Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería del Software

CREACIÓN, TESTEO Y MONITORIZACIÓN DE MÉTODOS DEL ESTÁNDAR EUROPEO SIRI

Alumno: Álvaro Gamarra Martín

Tutor: César Llamas Bello

Agradecimientos

En primer lugar, quiero agradecer a todas las personas que han estado a mi lado a lo largo de la carrera. Parece que fuera ayer cuando llegué a casa con las notas del instituto y mi padre me dijo: «Ahora tienes que pensar qué quieres estudiar». Pasaron los meses y yo seguía dudando entre Informática, Teleco o el doble grado de Informática y Estadística. Y, en este momento, después de esta larga travesía, estoy aquí, comenzando mi Trabajo de Fin de Grado. Por ello, a las primeras personas que quiero mostrar mi agradecimiento es a mi familia, que ha estado ahí siempre por encima de todo, ayudándome en la medida de lo posible. En segundo lugar, a mi tutor de la UVA, que me ha apoyado y dirigido en todo el proceso. Después, a todos mis amigos, tanto de la universidad, como de fuera de ella.

Resumen

La presente memoria de trabajo de fin de grado detalla la creación, para su posterior testeo y monitorización, de un servidor web remoto de tipo SOAP que implementa el estándar europeo SIRI (Standard Interface for Real-time Information) de posicionamiento a tiempo real de operaciones de flotas de transporte público. Los productos elaborados comprenden el desarrollo del método de consulta de topología en el servidor, creación de una página web que permitirá probar de forma manual varios métodos implementados en él, y una aplicación de Windows que consultará en modo continuo uno de los métodos que se prueban, la monitorización de paradas. Con la intención de comprobar su buen funcionamiento a lo largo del tiempo.

Este trabajo se ha desarrollado en un contexto real donde se han aprovechado las necesidades reales de un cliente-usuario del sistema para elaborar los requisitos de los prototipos y el sistema final. La metodología empleada en el proyecto obedece a un modelo de desarrollo evolutivo basado en prototipos, lo que permitirá que el cliente, de forma periódica, vea los avances que se han realizado y valore cambios o mejoras.

Abstract

This documents details the final degree project that consist in the creation, for subsequent testing and monitoring, of a SOAP-type remote web server that implements the European Standard SIRI (Standard Interface for Real-time Information) that specifies a mechanism for exchanging information about the planned, current or projected performance of real-time public transport operations between different computer systems. The main results of this project include three main parts, the development of the topology query method on the server, a web page that will allow you to manually test various methods implemented on the server and a Windows application that will constantly consult one of the methods that are going to be tested, the stop monitoring. With the intention of verifying its proper functioning over time.

This work has been developed in a real context where the real needs of a client-user of the system have been used to develop the requirements of the prototypes and the final system. The methodology used in the project obeys an evolutionary development model based on prototypes, which will allow the client to periodically see the progress that has been made and assess changes or improvements.

Índice general

Capítulo 1

Introducción

En el contexto de los sistemas de información de operaciones de transporte público en tiempo real, SIRI (Standard Interface for Real-time Information) trata de convertirse en un estándar para la interconexión de sistemas[1] Con la pretensión de obtener un conocimiento más profundo y estudiar la viablidad de su utilización práctica en sistemas reales en un entorno empresarial, este trabajo refleja un proyecto de creación de un cliente web, el desarrollo de un método de los que especificados por SIRI, un Web Service y una aplicación de Windows para el testeo y la monitorización de una serie de funcionalidades de un servicio SOAP.

La metodología utilizada para la página web, la aplicación de Windows y el método de consulta de topología es la de desarrollo o construcción de prototipos, que pertenece a los modelos de proceso evolutivo. Los motivos por los que se escogió esta metodología son varios:

- El beneficiario, en este caso el cliente de la empresa GMV, podrá validar el producto durante todo el proceso, dando a conocer sus necesidades.
- Permite que los desarrolladores puedan entender mejor el sistema, lo que da más calidad al código que, además, es reutilizable.

Entre todos los tipos de prototipos se escogió el vertical, puesto que cuentan con funcionalidades limitadas de producto final, pero en una fase muy avanzada de su desarrollo. El motivo de esta elección fue, sobre todo, la decisión del cliente que quería ir viendo funcionalidades completas con premura, permitiendo que pudiese validar partes funcionales, lo que da una imagen de profesionalidad y constante mejoría.

- *Primer prototipo*: Supondrá la creación de una página web sencilla y de la implementación del funcionamiento completo de la consulta de topología. Esto incluye la implementación de esta parte en el Web Service ya desarrollado por GMV para el resto de métodos.

[1] Standard Interface for Real-time Information[25]

- *Segundo prototipo*: Se mejorará la página web creada en el primer prototipo, convirtiéndola en usable y profesional para el testeo de la información de las paradas y para la consulta de topología, además del funcionamiento de la relación entre ellas. Incluirá los requisitos surgidos de la revisión y la validación del primer prototipo por parte del cliente.

- *Tercer prototipo*: Consistirá en la adaptación de la página web a la monitorización automática de paradas, y de la implementación del funcionamiento completo de esta parte, que serán una aplicación de Windows y un aplicación de servicios WCF. Incluirá también, los requisitos surgidos de la revisión y la validación del segundo prototipo por el cliente.

1.1. Motivación

La principal motivación para la realización de este proyecto como TFG fue la propuesta de GMV, empresa en la que realicé las prácticas de la carrera universitaria y en la que trabajo hoy en día. Desde el principio, GMV planteó la posibilidad de hacer una serie de tareas que podían dar lugar a un Trabajo de Fin de Grado muy interesante.

Entre las tareas propuestas, esta fue la seleccionada por una serie de motivos:

- **Novedad**: una de las principales motivaciones fue la posibilidad de trabajar con un lenguaje de programación totalmente nuevo para mí y con un Servicio Web de tipo SOAP que, aunque había estudiado a nivel teórico, nunca había trabajado con ello.

- **Interés**: otra de las motivaciones fue el estándar SIRI: no se podía desaprovechar la oportunidad de trabajar con un estándar europeo.

- **Oportunidad**: la empresa y el tutor nombrado por ella han dado siempre el soporte necesario para poder continuar con las tareas, aspecto que es muy de agradecer.

1.2. Objetivos específicos del sistema

Los objetivos de este proyecto son los siguientes:

- *La página web debe ser compatible con cualquier Servicio Web que implemente métodos del estándar europeo SIRI.*

- *Debe también permitir consultar la topología de líneas, trayectos y paradas del Servicio Web utilizado.*

- *Debe dar la opción de consultar paradas a tiempo real.*

- *La aplicación de Windows* [2] *deberá consultar de forma periódica la topología como las paradas, para comprobar el correcto funcionamiento del subsistema.*
- *La página web deberá mantener un histórico del funcionamiento automático del subsistema.*

1.3. Contenidos de la memoria

En esta sección se hará una breve descripción de lo que constituirá cada uno de los puntos de la memoria. Como ya previamente se documentó lo que incluye cada prototipo, solo se explicarán las fases internas comunes a los tres.

- *SIRI*: A lo largo de este apartado se describirá en que consiste el estándar, quiénes son los creadores, cuáles son sus usos principales y que métodos constituyen el mismo.
- *Plataformas*: Incluirá una descripción de los entornos de ejecución y desarrollo utilizados a lo largo del proyecto.
- *Análisis del problema*: Apartado que consistirá en describir el entorno inicial, los problemas que tratará de solucionar el proyecto y su estudio de viabilidad.
- *Metodología*: Describe la metodología que se va a utilizar: desarrollo evolutivo por prototipos. Explica los diferentes tipos de prototipos que existen y las razones por las que se ha escogido el prototipo vertical.
- Los apartados correspondientes a los tres prototipos describen la especificación de los requisitos y la batería de pruebas seleccionada, su análisis y sus diagramas pertinentes, el diseño en el que se describirá su arquitectura en profundidad, su implementación, que contendrá en detalle todo el desarrollo del mismo y, por último, su validación relacionando las pruebas realizadas.
- *Trabajo futuro*: en este apartado se propondrán mejoras diversas que podrían dar lugar a ampliaciones del proyecto.
- *Conclusiones*: La aplicación cumple los objetivos inicialmente definidos: tiene el alcance planteado desde un inicio, ofreciendo incluso más funcionalidades de las previstas inicialmente. Se puede valorar, desde nuestro punto de vista, muy positivamente el resultado del trabajo.
- *Lista de figuras, de tablas y bibliografía*: Para finalizar el trabajo se han incluido las figuras, tablas y bibliografía más importante de la utilizada para realiza este TFG.

[2] Windows™ es una marca registrada de Microsoft Corporation.

Capítulo 2

SIRI

2.1. ¿Qué es SIRI?

Los pasajeros de los servicios de transporte público solicitan cada vez más información y más variada: horarios, trayectos, información en tiempo real sobre el funcionamiento del servicio, etc. Todo ello lo proporcionan los sistemas informáticos, que además pueden emitir billetes de viaje, abonos y recibos, administrar flotas de vehículos en función de las necesidades reales o puntuales, etc.

SIRI [30] es un estándar de europeo que se utiliza entre diferentes sistemas informáticos para el intercambio de información sobre el rendimiento, ya sea planificado, real o proyectado, de las operaciones de transporte público en tiempo real. Características:

- Comprende un conjunto cuidadosamente modularizado de servicios funcionales para operar sistemas de información de transporte público, que aportan información sobre:
  - El intercambio de horarios planificado y en tiempo real.
  - La actividad de los vehículos en las paradas.
  - El movimiento de los vehículos.
  - Información para asistir en la provisión de conexiones confiables entre vehículos.
- Su objetivo es incorporar lo mejor de varios estándares nacionales y patentados de toda Europa y entregarlos utilizando un esquema XML moderno, terminología TransModel y conceptos de modelado.

- Todos los servicios SIRI se proporcionan a través de una capa de comunicaciones estandarizada, basada en una arquitectura de servicios web, que mantiene un enfoque para todos los servicios funcionales:
  - Seguridad.
  - Autenticación.
  - Control de versiones.
  - Recuperación/reinicio.
  - Control de acceso/filtrado.

Se admiten dos patrones principales que engloban diferentes requisitos operativos:

- Protocolo de solicitud/respuesta inmediata.
- Protocolo de publicación/suscripción asíncrona, que puede ser más elaborada y optimizada.

SIRI es un estándar en proceso, al que se espera que se agreguen servicios adicionales con el tiempo utilizando el mismo portador de comunicaciones. Su modularización permite:

- Dividir la complejidad de un problema convirtiendo problemas complejos en un conjunto de problemas más simples y por tanto más sencillos de implementar.
- Reutilizar el código de un programa en cualquier momento de su ejecución.

Permite, además, un enfoque incremental, es decir, una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema. De esta forma, únicamente el subconjunto de servicios realmente requerido debe implementarse para una aplicación en particular. El objetivo sería que los usuarios pudieran comenzar con algún servicio y, con el tiempo, aumentaran la cantidad de servicios y opciones admitidas. El mismo proceso se podría producir entre los proveedores.

SIRI lleva a cabo un análisis muy completo de todos los servicios de información, de datos, de modelos de estos, de transporte y de medición en tiempo real. Ello conlleva que haya precisión en el modelo subyacente porque los servicios en tiempo real frecuentemente solo intercambian cambios en tiempo real en los datos.

2.2. ¿Por qué se creó SIRI?

En toda Europa, las interfaces de software abierto y bien definidas tienen un papel muy destacado en la mejora de la viabilidad económica y técnica de los sistemas de información en cualquier transporte público.

- Usando interfaces estandarizadas, los sistemas pueden implementarse como módulos enchufables discretos que pueden elegirse entre una amplia variedad de proveedores en un mercado competitivo, en lugar de sistemas patentados de un solo proveedor.
- Las interfaces bien definidas también permiten la prueba automatizada sistemática de cada módulo funcional, vital para gestionar la complejidad de los crecientes sistemas grandes y dinámicos a los que conduce la interconexión de redes.
- Los módulos funcionales individuales se pueden reemplazar o evolucionar, sin rupturas inesperadas de funciones excesivamente dependientes.
- El uso de interfaces comunes permite que los datos en tiempo real de diferentes áreas se vinculen sin problemas, tanto dentro como fuera de las fronteras europeas, lo que permite verdaderos sistemas de información de viajes transeuropeos.

Los motivos principales de las partes interesadas son:

- Los compradores de sistemas quieren una forma sencilla y segura de adquirir diferentes componentes de un sistema de información de transporte público de diferentes proveedores, y tener la confianza de que estos diversos componentes funcionarán juntos. También quieren una protección a largo plazo de la inversión, de modo que haya un soporte continuo para sus sistemas, un proceso coherente para evolucionar los sistemas y la comodidad de que los componentes adicionales que se pueden comprar en el futuro también se integrarán en el sistema.
- Los proveedores de sistemas quieren desarrollar productos para un mercado europeo, asegurándose de que sus sistemas se puedan usar en todos los países sin necesidad de implementar diferentes estándares de interfaz en cada región. También quieren protección de la inversión y un enfoque incremental que les permita distribuir su inversión en estándares durante un período prolongado y tener en cuenta diferentes conjuntos de datos nacionales y procesos comerciales para la gestión de datos.

SIRI también abordó la creciente necesidad de actualizar varios estándares nacionales y patentados para cumplir con las metodologías y tecnologías modernas, en particular los estándares modernos de intercambio de datos (XML), para usar términos de dominio de aplicación de transporte público estándar y conceptos de modelado en línea con el estándar europeo TransModel, y para construir sobre otros estándares técnicos relevantes de CEN, ISO y W3C, por ejemplo, para posición geoespacial, códigos de idioma, etc.

2.3. ¿Quién creó SIRI?

SIRI ha sido creado por un conjunto de organizaciones e instituciones: proveedores de equipos, autoridades de transporte, operadores y consultores de transporte de ocho países europeos (República Checa, Alemania, Dinamarca, Francia, Noruega, Suecia, Reino Unido), con el respaldo de VDV en Alemania, la Direction des Transports Terrestres de la Ministerio de Transportes de Francia y RTIG en el Reino Unido, y basándose en el trabajo del proyecto Trident de la UE.

2.4. ¿Qué servicios están disponibles?

SIRI posee actualmente los siguientes servicios funcionales:

2.4.1. Servicio de Horarios de Producción (PT)

Proporciona información sobre el funcionamiento esperado de una red de transporte para un día concreto en un futuro próximo. Por lo general, esto se produce unas horas o días antes del día en cuestión e incorpora cualquier cambio en los horarios conocidos en ese momento. Un Horario de Producción puede ser filtrado por Operador, Línea y Rango de Fechas, permitiendo seleccionar sólo la sección del horario de interés. Es un servicio muy adecuado para el aprovisionamiento de sistemas AVL y dispositivos inteligentes con horarios base.

2.4.2. El Servicio de Horario Estimado (ET)

Da detalles del funcionamiento de la red de transporte para un periodo dentro del día en curso, proporcionando, en tiempo real las desviaciones de los horarios y las acciones de control que afectan al Horario (anulaciones, viajes adicionales y desvíos). Se puede filtrar un horario estimado por Operador o por Línea, permitiendo seleccionar sólo la sección del horario que sea de interés. Es también adecuado para aprovisionar sistemas AVL y dispositivos inteligentes con horarios en tiempo real.

2.4.3. Los Servicios de Parada

Existen dos, el Horario de Paradas o Stop Timetable (ST) y la Monitorización de Paradas o Stop Monitoring (SM) Brindan información centrada en las paradas sobre las llegadas y salidas de vehículos actuales y futuras en una parada designada o punto de monitoreo, generalmente para salidas dentro de los próximos 20 a 60 minutos, para mostrar al público. El servicio SM es adecuado en particular para proporcionar tablas de salida en todas las formas de dispositivo.

2.4.4. El Servicio de Seguimiento de Vehículos (VM)

Proporciona información sobre la ubicación actual y las actividades esperadas de un vehículo en particular, y puede brindar el Viaje actual y posterior y los Puntos de llamada en cada viaje, junto con los tiempos de llegada programados y esperados. El servicio VM es adecuado en particular para pantallas a bordo y visualización del movimiento del vehículo, y para intercambiar información sobre vehículos itinerantes entre diferentes sistemas de control. También constituye una fuente de registro detallada adecuada para recopilar datos históricos sobre el rendimiento contra el cronograma.

2.4.5. Los Servicios de Protección de Conexión (Horario de Conexión – CT- y Monitoreo de Conexión – CM-)

Permiten a los operadores de transporte intercambiar información sobre la gestión en tiempo real de los intercambios entre vehículos alimentadores y distribuidores que llegan y salen de un punto de conexión, por ejemplo, para informar a los pasajeros de un tren retrasado que un servicio de autobús local los esperará.

2.4.6. Servicio de Mensajería General (GM)

Es una forma estructurada de intercambiar mensajes informativos arbitrarios entre los participantes, como noticias de viajes, consejos operativos. Se puede utilizar para vincular sistemas de gestión de incidentes en una arquitectura de almacenamiento y reenvío.

2.5. ¿Cómo funciona SIRI?

Como ya se ha apuntado, SIRI es una interfaz destinada a ser utilizada para intercambiar información entre servidores que contienen datos de tiempo de viaje o vehículos de transporte público en tiempo real, que incluyen los centros de control de los operadores de transporte y los sistemas de información que utilizan información del vehículo en tiempo real para operar el sistema, y los sistemas posteriores que brindan información de viaje al público sobre las paradas y pantallas a bordo, dispositivos móviles, etc. Características importantes.

- Utiliza el lenguaje de marcado extensible (XML) para definir sus mensajes. Se hace una cuidadosa separación entre Transporte (cómo se transportan los datos) y Carga útil (los datos de dominio intercambiados), de modo que los mensajes SIRI se pueden intercambiar como documentos XML con http POST o utilizando el Protocolo simple de acceso a objetos (SOAP). Una definición de servicio web El enlace de idioma (WSDL) también se define para este último.

- El modelo de carga útil está envuelto en una capa de mediación, también descrita con XML, que proporciona funciones de gestión comunes y también describe formalmente como políticas los aspectos parametrizados de comportamiento de mediación o intercambio o que puede llevar a cabo un servicio.
- La terminología y las relaciones de CEN TransModel se siguen en el modelo de datos de la aplicación PT subyacente.

SIRI está diseñada para un funcionamiento eficiente en una amplia variedad de contextos. Se puede utilizar tanto para la canalización masiva de grandes cantidades de datos entre diferentes sistemas informáticos como para consultas ad-hoc de menor tráfico.

Utiliza un conjunto de protocolos generales de comunicación para intercambiar información entre el cliente y el servidor, con unos mismos patrones comunes de intercambio de mensajes que se utilizan en todas las diferentes interfaces funcionales, con dos patrones específicos. Se utilizan dos patrones específicos bien conocidos de interacción cliente-servidor:

- Solicitud/Respuesta permite el intercambio apropiado de datos bajo demanda del cliente.
- Publicación/Suscripción permite la inserción asincrónica repetida de notificaciones y datos para distribuir eventos y situaciones detectadas por un servicio en tiempo real. Esto puede ser mucho más eficiente para algunos tipos de comunicación ya que el cliente no necesita sondear para detectar cambios en los datos; más bien, el servicio de notificación desencadena un intercambio de datos solo cuando detecta un evento. El protocolo de publicación/suscripción de SIRI prescribe una mediación particular para filtrar la cantidad de mensajes devueltos, por ejemplo, solo creando actualizaciones si las predicciones en tiempo real cambian más de un cierto umbral con respecto a un valor anterior.

Capítulo 3

Plataformas

En este apartado se tratarán las diferentes herramientas necesarias, tanto para la ejecución como para el desarrollo del proyecto. Por ello se diferenciarán dos tipos: la plataforma de ejecución y la plataforma de desarrollo.

La elección de las plataformas ha estado en parte limitada por las especificaciones de requisitos habituales en el contexto de la empresa GMV, en la que se ha realizado el proyecto y beneficiaria de él. Por ello, algunas de las decisiones de los entornos utilizados han sido previamente pactados con la empresa, entre los que se encuentran C# como lenguaje central de programación, Sql Server como Sistema Gestor de Bases de Datos y tanto Microsoft Visual Studio 2015 y 2019 como IDE en el que se programa.

3.1. Plataformas de ejecución

La plataforma de ejecución, o solution stack, es un conjunto de sistemas o componentes necesarios para realizar una solución funcional y robusta. Incluye todas las herramientas necesarias para la ejecución de la aplicación (o las aplicaciones).

En este caso, el proyecto constará de una aplicación web, dos *web services* y una aplicación de Windows. Con estos requirimientos se precisará:

- En primer lugar, contar con un sistema operativo Windows. Entre toda la amplia gama de versiones de Windows que existen sería recomendable una lo más actualizada posible, para evitar exploits y la versión Server, que está especialmente orientada en rendimiento para programas informáticos.
- En segundo lugar, dependiendo de cómo se hayan publicado las aplicaciones, se necesitará tener .NET instalado o no. Lo usual es que una vez que se publique la aplicación ya genere las dependencias, esto evita que la máquina donde se va a desplegar tenga que tener .NET.

- En tercer lugar, será necesario tener instalado el Internet Information Server (IIS), que es una característica configurable de Windows y es imprescindible para la publicación de los Web Services.
- En cuarto lugar está el navegador web; realmente cualquier navegador valdría, ya que el único requisito es que sea compatible con Java Script y a día de hoy todos los navegadores modernos cuentan con esta compatibilidad. Java Script es un lenguaje interpretado de programación que opera dinámicamente por lo que cabe mencionarlo en este punto. De todas formas, al ser un lenguaje de programación, se tratará en la Plataforma de Desarrollo.

Navegador	Versión
Google Chrome	28.0+
Mozilla Firefox	26.0+
Opera	17.0+
Microsoft Edge	25.10+
Windows Internet Explorer	NA
Apple Safari	5+
Navegador Android	4.3+

Tabla 3.1: Navegadores compatibles con Java Script.

- Por último, será necesario un Sistema Gestor de Bases de Datos (SGDB). Se recomienda especialmente SQL Server, aunque es compatible con el resto de SGDBs, salvo con Oracle y los que compartan sintaxis con éste.

