

## Trabajo de Fin de Grado

---

# Portal de gestión de pedidos online orientado al pequeño comercio

---

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Juan José Díaz Santana

---

TUTORIZADO POR:

Dr. Francisco Alexis Quesada Arencibia

Fecha: Julio 2024

# Agradecimientos

*En especial a mi madre, que tenía mucha ilusión de verme terminar el grado y, por desgracia, no pudo ser así.*

*A mi familia y a mi pareja Gemma por apoyarme para que avance con el grado y el TFG.*

*Y, por último, a mi tutor, que después de tantos años, seguía dispuesto a ayudarme con el trabajo.*

# Resumen

Este trabajo de fin de grado ha sido impulsado durante la pandemia del coronavirus, donde las rutinas de la sociedad se veían afectadas por una disminución del contacto humano, acompañadas de un cambio generalizado de simplificar procesos.

Se elabora una plataforma de pedidos online enfocada a los pequeños comercios, especialmente los más afectados durante la pandemia. Los clientes pueden encontrar su tienda de confianza o productos que necesiten, realizando pedidos de diferentes formas con el fin de minimizar el tiempo que pasan en el establecimiento.

Además, también se busca aumentar el alcance y visibilidad de los pequeños comercios, con una interacción sencilla, rápida e intuitiva para usuarios de todas las edades.

# Abstract

*This thesis has been driven by the coronavirus pandemic, where society's routines were affected by a decrease in human contact, accompanied by a generalised change in simplifying processes.*

*An online ordering platform is developed with a focus on small businesses, especially those most affected during the pandemic. Customers can find their trusted shop or products they need, ordering in different ways in order to minimise the time they spend in the shop.*

*In addition, it also seeks to increase the reach and visibility of small businesses, with a simple, fast and intuitive interaction for users of all ages.*

# Índice

1. Introducción .....	12
1.1. Descripción general .....	12
1.2. Estado del arte.....	13
1.2.1. SAGE.....	13
1.2.2. Bitrix.....	15
2.1.3. Square .....	16
2. Estado actual y objetivos iniciales.....	17
2.1. Estado actual .....	17
2.2. Objetivos iniciales.....	17
3. Competencias.....	18
4. Normativa y legislación.....	18
4.1. Legislación .....	19
4.1.1. Ley Orgánica de Protección de Datos .....	19
4.1.2. Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico .....	19
4.2. Normativas .....	20
4.2.1. CC.....	20
4.2.2. MIT .....	20
4.2.3. GPL.....	20
4.2.4. APACHE 2.0 .....	20
5. Metodología y planificación .....	21
5.1. Metodología .....	21
5.2. Planificación .....	21
6. Tecnologías y herramientas utilizadas.....	22
6.1. Diseño .....	22
6.2. Desarrollo.....	23
7. Análisis .....	24
7.1 Actores .....	24
7.1.1. Administrador de la plataforma .....	24
7.1.2. Empleados de una tienda .....	25
7.1.3. Clientes .....	25
7.2. Casos de uso.....	25
7.2.1. Administrador de la plataforma .....	25
7.2.2. Empleados de una tienda .....	26

7.2.3.	Clientes .....	27
7.3.	Requisitos.....	28
7.3.1.	Requisitos funcionales .....	28
7.3.2.	Requisitos no funcionales .....	29
8.	Diseño .....	29
8.1.	Marca .....	29
8.1.1.	Significado .....	29
8.1.2.	Logotipo .....	30
8.1.3.	Paleta de colores.....	30
8.2.	Mockups.....	31
9.	Desarrollo .....	45
9.1	Lenguaje de programación.....	45
9.2	Estructura del proyecto .....	45
9.3.	Funciones implementadas destacadas .....	52
9.3.1.	Búsqueda de tiendas o productos .....	52
9.3.2.	Pedidos.....	54
9.3.3.	Pizarras.....	55
9.3.4.	Carta.....	59
10.	Despliegue del aplicativo .....	64
10.1.	Compra del dominio .....	64
10.2.	Compra del servidor VPS .....	65
10.3.	Configuraciones del servidor .....	66
10.3.1.	Instalación Plesk.....	66
10.3.2.	DNS.....	67
10.3.3.	SSL .....	68
10.3.4.	Puertos .....	69
10.4	Configuración con Git.....	70
10.5	Despliegue del proyecto .....	72
11.	Campaña de lanzamiento .....	72
11.1.	Prelanzamiento.....	73
11.2.	Lanzamiento.....	73
11.3.	Post-lanzamiento.....	74
12.	Resultados, conclusiones y trabajos futuros .....	74
12.1.	Resultados .....	74
12.2.	Conclusiones .....	75

