UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf
Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Propuesta de modelo de gestión para la mejora del proceso de abastecimiento y distribución del pescado en un establecimiento de comercialización alimenticia

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Jesús Yáñez Signes

Tutor: Ignacio Gil Pechuán

Curso 2019-2020

Resumen

El objetivo principal de la siguiente propuesta de TFG consiste en detallar la elaboración de un modelo y prueba de concepto que lo use. Con dicha prueba de concepto se busca mejorar el proceso de abastecimiento y distribución en la relación de gestión de proveedores de pescado sobre un establecimiento comercial.

Para esto, se propondrá un modelo matemático, el cual será utilizado en un algoritmo de optimización que se usará en la prueba de concepto a detallar. A través de este modelo y la PoC, se pretenden resolver situaciones reales en las que, tras una identificación de sus necesidades y procedimientos actuales, se pueda aumentar la productividad de los recursos en un establecimiento comercial. Con esto, se espera obtener una mejora de los rendimientos por dicha gestión mediante una optimización de los procesos de transporte y abastecimiento.

Se hará uso de una función multiobjetivo que minimice el coste del transporte de mercancías entre diferentes puntos, siendo estos lonjas, almacenes y tiendas de un establecimiento de venta de géneros alimenticios al por mayor. Además, se detallarán las principales tecnologías utilizadas, la arquitectura del proyecto, el proceso de preparación del entorno y se mostrarán algunas pruebas realizadas.

Palabras clave: optimización, modelo matemático, algoritmo, prueba de concepto, gestión.

Abstract

The main purpose of the following TFG proposal is to detail the development of a model and a proof of concept that uses it. This proof of concept seeks to improve the supply and distribution process existing in the management relationship of fish suppliers over a commercial establishment.

To carry this out, a mathematical model will be proposed, which will be used in an optimization algorithm. Also, said algorithm will be used in the proof of concept. Through this model and the PoC, is aimed to solve real situations in which, after identifying their current needs and procedures, a commercial establishment could increase the productivity of resources. With this, it is expected to obtain an improvement in the efficiency by said management through an optimization of the transport and supply processes.

An objective function will be used to minimize the cost of transporting goods between different points, being these markets, stores, and shops of an establishment of wholesale food sales. In addition, the main technologies used, the project architecture, the environment preparation process and some tests performed will be detailed.

Keywords: optimization, mathematical model, algorithm, proof of concept, management.

Resum

L'objectiu principal de la següent proposta de TFG consisteix en detallar l'elaboració d'un model i prova de concepte que l'utilitze. Amb dita prova de concepte es busca millorar el procés d'aprovisionament y distribució en la relació de gestió de proveïdors de peix sobre un establiment comercial.

Per a dur a terme açò, es proposarà un model matemàtic, el qual serà utilitzat en un algoritme d'optimització que s'utilitzarà a la prova de concepte a descriure. A través d'aquest model y la PoC, es pretenen resoldre situacions verídiques en les quals, després d'una identificació de les seves necessitats y procediments actuals, es pugui augmentar la productivitat dels recursos en un establiment comercial. Amb açò, s'espera obtenir una millora dels rendiments per dita gestió mitjançant una optimització dels processos.

Es farà ús d'una funció objectiu que minimitzi el cost del transport de mercaderies entre diferents punts, sent aquests, llotges, magatzems i tendes d'un establiment de venta de gèneres alimentaris a l'engròs. A més, es detallaran les principals tecnologies utilitzades, l'arquitectura del projecte, el procés de preparació de l'entorn y es mostraran algunes de les proves realitzades.

Paraules clau: optimització, model matemàtic, algoritme, prova de concepte, gestió.

Tabla de contenidos

1. Introducción

Los supermercados o establecimientos de comercialización alimenticia cubren hoy en día un papel muy importante en nuestra sociedad. Albergan en ellos una amplia gama de productos, ahorrándonos la necesidad de tener que visitar varios establecimientos para conseguirlos y ofreciéndonoslos a precios relativamente asequibles.

La mayoría de las tiendas tradicionales han ido evolucionado hasta el punto de convertirse en grandes establecimientos de comercialización cuyo objetivo es cubrir el más amplio número de necesidades de los compradores o consumidores posibles. Cada vez son más las carnicerías, pescaderías, verdulerías…que han ampliado su repertorio de venta al cliente, tendiendo a simular pequeños supermercados.

Ser una empresa de comercialización de productos supone estar en contacto directo con los distintos proveedores, llevar un control del stock, una buena gestión del inventario de productos, llevar a cabo tareas de transporte de las mercancías… Todo lo dicho anteriormente, supone unos costes constantes e importantes para la empresa, es por ello, que estas buscan minimizar dichos costes, optimizando de la mejor manera posible, las tareas logísticas que incluyen la producción, abastecimiento, transporte, paletización… de la mercancía pertinente.

En esta propuesta de trabajo de fin de grado se planteará una prueba de concepto que ayudará a los supermercados que basan su estrategia competitiva en ser líderes en costes, a optimizar al máximo sus tareas de abastecimiento de pescado. Con esto se conseguirá un augmento del margen de beneficio obtenido.

Antes de entrar en materia, resultará interesante saber cuáles son las partes en las que se basa la estructura de una empresa o establecimiento de comercialización desde la perspectiva de su organización. Saber esto permitirá comprender mejor el funcionamiento de un supermercado e identificar donde se llevará a cabo el proceso de optimización. Para ello, se describirá una configuración organizacional bastante común en las empresas del sector en el siguiente punto.

1.1 Configuración organizativa de una empresa:

Cabe destacar que no todas las empresas dedicadas a la compraventa de productos alimenticios siguen estrictamente la misma estructura organizativa. Aun así, resultará útil describir a grandes rasgos una estructura frecuente en la mayoría de estas. Con esto, se identificarán cuáles son las partes a las que la implementación del proyecto de optimización facilitará las tareas.

Una posible organización, basándose en la configuración organizacional según Henry Mintzberg, profesor y académico internacionalmente reconocido y autor de varias publicaciones acerca de gestión y negocio, nos presenta una división de la empresa en un total de cinco partes:

- Núcleo de operaciones.
- Personal de apoyo.
- Cumbre estratégica.

- Línea media.
- Estructura técnica o tecnoestructura.

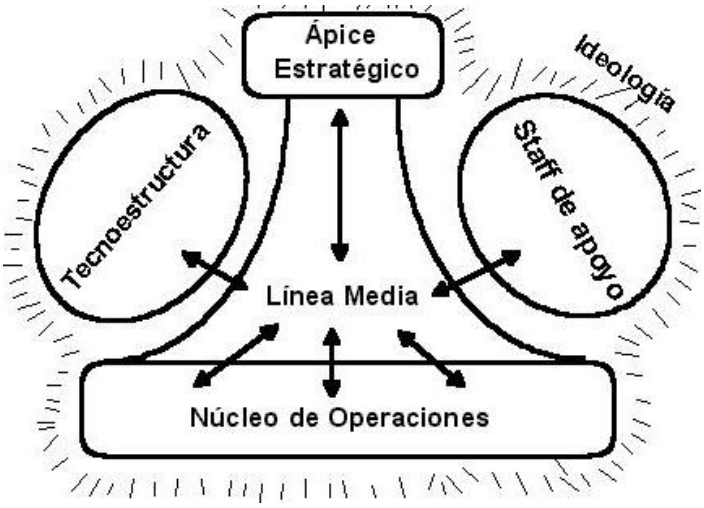

Figura 1: Esquema de la configuración organizativa de una empresa. Fuente: [1].

A continuación, se describirán con más detalle cada una de estas partes, mostradas en la figura anterior, destacando en cada una de ellas las figuras o grupos existentes que se pueden encontrar en un supermercado. Para ello, se utilizará como base el artículo publicado por Isabel Garrido en empresas.infoempleo.com titulado *El Modelo Mintzberg, una organización estructurada en la empresa* [2].

Para situar cada figura del ámbito de un establecimiento de comercialización en cada una de las partes, se seguirá el esquema de análisis del TFG de Marina Tripiana Rodríguez-Belza, alumna del Grado en Administración y Dirección de Empresas en la Facultad de Ciencias Económicas y Empresariales de la Universidad de Almería. Concretamente, me basare en el *Ánalisis del diseño estructural de la empresa* [56] que esta realiza, para desarrollar este apartado.

Cumbre estratégica

Es la parte más alta de la organización. Formada por el director de la empresa u organización y autoridades más significativas, así como también aquel personal que les presta apoyo directo. Los que pertenecen a dicha parte de la empresa, son los encargados de la visión global de la empresa, así como la definición de los objetivos de la organización y la relación con el entorno, los aspectos legales, etc.

Encontramos pues en esta parte a los directores regionales y el Comité de dirección, junto con el Consejo de Administración, encabezado por el presidente o presidentes de la empresa.

Línea media

Es la parte que se encuentra entre la cumbre estratégica (alta dirección) y el núcleo operacional. Dicha parte la forman los supervisores, gerentes y responsables encargados de asignar las tareas a aquellos cuyo objetivo es llevarlas a cabo. El propósito principal de esta parte de la organización es alcanzar las metas y objetivos definidos por los superiores.

Cabe destacar que, en algunos establecimientos de comercialización o supermercados, podemos encontrar dos figuras distintas al personal de tienda y que pertenecen también a la línea media. Estas son:

- **Coordinador de personal**
  Encargado de realizar funciones de coordinación, dirección de los trabajadores o personal de tienda a su cargo y tareas de planificación. Además de esto, es la figura encargada de supervisar que aspectos como la reposición de los distintos productos, atención al cliente, realización de pedidos y limpieza, se estén llevando a cabo correctamente.
- **Jefe de tienda**
  Se encarga de organizar las distintas actividades de la tienda, asegurándose que en esta se cumplan con los requisitos de atención al cliente, seguridad, calidad e higiene dictados por la organización. Es también el encargado de comunicar la situación al jefe de zona.

La figura del **jefe de zona** hace referencia a aquella persona encargada de coordinar varios supermercados próximos. Se encarga de supervisar que todo se esté llevando a cabo tal y como lo dicta la alta gerencia, es decir, cumpliendo con los estándares. Además, debe llevar a cabo un seguimiento de los distintos resultados y asegurarse que estos cumplen con la productividad, rentabilidad y presupuesto de ventas esperados.

Núcleo de operaciones

En esta parte encontramos a la mayoría de los trabajadores de la empresa, encargados de realizar las tareas sobre las cuales la empresa construye sus fundamentos. Es la parte formada por un conjunto de operarios encargados de realizar tareas de provisionamiento de servicios, así como de producción.

En nuestro caso, encontramos al personal de tienda y de logística. Entre las principales actividades en los centros logísticos podemos encontrar algunas como las de preparación de pedidos y las relacionadas con el transporte desde las fábricas de los proveedores (o lonjas) hasta los bloques logísticos (o almacenes) y desde estos últimos a las tiendas.

Personal de apoyo

Aquí se encuentra todo aquel personal y/o unidades encargadas de ofrecer servicios y que realizan funciones para la organización, sin pertenecer a su estructura operacional. Este personal es también el encargado de que la organización o empresa se adapte de mejor forma al entorno.

Encontramos en esta parte al personal de seguridad, así como al personal técnico de investigación, encargado de captar las distintas necesidades de los clientes, junto con el personal sanitario.

Estructura técnica o tecnoestructura

El personal perteneciente a esta parte de la empresa es el encargado de controlar y estandarizar los procesos de trabajo. Para ello, diseñan el planteamiento formal de cada proceso existente en la empresa y se encargan de controlarlo.

A esta parte de la empresa pertenece el departamento de recursos humanos y el de atención al cliente, entre otros.

Una vez descrita a grandes rasgos la posible organización en un supermercado, interesa saber en qué se basa dicho establecimiento a la hora de tomar las decisiones. Para ello, se introducirá el concepto de modelo de calidad total.

1.2 Modelo de calidad total:

En cuanto a la toma de decisiones, suponemos que la empresa para la cual va orientada la PoC[1] a desarrollar se basa en un modelo de gestión de Calidad Total. Dicho modelo ayudará a la organización a conseguir la máxima excelencia en la realización de cada una de sus tareas o acciones.

Mediante el seguimiento de este modelo, la organización buscará satisfacer las necesidades de todos los grupos de interés, incluyendo trabajadores, clientes y la sociedad en general. Para ello, deberá cumplir con los 8 principios básicos sobre los que se basa todo el sistema de gestión de calidad. Dichos principios, incluidos en la norma ISO 9001 y tal como se nos detallan en blogdecalidadiso.es [3], donde se pueden encontrar otras publicaciones acerca de certificados, calidad y aspectos de la ISO, son:

1.- Enfoque al cliente: Al fin y al cabo, las empresas y organizaciones dependen de sus clientes. Para que estos sigan confiando en ellas, deberán tener en cuenta sus necesidades, preocupaciones, preferencias a la hora de tomar las decisiones.

2.- Liderazgo: Los líderes establecen la unidad de propósito y la orientación de la organización. Deben crear y mantener un ambiente interno, en el cual el personal pueda llegar a involucrarse en el logro de los objetivos de la organización.

3.- Participación del personal: El personal, a todos los niveles, es la esencia de la organización, y su total compromiso posibilita que sus habilidades sean usadas para el beneficio de la organización.

4.- Enfoque basado en procesos: El procedimiento de alcance de un resultado resulta más eficiente si los recursos y actividades que abarca se definen como un proceso.

5.- Enfoque de sistema para la gestión: Resulta también de gran interés para mejorar la eficacia y eficiencia de la empresa u organización, la identificación, gestión y entendimiento de los procesos afines entre sí, como si de un sistema se tratase.

[1] PoC: Proof of Concept (Prueba de concepto).

6.- Mejora continua: Se debe contribuir a mejorar continuamente el desempeño de la organización en su totalidad. Esta necesidad de mejora debe ser un objetivo siempre presente en la organización.

7.- Enfoque basado en hechos para la toma de decisiones: Es interesante analizar los datos e información previa de la empresa para llevar a cabo una toma de decisiones eficaz.

8.- Relaciones mutuamente beneficiosas con el proveedor: Se debe estar en continuo contacto con los proveedores. La organización es interdependiente de estos y la creación de esta relación es beneficiosa para ambos. Con esto se contribuye al augmento de capacidad para crear valor.

Como bien se ha dicho, una mejora continua es vital para el crecimiento y desarrollo de una empresa. Mediante la implementación de la PoC, se busca potenciar este objetivo. Con la trazabilidad y optimización del proceso de aprovisionamiento del pescado, desde la lonja a la tienda o punto de venta al público, le será más fácil la gestión al departamento logístico de la empresa y se corregirán posibles fallos que puedan suponer un incremento en los costes de abastecimiento. A toda esta mejora del proceso de gestión, la PoC a implementar permitirá tener en cuenta otros parámetros de interés para la empresa, los cuales serán detallados más adelante.

Antes de entrar en detalle con la propuesta de optimización, interesará hablar acerca del proceso de abastecimiento de pescado llevado a cabo en España. Posteriormente se hablará de como el proyecto contribuirá a la mejora de dicho proceso.

1.3 Abastecimiento de pescado en España

Como punto de partida, quisiera citar a Javier Casares, miembro del Departamento de Economía y Ciencias Sociales Agrarias en la Universidad Computense de Madrid. Casares, en el libro llamado *Comercialización y distribución de productos pesqueros,* en el apartado de *Estructura de la distribución de productos pesqueros* [57], habla acerca del abastecimiento de pescado en España. Me basaré en este apartado pues, para resumir algunos conceptos comentados por Casares en dicho libro.

La distribución comercial del pescado ha experimentado un cambio drástico. Dicho cambio podría situarse en los años 80 y primera mitad de los 90. La situación actual, nos muestra la necesidad de tener en consideración una nueva interpretación del término de la distribución comercial. Cada vez más, hay que tener en cuenta los distintos cambios existentes en la producción y el consumo y en la evolución del sector distributivo.

Actualmente, los mercados tienden a orientarse por los deseos. Se ha pasado de la necesidad genérica, como puede ser la de tener una vivienda o poder alimentarse, a otra derivada, influida por la publicidad constante y las distintas marcas. Es por esto por lo que resulta cada vez más importante para las empresas, ofrecer al cliente una imagen de calidad en la que este esté dispuesto a confiar. Para esto, un buen establecimiento de comercialización alimenticia deberá disponer de los productos necesarios en todo momento y llevar un correcto control de lo que se adquiere a proveedores.

En el ámbito de los canales de comercialización del sector pesquero, existen una gran variedad de operaciones y agentes en el proceso que abarca las actividades a llevar a cabo desde el desembarque del pescado hasta que llega al consumidor final.

En cuanto a lo que se espera abastecer respecto al consumo total de pescado, el desglose es el siguiente:

- Conservas 10%
- Congelado 30%
- Fresco 60%

La demanda de pescado fresco es pues, la principal a tener en cuenta, debido a su alto consumo.

1.3.1 Comercialización de pescado fresco

El proceso de comercialización de pescado fresco sobre el año 2000 se llevaba a cabo siguiendo el esquema que se puede observar en la siguiente figura, extraída del libro citado en el apartado anterior [57].

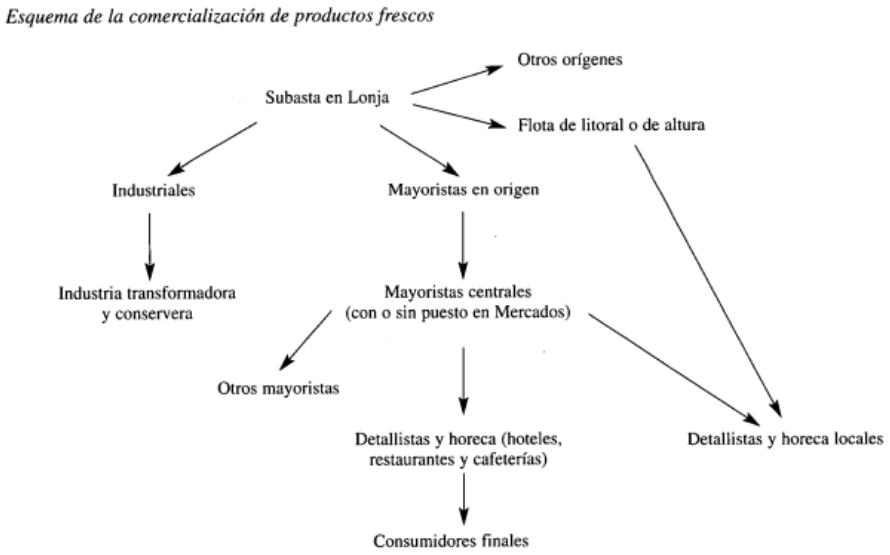

Figura 2: Esquema de comercialización del pescado fresco. Fuente:[57].

En muchos países, incluido España, la comercialización o venta de dicho pescado, se realizaba (y se sigue realizando) a través de Lonjas, el funcionamiento de las cuales se detalla a continuación.

Todo lo que se pesca llega al puerto y seguidamente pasa a la lonja. Es entonces donde se subasta, siendo esta subasta pública, a la baja y con precio libre. Al inicio de dicha subasta, el vendedor coloca una muestra del pescado en un mostrador y da un precio de salida, a partir del cual los compradores deberán pujar con precios superiores.

Actualmente, el proceso descrito ha cambiado en muchos sitios debido a la introducción de la etiqueta identificativa en la comercialización de pescado fresco. Dicha etiqueta da información acerca del pescado, incluyendo: nombre del pescado (vulgar y científico), origen, grado de calidad, peso, fecha de captura, tamaño, vendedor y precio.

Con esto se busca simplificar el proceso de subasta en las lonjas, así como una mayor estandarización de las mercancías. Además, dichas etiquetas facilitan el abastecimiento de pescado en los establecimientos de comercialización. Algunos de ellos, disponiendo de dicha información, realizan acuerdos con cofradías y pescadores alrededor de España para garantizar el abastecimiento de pescado fresco en sus tiendas.

Como se ha dicho, la demanda de pescado fresco en España asciende a un 60% del total de demanda de pescado en dicho país. Es por esto por lo que diversos establecimientos de comercialización tratan de poner sus máximos esfuerzos en hacer que dicho pescado llegue de las lonjas a sus tiendas antes de pasadas 24h.

Para poder abastecerse en tiempo récord, los establecimientos que ofertan pescado fresco a sus clientes deberán disponer de una logística de máxima eficacia. Es aquí donde entra en juego la optimización del transporte de dichas mercancías.

Para la elaboración de este apartado, al igual que en el anterior, se ha obtenido la información del libro anteriormente citado [57]. A continuación, se seguirá con el punto de motivación, donde se mostrarán cuáles fueron los motivos que hicieron elegir este tema como propuesta de TFG.

1.4 Motivación

A nivel personal, la motivación principal que ha impulsado a la elección de este tema para la elaboración del TFG ha venido dada por ser este el primer proyecto de dimensiones relativamente grandes en el cual se ha formado parte. Este ha sido el primer enfrentamiento real al mundo laboral en el papel de un informático. Además, cabe destacar que, a lo largo de la realización de este proyecto, se han ido aprendiendo nuevos lenguajes de programación, así como formas de comunicación y funcionamiento de trabajo dentro de una empresa grande como es Capgemini.

A nivel de la empresa interesada, el interés en la elaboración de este modelo y prueba de concepto viene dado por la mejora de eficacia en el departamento de logística. Con la elaboración de esta futura aplicación se reducirá de manera drástica el tiempo invertido en las tareas de cálculo de rutas de abastecimiento, así como las de gestión y seguimiento de estas y de las mercancías transportadas.

1.5 Objetivos

El objetivo principal de la siguiente propuesta de TFG consiste en detallar la elaboración de un modelo matemático y mostrar la estructura y funcionamiento de la prueba de concepto que hace uso de este. Dicha PoC busca mejorar el proceso de abastecimiento y distribución en la relación de gestión de proveedores de pescado sobre un establecimiento comercial.

A este objetivo principal se le podrían añadir los siguientes:

- Implementar un enlace entre la gestión y la producción y abastecimiento óptimos.
- Agilizar y optimizar las tareas de cálculo y planificación de rutas de abastecimiento al departamento de logística de un establecimiento de comercialización.

- Aplicar nuevas soluciones tecnológicas para la mejora de la gestión de un proceso logístico.
- Reducir el impacto de las emisiones de CO2 proponiendo una solución amigable para el medio ambiente.
- Entender el proceso de comunicación y transferencia de datos entre las distintas partes de la prueba de concepto desarrollada.

1.6 Estructura del trabajo

En esta propuesta de TFG se empezará hablando acerca de la situación actual referida a la contaminación por Co2, ámbito o problemática en el cual surge la idea de desarrollo de la prueba de concepto, junto con la necesidad de abastecimiento inmediato. Se continuará comparando aplicaciones existentes orientadas a la optimización de rutas con la PoC propuesta, remarcando los valores añadidos que aporta el proyecto a realizar.

Lo siguiente será analizar el problema que se desea cubrir con dicha PoC, describiendo el esquema a realizar y los diversos elementos a tener en consideración. Seguidamente, se presentará en que se basa el proyecto implementado para luego centrarse en su diseño. Concretamente, se definirá el problema a tener en consideración, junto con el modelo en el cual se basa el algoritmo definido para solventarlo.

Finalmente, se incluirá algún caso sencillo de prueba de la PoC desarrollada, seguido de una conclusión general (acerca de la realización del trabajo, necesidades cubiertas y relación de los conceptos desarrollados con el grado cursado) y proponiendo futuros aspectos a tener en consideración en la PoC.

1.7 Colaboraciones y agradecimientos

La elaboración de la prueba de concepto ha sido llevada a cabo por un gran conjunto de estudiantes de prácticas, cada uno perteneciente a una parte del desarrollo. De cada una de las partes, se ha recogido y detallado la información necesaria para la elaboración del trabajo.

Considerando como equipo de desarrollo, el existente desde mi incorporación a la empresa, sus miembros han sido los siguientes:

Onofre Sanmartín y Adrián Zarzoso han sido los principales encargados de desarrollar el modelo y algoritmo matemático del que se hace uso para el cálculo de rutas. Dragos Marián ha sido el principal encargado de la adecuación y diseño de la base de datos.

Juan Mezquita, todo y que ya no está implicado en el proyecto, fue el programador que empezó con la implementación del *frontend y* parte del *backend* de la aplicación en sus primeras fases. Sin él no habría sido posible avanzar en diversas situaciones ni solucionar dudas necesarias para el progreso del proyecto.

Toni González ha sido el tutor de empresa y jefe de proyecto durante mi estancia en Capgemini. Agradecerle a él su implicación y empatía durante la realización de las prácticas.

Ignacio Gil Pechuán, tutor de TFG y de prácticas, me ha ido orientando a la hora de la realización del trabajo. Quisiera agradecerle a él su disponibilidad, amabilidad y comprensión en las distintas reuniones realizadas para concretar aspectos acerca del TFG.

Mi función en el proyecto ha sido continuar con la implementación y corrección de la PoC. El trabajo realizado en la empresa ha consistido en adaptar el *frontend* a las nuevas funcionalidades y requerimientos de la aplicación, así como la reimplementación y modificación de partes del *backend* para el correcto funcionamiento con la nueva base. Además de esto, he estado en contacto con los matemáticos durante el proceso de elaboración del algoritmo basado en el modelo matemático de la PoC.

2. Estado del arte

Actualmente, vivimos en un mundo en continuo movimiento. Todo se mueve; las personas, productos, mercancías, la información y con ella datos de vital interés para negocios, establecimientos y/o empresas…

La globalización ha hecho posible que algo que nace en un lugar situado a miles de kilómetros nos llegue mucho antes de lo que lo hacía años atrás, como podemos comprobar en la difícil situación que atravesamos. Artículo interesante sobre este tema es el de Laura Martín, titulado *La logística se alía con el medio ambiente para evolucionar* y publicado en la revista digital Compromiso Empresarial [4]. Como bien dice Laura en dicho artículo, el auge del comercio online ha hecho incrementar notablemente el movimiento alrededor del mundo.

Cada vez son más las empresas internacionales que ofrecen productos a precios altamente asequibles, provenientes de países extranjeros como China o Estados Unidos. Ejemplo de estas podrían ser ebay o SHEIN, las cuales deben realizar un largo trayecto para entregar el producto al cliente.

El crecimiento e internacionalización de las empresas, ha obligado a estas a tener mucho más en cuenta el capital destinado al transporte de mercancías y todo lo que lo engloba. Es por esto por lo que cada vez se está prestando más atención a la economía del movimiento. Se pretenden optimizar los movimientos o desplazamientos, de manera que se consuman los mínimos recursos y se lleve a cabo el menor esfuerzo posible durante su realización.

En el ámbito de las empresas, según la Agencia Europea de Medio Ambiente, las operaciones de transporte y logísticas son las culpables del 25% de las emisiones de Co2 en España y del 10% a nivel mundial, cifra que incrementará debido al comentado auge de la venta online [4]. Resulta pues, de una necesidad creciente, intentar economizar al máximo el movimiento a la hora de distribuir mercancías. Es pues, de vital importancia para mantener un equilibrio entre el crecimiento económico actual y la conservación y protección de los recursos naturales y entorno en el que vivimos, llevar a cabo una distribución inteligente de estas.

Resulta imprescindible analizar la huella de carbono causada y apostar por una movilidad sostenible y las buenas prácticas.

Últimamente, la venta de vehículos eléctricos o híbridos ha augmentado notablemente. También, ha habido un augmento de desarrollo de aplicaciones concienciadas con el medio ambiente. Estas *apps* tienen como objetivo economizar el consumo de combustible, esto es, optimizar los recorridos o rutas que realizamos con nuestros automóviles a la hora de desplazarnos.

Podemos encontrar una lista de 10 de estas aplicaciones de optimización en el artículo de nombre *Diez apps para la gestión logística*, publicado en la revista el Vigía el 27 de abril del 2019 [5]. A continuación, partiendo de la información de este artículo se hablará de 2 de ellas: *WebFleet Mobile* y *SmartMonkey*.

SmartMonkey es una herramienta, el objetivo de la cual, como bien se ha introducido, es el de optimizar las rutas. Va dirigida principalmente a pymes y autónomos y está integrada en *Google Spreadsheet.* Según la definición de Wikipedia, *Google Spreadsheet* es un servicio en la red de hojas de cálculo (similar al conocido Excel), realizado en tecnología AJAX[2] [6].

Figura 3: SmartMonkey logo. Fuente: [54].

Mediante esta aplicación, podemos generar rutas óptimas de manera sencilla y rápida. Haciendo uso de esta, podremos reducir los costes operativos, podrá mejorarse la calidad en el servicio de entrega, además de aumentar la productividad debido a una correcta gestión de la cadena logística.

WebFleet Mobile es el equivalente a *Tom Tom* para móviles. Mediante dicha aplicación, podemos ver la posición actual de los vehículos disponibles y con cuales resultaría más asequible realizar un trabajo. Esta aplicación tiene en cuenta la situación del tráfico en cada momento, controlando al personal repartido y disponible para realizar cualquier pedido o tarea [5].

Figura 4: WebFleet Mobile logo. Fuente: [55].

Al final del trabajo, en el apartado de Referencias, se pueden consultar varios enlaces a videos de uso de estas dos aplicaciones [47][48].

En el ámbito de un establecimiento de comercialización alimenticia, interesa ahorrar al máximo en términos de coste de desplazamiento de mercancías. Además, cuando al hablar de mercancía, hablamos de pescado, especialmente en el caso del pescado fresco, la rapidez del proceso de transporte de las lonjas hasta la tienda es de suma importancia. Se necesita que el cliente sea capaz de adquirir el pescado en la mejor de las condiciones.

Desde Capgemini, empresa encargada del desarrollo de la PoC a detallar en este trabajo y en la cual han sido realizadas las prácticas en empresa, se quiere contribuir también a la optimización de rutas. Es por esto, que la empresa se compromete mediante el desarrollo de la PoC de optimización de redes, la cual se irá detallando a lo largo del documento, a optimizar el proceso de abastecimiento de pescado.