3.2. Plataformas de desarrollo (Development Stack)

La plataforma de desarrollo o Development Stack es el conjunto de lenguajes, bibliotecas, IDE y herramientas que se utilizan para el desarrollo de aplicaciones.

3.3. Lenguajes utilizados

A lo largo del proyecto se utilizará de forma mayoritaria C#, tanto del Web Service, como para la página web y la aplicación de consola.

C#, es una evolución que Microsoft realizó con lo mejor de los lenguajes C y C++, y ha continuado añadiéndole funcionalidades, que ha tomado de otros lenguajes. En la versión de .NET Core, que es la que se utilizará en este proyecto, se ha reconstruido por completo su compilador, haciendo las aplicaciones un 600 % más rápidas.

La decisión de hacerlo en C# fue sobre todo de la empresa, en tanto que era preciso continuar un desarrollo que ya se había iniciado en este lenguaje. Desde el principio fue un reto muy ilusionante. Por otro lado, durante gran parte de la carrera se estudia Java y ambos lenguajes guardan una estrecha relación, por lo que es posible, poco a poco, adaptarse.

Para el desarrollo web se han utilizado varios lenguajes:

- **HTML o Hyper Text Markup Language** es el lenguaje que se usa para escribir toda la parte estática de una página web. Está compuesto por etiquetas que marcan el inicio y fin de una estructura como, por ejemplo, un párrafo de texto, una imagen o un enlace. HTML no es un lenguaje de programación, pues carece de características como secuencias, condicionales y ciclos.

- **Ajax**: no es un lenguaje de programación, pero merece la pena mencionarlo en este apartado ya que se encuentra justo entre el código de Java Script y el código en C#. Ajax (Asynchronous JavaScript and XML) se refiere a un grupo de tecnologías que se utilizan para desarrollar aplicaciones web. Al combinar estas tecnologías, las páginas web parece que son más receptivas puesto que los paquetes pequeños de datos se intercambian con el servidor y las páginas web no se vuelven a cargar cada vez que un usuario realiza un cambio de entrada. Ajax [7] permite que un usuario de la aplicación web interactúe con una página web, sin la interrupción que implica volver a cargar la página.

- **Java Script**: lenguaje de programación interpretado, que los desarrolladores utilizan para hacer páginas web interactivas. Los últimos años ha tomado muchísima fuerza, posicionándose a día de hoy como el más usado. Java Script se utilizará para prácticamente toda la parte dinámica de la web.

- **C#** se utilizará para el controlador, es decir, procesar los formularios HTML y para procesar las peticiones de la parte de Java Script mediante la tecnología Ajax. También, gracias a que el proyecto es un ASP.NET MVC, se podrá utilizar el propio código C# en algunas partes de las vistas.

- **Razor** es una sintaxis de programación ASP.NET utilizada para crear páginas web dinámicas con los lenguajes de programación C# o VB.NET . Razor [24] no es un lenguaje de programación, sino simplemente un motor de vistas, herramienta que creó ASP.NET MVC para poder utilizar algo de código del controlador en la vista, únicamente para mostrar información, ya que, de otra forma, no se cumpliría el patrón MVC. En este caso se utilizará Razor para poder intercambiar más fácilmente la información que se quiere mostrar entre el controlador y la vista.

3.4. Herramientas utilizadas

Para el desarrollo del proyecto se utilizarán también una serie de herramientas para el desarrollo:

- **Microsoft Visual Studio 2015 y 2019**: es un entorno de desarrollo integrado para Windows y MacOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic. En este caso se utilizará la versión 2019 para la aplicación web, para la aplicación de Windows y para el Web Service que consulta la monitorización automática. Y la 2015 para el desarrollo del método que implemente el estándar europeo SIRI, se planteó así ya que otros métodos, que ya se implementaron en el pasado, se realizaron de esta forma.
- **Microsoft Excel**: se utilizará para realizar los cálculos necesarios que se necesiten a lo largo del proyecto.
- **Overleaf**: herramienta con la que se redactará online el documento en Latex.
- **Latex**: es un sistema de software para la preparación de documentos sin formato. AL escribir, se utilizan convenciones de marcado de etiquetas para definir la estructura general de un documento, dar estilo al texto en todo el documento (como negrita y cursiva) y agregar citas y referencias cruzadas.
- **Postman**: es una plataforma de API para que los desarrolladores diseñen, construyan, prueben e iteren sus API.
- **JabRef**: es un Software de gestión bibliográfica que utiliza BibTeX como formato nativo.

Capítulo 4

Análisis del problema

Comprender la situación y el entorno es básico para comprender el problema que se pretende resolver.

4.1. Entorno del que partimos

GMV proporciona una serie de aplicaciones que permiten, entre muchas otras cosas, conocer el posicionamiento a tiempo real de autobuses y trenes. Es preciso explicar esto ya que, a partir del posicionamiento a tiempo real del autobús, el recorrido que hace, el histórico de tiempos del trayecto y otra serie de factores, se sondean los datos.

Estos datos siguen el formato de GMV y, por tanto, se necesita que una herramienta, que en este caso será un Web Service, los convierta a las estructuras de datos que proporciona SIRI, y así intercambiarlo con el Puesto Web que permita verlas. Este tema se tratará con detalle más adelante.

Lo importante en este momento es conocer el entorno del que se parte.

4.2. Problema a solucionar

El problema que se trata de solucionar nace de la necesidad, en determinados proyectos, de conocer desde sus servidores el tiempo que queda para que llegue cada bus/tren a cada parada.

Este aspecto es especialmente importante, ya que en los paneles de cada parada se ve el tiempo que, en teoría, va a tardar en llegar el bus y, en muchas ocasiones, se parte únicamente del histórico del recorrido, por lo que siempre hay un alto porcentaje de error.

También hay gran cantidad de veces en que el software de los paneles, o el que da la información a los paneles, pueda tener algún problema o estar apagado, por lo que no se muestra la información.

Por todo ello, diferentes proyectos solicitaron una herramienta que les permitiese poder conocer los datos de cada parada a tiempo real.

Para hacerlo, GMV, antes del planteamiento de este TFG, comenzó un desarrollo que, básicamente, consultaba esta información en la base de datos y la devolvía en el formato de SIRI. Pero para poder ver esta información desde el punto de vista del usuario se necesita una herramienta con interfaz, sea una aplicación de escritorio, web o móvil, que no existía.

Esta problemática, sumada a qué para poder consultar los datos se necesitaba el identificador público de la parada y que no sé monitorizaba que las paradas devolvieran datos de formas constante, dio lugar a la necesidad de crear una página web para consultar la información.

4.3. Estudio de viabilidad

Antes de comenzar el proyecto es necesario plantear un estudio de viabilidad [8], también conocido como estudio de factibilidad o perfil, que se realiza para determinar el éxito que puede alcanzar un proyecto a partir de la evaluación de sus aspectos técnicos y económicos.

Para estudiar la viabilidad del proyecto existen seis áreas imprenscindibles [17]:

- Viabilidad técnica
- Viabilidad económica
- Viabilidad del mercado
- Viabilidad legal
- Viabilidad temporal

En este proyecto en particular hay varios puntos que no tienen relevancia en el cálculo. Estas son viabilidad económica o la legal [19]: la primera debido a que no hay coste monetario real detrás del proyecto, puesto que es un Trabajo de Fin de Grado; y la legal porque el proyecto ha sido aceptado por el departamento legal de la empresa GMV, además de que su uso será estrictamente profesional.

4.3.1. Viabilidad técnica del proyecto

La viabilidad técnica de un proyecto mide el nivel de recursos técnicos que una organización tiene a su alcance, es decir, estudia si se dispone de los recursos, las herramientas y la tecnología necesarias para llevar a cabo el desarrollo del proyecto.

Este punto tiene especial importancia en este proyecto ya que hay varios entornos y herramientas desconocidos.

Para poder obtener los conocimientos necesarios fue preciso realizar formación básica, tanto dirigida por la empresa, como de forma autodidacta, para poder hacer frente a la incertidumbre y al desconocimiento de algunos entornos.

4.3.2. Viabilidad del mercado del proyecto

La viabilidad de mercado para este proyecto es un aspecto no tan relevante, ya que será una nueva propuesta de venta por parte de GMV a los proyectos con los que ya cuenta. Además, la empresa presenta propuestas conjuntas de los diferentes subsistemas con el objetivo de mantener versiones activas para un mercado específico, como es el de los Sistemas de Transporte Inteligentes (ITS).

4.3.3. Viabilidad temporal del proyecto

La viabilidad temporal se refiere a la asignación de un tiempo determinado para llevar a cabo la ejecución del proyecto y los plazos de cumplimiento para cada fase del mismo.

Para poder estimar la viabilidad temporal se determinó que el proyecto comenzaría la tercera semana de Febrero de 2022, en concreto el 14 de Febrero, y finalizaría la segunda semana de Junio 2022, en torno al 10 de Junio. Supondría invertir en torno a 5 horas diarias, puesto que no existe disponibilidad total por motivos laborales.

Habiendo establecido las fechas de inicio y fin del proyecto, se estimaron las horas por fase del desarrollo, incluyendo las horas de formación.

Fases de desarrollo	Duración	Horas
Formación	10 días	50 horas
Análisis y especificación de los requisitos	14 días	70 horas
Diseño	14 días	70 horas
Implementación	36 días	180 horas
Validación	10 días	50 horas

Tabla 4.1: Previsión inicial del proyecto.

Una vez determinado el tiempo necesario para cada etapa, es preciso establecer el diagrama de Gantt asociado a todas las decisiones tomadas en lo que respecta a la planificación inicial (la planificación se puede ver en la Figura 4.1 y su representación en la Figura 4.2).

	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Formación	8 días	lun 14/02/22	mié 23/02/22	
2		**Primer prototipo**	**21 días**	**jue 24/02/22**	**jue 24/03/22**	
3		Análisis y especificación	4 días	jue 24/02/22	mar 01/03/22	1
4		Diseño	4 días	mié 02/03/22	lun 07/03/22	3
5		Implementación	10 días	mar 08/03/22	lun 21/03/22	4
6		Validación	3 días	mar 22/03/22	jue 24/03/22	5
7		**Segundo prototipo**	**32 días**	**vie 25/03/22**	**lun 09/05/22**	
8		Análisis y especificación	6 días	vie 25/03/22	vie 01/04/22	6
9		Diseño	6 días	lun 04/04/22	lun 11/04/22	8
10		Implementación	16 días	mar 12/04/22	mar 03/05/22	9
11		Validación	4 días	mié 04/05/22	lun 09/05/22	10
12		**Tercer prototipo**	**21 días**	**mar 10/05/22**	**mar 07/06/22**	
13		Análisis y especificación	4 días	mar 10/05/22	vie 13/05/22	11
14		Diseño	4 días	lun 16/05/22	jue 19/05/22	13
15		Implementación	10 días	vie 20/05/22	jue 02/06/22	14
16		Validación	3 días	vie 03/06/22	mar 07/06/22	15

Figura 4.1: Diagrama de Gantt de la previsión inicial del proyecto (Parte 1).

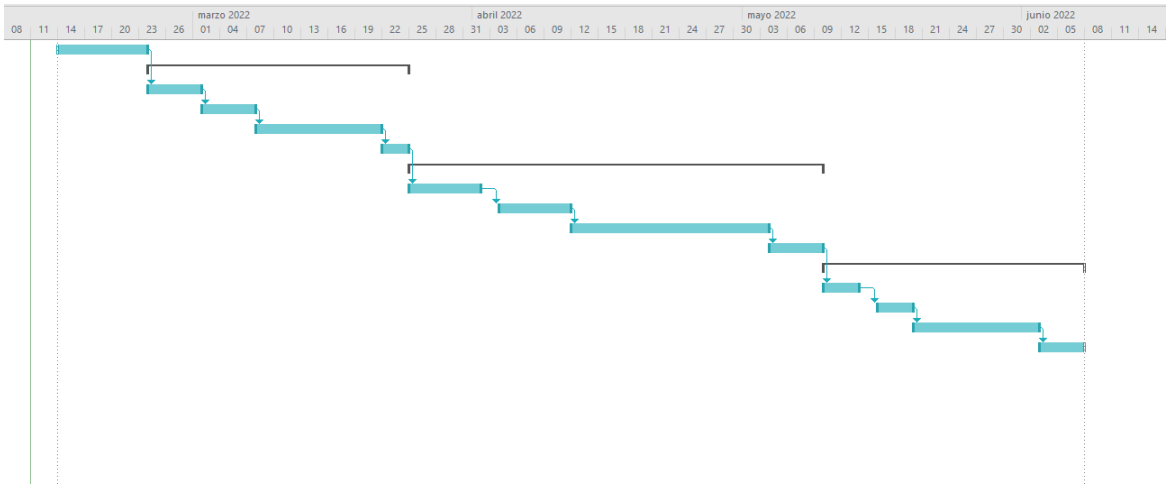


Figura 4.2: Diagrama de Gantt de la previsión inicial del proyecto (Parte 2).

Capítulo 5

Metodología

La página web, la aplicación de Windows y el método de consulta de topología seguirán una metodología de desarrollo o construcción de prototipos, que pertenece a los modelos de proceso evolutivo [26]. Es el método que más se ajusta a las condiciones, tal y como se desarrollará más adelante en este trabajo.

Los modelos evolutivos son iterativos, es decir, se caracterizan por la manera en la que permiten desarrollar versiones cada vez más completas del software. Características:

- No sufren tanto la posibilidad de que los requerimientos, tanto del negocio como del producto, varíen en el tiempo.
- Es importante tener en cuenta que, a veces, los plazos apretados del mercado, hacen imposible el desarrollar un software perfecto.
- Responden a la necesidad de entregar un proyecto, o una parte de él, con la mayor prontitud posible para aliviar la presión o hacer frente a la competencia de mercado existente.
- Estos modelos son interesantes en aquellos proyectos en los que se comprenden bien los requerimientos, pero las extensiones del sistema y los detalles del producto todavía no están muy claros.

5.1. Tipos de prototipos

Existen diferentes tipos de modelos de prototipos que se utilizan dependiendo del tipo de producto a desarrollar o el objetivo que se persigue en el desarrollo.

- Prototipos rápidos o desechables: en este modelo el objetivo es el desarrollo rápido para conseguir una evaluación del modelo con cierta velocidad. Normalmente, esto se hace en aquellos que se quiere estudiar su viabilidad y rentabilidad en el tiempo.
- Prototipos evolutivos: este modelo rompe con la idea prototipo tradicional. Un prototipo evolutivo es una pieza funcional de software, no algo que se pueda probar o enseñar al cliente. La creación de prototipos evolutivos comienza con un producto que cumple únicamente con los requisitos que se comprenden en su plenitud. A medida que evoluciona su desarrollo, tanto el cliente como el desarrollador irán comprendiendo más requisitos, lo que dará lugar a prototipos más completos y complejos.
- Prototipos incrementales: la creación de prototipos incrementales es útil para el software empresarial que tiene muchos módulos y componentes que pueden estar vagamente relacionados entre sí. En la creación de prototipos incrementales, se construyen pequeños prototipos separados en paralelo. Los prototipos individuales se evalúan y refinan por separado, y luego se fusionan en un todo integral.
- Prototipos horizontales: cuentan con la mayoría de las funcionalidades finales del producto final pero en una primera fase de desarrollo. Con este tipo de prototipos se trata de medir el alcance del producto, no su grado de funcionalidad. Un ejemplo de esto por ejemplo podría ser una interfaz con todos los elementos pero sin funcionalidad real.
- Prototipos verticales: son la antítesis de los horizontales, es decir, cuentan con muy pocas funcionalidades del producto final pero en una fase muy avanzada de desarrollo. Un ejemplo podría ser una aplicación móvil que permite enviar mensajes, en la que en el prototipo únicamente incluya mensajes unidireccionales.

5.2. Ventajas

Las principales ventajas son las siguientes:

- La versión final cubrirá las necesidades del cliente, ya que constantemente validará el producto y precisará sus necesidades.
- Los desarrolladores logran un mayor entendimiento del sistema, lo que, a la larga, suele dar lugar a una mayor calidad del código.
- Las especificaciones, necesidades y requerimientos pueden desarrollarse de forma creciente.
- El código es reutilizable.
- Permite detectar problemas antes de lanzar la versión final del producto.

5.3. Inconvenientes

Las principales inconvenientes son las siguientes:

- Con el objetivo de crear prototipos de forma rápida es fácil que se desatiendan aspectos importantes como son la calidad y el mantenimiento a largo plazo.
- No es una metodología recomendable en sistemas grandes y complejos, ya que, desarrollar todas las características puede suponer un problema.
- Puede ocurrir que el prototipo se descarte en su totalidad debido a que la rapidez de desarrollo ha resultado en una mala especificación de requisitos, diseño o implementación.
- Desarrollar cada iteración tomando como punto de partida el prototipo resultado de la iteración anterior puede resultar en un código desestructurado, con defectos y de baja calidad.

5.4. Estructura del proyecto

Antes de continuar, se requiere tomar dos decisiones importantes para el futuro del proyecto:

5.4.1. ¿Qué tipo de prototipos van a ser?

Esta decisión fue sencilla ya que, a partir de las diferentes reuniones con el cliente, quedó clara la necesidad de ver partes funcionales rápidamente. De acuerdo con ello, lo idóneo para este proyecto son prototipos verticales ya que, lo que queremos es que se desarrolle por funcionalidades, pero eso sí, cada una de ellas en profundidad.

Aunque también se podría haber optado por prototipos evolutivos ya que aportarán funcionalidades rápidamente, se tomo la decisión de prototipos verticales porque, al centrarse en partes funcionales, podrían tener objetivos más visibles que los evolutivos, al menos en las primeras fases de desarrollo.

5.4.2. Cantidad de prototipos

En cuanto a la cantidad, no ha sido fácil establecer la división puesto que se pretendía que tuvieran, aproximadamente, la misma carga.

En principio, podría parecer sencillo porque el proyecto consta de tres partes: desarrollo del método de topología en el servidor, creación de la página web para consulta de los dos métodos elegidos (el de topología y el de las paradas) y, por último, creación de un proceso que monitorice las diferentes paradas. Pero, aunque haya tres partes diferenciadas, para que los resultados sean comprobables en el desarrollo del servidor y en la monitorización, es necesario un respaldo con una interfaz.

Por ello, se ha decidido que serán tres prototipos verticales que incluyan una interfaz que respalde la funcionalidad realizada:

- *Primer prototipo*: Supondrá el desarrollo de una página web sencilla y de la implementación del funcionamiento completo de la consulta de topología. Esto incluye la implementación de esta parte en el Web Service ya desarrollado por GMV para el resto de métodos.
- *Segundo prototipo*: la página web, creada en el primer prototipo se mejorará para convertirla en usable y profesional para el testeo de la información de las paradas y para la consulta de topología, además del funcionamiento de la relación entre ellas. Incluirá los requisitos surgidos de la revisión y la validación del primer prototipo.
- *Tercer prototipo*: Consistirá en la adaptación de la página web a la monitorización automática de paradas, y de la implementación del funcionamiento completo de esta parte, que serán una aplicación de Windows y un aplicación de servicios WCF. Incluirá también, los requisitos surgidos de la revisión y la validación del segundo prototipo.

Capítulo 6

Primer prototipo

6.1. Especificación de requisitos

Al principio del proyecto se programaron una serie de reuniones con el cliente para comprender y entender los requisitos software. Para obtener los requisitos del primer prototipo del proyecto se realizaron cuatro reuniones en un plazo de una semana. Cada una de ellas tuvo un objetivo:

1. La primera reunión sirvió para conocer globalmente las necesidades del cliente, lo que permitió acotar más el problema.

2. En la segunda se preparó un mockup de la interfaz para mostrárselo al cliente y así tener una primera opinión.

3. En la tercera se realizaron las modificaciones solicitadas en el mockup y se debatió el alcance del prototipo. En esta, también se estableció el modelo de trabajo en común, programando reuniones semanales para esclarecer las dudas que puedan surgir y mostrar los resultados.

4. En la cuarta, una vez que ya se tenía clara una versión inicial de lo que se quería hacer, se realizó una batería de pruebas para poder comprobar el porcentaje de éxito del prototipo.

Por último, se planificó una sesión de revisión final tras la implementación del prototipo, con el objetivo de comprobar que se ha llegado al alcance y tener una opinión directa del cliente.

Este procedimiento se aplicará también a los otros dos prototipos por lo que únicamente se documentará las partes nuevas o modificadas.

En este primer prototipo únicamente se requiere una interfaz sencilla que permita mostrar las funcionalidades desarrolladas. En este caso, como en los siguientes prototipos, se ha propuesto una página web. La primera versión de esta tendrá los requerimientos básicos.

6.1.1. Requisitos

Los requisitos del proyecto son los servicios y restricciones que debe cumplir la aplicación final. Para poder facilitar la comprensión de esta parte será preciso centrarse únicamente en documentar los requisitos funcionales y no funcionales que afecten a este prototipo.

Antes de entrar en los requisitos se establecerá un vocabulario inicial para poder explicar los diferentes requerimientos:

El **sistema** es un conjunto ordenado de elementos interrelacionados y que interactúan entre sí cuyo objetivo es resolver una necesidad real.

El **usuario** es el elemento del sistema que interactuará con el mismo y al que se destina la resolución de la necesidad planteada.

La **topología** está compuesta por las líneas, trayectos y paradas vigentes que hay en un lugar y momento concreto.

Una **línea** está formada por todos los recorridos planteados o trayectos que existen entre dos zonas o barrios. Una línea normalmente consta de dos **trayectos**, uno de ida y otro de vuelta (Línea 8, con dos trayectos Parquesol-Belén y Belén-Parquesol).

Hay algunas excepciones, como son los **refuerzos**, que son trayectos especiales que se generan para suplir las necesidades de movilidad ciudadana que pueda haber (Trayectos Búho).