12.3. Futuras mejoras .....	75
13. Bibliografía .....	76
14. Glosario.....	76
Anexo I. Manual de usuario.....	77
A1.1. Login.....	77
A1.2. Registro de usuarios .....	78
A1.3. Registro de tiendas .....	79
A1.4. Página principal.....	80
A1.5. Búsqueda de tiendas o productos .....	81
A1.6. Detalles de una tienda .....	82
A1.7. Resumen del pedido .....	85
A1.8. Perfil del usuario .....	86
A1.9. Panel de administración de administradores .....	86
A1.10. Panel de administración de tiendas.....	91

# Índice de figuras

Figura 1 - Gráfica comparativa del estudio "Ferquency Meal Delivery" .....	12
Figura 2 - Vista de productos en SAGE.....	14
Figura 3 – E-commerce generado por SAGE .....	15
Figura 4 - Pantalla de gestión de Bitrix .....	16
Figura 5 - Square resumen de utilización .....	17
Figura 6 - Casos de uso de un administrador de la plataforma.....	26
Figura 7 - Casos de uso de empleados de una tienda.....	26
Figura 8 - Casos de uso de los clientes .....	27
Figura 9 - Logo de "Ordery" transparente .....	30
Figura 10 - Logo de "Ordery" con fondo verde .....	30
Figura 11 - Paleta de colores.....	31
Figura 12 - Login .....	32
Figura 13 - Registro de cliente.....	33
Figura 14 - Registro de una tienda.....	33
Figura 15- Página principal.....	34
Figura 16 - Página de búsqueda .....	35
Figura 17 – Vista de tienda .....	36
Figura 18 - Vista Checkout.....	37
Figura 19 - Perfil de usuario .....	38
Figura 20 - Vista de tienda con productos en el carrito .....	39
Figura 21 - Dashboard de empleados de una tienda .....	39
Figura 22 - Pedidos de empleados de una tienda.....	40
Figura 23 - Detalles del pedido.....	41
Figura 24 - Pizarra de empleados.....	41
Figura 25 - Formulario de empleados .....	42
Figura 26 - Pizarra de todas las tiendas .....	43
Figura 27 - Formulario de tienda .....	43
Figura 28 - Pizarra de empleados.....	44
Figura 29 - Formulario de empleados .....	44
Figura 30 - Logo Laravel .....	45
Figura 31 - Estructura de Laravel .....	46
Figura 32 - Carpeta App .....	46
Figura 33 - Ejemplo de controlador.....	47
Figura 34 - Ejemplo de modelo .....	47
Figura 35 - Carpeta config .....	48
Figura 36 - Carpeta database.....	49
Figura 37 - Ejemplo de una migración .....	49
Figura 38 - Carpeta resources .....	50
Figura 39 - Carpeta routes.....	50
Figura 40 - Fragmento del archivo "web.php" .....	51
Figura 41 - Carpeta storage.....	51
Figura 42 - Lógica de búsqueda.....	53
Figura 43 - Resultados de una búsqueda con el criterio "Pan".....	54
Figura 44 - Lógica de guardado de pedidos.....	54
Figura 45 - Lógica de cálculo de precio.....	55



