

INSTITUTO DE EDUCACIÓN SECUNDARIA JOSÉ PLANES

Departamento de Informática y Comunicaciones
Técnico Superior en Desarrollo de aplicaciones Web

C/ Maestro Pérez Abadía, 2
30100 Espinardo – Murcia
T. 968 834 605
30010577@murciaeduca.es
www.iesjoseplanes.es

Memoria del proyecto

Desarrollo de aplicaciones web

Bus&Co

Autores/as:

Zaén Enrique Martínez Abellán

Antonio Gallego Peñalver

Lorena Martínez Martínez

Profesor/a-coordinador/a:

María del Carmen Martínez Sánchez

Murcia, junio de 2024



Esta obra está bajo una [licencia de Creative Commons Reconocimiento-CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-sa/4.0/).

Queremos trasladar nuestro más sincero agradecimiento a aquellas personas que han sido un apoyo incondicional para que podamos presentar este proyecto.

En primer lugar, a todos las profesoras y profesores de ciclo de grado superior de Desarrollo de Aplicaciones Web que han sido nuestros guías en este camino formativo. Sin su ayuda, enseñanzas y esfuerzo, no habríamos alcanzado el desarrollo profesional y personal del que hoy podemos hacer alarde. Destacando una mención especial a nuestra tutora, Carmen, que nos ha acompañado con su compromiso, consejos y confianza a lo largo del desempeño de este trabajo.

A nuestros amigos y familiares por darnos su apoyo y comprensión en esta aventura académica, y que han sido testigos de nuestros malos y buenos momentos, ayudándonos a superar cada obstáculo.

A nuestros compañeros de clase, que han sido dignos acompañantes en estos dos años de ciclo y de los cuales también hemos aprendido, compartido momentos y objetivos comunes.

Y a Trino, Sabina, y Alberto por su inestimable ayuda.

Gracias de todo corazón.

Contenido

1	Resumen extendido.	4
2	Palabras clave.	5
3	Introducción.	5
4	Estado del arte/trabajos relacionados.	6
4.1	Webs de empresas concesionarias	6
4.2	Aplicaciones móviles de empresas concesionarias	9
4.3	Aplicaciones externas.	9
4.4	Aspectos comunes.	10
4.5	Valor añadido de nuestro proyecto.	10
5	Análisis de objetivos y metodología.	11
5.1	Objetivos del proyecto.	19
5.2	Licencia.	20
6	Diseño y resolución del trabajo realizado.	21
6.1	Diseño de la interfaz.	21
6.1.1	Logotipo.	30
6.2.2	Accesibilidad y usabilidad.	31
6.2	Desarrollo web.	35
6.3.1	Esquema de la base de datos	35
6.3.2	Desarrollo lado servidor	36
6.3.3	Desarrollo del entorno de cliente	46
6.3.4	Despliegue de la aplicación	54
7	Uso de la Inteligencia Artificial	62
8	Presupuesto	66
9	Conclusiones y vías futuras	67
10	Bibliografía/Webgrafía.	68

10.1 Atribuciones.	68
10.1.1 Imágenes.	68
10.1.2 Elementos HTML y CSS.	69
10.1.3 Fuentes.	69
11 Anexos	70
11.1 Anexo I: Pasos a seguir para la instalación y puesta en marcha del proyecto.	70
11.2 Anexo III: Enlaces de interés.	71

1 Resumen extendido.

Bus&Co es un prototipo de aplicación web que ha sido desarrollada mediante el uso de Angular y TypeScript como lenguaje de programación en el entorno cliente, así como Symfony y PHP como lenguaje de programación en el entorno de servidor, y librerías como PHPUnit para realizar test unitarios o LexikJWTAuthentication para autenticación.

HTML5 y CSS3 se han usado tanto para crear la estructura de contenido como la estructura de presentación de las distintas pantallas de la aplicación.

El objetivo principal de Bus&Co es ofrecer una potencial solución de consulta de información unificada a los usuarios de transporte público en autobús del municipio de Murcia. Podrán buscar líneas de autobuses seleccionando origen y destino del viaje, realizar consultas sobre información tanto de línea como paradas, horarios, recorrido, etc. Así como consultar noticias relativas al transporte público y posibles incidencias que puedan afectar a su trayecto.

Además, los administradores asociados a las distintas empresas de transporte, por su parte, podrá añadir, editar o eliminar cualquier información relativa a las líneas de forma sencilla, a través del uso de formularios intuitivos.

2 Palabras clave.

Transporte público, autobús, Angular, Symfony, diseño responsive, MariaDB, accesibilidad, usabilidad, AWS, Vercel, tests, JWT, Typescript, Php, Xampp, Lamp, diseño, inteligencia artificial.

3 Introducción.

La sociedad actual está cada vez más concienciada con el respeto y cuidado al medio ambiente. Así, numerosos gobiernos, corporaciones locales y entes públicos están tomando medidas drásticas para frenar la contaminación en las ciudades para mejorar la calidad de vida de los ciudadanos. Es por ello que el transporte público cobra más relevancia que nunca.

El municipio de Murcia tiene un elevado tamaño territorial, 881,8 km² y además cuenta con la existencia de numerosos núcleos urbanos dispersos, por lo que la movilidad es algo de especial importancia. El transporte de viajeros en autobús es el medio de transporte público con mayor alcance del municipio y una alternativa excelente para paliar los efectos de la contaminación.

Tener un buen sistema de transporte público en autobús es vital, al igual que facilitar toda la información relativa sobre éste a la población, ya que cuando mejor sea ese acceso a la información más se fomentará su uso.

Actualmente, existen más de 5 empresas concesionarias de transporte público en autobús que operan en el municipio. Cada una de ellas dispone de una web independiente, con formas distintas de presentar la información y divididas en zonas geográficas diferentes, por lo que es muy complicado para los usuarios poder encontrar de forma fácil los datos que necesitan, especialmente aquellos que viven en pedanías o núcleos urbanos distintos de la ciudad de Murcia.

Es por todo ello, que tomamos la decisión de crear un prototipo de aplicación que pudiera abordar este problema y darle solución. Enfocando el diseño de la misma

a una apariencia sencilla y accesible que facilitara su uso a los colectivos más vulnerables como las personas de edad avanzada.

4 Estado del arte/trabajos relacionados.

Ya que el prototipo de aplicación web tiene como temática principal el transporte público en autobús, es vital tomar como punto de partida el análisis de las webs y aplicaciones móviles de las empresas de autobuses disponibles, así como de las aplicaciones externas que actualmente son las utilizadas por los usuarios

4.1 Webs de empresas concesionarias

Teniendo en cuenta las webs de las distintas empresas concesionarias, destacamos las de las empresas TMP Murcia (antigua Latbus) y TMurcia (“los coloraos”) como las webs de referencia.

En la primera, podemos destacar que se trata de un diseño sencillo, pero con cierta desproporción en los elementos, como se puede apreciar en la siguiente imagen.



Figura 1. Captura de la página de inicio de la web de TMP Murcia

La información de las líneas es clara, y se pueden hacer búsquedas por origen y destino incluyendo combinaciones, mostrando la información relevante en cuando a paradas, expediciones, precio o kilómetros recorridos:

Líneas y horarios

Origen destino

Planos de líneas

Tarifas y bonos

Contacto

Noticias

Trabaja con nosotros

Líneas y horarios

home

resultados de la búsqueda

Volver

ORIGEN

MURCIA

DESTINO

MOLINA DE SEGURA

Línea

Nombre

Expediciones

Precio

Km

Bono

Línea

Parada

Expediciones

Precio

Km

Bonos

44

EXTRANJERIA
HERO

9

2,00 Euros

14,59

!

14

HERO
LA VIA II

8

2,00 Euros

5,2

Figura 2. Captura de la página de líneas y horarios de la web de TMP Murcia

Como parte relevante de la web, nos encontramos con un desplegable con todas las líneas de la compañía mediante el cual se puede seleccionar aquella de la que se quiera descargar el plano en PDF:

Líneas y horarios	Origen destino	Planos de líneas	Tarifas y bonos	Contacto	Noticias	Trabaja con nosotros
Planos de líneas						↑ home ← Volver
búsqueda directa						
<div>Urbanas e Interurbanas</div> <div>1: SAN GINES - MURCIA</div>						
<div> <div>✕</div> <div>▶</div> </div> <div> <div>borrar</div> <div>confirmar</div> </div>						

Figura 3. Captura de la página Planos de líneas de la web de TMP Murcia.

Al analizar la segunda web correspondiente a la empresa TMurcia, nos encontramos con un diseño mucho más complejo. El color de la fuente en el menú lateral no tiene un buen contraste como se puede apreciar en la siguiente imagen, donde lo analizamos con la herramienta Color Contrast Checker para navegador Chrome:

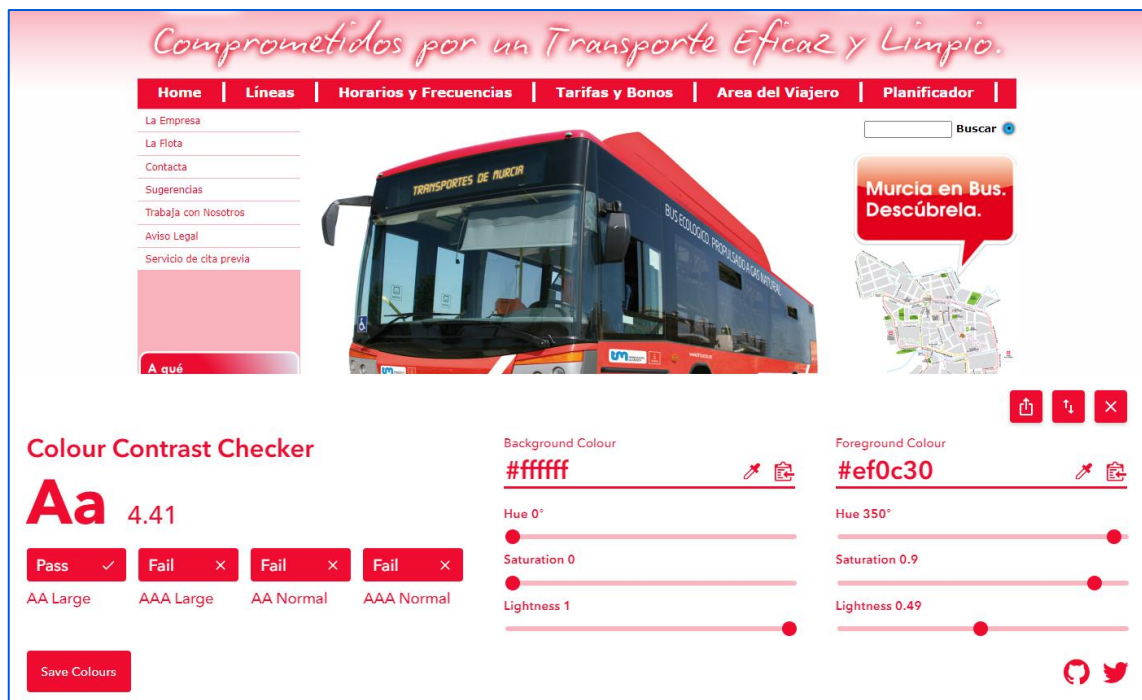


Figura 4. Captura de pantalla de la página de inicio de la web de TMurcia.

En cuanto a la información detallada ofrecida por esta, la gran diferencia con respecto a la anterior es que no proporciona la búsqueda por origen y destino, tan sólo un planificador de rutas cuya funcionalidad no es nada práctica.



Figura 5. Captura de pantalla de la página de horarios y frecuencias de la web de TMurcia.

Mientras que como puntos comunes en ambas se encuentra la posibilidad consultar los datos de las líneas y las paradas y de descargar o visualizar los planos en pdf.

4.2 Aplicaciones móviles de empresas concesionarias

En cuanto a las aplicaciones de las empresas concesionarios, nos encontramos que sólo existe una correspondiente a la empresa TMurcia.



Figura 6. Captura de pantalla de la aplicación de TMurcia

En ella se pueden realizar las mismas acciones que en la web y además ofrece información adicional como puntos turísticos de la ciudad, paradas cercanas o los tiempos esperados de llegadas de los autobuses a las paradas.

4.3 Aplicaciones externas.

Moovit es una aplicación web con versión móvil que actúa como buscador de transporte público entre dos puntos geográficos. Opera a nivel nacional y entre sus funciones encontramos la de buscar rutas, obtener información de líneas y horarios o conocer los datos de las empresas que operan en una región concreta.

Su principal inconveniente es que se trata de una web con versión de pago y versión gratuita con publicidad. Esta última opción resulta muy incómoda para el usuario, siendo a veces, muy complicado encontrar la información de forma fácil.

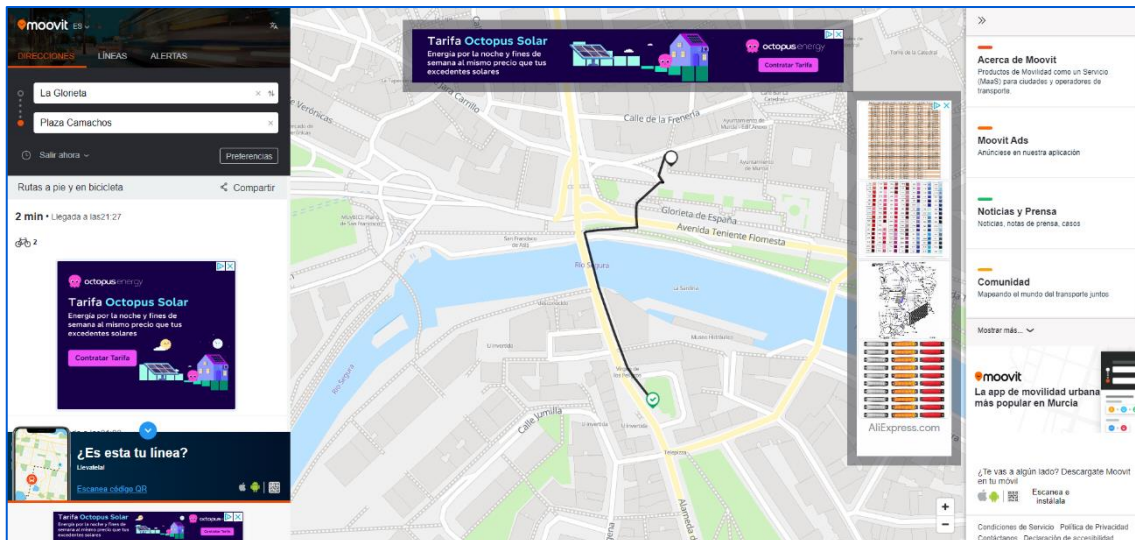


Figura 7. Captura de pantalla de la aplicación de Moovit

4.4 Aspectos comunes.

A pesar de haber dividido las soluciones disponibles en distintos apartados, éstas presentan características comunes, tales como:

- **Información sobre líneas, paradas y horarios:** Todas cuentan con información del nombre o referencia de cada línea, las paradas asociadas a ellas y los horarios de cada parada.
- **Sección de noticias:** Un apartado donde se puede consultar noticias e información de interés relacionada con el transporte público.
- **Sección de contacto:** Información relativa al contacto de la empresa concesionaria, con datos como nombre, dirección física, teléfono, email o web.
- **Planos o itinerarios de rutas:** Pestaña donde consultar todos recorridos de las líneas, mediante plano o esquema circular.

4.5 Valor añadido de nuestro proyecto.

Las distintas ventajas competitivas de nuestro proyecto con respecto al resto son, en primer lugar, la mejora en la usabilidad de la aplicación. Siendo nuestra web una opción mucho más sencilla, intuitiva y atractiva. En segundo lugar, la accesibilidad, ya que hemos diseñado y desarrollado la aplicación teniendo en cuenta los criterios de la WCAG 2, para que pueda ser usada por la mayoría de la población con independencia de sus capacidades. Y, por último, la principal ventaja es que se trata de una solución unificada y gratuita para el usuario, que pretende ofrecer toda la información del

transporte en autobús del municipio de Murcia de forma estandarizada. Esto evitará posibles errores en los trayectos de los viajeros.

5 Análisis de objetivos y metodología.

Una vez realizado el estudio de los trabajos relacionados y webs similares, es necesario detallar en colaboración con el cliente los requerimientos funcionales y no funcionales del sistema.

En una primera entrevista inicial con el cliente, el Ayuntamiento de Murcia, realizamos la toma de requisitos para así poder detallar lo que se esperaba de la aplicación. En la reunión, nos explicó, que lo que busca es una web donde los usuarios puedan consultar fácilmente la información del transporte en autobús del municipio. Para la primera versión, solicitan que puedan realizar consultas de líneas de autobuses buscando por origen y destino, pudiendo seleccionar la fecha y la hora deseadas. También la opción de ver un listado de las líneas disponibles con la posibilidad de buscar y filtrar por tipo de línea o empresa. Además, se debe acceder a la información detallada de cada línea, incluyendo el mapa con el recorrido, el listado de paradas que cambiará según la dirección de la marcha y los horarios de cada parada según el día (laboral, sábado o festivo).

Para la parte del administrador, éste debe poder acceder a la parte privada mediante nombre de usuario y contraseña. En esta área privada, podrá consultar el listado de líneas activas e borradas, crear nuevas líneas, editarlas, borrarlas y restaurarlas.

Los usuarios administradores no podrán darse de alta por sí mismos, ya que, al tratarse de administradores corporativos de las empresas concesionarias, debemos ser los administradores de la aplicación quienes creen sus perfiles. Además, los usuarios administradores tampoco deben tener la capacidad de crear empresas, sino que esto también debe ser gestionado por los súper administradores.

Como siguiente paso para la planificación y diseño de la aplicación, se elaboró un documento siguiendo la metodología MoSCoW, para priorizar las funcionalidades en función de los requerimientos explicados por el cliente:

Las letras resaltadas corresponden al acrónimo:

M for Must Have (Debe tener): estos requisitos son indispensables para el éxito del proyecto. Por lo tanto, deben llevarse a cabo en primer lugar y son innegociables.

S de Should Have (Debería tener): esta categoría agrupa los requisitos importantes a realizar una vez finalizados los anteriores. Aportan un valor añadido real al proyecto y contribuyen al logro de los objetivos, pero a diferencia de los «imprescindibles», pueden alargarse en el tiempo.

C de Could have (Podría tener): estas son las tareas de comodidad que se realizarán mientras sea posible, si hay tiempo una vez finalizadas las tareas de las dos primeras categorías. Su realización no debe afectar a las demás tareas.

W de Won't have but would like to in the future (No tendrá, pero me gustaría en el futuro): estas son las tareas secundarias que nos gustaría realizar algún día, pero que dejamos de lado por ahora, por falta de presupuesto y / o falta de tiempo.

Esta técnica se suele usar en metodologías ágiles y se debe revisar en cada actualización del producto. Pero en nuestro caso, realizamos una adaptación y la usamos para determinar junto al cliente aquellas funcionalidades básicas y ampliadas que debía tener la aplicación como mínimo, para priorizar lo importante y ajustarnos al tiempo limitado disponible para desarrollar la aplicación.

Must Have	Should Have
<p>Usuario</p> <ul style="list-style-type: none">-Búsqueda por origen y destino con fecha y hora-Vista generada debe ser un listado de las líneas que cumplen los criterios, primero las directas y luego las combinaciones (mostrando el nombre de la línea, origen-destino, 3 siguientes horarios)-Vista de incidencias surgidas-Sección de noticias relevantes-Sección de contacto <p>Admin</p> <ul style="list-style-type: none">-Acceso con seguridad al área privada-Vista con formulario para introducir datos de las líneas-Vista formulario para modifica datos de las líneas	<ul style="list-style-type: none">-Filtro por compañía (selector de empresa)-Filtro de fecha-Filtro por hora-Incluir mapa en el detalla de línea

-Opción de borrado lógico de las líneas y su posterior recuperación	
Could Have	Won,t have
-Iconos de advertencia de incidencia junto a las líneas afectadas	-Geolocalización del usuario -Rutas favoritas -Plataforma de pago

Tabla 1. Adaptación método Moscow

Como segundo paso de la planificación y diseño, se elaboró un diagrama de casos de uso (Figura 1) en el que se representan los actores y su interacción con el sistema. Este esquema nos sirvió para poder tener mucho más claro los distintos casos de uso y escenarios posibles y que, tanto el desarrollo posterior como la definición de los requisitos funcionales, resultaran más sencillos.

Los actores principales son el usuario de autobuses y el administrador de la empresa.