6.1.1.1. Requisitos funcionales

Los requisitos funcionales son las funcionalidades que debe presentar y, los que afectan a este primer prototipo, son:

P1-RF1: *El sistema deberá permitir a los usuarios consultar la topología.*

P1-RF2: *El sistema deberá permitir a los usuarios ver los trayectos de cada línea.*

P1-RF3: *El sistema deberá permitir a los usuarios ver las paradas de cada trayecto.*

P1-RF4: *El sistema deberá permitir a los usuarios cambiar la información de idioma.*

6.1.1.2. Requisitos no funcionales

Los requisitos funcionales son las restricciones que debe cumplir un sistema software y, los que afectan a este primer prototipo, son:

P1-RNF1: *El sistema debe ser capaz de consultar los datos del sistema en menos de 30 segundos.*

P1-RNF2: *El sistema debe ser capaz de consultar los trayectos de cada línea en menos de 1 segundo.*

P1-RNF3: *El sistema debe ser capaz de consultar las paradas de cada trayecto en menos de 2 segundos.*

6.1.2. Batería de pruebas

Para poder comprobar que se han alcanzado los requisitos del prototipo, se va a preparar una batería de pruebas por prototipo, que incluirá una serie de pruebas de caja negra, es decir, no se entrará en detalles de como se ha logrado o como funciona, únicamente se comprobará que ha tenido el resultado esperado. En función de la cantidad de pruebas que sean exitosas se podrá valorar si se han cumplido los requisitos solicitados por el cliente.

Hay muchas formas de realizar una batería de pruebas de caja negra, pero la más acorde a las que se van a realizar es en formato de tabla.

Id	Descripción	Propuesta por	Resultado
P1-P01	Consultar topología (líneas, trayectos y paradas vigentes)	Cliente	Muestra la lista de lineas, trayectos y paradas
P1-P02	Ver línea	Cliente	Lista todos los trayectos asociados a una línea
P1-P03	Ver trayecto	Cliente	Lista todas las paradas asociadas a un trayecto
P1-P04	Cambiar idioma, permitirá cambiar del idioma local al inglés y viceversa	Cliente	Cambiará de idioma todas las cadenas referentes a la topología de idioma
P1-P05	Comprobar que se podrán seleccionar tantas líneas y trayectos como se quiera	GMV	Se deberán listar todas los trayectos por línea y todas las paradas por trayecto que se hayan seleccionado

Tabla 6.1: Batería de pruebas del primer prototipo de la aplicación (P1).

6.2. Análisis

El análisis del problema funciona a modo de .^enunciado del problema"para el que diseñaremos nuestro sistema, y permitirá conocer el contexto en que este sistema debe funcionar y qué deberá ser capaz de hacer.

El resultado de la fase de análisis es un conjunto de modelos que especifican cómo ha de comportarse el sistema desde un punto de vista externo. Estos modelos se realizarán utilizando UML (Unified Modeling Language), que es el modelo de lenguaje unificado más conocido y utilizado en la actualidad. Este estándar permite realizar esquemas, diagramas y documentación relativa a los desarrollos de software. Es importante destacar que llamar a UML lenguaje no es çorrecto", ya que es un conjunto de normas y estándares gráficos.

Dentro de UML existen distintos tipos de diagramas e informes que permiten analizar los requisitos de un sistema software. En este caso, se utilizarán diagramas de casos de uso para describir las acciones que se podrán realizar en el sistema desde el punto de vista del usuario. Los casos de uso se componen de un conjunto de posibles secuencias de interacciones entre sistemas y usuarios.

6.2.1. Diagrama de casos de uso

El diagrama de casos de uso para este primer prototipo se muestra en la Figura 6.1:

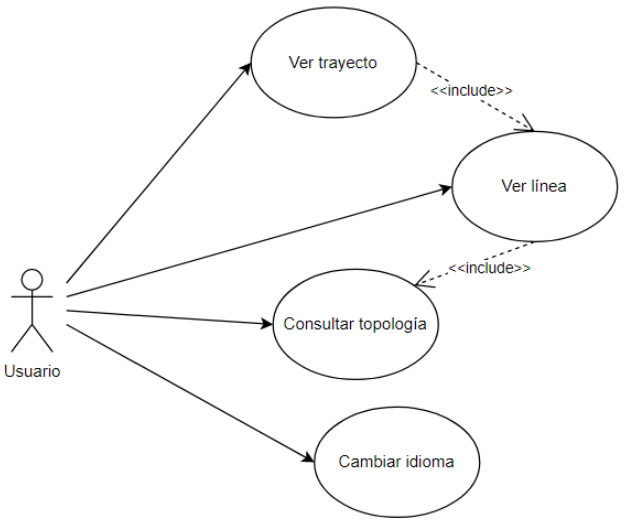


Figura 6.1: Modelo de casos de uso obtenido en el análisis del primer prototipo de la aplicación.

En este momento, es preciso especificar cada uno de los casos de uso. Para ello se dividirá cada uno de ellos en varios puntos: nombre del caso de uso, breve descripción, actor principal, condiciones previas, flujo de eventos y condiciones posteriores. Para poder recoger toda esta información se utilizará una tabla por caso de uso:

P1-CU01	Caso de uso: Consultar topología
Breve descripción	Consultar todos los datos de líneas, trayectos y paradas
Actor	Usuario
Precondiciones	-
Flujo de eventos	1. El sistema muestra una vista con varios botones, entre ellos el que permite consultar la topología 2. Pulsar el botón de consultar topología
Postcondiciones	Aparecerá toda la información consultada en forma de árbol

Tabla 6.2: Caso de uso: Consultar topología (P1-CU01).

P1-CU02	Caso de uso: Ver línea
Breve descripción	Consulta todos los trayectos de la línea seleccionada
Actor	Usuario
Precondiciones	Haber consultado la topología
Flujo de eventos	1. El sistema muestra una lista de líneas al usuario 2. Buscar la línea que se desea 3. El usuario pulsará en dicha línea
Postcondiciones	Se mostrarán todos los trayectos asociados a esa línea (normalmente son dos, ida y vuelta)

Tabla 6.3: Caso de uso: Ver línea (P1-CU02).

P1-CU03	Caso de uso: Ver trayecto
Breve descripción	Consulta todas las paradas del trayecto solicitado
Actor	Usuario
Precondiciones	▪ Haber consultado la topología ▪ Haber seleccionado una línea
Flujo de eventos	1. El sistema muestra una lista de los trayectos al usuario 2. Buscar el trayecto que se desea 3. Pulsar en ese trayecto
Postcondiciones	Se mostrarán todos los trayectos asociados a esa línea (normalmente son dos, ida y vuelta)

Tabla 6.4: Caso de uso: Ver línea (P1-CU03).

P1-CU04	Caso de uso: Cambiar idioma
Breve descripción	Cambia entre los diferentes idiomas que tengan disponibles cada línea, trayecto y parada. Normalmente es el local de la zona y inglés
Actor	Usuario
Precondiciones	Haber consultado la topología
Flujo de eventos	1. El sistema muestra una vista con varios botones, entre ellos el que permite cambiar el idioma 2. Pulsar el botón de cambiar idioma
Postcondiciones	Se mostrarán todos los datos de líneas, trayectos y paradas con el idioma cambiado

Tabla 6.5: Caso de uso: Cambiar idioma (P1-CU04).

6.2.1.1. Modelo de dominio

Para terminar la etapa del análisis, será necesario establecer cuáles son los objetos de interés y cómo se relacionan entre ellos. La herramienta utilizada en el proceso es la que se denomina modelo de dominio, que es una representación de las clases conceptuales del mundo real. En él se describen las distintas entidades, sus atributos, papeles y relaciones, además de las restricciones que rigen el dominio del problema. Es importante aclarar que el modelo de dominio representa clases conceptuales, no componentes de software.

A lo largo del tiempo ha habido muchas discusiones sobre este tema. Aclarar esta polémica no es objetivo del TFG, por lo que simplemente se apuntará que, lo que se modela en esta parte son clases conceptuales, es decir, entidades reales. En cambio, los componentes software son unidades modulares que pueden incluir una o más clases y que están más relacionados con la etapa de diseño.

El modelo de dominio para este primer prototipo es:

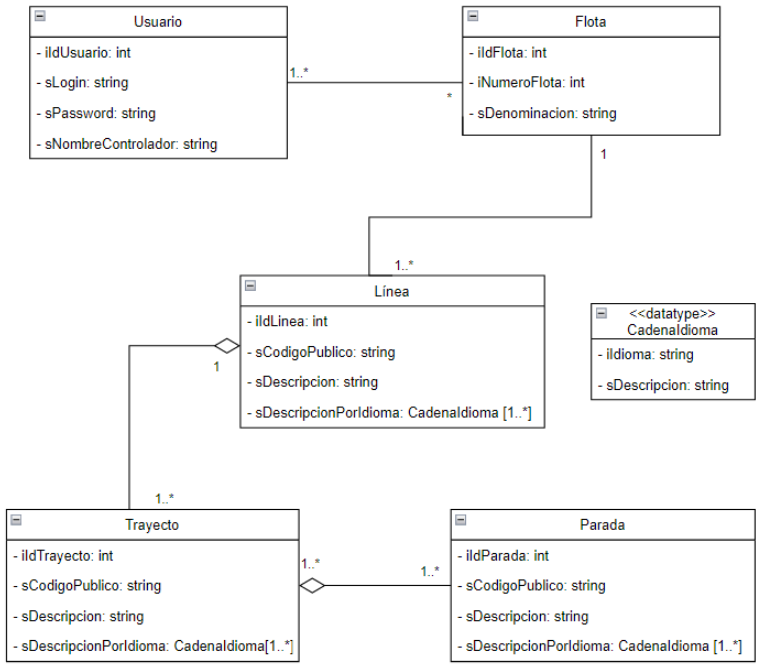


Figura 6.2: Diagrama de Clases del Modelo de Dominio en análisis del primer prototipo.

Aunque el modelo de dominio habla por sí solo, merece la pena comentar el tipo compuesto CadenaIdioma. Una de las opciones que se ofrece en los casos de uso es cambiar el idioma de la topología, por lo que hay que añadir las posibles cadenas que existan en diferentes idiomas. Para ello, y como de lo que se dispone es del idioma y cada cadena, la mejor forma de modelarlo es con un DataType que tenga dos strings, uno que nos sirva para identificar el idioma y el segundo que es el nombre de la línea, trayecto o parada en ese idioma.

Un ejemplo explicativo sería el siguiente:

```
L: L1 - City Centre L1
L: L1 - (el) Κέντρο της πόλης Λ1
```

En este caso como se puede ver en la Figura **??** la segunda cadena tiene un (el), que significa griego. Para establecer la cadena que identifica cada idioma se utilizó el estándar ISO 639-2 que determina entre otras cosas la cadena que identifica a cada país. Uno mucho más conocido y al uso es el 'en' que significa inglés o el 'es' que significa español.

6.2.1.2. Diagrama de secuencia

En este punto se mostrará el diagrama de secuencia del caso de uso más relevante en cada prototipo. Se hará así porque los casos de uso que únicamente modifican elementos de la vista serían demasiado cortos y no aportarían nada al documento. En este caso se hará el diagrama de secuencia del CU Consultar topología (P1-CO1), que se encuentra en la Figura 6.3.

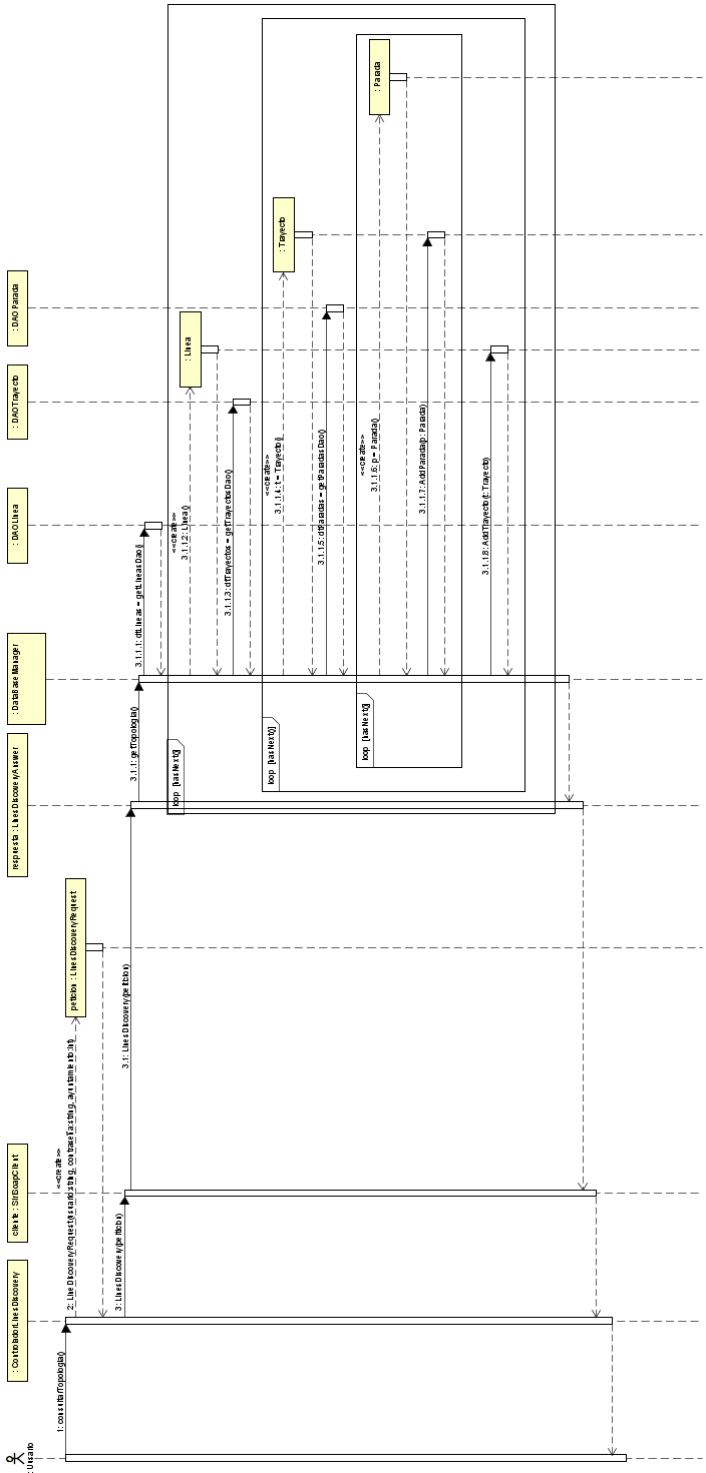


Figura 6.3: Diagrama de secuencia del caso de uso CU01.

6.3. Diseño

En esta fase se estudian las posibles opciones de implementación para el software que hay que construir, así como decidir la estructura general del mismo. Esta etapa es compleja y, por ello, hay posibilidades de que la primera solución propuesta no sea la definitiva. Para evitar que esto pueda pasar se hace de forma iterativa y mediante patrones, que recogen buenas praxis, lo que permite evitar posibles errores futuros.

6.3.1. Arquitectura del sistema

En la arquitectura del sistema se decide cuál va a ser la estructura general del sistema. Este punto es el que precede a los diagramas que darán lugar después al desarrollo.

Una particularidad de este prototipo es que, aunque consulta datos reales de una BD situada en un servidor de producción, no se insertarán datos hasta el último prototipo, el dedicado a la monitorización. Además, como gran parte del Web Service del que provienen los datos ya está desarrollado, para este TFG esa parte será, salvo en el caso de la consulta de topología, una caja negra. Realmente, aunque estén desarrollados casi todos los métodos que implementan el estándar, en este proyecto solo se va a utilizar uno de ellos. El otro método que trataremos sí se desarrollará por completo.

6.3.2. Arquitectura de tres capas

Como ya se ha comentado, para este prototipo no va a haber una persistencia de datos, pero aun así se necesita una capa de persistencia para dar acceso a datos persistentes. Por ello, hablaremos de Arquitectura de tres capas [2]:

- **Capa de presentación**: Es la que interactúa el usuario, por lo que también se denomina capa de usuario. Se encarga de presentar el sistema al usuario, de comunicarle información y de atender los eventos que puede este generar (pulsar botones, insertar datos...). Se detalla en la Sección 6.3.2.1.

- **Capa de lógica de negocio**: En ella se encuentran los requerimientos funcionales del sistema, es decir, las reglas de negocio, el workflow del negocio y todas las operaciones no persistentes que pueda haber. Es importante, ya que todas las operaciones persistentes pertenecen a la capa de acceso a datos.

- **Capa de acceso a datos**: Aquí se encuentran los datos persistentes, que deben ser almacenados y accesibles con el paso del tiempo. Para ello deberá utilizarse una Base de Datos.

6.3.2.1. Capa de presentación

En la capa de presentación se implementará el patrón de diseño arquitectónico MVC. Se ha tomado esta decisión porque para una página web es la mejor forma de que la aplicación sea escalable, mantenible y fácil de expandir. Además, como se va a hacer en C#, se puede utilizar el completo marco de trabajo ASP.NET Core MVC.

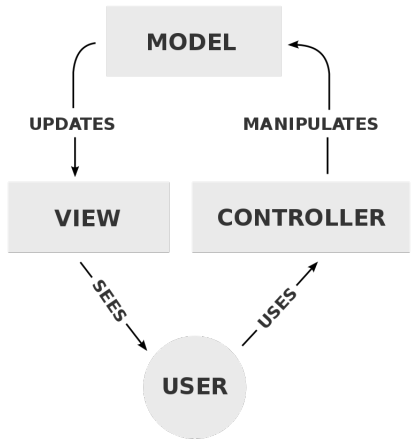


Figura 6.4: Representación visual del patrón Model-View-Controller (MVC) (fuente: Wikipedia)[1].

Se utilizará MVC porque es el que más se acopla a las necesidades: el modelo de las clases que se van a mostrar, una vista para poder intercambiar, mostrar información al usuario y corresponder con los eventos con los que interactúe y, por último, un controlador que permita responder a los eventos, modificar el modelo, etc.

Es importante diferenciar la Arquitectura de Tres Capas del MVC [23], puesto que, en ocasiones se confunden y, sin embargo, son muy diferentes:

- **Arquitectura por capas**: La arquitectura de tres niveles se refiere a un nivel topológicamente más alto, entre diferentes sistemas. Otra gran diferencia es su flujo de comunicación: en la arquitectura de tres niveles el flujo es lineal. De la capa de presentación al negocio, del negocio a los datos y al revés.
- **MVC**: El patrón de diseño MVC se refiere al código de una aplicación (es decir, el software) en sí mismo, lo que el usuario ve y con lo que interactúa. El flujo de comunicación en el MVC, en cambio, es triangular. La Vista envía comandos al Controlador, que actualiza el Modelo, y la Vista accede a los nuevos datos del Modelo.

Para este primer prototipo, a partir de la información que proporcionó el cliente, se construyó un mockup. Este boceto se planteó en las primeras reuniones y cuenta con la aprobación del cliente.

[1] `https://en.wikipedia.org/wiki/Model-view-controller#/media/File:MVC-Process.svg`

Es una versión primaria de la web, como ya se apuntó, la que se incluye en este primer prototipo. Esta primera versión será el puente a lo que luego se desarrollará para los otros dos prototipos, por lo que se hará de la forma más fiel posible al objetivo final.

La información mostrada en el mockup es, como se planteó que podía ser, una buena forma de reflejar la topología. Entre las propuestas planteadas estaba la de una tabla, que no reflejaba del todo la relación entre líneas, trayectos y paradas. Y un árbol, que es perfecto para poder anidar la información.

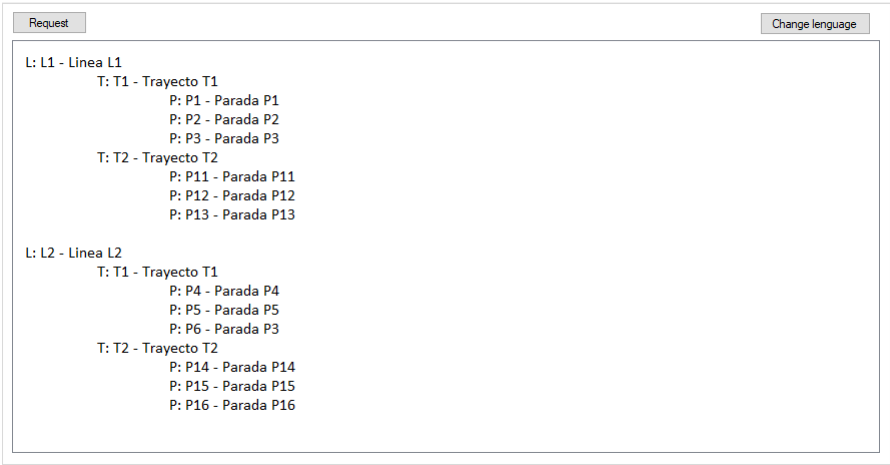


Figura 6.5: Mockup de la vista para la consulta de topología (Primera Versión).

6.3.2.2. Capa de lógica de negocio

Esta capa se dará lugar en un WS que implementará el estándar europeo SIRI para poder intercambiar información con la vista. En esta parte, estará toda la lógica que se encargue de modificar y consultar los datos. Además, será la que esté directamente relacionada con la capa de persistencia.

6.3.2.3. Capa de persistencia

Para la capa de persistencia, durante el desarrollo se utilizará una BD que estará en nuestra máquina local pero, una vez terminado el desarrollo y desplegado (que será al finalizar el primer prototipo), utilizaremos la base de datos real de producción. Está parte es urgente, porque permitirá poder usar datos reales de los autobuses funcionando, lo que es totalmente necesario para el caso del segundo y tercer prototipo.