Figura 46 - Lógica de lado de servidor del datatable.....	56
Figura 47 - Método de inicialización datatable.....	56
Figura 48 - Definición de la variable "columns" .....	57
Figura 49 - Definición de la variable "columnDefs" .....	57
Figura 50 - Definición de la variable "filtros" .....	58
Figura 51 - Definición de la variable "options" .....	58
Figura 52 - Inicialización de dragula para categorías .....	59
Figura 53 - Lógica de creación y destrucción dragula.....	60
Figura 54 - Editar categorías en la carta .....	60
Figura 55 - Editar productos en la carta.....	60
Figura 56 - Lógica de guardado de carta .....	61
Figura 57 - Lógica de envío de la carta.....	62
Figura 58 - Guardado de categorías de la carta.....	62
Figura 59 - Guardado de productos de la carta .....	63
Figura 60 - Guardado de imágenes de la carta.....	63
Figura 61 - Búsqueda de dominio en IONOS .....	64
Figura 62 - Resultados búsqueda de dominios en IONOS .....	64
Figura 63 - Requisitos servidor VPS Linux S.....	65
Figura 64 - Login Plesk.....	66
Figura 65 - Entorno de plesk.....	67
Figura 66 - Plantilla de DNS de IONOS.....	67
Figura 67 - Herramienta DNSChecker revisando el dominio.....	68
Figura 68 - Vista de dominio en Plesk.....	68
Figura 69 - Botón de instalar certificado Lets Encrypt.....	69
Figura 70 - Registro TXT para validar el certificado .....	69
Figura 71 - Firewall de IONOS .....	70
Figura 72 - Botón de git en Plesk.....	70
Figura 73 - Ventana de creación de un nuevo repositorio en Plesk.....	71
Figura 74 - Creación de una deploy Key en github .....	71
Figura 75 - Botón de SSH Terminal del dominio en Plesk.....	72
Figura 76 - Login .....	78
Figura 77 - Registro de usuarios .....	79
Figura 78 - Registro de tiendas .....	80
Figura 79 - Página principal.....	81
Figura 80 - Página principal, parte inferior.....	81
Figura 81 - Listado de tiendas tras una búsqueda .....	82
Figura 82 - Detalles de una tienda .....	83
Figura 83 - Detalles de la tienda con imagen de nota de compra .....	83
Figura 84 - Detalles de la tienda con nota de compra.....	83
Figura 85 - Carrito de compra.....	84
Figura 86 - Carrito de compra sin hacer login .....	84
Figura 87 - Alerta cuando no hay productos en una tienda .....	85
Figura 88 - Resumen del pedido .....	85
Figura 89 - Pedido realizado con éxito .....	86
Figura 90 - Perfil de usuario .....	86
Figura 91 - Dashboard administrador .....	87

Figura 92 - Pizarra de empleados.....	87
Figura 93 - Formulario de nuevo empleado.....	88
Figura 94 - Pizarra de tiendas .....	88
Figura 95 - Filtros de las pizarras .....	89
Figura 96 - Formulario de tienda .....	89
Figura 97 - Métodos de pago disponibles .....	90
Figura 98 - Configuración de Paypal .....	90
Figura 99 - Modo noche del panel admin .....	91
Figura 100 - Accesos directos laterales .....	91
Figura 101 - Dashboard de empleado de una tienda.....	91
Figura 102 - Estadísticas de una tienda.....	92
Figura 103 - Pizarra de empleados.....	92
Figura 104 - Filtros de la pizarra de empleados.....	93
Figura 105 - Filtro de visibilidad del datatable.....	93
Figura 106 - Acciones de la pizarra de empleados .....	93
Figura 107 - Ejemplo de exportación en PDF de empleados.....	94
Figura 108 - Datatable de pedidos .....	94
Figura 109 - Resumen del pedido para la tienda.....	95
Figura 110 - Editando el total de un producto del pedido.....	95
Figura 111 - Mensaje de pedido actualizado .....	96
Figura 112 - Pedido con imagen subida por el usuario.....	96
Figura 113 - Zoom de la imagen subida por el usuario .....	96
Figura 114 - Carta editando categorías .....	97
Figura 115 - Editando una categoría del catálogo .....	97
Figura 116 - Producto en una categoría del catálogo.....	98
Figura 117 - Editando un producto del catálogo.....	98

# Índice de tablas

Tabla 1 - Tabla de competencias .....	18
Tabla 2 - Tabla de planificación TFT01 .....	22
Tabla 3 - Descripción de los casos de uso .....	28
Tabla 4 - Comandos para instalar Plesk.....	66
Tabla 5 - Comandos de despliegue de un proyecto Laravel .....	72

# 1. Introducción

## 1.1. Descripción general

Cada día hay personas que deciden emprender y crear la empresa o pyme que siempre han soñado. Desgraciadamente, en 2019 se presenta uno de los acontecimientos más importantes a nivel médico del último siglo, haciendo que los hábitos y costumbres de las personas cambien por completo de un día para otro.

Una de las medidas para la contención del coronavirus<sup>1</sup>, fue la limitación de la cantidad de personas en espacios públicos, haciendo que las personas quedaran confinadas en sus casas. Esto fomentó el crecimiento exponencial de los repartos a domicilio y del modelo de negocio “click & collect”<sup>2</sup>.

A todo lo anterior, hay que añadir la brecha digital en personas mayores de 65 años, que complican su acceso a la transición que está sufriendo la sociedad con el paso del coronavirus y el cambio de sus costumbres.

