Trabajo de Fin de Grado

Sistema móvil de visualización y control en planificación de rutas

Mobile visualization and control system in routes planning

Ángel Rodríguez Negrín

La Laguna, 4 de junio de 2018

Don **Israel López Plata**, con N.I.F. 42.193.801-W profesor contratado de Universidad adscrito al Departamento Ingeniería Informática y de Sistemas de la Universidad de La Laguna y Don **Airam Expósito Marquez**, con N.I.F. 54.056.048-E personal investigador adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna,

C E R T I F I C A N

Que la presente memoria titulada:

”*Sistema móvil de visualización y control en planificación de rutas*”

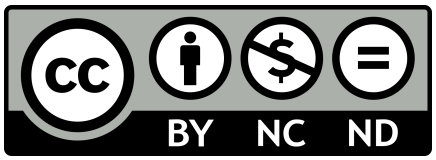
ha sido realizada bajo su dirección por Don **Ángel Rodríguez Negrín**, con N.I.F. 51.147.121-C.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 4 de junio de 2018

Agradecimientos

A todas esas personas que me han apoyado a lo largo de estos meses de trabajo.

Licencia



Resumen

En el presente proyecto, titulado ’Sistema móvil de visualización y control en planificación de rutas’, se realiza la construcción de una plataforma tanto móvil como web que posibilite a las empresas de transporte optimizar la planificación de las rutas a seguir y analizar el rendimiento de sus flotas de vehículos.

Este proyecto se basa en la tecnología de mapas Leaflet, así como en la librería Leaflet Routing Machine tanto para poder visualizar mapas dentro de la aplicación como para realizar cálculos de rutas en tiempo real. Estas herramientas, junto con tecnologías ampliamente utilizadas actualmente como Angular 5 o MongoDB conforman una plataforma que brinda a las empresas de transporte una solución para realizar el seguimiento y análisis de su proceso de reparto de mercancías.

La herramienta permite a los conductores de los vehículos de una empresa de transporte enviar información de los servicios realizados a clientes a través de la aplicación móvil. Mientras, en el gestor web se visualiza el recorrido de los vehículos, se modifican las rutas de reparto y se muestran datos estadísticos de la flota recogidos a través de la aplicación móvil.

Todo este proceso de comunicación entre los diferentes componentes de la plataforma se pone de manifiesto la necesidad de poder contar con funcionalidades y tecnologías que faciliten intercambios de información en tiempo real, siendo esto uno de los pilares fundamentales sobre los que se sostiene el proyecto.

Palabras clave: *Tiempo real, cadena de suministros, gestión de transporte, cálculo de rutas.*

Abstract

This project, entitled ‘Mobile visualization and control system in routes planning’, aims to build a mobile and web platform that allows transport companies to optimize the planning of the routes to be followed and analyze the performance of their vehicle fleets.

This project is based on Leaflet mapping technology and the Leaflet Routing Machine library, both for viewing maps within the application and for real-time route calculations. These tools, along with widely used technologies such as Angular 5 or MongoDB, offers a platform that provides transport companies with a solution for tracking and analyzing their freight delivery process.

The tool allows drivers of a transport company’s vehicles to send information on the services provided to customers via the mobile application. Meanwhile, the web manager displays the route of the vehicles, modifies the delivery routes and shows fleet statistics collected through the mobile application.

All this communication process between the different components of the platform shows the need to be able to count on functionalities and technologies that facilitate the exchange of information in real time, being this one of the fundamental pillars on which the project is based.

Keywords: *Real-time, Supply Chain, Transportation Management, Routing.*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

1.1. Contexto

La correcta administración y gestión del transporte es fundamental a la hora de implementar los procesos logísticos en la denominada Cadena de Suministros[1]. Las empresas del sector tratan de optimizar los planes de trasporte de mercancías, con el objetivo de poder entregar los bienes dentro del tiempo establecido y con unos costes adecuados. Estos elementos se traducen en una mejor percepción del servicio por parte de los clientes, que ante la creciente demanda de productos que han de ser servidos por las empresas de transporte exigen un trabajo de reparto con la mayor calidad y eficiencia posible. En esta línea, el transporte de mercancías requiere de un servicio de alta calidad en lo que respecta al tiempo, la eficiencia y la seguridad.

Para poder cumplir con este tipo de servicios de entrega de mercancías, las empresas disponen de flotas de vehículos que realizan el reparto a los diferentes clientes. Es imprescindible que tanto el desempeño como la productividad de los vehículos de las empresas proveedoras sean lo mejor posible para poder obtener un servicio con la calidad esperada por todos los actores de la cadena de suministro[2]: desde los proveedores, quienes conceden el uso de un determinado bien o servicio hasta el cliente o consumidor final[3], pasando por los fabricantes, los distribuidores y los detallistas o comerciantes al por menor. Poder agilizar todos los procesos y las comunicaciones dados a lo largo de esta cadena de suministros es la premisa para lograr una mejora de todo este sistema.

Los objetivos a cumplir por parte de dichas compañías deben estar

enfocados en incrementar la eficiencia de su flota y recursos móviles, así como en gestionar correctamente el rendimiento de los mismos, respetando en todo momento las normativas pertinentes del sector.

Las plataformas móviles para el seguimiento y la gestión de flotas de vehículos son una solución cada vez más extendida para solventar los problemas relacionados con los procesos logísticos asociados a la gestión y reparto de mercancías[4]. Muchos elementos que componen estas plataformas son clave en una correcta optimización de las rutas, desde los GPS que permiten realizar seguimientos a través de los dispositivos móviles hasta los algoritmos de planificación y optimización de rutas, pasando por las herramientas de análisis y gestión de rendimiento. Todos ellos son imprescindibles en la implementación de una plataforma competente que pretenda resolver problemas de este calibre.

Con el objetivo de construir una plataforma para el seguimiento y gestión de una flota de vehículos basada en los requerimientos previamente mencionados, a lo largo del presente proyecto se plantea el diseño, implementación y validación de un sistema de información que permita el seguimiento y la planificación de rutas, así como medir su rendimiento.

1.2. Motivación

Durante los últimos años, principalmente a causa de la proliferación de las páginas de compra por Internet, el número de clientes que deben ser atendidos por las empresas de transporte ha crecido de forma exponencial. Con el fin de poder atender a todos estos clientes garantizando la mayor efectividad posible, estas empresas se han visto obligadas a invertir un mayor número de recursos a la hora de gestionar y planificar los servicios de reparto.

Ante estas circunstancias, la elaboración de planes compuestos por rutas de transportes para la distribución a clientes se antoja un proceso de enorme importancia en el sector. Es fundamental poder optimizar los recursos de transporte disponibles, permitiendo a las empresas ahorrar tiempo y dinero a la par que se mejora el servicio prestado.

A la hora de elegir de qué forma poder llevar un control del proceso

de transporte cada vez más empresas proponen la utilización de herramientas software que gestionen toda esta logística. Actualmente existen diversas herramientas con la finalidad de gestionar las flotas de vehículos de empresas del sector del transporte. Éstas permiten controlar de manera exhaustiva elementos como la planificación de rutas, la posición de los vehículos o la desviación respecto a la planificación inicial establecida, siendo esto último lo más importante. Es necesario el uso de estas herramientas, puesto que permiten optimizar las rutas a seguir mediante algoritmos heurísticos, mejorando la planificación tanto inicial como final como su optimización, a la vez que se reduce el trabajo humano.

Por ejemplo **Verizon NetworkFleet**[1] es una herramienta de pago principalmente enfocada al seguimiento y análisis de flotas. Destaca por disponer de un análisis estadístico bastante amplio pero en contraposición no dispone de opciones para la edición de rutas en tiempo real.

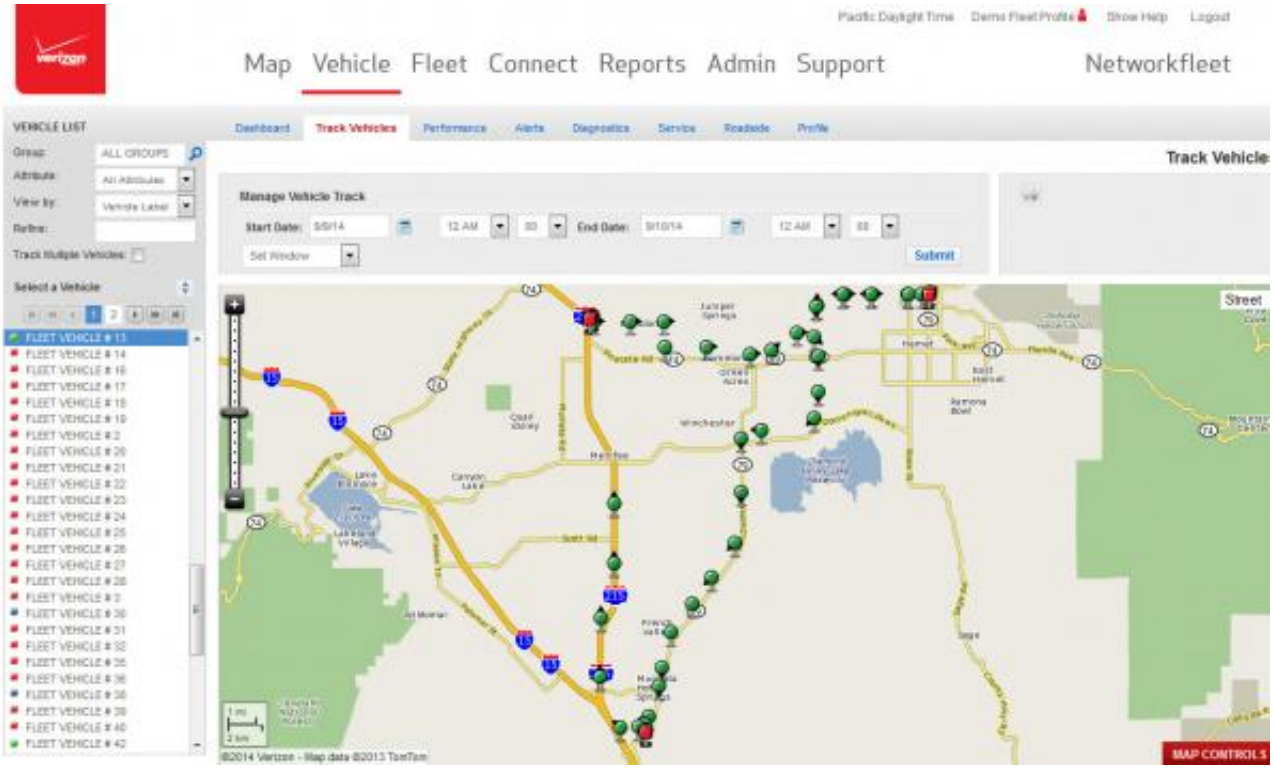


Figura 1.1: Interfaz Verizon NetworkFleet

Otra de las herramientas a considerar es **Routing Maps**[2], un software de pago dedicado a la optimización de rutas. Dispone de un producto web que calcula las rutas a realizar, muestra datos estadísticos y cuenta con un interesante simulador de decisiones de planificación, pero su interfaz

[1] http://www.verizonenterprise.com/view/12268/networkfleet-wireless-fleet-management
[2] https://www.routingmaps.com/

resulta bastante compleja y poco intuitiva. También cuenta con una aplicación móvil pero la poca información que ofrecen acerca de ella es que está pensada para realizar tracking de vehículo e informar de incidencias.

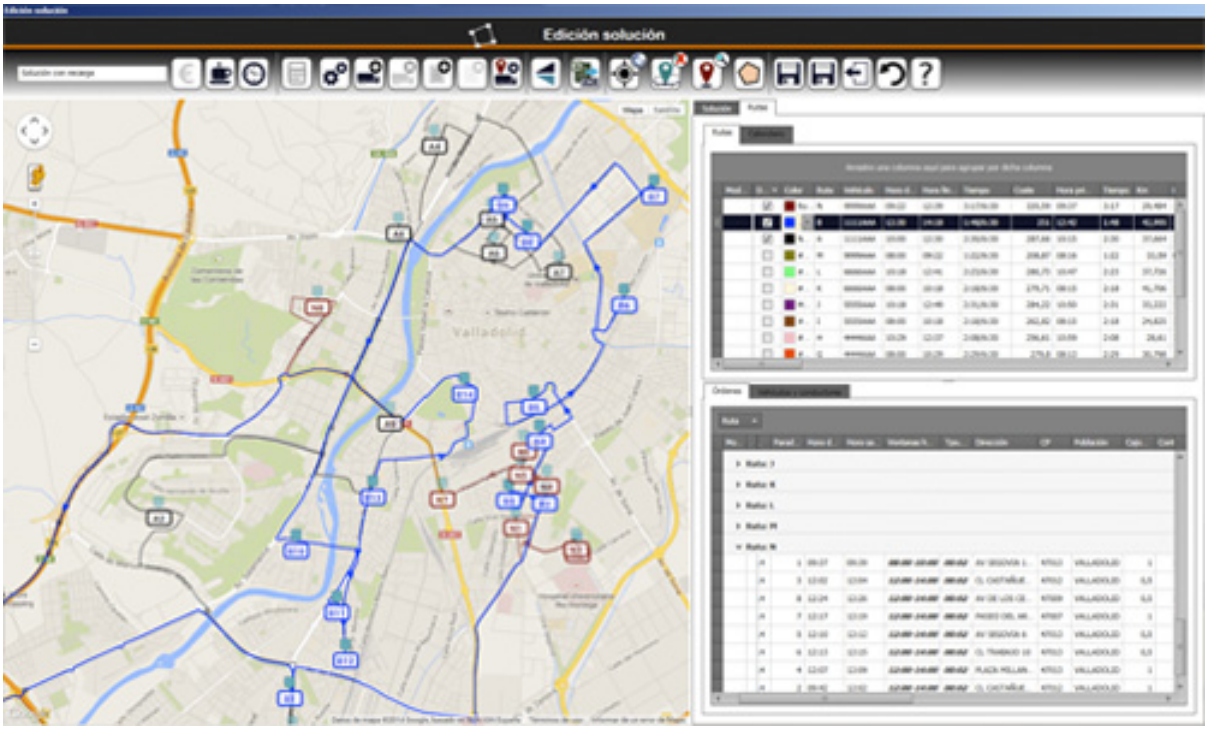


Figura 1.2: Interfaz Routing Maps Web

1.3. Objetivo del proyecto

El objetivo principal del presente proyecto es la creación y el desarrollo de una plataforma que permita la creación de rutas así como el análisis y seguimiento de la flota de vehículos que realiza dichas rutas. Dicha plataforma está basada en una arquitectura cliente-servidor compuesta en este caso por dos clientes: una aplicación móvil y una aplicación web, y un servidor estará conectado a una base de datos.

- El servidor:
  - Ofrece a ambos clientes los datos necesarios para representar geográficamente las rutas planificadas.
  - Se conecta con la base de datos para actualizar y consultar la información.
  - Dispone de mecanismos de comunicación con los clientes.

- El cliente móvil:
  - Muestra la ruta de servicio del camión en cuestión.
  - Permite actualizar el estado de los clientes: si están siendo atendidos, si han sido atendidos, etc.
  - Envía datos estadísticos al servidor: tiempo de servicio de cada cliente, distancia recorrida, etc.
  - Recoge la posición del camión para poder realizar su seguimiento.
- La aplicación web:
  - Permite editar las rutas de servicio de los camiones.
  - Permite visualizar el recorrido de los camiones en tiempo real.
  - Muestra un histórico de datos de cada camión.
  - Muestra datos estadísticos que permiten realizar análisis de redimiento de la flota.