En el siguiente punto, se hablará con más detalle acerca de la empresa, su trayectoria y los servicios que presta, para entrar luego en detalle en el proceso de elaboración de la PoC.

2.1 La empresa consultora: Capgemini

Capgemini es una corporación multinacional francesa fundada el 1967 por Serge Kampf y con sede en París, especializada en proporcionar servicios de TI y consultoría y que cuenta con más de 200.000 empleados en más de 40 países.

[2] AJAX: Asynchronous JavaScript And XML. Técnica para el Desarrollo de aplicaciones webs interactivas [65].

La **visión** de Capgemini dice que:

'El valor de negocio de la tecnología viene desde, y a través, de las personas'

Con esto, la empresa quiere recalcar que, para ellos, el valor del negocio no se puede alcanzar únicamente a través de la tecnología. El éxito empieza en las personas, las cuales, trabajando en conjunto son capaces de desarrollar u obtener mejores soluciones y resultados. Este enfoque tecnológico centrado en las personas es lo que empuja a prosperar a la empresa.

Figura 5: Logo Capgemini. Fuente:[7].

En cuanto a la **misión**, Capgemini busca:

'Crear y entregar soluciones de negocios y tecnología que se ajusten a sus necesidades e impulsen sus resultados'

La empresa afirma que su objetivo consiste en proporcionar capacidad para responder rápidamente al cambio constante de la dinámica del mercado actual. Ayuda a quienes confían en ella a aprovechar la tecnología adecuada para ser más ágiles y competitivos.

Colaborar es fundamental para la empresa, es su forma de hacer negocios. A este enfoque lo llaman el Collaborative Business Experience®.Con esto buscan forjar relaciones más cercanas y eficaces.

2.1.1 Historia

Serge Kampf fundó Sogeti en 1967 en Grenoble. En 1975, con la adquisición de dos compañías de servicios de tecnologías de la información, CAP y *Gemini Computer Systems*, la empresa se convierte en líder en Europa con presencia en 21 países.

Figura 6: Serge Kampf en la fundación de Sogeti. Fuente:[53].

Continua la expansión y la empresa sigue creciendo, centrándose en campos que van desde soluciones de inversión de capital hasta servicios intelectuales. Es en 1989 cuando la reestructuración interna, la expansión europea y la penetración en el mercado estadounidense permiten a Capgemini convertirse en uno de los líderes mundiales en este sector.

Tan solo un año después, Capgemini empieza a desarrollar su práctica de consultoría de gestión gracias a una serie de adquisiciones, incluidas *United Research* (1990) y *Mac Group* (1991) en los Estados Unidos. La empresa también amplía las actividades en Europa a través de una serie de adquisiciones, como la de *Bossard* en Francia. La adquisición de *Ernst & Young Consulting* en 2000 marca un punto de inflexión para la empresa, fortaleciendo así la práctica de consultoría y la presencia en los Estados Unidos.

A finales de los años 90, Capgemini comienza su desarrollo en India, lo que da como resultado 85,000 empleados a fines de 2015. En 2002, Serge Kampf le entrega el papel de CEO a Paul Hermelin. En 2010, la empresa adoptó una nueva firma de marca: "Las personas importan, los resultados cuentan". Esta frase expresa la visión única de que la tecnología sea desarrollada por y para sus usuarios al servicio de los resultados comerciales. En 2012, después de 45 años, Serge

Kampf deja su cargo de presidente de Capgemini. Propone a Paul Hermelin como su sucesor y permanece activo hasta su fallecimiento, como vicepresidente de la Junta.

2.1.2 Servicios y sectores

Dentro de los servicios prestados por la compañía podemos encontrar:

Digital services

Servicios que pueden ser proporcionados a través de una infraestructura de información como puede ser Internet. Dichos servicios incluyen la entrega de información, datos o contenido digital y servicios transaccionales como formularios online y aplicaciones de beneficios. Estos se entregan a través de una amplia variedad de dispositivos, plataformas y mecanismos de entrega como páginas web, aplicaciones móviles y redes sociales.

Servicios orientados a la transformación e innovación

Con estos servicios, Capgemini busca fomentar la innovación enseñando a las empresas a aplicarla en toda su organización. Para ello ofrecen ideas y herramientas que ayuden a resolver los principales problemas empresariales. Con estos servicios, las empresas serán capaces de lograr ventaja competitiva frente a las otras dentro de lo que se conoce actualmente como la era digital.

Technology Operations

Servicios orientados a ayudar a los CIO y a los líderes de TI a reducir costes y respaldar una mayor agilidad comercial mediante la modernización de su infraestructura y aplicaciones de tecnología de la información. Dichos servicios tecnológicos engloban desde la automatización, gestión de la infraestructura, desarrollo y mantenimiento de aplicaciones hasta el *testing* y servicios dirigidos al usuario final.

Business Services

Estos servicios van destinados a ayudar a los negocios en sus objetivos de transformación. Permiten a estos optimizar sus operaciones de tecnología, así como mejorar las interacciones con sus clientes. Además, ofrecen también la posibilidad de obtener información mediante la analítica del negocio, haciendo uso de los datos generados por los clientes, así como las operaciones comerciales. También incluyen servicios relacionados con el manejo correcto de los riesgos de la empresa.

Ciberseguridad

Capgemini ayuda a las empresas y negocios a llevar a cabo su proceso de transformación en la era digital explorando las oportunidades de la era digital y protegiendo los activos del negocio. Con estos servicios, se pretende guiar a las organizaciones de manera segura a mejorar su estrategia de seguridad.

Toda la información necesaria para la elaboración de este punto ha sido obtenida de la página oficial de Capgemini, la cual se puede consultar en las referencias, al final del trabajo [7].

2.1.3 Marco de las prácticas realizadas

En las prácticas realizadas en la empresa, se formó parte del módulo CSD[3], en el cual principalmente se trabaja la implementación y prueba de aplicaciones y pruebas de concepto, entre ellas, la que se detallará en este TFG.

El equipo en que me encontraba estaba compuesto por matemáticos e informáticos, cada uno asignado a una parte o proyecto relacionado con aspectos de un establecimiento de comercialización. La labor de los primeros era la de la implementación de las distintas bases de datos del proyecto general, así como el planteamiento, definición y programación de los distintos modelos y algoritmos principales de cálculo. En cuanto al proyecto que me asignaron, el referente a la parte de optimización de redes, los dos matemáticos de mi equipo se encargaron principalmente del algoritmo de cálculo de las rutas óptimas.

Entre los otros proyectos o partes existentes, podemos encontrar la de gestión de turnos de los empleados del supermercado. Otra parte se encargaba de la gestión y control automatizado de las baldas de un establecimiento de comercialización.

La gran mayoría de los equipos de trabajo pertenecientes al módulo CSD de la empresa, hacen uso de los lenguajes de programación y gestión de bases de datos que fueron y siguen siendo usados para la elaboración de la PoC de optimización de redes.

Dichos lenguajes son principalmente HTML, TypeScript, entornos de programación como Node.js, Angular y librerías bastante utilizadas para la implementación de aplicaciones web escalables como Angular Material. Para la implementación y gestión de bases de datos se ha hecho uso de PostgreSQL. Además, para facilitar el despliegue de la parte de la prueba de concepto que se conecta a la base de datos se ha utilizado Spring Boot.

Aprovecharé por último este punto para comentar un poco mi paso por la empresa. Así pues, comentaré algunas de las tareas que me fueron mandando a lo largo de mi estancia y que me tuvieron ocupado durante mi periodo de prácticas.

2.1.4 Mi paso por Capgemini

Actualmente, ya no sigo dentro de la empresa. Mi último convenio con Capgemini fue realizando prácticas extracurriculares, durante las cuales me encargué de llevar a cabo las últimas correcciones de la prueba de concepto para poder realizar este TFG.

Empecé mi estancia en la empresa con el período de formación inicial, basado en la visualización de cursos en la plataforma de Pluralsight e implementando alguna que otra interfaz sencilla de prueba. En dicha plataforma, adquirí conocimientos básicos acerca de los lenguajes y *frameworks*[4] anteriormente nombrados, que me sirvieron a la hora de entrar en contacto con el proyecto.

Pasadas unas semanas, se me otorgó acceso al repositorio de la empresa donde se encontraban los archivos necesarios para empezar a trabajar en la implementación de la PoC. Tras la configuración correspondiente del entorno y lanzamiento de la prueba (la cual comentaré más adelante), empecé a realizar tareas relacionadas con la corrección de los errores existentes en la implementación de

[3] CSD: Custom software development.

[4] Framework: Entorno de trabajo con sus correspondientes artefactos orientado al desarrollo de software.

la interfaz de la prueba de concepto. En dicho proyecto, había cosas que no funcionaban y algunas funciones o métodos estaban a medio implementar. Mi tarea consistió pues en corregir dichas faltas y probar el correcto funcionamiento de la PoC.

El mapa de la interfaz donde se visualizaban las rutas calculadas y marcadores con los distintos proveedores no disponía de *API key*[5] y al poco tiempo, me pidieron que buscara alternativas, con lo que estuve investigando e implementando distintos mapas alternativos a la API[6] de *Google Maps* (Leaflet, OpenStreetMap…). Finalmente se optó por conseguir una nueva licencia y volví a adaptar la aplicación con dicha API. La licencia que se consiguió era la más básica, por lo que había un máximo de consultas por segundo y esto a la hora de realizar pruebas de funcionamiento de la *app* era bastante engorroso. Se optó por implementar ciertos retardos y *loaders*[7] para impedir que el usuario pudiera sobrepasar el límite de consultas de los servicios de Google.

Lo próximo fue corregir un error que se dio en la red y que provocó que el cliente local no pudiera conectarse al servidor implementado en Linux en una máquina virtual. Tuve que migrar el *backend* de la aplicación, es decir, todo el servidor en sí, a local, teniendo que corregir errores de dependencias en cuanto a Cplex, que es el software utilizado para realizar los algoritmos de optimización. Así pues, también se debieron solucionar ciertos problemas de compatibilidad entre Linux y Windows, que impedían que se ejecutase correctamente el Tomcat a la hora de lanzar el servidor.

Durante el período inicial, se modificó la base de datos del proyecto por parte de otro estudiante en prácticas del equipo, y se volvieron a implementar de mejor forma varios algoritmos, cosa que obligó a volver a implementar el *backend* de la aplicación. Es en este punto donde llevé a cabo la mayoría de las tareas en la empresa. Tuve que implementar las nuevas *queries*[8] que accedían a la nueva base de datos para obtener la información necesaria a la hora de calcular rutas y mostrar estas por pantalla junto a otras funcionalidades.

Un grupo formado por dos estudiantes de matemáticas se encargaron de realizar los distintos algoritmos dedicados a la lectura e inserción de datos en la nueva base. A su vez, estando en contacto continuo con ellos, se revisó el modelo matemático y se acabaron de solucionar todos los problemas relativos a la compatibilidad con la nueva base.

A continuación, se hablará un poco acerca de las distintas tecnologías utilizadas durante el proceso de implementación de la PoC.

2.2 Tecnologías para el desarrollo

2.2.1 Devonfw

Devon fue la tecnología utilizada para la elaboración de la versión inicial del *backend*. Como bien se puede encontrar en la página web oficial de Capgemini [63]: "Devon es el *framework* de aplicaciones Java EE estándar de Capgemini, orientado a servicios, que

Figura 7: Logo devonfw. Fuente: [52].

[5] API key: Clave necesaria para poder utilizar distintos recursos o servicios orientados al desarrollo de aplicaciones.

[6] API: *Application Programming Interface* (interfaz de programación de aplicaciones).

[7] *Loader*: Componente visual que indica que se está cargando información.

[8] Queries: consultas a una base de datos.

permite mejorar la productividad, calidad, reutilización y satisfacción del usuario en los proyectos de desarrollo a medida Java." (Capgemini, 2020).

Según la documentación de devonfw (*Devon Framework*) disponible en Github [8] y en su página web oficial [52], este surge de la idea de mejora de industrialización del módulo CSD de Capgemini. Con dicho *framework* se busca disponer de una plataforma estandarizada para el desarrollo software dentro de la empresa. Dicha plataforma es utilizada en aquellos proyectos donde el cliente no obliga a la utilización de un determinado *framework*, software o tecnología. Con esto se busca poder ofrecer una solución factible, basandose en la experiencia de grupo de los programadores de la empresa.

Mediante la utilización de *Devon Framework* se pretende facilitar el proceso de desarrollo de una aplicación, estandarizando los procesos y aumentando la productividad. Gracias a la utilización de este *framework*, el proceso de construcción del plan de arquitectura de aplicaciones JavaScript o Java, resulta mucho menos tedioso. Además, Devon proporciona un conjunto de herramientas para conseguir un entorno de desarrollo listo para usar.

Una de las herramientas más remarcables incluidas en el contexto de devonfw es CobiGen. Mediante el uso de esta herramienta, los programadores serán capaces de 'autogenerar' toda la estructura y componentes de la aplicación. Esto ayudará a ahorrar tiempo malgastado en tareas repetitivas de definición de componentes. Como se puede ver en la siguiente figura, Cobigen permite seleccionar las distintas organizaciones de paquetes a generar de forma automática.

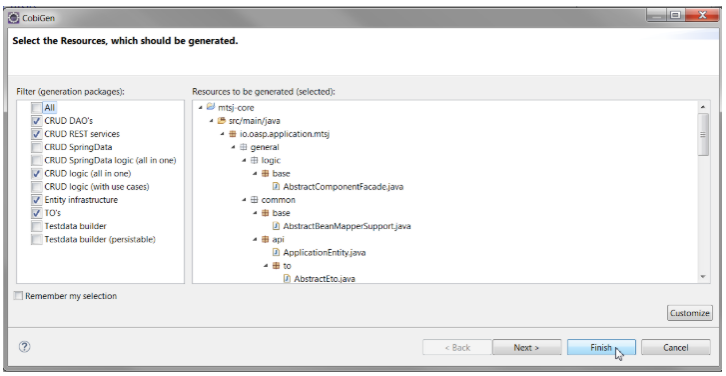

Figura 8: Captura de utilización de Cobigen. Fuente:[8].

Para poder trabajar con Devon, se deberá tener configurado correctamente Node.js en nuestro equipo. Así pues, también se deberá tener preinstalado Maven para la gestión de dependencias y Tomcat como servidor web listo para probar nuestros 'artefactos'. De todo este proceso de configuración e instalación se encarga el ejecutable contenido en la distribución o carpeta de Devon. Todo el proceso de descarga, instalación y preparación del entorno se encuentra disponible en la documentación de devonfw, en Github [8] ampliada en su página oficial [52].

A continuación, una vez introducido el *framework* de devonfw, se describirán brevemente las principales características de las principales tecnologías y lenguajes utilizados en la elaboración de la PoC. Dichas tecnologías son TypeScript, Angular, PostgreSQL, Maven, Spring Boot y Tomcat.

2.2.2 TypeScript

Partiendo de la definición que podemos encontrar en Wikipedia, podemos decir que TypeScript "es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft" (Wikipedia, 2020) [9]. Como bien nos dice Miguel Ángel Gómez, especializado en tecnologías y artesanía del software y autor de libros acerca de la limpieza del código y *testing* orientado a JavaScript [10], TypeScript es un superconjunto de JavaScript, esto quiere decir que amplía este lenguaje con una sintaxis nueva, añadiendo cosas como genéricos, decoradores, tipado estático opcional y elementos de programación orientada a objetos.

Con este lenguaje de programación podemos compilar código en JavaScript, ejecutado en cualquier navegador, en Node.js o en cualquier motor de JavaScript que admita ECMAScript[9] 3 o alguno más reciente.

TypeScript intenta resolver la mayoría de los problemas que puedan surgir al programar con JavaScript, mejorando la productividad de los desarrolladores. Permite la utilización de técnicas como la encapsulación para generar código más escalable y mantenible que con JavaScript básico o tradicional, así como el tipado estático opcional [10].

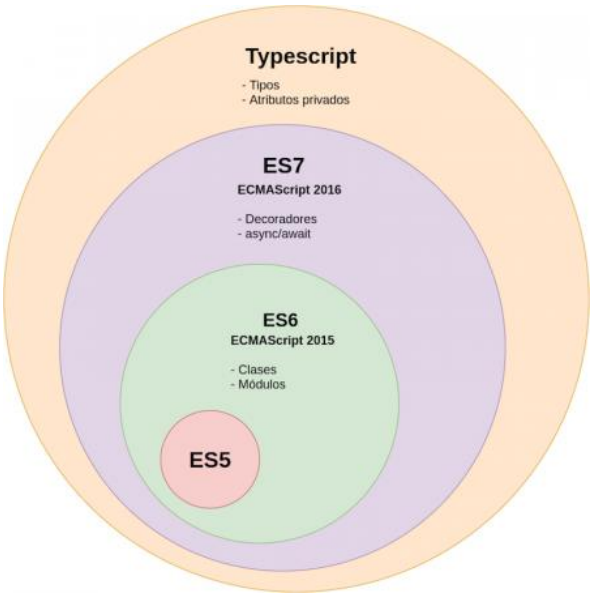

Figura 9: Nuevos aspectos que añade Typescript. Fuente: [10].

2.2.3 Node.js

Como bien podemos encontrar en la tan conocida Wikipedia, Node.js fue creado por Ryan Dahl en 2009 teniendo como propósito ser útil en la creación de programas de red altamente escalables, incluyendo los servidores web. Es una librería y entorno de programación de E/S (entrada y salida) dirigida por eventos, en tiempo de ejecución multiplataforma. Dicha E/S es por tanto asíncrona y se

Figura 10: Logo Node.js. Fuente: [11].

9 ECMAScript: "ECMAScript es una especificación de lenguaje de programación publicada por ECMA International." (Wikipedia, 2020) [58].

ejecuta sobre el intérprete o motor de JavaScript creado por Google V8, para uso de su navegador Chrome [11].

Tras consultar información sobre Node.js en la web de Consultoría Digital y Márquetin apasionados.es [12], puede decirse que Node.js es un entorno JavaScript del lado del servidor, basado en eventos y muy utilizado por los programadores. Gracias al motor V8, Node proporciona un entorno de ejecución que compila y ejecuta JavaScript con un alto rendimiento, es decir a altas velocidades.

En la prueba de concepto que se planteará, se ha utilizado Node.js para la gestión de las dependencias tanto en la parte del *frontend* como el *backend*, ya que, todo y ser un entorno del lado del servidor, no se reduce solo a ello. Así también ha sido utilizado su gestor de paquetes npm para la instalación de las distintas librerías o módulos necesarios.

2.2.4 Angular

Tal como se puede encontrar en Wikipedia, Angular es un *framework* mantenido por Google y utilizado en aplicaciones web. Dicho *framework* está desarrollado en TypeScript y es *open source*, es decir, de código abierto. Angular es usado para la programación *frontend* y hace uso de Node.js para la gestión de paquetes y dependencias.

Figura 11. Logo Angular. Fuente: [13].

Su principal objetivo es el de aumentar las aplicaciones basadas en navegador con capacidad de Modelo Vista Controlador, con el propósito de que el desarrollo y las pruebas sean más fáciles. [13]

En el punto posterior de Diseño de la solución, entraré con más detalle a hablar de algunas de las ventajas de programar con Angular.

A continuación, basándome en el artículo de título *Angular: Mucho más que un framework*, publicado por Jorge Cano en la revista #56 de Software Guru [14] se describirá lo necesario para poder construir nuestras aplicaciones con Angular. Las distintas partes o componentes en los que se basa una aplicación Angular y que, por tanto, deberemos crear son:

- Los *templates* o hojas HTML: Estos archivos contendrán las etiquetas especiales de Angular y se relacionarán con los distintos compontes mediante *bindings*[10] de propiedades.

- Los componentes de clase: Estos gestionan la funcionalidad de los *templates* anteriores. En ellos se definen los distintos atributos, constructores y/o funciones relativos al componente.

- Los servicios: Estos encapsularán la lógica de la aplicación. Poniendo como ejemplo nuestra prueba de concepto, en cada servicio relativo a las tiendas, proveedores, almacenes o productos, se gestionarán las peticiones http al servidor. Todo esto será detallado más adelante.

[10] Binding: Enlace entre uno o varios componentes con alguna de sus propiedades o propiedades de otro componente.

- Los módulos: que organizan los distintos componentes y servicios. Póngase como ejemplo una aplicación de alquiler de vehículos que considera coches, motos y bicicletas. Se podría tener un módulo que encapsulara la lógica y los componentes relativos a cada tipo de móvil, haciendo así más entendible y organizada la aplicación.

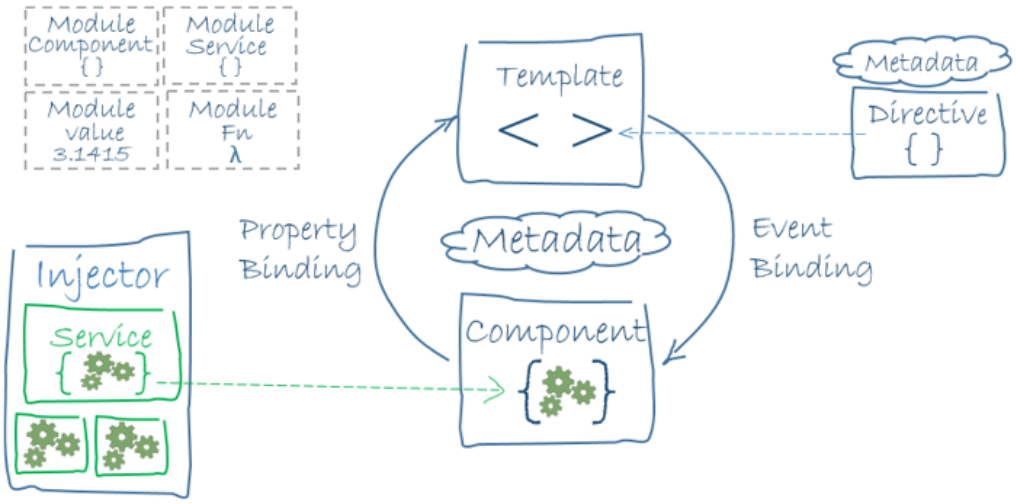

Figura 12: Esquema de funcionamiento de Angular. Fuente: [14].

En la figura anterior, obtenida del artículo de Jorge Cano citado anteriormente [14], se puede ver a grandes rasgos el funcionamiento de Angular. Los componentes son los que definen el contenido a mostrar en pantalla y se define en ellos también como actuar frente a las interacciones de los usuarios. La lógica detrás de los componentes suele estar definida en los servicios, que como bien se ha dicho, se encargarán de llevar a cabo las distintas actuaciones. Seguidamente, mediante las distintas propiedades de los componentes y *bindings*, se mostrarán los cambios en pantalla, actualizándose para aquellos componentes que en la definición del *template*, hagan uso de alguna propiedad que haya cambiado, debido a la ejecución del método o servicio, como respuesta a la interacción del usuario.

Angular, así como su herramienta de línea de comandos Angular CLI y las aplicaciones basadas en este *framework*, necesitarán de funcionalidades y librerías adicionales. Dichas librerías son accesibles a través de paquetes npm, siendo npm el manejador de paquetes de Node.js (Node.js *package manager*), como bien se ha dicho anteriormente.

2.2.5 Angular Material

Una de las principales librerías utilizadas en la parte del cliente o *frontend* de la PoC ha sido Angular Material. Tras consultar la publicación de José Carlos Fatjó en la web de Tribalyte Technologies, compañía orientada a la mejora y desarrollo de plataformas tecnológicas [15], y la web oficial de Angular Material [16], puede decirse que se trata de un módulo rápido, consistente y versátil construido para Angular y por los mismos que desarrollaron el *framework*.

Con Angular Material se pueden implementar componentes de Angular con un diseño atractivo y minimalista basado en *Material Design*. Además, toda la documentación acerca de los distintos componentes y ejemplos de *templates* pueden encontrarse en la web oficial de Material [16].

2.2.6 Maven

En la página oficial de Maven encontramos la siguiente definición (traducida al español): "Apache Maven es una herramienta de gestión y comprensión de proyectos de software. Basado en el concepto de un modelo de objeto de proyecto (POM), Maven puede administrar la construcción, los informes y la documentación de un proyecto a partir de una información central." (Apache Maven Project, 2020) [17]

Básicamente, Maven se encarga de gestionar de manera automática todas las dependencias definidas en el fichero POM, necesarias para nuestro proyecto. En la PoC a implementar, se hará uso de Maven en la parte del servidor o *backend* para gestionar todas las dependencias con librerías y módulos necesarios.

2.2.7 Spring Boot y Apache Tomcat

Para la elaboración de este punto me he valido de información detallada por Pablo Miranda [18] y Luís Alberto [19] en sus respectivas publicaciones acerca de Sping Boot en la red.

Partiendo del *framework* de nombre Spring, dedicado al desarrollo de aplicaciones Java, Spring Boot lo amplía, pudiéndose considerar una extensión de este. Con Spring Boot, el programador se ahorra todo el trabajo relacionado con la puesta en marcha o levantamiento de una aplicación Spring.

A su vez, Spring Boot contiene un Tomcat embebido que se encarga de 'simular' un servidor. Concretamente, Apache Tomcat es un software desarrollado con Java, lo cual le permite funcionar en cualquier sistema operativo con su correspondiente máquina virtual Java. Este software, sirve como un servidor web, el cual tiene soporte de *servlets*[11] y *Java Server Pages*[12]. Para ampliar información acerca de Tomcat, se pueden consultar las referencias al final del trabajo [20][21].

Ambas tecnologías han sido usadas en la parte correspondiente al *backend* de la PoC. Con esto, se pretende ahorrar tareas tediosas de preparación y levantamiento de dicha parte de la prueba de concepto.

2.2.8 PostgreSQL

Como bien se puede entender al leer la entrada correspondiente a PostgreSQL en Wikipedia [22], este se trata de un gestor de bases de datos relacionales orientado a objetos. Dicho gestor, ha sido desarrollado por una comunidad de desarrolladores de manera altruista, es por esto por lo que es gratuito. Además, se trata de un software *open source*, libre y con un elevado número de opciones avanzadas.

La denominada pgAdmin es la herramienta oficial utilizada para la administración de bases de datos en PostgreSQL. Con ella podemos realizar desde consultas a nuestra base de datos hasta desarrollar toda nuestra base de datos de manera fácil e intuitiva hacienda uso de la interfaz gráfica. Se trata de una aplicación software bastante similar al gestor de bases de datos 'Oracle' lanzada en la nube y de funcionamiento, a primera vista, más intuitivo que esta última.

[11] Servlet: Clase java que amplía las capacidades de un servidor [59].

[12] Java Server Pages (JSP): Tecnología utilizada en el ámbito del desarrollo de software utilizada para crear páginas web activas que hacen uso de XML y HTML [60].

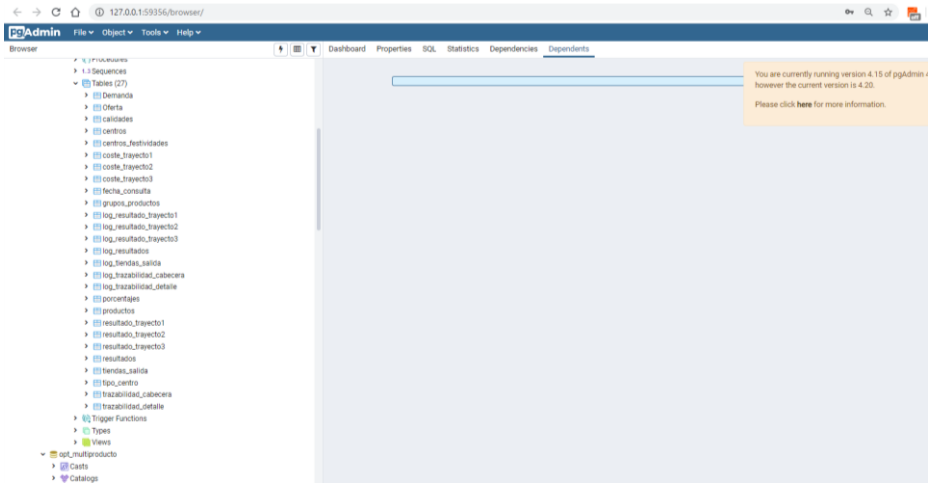

Figura 13: Captura de la interfaz del gestor PostgreSQL. Se muestran en ella las distintas tablas existentes en la base de datos de la PoC. Fuente: BD PoC Optimización de Redes (Capgemini).

Algunas de las principales características de PostgreSQL que se pueden destacar, como bien se puede encontrar en Platzi [23] son:

- Su alta concurrencia. PostgreSQL permite que mientras un usuario o proceso está escribiendo en una tabla, otros accedan a la misma tabla sin necesidad de bloqueos, obteniendo cada usuario una visión consistente de la base.
- Amplia variedad de tipos nativos. Provee soporte para:
    - *Arrays*.
    - Números de precisión arbitraria.
    - Texto de extensión o largo ilimitado.
    - Bloques de direcciones estilo CIDR
    - Figuras geométricas.
    - DireccionesIPV4 y IPV6.
    - Direcciones MAC.

Como bien se ha dicho en apartados anteriores, PostgreSQL y pgAdmin, han sido unas de las principales herramientas utilizadas en el desarrollo de la PoC. Estas han jugado un papel fundamental en el proceso de creación de tablas y realización de pruebas con la base de datos.

2.3 Crítica al estado del arte

En el apartado referente al estado del arte, se ha hablado del valor que intentaban aportar algunas de las *apps* existentes hoy en día, en cuanto a la optimización de rutas. Estas aplicaciones buscan reducir principalmente tiempos de desplazamiento y combustible, minimizando así el impacto de CO_2 y aprovechando al máximo el tiempo disponible para realizar una determinada tarea de desplazamiento.

Desde el punto de vista de un establecimiento de comercialización, interesa conseguir esto también. No obstante, se necesita considerar otros parámetros a la hora de elegir un u otro

proveedor para abastecerse. Es por esto, que la PoC que se detallará en este trabajo de fin de grado, considerará otras variables de interés para la empresa.