Tras el paso del coronavirus y los cambios y consecuencias que supuso para la sociedad, se fomentó la tendencia a realizar pedidos a domicilio que hoy en día se mantiene y sigue creciendo de manera exponencial. Según el estudio “Frequency Meal Delivery” por “Kantar” realizado en los años 2020, 2021 y 2022, confirma lo siguiente: “We’re also seeing a universally upward trend in how frequently people are ordering delivered food, with total foodservice occasions up to 85.5 per year in average for 2022, 31% of which are delivery instances” (en español, “También estamos observando una tendencia universal al alza en la frecuencia con la que la gente pide comida a domicilio, con un total de 85,5 ocasiones al año de media para 2022, el 31% de las cuales son entregas a domicilio.”)

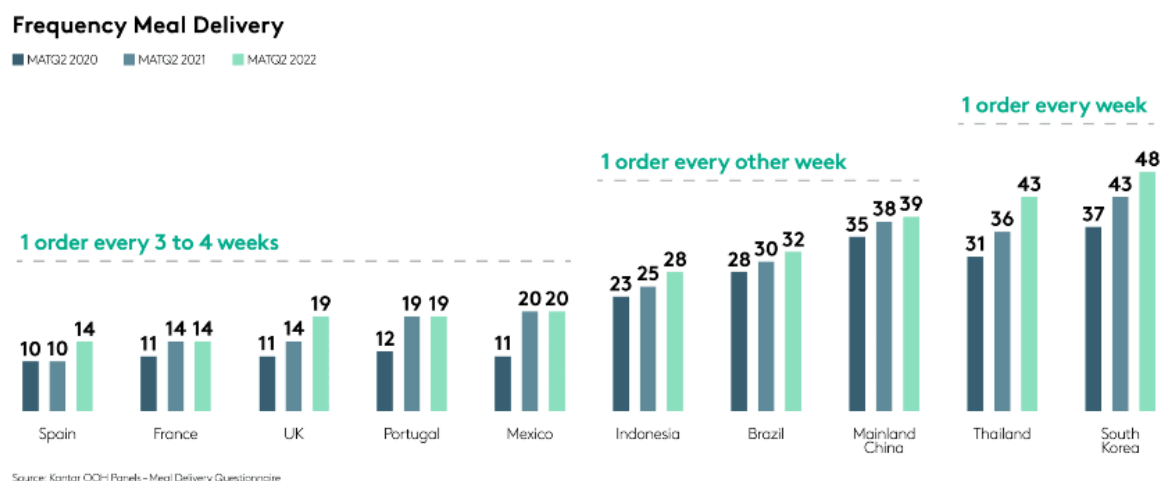


Figura 1 - Gráfica comparativa del estudio "Frequency Meal Delivery"

<sup>1</sup> El coronavirus, o también llamado COVID-19, fue una pandemia originada en 2019, donde las personas que contraían el virus sufrían una enfermedad respiratoria, en muchos casos, grave.

<sup>2</sup> Es un modelo de negocio que permite al cliente hacer una compra por internet y recogerlo en una tienda física.

Este incremento de pedidos a domicilio también tiene que ser reflejado en las capacidades de las empresas tanto grandes como pequeñas para satisfacer a los clientes.

Con el fin de buscar una alternativa que incluya: optimización de tiempo para los clientes, facilidades de gestión para las empresas y una herramienta fácil de usar e intuitiva para los adultos, se ha llevado a cabo la creación de *Ordery*, que pretende dar solución a las empresas que no tienen facilidades económicas para disponer de una plataforma donde gestionar pedidos online.

## **1.2. Estado del arte**

Para ayudar a los pequeños comercios a adentrarse en el mundo de las ventas de una forma sencilla y versátil, hay varias plataformas que ofrecen servicios bastante completos, aunque ninguna que lo haga con la intención y la facilidad que buscamos.

Hay aplicaciones muy populares multisectoriales que permiten mantener una gestión de pedidos e incluso algunos tienen “*addons*” (en español, extensiones) que permiten crear su tienda online como puede ser *SAGE*, *bitrix24* o *squareup*.

Para entender más el funcionamiento de cada herramienta mencionada, desglosaremos y comentaremos las fortalezas y debilidades de cada una de ellas.

### **1.2.1. SAGE**

Esta herramienta permite tener un “*TPV Online*”. Se conoce como TPV Online a todos aquellos programas o aplicativos que permitan a un negocio aceptar pagos online. Aunque *SAGE* disponga de una interfaz con una curva de aprendizaje bastante prolongada, dispone de bastante funciones y características para tener en cuenta. En la *Figura 2* se muestra una vista del listado de productos dentro de *SAGE* donde se puede ver la multitud de información que aporta para el trabajo diario de los empleados que lo usen.