Para poder desarrollar todos estos componentes de la plataforma es necesario realizar un estudio de las tecnologías necesarias, así como establecer de qué forma deben conectarse o cuál será el esquema de datos de la información que se debe almacenar. Por tanto, otro de los objetivos marcados para este proyecto y que es necesario para lograr el objetivo principal es la adquisición de conocimientos previos necesarios para la construcción del sistema.

1.4. Contenido del documento

Tras este capítulo 1 de introducción y resumen, estos son los contenidos a tratar en la presente memoria:

- **Capítulo 2:** Descripción de tecnologías y herramientas utilizadas durante el desarrollo del trabajo, justificando su uso y argumentando su importancia en el mismo.
- **Capítulo 3:** Descipción de la arquitectura del proyecto, los componentes que la conforman y de qué forma se relacionan entre ellos.

- **Capítulo 4:** Muestra de resultados del trabajo y explicación de su funcionamiento paso a paso.
- **Capítulo 5:** Explicación de las líneas futuras de trabajo y posibles mejoras o cambios planteados una vez finalizado el desarrollo.
- **Capítulo 6:** Desarrollo de conclusiones obtenidas tras la realización de este Trabajo de Fin de Grado.
- **Anexos y bibliografía:** Aportaciones adicionales que complementan los contenidos desarrollados en la memoria.

Capítulo 2

Tecnologías y herramientas

El presente capítulo tiene como objetivo ofrecer descripciones de las diferentes tecnologías y herramientas que se han escogido a la hora de desarrollar este trabajo.

2.1. Lenguajes de programación

- **JavaScript**[1]**:** JavaScript es un lenguaje de programación desarrollado por Netscape que actualmente se utiliza en la mayoría de páginas web y aplicaciones de servidor. Es un lenguaje dinámico que soporta construcción de objetos basados en prototipos, es decir, que no distingue entre clase e instancia y se limita a trabajar con objetos. Además tiene la característica de que puede funcionar como lenguaje procedimental y como lenguaje orientado a objetos. En el proyecto se utiliza JavaScript para desarrollar enteramente la parte del servidor. Su utilización se ve motivada por la sencillez de su uso y su baja curva de dificultad. Gracias a esto se pueden obtener unos resultados más que satisfactorios en poco tiempo.

[1] `https://www.javascript.com/`

Figura 2.1: Logo JavaScript

- **TypeScript**[2]**:** TypeScript es un lenguaje de programación de código abierto desarrollado por Microsoft. Es una extensión de JavaScript que tiene como añadidos principales el tipado estático y los objetos basados en clases. Se puede utilizar para desarrollar aplicaciones JavaScript, puesto que a la hora de compilar el código es traducido a dicho lenguaje. La parte lógica del cliente se desarrolla en este lenguaje. TypeScript está presente en este proyecto debido a que se utilizan los framework Ionic y Angular, como se comentará en este mismo apartado, y ambos están desarrollandos en este lenguaje.

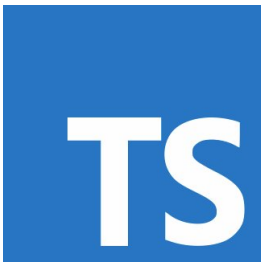


Figura 2.2: Logo TypeScript

- **HTML+CSS:** HTML o HyperText Markup Language es un lenguaje estándar utilizado para la definición de contenido de páginas web. HTML utiliza markup o marcado para identificar los elementos que se mostrarán en el navegador web y mediante etiquetas HTML se pueden definir bloques que darán forma a la estructura de la página. Por otra parte, CSS es un mecanismo basado en reglas que describe la forma en que se muestra el contenido del código escrito en HTML. Esta combinación de tecnologías se utiliza para desarrollar la parte visual del cliente web y el cliente móvil.

[2] `https://www.typescriptlang.org/`

Figura 2.3: Logo HTML+CSS

2.2. Tecnologías

2.2.1. Para backend y frontend

- **Socket.IO**[3]**:** Esta librería de JavaScript proporciona mecanismos de comunicación bi-direccional y en tiempo real mediante eventos. Está basada en Websockets y tiene la peculiaridad de que fue concebida para facilitar la comunicación en sistemas basados en la arquitectura cliente-servidor. Para enlazar la parte cliente y la parte servidor es necesario que ambas implementen sus respectivas funcionalidades de emisión y recepción de eventos, pero gracias a que el código es similar para ambas, este proceso se simplifica enormemente. Socket.IO hace posible la implementación de funcionalidades en tiempo real de este proyecto y su uso ha sido fundamental en el desarrollo del mismo.

Figura 2.4: Logo Socket.IO

[3] https://socket.io/

- **Node.js**[4]**:** Node.js es un entorno de ejecución de código abierto, multi-plataforma, construido con el motor de JavaScript 8 de Chrome. Es altamente recomendado para desarrollar aplicaciones web y dispone de un gran ecosistema de librerías, NPM, que nos aporta numerosos módulos reutilizables que facilitan el desarrollo del proyecto. Esta característica es de gran ayuda en la construcción de proyectos altamente escalables, puesto que además de contar un varios módulos básicos, permite instalar los módulos que se requieran a medida que avanza el proyecto. Además de esto, ofrece otras ventajas, como el hecho de estar diseñado para optimizar el rendimiento y la estabilidad de las aplicaciones o el tener una alta portabilidad.

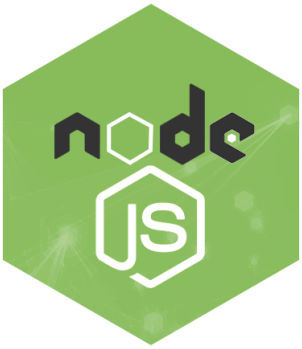


Figura 2.5: Logo Node.js

- **JSON**[5]**:** JSON o JavaScript Object Notation es un formato de texto ligero utilizado para realizar intercambios de datos. Este formato soporta diversos tipos de datos: números, cadenas, booleanos, null, arrays y objetos. Ante otras alternativas como XML, JSON se presta a un parseo de datos mucho más sencillo y es por ello que este formato es altamente recomendado en situaciones en las que se requiera de un gran flujo de datos. En este proyecto se maneja la información de un gran número de clientes, lo que hace que este formato sea el adecuado para la transmisión de datos.

Figura 2.6: Logo JSON

[4] https://nodejs.org/es/
[5] https://www.json.org/

2.2.2. Específicas de backend

- **Express**[6]**:** Express es un framework ampliamante utilizado de Node diseñado para facilitar el desarrollo de aplicaciones web. Proporciona diversos mecanismos de gran utilidad, como gestión de puertos o procesamiento de peticiones middleware. Existen diversas librerías que permiten trabajar con elementos de sesión, cookies, etc, pero en este caso en concreto, Express se ha utilizado para desarrollar la API REST que conecta ambos clientes con el servidor de la aplicación, gracias a los manjeadores de peticiones HTTP de los que se sirve. Además reduce en gran medida la complejidad de desarrollo de diversos elementos del backend, ya que se consiguen grandes resultados con unas pocas líneas de código.

- **MongoDB**[7]**:** MongoDB es un sistema de base de datos no relacional de código abierto y que pertenece a la familia de NoSQL. Esto quiere decir que en lugar de almacenar datos en tablas, Mongo almacena los datos en documentos BSON. Dichos BSON son muy similares a los JSON pero con la particularidad de que disponen de un esquema dinámico, lo que favorece la rápidez en la integración de datos. Debido a que en este proyecto se ha optado por utilizar JSON como formato de intercambio de datos, es lógico utilizar una base de datos Mongo para su almacenamiento, además de por el hecho de ser una de las bases de datos más utilizadas en la actualidad, debido a su elevada potencia y escalabilidad.

Figura 2.7: Logo MongoDB

- **Mongoose**[8]**:** Esta librería disponible a través de node ofrece todas las herramientas necesarias para poder trabajar con una base de datos mongo desde el servidor. Gracias a su sistema de modelado de objetos, es posible definir un esquema a través del cual persistir o consultar objetos alojados en la base de datos especificada.

[6] http://expressjs.com/
[7] https://docs.mongodb.com/
[8] http://mongoosejs.com/

- **Docker**[9]**:** Es una plataforma que permite desplegar diversas aplicaciones dentro de contenedores de software. Cada uno de estos contenedores ejecuta procesos de manera aislada, de forma que podemos trabajar con cada uno de ellos como máquinas virtuales independientes. En este caso, se utiliza un contenedor virtual de Docker para alojar la base de datos Mongo. Debido a la naturaleza de los contenedores es posible trabajar con dicha base de datos de forma muy rápida y consumiendo muy pocos recursos de la máquina. A pesar de que en este proyecto es utilizada con un único fin, Docker es una herramienta con un gran potencial que posibilita la creación y despliegue de sistemas altamente distribuídos.

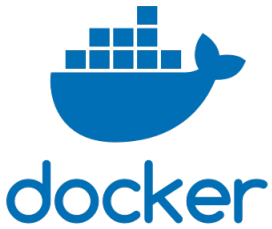


Figura 2.8: Logo Docker

2.2.3. Específicas de frontend

- **Angular 5**[10]**:** Angular 2+ es un framework open source utilizado para desarrollo web. Está desarrolado en TypeScript y es mantenido por Google. El objetivo principal de Angular 2+ es el de proporcionar una estructura de proyecto que se adecúe al MVC, Modelo-Vista-Controlador, con el fin de facilitar el desarrollo de proyectos y sus correspondientes pruebas. Permite desarrollar tanto aplicaciones de escritorio como web o nativas y se caracteriza por tener un buen rendimiento al estar enfocado en obtener la mayor productividad posible. El cliente web de este proyecto ha sido desarrollado en su versión 5, por ser la iteración de Angular más actualizada en el momento de desarrollo del proyecto.

[9] https://www.docker.com/
[10] https://angular.io/

Figura 2.9: Logo Angular

- **Ionic**[11]**:** Este kit de desarrolo software basado en Angular permite construir aplicaciones móviles tanto nativas para Windows, Android e iOS, como híbridas. En el caso de las aplicaciones híbridas, éstas utilizan tecnologías web como HTML y se despliegan utilizando Cordova, por lo que es posible emular el funcionamiento de la aplicación en un navegador web. Se puede descargar vía node (npm install) e incluye numerosos elementos para el desarrollo móvil como componentes, tipografías o temas. Desarrollar en Ionic ofrece muchísimas ventajas frente a otras alternativas como Android Studio, como por ejemplo el ser mucho más ligero al poder mostrar la aplicación en el propio navegador y no necesitar un emulador de móvil, o el hecho de que permite desarrollar en nativo para todas las plataformas móviles.

Figura 2.10: Logo Ionic

- **Leaflet[5]**[12]**:** Es una librería de JavaScript cuya función es construir mapas para aplicaciones, ya sean web o móvil. A pesar de que GoogleMaps es quizás la opción estándar para trabajar con mapas, Leaflet ofrece varias ventajas que la hacen una opción a considerar para desarrollar aplicaciones de este tipo. Una de ellas es que resulta muy cómoda para trabajar, ya que su curva de aprendizaje es bastante baja y, al contrario que GoogleMaps, no requiere enlazar el proyecto a ninguna cuenta, por lo que tras una simple instalación ya es posible

[11] https://ionicframework.com/
[12] https://leafletjs.com/

utilizarla. Pero quizás lo más llamativo de esta alternativa es que es Open Source, lo que la hace ideal para aplicarla en proyectos como éste, no solo por el hecho de ser gratuíta, sino también por la cantidad de apoyo que recibe por parte de la comunidad en forma de plugins. En la figura 2.12 se muestra el resultado final de un mapa creado con esta tecnología.

Figura 2.11: Logo Leaflet

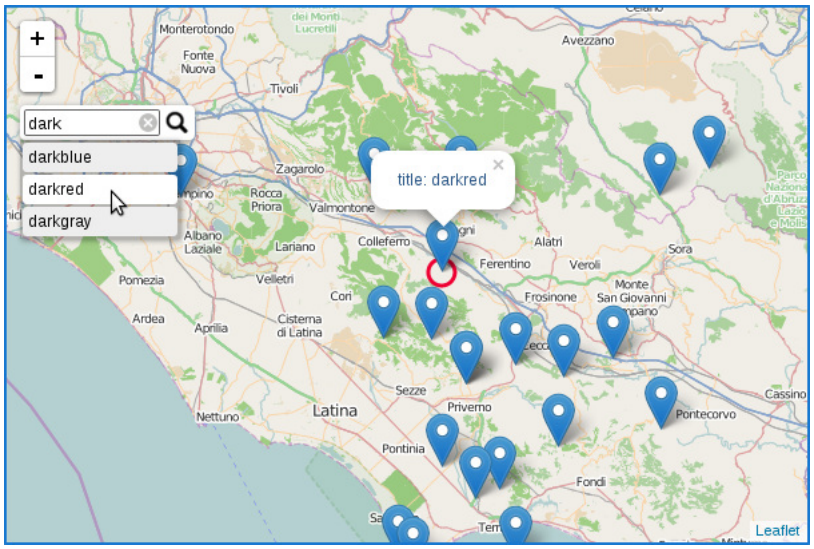


Figura 2.12: Mapa en Leaflet

- **Leaflet Routing Machine[6][13]:** Esta librería, desarrollada por el usuario de github Per Liedman y descargable vía npm install, ofrece varias herramientas que añaden funcionalidades de enorme importancia a un proyecto con Leaflet. Además de diversas opciones de configuración sobre Leaflet, Leaflet Routing Machine facilita diversos objetos, funciones y mecanismos para poder trabajar con rutas dentro de una aplicación. Gracias a ella podemos gestionar y mostrar rutas, calcular distancias o trabajar con waypoints. Es una herramienta que agrega un enorme potencial a Leaflet, habilitando muchísimas más posibilidades de las que ya dispone y es por ello que se convierte en la alternativa elegida para el desarrollo del presente trabajo.

[13] http://www.liedman.net/leaflet-routing-machine/

- **PrimeNG**[14]**:** Es una librería de código abierto desarrollada por PrimeTek Informatcs que dispone de una gran cantidad de elementos reutilizables para el desarrollo de páginas web y aplicaciones móviles. Ofrece un gran número de componentes, desde tablas hasta diferentes tipos de menús, así como plantillas y diversos estilos y temas. Su utilización se ve motivada no solo por la facilidad de su uso, sino porque los elementos disponibles cuentan con numerosas funcionalidades muy potentes que ya están implementadas y que suponen un gran ahorro de tiempo, como por ejemplo poder exportar el contenido de una tabla a un documento.

Figura 2.13: Logo PrimeNG

2.3. Editor de código

El editor utilizado para desarrollar el proyecto en su totalidad ha sido **Visual Studio Code**[15]. Desarrollado por Microsoft, este editor de código fuente está basado en Electron, un framework que se utiliza principalmente para la implementación de aplicaciones con Node.js. Entre las características que ofrece y por las que ha sido la herramienta escogida para el desarrollo del código del proyecto nos encontramos con soporte para la depuración, alta personalización o control integrado de git, entre otros. Además, es gratuíto y de código abierto.