No solo se tratará de optimizar en cuanto al consumo de combustible, reduciendo el impacto medioambiental causado por el transporte de mercancías. Además de esto, se tendrán en cuenta otros aspectos de interés para el supermercado en cuestión, como pueden ser el precio de compra al proveedor, la calidad del pescado a adquirir y la previsión de venta que se espera tener de cada pescado en cada una de las tiendas. Se podrá considerar también que la distribución de los pedidos sea lo más homogénea posible y que se tenga en cuenta la denominación de origen de los productos de cada región.

3. Identificación de la necesidad

La propuesta a desarrollar tratará de cubrir todas las necesidades de abastecimiento óptimo de los establecimientos de comercialización nombradas en el capítulo anterior.

Utilizando como punto de partida un modelo matemático y un algoritmo de optimización implementado haciendo uso de la librería CPLEX, se desarrollará una prueba de concepto.

CPLEX no es más que, como bien lo describen sus desarrolladores en la web oficial de IBM, "un solucionador de programación matemática de alto rendimiento para programación lineal, programación de enteros combinada y programación cuadrática" (IBM, 2020) [24]. Además de esta librería, se hará uso de las tecnologías también descritas anteriormente.

Dicho proyecto, además de cumplir con los objetivos de una aplicación como *SmartMonkey*, permitirá al usuario de la PoC, es decir, al personal logístico de la empresa, tomar la decisión correcta en la situación indicada y proporcionar el abastecimiento de pescado al supermercado. Todo esto será llevado a cabo teniendo en consideración la mejor combinación de factores de decisión.

Cabe destacar, que el enfoque que se le dará al trabajo no es el de desarrollo de una aplicación software, sino, el de especificar y detallar la elaboración del modelo matemático en el que se basa el algoritmo de la PoC. Se presentará también la PoC desarrollada, explicando (sin entrar mucho en detalle) sus diversas partes y la estructura de la base de datos en la cual se basa.

3.1 Identificación de la lógica subyacente: propuesta de algoritmo base

Como bien se ha dicho, el objetivo principal de este TFG consiste en la elaboración de una prueba de concepto que permita a un supermercado o establecimiento comercial ser más eficiente a la hora de transportar sus mercancías provenientes de lonjas o sus proveedores de pescado.

Para dar solución a dicho problema, se analizarán cuáles son los requisitos que deberá satisfacer la PoC. Estos requisitos describirán las necesidades que se esperan satisfacer al elaborar la prueba de concepto.

La visión de cumplimiento de los requisitos por parte de la PoC es de tipo incremental, esto es, se irán añadiendo distintos requerimientos una vez se tenga una versión funcional del proyecto que cumpla con los requisitos pertenecientes.

Para poder entender mejor los requerimientos que se desean cumplir una vez implementada la prueba de concepto, se dará una idea general acerca del algoritmo, sobre el cual se hablará con más profundidad más adelante.

3.1.1 Algoritmo base

Para obtener una solución del problema se creará un primer algoritmo en el que se tendrá en cuenta la oferta y demanda de un solo producto, el cual deberá pasar desde los nodos origen (proveedores) a un primer almacén para posteriormente ser transportado a otro con el fin de que a través de éste se proceda al reparto entre los distintos nodos destino (tiendas). Desde cada proveedor se puede suministrar a uno o más almacenes o incluso, si se diera el caso, se podría prescindir del uso de alguno de ellos y dichos almacenes tan sólo repartirán a ciertas tiendas en concreto.

El problema debe contener la cantidad de mercancía suministrada por los proveedores y la cantidad demandada de cada tienda a la que se debe distribuir. El coste de cada arco se calculará como la distancia entre ambos nodos del arco, por lo que el coste total será la suma de las distancias de cada par de nodos origen y destino por los que se realizan los distintos caminos. El coste de cada arco vendrá dado en una tabla leída de la base de datos.

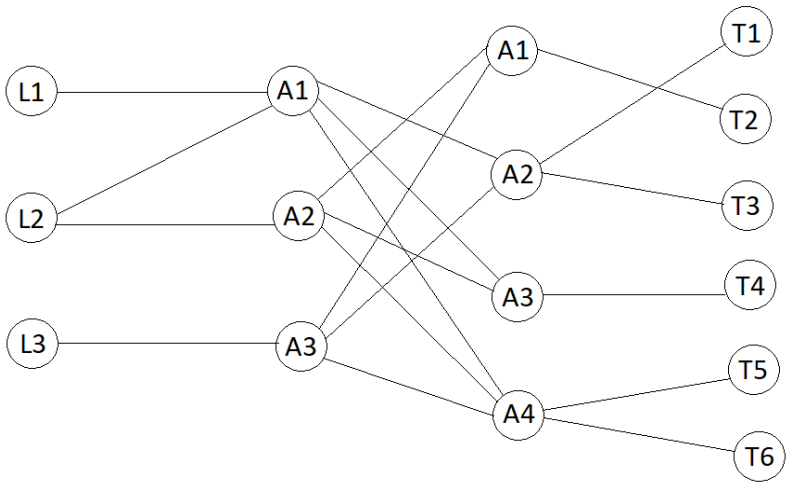

Figura 14: Esquema ejemplo de los nodos existentes en el problema a optimizar. Fuente: Documentación de la PoC (Capgemini).

3.1.2 Identificación del modelo: análisis de requerimientos:

Tras el modelo base propuesto en el punto anterior, se ampliarán los siguientes aspectos:

1. Se tendrá en cuenta que la cantidad de mercancía adquirida puede no ser fija, es decir, se trabajará con un rango de cantidades máximo y mínimo.

2. Se priorizarán los proveedores con los precios del producto más bajos. Además, se dará prioridad a aquellos donde haya mayor cantidad de producto comprado con el objetivo de ahorrar costes en el transporte.

3. Se priorizará la calidad del producto de cada proveedor mediante un parámetro estableciendo si la calidad es alta, media o baja.

4. En caso de que no sea posible abastecer las cantidades requeridas por todas las tiendas, se repartirá mayor cantidad a las que tengan los precios más altos del producto.

5. Se priorizará por las previsiones de venta en tienda más altas, es decir, en caso de que no sea posible abastecer las cantidades requeridas por todas las tiendas se repartirá mayor cantidad a las que tengan mayor previsión.

6. La distribución de los pedidos en tienda será lo más homogénea posible en caso de que no sea posible abastecer las cantidades requeridas por todas las tiendas.

Una vez finalizadas estas casuísticas, se procederá a la implementación del modelo con una red completa en la que formarán parte multitud de productos. La PoC pasa ahora entonces a ser de optimización 'multi-producto' y añadirá los siguientes requerimientos:

7. Existirá la posibilidad de obtener productos sustitutos en el caso de que sea imposible comprar el producto deseado, habiendo una penalización en la cantidad de compra.

8. Se tratará de agrupar los productos comprados en el mínimo número de proveedores, lo que ahorrará costes de transporte.

9. Se tendrán en cuenta los días en que proveedores, almacenes o tiendas estén cerrados.

Tras cumplir los requisitos descritos anteriormente, la PoC dará una solución completa al problema identificado anteriormente. A continuación, en el siguiente punto, se detallarán la serie de actividades que serán necesarias para llevar a cabo la realización del proyecto, de forma organizada y correcta.

3.2 Plan de trabajo

En este apartado, se mostrará un sencillo diagrama de Gantt, el cual reflejará el plan de trabajo del proyecto. A su vez, dicho diagrama englobará en él las actividades más importantes o considerables, llevadas a cabo durante el desarrollo de este.

Como paso previo a la realización del diagrama, se debe considerar cual es la 'estrategia' o metodología seguida a la hora de la implementación de la PoC. Para ello, podrá considerarse alguna de las metodologías de desarrollo software, las cuales, al fin y al cabo, pueden aplicarse a la hora de la realización de la PoC así como al modelo y algoritmo matemático.

Se sabe que el proyecto cuenta con un conjunto de requisitos a cumplimentar de manera incremental. Esto es, deben ir añadiéndose nuevos aspectos a tener en cuenta a lo largo del tiempo. Debido a esto, se deberán realizar varios evolutivos de la PoC. Teniendo pues, todo esto en consideración y tras considerar los conceptos vistos en la asignatura de ISW y consultar el trabajo sobre metodologías de desarrollo publicado en la web por Gastón Nicolás Jara Rodríguez y Débora Mattos, titulado *Metodologías de Desarrollo de Software, Desarrollo Evolutivo: Metodología Incremental* [25], se llega a una conclusión bastante clara. Resultará conveniente considerar una metodología incremental, una vez asignadas las tareas y el puesto de trabajo a cada uno de los matemáticos e informáticos implicados en el proyecto.

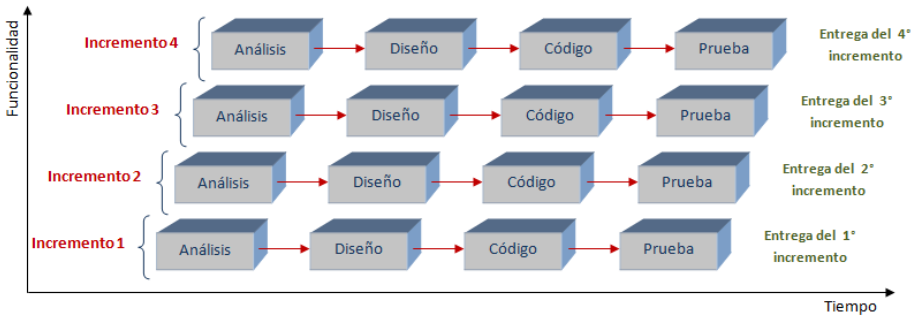

Figura 15: Modelo incremental. Fuente:[25].

Ahora pues, el diagrama de Gantt podría dividirse en dos diagramas distintos. En uno de ellos se representarían aquellas tareas previas a la puesta en marcha de los trabajadores implicados en el proyecto, correspondientes a las fases de análisis y planificación del proyecto. En el segundo, se encontrarían las referentes a la ejecución, seguimiento y cierre de cada una de las versiones del proyecto. Se muestran en la siguiente figura las distintas tareas junto con sus inicios, fines y duraciones, medidas en semanas.

Actividades de análisis y planificación	Inicio	Duración	Fin
Analisis de riesgos	1	1	2
Analisis de viabilidad	2	1	3
Análisis de alcanze	1	3	4
Definición del plan de proyecto	4	2	6
Negociación del contrato	6	2	8
Establecimiento entorno de trabajo	8	1	9
Formación trabajadores	9	3	105
Asignación de tareas	9	1	10

Tabla 1: Actividades de análisis y planificación. Fuente: elaboración propia.

Se puede observar que la tarea de formación sigue llevándose a cabo actualmente. Como en todos los proyectos de este tipo, la formación nunca deja de estar presente, ya que, más aún para un informático, la formación es una actividad necesaria en este entorno continuamente cambiante.

Actividades de ejecución, seguimiento y cierre	Inicio	Duración	Fin
Definición o modificación de la base de datos	10	5	15
Definición o modificación del modelo mat.	10	6	16
Implementación y/o modificación del back	16	6	22
Implementación y/o modificación del front	16	6	22
Pruebas y correcciones	22	4	26
Gestión de incidencias	23	3	26
Generación de documentación e informes	25	3	28
Backup del proyecto	26	1	27
Análisis de resultados	28	1	29
Definición/Análisis de nuevos requerimientos	29	1	30

Tabla 2: Actividades de ejecución, seguimiento y cierre. Fuente: elaboración propia.

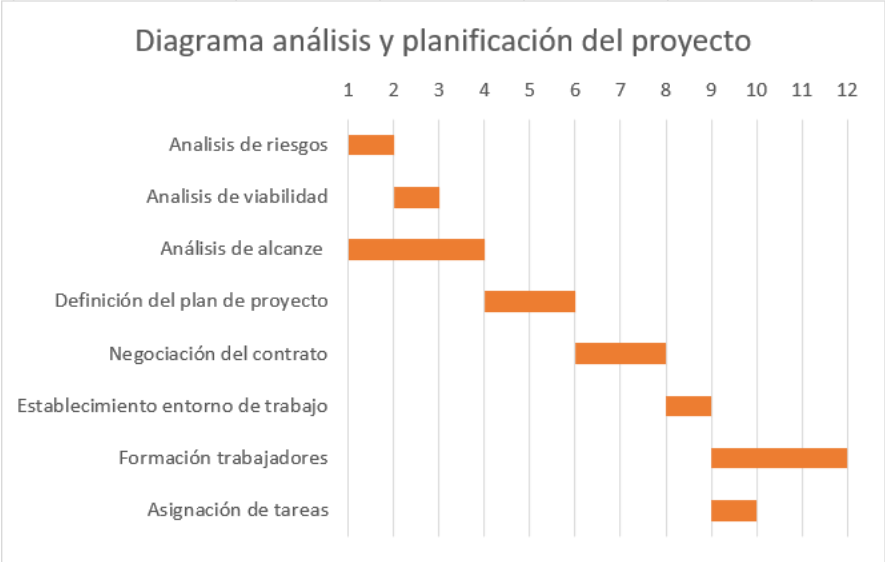

Figura 16: Diagrama análisis y planificación del proyecto. Fuente: elaboración propia.

Se puede observar en el diagrama anterior, que la tarea de formación empieza junto con la de asignación de tareas. Esto es debido a que los trabajadores empiezan a formarse al mismo tiempo que se les asignan las primeras tareas del proyecto. Dicha formación, todo y que en el diagrama finalice en la semana 12, seguirá presente a lo largo del desarrollo del proyecto.

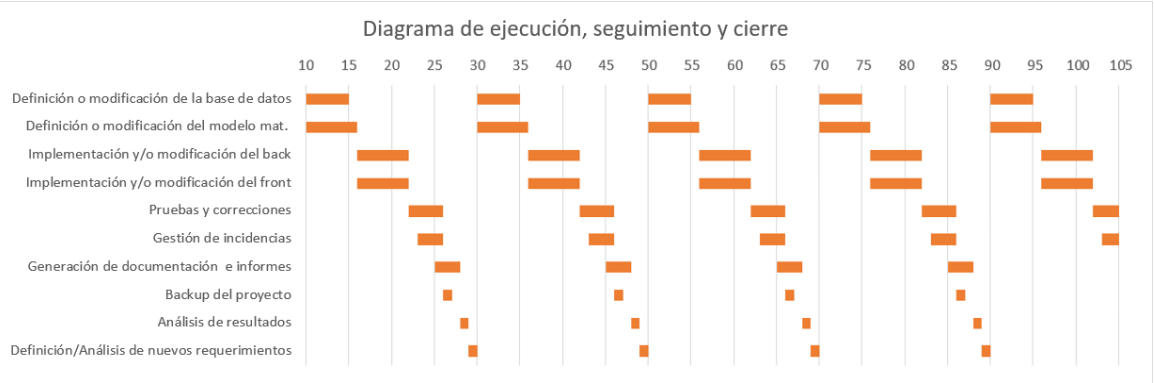

Figura 17: Diagrama ejecución, seguimiento y cierre del proyecto. Fuente: elaboración propia.

Como puede observarse en la figura anterior, el diagrama muestra la continua repetición de 10 tareas. Esto es debido, como se ha dicho, a que se sigue una metodología de desarrollo incremental del proyecto. Cada vez que se realiza la tarea de *backup* del proyecto, se sube una versión funcional del proyecto al repositorio. Además de esto, se genera documentación referente al lanzamiento, modificación de la base, así como del modelo matemático.

Una vez se dispone de una versión funcional, se analizan los resultados del proyecto y se definen los nuevos objetivos o mejoras. Seguidamente se procederá a la modificación de las distintas partes (base de datos, modelo matemático, *frontend* y *backend* de la aplicación).

3.3 Viabilidad económica: Presupuesto

El objetivo de este apartado es elaborar un presupuesto estimado del proyecto. Para ello se tendrá en cuenta tanto el software como el hardware más importante utilizado. Además, se contabilizará también la mano de obra de los implicados en el proyecto.

Se empezará a analizar el coste del personal. Cabe decir, que dicho proyecto, se trata de un proyecto o propuesta novedosa de optimización que ha surgido dentro del ámbito de la beca de formación destinada a la empresa por la incorporación de personal de prácticas en ella. Es por esto por lo que lleva en desarrollo un período considerable de tiempo de 2 años, aproximadamente. Si a este período de tiempo, le descontamos el tiempo en el cual no se ha estado trabajando en el proyecto (dicho de otra forma, si se considera, que se trabajan de 5 a 6 horas al día, 5 días a la semana durante estos 2 años), obtenemos un tiempo aproximado total de trabajo de 2625 horas, lo cual son aproximadamente 4 meses.

Como bien se dijo, el proyecto de optimización de redes forma parte de un proyecto global más grande, destinado a la optimización general de las tareas en un establecimiento de comercialización. En esta aproximación de presupuesto se tendrá en consideración el tiempo dedicado únicamente al proyecto de redes en el cual se elaboró el modelo y la PoC. En la siguiente tabla se observa el personal dedicado al proyecto junto con el salario recibido.

Puesto de trabajo	Cantidad	Salario(mensual)
Matemático	2	450
Informático	2	450

Capital total invertido: 43200€

Para contabilizar el capital destinado a los trabajadores del proyecto se ha tenido en cuenta el salario pagado a estos durante los 2 años que lleva el proyecto. El cálculo total es aproximado, ya que no se están considerando festivos ni períodos de vacaciones.

En cuanto al coste del software utilizado y las distintas licencias que se han necesitado para la elaboración de la solución o prueba de concepto se incluye lo resumido en la siguiente tabla:

Nombre del software o licencia	Coste (en euros)
pgAdmin	0
Microsoft Excel	145 (coste conjunto anual)
Microsoft Word	
Microsoft Teams	
Skype for Business	5 (coste mensual por usuario)
PostgreSQL	0
Librería CPLEX	0
Visual Studio Code	0
TortoiseSVN	0
Google Maps API key	(Precios desglosados en las figuras)
Eclipse	0

Capital total invertido: 890€

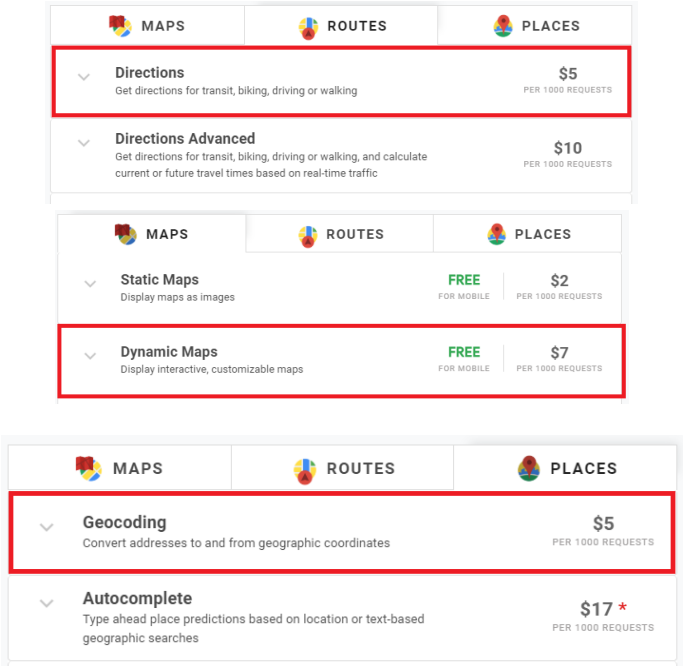

Figura 18: Composición de capturas referentes a los precios de los distintos servicios de Google. Precios consultados el 20/03/20. Fuente:[26].

Se pueden ver, en las figuras anteriores, los costes referentes a los servicios utilizados de Google. Dichos costes ascienden a 5$ por cada 1000 consultas al servicio *Directions*, el cual permite trazar rutas entre distintos puntos. En el caso de los mapas dinámicos, como es el caso de nuestro proyecto, se deberán pagar 7$ cada 1000 consultas. Así pues, se pagarán 5$ por cada consulta de Geolocalización del servicio *Places*. No obstante, Google proporciona 200$ al mes a aquellos que dispongan de una de sus *API keys*, los cuales se pueden utilizar con cualquiera de sus servicios. Por lo tanto, podemos considerar el coste de los servicios de Google como nulo, ya que no se llegarán a sobrepasar el numero necesario de consultas para que se sobrepasen los 200$ que se otorgan mensualmente.

En cuanto a Skype, el precio total aproximado será de 25€ mensuales, al considerar los 4 trabajadores del proyecto, más el jefe de este. Esto dará un total de 600€ destinados a Skype en los 2 años de vida del proyecto. A esto se le sumará la licencia de Microsoft durante estos 2 años, con un coste de 290€. Obtenemos pues un capital total destinado a software que asciende a 890€.

También, se tendrá en cuenta el equipo utilizado por el personal de prácticas implicado en la empresa. Para ello se hará una estimación de precio del hardware necesario para cada uno de los trabajadores del proyecto.

Dispositivo	Coste aproximado (euros)
Teclado	700 (por trabajador)
Portátil	
Auriculares usb	
Mouse	

Teniendo en consideración todos los recursos descritos y realizando la suma de los distintos totales se obtiene un capital total invertido en el proyecto de 44790€. Como bien se ha dicho, se trata de una cifra estimada, pero que da una idea acerca del coste de realización del proyecto.

Todos los precios han sido consultados en las correspondientes páginas web oficiales de los softwares, así como en lizengo.com. Dicha web, dedicada a la venta de software, ofrece el pack *Microsoft 365 para Empresa Estándar* [27] que incluye algunas de las herramientas que han sido necesarias a la hora de la elaboración del proyecto. Aun así, como bien se ha dicho, son precios aproximados, que, aun no siendo exactos, dan una aproximación útil para la realización del presupuesto. En el caso de Skype, sigue usándose la versión *for Business* que ha sido ahora sustituida por *Microsoft Teams*. El precio de dicha versión se ha consultado en getapp.com [28], web dedicada a la reseña, evaluación y comparación de software dedicado a la empresa.

En los siguientes puntos, se hablará de cómo se pretende cumplir con los requerimientos que esperan satisfacerse con la PoC. Se empezará por detallar el diseño necesario para llevar a cabo la solución propuesta.

4. Diseño de la solución

Lo primero pues, después de haber concretado los requisitos del proyecto a desarrollar, es describir el esquema o diagrama conceptual de la base de datos, en él se mostrarán las entidades necesarias para poder llevar a cabo el funcionamiento de la PoC.

4.1 La base de datos: punto de partida y rediseño

En la versión del proyecto existente en el período de incorporación a la empresa, el esquema de la base de datos en el que se basaba dicha primera versión carecía por completo de sentido. La base de datos no presentaba más que una relación entre dos tablas y la información estaba mal estructurada. En la siguiente figura puede observarse como existían tablas cuya clave primaria era el conjunto total de sus atributos. En conclusión, el esquema no era correcto.

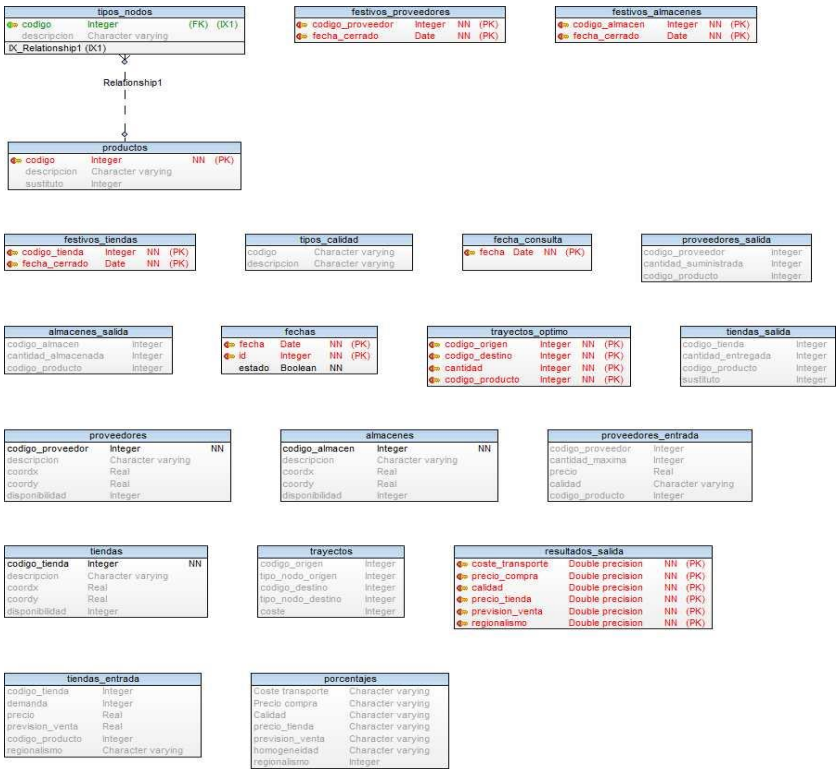

Figura 19: Esquema antigua base. Fuente: documentación Capgemini.

Estaba entonces finalizándose la estructura de la nueva base con la que trabajar. Esta nueva base, organizaba mejor la información y representaba las diversas relaciones existentes entre las tablas. Se puede ver un esquema de dicha base en la siguiente figura.

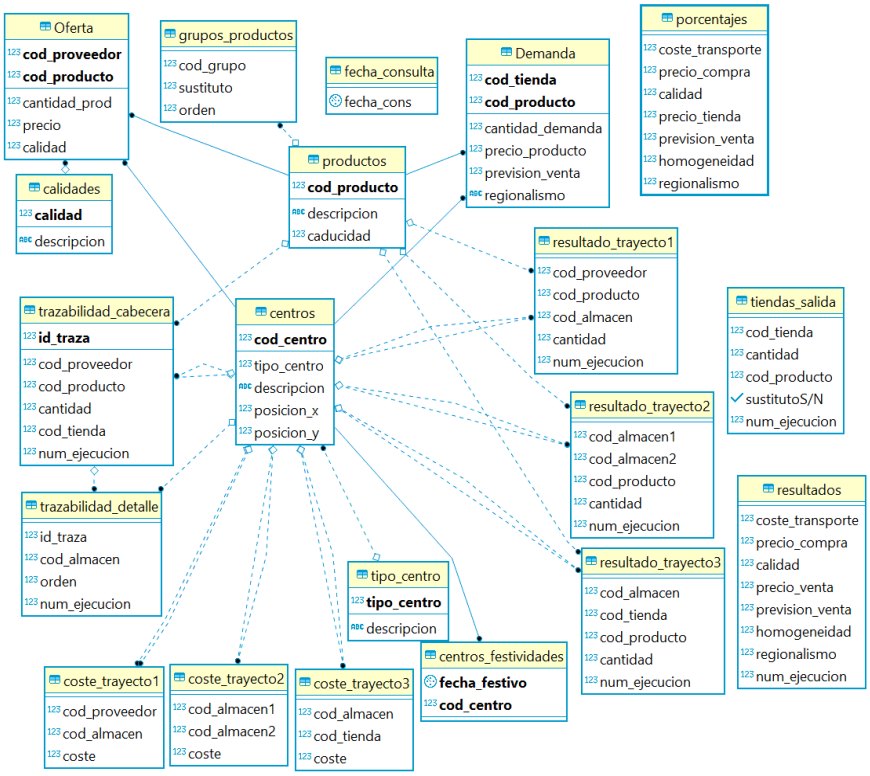

Figura 20: Esquema ER de la base actual. Fuente: documentación Capgemini.

Viendo la figura anterior, se pueden identificar tablas sin claves primarias. Esto podría considerarse un fallo en el diseño del esquema, el cual, podría haberse evitado habiendo asignado dicha tarea a un informático en vez de a un matemático. Al tratarse de un proyecto de grandes dimensiones en el que ninguno de los trabajadores son expertos y que continuamente iban surgiendo nuevas tareas, no se le dio importancia a este aspecto en su momento. Aun así, hay que recalcar que esto se trata de un aspecto a pulir en futuras versiones de la PoC.

Como se observa, se tiene un total de 20 tablas o relaciones. Mediante estas, se explicará el funcionamiento de la prueba de concepto a implementar. Cabe destacar que el atributo 'num_ejecución', existente en varias de las relaciones de la base de datos, no aporta información relativa al problema. Dicho atributo se incluyó por parte de los matemáticos para tener un control de las veces que se realizaba un cálculo de ruta óptima.

Procediendo pues, con la explicación de la base de datos tenemos lo siguiente:

En primer lugar, se tiene la tabla referente a los ***centros*** que intervienen en el cálculo de las rutas óptimas. Se tiene un atributo 'cod_centro' que los identifica, así como la 'descripción' que indica su nombre. Además, hay otro denominado 'tipo_centro' que nos dice si se trata de una tienda, almacén o proveedor mirando en la relación ***tipo_centro***. Por último, los atributos 'posicion_x' y 'posicion_y' serán necesarios para situar cada centro en el mapa de la prueba de concepto.

En segundo lugar, existen tablas referentes a la ***Oferta*** y la ***Demanda*** que a su vez van relacionadas con la tabla ***productos***. En la primera se almacenará la información relativa a los productos ofrecidos por cada proveedor, dando además información acerca de la cantidad, precio y calidad (relacionándose con la tabla ***calidades***) del producto ofertado. En la segunda, se tiene lo

referente a lo demandado por las tiendas de cada producto, incluyendo la cantidad demandada y precio de cada producto, lo que se espera vender de este y el nivel de regionalismo de tal. Se puede ver, que en la tabla ***productos*** se nos informa acerca del código de cada producto 'cod_producto', su 'descripción' y 'caducidad'. Además, esta se relaciona con la tabla ***grupos_productos***, la cual nos informa de los distintos grupos de productos existentes que se sustituyen entre sí. Concretamente, se tiene el atributo 'cod_grupo' que lo identifica, así como el atributo 'sustituto', que indica el código del producto y el 'orden'. El atributo 'orden' tomará el valor 1 para aquellos productos que sean principales en un grupo, siendo el primer sustituto el que tenga el valor 2 y así sucesivamente.