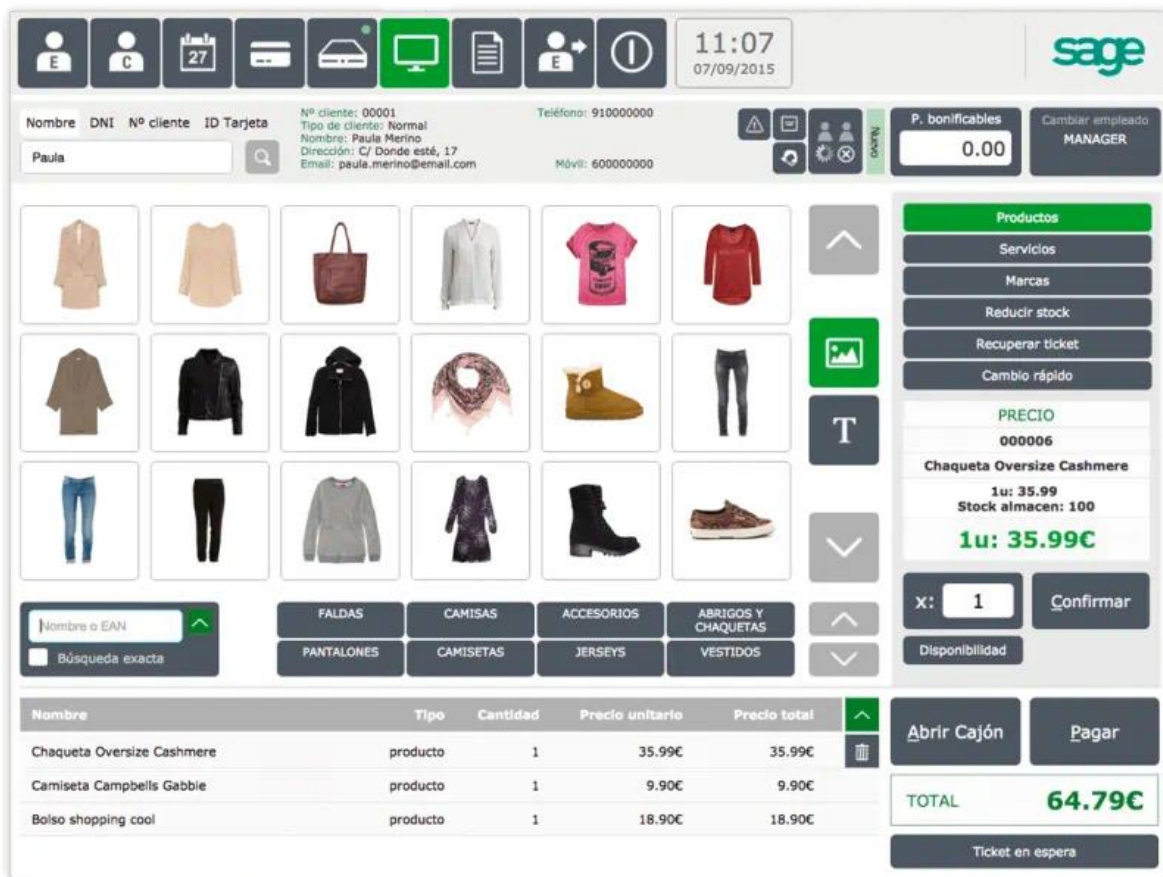


Figura 2 - Vista de productos en SAGE

Algunas de las características más destacables de la herramienta:

- Es una herramienta multisectorial, por lo que se utilizará para prácticas comerciales de cualquier tipo.
- Permite controlar el stock de productos y almacenes de la tienda. Esto favorece en gran medida a la escalabilidad del negocio, puesto que, si es una pequeña empresa y crece en los próximos años, no tendrá que migrar a otra herramienta.
- La herramienta da la posibilidad de crear una solución “e-commerce” (en español, comercio electrónico), es decir, crear una página web sencilla donde mostrar y vender productos (Figura 3).
- Tiene una caja para procesar las compras donde se pueden devolver, vender tarjetas regalos y muchas funciones cotidianas.
- Gestión de proveedores.
- Gestión de empleados.
- No requiere de instalación para usar la herramienta.
- Es una plataforma que requiere de pago mensual / anual. Un experto de la plataforma se pone en contacto con el cliente interesado para generar un presupuesto según las necesidades y escala de la empresa.
- La aplicación está disponible en diferentes idiomas.
- Tiene una tienda interna donde poder adquirir soluciones útiles según la línea de mercado que quiera tomar la empresa. Por ejemplo, una aplicación que se pueda

conectar directamente a *Prestashop* para sincronizar el catálogo, stock, almacenes, etc.