Figura 2.14: Logo Visual Studio Code

[14] https://www.primefaces.org/primeng/
[15] https://code.visualstudio.com/

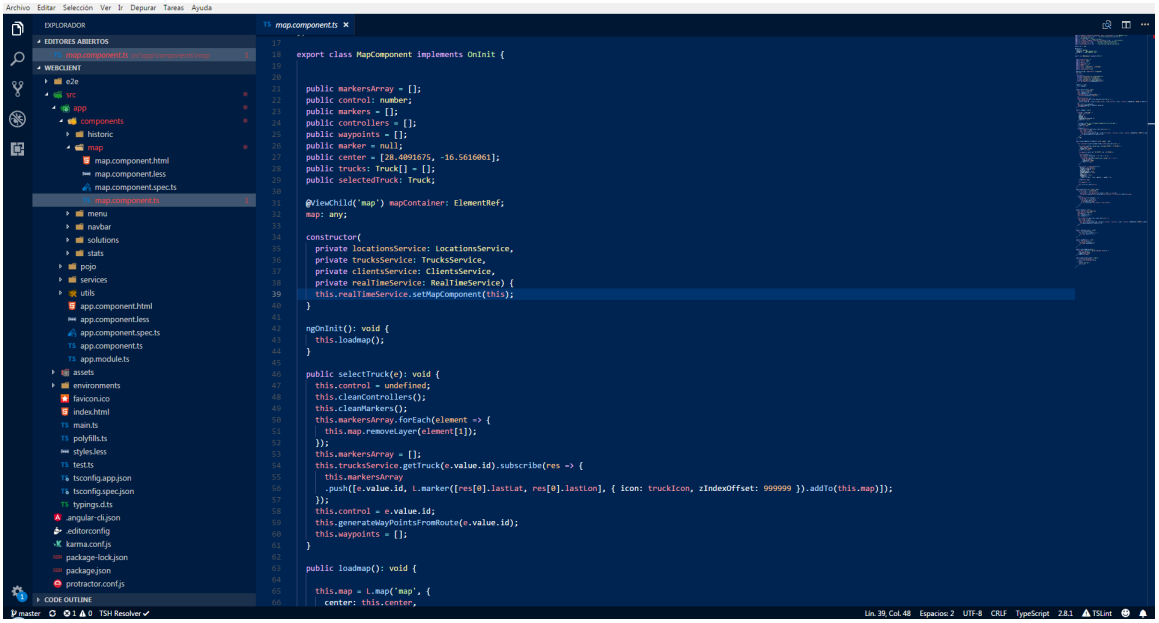


Figura 2.15: Proyecto en Visual Studio Code.

Capítulo 3

Arquitectura del sistema

A la hora de construir todo el ecosistema de componentes que conforman el proyecto se ha optado por utilizar la clásica **arquitectura cliente-servidor**. La filosofía que se sigue tras este modelo de diseño software es el reparto de tareas entre los servidores, proveedores de recursos o servicios y clientes, que son aquellas partes que demandan la información servida. Para poder realizar este intercambio de información, ambas partes están conectadas mediante una red determinada y a través de los mecanismos de comunicación definidos[7].

La arquitectura propuesta para la construcción del sistema, siguiendo los principios de la estructura cliente-servidor queda distribuída tal como se muestra en la figura 3.1.

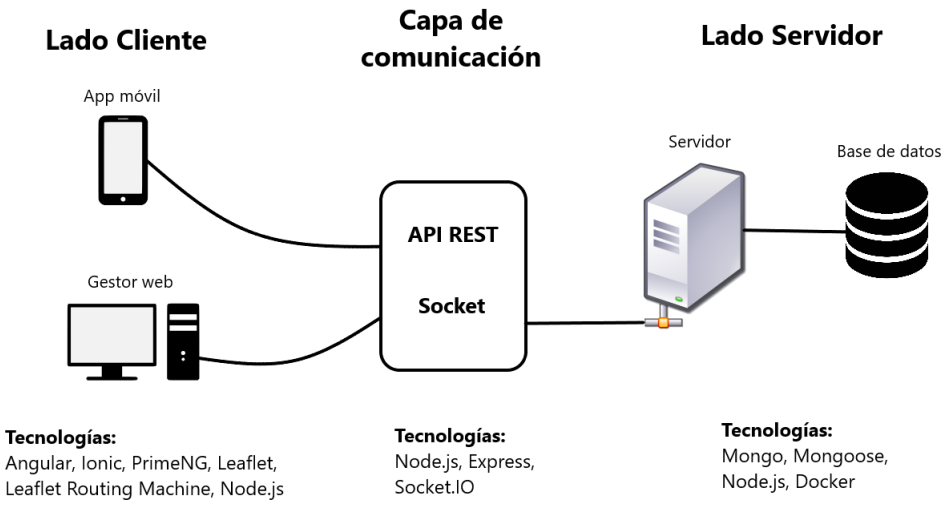


Figura 3.1: Arquitectura de la plataforma.

En este caso el lado del servidor está compuesto por el propio servidor conectado a una base de datos Mongo encargada de alojar todos los datos necesarios. Dentro del código del servidor se describen dos mecanismos de comunicación que servirán como una capa intermedia entre lado cliente y lado servidor: una API REST con varios endpoints definidos que son accesibles desde los clientes y un socket para poder implementar funcionalidades que requieran de interacciones en tiempo real. Por último, el lado cliente está compuesto por dos herramientas: un cliente móvil desarrollado en Ionic y un gestor web desarrollado en Angular. Ambos consumirán la información servida y disponen de toda la lógica necesaria para desempeñar las funcionalidades planteadas en el proyecto.

3.1. Servidor

El servidor se encarga de proporcionar la información necesaria a los clientes web y móvil. Esta parte está desarrollada en Javascript y cuenta con dos mecanismos de comunicación: una API REST y un socket, ambos descritos en su apartado correspondiente. Además, la información a servir y almacenar estará alojada en una base de datos Mongo, la cual será accedida gracias a los endpoints definidos en la API REST y la librería Mongoose.

3.1.1. Base de datos:

La base de datos de la cual el servidor recoge la información se encarga de almacenar los datos de los clientes a los que se debe visitar, los camiones que componen la flota de la empresa y el listado de servicios realizados por los vehículos.

Dado que MongoDB es una base de datos no relacional, los objetos se guardan directamente como documentos BSON y estos objetos se almacenan en colecciones, que son el equivalente a las tablas del modelo relacional de datos.

De acuerdo con lo anteriormente mencionado, las tres colecciones planteadas para este proyecto son clientes, camiones y servicios. A continuación se describe cada una de ellas:

- **Clientes:** Esta colección almacena la información de todoslos clientes que deben de ser atendidos por la empresa de transporte. A efectos de simplificación del proceso de asignación de clientes a los diferentes camiones, se ha planteado que cada uno de estos clientes solo sea atendido por un único camión y pueden existir clientes que no tengan ningún camión asociado. Cada uno de ellos dispone de un atributo identificador para poder ser diferenciado y la latitud y longitud de su ubicación para generar un marcador en los mapas de la aplicación. También se ofrecen datos para aportar más información al camionero que deba realizar el servicio, como la dirección o el horario de apertura. Por último, se han definido varios atributos para poder controlar el estado actual del cliente entre tres posibles: siendo atendido, atendido o sin atender, que son fundamentales para el correcto funcionamiento de la plataforma.

- **Camiones:** La información recogida en esta colección representa a cada uno de los camiones que forman parte de la flota de vehículos de la empresa. Incluye tanto un atributo ID para identificar a cada uno de ellos, como atributos relacionados con las estadísticas: distancia recorrida, número de clientes servidor, tiempo de inicio y final de su recorrido, entre otros. Además, en esta colección se almacenan los datos de localización del camión, tanto la posición donde comienzan la ruta para poder generar un marcador especial, como su última ubicación registrada para poder llevar un control de la misma desde las aplicaciones cliente.

- **Servicios:** La última de las tres colecciones se encarga de almacenar los servicios realizados por los vehículos de la empresa. Como ya se ha comentado anteriormente, cada cliente solo puede ser servido por un único camión, así que cada una de las t-uplas de esta colección es única, en cuanto a que no puede existir una repetición del par de atributos ’truckId’ y ’clientId’. Además de contar con estos dos atributos, cada documento de esta colección indica el momento de inicio del servicio, el momento de finalización del servicio y el tiempo de duración. Todos estos datos se utilizan para las funciones de análisis de rendimiento y estadísticas implementadas en el gestor web.

3.2. Capa de comunicación

En el lado servidor del proyecto se establecen dos mecanismos que definen de qué forma los clientes de la aplicación se conectan con el servidor. Uno de ellos es una API REST. Desarrollada con la librería Express, esta interfaz se encarga de definir diversos endpoints a través de los cuales los clientes acceden a la información alojada en la base de datos. El otro mecanismo es un socket, construído gracias a la librería Socket.IO. El socket se encarga de proveer un canal de comunicación bidireccional a través del cual se controlan las conexiones que realizan los clientes. Además, mediante el sistema de gestión de eventos que ofrece, permite realizar intercambios de información en tiempo real.

3.2.1. API REST:

Para poder acceder a los recursos ofrecidos por el servidor es necesario definir una capa intermedia que permita controlar de qué forma pueden ser consumidos dichos recursos desde los clientes. Por este motivo se ha optado por utilizar una API REST, desarrollada con la librería Express, como uno de los elementos de comunicación entre clientes y servidor.

Describiéndola brevemente, REST es una interfaz que normalmente se define sobre HTTP y que se utiliza para poder generar datos u operar sobre ellos[8]. Las diferentes operaciones que se pueden definir a la hora de trabajar con una interfaz REST son: POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar).

La API desarrollada para este proyecto dispone de varios endpoints, es decir, puntos de conexión donde quedan expuestos los recursos a consumir:

- **/clients:** Endpoint encargado de proveer la información almacenada en la colección ’clients’ de la base de datos. Permite filtrar resultados en función de los queryparams que reciba y tiene definidos los métodos GET para obtener información de clientes y PUT para actualizar información de los mismos.
- **/trucks:** Endpoint encargado de proveer la información almacenada en la colección ’trucks’ de la base de datos. Permite filtrar resulta-

dos en función de los queryparams que reciba y tiene definidos los métodos GET para obtener información de los vehículos y PUT para actualizar información de los mismos.

- **/services:** Endpoint encargado de proveer la información almacenada en la colección 'services' de la base de datos. Permite filtrar resultados en función de los queryparams que reciba y tiene definidos los métodos GET para obtener información de los servicios y POST para añadir los servicios realizados a la base de datos.

- **/locations:** Además de las colecciones de la base de datos, se ha planteado almacenar un registro de las últimas ubicaciones de los camiones. Esta información es volátil, es decir, que no está pensada para ser persistida, y por tanto no se almacena en base de datos. En su lugar se almacenan en el propio servidor. Un objeto de tipo locations dispone de tres parámetros: latitud, longitud y el id del camión al que corresponde dicha ubicación. Tiene definidos los métodos GET para obtener información de las últimas localizaciones de un determinado vehículo y POST para añadir ubicaciones conforme se desplaza el camión en la ruta.

En el diagrama de la figura 3.2 se refleja un sencillo caso de uso de una llamada a la API REST desde uno de los clientes. En este caso desde el cliente móvil se desea llamar al endpoint '/services' para consultar los servicios realizados por el camión número 3:

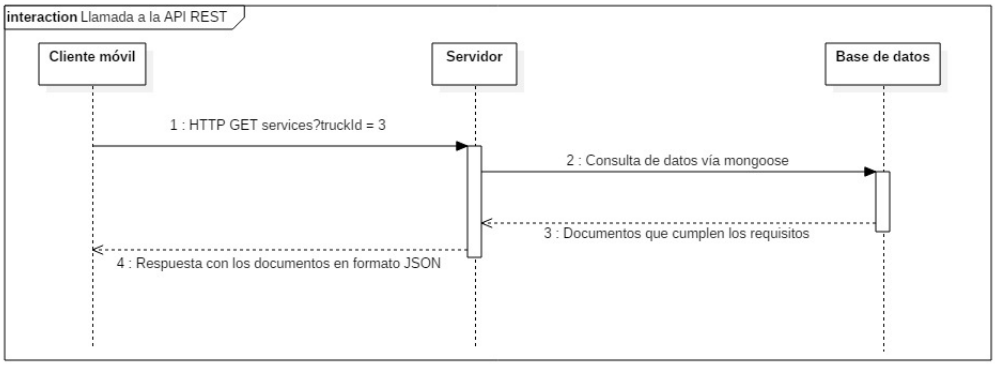


Figura 3.2: Caso de uso de una llamada a la API REST.

3.2.2. Socket:

El segundo mecanismo de comunicación definido es un socket creado con la librería Socket.IO. Un socket es un mecanismo que permite establecer una comunicación bidireccional entre dos componentes, pudiendo intercambiar información entre ellos[9]. La utilización de esta tecnología dentro del proyecto es totalmente necesaria, ya que permite llevar un control de las conexiones de los clientes y gracias al manejo de eventos es posible implementar varias de las funcionalidades planteadas que operan en tiempo real.

Para poder establecer una comunicación vía socket entre los clientes y el servidor se han de implementar mecanismos de emisión y recepción de eventos en ambas partes, en función de las necesidades de la aplicación.

En el lado servidor el funcionamiento del socket es bastante sencillo: realizar un envío simultáneo de eventos y datos a todos los clientes que estén conectados. Es decir, cuando un cliente emite un determinado evento en el mismo canal que el servidor, éste se encargará de realizar un broadcast al resto de clientes conectados. Por otra parte, los clientes serán los encargados de envíar y recibir los eventos una vez se conecten al canal de comunicación existente entre ellos y el servidor.

El diagrama de secuencia de la Figura 3.3 muestra un caso de uso de la comunicación vía socket dentro del proyecto, donde ambos clientes se conectan al servidor. El cliente móvil realiza un envío de datos, en este caso datos de su posición actual, el servidor realiza un broadcast de esos datos recibidos y por último, el gestor web procesa la información que llega desde el servidor para realizar las operaciones necesarias para replicar el movimiento del camión.

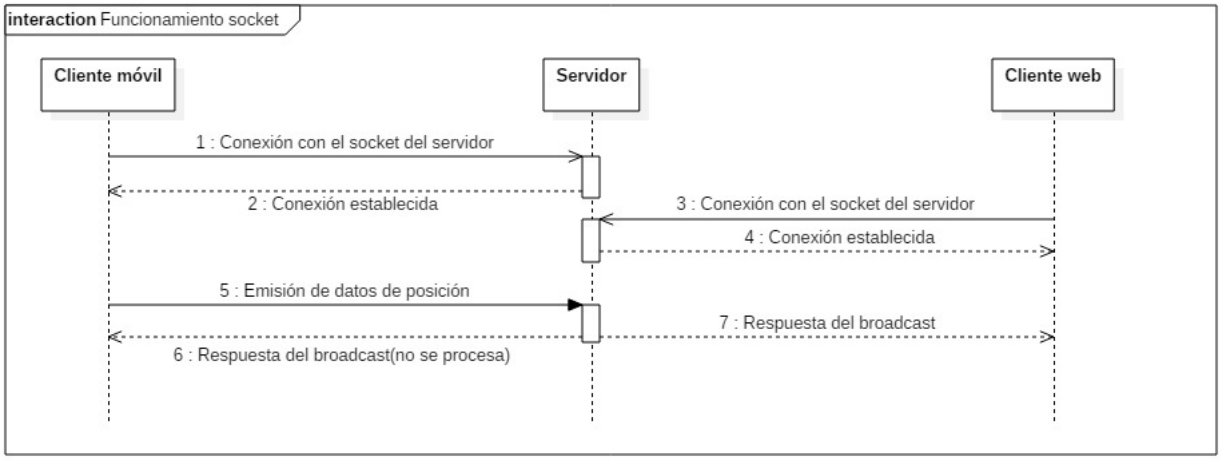


Figura 3.3: Ejemplo de funcionamiento del socket.

3.3. Clientes

La información que el servidor expone a través de los mecanismos de comunicación definidos y explicados en el apartado anterior será consumida por la parte cliente. En este caso se han desarrollado dos clientes diferentes: uno de ellos es un cliente móvil desarrollado con ionic, el cual será utilizado por el conductor del vehículo de reparto mientras que el otro es una aplicación web desarrollada en Angular que permitirá realizar la gestión de los diferentes clientes y camiones.