En tercer lugar, existen relaciones relativas a los costes entre los distintos centros. En la primera, ***coste_trayecto1*** se muestra cuál es el coste entre cada uno de los centros del primer tipo y del segundo, es decir, el coste entre todos los caminos entre un proveedor y un almacén. La segunda, ***coste_trayecto2***, indica el coste de todos los caminos existentes entre dos almacenes. Por último, ***coste_trayecto3*** informa del coste del camino entre un almacén y una tienda.

En cuarto lugar, se tienen relaciones del tipo ***resultado_trayecto1***, ***resultado_trayecto2*** y ***resultado_trayecto3***. Dichas tablas dan información acerca de lo transportado en cada trayecto de una ruta óptima. Más concretamente, ***resultado_trayecto1***, informa de la 'cantidad' de producto de código 'cod_producto' transportada del proveedor con código 'cod_proveedor' al almacén con código 'cod_almacen'. Las otras dos tablas indican la misma información, referente al trayecto entre dos almacenes (con código 'cod_almcen1' y 'cod_almacen2') y entre un almacén (con código 'cod_almacen') y una tienda (con código 'cod_tienda'), respectivamente.

En quinto lugar, se encuentran las tablas denominadas ***trazabilidad_cabecera*** y ***trazabilidad_detalle***. En la primera, se almacena la información referente a una ruta identificada con el atributo 'id_traza'. De dicha traza se sabe el 'cod_proveedor' del proveedor origen, así como el código ('cod_tienda') de la tienda destino. Se sabe, además, el código ('cod_producto') y la 'cantidad' de cada producto transportado en dicha ruta. La segunda tabla, reúne la información referente a los almacenes por los que pasa una determinada traza de código 'idtraza'. Se detalla en esta, la información descrita en ***trazabilidad_cabecera***, esto es, nos indica por cuales almacenes se ha pasado al transportar el producto. El atributo 'orden' de esta relación indica el orden de recorrido de los almacenes, siendo el almacén de 'orden' 1 el primero, y así sucesivamente.

En sexto lugar, se tienen las tablas de ***porcentajes***, ***resultados*** y ***tiendas_salida***. En ***porcentajes*** se almacenan los valores introducidos por el usuario que posteriormente se recuperarán para el cálculo óptimo de rutas. En ***resultados*** se almacenará información general una vez realizado el cálculo de rutas ('coste_transporte' total, 'precio_compra' total, 'calidad' media de los productos suministrados…etc.). En ***tiendas_salida*** se almacenará información de la 'cantidad' de producto (de código 'cod_producto') recibido en cada tienda, identificada por su código 'cod_tienda'. Además, se podrá saber con el atributo 'sustitutoS/N', si el producto que llega a tienda se trata de un sustituto o es un producto principal.

Finalmente, queda hablar de las relaciones ***fecha_consulta*** y ***centros_festividades***. En la primera, se almacena la fecha de la simulación, es decir, el día en el que se desea llevar a cabo el transporte óptimo de mercancías. Dicha fecha, se almacena en el sistema en el atributo 'fecha_cons'. En la segunda relación, ***centros_festividades***, se almacena el código de los centros ('cod_centro'), junto con las fechas ('fecha_festivo') en las que permanecen cerrados. En versiones posteriores de la

prueba de concepto, se espera tener dicha información en consideración. Para esto, se descartarán aquellos centros cerrados en la fecha que se va a simular, del total de centros participantes en el cálculo de rutas óptimas.

A continuación, se procederá a describir la arquitectura del proyecto desde el punto de vista de las distintas partes que lo forman, concretando así, la arquitectura del sistema a desarrollar.

4.2 Arquitectura del sistema

En cuanto a la prueba de concepto a implementar, esta se basa en una aplicación con arquitectura cliente - servidor (*frontend* y *backend* respectivamente). Más concretamente, las acciones que realiza el usuario serán gestionadas en la parte del *frontend*, el cual a su vez solicitará servicios a un API REST.

Para entender mejor los conceptos, me basaré en resumir cierta información detallada por Cecilio Álvarez Caules, escritor de varios libros sobre programación en Java y otros *frameworks*, en arquitecturajava.com [29]. La forma de acceder a los servicios implementados en el *backend*, se lleva a cabo mediante el uso del módulo *HttpClient*, el cual envía las distintas peticiones al servidor REST o *backend*.

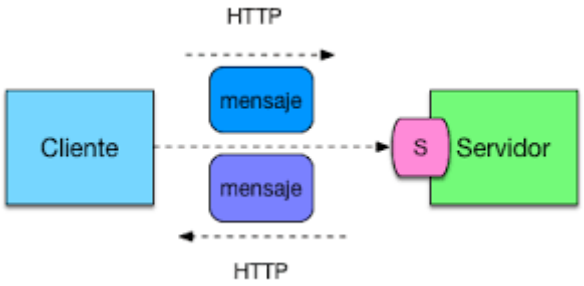

Figura 21: Figura ilustrativa de la comunicación entre cliente y servidor. Fuente:[29].

Una vez recibida una solicitud del cliente, el servidor realiza las operaciones convenientes con la base de datos y actualiza o recupera las filas solicitadas en la petición de este. Seguidamente, retorna, si es el caso, la información solicitada.

Por ahora, se sabe que el cliente pide datos al servidor, los cuales obtiene de la base de datos al que este último se conecta. ¿Pero, como se lleva a cabo este proceso? En el siguiente apartado se dará respuesta a la anterior cuestión.

4.3 Diseño Detallado

El presente punto se centrará en detallar el proceso de comunicación entre el cliente y servidor. Para ello, se hablará de los datos necesarios para realizar dicha comunicación y de cómo se trabaja con estos. Seguidamente se detallará cada una de las partes de la arquitectura de la prueba de concepto.

4.3.1 Transmisión de datos entre cliente y servidor

Para la elaboración de este punto, me basaré en resumir algunos conceptos destacados por Yone Moreno Jiménez [31], obtenidos de uno de los cursos estudiados en el período de incorporación a la empresa. Estos conceptos de iniciación a Angular se detallaron en el curso de formación de Deborah Kurata, *Angular: Getting Started*, accesible desde Pluralsight.com [30].

En el escenario que se nos presenta, el cliente realizará peticiones en un momento X, no obstante, la respuesta del servidor será obtenida en un instante de tiempo posterior. A este concepto se le llama asincronía.

Serán necesarios pues, un tipo de datos que permitan abordar dicha situación asíncrona. Este tipo de datos son los *Observables*. Mediante los *Observables*, será posible solventar los problemas existentes en estas situaciones. Dichos datos, son una especie de *arrays* que tratan los eventos y respuestas del servidor como si de una colección de cosas se tratase. Angular hace uso de este tipo de datos mediante una librería de terceros llamada RxJS.

Como bien se detalla en el artículo de Osman Cea de título *RxJS: de cero a experto en 15 minutos* [32], RxJS es una librería de programación reactiva, el fin de la cual es la composición de código asíncrono. Dicho código se basa en eventos a través de secuencias observables. Mediante esta librería, se logra proveerse de la estructura de datos anteriormente nombrada *Observable* y de operadores para manipularla.

Bien, se sabe que los datos necesarios para poder llevar a cabo las comunicaciones de forma correcta son los *Observables*. Para concretar más acerca de esto, hay que hablar acerca de cómo se trabaja con dichos datos.

Un *Observable* es un objeto que dispone de un método *subscribe* que toma como parámetro un observador, devolviendo una suscripción. El observador se subscribe al *Observable* y cada vez que este emita un valor, dicho observador es notificado.

A diferencia de las promesas existentes en JavaScript, un *Observable* recibe y emite distintos eventos a la vez. Además, por defecto, no emite nada, simplemente espera a recibir datos. Es un objeto cancelable, es decir, permite la cancelación de las suscripciones haciendo uso del método *unsubscribe*. Por último, dicho objeto acepta operadores como *map* y *reduce* que aplican una determinada función al flujo de eventos que recibe el *Observable* en orden.

4.3.2 MóduloHttpClient

Una vez introducidos los objetos necesarios para realizar la comunicación entre los componentes de la arquitectura del sistema, interesará saber cómo es llevado a cabo todo este proceso de transferencia y/o modificación de datos.

Para ello, se ha nombrado anteriormente el módulo *HttpClient*, encargado de llevar a cabo dicha comunicación entre componentes. Se detallará ahora pues, como hacer uso de dicho módulo.

Como se introdujo al hablar de Angular, dicho *framework* presenta una estructura de funcionamiento basada en componentes, módulos y servicios. En los servicios es pues, donde será necesario hacer uso del módulo *HttpClient* para llevar a cabo las peticiones del cliente al servidor.

Para ejemplificar el proceso de realización de una petición HTTP, se hará uso del servicio definido para las tiendas, nombrado *shops.service.ts*. Dicho servicio mediante la utilización del módulo http será capaz de realizar los GET[13] y POST[14] necesarios al servidor.

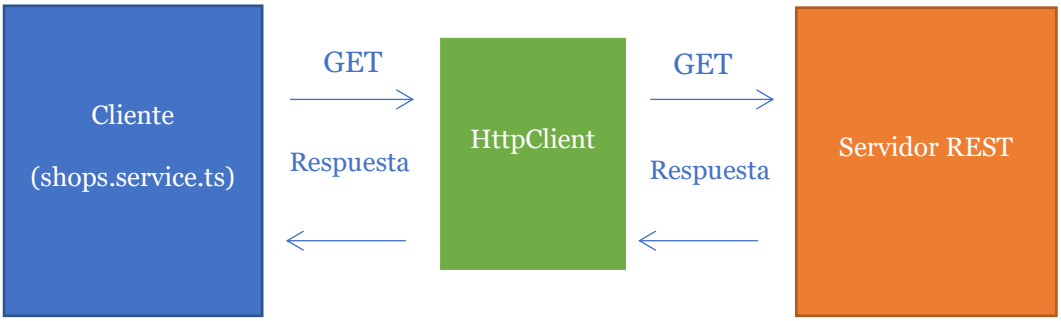

Figura 22: Esquema de envío de una petición HTPP. Fuente: elaboración propia.

Se tendrá para ello, declarada una variable encargada de almacenar la URL del servidor de donde se quieren obtener o modificar los recursos. Dicha variable, en el caso de la prueba de concepto que se está detallando, se denomina *config*. Se trata de una constante de tipo arbitrario, que a su vez define en ella una variable *basePath*, como bien se puede observar en la siguiente figura.

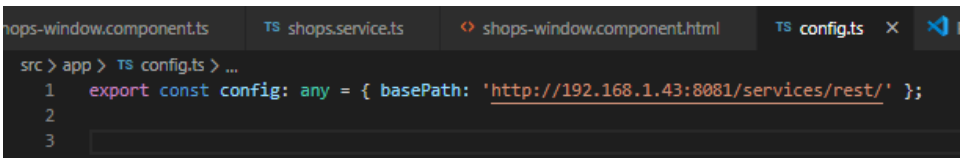

Figura 23: Variable que contiene la ruta de conexión al backend de la PoC. Fuente: Captura de código de la PoC.

Luego, en *shops.service.ts* se tendrá, entre los *imports*, los correspondientes a dicha variable, al módulo *HttpClient* y a *Observable*, de la librería rxjs.

Figura 24: Distintos imports de shop.service.ts. Fuente: Captura de código de la PoC.

En dicho *service* de la prueba de concepto, se encuentran también, distintas funciones que servirán para obtener y modificar cada uno de los recursos relacionados, en este caso, con las tiendas de nuestra prueba de concepto. Puede verse en la figura siguiente, que la función *getShops()* hace uso del módulo *HttpClient* (objeto con nombre http en la figura). Se realiza una petición GET al servidor, retornando un *Observable*. Se observa, además, que se utiliza la variable que almacena el *basePath* donde se sitúan los recursos, seguida del nombre del servicio REST al que se desea acceder, en este caso el servicio 'shops'.

[13] GET: Solicitud de registros de la base de tatos.
[14] POST: Modificación de registros de la base de datos.

```
getShops(): Observable<any> {
  return this.http.get(`${config.basePath}shops`).map(response => {
    return response;
  });
}
```

Figura 25: Ejemplo ilustrativo de petición http. Fuente: Captura de código de la PoC.

La prueba de concepto, hará uso de dicha función de *shops.service* cada vez que necesite obtener las tiendas de la base de datos. Puede verse un ejemplo de obtención de dicha información y de utilización del método *subscribe* en *shops.component*. Para verlo, hay que prestar atención al método *OnInit()* del componente referente a las tiendas. En un momento determinado de dicho método, se obtienen los registros correspondientes a la información de las distintas tiendas de la *DB*[15]. Esta información será necesaria para poder mostrar lo correspondiente en pantalla.

```
this.service.getShops().subscribe(dataShops => {
  if (dataShops != null) {
    this.data = dataShops;
```

Figura 26: Fragmento ilustrativo de uso de la función getShops() en shops.component.ts. Fuente: Captura de código de la PoC.

Se ha nombrado anteriormente que el cliente solicita servicios a una API REST, pero ¿a qué se refiere esto de REST? Para concretar conceptos y poder seguir detallando la estructura de la PoC, se hablará en el siguiente punto sobre qué es lo que se entiende por REST.

4.3.3. REST

Para acabar de aclarar las posibles dudas que pudieran surgir acerca del concepto de REST, se resumirán en este punto los conceptos esenciales descritos en el artículo citado anteriormente de Álvarez Caules [29].

El término REST hace referencia a *Representational State Transfer*. Hay cierta confusión en cuanto este concepto. Se dice que REST se trata de un patrón de diseño, otra gente afirma que es una API, para otros REST es una arquitectura. En definitiva, REST es un estilo de arquitectura utilizado a la hora de realizar comunicaciones entre cliente y servidor.

Normalmente, un cliente se comunica con un servidor a través de un punto de acceso haciendo uso de una comunicación http, vista anteriormente. Con esto se consigue que la comunicación sea abierta y se pueda acceder desde cualquier sitio. En esta comunicación, se intercambian mensajes entre el cliente y el servidor, estos suelen estar en formato JSON[16] o XML[17].

Sin tener nada más en consideración, con lo dicho hasta ahora se tendría lo denominado un nivel de organización 0, es decir, no existe ningún nivel de organización. Esto podría considerarse una situación de caos. Interesaría pues, organizar la información de manera que, al realizar las peticiones desde el cliente, fuese más sencillo dar con los datos deseados.

[15] DB: Database (base de datos).

[16] JSON: JavaScript Object Notation.

[17] XML: Extensible Markup Language.

Mediante la arquitectura REST, se pretende que los servicios que ofrece nuestra aplicación, ya sea un navegador, o para ser utilizados por otra aplicación, puedan ser definidos en términos de recursos y mediante las operaciones provistas por el protocolo http. Más adelante se verán reflejados estos conceptos en la descripción de la estructura del *backend* de la PoC.

A continuación, se pasará a describir las distintas partes de la estructura de la prueba de concepto, entrando con más profundidad a describir el problema y el modelo con el que se intentará resolver en el próximo capítulo.

4.3.4 El frontend

Existen multitud de sitios donde encontrar definiciones referentes a lo que se entiende por *frontend* y *backend*. Tras consultar Wikipedia [33], publicaciones como la de Víctor Manuel Acosta en Revista Digital INESEM [34] y webs educativas como Platzi [35] se pueden esclarecer un poco los conceptos.

La parte del *frontend* de la PoC es la encargada de llevar a cabo la interacción con los usuarios, es por ello por lo que también se denomina cliente o parte del cliente. Los encargados de diseñar dicha parte pues, se encargan del diseño, experiencia de usuario y funcionalidad de la aplicación o prueba de concepto.

En el caso de nuestra prueba de concepto, la parte del *frontend* ha sido implementada en Angular. Como bien se ha visto en el apartado del Estado del arte, este es un *framework* basado en TypeScript, que a su vez necesita de Node.js para funcionar y gestionar las dependencias.

Como bien dice Acosta [34], existen ciertos beneficios derivados de utilizar un *frontend* basado en Angular. Dichos beneficios agilizan y facilitan el desarrollo y programación de aplicaciones. Algunos de ellos son los siguientes:

La reusabilidad: Gracias a la estructura basada en componentes presentada por Angular, se permite que estos sean utilizados reiteradamente en toda nuestra prueba de concepto. Como ejemplo, en nuestra PoC se reutilizan constantemente componentes referentes a botones, así como *sliders* u otros. Esto ahorra tiempo en tareas de redefinición de elementos o componentes.

Independencia en cuanto a las pruebas: Debido a que cada componente es independiente entre sí, las pruebas independientes o unitarias serán mucho más fáciles de llevar a cabo. Esto podría ejemplificarse a la hora de probar el funcionamiento de uno de los *sliders* de nuestra prueba de concepto. Las modificaciones que realicemos sobre dicho componente, cada vez que se quiera probar un determinado funcionamiento de este, no afectarán al funcionamiento de los otros componentes existentes.

La parte del *frontend* de la PoC, será la encargada pues, de hacer uso del módulo *HttpClient* descrito anteriormente para comunicarse con el servidor o *backend*.

4.3.5 El backend

Cuando se habla de *backend*, se está refiriendo a la parte del sitio web (en nuestro caso, de la PoC) que conecta con la base de datos y el servidor que este utiliza, es por esto por lo que también se suele denominar parte del servidor.

En el caso que nos ocupa, el *backend* está implementado en Java, siguiendo una arquitectura REST para las comunicaciones con el cliente. En él se define toda la lógica de los servicios, accesibles desde el *frontend* mediante el uso del módulo *HttpClient*.

El servidor, en nuestro caso, expondrá sus servicios en la clase *VisitorRestService.java*, interfaz que será implementada en *VisitorRestServiceImpl.java* (dichas clases serán mostradas más adelante). La clase nombrada anteriormente pues, se encargará de atender las peticiones del *frontend* modificando los registros y proporcionando los datos solicitados al cliente.

La implementación de una nueva base de datos para la prueba de concepto obligó a la adaptación de los servicios existentes en el *backend* o servidor de esta, así como la implementación de otros nuevos para cubrir nuevas necesidades. Esto supuso tener que comprender todo el proceso de comunicación entre las dos partes, para poder implementar correctamente la transferencia de datos.

A continuación, mostraré todo el proceso de comunicación descrito, llevado a cabo entre el *frontend* y *backend* de la PoC. Me centraré en la parte correspondiente al *backend*, ya que es la que se encarga de obtener los datos, mediante sencillas consultas, y devolverlos al cliente. Además, la estructura de esta parte de la PoC (del *backend*) ha sido implementada haciendo uso de Devon, *framework* nacido en la empresa.

Como ejemplo de todo esto, mostraré el proceso de cálculo de rutas óptimas, proceso en el que basa su principal funcionamiento la PoC.

4.3.6 Ejemplo de comunicación entre cliente y servidor

Se parte del archivo HTML perteneciente al componente que incluye el mapa de la app o PoC, junto con los controles de coeficientes y botones de cálculo de rutas y navegación. Aquí pues, interesa centrarse, para el procedimiento de cálculo de rutas, en el botón de id 'btnRuta'. Este, al ser pulsado, llamará al método *calculateRouteMap()* implementado en el archivo .ts referente al componente del mapa.

```
<button class="btnRuta" id="btnRuta" (click)="calculateRouteMap()" onClick ="desactivar()">Calcular ruta</button>
<button class="btnClean" id="btnClean" (click)="cleanMap()" onClick ="activar()">Limpiar</button>
<button class="btnDatos" id="btnData" (click)="toData()">Ver datos</button>
```

Figura 27: Fragmento de map.html. Fuente: Captura de código de la PoC.

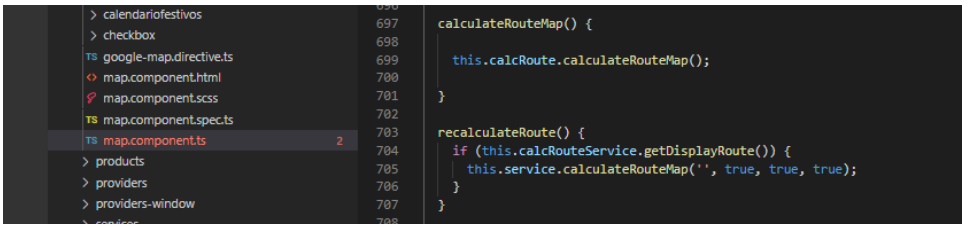

Figura 28: Captura esquemática de código correspondiente a map.component.ts. Fuente: Captura de código de la PoC.

Seguidamente, dicho método llamará al correspondiente del servicio de cálculo de rutas. En este, se comprueba la corrección de los datos introducidos por el usuario para el cálculo. A continuación, mediante una serie de llamadas a métodos, se solicita al servicio de cálculo de rutas óptimas, los resultados necesarios para representar toda la información en el mapa. No se entrará en detalle en la parte del cliente, ya que lo que interesará es mostrar el flujo de los datos una vez recibida la petición en el *backend*.

Aun así, decir que, para dibujar las rutas en el mapa de la PoC, se ha hecho uso del *Google Maps API Directions Service*. Dicho *service*, permite dibujar de manera dinámica, respondiendo en tiempo real al usuario tras solicitar el cálculo, las rutas optimizadas. Concretamente, mediante el objeto *DirectionsRenderer*, se mostrarán los caminos óptimos entre los 2 o más puntos especificados. Para más información, consultar la documentación de la plataforma de Google Maps [36].

```
this.http
  .post(`${config.basePath}coeficientes`, {
    calidad: calidad,
    coste: coste,
    precio_coste: precio_coste,
    prevision_tienda: prevision_tienda,
    precio_tienda: precio_tienda,
    homogeneidad: homogeneidad,
    regionalismo: regionalismo
  }).subscribe(data => {
    this.setData(data.json());
    this.setJourney(this.data[0]);
    if (this.data[0].length === 0 || !displayRoute) {
    } else {
      this.displayRoute = true;
      setTimeout(() => {
        this.calculateRouteMap(
          this.data[0],
          this.getVisibleProviders(),
          this.getVisibleShops(),
          this.getVisibleStores()
        );
        this.setStoresOut(this.data[1]);
        this.setProvidersOut(this.data[2]);
        this.setShopsOut(this.data[3]);
      }, 100);

    }
  });
```

Figura 29: Petición de cálculo de rutas por parte del cliente al servidor.
Fuente: Captura de código de la PoC.

Se observa en la figura anterior, la parte correspondiente a la solicitud de datos al servidor o *backend*, hallada en el método correspondiente de *map.service.ts*. Se puede ver como se hace uso del método subscribe, anteriormente nombrado, para la espera, recepción y posterior tratamiento de los datos recibidos por parte del servidor REST.

Ahora pues, se verá cómo se gestiona dicha petición desde la parte del servidor. Como bien se dijo, una vez recibida la petición del cliente, dicho servidor realizará las consultas y *updates*[18] necesarios para retornar los datos solicitados

Antes de seguir con el ejemplo de comunicación que se está detallando, interesa tener una idea de cómo se ha implementado y estructurado la parte relativa al *backend* de la PoC. El proyecto referente a la parte del servidor se implementó siguiendo, como bien se ha dicho antes, una estructura cliente-servidor basada en *REST-Services*. Para ello, se partió de la estructura conocida de un proyecto prueba, accesible en Github desarrollado con Devon [8].

Partiendo del ejemplo de servidor dicho y siguiendo la guía de desarrollo del equipo de Devon, se creó la estructura inicial del proyecto, automatizando el proceso gracias a CobiGen. Seguidamente, se añadieron las interfaces y clases necesarias en cada uno de los niveles de la

[18] *Updates*: Actualizaciones de la base de datos.

arquitectura que se comentará a continuación. En la siguiente figura se observan los 3 directorios principales sobre los que basa su funcionamiento el *backend* de la PoC.

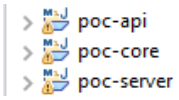

Figura 30: Directorios principales estructura backend.
Fuente: Captura de la estructura del backend de la PoC.

En el proyecto poc-core se encuentra todo aquello relacionado con la lógica y funcionalidad de la PoC en concreto. En poc-api están contenidas las definiciones de todas las interfaces de la prueba de concepto. El directorio denominado poc-server contiene lo necesario para para la 'paquetización' de la PoC.

Interesa ahora entrar, con un poco más de detalle, a describir lo contenido en el directorio 'poc-core'. La arquitectura implementada consta de tres capas: servicio, lógica y datos. Dichas capas pueden observarse en la siguiente figura, en la cual se pueden ver también *packages* referentes a configuración, seguridad, constantes y otros aspectos. Dichos paquetes, denominados como 'general', fueron creados de manera automática durante el proceso de construcción de la estructura base del proyecto, siguiendo la guía de Devon [8].

Los *packages* de interés son los que contienen el nombre 'project' en ellos. Estos fueron los creados a posteriori para implementar el funcionamiento del *backend* o servidor de la PoC.

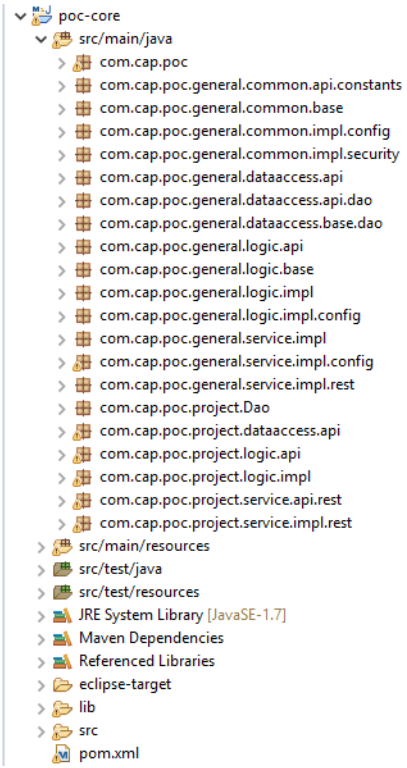

Figura 31: Estructura de directorios y packages de poc-core.
Fuente: Captura de la estructura del backend de la PoC.

A modo de esquema, se muestra en la siguiente figura el diagrama de clases correspondiente a la parte específica del proyecto del *backend*. Se trata de un diagrama simplificado. En este solo se muestran los métodos equivalentes a los servicios en las clases que los exponen, así como en cada una de las interfaces de los objetos que los definen.

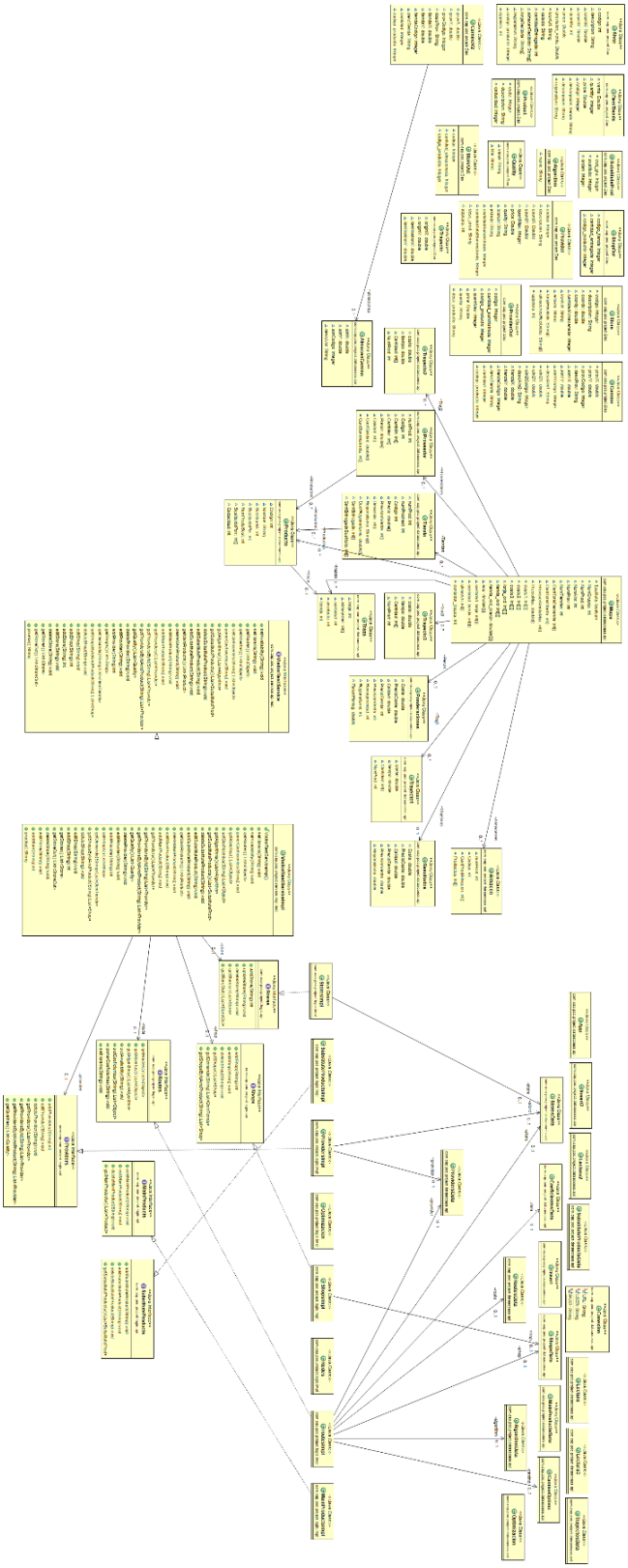

Figura 32: Diagrama de clases del backend de la PoC. Fuente: Backend de la PoC.

Una vez mostrada la estructura de directorios, paquetes y el diagrama de clases correspondiente al *backend*, se continuará detallando el proceso de comunicación previamente introducido. Ahora pues, desde la parte del servidor, una vez hecha la petición desde el *frontend* para obtener los resultados de cálculo de rutas óptimo, se recibirá la solicitud del cliente. Tanto esta, como todas las solicitudes recibidas, se centralizan en la capa de servicio del *backend*, más concretamente en la clase *VisitorRestServicesImpl.java*.