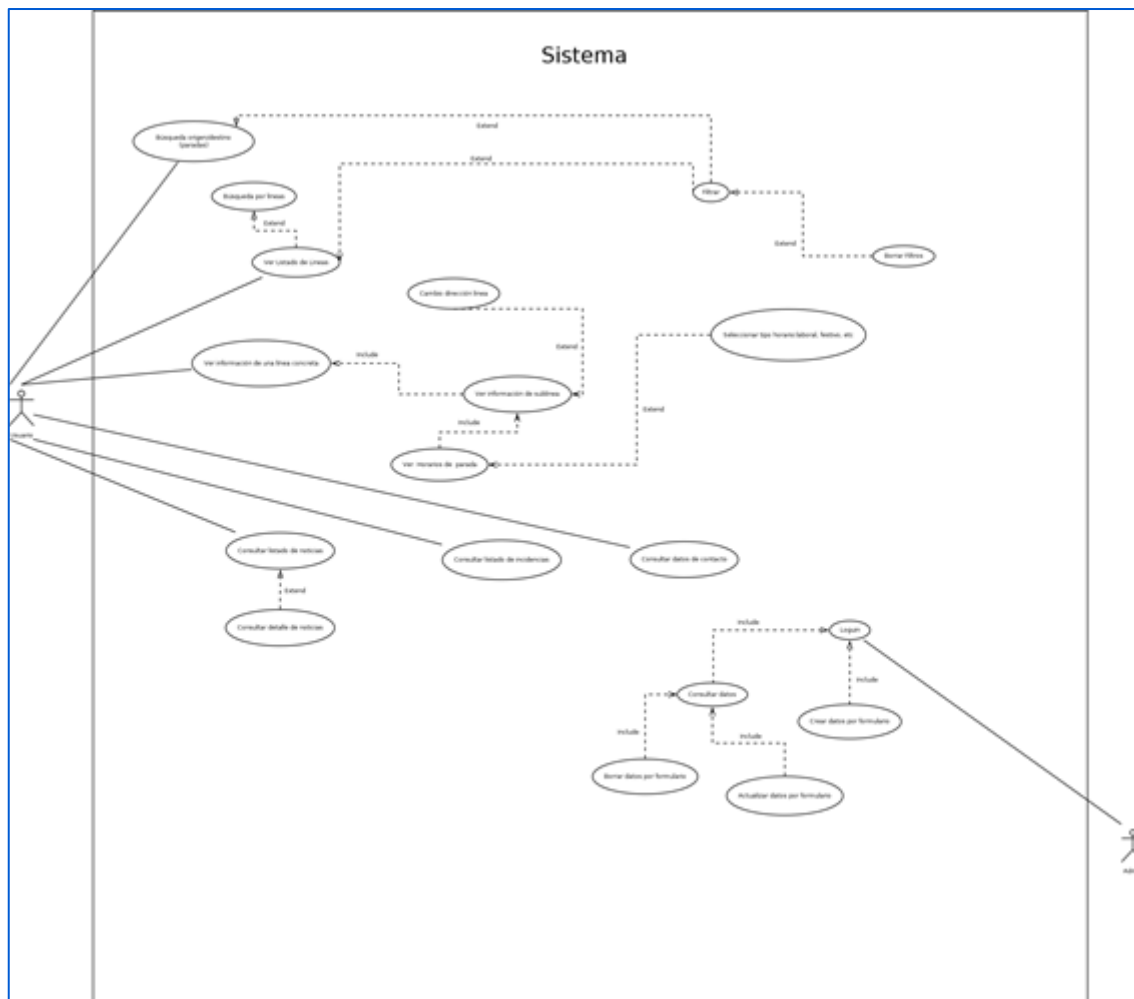


Figura 8. Diagrama de casos de uso elaborado

Tras la aprobación del cliente, que considera que quedan correctamente plasmadas todas las funciones que se precisan para la primera versión, se pasa a elaborar de forma más detallada unas tablas con dichos requerimientos funcionales y no funcionales. Entendiendo como requerimientos funcionales los que determinan las funciones y procesos de la web.

Actor	Requerimiento Funcional	Descripción
Usuario	Búsqueda por origen y destino	El usuario introducirá un origen tecleando caracteres en la caja de búsqueda, le aparecerá un desplegable con posibles coincidencias a medida que va incorporando más caracteres. Podrá seleccionar la coincidencia elegida y ésta se quedará fija en Origen. Para el destino procederá de la misma manera. Finalmente hará click en el botón de búsqueda.
Usuario	Búsqueda por Línea	El usuario introducirá un texto tecleado en la caja de búsqueda para localizar la línea buscada, el listado se actualiza a medida que va tecleando los caracteres y mostrando las coincidencias.
Usuario	Filtros	El usuario puede seleccionar los filtros de empresa, tipo y fecha para que el listado de líneas sólo muestre las líneas que cumplan esos criterios.
Usuario	Borrar Filtros	El usuario puede clickar en botón Borrar Filtros y los valores filtrados se resetean.
Usuario	Ver Listado de Líneas	El usuario podrá visualizar el listado completo de las líneas en la pestaña correspondiente
Usuario	Ver información de una línea concreta	El usuario podrá visualizar la información de la línea con una sublínea por defecto que haya seleccionado desde la pantalla de ver líneas

Usuario	Cambiar Sublinea seleccionada	El usuario hará click en el selector para cambiar de sublinea
Usuario	Cambio de dirección	El usuario podrá cambiar la dirección del recorrido de la sublinea haciendo click en el botón "Cambiar Dirección"
Usuario	Consultar horarios	El usuario hará click en el botón "ver Horario" de una parada en concreto y podrá visualizar los horarios de la misma
Usuario	Seleccionar tipo de horario	El usuario podrá hacer click en el selector de tipo de horario para mostrar las horas correspondientes
Usuario	Consultar listado noticias	
Usuario	Consultar detalle noticia	
Usuario	Consultar listado incidencias	
Usuario	Consultar datos de contacto	
Admin	Login	El usuario introduce usuario y contraseña para iniciar sesión en Área Privada
Admin	Consulta de datos	El usuario puede ver la información de las líneas e incidencias asociadas a su empresa
Admin	Crear datos	El usuario podrá crear los datos de una nueva línea mediante formulario

Admin	Actualizar datos Líneas	El usuario podrá seleccionar la línea y podrá modificar su información
Admin	Actualizar datos Incidencias	El usuario podrá seleccionar la incidencia y podrá modificar su información
Admin	Borra datos de línea	El usuario podrá borrar los datos de la Línea seleccionada, deberá tener doble confirmación para ello

Tabla 2. Requerimientos funcionales del sistema.

Pasando a continuación a la definición de los requerimientos no funcionales (Tabla 2), que, aunque no son estrictamente necesarios para la funcionalidad del sistema, sí son importantes para un buen desarrollo.

Requerimiento No Funcional	Descripción
Diseño responsive	El diseño de la web deberá adaptarse adecuadamente a los diferentes tamaños de pantalla existentes.
Colores	La web se limitará al uso de los colores: blanco, dos tonos de azul y amarillo.
Accesibilidad	La web contará con medidas para que sea usable por todo tipo de perfiles, haciendo uso de, por ejemplo, un contraste adecuado.
Diseño intuitivo y consistente	El software tiene que ser fácil de entender y con un diseño consistente visualmente en cuanto a tipografía, colores y disposición de los elementos.
Privacidad	Se debe garantizar la protección de los datos de los usuarios y prevenir el acceso no autorizado a ciertas secciones de la web.
Gestión adecuada del proyecto	El desarrollo se hará en continua comunicación con el cliente y con los miembros del equipo.
Consumo de recursos	El software resultante no debe consumir demasiados recursos del sistema.

Despliegue	El software debe funcionar adecuadamente en un entorno de producción, garantizando tanto la seguridad como la escalabilidad.
------------	--

Tabla 3. Requerimientos no funcionales del sistema.

Siguiendo con los aspectos metodológicos, hemos convenido hacer una recopilación de todas las herramientas y frameworks de los que se hará uso (Tabla 3) para la adecuada obtención del resultado final en su primera versión, siendo éste ampliable según surjan nuevas necesidades o funcionalidades.

Herramienta	Uso
Microsoft Teams	Herramienta de comunicación y gestión de proyectos, donde poder almacenar archivos, información, compartir recursos, planificar flujos de trabajo y realizar videollamadas.
GitHub / GitHub Desktop	Herramienta utilizada para el control de versiones y gestión del proyecto.
WhatsApp / Telegram	Canales utilizados para una comunicación más ágil e inmediata del equipo.
Angular	Framework de desarrollo web de código abierto, basado en el lenguaje TypeScript que permite crear aplicaciones web mediante el uso de componentes, servicios, enrutamiento, inyección de dependencias y otros conceptos avanzados de programación.
Symfony	Entorno de desarrollo web, basado en el patrón modelo, vista, controlador, desarrollado completamente en PHP. Permite crear aplicaciones web y APIS.
Xdebug	Extensión para Visual Studio Code cuya función es la ayudar en la depuración de código PHP.
Figma	Herramienta utilizada para el diseño de prototipos, tanto de bajo como de alto nivel.
Contrast Finder	Herramienta en línea que permite verificar el contraste entre dos colores y evaluar si cumplen con las pautas de accesibilidad para personas con discapacidad visual.
Dia	Herramienta en línea la cual permite hacer diagramas de cualquier tipo.

Lighthouse	Herramienta automatizada y de código abierto que nos ofrece la posibilidad de medir la calidad de una determinada página web, en cuando a rendimiento, accesibilidad, etc.
Responsively App	Herramienta que facilita el desarrollo y la depuración de sitios web responsivos al permitir a los desarrolladores visualizar y probar la apariencia de un sitio en diferentes dispositivos y tamaños de pantalla simultáneamente
SortSite	Rastreador web que escanea sitios web completos en busca de problemas de calidad, incluidos la accesibilidad, la compatibilidad del navegador, los enlaces rotos, el cumplimiento legal, la optimización de búsqueda, la usabilidad y el cumplimiento de los estándares web.
Postman	Herramienta que permite realizar peticiones de una manera simple para testear APIs de tipo REST propias o de terceros.

Tabla 4. Recopilación de recursos/herramientas y uso dado.

5.1 Objetivos del proyecto.

Una vez comentados y analizados tanto los requerimientos como las herramientas, pasamos a realizar un breve análisis de los objetivos principales del proyecto que establecimos en nuestra propuesta inicial y de cómo los hemos abordado. A continuación, se mencionan dichos objetivos de forma numerada, para que a lo largo de este trabajo se pueda hacer mención a ellos fácilmente, indicando en qué partes del proyecto se llevan a cabo acciones destinadas a cumplirlos.

Objetivo 1. Gestionar y crear una base de datos de muestra: Hemos conseguido generar una base de datos relacional en MariaDB que representa todas las entidades y sus relaciones. Si bien nos hemos encontrado la dificultad de poblarla con datos. En intento de automatizar este proceso, realizamos una investigación sobre web scraping¹ pero rápidamente lo descartamos por la dificultad en la extracción de datos debido a la maquetación de las páginas web de referencia.

Objetivo 2. Crear una interfaz amigable y fácil de usar por todos los usuarios: Creemos que hemos sido fieles a este objetivo, y hemos conseguido una interfaz bastante intuitiva. Los elementos y el diseño son simples pero muy coloridos y hace que la experiencia de usuario sea óptima.

¹ Web scraping o raspado web es una técnica utilizada mediante programas de software para extraer información de sitios web a través de sus marcas HTML.

Objetivo 3. Realizar un diseño adaptativo (responsive y mobile first): Desde la realización de los prototipos nos centramos en que el diseño de la web fuera adaptable a cualquier dispositivo, no sólo móvil, sino tablet.

Objetivo 4. Añadir funcionalidad a una aplicación web: Hemos conseguido implementar la funcionalidad básica para una primera versión de la web, aunque también es cierto que, por falta de tiempo, muchas de las funcionalidades que inicialmente se plantearon se van a incorporar como futuras mejoras.

Objetivo 5. Cumplir con la normativa vigente: Sobre todo en materia de accesibilidad web, se ha desarrollado la aplicación para obtener el criterio AA según las pautas de accesibilidad web de la WAGC. Ya que nuestro cliente se trata de un organismo público debe cumplir la legislación establecida por el Real Decreto 1112/2018, de 7 de septiembre, sobre accesibilidad de los sitios web y aplicaciones para dispositivos móviles del sector público.

Objetivo 6. Incluir alguna tecnología no vista en el curso: A pesar de haber usado tecnologías ya vistas en clase, si consideramos que hemos cumplido con este objetivo, ya que hemos incorporado e investigado nuevas funciones que mejoran y complementan lo aprendido.

Objetivo 7. Desplegar la aplicación en un entorno de producción: Hemos conseguido desplegar tanto la parte de entorno cliente como la parte de entorno servidor en distintas plataformas de las cuales hemos investigado y aplicado nuevos conocimientos.

5.2 Licencia.

Para finalizar con este apartado, queremos dedicar un pequeño espacio a hablar sobre la licencia bajo la que se enmarca este proyecto: **Creative Commons Reconocimiento-CompartirIgual (CC BY-SA)**. Haciendo uso de esta licencia se permite que el proyecto se comparta, copie, distribuya, exhiba, e incluso que se realicen obras derivadas, siempre y cuando se nos de una correcta atribución y se compartan bajo los mismos términos. Todo esto permite que la obra llegue a una audiencia más amplia, lo que da pie a que aumente su visibilidad e impacto y, a su vez, limita al resto de usuarios para hacer lo que quieran con esta obra, ya que es necesaria tanto la atribución como el hecho de que se comparta bajo los mismos términos. Esto último nos permite tener cierto control sobre cómo se utiliza y comparte la obra.

6 Diseño y resolución del trabajo realizado.

Una vez que los objetivos y requerimientos están bien definidos y la metodología está clara, es hora de pasar a la fase de diseño.

6.1 Diseño de la interfaz.

El primer paso dentro de este apartado consistió en el diseño de la interfaz, empezando por la elección de la paleta de colores (Figura 4, Tabla 4) y tipografía (Figura 5) que serán los protagonistas en el posterior diseño de prototipos en Figma.

Respecto a los colores, elegimos una paleta con un azul muy oscuro (#073B4C), que usamos para todos los textos de la aplicación, y como color principal, para los botones más importantes de la aplicación, un amarillo bastante llamativo (#FFD166), que combina muy bien con el azul del texto, ya que se encuentra casi en el otro extremo del círculo cromático. Además, como colores intermedios, probamos varios colores que combinaban bien con los dos principales, y al final elegimos un verde claro (#06D6A0) y otro azul (#117286), bastante más claro que el del texto, pero que contrasta bien con los distintos tonos claros de la aplicación. Estos dos colores, los usamos sobre todo para “hovers” de botones y enlaces, así como para otros pequeños detalles de la aplicación. Además, como colores de fondo, utilizamos el blanco (#FFFFFF) y un gris muy claro (#EEEEEE).

A continuación, se muestra la paleta de colores, así como una recopilación de los contrastes de las combinaciones más relevantes.

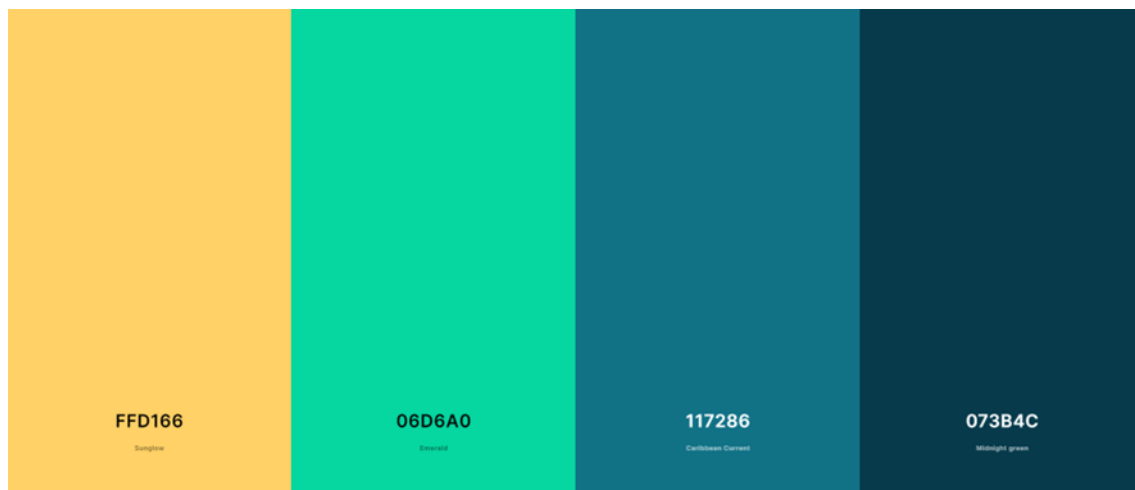


Figura 9. Paleta de colores elegida para la web.

Color del texto	Color de fondo	Contraste
#073B4C	#FFFFFF	12,08 (AAA)
#073B4C	#EEEEEE	10,41 (AAA)
#073B4C	#FFD166	8,38 (AAA)
#073B4C	#06D6A0	6,40 (AA)
#117286	#FFFFFF	5,56 (AA)
#117286	#FFE8AF	4,61 (AA)
#FFFFFF	#073B4C	12,08 (AAA)
#073B4C	#AFF2E0	9,55 (AAA)

Tabla 5. Contrastes de las combinaciones de colores más comunes (utilizando la herramienta Color Contrast Checker, de Coolers).

Centrándonos ahora en la tipografía, hemos elegido una única fuente para toda la web: Urbanist, diseñada por Corey Hu, disponible en [Google Fonts](#). Es una fuente sans-serif, que destaca por su diseño simple y claro, lo que hace que todo el texto sea muy legible, y otorga a la web un aspecto bastante moderno y elegante. El utilizar una sola tipografía da un aspecto de unidad a la web, y para diferenciar los textos que son más o menos importantes jugamos con distintos tamaños y grosores de fuente.



Figura 10. Fuente utilizada en la web.

Uno de los aspectos más relevantes en el proceso de diseño de nuestra web ha sido, sin duda, el diseño de prototipos, para lo que usamos Figma, una herramienta de prototipado y diseño de interfaces. Aunque todos los prototipos (tanto en su versión para escritorio como para móvil) se encuentran disponibles en los enlaces del final de este apartado, mostramos aquí los diseños de las pantallas y componentes que consideramos más importantes. Cabe destacar que hemos creado nuestro diseño desde cero, sin hacer uso de plantillas de terceros. Por esto, lo primero que hicimos fue tomar referencias de las distintas páginas de transportes que conocíamos (como thetrainline.com, omio.es o tmpmurcia.es), e inspirarnos en los elementos de cada una que más nos gustaron. Además, a lo largo del desarrollo Front end nos hemos encontrado con diversas necesidades y cambios de opinión, por lo que mostramos tanto el diseño inicial que se pretendía obtener como el diseño que finalmente se implementó, acompañado de breves comentarios.

- **Botones.** Al empezar el prototipado, en toda la aplicación teníamos una sola versión de los botones, que eran todos con fondo amarillo y texto azul oscuro, y en el hover el fondo pasaba a verde. No obstante, nos dimos cuenta de que ese botón estaba muy bien como botón primario, para utilizarlo en las acciones principales de cada pantalla, pero era demasiado llamativo como para utilizarlo en otras acciones más secundarias, así que hicimos dos versiones más de ese botón, una con fondo blanco, y borde de los colores de fondo del botón primario, y otro directamente sin borde ni fondo, y que con el hover el texto se subraye (estilo enlace).

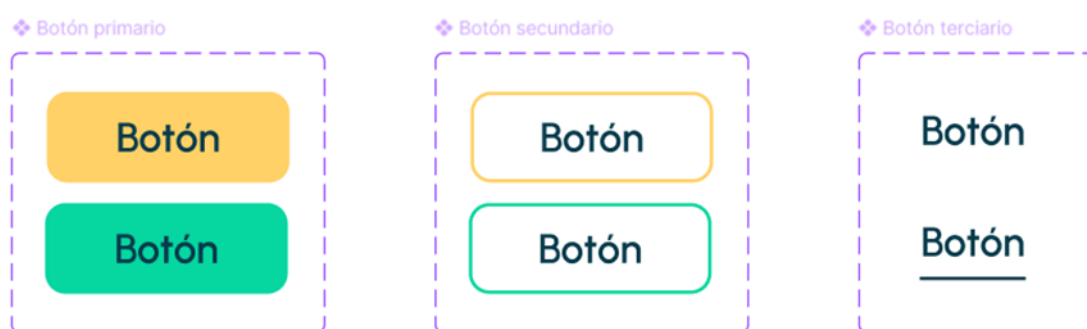
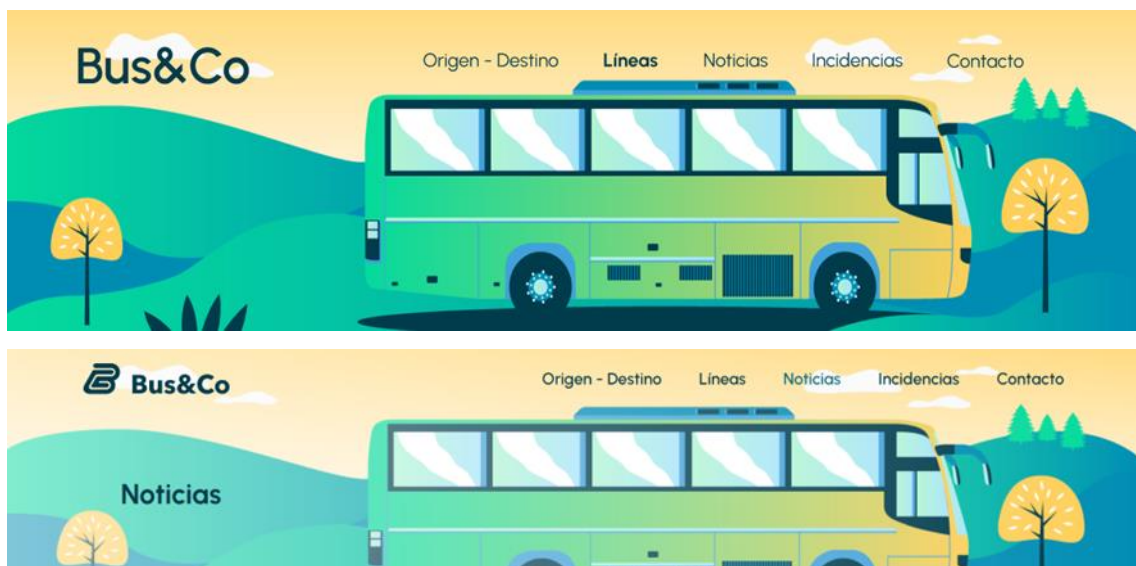


Figura 11. Botones en sus diferentes versiones.