3.3.1. Cliente móvil

Desarrollado en TypeScript y mediante el framework Ionic, el cliente móvil es el responsable de envíar y consumir la información necesaria para localizar al camionero en tiempo real, notificar los servicios realizados a los clientes asignados y registrar estadísticas relacionadas con el vehículo y el recorrido a realizar: tiempo de duración del trayecto, distancia recorrida, etc.

Una de las funcionalidades más importantes del cliente es el poder visualizar mapas y rutas para que el camionero sepa a qué lugares debe dirigirse. En este caso se utilizan las tecnologías Leaflet y Leaflet Routing Machine, que disponen de todas las funcionalidades necesarias para la creación de mapas y visualización de rutas. Por un lado, Leaflet se encarga

de la creación del mapa y varios aspectos asociados al mismo como los marcadores o el zoom. Mientras que Leaflet Routing Machine calcula y dibuja en el mapa las rutas a seguir. A partir de array ordenado de coordenadas o ’waypoints’ esta librería permite obtener la ruta entre ellos, y una gran cantidad de información acerca de la misma, como por ejemplo el tiempo y la distancia estimados para el recorrido o las direcciones que hay que tomar para llegar al destino. Muchos de estos datos son recogidos dentro de la propia aplicación para poder ser enviados al gestor web.

Otro de los aspectos clave del cliente y que resulta imprescindible para el funcionamiento de la plataforma es la conexión vía socket con el servidor. La implementación de la librería Socket.IO client dentro de la aplicación móvil permite establecer una conexión con el socket del servidor y realizar intercambios de información en tiempo real. Gracias a esto, el cliente móvil es capaz de envíar al servidor su posición actual cada ciertos segundos para posteriormente replicar el movimiento en el gestor web.

El resto de funcionalidades hacen uso de la información liberada a través de los endpoints de la API REST, ya que son principalmente consultas y actualizaciones de datos. Por ejemplo, desde la pantalla de login se lanzan consultas para comprobar si el ID del camión introducido existe en la base de datos, o para mostrar los servicios realizados por un determinado camión, etc. En Ionic, con la finalidad de organizar el código de una forma adecuada, las funciones relacionadas con las llamadas a endpoints se realizan dentro de unos componentes llamados proveedores o ’providers’. Para cada endpoint a atacar se define un provider, y además existe uno en específico que se encarga de la implementación del socket.

3.3.2. Cliente web

El cliente web o gestor, desarrollado en TypeScript mediante el framework Angular 5, es el encargado de gestionar toda la información acerca de los vehículos que componen la flota de la empresa. No solo permite modificar las rutas de los camiones, añadiendo, quitando y reasignando clientes en tiempo real, sino que además permite visualizar su recorrido y consultar estadísticas.

Este cliente comparte algunas similitudes con el cliente móvil. Por ejemplo, una de las pantallas de las que dispone la web muestra el recorrido en tiempo real de los camiones mientras realizan su trayecto.

Leaflet y Leaflet Routing Machine se utilizan también en este caso para poder visualizar el mapa y las rutas de los diferentes camiones. Esta funcionalidad también requiere de una implementación de Socket.IO client, aunque difiere de la de la aplicación móvil, puesto que en este caso se añaden funcionalidades para recibir datos del broadcast realizado en la implementación del socket del servidor. Es de gran importancia controlar la recepción de eventos que maneja el socket, puesto que en el caso de que existan varias aplicaciones móviles conectadas al mismo tiempo y solo se desee recibir datos de una de ellas, es necesario filtrar la información recibida a través del canal de comunicación.

Gran parte de la construcción de este gestor web está realizada con la librería PrimeNG. Esta tecnología ofrece componentes ya construídos que facilitan el diseño de la web y permiten ahorrar tiempo en su desarrollo. Por ejemplo, la pantallas que muestran el histórico de los vehículos y el editor de rutas utilizan tablas de PrimeNG. Estas tablas disponen de funcionalidades muy interesantes como por ejemplo exportación de información a CSV o filtros de búsqueda. Además, también se han utilizado otros elementos de la librería, como las gráficas para implementar el módulo de estadísticas, que disponen de funcionalidades de gran utilidad, como redimensión automática o filtro de datos.

Por último, cabe destacar que toda la información mostrada en las pantallas mencionadas se extrae de la API REST. En este caso se aplica la misma regla que para el cliente móvil: las funcionalidades que ataquen a los diversos endpoints deben de implementarse en componentes específicos de Angular llamados servicios o ’services’ para poder estructurar el código respetando la filosofía de Modelo Vista Controlador.

Capítulo 4

Descipción funcional

Tras detallar las tecnologías utilizadas y la arquitectura planteada para este proyecto, en este capítulo se muestran los resultados obtenidos y se realiza una descipción de las funcionalidades de los mismos.

4.1. Aplicación móvil

4.1.1. Pantalla de login

Comenzando por la descripción de funcionalidades del cliente móvil, lo primero que se muestra en esta aplicación es una sencilla pantalla de login. Tal y como se aprecia en la Figura 4.1, se solicitan cinco parámetros a introducir: El usuario del camión que va a utilizar la aplicación, una contraseña, la IP del servidor donde están alojados los mecanismos de comunicación y sus puertos.

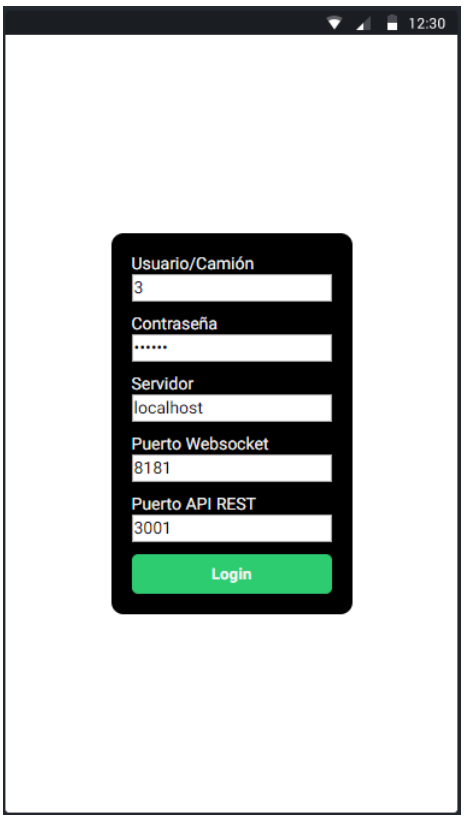


Figura 4.1: Pantalla de login.

Una vez introducidos los datos, se debe pulsar el botón de Login para acceder a la aplicación. Este acceso verifica que existe comunicación con el servidor a través de los respectivos puertos y realiza una comprobación de la existencia del usuario proporcionado.

4.1.2. Pantalla y menú de conexión

Si las operaciones realizadas en la pantalla de login tras pulsar el botón correspondiente son exitosas se abre la pantalla de conexión, la cual se muestra en la Figura 4.2. En esta pantalla se llevan a cabo una serie de operaciones previas al inicio del recorrido del camión.

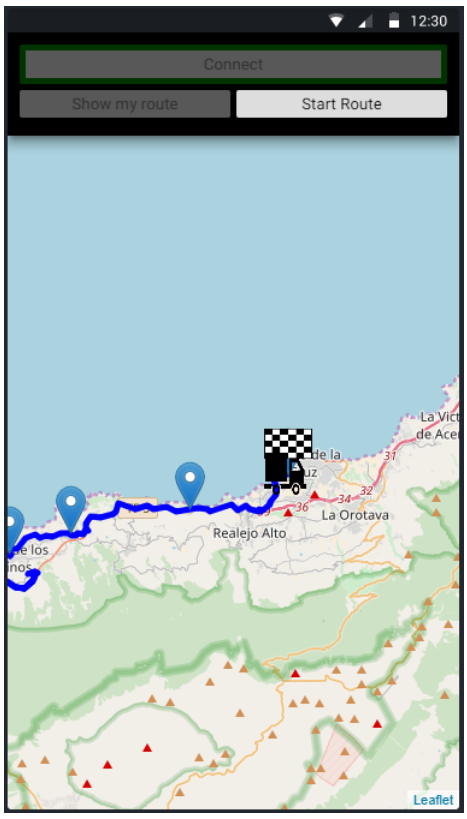


Figura 4.2: Menú previo al recorrido.

En la parte superior de este menú se aprecia que está compuesto por tres botones que implementan las siguientes funcionalidades:

- **Connect:** Se encarga de cargar los datos de los clientes asociados al camión.
- **Show my route:** Utiliza los datos cargados para llamar a las funciones de la librería Leaflet Routing Machine y mostrar la ruta en el mapa.
- **Start route:** Da paso al inicio del recorrido.

Se ha de tener en cuenta que estos tres botones están diseñados para ejecutar esta secuencia de pasos en el orden en el que se han numerado. Una vez realizada dicha secuencia, el menú de la aplicación cambia y se muestra la siguiente pantalla.

4.1.3. Pantalla y menú de ruta

Este es el aspecto de la aplicación en pleno funcionamiento, tal y como se muestra en la Figura 4.3. El diseño ha sido planteado de esta forma para poder concentrar tanto funcionalidades como la información a mostrar de cara al camionero que la use.

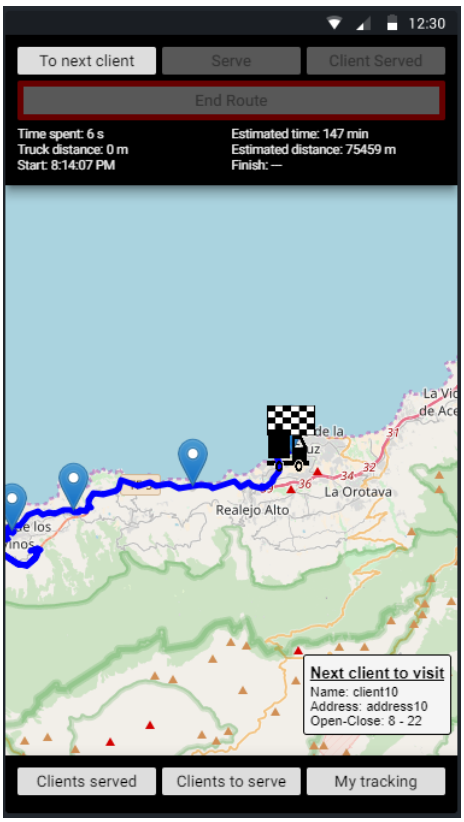


Figura 4.3: Comienzo del recorrido.

En el menú superior se observan cuatro botones y una serie de parámetros relativos al recorrido del camión. Comenzando con la descripción de dichos parámetros, se observan seis en total:

- **Time spent:** Un indicador del tiempo que lleva el camión realizando la ruta. La cuenta comienza desde que se accede a esta pantalla. Se mide en segundos debido al poco tiempo que duran las simulaciones realizadas.
- **Estimated time:** Tiempo en minutos estimado para la realización del recorrido. Este parámetro es calculado por la librería Leaflet Routing Machine y si la ruta cambia se actualiza automáticamente.

- **Truck distance:** Distancia en metros que ha recorrido el camión. Se actualiza en tiempo real según se desplaza el vehículo.
- **Estimated distance:** Distancia estimada del recorrido en metros. Este parámetro es calculado por la librería Leaflet Routing Machine y si la ruta cambia se actualiza automáticamente.
- **Start:** Indica la hora de inicio del recorrido.
- **Finish:** Indica la hora de finalización del recorrido. Se muestra tras pulsar el botón ’End route’.

En cuanto a los botones, existen cuatro diferentes y están diseñados para seguir una secuencia lógica de funcionamiento según el orden que se describe a continuación:

- **To next client:** Este botón es el que inicia la secuencia de funcionamiento. Cuando el camionero lo pulse se considera que comienza su desplazamiento hacia el siguiente cliente. Para la entrega de este proyecto el recorrido se realiza de forma automática tras pulsar el botón y debido a esto es necesario realizar una comprobación de la ruta, ya que en caso de que se hayan añadido o quitado clientes cambian los puntos que se visualizan en el mapa.
- **Serve:** Una vez el camionero llegue al cliente debe pulsar este botón. Da inicio a un contador de tiempo para registrar la duración del servicio. Además, también registra el momento en el que da comienzo dicho servicio y actualiza el siguiente cliente a atender para indicar que va a ser servido.
- **Client served:** Una vez el cliente haya sido atendido, el camionero pulsa este botón para parar la cuenta de tiempo de servicio, actualizar los datos del cliente y mostrar un marcador en el mapa que indica que el servicio ha sido realizado. Tras esto se comprueba si quedan mas clientes por atender. Si se cumple esta condición, se activa el botón ’To next client’, actualiza la ruta para comprobar si se han añadido o quitado clientes y la secuencia comienza de nuevo. En caso contrario se activa el botón ’End route’.
- **End route:** Este botón se muestra en el caso de no existir mas clientes por atender en la ruta del camión. Tras pulsarlo finaliza el recorrido.

Las figuras 4.4 y 4.5 muestran ejemplos del avance del recorrido del camión.

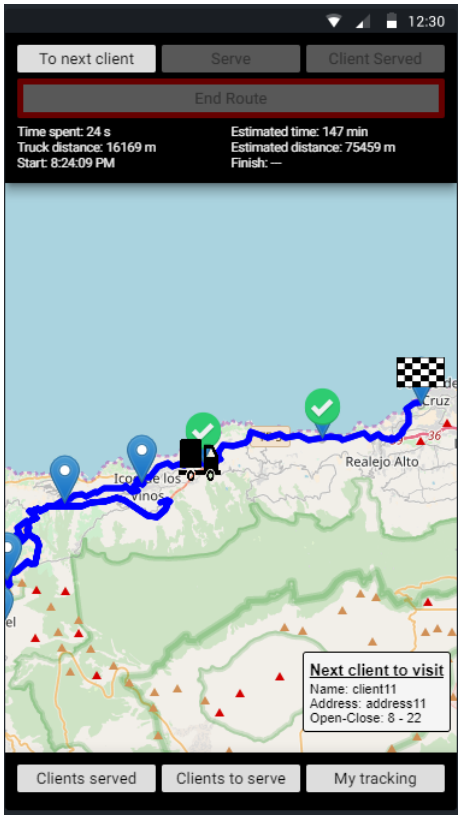


Figura 4.4: Camión realizando recorrido.

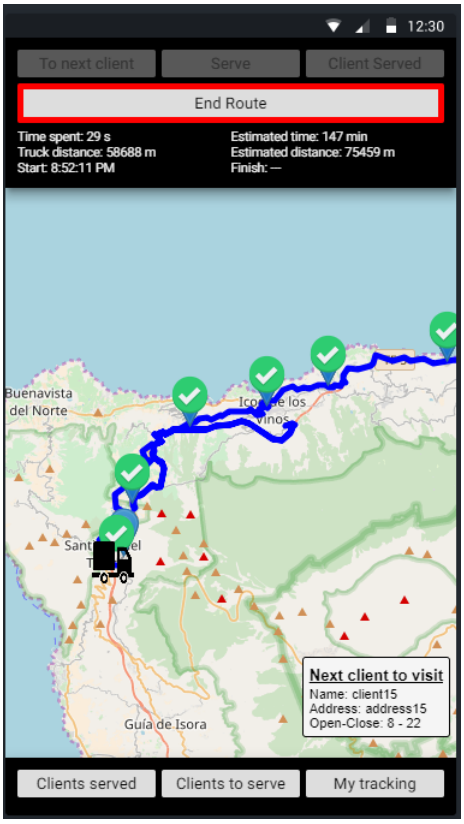


Figura 4.5: Final del recorrido.

