la interacción del usuario con la plataforma se ha definido una plataforma web en la que el usuario podrá no sólo realizar las solicitudes indicando las paradas origen y destino, sino que también permite la visualización del recorrido retornado, si fuera necesario, los lugares donde realizar los transbordos y hacia qué línea.

Se puede considerar que, aunque los sistemas planificadores de rutas sean ampliamente utilizados a día de hoy, la principal característica de este proyecto se basa en la posibilidad de configurar el sistema según cualquier requerimiento o métrica externa que se quiera implementar en la plataforma. El nivel de personalización que ofrece la plataforma permite adaptarse al entorno en el que esté implementado el sistema, es decir, es posible introducir características específicas o únicas sobre recorridos, tiempos, autobuses o cualquier variable. El conjunto de estas características permiten modelar una planificación, única, precisa y adaptable a cualquier situación o imprevisto en tiempo real.

En este sentido, aunque el planificador podría exportarse a cualquier ciudad o entorno, para el caso particular de la ciudad de Santander y que ha sido el tratado en este proyecto, el sistema experto podría aprovechar la cantidad de datos recopilada por el proyecto *SmartSantander* (tráfico, meteorología, etc.) para poder ofrecer una planificación precisa y a medida del usuario final.

Sin embargo, en el ámbito de las telecomunicaciones ningún proyecto puede considerarse terminado definitivamente, ya que siempre pueden existir matices susceptibles de modificaciones, como se explicará en el apartado de líneas futuras que sigue a continuación.

6.2. Líneas futuras

La característica de configurabilidad abre un gran abanico de posibilidades de mejora o extensión del planificador desarrollado. Como ya se ha apuntado, la evolución natural es hacia un sistema experto donde se haga uso de toda la información disponible a través de la plataforma *SmartSantander* [30]. A partir de toda la información recopilada por los más de 20000 sensores distribuidos por la ciudad, a lo que se le sumaría la posibilidad de conocer de antemano las costumbres o necesidades de los usuarios, se podría ofrecer un servicio más completo y efectivo a los ciudadanos. Esta línea de trabajo se podría aplicar interactuando inicialmente con la plataforma pública de *Big Data* que el Ayuntamiento de Santander proporciona de manera abierta y libre. A partir de los datos adquiridos, se ponderarían los valores para aplicarlos el grafo de la red de transporte.

Además de ésto, se plantean otro conjunto de acciones destinadas a mejorar el procedimiento de planificación y proporcionar una mayor exactitud a las estimaciones de rutas realizadas. Algunas de ellas se enumeran a continuación.

- **Estimación tiempo de espera de un autobús en cada parada**: El estudio de este tiempo permitiría sustituir el retardo de 1 minuto impuesto por defecto y mejorar dicha estimación. Este valor, como se ha demostrado a lo largo del proyecto, puede dar lugar a una incorrecta estimación del tiempo empleado por un autobús en recorrer la ruta calculada por el algoritmo. Este fenómeno puede presentarse en trayectos largos donde intervienen un número de paradas relativamente alto. Una posible solución sería modelar la estimación del tiempo como una variable aleatoria correspondiente a la probabilidad de que haya otros viajeros en espera de un autobús en cada parada o en una determinada zona de la ciudad. Sin duda alguna, resultaría interesante poder relacionar este estudio con la densidad de población de cada zona específica. De esta forma, se podría estimar la posibilidad de que un autobús efectúe una parada y el tiempo de espera en dicha parada. Se ha de recordar que los resultados pueden verse influenciados por variables externas, como la temporada en la cual se realiza el estudio, las condiciones meteorológicas, la concentración de personas en la ciudad debido a algún evento especial, etc., lo cual determina que el proceso de estimación resulte mucho más complejo.

- **Estimación de la velocidad de los autobuses**: Una de las ventajas que ofrecen las aplicaciones de tiempo real es la posibilidad de conocer en todo momento la posición exacta de los autobuses circulantes por la estructura de la red. Este parámetro lleva a cabo un rol fundamental para poder estimar en qué zona de la red se encuentra el autobús y cuánto tiempo puede tardar dicho vehículo en llegar a una parada concreta. Gracias a estos parámetros y a otros a considerar, sería posible estimar la velocidad media de los autobuses que viajan por un determinado trayecto, lo que permitiría calcular con más exactitud los costes asignados a cada arco del grafo.

- **Aumentar la estructura de la línea *Walking***: Se han realizado una serie de pruebas para permitir el desplazamiento a pie, por parte de un pasajero, desde una parada a otra con la posibilidad de seguir estando en el servicio de transporte urbano y que el sistema calcule la mejor ruta. Una potencial mejora del sistema consistiría en ampliar este servicio e introducir, en la base de datos, una estructura de red dedicada a esta línea en especial. El objetivo se basa en ofrecer al usuario nuevas opciones para poder ahorrar tiempo, transbordos y, en determinados casos, abaratar el coste económico para el usuario. Conociendo la posición en coordenadas geográficas de cada parada, se debería detectar las paradas que estén a una distancia inferior a un valor dado, con la condición de que no pertenezcan al trayecto de una misma línea, para poder conectar las paradas en el grafo mediante la línea *Walking*. Para ello, sería preciso utilizar la información almacenada en la Tabla *Paradas* de la base de datos y, para ofrecer una mejor estimación del tiempo a emplear en realizar el trayecto, aplicar sistemas GIS.