Antes de terminar esta etapa de diseño, se adjuntará un diagrama de despliegue para mostrar de forma gráfica la arquitectura del sistema [27]

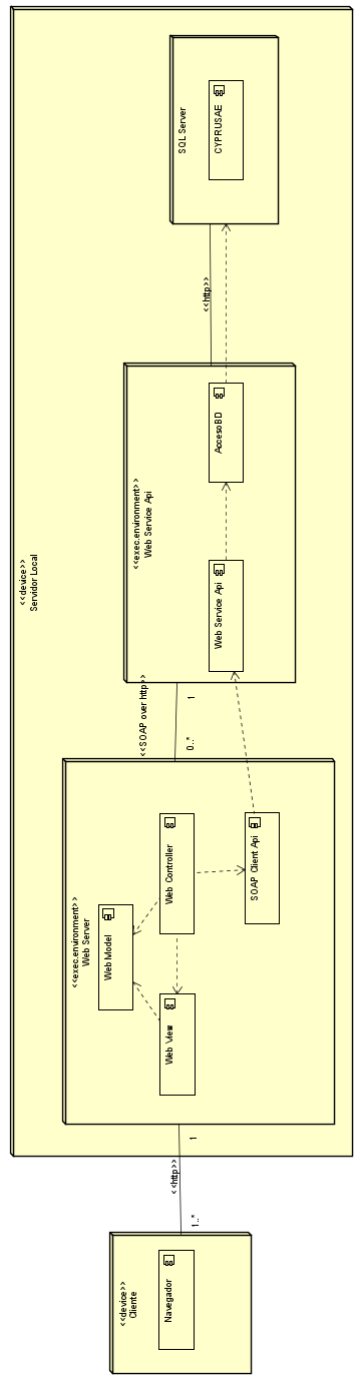


Figura 6.6: Diagrama de despliegue del primer prototipo.

Este diagrama se realizará también para el resto de prototipos, con la intención de mostrar cómo evoluciona el proyecto. La idea es que, partiendo del diagrama del prototipo anterior, se pueda observar la evolución del siguiente desde el punto de vista de la arquitectura.

6.4. Implementación

En este momento del trabajo, queremos dejar constancia de las diferentes barreras que han ido surgiendo y de cómo se han ido resolviendo.

6.4.1. Desarrollo y desafíos de la implementación

El primer gran desafío fue entender el funcionamiento de la lógica del WS desarrollado por GMV ya que es código legacy que tiene que ser ejecutado en VS2015. Una vez obtenido el código en local, se pudo trastear con él. La lógica no era muy diferente a las versiones posteriores en las que sí habíamos podido programar. La versión en la que se hizo este desarrollo, al que añadimos la implementación del método Lines Discovery está en .NET Framework 4. Una versión muy anterior a la que se utilizará para la página web que será .NET Core 3.1, que además tiene compatibilidad a largo plazo.

Una vez tuvimos la aprobación de GMV para el desarrollo, nos pusimos manos a la obra. En primer lugar había que adaptar el estándar SIRI desde este lado, es decir, añadir la referencia de servicio WCF a partir del wsdl que se genera al publicar el Web Service de GMV en el IIS.

WSDL (Web Services Description Language [6]) es una especificación estándar para describir servicios basados en XML de red. Proporciona a los proveedores de servicios un modo sencillo de describir el formato básico de las peticiones a sus sistemas independientemente de la implementación del motor de ejecución subyacente.

A partir de un wsdl, descargado de la página original del estándar, en el proyecto original obtuvieron las diferentes interfaces de los métodos SOAP así como las estructuras del mismo. A partir de este wsdl se generaron automáticamente el ISiriProducerDocBinding20g que contendría todas las interfaces de los métodos y el SiriServices20g que tendría todas las estructuras necesarias para el intercambio de información. En ambos se observa el sufijo 20g, que indica la versión. El punto más interesante del estándar es que las estructuras que proporciona permiten compartir grandes cantidades de información, pero que también da la opción de no utilizar. De hecho, en este trabajo no utilizaremos muchos de los campos que proporciona porque, para lo que solicita el cliente, no son necesarios.

SOAP (originalmente las siglas de Simple Object Access Protocol [21]) es un protocolo estándar que define cómo dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML[3].

Una vez conocido el trabajo requerido y la interfaz del método, junto con los parámetros y estructura a desarrollar fue sencillo.

El desarrollo del trabajo fue el siguiente:

1. En primer lugar, a partir del parámetro, pudimos conocer la estructura en la que recibía los datos de la petición. Esta estructura se denomina WsLinesDiscoveryStructure
2. En segundo lugar, comprobamos qué datos de la estructura realmente se necesitaban, ya que para la versión más básica únicamente con un usuario y contraseña que tuviera acceso a los datos valía. Obviamente, después ya se utilizarían más campos de la estructura, como puede ser el municipio a elegir.
3. En tercer lugar, fue preciso utilizar la lógica para relacionar los datos recibidos con el DAO que permite acceder a la BD.
4. En cuarto lugar, plantear la posible consulta que había que relacionar la tabla de líneas con la de trayectos y paradas
5. En quinto y último lugar, rellenar con los datos recibidos de la consulta la estructura de respuesta que devolverá el método, que se denomina WsLinesDiscoveryAnswerStructure.

Para probar toda esta parte hubo que utilizar la herramienta SoapUI, que permite simular ser un cliente del WS. Aunque no fue todo lo claro que se podía esperar, permitió ver el XML que se intercambiaba en el que ver los datos.

Lo bueno en este caso es que, como se utilizó un backup de una BD real para crear este método, siempre tuvimos las tablas con las columnas y datos necesarios. Esto permitió que, únicamente conociendo las tablas de líneas, trayectos y paradas de la versión topológica actual (terminología que utilizamos para saber que líneas, trayectos y paradas están en uso realmente), se pudiera consultar toda la información.

Una vez desarrollada esta parte ya se podía empezar con el verdadero gran reto: probar el funcionamiento del WS. Para ello creamos un proyecto ASP.NET Core MVC [22] en la versión 3.1 para asegurar la compatibilidad a largo plazo.

Para poder crear una página web que consultase al WS soap, se necesitaba importar el wsdl del WS al que se va a conectar. Como, a ojos de la página web, solo se verá que es un WS Soap y las estructuras de datos del SIRI, cualquier WS que cumpla estas condiciones podrá utilizarla como cliente. No fue tarea fácil, hubo que buscar información y documentarse durante bastante tiempo para entender como importar el wsdl.

Una vez estuvimos conectados al WS Soap a través del endpoint, lo que al estar ambos en nuestra máquina local fue sencillo, ya se podía comenzar con la capa de presentación.

El que fuese MVC en C# permitió utilizar con mayor facilidad la tecnología Razor [24], que es una sintaxis de marcado para insertar código basado en .NET en páginas web. Razor permite, básicamente, utilizar variables de C# en páginas HTML. Esta herramienta y el patrón arquitectónico de diseño permitió poder, teniendo un ViewModel, intercambiar información rápidamente entre el Controlador y la Vista.

Lo primero que había que hacer era crear la lógica necesaria en el controlador para solicitar la información al WS e imprimir los datos en la web. Cuando logramos esa parte, se pudo ya plantear el árbol en el mockup. Esto requería de conocimientos de JavaScript, por lo que fue preciso algo de tiempo para adaptarnos y lograrlo.

El último reto fue el caso de uso de Cambio de idioma, ya que, como los datos que se recibían estaban en varios idiomas, simplemente se tenía que permitir cambiar entre uno y otro. El problema era que al pulsar el botón de cambio de idioma se generaba un nuevo controlador, que ya no tenía el ViewModel con la topología. Para solucionarlo, lo que hubo que hacer fue utilizar el concepto de sesión. A partir de ahora se guardarían los datos consultados en la sesión, lo que permitiría no tener que volver a consultar la topología en cada ocasión y así poder cambiar el idioma rápidamente.

6.5. Validación

Para la validación del prototipo se programó una reunión con el cliente, en la que se le presentaron todas las funcionalidades. Quedó muy satisfecho, sobre todo con la estructura de árbol que era muy agradable y cómoda. A lo largo de esta reunión también se pasó una vez más la batería de pruebas.

Id	Descripción	Propuesta por	Resultado	Éxito
P1-P01	Consultar topología (líneas, trayectos y paradas vigentes)	Cliente	Muestra la lista de lineas, trayectos y paradas	SÍ
P1-P02	Ver línea	Cliente	Lista todos los trayectos asociados a una línea	SÍ
P1-P03	Ver trayecto	Cliente	Lista todas las paradas asociadas a un trayecto	SÍ
P1-P04	Cambiar idioma, permitirá cambiar del idioma local al inglés y viceversa	Cliente	Cambiará de idioma todas las cadenas referentes a la topología de idioma	SÍ
P1-P05	Comprobar que se podrán seleccionar tantas líneas y trayectos como se quiera	GMV	Se deberán listar todas los trayectos por línea y todas las paradas por trayecto que se hayan seleccionado	SÍ

Tabla 6.6: Batería de pruebas comprobada del primer prototipo de la aplicación (P1).

Entre los aspectos a mejorar de la vista que se muestra en la Figura 6.7 se habló de la cercanía de los botones y los "particularesçolores elegidos para la cabeceara y fondo. Todos estos aspectos no entraban dentro del prototipo, ya que se iba a desarrollar una interfaz lo más sencilla posible. Aun así, se rediseñará la interfaz de la página porque se utilizará también en el siguiente prototipo.

En la revisión del prototipo con el tutor, se propusieron alternativas muy interesantes. La primera fue la de modificar el carácter de inicio en las líneas (L), trayectos (J) y paradas (S) por una combinación de signos presentes en el teclado de la computadora u ordenador, con la que se exprese gráficamente lo mismo, pero de una forma más visual. También, la presencia de una letra muy pequeña para la cantidad de espacio que hay. Y otra serie de mejoras en la memoria que se implementaron.

Después de las revisiones se realizaron las mejoras sencillas que se plantearon, el resto se realizarán en el siguiente prototipo. Estas mejoras se pueden observar en la Figura 6.8.

Figura 6.7: Vista Lines Discovery una vez terminada este primer prototipo.

Figura 6.8: Vista Lines Discovery después de las mejoras planteadas en las revisiones.

Capítulo 7

Segundo prototipo

En este segundo prototipo y en el tercero y último, muchos puntos se repetirían, por lo que nos centraremos en documentar únicamente las partes nuevas y las modificaciones de lo ya realizado.

7.1. Especificación de requisitos

7.1.1. Interfaces de usuario

En el segundo prototipo se requiere una nueva vista para la página web que permita monitorizar las paradas. Con ella, se podrá probar otro de los métodos que implementan el estándar SIRI, uno ya desarrollado previamente en la empresa. También se incluirá la relación entre la vista del primer prototipo y esta nueva que se va a desarrollar.

7.1.2. Requisitos

7.1.2.1. Requisitos funcionales

Los requisitos funcionales que afectan a este segundo prototipo son:

P2-RF1: *El sistema deberá permitir a los usuarios monitorizar la información de la parada elegida en tiempo real a partir de su código público.*

P2-RF2: *Deberá también permitir monitorizar la información en tiempo real de la parada elegida a partir de una referenica o enlace de cada parada en la consulta de topología.*

P2-RF3: *Además, deberá permitir monitorizar varias paradas a la vez, a partir del botón añadir de la consulta de topología.*

P2-RF4: *Deberá permitir combinar las tres formas diferentes de consulta de información en tiempo real.*

P2-RF5: *El sistema deberá permitir al usuario poder abrir más de una pestaña web con información diferente para poder consultar más paradas al mismo tiempo.*

P2-RF6: *Deberá también permitir cambiar el idioma de la información mostrada.*

P2-RF7: *El sistema deberá recargar la información de las paradas en tiempo real.*

7.1.2.2. Requisitos no funcionales

Los requisitos no funcionales que afectan a este segundo prototipo son:

P2-RFN1: *También debe ser capaz de consultar los datos del sistema en menos de 30 segundos, dependiendo de la cantidad de paradas consultadas.*

P2-RFN2: *El sistema debe ser capaz de admitir hasta 5 pestañas web con información diferente.*

P2-RFN3: *Debe ser capaz, además, de añadir al menos 10 paradas mediante el botón añadir de la consulta de la topología. A estas 10 paradas hay que sumarle una más que se consulta manualmente por su código público o por su enlace. Esto sumando a que se permiten hasta 5 pestañas posibles, permitirá consultar un máximo de 55 paradas al mismo tiempo.*

P2-RFN4: *El sistema deberá permitir intercalar entre la consulta de topología y la monitorización de paradas.*

7.1.3. Batería de pruebas

En la batería de pruebas de este prototipo, además de añadir todas las pruebas asociadas a este, se añadirán aquellas nacidas del conocimiento de la interfaz y funcionalidades asociadas al primer prototipo.

Todas las pruebas que se denominen con P1 harán referencia al primer prototipo y, con ello, a la vista de la consulta de topología. En cambio, todas las pruebas que tengan P2 podrán hacer referencia a tanto la vista de topología como a la de monitorización manual de paradas. Para poder esclarecer esta situación se incluirá CT en el caso de que afecte a la vista de consulta de topología y MMP en caso de que sea esta segunda. En caso de afectar ambas y por tanto a la relación entre ellas se especificará CT/MMP, estableciendo así, también el orden de cuál iría primero y cuál después.

Id	Descripción	Propuesta por	Resultado
P1-P06 (CT)	Comprobar que se cumple la estructura de árbol en las líneas, trayectos y paradas	GMV	Lista todas las paradas asociadas a un trayecto
P2-P01 (MMP)	Consultar una parada escribiendo su código publico	Cliente	Lista todas las líneas de petición de la parada solicitada
P2-P02 (CT/MMP)	Consultar una parada pulsando en la parada que se quiera en la vista de la consulta de topología	Cliente	Lista todas las líneas de petición de la parada solicitada
P2-P03 (CT/MMP)	Consultar una parada añadiendola a la lista de paradas solicitadas por puesto	Cliente	Lista todas las líneas de petición de la parada solicitada
P2-P04 (MMP)	Cambiar idioma, permitirá cambiar del idioma local al inglés y viceversa	Cliente	Cambiará de idioma todas las cadenas referentes a la petición
P2-P05 (CT/MMP)	Consultar más de una parada añadiéndolas a la lista de paradas solicitadas por puesto	Cliente	Lista todas las líneas de petición de la paradas solicitadas, distinguiendo entre cada una de ellas
P2-P06 (CT/MMP)	Consultar hasta diez paradas añadiéndolas a la lista de paradas solicitadas por puesto	Cliente	Lista todas las líneas de petición de la paradas solicitadas, distinguiendo entre cada una de ellas
P2-P07 (CT/MMP)	Consultar que no se pueden consultar más de diez paradas añadiéndolas a la lista de paradas solicitadas por puesto	GMV	Lista todas las líneas de petición de la paradas solicitada, distinguiendo entre cada una de ellas
P2-P08 (CT/MMP)	Eliminar la lista de paradas solicitadas en un puesto	Cliente	Comprobar que ya no se listan las líneas de petición de las paradas previamente solicitadas en ese puesto
P2-P09 (CT/MMP)	Cambiar de puesto	GMV	Comprobar que el número de puesto ha cambiado
P2-P10 (CT/MMP)	Una vez se ha cambiado de puesto, realizar las pruebas P01, P02, P03, P04, P05, P06 y P07	GMV	Lista todas las líneas de petición de la paradas solicitadas, distinguiendo entre cada una de ellas
P2-P11 (CT/MMP)	Eliminar la lista de paradas solicitadas en más de un puesto	GMV	Comprobar que ya no se listan las líneas de petición de las paradas previamente solicitadas en esos puestos

Tabla 7.1: Batería de pruebas del segundo prototipo de la aplicación (P2).

7.2. Análisis

En esta parte, se partirá del diagrama de casos de uso y de dominio del anterior prototipo para aumentarlo hasta el alcance de éste. El mismo proceso se realizará en el último prototipo para tener los diagramas del subsistema completo. Eso sí, los casos de uso ya especificados no se volverán a repetir, excepto en el caso de que se añada un flujo alternativo u otros pasos.

7.2.1. Diagrama de casos de uso

El diagrama de casos de uso para este segundo prototipo se muestra en la Figura 7.1 es:

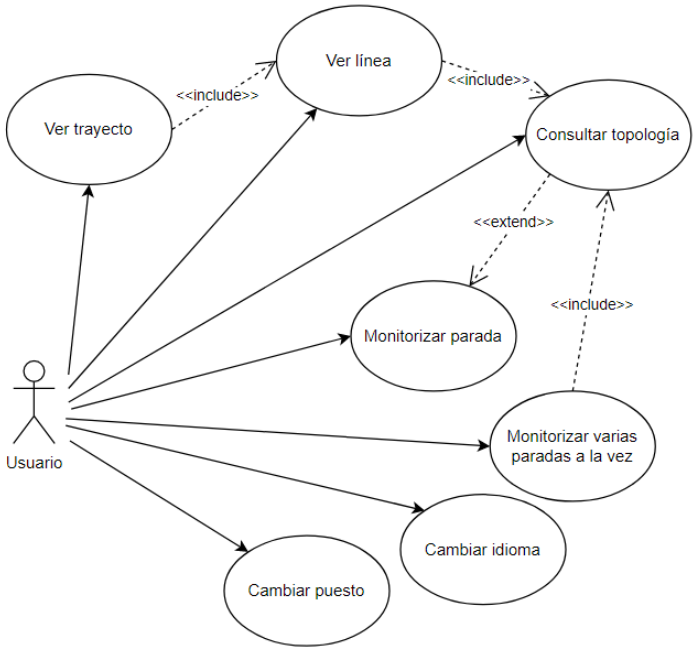


Figura 7.1: Modelo de casos de uso obtenido en el análisis del segundo prototipo de la aplicación.

Ahora se especificará cada uno de los casos de uso:

P2-CU01	Caso de uso: Monitorizar parada.
Breve descripción	Permite monitorizar una única parada, es decir, permite saber las líneas que pasan por esa parada, el tiempo que tarda en llegar el bus, la hora consultada, la hora estimada de llegada a la parada...
Actor	Usuario
Precondiciones	Flujo básico: (véase a continuación) Flujo alternativo: ▪ Haber consultado la topología. ▪ Haber seleccionado una línea. ▪ Haber seleccionado un trayecto.
Flujo de eventos	Flujo básico: 1. El sistema muestra un campo de texto para ingresar el código público de la parada y un botón para solicitar monitorizar la parada. 2. El usuario ingresa el código público de la parada. 3. El usuario pulsa el botón para monitorizar la parada. Flujo alternativo: 1. El sistema muestra una lista de paradas que son enlaces. 2. El usuario pulsa el enlace de la parada que quiera solicitar.
Postcondiciones	El sistema mostrará la información que permite monitorizar la parada en forma de tabla.

Tabla 7.2: Caso de uso: Monitorizar parada (P2-CU01).

P2-CU02	Caso de uso: Monitorizar varias paradas a la vez
Breve descripción	Permite monitorizar más de una parada a la vez.
Actor	Usuario
Precondiciones	▪ Haber consultado la topología. ▪ Haber seleccionado una línea. ▪ Haber seleccionado un trayecto.
Flujo de eventos	Flujo básico: 1. El sistema muestra una lista de paradas que tienen un botón que nos permite añadir el código público a una lista. 2. El usuario añade las paradas que desee a una lista. 3. El usuario cambia a la vista de Monitorización de paradas.
Postcondiciones	El sistema permitirá monitorizar todas las paradas seleccionadas.

Tabla 7.3: Caso de uso: Monitorizar varias paradas a la vez (P2-CU02).

P2-CU03	Caso de uso: Resetear las paradas
Breve descripción	Permite eliminar las paradas solicitadas. en un puesto
Actor	Usuario
Precondiciones	▪ Haber consultado la topología. ▪ Haber seleccionado una línea. ▪ Haber seleccionado un trayecto.
Flujo de eventos	Flujo básico: 1. El sistema muestra una lista de paradas que tienen un botón que nos permite añadir el código público a una lista. 2. El usuario añade las paradas que desee a una lista. 3. El usuario cambia a la vista de Monitorización de paradas.
Postcondiciones	El sistema permitirá monitorizar todas las paradas seleccionadas.

Tabla 7.4: Caso de uso: Resetear las paradas (P2-CU03).

P2-CU04	Caso de uso: Cambiar de puesto
Breve descripción	Permite poder cambiar de puesto para poder consultar más de 10 paradas. Además, también al ser puestos diferentes permite poder monitorizar más paradas utilizando otra pestaña.
Actor	Usuario
Precondiciones	-
Flujo de eventos	Flujo básico: 1. El sistema muestra una vista con varios botones, entre ellos el que permite cambiar de puesto. 2. El usuario pulsa el botón de cambiar puesto. 3. El usuario cambia a la vista de Monitorización de paradas o pulsa una parada para Monitorizarla (caso de uso Monitorizar parada con el flujo alternativo).
Postcondiciones	El sistema permitirá ahora añadir paradas a un nuevo puesto. Esto se podrá hacer hasta 4 veces, siendo un máximo de 5 puestos.

Tabla 7.5: Caso de uso: Cambiar de puesto (P2-CU04).