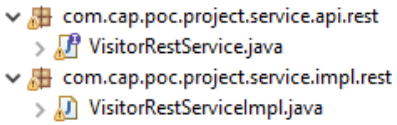

Figura 33: Captura de las clases de la capa de servicios del backend. Fuente: Código del backend de la PoC.

En la figura 33 se puede ver la definición de algunos de los servicios expuestos en la interfaz *VisitorRestService* del *backend*, entre ellos, el de cálculo de rutas óptimas. La clase *VisitorRestServicesImpl.java* pues, implementará dicha interfaz y redirigirá después las peticiones a quién debe encargarse de la petición.

```
@Named("VisitorRestService")
public class VisitorRestServiceImpl implements VisitorRestService {

  @Inject
  private Stores store;

  @Inject
  private Shops shop;

  @Inject
  private Providers provider;

  @Inject
  private Routes route;

@Override
public List<Object> setCoeficientes(String obj)
        throws ClassNotFoundException, SQLException, JSONException {
  return this.route.setCoeficientes(obj);
}
```

Figura 34: Recorte de código de la clase VisitorRestServiceImpl. Fuente: Código del backend de la PoC.

Puede verse en la figura anterior, que la petición se redirige haciendo uso del objeto *route*, de tipo definido en la interfaz *Routes.java* e implementado en *RoutesImpl.java*. Ahora pues, se ha pasado de la capa de servicios a la capa lógica, como bien se puede comprobar en la siguiente figura.

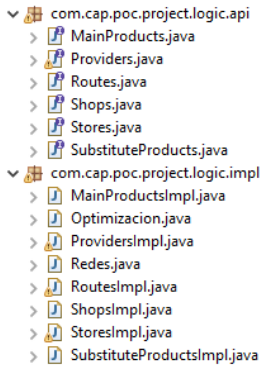

Figura 35: Captura de las clases de la capa lógica del backend. Fuente: Código del backend de la PoC.

En la capa actual, se recibirá el objeto JSON enviado por el cliente. De dicho objeto, se extraerán los datos necesarios para realizar las consultas y/o modificaciones a la base de datos. Todo esto puede verse en la figura siguiente.

```
public List<Object> setCoeficientes(String coeficientes)
    throws NumberFormatException, JSONException, ClassNotFoundException, SQLException {

  JSONObject json = new JSONObject(coeficientes);
  System.out.println(json);

  double calidad = json.getInt("calidad");
  double coste = json.getInt("coste");
  double precio_coste = json.getInt("precio_coste");
  double regionalismo = json.getInt("regionalismo");
  int prevision_tienda = json.getInt("prevision_tienda");
  int precio_tienda = json.getInt("precio_tienda");
  int homogeneidad = json.getInt("homogeneidad");

  this.data.setCoeficientes(calidad, coste, precio_coste, regionalismo, prevision_tienda, precio_tienda,
      homogeneidad);

  try {
      App.rellenardb();
  } catch (IloException e) {

      e.printStackTrace();
  } catch (IOException e) {

      e.printStackTrace();
      System.out.println("ERRO000000000000000R");
  }
  return this.camino.getCaminoOptimo();

}
```

Figura 36: Código del método setCoeficientes() de la capa lógica del backend.
Fuente: Código del backend de la PoC.

Tras obtener los atributos necesarios (de nombre conocido) del objeto recibido, se llamará al método correspondiente de la capa de datos. Este método actualizará la tabla de coeficientes de la base. Dicha tabla será leída junto con los otros datos necesarios para el cálculo de rutas en el algoritmo matemático (todo esto se realiza en la llamada a *App.rellenardb()* de la figura anterior).

```
Conexion conexion = new Conexion();
con = conexion.crearConexion();
rs = st.executeQuery("SELECT * FROM trazabilidad_cabecera");
```

Figura 37: Lineas de código del método correspondiente a la capa de datos del backend.
Fuente: Captura de código de la PoC.

Una vez obtenidos los resultados, se llamará al método correspondiente de la capa de datos que obtenga la información necesaria a devolver al cliente. En este caso *getCaminoOptimo()*. En este método de la capa de datos, se crean las conexiones a la base y se implementan las consultas que obtendrán los resultados del cálculo de rutas. A modo de ejemplo esquemático, esto puede verse en la figura anterior (figura 37), en la cual se muestran algunas líneas del método en cuestión.

Durante la implementación del *backend* de la POC se intentaron configurar las conexiones con la base de datos en el fichero *application.properties* de Spring. Con esto Spring se encargaría de gestionar las conexiones. No obstante, por determinados aspectos de Devon, no tener soporte y tampoco tiempo suficiente debido a la exigencia de evolutivos, se tuvo que recurrir al cierre manual de las conexiones, debido al límite de estas.

Para devolver al cliente la información necesaria, se deberán realizar varias consultas. Como se observa en la figura anterior, la primera se hará sobre la tabla ***trazabilidad_cabezera***. Dicha tabla, como ya se ha dicho, contiene información general sobre cada traza o recorrido óptimo. De aquí, se obtendrán los códigos de los distintos proveedores y tiendas que forman parte de la solución junto con el id de la traza correspondiente.

Seguidamente, se consultará la tabla ***centros*** para construir los objetos relativos a los proveedores y tiendas, para las cuales es necesario conocer las coordenadas. Como último paso, se concatenará pues, la tabla centros con la de ***trazabilidad_cabecera***, haciendo uso del código del primer almacén y el id de la traza. Con esto se obtendrá la secuencia de almacenes correspondiente a cada traza.

Finalmente, se devolverá un *array* que contenga cada uno de los caminos solución, así como el conjunto de proveedores, tiendas y almacenes implicados. Ahora pues, este método de la capa de datos devolverá la información a la capa lógica. Esta última, a su vez, retornará los datos a la capa de servicios, la cual devolverá los resultados solicitados por el cliente en el *frontend*.

5. Modelo propuesto

Como bien se ha dicho a la hora de introducir la propuesta de TFG, este trabajo no va orientado al desarrollo de una aplicación o software. El propósito de este trabajo es detallar el proceso de implantación de una mejora en la optimización de procesos logísticos y de gestión de un establecimiento de comercialización. Se ha pretendido también, darle importancia al cómo fluye la información entre las distintas partes de la arquitectura.

Este proyecto pues, no podría categorizarse dentro del ámbito de desarrollo software, ya que no he sido el principal encargado del proceso de realización del proyecto, por ser este de amplias dimensiones y abarcar a mucha gente en su realización, además de no haber estado presente durante todo el proceso de desarrollo. La implementación de la base no corrió a mi cargo, al igual que otras decisiones y funcionalidades de la prueba de concepto. Durante mi estancia en la empresa he tenido que ceñirme a lo que se me encargaba, con posibilidad de opinión y libre expresión, pero atado al fin y al cabo a las tareas que me correspondían.

Es por esto por lo que se ha enfocado el proyecto al entendimiento o comprensión del funcionamiento de la PoC. Así pues, el enfoque que se le va a dar a este punto, tras haber detallado el procedimiento de comunicación entre las distintas partes de la arquitectura de la PoC, consistirá en detallar el modelo matemático en el que se basa toda la funcionalidad del proyecto.

En el capítulo anterior, se ha mostrado que, una vez modificada la relación de coeficientes de la base, se llama al algoritmo que hará el cálculo de rutas. En este apartado pues, se detallará el modelo matemático a tener en consideración a la hora de implementar el algoritmo.

Se empezará pues, con la descripción del problema, siguiendo con la definición del modelo y sus respectivas restricciones y funciones, sobre las cuales se construye todo lo demás. Seguidamente, se mostrarán pequeños fragmentos de código en los que se puede ver la utilización de aspectos teóricos definidos en el modelo matemático, haciendo uso de la librería CPLEX.

5.1 Descripción del problema

El objetivo principal del proyecto, como ya se sabe, es la optimización de un sistema de rutas para distribuir productos provenientes del mar a diferentes tiendas. Interesará que el pescado fresco que se venda en los establecimientos de comercialización interesados llegue lo más rápido posible al punto de venta, minimizando su pérdida de frescura.

Se considera que el producto parte de unos nodos llamados lonjas y ha de pasar necesariamente por nodos del tipo almacén para llegar a nodos de tipo tienda. La intención es obtener la cantidad de cada producto que se envía a cada tienda desde cada lonja y cuál es la ruta que deben seguir para satisfacer las preferencias del usuario. Además de esto, durante su transporte, el producto podrá ser almacenado en distintos almacenes como stock. La secuencia de los almacenes en los que podrían almacenarse dichos productos también tendrá que ser representada.

Se trata de un problema de optimización multiobjetivo, ya que el problema se resuelve según diferentes funciones objetivo. Las preferencias del usuario sobre cada objetivo se expresan

mediante un sistema de pesos. A menudo, la oferta puede ser insuficiente para cubrir la demanda de un producto determinado. Esto es, la cantidad total de producto demandada por el total de las distintas tiendas podrá ser mayor a la cantidad total que pueden ofrecer las distintas lonjas. Debido a esto, cada producto puede tener asociado otro producto que actúe como sustituto de este. El objetivo de este producto sustituto es satisfacer, en la medida de lo posible, la demanda del producto a quien sustituye.

Además, se considera la posibilidad de que algún nodo se encuentre cerrado el día en que el usuario quiere calcular la ruta óptima. Directamente, se leerán los datos de aquellos nodos que estén disponibles. Dicha información relativa a las fechas festivas de cada uno de los centros se almacena en una de las relaciones de la base del proyecto, como bien se ha visto en la descripción de esta.

5.2 Objetivos del problema

El problema cuenta con diferentes objetivos mediante los cuales podemos optimizar las rutas para distribuir los productos. Distinguiremos dos grupos dentro de estos objetivos, aquellos regulados con pesos reales comprendidos entre 0 y 1 y otros que solo podrán valer exclusivamente 0 o 1. A nivel práctico, los objetivos regulados con pesos binarios se verán como modos y la activación de uno de ellos excluye al resto de ser activados.

Los objetivos que pueden ser calibrados con pesos de 0 a 1 son los siguientes:

1. Distancia recorrida: minimizar la suma de los costes de los trayectos utilizados.

2. Calidad de producto: a cada producto se le asigna un valor entre "baja", "media" y "alta" en cada lonja asociado a su calidad, este objetivo maximiza la calidad del producto que llega a tienda.

3. Coste de producto: minimizar el precio pagado por cada tienda por el total de producto adquirido.

Los objetivos definidos como modos son:

1. Regionalismo: cada tienda asigna a cada producto un valor entre "bajo", "medio" y "alto" asociado al consumo tradicional del producto en la región donde está situada la tienda. Este modo prioriza el reparto de productos a aquellas tiendas que tengan un mayor valor de regionalismo.

2. Previsión de ventas: priorizar el reparto de productos a aquellas tiendas con mayor previsión de ventas.

3. Precio tienda: priorizar el reparto de productos a aquellas tiendas con precio de producto más elevado.

4. Homogeneidad: equilibrar el número de producto que llega a cada tienda en el caso en que no haya oferta suficiente para cubrir la demanda. Aseguramos un mínimo de producto a cada tienda proporcional a su demanda. Este modo aparece en forma de restricción cuando la oferta sea menor que la demanda.

5.3 El modelo

Para reflejar todo lo dicho anteriormente se necesitará modelar el problema, es decir, construir un modelo que reúna todas las características que se quieren tener en cuenta. Dicho modelo constará pues, de varias variables de decisión, restricciones y diversas funciones objetivo (nos encontramos pues, ante un problema de optimización multiobjetivo). Al introducir los pesos por pantalla, se le dará más o menos importancia a uno u otro factor, influyendo esto en el cálculo final de las distintas rutas óptimas.

5.3.1 Variables decisión del modelo

Estas variables serán las que definan los datos solución que se desean obtener. Como principal solución al problema, interesa saber la cantidad de producto que se mueve entre cada tipo de centros. Como bien se sabe, existen 3 tipos de trayectos: de proveedor a almacén, de un almacén a otro y finalmente de un almacén a tienda. Las variables decisión a definir pues, serán las siguientes:

- $\mathbf{CantTransp1}_{i,j,p}$**:** cantidad transportada del producto p de la lonja i al almacén j.
- $\mathbf{CantTransp2}_{j1,j2,p}$**:** cantidad transportada del producto p del almacén j_1 al j_2.
- $\mathbf{CantTransp3}_{j,k,p}$**:** cantidad transportada del producto p del almacén j a la tienda k.

Se tendrá también un conjunto de variables binarias o booleanas. Dichas variables tomarán el valor 1 o 0 dependiendo de lo que se desee modelar en el problema. Las variables binarias a incluir serán las siguientes:

$\mathbf{x1}_{i,j}$ **:** variable booleana que vale 1 si se transporta algún producto por el trayecto que va del proveedor i al almacén j.

$\mathbf{x2}_{j1,j2}$**:** variable booleana que vale 1 si se transporta algún producto por el trayecto que va del almacén j_1 al almacén j_2.

$\mathbf{x3}_{j,k}$**:** variable booleana que vale 1 si se transporta algún producto por el trayecto que va del almacén j a la tienda k.

5.3.2 Constantes conocidas

Además de las variables decisión y binarias descritas, deberemos incluir otro tipo de datos. Existe un conjunto de valores conocidos que no alteraran su valor durante el cálculo de rutas, estos son:

$\boldsymbol{w_i}$**:** peso asociado a cada objetivo i.
$\mathbf{Dist1}_{i,j}$ **:** distancia del trayecto que va del proveedor i al almacén j.
$\mathbf{Dist2}_{j1,j2}$ **:** distancia del trayecto que va del proveedor j_1 al almacén j_2.
$\mathbf{Dist3}_{j,k}$**:** distancia del trayecto que va del almacén j a la tienda k.
$\mathbf{CoefCalidad}_{i,p}$**:** coeficiente asociado al producto p en la lonja i.
$\mathbf{Precio}_{i,p}$**:** precio del producto p en la lonja i.
$\mathbf{PrevVenta}_{j,p}$**:** previsión de venta del producto p en la tienda j.

$\mathbf{PrevVentaMax_p}$: previsión de venta más alta de entre todas las previsiones del problema para cada producto p.
$\mathbf{CoefReg_{j,p}}$: coeficiente asociado al producto p de la tienda j.
$\mathbf{PrecioVenta_{j,p}}$: precio de venta del producto p en la tienda j.
$\mathbf{PrecioVentaMax_p}$: precio de venta más alto de entre todos los precios de la tienda del problema para cada producto p.
$\mathbf{CantMax_{i,p}}$: cantidad máxima del producto p que puede distribuir la lonja i.
$\mathbf{Demanda_{j,p}}$: demanda de la tienda j del producto p.
$\mathbf{CantTotalDem_p}$: cantidad total de demanda del producto p.
$\mathbf{CantTotalOferta_p}$: cantidad total de oferta del producto p.
ParamHomog: parámetro utilizado en el modo Homogeneidad que indica el mínimo de producto que ha de recibir como mínimo cada tienda. Su valor puede variar entre el intervalo [0, 1].

n: número total de productos.
$\mathbf{n_P}$: número total de proveedores.
$\mathbf{n_A}$: número total de almacenes.
$\mathbf{n_T}$: número total de tiendas.

5.3.3 Función objetivo

En primer lugar resolvemos el problema de optimización para cada función objetivo para calcular el valor óptimo de cada objetivo z_i^* para todo $i \in \{1,\ldots,6\}$.

A continuación, con estos valores óptimos, construiremos la desviación de cada uno de los objetivos:

$$\mathrm{desv}_i(z) = \frac{z_i^* - z_i}{z_i^*} \quad 1 \leq i \leq 6.$$

Figura 38: Formula desviación típica de los objetivos. Fuente: Modelo matemático de la PoC.

Sean $\{w_1, \ldots, w_6\}$los pesos asociados a cada objetivo, la función objetivo del problema será la siguiente:

$$Z = \sum_{i=1}^{6} (-1)^{n_i} w_i \frac{z_i^* - z_i}{z_i^*},$$

Figura 39: Función objetivo del problema. Fuente: Modelo matemático de la PoC.

donde z_i es la función objetivo asociada a cada objetivo y n_i = 0 si el objetivo i se quiere maximizar y n_i = 1 en caso contrario. En particular:

- Distancia recorrida

$$z_1 = \sum_{i=1}^{n_P} \sum_{j=1}^{n_A} Dist1_{i,j} \cdot x1_{i,j} + \sum_{j_1=1}^{n_A} \sum_{j_2=1}^{n_A} Dist2_{j_1,j_2} \cdot x2_{j_1,j_2} + \sum_{j=1}^{n_A} \sum_{k=1}^{n_T} Dist3_{j,k} \cdot x3_{j,k},$$

Figura 40: Función objetivo asociada a la distancia recorrida. Fuente: Modelo matemático de la PoC.

- Calidad de producto

$$z_2 = \sum_{p=1}^{n} \sum_{i=1}^{n_P} \sum_{j=1}^{n_A} CantTransp1_{i,j,p} \cdot CoefCalidad_{i,p},$$

Figura 41: Función objetivo asociada a la calidad del producto. Fuente: Modelo matemático de la PoC.

- Coste de producto

$$z_3 = \sum_{p=1}^{n} \sum_{i=1}^{n_P} \sum_{j=1}^{n_A} CantTransp1_{i,j,p} \cdot Precio_{i,p},$$

Figura 42: Función objetivo asociada al coste del producto. Fuente: Modelo matemático de la PoC.

- Previsión de ventas

$$z_4 = \sum_{p=1}^{n} \sum_{j=1}^{n_A} \sum_{k=1}^{n_T} CantTransp3_{j,k,p} \cdot (PrevVentaMax_p - PrevVenta_{j,p}),$$

Figura 43: Función objetivo asociada a la previsión de ventas de cada tienda. Fuente: Modelo matemático de la PoC.

- Regionalismo

$$z_5 = \sum_{p=1}^{n} \sum_{j=1}^{n_A} \sum_{k=1}^{n_T} CantTransp3_{j,k,p} \cdot CoefReg_{j,p},$$

Figura 44: Función objetivo asociada al Regionalismo. Fuente: Modelo matemático de la PoC.

- Precio tienda

$$z_6 = \sum_{p=1}^{n} \sum_{j=1}^{n_A} \sum_{k=1}^{n_T} CantTransp3_{j,k,p} \cdot (PrecioVentaMax_p - PrecioVenta_{j,p}).$$

Figura 45: Función objetivo asociada al precio de venta de las tiendas. Fuente: Modelo matemático de la PoC.

5.3.4 Restricciones

La función objetivo compuesta de la sección anterior se optimiza sujeta a:

<u>Restricciones que regulan lo que sale de proveedores</u>

- La suma de los productos que salen a los almacenes desde cada proveedor y para cada producto ha de ser menor que la cantidad máxima del proveedor.

$$\sum_{j=1}^{n_A} CantTransp1_{i,j,p} \leq CantMax_{i,p} \quad 1 \leq p \leq n, \quad 1 \leq i \leq n_P.$$

Figura 46: Restricción que controla que no se supere la cantidad ofertada por proveedores.
Fuente: Modelo matemático de la PoC.

- Si la demanda total es mayor o igual que la oferta total para el producto p, se añadirá la siguiente restricción:

$$\sum_{j=1}^{n_A} CantTransp1_{i,j,p} = CantMax_{i,p} \quad 1 \leq p \leq n, \quad 1 \leq i \leq n_P.$$

Figura 47: Restricción que indica Demanda ≥ Oferta en un producto.
Fuente: Modelo matemático de la PoC.

- Si no existe demanda para el producto y el producto sustituye a otro producto y la demanda del producto al que sustituye es mayor o igual que su oferta total, se añadirá también la restricción anterior (figura 47).

Restricciones que regulan lo que entra y sale de almacenes

En los almacenes ha de salir la misma cantidad de productos que ha entrado. Con estas restricciones se habilitará el paso de mercancía entre proveedores, almacenes y tiendas.

$$-\sum_{i}^{n_P} CantTransp1_{i,j,p} + \sum_{j2}^{n_A} CantTransp2_{j,j2,p} - \sum_{j2}^{n_A} CantTransp2_{j2,j,p} + \sum_{k}^{n_T} CantTransp3_{j,k,p} = 0, \quad 1 \leq p \leq n, \quad 1 \leq j \leq n_A.$$

Figura 48: Restricción que controla que los que entra en un almacén sea equivalente a lo que sale.
Fuente: Modelo matemático de la PoC.

Notemos que el trayecto que va desde un almacén a sí mismo no está definido. Por tanto, se tendrá que escribir la siguiente restricción para evitar que las variables $\boldsymbol{CantTransp2_{j,j}}$ sean mayores que cero.

$$CantTransp2_{j,j} = 0, \quad 1 \leq j \leq n_A.$$

Figura 49: Restricción que evita caminos de un almacén con él mismo.
Fuente: Modelo matemático de la PoC.

Restricciones que regulan lo que entra a tienda

- Si la demanda total de un producto es menor que la oferta total, añadimos la siguiente restricción: la suma de los productos que llegan a una tienda ha de ser mayor que la demanda de la tienda para cada producto.

$$\sum_{j}^{n_A} CantTransp3_{j,k,p} \geq Demanda_{k,p} \quad 1 \leq p \leq n, \quad 1 \leq k \leq n_T.$$

Figura 50: Restricción que indica que la demanda de un producto es menor que la oferta de este. Fuente: Modelo matemático de la PoC.

- Si la demanda total de un producto es mayor que la oferta total añadimos la siguiente restricción:

$$\sum_{j}^{n_A} CantTransp3_{j,k,p} \leq Demanda_{k,p} \quad 1 \leq p \leq n, \quad 1 \leq k \leq n_T.$$

Figura 51: Restricción que indica el caso contrario a lo dicho en la figura 50. Fuente: Modelo matemático de la PoC.

  - Si se activa el modo homogeneidad se añade la restricción para

    $1 \leq p \leq n, 1 \leq k \leq n_T$:

$$\sum_{j}^{n_A} CantTransp3_{j,k,p} \geq ParamHomog \cdot \frac{Demanda_{k,p}}{CantTotalDem_p} \cdot CantTotalOferta_p.$$

Figura 52: Restricción que intenta repartir producto a todas las tiendas. Fuente: Modelo matemático de la PoC.

  - Ahora es posible que exista más de un producto sustituto, por ello vamos a llamar n_S al número de sustitutos siendo p_S un sustituto del producto p:

    - Si la suma de las ofertas totales del producto y de sus sustitutos es mayor que la suma de las demandas totales de estos productos añadimos la siguiente restricción para $1 \leq p \leq n, 1 \leq k \leq n_T$:

$$\sum_{j}^{n_A} CantTransp3_{j,k,p} + \sum_{l}^{n_s} \sum_{j}^{n_A} CantTransp3_{j,k,p_{Sl}} = Demanda_{j,p} + \sum_{l}^{n_s} Demanda_{j,p_{Sl}}$$

Figura 53: Restricción donde la oferta de productos y sustitutos > demanda de estos. Fuente: Modelo matemático de la PoC.

    - Si la suma de las ofertas totales del producto y de sus sustitutos es menor que la suma de las demandas totales de estos productos añadimos la siguiente restricción para $1 \leq p \leq n, 1 \leq k \leq n_T$:

$$\sum_{j}^{n_A} CantTransp3_{j,k,p} + \sum_{l}^{n_s} \sum_{j}^{n_A} CantTransp3_{j,k,p_{Sl}} \leq Demanda_{j,p} + \sum_{l}^{n_s} Demanda_{j,p_{Sl}}$$

Figura 54: Restricción que indica lo contrario que la figura 53. Fuente: Modelo matemático de la PoC.

Restricciones bandera

Las siguientes restricciones sirven para relacionar el valor de las variables del tipo ***CantTransp1***, ***CantTransp2***, ***CantTransp3*** con el de las variables del tipo ***x1***, ***x2***, ***x3***:

$$x1_{i,j} \leq \sum_{p}^{n} CantTransp1_{i,j,p} \leq \sum_{p}^{n} CantTotalOferta_p \cdot x1_{i,j}, \quad 1 \leq i \leq n_P, \quad 1 \leq j \leq n_A,$$

$$x2_{j_1,j_2} \leq \sum_{p}^{n} CantTransp2_{j_1,j_2,p} \leq \sum_{p}^{n} CantTotalOferta_p \cdot x2_{j_1,j_2}, \quad 1 \leq j_1 \leq n_A, \quad 1 \leq j_2 \leq n_A,$$

$$x3_{j,k} \leq \sum_{p}^{n} CantTransp3_{j,k,p} \leq \sum_{p}^{n} CantTotalOferta_p \cdot x3_{j,k}, \quad 1 \leq j \leq n_A, \quad 1 \leq k \leq n_T.$$

Figura 55: Restricciones que indican que, de transportarse producto en un determinado trayecto, la cantidad transportada de este será >0. Fuente: Modelo matemático de la PoC.

El número relativo al sumatorio de $\boldsymbol{CantTotalOferta_p}$ ha de ser un número suficientemente grande como para que la suma de cantidad de producto nunca pueda ser superado por este. Aunque este valor sería suficiente, en JAVA se ha utilizado el valor *Integer.MAX_VALUE* que utiliza el mayor valor entero disponible.

5.4 Ejemplos de utilización del modelo en CPLEX

Como bien se vio, para devolver los resultados demandados por el cliente, el *backend* deberá realizar el cálculo óptimo de rutas. Para ello se hará uso del optimizador CPLEX y distintos objetos de la librería *ilog*. Para más información, consultar la bibliografía correspondiente a dicho apartado en las referencias [24].

A continuación, se mostrarán algunos ejemplos de código implementado haciendo uso de dicha librería para modelar el modelo matemático. No se pretende detallar la totalidad del algoritmo, sino de mostrar algunos fragmentos de utilización de dicho optimizador.

```
// Variables

IloNumVar[][] x1 = new IloNumVar[Map.NumProv][Map.NumAlm]; // Se coge el camino i para llevar el producto j?
IloNumVar[][] x2 = new IloNumVar[Map.NumAlm][Map.NumAlm];
IloNumVar[][] x3 = new IloNumVar[Map.NumAlm][Map.NumTiendas];
IloNumVar[][][] CantTransp1 = new IloNumVar[Map.NumProv][Map.NumAlm][]; // La cantidad transportada del
                                                                        // camino j por el camino i
IloNumVar[][][] CantTransp2 = new IloNumVar[Map.NumAlm][Map.NumAlm][];
IloNumVar[][][] CantTransp3 = new IloNumVar[Map.NumAlm][Map.NumTiendas][];
```

Figura 56: Variables decisión del modelo, definidas en cplex. Fuente: Código del backend de la PoC.

En la anterior figura se puede observar la definición de las distintas variables decisión del modelo matemático. Como se puede ver, son las mismas que las vistas en el apartado de definición del problema. El objeto *Map* es una matriz multidimensional que contiene la información necesaria para el cálculo de rutas, la cual obtiene de la base de datos.

```
for (int i = 0; i < Map.NumProv; i++) {
    for (int j = 0; j < Map.NumAlm; j++) {
        x1[i][j] = cplex.boolVar();//x1 valdrá 0 ó 1
        CantTransp1[i][j] = cplex.intVarArray(Map.NumProd, 0, Integer.MAX_VALUE);
        /*variable entera de vectores cuyo tamaño es Map.NumProd y sus
         * valores oscilan entre 0 y Integer.MAX_VALUE */
    }
}
```

Figura 57: Bucle de instanciación de las variables referentes a los trayectos entre proveedor y almacén. Fuente: Código del backend de la PoC.

Seguidamente, se deberán instanciar cada una de las posiciones de las matrices referentes a los distintos trayectos y cantidades transportadas. En la figura de arriba se puede observar la instanciación de los trayectos entre proveedor y almacén. Equivalentemente se hará con los otros trayectos.

Además, las variables ***x1***, ***x2***, ***x3*** del modelo, como bien se ha dicho, indicarán si se debe tomar el camino i para transportar el producto j. Tras realizar los cálculos y obtener las distintas soluciones posibles, estas variables tomarán valor 1, si se toma dicho camino, o 0 en caso contrario. A su vez, las matrices de cantidades transportadas en cada trayecto tomarán los valores correspondientes. De entre todas las soluciones, Cplex seleccionara la óptima, siguiendo el criterio de optimización (minimizar en este caso).

Describir el proceso de cálculo del algoritmo sería muy engorroso y desbordaría los límites de esta propuesta de TFG. No obstante, conviene dar una idea general del funcionamiento del objeto Cplex para el cálculo de las rutas óptimas. Toda la documentación referente a la utilización del optimizador se puede consultar en las referencias [24].

```
IloCplex cplex = new IloCplex();
```

Figura 58: Objeto cplex utilizado para solucionar el problema matemático. Fuente: Código del backend de la PoC.

Haciendo uso del objeto que se muestra en la figura anterior, será posible definir una función objetivo a optimizar, con sus términos y restricciones pertinentes. Como se dijo en la descripción del modelo matemático, dicha función objetivo se calculará como la suma de las desviaciones de cada uno de los objetivos por separado.

```
IloLinearNumExpr coste_camino = cplex.linearNumExpr();

for (int i = 0; i < Map.NumProv; i++) {
    for (int j = 0; j < Map.NumAlm; j++) {
        coste_camino.addTerm(Map.Traj1[i][j].coste, x1[i][j]);
    }
}
```

Figura 59: Bucle de adición de los distintos términos para el cálculo de la función óptima del objetivo 'coste_camino'. Fuente: Código del backend de la PoC.

Tomando como ejemplo de cálculo de solución óptima parcial, el objetivo de coste del camino se observa lo mostrado en la figura anterior. Se creará una expresión lineal, la cual contendrá como términos los costes de cada uno de los trayectos, multiplicados por la variable conveniente que indicará si se deben coger o no.

```
cplex.addMinimize(coste_camino);

// Subject to
Restricciones(Map, cplex, x1, x2, x3, CantTransp1, CantTransp2, CantTransp3, Principal );

coste_camino.setConstant(1);
// Solución modelo
cplex.setParam(IloCplex.Param.MIP.Display, 0);

if (cplex.solve()) {

    optimo = cplex.getValue(coste_camino);

} else {
    optimo = 1;
}
```

Figura 60: Adición de restricciones y obtención del óptimo. Fuente: Código del backend de la PoC.