En la parte inferior derecha del mapa se encuentra un componente que muestra **la información del siguiente cliente a atender**. Se visualiza el nombre del cliente, su dirección y las horas de apertura y cierre. Esta información se actualiza en tiempo real según se vayan atendiendo clientes. Concretamente el botón **Client served** es el encargado de lanzar una petición para consultar cuál es el siguiente cliente y así actualizar este cuadro.

Por último, en el menú situado en la parte inferior de la pantalla existen tres botones cuyas funcionalidades son las siguientes:

- **Clients served:** Permite visualizar una tabla que contiene el listado de servicios realizados por el camión. La información se obtiene mediante una petición al endpoint ’/services’ accesible a través de la API REST. Para cada uno de los servicios se muestra el ID del cliente atendido, la hora de inicio del servicio, la hora de finalización del servicio y el tiempo total empleado en segundos. La Figura 4.6 muestra los clientes servidos correspondientes al recorrido visto en la Figura 4.4.

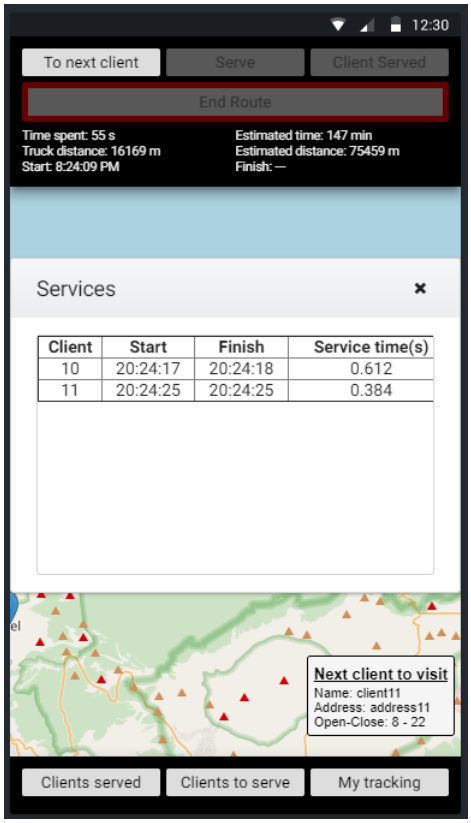


Figura 4.6: Listado de clientes servidos.

- **Clients to serve:** Muestra una tabla que contiene el listado de clientes que deben de ser atendidos por el camión. La información en este caso se obtiene mediante una petición aplicando un filtro con dos parámetros: el ID del camión que está usando y el estado de los clientes a devolver, es decir, que no hayan sido atendidos aun. En la Figura 4.7 aparecen los clientes servidos al finalizar la ruta del vehículo.

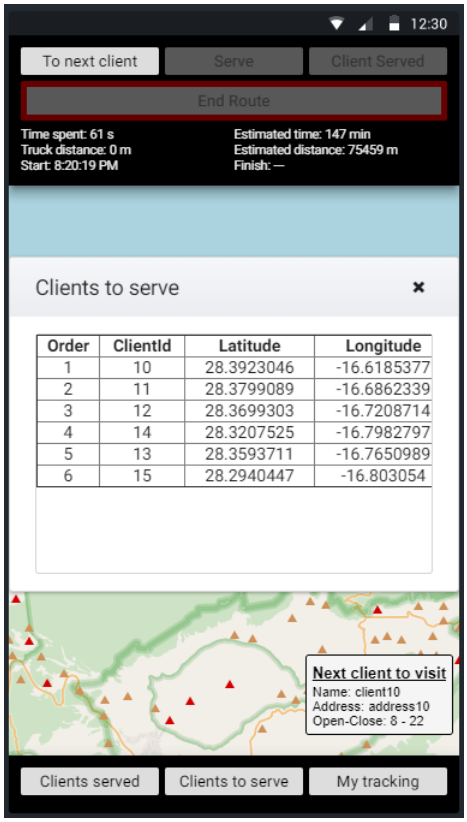


Figura 4.7: Listado de clientes a atender.

- **My tracking:** Carga el listado de localizaciones registradas del camión. Para cada localización registrada se visualiza su latitud y longitud, tal y como se muestra en la Figura 4.8

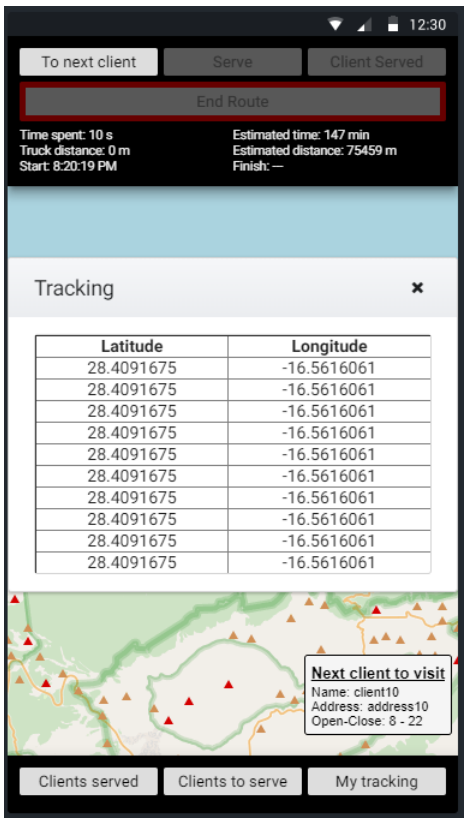


Figura 4.8: Tracking del vehículo.

4.2. Gestor web

La herramienta web se encarga de gestionar la información acerca de las rutas y los datos estadísticos de los vehículos. Dispone de un sencillo menú en la parte izquierda compuesto por cuatro botones. Cada uno de estos botones carga una pantalla diferente, las cuales son descritas a continuación:

4.2.1. Pantalla de edición de rutas

La pantalla de edición de rutas permite asignar, borrar y redistribuir clientes entre los diferentes camiones que componen la flota de vehículos de la empresa, así como crear rutas desde cero. En este caso se considera que un un conjunto ordenado de clientes que tienen que ser atendidos por un camión en concreto conforman una ruta. En la figura 4.9 se muestra una captura de dicha pantalla:

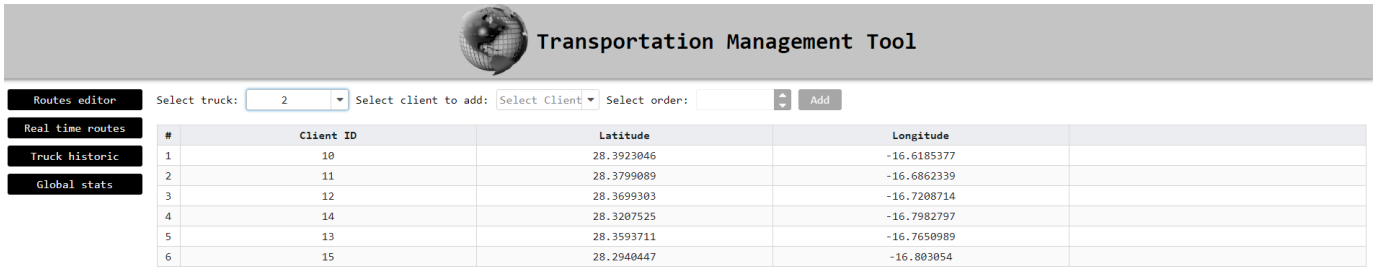


Figura 4.9: Pantalla de edición de rutas.

Para poder visualizar, añadir y quitar clientes de la ruta se ha implementado un mecanismo basado en el uso de los elementos de la parte superior de la pantalla.

El primero de ellos es el combo box **Select truck** situado en la parte superior izquierda, que permite seleccionar un identificador de camión entre una lista extraída de una consulta realizada a la base de datos. Dependiendo del valor del combo box, se muestra un listado de clientes u otro en la tabla de la parte inferior. Esta tabla, creada gracias a la librería PrimeNG muestra para cada cliente su orden de prioridad, su ID y la latitud y la longitud de su ubicación, así como un botón para poder eliminar al cliente de la ruta, en caso de ser posible.

Para poder añadir clientes se utiliza el segundo combo box, **Select client to add**, que muestra los clientes que aun no están asignados a ningún camión y que por tanto pueden ser añadidos a la ruta del camión seleccionado en el primer combo box.

Un vez seleccionado el cliente a añadir, el selector numérico **Select order** permite elegir en qué posición de la ruta se desea colocar. Aquí se implementa un mecanismo de control del orden en el que el cliente se añade a la ruta, ya que solo puede ser incluido después de los clientes que han sido atendidos. En la Figura 4.10 se ejemplifica este proceso, puesto que se observa que en los combo box se ha seleccionado un cliente y un orden.

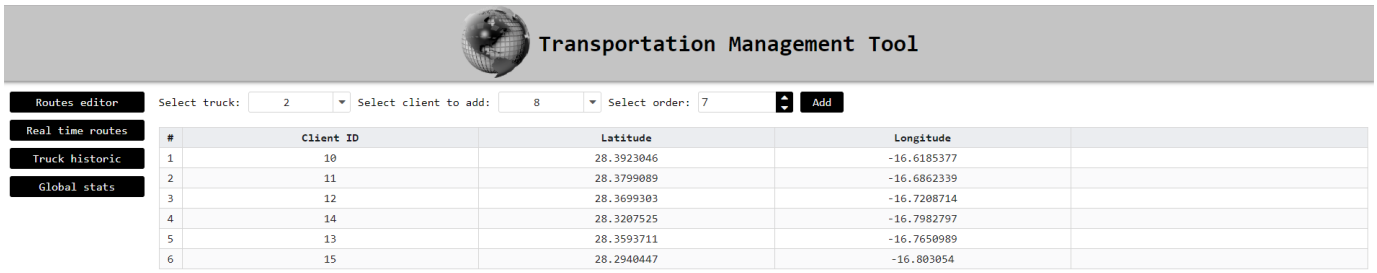


Figura 4.10: Añadiendo un cliente (1/2).

En este caso el único orden en el que se puede añadir es en la posición 7 porque los otros 6 clientes asociados al camión ya han sido atendidos. Una vez seleccionado el camión, el cliente y el orden, es posible pulsar el botón **Add** para realizar la operación. Tras eso, en la Figura 4.11 se muestra como el cliente se ha añadido en la tabla.

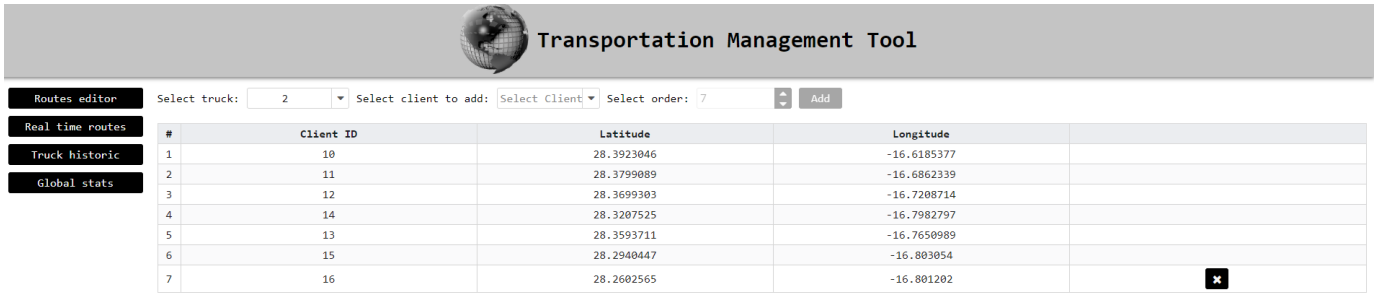


Figura 4.11: Añadiendo un cliente (2/2).

Como el cliente recién incluido aun no ha sido atendido, es posible borrarlo de la ruta gracias al botón de eliminar que aparece en su fila correspondiente de la tabla. Este botón desaparecerá en cuanto el camión comienza el servicio a ese cliente, impidiendo que pueda ser eliminado de la ruta.

4.2.2. Pantalla de seguimiento de vehículos

La pantalla de visualización de rutas en tiempo real comparte muchos elementos funcionales con la aplicación móvil al estar compuesta por un mapa creado con Leaflet. Desde esta sección se pueden visualizar las rutas compuestas por los diferentes clientes a atender, así como el movimiento de los camiones en tiempo real mediante la utilización de las funciones que implementan la librería Socket.IO.

En la parte superior izquierda de la pantalla existen dos elementos a través de los cuales se controla la visualización de las rutas. El primero de ellos es un combo box que muestra los ID de los camiones que componen la flota, extraídos mediante una consulta a la base de datos. La ruta a visualizar depende del ID seleccionado en este elemento. En las Figuras 4.12 y 4.13 se aprecian dos rutas distintas generadas gracias a este mecanismo.

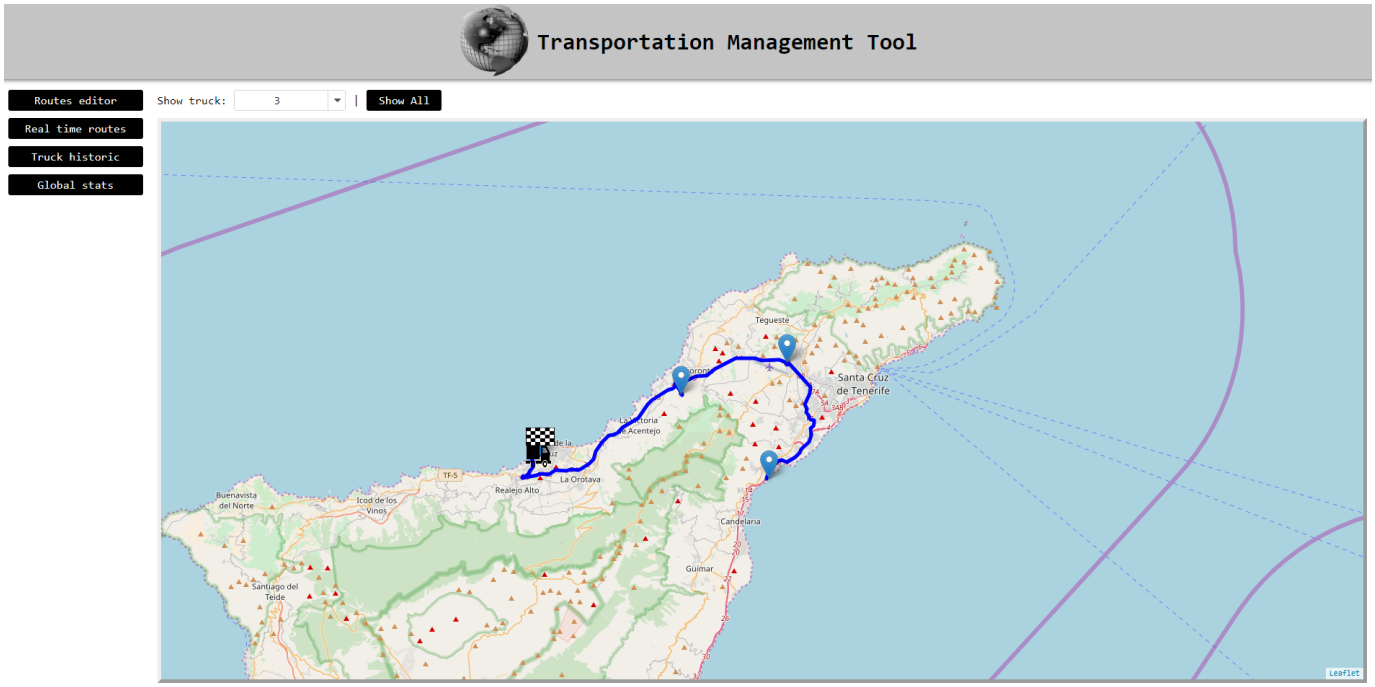


Figura 4.12: Ruta del camión número 3.