- **Cabecera.** Para el diseño de la cabecera, nos inspiramos mucho en la de Omio, y generamos una ilustración relacionada con la temática de nuestra web, y que usara la paleta de colores que habíamos elegido. El “prompt” inicial utilizado fue el siguiente: “Genérame una ilustración de 1500px x 500px que muestre un autobús utilizando la siguiente paleta de colores: #FFD166, #06D6A0, #117826, #073B4C”. No obstante, tras este “prompt” inicial, hubo que hacer muchas correcciones, ya que los primeros resultados no se acercaban mucho a lo que queríamos. Como podemos observar en las imágenes de abajo, en el prototipo inicial, la imagen se ajustaba bien a la cabecera, pero al final, cuando desarrollamos los estilos en la aplicación, nos dimos cuenta de que el alto de ésta era demasiado grande, ya que ocupaba más de la mitad de la pantalla, por lo que tuvimos que reducir sus dimensiones, por lo que en muchos tamaños la imagen aparece recortada por abajo.

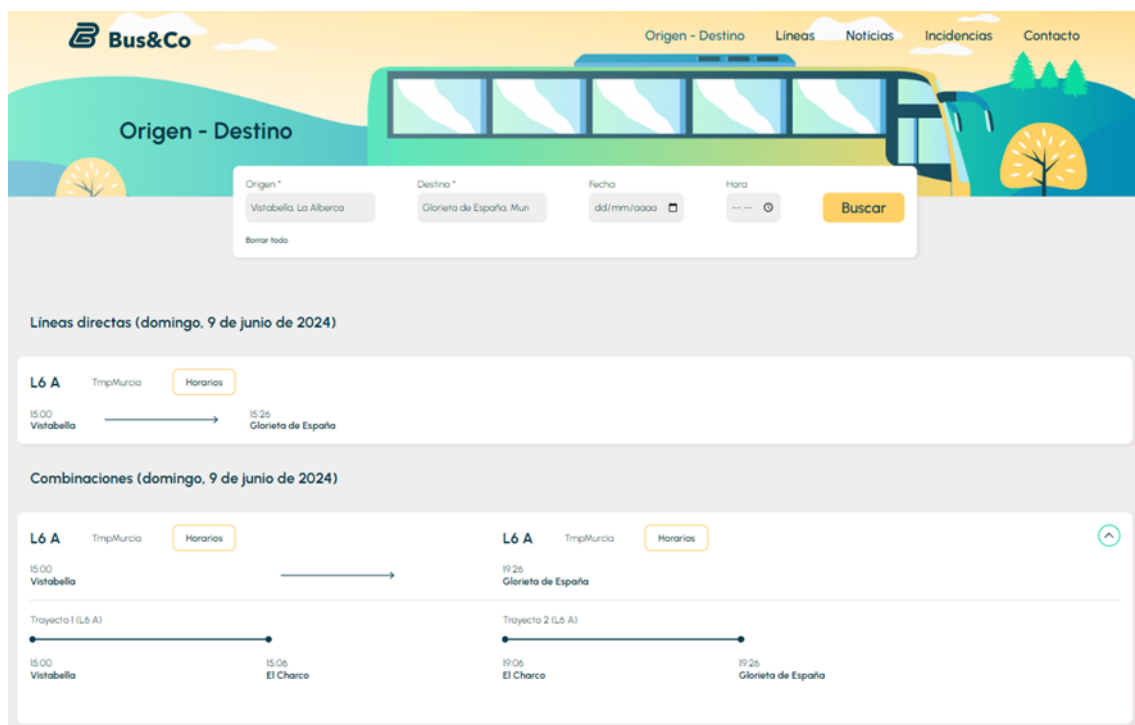
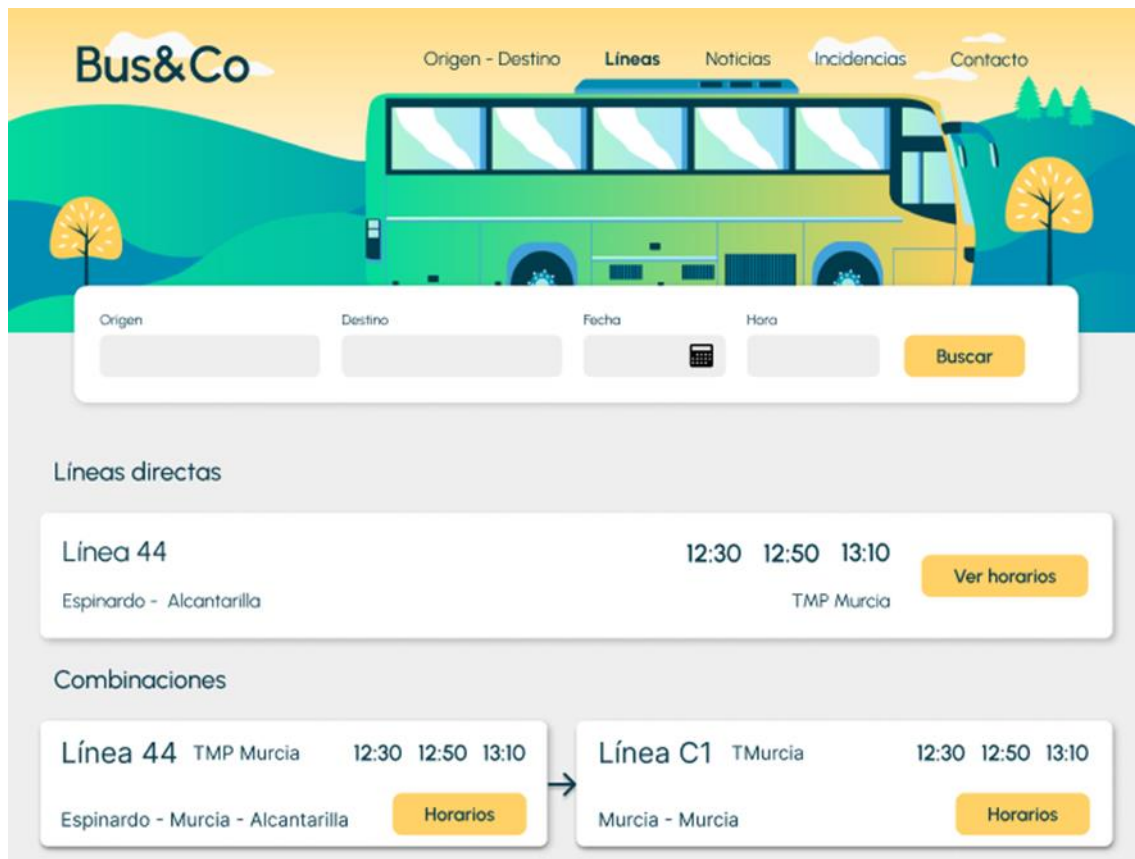


Figuras 12 y 13. Diseño en Figma y resultado final – Cabecera (Versión de escritorio).

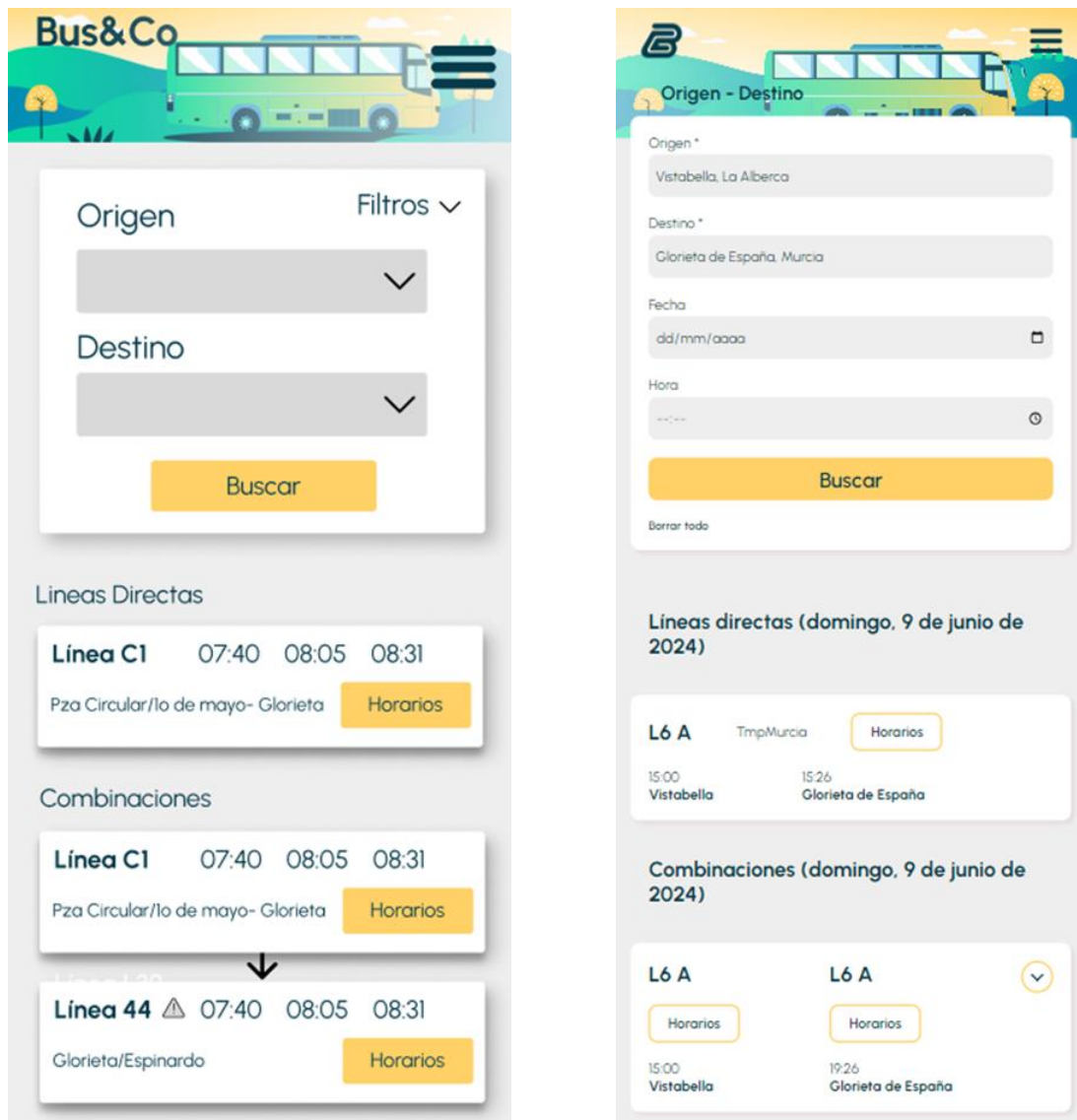


Figuras 14 y 15. Diseño en Figma y resultado final – Cabecera (Versión de móvil).

- **Origen - Destino.** La pantalla de origen y destino ha sido, sin duda alguna, la que más cambios y dudas ha generado en cuanto a funcionalidad y diseño, por lo que la versión inicial de Figma difiere bastante del resultado final. En esta pantalla tenemos tres elementos principales: el buscador, la tarjeta de líneas directas, y la tarjeta de combinaciones. El diseño del buscado vuelve a estar inspirado en el de la aplicación Omio, de la que tomamos su forma, y su disposición a medio camino entre la cabecera y el contenido principal de la página. Respecto a las tarjetas de línea directa y combinación, la primera versión mostraba los siguientes horarios desde la hora seleccionada (o en su defecto, la hora actual), pero nos dimos cuenta de que eso era bastante confuso, ya que no había información de las paradas de origen y de destino, así como la parada de transbordo en el caso de la combinación. Por eso, al final se optó por mostrar el siguiente autobús, pero detallando los horarios de paso por cada parada.



Figuras 16 y 17. Diseño en Figma y resultado final – Origen y Destino (versión de escritorio).



Figuras 18 y 19. Diseño en Figma y resultado final – Origen y Destino (versión de móvil).

- **Línea detallada.** Esta pantalla es de las que más elementos visuales tiene, y por lo tanto una de las que más complicado fue hacer el diseño inicial, optamos por dividirla en una cabecera, que mostrara la información de la línea, la sublínea, la dirección, la empresa y el mapa con el recorrido, y debajo una tabla con las distintas paradas, pudiendo acceder a los horarios de cada parada. Para esta pantalla nos inspiramos en cómo tenía estructurada la información la web de TmpMurcia, que, aunque quizá el diseño en cuanto a colores y proporciones no sea demasiado atractivo, si que está bastante bien organizado en cuanto a estructura.

Línea 44 A

Sublínea

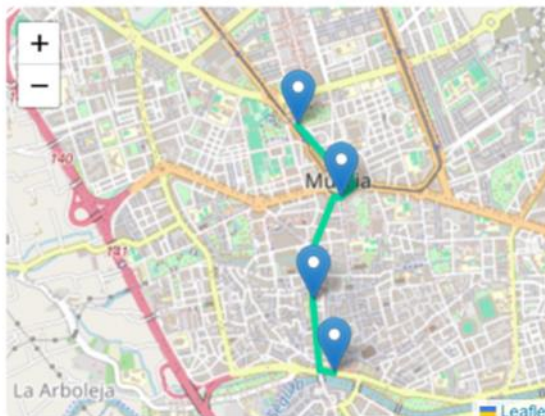


TMP Murcia



Espinardo - Alcantarilla

Cambio de dirección



Población	Parada	Ver horarios	Enlace con líneas
Espinardo	Instituto	Horarios	
Espinardo	Iglesia	Horarios	39
Murcia	Carrefour Zaraiche	Horarios	29, 39
Murcia	Plaza Camachos	Horarios	1, 6, 26, 28, 29, 39, 62, 91
Alcantarilla	La Palmera	Horarios	13

[Origen - Destino](#)
[Lineas](#)
[Noticias](#)
[Incidencias](#)
[Contacto](#)

Líneas

L6 A

Sublínea

L6 A

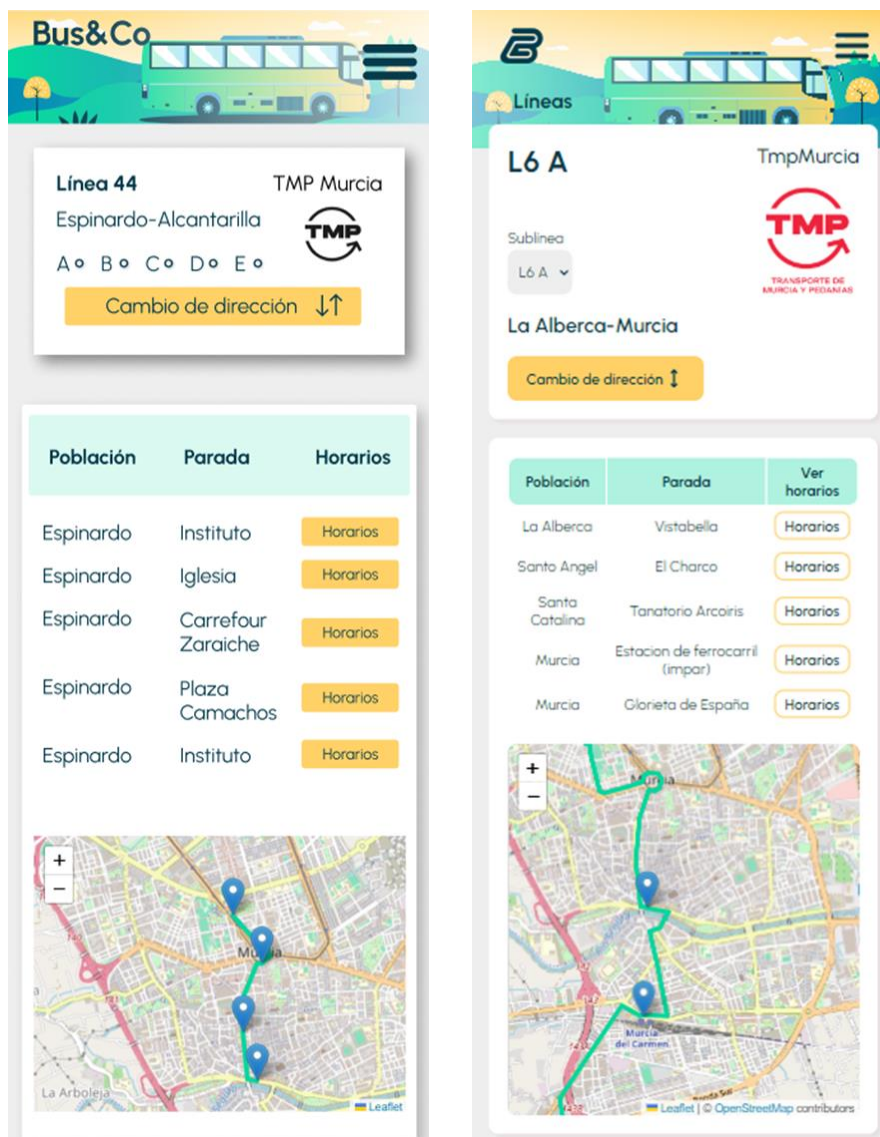
La Alberca-Murcia

Cambio de dirección

TmpMurcia

Población	Parada	Ver horarios	Enlace con líneas
La Alberca	Vistabella	Horarios	
Santo Ángel	El Charco	Horarios	
Santa Catalina	Tanatorio Arcoiris	Horarios	
Murcia	Estacion de ferrocarril (impar)	Horarios	
Murcia	Glorieta de España	Horarios	L1

Figuras 20 y 21. Diseño en Figma y resultado final – Línea detallada (versión de escritorio).



Figuras 22 y 23. Diseño en Figma y resultado final – Línea detallada (versión de móvil).

6.1.1 Logotipo.

Para la creación del logo de nuestra aplicación hicimos uso de [design.com](https://www.design.com), una herramienta generadora de logotipos, tarjetas de negocio, post en redes sociales y mucho más. Para generar el logotipo tuvimos que introducir el nombre de nuestro “negocio”, las palabras clave (transporte y autobús), elegir el estilo del logo, y la herramienta nos proporcionó distintas opciones. Elegimos la que creíamos que más se adecuaba al estilo de nuestra interfaz. Después cambiamos el color de fondo al azul oscuro de nuestra paleta de colores, y añadimos una versión con el nombre de la aplicación a la derecha del logo, usando la tipografía que utilizamos en toda la web. Por último, hicimos una versión del logo en blanco, para utilizarla en el pie de página, cuyo fondo es de color azul oscuro.



Figura 24. Logo generado para Bus&Co.



Figura 25. Logo con el título añadido.

6.2.2 Accesibilidad y usabilidad.

Aunque ya se han comentado ciertas cuestiones relacionadas con el tema que nos ocupa en este punto como, por ejemplo, el contraste entre colores o la existencia de ciertos elementos, queríamos dedicar un subapartado de la memoria a este tema, ya que para nosotros ha sido importante que la web esté a disposición de cualquier tipo de usuario y sea usable desde cualquier dispositivo (Objetivos 2 y 5).

Comenzando con la accesibilidad, hemos optado por presentar este punto haciendo referencia a los aspectos más destacables de nuestra web en base a los 4 principios de accesibilidad de WCAG 2.1.

- **Perceptible.**
 - **Alternativas textuales al contenido no textual.** Hemos añadido etiquetas *alt* tanto al mapa como a las imágenes de la web.
 - **Utilizar colores contrastantes.** Como hemos visto anteriormente, se usan contrastes de colores óptimos para leer el texto correctamente.
 - **Información y Relaciones:** Se ha estructurado la web con el máximo uso posible de etiquetas semánticas, para que la información y las relaciones de los elementos de la página se representen de manera

que los usuarios invidentes puedan relacionar estos elementos entre sí gracias a los lectores de pantalla.

- **Operable.**
 - **Acceso a las funcionalidades mediante el teclado.** Nos hemos asegurado de que los botones e inputs sean accesibles con, al menos, dos métodos de entrada: ratón y teclado.
- **Comprensible.**
 - **Texto legible y comprensible.** Hemos elegido una tipografía legible y el texto de la web está expresado de forma clara y sencilla.
 - **Organización del contenido de manera lógica y estructurada.**
- **Robusta.**
 - **Poner atención a la compatibilidad con herramientas que permiten una mayor accesibilidad.** Hemos intentado maximizar la legibilidad de la aplicación web haciendo uso de lectores de pantalla. Para ello, hemos procurado que la mayoría de los aspectos de la web sean tabulables haciendo uso de la propiedad *tab-index*, así como, de las etiquetas *aria-label* para que los usuarios que usan estos lectores puedan, por ejemplo, saber para qué sirve cada botón.

Una herramienta que nos ha sido fundamental para el desarrollo en términos de accesibilidad ha sido la herramienta Lighthouse de Chrome, que permite ver fácilmente errores o alertas en aspectos como: contraste entre colores, orden de tabulación, uso de etiquetas semánticas, etiquetas ARIA, etc. A continuación, se muestran algunas imágenes demostrativas de la herramienta:

DevTools - localhost:4200/origen-destino

Elementos Consola Fuentes Red Rendimiento Lighthouse >> 1

+ (nuevo informe)

Generar un informe de Lighthouse

Analizar carga de la página

Modo [Más información](#)

☒ Navegación (predeterminada)
☐ Tiempo
☐ Instante

Dispositivo

☐ Móvil
☒ Ordenador

Categorías

☒ Rendimiento
☒ Accesibilidad
☒ Prácticas recomendadas
☒ SEO
☒ Aplicación web progresiva

100

Accesibilidad

Estas comprobaciones permiten identificar oportunidades para [mejorar la accesibilidad de tu aplicación web](#). La detección automática solo puede detectar un subconjunto de problemas y no garantiza la accesibilidad de tu aplicación web, por lo que también te recomendamos que hagas [pruebas manuales](#).

