UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA
CAMPUS D'ALCOI

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Politécnica Superior de Alcoy

Aplicación web para agentes comerciales

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Ortega Morales, Eduardo

Tutor/a: Esparza Peidro, Javier

CURSO ACADÉMICO: 2023/2024

RESUMEN

En este proyecto se pretende desarrollar una aplicación para agentes comerciales en la que puedan administrar la información de fabricantes y de clientes con los que trabaja, tener ordenada la información personal, tramitar propuestas de pedidos del cliente al fabricante y viceversa, informar automáticamente a ambas partes de cada cambio de estado en el que esté el pedido mediante correos automatizados, ofrecer un apartado para almacenar y mostrar los catálogos del agente comercial, ofrecer usuarios a fabricante y cliente con permisos restringidos para acceder a la aplicación y gestionar eventos.

PALABRAS CLAVE

Aplicación web, MySql, Symfony, Sonata, Agentes comerciales.

SUMMARY

This project aims to develop an application for commercial agents in which they can manage the information of manufacturers and customers with whom they work, have their personal information organized, process order proposals from the customer to the manufacturer and vice versa, automatically inform both parties of each change in the status of the order through automated emails, offer a section to store and display the catalogs of the commercial agent, offer users to manufacturer and customer with restricted permissions to access the application and manage events.

KEYWORDS

Web application, MySql, Symfony, Sonata, Commercial agents.

RESUM

En aquest projecte es pretén desenvolupar una aplicació per a agents comercials en què puguin administrar la informació de fabricants i de clients amb què treballa, tenir ordenada la informació personal, tramitar propostes de comandes del client al fabricant i viceversa, informar automàticament a ambdues parts de cada canvi d'estat en què estigui la comanda mitjançant correus automatitzats, oferir un apartat per emmagatzemar i mostrar els catàlegs de l'agent comercial, oferir usuaris a fabricant i client amb permisos restringits per accedir a l'aplicació i gestionar esdeveniments.

PARAULES CLAU

Aplicació web, MySql, Symfony, Sonata, Agents comercials.

Índice

Figuras

1. Introducción

1.1 Motivación

En esta época en la que vivimos es bastante normal ver casi todo lo que nos rodea adaptado a las nuevas tecnologías que van surgiendo, y cada vez son más las empresas que sacan presupuestos y hacen proyectos para adaptar su forma de trabajo a algo más digital y poder hacer el mismo trabajo, pero más rápido y con más calidad.

Pero adaptarse a estos cambios y formas de trabajar puede ser algo difícil para mucha gente que, en todos los años que llevan trabajando, han tenido siempre la misma forma de trabajar. Así que aún quedan muchas otras empresas que por esta razón o por alguna otra siguen usando los mismos métodos y las mismas herramientas que han estado utilizando, provocando que tengan resultados menos óptimos que otras empresas que sí se adaptan a las nuevas tecnologías, y esto puede crear estrés en el ambiente laboral.

Y aquí es donde entra el objetivo del proyecto. La idea de comenzar con esto, como en todo gran proyecto, vino de casualidad y sin buscarla, ya que todo empezó en una reunión familiar. Una de las personas asistentes a esta reunión se dedica a hacer de intermediario entre un fabricante y un cliente final, lo que viene siendo un agente comercial, y empezó a comentar que últimamente nota que le cuesta más hacer su trabajo y que en comparación con otros compañeros de profesión, él siente que va más lento y necesita más esfuerzo para llegar a los mismos resultados que ellos. Por curiosidad, se le preguntó acerca de que usaba para trabajar y cuáles suelen ser las rutinas que lleva, y claro, no se tardó en descubrir cuál podría ser una de las razones por las que le ocurre esto. Esta persona lleva más de 30 años trabajando en este sector, y cuando empezó, todo se hacía a papel y realizaba tareas, a priori simples, que ocupaban demasiado tiempo. Esto le hace perder eficacia y cada vez le costaba más estar bien organizado. Pero no solo eso, sino que casi toda la gente con la que trabaja sí que se están adaptando a los nuevos tiempos, y esto genera que tenga que hacer el doble de esfuerzo para adaptar su forma de trabajar a la de ellos.

Entonces apareció la idea de poder desarrollar una aplicación web, en la cual se pueda tener toda la información necesaria y poder realizar todas las acciones posibles que realiza diariamente desde un mismo sitio y tenerlo todo con más organización y con más rapidez. Esto no solo le podría venir bien al agente comercial, sino que también se podrían beneficiar las personas con las que trabaja, tanto en comunicación como en obtención de información.

1.2 Estudio del mercado

Con la idea ya en mente, habría que investigar si ya existen aplicaciones similares a la que se quiere desarrollar y ver cuál es el porcentaje de agentes que usan estas aplicaciones y cuáles son las razones por las que las usan y por las que no. Entonces me puse a investigar y pregunté a varios agentes comerciales que conocía la persona con la que hablé, y sí que es verdad que existen algunas aplicaciones que usan algunos de ellos, pero hay muchos que están en la misma situación que la de esta persona. Casi todos coincidían en que creen que no les merece la pena cambiar su forma de trabajo, otros que no se aclaran con estas aplicaciones, y también que las aplicaciones existentes no estaban mal, pero solo cubren algunos aspectos de su trabajo y es casi más trabajo usar esas aplicaciones por separado. Hay otras que tienen muchas funcionalidades, pero tienen un precio que no les merece la pena o que no usarían la aplicación al 100% y creen que perderían dinero. Algunos ejemplos de las aplicaciones más usadas entre ellos son Todoist, Trello o AgentesCloud.

Y entonces lo vi claro. Comprendí cuál era la competencia a la que me enfrentaba y cuáles eran los motivos de varios agentes comerciales para no optar por cambiar su forma de trabajo. Creo que, si tuviesen una aplicación web que no suponga un gran gasto al año y que es sencilla, pero a la vez cubre las necesidades básicas de su trabajo podría tener un buen futuro.

1.3 Objetivos del proyecto

A grandes rasgos y gracias a la información que se obtuvo al hablar con varios agentes comerciales, la aplicación web debe tener una interfaz limpia y amigable con el agente comercial, ya que tiene que ser lo bastante atractiva como para que decidan cambiar su forma de trabajo para usar esta aplicación.

También debe cubrir las necesidades básicas que tienen los agentes a lo largo de sus jornadas laborales. Es cierto que se pueden añadir muchas funcionalidades y seguro que serán muy prácticas, pero a veces cantidad no es calidad y es suficiente con que se pueda hacer lo necesario y con efectividad.

El objetivo principal que debe tener esta aplicación es dar al agente una forma rápida de hacer su trabajo y tener más tiempo para poder centrarse en otros aspectos. Esto se va a conseguir haciendo que el agente tenga una mejor comunicación con los fabricantes y los clientes, automatizando algunas labores o poder realizar facturas y propuestas de pedidos más fácilmente, así como tener toda la información relevante recogida en un mismo sitio y accesible rápidamente.

1.4 Estructura del resto de la memoria

Esta memoria está estructurada para que se tenga una comprensión clara y detallada del proyecto, abarcando desde la identificación del problema hasta la implementación de la solución y los resultados obtenidos. A continuación, se describen brevemente los contenidos de cada sección:

- **2. Definición del problema**: En este apartado se explican con detalle los requisitos funcionales y no funcionales del sistema, es decir, las características y funcionalidades que debe tener la aplicación para satisfacer las necesidades de los agentes comerciales. Se incluyen diagramas UML, como el diagrama de casos de uso, que representan gráficamente las interacciones entre los usuarios y el sistema, o los diagramas de secuencia de clientes y fabricantes, en los que se puede apreciar el procedimiento que pueden tener estos usuarios para realizar sus interacciones. Con este apartado se quiere hacer entender el alcance del proyecto y las necesidades específicas que se abordarán.

- **3. Solución del problema**: El diseño de la solución se presenta en este apartado, donde se describe la arquitectura general de la aplicación y las decisiones de diseño tomadas para abordar los requisitos identificados en el apartado anterior. Se divide en varios subapartados para poder cubrir los distintos aspectos del diseño:

- 3.1 Arquitectura de la aplicación: Se detalla la estructura general de la aplicación, explicando cómo se organizan los diferentes componentes que conforman el sistema. Se explica el funcionamiento del framework Symfony con el que se ha desarrollado la aplicación.

- 3.2 Tecnologías utilizadas: Se proporciona una descripción de las tecnologías y herramientas seleccionadas para el desarrollo del proyecto, justificando su elección en función de las necesidades del sistema, el nivel de compatibilidad e integración entre ellas y las ventajas que ofrecen.

- 3.3 Modelo: Se explica la gestión de datos mediante Doctrine ORM y MySQL, detallando las entidades y repositorios utilizados para la persistencia y obtención de los datos que se vayan registrando en la aplicación.

- 3.4 Vista: Se detalla el diseño de la interfaz de usuario, describiendo las plantillas Twig y el uso de CSS para crear una experiencia de usuario atractiva y responsiva [13].

- 3.5 Controlador: Se describen los archivos creados y configurados en el proyecto, incluyendo los admins, controladores, servicios, comandos y eventos que manejan las funciones de la aplicación.

- **4. Resultado de la aplicación**: En este apartado se presentan los resultados del proyecto a través de un recorrido detallado por la aplicación web desarrollada. Se incluyen capturas de pantalla y descripciones de las principales funcionalidades implementadas, mostrando cómo los usuarios pueden interactuar con la aplicación para llevar a cabo sus tareas. Con este tour visual se puede apreciar el producto final y de qué forma cumple con los objetivos establecidos.

- **5. Conclusiones y trabajos futuros**: La memoria concluye con una reflexión sobre los logros alcanzados y las lecciones aprendidas durante el desarrollo del proyecto. Se resumen los resultados obtenidos y se evalúa el grado de cumplimiento de los objetivos planteados inicialmente. Además, se plantean posibles mejoras y extensiones para la aplicación web que podrían enriquecer y ampliar la funcionalidad de la aplicación.

- **6. Bibliografía**: En este apartado se enumeran todas las fuentes de información consultadas y utilizadas a lo largo de todo el proyecto. La bibliografía incluye consultas de dudas específicas en foros, artículos, documentación técnica y otros recursos que han sido de mucha ayuda para el correcto desarrollo de la aplicación y de la memoria. Cada referencia se cita adecuadamente en el apartado, permitiendo al lector acceder a las fuentes originales para obtener información adicional.

2. Definición del problema

2.1 Requisitos funcionales y no funcionales

Para desarrollar una aplicación web eficiente para la gestión de los agentes comerciales, hay que dejar claros los requisitos funcionales y no funcionales, ya que puede ayudar mucho a saber que se espera de la aplicación y dan una guía para desarrollarla.

Requisitos Funcionales:

1. Gestión de Usuarios:
   Aunque la aplicación sea de uso casi exclusivo para el agente comercial, debe poder dar la opción de crear, editar y eliminar usuarios con acceso a la aplicación, pero estos usuarios tendrán unos roles (administrador, cliente, fabricante) y unos permisos específicos para controlar las funcionalidades que pueden usar y lo que pueden ver. Estos usuarios deben poder iniciar sesión con un nombre y una contraseña que serán proporcionados por el agente comercial.

2. Gestión de Pedidos:
   Uno de los aspectos más importantes en el trabajo de un agente comercial son los pedidos que le hacen los clientes para que los elaboren los fabricantes, así que deben poder crear, editar y eliminar estos pedidos.
   Deben incluir información sobre el cliente que lo solicita, el fabricante que se va a hacer responsable, los productos que quiere el cliente, sus precios y el estado en el que se encuentra el pedido (validado, pendiente, preparando, enviando, enviado, cancelado), ya que este va a ir cambiando en función de cómo vaya avanzando y saber cómo va. Cada pedido debe tener un historial de cambios de estado, en el que se detalla el usuario que ha hecho el cambio, el día que se ha hecho y a qué estado ha cambiado.
   Con todos los datos del pedido se generará una plantilla totalmente personalizada al gusto del agente comercial, y se podrá imprimir y descargar en el ordenador directamente desde la aplicación.

3. Gestión de Catálogos:
   Los fabricantes deben poder subir y gestionar sus catálogos de productos en formato PDF. Los agentes deben poder también añadir y acceder a todos estos catálogos y utilizarlos para enseñarlos a los clientes y poder realizar los pedidos. Estos catálogos se quedarán guardados en una galería de archivos multimedia en la aplicación, donde se podrán ver todos los archivos subidos y filtrarlos por contextos.

4. Gestión de Eventos:
   En la aplicación se debe poder registrar, editar y eliminar todos los eventos que tenga el agente comercial. En el inicio de la aplicación habrá un calendario en el que aparecerán todos los eventos registrados y se podrá pinchar en ellos para ver toda la información sobre ellos. Este calendario tiene cuatro vistas posibles (mes, semana, día, lista).
   Todos los días a las 8:00 de la mañana se lanzará un comando para comprobar si hay algún evento al día siguiente, en cuyo caso se enviará un recordatorio al correo del agente recordándole los eventos del día siguiente.

5. Correos electrónicos:
   La aplicación va a mejorar la comunicación del agente con los clientes y fabricantes, automatizando el envío de emails a estos para notificar los cambios de estado en los pedidos.
   Como un pedido tiene muchos estados, se crearán unos formularios en los que el agente comercial podrá editar siempre que quiera el asunto y el contenido del email, y la aplicación automáticamente recogerá la plantilla del estado en cuestión y enviará el email directamente. En estas plantillas se podrán poner Palabras clave (señalizadas justo al lado del formulario) que en el email se cambiarán por valores del pedido, como por ejemplo [identificador].

6. Página de inicio (dashboard):
   En la primera página que aparecerá nada más iniciar sesión, aparte del calendario de eventos, estarán seis datos importantes que el agente quiere ver rápidamente, que son el total de ventas (en euros) del mes pasado, las ventas que van este mes, los catálogos activos, los pedidos que no han acabado ni están cancelados, los clientes en activo y los fabricantes en activo.
   Al pinchar en cada uno de ellos hará una función. Si se pincha en el de las ventas del mes pasado, se generará un historial de las ventas del mes pasado, con datos como el mes, el año, el total, pedidos terminados y pedidos cancelados. Estos historiales se podrán ver en una pestaña de la aplicación llamada Ventas. Si se pincha en los otros llevará a los listados correspondientes, pero con los filtros activos para mostrar solo los activos (en el de ventas del mes actual te lleva al listado sin filtros).

Requisitos No Funcionales:

1. Seguridad:
   La aplicación debe tener controles de acceso basados en los roles para asegurar que solo los usuarios autorizados puedan acceder a ciertas funcionalidades. El agente comercial (ROLE_ADMIN) puede hacer uso pleno de la aplicación menos añadir plantillas para los emails, ya que para hacer que esa plantilla se usa, se requiere de código, pero si se necesita alguna plantilla se comunica y se añade.

Los usuarios de fabricantes (ROLE_FABRICANTE) solo tendrán acceso a sus catálogos y a sus pedidos, pudiendo editar y añadir catálogos (sin poder ponerlos en activo para que el agente pueda ver si se han añadido nuevos), y pudiendo editar el estado, descargarlos e imprimirlos, pero no añadir pedidos.
Los usuarios de clientes (ROLE_CLIENTE) solo podrán ver, descargar e imprimir sus pedidos. Ninguno de estos dos usuarios verá la página de inicio descrita antes, solo verán un listado con lo que pueden hacer en la aplicación

2. Usabilidad:
   La interfaz de usuario debe ser intuitiva y fácil de usar. Usando un estilo de colores afines a la empresa del agente comercial. El diseño debe ser responsivo para garantizar una buena experiencia en dispositivos móviles.