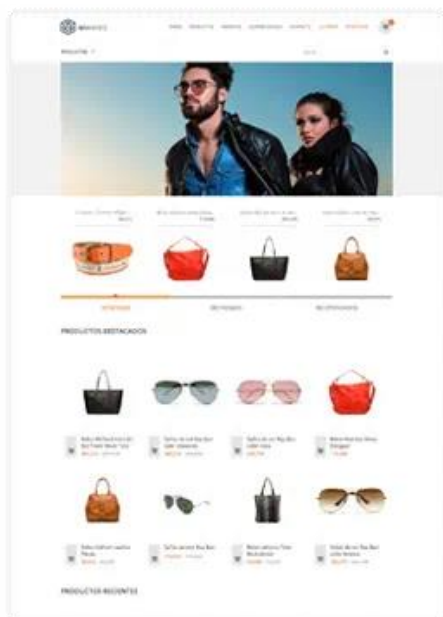


Figura 3 – E-commerce generado por SAGE

Remarcando lo ya comentado anteriormente, en la experiencia profesional del autor del trabajo, se ha podido comprobar como SAGE es utilizado de forma activa en el sector marítimo. Para ello, las diferentes empresas han tenido que contratar y desarrollar diferentes extensiones o ampliaciones dentro de la propia plataforma para permitirles guardar la información necesaria como, en este caso, a nivel de los buques que hacen escala en el puerto como de los datos relevantes de facturación para los mismos.

A modo de resumen, SAGE es una herramienta bastante completa que se ha labrado durante los últimos años bastantes funcionalidades útiles para pequeñas y grandes empresas. Aunque es un aplicativo capaz de permitirte gestionar todo lo necesario para un negocio, resulta bastante complicado adaptarse al entorno y a su funcionamiento.

### 1.2.2. Bitrix

Bitrix es una herramienta que permite la creación y gestión de los pedidos, pudiendo elaborar el catálogo de productos que será visible para los visitantes.

Las ventajas que tiene Bitrix son:

- Permite elegir entre un abanico de plantillas.
- Permite configurar múltiples métodos de pago.
- Permite conexiones con aplicaciones externas.
- Gestión de productos, stock y ventas.

Las principales desventajas son:

- Es un software privativo, por lo que tiene un coste mensual.
- La edición y personalización es muy limitada.