ELEMENTOS ADICIONALES QUE SE DEBEN COMPROBAR MANUALMENTE (10) Ocultar

- ☐ Interactive controls are keyboard focusable
- ☐ Interactive elements indicate their purpose and state
- ☐ The page has a logical tab order
- ☐ Visual order on the page follows DOM order
- ☐ User focus is not accidentally trapped in a region
- ☐ The user's focus is directed to new content added to the page
- ☐ HTML5 landmark elements are used to improve navigation

AUDITORÍAS APROBADAS (15)		Ocultar
●	[aria-hidden="true"] no se encuentra en el documento <body>	▼
●	Los botones tienen nombres accesibles	▼
●	Los elementos de imagen tienen atributos [alt]	▼
●	[user-scalable="no"] no se utiliza en el elemento <meta name="viewport"> y el valor del atributo [maximum-scale] no es inferior a 5.	▼
●	Los elementos [aria-hidden="true"] no contienen ningún elemento inferior seleccionable	▼
●	Los colores de fondo y de primer plano tienen una relación de contraste adecuada	▼
●	El documento tiene un elemento <title>	▼
●	El elemento <html> tiene un atributo [lang]	▼
●	El atributo [lang] del elemento <html> tiene un valor válido	▼

Figuras 26 a 28. Imágenes demostrativas del uso de Lighthouse.

En cuanto a la usabilidad se han tenido en cuenta los siguientes aspectos:

- **Diseño responsive.** Desde el comienzo del diseño de la interfaz, seguimos un enfoque *mobile-first*, es decir, priorizar el diseño y desarrollo para dispositivos móviles, antes que el de escritorio. Todas las pantallas se pueden ver y utilizar perfectamente en dispositivos con un ancho mínimo de 300px, y para lograr esto hemos usado las propiedades “display: flex” y “display: grid”, así como las llamadas “media queries” de CSS. Para comprobar los diseños en cada tamaño, hemos hecho uso de la barra herramienta de dispositivo, dentro de las herramientas de desarrollador de Chrome, que permite simular la vista de la aplicación en cualquier tamaño.
- **Orden y coherencia.** El contenido se muestra en un orden coherente. Esto, junto con un diseño simple, contribuye a hacer la web más usable para el usuario.
- **Página 404.** Para ello hemos creado una ruta en el archivo router link para que cualquier error de enrutamiento a nuestro programa aparezca un error indicando que la ruta no es correcta.
- **Título.** En todo momento, tanto en el título de la pestaña del navegador, como en la cabecera, aparece el título de la pantalla en la que nos encontramos, para que así el usuario esté orientado en todo momento.

- **Logo vinculado a la página de inicio.** Hemos creado un link en la imagen para que retorne al apartado de inicio de nuestra página (en este caso, la pantalla de origen y destino). La función que buscamos con esto es poder volver al inicio de una manera fácil y rápida.

6.2 Desarrollo web.

6.3.1 Esquema de la base de datos

Cuando planteamos el primer modelo de datos necesario para la nuestra aplicación, nos dimos cuenta de que necesitábamos una base de datos relacional, debido a las múltiples relaciones de distintos tipos entre las distintas entidades, por lo que optamos por utilizar MariaDB como gestor de bases de datos, y Doctrine como ORM (Object Relational Mapping), ya que es el que utiliza Symfony por defecto. Desde este punto de vista inicial, se planificó un modelo relacional cuya estructura se representa en la siguiente figura:

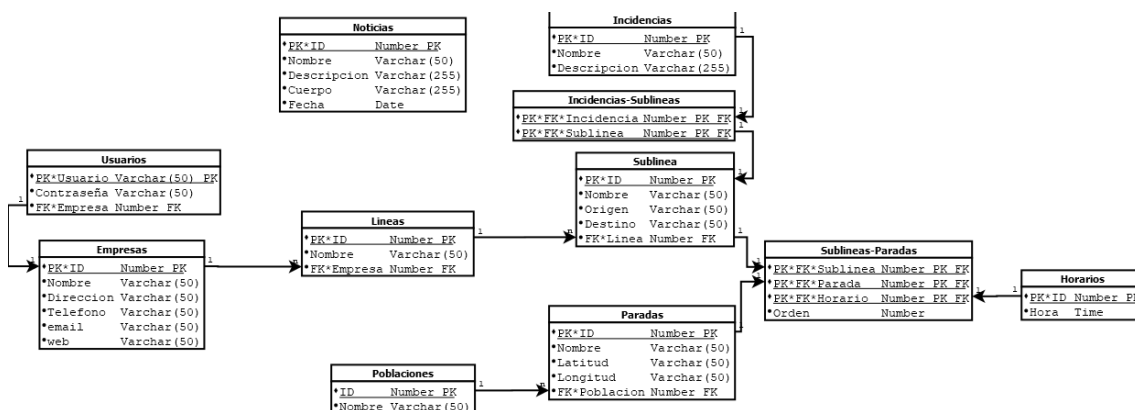


Figura 29. Esquema entidad-relación inicial.

Gracias al desarrollo de este diagrama, pudimos visualizar mucho mejor el funcionamiento interno de la aplicación. No obstante, a medida que avanzamos en el desarrollo del proyecto, tuvimos que modificar varias veces el esquema inicial, debido a las diferentes necesidades que fueron surgiendo en los distintos puntos del desarrollo. En la siguiente se muestra la versión definitiva del modelo relacional empleado en nuestra base de datos:

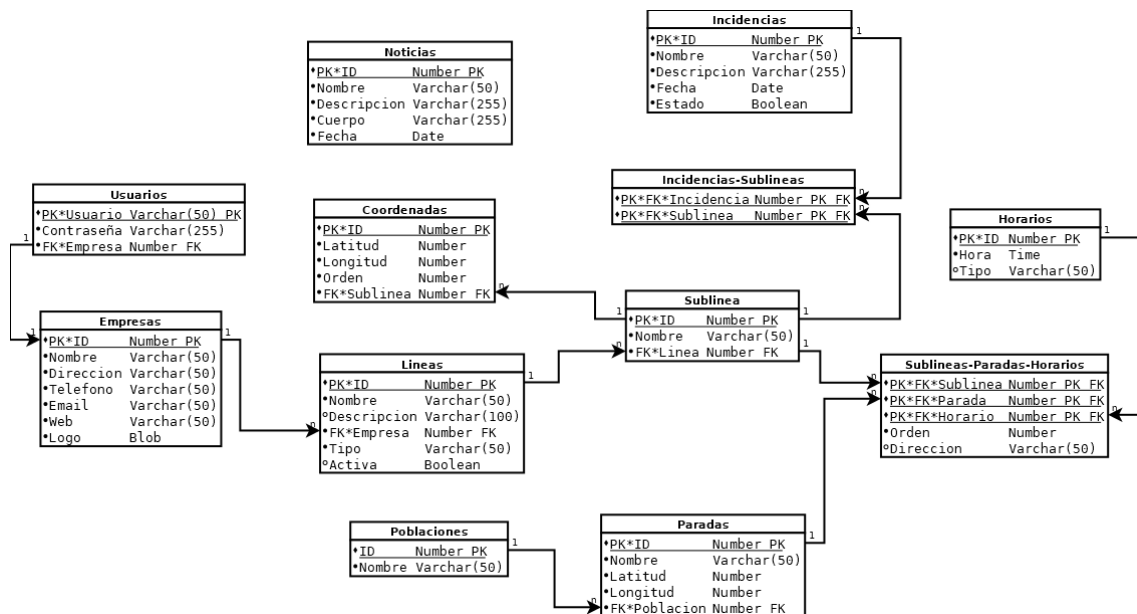


Figura 30. Esquema entidad-relación final.

6.3.2 Desarrollo lado servidor

El desarrollo en la parte servidor se ha llevado a cabo con Symfony, debido a que es un framework del que teníamos conocimientos todos los miembros del grupo y por tanto, era más fácil aportar ideas o soluciones a posibles problemas de manera conjunta. Además, Symfony es un framework que aporta un modelo robusto de desarrollo y tiene versiones muy estables con soporte continuado en el tiempo y alta compatibilidad con versiones anteriores.

En nuestro caso, al no necesitar Vistas, creamos la aplicación en formato API REST incorporando el bundle FOSRest para la funcionalidad Api y el NelmioCors para incorporar la información requerida para las cabeceras en las peticiones HTTP.

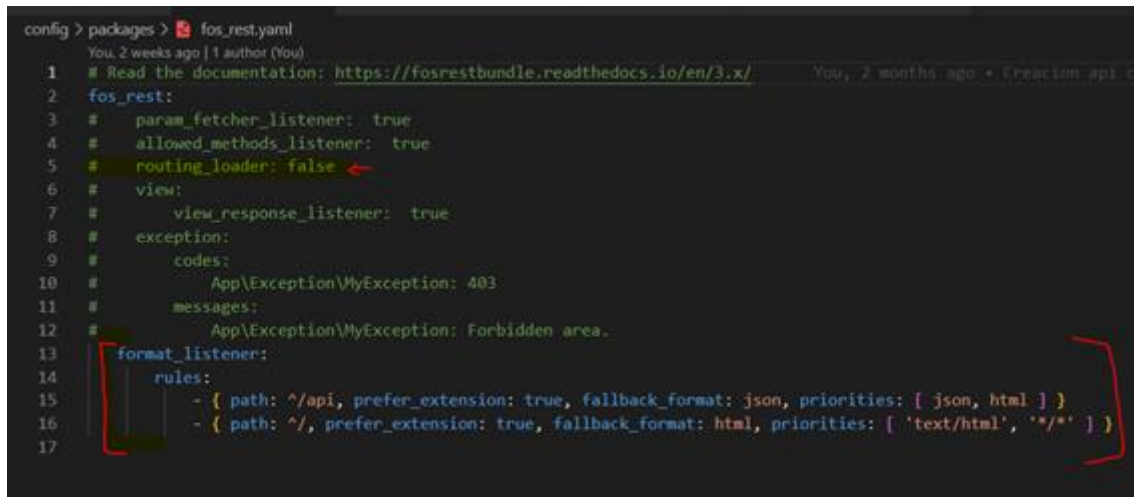
```

1  <?php
2
3  return [
4      Symfony\Bundle\FrameworkBundle\FrameworkBundle::class => ['all' => true],
5      JMS\SerializerBundle\JMSSerializerBundle::class => ['all' => true],
6      FOS\RestBundle\FOSRestBundle::class => ['all' => true],
7      Symfony\Bundle\MakerBundle\MakerBundle::class => ['dev' => true],
8      Doctrine\Bundle\DoctrineBundle\DoctrineBundle::class => ['all' => true],
9      Doctrine\Bundle\MigrationsBundle\Doctrine\MigrationsBundle::class => ['all' => true],
10     Nelmio\CorsBundle\NelmioCorsBundle::class => ['all' => true],
11     Symfony\Bundle\SecurityBundle\SecurityBundle::class => ['all' => true],
12     Lexik\Bundle\JWTAuthenticationBundle\JWTAuthenticationBundle::class => ['all' => true],
13 ];
14

```

Figura 31. Captura del fichero bundles.php del proyecto de Symfony

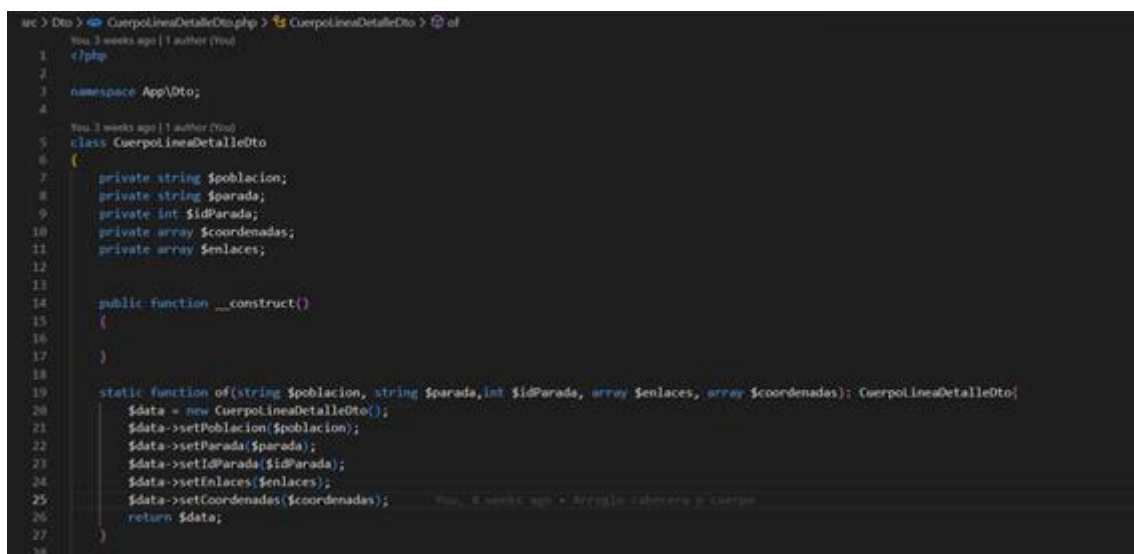
Además, hemos configurado el fichero fos_rest.yaml para que al devolver un valor desde una acción, se serialice a JSON de forma automática añadiendo los distintos path en el “format listener”. Comentando la orden “routing_loader: false” conseguimos que el enrutamiento se haga de forma también automática en función de los nombres de los métodos de acción de los controladores:



```
config > packages > fos_rest.yaml
You, 2 weeks ago | 1 author (You)
1 # Read the documentation: https://fosrestbundle.readthedocs.io/en/3.x/
2 fos_rest:
3 #   param_fetcher_listener: true
4 #   allowed_methods_listener: true
5 #   routing_loader: false
6 #   view:
7 #     view_response_listener: true
8 #   exception:
9 #     codes:
10 #       App\Exception\MyException: 403
11 #     messages:
12 #       App\Exception\MyException: Forbidden area.
13 format_listener:
14   rules:
15     - { path: ^/api, prefer_extension: true, fallback_format: json, priorities: [ json, html ] }
16     - { path: ^/, prefer_extension: true, fallback_format: html, priorities: [ 'text/html', '*' ] }
```

Figura 32. Captura del fichero fos_rest.yaml del proyecto de Symfony

Hemos incorporado el uso de DTO’s (Data Transfer Objects) con el objetivo de controlar la información transmitida entre cliente y servidor, además si el modelo de datos cambia, el cliente no se verá afectado porque seguirá recibiendo el mismo DTO. También aumenta la seguridad ya proporcionada por el ORM de Symfony, Doctrine, al incorporar una capa más entre el modelo y el resto de las capas.



```
src > Dto > CuerpoLineaDetalleOto.php > CuerpoLineaDetalleOto > of
You, 2 weeks ago | 1 author (You)
1 <?php
2
3 namespace App\Dto;
4
5 You, 3 weeks ago | 1 author (You)
6 class CuerpoLineaDetalleOto
7 {
8     private string $poblacion;
9     private string $parada;
10    private int $idParada;
11    private array $coordenadas;
12    private array $enlaces;
13
14    public function __construct()
15    {
16    }
17
18
19    static function of(string $poblacion, string $parada, int $idParada, array $enlaces, array $coordenadas): CuerpoLineaDetalleOto
20    {
21        $data = new CuerpoLineaDetalleOto();
22        $data->setPoblacion($poblacion);
23        $data->setParada($parada);
24        $data->setIdParada($idParada);
25        $data->setEnlaces($enlaces);
26        $data->setCoordenadas($coordenadas);
27        return $data;
28    }
29 }
```

Figura 33. Captura de la clase DTO “CuerpoLinea”, atributos y función de mapeo

Para poder trabajar estos objetos se ha implementado el bundle JMSSerializer y la elaboración de varios métodos en la clase creada “TransformDto” dentro del

directorio “Utils” del proyecto. Éstos se llaman en cada método del controlador para transformar los objetos a JSON y viceversa.

```
11 class TransformDTO {
12     public function __construct() {}
13
14     public function encoderDto (array $dtoList){
15         $encoders = [new JsonEncoder()];
16         $normalizers = [new ObjectNormalizer()];
17         $serializer = new Serializer($normalizers, $encoders);
18
19         $jsonContent = json_decode($serializer->serialize($dtoList, 'json'));
20         return $jsonContent;
21     }
22
23     public function decoderDtoObject ($dtoList){
24         $decoders = [new JsonEncoder()];
25         $normalizers = [new ObjectNormalizer()];
26         $serializer = new Serializer($normalizers, $decoders);
27
28         $jsonContent = json_encode($serializer->serialize($dtoList, 'json'));
29         return $jsonContent;
30     }
31
32     public function encoderDtoObject ($dtoList){
33         $encoders = [new JsonEncoder()];
34         $normalizers = [new ObjectNormalizer()];
35         $serializer = new Serializer($normalizers, $encoders);
36
37         $jsonContent = json_decode($serializer->serialize($dtoList, 'json'));
38         return $jsonContent;
39     }
40
41     public function decoderDto (array $dtoList){
42         $decoders = [new JsonEncoder()];
43         $normalizers = [new ObjectNormalizer()];
44         $serializer = new Serializer($normalizers, $decoders);
45
46         $jsonContent = json_encode($serializer->serialize($dtoList, 'json'));
47         return $jsonContent;
48     }
49 }
```

Figura 34. Métodos de transformación de DTOs que son reutilizados en los Controladores

Como parte destacable del código, nos gustaría remarcar el método del Controlador de Usuario llamado “origenDestino” por la complejidad que supuso a la hora de su desarrollo. Se han incluido comentarios en el propio código para una mayor comprensión. Dicho método recibe por parámetros dos String correspondientes a nombres de paradas, y debe devolver un objeto DTO que incluye a su vez otros objetos DTOs anidados. El resultado es un JSON en el que por cada línea activa se obtienen las sublíneas asociadas y sus datos, incluyendo las direcciones. Y a su vez, por cada dirección, se obtiene el listado de paradas con población y horario asociados.

Además de hacer uso de varias entidades y métodos creados manualmente en los distintos repositorios, se debían generar varios bucles anidados, con comprobaciones y condicionales para enviar la información necesaria al entorno

cliente y que además estuviera en la estructura correcta. Este método se usa para la interfaz de búsqueda de líneas por origen y destino y cuya funcionalidad total se completa con la lógica creada en el entorno cliente, mostrando los itinerarios directos y combinados junto al horario según el tipo (laboral, sábado, festivo) en función de las paradas, fecha y hora introducidas por el usuario.

```

95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
class UsuarioController extends AbstractController
{
    public function origenDestino(Request $request): JsonResponse
    {
        $pOrigen = $this->em->getRepository(Parada::class)->find($origen);
        $pDestino = $this->em->getRepository(Parada::class)->find($destino);

        //Recorremos array de sublineas que incluyen las paradas de Origen y/o Destino
        $sublineasBusqueda = $this->em->getRepository(Sublinea::class)->findSublineasByParadas($pOrigen, $pDestino);
        if($sublineasBusqueda){
            foreach($sublineasBusqueda as $sublineaBusqueda){
                //Recuperamos las direcciones de la sublinea
                $direccionesSublinea = $this->em->getRepository(SublineasParadasHorarios::class)->findDireccionesBySublinea($sublineaBusqueda->getId());

                //Recorremos dichas direcciones y recuperamos las paradas de cada direccion
                foreach($direccionesSublinea as $direccionSublinea){
                    if($direccionSublinea->count($direccionSublinea)>0){
                        $paradasDireccion = $this->em->getRepository(Parada::class)->findParadasByDireccion($direccionSublinea->getDireccion());

                        //Recorremos las paradas y recuperamos los horarios y el orden
                        foreach($paradasDireccion as $paradaDireccion){
                            if($paradaDireccion){
                                $horariosParada = $this->em->getRepository(Horario::class)->findHorariosByParadaSublineaDireccion(
                                    ($sublineaBusqueda->getId(), $paradaDireccion->getId(), $direccionSublinea);
                                $ordenParada = $this->em->getRepository(SublineasParadasHorarios::class)->findOrdenByParadaDireccion(
                                    ($paradaDireccion->getId(), $direccionSublinea->getDireccion());
                                if(count($ordenParada) == 0){
                                    return $this->json(["error" => "No se han encontrado Datos"], 404);
                                }else{
                                    //Generamos el dto con Datos de las Paradas
                                    $dtoParada = COParadasDto::of($paradaDireccion->getId(),
                                        $paradaDireccion->getNombre(),
                                        $paradaDireccion->getPoblacion()->getNombre(),
                                        $ordenParada[0]['orden'],
                                        $horariosParada);
                                    array_push($dtoParadas, $dtoParada);
                                }
                            }
                        }
                    }
                }
            }
        }
        $horariosParada = [];
    }
}

```

Figura 35. Captura de fragmento del método origenDestino del controlador Usuario.