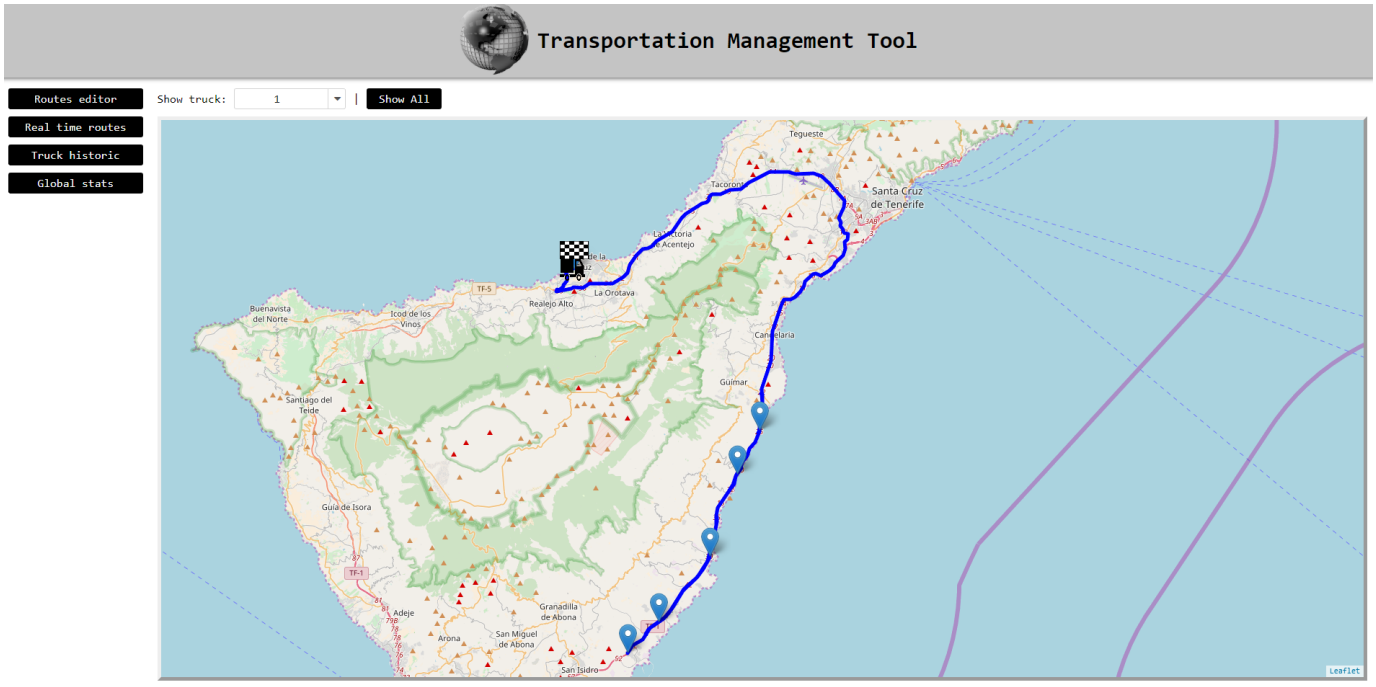


Figura 4.13: Ruta del camión número 1.

El segundo elemento es un botón que muestra todas las rutas, tal y como se aprecia en la Figura 4.14. Cabe destacar que en el mapa se visualizan los clientes que han sido servidos gracias a un marcador de color verde. Este marcador se genera en tiempo real gracias a las funcionalidades del socket, por lo que si un camión finaliza un servicio se refleja al instante en esta pantalla.

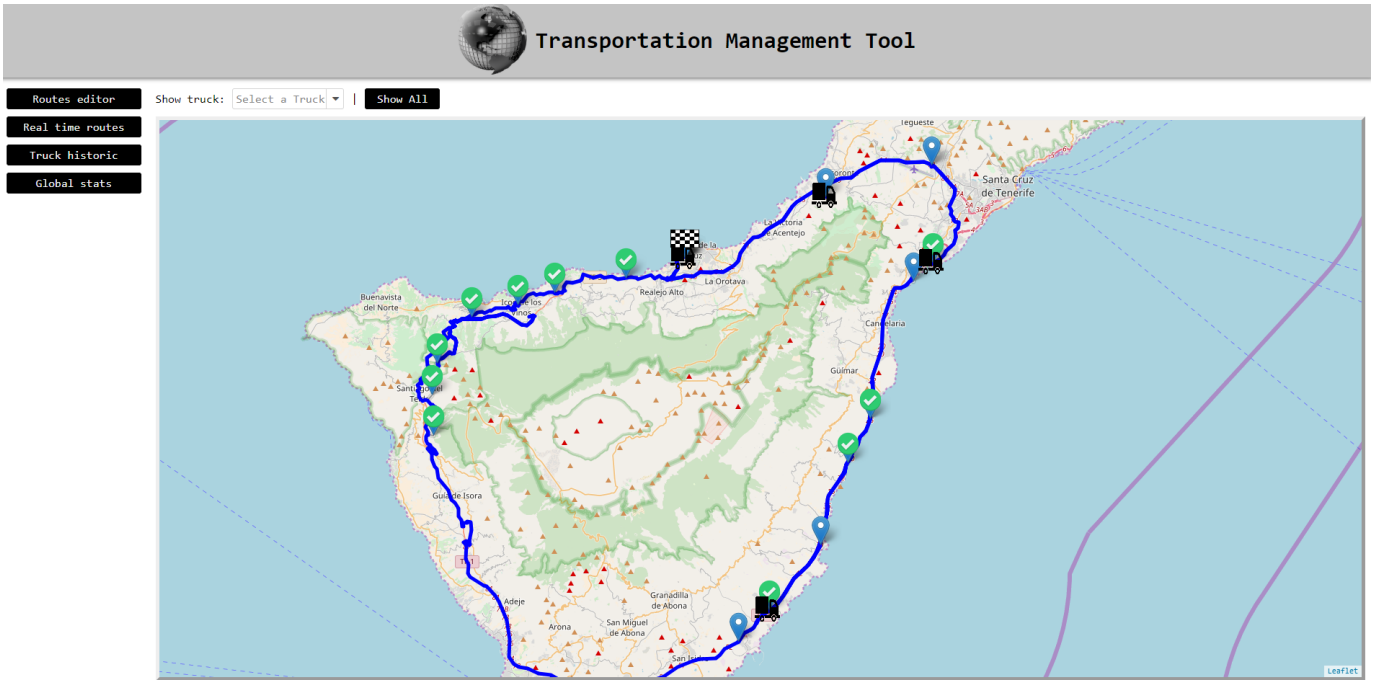


Figura 4.14: Visualización de todas las rutas.

4.2.3. Pantalla de histórico de vehículos

Esta sección de la web esta compuesta por una tabla que contiene los datos de los camiones recogidos a través de la aplicación móvil.

La tabla de esta pantalla está creada con PrimeNG y dispone de algunas funcionalidades que ha sido posible implementar gracias a esta librería. Una de ellas es un filtro, situado en la parte superior izquierda de la tabla, que permite al usuario realizar una búsqueda por identificador del camión cuya información desee visualizar. La otra funcionalidad es la generación de informes en CSV. Los botones situados en la parte superior derecha de la tabla son los encargados de realizar esta funcion. El primero de ellos descarga un CSV toda la información contenida en la tabla, mientras que el segundo descarga un fichero con la información de las filas de la tabla que el usuario seleccione. Además, al lado de los botones de descarga se encuentra un botón para actualizar la información de la tabla. Al ser pulsado realiza las consultas correspondientes a la base de datos y rellena la tabla con la información recogida. En la figura 4.15 se muestra el aspecto de esta pantalla.

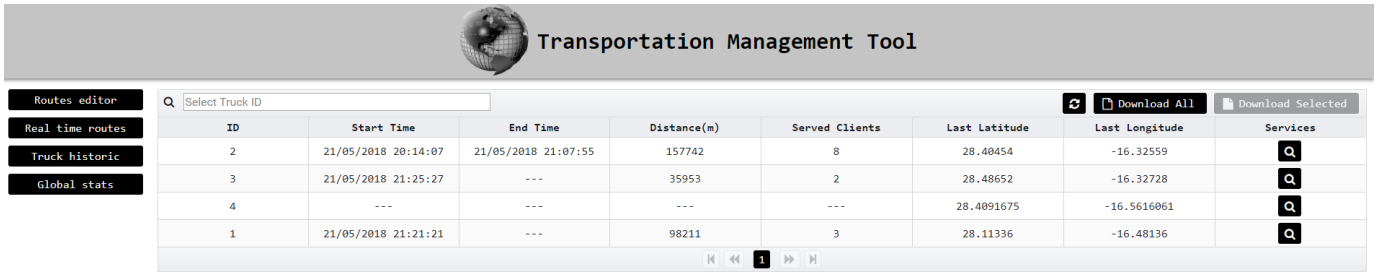


Figura 4.15: Histórico de vehículos.

Para cada uno de los vehículos se muestra su identificador, el momento de inicio de recorrido, el momento de finalización de recorrido, la distancia recorrida en metros, el número de clientes que ha servido, la latitud y longitud de la última ubicación registrada y los servicios realizados. La información de dichos servicios se muestra tras pulsar el botón correspondiente, tal y como se muestra en la Figura 4.16.

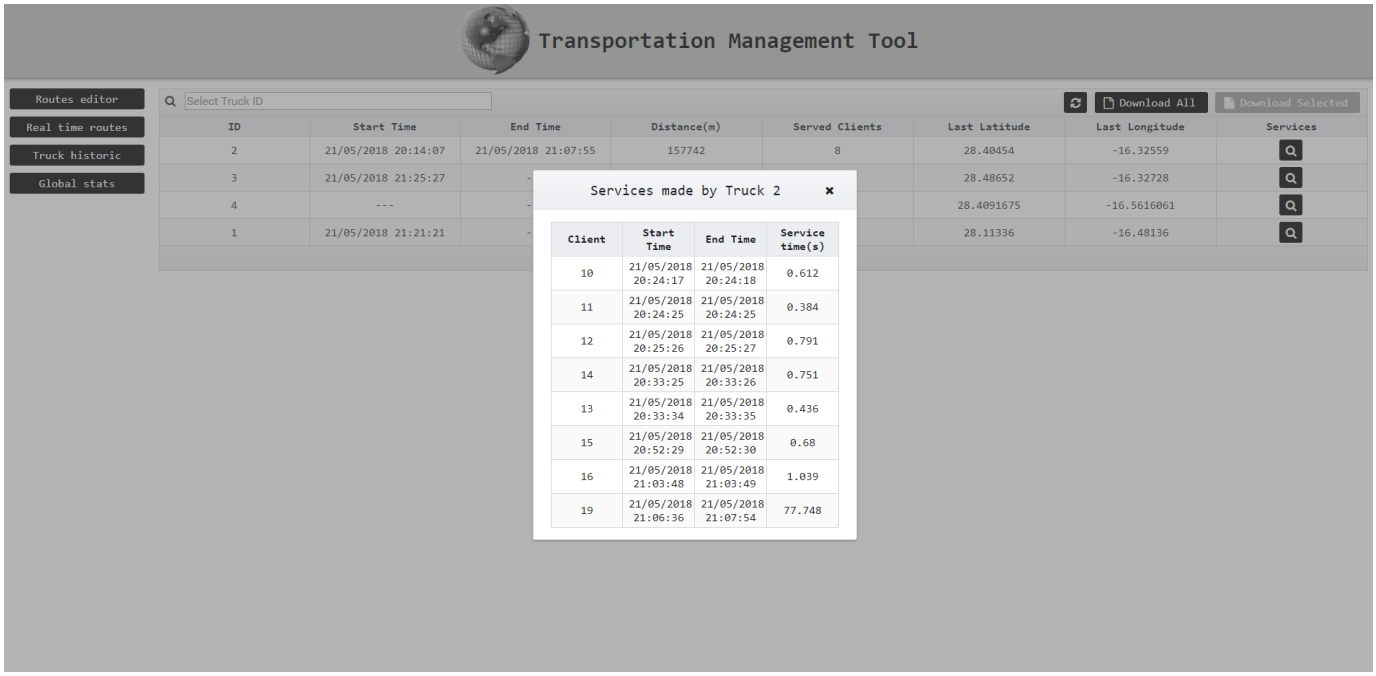


Figura 4.16: Servicios de un vehículo.

4.2.4. Pantalla de estadísticas globales

La última sección de la web es el módulo de estadísticas globales. La idea tras esta pantalla es poder organizar todos los datos recogidos a lo largo de la aplicación móvil, tanto de clientes como de camiones, en diferentes gráficas.

Estas gráficas se han creado gracias a la librería PrimeNG y obtienen los datos de los endpoints definidos en la API REST de la plataforma.

En la parte superior de esta pantalla existen diversas pestañas, una por cada gráfica definida, que si se pulsan muestran la gráfica en cuestión. Se han planteado tres gráficas distintas:

- **Porcentaje de tiempo invertido:** Esta gráfica, un diagrama de barras correspondiente a la Figura 4.17, muestra para cada vehículo el porcentaje de tiempo que ha invertido en desplazarse, en color azul, y el porcentaje de tiempo que ha dedicado a servir a los clientes, en color verde. En este caso el eje X representa a cada uno de los camiones y el eje Y, el porcentaje sobre 100.

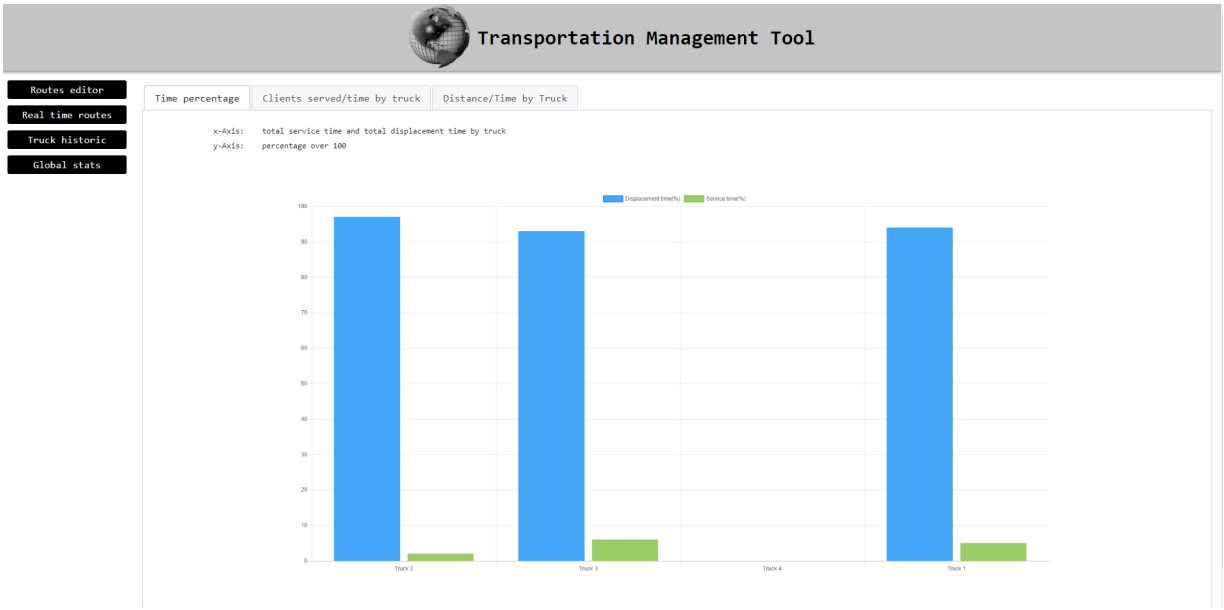


Figura 4.17: Gráfica de porcentaje de tiempo invertido.