3. Escalabilidad:
   La arquitectura del proyecto debe permitir la implementación de nuevas funcionalidades en el futuro de forma fácil.

2.2 Diagrama de Casos de uso

El diagrama de casos de uso es una herramienta útil para ver de una forma más simplificada las posibles interacciones entre los usuarios y la aplicación. A continuación, se presenta un diagrama de casos de uso que muestra a grandes rasgos todas las interacciones que un agente comercial puede realizar con la aplicación [5]. A la aplicación pueden acceder más usuarios aparte del agente comercial, pero él tiene permiso a todas las acciones posibles, por lo que el único actor del diagrama será el agente comercial. Se entrará más en detalle sobre las interacciones permitidas a clientes y fabricantes más adelante.

El agente comercial introduce sus credenciales en el inicio de sesión y seguidamente aparecerá la pantalla de inicio en donde estarán las interacciones directas, las cuales se representan con la flecha continua. A raíz de esas interacciones aparecerán otras interacciones específicas que se pueden realizar, las cuales se representan con las flechas discontinuas.

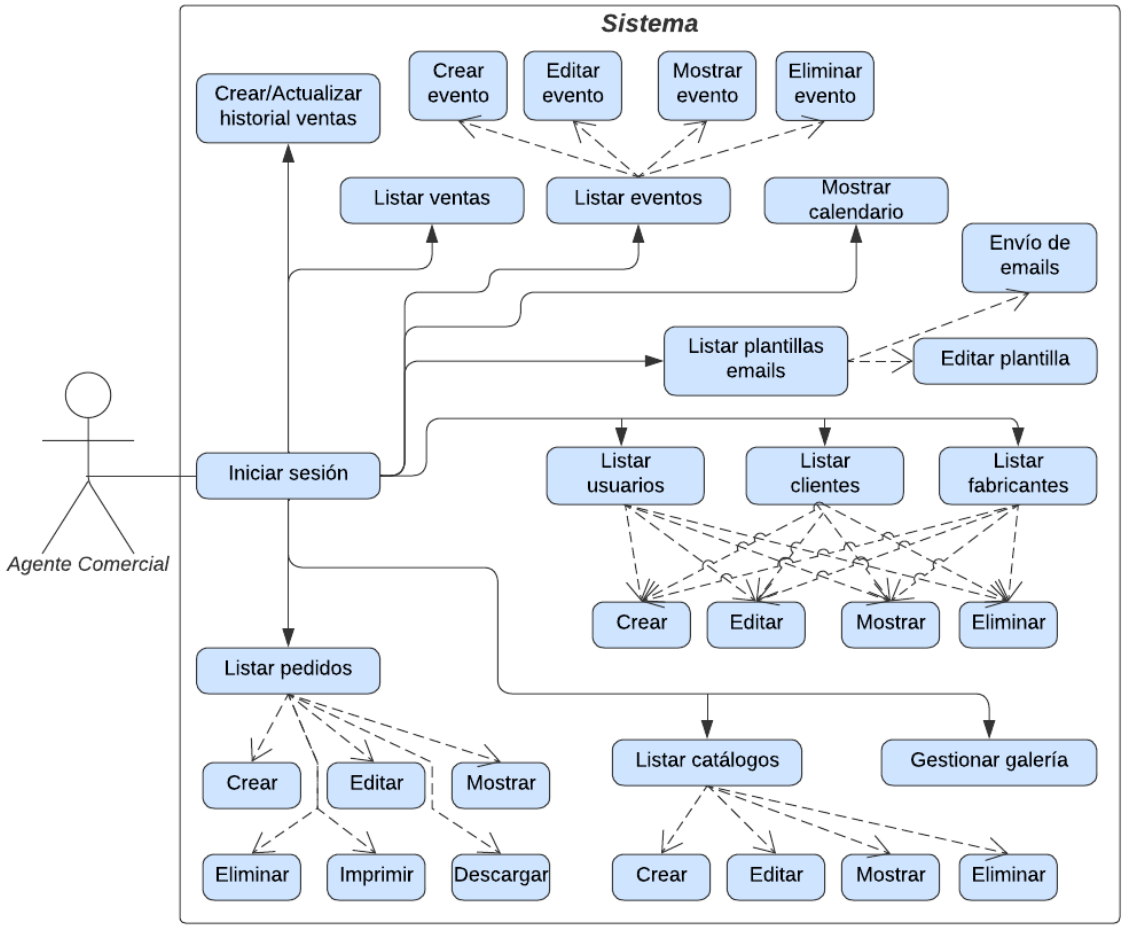


Figura 1. Diagrama de casos de uso

2.3 Diagramas de Secuencia

Con el diagrama de casos de uso se han podido ver todas las interacciones posibles entre un usuario y la aplicación, pero de una forma muy directa, así que aquí se van a presentar dos diagramas de secuencia que van a mostrar de una forma más detallada que es lo que ocurre en cada momento a medida que el usuario va interactuando con la aplicación [5]. Pero en este caso se va a mostrar desde la perspectiva de otro tipo de usuario para poder tener una mayor claridad sobre cómo funciona la aplicación.

En estos diagramas aparece el actor que lleva a cabo la interacción (cliente o fabricante), la propia aplicación, la base de datos, una línea discontinua que indica el tiempo que está pendiente ese objeto, unos tramos grises sobre la línea discontinua que indican los momentos en los que se activa el objeto, y las interacciones, que se representan con dos flechas distintas. La flecha con la punta rellena indica mensaje síncrono (mensaje que espera respuesta) y la flecha con punta delgada un mensaje asíncrono (mensaje que no espera respuesta).

La primera perspectiva es la de un cliente del agente comercial con la interacción de descargar un pedido.

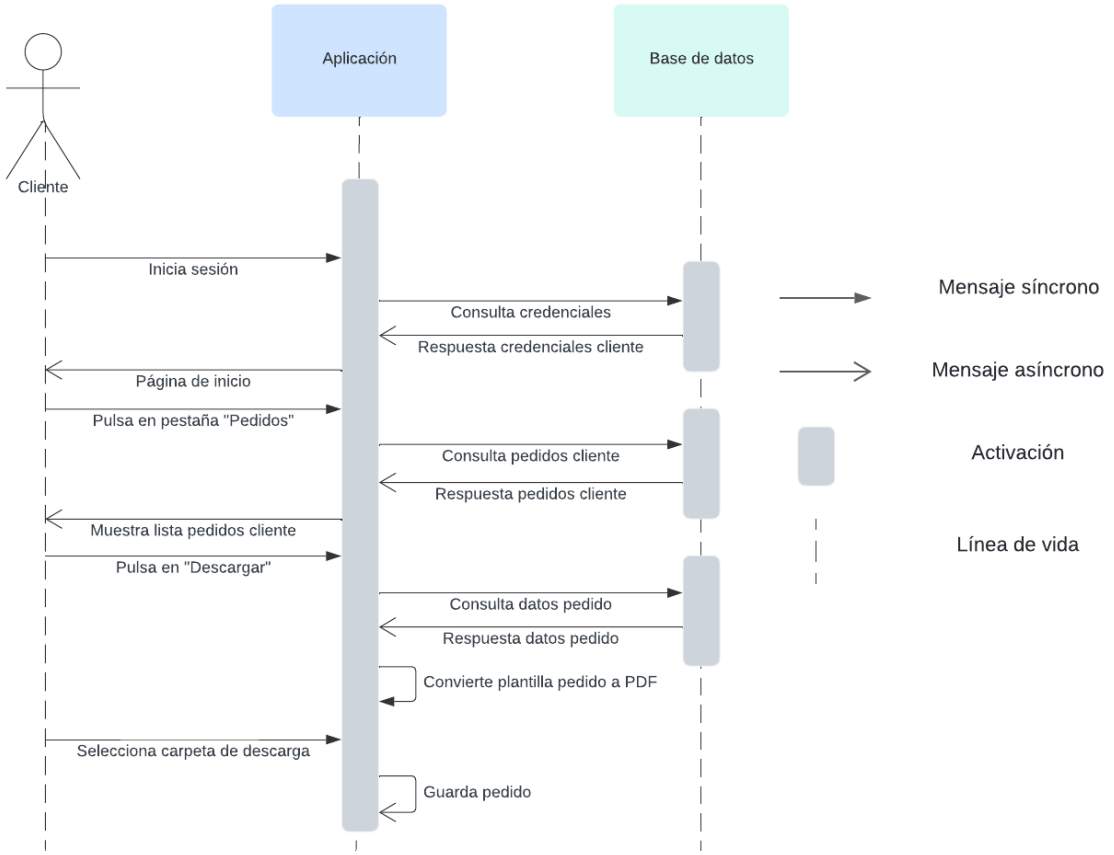


Figura 2. Diagrama de secuencia cliente

El segundo diagrama de secuencia será desde la perspectiva de un fabricante el cuál quiere añadir un nuevo catálogo a la aplicación (en este diagrama se usarán los mismos componentes que en el anterior):

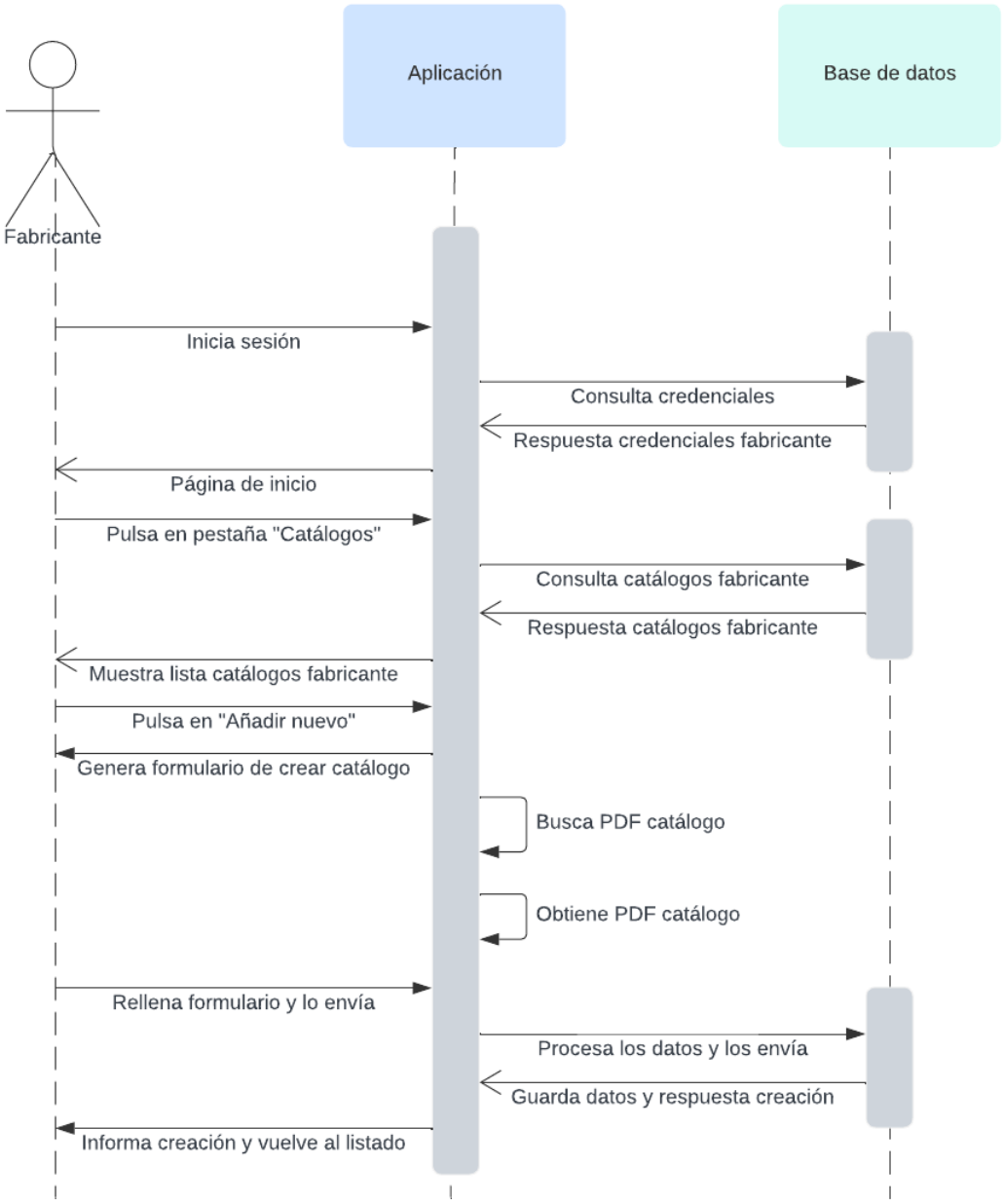


Figura 3. Diagrama de secuencia fabricante

Hay muchas interacciones entre el usuario y la aplicación, pero con estos dos diagramas junto con el diagrama de casos de uso se puede conseguir una idea bastante clara de cómo funciona la aplicación a grandes rasgos.

Con este apartado se ha proporcionar un análisis detallado del problema y se ha podido plantear una base para el diseño y desarrollo de la solución propuesta, así como las interacciones que tendrán los distintos usuarios en la aplicación.

3. Solución del problema

En este apartado se describe el diseño de la solución propuesta, incluyendo la arquitectura general con la que se ha construido la aplicación, las tecnologías que se han utilizado y una descripción detallada de las partes principales que componen la aplicación.

3.1 Arquitectura de la aplicación

Para el desarrollo de la aplicación se ha utilizado el framework de PHP Symfony en su versión 5 [12]. Este framework sigue el patrón Modelo-Vista-Controlador (MVC), que separa la lógica de negocio, la interfaz de usuario y el control de flujo en tres componentes distintos:

- **Modelo (Model)**: Representa la lógica de negocio y maneja el acceso a la base de datos.

- **Vista (View)**: Es el componente de presentación que muestra los datos al usuario mediante unas plantillas.

- **Controlador (Controller)**: Gestiona la comunicación entre el modelo y la vista, maneja las solicitudes HTTP y devuelve las respuestas HTTP.

A continuación, se muestra un diagrama de flujo que enseña cómo funciona Symfony cuando se recibe una petición HTTP:

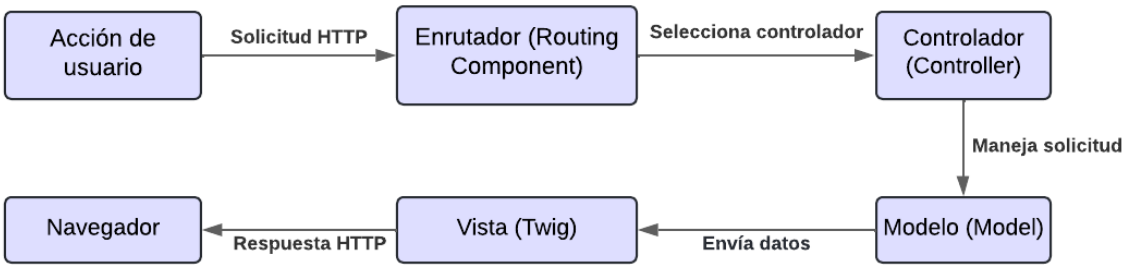


Figura 4. Diagrama Symfony

Cuando un usuario envía una solicitud HTTP (acceder al inicio de la aplicación por ejemplo), Symfony analiza la solicitud HTTP y determina el controlador adecuado basado en la configuración de enrutamiento definida en los archivos de configuración como 'routes.yaml'. El controlador correspondiente maneja la solicitud y puede interactuar con el modelo para recuperar datos de la base de datos y pasarlos a la

vista. Para manejar la vista se usa el motor de plantillas Twig, que recibe los datos del controlador y genera el HTML que se enviará al navegador. Finalmente, Symfony devuelve la respuesta HTTP que ha generado la vista al navegador.