- **Optimización del algoritmo de búsqueda:** Otro posible camino interesante para el futuro, sería estudiar la manera de implementar otros algoritmos, a diferencia del Dijkstra, para buscar la ruta de menor coste y comparar el rendimiento entre cada modelo empleado. El estudio realizado en el proyecto ha detallado el funcionamiento de algoritmos diferentes del implementado en origen. Una potencial solución sería introducir los algoritmos propuestos, además de otros que poseen cierta relevancia en la teoría de grafos. A modo de ejemplo sería posible implementar cualquiera de los siguientes: BFS con cola de prioridad ordenada por distancias, KSP, A*, etc. Evidentemente, para introducir cada opción propuesta puede que haya que modificar la estructura de la base de datos o la generación del grafo. Por esta razón, el tema necesita un estudio largo y profundizado.

- **Mejoras del interfaz de interacción**: En esta primera iteración del desarrollo del planificador los esfuerzos se han centrado en dotar de funcionalidad al sistema, por lo que se puede considerar que el margen de mejora de la interfaz visual de interacción con el usuario es bastante amplio. Además de proporcionar una interfaz algo más atractiva en términos de diseño, los resultados se podrían ofrecer representados en un mapa que incluyera la posición en tiempo real de los autobuses en el recorrido, pudiendo incluso llegar a definir un procedimiento de notificaciones que avisara al usuario de los instantes de llegada o de realización del transbordo.

Bibliografía

[1] RACC, Tribunas de la movilidad [sede Web]. Disponible en: http://tribunas.racc.es/es/transporte-publico

[2] FACUA,Consumidores en acción [sede Web]. Sevilla: FACUA Andalucía; 2007 [acceso el 10 de octubre de 2015]. El transporte público, guía del consumidor 2007. Disponible en: https://www.facua.org/es/guia.php?Id=77

[3] Bayon D. Best route planner apps for public transport. Alphr. [Revista en línea]. 2013 [acceso el 13 de octubre de 2015] Disponible en: http://www.alphr.com/features/381511/best-route-planner-apps-for-public-transport

[4] Google Transit [sede Web]. Disponible en: https://developers.google.com/transit/?hl=es

[5] GTFS [sede Web]. Disponible en: https://developers.google.com/transit/overview?hl=es

[6] Public Feeds [sede Web]. Disponible en: https://code.google.com/p/googletransitdatafeed/wiki/PublicFeeds

[7] Here Transit [sede Web]. Disponible en: https://company.here.com/en/here/

[8] Open Trip Planner (OTP)[sede Web]. Disponible en: http://docs.opentripplanner.org/en/latest

[9] OpenTripPlanner tool and transit mobility maps for Chicago [sede Web]. Disponible en: www.github.com/dssg/cta-otp

[10] TomTom [sede Web]. Disponible en: http://www.tomtom.com

[11] Cardozo O.D, Gómez E.L., Parras M.A. Teoría de Grafos y Sistemas de Información Geográfica aplicados al Transporte Público de Pasajeros en Resistencia. Revista Transporte y Territorio, Universidad de Buenos Aires. 2009; núm.1. pp. 89-111. Disponible en: http://www.redalyc.org/pdf/3330/333027079005.pdf

[12] Aldous J.M,Wilson R.J. Graph and applications, An Introductory Approach. 3ª ed. Londres: Springer; 2000.

[13] Pelegrín B, Cánovas L, Fernández P. Algoritmos en grafos y redes. 1ª ed.Barcelona: PPU; 1992.

[14] Yen, J. Y. Finding the K Shortest Loopless Paths in a Network. Management Science. 1971;17 (11): 712?716.

[15] Cormen T.H Leiserson C.E, Rivest R.L,Stein C. Introduction to Algorithms. 3ª ed. MIT Press and McGraw-Hill, 2009.

[16] Rabadán C.M, Algoritmos para el estudio de técnicas multi-path y network-coding en redes inalámbricas multi-salto [Trabajo de Fin de Grado]. Santander: Universidad de Cantabria; 2013.

[17] JavaScript [sede Web]. Disponible en: `https://www.javascript.com`

[18] Node.js [sede Web]. Disponible en: `https://nodejs.org`

[19] Vert.x, Asynchronous event-driven Web applications on the JVM [sede Web]. Disponible en: www.slideshare.net

[20] Librería Dijkstra Edsger [sede Web]. Disponible en: `https://www.npmjs.com/package/dijkstra-edsger`

[21] SoapUI [sede Web].Disponible en:`http://www.soapui.org`

[22] Hablemos de la arquitectura SoA [sede Web]. Disponible en: `www.islavisual.com`

[23] Librería Soap [sede Web]. Disponible en: `https://www.npmjs.com/package/soap`

[24] Librería Mysql. [sede Web]. Disponible en: `https://www.npmjs.com/package/mysql`

[25] Recurso web para Conversión ETR89, WGS84. Disponible en: `http://www.killetsoft.de/t_1009_e.htm`

[26] Servicio Web de Transformación de Coordenadas. [sede Web]. Disponible en: `http://www.ign.es/wcts-app/`

[27] Silberschatz A, Korth H.F, Sudarshan S. Fundamentos de bases de datos 5ª e. Madrid:McGraw-Hill;2006.

[28] Newcomer, E. Understanding Web Services: XML, WSDL, SOAP, and UDDI [Monografía en línea] Boston:Addison-Wesley; 2002.

[29] Transporte público y movilidad urbana sostenible [sede Web] Madrid: Ecomovilidad.net; 2012 [acceso 12 de octubre de 2015] De Córboda F. ¿A qué velocidad va el transporte público?. Disponible en: `https://ecomovilidad.net/madrid/a-que-velocidad-va-el-transporte-publico/`

[30] SmartSantander [sede Web]. Disponible en `http://www.smartsantander.eu/`