7.2.1.1. Modelo de dominio

El modelo de dominio para este segundo prototipo es:

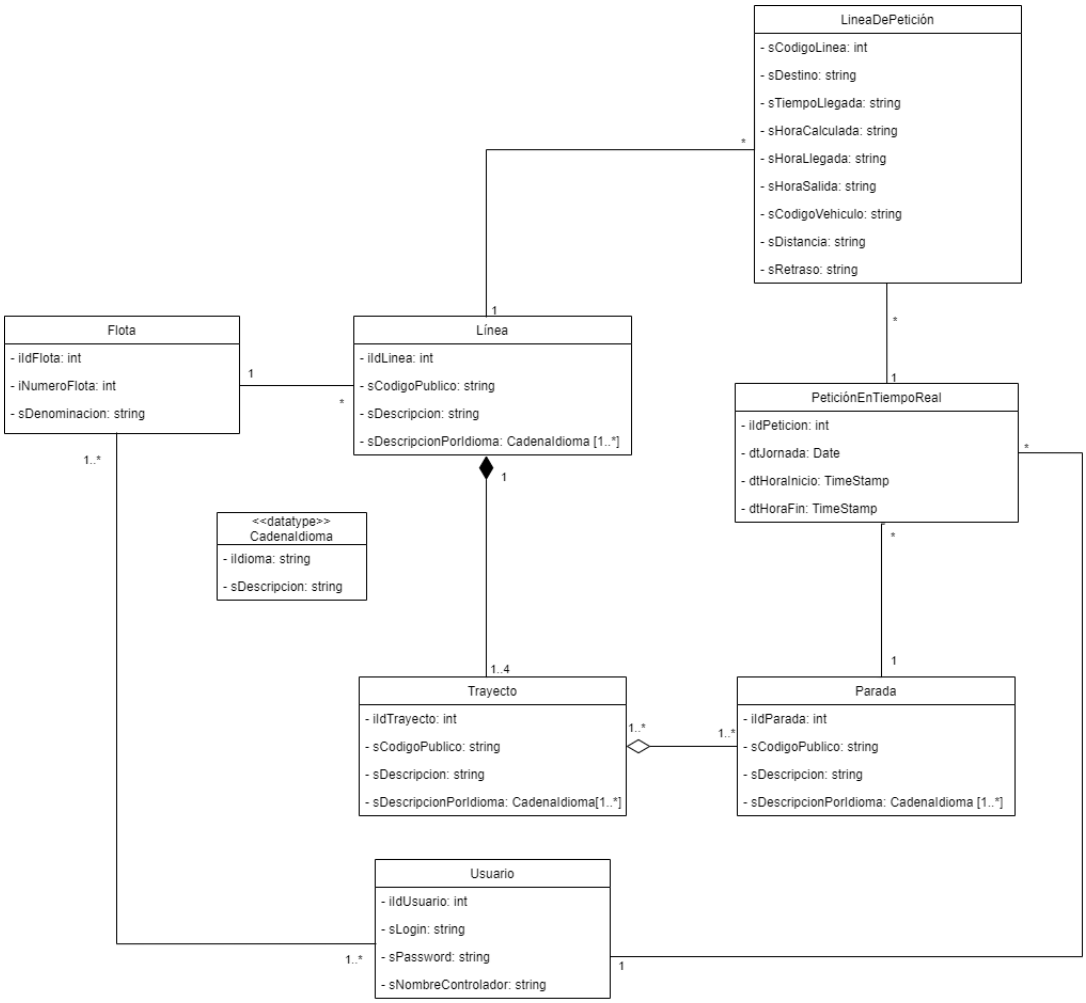


Figura 7.2: Diagrama de Clases del Modelo de Dominio en análisis del segundo prototipo.

7.2.1.2. Diagrama de secuencia

Para este prototipo de uso solo se ha realizado el diagrama de secuencia del caso de uso Monitorizar parada (P2-CU01), que se puede observar en la Figura 7.3. Se ha escogido este sistema porque este caso de uso es el que mejor explica el cometido de este prototipo, sin añadirle la complejidad de otros CU. Además, este diagrama también será incluido en, por ejemplo, el caso de uso Monitorizar varias paradas a la vez (P2-CU02).

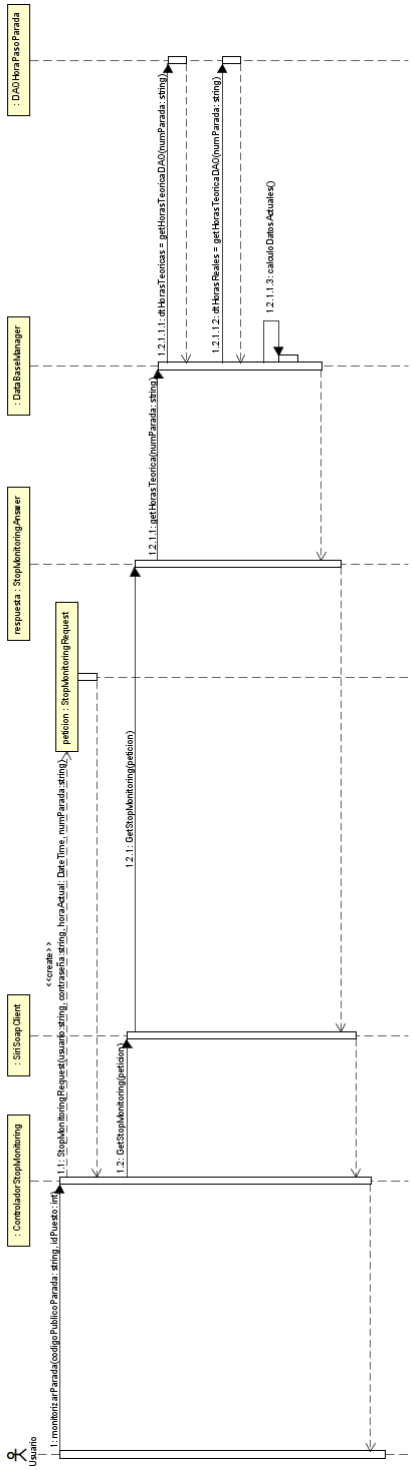


Figura 7.3: Diagrama de secuencia del caso de uso P2-CU01.

7.3. Diseño

7.3.1. Arquitectura del sistema

En este prototipo nos centraremos en la capa de presentación. Contamos con un Web Service con toda la funcionalidad esperada para la versión final pero, en cambio, una aplicación web que solo permite consultar la topología.

Cuando se planteó el proyecto se dividió en tres prototipos verticales para que, en cada uno, se explorase en profundidad al menos una de las capas:

El primero se centra en las capas de la lógica de negocio y en la capa de persistencia, aunque únicamente en la consulta de datos. Para ello, lo que se hizo es crear un nuevo método en el WS creado por GMV para la consulta de topología y una página web muy sencilla para testearlo.

El segundo prototipo se centra en la capa de presentación. Para ello se mejorará la vista creada en el primer prototipo, se creará una nueva vista para el método de la monitorización de paradas y, por último, se establecerá una relación entre ellas. También se profesionalizará la página web con la intención de tener una versión prácticamente final de la misma.

Por último, el tercer prototipo se centrará en las tres capas citadas, aunque en mayor medida, en la capa de lógica y en la capa de persistencia. En este último prototipo se creará una aplicación de consola que monitorizará de forma constante qué paradas no reciben respuesta cada cierto tiempo, cuáles tienen un tiempo de respuesta demasiado alto, etc. Esta herramienta se complementará con un informe web que les permitirá consultar esta información persistente en la BD. Por ello, este tercer prototipo sí insertará y modificará datos en la BD, por lo que complementará al primero, en el que únicamente se consultaba topología. El diagrama de despliegue del segundo prototipo se puede observar en la Figura 7.4

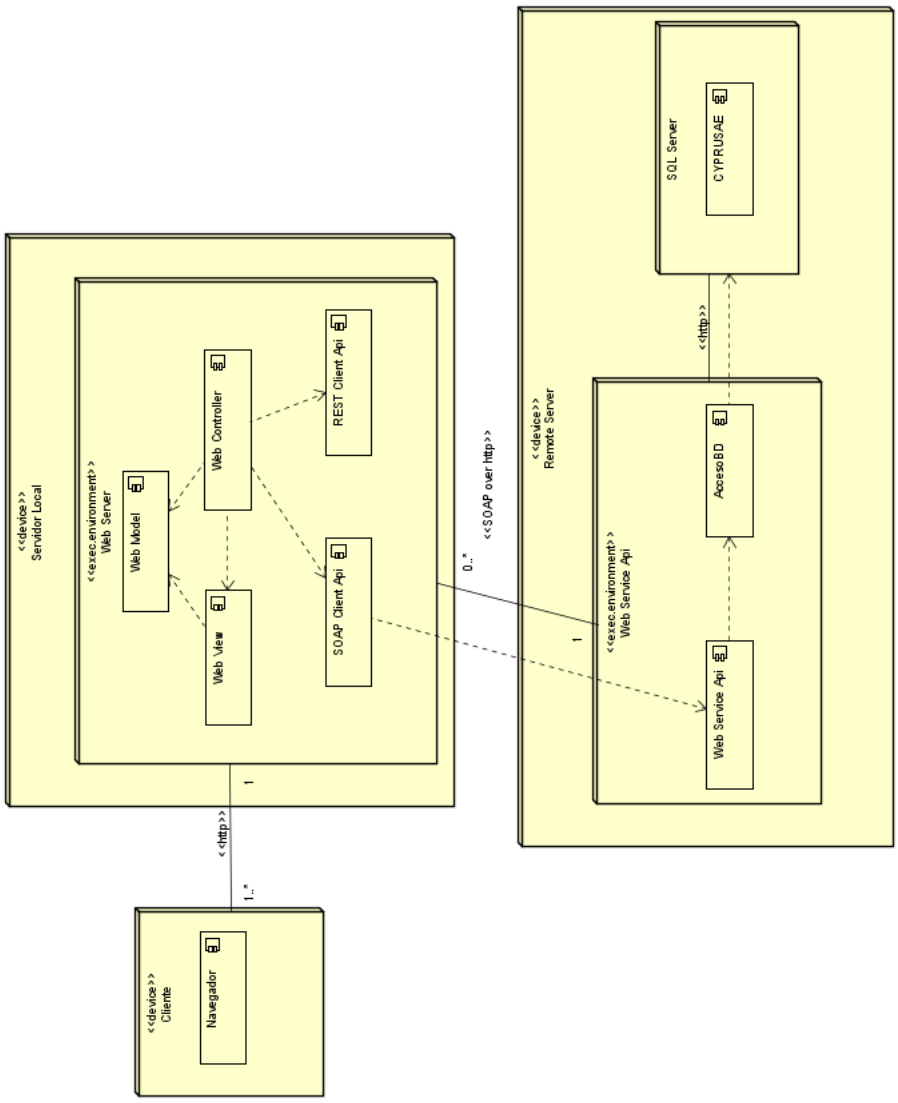


Figura 7.4: Diagrama de despliegue del segundo prototipo.

7.3.2. Capa de presentación

Como ya se ha apuntado, este segundo prototipo se centra en la capa de presentación, por lo que se anotará un listado de focos de atención:

1. Mejorar la vista básica desarrollada para el primer prototipo.
2. Crear la vista de la monitorización manual de paradas.
3. Establecer la relación entre ambas vistas.
4. Profesionalizar la página globalmente.

Después de la reunión con el cliente se plantearon una serie de propuestas para la nueva vista y para completar la de la consulta de topología desarrollada en el primer prototipo.

Antes de continuar, es necesario explicar el concepto de puesto (Workstation). El puesto viene determinado por un número que varía entre 1 y 5 y corresponde a qué consulta de paradas se va a realizar. Si, por ejemplo, un usuario se encuentra en el puesto 1 estará consultando las paradas que se hayan añadido al puesto 1 únicamente, si está en el 2 a las del 2 y así sucesivamente. Esto se creó a partir de la petición expresa del cliente de poder monitorizar al mismo tiempo muchas paradas y, de esta forma, podrá tener hasta 5 pestañas, que corresponden a los 5 puestos, con información diferente.

En este primer mockup vemos la nueva vista que consta de tres partes:

1. La primera parte cuenta solo con barra de navegación que permite cambiar de una vista a otra fácilmente.
2. La segunda parte tiene un texto que pone Workstation Number en el que se puede ver el puesto en el que está el usuario y un botón para cambiar el idioma (Change lenguage).
3. La tercera parte cuenta con un campo de texto en el que se añadiría el código público de la parada que se quiere consultar, un botón para monitorizar la parada y una tabla que mostraría los datos.
4. Y, por último, en la parte inferior, aparecerá el nombre de la parada junto con su código público y la tabla asociada a esa parada. Esta parte es la más importante, ya que se monitorizarán las paradas que se soliciten desde la vista de la consulta de la topología. Se planteó así ya que conocer el código público de la parada es infinitamente más complicado que consultarlo en la topología a partir de su nombre.

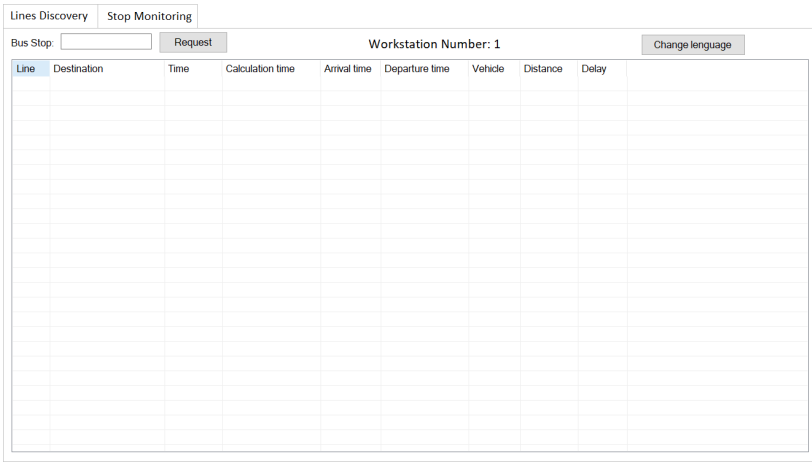


Figura 7.5: Mockup de la vista para la monitorización de paradas.

Como se puede observar, al mockup de la vista consulta de de la topología, que se encuentra en la Figura 7.6 se le han añadido una serie de aspectos importantes:

- Lo primero y más obvio es que todas las paradas tienen ahora un botón al final etiquetado con 'Add', que permitirá añadir esa parada a las que se van a consultar para ese número de puesto.
- Después, el 'Workstation Number' permitirá saber a qué puesto se están añadiendo las paradas que el usuario requiere pulsando en cada botón 'Add'.
- En la línea del anterior, habrá un botón de nombre 'Change Workstation' para poder cambiar de un puesto a otro. Cuando se cambie de puesto, todas las paradas que se selecionen a partir de ese momento se asignarán al nuevo número de puesto. En esta vista, que se trate un puesto u otro no tiene importancia, pero, en cambio, en la vista de monitorización manual determinará las paradas que se van a consultar.
- Y por último, un botón que se llamará 'Reset' que nos permitirá borrar la lista de paradas que se quieren consultar en ese puesto.

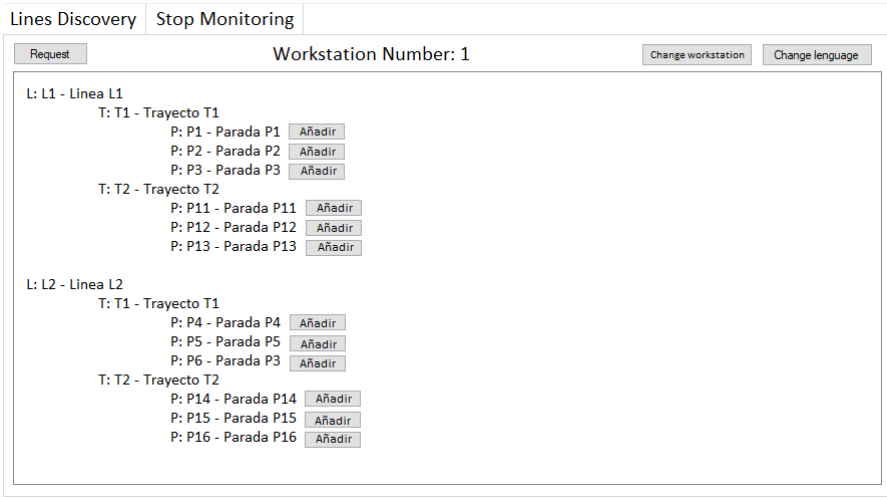


Figura 7.6: Mockup de la vista para la consulta de topología (Segunda Versión).

7.4. Implementación

7.4.1. Desarrollo y desafíos de la implementación

En este prototipo los desafíos tienen especial importancia ya que afectan directamente a la opinión del cliente de la aplicación web.

El primer desafío era adaptar la vista que se desarrolló en el primer prototipo a las nuevas necesidades de este segundo. Esta parte consistía en:

- Añadir un botón '**Change Workstation**' que permitiera cambiar de puesto.
- Añadir un botón '**Reset**' que eliminará la lista de paradas a consultar de cada puesto.
- Añadir un botón '**Add**' a cada parada para poder insertarla a la lista de las consultadas por puesto.

La funcionalidad de todos estos botones ya ha sido explicada en el punto anterior.

El segundo desafío era conseguir mejorar la vista que se desarrolló en el primer prototipo con las nuevas funcionalidades añadidas. Para ello, se planteó un mejor posicionamiento para todos los elementos de la página, y se trató de diseñarla de forma que permitiese que el resto de vistas pudieran seguir el patrón. Una vez planteada una vista genérica, únicamente habría que cambiarla para adaptarla a cada vista nueva que se quisiera desarrollar.

A partir de esta vista genérica, se creó una nueva para la monitorización manual de paradas. En ella se consultaría la información de las paradas elegidas por puesto. Lo realmente importante de esta vista es que tendrá asignado un número de puesto, lo que determinará qué lista de paradas se consultará. Además tendría que estar relacionada con la ya realizada en el anterior prototipo, esto se hizo a partir de una navbar, común a las dos vistas, y una redirección a cada vista [5].

El tercer desafío y el más importante que ha tenido hasta ahora este proyecto, era conseguir que todas las paradas monitorizadas por puesto se recargasen a tiempo real. Para conseguirlo se necesitaban varias cosas:

1. En primer lugar, encontrar una forma de recargar la página: necesariamente era un tema de la parte cliente, por lo que tenía que hacerse mediante Java Script.

2. En segundo lugar, había que tener en cuenta toda la información que se necesita para recargar la vista. Este punto llevó bastante tiempo, ya que se necesitaba una forma de que persistiera la información una vez la página había sido recargada. Para ello, se utilizó el concepto de session. En este punto, es preciso explicar qué es la session y los tipos que hay. El protocolo HTTP no tiene estado, es decir, un ordenador cliente que ejecuta un navegador web tiene que establecer una conexión TCP nueva con cada petición GET o POST. El servidor web, por lo tanto, no puede confiar en una conexión TCP establecida para más de una única operación GET o POST. La administración de la session es la técnica que utiliza el desarrollador web para dar soporte de estado al protocolo HTTP, es decir, la herramienta que tiene para poder almacenar información en el tiempo [1]. A lo largo de esta y todas las vistas de la aplicación web se utilizarán dos tipos de sessions:

   - `HttpContext.Session`: es como se denomina la session de ASP.NET Core. Se almacena en el lado del servidor, lo que limita mucho su uso. Es especialmente útil para intercambiar información entre vistas.
   - `Window.sessionStorage`: es como se denomina a la session de JavaScript. Es extremadamente potente y útil, ya que se aloja en la pestaña del navegador del usuario. La session de la página perdura mientras el navegador se encuentra abierto, y se mantiene aunque se recargue la página. Abrir una página en una nueva pestaña o ventana iniciará una nueva session y cerrarla implicará el fin de la misma.

3. Una vez almacenada la información en la session, al recargar la página con JavaScript se lanzaría un metodo GET del Controlador que volvería a realizar todas las peticiones a la API SOAP para luego mostrar la información en la vista.

[1] Session o Sesión (Wikipedia): `https://es.wikipedia.org/wiki/Sesin_(informtica)#Administracin_de_sesin_del_servidor_web`

7.5. Validación

Una vez finalizado el segundo prototipo, se programó una reunión con el cliente para mostrárselo, que tuvo que realizar una revisión prácticamente completa del proyecto, puesto que el segundo prototipo necesita del primero. Para ello, se pasó toda la batería de pruebas de este segundo prototipo, como se puede observar en la Tabla 7.6.

Hasta este punto se tenía toda la funcionalidad de, tanto la vista Lines Discovery que se puede ver en la Figura 7.7, como la de Stop Monitoring (Figura 7.8).

Figura 7.7: Vista Lines Discovery una vez terminado el segundo prototipo.

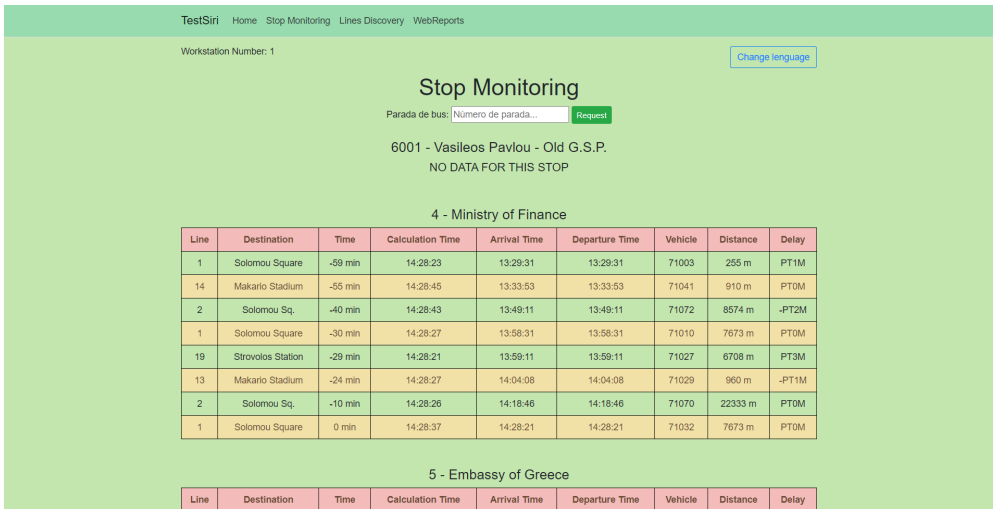


Figura 7.8: Vista Stop Monitoring una vez terminado el segundo prototipo.

El cliente se mostró muy satisfecho y nos felicitó por el rumbo que estaba tomando el proyecto. Reconoció que teníamos razón cuando le dijimos que el problema era la estructuración y posicionamiento de los elementos de la vista, no los colores de fondo y cabecera como expuso él en la revisión final del primer prototipo.

En la revisión del prototipo con el tutor también se mostró satisfecho con el trabajo realizado, si bien, realizó una serie de propuestas de mejora que han sido añadidas ya al proyecto.