Lo siguiente será indicar que dicha función se desea minimizar, incluir las restricciones sujetas a dicho objetivo e indicar a Cplex que calcule la solución óptima. Todo esto puede observarse en la figura anterior.

Ahora pues, se hará lo mismo con los otros objetivos. Se calculará después la desviación respecto a cada uno de los óptimos y se realizará la suma de todas ellas, minimizando la función objetivo resultante. Con esto pues, se obtendrán los resultados de cálculo de rutas óptimos que, tras ser insertados en la base de datos, se enviarán al cliente.

Se ha detallado en este capítulo pues, el modelo matemático junto a su utilización mediante Cplex en el algoritmo de cálculo de rutas. En el siguiente capítulo se mostrarán los pasos a seguir para preparar el entorno de ejecución de la PoC, el funcionamiento principal de la cual, ha sido mostrado.

6. Implantación de la solución

En este capítulo se hablará acerca del proceso de lanzamiento del sistema, suponiendo que se cuenta con un *backup* del proyecto y que se tiene instalado Visual Studio Code y Eclipse en nuestro ordenador. La versión de Eclipse instalada en el equipo deberá ser la Eclipse Jee 2019-09. Dicha versión cuenta con un Maven embebido para la gestión de las dependencias. Se considerará que se han configurado debidamente dichas dependencias necesarias con Maven para llevar a cabo el lanzamiento de la parte correspondiente al *backend* del proyecto.

Se empezará detallando el proceso de puesta en marcha del servidor de base de datos, siguiendo con el *backend*, que realizará las conexiones a la base. Finalmente se mostrará como lanzar la aplicación cliente que se conecta a dicho servidor REST o *backend*.

6.1 Instalación de PostgreSQL y pgAdmin:

Lo primero que se debe hacer es realizar la instalación de PostgreSQL. Esto lo conseguiremos descargando el paquete binario PostgreSQL 10.5 (o una versión semejante más actualizada) en nuestro disco C. Obtendremos el paquete de la página oficial de descargas de PostgreSQL [37]. Una vez hecho esto y tras haber descomprimido el paquete, se cambiará el nombre de la carpeta a psql10 y se creará en ella una carpeta *data*. Seguidamente se procederá a la instalación.

Se deberá abrir una terminal y desplazarse a la carpeta C:\pgsql10\data. Una vez allí, ingresaremos el siguiente comando para realizar la instalación:

initdb.exe -U postgres -A password -E utf8 -W -D C:\pgsql10\data

Lo siguiente será ingresar la clave de ‘superusuario’ o administrador, la cual será utilizada más adelante.

Tras hacerlo, se nos crearán en nuestra carpeta los archivos necesarios para la utilización de PostgreSQL. Entre dichos archivos estarán los denominados archivos de registro, que servirán para levantar el servidor, iniciando o deteniendo los servicios.

6.1.1 Lanzamiento de PostgreSQL:

Para llevar a cabo el lanzamiento o inicio del servidor de PostgreSQL deberemos situarnos otra vez en el directorio C:\pgsql10\data, creado en el apartado de instalación. Una vez ahí, para iniciar los servicios de PostgreSQL se deberá ingresar el siguiente comando:

pg_ctl -D ^"C^:^\pgsql10^\data^" -l archivo_de_registro start

Seguidamente, se deberá ejecutar la aplicación de pgAdmin instalada, de la cual se habrá tenido que descargar la versión más actualizada en la web de descargas oficial [38]. Se nos abrirá el gestor de bases en nuestro navegador. Ahora pues, agregaremos un servidor PostgreSQL10 en el

Dashboard[19] de PGAdmin4. Recordemos que la contraseña del usuario postgres es la que se ha introducido anteriormente.

Una vez iniciados los servicios, si lo que interesa es detenerlos, se ingresará el siguiente comando:

pg_ctl -D ^"C^:^\pgsql10^\data^" -l archivo_de_registro stop

Figura 61: Captura del proceso de inicio de los servicios.
Fuente: Terminal de mi ordenador en Capgemini.

Ahora que se ha configurado correctamente el entorno con lo necesario para trabajar con PostgreSQL y pgAdmin, lo siguiente será configurar la base de datos de la PoC.

6.2 Preparación de la base de datos

Para poder trabajar con la base de datos del proyecto podrá hacerse uso, si se dispone de él, de un *backup* o copia de dicha base. Si lo anterior no es posible, deberán fusionarse los esquemas de la estructura de la base y el de inserción de datos antes de lanzar el servidor. Es decir, deberá crearse la estructura para luego proceder a la inserción de los datos.

El segundo procedimiento, todo y que resulta un poco más engorroso, facilitará el proceso de pruebas con distintos datos de la base. Se podrá rellenar una misma estructura seleccionando uno u otro fichero con los datos de interés.

A continuación, se detallarán los pasos a seguir en cada uno de los casos de configuración mencionados anteriormente.

6.2.1 Restaurar un backup de la base

Si se dispone de una copia o *backup* de la base en cuestión, se deberá copiar el archivo .sql en el directorio de ruta C:\pgsql10\bin. Una vez hecho esto, habiendo lanzado previamente el servidor creado desde pgAdmin, recuperaremos dicha base de datos.

Para ello, se deberá crear una nueva base de datos en el servidor anteriormente creado, dándole el nombre que se desee. Hecho esto, mediante la función *Restore*, se recuperará la base, seleccionando para ello el archivo .sql situado en el directorio dicho anteriormente. Después de hacer esto, ya se tendrá la base de datos accesible desde pgAdmin.

[19] *Dashboard*: Panel de herramientas o instrumentos.

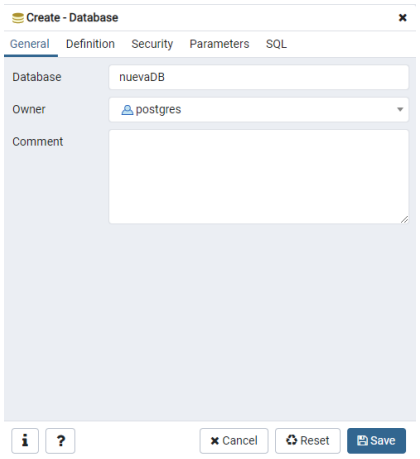

Figura 62: Captura del proceso de creación de la nueva DB. Fuente: servidor PostgreSQL en Capgemini.

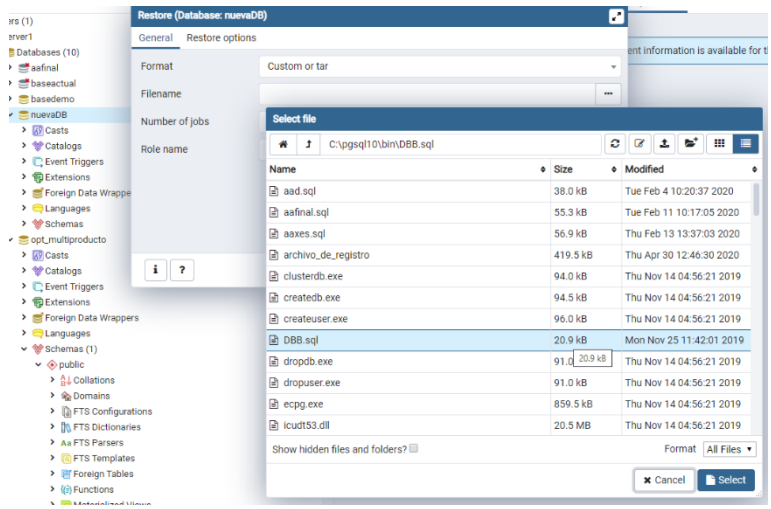

Figura 63: Captura del proceso de restauración de la DB. Fuente: servidor PostgreSQL en Capgemini

6.2.2 Generar la base a partir de la estructura y los datos a insertar

Una vez instalado y configurado todo lo referente a PostgreSQL y pgAdmin, accederemos al pgAdmin y crearemos una base de datos, por ejemplo "TEST".

El siguiente paso es generar la estructura de la base (las tablas, relaciones y funciones). Para ello se deberá acceder a la CMD o terminal (como en la instalación de postgresql) y ejecutar el siguiente comando en la carpeta *bin*:

psql -h localhost -f Generated.sql TEST

Con esto ya tenemos la estructura (la cual se construirá a partir de comandos DDL[20]) de la base "TEST". Ahora deberemos insertar los datos (mediante instrucciones DML[21]). Para ello deberemos disponer pues, de un archivo *dump.sql* que contenga los datos a insertar. Ahora pues, se deberá introducir el siguiente comando en la CMD:

psql TEST<dump.sql

Al usar el comando anterior, se nos pedirá la contraseña de 'superusuario' de inicio de pc, cosa que nos dará error, debido a que no se dispone de los privilegios de objeto referentes a la base y necesarios para realizar la inserción. Para solucionar este problema, haremos lo siguiente en una consulta del pgAdmin:

CREATE USER ana WITH PASSWORD

'myPassword';

GRANT ALL PRIVILEGES ON DATABASE laBase to ana;

[20] DDL: Data Definition Language (Lenguaje de definición de datos). Incluye las instrucciones de definición y modificación de las estructuras de objetos de almacenamiento de datos. (CREATE, DROP y ALTER) [61].

[21] DML: Data Modeling Language (Lenguaje de manipulación de datos). Incluye las instrucciones de inserción, consulta y modificación de datos de la base de datos [62].

Donde en vez de ‘ana’ pondremos nuestro nombre de usuario del pc, y donde ponga ‘laBase’ nuestra base de datos, en este caso ‘TEST’. En ‘myPassword’ le pondremos la contraseña que deseemos. Con esto se solucionará el error de la *password*, y tras ejecutar otra vez el comando de inserción de datos, se realizará correctamente.

6.3 Lanzamiento del servidor o backend

Se deberá tener instalada en nuestro equipo, la versión de Eclipse dicha al inicio del capítulo (Eclipse IDE for Enterprise Java Developers Version: 2019-09 R 4.13.0). Dicha versión podrá ser descargada desde la web oficial de Eclipse [39].

Una vez instalado y abierto el entorno, se configurará un *workspace* en el cual se trabajará a partir de ahora. Lo siguiente será abrir la carpeta correspondiente al *backend* de la PoC con *File – Open Projects from File System* y seleccionándola de nuestro sistema de archivos. Dicho directorio tiene como ruta ‘Devon-dist_2.3.1_Linux/workspaces/back/poc’ y está contenido en la carpeta comprimida que contiene el *backend* de la PoC, accesible desde el repositorio SVN de la empresa.

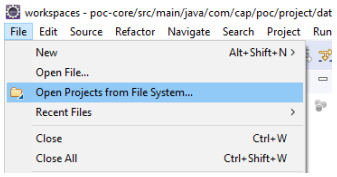

Figura 64: Abrir proyecto existente en Eclipse. Fuente: Backend PoC Capgemini.

Lo siguiente a hacer será lanzar el servidor. Para llevar a cabo su lanzamiento, durante el proceso de creación de la estructura base del proyecto con Devon, se automatizó el proceso haciendo uso de Spring Boot. Para ello, se definieron las dependencias correspondientes en el archivo *pom.xml* y se configuraron los parámetros necesarios en el archivo de configuración de Spring, pero no voy a entrar en detalle en este tema, ya que no estuve implicado en el proceso y este es bastante tedioso.

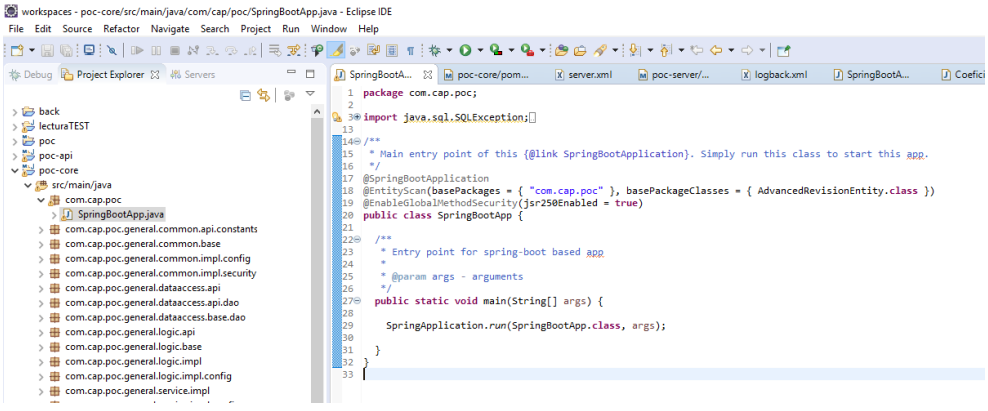

Figura 65: Captura del fichero SpringBootApp.java del backend. Fuente: Backend PoC Capgemini.

Bastará pues, en situarse en el archivo ‘SpringBoot.java’ (el cual puede verse en la figura anterior) situado en el siguiente directorio:

poc-core/src/main/java/com/cap/poc/SpringBootApp.java

Se ejecutará dicho archivo con *Run As - Java Application*. Con esto, el servidor se levantará, en el puerto indicado en el archivo de configuración de Spring *application.properties*, exponiendo los distintos servicios que se implementan en él. Estos servicios serán accesibles desde el *frontend* como bien se explicó en puntos anteriores. Algunos de ellos pueden verse en la siguiente figura.

```
@POST
@Path("/setHorario/")
public void setHorario(String obj)
    throws JSONException, SQLException, NumberFormatException, ClassNotFoundException;

@GET
@Path("/routes/")
public List<Object> getRoutes() throws JSONException, SQLException;

@POST
@Path("/coeficientes/")
public List<Object> setCoeficientes(String obj) throws ClassNotFoundException, SQLException, JSONException;

@POST
@Path("/ponerCoeficientes/")
public void ponerCoeficientes(String obj) throws ClassNotFoundException, SQLException, JSONException;

@GET
@Path("/algorithms/")
public List<Algorithm> getAlgorithms() throws JSONException, SQLException;
```

Figura 66: Captura de algunos de los servicios expuestos por el backend en VisitorRestService.java. Fuente: Backend PoC Capgemini

6.4 Preparación y lanzamiento del cliente

6.4.1 Instalación Node.js

Lo primero que se tendrá que hacer para poder lanzar la parte cliente de la prueba es instalar los entornos, lenguajes y obtener las librerías o módulos necesarios. Para ello, se presupone que tenemos instalado en nuestro ordenador Visual Studio Code, el cual es gratuito y se puede descargar directamente desde la página web oficial [51]. Se deberá descargar Node.js si no se tiene instalado en el equipo. Para esto hará falta entrar a su web oficial [40] y descargar el instalador de la versión 10.16.3, si no, el más actualizado. Al instalar Node.js, se instalará también por defecto el gestor de paquetes npm.

Lo siguiente será comprobar qué versión se nos ha instalado haciendo uso del comando 'node -v'. Si la versión de la que se dispone no es la 10.16.3, se deberá instalar dicha versión. Para ello, se podrá seguir la siguiente secuencia de comandos, como bien nos propone Eborio Linárez [41]:

$ sudo npm cache clean -f

$ sudo npm install -g n

$ sudo n 10.16.3

Durante este proceso y en instantes posteriores podrán surgir problemas referentes a vulnerabilidades por dependencias. Para fijar algunas de las vulnerabilidades por dependencias que puedan surgir se podrá hacer uso del comando 'npm audit fix'.

Si nunca se ha trabajado antes en ningún otro proyecto Angular, ni se tiene instalado TypesScrypt, lo más probable es que no se tengan instalados la mayoría de los paquetes necesarios para lanzar la aplicación. En esta situación, se deberían instalar desde 0 todos los módulos, *frameworks* y librerías necesarias en vez de configurar las versiones.

Lo dicho anteriormente, aunque no lo parezca, es un proceso relativamente fácil. Bastará con situarse en la carpeta raíz del proyecto *frontend* (con la terminal de VS Code, o una Node.js) y

teclear el comando 'npm install'. Dicho comando configurará automáticamente todas las dependencias listadas en el fichero *package.json* del proyecto e instalará todo lo necesario para poder lanzarlo.

En caso de haber programado anteriormente en Angular, se deberán tener las versiones específicas de Angular, Angular CLI y Angular Material instaladas. En los siguientes puntos se detallará como configurar las versiones correctas de estos.

6.4.2 Configurar versión de Angular y Angular CLI

Para poder llevar a cabo el lanzamiento del cliente de la PoC deberá instalarse la versión correspondiente de Angular CLI. Dicha herramienta, como bien dice Alberto Basalo en su tutorial de introducción y primeros pasos con Angular CLI, publicado en Academia Binaria[22] [42], permitirá depurar, crear y publicar aplicaciones Angular desde la línea de comandos. En este caso, se instalará la versión 1.7.4.

Al instalar la versión dicha en el párrafo anterior, se instalará con ella la versión 5.2.10 de Angular en nuestro equipo. Como se observa, son versiones antiguas de Angular, pero fueron con estas con las que se desarrollaron las primeras versiones del cliente de la PoC.

Si se tiene instalado ya Angular CLI, pero no se tiene seguridad de que versión se dispone, lo primero será desinstalar-lo con:

$ npm uninstall -g @angular/cli

Como siguiente paso, será necesario limpiar la memoria cache e instalar la versión deseada de Angular CLI. Se ejecutará para ello pues, la siguiente secuencia de comandos:

$ npm cache clean

$ npm install -g @angular/cli@1.7.4

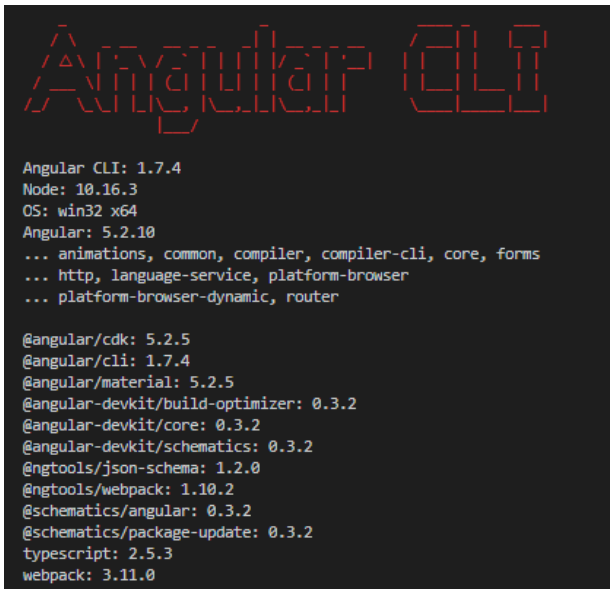

Figura 67: Captura ilustrativa de las distintas versiones de Angular y Node necesarias.

[22] Academia Binaria: Web de formación online con tutoriales y artículos interesantes sobre TypeScript, Java y Angular [64].

Puede que, durante este proceso, al consultar la versión de Angular instalada con 'ng --version', no aparezca actualizada. Para solucionar esto, cerraremos y volveremos a abrir Visual Studio.

Todos los comandos necesarios para la instalación de Angular y Angular CLI fueron indicados en algunos de los tutoriales de formación inicial en la empresa, accesibles desde Pluralsight [66]. Así también, puede consultarse la documentación de Angular [43]. Además, se pueden ampliar conocimientos con algunas publicaciones acerca de dicho proceso de instalación y actualización de versiones [44].

6.4.3 Configurar versión de Material

Se deberá comprobar, si se tiene instalada una versión de material, con el comando 'npm angular/material -v'. Si esta es la 6.9.0, no será necesario hacer nada. Si no es así, desinstalar, si se tiene, la versión de material existente en el pc con 'npm uninstall @angular/material'. Seguidamente con 'npm install @angular/material@6.9.0' se instalará la versión que se necesita.

El conocimiento necesario para poder llevar a cabo la instalación de una versión especifica de Material fue obtenido de distintos hilos de StackOverflow.com, accesibles en las referencias [45] [46].

6.4.4 Lanzamiento del frontend

Una vez instalado y configurado todo el entorno relativo a la parte del cliente, el siguiente paso será lanzarlo. Para ello, se realizarán los pasos descritos a continuación.

Lo primero, será cambiar del archivo *config.ts*, la ip que aparece a la de nuestro equipo (la de la vpn a la que estemos conectados o la de nuestra red wifi o por cable).

```
src > app > TS config.ts > ...
1   export const config: any = { basePath: 'http://192.168.1.43:8081/services/rest/' };
2
```

Figura 68: Variable basePath donde se define la ip. Fuente: Código del frontend de la PoC (Capgemnini).

Lo segundo, será situarse en la carpeta *client*, y lanzar el *frontend*. Angular lanzará el cliente por defecto en el puerto 4200. Para ello, se usará el comando: 'ng serve -o --host NUESTRA_IP'

Es posible que debido a problemas con la versión de Visual Studio Code, no se reconozca el anterior comando. Si es así, lo introduciremos, situándonos en la carpeta *client*, desde una terminal Node.js, donde sí que será posible ejecutarlo. Automáticamente, tras la compilación y lanzamiento del cliente, se abrirá una ventana en el navegador en la que se mostrará la pantalla principal de la prueba de concepto, como bien puede verse en la siguiente figura.

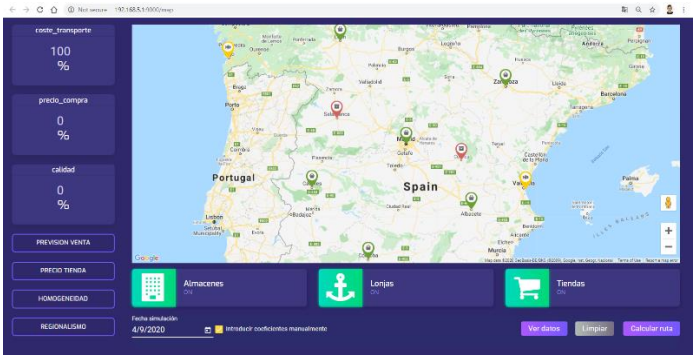

Figura 69: Pantalla inicial de la PoC. Fuente: PoC Optimización de Redes (Capgemini).

7. Pruebas

Una vez se han puesto en marcha las distintas partes de la prueba de concepto, es el momento de comprobar que esta funciona correctamente. Comprobar el funcionamiento completo del proyecto excedería los límites de extensión del TFG, ya que deberían considerarse multitud de casos e incluir numerosas capturas. Para ello, se detallará en este capítulo algunas de las pruebas realizadas, comprobando la validez de los datos y la correcta actualización de la base.

7.1 Pruebas sobre un cálculo sencillo de rutas

Partiendo de la pantalla principal de la prueba de concepto, se procede a realizar una prueba sencilla de cálculo de rutas. Para ello, se otorga el total del peso de los coeficientes a la variable 'calidad'. Con esto, se busca obtener las rutas óptimas teniendo en consideración principal, la calidad de los productos ofertados por los proveedores. Se espera pues, que las tiendas se abastezcan con aquellos proveedores que ofrezcan los productos que estas demandan de mayor calidad.

Después de esto (aunque podría haberse visualizado anteriormente), visualizamos los datos de la tabla de la tienda que queramos tener en consideración a la hora de realizar la prueba. A modo de ejemplo, escogemos la tienda de Madrid.

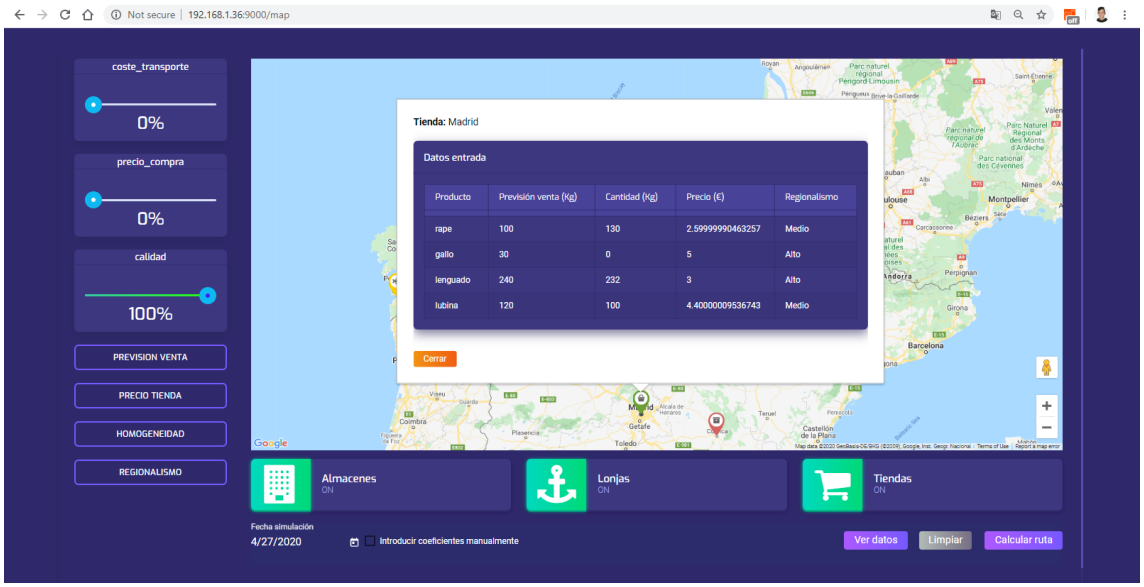

Figura 70: Captura de pantalla de la PoC que ilustra los pasos dichos en el párrafo anterior. Fuente: PoC Optimización de Redes (Capgemini).

Vemos, al pulsar sobre el marcador correspondiente a la tienda de Madrid, un pop-up, el cual incluye una tabla con la información correspondiente a dicha tienda. Se observa que esta tiene una demanda de lenguado de 232 kg, la cual, en el caso de que la oferta total de lenguado ofrecida por los proveedores supere o sea igual a la demanda total de las tiendas, debería verse abastecida en su totalidad.

Seguidamente, se observa la información correspondiente a cada uno de los proveedores. Para ello, al igual que en el caso anterior, pulsamos sobre los marcadores correspondientes.

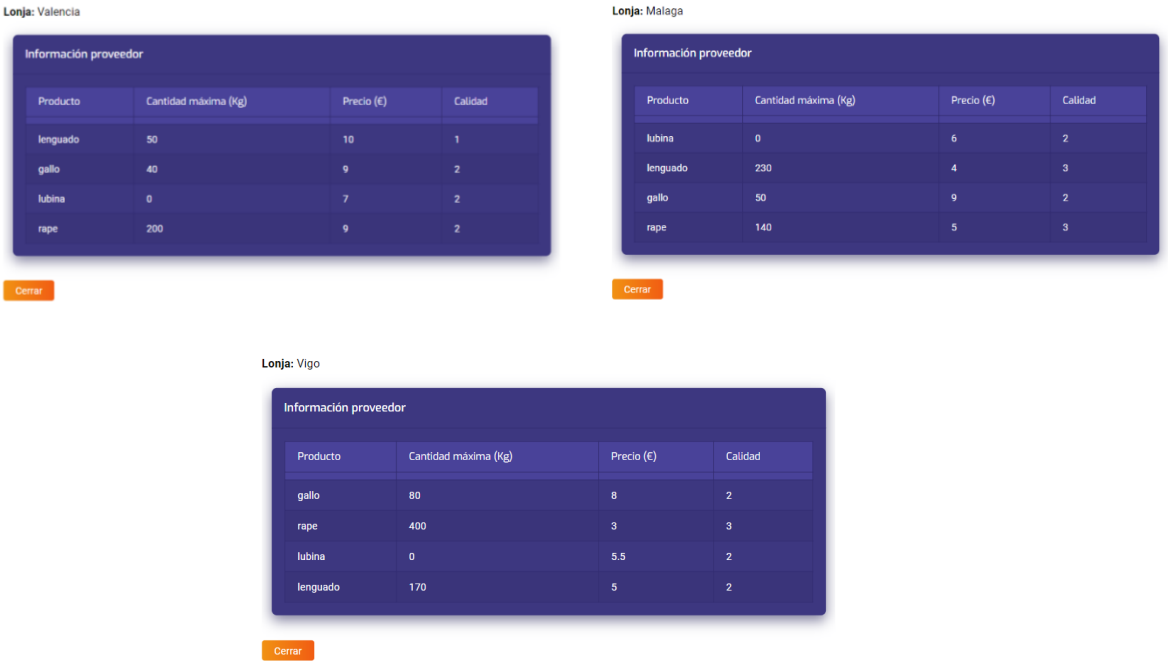

Figura 71: Composición de capturas de la información de las lonjas mostrada en las tablas. Fuente: PoC Optimización de Redes (Capgemini).

Una vez visualizadas las tablas (mostradas en la figuran anterior), conviene centrarnos en la calidad del lenguado ofrecido por cada uno de los anteriores proveedores. Como se ha dicho, la tienda en cuestión buscará abastecer su demanda de lenguado de aquellos proveedores que lo ofrezcan con una calidad mayor. En caso de que dicho proveedor no tenga suficiente oferta, la tienda adquirirá el producto de otros proveedores. Se observa que Málaga ofrece el lenguado a un nivel 3 de calidad, seguida por Vigo, a un nivel 2 y finalmente Valencia (nivel 1). Tendrá que probarse pues, que Madrid cubrirá, total o parcialmente, su demanda, abasteciéndose, a ser posible, de aquellas lonjas de más calidad.

Iniciamos pues, el procedimiento de cálculo de ruta. Al pulsar el botón `Calcular ruta`, el *frontend* recuperará los coeficientes introducidos por pantalla por el usuario y solicitará, mediante el servicio de cálculo de rutas óptimas, la información correspondiente al *backend*. Al comprobar la tabla porcentajes, haciendo uso de pgAdmin, se observa que efectivamente ha sido almacenado el dato referente al 100% de calidad a tener en cuenta en el cálculo de las rutas.