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
{
  "sublinea": 5,
  "linea": "L3",
  "empresa": "TropMurcia",
  "idSublinea": 5,
  "sublinea": "A",
  "direcciones": [
    {
      "direccion": "San Gines-Murcia",
      "paradas": [
        {
          "idParada": 5,
          "parada": "Escuela de España",
          "poblacion": "Murcia",
          "orden": 1,
          "horario": [
            {
              "id": 76,
              "hora": "07:35",
              "tipo": "Laboral"
            },
            {
              "id": 77,
              "hora": "08:05",
              "tipo": "Laboral"
            },
            {
              "id": 78,
              "hora": "08:35",
              "tipo": "Laboral"
            },
            {
              "id": 80,
              "hora": "08:35",
              "tipo": "Laboral"
            },
            {
              "id": 85,

```

Figura 36. Captura de la respuesta de Postman a la llamada al método origenDestino

En muchas ocasiones hemos necesitado crear consultas personalizadas en los repositorios con los que extraer la información necesaria y evitar así una excesiva lógica en el controlador. Para ello, hemos utilizado Doctrine Query Language, un lenguaje de consulta específico del ORM (Object Relational Mapping o Mapeo Objeto-Relacional) para realizar consultas sobre nuestro modelo de entidades.

```
public function findHorariosByParadaSublineaDireccion($idSublinea,$idParada, $direccion)
{
    $query = $this->em->createQuery("SELECT ho.id, ho.hora, ho.tipo FROM App\Entity\Horario ho
    JOIN ho.sublineasParadasHorarios sub
    JOIN sub.parada pa
    JOIN sub.sublinea sl
    WHERE sl.id = ?1 and pa.id = ?2 and sub.direccion = ?3");
    $query->setParameter(1, $idSublinea);
    $query->setParameter(2, $idParada);
    $query->setParameter(3, $direccion);
    return $query->getResult();
}
```

Figura 37. Captura función query en DQL

Para el acceso al área privada, queremos destacar que hemos usado JWT (Json Web Token) como estándar de autenticación. Para ello se ha hecho uso de los bundles SecurityBundle y LexikJWTAuthenticationBundle. Además, se debe realizar la configuración de las claves privadas y públicas, así como de firewalls y control de accesos dentro del fichero de security.yaml. Esto determina qué rutas serán de acceso público y cuáles serán de acceso privado dentro del sistema.


```

config > packages > security.yaml
You 2 weeks ago | 1 author (You)
1 security: You 3 weeks ago • seguridad y población en búsqueda
2 enable_authenticator_manager: true
3 # https://symfony.com/doc/current/security.html#registering-the-user-hashing-passwords
4 password_hashers:
5     Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
6         algorithm: 'auto'
7         cost: 15
8 # https://symfony.com/doc/current/security.html#loading-the-user-the-user-provider
9 providers:
10     # used to reload user from session & other features (e.g. switch_user)
11     app_user_provider:
12         entity:
13             class: App\Entity\Usuario
14             property: username
15 firewalls:
16     login:
17         pattern: ^/api/login
18         stateless: true
19         json_login:
20             check_path: /api/login_check
21             success_handler: lexik_jwt_authentication.handler.authentication_success
22             failure_handler: lexik_jwt_authentication.handler.authentication_failure
23     api:
24         pattern: ^/api
25         stateless: true
26         jwt: ~
27     dev:
28         pattern: ^/(_(profiler|wdt)|css|images|js)/
29         security: false
30 main:
31     lazy: true
32     provider: app_user_provider
33     form_login:
34         login_path: app_login
35         check_path: app_login
36     logout:
37         path: app_logout
38         target: api_
39

```

Figura 38. Captura de configuración de Firewalls en fichero security.yaml

```

# Easy way to control access for large sections of your site
# Note: Only the *first* access control that matches will be used

access_control:
    # - { path: ^/admin, roles: ROLE_ADMIN }
    # - { path: ^/profile, roles: ROLE_USER }
    - { path: ^/registro/api, roles: PUBLIC_ACCESS }
    - { path: ^/login, roles: PUBLIC_ACCESS }
    - { path: ^/api, roles: IS_AUTHENTICATED_FULLY }

when@test:
    security:
        password_hashers:
            # By default, password hashers are resource intensive and take time. This is
            # important to generate secure password hashes. In tests however, secure hashes
            # are not important, waste resources and increase test times. The following
            # reduces the work factor to the lowest possible values.
            Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface:
                algorithm: auto
                cost: 4 # Lowest possible value for bcrypt
                time_cost: 3 # Lowest possible value for argon
                memory_cost: 10 # Lowest possible value for argon

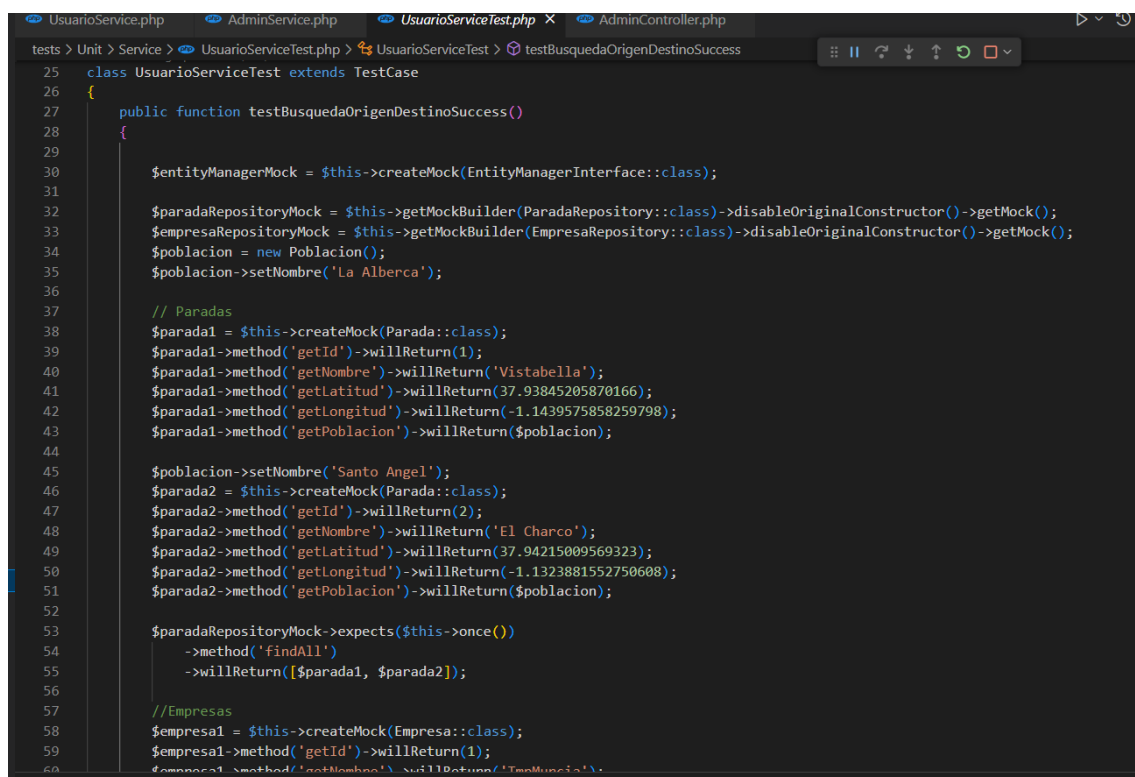
```

Figura 39. Captura de configuración de access control en fichero security.yaml

Este estándar de seguridad es uno de los más usados en web, ya que desde el entorno cliente al introducir usuario y contraseña se lanza una petición que es respondida con un token de seguridad, y el cual será necesario para cualquier petición de recursos al área privada de la API rest.

Por último, hemos querido incorporar una parte de testeo en la aplicación a través de test unitarios en algunos métodos de la aplicación y su ejecución de forma automática. Para ello realizamos distintas pruebas con un proyecto a modo de ejemplo, y procedimos a investigar la forma de automatizarlos.

Los test unitarios se ejecutan sobre el servicio del controlador Usuario, que es el que contiene casi toda la lógica. Para ello, se ha creado una clase test de dicho servicio y se han incluido “mocks” u objetos que suplantan la dependencia de una clase, con el fin de poder crear objetos sobre los que tenemos el control sobre los valores que devuelven. Esto lo hemos hecho para no tener que configurar una base de datos extra para las pruebas.



```
25 class UsuarioServiceTest extends TestCase
26 {
27     public function testBusquedaOrigenDestinoSuccess()
28     {
29         $entityManagerMock = $this->createMock(EntityManagerInterface::class);
30
31         $paradaRepositoryMock = $this->getMockBuilder(ParadaRepository::class)->disableOriginalConstructor()->getMock();
32         $empresaRepositoryMock = $this->getMockBuilder(EmpresaRepository::class)->disableOriginalConstructor()->getMock();
33         $poblacion = new Poblacion();
34         $poblacion->setNombre('La Alberca');
35
36         // Paradas
37         $parada1 = $this->createMock(Parada::class);
38         $parada1->method('getId')->willReturn(1);
39         $parada1->method('getNombre')->willReturn('Vistabella');
40         $parada1->method('getLatitud')->willReturn(37.93845205870166);
41         $parada1->method('getLongitud')->willReturn(-1.1439575858259798);
42         $parada1->method('getPoblacion')->willReturn($poblacion);
43
44         $parada2 = $this->createMock(Parada::class);
45         $parada2->method('getId')->willReturn(2);
46         $parada2->method('getNombre')->willReturn('El Charco');
47         $parada2->method('getLatitud')->willReturn(37.94215009569323);
48         $parada2->method('getLongitud')->willReturn(-1.1323881552750608);
49         $parada2->method('getPoblacion')->willReturn($poblacion);
50
51         $paradaRepositoryMock->expects($this->once())
52             ->method('findAll')
53             ->willReturn([$parada1, $parada2]);
54
55         // Empresas
56         $empresa1 = $this->createMock(Empresa::class);
57         $empresa1->method('getId')->willReturn(1);
58         $empresa1->method('getNombre')->willReturn('Ten Murcia');
```

Figura 40. Captura de test unitario de la clase UsuarioService

En un primer momento, para la automatización, valoramos la opción de usar GitHub Actions, ya que con esta herramienta de Github se pueden automatizar flujos de trabajo a través de ficheros yaml. Estos ficheros están ya disponibles en la biblioteca de GitHub Actions, y concretamente existe uno para las pruebas unitaria en PHP.

Choose a workflow

Build, test, and deploy your code. Make code reviews, branch management, and issue triaging work the way you want. Select a workflow to get started.

Skip this and [set up a workflow yourself](#) →

Categories

Deployment

Continuous integration

Automation

Pages

Found 1 workflow

Symfony

By GitHub Actions

Test a Symfony project.

Configure

PHP



© 2024 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)

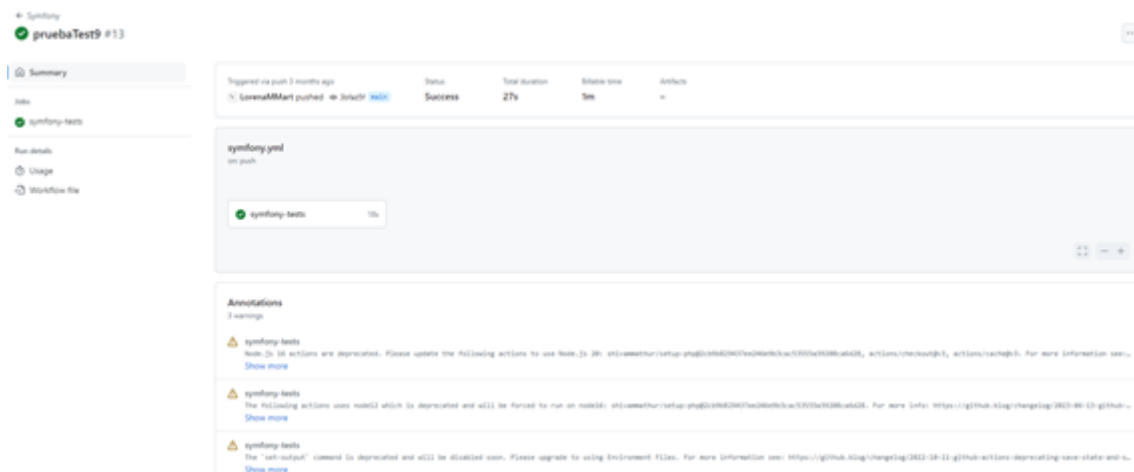
Figura 41. Captura de búsqueda del flujo de trabajo de pruebas test Symfony en Github

Una vez subido el proyecto a Github y configurado el flujo de trabajo, se ejecutan de forma automática los test y lanza un error si un test ha fallado.

The screenshot shows the GitHub Actions interface for a repository named 'LorenaMMart / PruebaTestPHP'. The 'All workflows' tab is selected, showing a list of workflow runs. The table has columns for workflow name, event, status, branch, and actor. The runs are filtered by 'pruebaTest'.

Workflow name	Event	Status	Branch	Actor
pruebaTest9	Symfony #13: Commit 33a2f2f pushed by LorenaMMart	Success	main	LorenaMMart
pruebaTest8	PHP Composer #5: Commit 33a2f2f pushed by LorenaMMart	Failure	main	LorenaMMart
pruebaTest6	Symfony #12: Commit 33a2f2f pushed by LorenaMMart	Success	main	LorenaMMart
pruebaTest7	Symfony #11: Commit 33a2f2f pushed by LorenaMMart	Success	main	LorenaMMart
pruebaTest5	PHP Composer #2: Commit 33a2f2f pushed by LorenaMMart	Failure	main	LorenaMMart

Figura 42. Captura de las pruebas realizadas con el proyecto de ejemplo.



Descartamos esta opción porque para proteger la rama principal y que no se hiciera push desde el repositorio local si no se pasaban los test, debíamos tener una cuenta corporativa. El flujo se ejecutaba, pero si fallaban los test, el push seguía adelante.

Por tanto, buscamos otra opción que nos garantizara que el proceso de actualización de las versiones del código al repositorio en remoto se hiciera sin errores. Para ello, usamos git hooks, es decir, configuramos un script personalizado de pre-commit con el que se ejecutan los tests automáticamente cuando se intenta hacer commit del proyecto, y si éstos dan fallo, no se lleva a cabo, imposibilitando la subida del código al repositorio en remoto hasta que los tests sean correctos.

<input type="checkbox"/>	Nombre	Fecha de modificación	Tipo	Tamaño
<input type="checkbox"/>	applypatch-msg.sample	29/03/2024 19:29	Archivo SAMPLE	1 KB
<input type="checkbox"/>	commit-msg.sample	29/03/2024 19:29	Archivo SAMPLE	1 KB
<input type="checkbox"/>	fsmonitor-watchman.sample	29/03/2024 19:29	Archivo SAMPLE	5 KB
<input type="checkbox"/>	post-update.sample	29/03/2024 19:29	Archivo SAMPLE	1 KB
<input type="checkbox"/>	pre-applypatch.sample	29/03/2024 19:29	Archivo SAMPLE	1 KB
<input checked="" type="checkbox"/>	pre-commit	29/03/2024 20:45	Archivo	1 KB
<input type="checkbox"/>	pre-commit.sample	29/03/2024 19:29	Archivo SAMPLE	2 KB
<input type="checkbox"/>	pre-merge-commit.sample	29/03/2024 19:29	Archivo SAMPLE	1 KB
<input type="checkbox"/>	prepare-commit-msg.sample	29/03/2024 19:29	Archivo SAMPLE	2 KB
<input type="checkbox"/>	pre-push.sample	29/03/2024 19:29	Archivo SAMPLE	2 KB
<input type="checkbox"/>	pre-rebase.sample	29/03/2024 19:29	Archivo SAMPLE	5 KB
<input type="checkbox"/>	pre-receive.sample	29/03/2024 19:29	Archivo SAMPLE	1 KB
<input type="checkbox"/>	push-to-checkout.sample	29/03/2024 19:29	Archivo SAMPLE	3 KB
<input type="checkbox"/>	sendemail-validate.sample	29/03/2024 19:29	Archivo SAMPLE	3 KB
<input type="checkbox"/>	update.sample	29/03/2024 19:29	Archivo SAMPLE	4 KB

Figura 44. Captura de lista de hooks en directorio Git del proyecto

```
pre-commit: Bloc de notas
Archivo Edición Formato Ver Ayuda
#!/usr/bin/php
<?php
printf("%s > pre-commit > ejecutando los tests ... %1\\$s", PHP_EOL);

$proyecto = basename(getcwd());

exec('php bin/phpunit', $output, $returnCode);

if ($returnCode != 0) {
    $minimalTestSummary = array_pop($output);

    printf(" [ERROR] Algún test de %s ha producido un error: ", $proyecto);
    printf("( %s ) %s%2\\$s", $minimalTestSummary, PHP_EOL);

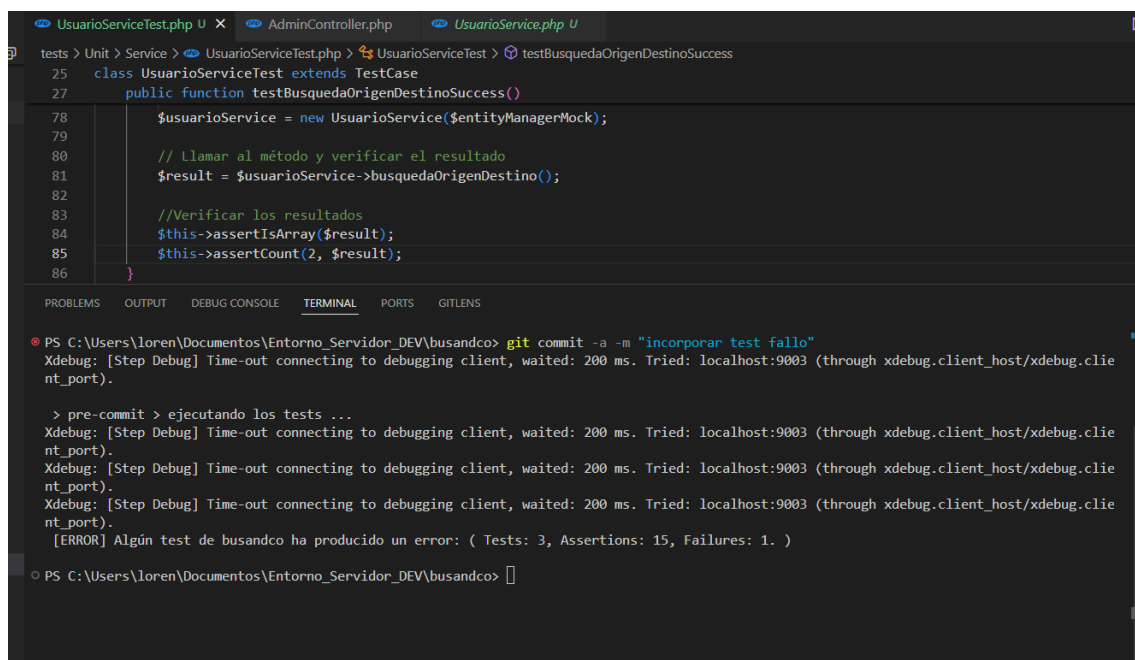
    exit(1);
}

printf(" [OK] Los tests de %s han pasado correctamente.%s%2\\$s", $proyecto, PHP_EOL);

exit(0);
```

Línea 1 columna

Figura 45. Captura del Script pre-commit



```
UsuarioServiceTest.php U x AdminController.php UsuarioService.php U
tests > Unit > Service > UsuarioServiceTest.php > UsuarioServiceTest > testBusquedaOrigenDestinoSuccess
25 class UsuarioServiceTest extends TestCase
27 public function testBusquedaOrigenDestinoSuccess()
78     $usuarioService = new UsuarioService($entityManagerMock);
79
80     // Llamar al método y verificar el resultado
81     $result = $usuarioService->busquedaOrigenDestino();
82
83     //Verificar los resultados
84     $this->assertIsArray($result);
85     $this->assertCount(2, $result);
86 }
```

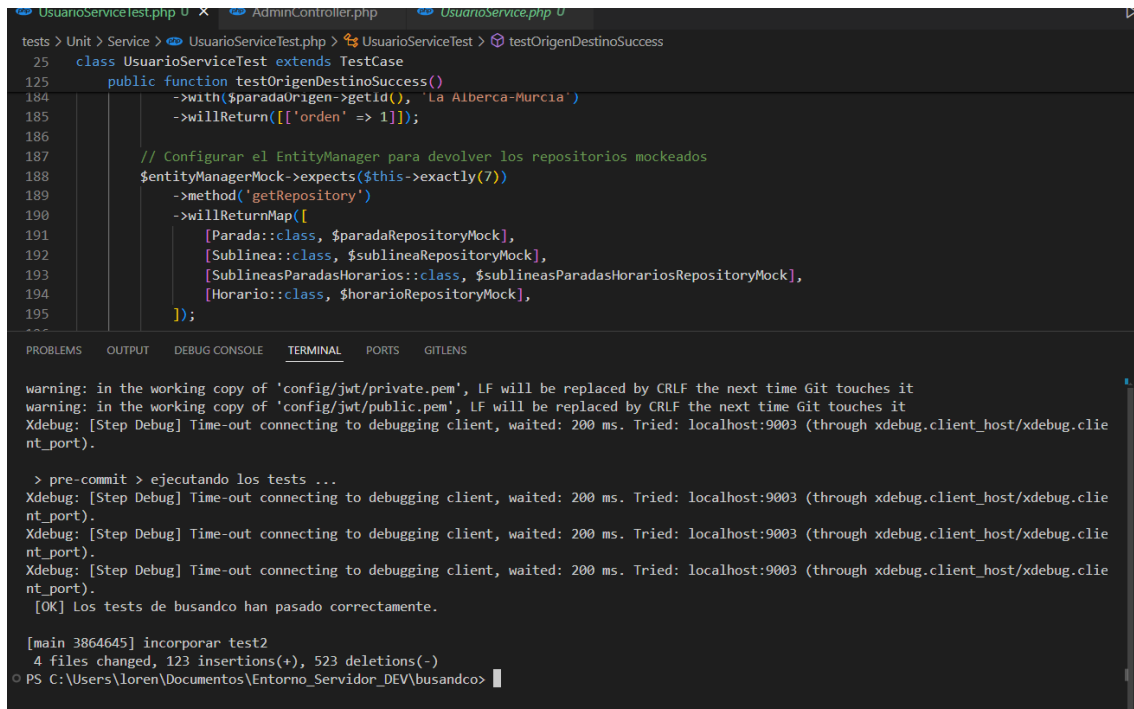
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS C:\Users\loren\Documents\Entorno_Servidor_DEV\busandco> git commit -a -m "incorporar test fallo"
Xdebug: [Step Debug] Time-out connecting to debugging client, waited: 200 ms. Tried: localhost:9003 (through xdebug.client_host/xdebug.client_port).

> pre-commit > ejecutando los tests ...
Xdebug: [Step Debug] Time-out connecting to debugging client, waited: 200 ms. Tried: localhost:9003 (through xdebug.client_host/xdebug.client_port).
Xdebug: [Step Debug] Time-out connecting to debugging client, waited: 200 ms. Tried: localhost:9003 (through xdebug.client_host/xdebug.client_port).
Xdebug: [Step Debug] Time-out connecting to debugging client, waited: 200 ms. Tried: localhost:9003 (through xdebug.client_host/xdebug.client_port).
[ERROR] Algún test de busandco ha producido un error: ( Tests: 3, Assertions: 15, Failures: 1. )

PS C:\Users\loren\Documents\Entorno_Servidor_DEV\busandco>
```