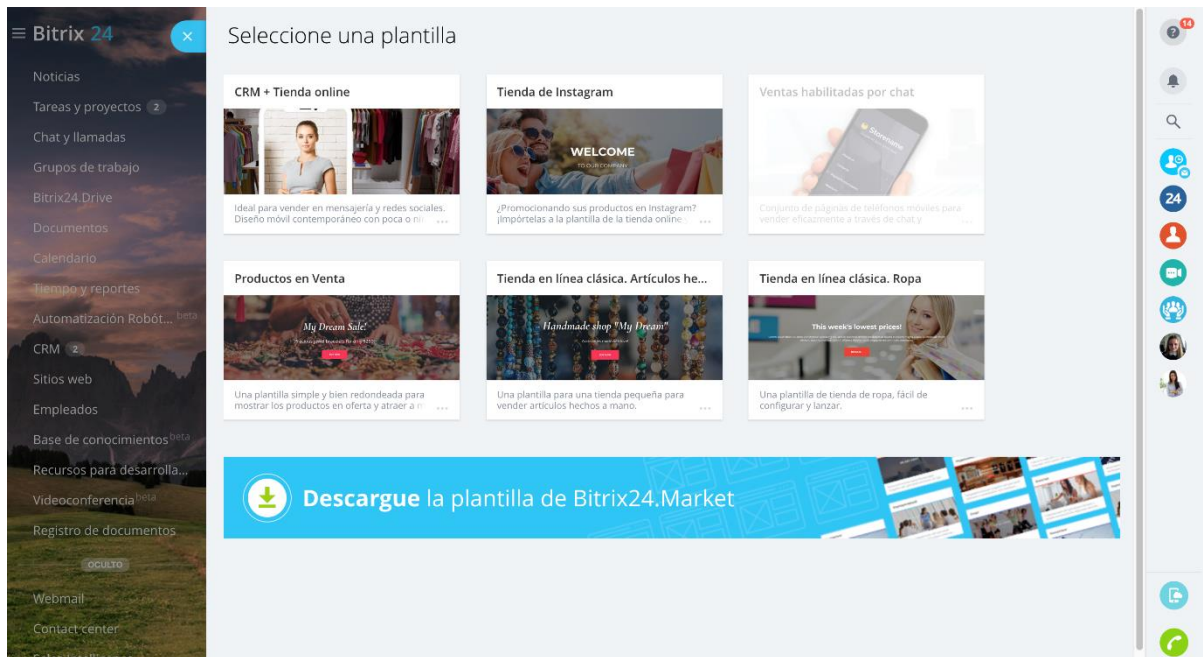


Figura 4 - Pantalla de gestión de Bitrix

En definitiva, bitrix es una plataforma capaz de proporcionar a los clientes que necesitan una web la posibilidad de tenerla, con sus limitaciones. Por la parte de *Ordery*, los clientes tendrán la posibilidad de no preocuparse por su web ni por su mantenimiento, y además, podrán tener todo el catálogo de una manera sencilla y sin preocuparse por comprar servidores, dominios, pagos de mantenimiento, etc.

### 2.1.3. Square

Square es una plataforma enfocada a la gestión y crecimiento de los comercios. La virtud de esta plataforma es la gran personalización en cualquier sector, por lo que permite adaptarse a cualquier entorno fácilmente. Además, ofrece la posibilidad de crear una tienda online básica.

Las principales ventajas de Square son:

- Gestión de pagos.
- Fácil adaptación multisectorial.
- Gestión de reservas de productos.
- Gestión de empleados.
- Gestión de clientes.
- Compatibilidad con terminales físicos.

Por otro lado, las desventajas de esta plataforma:

- Altas comisiones de uso de la plataforma y cobros.
- No favorece la visibilidad de los negocios ni fomenta su perfil público.





Figura 5 - Square resumen de utilización

Square posee una plataforma capaz de integrar soluciones físicas para las tiendas, enfocándose más en este último ya que es donde ellos recogen beneficios. Esta plataforma difiere bastante de la idea original de *Ordery*, puesto que la aplicación elaborada en este trabajo no cobra comisiones de creación de tienda, ni comisiones por pedido ni similares.

## 2. Estado actual y objetivos iniciales

### 2.1. Estado actual

Actualmente, el trabajo de fin de grado se encuentra en un proceso muy avanzado, donde podría llevarse a producción tras una intensiva depuración de todas las características que se realizarán en el proyecto.

Como es normal, durante el desarrollo se realizan exhaustivas pruebas de funcionalidades, pero siempre es conveniente que personas ajenas al desarrollo prueben y usen el aplicativo para detectar cualquier problema en el código o de usabilidad.

El proyecto cuenta con el código completo, una base de datos relacional funcional e incluso migraciones y “seeders”<sup>3</sup> para crear una base de datos nueva con la misma estructura.

Además, se ha llevado a cabo usos prolongados con personas ajenas al proyecto para comprobar que no existían errores y que la usabilidad es la esperada.

### 2.2. Objetivos iniciales

Este trabajo ha perseguido muy estrictamente los objetivos iniciales, sin dejar de lado la finalidad y el objetivo con el que se ha elaborado esta idea. Se enumera a continuación los principales puntos objetivos a cumplir con este trabajo.

- a. Ayuda a los pequeños comercios. Uno de los principales objetivos es el apoyo a los pequeños comercios que han sufrido las consecuencias de la pandemia, proporcionando una plataforma capaz de visibilizar tanto su tienda como sus productos. Además, permitiendo que los clientes potenciales puedan hacer pedidos y recoger en la tienda de forma rápida.
- b. Brecha digital. Proporcionar una interfaz sencilla, con claros iconos y texto legible para que cualquier usuario sin importar su edad, pueda interactuar y entender que está

---

<sup>3</sup> La palabra anglosajón “*Seeder*” se traduce textualmente como “sembradores”. Es una funcionalidad que dispone los lenguajes de programación más modernos para poblar una base de datos. Son capaces de llenar muchas filas de la base de datos con información aleatorios.

sucediendo en todo momento. Con esto conseguiremos fomentar la franja de edad de uso del aplicativo, haciendo que el rango de clientes potenciales sea mayor.

- c. Eficiencia. Facilitar a los clientes el contacto y solicitar productos a una tienda, ya que simplemente tendrá que pagar y recoger su pedido que ya estará preparado previamente. Esto contribuye a las políticas de restricción de la pandemia, donde había usuarios que no quieren pasar demasiado tiempo dentro del comercio por el temor a contraer algún tipo de infección y, sobre todo, actualmente a aquellas personas que quieren agilizar sus rutinas diarias y desean hacer el proceso de comprar productos mucho más rápido.

### 3. Competencias

Durante el desarrollo de este trabajo, se ha podido profundizar entre las siguientes competencias, poniendo en práctica los conocimientos adquiridos durante el grado.