- **Número de clientes atendidos durante el recorrido:** En esta gráfica correspondiente a la Figura 4.18 el eje X representa el número de clientes atendidos y el eje Y el tiempo de recorrido en segundos. Con esta gráfica se pretende ver para cada camión cuántos clien-

tes atiende durante su recorrido, por lo que cada una de las líneas dibujadas representa a un vehículo diferente.

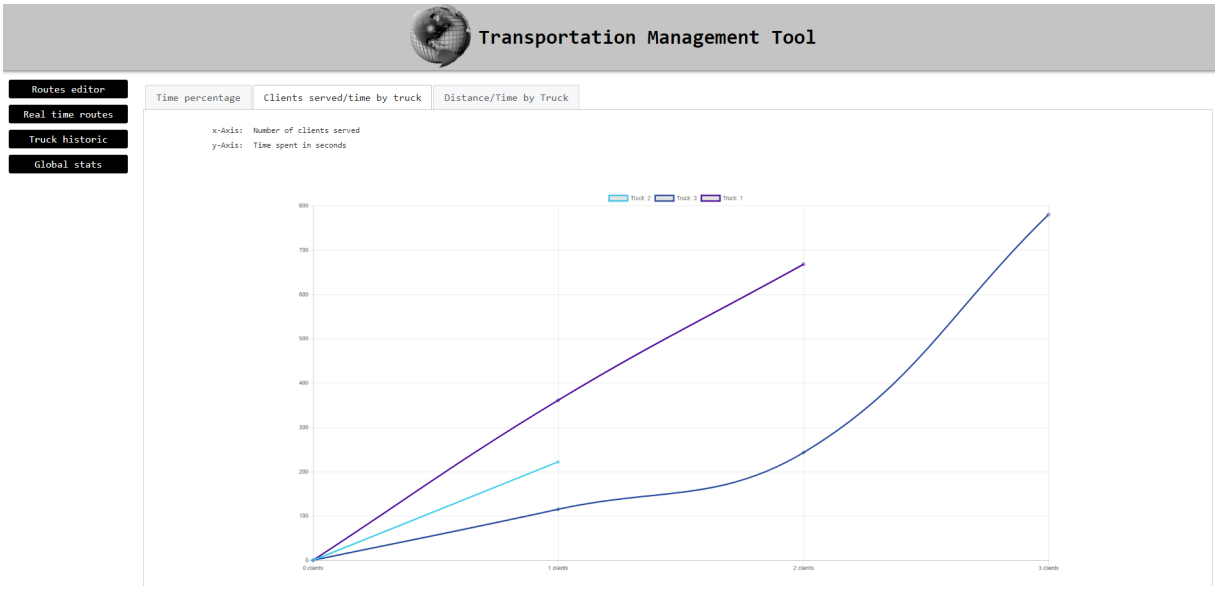


Figura 4.18: Gráfica de clientes atendidos durante el recorrido.

- **Relación distancia/tiempo:** La última gráfica refleja para cada vehículo la relación entre la distancia que han recorrido y el tiempo que han tardado en realizar su ruta. En caso de que su ruta no haya finalizado en el momento de realizar la consulta de datos, se tomará como referencia la fecha actual. En el eje X se muestra el tiempo del recorrido en segundos mientras que en el eje Y aparece la distancia recorrida en metros. La figura 4.19 muestra esta gráfica.

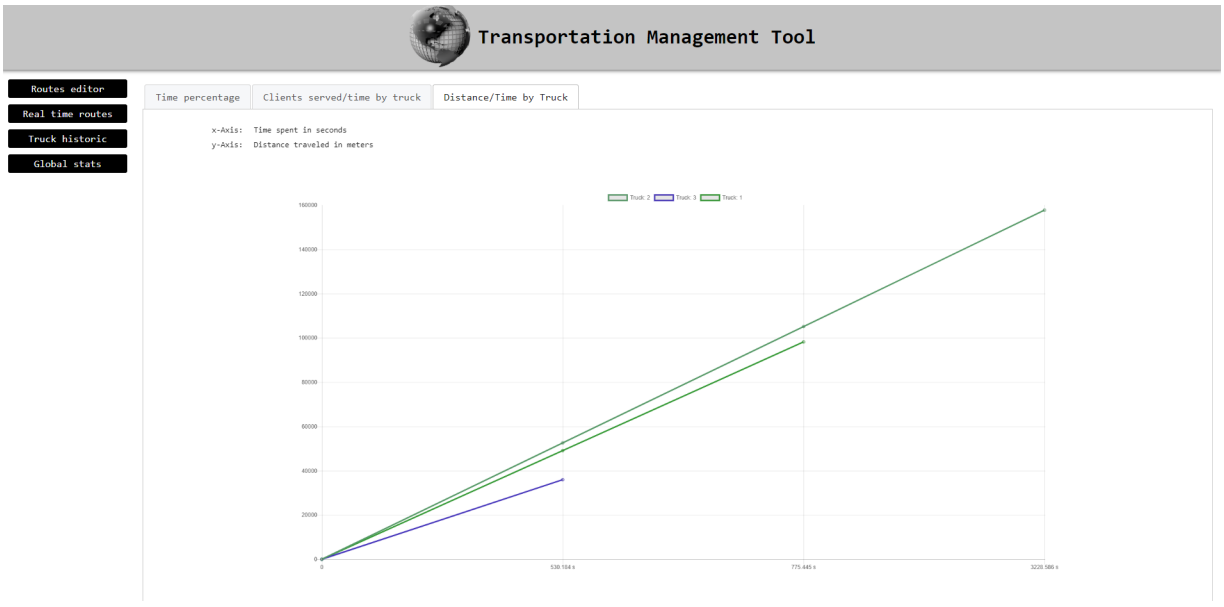


Figura 4.19: Gráfica de relación distancia/tiempo por vehículo.

Capítulo 5

Líneas futuras de trabajo

5.1. Versión en castellano

Una vez descrito el proyecto en su totalidad se pueden plantear cuáles serían los posibles cambios o mejoras planteados de cara a una continuación del desarrollo del mismo:

- **Combinar la herramienta web con un software de optimizacion:** Desde un primer momento la herramienta ha sido concebida para poder trabajar con un software de optimización de rutas que opere en el lado del servidor. En el modelo de datos planteado, concretamente en la entidad cliente, existe un atributo llamado orden que define la prioridad con la que dicho cliente debe de ser atendido dentro de una ruta para un camión concreto. Es por ello que cualquier petición lanzada al servidor devolverá la ruta con los clientes ordenados según este atributo. En el producto actual el orden de servicio de clientes se modifica desde el cliente web para poder adaptar las soluciones en tiempo real, pero sería ideal disponer de un software de optimización de rutas que permita realizar esto de forma automática. De esta forma la potencia de la herramienta se incrementaría considerablemente.
- **Geolocalización:** El producto actual cumple a modo de simulador, puesto que es necesario simular el movimiento del camión en el cliente móvil para poder realizar una demostración del funcionamiento de la app sin necesidad de desplazamiento. De cara a tener un producto totalmente adaptado a situaciones reales, sería necesario cambiar el

cliente móvil y añadir un sistema de geolocalización que ubique la posición del camión en tiempo real. Afortunadamente este cambio es bastante sencillo puesto que la lógica de la aplicación móvil ha sido diseñada teniendo este apartado en cuenta.

- **Creación y edición de clientes y vehículos desde el cliente web:** Tal y como está planteado el producto de cara a su presentación, los clientes y camiones son insertados en la base de datos antes de poner a funcionar todos los componentes del mismo. Todas las funcionalidades de edición actualmente implementadas están relacionadas con la modificación de las rutas en tiempo real, pero también sería interesante de cara al futuro el poder contar con un editor de clientes que permita añadirlos a la base de datos o que permita cambiar la información de los mismos, como la dirección o el nombre. Lo mismo para añadir o modificar los camiones de la flota de la empresa.

- **Migración del backend a Spring**[1]**:** Para la realización del proyecto se ha optado por la utilización de JavaScript como lenguaje de programación del backend. Una de las principales motivaciones que llevaron a la elección de JavaScript es que ofrece una curva de aprendizaje bastante baja, permitiendo obtener resultados bastante eficaces en un corto periodo de tiempo. Una vez finalizado el desarrollo del proyecto se plantea aplicar tecnologías que permitan ir un paso más allá. Spring actualmente es una de las soluciones más utilizadas como tecnología de backend, debido a la efectividad que ofrece a la hora de realizar operaciones con bases de datos o controlando sesiones de usuarios.

- **Construcción de un sistema de login robusto:** El login con el que cuenta la parte móvil del proyecto actualmente dispone de un mecanismo bastante simple de inicio de sesión que se limita a cumplir con las exigencias mínimas debido a que todo el trabajo se ha dedicado a la funcionalidad de la aplicación y no a la seguridad. Es por ello que una de las futuras mejoras del proyecto es implementar un mecanismo de login con seguridad y manejo de sesiones con tokens. A pesar de que JavaScript ofrece herramientas para ello, lo ideal sería aprovechar la migración del backend a Spring anteriormente mencionada y utilizar Spring Security para realizar la implementación del nuevo sistema seguro de inicio de sesión.

[1] `https://spring.io/`

- **Mejorar la interacción en tiempo real:** Gracias a socket.IO se han podido implementar varias funcionalidades que trabajan en tiempo real, como por ejemplo el replicar el movimiento del camión en el cliente web. En esta línea y aprovechando el potencial que ofrece la librería socket.IO sería interesante contar con más funcionalidades en tiempo real. Una de esas funcionalidades planteadas para su futura implementación es un sistema de envío de mensajes que mejore la interacción entre cliente web y móvil. De esta forma sería posible que el cliente móvil reciba notificaciones cuando haya cambiado su ruta, cuando un cliente al que tenga que atender esté a punto de cerrar, etc.

- **Añadir más gráficas al módulo de estadísticas:** Las gráficas actualmente implementadas dentro del módulo de estadísticas del cliente web son solo ejemplos que muestran información que se ha considerado interesante visualizar. Tal y como está construída la aplicación es posible diseñar otras gráficas que permitan visualizar más información recogida por el cliente móvil. Por ello es interesante que de cara al futuro se realizara un estudio a empresas del sector para conocer qué datos sería adecuado mostrar en las gráficas y añadirlos para poder adaptar aun mas el producto a las necesidades reales de los clientes.

- **Versión de la app móvil para iOS:** Una de las ventajas que ofrece Ionic y que motivan su uso es el poder desarrollar una aplicación móvil y posteriormente generar una versión tanto para Android como para iOS. Desde el principio se había planteado la realización de una aplicación móvil para dispositivos Android y es por ello que es la versión escogida para la entrega de este proyecto, pero en un futuro también es posible generar una versión para dispositivos Apple.

5.2. English version

Once the project has been described in its entirety, it is possible to consider what the possible changes or improvements might be in order to continue the development of the project:

- **Combine the web tool with an optimization software:** From the very beginning the tool has been designed to work with a route optimization software that operates on the server side. In the proposed data model, specifically in the client entity, there is an attribute called order that defines the priority with which the client must be attended within a route for a specific truck. This is why any request launched to the server will return the path with the clients sorted according to this attribute. In the current product the customer service order is modified from the web client in order to be able to adapt the solutions in real time, but it would be ideal to have a route optimisation software that allows this to be done automatically. This would considerably increase the power of the tool.
- **Geolocation:** The current product acts as a simulator, since it is necessary to simulate the movement of the truck in the mobile client in order to demonstrate the operation of the app without the need to travel. In order to have a product totally adapted to real situations, it would be necessary to change the mobile client and add a geolocation system that locates the position of the truck in real time. Fortunately this change is quite simple as the logic of the mobile application has been designed with this section in mind.
- **Creation and edition of clients and vehicles from the web client:** As the product is presented, the clients and trucks are inserted in the database before all the components of the same are put into operation. All the editing functionalities currently implemented are related to the modification of the routes in real time, but it would also be interesting for the future to have a client editor that allows adding them to the database or changing their information, such as address or name. The same goes for adding or modifying the trucks in the company's fleet.
- **Backend migration to Spring[2]:** For the implementation of the project, JavaScript has been chosen as the backend programming langua-

[2] https://spring.io/

ge. One of the main motivations that led to the choice of JavaScript is that it offers a rather low learning curve, allowing you to obtain quite effective results in a short period of time. Once the project has been completed, the idea is to apply technologies that allow us to go one step further. Spring is currently one of the most widely used solutions as a backend technology, due to the effectiveness it offers when performing operations with databases or controlling user sessions.

- **Construction of a robust login system:** The login of the mobile part of the project currently has a fairly simple login mechanism that is limited to meeting the minimum requirements because all the work has been dedicated to the functionality of the application and not to security. That is why one of the future improvements of the project is to implement a login mechanism with security and session management with tokens. Although JavaScript provides tools to do this, ideally you should take advantage of the backend migration to Spring mentioned above and use Spring Security to implement the new secure logon system.

- **Improve real-time interaction:** Thanks to socket.IO, several features have been implemented that work in real time, such as replicating the movement of the truck in the web client. In this line and taking advantage of the potential offered by the socket.IO library, it would be interesting to have more features in real time. One of those functionalities proposed for its future implementation is a message sending system that improves the interaction between web and mobile clients. This way it would be possible for the mobile client to receive notifications when it has changed its route, when a customer it has to serve is about to close, etc.

- **Add more graphs to the statistics module:** The graphs currently implemented within the web client statistics module are only examples that show information that has been considered interesting to visualize. As the application is built it is possible to design other graphs that allow you to visualize more information collected by the mobile client. For this reason, it is interesting that in the future a study of companies in the sector will be carried out to find out which data it would be appropriate to show in the graphs and add them in order to be able to adapt the product even more to the needs of the market.

- **Version of the mobile app for iOS:** One of the advantages that Ionic offers and motivates its use is the ability to develop a mobile application and then generate a version for both Android and iOS. From the beginning, the idea was to create a mobile application for Android devices and that is why it is the version chosen for the delivery of this project, but in the future it is also possible to generate a version for Apple devices.

Capítulo 6

Conclusiones

6.1. Versión en castellano

Desde un principio, este proyecto se planteó con un claro objetivo: desarrollar una plataforma que ayudara a las empresas de transporte a lidiar con la planificación de sus rutas de reparto, optimizando así sus procesos logísticos de la forma más efectiva posible.

El resultado final de este proceso de desarrollo ha sido más que satisfactorio. La herramienta resultante no solo se ha creado con las tecnologías más actuales y que son utilizadas por numerosas empresas en el mundo laboral, sino que además recoge los aspectos más destacados de otros softwares similares y los reune en un mismo ecosistema: tanto la edición de rutas en tiempo real, como las funcionalidades de GPS o el registro de estadísticas, entre otras.

De cara al futuro la herramienta aun tiene mucho que ofrecer. Muchas ideas y funcionalidades han quedado en el tintero debido a las limitaciones de tiempo y a los requerimientos funcionales del prototipo. Por tanto, aunque el producto actual es completamente funcional, aún queda trabajo por delante y sería interesante seguir realizando mejoras como las comentadas en el capítulo 5 que lo hagan aun más completo.

El proceso de investigación implícito en el desarrollo de un proyecto como este ha supuesto todo un reto. Poder solventar los problemas dados durante todo este tiempo, así como trabajar con tecnologías que desconocía hasta ahora y establecer mecanismos para definir un funcionamiento correcto de la aplicación han supuesto un gran esfuerzo y dedicación que

se traducen un crecimiento no solo como ingeniero informático, sino como persona. En resumidas cuentas, este trabajo me ha preparado enormemente de cara al futuro y mucho de lo aquí aprendido lo aplicaré en futuros proyectos.