Figura 46. Captura de commit fallido por error en los tests



The image shows a code editor with a dark theme. The top part displays a PHP file named `UsuarioServiceTest.php` with the following code:

```
25 class UsuarioServiceTest extends TestCase
125 public function testOrigenDestinoSuccess()
184     ->with($paradaOrigen->getId(), 'La Alberca-Murcia')
185     ->willReturn(['orden' => 1]);
186
187 // Configurar el EntityManager para devolver los repositorios mockeados
188 $entityManagerMock->expects($this->exactly(7))
189     ->method('getRepository')
190     ->willReturnMap([
191         [Parada::class, $paradaRepositoryMock],
192         [Sublinea::class, $sublineaRepositoryMock],
193         [SublineasParadasHorarios::class, $sublineasParadasHorariosRepositoryMock],
194         [Horario::class, $horarioRepositoryMock],
195     ]);
```

The bottom part of the image shows a terminal window with the following output:

```
warning: in the working copy of 'config/jwt/private.pem', LF will be replaced by CRLF the next time Git touches it
warning: in the working copy of 'config/jwt/public.pem', LF will be replaced by CRLF the next time Git touches it
Xdebug: [Step Debug] Time-out connecting to debugging client, waited: 200 ms. Tried: localhost:9003 (through xdebug.client_host/xdebug.client_port).
> pre-commit > ejecutando los tests ...
Xdebug: [Step Debug] Time-out connecting to debugging client, waited: 200 ms. Tried: localhost:9003 (through xdebug.client_host/xdebug.client_port).
Xdebug: [Step Debug] Time-out connecting to debugging client, waited: 200 ms. Tried: localhost:9003 (through xdebug.client_host/xdebug.client_port).
Xdebug: [Step Debug] Time-out connecting to debugging client, waited: 200 ms. Tried: localhost:9003 (through xdebug.client_host/xdebug.client_port).
[OK] Los tests de busandco han pasado correctamente.
[main 3864645] incorporar test2
4 files changed, 123 insertions(+), 523 deletions(-)
PS C:\Users\loren\Documents\Entorno_Servidor_DEV\busandco>
```

Figura 47. Captura de commit realizado con éxito por tests correctos

6.3.3 Desarrollo del entorno de cliente

Pasando al apartado del desarrollo del entorno cliente, como ya se ha mencionado anteriormente, todo el Front end se ha desarrollado con Angular 17, usando TypeScript, HTML5 y CSS3 como lenguajes de programación. Se ha elegido este framework por varios motivos. El primero es que todos los integrantes del proyecto conocíamos el Angular, ya que habíamos trabajado con él durante toda la segunda evaluación de este segundo año de Desarrollo de Aplicaciones Web. Por otro lado, queríamos que la aplicación fuera una SPA (single page application), y para esto Angular es ideal, además de que su arquitectura basada en componentes y servicios facilita mucho la interacción de los distintos elementos y pantallas de la aplicación.

Desarrollo de los estilos en CSS3.

En lo que se refiere a los estilos, se planteó usar algún framework de CSS, como Bootstrap, o incluso alguna plantilla con un diseño ya establecido, pero se descartaron estas opciones, debido a que queríamos que el aspecto de la aplicación fuera lo más fiel posible a los prototipos de Figma que habíamos elaborado, y seguramente habríamos tardado más tiempo en modificar una plantilla o los componentes ya estilados que proporciona Bootstrap, que en elaborar todos los estilos desde cero. No obstante, esta elaboración de los estilos sin ningún tipo de plantilla o framework, ha resultado ser bastante costosa y, de hecho, un apunte interesante es que cuando intentamos compilar el proyecto, obtuvimos un error debido a que el

archivo de CSS del componente Combinación, que cuenta con casi cuatrocientas líneas de código, excedía el Budget por defecto de Angular, así que tuvimos que cambiar la configuración en el archivo angular.json, y ampliar ese parámetro.

Con el objetivo de reutilizar código, empleamos la metodología de crear clases generales para elementos que se repetían bastante dentro de la aplicación, como los botones, o las tarjetas, para que así, solo aplicando esa clase a cada etiqueta, ya obtuviéramos todos los estilos necesarios (parecido al funcionamiento de un framework de CSS). Además, todos los colores empleados se guardaron en variables, y siempre se referencia a esas variables, para que, si en algún momento hay que cambiar alguno, modificando la variable se cambie en todos los elementos que lo usan. En las siguientes imágenes se muestran algunos de los fragmentos más representativos del código CSS:

```
.cabecera{
  width: 100%;
  min-height: 300px;
  background: linear-gradient(270deg, rgba(255, 255, 255, 0.0) 0%, rgba(255, 255, 255, 0.5) 100%), url('../assets/imagenes/Zaen-header.png');
  background-size: cover;
  background-position-x: center;
  background-position-y: 38%;
  background-repeat: no-repeat;
  color: var(--azul-oscuro);
  display: grid;
  grid-template-columns: repeat(10, 1fr);
  grid-template-rows: 4fr 3fr;
}
```

Figura 48. Ejemplo de código CSS de una parte de la cabecera de la aplicación.

```
.boton{
  height: 48px;
  font-weight: 600;
  font-family: Urbanist;
  font-size: 16px;
  color: var(--azul-oscuro);
  background-color: var(--amarillo);
  border: 0;
  border-radius: 10px;
  transition: background-color 0.3s;
}
.boton:hover{
  cursor: pointer;
  background-color: #06D6A0;
  transition: background-color 0.3s;
}
```

Figura 49. Código CSS que se aplica a todos los botones primarios de la aplicación.


```

.linea-trayecto {
  box-sizing: border-box;
  position: relative;
  display: block;
  transform: scale(var(--ggs,1));
  background-color: var(--azul-oscuro);
  width: 52.5%;
  height: 3px;
}
.linea-trayecto::after,
.linea-trayecto::before {
  content: "";
  display: block;
  box-sizing: border-box;
  position: absolute;
}

```

```

.linea-trayecto::after {
  background-color: var(--azul-oscuro);
  width: 10px;
  height: 10px;
  border: 2px solid;
  border-radius: 8px;
  right: 0;
  bottom: -3.5px;
}
.linea-trayecto::before {
  background-color: var(--azul-oscuro);
  width: 10px;
  height: 10px;
  border: 2px solid;
  border-radius: 8px;
  left: -1px;
  bottom: -3.5px;
}

```

Figuras 50 y 51. Código css de las líneas que se emplean para representar los trayectos y sus paradas en el componente Combinación.

Implementación del mapa con Leaflet y Open Street Maps

En la página de línea detallada, se representa el recorrido de cada línea mediante un mapa interactivo, con un marcador en cada parada, y el recorrido marcado con una línea. Para esta funcionalidad empleamos la librería Leaflet, que permite añadir a tu web y modificar un mapa interactivo, de manera gratuita, siempre y cuando se de reconocimiento a Open Street Maps y sus colaboradores.

```

configurarMapa() {
  this.mapa = L.map(this.id).setView([37.9815, -1.1277], 14);
  L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution:
      '&copy; <a href="https://www.openstreetmap.org/copyright">OpenStreetMap</a> contributors',
  }).addTo(this.mapa);
  this.recorrido.forEach((punto: any, index: any) => {
    if (index > 0) {
      const puntoAnterior = this.recorrido[index - 1];
      const puntoActual = punto;
      const ruta = L.polyline(
        [
          [puntoAnterior.latitud, puntoAnterior.longitud],
          [puntoActual.latitud, puntoActual.longitud],
        ],
        { color: '#06D6A0', weight: 5 }
      ).addTo(this.mapa);
    }
  });
  this.paradas.forEach((parada: any) => {
    L.marker([parada.coordenadas[0].latitud, parada.coordenadas[0].longitud])
      .addTo(this.mapa)
      .bindPopup(`<b>${parada.parada}</b>`);
  });
}

```

Figura 52. Código TypeScript empleado para añadir al mapa las líneas del recorrido y los marcadores de cada parada.

Almacenamiento en el Local Storage

En muchas de las pantallas de la aplicación, se hacen peticiones a la API del Back end, para recuperar las líneas, empresas, incidencias, paradas, etc. Esos datos no van a ser muy susceptibles de cambios mientras el usuario navegue por la aplicación, por lo que hacer una petición al Backend cada vez que el usuario entra en una de estas pantallas, puede ser innecesario y afectar al rendimiento. Es por eso, que estas peticiones solo se hacen la primera vez que se carga cada pantalla, o cuando se refresca la página, y mientras tanto, se guardan en el Local Storage del navegador, aumentando así la velocidad de carga de muchos componentes. No obstante, es necesario que esos datos se refresquen cuando el usuario entra por primera vez a la aplicación, o refresca la página, y para ello, en el componente principal de la aplicación (app.component) se borran los datos, que se volverán a cargar según los necesite el usuario.

```
empresas:any = JSON.parse(localStorage.getItem('empresas') || '[]')
constructor(private comunicacionService: ComunicacionService, protected contactoService: ContactoService){
  if(localStorage.getItem('empresas') == null){
    this.contactoService.setEmpresas().subscribe((json:any) => {
      this.empresas = json
      localStorage.setItem('empresas', JSON.stringify(this.empresas));
    });
  }
}
```

Figura 53. Código TypeScript empleado para guardar en el Local Storage los datos de las empresas, y solo hacer la petición al Back end en caso de que no estén cargados

```
export class AppComponent implements OnInit {
  constructor(private router: Router) {
    localStorage.removeItem('empresas');
    localStorage.removeItem('lineas');
    localStorage.removeItem('incidencias');
    localStorage.removeItem('paradas');
  }
}
```

Figura 54. Código TypeScript empleado para borrar los datos del Local Storage

Funcionalidad de búsqueda por origen y destino

Podríamos decir que la funcionalidad principal de la aplicación es la búsqueda por origen y destino, en la que el usuario introduce una parada de origen, otra de destino, opcionalmente puede incluir una fecha y una hora, y obtiene los resultados de las líneas directas entre esas dos paradas, y las combinaciones de dos líneas como alternativa, o en caso de que no haya líneas directas entre las paradas seleccionadas. Esta funcionalidad ha sido, sin duda alguna, el mayor reto en cuanto al desarrollo de la aplicación, llegando a cambiarse el enfoque de ésta varias veces, y el código empleado otras tantas.

En la versión final de esta funcionalidad, se hace una petición al Back end solicitando todos los datos de las líneas que contengan la parada de origen o la de destino introducidas por el usuario, y a partir de ese listado de líneas, se aplica toda la lógica para buscar las líneas directas o las combinaciones, teniendo en cuenta también el tipo de horario según la fecha (laboral, sábado o festivo), y la hora seleccionada por el usuario.

Esta funcionalidad es demasiado extensa como para reflejarla con imágenes del código en esta memoria, por lo que aquí se muestra una parte, que podría ser la más complicada, como es la función que establece las posibles combinaciones de líneas según las paradas y la fecha y hora proporcionada.

```
//recorrer todas las direcciones de origen y destino para buscar combinaciones
direccionesOrigen.forEach((lineaOrigen: any) => {
    let paradaOrigen = this.findParadaByDireccion(
        lineaOrigen,
        this.paradaOrigen.idParada
    );
    direccionesDestino.forEach((lineaDestino: any) => {
        let paradaDestino = this.findParadaByDireccion(
            lineaDestino,
            this.paradaDestino.idParada
        );
        //recorrer todas las paradas de las direcciones de origen y destino para buscar combinaciones
        lineaOrigen.paradas.forEach((paradaLineaOrigen: any) => {
            lineaDestino.paradas.forEach((paradaLineaDestino: any) => {
                //comprobar que la parada en la que vamos a hacer transbordo no sea la parada de origen ni la de destino
                if (
                    paradaLineaOrigen.idParada !== this.paradaOrigen.idParada &&
                    paradaLineaOrigen.idParada !==
                        this.paradaDestino
                            .idParada /*&& lineaOrigen.idSublinea !== lineaDestino.idSublinea*/
                ) {
                    //en las paradas coincidentes entre las dos líneas, comprobar el orden de las paradas
                    if (
                        paradaLineaOrigen.idParada === paradaLineaDestino.idParada &&
                        paradaLineaOrigen.orden > paradaOrigen.orden &&
                        paradaLineaDestino.orden < paradaDestino.orden
                    ) {
                        //comprobar que existen horarios en las paradas de origen y destino
                        if (
                            this.getHorarioOrigen(
                                lineaOrigen,
                                this.paradaOrigen,
                                this.getHora()
                            ) !== undefined &&
                            this.getHorarioDestino(lineaOrigen, paradaLineaOrigen) !==
                                undefined &&
                            this.getHorarioOrigen(
                                lineaDestino,
                                paradaLineaDestino,
                                this.getHorarioDestino(lineaOrigen, paradaLineaOrigen)
                            ) !== undefined &&
                            this.getHorarioDestino(lineaDestino, this.paradaDestino) !==
                                undefined
                        ) {
                            if (this.combinaciones.length !== 0) {
                                //si ya hay combinaciones, antes de añadir una nueva, comprobar que no exista ya una combinación entre las mismas líneas
                                this.combinaciones.forEach((combinacion: any) => {
                                    if (
                                        lineaOrigen.idSublinea !==
                                            combinacion.lineaOrigen.idSublinea &&
                                        lineaDestino.idSublinea !==
                                            combinacion.lineaDestino.idSublinea
                                    ) {
                                        //añadir la combinación
                                        this.combinaciones.push({
                                            lineaOrigen: lineaOrigen,
                                            lineaDestino: lineaDestino,
                                            paradaOrigen: this.paradaOrigen,
                                            paradaDestino: this.paradaDestino,
                                            horaOrigen: this.getHora(),
                                            horaDestino: this.getHora(),
                                            ordenOrigen: paradaLineaOrigen.orden,
                                            ordenDestino: paradaLineaDestino.orden
                                        });
                                    }
                                });
                            } else {
                                //añadir la combinación directamente
                                this.combinaciones.push({
                                    lineaOrigen: lineaOrigen,
                                    lineaDestino: lineaDestino,
                                    paradaOrigen: this.paradaOrigen,
                                    paradaDestino: this.paradaDestino,
                                    horaOrigen: this.getHora(),
                                    horaDestino: this.getHora(),
                                    ordenOrigen: paradaLineaOrigen.orden,
                                    ordenDestino: paradaLineaDestino.orden
                                });
                            }
                        }
                    }
                }
            })
        })
    })
})
```

```

this.combinaciones.push({
  idCombinacion: this.combinaciones.length + 1,
  lineaOrigen: {
    idLinea: lineaOrigen.idLinea,
    linea: lineaOrigen.linea,
    empresa: lineaOrigen.empresa,
    idSublinea: lineaOrigen.idSublinea,
    sublinea: lineaOrigen.sublinea,
    direccion: lineaOrigen.direccion,
    paradaOrigen: this.paradaOrigen.nombre,
    horarioOrigen: this.getHorarioOrigen(
      lineaOrigen,
      this.paradaOrigen,
      this.getHora()
    ),
    paradaDestino: paradaLineaOrigen.parada,
    horarioDestino: this.getHorarioDestino(
      lineaOrigen,
      paradaLineaOrigen
    ),
  },
},

```

```

    lineaDestino: {
      idLinea: lineaDestino.idLinea,
      linea: lineaDestino.linea,
      empresa: lineaDestino.empresa,
      idSublinea: lineaDestino.idSublinea,
      sublinea: lineaDestino.sublinea,
      direccion: lineaDestino.direccion,
      paradaOrigen: paradaLineaDestino.parada,
      horarioOrigen: this.getHorarioOrigen(
        lineaDestino,
        paradaLineaDestino,
        this.getHorarioDestino(
          lineaOrigen,
          paradaLineaOrigen
        )
      ),
      paradaDestino: this.paradaDestino.nombre,
      horarioDestino: this.getHorarioDestino(
        lineaDestino,
        this.paradaDestino
      ),
    },
  },
});
}
});

```

```

    else {
      this.combinaciones.push({
        idCombinacion: this.combinaciones.length + 1,
        lineaOrigen: {
          idLinea: lineaOrigen.idLinea,
          linea: lineaOrigen.linea,
          empresa: lineaOrigen.empresa,
          idSublinea: lineaOrigen.idSublinea,
          sublinea: lineaOrigen.sublinea,
          direccion: lineaOrigen.direccion,
          paradaOrigen: this.paradaOrigen.nombre,
          horarioOrigen: this.getHorarioOrigen(
            lineaOrigen,
            this.paradaOrigen,
            this.getHora()
          ),
          paradaDestino: paradaLineaOrigen.parada,
          horarioDestino: this.getHorarioDestino(
            lineaOrigen,
            paradaLineaOrigen
          ),
        },
        lineaDestino: {
          idLinea: lineaDestino.idLinea,
          linea: lineaDestino.linea,
          empresa: lineaDestino.empresa,
          idSublinea: lineaDestino.idSublinea,
          sublinea: lineaDestino.sublinea,
          direccion: lineaDestino.direccion,
          paradaOrigen: paradaLineaDestino.parada,
          horarioOrigen: this.getHorarioOrigen(
            lineaDestino,
            paradaLineaDestino,
            this.getHorarioDestino(lineaOrigen, paradaLineaOrigen)
          ),
          paradaDestino: this.paradaDestino.nombre,
          horarioDestino: this.getHorarioDestino(
            lineaDestino,
            this.paradaDestino
          ),
        },
      });
    }
  }
}

```

Figura 55 - 60. Función que establece las distintas combinaciones entre dos líneas.

Seguridad con Json Web Token y Angular Guards

Uno de los apartados más importantes de una aplicación web, es la seguridad. En este caso, al tratarse de una aplicación de Angular en el Front end con una API de Symfony en el Back end, hemos optado por implementar la funcionalidad de login del usuario administrador, y las peticiones POST que debe hacer al Back end (como crear, editar, o eliminar una línea) usando el estándar JWT (Json Web Token). Además, se emplean los Guards de Angular, que son servicios que permiten controlar el acceso a ciertas rutas de la aplicación, en función de si el usuario está logueado o no.

```

export class AutenticacionService {
  private readonly JWT_TOKEN = 'JWT_TOKEN';
  private loggedUser?: string;
  private isAuthenticatedSubject = new BehaviorSubject<boolean>({false});
  private http = inject(HttpClient);

  constructor() {}

  login(user: { username: string; password: string }): Observable<any> {
    return this.http
      .post<any>('http://localhost:8000/api/login_check', user)
      .pipe(
        tap((tokens) => {
          this.doLoginUser(user.username, tokens);
        })
      );
  }

  isLoggedIn() {
    return this.isAuthenticatedSubject.value;
  }

  private doLoginUser(username: string, tokens: any) {
    this.loggedUser = username;
    this.storeJwtToken(tokens.token);
    this.isAuthenticatedSubject.next(true);
  }

  private storeJwtToken(jwt: string) {
    localStorage.setItem(this.JWT_TOKEN, jwt);
  }

  logout() {
    localStorage.removeItem(this.JWT_TOKEN);

    this.isAuthenticatedSubject.next(false);
  }
}

```

Figura 61. Servicio utilizado para la autenticación de los usuarios administradores.

```
export const loginGuard: CanActivateFn = (route, state) => {
  const autenticacionService = inject(AutenticacionService);
  const router = inject(Router);
  if (autenticacionService.isLoggedIn()) {
    return true;
  }
  else {
    const urlTreeReturn = router.createUrlTree(['/login']);
    return urlTreeReturn;
  }
};
```

```
{
  path: 'admin-listado',
  component: AdminListadoComponent,
  canActivate: [loginGuard],
  title: 'Líneas - Administración - Bus&Co',
},
{
  path: 'admin-crear',
  component: AdminCrearComponent,
  canActivate: [loginGuard],
  title: 'Crear línea - Administración - Bus&Co',
},
```

Figuras 62 y 63. Código empleado para proteger las rutas de las pantallas de administración mediante los Guards de Angular.