Id	Descripción	Propuesta por	Resultado	Éxito
P1-P06 (CT)	Comprobar que se cumple la estructura de árbol en las líneas, trayectos y paradas	GMV	Lista todas las paradas asociadas a un trayecto	SÍ
P2-P01 (MMP)	Consultar una parada escribiendo su código publico	Cliente	Lista todas las líneas de petición de la parada solicitada	SÍ
P2-P02 (CT/MMP)	Consultar una parada pulsando en la parada que se quiera en la vista de la consulta de topología	Cliente	Lista todas las líneas de petición de la parada solicitada	SÍ
P2-P03 (CT/MMP)	Consultar una parada añadiéndola a la lista de paradas solicitadas por puesto	Cliente	Lista todas las líneas de petición de la parada solicitada	SÍ
P2-P04 (MMP)	Cambiar idioma, permitirá cambiar del idioma local al inglés y viceversa	Cliente	Cambiará de idioma todas las cadenas referentes a la petición	SÍ
P2-P05 (CT/MMP)	Consultar más de una parada añadiéndolas a la lista de paradas solicitadas por puesto	Cliente	Lista todas las líneas de petición de la paradas solicitadas, distinguiendo entre cada una de ellas	SÍ
P2-P06 (CT/MMP)	Consultar hasta diez paradas añadiéndolas a la lista de paradas solicitadas por puesto	Cliente	Lista todas las líneas de petición de la paradas solicitadas, distinguiendo entre cada una de ellas	SÍ
P2-P07 (CT/MMP)	Consultar que no se pueden consultar más de diez paradas añadiéndolas a la lista de paradas solicitadas por puesto	GMV	Lista todas las líneas de petición de la paradas solicitada, distinguiendo entre cada una de ellas	SÍ
P2-P08 (CT/MMP)	Eliminar la lista de paradas solicitadas en un puesto	Cliente	Comprobar que ya no se listan las líneas de petición de las paradas previamente solicitadas en ese puesto	SÍ
P2-P09 (CT/MMP)	Cambiar de puesto	GMV	Comprobar que el número de puesto ha cambiado	SÍ
P2-P10 (CT/MMP)	Una vez se ha cambiado de puesto, realizar las pruebas P01, P02, P03, P04, P05, P06 y P07	GMV	Lista todas las líneas de petición de la paradas solicitadas, distinguiendo entre cada una de ellas	SÍ
P2-P11 (CT/MMP)	Eliminar la lista de paradas solicitadas en más de un puesto	GMV	Comprobar que ya no se listan las líneas de petición de las paradas previamente solicitadas en esos puestos	NO

Tabla 7.6: Batería de pruebas del segundo prototipo de la aplicación (P2).

Capítulo 8

Tercer prototipo

8.1. Especificación de requisitos

8.1.1. Interfaces de usuario

En el tercer y último prototipo se requiere de una nueva vista para la página web que permita consultar la información asociada a la monitorización automática de paradas. Antes de detallar en qué consiste la vista, y de qué está compuesta, es preciso explicar que el tercer prototipo constará de tres partes: una nueva vista en la aplicación web de los anteriores prototipos, una aplicación de Windows y un WebService que implementará una api Rest.

La aplicación de Windows servirá para monitorizar cada cierto tiempo todas las paradas que existan en la topología activa 6.1.1. Ante la necesidad de modelar un elemento que contenga todas las peticiones consultadas en unas horas determinadas, nace el contenedor de peticiones.

Por tanto, como **contenedor de peticiones** nos referimos al conjunto de peticiones que se realizan en el mismo bloque, es decir, todas aquellas peticiones que se realicen consecutivamente de forma automática en un intervalo de tiempo.

Para el caso de las **peticiones** se mantiene la definición del segundo prototipo, añadiéndole las peticiones automáticas, las que no han sido realizadas de forma manual.

En esta vista se distinguirán dos partes a las que se denominará subvistas: una para consultar el histórico de los contenedores de peticiones y otra para el histórico de las peticiones. Se han planteado dentro de la misma vista puesto que existe una relación directa entre contenedor y petición: toda petición realizada de forma automática pertenecerá a un contenedor.

8.1.2. Requisitos

8.1.2.1. Requisitos funcionales

Los requisitos funcionales que afectan a este tercer prototipo son:

P3-RF1: *El sistema deberá permitir a los usuarios consultar el histórico de todos los contenedores de petición.*

P3-RF2: *El sistema deberá permitir a los usuarios consultar el histórico de los contenedores de petición asociados a una jornada.*

P3-RF3: *El sistema deberá permitir a los usuarios consultar el histórico de todas las peticiones.*

P3-RF4: *El sistema deberá permitir a los usuarios consultar el histórico de todas las peticiones asociadas a una jornada.*

8.1.2.2. Requisitos no funcionales

Los requisitos no funcionales que afectan a este tercer prototipo son:

P3-RFN1: *El sistema debe ser capaz de consultar los datos del sistema en menos de 30 segundos.*

P3-RFN2: *El sistema debe ser capaz de admitir hasta 5 pestañas web con información diferente.*

P2-RFN3: *El sistema deberá permitir intercalar entre la consulta de topología, la monitorización de paradas y la consulta de históricos.*

8.1.3. Batería de pruebas

En la batería de pruebas de este prototipo, además de añadir todas las pruebas asociadas a este, se añadirán aquellas nacidas del conocimiento de la interfaz y funcionalidades asociadas al segundo prototipo. Además, en este caso, en la revisión final del segundo prototipo una de las pruebas no sé pasó con éxito, por lo que también será añadida a este.

En este tercer y último prototipo todas las pruebas nuevas afectarán únicamente a la vista nueva de informes web (IW), exceptuando aquellas que provengan del anterior.

Id	Descripción	Propuesta por	Resultado
P2-P11 (CT/MMP)	Eliminar la lista de paradas solicitadas en más de un puesto	GMV	Comprobar que ya no se listan las líneas de petición de las paradas previamente solicitadas en esos puestos
P2-P12 (CT/MMP)	Ver que cada parada se deshabilita al ser añadida a la lista de paradas solicitadas por puesto	Cliente	Comprobar que se deshabilita la parada al añadirla
P2-P13 (CT/MMP)	Ver que al resetear un puesto se habilitan todas las paradas previamente deshabilitadas	GMV	Comprobar que se habilitan las paradas
P2-P14 (CT/MMP)	Ver que al resetear un puesto se habilitan todas las paradas previamente deshabilitadas	GMV	Comprobar que se habilitan las paradas
P3-P01 (IW)	Cambiar entre las dos subvistas (la de peticiones y la de contenedores)	Cliente	Comprobar que en la subvista de peticiones hay un campo desplegable con el filtro de tipo de petición y en la de contenedores no
P3-P02 (IW)	Solicitar las peticiones sin filtro	Cliente	Comprobar que se han listado todas las peticiones sin tener en cuenta la fecha ni el tipo
P3-P03 (IW)	Solicitar las peticiones para una jornada específica	Cliente	Comprobar que se han listado todas las peticiones para esa fecha específica
P3-P04 (IW)	Solicitar las peticiones para una fecha y un tipo de petición específica	Cliente	Comprobar que se han listado todas las peticiones para esa fecha y ese tipo de petición
P3-P05 (IW)	Solicitar los contenedores sin filtro	Cliente	Comprobar que se han listado todos los contenedores sin tener en cuenta la fecha ni el tipo
P3-P06 (IW)	Solicitar los contenedores para una jornada específica	Cliente	Comprobar que se han listado todas los contenedores para esa fecha específica

Tabla 8.1: Batería de pruebas del tercer prototipo de la aplicación (P3).

8.2. Análisis

En este tercer prototipo se partirá del diagrama de casos de uso del segundo prototipo, completándolo. Como en el anterior prototipo, solo se especificarán los casos de uso nuevos y aquellos casos de uso que tengan alguna modificación.

8.2.1. Diagrama de casos de uso

El diagrama de casos de uso para este tercer prototipo se muestra en la Figura 8.1:

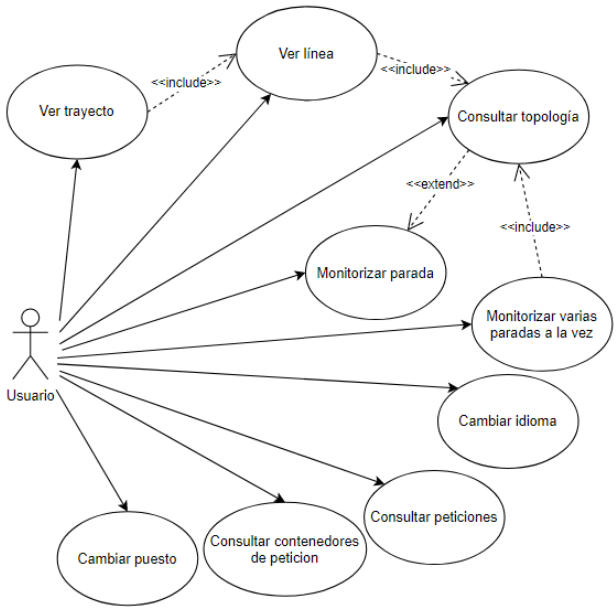


Figura 8.1: Modelo de casos de uso obtenido en el análisis del tercer prototipo de la aplicación.

Especificación de cada uno de los casos de uso:

P3-CU01	Caso de uso: Consultar peticiones
Breve descripción	Permite consultar el histórico de peticiones.
Actor	Usuario
Precondiciones	Haber pulsado el botón de informe de peticiones.
Flujo de eventos	Flujo básico: 1. El usuario pulsa el botón de consultar histórico de peticiones. Flujo alternativo: 1. El sistema muestra un campo de fecha para poder obtener los datos de un día de particular. 2. El usuario ingresa la fecha de la que quiere datos. 3. El usuario pulsa el botón para consultar el histórico de peticiones.
Postcondiciones	El sistema mostrará la información de todas las peticiones consultadas automáticamente.

Tabla 8.2: Caso de uso: Consultar peticiones (P3-CU01).

P3-CU02	Caso de uso: Consultar contenedor de peticiones.
Breve descripción	Permite consultar el histórico de contenedores.
Actor	Usuario
Precondiciones	Haber pulsado el botón de informe de contenedores.
Flujo de eventos	Flujo básico: 1. El usuario pulsa el botón de consultar histórico de contenedores. Flujo alternativo: 1. El sistema muestra un campo de fecha para poder obtener los datos de un día de particular. 2. El usuario ingresa la fecha de la que quiere datos. 3. El usuario pulsa el botón para consultar el histórico de contenedores.
Postcondiciones	El sistema mostrará la información de todos los contenedores o bloques de peticiones realizados automáticamente.

Tabla 8.3: Caso de uso: Consultar contenedor de peticiones (P3-CU02)

El modelo de dominio para este tercer prototipo es:

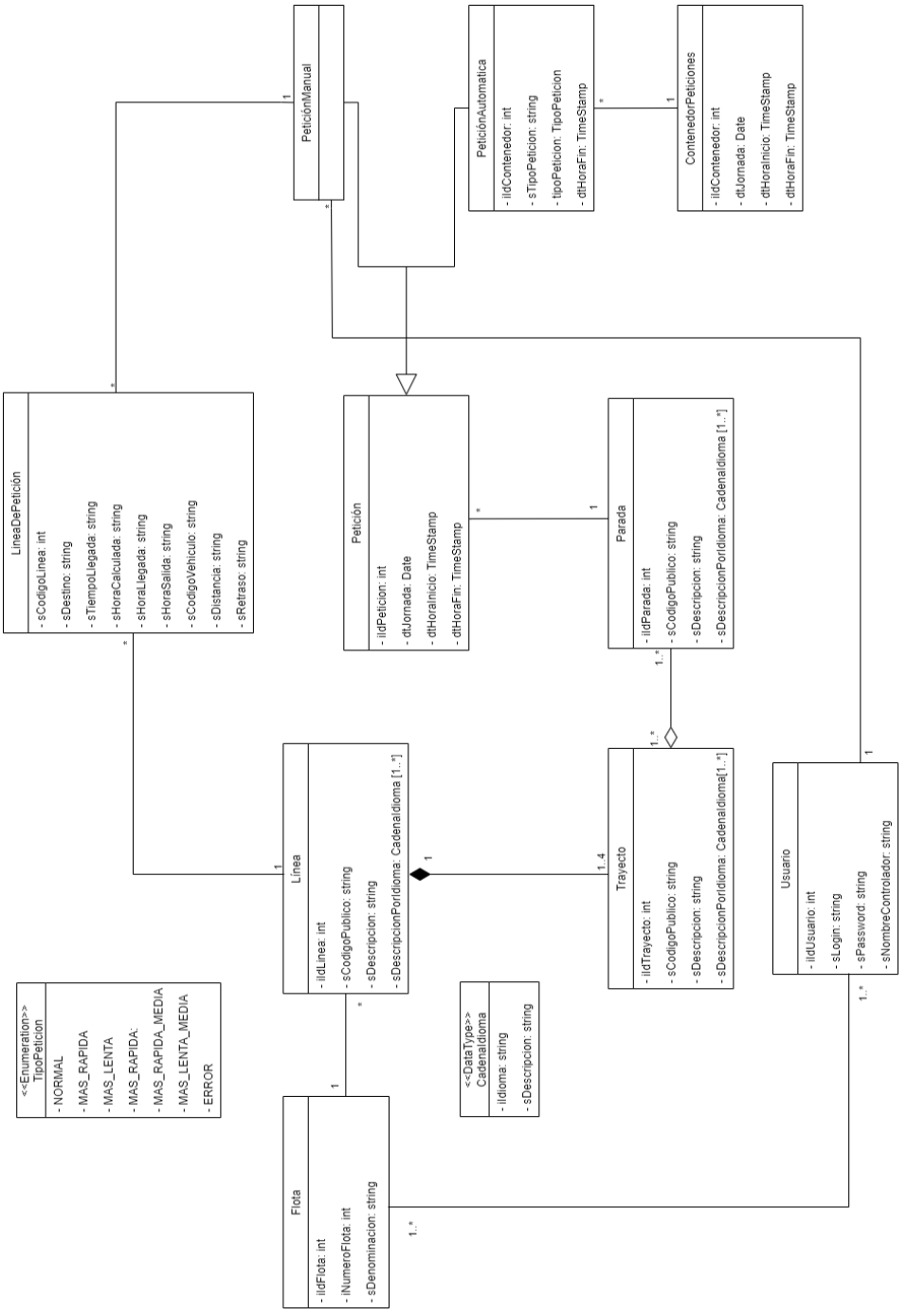


Figura 8.2: Diagrama de Clases del Modelo de Dominio en análisis del tercer prototipo.

8.2.1.1. Diagrama de secuencia

En este prototipo los CU complejos son extremadamente parecidos. Únicamente se diferencian si se consultan peticiones o contenedores y si la petición es con o sin filtro de fecha. Por ello, se ha realizado el diagrama de secuencia de, por ejemplo, el caso de uso Consultar peticiones (P3-CU01), que se puede observar en la Figura 8.3.

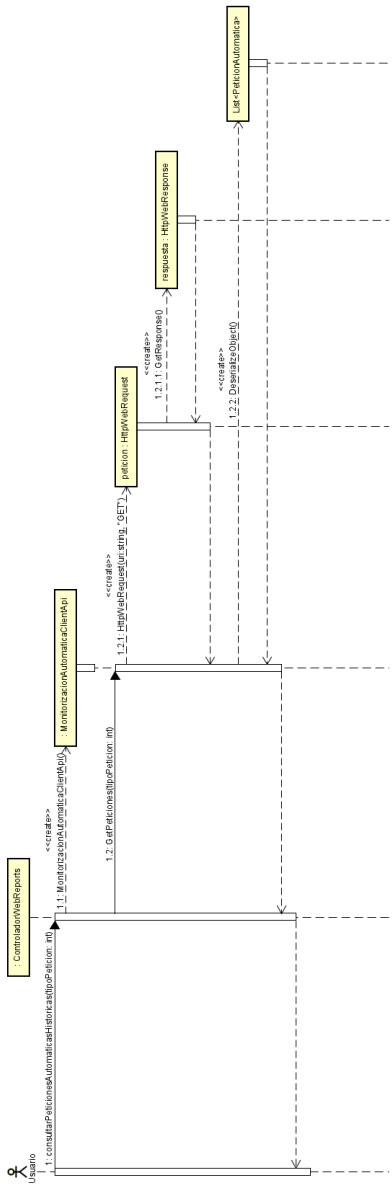


Figura 8.3: Diagrama de secuencia del caso de uso P3-CU01.

8.3. Diseño

8.3.1. Arquitectura del sistema

En este prototipo este apartado tiene especial importancia porque se añaden al proyecto tres nuevos elementos:

1. Una aplicación Windows que monitorizará de forma automática todas las paradas vigentes de la topología y que insertará los datos en la BD.
2. Una aplicación de servicios WCF (Windows Communication Foundation) que permitirá crear una Api Rest para acceder a los datos insertados. Se ha decidido hacer con este tipo de aplicacion ya que son las más especializadas para crear servicios que posteriormente se desplegarán en el IIS (Internet Information Services).
3. Una nueva vista en la aplicación web para poder mostrar al usuario los datos consultados a partir de la Api Rest.

De acuerdo con lo explicado, es obvio que este prototipo contendrá partes de las tres capas. Aunque, como ya se comentó en el prototipo anterior, nos centraremos en las capas de lógica de negocio y persistencia. El diagrama de despliegue del tercer prototipo se encuentra en la Figura 8.4

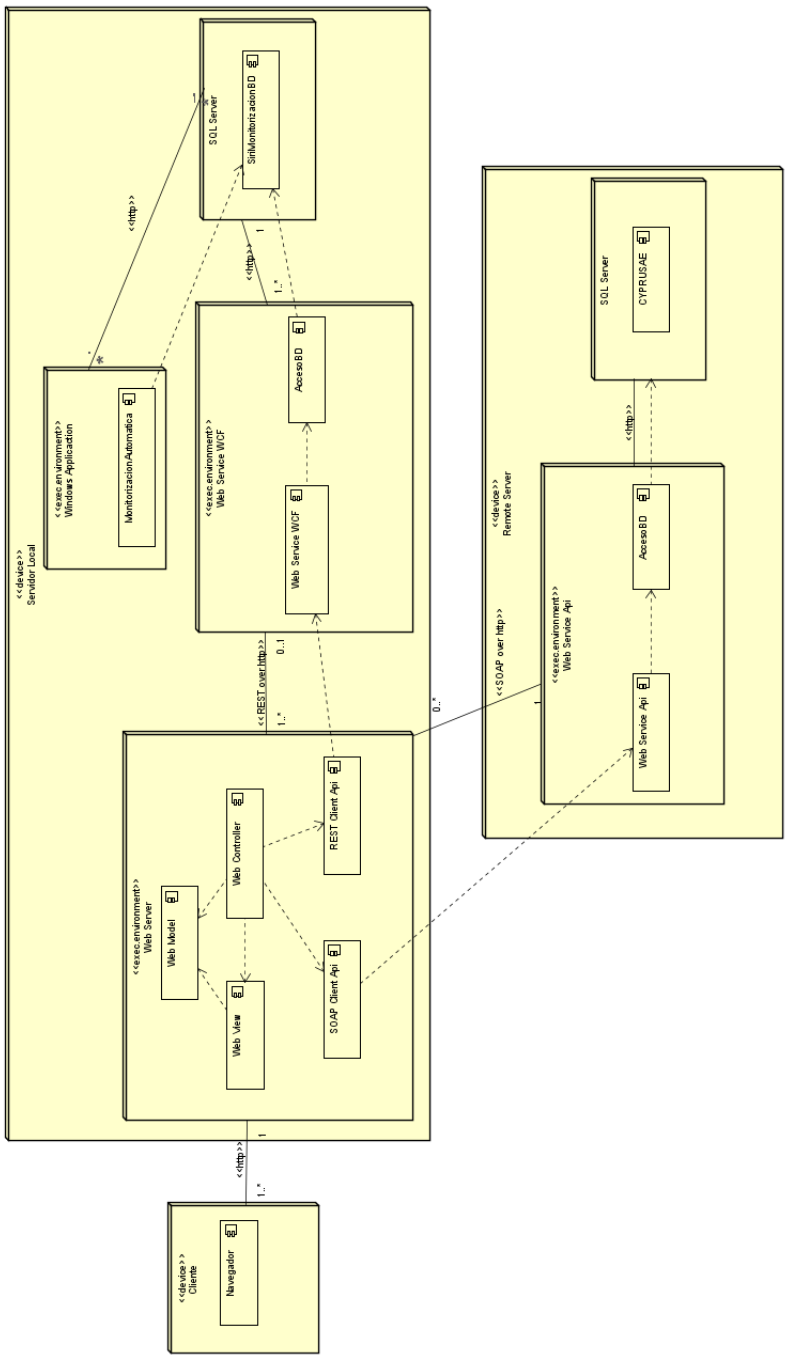


Figura 8.4: Diagrama de despliegue del tercer prototipo.

8.3.2. Capa de presentación

En esta capa únicamente se harán las modificaciones en las vistas del anterior prototipo solicitadas por el tutor y el cliente y se añadirá la nueva vista.

Constará, como ya se ha comentado, de dos subvistas, ambas dedicadas a la consulta de históricos, pero en un caso al de peticiones y en el otro al de contenedores de petición. Se decidió así porque el cliente quería que ambas estuvieran en la misma pestaña, además de por su correlación, que se denominaría 'Web Reports'.

Para terminar, con la capa de presentación se adjuntarán los dos mockups realizados en las primeras reuniones con el cliente, uno para cada parte de la pestaña 'Web Reports'. La idea de estos mockups es que, en la misma vista, dependiendo del botón pulsado, se muestre una información u otra. Por ejemplo, en el primero de ellos se puede observar en color morado el botón "Request report", lo que significaría que, si se solicitasen los datos, se consultaría el histórico de peticiones. En caso contrario, si el botón morado fuese el "Container report", significaría que se van a consultar los históricos de contenedores de peticiones (segundo mockup). También en la tabla asociada a cada uno de ellos se podrá observar las columnas que tendrá.

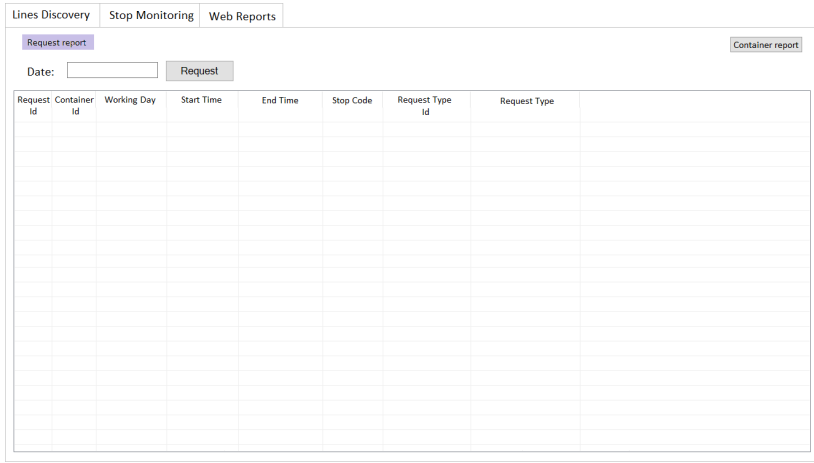


Figura 8.5: Mockup de la subvista para consultar peticiones.