6.2. English version

The main goal of this project is to develop a platform that helps transport companies to deal with the planning of their delivery routes, optimizing their logistics processes as effectively as possible.

The resulting software has been created with the latest technologies, used by many companies nowadays. The platform also includes the most important aspects of other similar softwares and brings them together: the editing of routes in real time, GPS features or the recording of statistics, among others.

There are many ideas and functionalities that have been proposed as future requests, that haven’t been implemented due to time constraints and functional requirements of the prototype. Therefore, although the current product is fully functional, there is still work to be done and it would be interesting to continue making improvements such as those discussed in chapter 5 that make it even more complete.

The research process involved in the development of such a project has been a challenge for me. Being able to solve the problems encountered during all this time, as well as working with technologies that I was unaware of until now and establishing mechanisms to define the correct functioning of the application have meant a great effort and dedication that have resulted in growth not only as a computer engineer, but as a person. In short, all this work has prepared me enormously for the future and I will apply all I have learned here in future projects for sure.

Apéndice A

Instrucciones de despliegue

Este anexo sirve de manual de instrucciones para poder poner en funcionamiento el proyecto en local. Todo el código se encuentra en mi repositorio de GitHub: `https://github.com/alu0100782974/TFG`

A.1. Descripción del proceso

Antes de proceder con las instrucciones es necesario describir los pasos que se realizan durante todo el proceso de despliegue de la plataforma.

- **Crear contenedor Docker con la base de datos**

  Tal y como se ha planteado para este proyecto, la base de datos está alojada en un contenedor Docker que extiende de una imagen base de MongoDB. Por tanto, se necesita crear dicho contenedor y realizar la carga inicial de datos. Para ello se utiliza un fichero JavaScript donde se define un usuario para acceder a la base de datos, las colecciones de datos y los diferentes documentos que definen los objetos a insertar. Las operaciones de creación de colecciones y usuarios y las inserciones de documentos se delegan a la librería Mongoose.

- **Arrancar el servidor**

  Para que los clientes puedan acceder a la información alojada en la base de datos es necesario ejecutar el código del servidor. Este código incluye tres mecanismos fundamentales. Primero la conexión con la base de datos y los mapeos de las colecciones creadas. De esta forma es posible establecer una conexión y consultar y actualizar documentos. Segundo, la API REST que define los endpoints a través de los cuales los clientes pueden acceder a la información. Por último, la implementación del socket, con las funciones necesarias para realizar los broadcast de datos y comprobar las conexiones de los clientes.

- **Arrancar los clientes**

  Una vez la base de datos esta creada y el servidor esta en funcionamiento, es posible ejecutar el código de los dos clientes. Para el cliente móvil existen dos posibles formas: ejecutarlo mediante la apk generada a partir del proyecto dentro de un dispositivo móvil o mediante el comando 'ionic serve' de Ionic que lo despliega en un navegador web. Mientras tanto, el cliente web se despliega en un navegador web gracias al comando 'ng serve' de 'angular cli'. Una vez funcionando, ambos clientes se conectan al servidor a través de la API REST y el socket y la plataforma estaría completamente desplegada.

A.2. Requisitos

- **Node.js:** Para instalar las librerías requeridas durante el despliegue es necesario que el sistema disponga de Node.js. Se puede descargar desde la url: `https://nodejs.org/es/download/`

- **Docker:** Para poder generar el contenedor que aloja la base de datos es necesario disponer de esta herramienta en la máquina donde se desea realizar el despliegue. Se puede descargar desde la url: x`https://www.docker.com/community-edition`

- **Windows 10:** Las pruebas de despliegue se han realizado utilizando el Sistema Operativo Windows 10, ya que la herramienta Docker viene integrada y los comandos de despliegue están adaptados a este entorno. En versiones anteriores de Windows es posible que el

despliegue automático no funcione correctamente y sería necesario realizarlo manualmente.

- **Git:** Debido a que se necesita trabajar con un repositorio, se requiere de Git para poder realizar la descarga automática de los directorios del proyecto. Se puede descargar Git for Windows en el siguiente enlace: `https://git-scm.com/downloads`

A.3. Pasos a seguir

- Abrir una terminal de Docker para arrancar el servicio.
- Seleccionar un directorio de trabajo donde descargar las carpetas del repositorio.
- Ir al repositorio `https://github.com/alu0100782974/TFG` y descargar los scripts de despliegue: **deployServer.sh y deployWeb.sh.** Para el cliente móvil o bien se descarga la apk **android-debug.apk** que está en el mismo repositorio o bien el script **deployApp.sh** para ejecutarlo en el navegador de la máquina.
- Abrir el directorio de trabajo y guardar los scripts en él.
- Abrir una terminal bash y ejecutar deployServer.sh. **Es necesario que este script se ejecute primero.**
- Abrir otra terminal bash y ejecutar deployWeb.sh. **Una vez desplegado, se debe visualizar la web en localhost:4200.**
- En caso de haber descargado el script de despliegue de la app, abrir otra terminal bash y ejecutar deployApp.sh. **Una vez desplegado, debe ser accesible a través de la URL localhost:8100 (visualización web) o localhost:8100/ionic-lab (visualización móvil)**. Los datos necesarios para la conexión en el login son servidor: **localhost**, puerto API REST: **3001** y puerto Socket: **8100** .
- En caso de querer ejecutar la app en un dispositivo móvil se deben habilitar previamente los permisos de ejecución de aplicaciones en el dispositivo. A continuación solo hay que pasar el fichero android-debug.apk al móvil, instalarlo y ejecutar la aplicación. los datos necesarios para la conexión desde el dispositivo móvil son: puerto API

REST: **3001**, puerto Socket: **8100** y el servidor debe ser **la dirección IP de la máquina donde se haya desplegado el servidor.** Para este caso es necesario que el dispositivo móvil esté en la misma red que la máquina donde se ejecutaron los scripts y es posible que se requiera desactivar su firewall.

Apéndice B

Esquema de colecciones de la base de datos

B.1. Servicios

Servicios		
Atributo	**Tipo de dato**	**Descripción**
truckId	Number	Identificacdor del camión que ha realizado el servicio.
clientId	Number	Identificador del cliente que ha sido atendido por el camión con el id "truckId".
start	Date	Inicio del servicio.
end	Date	Final del servicio.
serviceTime	Number	Tiempo total del servicio en segundos.

Tabla B.1: Esquema de servicios

B.2. Camiones

Camiones		
Atributo	**Tipo de dato**	**Descripción**
id	Number	Cada camión dispone de un id representado por un número entero. Este id se utiliza para identificar a cada camión y para controlar el login de la app móvil.
startTime	Date	Indica el momento en la que da comienzo el recorrido del camión.
endTime	Date	Indica el final del recorrido del camión. En caso de que se agreguen más clientes tras finalizar la ruta, este atributo se actualiza.
distance	Number	Atributo numérico que representa la distancia total recorrida por el camión. Dicha distancia se recoge en metros. Se utiliza en el módulo de estadísticas.
clientsServed	Number	Indica el número de clientes atendidos por el camión en cuestión.También es utilizado en el módulo de estadísticas.
lastLat	Number	Representa la latitud de la última ubicación registrada.
lastLon	Number	Representa la longitud de la última ubicación registrada.
startLat	Number	Representa la latitud de la posición geográfica en la que el camión inicia su recorrido.
startLon	Number	Representa la longitud de la posición geográfica en la que el camión inicia su recorrido.

Tabla B.2: Esquema de camiones

B.3. Clientes

Clientes		
Atributo	**Tipo de dato**	**Descripción**
id	Number	Cada cliente dispone de un id representado por un número entero. Este id se utiliza para identificar a cada cliente y facilitar su búsqueda.
name	String	El nombre de cliente, representado por un string se utiliza dentro de la aplicación para ofrecer más información al camionero.
lat	Number	Latitud de la ubicación del cliente.
lon	Number	Longitud de la ubicación del cliente.
truckId	Number	Representa el camión que atiende al cliente.
order	Number	Indica el orden en el que el cliente debe ser atendido dentro de la ruta. Es decir, su prioridad de servicio.
address	String	String con la dirección donde se realiza el servicio. Ayuda al camionero a conocer a dónde debe dirigirse para realizar su siguiente entrega. En el producto actual su uso es meramente anecdótico, ya que cada cliente queda localizado según sus coordenadas.
open	Number	Hora de apertura del cliente.
close	Number	Hora de cierre del cliente.
closed	Boolean	atributo booleano que, tras comprobaciones del tiempo actual con el horario del cliente, obtiene un valor true o false.
served	Boolean	Atributo booleano, con valor false por defecto, que indica si el cliente ha sido servido o no.
serving	Boolean	Indica si el cliente está siendo atendido por el camión correspondiente. Es de vital importancia tener este aspecto bajo control, puesto que un cliente que está siendo atendido no puede ser modificado dentro de una ruta.

Tabla B.3: Esquema de clientes

Apéndice C

API REST

Este anexo sirve a modo de documentación de la API REST creada para este proyecto. Esta API está compuesta de cuatro endpoints diferentes que son accesibles desde los clientes: /clients, /trucks, /services y /locations. Para cada uno de ellos se específican los verbos con los que pueden ser llamados y para cada posible llamada se indica:

- Método HTTP
- Acción que realiza
- Parámetros obligatorios y/o opcionales
- Respuesta, código y cabecera de la misma
- Ejemplo de uso

Cabe aclarar que cualquier llamada realizada con un verbo diferente de los aquí descritos devuelve un error 404.

C.1. /clients

URL	/clients
Método HTTP	GET
Acción	Consulta de clientes
Parámetros opcionales	id=[integer] name=[string] lat=[double] lon=[double] truckId=[integer] order=[integer] address=[string] open=[integer] close=[integer] served=[boolean] serving=[boolean]
Código de respuesta	200
Cabecera de respuesta	application/json
Respuesta	JSON con los clientes que cumplan las condiciones especificadas en la query
Ejemplo de uso	GET /clients?served=true&closed=false

Tabla C.1: GET /clients

URL	/clients
Método HTTP	PUT
Acción	Actualización de clientes

Parámetros obligatorios	Documento JSON de un cliente.
	Estructura:
	{
	'lat': [double],
	'lon': [double],
	'id':[integer],
	'name': [string],
	'address': [string],
	'open': [integer],
	'close': [integer],
	'truckId': [integer],
	'served': [boolean],
	'serving': [boolean],
	'closed': [boolean],
	'order':[integer]
	}

Código de respuesta	200
Cabecera de respuesta	application/json
Respuesta	Copia del objeto modificado

Ejemplo de uso	PUT /clients
	Body:
	{
	'lat': 28.2741662,
	'lon': -16.3852894,
	'id': 3,
	'name': 'client3',
	'address': 'address3',
	'open': 8,
	'close': 22,
	'truckId': 1,
	'served': false,
	'serving': false,
	'closed': false,
	'order': 1
	}

Tabla C.2: PUT /clients

C.2. /trucks

URL	/trucks
Método HTTP	GET
Acción	Consulta de camiones
Parámetros opcionales	id=[integer], startTime= [date], endTime=[date], distance=[integer], clientsServed=[integer], lastLat=[double], lastLon=[double], startLat=[double], startLon=[double],
Código de respuesta	200
Cabecera de respuesta	application/json
Respuesta	JSON con los camiones que cumplan las condiciones especificadas en la query
Ejemplo de uso	GET /trucks?id=3

Tabla C.3: GET /trucks

URL	/trucks
Método HTTP	PUT
Acción	Actualización de un camión

Parámetros obligatorios	Documento JSON de un camión.
	Estructura:
	{
	’id’: [integer],
	’startTime’: [date],
	’endTime’: [date],
	’distance’: [integer],
	’clientsServed’: [integer],
	’lastLat’: [double],
	’lastLon’: [double],
	’startLat’: [double],
	’startLon’: [double],
	}

Código de respuesta	200
Cabecera de respuesta	application/json
Respuesta	Copia del objeto modificado.

Ejemplo de uso	PUT /trucks
	Body:
	{
	’id’: 2,
	’startTime’: null,
	’endTime’: null,
	’distance’: 0,
	’clientsServed’: 0,
	’lastLat’: 28.4091675,
	’lastLon’: -16.5616061,
	’startLat’: 28.4091675,
	’startLon’: -16.5616061,
	}

Tabla C.4: PUT /trucks

C.3. /services

URL	/services
Método HTTP	GET
Acción	Consulta de servicios
Parámetros opcionales	truckId=[integer], clientId=[integer], start= [date], end=[date], serviceTime=[integer],
Código de respuesta	200
Cabecera de respuesta	application/json
Respuesta	JSON con los servicios que cumplan las condiciones especificadas en la query
Ejemplo de uso	GET /services?truckId=3

Tabla C.5: GET /services

URL	/services
Método HTTP	POST
Acción	Inserción de un documento de servicios

Parámetros obligatorios	Objeto de tipo servicio.
	Estructura:
	{
	’clientId’: [integer],
	’truckId’: [integer],
	’start’: [date],
	’end’: [date],
	’serviceTime’: [integer]
	}

Código de respuesta	200
Cabecera de respuesta	application/json
Respuesta	Copia del objeto insertado en base de datos

Ejemplo de uso	POST /services
	Body:
	{
	’clientId’: 17,
	’truckId’: 3,
	’start’: ’2018-05-30T19:42:27.740Z’,
	’end’: ’2018-05-30T19:42:28.593Z’,
	’serviceTime’: 0.853,
	}

Tabla C.6: POST /services

C.4. /locations

URL	/locations
Método HTTP	GET
Acción	Consulta del histórico de ubicaciones
Parámetros opcionales	truckId=[number]
Código de respuesta	200
Cabecera de respuesta	application/json
Respuesta	Ubicaciones del camión con el identificador especificado
Ejemplo de uso	GET /locations?truckId=3

Tabla C.7: GET /locations

URL	/locations
Método HTTP	POST
Acción	Inserción de un objeto localización

Parámetros obligatorios	Objeto de tipo localización.
	Estructura:
	{
	"truckId": [number],
	"lat": [double],
	"lon": [double]
	}

Código de respuesta	200
Cabecera de respuesta	application/json
Respuesta	Copia del objeto insertado

Ejemplo de uso	POST /locations
	Body:
	{
	"truckId": 3,
	"lat": 28.40875,
	"lon": -16.56746
	}

Tabla C.8: POST /locations

Bibliografía

[1] Dawei Lu. *Fundamentals of Supply Chain Management*. Ventus Publisher Aps, 2014.

[2] Peter Baker Alan Rushton, Phil Croucher. *The handbook of logistics and distribution management*. Kogan Page Limited, 2010.

[3] Rodrigo Rojas. La importancia del transporte en la cadena logística. `https://mba.americaeconomia.com/`, 2014.

[4] Richard van Hooijdonk. *The Future of Logistics and Supply Chain Management*. 2018.

[5] Vladimir Agafonkin. Leaflet api documentation. `https://leafletjs.com/reference-1.3.0.html`, 2017.

[6] Pier Liedman. Leaflet routing machine api documentation. `http://www.liedman.net/leaflet-routing-machine/api/`, 2015.

[7] Andrew Lombardi. *Websocket, Lightweight client-server communications*. O'Reilly Media Inc, 2015.

[8] Mark Masse. *REST API Design Rulebook*. O'Reilly Media Inc, 2011.

[9] Casimir Saternos. *Client-Server Web Apps with JavaScript and Java*. O'Reilly Media Inc, 2014.