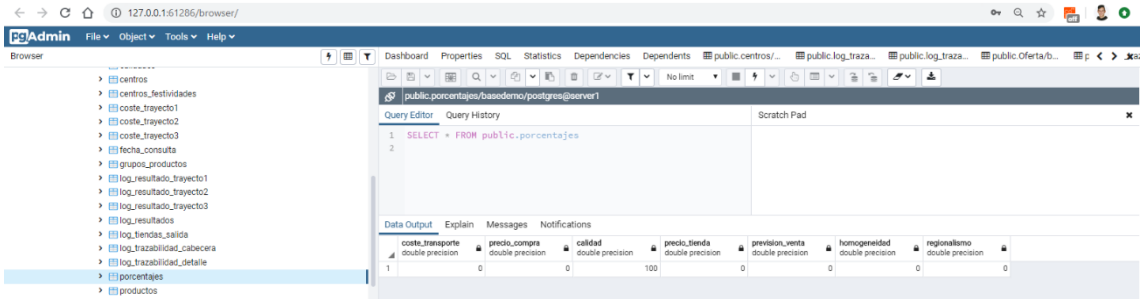

Figura 72: Captura de la tabla porcentajes después de la introducción de datos. Fuente: BD PoC Optimización de Redes (Capgemini).

Una vez realizado el cálculo, el *backend* actualizará como es debido los registros correspondientes de la base de datos y devolverá la información pertinente al *frontend*. Este la utilizará para dibujar las distintas rutas haciendo uso del 'renderizador' de direcciones del API de *Google Maps*, como bien puede verse en la siguiente figura. Además, se cargarán en las distintas tablas la información correspondiente a los resultados del cálculo de rutas.

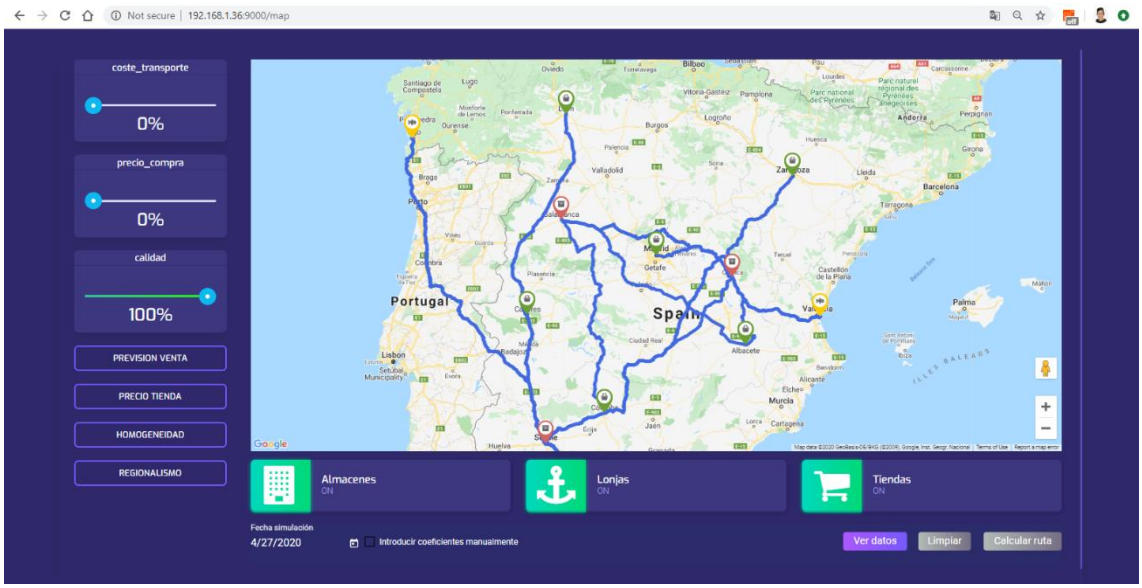

Figura 73: Captura del resultado de cálculo de rutas de la PoC.
Fuente: PoC Optimización de Redes (Capgemini).

Para observar de mejor forma la información recuperada tras el cálculo de rutas, se visualizarán los resultados en la ventana de datos. Haciendo esto, obtendremos la información necesaria para comprobar que la solución obtenida tiene sentido y que Madrid, tienda escogida como referencia, se abastece correctamente.

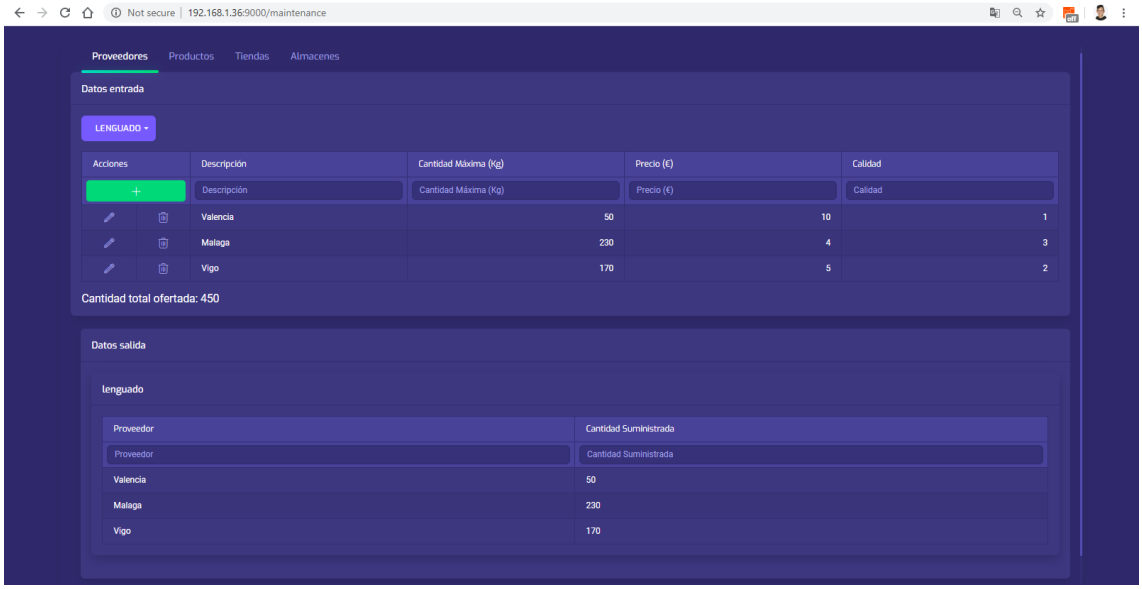

Figura 74: Captura de la información de los proveedores de la PoC.
Fuente: PoC Optimización de Redes (Capgemini).

Observamos en la figura anterior que la cantidad total ofertada de lenguado es de 450 kg. Cada proveedor abastece el máximo permitido en cada caso. Esto puede ser debido a que la demanda

total de lenguado por parte de las tiendas coincide exactamente con la oferta total, o que la demanda de dicho producto es superior a lo que pueden ofrecer los proveedores.

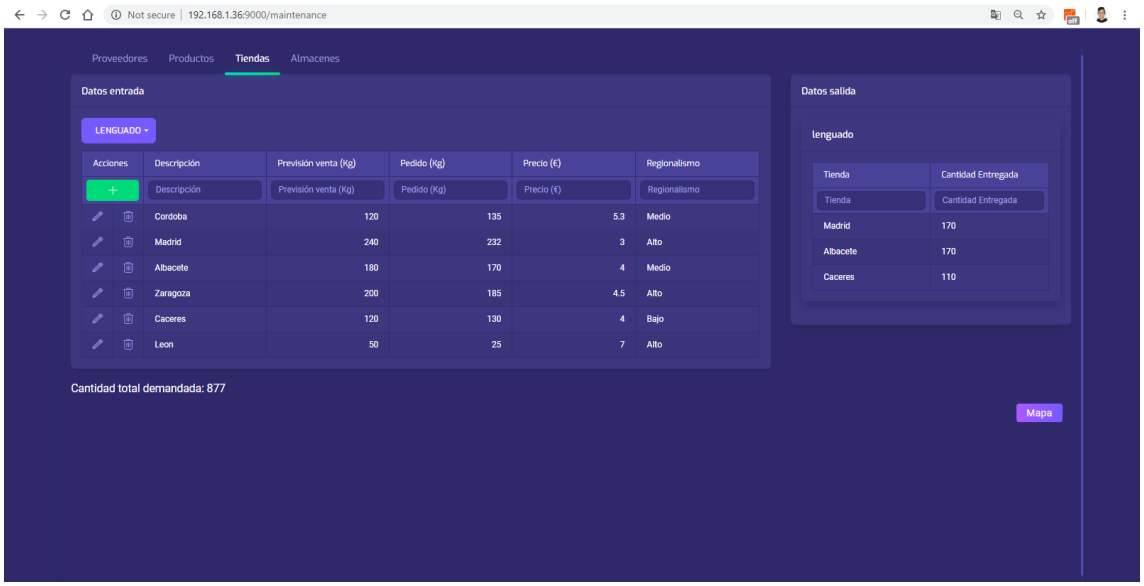

Figura 75: Captura de la información de las tiendas de la poC.
Fuente: PoC Optimización de Redes (Capgemini).

Vemos pues, en la anterior figura, que la demanda total de lenguado es de 877 kg, superando los 450 kg que podían ofrecer las 3 lonjas conjuntamente. Debido a esto, algunas de las tiendas no serán abastecidas completamente. Podemos ver como solo 3 de las 6 tiendas reciben mercancía, ya que la variable a tener en cuenta era la calidad de los proveedores, por esto, no se ha buscado un abastecimiento repartido. Además, Madrid no abastece su demanda completa, recibiendo solo 170kg. Observamos en la siguiente figura, viendo la tabla de datos de salida correspondiente a la tienda de Madrid, que esta tampoco recibe el lenguado del proveedor de mayor calidad.

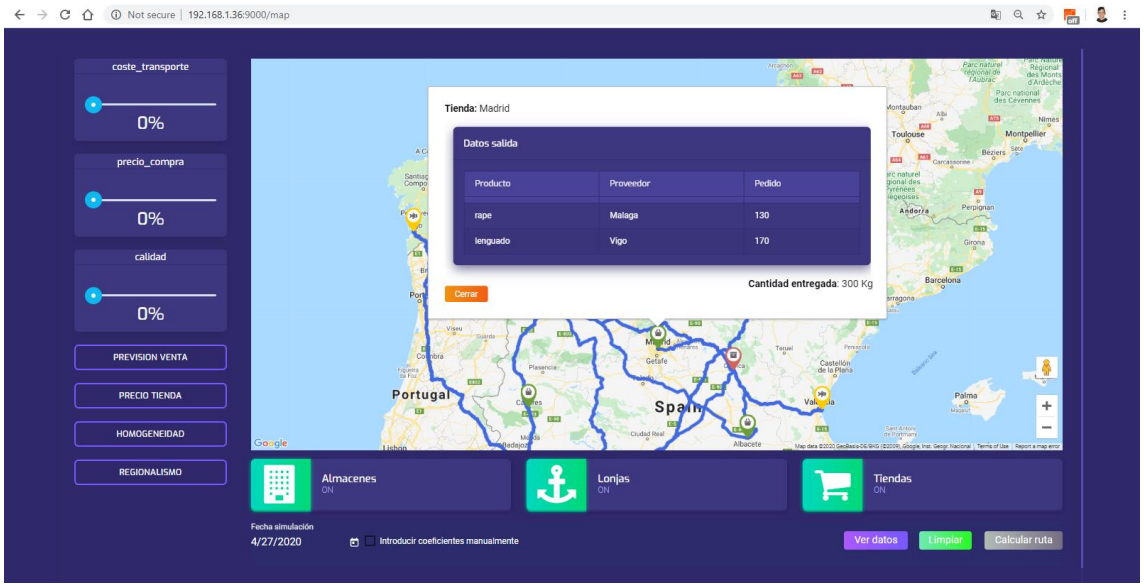

Figura 76: Captura de la información de abastecimiento de Madrid.
Fuente: PoC Optimización de Redes (Capgemini).

Se observa pues, que Madrid recibe lenguado de Vigo, por lo tanto, en la tabla de datos de salida de dicho proveedor debería aparecer esta información. Efectivamente, los 170kg recibidos en la tienda de Madrid aparecen registrados, como se muestra en la siguiente figura.

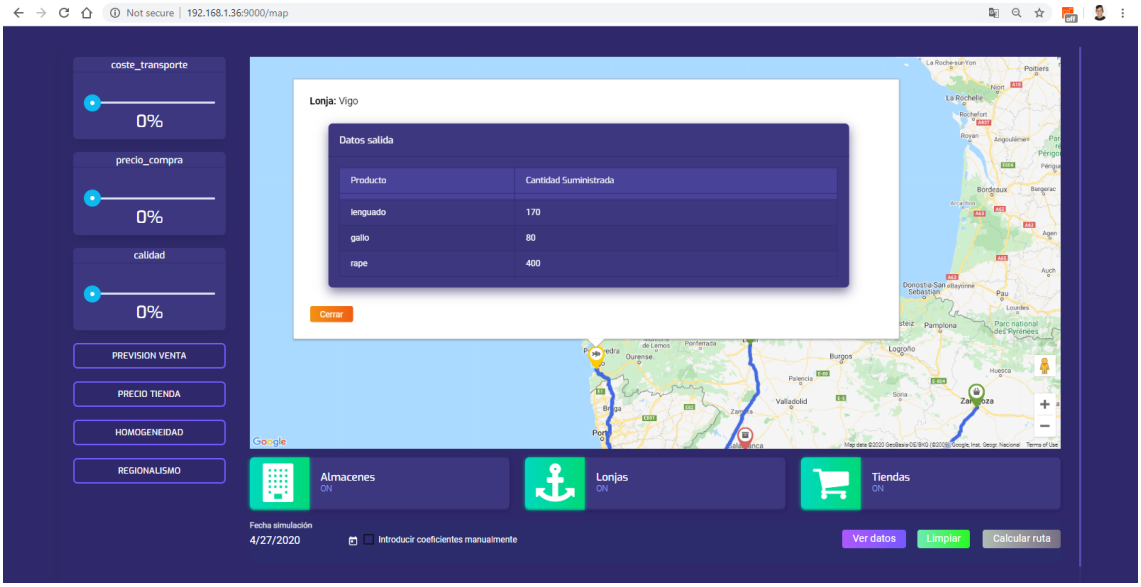

Figura 77: Captura de la información de suministro de salida de Vigo. Fuente: PoC Optimización de Redes (Capgemini).

Se puede ver también, en la figura anterior, como Vigo únicamente abastece a Madrid. Anteriormente, se ha visto que 3 de las tiendas totales no recibían lenguado, por lo que, de ampliar la oferta en algunos de los proveedores, tal vez llegase algo a dichas tiendas. Procedemos pues a realizar dicha modificación en la base.

7.2 Modificando la base

Como paso previo a la modificación de la base con pgAdmin, deberá cerrarse la conexión del *backend* a esta. Después, se detendrá la aplicación *frontend*, ya que, de realizar alguna consulta al *backend*, esta no obtendría resultados. Detenemos las dos partes como se muestra en las siguientes figuras.

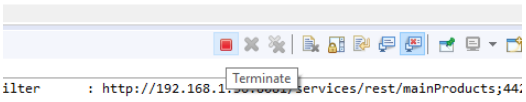

Figura 78: Cierre de conexión del backend. Fuente: Backend de la PoC (Capgemini).

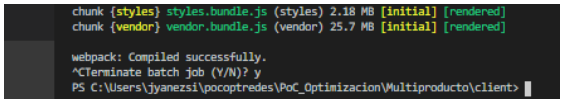

Figura 79: Cierre de conexión del frontend. Fuente: Frontend de la PoC (Capgemini).

Seguidamente, se deberá saber cuál es el código referente al proveedor de Vigo, así como el código identificativo del lenguado. Dicha información no es obtenible desde la tabla Oferta, la cual se quiere modificar. Para ello, se consultarán las tablas relativas a los centros y a los productos. Puede verse esta información en las dos figuras siguientes.

Data Output | Explain | Messages | Notifications

	cod_centro [PK] integer	tipo_centro integer	descripcion character (10)	posicion_x double precision	posicion_y double precision
1	4001	1	Valencia	39.4480857849121	-0.316926687955856
2	4002	1	Malaga	36.7096252441406	-4.42206716537476
3	4003	1	Vigo	42.2311210632324	-8.74261093139648
4	5001	2	Salamanca	40.970100402832	-5.66353988647461
5	5002	2	Sevilla	37.3890991210938	-5.98445987701416
6	5003	2	Cuenca	40.0703926086426	-2.13741612434387
7	6002	3	Madrid	40.4168014526367	-3.70378994941711
8	6004	3	Zaragoza	41.6487998962402	-0.889084994792938
9	6005	3	Caceres	39.4752998352051	-6.37241983413696
10	6003	3	Albacete	38.9944000244141	-1.85854005813599
11	6006	3	Leon	42.5987281799316	-5.56709575653076
12	6001	3	Cordoba	37.8881988525391	-4.77937984466553

Figura 80: Captura de la table 'centros' en pgAdmin. Fuente: BD de la PoC (Capgemini).

Data Output | Explain | Messages | Notifications

	cod_producto [PK] integer	descripcion text	caducidad integer
1	105	rape	4
2	106	gallo	2
3	107	lenguado	3
4	108	lubina	3

Figura 81: Captura de la tabla 'productos' en pgAdmin. Fuente: BD de la PoC (Capgemini).

Se sabe ahora, habiendo visualizado las correspondientes tablas, que el código referente a Vigo es el 4003 y que el número 107 identifica al rape. Ahora pues, deberá modificarse la fila de la tabla Oferta, que incluya ambos códigos entre sus atributos. Lo que se trata es de aumentar la oferta de lenguado de Vigo. Se augmentará pues, la oferta de lenguado de Vigo hasta 600kg, asegurando así que la oferta total de dicho producto resultante (880 kg), sea superior a la demanda total de las tiendas (877 kg). Para ello, se modifica la tabla Oferta como se ve en la siguiente figura.

basedemo/postgres@server1

Query Editor | Query History

```
1  UPDATE public."Oferta"
2      SET cantidad_prod=600
3      WHERE cod_proveedor=4003 AND cod_producto=107;
```

Data Output | Explain | Messages | Notifications

```
UPDATE 1

Query returned successfully in 91 msec.
```

Figura 82: Captura del 'update' de modificación de la oferta de Vigo en pg Admin. Fuente: BD de la PoC (Capgemini).

Una vez hecho esto, se relanza el *backend*, seguido del *frontend* y se vuelve a visualizar la pantalla de la PoC. Podrá comprobarse entonces que, en la tabla de información correspondiente a Vigo, la oferta de lenguado será ahora 600 kg, como bien puede verse en la figura siguiente.

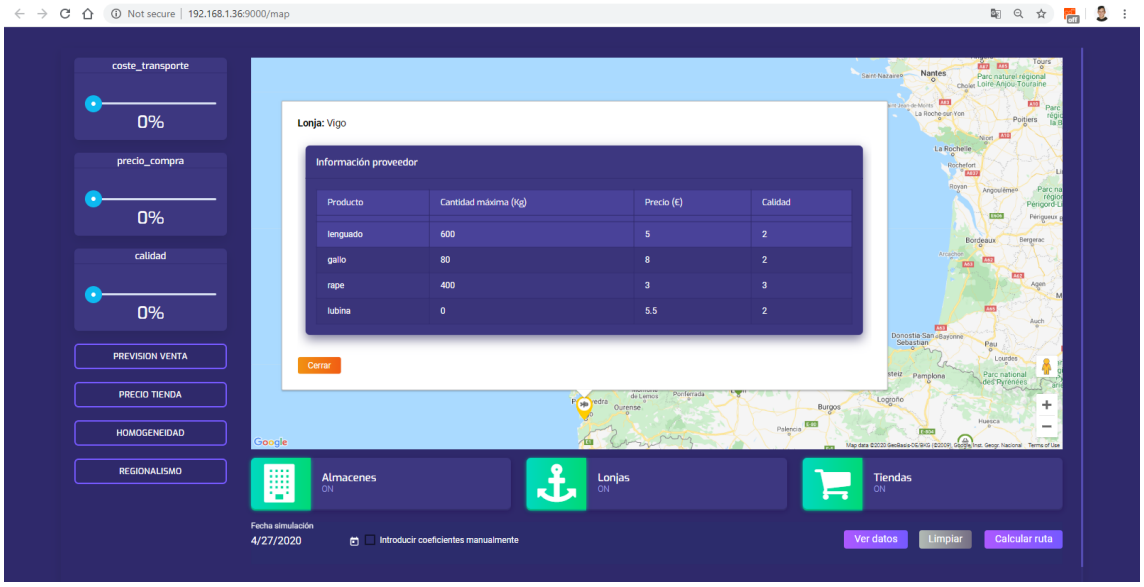

Figura 83: Captura de la información de suministro de entrada de Vigo.
Fuente: PoC de Optimización de Redes (Capgemini).

Tras realizar el cálculo óptimo de rutas de forma equivalente al realizado anteriormente, obtenemos un nuevo conjunto de rutas óptimas. Ahora deberemos comprobar en la ventana de datos, si existen nuevas tiendas que reciben mercancía tras la modificación realizada en la oferta. Consultando la información de las tiendas tras el cálculo de rutas obtenemos los resultados mostrados en la siguiente figura.

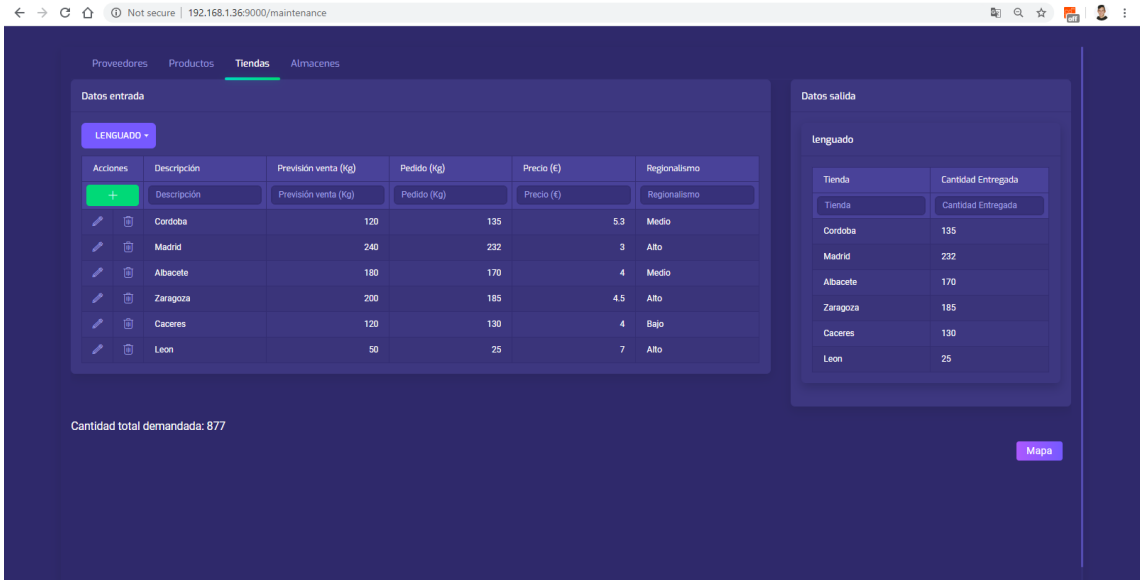

Figura 84: Captura de la información de las tiendas tras la modificación.
Fuente: PoC de Optimización de Redes (Capgemini).

Como se observa, cada una de las tiendas ve abastecida su demanda. Esto es debido a que la oferta de los tres proveedores es mayor a la demanda de las tiendas. Gracias al incremento realizado en la oferta de Vigo, vemos (en la figura mostrada a continuación) como en nuestra tienda de referencia (Madrid), se abastece la demanda.

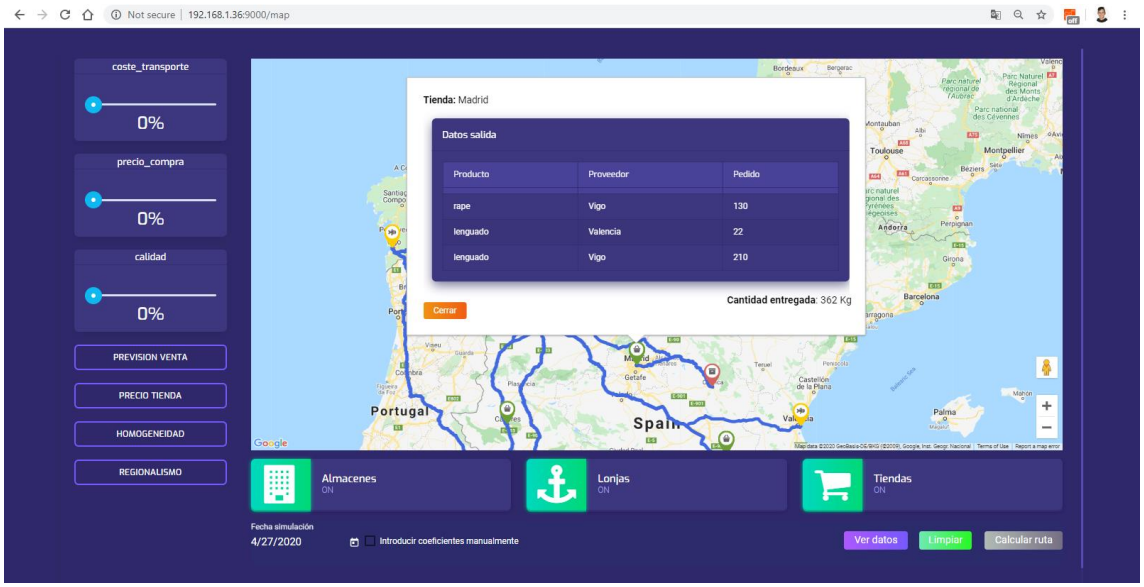

Figura 85: Captura de la información de abastecimiento de Madrid.
Fuente: PoC de Optimización de Redes (Capgemini).

Aun así, se observa que dicha demanda no llega a ser abastecida por Málaga, el proveedor que ofrece el lenguado a mayor calidad. Esto es debido a que, al terminarse la cantidad ofertada de este producto en el proveedor, las tiendas restantes tienen que ser abastecidas por el siguiente con más calidad. Puede verse también (en la siguiente figura) como Córdoba, tienda más cercana a Málaga, se abastece de lenguado proveniente de este.

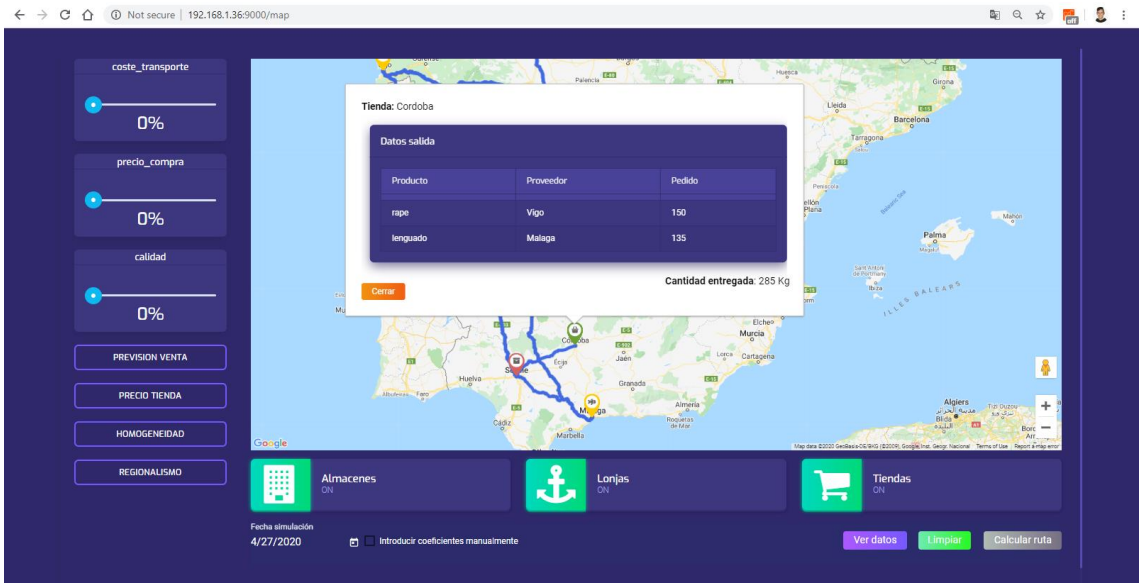

Figura 86: Captura de la información de abastecimiento de Córdoba.
Fuente: PoC de Optimización de Redes (Capgemini).

¿Como estamos probando entonces que el principal factor a tener en cuenta es la calidad del proveedor? Para ello, se augmentará la oferta de lenguado en Málaga a 600, dejando la de Vigo también a esta cantidad. Con esto se comprobará que, teniendo un caso en el que la oferta sea mayor a la demanda, se escogen prioritariamente aquellos proveedores con más calidad en los productos.

Se realiza pues, un *update* en la tabla de Oferta. Asignaremos una cantidad de lenguado disponible de 600 kg en el proveedor de Málaga, el cual se identifica con el código 4002. Tras realizar la modificación, se tienen 600kg de oferta de lenguado tanto en Vigo como en Málaga, como bien se puede ver en la siguiente figura.

Data Output | Explain | Messages | Notifications

	cod_proveedor integer	cod_producto integer	cantidad_prod integer	precio double precision	calidad integer
1	4001	107	50	10	1
2	4001	106	40	9	2
3	4002	106	50	9	2
4	4003	106	80	8	2
5	4001	108	0	7	2
6	4002	108	0	6	2
7	4003	108	0	5.5	2
8	4002	105	140	5	3
9	4001	105	200	9	2
10	4003	105	400	3	3
11	4003	107	600	5	2
12	4002	107	600	4	3

Figura 87: Captura de la tabla 'Oferta' en pgAdmin.
Fuente: BD PoC de Optimización de Redes (Capgemini).

Ahora, tras volver a realizar el proceso de lanzamiento y proceder con el cálculo de rutas óptimas, obtendremos lo mostrado en la figura incluida a continuación. En la ventana de datos de los proveedores, vemos como se utilizan únicamente Málaga y Vigo para satisfacer la demanda de las tiendas.