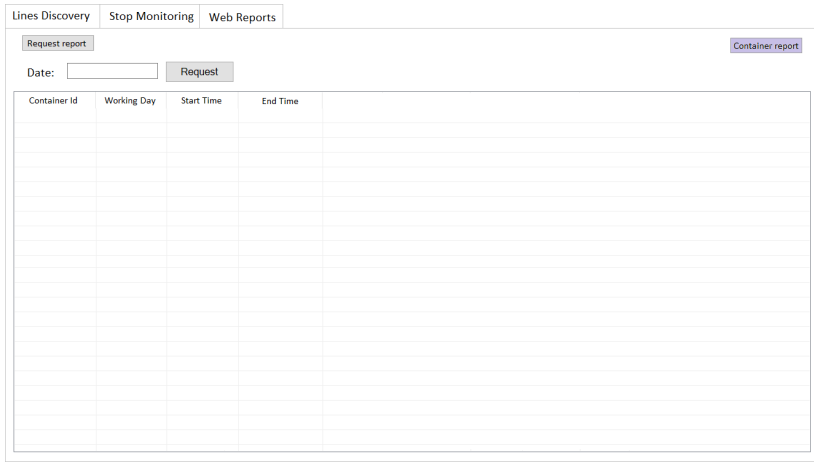


Figura 8.6: Mockup de la subvista para consultar contenedores de petición.

8.3.3. Capa de lógica de negocio y capa de persistencia

En esta capa nos centraremos específicamente en los componentes que tengan lógica de negocio y persistencia de datos, como son la aplicación de Windows y la aplicación de servicios WCF con la API Rest.

Para la persistencia de datos, a lo largo de este prototipo, se utilizará una BD en local. Todas estas tablas, una vez se pase la validación, serán insertadas a la BD remota en el servidor de producción a la que ya accede el Web Service al que se consultan los datos.

8.3.4. Funcionalidades desarrolladas

La aplicación de Windows se encargará de:

1. Solicitar la topología vigente a partir del método SOAP `LinesDiscovery` realizado en el primer prototipo.
2. Una vez obtenida la topología, lo que se precisa únicamente son las paradas, por lo que a partir de todas las líneas se obtendrán todos los trayectos y a partir de los trayectos todas las paradas (7.2).
3. Después se creará un objeto `Contenedor` (8.2), que estará formado por todas las paradas que se consulten posteriormente hasta que termine la ejecución. Teniendo en cuenta que solo parará en caso de que se cierre manualmente, casi siempre estará formado por todas las paradas vigentes en la topología.
4. Una vez creado el contenedor se almacenará en la base de datos (INSERT). Únicamente contará con identificador, jornada y hora inicio.

5. Una vez conocidas todas las paradas, será preciso monitorizar cada una de ellas a partir del método SOAP StopMonitoring (utilizado en el segundo prototipo para monitorizar manualmente paradas).

6. Al tener la hora inicio antes de monitorizar la parada y la hora fin, se podrá calcular el tiempo de respuesta. Dependiendo de este el tipo de petición será uno o otro. Si se considera, por ejemplo, 0.3 segundos un tiempo medio de respuesta, todas las paradas que tarden menos tendrán el estado 4 (MAS_RAPIDA_MEDIA) y viceversa. Será el mismo caso para el resto de los estados. Es importante aclarar que al estar esta aplicación de Windows en la misma máquina que el WS que proporciona los datos, no existirán retardos por culpa de la red.

7. Después, se almacenarán todas las peticiones en la BD (INSERT), contando con toda la información necesaria para su almacenamiento: identificador, contenedor al que pertenecen, jornada, hora inicio, hora fin y tipo de petición.

8. Finalmente, una vez que haya terminado con todas las peticiones, se actualizará la hora fin del contenedor (UPDATE).

La aplicación de servicios WCF se encargará de:

1. Contar con una Api Rest para consultar contenedores y peticiones.

2. Disponer de dos métodos GET para poder acceder a los históricos, tanto de peticiones como contenedores. El método de peticiones tendrá como parámetros una fecha y un filtro de petición, ambos opcionales. El filtro de petición8.2, es la forma en que el cliente filtraría únicamente desde la interfaz las peticiones que hayan dado error, que sean más lentas que la media... El método de contenedores de petición tendrá como parámetro únicamente la fecha.

3. Estos métodos consultarán en la BD todos las peticiones o contenedores asociados a un día en caso de que se haya añadido, o todos los existentes en otro caso (SELECT). En el caso de las peticiones también podrá consultar únicamente los que estén en un estado.

4. Después, se crearán los objetos y se añadirán a una lista para ser devuelta a la interfaz o aplicación web, que será quién realice las peticiones.

Como ya se habrá podido observar, la aplicación de servicios WCF será el puente entre la interfaz, es decir, la aplicación web y la aplicación de Windows que monitoriza de forma automática los datos.

8.3.5. Tiempo medio de respuesta

El tiempo medio de respuesta es, por tanto, el factor más importante en este tercer y último prototipo. Calcularlo ha sido un proceso complejo, puesto que, determinar un tiempo medio de repuesta base acorde a lo real es crucial para que los datos consultados mediante la aplicación web tengan sentido.

Uno de los primeros pasos fue calcular el tiempo de respuesta de dos paradas aleatorias de entre todas aquellas que tienen datos reales. Para que estos datos fueran coherentes se realizó la misma petición 15 veces por parada y se almacenaron todos los datos. Se planteó así porque, al estar el Web Service al que se consultan los datos reales de la parada en la máquina remota de producción, se podrían ver afectados por los problemas de red que pueda haber en algún momento determinado, además del obvio retraso que tendrá de base al no estar en la misma máquina. Ello es inevitable si se quieren hacer pruebas con paradas reales. Se realizaron pruebas con muchas paradas, por ejemplo estas que fueron a la 4 y a la 73 (Figura 8.7 y Figura 8.8):

sCodigoParada = 4	Petición	Tiempo Respuesta	Tiempo Medio Respuesta	Tiempo Medio Base	Media Final
	1	254,00	283,47	250,00	266,73
	2	482,00			
	3	275,00			
	4	262,00			
	5	259,00			
	6	256,00			
	7	261,00			
	8	268,00			
	9	256,00			
	10	263,00			
	11	373,00			
	12	251,00			
	13	252,00			
	14	257,00			
	15	283,00			

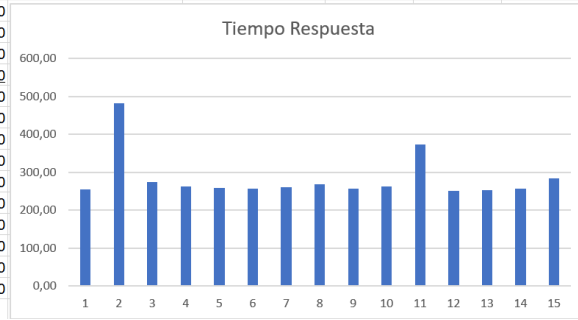


Figura 8.7: Tiempo medio de respuesta de la parada 4.

sCodigoParada = 73	Petición	Tiempo Respuesta	Tiempo Medio Respuesta	Tiempo Medio Base	Media final
	1	290,00	290,60	250,00	270,30
	2	246,00			
	3	268,00			
	4	344,00			
	5	261,00			
	6	343,00			
	7	250,00			
	8	341,00			
	9	313,00			
	10	280,00			
	11	271,00			
	12	361,00			
	13	261,00			
	14	274,00			
	15	256,00			

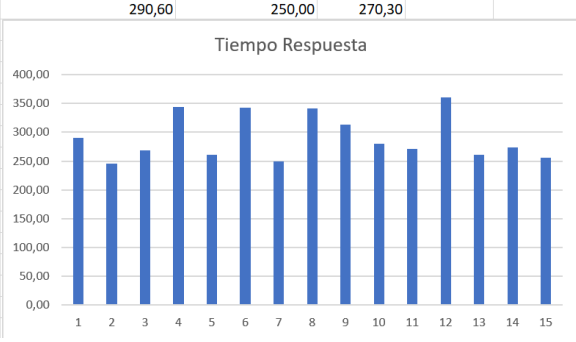


Figura 8.8: Tiempo medio de respuesta de la parada 73.

Una vez realizadas las pruebas con paradas individuales se llevaron a cabo varias pruebas con todas las paradas, que ya no consistían en la repetición de una en hasta quince ocasiones, sino en hacer una petición a todas las paradas y hacer una media de todas ellas. Actualmente en el subsistema están dadas de alta 16907 paradas por lo que serán 16907 tiempos de respuesta a comparar. Por ello, solo se adjuntará el gráfico resultante y los cálculos realizados (Figura 8.9):

Tiempo Medio Respuesta	Tiempo Medio Base	Media Final
226,19	250,00	238,09

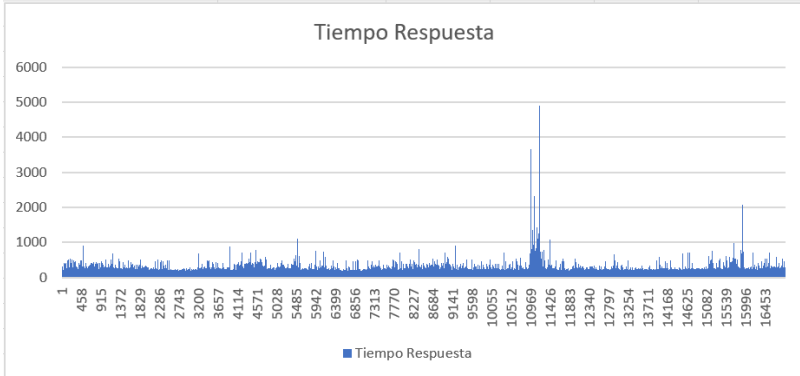


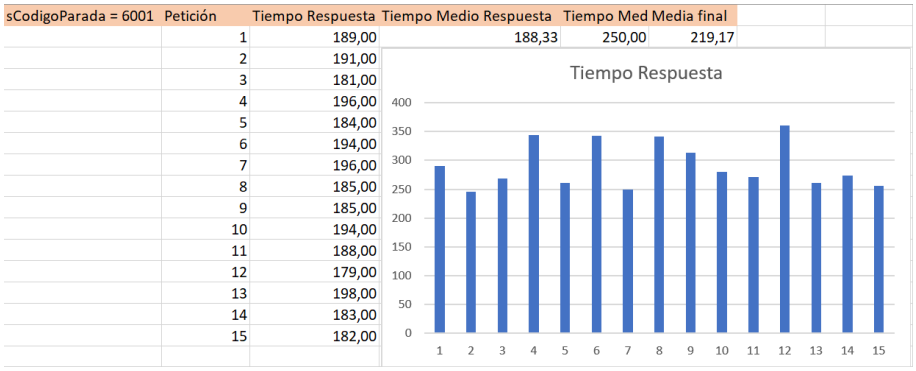
Figura 8.9: Tiempo medio de respuesta de todas las paradas.

Aunque sorprenda que exista tanta diferencia entre el promedio de las paradas individuales (283.47 ms para la parada 4 y 290.60 ms para la parada 73) y el de todas (226.19 ms), y más aún que sea menor en el caso de todas, realmente tiene lógica: las dos paradas individuales adjuntadas tienen datos reales de forma constante y, en cambio, muchas otras de las que se incluyen en todas no. Es así porque la versión piloto de Stop Monitoring desplegada en producción no devuelve datos de todas las paradas, sino únicamente de unas pocas.

Asimismo, también hay que tener en cuenta que, por ejemplo, en horarios nocturnos ninguna parada tiene datos. En esos horarios no hay buses operando, por lo que entre las 12:00 de la noche y a las 6:00 de la mañana los datos tendrán un promedio mucho menor. Lo mismo puede ocurrir en algunas líneas a determinadas horas, aunque también puede darse el caso contrario, con trayectos de refuerzo por eventos locales.

Para que el proyecto no pueda verse afectado por la problemática del horario nocturno, los cálculos siempre se realizarán en horario de mañana y de tarde.

Por último, se adjuntará el tiempo de respuesta medio de una parada que no devuelva datos, como puede ser la 6001 (Figura 8.10):



sCodigoParada = 6001	Petición	Tiempo Respuesta	Tiempo Medio Respuesta	Tiempo Med	Media final
	1	189,00	188,33	250,00	219,17
	2	191,00			
	3	181,00			
	4	196,00			
	5	184,00			
	6	194,00			
	7	196,00			
	8	185,00			
	9	185,00			
	10	194,00			
	11	188,00			
	12	179,00			
	13	198,00			
	14	183,00			
	15	182,00			

Figura 8.10: Tiempo medio de respuesta de la parada 6001.

Esto, por tanto, explica y justifica que la media de todas las paradas sea más baja, puesto que cuenta con algunas con datos y otras sin datos.

En las cuatro capturas adjuntadas se puede observar una columna con el Tiempo medio de respuesta base, que se ha calculado en función de todos los cálculos realizados a cada parada y al conjunto de todas. Se ha intentado que sea un número redondo para que sea más adaptable a otros proyectos. Realmente este valor solo tiene relevancia en el origen, ya que con cada ejecución se reajustará acercándose cada vez más al valor ideal.

El proceso de reajuste consistirá en que cada ejecución almacenará un promedio de tiempo de respuesta de todas las paradas y, a partir de este y el valor base, se realizará una media ponderada. Este cálculo sencillo permitirá que, con el paso del tiempo, cada vez sea posible acercarse más al valor estándar, alejándose así de los valores atípicos.

Otro aspecto a tener en cuenta es que, cuando todas las paradas devuelvan datos reales, habrá que recalcular este índice base. Realmente el paso del tiempo ya reajustará el valor al real.

8.4. Implementación

8.4.1. Desarrollo y desafíos de la implementación

Este prototipo, como era de esperar por ser el último, tuvo grandes desafíos. Todos ellos se explicarán a medida que se fueron presentando.

La implementación de este prototipo se realizó por fases, ya que, para consultar por ejemplo las peticiones de un día determinado, se necesita que existan esos datos. Por ello, se hablará de tres fases:

1. Aplicación de Windows.
2. Aplicación de servicios WCF.
3. Nueva vista en la aplicación web, con sus dos partes.

Tanto la aplicación de Windows como la aplicación de servicios WCF se han realizado en .NET Framework 4.7.2. Se tomó esta decisión porque todos los proyectos GMV que quieran utilizar trazas, tener acceso a BDs, convertir los nombres de las tablas al formato específico y muchas otras cosas, deben utilizar la Arquitectura de GMV, y esta no existe para .NET Core. En el caso de la aplicación web no era necesario utilizar la Arquitectura ya que, al ser únicamente la capa de presentación, ni accede a datos de la BD, ni requiere de trazas de log, porque se puede utilizar la propia página con información visual.

Desde el principio del desarrollo del prototipo se planteó un desafío: el hecho de que tengan que coexistir en el mismo subsistema dos versiones de .NET dio lugar a un problema en el proyecto del Modelo. Este aspecto no había tenido relevancia hasta ahora porque, hasta este momento, en la solución únicamente se utilizaba la aplicación web en .NET Core 3.1, lo que evitaba el problema que existe al coexistir varias versiones. Para solucionar este problema se creó un nuevo proyecto Modelo en .NET Standard 2.0, que es compatible con las dos versiones que utilizaremos como se puede observar en la Figura **??**; de hecho es la versión más actual que tiene compatibilidad con versiones de .NET Framework[1].

Cabría preguntarse por qué en el primer prototipo en el que se desarrolló la consulta de toplogía en el Web Service que implementa los métodos SIRI, no hubo este problema de versión cuando se especificó que fue en .NET Framework 4, anterior incluso al .NET Framework 4.7.2 que utilizaremos en el prototipo actual. Esta pregunta tiene fácil respuesta: porque el desarrollo del método de consulta de topolgía (LinesDiscovery) se hizo en la solución que desarrolló GMV para otros métodos del estándar; en cambio todos los cambios de este prototipo serán en la misma solución que se encuentra la aplicación web.

[1].NET Standard (Microsoft Learn): `https://learn.microsoft.com/es-es/dotnet/standard/net-standard?tabs=net-standard-2-0`)

.NET Standard 2.0 tiene 32 638 de las 37 118 API disponibles.

Implementación de .NET	Compatibilidad con versiones
.NET y .NET Core	2.0, 2.1, 2.2, 3.0, 3.1, 5.0, 6.0, 7.0
.NET Framework [1]	4.6.1 [2], 4.6.2, 4.7, 4.7.1, 4.7.2, 4.8, 4.8.1
Mono	5.4, 6.4
Xamarin.iOS	10.14, 12.16
Xamarin.Mac	3.8, 5.16
Xamarin.Android	8.0, 10.0
Plataforma universal de Windows	10.0.16299, TBD
Unity	2018.1

8.4.1.1. Aplicación de Windows

Una vez hubo acceso al Modelo desde este nuevo proyecto de solución, se necesitaba también acceso a los métodos del estándar SIRI del Web Service remoto. No supuso un gran desafío, porque se hizo previamente en el primer prototipo: se añadió la referencia de servicio WCF y, a partir del wsdl, se obtuvo el acceso a todos los métodos y estructuras del estándar.

Otro reto fue el de utilizar desde cero la Arquitectura de GMV para poder insertar, actualizar y consultar datos en la BD. En el primer prototipo también llegamos a consultar datos en la BD, pero al ser un proyecto ya formado y con la arquitectura ya en uso, no hubo tanto problema.

8.4.1.2. Aplicación de servicios WCF

Para poder intercambiar información entre la vista y los datos de la BD se creó una Api Rest. El objetivo era poder, a partir de la api, consultar los datos insertados previamente por la aplicación de Windows en la vista, y así poder informar al cliente de la monitorización de paradas automática.

Para desarrollar la api hubo varios desafíos:

1. Recordar los conocimientos necesarios para desarrollar una api Rest sencilla.

2. Cambiar la fecha a un formato universal porque, si no, en la api recibía una fecha errónea o el valor mínimo de la fecha. A pesar de que parece un problema menor, fue complicado ver el origen para resolverlo.

3. Hubo también dificultades en la forma en que se pasaría el filtro de tipo de petición, puesto que, al ser un tipo enumerado con enteros asignados, lo pasaba como entero en la api. Esto no se puede hacer, ya que para la api toda la uri y con ello sus parámetros deben ser strings.

Para poder ir viendo los avances en la api se utilizó la herramienta llamada Postman, que permitió ir haciendo comprobaciones de la información que recibiría la aplicación web. Para ello, se realizaban las peticiones GET con la uri específica y pasando los parámetros necesarios.

8.4.1.3. Nueva vista para la monitorización automática

El último punto para finalizar el prototipo, fue crear una vista con el objetivo de poder mostrar toda esta información al cliente. Partiendo de la vista genérica que se realizó en el segundo prototipo, en un primer paso se crearon las dos partes diferenciadas, una para las peticiones y otra para los contenedores de petición. Supuso una gran diferencia respecto al resto de vistas, ya que la información mostrada cambiaría en función de un parte de la vista y la otra. Lo que se hizo fue dar un valor entero a cada parte, siendo el valor 1 para la vista de peticiones y el valor 2 para la de contenedores. El dar esta distinción permitió establecer que se tendría que ver en cada una.

La subvista de petición, tal y como se puede ver en la Figura8.5, constaría de un campo de texto con formato fecha, un desplegable para poder seleccionar el filtro por tipo de petición, el botón request para solicitar la información y una tabla con una serie de columnas diferentes de la subvista de los contenedores.

La subvista de contenedor: Figura8.6 constaría de un campo de texto con formato fecha y el botón request para solicitar la información.

Para poder facilitar la compresión de que tipo de petición era cada una de las filas de la tabla se coloreo el fondo de una de las celdas [15]. La transición de colores elegida para representar los tipos fue de verde a rojo, siendo la verde la petición más rápida y la roja la más lenta (verde y roja, por tanto, solo habrá una por contenedor de petición). El resto de colores, por ejemplo, el naranja que es para el tiempo de respuesta medio si podrán repetirse.

Finalmente y debido a que tardaba mucho en cargar las peticiones cuando se seleccionaba un día con muchas peticiones, se puso un máximo de 5000 peticiones por tabla y unos botones para iterar entre la lista.

8.4.2. Uso de un Centinela para la prueba

Lo único que faltaría para terminar el prototipo es conseguir que la aplicación de Windows se ejecute de forma automática, ya que el objetivo de este prototipo es que cada X tiempo se monitoricen las paradas, para así saber cuáles están fallando o tardan más de lo que deben.

Para conseguirlo hay variadas formas. La primera, quizá excesivamente farragosa, es con un while true y un sleep del tiempo que se quiera. Permitiría que se ejecute constantemente, una vez se termine el timeout. Tiene la ventaja de que no requiere de otras herramientas, pero sus desventajas son grandes: es una aplicación consumiendo recursos constantemente, sólo se podría parar apagándola manualmente y, por último, hay que darle autonomía para reiniciarse constantemente.

Otra forma de lograr ese automatismo es la herramienta 'Centinela' desarrollada para esta finalidad por GMV. Normalmente se utiliza para levantar las aplicaciones, servicios o procesos que, por el motivo que sea, se han parado o caído. En otras palabras, funciona como un monitor y supervisor de procesos. Esa sería su funcionalidad estándar, pero la que utilizaremos en este caso es su funcionalidad secundaria: ejecutar aplicaciones y procesos una vez termine el timeout configurable. Esta manera de hacerlo tiene indudables ventajas:

- Permitirá comprobar que no estén parados o caídos diferentes procesos, aplicaciones y servicios.
- Permitirá iniciar procesos y aplicaciones con un timeout configurable por aplicación y proceso.
- Permitirá quitar la autonomía a la aplicación o proceso de reiniciarse automáticamente.
- Si se apagará el centinela o se modificará su configuración, no realizaría ninguna de sus funciones. Esto puede servir en caso de que la máquina, por ejemplo, esté muy saturada.