3.2 Tecnologías utilizadas

Para el desarrollo de esta aplicación, se han seleccionado varias tecnologías que permiten cumplir con los requisitos funcionales y no funcionales establecidos. A continuación, se describen las tecnologías principales utilizadas:

- **Symfony 5**: Es un framework de PHP muy popular de código abierto óptima para la creación de aplicaciones web robustas y escalables. Symfony proporciona una estructura organizada para el proyecto y componentes reutilizables y muy potentes que optimiza el desarrollo de la aplicación. Aunque no es la última versión disponible, se ha elegido la 5 porque es la que más estabilidad tiene por ahora [12].

- **Sonata Admin Bundle**: Es un paquete de Symfony que permite crear interfaces de administración de manera rápida y eficiente. Facilita la gestión de entidades y operaciones CRUD, además de que tiene un gran nivel de personalización con la interfaz y las funciones [6].

- **Sonata User Bundle**: Este paquete extiende las funcionalidades de gestión de usuarios en Symfony, proporcionando un conjunto de herramientas para la administración de roles y permisos. Con él puedo administrar también la página de inicio de sesión. Se integra perfectamente con las dos anteriores tecnologías [8].

- **Sonata Media Bundle**: Este paquete me permite gestionar los diferentes archivos que se suben a la aplicación. También se integra perfectamente con las tecnologías anteriores y da la posibilidad de subir no solo PDF, también imágenes (svg, png, jpg, …) para alguna posible función futura [7].

- **Doctrine ORM**: Es un mapeador objeto-relacional (ORM) para PHP con el que configuro las interacciones con la base de datos. Permite mapear clases PHP a tablas de la base de datos y simplifica las operaciones CRUD.

- **MySQL**: Sistema de gestión de bases de datos relacional utilizado para almacenar los datos de la aplicación. Es de código abierto y estoy bastante familiarizado con él.

- **Twig**: Un motor de plantillas para PHP utilizado en Symfony para generar la interfaz de usuario de manera eficiente. Permite mezclar el código HTML con la información de la base de datos [13].

- **FullCalendar**: Una librería JavaScript utilizada para mostrar eventos en un calendario interactivo. Es bastante ligera y ofrece un gran abanico de posibilidades al configurarlo [2].

- **Dompdf**: Es una biblioteca de PHP que me permite convertir documentos HTML a formato PDF. Es muy ligero y se integra muy bien con Symfony y soporta CSS.

- **Leaflet**: Es una biblioteca de código abierto desarrollada en JavaScript que se usa para generar mapas en aplicaciones web. Es muy ligera y bastante eficiente, tiene compatibilidad con todos los navegadores principales y con dispositivos móviles, aparte de que tiene muchas opciones de personalización [4].

- **SMTP de Gmail**: Gmail proporciona un servidor SMTP con tu cuenta para que se puedan enviar correos desde aplicaciones web u otros servicios externos. Me permite una conexión segura (con usuario y contraseña) y encriptación TLS. Para poder utilizarlo hay que tener activado la Verificación en dos pasos y crear una contraseña de aplicación. Es gratis, pero tiene la limitación de un máximo de 100 destinatarios y 500 correos al día, aunque no se llegará a este límite.

3.3 Modelo

Este componente es el encargado de la gestión de la persistencia y recuperación de los datos. Se encarga de interactuar con la base de datos, asegurando que los datos se almacenan y se recuperan de manera eficiente y segura.

Estructura:

- Entidades:
  Las entidades son clases que sirven para configurar los campos en la base de datos y sus relaciones. En aplicaciones web, las entidades se utilizan para mapear las tablas de la base de datos a objetos en el código.
  A continuación, se puede ver un diagrama ERD de la base de datos para mostrar todas las tablas creadas, con todos los campos que las conforman (con el tipo de dato que es) y las relaciones con las otras tablas [14]. Los campos que el nombre acaba por Id representan la relación con la otra entidad, y quiere decir que la entidad que tiene este campo, es la que solo puede estar con un elemento de la otra entidad. Esto se puede ver también con las líneas de las relaciones, la entidad con la raya es la que puede tener muchos elementos de la otra entidad:

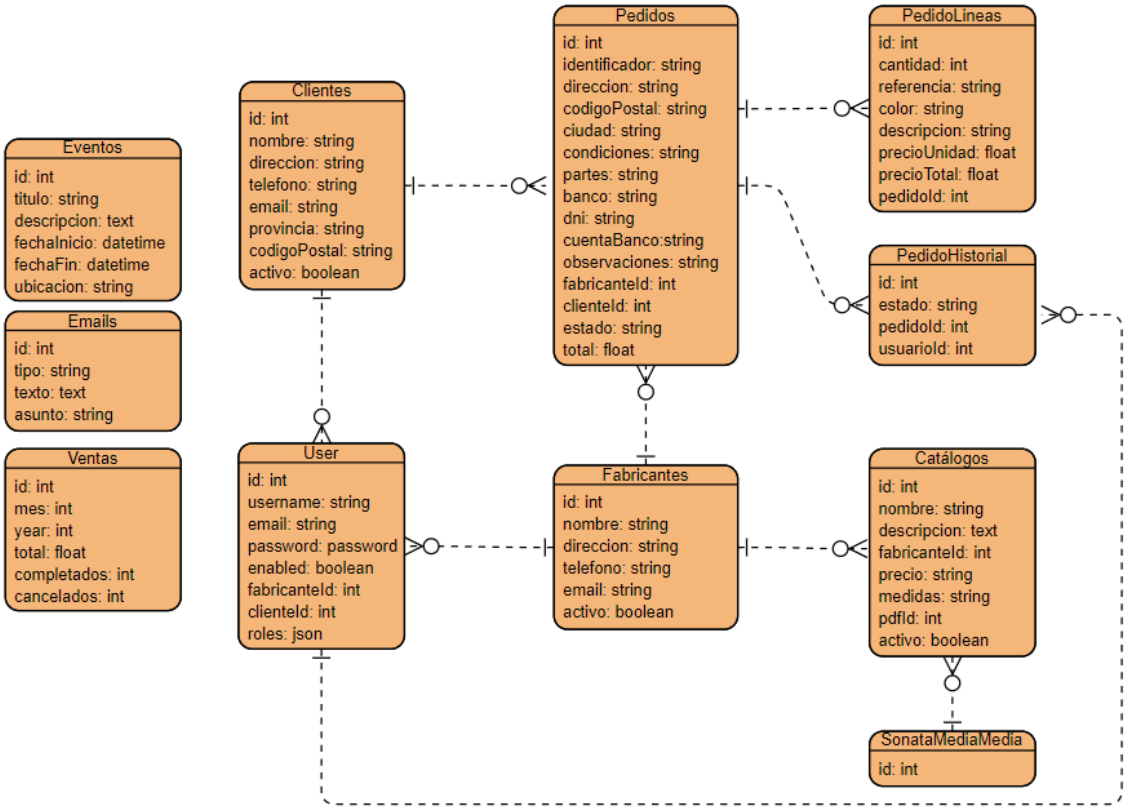


Figura 5. Diagrama base de datos

- Repositorios:
  Los repositorios son componentes que se encargan de la interacción con la base de datos. Proporcionan métodos para realizar operaciones CRUD (crear, leer, actualizar, eliminar) y consultas personalizadas. Los repositorios facilitan el acceso y manipulación de los datos, y cada entidad tiene un repositorio adjuntado.

- Doctrine (ORM):
  Doctrine es una herramienta que permite mapear las entidades del negocio a tablas de la base de datos, facilitando el trabajo con bases de datos relacionales mediante objetos.

Desarrollo:

1. Para conectar la aplicación con la base de datos creada primero se debe de crear un usuario que tenga permisos para editar la base de datos. En este caso se ha creado el usuario *admin* con la contraseña *admin*. La base de datos que se utiliza se llama representaciones y utiliza la versión 8.0 de MySql. Con el usuario ya creado se debe de configurar la URL de la base de datos que se va a utilizar desde el archivo .env del proyecto. Esta URL tiene esta forma:

   DATABASE_URL="mysql://admin:admin@localhost:3306/representaciones?serverVersion=8.0&charset=utf8mb4"

   La base de datos se ha configurado con línea de comando a través del cmd.

2. Para añadir tablas a la base de datos se han utilizado las entidades. En estos archivos se añaden todos los campos que deben de tener las tablas y un método get y set por cada campo para poder obtener y editar los valores de estos. En algunos casos, como es el ejemplo de la entidad 'Clientes.php', se pueden crear métodos personalizados para funciones específicas. En este caso se ha creado el método getProvinciasLista() para que al crear un cliente en la aplicación se obtengan todas las provincias de España y se pueda seleccionar una. En estas entidades hay campos que sirven de relación con otras entidades, y permiten conectar las tablas entre sí para una consulta de información más rápida y para desarrollar funciones nuevas. Este es el caso de la entidad 'Pedidos.php', la cual está relacionada con las entidades 'PedidoLineas.php' y 'PedidoHistorial.php'. Estas dos entidades están relacionadas con los pedidos para poder extender las posibilidades de estos, ya que me permite tener un historial para cada uno y poder añadir todos los productos que sean necesarios al pedido con sus datos específicos. La entidad del historial de pedidos también está relacionada con la entidad 'User.php', puesto que en cada historial se registra qué usuario ha realizado el cambio. A su vez, la entidad 'User.php' está relacionada con las entidades 'Clientes.php' y 'Fabricantes.php', ya que, quitando el usuario del agente comercial, los demás usuarios deben de tener un cliente o un fabricante asignados para saber qué pueden ver y a qué tiene acceso en la aplicación. Se ha creado la entidad 'Catálogos.php', que está relacionada con los fabricantes para saber a quién pertenece ese catálogo, pero también está relacionada con la entidad 'SonataMediaMedia.php', que es una entidad que se debe de crear para que Sonata Media pueda guardar el PDF en la galería [7]. También hay varias entidades que no están relacionadas con ninguna otra entidad, puesto que para desarrollar sus funciones solo se necesita su entidad. Estas entidades son 'Eventos.php', 'Emails.php' y 'Ventas.php'.
   Todas las relaciones que hay entre entidades son *OneToMany*. Esto quiere decir que, en la relación, un elemento de una de las entidades puede estar relacionado con muchos elementos de la otra entidad, pero un elemento de esta entidad solo puede estar con uno de la otra. Como ejemplo se puede ver la entidad 'Pedidos.php' con la entidad 'PedidoHistorial.php', ya que un pedido puede tener muchas líneas de historial, pero una línea de historial solo puede estar en un pedido.

3. Para poder hacer consultas a la base de datos existen los repositorios, en los que se han creado diversas consultas especiales para devolver datos más específicos. Se ha creado y asignado un repositorio para cada una de las entidades existentes, pero hay algunos repositorios con consultas especiales, como en el 'EventosRepository.php', en el que se ha creado la consulta findByDayeRange(). A esta consulta se le pasan dos fechas (inicio y fin) y te devuelve los eventos que haya entre esas dos fechas. O en el 'PedidosRepository.php', en el que se han creado las consultas totalPedidosMes() y pedidosActivos(). A la primera consulta

también se le pasan dos fechas y te devuelve todos los pedidos que estén entre esas fechas y no estén cancelados. La segunda consulta devuelve todos los pedidos que no estén ni cancelados ni enviados.
En estas consultas se debe de configurar las condiciones que deben de cumplir los elementos para que se devuelvan y el valor de los parámetros que se usan en esas condiciones.
Las demás consultas que se usan en el proyecto son consultas de filtrar por campos de la entidad.

4. La conexión entre la base de datos y las entidades se lleva a cabo con Doctrine, ya que, al crear una entidad en el proyecto, para que se cree la tabla y se haga persistente se tienen que usar unos comandos. El primero es 'php bin/console make:migration' para crear una migración con todos los datos que se van a subir a la base de datos (estas migraciones se pueden revisar en la carpeta *migrations* del proyecto). Estas migraciones se quedan registradas para poder volver a una versión anterior de la base de datos. Y el segundo comando es 'php bin/console doctrine:migrations:migrate', que sube la última migración creada a la base de datos y se hace persistente.

3.4 Vista

Este componente representa la interfaz en la que van a aparecer todos los datos y funcionalidades con las que van a interactuar los usuarios. En esta aplicación, la vista está desarrollada utilizando Twig y CSS, lo que garantiza una experiencia de usuario moderna y responsiva para que se pueda usar perfectamente desde el móvil.

Estructura:

- Plantillas Twig: Este tipo de plantillas son utilizadas para generar las páginas HTML de manera eficiente y mantenible [13]. Facilita la separación de la lógica de presentación de la lógica de negocio, permitiendo que el código HTML sea más limpio y fácil de mantener. Cada página de la aplicación está representada por una plantilla Twig que define la estructura y el contenido de la página.
  Twig tiene su propia sintaxis, con la que se pueden mostrar las variables que le pasa el controlador ({{ variable }}), hacer operaciones como bucles o extensiones de otras plantillas ({% for %}) o hacer comentarios ({# Comentario #}).
  Para poder usar este motor de plantillas primero se configura un controlador en Symfony para manejar la solicitud HTML y una vez listo se usa el método render() para generar el contenido. Esto llama al motor Twig que cargará la plantilla que se ha indicado en el método render(). Se rellenará con los datos pertinentes, obtenidos a través del controlador, y se enviará el HTML completo al cliente. Aquí se muestra un diagrama para explicar mejor el proceso.

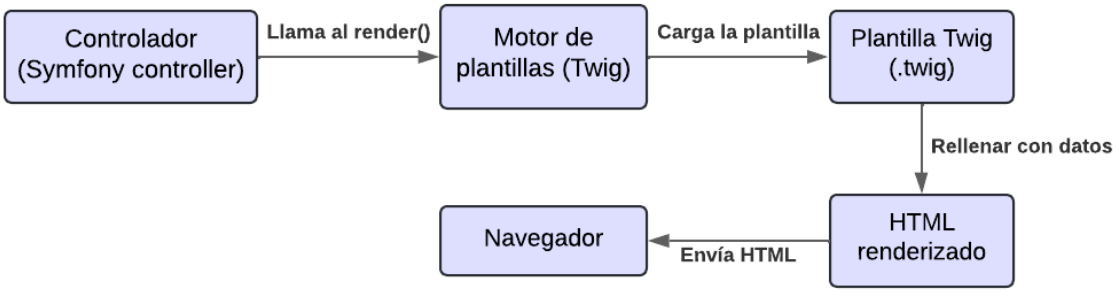


Figura 6. Diagrama Twig

- CSS: Significa Cascading Style Sheets u Hojas de Estilo en Cascada, es un lenguaje utilizado para describir la presentación de documentos HTML o XML. Mientras que HTML se encarga de la estructura y el contenido de la página web, CSS define el estilo que van a tener esos elementos cuando se muestren por pantalla en la aplicación.