6.3.4 Despliegue de la aplicación

El despliegue de la parte desarrollada para el lado servidor y la base de datos se ha realizado en AWS (Amazon Web Services). Para ello, nos hemos dado de alta con una cuenta de estudiante y hemos creado una instancia en EC2, un servicio que ofrece instancias en la nube de máquina virtuales. En nuestro caso, configuramos una con Ubuntu Server, con 8 Gigabytes de memoria RAM y 1 CPU.

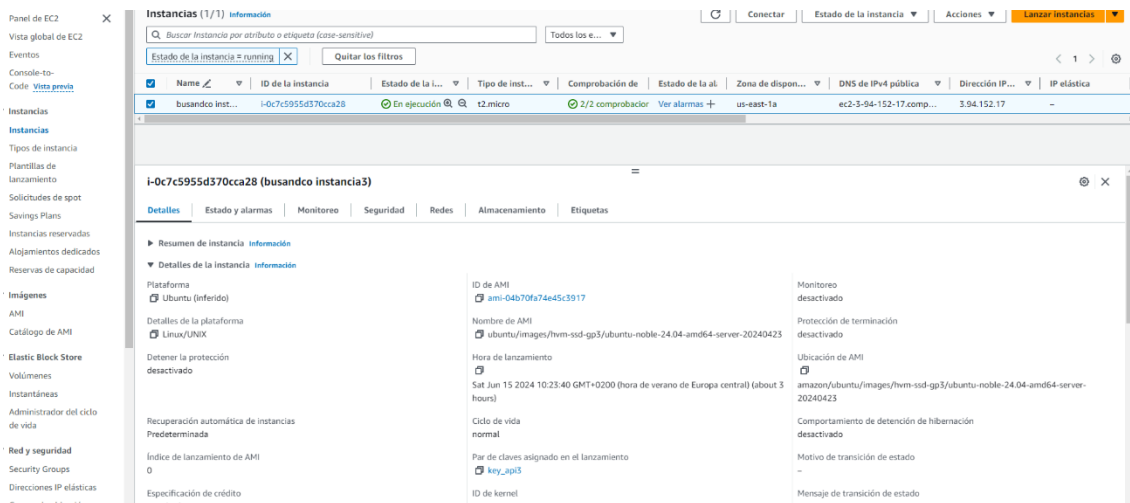


Figura 64. Captura de instancia de AWS.

Para conectarnos a la instancia lo hicimos mediante Putty, con clave pública y las DNS (sistema de nombres de dominio) generado por la instancia.

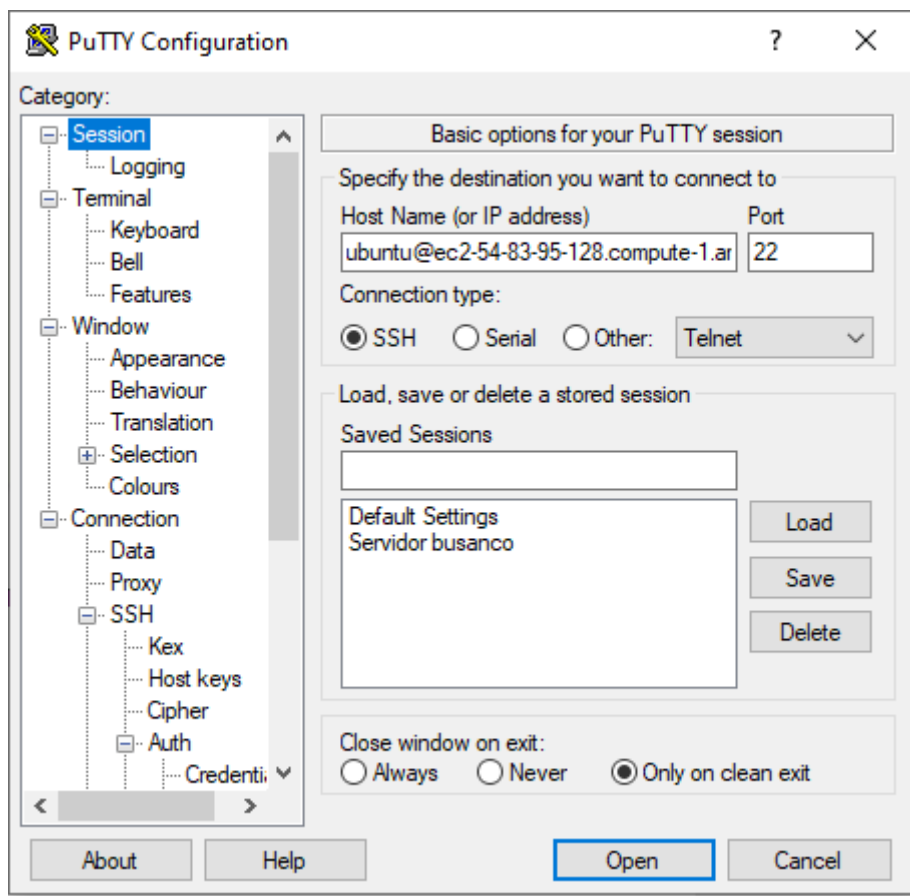
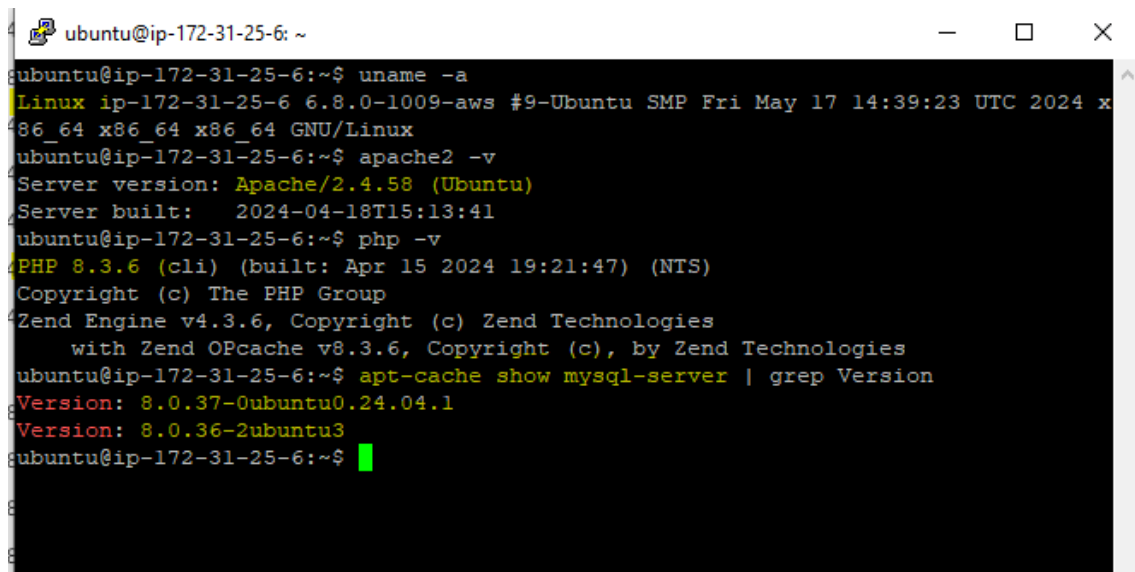


Figura 65. Captura de conexión por PuTY a la Instancia de EC2.

Una vez conectados a la máquina virtual, instalamos Lamp, acrónimo de Linux, Apache, MySQL y PHP, clonamos el repositorio en el directorio /var/www de Ubuntu Server y creamos la base de datos, mediante comandos.

A terminal window titled 'ubuntu@ip-172-31-25-6: ~' with standard window controls. The terminal shows the following commands and output:

```
ubuntu@ip-172-31-25-6:~$ uname -a
Linux ip-172-31-25-6 6.8.0-1009-aws #9-Ubuntu SMP Fri May 17 14:39:23 UTC 2024 x
86_64 x86_64 x86_64 GNU/Linux
ubuntu@ip-172-31-25-6:~$ apache2 -v
Server version: Apache/2.4.58 (Ubuntu)
Server built: 2024-04-18T15:13:41
ubuntu@ip-172-31-25-6:~$ php -v
PHP 8.3.6 (cli) (built: Apr 15 2024 19:21:47) (NTS)
Copyright (c) The PHP Group
Zend Engine v4.3.6, Copyright (c) Zend Technologies
with Zend OPcache v8.3.6, Copyright (c), by Zend Technologies
ubuntu@ip-172-31-25-6:~$ apt-cache show mysql-server | grep Version
Version: 8.0.37-0ubuntu0.24.04.1
Version: 8.0.36-2ubuntu3
ubuntu@ip-172-31-25-6:~$
```

Figura 66. Captura versiones instaladas de LAMP.

Cuando terminamos el clonado, ejecutamos el comando de “composer install” para que se descargaran todas las librerías y dependencias necesarias, al igual que se haría en un proyecto en local.

Un aspecto a tener en cuenta es que cuando creamos la base de datos debemos cambiar la ruta URL en el fichero .env del proyecto para que la conexión se ejecute correctamente.

A continuación, modificamos tanto el archivo apache2.conf, como el 000-default.conf de apache. En el primer de ellos, configuramos el servidor apache configurar el puerto, la raíz de los documentos, los módulos, los archivos de registros, los hosts virtuales, etc.


```
GNU nano 7.2                                apache2.conf
# not allow access to the root filesystem outside of /usr/share and /var/www.
# The former is used by web applications packaged in Debian,
# the latter may be used for local directories served by the web server. If
# your system is serving content from a sub-directory in /srv you must allow
# access here, or in any related virtual host.
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Require all denied
</Directory>

<Directory /usr/share>
    AllowOverride None
    Require all granted
</Directory>

<Directory /var/www/busandco/public>
    AllowOverride None
    Require all granted
    <IfModule mod_rewrite.c>
        Options -Multiviews
        RewriteEngine On
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteRule ^(.*)$ index.php [QSA,L]
    </IfModule>
</Directory>

#<Directory /srv/>
#     Options Indexes FollowSymLinks
#     AllowOverride None
#     Require all granted
#</Directory>
```

Figura 67. Captura de configuración del fichero apache2.conf

En el segundo, incluimos la ruta a la que se dirigirán las peticiones que entren al servidor apache para que lleguen a la API.

```
ubuntu@ip-172-31-25-6: /etc/apache2/sites-available
GNU nano 7.2 000-default.conf
VirtualHost *:80>
# The ServerName directive sets the request scheme, hostname and port that
# the server uses to identify itself. This is used when creating
# redirection URLs. In the context of virtual hosts, the ServerName
# specifies what hostname must appear in the request's Host: header to
# match this virtual host. For the default virtual host (this file) this
# value is not decisive as it is used as a last resort host regardless.
# However, you must set it for any further virtual host explicitly.
#ServerName www.example.com

ServerAdmin webmaster@localhost
DocumentRoot /var/www/busandco/public

# Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
# error, crit, alert, emerg.
# It is also possible to configure the loglevel for particular
# modules, e.g.
#LogLevel info ssl:warn

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

# For most configuration files from conf-available/, which are
# enabled or disabled at a global level, it is possible to
# include a line for only one particular virtual host. For example the
# following line enables the CGI configuration for this host only
# after it has been globally disabled with "a2disconf".
#Include conf-available/serve-cgi-bin.conf
</VirtualHost>
```

Figura 68. Captura de configuración del fichero 000-default.conf

Por último, cargamos el fichero con el script (conjunto de instrucciones) de las inserciones de la base de datos mediante FileZilla.

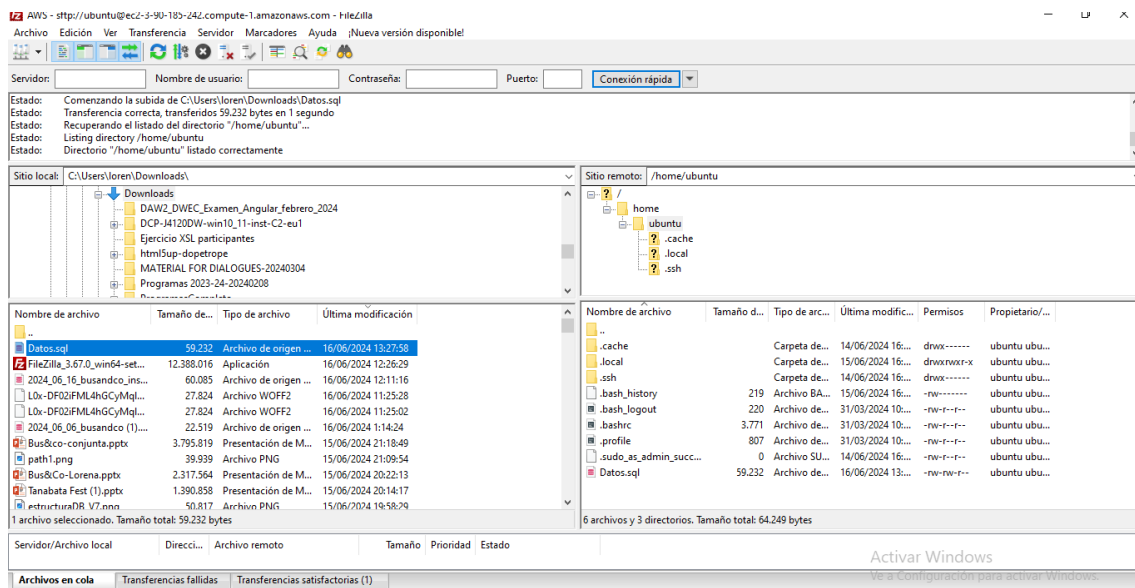


Figura 69. Captura de carga del fichero Datos.sql a instancia AWS por Filezilla.

```
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+-----+
5 rows in set (0.01 sec)

mysql> use busandco;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> source /home/ubuntu/Datos.sql
Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

Figura 70. Inserción de los datos en la base de datos por comando.

Pasando al despliegue del lado cliente, cabe destacar que ha sido mucho más sencillo que el del lado servidor. En este caso utilizamos la capa gratuita de Vercel, un hosting especializado en proyectos de JavaScript y TypeScript, y que automatiza todo el proceso de despliegue, para que el usuario solo tenga que importar el repositorio del proyecto. A continuación, se detalla el proceso completo, acompañado de capturas de pantalla.

Tras iniciar sesión con la cuenta de GitHub, para que así Vercel tenga acceso al repositorio del proyecto, debemos seleccionar Add new – Project.

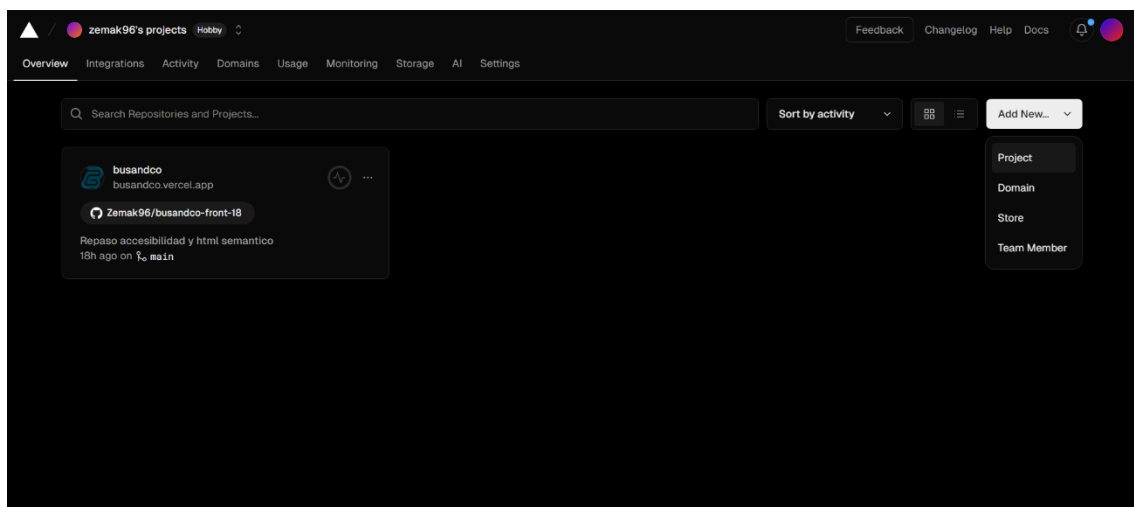


Figura 71. Captura de la página principal del panel de control de Vercel.

Ahora tendremos que seleccionar el repositorio que queremos importar (en este caso el del Frontend de Bus&Co:

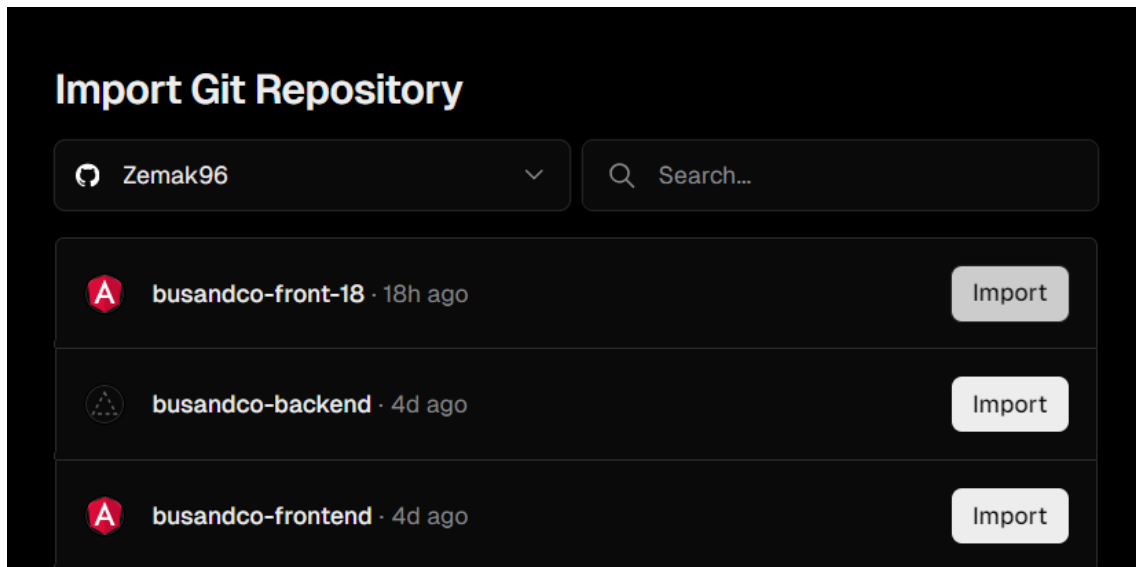
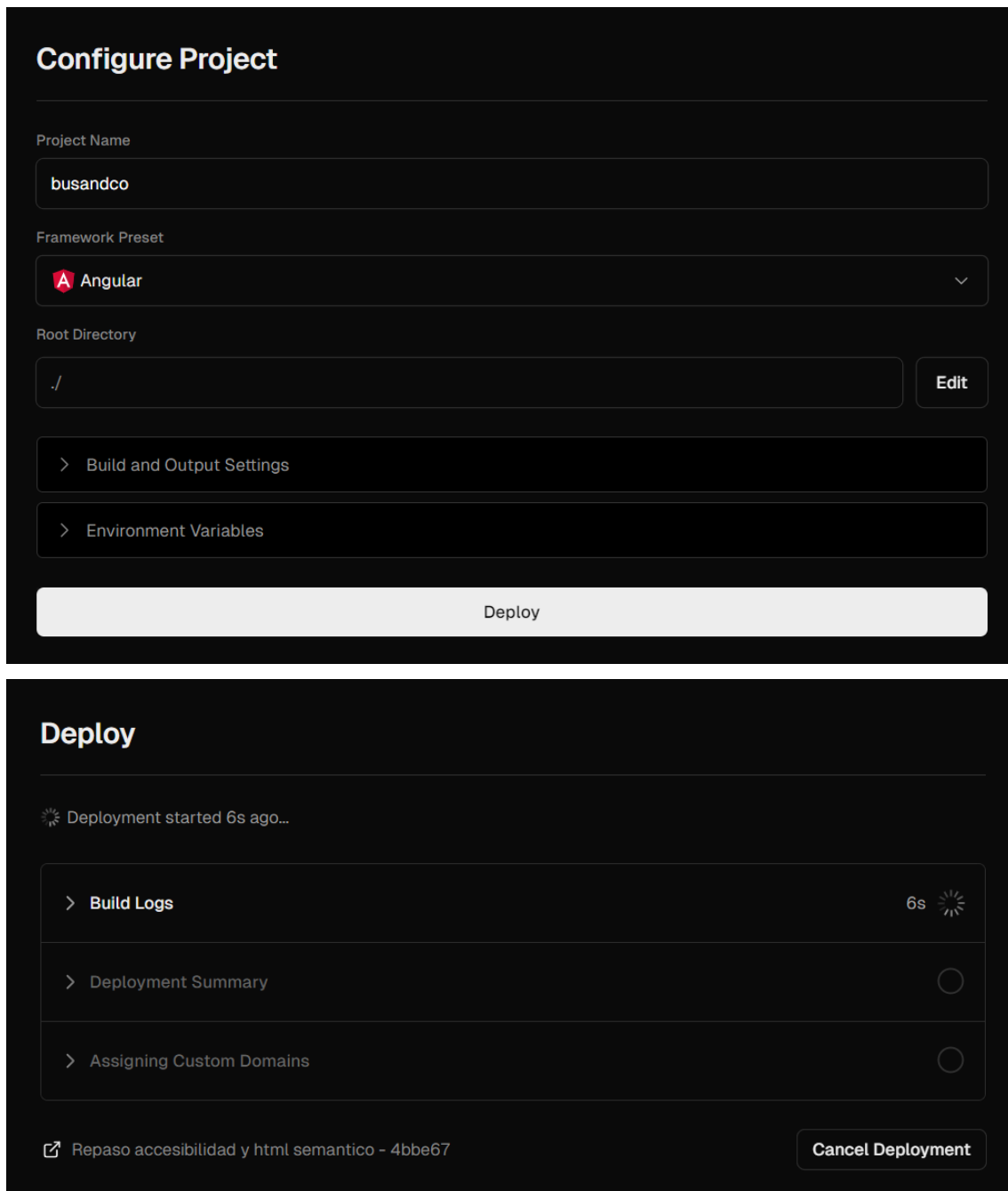


Figura 72. Captura del proceso de importar repositorio en Vercel.

En este punto, elegiremos el nombre del proyecto, y las diferentes configuraciones (directorio raíz, opciones de compilación, variables de entorno, etc.), y seleccionaremos Deploy:



Figuras 72 y 73. Capturas del proceso de despliegue en Vercel.

En este momento comenzará el proceso de compilar y desplegar el proyecto, y si no hay ningún error, en cuanto se complete, tendremos nuestro proyecto desplegado. Cabe destacar que en este punto tuvimos que subsanar un error que nos costó bastante tiempo resolver, y es que un archivo CSS de la librería Leaflet, que usamos para el mapa del recorrido de las líneas, no podía cargar las imágenes que usaba de fondo en diferentes clases. Para arreglar este error, tuvimos que hacer una copia del archivo css y las imágenes fuera de la carpeta node_modules, en la que están todos los archivos de las librerías, y enlazar a esas copias en lugar de a los archivos originales.

7 Uso de la Inteligencia Artificial

A lo largo del proyecto hemos hecho uso de la inteligencia artificial para intentar automatizar procesos o generar recursos, pero siempre teniendo en cuenta la ética y sobre todo evitando su uso en el propio desarrollo de la aplicación. Creemos que usar este tipo de tecnología es una ayuda, pero en ningún caso debe suplantar nuestro conocimiento ni nuestra capacidad de resolución.

Un ejemplo de uso en nuestro proyecto ha sido la generación de las inserciones de datos en la base de datos para nutrir nuestro esquema. En las siguientes figuras se muestra la instrucción dada (prompt) y el resultado obtenido.