8.5. Validación

Habiendo finalizado el tercer prototipo y siendo el último, es necesario hacer una validación completa del sistema. En primer lugar, se validarán únicamente las pruebas del tercer prototipo (Tabla **??**) y después se realizarán todas las pruebas realizadas a lo largo del proyecto, que sería realizando las otras dos, de los prototipos anteriores.

Por último se adjuntarán las dos subvistas de este último prototipo, que se pueden observar en las Figuras 8.11 y 8.12.

Finalmente, se realizaron las dos últimas revisiones, una con el cliente y otra con el tutor, en las que se mostraron en formato de demo todas las funcionalidades del subsistema.

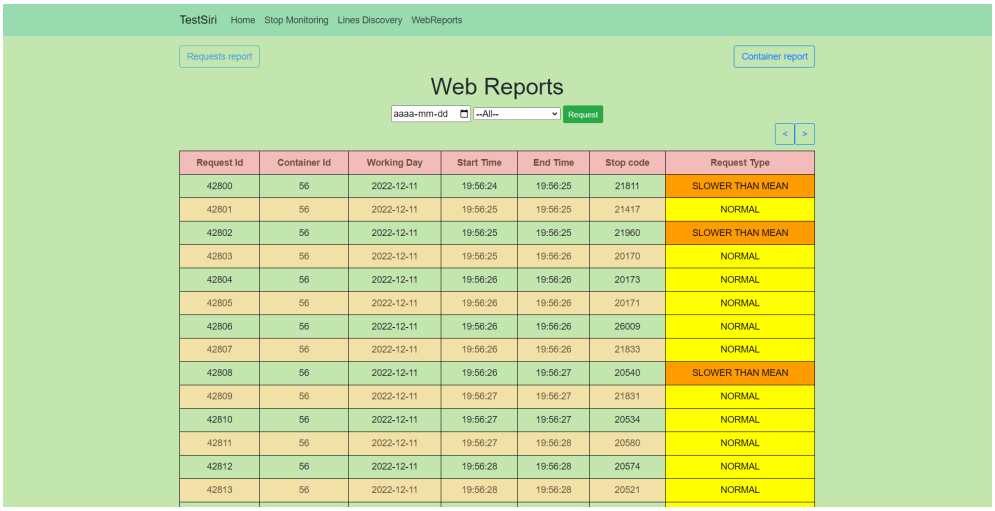


Figura 8.11: Subvista Web Reports para las peticiones una vez terminado el tercer prototipo.

Figura 8.12: Subvista Web Reports para los contenederos de petición una vez terminado el tercer prototipo.

Id	Descripción	Propuesta por	Resultado	Éxito
P2-P11 (CT/MMP)	Eliminar la lista de paradas solicitadas en más de un puesto	GMV	Comprobar que ya no se listan las líneas de petición de las paradas previamente solicitadas en esos puestos	SÍ
P2-P12 (CT/MMP)	Ver que cada parada se deshabilita al ser añadida a la lista de paradas solicitadas por puesto	Cliente	Comprobar que se deshabilita la parada al añadirla	SÍ
P2-P13 (CT/MMP)	Ver que al resetear un puesto se habilitan todas las paradas previamente deshabilitadas	GMV	Comprobar que se habilitan las paradas	SÍ
P2-P14 (CT/MMP)	Ver que al resetear un puesto se habilitan todas las paradas previamente deshabilitadas	GMV	Comprobar que se habilitan las paradas	SÍ
P3-P01 (IW)	Cambiar entre las dos subvistas (la de peticiones y la de contenedores)	Cliente	Comprobar que en la subvista de peticiones hay un campo desplegable con el filtro de tipo de petición y en la de contenedores no	SÍ
P3-P02 (IW)	Solicitar las peticiones sin filtro	Cliente	Comprobar que se han listado todas las peticiones sin tener en cuenta la fecha ni el tipo	SÍ
P3-P03 (IW)	Solicitar las peticiones para una jornada específica	Cliente	Comprobar que se han listado todas las peticiones para esa fecha específica	SÍ
P3-P04 (IW)	Solicitar las peticiones para una fecha y un tipo de petición específica	Cliente	Comprobar que se han listado todas las peticiones para esa fecha y ese tipo de petición	SÍ
P3-P05 (IW)	Solicitar los contenedores sin filtro	Cliente	Comprobar que se han listado todos los contenedores sin tener en cuenta la fecha ni el tipo	SÍ
P3-P06 (IW)	Solicitar los contenedores para una jornada específica	Cliente	Comprobar que se han listado todas los contenedores para esa fecha específica	SÍ

Tabla 8.4: Batería de pruebas del tercer prototipo de la aplicación (P3).

Capítulo 9

Conclusiones y trabajo futuro

9.1. Conclusiones

Esta memoria recoge la documentación elaborada durante del Trabajo de Final de Grado que ha consistido en el desarrollo de una aplicación web aunque, para lograr los objetivos, también cuenta con una aplicación de servicios WCF, una aplicación de Windows y un Web Service.

El resultado del proyecto es una aplicación totalmente funcional, capaz de consultar la topología vigente en una zona, monitorizar a tiempo real las paradas y comprobar el rendimiento y funcionamiento del sistema, mediante una monitorización automática de paradas en segundo plano.

La aplicación se ha desarrollado siguiendo una metodología de desarrollo evolutivo por prototipos, lo que ha permitido especificar más eficazmente los requisitos del cliente. También ha ayudado en la comunicación con él porque, tras cada validación, era posible definir qué aspectos o requisitos se deseaban mejorar o modificar para ajustarlos a sus necesidades.

La aplicación cumple los objetivos inicialmente definidos: tiene el alcance planteado desde un inicio. Incluso ofrece más funcionalidades de las previstas inicialmente como, por el ejemplo, poder moverse entre diferentes páginas en los Web Reports para mejorar el rendimiento de la vista. Este requisito se añadió durante el desarrollo del tercer prototipo y no estaba previsto en la estimación inicial. A pesar de ello, el resultado ha sido excelente y se ha realizado una correcta implementación de cada uno de los requisitos definidos.

Este Trabajo de Fin de Grado debería haber sido presentado en junio de 2022 como se ha detallado en la memoria, pero diferentes circunstancias han hecho que finalmente se haya retrasado.

- Por una parte, el hecho de que GMV después de terminar las prácticas me contratará, lo que supuso un aumento de la carga de trabajo dedicada a la empresa y una lógica reducción de las horas que podía dedicar al TFG. Además, fue imprescindible el documentarse y formarse en muchas herramientas para poder estar a la altura de la situación.
- Por otra parte, el objetivo de este TFG era intentar hacer un buen trabajo, muy testado y con unos resultados comprobables. Y todo este proceso lleva mucho tiempo y mucho ensayo y error, sin contar con el tiempo que ha sido preciso documentarse a priori y durante el propio desarrollo.

Por todo ello, finalmente, solo se pudo realizar el primer prototipo entre las fechas que se estimaron originalmente de inicio y fin de proyecto, ya que al comprobar el retraso que llevaba respecto a las fechas estimadas por prototipo se planteó retrasarlo hasta estas fechas (Figura 9.1), lo que se ve reflejado en los meses situados en el medio del gráfico que no hubo avances (Figura 9.2).

	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Formación	12 días	lun 14/02/22	mar 01/03/22	
2		**Primer prototipo**	**38 días**	**mié 02/03/22**	**vie 22/04/22**	
3		Análisis y especificación	6 días	mié 02/03/22	mié 09/03/22	1
4		Diseño	6 días	jue 10/03/22	jue 17/03/22	3
5		Implementación	20 días	vie 18/03/22	jue 14/04/22	4
6		Validación	6 días	vie 15/04/22	vie 22/04/22	5
7		**Segundo prototipo**	**32 días**	**mar 20/09/22**	**mié 02/11/22**	
8		Análisis y especificación	6 días	mar 20/09/22	mar 27/09/22	6
9		Diseño	6 días	mié 28/09/22	mié 05/10/22	8
10		Implementación	16 días	jue 06/10/22	jue 27/10/22	9
11		Validación	4 días	vie 28/10/22	mié 02/11/22	10
12		**Tercer prototipo**	**21 días**	**jue 03/11/22**	**jue 01/12/22**	
13		Análisis y especificación	4 días	jue 03/11/22	mar 08/11/22	11
14		Diseño	4 días	mié 09/11/22	lun 14/11/22	13
15		Implementación	10 días	mar 15/11/22	lun 28/11/22	14
16		Validación	3 días	mar 29/11/22	jue 01/12/22	15

Figura 9.1: Diagrama de Gantt final (Parte 1).

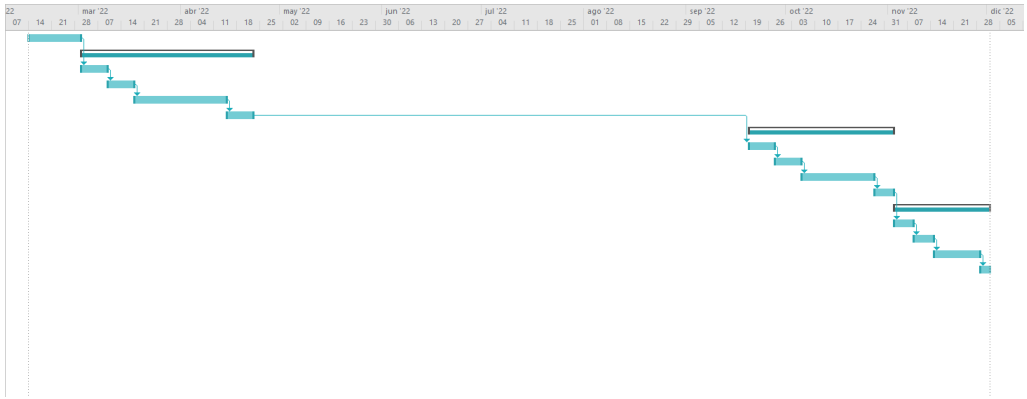


Figura 9.2: Diagrama de Gantt final (Parte 2).

9.2. Trabajo futuro

Este TFG responde a una necesidad y puede considerarse un proyecto completo. No obstante, existen ciertos puntos que podrían ser objeto de mejora en un futuro:

- Una de las mejoras más obvias y que tarde o temprano se realizarán, es extender el alcance del proyecto al resto de los métodos SIRI que se vayan desarrollando o se hayan desarrollado. Un ejemplo de otro de los métodos que se podrían adaptar es el Vehicle Monitoring, que mide el posicionamiento a tiempo real del autobús.
- Otra mejora que se podría llevar a cabo, una vez se publicase o desplegase todo el subsistema en un entorno de producción, sería unificar la BD con la intención de reducir la complejidad de la arquitectura. El diagrama de despliegue que resultaría, una vez se desplegase en el servidor de producción, se puede observar en la Figura 9.3.
- También podrían realizarse cambios en las interfaces, en caso de que el cliente lo requiera. Aunque el proyecto se ha revisado de forma continua con el cliente, es posible que quiera cambiar algo una vez lo pruebe más en detalle.
- Otra posibilidad a explorar sería mostrar la línea que corresponde con el recorrido más corto entre dos puntos. Esto podría resultar útil para viajes que requieran de transbordos.
- Este proyecto no es extensible a bases de datos de Oracle, por lo que una propuesta interesante sería realizar este desarrollo y hacerlo configurable para que, el usar un Sistema Gestor de Bases de Datos u otro, fuera decisión del cliente. Ello le daría una mayor versatilidad.

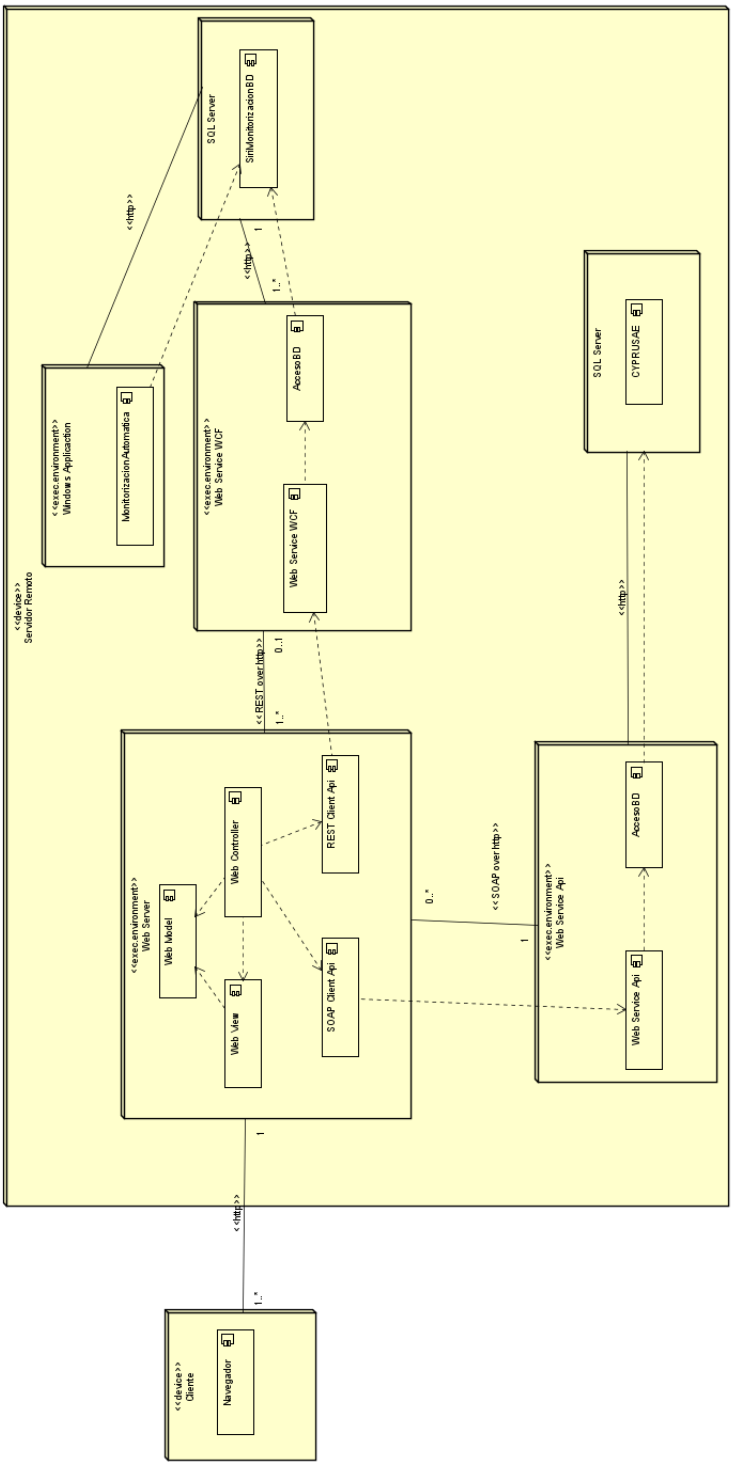


Figura 9.3: Diagrama de despliegue una vez se haya publicado en un servidor de producción real.

9.3. Código fuente de la aplicación

Todo el trabajo reflejado en esta memoria ha sido realizado íntegramente por el autor. La planificación y seguimiento, análisis, diseño y codificación han sido realizados en el marco de las actividades del autor para el estudio de las posibilidades de SIRI en el contexto indicado del grupo empresarial GMV. Por esta razón, el código se encuentra sujeto a las condiciones contractuales del autor con la empresa en el marco regulado de la Ley de la Propiedad Intelectual y no es público. El tribunal puede revisar y consultar este código si así lo desea bajo la condición de confidencialidad, si lo estima conveniente y así lo solicita.

9.4. Agradecimientos

Finalmente, es preciso dejar constancia y agradecer el seguimiento continuo de mi tutor de la UVA, Don César Llamas Bello, que me ha hecho valiosas aportaciones y me ha motivado durante todo este tiempo, especialmente cuando tuve que posponerlo unos meses cuando empecé a trabajar a GMV. También debo agradecer su apoyo a mis compañeros de la empresa, en particular a mi tutor, Don Antonio Abascal Lagunas.

Lista de Figuras

Lista de Tablas

Bibliografía

[1] Rick Anderson. Referencia sobre la sintaxis de Razor para ASP.NET Core. https://learn.microsoft.com/es-es/aspnet/core/mvc/views/razor?view=aspnetcore-7.0, 2022. visited on 2022-03-12.

[2] Martin Bertucelli. Arquitectura de capas. https://somospnt.com/blog/118-arquitectura-de-capas, 2019. visited on 2022-02-25.

[3] David Britch. Consumo de un servicio web ASP.NET (ASMX). https://learn.microsoft.com/es-es/xamarin/xamarin-forms/data-cloud/web-services/asmx, 2022. visited on 2022-02-20.

[4] Luis del Valle Hernández. Peticiones Ajax desde ASP.NET MVC. https://programarfacil.com/tutoriales/fragmentos/realizado-peticiones-ajax-desde-asp-net-mvc/, 2017. visited on 2022-03-30.

[5] Dondon. Set asp-route-id dynamically. https://learn.microsoft.com/en-us/answers/questions/838864/set-asp-route-id-dynamically.html, 2022. visited on 2022-04-12.

[6] IBM. WSDL (Web Services Description Language). https://www.ibm.com/docs/es/rsas/7.5.0?topic=standards-web-services-description-language-wsdl, 2021. visited on 2022-03-17.

[7] IBM. ¿Qué es Ajax? https://www.ibm.com/docs/es/rational-soft-arch/9.6.1?topic=page-asynchronous-javascript-xml-ajax-overview, 2021. visited on 2022-03-26.

[8] Infinitia. Estudio de viabilidad de un proyecto ¿Cómo realizarlo? https://www.infinitiaresearch.com/noticias/estudio-de-viabilidad-de-un-proyecto-como-realizarlo/#viabilidad_operativa, 2021. visited on 2022-03-08.

[9] Jimi. Converting URL to Action Link @url.action. https://stackoverflow.com/questions/71631937/converting-url-to-action-link-url-action, 2022. visited on 2022-03-06.

[10] jitt. HTML — ¡td¿bgcolor Attribute. https://www.geeksforgeeks.org/html-td-bgcolor-attribute/, 2022. visited on 2022-11-10.

[11] Jmancherje. How to extract values from HTML ¡input type=”date» using jQuery. https://stackoverflow.com/questions/33312318/how-to-extract-values-from-html-input-type-date-using-jquery, 2015. visited on 2022-11-10.

[12] Sachin Kainth. How do I add a .svc file in Visual Studio. https://stackoverflow.com/questions/11760138/how-do-i-add-a-svc-file-in-visual-studio, 2012. visited on 2022-11-20.

[13] Luis. ¿Cómo girar una tabla con LaTeX? http://minisconlatex.blogspot.com/2014/10/como-girar-una-tabla-con-latex.html, 2014. visited on 2022-09-16.

[14] Pedro Martín. ¿Qué es C#? https://bsw.es/que-es-c/, 2021. visited on 2022-02-18.

[15] Windson Mateus. How do you change background color on td in CSS? https://stackoverflow.com/questions/66233077/how-do-you-change-background-color-on-td-in-css, 2021. visited on 2022-11-10.

[16] Mitsurugi. parseInt(). https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Global_Objects/parseInt, 2022. visited on 2022-03-03.

[17] Marytere Narvaez. ¿Qué es un estudio de viabilidad y cómo se realiza? https://www.questionpro.com/blog/es/estudio-de-viabilidad/, 2018. visited on 2022-03-06.

[18] Visual Paradigm. Seis tipos de relaciones en uml. https://blog.visual-paradigm.com/es/what-are-the-six-types-of-relationships-in-uml-class-diagrams/, 2022. visited on 2022-04-10.

[19] Carlos Quintana. Estudio de viabailidad de un proyecto: ¿Qué es? y ¿cómo se hace? https://www.oberlo.es/blog/viabilidad-de-un-proyecto, 2021. visited on 2022-03-10.

[20] Niklas Rosencrantz. How to properly distinguish between mvc and three-tier? https://softwareengineering.stackexchange.com/questions/228978/how-to-properly-distinguish-between-mvc-and-three-tier, 2014. visited on 2022-03-08.

[21] Sandy. Conceptos básicos de Servicios Web SOAP, WSDL y XSD. http://desarrolloconsoa.blogspot.com/2014/02/conceptos-basicos-de-servicios-web-soap.html, 2014. visited on 2022-03-14.

[22] Steve Smith. Información general de asp.net core mvc. https://learn.microsoft.com/es-es/aspnet/core/mvc/overview?view=aspnetcore-7.0, 2016. visited on 2022-02-15.

[23] Fabio Terracini. The mvc design pattern on web applications and the three tier architecture. https://terracini.com/mvc-design-pattern-three-tier-architecture/, 2020. visited on 2022-02-28.

[24] Eduard Tomàs. Analizamos la sintaxis y sus consideraciones para el motor de vistas Razor en .NET. https://desarrolloweb.com/articulos/motor-vistas-razor-dotnet.html, 2011. visited on 2022-03-16.

[25] Transmodel. Standard Interface for Real-time Information. https://www.transmodel-cen.eu/siri-standard/, 2018. visited on 2022-02-16.

[26] Jorge Trejos. Modelo Evolutivo. http://jorgetrejos.blogspot.com/2010/08/modelo-evolutivo.html, 2010. visited on 2022-02-27.

[27] Ukpai Ugochi. Deployment diagrams explained in detail, with examples. https://www.plutora.com/blog/deployment-diagrams-explained-in-detail-with-examples, 2021. visited on 2022-11-10.

[28] Rob Volk. LINQ: Select an object and change some properties without creating a new object. https://stackoverflow.com/questions/807797/linq-select-an-object-and-change-some-properties-without-creating-a-new-object, 2009. visited on 2022-10-04.

[29] Bill Wagner. Herencia en C# y .NET. https://learn.microsoft.com/es-es/dotnet/csharp/fundamentals/tutorials/inheritance, 2022. visited on 2022-03-05.

[30] Wikipedia. Service interface for real time information. https://en.wikipedia.org/wiki/Service_Interface_for_Real_Time_Information, 2016. visited on 2022-02-15.