A continuación, se muestra un mapa de navegación en el que se puede como se salta entre las distintas páginas de la aplicación:

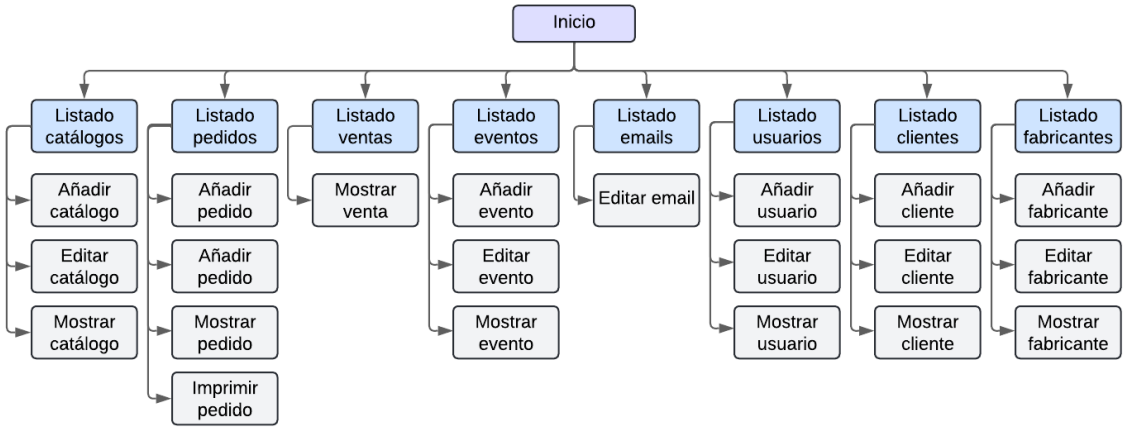


Figura 7. Mapa de navegación

Listado de archivos:

Admin

catalogo_pdf_list_field.html.twig
dashboard.html.twig
descargar_pedido.html.twig
estados_pedido.html.twig
evento.edit.html.twig
imprimir_pedido.html.twig
mapa_evento.html.twig
pedido_historial.html.twig
pedido_lineas.html.twig
plantilla_pedido.html.twig

show_pdf.html.twig
standard_layout.html.twig
tipos_email.html.twig

Dashboard
datos_dasboard.html.twig
index.html.twig

Email
recordatorio_evento.html.twig
variables_email.html.twig

base.html.twig

Desarrollo:

Estas plantillas Twig se han ido utilizando para distintos fines:

1. Para generar los correos de recordatorios de eventos e información de cambios de estado de pedidos. Para lo primero se ha creado la plantilla 'recordatorio_evento.html.twig', la cual tiene un elemento <p> en el que hay un texto con una variable que se refiere a la fecha del día de los eventos, y un bucle for (el cual recorre todos los eventos obtenidos para el día siguiente) con otro elemento <p> con el título del evento y la hora de inicio.
   La segunda plantilla creada relacionada con los correos es 'variables_email.html.twig', que es para los correos de aviso de cambio de estado. El agente comercial tendrá a su disposición un editor de texto para cada estado del pedido, en el que escribirá el contenido del email [10]. Como se ha explicado antes, el controlador obtiene ese contenido y lo pasa a la plantilla, donde mostrará el contenido, pero con el filtro *replace* para cambiar las Palabras clave por los valores del pedido.

2. Para editar la forma en la que se visualizan algunos elementos en los listados o añadir alguna funcionalidad. Una de estas plantillas es 'catalogo_pdf_list_field.html.twig', en la que se edita uno de los datos del listado de un catálogo para que sea un enlace a otra página del navegador, donde se podrá ver el PDF del catálogo perfectamente (se hace uso de un if para comprobar que haya un PDF, si no se mostrará el texto *No PDF*).
   Otras dos plantillas de este estilo son 'estados_pedido.html.twig' y 'tipos_email.html.twig', en las que se comprueban cuál es el estado del pedido o del email y se edita el texto del estado para que aparezca rodeado de un color especial (por ejemplo, verde para el estado *Validado*). Para dar estos colores se usa un if, y dependiendo del estado entrará en una condición donde estará un <td> para crear el campo en el listado, y dentro de él un con clases distintas (*label-success* para Validado o *label-danger* para Cancelado).

También están las plantillas 'descargar_pedido.html.twig' e 'imprimir_pedido.html.twig' que son las que se han creado para incluir en el listado de pedidos las funcionalidades de descargar e imprimir el pedido. Básicamente, son elementos <a> en los que su enlace (href) es una ruta creada y configurada para cumplir la función concreta.
El archivo del pedido que se va a poder descargar o imprimir se va a generar a partir de una plantilla llamada 'plantilla_pedido.html.twig'. Esta plantilla ha sido estructurada y editada en base a las preferencias del agente comercial. En esta plantilla se aprecia una cabecera con información del agente comercial, unas líneas con los datos generales del pedido y una tabla con la lista de productos del pedido. Se ha añadido una parte de JavaScript para que, al seleccionar la opción de imprimir, se active la función de imprimir la plantilla.

3. Para configurar campos en los formularios de editar/crear, como es el caso de la plantilla 'evento.edit.html.twig'. En esta plantilla se añade un mapa del mundo en el formulario, el cual se podrá manejar para encontrar la ubicación donde se celebrará el evento y seleccionar esa ubicación, rellenando automáticamente el campo del formulario *Ubicación* y dejando la marca en el mapa. Para conseguir esto, esta plantilla se ha extendido a la plantilla base 'base_edit.html.twig' (generada por Sonata para tener la base de los formularios en la aplicación), ya que se necesita editar el bloque *stylesheets* de esta plantilla para añadir el css de la biblioteca Leaflet y el css propio [4]. Pero no solo eso, sino que se tiene que editar el bloque *javascripts*, donde se configurará el código JavaScript para crear el mapa personalizado. En este JavaScript se configuran los atributos del mapa (zoom, ubicación en la que aparece y la marca de la ubicación), una función para guardar la longitud y latitud en el campo de la ubicación al pinchar en el mapa, y una función para poner la marca si se está editando el evento y ya hay una ubicación seleccionada. Por último se edita el bloque *sonata_tab_content* para añadir un elemento con *id="map"*, que es donde se generará el mapa.
   Cuando se extiende de otra plantilla y se edita un bloque de dicha plantilla, si se quiere mantener el contenido del bloque para añadir algo, se pone {{ parent() }}, y esto dice a Twig que se respete el contenido de la plantilla de la que se extiende.

4. Para editar la visualización de algunos campos en la pantalla de mostrar datos de pedidos, eventos o catálogos. Empezando por los eventos, se crea la plantilla 'mapa_evento.html.twig' para mostrar el mapa con la marca de la ubicación. Esta plantilla es muy parecida a la anterior, con algunas modificaciones en el HTML y CSS, pero esta vez no se extiende de ninguna otra plantilla, por lo que no se editan bloques. El JavaScript se modifica un poco, ya que solo se requiere mostrar el mapa con la marca.
   Para los catálogos se ha creado la plantilla 'show_pdf.html.twig', que es un elemento <iframe> al que se le pasa como origen la referencia en la galería del PDF del catálogo. Esto permite poder ver el PDF incrustado en esta página.

Y para los pedidos se crean dos plantillas, 'pedido_historial.html.twig' para mostrar el historial de cambios de estado del pedido, y 'pedido_lineas.html.twig' para mostrar los productos del pedido. Estas dos plantillas con elementos <table> que mostrarán los datos necesarios en cada caso. En el historial se muestra el estado, el usuario y la fecha, y en los productos se muestra la cantidad, referencia, color, descripción, precio de unidad y precio total. Estas tablas tienen un <thead> que define las columnas de la tabla con los nombres de los datos, y un <tbody> con los valores de cada producto.

5. Para editar la página de inicio de la aplicación se han creado dos plantillas, una para los datos relevantes que aparecen al inicio, llamada 'datos_dashboard.html.twig', y la otra para el calendario de eventos, llamada 'index.html.twig'. Estas dos plantillas van a ser llamadas desde otra plantilla llamada 'dashboard.html.twig', que extenderá de la plantilla con el mismo nombre que genera Sonata para iniciar la aplicación. Esta plantilla es la que se ha configurado para que sea la plantilla del inicio. Esto se ha llevado a cabo de esta forma para poder distinguir qué se va a visualizar dependiendo del usuario que entre a la aplicación, ya que si es el agente comercial se llamará a las dos plantillas anteriores con un render(), si no, se mostrará la lista de acciones disponibles.
   La plantilla de los datos (datos_dasboard.html.twig) está compuesta por dos a modo de líneas, en las que hay otros tres en cada una. Estos contienen un elemento <a> para que al pinchar en ellos se ejecute la acción específica del recuadro, y dos , uno con el título del dato y otro con el valor del dato.
   La plantilla del calendario (index.html.twig) es parecida a la del mapa en el evento. Esta extiende de la plantilla 'base.html.twig', y edita el bloque *body* para crear un elemento con *id="calendar"* donde se mostrará el calendario. También edita el bloque *javascripts,* donde se importará y configurará el código de la librería FullCalendar con la que se generará el calendario [2]. Algunos de los atributos con los que se ha configurado el calendario son que aparezca por defecto con la vista del mes, que se muestren las semanas justas del mes, el tamaño del calendario, como se organiza la cabecera del calendario y con un bucle for se van añadiendo los eventos que pasa el controlador a la plantilla.

6. Por último comentar que se crea la plantilla 'standard_layout.html.twig', que se extiende de la plantilla con el mismo nombre. Esa plantilla es por así decirlo la plantilla madre de la aplicación, así que se extiende de esta plantilla para editar el bloque *stylesheets* y poder incluir el archivo 'admin.css', que es donde están casi todas las reglas css de la aplicación. Este archivo se ha creado en el directorio *public/css* para que el navegador pueda acceder a él.

El resultado de todas estas plantillas se podrá ver en el apartado 4.

3.5 Controlador

Este componente es la parte que da vida a las aplicaciones web. Es responsable de gestionar la lógica de la interacción entre los datos, la lógica del usuario y la presentación. En una aplicación web que utiliza Twig y CSS, los controladores se encargan de procesar las solicitudes del usuario, gestionar los datos y preparar la información que se enviará a la vista. Pero para preparar la información no solo se utilizan controladores, puesto que también se pueden usar otros archivos para distintas funcionalidades, como es el caso de los admins o los servicios.

Estructura:

- Admins:
  Estos archivos son los encargados de generar la página de administración para las entidades, con los formularios de edición/creación, los listados y las páginas de mostrar los datos. Para ello este archivo se divide en varios métodos en los que se configurarán los campos de la entidad que aparecerán en cada caso y como agruparlos:
    - FormMapper: Para los formularios de creación y edición.
    - ListMapper: Para los listados de los elementos (aquí se decide también qué acciones se podrán realizar con los elementos).
    - ShowMapper: Para la página de mostrar los datos del elemento.
    - DatagridMapper: Se eligen los campos por los que se puede filtrar el listado.
    - ConfigureQuery: Sirve para crear consultas con las que poder controlar que se va a mostrar en el listado (por ejemplo, para que un cliente solo vea sus pedidos.

  Estos archivos deben de registrarse en el archivo de configuración 'services.yaml' y llamar desde el archivo 'sonata_admin.yaml' para que se puedan mostrar. En este último archivo es donde se decide la estructura del menú de la aplicación y los grupos que lo forman.

- Controladores:
  Los controladores son componentes que gestionan las solicitudes HTTP entrantes, procesan los datos necesarios y determinan qué vista se debe renderizar. Son los intermediarios entre el usuario y la aplicación. En Symfony, los controladores son métodos de clases que responden a rutas específicas.

- Servicios:
  Los servicios son funciones específicas que pueden ser reutilizados en diferentes partes de la aplicación. En este caso incluye la lógica para enviar los emails y los recordatorios. Son un buen componente para la reutilización, manteniendo el código modular y escalable.

- Comandos:
  Los comandos son componentes parecidos a los controladores, pero en vez de responder a rutas tienen un nombre, el cual se usa en la línea de comandos para ejecutarse y realizar la lógica de negocio que tienen en su función. También se puede programar la ejecución de un comando sin necesidad de usar la línea de comandos.

- Eventos (listeners):
  Los Listeners son componentes que se ejecutan en el tramo de tiempo entre que se envía un formulario en la aplicación y se guardan los datos en la base de datos. Dependiendo de la función que uses, la lógica de negocio que describe se ejecutará en un momento concreto u en otro.
- Ahora se va a mostrar la estructura principal del proyecto:

```
.
|-> config
|   |-> packages
|   |   |-> …
|   |-> routes
|   |   |-> …
|   |-> routes.yaml
|   |-> services.yaml
|-> migrations
|   |-> ...
|-> public
|   |-> css
|   |-> img
|   |-> uploads
|   |   |-> media
|   |   |   |-> …
|   |-> index.php
|-> src
|   |-> Admin
|   |   |-> ...
|   |-> Command
|   |   |-> ...
|   |-> Controller
|   |   |-> ...
|   |-> Entity
|   |   |-> ...
|   |-> EventListener
|   |   |-> ...
|   |-> Form
|   |   |-> ...
|   |-> Repository
|   |   |-> ...
|   |-> Services
|   |   |-> ...
|   |-> ...
|-> templates
|   |-> admin
```

```
|   |   |-> ...
|   |-> dashboard
|   |   |-> ...
|   |->email
|   |   |-> ...
|   |-> security
|   |   |-> ...
|   |-> ...
|-> var
|   |-> log
|   |   |-> ...
|-> .env
|-> ...
```

Listado de archivos:

Admin
- CatalogosAdmin.php
- ClientesAdmin.php
- EmailsAdmin.php
- EventosAdmin.php
- FabricantesAdmin.php
- PedidosAdmin.php
- UserAdmin.php
- VentasAdmin.php

Command
- RecordatoriosEventosCommand.php

Controller
- Dashboardcontroller.php
- PedidosController.php

EventListener
- PedidoListener.php

Form
- PedidoLineasType.php

Services
- Mailer.php

Desarrollo:

Se va a ver cuáles han sido los archivos creados de cada tipo:

1. No todas las entidades tienen un Admin asociado, ya que por ejemplo la entidad *PedidoLineas* existe para la entidad *Pedidos*, y se maneja de una forma diferente. Pero quitando algunas, la mayoría de entidades tienen un Admin asociado para que se puedan administrar en la aplicación.
   Dependiendo de la entidad se usarán unos métodos u otros de los nombrados arriba, pero la estructura de los archivos es la misma para todos. Uno de los

Admins es el 'PedidosAdmin.php', en el que se han usado los cinco métodos. En FormMapper se han elegido los campos que aparecerán, el nombre asociado que tendrán, el tipo de campo que es en el formulario y en qué grupo o pestaña aparecerán, si es que se crean. La entidad *PedidoLineas* está relacionada con la entidad Pedidos por un campo, y este campo es especial, ya que debe hacer uso de un Form llamado 'PedidoLineasType.php'. Este Form es un archivo creado para dar forma al formulario de *PedidoLineas* (en vez de usar un Admin, se usa un Form), de esta forma se puede incrustar el formulario de esta entidad en el Admin de *Pedidos* y poder añadir productos al pedido. Entonces se le indica al campo que hace de relación con la entidad *PedidoLineas* que es un *CollectionType*, diciéndole cuál es el formulario que debe usar y si se permite ir añadiendo elementos.

En el ListMapper se indican los campos que aparecerán en el listado y las acciones posibles con los elementos, que en este caso son editar, mostrar, imprimir y descargar. Como se ha explicado en el apartado 3.3, se han usado plantillas para editar campos en el listado, y este es el caso de las dos últimas acciones, ya que son creadas desde cero. Estas acciones tendrán una ruta personalizada (registradas en el archivo de configuración 'routes.yaml') que llamará a un método en algún controlador para ejecutar la acción. Para que un campo use una plantilla específica se le indica con el atributo *template*.

En el ShowMapper es más de lo mismo, se eligen los campos que se mostrarán y la estructura. Aquí también se usan plantillas específicas para mostrar algunos campos, como los productos o el historial.

En ConfigureQuery se hacen consultas a la base de datos para obtener los pedidos correspondientes al usuario.