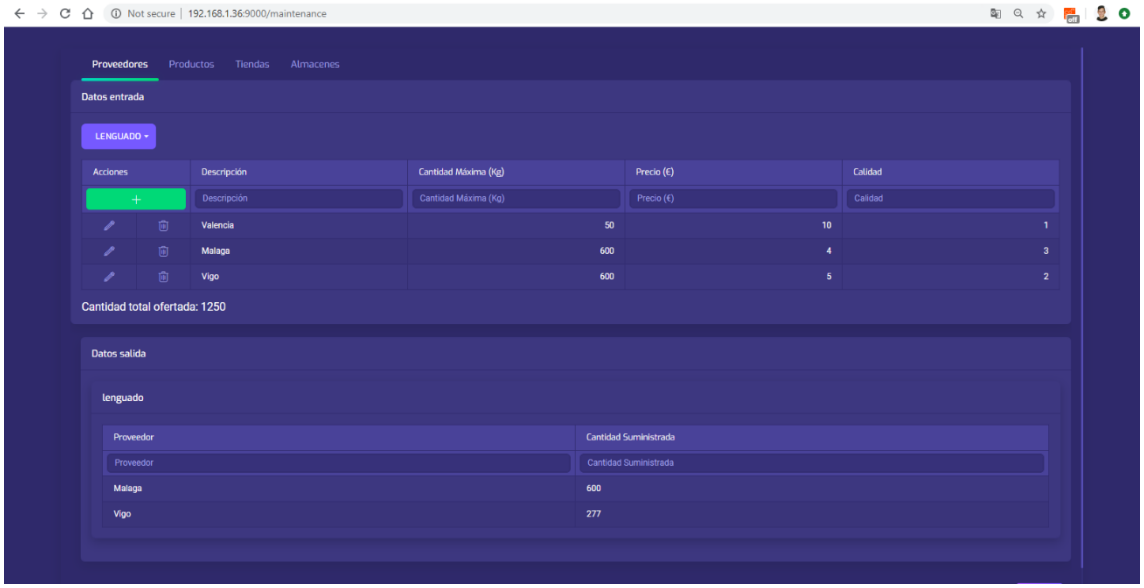

Figura 88: Captura de la información de los proveedores.
Fuente: PoC de Optimización de Redes (Capgemini).

Como bien se ha mostrado al inicio, Valencia era el proveedor con menor calidad de lenguado, es por esto, que el algoritmo no lo tiene en consideración. Si la demanda fuese mayor, como en casos anteriores, el algoritmo no tendría más remedio que considerar el abastecimiento de lenguado desde Valencia. Para lo dicho anteriormente, los proveedores con mayor calidad del producto tendrían que haber agotado su oferta.

Se comprueba ahora como Madrid abastece su demanda recibiendo de dichos proveedores de mayor calidad. La tienda recibe la mayoría de la mercancía de Málaga, completando su abastecimiento de lenguado proveniente de Vigo. Esto puede verse en la figura siguiente.

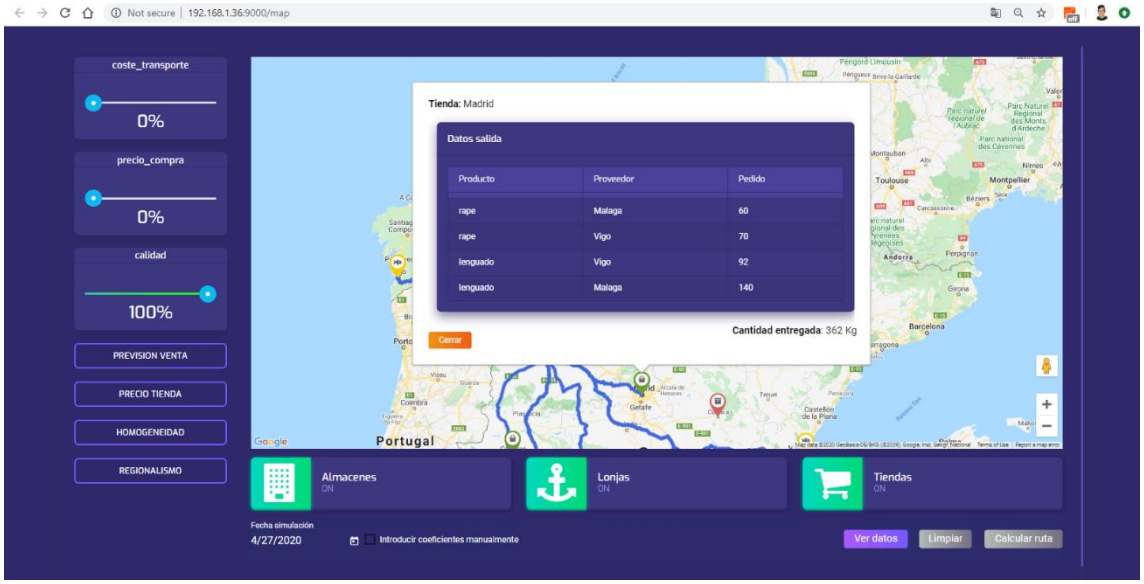

Figura 89: Captura de la información de abastecimiento de Madrid.
Fuente: PoC de Optimización de Redes (Capgemini).

Se observa, además, en la siguiente figura, como otras tiendas (como León) reciben la mercancía de proveedores de alta calidad del producto, aun estando cerca de otros proveedores. Queda demostrado con esto que el principal factor a tener en consideración para el cálculo de las rutas no es otro sino la calidad del producto ofertado.

Figura 90: Captura de la información de abastecimiento de salida de León.
Fuente: PoC de Optimización de Redes (Capgemini).

8. Conclusiones

En este capítulo se detallarán las conclusiones a las cuales se ha llegado durante el proceso de realización del trabajo, así como al final de este. El objetivo de ello es pues, concretar los aspectos tratados, así como también los distintos conceptos reforzados o aprendidos. Además de esto, se relacionarán los conceptos desarrollados en la elaboración del TFG con las materias vistas en las asignaturas durante el grado de ingeniería informática. Para finalizar, se relacionará el trabajo realizado con las distintas competencias transversales de la UPV y se mostrarán las principales limitaciones y futuras mejoras de la PoC.

La implicación en el proceso de elaboración, tanto en el modelo como en la prueba de concepto, ha obligado a hacer frente a situaciones en las que se debe llevar a cabo una abstracción de los conceptos. Con esto se ha conseguido pasar de la idea que se conocía de abastecimiento, a los algoritmos, métodos, tablas y relaciones, necesarias para su automatización y representación en la prueba de concepto.

El desarrollo de este proyecto, además, ha ayudado a reforzar la programación en general, poco presente en la rama de sistemas de información de la cual provengo. Gracias a haber implementado la parte *frontend* con Angular, se ha aprendido a programar lo básico en este lenguaje. Con esto, se ha comprendido que, efectivamente, es un lenguaje muy útil a la hora de reutilización de componentes y debido a su interdependencia, cosa que facilita las pruebas.

Por una parte, el haber estado trabajando en una empresa grande de TIC como es Capgemini, ha conseguido motivar durante el proceso de aprendizaje. A su vez, el método de formación autodidacta que sigue la empresa ha empujado en cierto modo el interés por saber y el afán de superar las tareas impuestas.

Por otra parte, lo más costoso de la elaboración del proyecto ha sido también, el proceso de formación o puesta en marcha. Se tuvo que dedicar bastante tiempo a la formación y comprensión de los conceptos, así como al funcionamiento de la prueba, antes de ponerse a trabajar en el proyecto.

Todo y que la PoC desarrollada no es definitiva, se ha obtenido una versión funcional capaz de dar solución al principal objetivo de optimización de rutas desde lonjas hasta supermercados. La economización del movimiento es crucial en los tiempos que corren y el desarrollo de aplicaciones o pruebas de concepto como esta, contribuyen de manera notable en la reducción de coste y contaminación en dicho movimiento.

Concretamente, mediante el desarrollo de la PoC detallada en este TFG, se han satisfecho los objetivos propuestos al inicio del trabajo:

- Se ha implementado un enlace entre las tareas de un establecimiento dedicadas a la gestión y la optimización de procesos. Mediante la implementación de la PoC, los costes de producción y abastecimiento serán los óptimos considerando los distintos factores a tener en cuenta por el establecimiento comercial.

- Mediante el enlace descrito anteriormente, se han simplificado las tareas logísticas de cálculo y planificación de rutas. Con la PoC implementada, el departamento de logística solo tendrá que introducir los valores de las distintas variables y realizar el posterior análisis de las rutas optimas.

- Se han introducido algunas de las principales tecnologías en el mundo de la comercialización y abastecimiento alimenticio. Aprender a utilizar dichas tecnologías resultará útil en cuanto a aspectos de optimización y futuros proyectos de mejora de la empresa.

- Además, se ha dado una explicación detallada del proceso de comunicación entre las partes de la arquitectura Cliente – Servidor, basada en *REST-Services*. La comprensión y ejemplificación del proceso de trato de los datos por parte del servidor ha permitido esclarecer y verlo menos ambiguo.

8.1 Relación del trabajo con los estudios cursados

Durante la elaboración del trabajo se han visto conceptos relacionados con las asignaturas vistas durante el grado, algunas de estas son:

- **TSR:** Se han aplicado algunos conceptos vistos acerca de JavaScript. También se ha hecho uso de Node.js, visto en la asignatura. La teoría vista en esta asignatura ha servido como base sobre la que partir a la hora de programar la parte del cliente de la prueba de concepto y entender su funcionamiento.

- **DBD y BDA:** Durante la implementación de la PoC se ha ido trabajando y realizando consultas y conexiones a distintas bases de datos. Todo y que se ha aprendido a trabajar con un gestor de base de datos nuevo como es PostgreSQL, la funcionalidad de este es comparable en grandes rasgos a Oracle.

- **COP:** En la implementación del modelo matemático se han aplicado los conceptos aprendidos en dicha asignatura. Se han definido las variables decisión referentes al problema del abastecimiento óptimo, planteado las distintas restricciones, así como las constantes y variables binarias necesarias para hacer el modelo correcto. Con todo lo dicho, se han planteado las distintas funciones objetivo, referentes a cada uno de los parámetros objetivo, introducidos por el usuario. Dichas funciones se agrupan formando una función multiobjetivo conjunta, con pesos asociados a dichos parámetros o variables introducidas.

- **COR:** Se han mostrado, en la introducción del proyecto, las distintas partes de la estructura organizativa de una empresa. En dicho apartado, se han descrito también las figuras de un establecimiento de comercialización que pertenecen a cada una de las partes de la estructura. Con esto se ha querido mostrar la veracidad del modelo de Mintzberg visto en la asignatura de comportamiento organizativo.

- **GPR:** De esta asignatura se han tenido en cuenta los conceptos relacionados con el plan de trabajo siguiendo a cabo un modelo. Se ha hablado acerca del modelo incremental de elaboración de software, aplicado al de la elaboración y continua evolución del modelo

matemático y la PoC desarrolladas. Se ha realizado también un presupuesto estimado, todo y que bastante simple, sirve para dar una idea acerca del coste del proyecto.

- **ISW, PRG, EDA:** Para llevar a cabo el desarrollo del proyecto, se ha tenido que programar y realizar pruebas de software. Todo y que estas pruebas no fueron exactamente las estudiadas en ISW, sí que se ha tenido en cuenta parte de la teoría vista en esta asignatura acerca de cómo llevarlas a cabo.

Aunque solo han sido nombradas algunas de las asignaturas cursadas, también han sido necesarios conocimientos vistos de manera indirecta en otras. Las tareas llevadas a cabo dentro de la empresa han tenido que realizarse teniendo en cuenta los aspectos de legalidad y confidencialidad vistos en asignaturas como GSE. Así también, asignaturas como DYP, SIO, MNE…han sido útiles para desenvolverse en el entorno de la empresa.

8.2 Relación del trabajo con las competencias transversales de la UPV

Resulta interesante también, mostrar cuales han sido las principales competencias transversales o *soft-skills* desarrolladas a lo largo del proyecto. Dichas competencias, favorecen la inserción en el mercado laboral y ayudan a ser más competente en este ámbito. Las que se han visto reforzadas en la elaboración del proyecto son:

- **El trabajo en equipo:** Las distintas partes dedicadas a la elaboración del proyecto de optimización de redes hemos estado en continuo contacto, ayudándonos los unos a otros a resolver las distintas dudas que iban surgiendo. El objetivo común de todos ha sido definir el modelo matemático, para su posterior utilización en la PoC.

- **La comprensión e integración:** Durante el proyecto se han tenido que asimilar nuevas ideas. Estas han sido referentes, tanto al funcionamiento y definición del modelo, como al aprendizaje de conceptos relativos a nuevos lenguajes.

- **La aplicación y el pensamiento práctico:** Se han aplicado los conceptos aprendidos en la etapa de formación en la empresa. Dichos conceptos aprendidos han servido para trabajar con nuevos gestores de bases de datos y a programar con *frameworks* como Angular. Se han aplicado también, los conocimientos vistos en las distintas asignaturas del grado, detallados en el apartado anterior.

- **El análisis y resolución de problemas:** Se ha tenido que considerar el problema del abastecimiento óptimo y proponer un modelo que diera una solución efectiva. Además, en cada una de las entregas se han tenido que analizar los nuevos requerimientos y buscar la forma de considerarlos en la PoC.

- **La innovación, creatividad y emprendimiento:** Se ha desarrollado un proyecto novedoso que además de optimizar en cuanto a costes de transporte, considera otras variables de gran interés para el establecimiento. Con esto se busca priorizar variables como el regionalismo o la homogeneidad por encima del coste del transporte.

- **El diseño y proyecto:** Se ha analizado el problema, definiendo el modelo matemático y posteriormente implementando una prueba de concepto funcional.

- **La responsabilidad ética, medioambiental y profesional:** Con la elaboración de esta mejora en el proceso de abastecimiento de pescado, se está contribuyendo a reducir la

huella de CO_2. Como bien se dijo, la idea del proyecto surge de la necesidad de optimizar al máximo el transporte del pescado, entre otros factores, reduciendo así la contaminación.

- **La comunicación efectiva:** En cada una de las entregas se han tenido que implementar los cambios propuestos por el jefe del proyecto. Así pues, este también ha tenido que estar informado continuamente de los avances del equipo, mediante reuniones donde se explicaban las mejoras realizadas.

- **El pensamiento crítico:** La comprensión del modelo matemático ha hecho posible entender y solventar de mejor forma los problemas que surgieron durante la implementación. Gracias a esto, ha sido posible analizar los requisitos que se pedían para cada evolutivo y encontrar la solución más viable teniendo en consideración la base teórica en que basa su funcionamiento la PoC.

- **La planificación y gestión del tiempo:** Se han establecido prioridades en cuanto a las tareas que realizar, de más a menos importantes. Además, mediante la fijación de reuniones y exposiciones, se ha 'obligado' a realizar dichas tareas dentro de un período de tiempo. Esto conlleva una correcta gestión del tiempo dedicado a la realización de cada una de las tareas.

- **El aprendizaje permanente:** En el mundo de la informática, todo está en continuo cambio. Se debe estar al día y en aprendizaje continuo para poder entender los nuevos lenguajes y programas. Durante el desarrollo del proyecto, la formación ha sido un proceso que no se ha visto nunca interrumpido.

- **El conocimiento de problemas contemporáneos:** La idea del proyecto, como bien se dijo en los primeros capítulos, surge, en parte, de la necesidad de dar solución al problema de contaminación por CO_2. Además, se ha sido consciente de la importancia que tiene para un establecimiento la optimización del tiempo transcurrido desde la adquisición del pescado en lonja, hasta su llegada a tienda.

- **La instrumental específica.** Se ha aprendido a programar en Angular, así como a manejar el gestor de bases de datos de PostgreSQL, necesarios para el desarrollo del proyecto.

8.3 Limitaciones y futuras mejoras

Para dar clausura a este TFG, dejar claros algunos aspectos finales. Durante el proceso de elaboración de este, se han adquirido conocimientos sobre las distintas tecnologías descritas. No obstante, no se ha profundizado lo suficiente como para poder considerarse expertos en Angular, Spring, Node.js, o cualquiera de las nuevas herramientas de programación utilizadas.

La formación en la empresa fue necesaria para poder desempeñar las tareas asignadas, pero no suficiente como para defenderse sin fallas en un mundo tan complejo y cambiante como es el de las TIC y la programación. Decir también, que algunos aspectos de la implementación, como es el caso de la definición de la base de datos, podrían haber sido mejores. Se insistió en intervenir en el proceso de definición de esta, pero finalmente, el trabajo se vio desviado hacia otros propósitos.

Como futuras mejoras, se podría considerar la corrección de la base, como bien se dijo en el capítulo de *Diseño de la solución*. Así también, podrían considerarse distintos aspectos relacionados con la lógica de la PoC.

Para dotar de mayor realismo el problema, podría considerarse que la fecha de caducidad de cada producto (o bien, sus días de vida determinados) influyesen en el cálculo de rutas. De esta forma, cada producto podría viajar por la ruta tanto tiempo como sus días de vida permitieran.

En esta versión de la PoC se ha considerado que un producto actúa como sustituto de otro para todas las tiendas. Para un programa futuro, se podría considerar que cada tienda tiene definidos sus productos sustitutos para cada producto, es decir, dos tiendas podrían tener dos productos sustitutos diferentes para un mismo producto.

9. Referencias

Introducción:

[1] Grandío, A. y Bou, J. C. (s. f.). *Burocracia y Liderazgo: De la Dirección de Operaciones a la de Recursos Humanos*. Recuperado 10 de junio de 2020, de https://www3.uji.es/~agrandio/Aedem94.htm

[2] Garrido, I. (2017, noviembre 24). Modelo Mintzberg, organización estructurada en la empresa | HR TRENDS. *HRTRENDS*. http://empresas.infoempleo.com/hrtrends/modelo-mintzberg-una-organizacion-estructurada-la-empresa

[3] Los 8 principios de gestión de la calidad. (2015, marzo 4). *Blog Calidad ISO*. http://blogdecalidadiso.es/los-8-principios-de-gestion-de-la-calidad/

Estado del arte:

[4] Martín, L. (2019, agosto 26). La logística se alía con el medio ambiente para evolucionar. *Compromiso Empresarial*. https://www.compromisoempresarial.com/rsc/2019/08/la-logistica-se-alia-con-el-medio-ambiente-para-evolucionar/

[5] Vigía, E. (2019, abril 27). *Diez apps para la gestión logística* -. http://elvigia.com/diez-apps-para-la-gestion-logistica/

[6] Google Spreadsheets. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Google_Spreadsheets&oldid=123695085

Capgemnini:

[7] Capgemini. (2017, agosto 4). Nuestra empresa. *Capgemini España*. https://www.capgemini.com/es-es/nuestra-empresa/

devonfw:

[8] *Devonfw/jump-the-queue*. (s. f.). GitHub. Recuperado 10 de junio de 2020, de https://github.com/devonfw/jump-the-queue

Typescript:

[9] TypeScript. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=TypeScript&oldid=125913536

[10] Gómez, M. A. (s. f.). *TypeScript: Los fundamentos en 10 min【2020】*. Recuperado 9 de junio de 2020, de https://softwarecrafters.io/typescript/typescript-tutorial-javascript-introduccion

Node.js:

[11] Node.js. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Node.js&oldid=126254418

[12] Node.js ¿Qué es y para qué sirve NodeJS?¿Qué no es NodeJS? (2015, septiembre 30). *Apasionados: Agencia de Marketing y Consultoría Estratégica Online*. https://apasionados.es/blog/nodejs-4430/

Angular:

[13] Angular (framework). (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Angular_(framework)&oldid=126367011

[14] Cano, J. (2018, mayo). Angular: Mucho más que un framework. *Software Guru*, *56*. https://sg.com.mx/revista/56/angular

Angular Material:

[15] Fatjó, J. C. (2019, octubre 25). Introducción a Angular Material. *Tribalyte Technologies*. https://tech.tribalyte.eu/blog-introduccion-angular-material

[16] Team, A. C. (s. f.). *Angular Material*. Angular Material. Recuperado 5 de junio de 2020, de https://material.angular.io/

Maven:

[17] *Maven – Welcome to Apache Maven*. (s. f.). Recuperado 8 de junio de 2020, de https://maven.apache.org/

Springboot y Tomcat:

[18] Miranda, P. (2019, junio 18). *Spring vs Spring Boot: ¿Cuál es la diferencia?* Mirandas. https://www.mirandas.work/es/posts/spring-vs-spring-boot-cual-es-la-diferencia/

[19] Alberto, L. (s. f.). *Iniciándose con Spring Boot (I) | El Recetario*. Recuperado 9 de junio de 2020, de http://ictblog.luisalbertogh.net/?p=424

[20] Tomcat. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Tomcat&oldid=126115277

[21] *Apache Tomcat®—Welcome!* (s. f.). Recuperado 11 de junio de 2020, de http://tomcat.apache.org/

PostgreSQL:

[22] PostgreSQL. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=PostgreSQL&oldid=126115356

[23] reizek. (s. f.). *Qué es PostgreSQL y cuáles son sus ventajas*. Platzi. Recuperado 10 de junio de 2020, de https://platzi.com/blog/que-es-postgresql/

Cplex:

[24] *CPLEX® Optimizers*. (s. f.). Recuperado 5 de junio de 2020, de https://www.ibm.com/support/knowledgecenter/SSSA5P_12.9.0/ilog.odms.cplex.help/CPLEX/homepages/cplex_KC_home.html

Plan de trabajo:

[25] Dbymatt. (2017, abril 3). Ingeniería de Software I - 2017: Metodologías de Desarrollo de Software. *Ingeniería de Software I - 2017*. http://isi17.blogspot.com/2017/04/metodologias-de-desarrollo-de-software.html

Presupuesto:

[26] *Directions API Usage and Billing*. (s. f.). Google Developers. Recuperado 11 de junio de 2020, de https://developers.google.com/maps/documentation/directions/usage-and-billing?hl=es-419

[27] *Microsoft 365 Empresa Estándar*. (s. f.). lizengo.es. Recuperado 9 de junio de 2020, de https://www.lizengo.es/office-suite/microsoft-office-365-business-premium

[28] *Skype for Business Pricing Plan & Cost Guide*. (s. f.). GetApp. Recuperado 12 de junio de 2020, de https://www.getapp.com/it-communications-software/a/skype-for-business/pricing/

Arquitectura del sistema:

[29] Caules, C. Á. (2016, noviembre 18). ¿Qué es REST? *Arquitectura Java*. https://www.arquitecturajava.com/que-es-rest/

[30] Kurata, D. (s. f.). *Angular: Getting Started* [Curso formativo en Pluralsight]. Recuperado 12 de junio de 2020, de https://www.pluralsight.com/courses/angular-2-getting-started-update

[31] Jiménez, Y. M. (2017, diciembre 31). Angular: ¿cómo narices se hace una petición http a una base de datos? *Medium*. https://medium.com/@yonem9/angular-c%C3%B3mo-narices-se-hace-una-petici%C3%B3n-http-a-una-base-de-datos-5735a26de2a2

[32] Cea, O. (2020, abril 14). RxJS: De cero a experto en 15 minutos. *Medium*. https://tech.cornershop.io/programaci%C3%B3n-reactiva-con-rxjs-bebc9432485f

Frontend y Backend:

[33] Front end y back end. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Front_end_y_back_end&oldid=125948469

[34] Acosta, V. M. (2019, marzo 4). Beneficios de la utilización de Frontend con Angular. *Canal Informática y TICS*. https://revistadigital.inesem.es/informatica-y-tics/frontend-con-angular-todo-lo-que-debes-saber-sobre-esta-herramienta/

[35] maldeadora. (s. f.). *Qué es Frontend y Backend*. Platzi. Recuperado 9 de junio de 2020, de https://platzi.com/blog/que-es-frontend-y-backend/

[36] *Directions Service | Maps JavaScript API | Google Developers*. (s. f.). Recuperado 14 de junio de 2020, de https://developers.google.com/maps/documentation/javascript/directions?hl=es-419

Implantación de la solución:

[37] *PostgreSQL Database Download | EnterpriseDB*. (s. f.). Recuperado 11 de junio de 2020, de https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

[38] *PgAdmin 4 (Windows) Download*. (s. f.). Recuperado 11 de junio de 2020, de https://www.pgadmin.org/download/pgadmin-4-windows/

[39] Foundation, E. (s. f.). *Eclipse Downloads | The Eclipse Foundation*. Recuperado 15 de junio de 2020, de https://www.eclipse.org/downloads/

[40] Node.js. (s. f.). *Node.js*. Node.js. Recuperado 15 de junio de 2020, de https://nodejs.org/es/

[41] Linárez, E. (2018, abril 1). *Actualizar NodeJS a una versión superior o inferior utilizando NPM*. Medium. https://medium.com/@eboriolinarez/actualizar-nodejs-a-una-versi%C3%B3n-superior-o-inferior-utilizando-npm-1a3c4ac194dd

[42] Alberto Basalo, A. (2020, marzo 4). *Hola Angular CLI*. Academia Binaria. https://academia-binaria.com/hola-angular-cli/index.html

[43] *Angular—Ng update*. (s. f.). Recuperado 15 de junio de 2020, de https://angular.io/cli/update

[44] avi.elkharrat. (s. f.). *angular—Instalación de una versión específica de angular con angular cli*. www.it-swarm.dev. Recuperado 5 de junio de 2020, de https://www.it-swarm.dev/es/angular/instalacion-de-una-version-especifica-de-angular-con-angular-cli/831135700/

[45] *angularjs—How to check the version of Angular Material*. (s. f.). Stack Overflow. Recuperado 15 de junio de 2020, de https://stackoverflow.com/questions/32566114/how-to-check-the-version-of-angular-material

[46] *How to install a specific version of Angular Material?* (s. f.). Stack Overflow. Recuperado 15 de junio de 2020, de https://stackoverflow.com/questions/57217229/how-to-install-a-specific-version-of-angular-material

Video Smart Monkey:

[47] SmartMonkey - Last-mile route optimization. (s. f.). *Primeros pasos—Optimizador de rutas con el plugin de SmartMonkey.io para Google Sheet* [Vídeo en Youtube]. Recuperado 10 de junio de 2020, de https://www.youtube.com/watch?v=FSVadDujE7w

Video WeebFleet Mobile:

[48] Abelcom Soluciones y Sistemas. (s. f.). *App WEBFLEET Mobile—ABELCOM* [Vídeo en Youtube]. Recuperado 10 de junio de 2020, de https://www.youtube.com/watch?v=riUUAFeQUts

Video creación diagrama de Gantt:

[49] Saber Programas. (s. f.). *Excel—Cómo hacer un diagrama de Gantt o cronograma utilizando los gráficos. Tutorial en español HD* [Vídeo en Youtube]. Recuperado 10 de junio de 2020, de https://www.youtube.com/watch?v=chR6kx4btDQ&t=242s

Relación trabajo con competencias transversales UPV:

[50] *Competencias Transversales: UPV*. (s. f.). Recuperado 3 de junio de 2020, de http://www.upv.es/contenidos/COMPTRAN/info/1026833normalc.html

Enlaces de descarga, libros y otras fuentes:

[51] *Download Visual Studio Code—Mac, Linux, Windows*. (s. f.). Recuperado 16 de junio de 2020, de https://code.visualstudio.com/Download

[52] *Devonfw guide*. (s. f.). Recuperado 16 de junio de 2020, de https://devonfw.com/index.html

[53] *Capgemini (@Capgemini) / Twitter*. (s. f.). Twitter. Recuperado 16 de junio de 2020, de https://twitter.com/capgemini

[54] Smart Monkey. (s. f.). *eShow | Congreso profesional de eCommerce y Marketing Digital*. Recuperado 17 de junio de 2020, de https://www.the-eshow.com/barcelona/expositores/smart-monkey/smart-monkey/

[55] *WEBFLEET Mobile*. (s. f.). App Store. Recuperado 17 de junio de 2020, de https://apps.apple.com/es/app/webfleet-mobile/id455972224

[56] Tripana, M. (2017). Análisis del diseño estructural de la empresa. En *Análisis del diseño organizativo de la empresa "Mercadona s.a"* (trabajo de fin de grado). Universidad de Almería, España. Recuperado de http://repositorio.ual.es/bitstream/handle/10835/6605/15307_TFG%20Marina%20Tripiana.pdf?sequence=1&isAllowed=y

[57] Casares, J. (2000). Estructura de la distribución de productos pesqueros. En *Comercialización y distribución de productos pesqueros* (1.ª ed., pp. 243–255). Madrid: Ministerio de Agricultura, Pesca y Alimentación. Recuperado de https://www.mapa.gob.es/ministerio/pags/biblioteca/fondo/pdf/29259_all.pdf

[58] ECMAScript. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=ECMAScript&oldid=126050779

[59] Java Servlet. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=Java_Servlet&oldid=125248440

[60] JavaServer Pages. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=JavaServer_Pages&oldid=126912998

[61] *Lenguaje de definición de datos (DDL)—SQL Is My Sin*. (s. f.). Recuperado 23 de junio de 2020, de https://sites.google.com/site/sqlismysin/home/lenguaje-de-definicion-de-datos-ddl

[62] Diferencias entre DDL, DML y DCL. (2017, noviembre 7). *TodoPostgreSQL*. https://todopostgresql.com/diferencias-entre-ddl-dml-y-dcl/

[63] Accelerated solution development: Devonfw. (2017, septiembre 25). *Capgemini Deutschland*. https://www.capgemini.com/de-de/devonfw/

[64] Basalo, A. (s. f.). *Academia Binaria*. Academia Binaria. Recuperado 27 de junio de 2020, de https://academia-binaria.com/index.html

[65] AJAX. (2020). En *Wikipedia, la enciclopedia libre*. https://es.wikipedia.org/w/index.php?title=AJAX&oldid=126854787

[66] *Pluralsight—Unlimited Online Developer, IT, and Cyber Security Training*. (s. f.). Recuperado 2 de julio de 2020, de https://www.pluralsight.com