```
genera un insert similar a : INSERT INTO `sublineas_paradas_horarios`  
(`id`,`sublinea_id`,`horario_id`,`parada_id`,`orden`,`direccion`) VALUES  
(31, 1, 31, 3, 3, 'La Alberca-Murcia'),  
(32, 1, 32, 3, 3, 'La Alberca-Murcia'),  
(33, 1, 33, 3, 3, 'La Alberca-Murcia'),  
(34, 1, 35, 3, 3, 'La Alberca-Murcia'),  
(35, 1, 36, 3, 3, 'La Alberca-Murcia'),  
(36, 1, 37, 3, 3, 'La Alberca-Murcia'),  
(37, 1, 38, 3, 3, 'La Alberca-Murcia'),  
(38, 1, 39, 3, 3, 'La Alberca-Murcia'),  
(39, 1, 40, 3, 3, 'La Alberca-Murcia'),  
(40, 1, 41, 3, 3, 'La Alberca-Murcia'),  
(41, 1, 42, 3, 3, 'La Alberca-Murcia'),  
(42, 1, 43, 3, 3, 'La Alberca-Murcia'),  
(43, 1, 44, 3, 3, 'La Alberca-Murcia'),  
(44, 1, 45, 3, 3, 'La Alberca-Murcia'),  
(45, 1, 34, 3, 3, 'La Alberca-Murcia'); quitanto el campo id, cambiando  
la sublinea_id por 6, el horario_id empezaría en 76 hasta 101, la  
parada_id es 5, y el orden 7
```

Figura 74. Captura de instrucción realizada a Chat GPT



Aquí tienes el nuevo insert siguiendo tus indicaciones:

```
sql Copiar código

INSERT INTO `sublineas_paradas_horarios` (`sublinea_id`, `horario_id`, `parada_id`, `orden`
(6, 76, 5, 7, 'La Alberca-Murcia'),
(6, 77, 5, 7, 'La Alberca-Murcia'),
(6, 78, 5, 7, 'La Alberca-Murcia'),
(6, 80, 5, 7, 'La Alberca-Murcia'),
(6, 81, 5, 7, 'La Alberca-Murcia'),
(6, 82, 5, 7, 'La Alberca-Murcia'),
(6, 83, 5, 7, 'La Alberca-Murcia'),
(6, 84, 5, 7, 'La Alberca-Murcia'),
(6, 85, 5, 7, 'La Alberca-Murcia'),
(6, 86, 5, 7, 'La Alberca-Murcia'),
(6, 87, 5, 7, 'La Alberca-Murcia'),
(6, 88, 5, 7, 'La Alberca-Murcia'),
(6, 89, 5, 7, 'La Alberca-Murcia'),
(6, 90, 5, 7, 'La Alberca-Murcia'),
(6, 91, 5, 7, 'La Alberca-Murcia'),
(6, 79, 5, 7, 'La Alberca-Murcia');
```

Figura 75. Captura de los resultados de Chat GPT

Además, también usamos ChatGPT para generar noticias sobre los autobuses del municipio de Murcia. A continuación, se muestra el prompt utilizado, y el resultado obtenido:

generame en JavaScript un array de 5 noticias sobre los autobuses del municipio de murcia. cada noticia tiene que tener las siguientes propiedades: id, titulo, descripcion, fecha, cuerpo1, cuerpo2, cuerpo3, cuerpo4 y cuerpo5. cada apartado del cuerpo tiene que tener un mínimo de 100 palabras

Figura 76. Prompt utilizado para generar las noticias

```
javascript Copiar código

const noticias = [
  {
    id: 1,
    titulo: "Nueva flota de autobuses en Murcia",
    descripcion: "Murcia presenta su nueva flota de autobuses ecológicos.",
    fecha: "2024-06-16",
    cuerpo1: "El municipio de Murcia ha dado un gran paso hacia la sostenibilidad con la i",
    cuerpo2: "Durante la presentación, el alcalde de Murcia destacó la importancia de esta",
    cuerpo3: "La adquisición de esta nueva flota ha sido posible gracias a una colaboraci",
    cuerpo4: "Los ciudadanos de Murcia han recibido con entusiasmo la llegada de los nuevo",
    cuerpo5: "El impacto de esta medida se espera que sea significativo en términos de red",
  },
  {
    id: 2,
    titulo: "Ampliación de rutas de autobuses en Murcia",
    descripcion: "Murcia amplía las rutas de autobuses para mejorar la conectividad.",
    fecha: "2024-06-16",
    cuerpo1: "En un esfuerzo por mejorar la conectividad y facilitar el acceso al transpor",
    cuerpo2: "El concejal de Transporte, Juan López, explicó que esta ampliación es parte",
    cuerpo3: "La implementación de las nuevas rutas ha sido precedida por un estudio exhaus",
    cuerpo4: "La reacción de los ciudadad",
    cuerpo5: "Además de las mejoras en las rutas, el municipio de Murcia también está trab",
  }
]
```

Figura 77. Incorporación noticias al código.

Por último, utilizamos Designer, una inteligencia artificial de Microsoft, para generar la imagen de la cabecera de la aplicación. En este caso, introdujimos las instrucciones en inglés indicando los colores de la paleta y el tamaño de la imagen a generar. Para tener una imagen que nos resultara atractiva para la cabecera, fuimos modificando las instrucciones en cada imagen generada hasta llegar al resultado final.

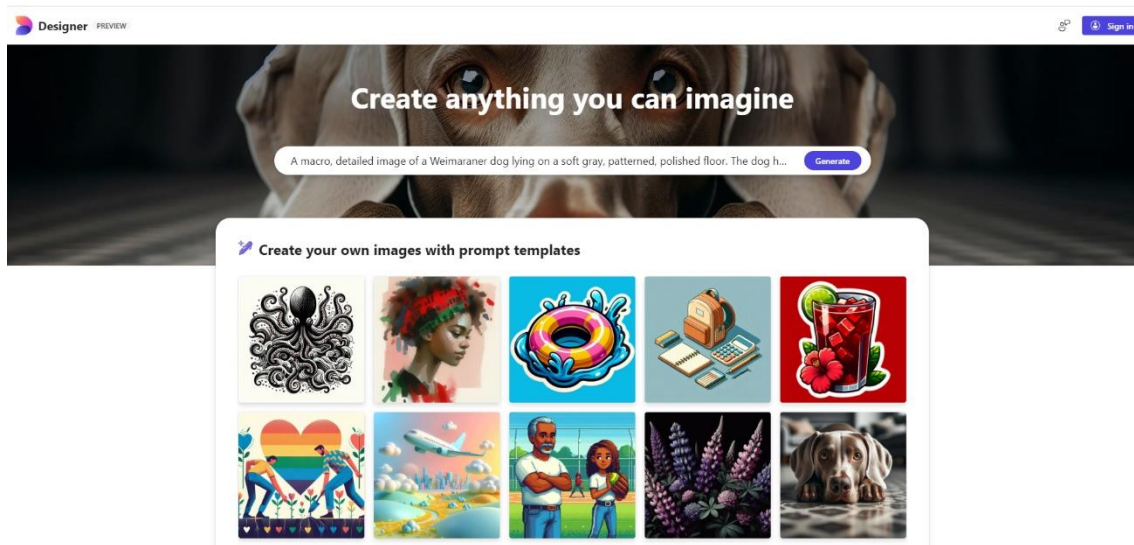


Figura 78. Captura de IA Microsoft Desingner

A continuación, indicamos el prompt usado:

“As a designer. create an illustration about a bus in the middle of a landscape with some trees, a mountain, with following colors: FFD166, 06D6A0, 117286, 073B4C and size 1500 x 500 pixels”.

8 Presupuesto

Hemos desglosado el trabajo en diversas etapas, teniendo en cuenta los recursos necesarios y el tiempo para cada una. Con el fin de brindar un cálculo preciso, hemos aplicado una tarifa horaria competitiva basada en la complejidad y la dedicación requerida.

El objetivo principal de este presupuesto es proporcionar una visión clara de los costes involucrados en la ejecución de Bus&Co.



Figura 79. Diagrama representativo de las horas empleadas en cada sección del proyecto.

A continuación, en la tabla se puede observar el precio por hora en euros en las distintas secciones de la web:

Sección	Tiempo	€/h	Coste
Diseño y reuniones	55	20	1100
Maquetación HTML y CSS	45	8	360
Desarrollo Frontend	75	20	1500
Desarrollo Backend	60	20	1200
Despliegue	30	15	450
Tiempo y coste total	265		4610

Figura 80. Desglose de los costes del proyecto.

Como podemos observar la inversión total en horas son 265, con un coste de proyecto del valor de 4610 €, más el tiempo empleado en investigaciones para su desarrollo. En cuanto al despliegue, hemos usado la opción gratuita de la plataforma Vercel para Angular, y AWS con cuenta de estudiante que dispone de 100 dólares de crédito para pruebas, en el caso de la API en Symfony y base de datos.

9 Conclusiones y vías futuras

El reto de este proyecto ha sido el de poder crear una solución en la que los usuarios de transporte público en autobús del municipio de Murcia, pudieran consultar de forma unificada toda la información relativa a dicho transporte, con independencia de su condición y capacidades. Era un reto muy ambicioso del que fuimos conscientes desde el primer momento, por ello, hemos puesto todo nuestro esfuerzo y empeño en poder realizar las funcionalidades necesarias para cumplir con este propósito. Creemos que hemos hecho un buen trabajo y hemos sido fieles a nuestra idea inicial, incorporando en la medida de lo posible, implementaciones y desarrollos que nos han obligado a salir de nuestra zona de confort y adquirir nuevos conocimientos.

A lo largo de la realización de este proyecto también hemos reforzado nuestros conocimientos en las tecnologías y herramientas vistas durante el curso, como el uso de los frameworks, Angular y Symfony, el manejo del control de versiones y los distintos comandos de Git, así como los repositorios en remoto de GitHub.

Como parte importante, destacar que al trabajar en equipo hemos adquirido competencias relacionadas con las habilidades blandas o soft skills. La coordinación entre los miembros del equipo, el aceptar las propuestas de todos con respeto, usar la asertividad para comunicar aquello con lo que no estamos de acuerdo y comprender las situaciones personales de cada uno adaptándonos para intentar sacar el mejor rendimiento del equipo.

Como vía futura de la aplicación, nos gustaría remarcar el conseguir implementar algunas de las funcionalidades que no hemos podido realizar por falta de tiempo en esta primera versión, como son la incorporación de todos los datos necesarios en la base de datos para un uso efectivo de la web, el almacenamiento de noticias en la base de datos mediante URL, ampliar la funcionalidad del área privada para que existan usuarios ligados a cada empresa y que éstos solo puedan acceder a la

información propia de su empresa y el tratamiento de archivos JSON para importar o exportar datos de la base de datos.

Para finalizar, mencionar que, en una segunda versión del proyecto, además de incorporar las funciones anteriormente citadas, nos gustaría añadir la posibilidad de crear cuentas con rol usuario para guardar rutas favoritas, consultar sus trayectos e información personal y su impacto en la huella de carbono, la geolocalización del usuario para ofrecer las paradas más cercanas en función de su ubicación, o la incorporación de una pasarela de pagos para poder adquirir billetes y bonos desde la propia aplicación y facilitar así la experiencia del usuario.

Como conclusión final, consideramos que nuestra web ha sido desarrollada con éxito cumpliendo su propósito de ofrecer una solución de consulta de información estandarizada, accesible y usable, aunque también somos conscientes de que tiene mucho potencial para seguir mejorando.

10 Bibliografía/Webgrafía.

10.1 Atribuciones.

Este apartado contribuye al cumplimiento del objetivo 5.

10.1.1 Imágenes.

Imagen noticia 1 por Ja Kubislav bajo licencia [Pexels](https://www.pexels.com/es-es/foto/ciudad-calle-autobus-bus-14585240/)

(<https://www.pexels.com/es-es/foto/ciudad-calle-autobus-bus-14585240/>)

Imagen noticia 2 por Ferliana Febritasari bajo licencia [Pexels](https://www.pexels.com/es-es/foto/colgar-asas-naranjas-en-transporte-publico-3766617/)

(<https://www.pexels.com/es-es/foto/colgar-asas-naranjas-en-transporte-publico-3766617/>)

Imagen noticia 3 por Scott Webb bajo licencia [Pexels](https://www.pexels.com/es-es/foto/foto-timelapse-de-la-parada-de-autobus-136739/)

(<https://www.pexels.com/es-es/foto/foto-timelapse-de-la-parada-de-autobus-136739/>)

Imagen noticia 4 por Jakob Scholz bajo licencia [Pexels](https://www.pexels.com/es-es/foto/asientos-de-banco-de-autobus-808846/)

(<https://www.pexels.com/es-es/foto/asientos-de-banco-de-autobus-808846/>)

Imagen noticia 5 por Daniel Cruz bajo licencia [Pexels](https://www.pexels.com/es-es/foto/gente-parabrisas-transporte-publico-autobus-13247069/)

(<https://www.pexels.com/es-es/foto/gente-parabrisas-transporte-publico-autobus-13247069/>)

Imagen logo TMPMurcia por Wikipedia bajo licencia [Wikimedia Commons](#)

(https://es.wikipedia.org/wiki/Transporte_de_Murcia_y_Pedan%C3%ADas#/media/Archivo:LogoTMP.svg)

Imagen logo Transportes de Murcia por Wikipedia bajo licencia [Wikimedia Commons](#)

(https://commons.wikimedia.org/wiki/File:Transportes_de_Murcia_logo.png)

Imagen logo Movibus por Wikipedia bajo licencia [Wikimedia Commons](#)

(<https://upload.wikimedia.org/wikipedia/commons/9/93/LogoMovibus.svg>)

10.1.2 Elementos HTML y CSS.

Info Icono por Chromicons bajo licencia MIT

(<https://lifeomic.github.io/chromicons.com/>)

BookOpen Icono Chromicons bajo licencia MIT

(<https://lifeomic.github.io/chromicons.com/>)

Alert-triangle Icono Chromicons bajo licencia MIT

(<https://lifeomic.github.io/chromicons.com/>)

Map-pin Icono Chromicons bajo licencia MIT

(<https://lifeomic.github.io/chromicons.com/>)

Vertical-Arrows Icono Chromicons bajo licencia MIT

(<https://lifeomic.github.io/chromicons.com/>)

10.1.3 Fuentes.

Binaryboxtuts (2023). How to make Symfony 6 REST API. Recuperado de:

(<https://www.binaryboxtuts.com/php-tutorials/how-to-make-symfony-6-rest-api/>)

Berrio, César (15 Agosto 2017). Buenas prácticas Diseño/Modelado Bases de Datos.

Recuperado de: (<https://berriotech.blogspot.com/2017/08/buenas-practicas-disenomodelado-base-de.html>)

Medium (14 de Mayo 2023). Convert dynamically Request content to DTO with

Symfony. Medium.com Recuperado de: (<https://medium.com/@etearner/how-to-transform-request-content-to-dto-with-symfony-6-2-84c9c8543200>)

Symfony (s.f) Symfony Documentation. Recuperado de:

(<https://symfony.com/doc/current/index.html>)

Angular (s.f) Angular Documentation. Recuperado de: <https://angular.dev/>

Doctrine (s.f) Doctrine Query Language. Recuperado de: <https://www.doctrine-project.org/projects/doctrine-orm/en/current/reference/dql-doctrine-query-language.html>

Tanya Webdev. (Productor) (17 de Marzo de 2023). Symfony 6 & JWT Authentication tutorial for beginners [Youtube]. De: <https://www.youtube.com/watch?v=rZIF9G-iZkQ>

Learning Partner (19 de Febrero de 2024). Angular 17 Login Refresh Token using interceptor [Youtube]. De: <https://www.youtube.com/watch?v=HBemrC8NDZU>

Grados Ramírez, Jimmy (04 de Abril de 2024). Maximizando la Eficiencia y Seguridad en tu Código: El poder de los DTOs.[Linkedin] Recuperado de: <https://www.linkedin.com/pulse/maximizando-la-eficiencia-y-seguridad-en-tu-c%C3%B3digo-el-grades-ram%C3%ADrez-ykzqf/>

Flores Arriaga, Luis (15 de Agosto de 2023) Dockerizando Symfony: Simplificando el Despliegue de Aplicaciones. [Linkedin] Recuperado de : <https://www.linkedin.com/pulse/dockerizando-symfony-simplificando-el-despliegue-de-flores-arriaga/>

Leaflet (s.f) Leaflet Documentation. Recuperado de: [Documentation - Leaflet - a JavaScript library for interactive maps \(leafletjs.com\)](https://leafletjs.com/)

Tempo (s.f) Tempo, FormKit Team. Recuperado de: [Tempo • Dates by FormKit](https://tempo.formkit.com/)

ChatGPT de OpenAI. Recuperado de: chat.openai.com

Designer de Microsoft. Recuperado de: <https://create.microsoft.com/es-es/features/ai-image-generator>

11 Anexos

11.1 Anexo I: Pasos a seguir para la instalación y puesta en marcha del proyecto.

Para poder poner en marcha el proyecto en un entorno local debemos tener instalado Xampp, para la base de datos, y los servicios de Apache y MySQL iniciados. Además, necesitamos tener instalado Symfony 6 y Angular 17. Entonces, seguiremos los siguientes pasos:

1. Primero clonamos el repositorio de la parte de backend con el comando "git clone <https://github.com/LorenaMMart/busandco>", e instalaremos las dependencias con "composer install".
2. Después, repetimos el proceso con el repositorio de frontend: "git clone <https://github.com/Zemak96/busandco-front-18>" para clonar el proyecto, y "npm install" para instalar las dependencias.
3. Ahora, debemos abrir phpmyadmin, y crear una base de datos llamada "busandco".
4. Después, con el comando "php bin/console doctrine:schema:update --force", actualizaremos la base de datos, y debemos ejecutar el script del archivo .sql que hay en el proyecto del backend.
5. Una vez ya tenemos la base de datos operativa, iniciaremos los dos servidores locales de ambos proyectos. En el caso del backend, el comando es "symfony server:start" y para el frontend "ng serve --open". En este momento, se nos abrirá la aplicación en nuestro navegador predeterminado, y podremos trabajar con ella.

11.2 Anexo III: Enlaces de interés.

Repositorio proyecto Backend: <https://github.com/LorenaMMart/busandco>

Repositorio proyecto Frontend: <https://github.com/Zemak96/busandco-front-18>

Proyecto de Figma prototipos versión móvil:

<https://www.figma.com/design/RS56P2SotHU2rPjo15nUiH/Bus%26Co?node-id=0-1&t=9InQ31QfPCo6y5mp-1>

Proyecto de Figma prototipos versión escritorio:

<https://www.figma.com/design/HLiWAqSIBm7ByX5LwEWcaM/Bus%26Co---Desktop?node-id=0-1&t=QCfxYMxV0YXXQvY0-1>

Enlace a la web: <https://busandco.vercel.app/>