Con la explicación de este Admin se ha podido comprobar que es lo que se ha ido configurando en el resto de Admin. Todos tiene sus peculiaridades, como el 'CatalogosAdmin.php', en el que se ha usado una plantilla para editar el campo del PDF en el listado, el 'EmailsAdmin.php', en el que se ha prohibido crear nuevos elementos, o el 'EventosAdmin.php', en el que se ha editado el campo de la ubicación en el formulario y en la página de mostrar para que aparezca el mapa. Los demás que se han creado son 'ClientesAdmin.php', 'FabricantesAdmin.php', 'VentasAdmin.php' y 'UserAdmin.php', pero este último es un poco especial, ya que se ha tenido que sobrescribir. Esto es porque Sonata User crea su propio Admin, pero se ha querido editarlo para adaptarlo a la aplicación [8]. Esto se ha conseguido registrando en el archivo de configuración 'sonata_user.yaml' que se debe usar 'UserAdmin.php' como Admin.

2. Par poder configurar las acciones especiales en la aplicación se han creado dos controladores. El primero es el 'Dashboardcontroller.php', en el que se han configurado tres métodos relacionados con la página de inicio de la aplicación:
    - Método index(): Es el que renderiza la plantilla del calendario. Se hace una consulta a la base de datos para obtener en un array todos los

eventos del agente comercial. Seguidamente, se llama al render('dashboard/index.html.twig') pasándole el array con los eventos para que puedan aparecer en el calendario.

- Método datosDashboard(): Es el que renderiza la plantilla de los datos del inicio. La primera parte del método es obtener el mes y año actual y el mes anterior. Con estos datos se crea la primera fecha y la última de los dos meses (teniendo en cuanta si tiene 28, 29, 30 o 31 días). Se llama a una consulta personalizada (totalPedidosMes) a la que pasándole una fecha inicio y una final, devuelve el total de pedidos no cancelados en ese intervalo (a esta consulta se le llama dos veces, para el mes pasado y el actual). Con un foreach se va sumando el total de dinero de cada pedido y se guarda en una variable el del mes pasado y en otra el del actual. A continuación, se llama a otra consulta (pedidosActivos) para obtener los pedidos que no están ni cancelados ni enviados, y se guarda el número en otra variable. La última parte del código es obtener el número de catálogos, fabricantes y clientes activos con consultas y guardar estos datos en variables. Una vez se ha obtenido toda la información, se hace el render('dashboard/datos_dashboard.html.twig') y se le pasan las variables obtenidas.
- Método crearVenta(): Es el que ejecuta una de las acciones al pinchar en uno de los recuadros de datos del inicio, en específico el primero (el del total del mes pasado). Se trata de que al pinchar se genere automáticamente un historial de ventas del mes pasado, y para que se ejecute este método, el enlace del recuadro será una ruta que se ha creado en el archivo de configuración 'routes.yaml' (llamada *crear_venta*), a la que se le pasan dos valores (mes y año) y apunta a este método del controlador. Primero se hace una consulta para obtener las ventas creadas, si existe un historial de ese mes se actualizará, si no, se creará uno nuevo. En este código se obtienen los pedidos activos del mes, los cancelados y el total del mes, junto al mes y año que se le pasan al método. Se guardan estos datos en el objeto creado y se guarda en la base de datos. Para finalizar me recarga la página y muestra un mensaje que avisa de cómo ha ido la acción.

El segundo controlador es 'PedidosController.php', en el que se han creado los métodos para imprimir y descargar los pedidos:

- Método printPedido(): En el que se imprime el pedido. Cuando el usuario pincha en el botón de imprimir, se lanza una ruta creada en 'routes.yaml' (llamada *imprimir_pedido*) a la que se le pasa el id del pedido como valor. En el controlador, se obtiene el valor del id y se encuentra el pedido mediante una consulta. El último paso es llamar al render('admin/plantilla_pedido.html.twig') pasándole el pedido para que se pueda crear el HTML. Una vez se ha creado, se inicia el JavaScript de la plantilla para que se pueda imprimir el pedido desde el navegador.

- Método generatePedidoPdf (): En el que se genera el PDF del pedido para descargarlo. A este método también se le llama por una ruta (llamada descargar_pedido) a la que se le pasa también el id del pedido. Se obtienen los datos del pedido como en el método anterior y se inicializa Dompdf con la opción de tipografía Arial. A continuación, se obtiene el HTML del pedido mediante un renderView() usando la plantilla con los datos, y se carga este HTML en el objeto Dompdf que se ha creado, asignándole el tamaño A4 y la orientación portrait. Lo último que se realiza en renderizar el documento con el render() y se envía el PDF generado al navegador para la descarga. El método devuelve una respuesta del contenido que se va a descargar.

3. Para el envío de los correos se ha creado un servicio llamado 'Mailer.php', cuyo constructor tiene tres argumentos. La interfaz MailerInterface [11], que es la encargada de hacer la propia acción de enviar el correo, la LoggerInterface, que se usa para dejar escrito que es lo que ocurre en el proceso (como los correos se envían de forma automática, si hubiese un error se puede consultar en el archivo *var/log/dev.log*), y el motor Twig, para poder generar el cuerpo de los correos.
   En el servicio se han creado dos métodos para enviar los correos, uno para los avisos de cambio de estado de los pedidos y otro para los recordatorios de los eventos para el agente comercial. En ambos métodos primero se crea el email con los atributos necesarios, que son el correo de quién lo envía (se pone el correo del agente comercial), el correo del destinatario, el asunto del correo y el cuerpo. Para generar el cuerpo se hace uso del render() de Twig para usar la plantilla 'variables_email.html.twig' en el caso de los avisos de estado, y la plantilla 'recordatorio_evento.html.twig' para los recordatorios del agente comercial. Por último, se manejan los posibles errores con la estructura try/catch de PHP, intentando ejecutar el envío del email, y si da algún error, se informa en el log.
   Cada servicio se debe de registrar en el archivo de configuración 'service.yaml', indicando los argumentos que se le pasan al constructor y el archivo del servicio. Para que se pueda enviar los correos desde la aplicación se ha usado el servicio SMTP que ofrece Gmail, y para poder utilizarlo se ha tenido que configurar el DSN desde el archivo .env. Hay que configurar una línea de código que tenga esta forma:

   MAILER_DSN=Gmail://correo@gmail.com:contraseña@default

   Se cambian las palabras clave por los datos de la cuenta de correo del agente comercial y en el archivo de configuración 'mailer.php' se indica que para el email se tiene que usar este DSN [11]. Para mayor seguridad, se ha registrado en el archivo 'services.yaml' que debe de usar como host *smtp.gmail.com* y como puerto el 587.

4. Para poder enviar el recordatorio de eventos al agente comercial se ha creado un comando llamado 'RecordatoriosEventosCommand.php'. Para empezar, cuenta con un constructor al que se le pasan dos argumentos, el repositorio de *Eventos* y el servicio creado Mailer. A continuación, en el método configure() se indica que nombre va a tener el comando (app:enviar-recordatorio-evento) y la descripción del mismo, y lo que va a ejecutar el comando está en el método execute(). Aquí se obtienen la primera hora del día siguiente (0, 0, 0) y la última (23, 59, 59). Se llama a una consulta (findByDateRange) a la que le paso las dos horas para que me devuelva todos los eventos del día siguiente. Si no hay eventos el comando devuelve un mensaje explicando eso, pero si sí hay eventos, se intenta enviar el recordatorio, y si falla se escribe en el log. Tanto si falla como si no, se devuelve un mensaje con la información pertinente.
   Este comando se ha programado desde el equipo para que se lance cada día a las 8:00 a.m. Se ha hecho programando una nueva tarea que lance el comando.

5. Como último componente están los Listeners, que en este caso se ha creado uno, llamado 'PedidoListener.php', para poder realizar varias acciones automáticamente tanto al crear como al editar pedidos. Para empezar, tiene un constructor con dos argumentos. Uno de los argumentos es UsageTrackingTokenStorage, que se usa para poder obtener el usuario que ha iniciado sesión se usa, y el otro argumento es el servicio Mailer.
   En el Listener hay dos métodos, el método preUpdate(), que sirve para hacer una acción justo en el momento entre que se envía el formulario y se guardan los datos del pedido en la base de datos al editar un pedido. El método prePersist(), va a realizar la misma función que el método anterior, pero para cuando se crea un pedido, y no se comprueba si ha cambiado el estado del pedido.
   Lo primero que se hace es obtener los datos que se están enviando en el formulario y comprobar que estos datos corresponden a la entidad Pedidos. Si es un pedido, se obtiene el precio de cada producto del pedido y se va sumando para obtener el total del pedido y guardarlo en un campo. Luego se revisa si ha cambiado el estado del pedido, y si es verdadero se obtiene el usuario que ha iniciado sesión y se crea un objeto de la entidad PedidoHistorial para dejar registrado el cambio de estado en el historial del pedido. Ahora se revisa a qué estado específicamente se ha cambiado, porque dependiendo del estado al que cambie se obtendrá una plantilla u otra, y se enviará el cliente, al fabricante o a ambos.
   Para terminar hay un método más, el postFlush(), que se encarga de hacer permanente en la base de datos las acciones de los otros dos métodos. Para ello se suben los datos de la entidad con persist() y se guardan definitivamente con flush().

4. Resultado de la aplicación

En este apartado se presentan los resultados obtenidos a lo largo del desarrollo del proyecto mediante un recorrido detallado por la aplicación web. Se incluyen capturas de pantalla con descripciones de las principales funcionalidades implementadas, demostrando cómo los usuarios van a poder interactuar con la aplicación para llevar a cabo sus tareas de manera eficiente y más concentrada. En este tour se va a tomar la identidad de un agente comercial en su mayor parte, al final de esta se comprobará lo que verán los clientes y los fabricantes.

4.1 Autenticación

La aplicación comienza con una página de inicio de sesión donde los usuarios pueden introducir sus credenciales para entrar a la administración. Esta página incluye campos para el nombre de usuario y la contraseña. Si por alguna circunstancia un cliente o un fabricante olvidan la contraseña, deberán pedirla al agente comercial. Se habilita también la opción de recordar la sesión para no tener que introducir las credenciales demasiadas veces (el logo es del agente comercial al que se le ha ofrecido hacer la aplicación y con el que se ha colaborado en todo el proceso).

Figura 8. Inicio de sesión

4.2 Página de inicio (dashboard)

Una vez que se ha comprobado que el usuario existe en la base de datos y tiene el rol de administrador, es dirigido al Dashboard, que es la página principal de la aplicación. Aquí, el agente comercial puede ver un resumen de sus eventos recientes y algunos datos relevantes.

Figura 9. Dashboard

Como se puede apreciar en la figura anterior, el Dashboard incluye:

- **Calendario de Eventos**: Un calendario interactivo que muestra todos los eventos registrados en la aplicación. Si el agente lo desea, puede cambiar las vistas del calendario arriba a la derecha del mismo, o cambiar de mes arriba a la izquierda. Cada evento es un enlace a una página con todos los detalles del evento.

- **Datos relevantes**: Son seis recuadros en los que se muestran datos importantes para poder acceder a ellos rápidamente. Dependiendo del recuadro en el que se pinche se generará una interacción personalizada.

- Si se pincha en el primero, se generará un historial del mes anterior con algunos datos como el total de dinero ganado, el mes, el año o los pedidos cancelados en ese mes.
- Pinchando en el segundo recuadro, llevará al agente a la pestaña de Ventas, un listado con todos los historiales de todos los meses pasados:

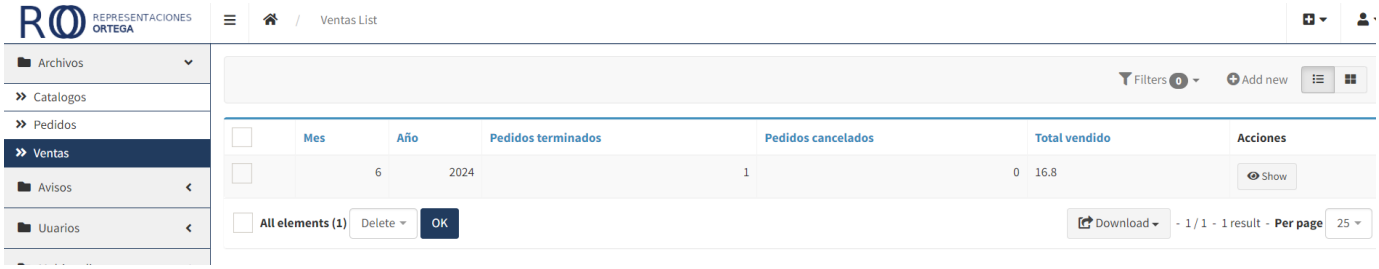


Figura 10. Listado ventas

- Y si se pincha en cualquiera de los otros cuatro recuadros, llevará al agente al listado correspondiente, pero con un filtro activo para que solo se muestren los que están marcados como activos (como ejemplo se muestra el listado de fabricantes con filtro):

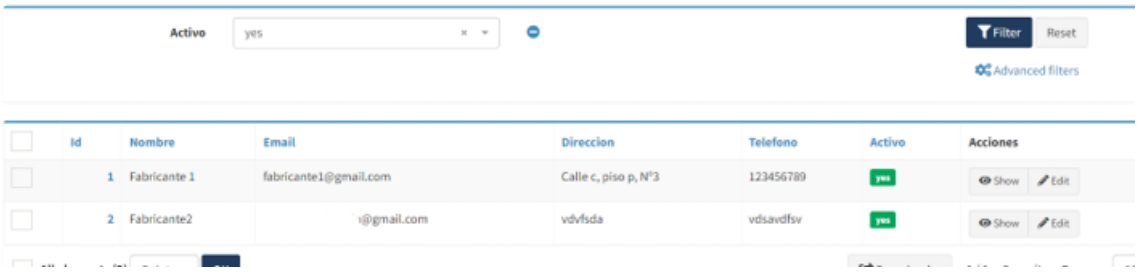


Figura 11. Listado fabricantes filtrado

4.3 Gestión de pedidos

Cuando el agente comercial consiga concretar un pedido con un cliente, deberá de añadirlo en la aplicación para poder empezar todo el procedimiento hasta que le llegue al cliente. El agente irá a la pestaña de Pedidos y agregará un pedido con los datos acordados con el cliente. Para ello pinchará en el botón 'Add new' y aparecerá un formulario con todos los campos necesarios:

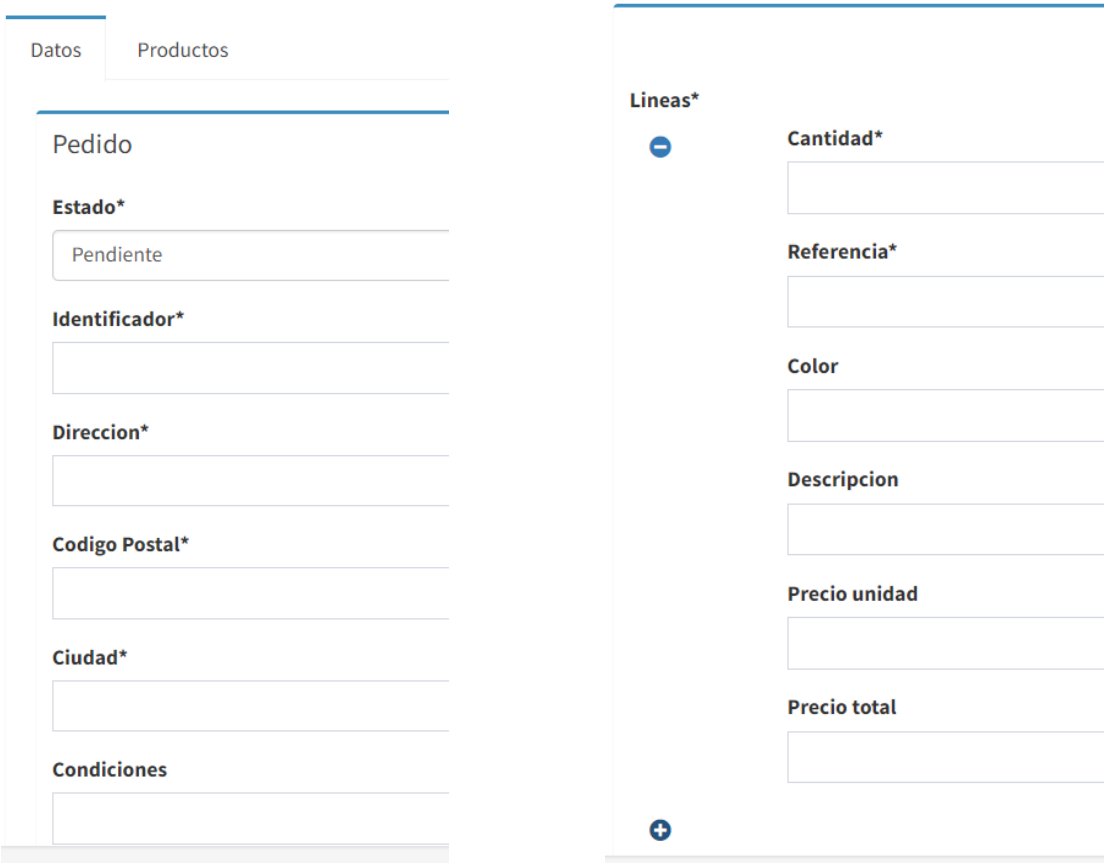
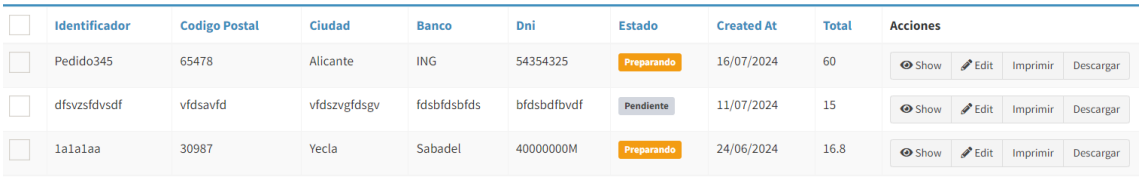


Figura 12. Formulario crear pedido

Se puede apreciar que este formulario consta de dos partes, la de la izquierda (datos) con todos los datos generales del pedido, y la de la derecha (productos), en la que el agente irá añadiendo los productos que conforman el pedido con todos sus datos. Se envía el formulario y la aplicación redirecciona al listado de pedidos para hacer interacciones con ellos.

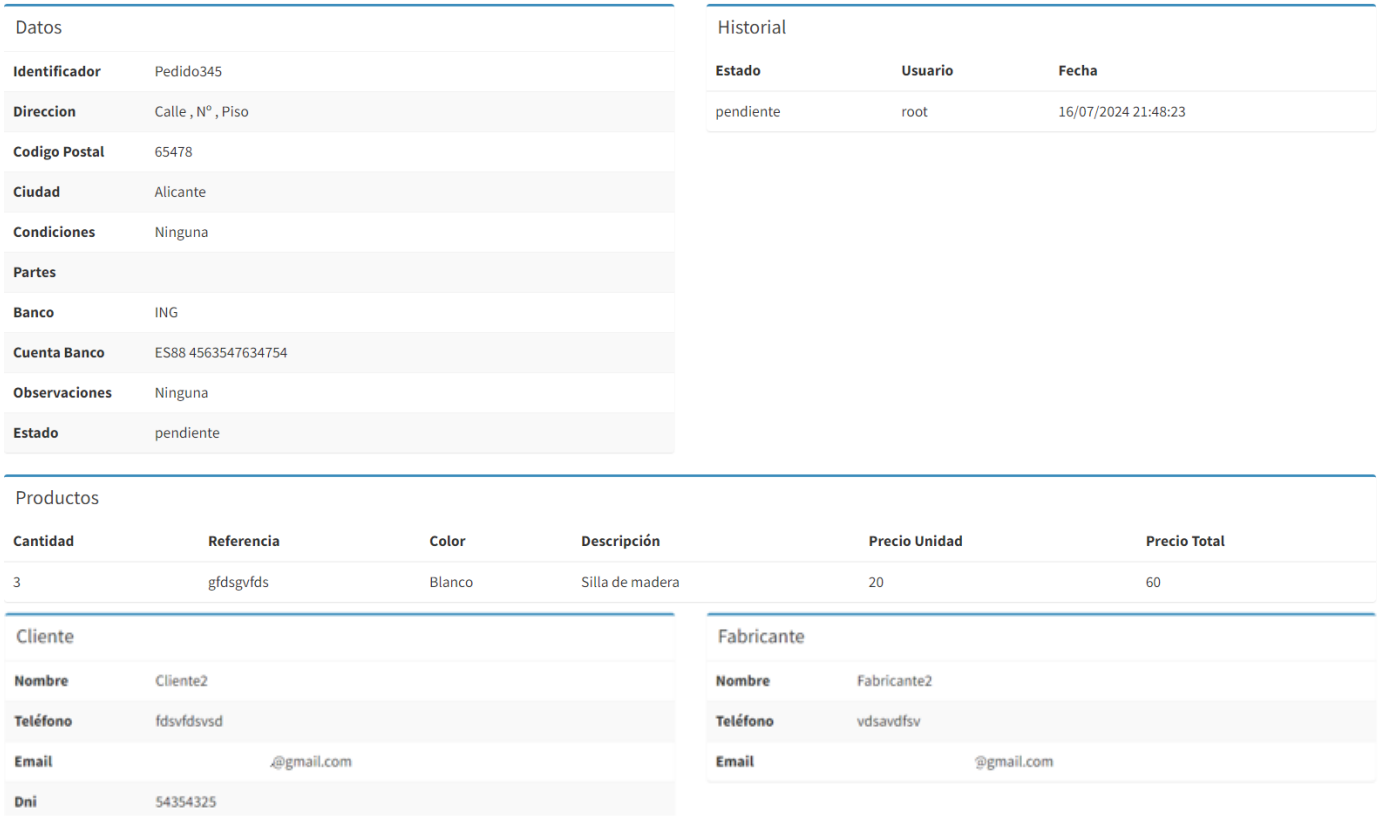


Identificador	Codigo Postal	Ciudad	Banco	Dni	Estado	Created At	Total	Acciones
Pedido345	65478	Alicante	ING	54354325	Preparando	16/07/2024	60	Show Edit Imprimir Descargar
dfsvzsfdvsdf	vfdsavfd	vfdszvgfdsgv	fdsbfdsbfds	bfdsbdfbvdf	Pendiente	11/07/2024	15	Show Edit Imprimir Descargar
1a1a1aa	30987	Yecla	Sabadel	40000000M	Preparando	24/06/2024	16.8	Show Edit Imprimir Descargar

Figura 13. LIstado pedidos

Los listados están ordenados de manera descendente, para que a primera vista aparezcan los elementos más nuevos.

Antes de generar el PDF del pedido, el agente debe asegurarse que están todos los datos correctamente, así que pincha en el botón 'Show' para mostrar una página con todo lujo de detalle sobre el pedido:



Datos

Identificador	Pedido345
Direccion	Calle , N° , Piso
Codigo Postal	65478
Ciudad	Alicante
Condiciones	Ninguna
Partes	
Banco	ING
Cuenta Banco	ES88 4563547634754
Observaciones	Ninguna
Estado	pendiente

Historial

Estado	Usuario	Fecha
pendiente	root	16/07/2024 21:48:23

Productos

Cantidad	Referencia	Color	Descripción	Precio Unidad	Precio Total
3	gfdsgvfds	Blanco	Silla de madera	20	60

Cliente

Nombre	Cliente2
Teléfono	fdsvfdsvsd
Email	@gmail.com
Dni	54354325

Fabricante

Nombre	Fabricante2
Teléfono	vdsavdfsv
Email	@gmail.com

Figura 14. Mostrar pedido

En esta pantalla el agente visualizará los datos generales, los de los productos, los datos del cliente, los datos del fabricante y un historial de cambios de estado del pedido, para saber qué usuario hizo el cambio, a qué estado se cambió y en qué fecha. Este historial

se rellena automáticamente al crear y editar el pedido, aunque solo cuando hay un cambio en el valor del estado.

Ya se ha asegurado el agente comercial de que todo está en orden, así que procede a descargar el pedido. Pincha en el botón de 'Descargar' que hay en el listado y genera la plantilla rellenada con los datos correspondientes:

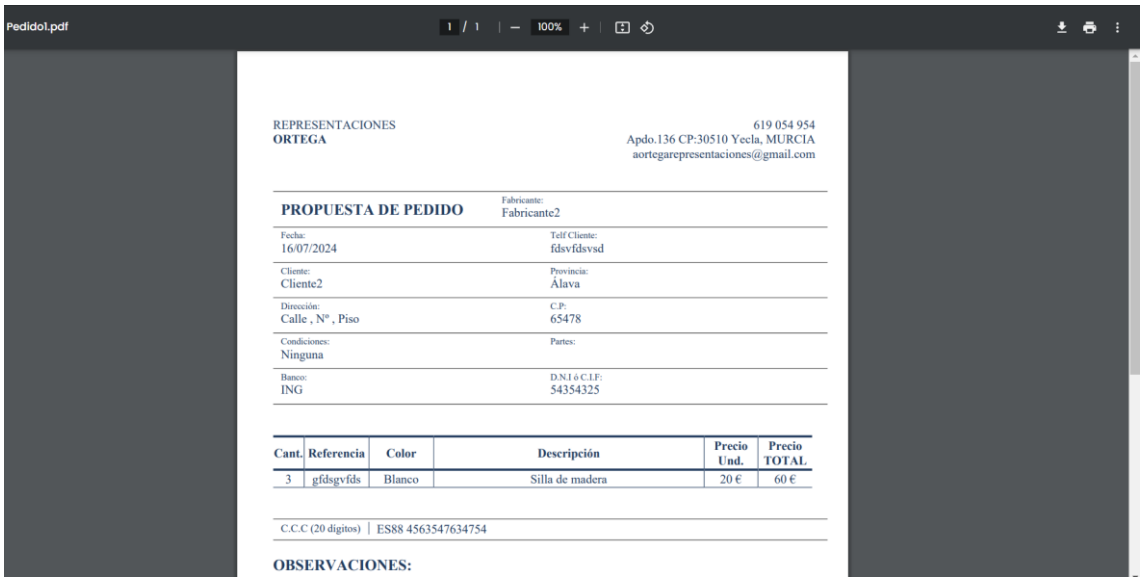


Figura 15. PDF descargado

Si por algún casual el agente no quisiese descargarlo, sino que necesitase imprimirlo, puede ir directamente a la opción de 'Imprimir':

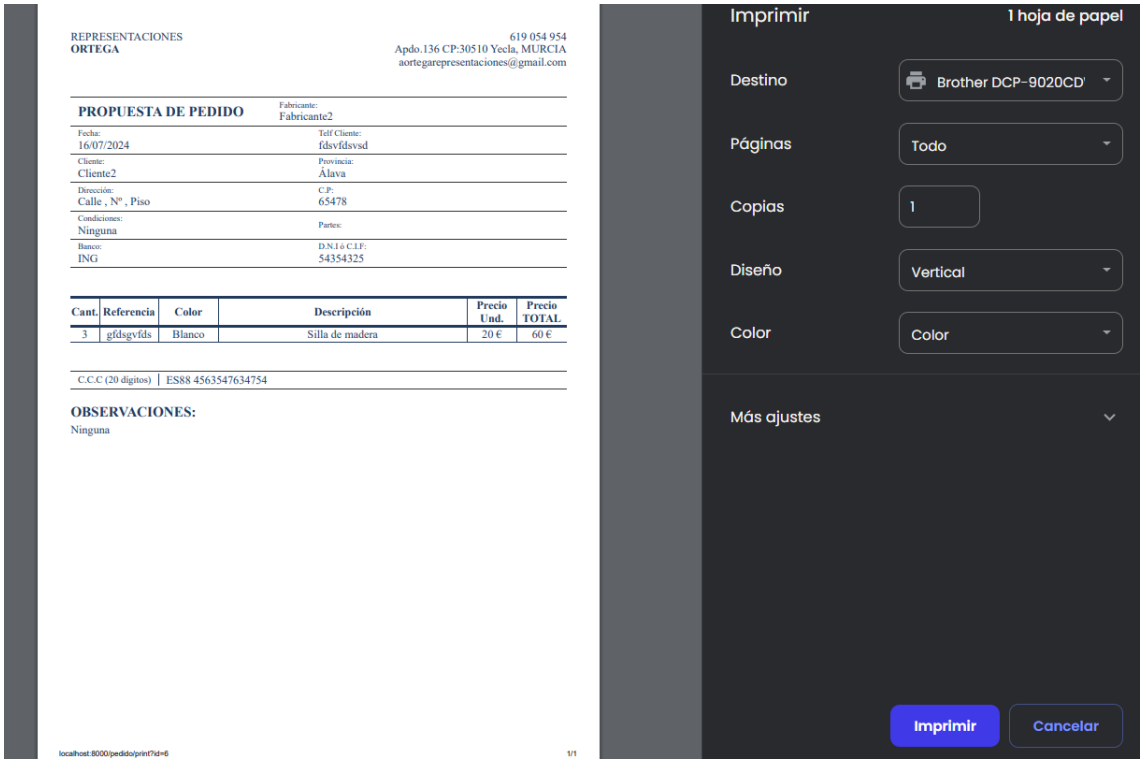


Figura 16. Imprimir pedido

4.4 Gestión de catálogos

Los agentes van a poder añadir y editar todos los catálogos que le proporcionen los fabricantes. Para ello deberá de ir a la pestaña Catálogos y en este caso, darle a 'Edit' para poder modificar alguna información o el archivo del catálogo.

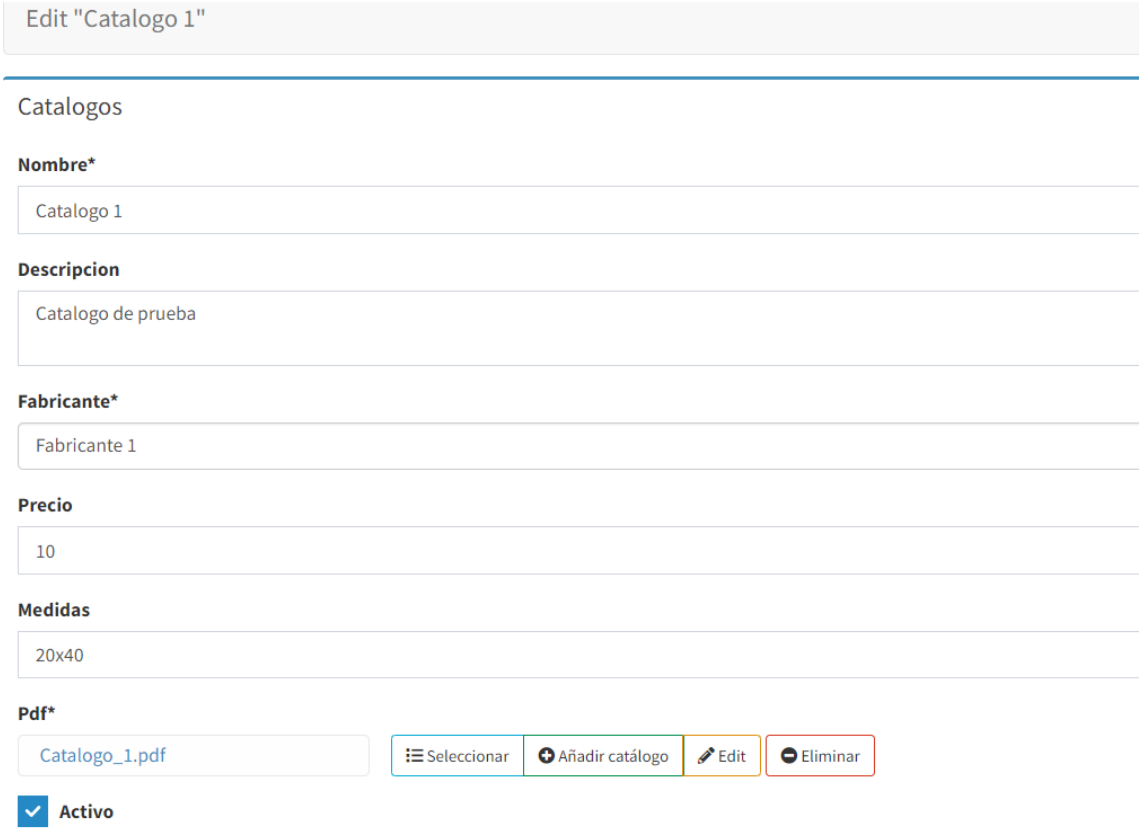


Figura 17. Editar catálogo

Como se puede apreciar en la imagen, es un formulario normal, en el que hay un desplegable para el campo Fabricante en el que se listan todos los fabricantes existentes en la aplicación y seleccionar solo uno.

Todos los campos son importantes, pero el que más peso tiene sin duda es el campo PDF. Se puede ver que con este campo se puede interactuar de varias formas. La primera opción básicamente te permite seleccionar un archivo que ya exista en la galería de la aplicación. La segunda es la más utilizada, ya que permite adjuntar un archivo que esté en tu equipo informático (ordenador, tableta, …). La tercera y la cuarta opción son para cuando ya se ha adjuntado un archivo, puesto que te permiten editar la información del archivo o eliminarlo.

Cuando se añade un nuevo archivo a la aplicación, este se almacena en una galería, a la cual se puede acceder desde la pestaña Galería del menú de la izquierda de la aplicación. Ahí aparecerán todos los archivos subidos, filtrados por el contexto en el que se hayan creado (en este caso el contexto es catálogos):

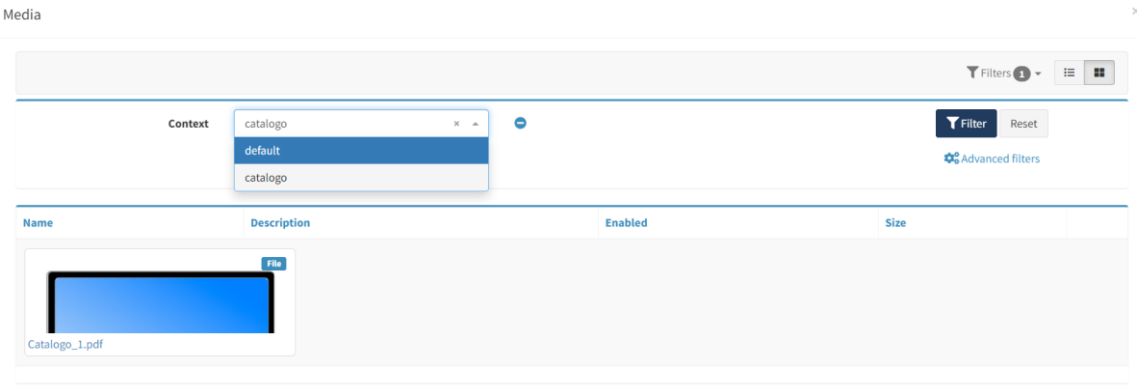


Figura 18. Galería

Para que el agente comercial pueda mostrar el catálogo al cliente sin mucho esfuerzo, se han habilitado dos opciones. En el listado de los catálogos, uno de los campos se ha editado para que sea un enlace a una página nueva en el navegador donde se muestre el PDF, pero si lo prefiere, puede optar por pinchar en el botón 'Show' para ver los datos del catálogo y el PDF en cuestión:

Figura 19. Ver PDF

4.5 Gestión de eventos

Un agente comercial va a estar repleto de eventos como reuniones, recogidas o exposiciones, y todo eso lo debe compaginar con sus tareas y con su vida privada, así que se ha habilitado la opción de que el agente pueda registrar todos estos eventos, con todos los detalles, y que aparezcan en el calendario del inicio. Así podrá organizar mucho mejor su trabajo y podrá ser más eficiente.

Para estos eventos se ha añadido un mapa del mundo para poder señalizar el lugar en el que va a acontecer. Para seleccionar la ubicación al crear o editar el evento, el agente solo tendrá que pinchar en el mapa en el sitio del evento. Cuando el agente vaya a la página para mostrar los datos del evento, aparecerá también el mapa con una marca en la localización, y se podrá navegar por todo el mapa.

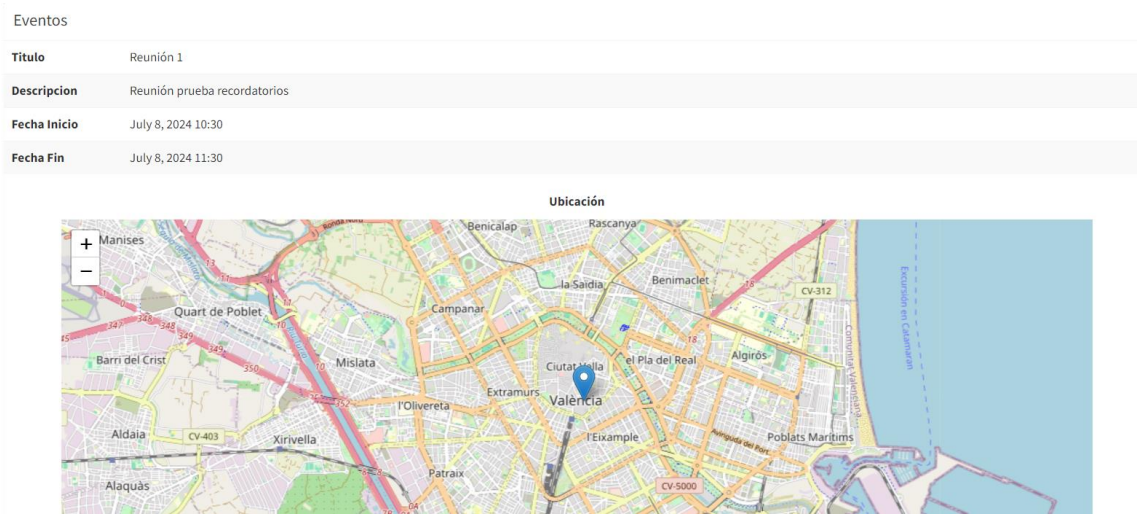


Figura 20. Ver evento

Como se ha estado comentando antes, todos estos eventos aparecerán en el calendario del inicio, y este aparecerá con el modo de visualización del mes, pero se podrá cambiar a cualquiera de los otros:

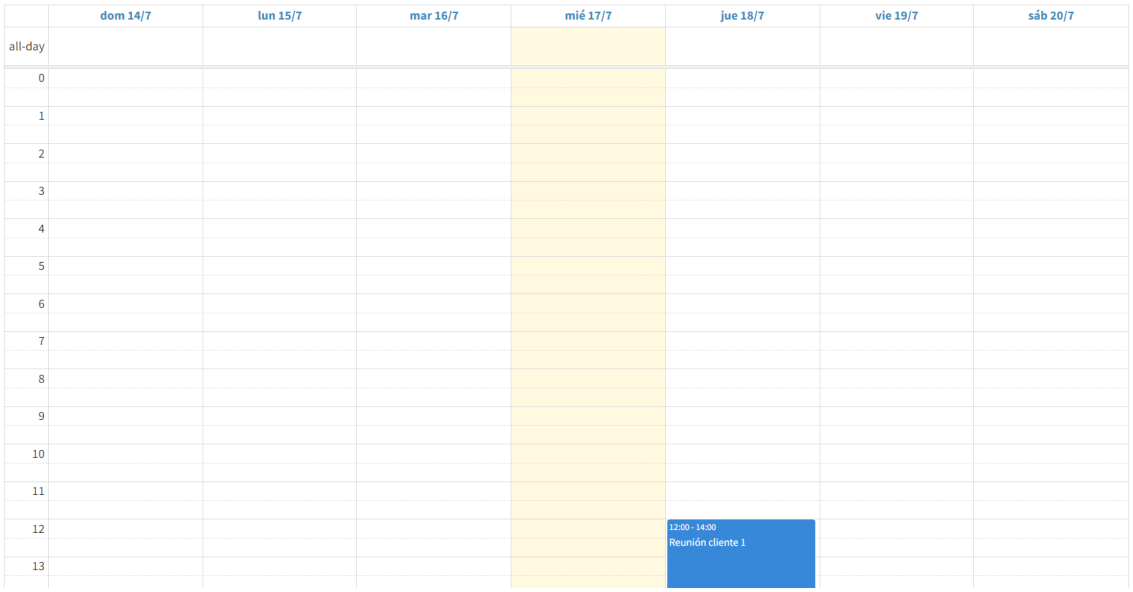


Figura 21. Calendario en semana

Para asegurar que el agente comercial va a recordar los eventos cercanos, se ha creado un comando el cual revisa si existe uno o varios eventos al día siguiente, en cuyo caso

enviará un recordatorio al correo del agente listando los eventos que acontecerán al día siguiente, con el nombre y la hora. Este comando se ejecutará todos los días a las 8:00 de la mañana (para no tener que esperar a las 8 de la mañana del siguiente día, se ha ejecutado el comando desde línea de comandos para comprobar el recordatorio).

Figura 22. Recordatorio

4.6 Gestión de correos

Cuando el agente comercial edita un pedido cambiado su estado, automáticamente se envía un correo al cliente o al fabricante, o a ambos, informando de ese cambio. Pero el contenido de estos correos no es estático, ya que el agente irá a la pestaña Emails y allí encontrará un listado con las plantillas para cada cambio de estado.

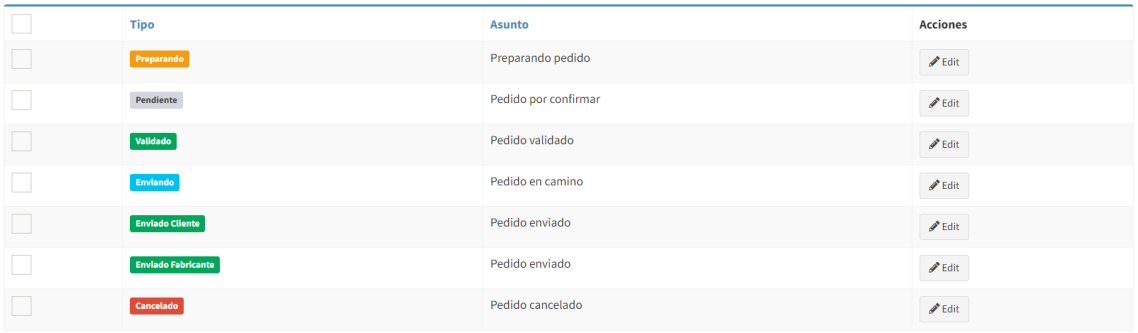


Figura 23. Plantillas email

Cuando necesite cambiar el cuerpo de alguno de estos correos, le dará a 'Edit' en la plantilla concreta y le aparecerá un formulario en el que no podrá cambiar el estado al que pertenece, pero sí el asunto y el cuerpo del correo.

Para que el agente tenga más libertad, se ha configurado el campo del cuerpo con opciones de edición de texto, asemejándose a la edición de un Word. Con esto el correo podrá ser más personalizable y de mejor calidad.

En este texto, se ha configurado la posibilidad de que el agente comercial pueda poner unas Palabras clave, las cuales se muestran justo debajo del editor de texto, para que

cuando se cree el correo para enviarlo, estas palabras sean intercambiadas por valores del propio pedido, como por ejemplo el identificador o el nombre del cliente.

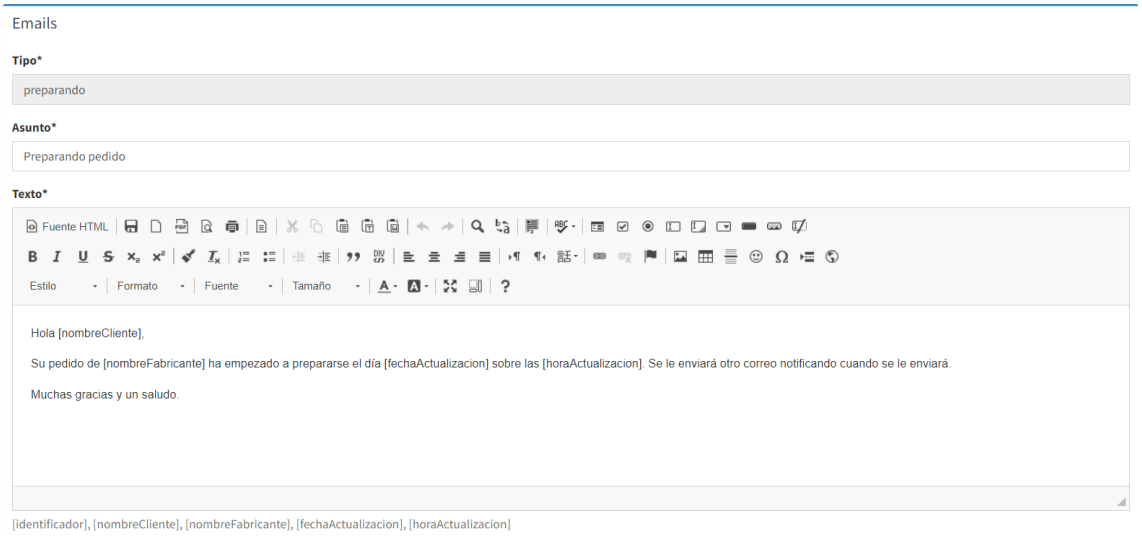


Figura 24. Editar plantilla email

Entonces, cuando el agente cambie un el estado de un pedido, se enviará un correo con la información proporcionada en la plantilla correspondiente.

Así es como se vería el correo con los valores de las Palabras clave:

Figura 25. Email Preparando

4.7 Otros usuarios

A esta aplicación van a tener acceso dos perfiles de usuario más, los clientes y los fabricantes, aunque con muchas más limitaciones que el agente comercial. Para administrar estos dos perfiles se ha añadido una pestaña para cada uno. En ellas el agente comercial irá añadiendo los fabricantes con los que trabaja y los clientes con los que interactúa, cada uno con sus datos.

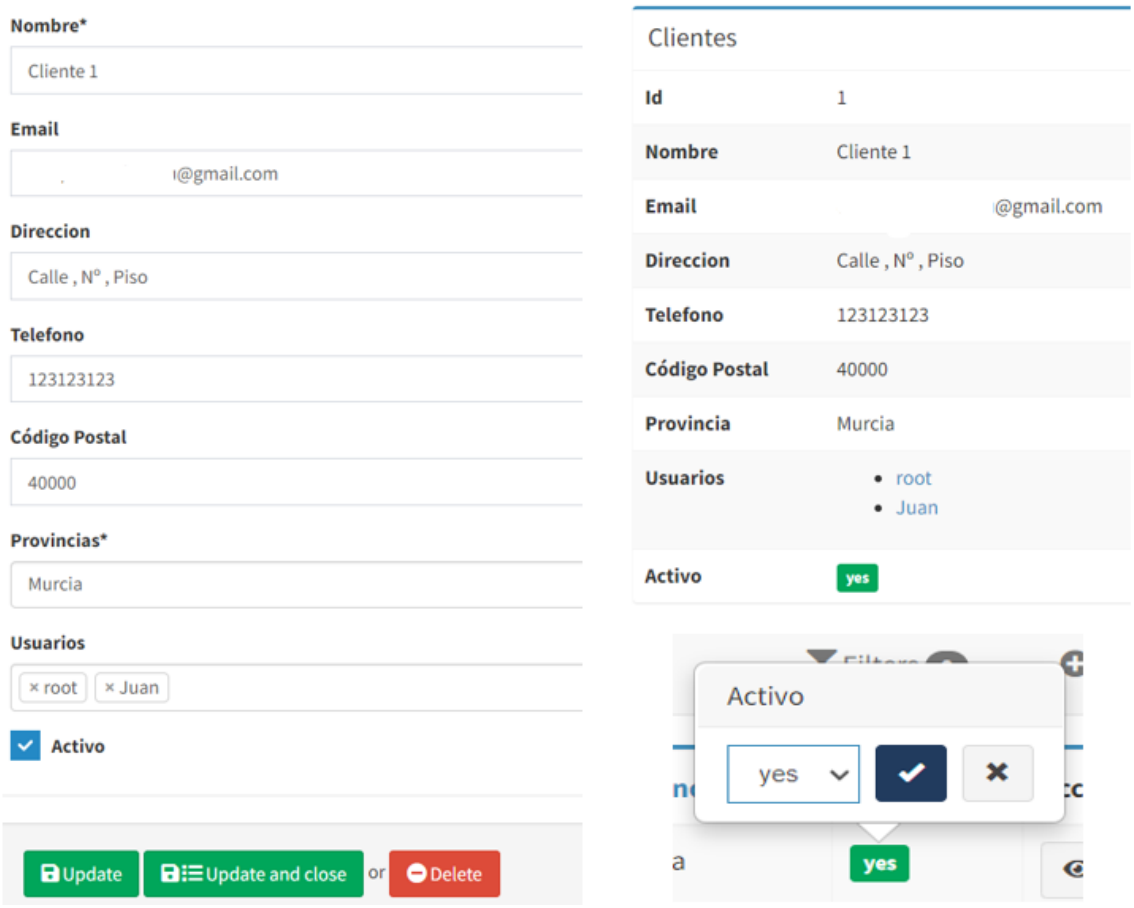


Figura 26. Clientes

Al crear un cliente o un fabricante, se pueden asociar a los usuarios que dé de alta el agente comercial, para que cuando una persona entre con las credenciales de ese usuario, se pueda saber qué pedidos (y catálogos en los fabricantes) se deben listar.

Si un cliente o fabricante ya no está activo se puede editar desde el mismo listado, sin tener que entrar al formulario de edición y enviarlo.

La pestaña de fabricantes es muy parecida a lo que se ha mostrado en la figura 19 sobre los clientes, solo que tienen distintos campos y son para otros usuarios.

Habiendo creado ya algunos clientes y algunos fabricantes, el agente comercial puede crear usuarios para cada uno de ellos con los accesos a la aplicación. El formulario para añadir o editar un usuario da la posibilidad de asignar los roles pertinentes a cada usuario. Ya se han configurado unos roles específicos con los permisos adecuados, pero si en algún caso particular, el agente comercial necesita dar un permiso concreto a un usuario concreto, puede hacerlo con una maya de permisos que aparece en el formulario. Justo debajo de esta maya aparecen los roles mencionados:

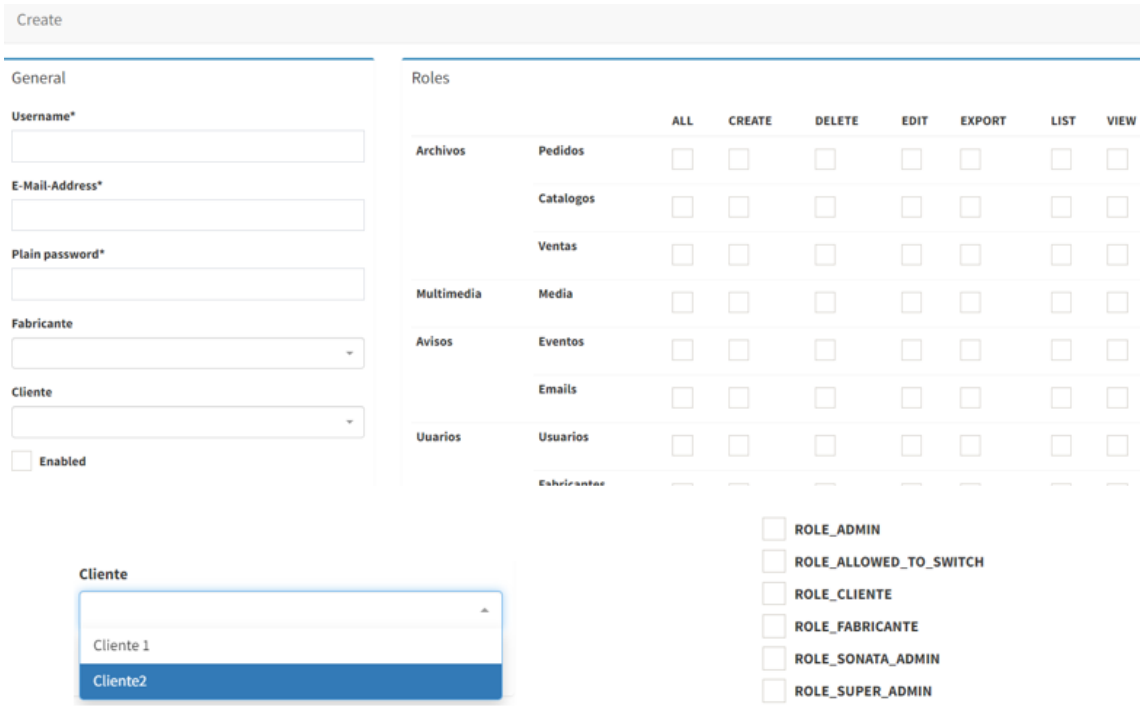


Figura 27. Crear usuario

Con esto el agente da de alta usuarios para que puedan acceder a la aplicación, pero ellos no van a ver lo que el agente comercial ve.

En el caso de los clientes, solo van a tener acceso al listado de sus pedidos, para verlos, descargarlos o imprimirlos, pero no editarlos:

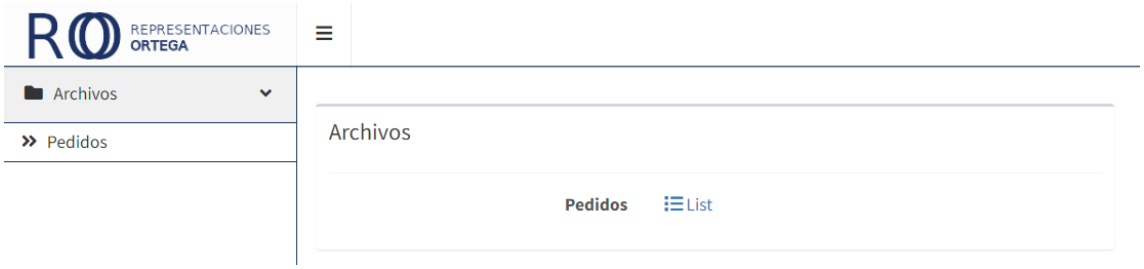


Figura 28. Dashboard cliente

En el caso de los fabricantes también van a poder ver, descargar e imprimir sus pedidos, pero también editar el estado del pedido. Pero no solo los pedidos, también van a poder ver sus catálogos y van a poder añadir nuevos, aunque solo podrán relacionarlos con ellos y no estarán activos. Lógicamente, al poder añadir catálogos, van a poder adjuntar archivos y subirlos a la aplicación, pero no van a poder ver la galería:

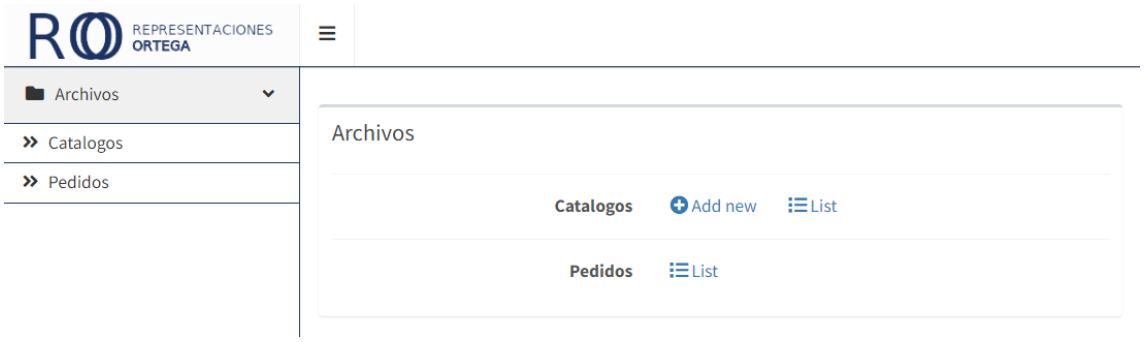


Figura 29. Dashboard fabricante

Y con esto concluiría el tour completo por la aplicación web. Con este viaje a través de la aplicación se muestra que se han implementado todas las funcionalidades clave definidas en los requisitos del proyecto. La aplicación proporciona una plataforma integrada y eficiente para que los agentes comerciales gestionen sus pedidos, catálogos y eventos, mejorando así su productividad y organización. Las interfaces de usuario son intuitivas, claras y responsivas, asegurando una experiencia de usuario positiva y agradable.

5. Conclusiones y trabajos futuros

5.1 Conclusiones

El desarrollo de esta aplicación web para la gestión de agentes comerciales ha sido un proyecto desafiante y enriquecedor. A lo largo de su desarrollo, se han alcanzado numerosos logros y se ha adquirido una valiosa experiencia. También se han encontrado alguna que otra traba y algunos errores improvistos, pero se han sabido solucionar con firmeza.

Cumplimiento de Objetivos

1. Interfaz de Usuario Intuitiva y Atractiva:
   Se ha logrado desarrollar una interfaz de usuario moderna y responsiva utilizando Twig y CSS. La aplicación es intuitiva y fácil de usar, lo que facilita la adopción por parte de los agentes comerciales. Se puede ver en las capturas de pantalla.

2. Gestión Eficiente de Pedidos y Catálogos:
   La aplicación permite la creación, edición y eliminación de pedidos y catálogos de manera eficiente. Los agentes comerciales pueden gestionar sus tareas desde un único lugar, mejorando su organización y productividad.

3. Control de Acceso Basado en Roles:
   Se ha implementado un sistema de control de acceso basado en roles, garantizando que solo los agentes comerciales puedan acceder a todas las funcionalidades. Estos roles son ‘administrador’, ‘cliente’ y ‘fabricante’, cada uno con diferentes permisos.

4. Correos y Recordatorios:
   Se ha logrado mejorar la comunicación de los agentes comerciales con clientes y fabricantes, así como mejorar la organización del tiempo con los eventos. Todo esto contribuye a que el trabajo del agente sea más eficiente y de calidad.

Qué se ha aprendido

1. Planificación Detallada:
   Durante el desarrollo del proyecto, se ha constatado la importancia de una buena planificación. Definir claramente los requisitos y diseñar una estructura firme desde el principio facilita el desarrollo y reduce la posibilidad de hacer cambios grandes en momentos ya avanzados.

2. Adaptabilidad y Flexibilidad:
   La capacidad de adaptarse a nuevos requisitos y cambios es muy importante en cualquier proyecto de desarrollo de software. En este proyecto, se realizaron varias modificaciones sobre la marcha, por lo que se aconseja ser flexible y adaptarse.

5.2 Trabajos futuros

A pesar de haber alcanzado los objetivos del proyecto, siempre debe haber espacio para mejoras y nuevas funcionalidades que podrían enriquecer la aplicación. A continuación, se presentan algunas ideas de trabajo futuro que podrían servir para mejorar aún más la aplicación:

1. Implementación de gráficas de datos
   Sería interesante poder introducir gráficas en la aplicación para poder contrastar datos entre meses o poder observar mejor el crecimiento o decrecimiento del negocio.

2. Soporte multilingüe
   Se podría mirar la posibilidad de contactar con algún traductor para trabajar conjuntamente en la traducción de la aplicación a varios idiomas, así se expandirían las fronteras en el mercado para que más gente se animase a usar la aplicación.

3. Optimización
   Una de las claves de las buenas aplicaciones web es el rendimiento, así que en un tiempo no muy lejano se podría empezar a trabajar en formas de optimizar las funcionalidades de la aplicación y que sea aún más rápida.

5.3 Reflexión

Se cree de verdad que esta aplicación puede tener un gran impacto para la gestión comercial, ya que como se dijo al principio, hay muchos agentes que están anticuados en sus métodos de trabajo, no teniendo buena comunicación con clientes o fabricantes, o perdiendo mucho tiempo con acciones que no deberían ni notarse, y con esta aplicación pueden centrarse en lo realmente importante. Aunque se han cometido errores, se han podido solucionar, así que ha sido y será una experiencia que no se olvidará fácilmente.

6. Bibliografía

1. Composer. (s.f.). Dependency Manager for PHP. En Composer. https://getcomposer.org/
2. FullCalendar. (s.f.). FullCalendar Documentation. En FullCalendar. https://fullcalendar.io/docs
3. GitHub. (s.f.). Gist. En GitHub. https://gist.github.com/
4. Leaflet. (s.f.). Leaflet Documentation. En Leaflet. https://leafletjs.com/
5. Lucidchart. (s.f.). Lucidchart Pages. En Lucidchart. https://www.lucidchart.com/pages/es
6. Sonata Project. (s.f.). Sonata Admin Bundle (4.x). En Sonata Project. https://docs.sonata-project.org/projects/SonataAdminBundle/en/4.x/
7. Sonata Project. (s.f.). Sonata Media Bundle (4.x). En Sonata Project. https://docs.sonata-project.org/projects/SonataMediaBundle/en/4.x/
8. Sonata Project. (s.f.). Sonata User Bundle (5.x). En Sonata Project. https://docs.sonata-project.org/projects/SonataUserBundle/en/5.x/
9. Stack Overflow. (s.f.). Questions. En Stack Overflow. https://stackoverflow.com/questions/
10. Symfony. (s.f.). FOSCKEditorBundle Installation. En Symfony. https://symfony.com/bundles/FOSCKEditorBundle/current/installation.html
11. Symfony. (s.f.). Mailer Documentation (5.x). En Symfony. https://symfony.com/doc/5.x/mailer.html
12. Symfony. (s.f.). Symfony Documentation (5.4). En Symfony. https://symfony.com/doc/5.4/index.html/
13. Twig. (s.f.). Twig Documentation (3.x). En Symfony. https://twig.symfony.com/doc/3.x/
14. Visual Paradigm. (s.f.). ERD Tool. En Visual Paradigm. https://online.visual-paradigm.com/es/diagrams/features/erd-tool/