Competencia	Definición	Justificación
TI1	<i>“Capacidad para comprender el entorno de una organización y sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones.”</i>	Se han identificado los puntos potenciales para que con este proyecto las empresas puedan trabajar de forma eficiente con la plataforma.
TI3	<i>“Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.”</i>	Esta competencia es uno de los objetivos principales del proyecto, ya que se pretende que los usuarios puedan realizar un pedido en su tienda favorita rápidamente, incluyendo una interfaz sencilla para que las personas que sufren la brecha digital se sientan más cómodas. Además, las empresas puedan gestionar los pedidos y procesarlos más cómodamente.
TI6	<i>“Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.”</i>	Para el desarrollo del aplicativo había que comprender y aplicar conocimientos en tecnologías de red, puesto que la comunicación con la base de datos, pedidos e incluso el despliegue de la aplicación en un servidor, hacen uso de esta competencia.

Tabla 1 - Tabla de competencias

### 4. Normativa y legislación

En este apartado abordaremos las leyes más importantes relacionadas con el proyecto y las licencias que hacen uso las herramientas que se han empleado para el desarrollo del trabajo.

## **4.1. Legislación**

Para que el proyecto tenga éxito, hay que tener en cuenta la legislación actual de España. El obligado cumplimiento de estas leyes es indispensable para proteger a las personas que hacen uso del aplicativo y evitar cuantiosas multas.

### **4.1.1. Ley Orgánica de Protección de Datos**

La Ley Orgánica de Protección de Datos (LOPD), fue publicada el 15 de diciembre de 1999, que casi 20 años después fue derogada por una ley mucho más restrictiva, en 2018.

La LOPD regula de forma activa la seguridad de los datos de carácter personal de los usuarios tanto en los sistemas de información como en otros sistemas.

Esta ley es importante tenerla en cuenta ya que en el aplicativo debe existir un apartado “Política de privacidad” que incluya al menos:

- a. Identidad y datos de contacto del responsable del tratamiento (la empresa)
- b. Finalidad del tratamiento de datos
- c. Derechos de los usuarios
- d. Comunicación de datos a terceras partes

Además, esta ley también regula las Cookies, tanto lo que recopilan y cómo lo hacen. En una primera instancia, la web no puede recopilar cookies hasta que el usuario no da su consentimiento. Como novedad en los últimos años, se ha profundizado más en como se muestra el mensaje de consentimiento de las cookies y prohibición de los muros que no dejan navegar en la web hasta que el usuario acepte las cookies.

Por otro lado, el proyecto garantiza la encriptación de la contraseña de los usuarios, la protección mediante firewall y medidas de seguridad avanzadas en consultas y rutas del proyecto. Programáticamente, también se controlan los privilegios de los usuarios y se prohíbe el acceso a zonas o información restringida.

Regularmente, estos apartados son revisados por el equipo de inspección la *AEPD* (Agencia Española de Protección de Datos).

### **4.1.2. Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico**

La Ley de Servicios de la Sociedad de la Información y del Comercio Electrónico (LSSICE), fue publicada el 11 de Julio de 2002.

Esta ley regula todo lo relacionado sobre los prestadores de servicios de los comercios electrónicos. Es importante tener en cuenta los siguientes aspectos en el aplicativo:

- a. Al igual que en la LOPD, obligación de informar de la política de privacidad.
- b. Obligación de informar el Aviso legal.
- c. Obligación de informar la Política de Cookies.
- d. Se prohíbe enviar correos a los usuarios sin previo consentimiento.
- e. Disponibilidad de las condiciones generales de contratación.

El aplicativo incluye cada uno de los apartados, donde son visibles y accesibles desde el pie de página de la web.

## 4.2. Normativas

### 4.2.1. CC

Las licencias CC (*Creative Commons*) dan libertad de compartir el proyecto en cuestión, pudiendo distribuirlo, copiarlo o incluso, realizar un proyecto nuevo basándose en el original. Existen hasta 6 tipo de licencias CC que se pueden configurar para restringir las acciones de los usuarios.

Programas que se han utilizado con esta licencia:

- Github

### 4.2.2. MIT

La licencia MIT (*Massachusetts Institute of Technology*) es una licencia muy comúnmente usada en software libre, por lo que no restringe tanto como otras licencias. Esta en particular, permite reutilizar el software. Además, es una licencia bastante versátil ya que puede combinarse con licencias como GPL.

Programas que se han utilizado con esta licencia:

- Laravel
- JQuery

### 4.2.3. GPL

La licencia GPL (*General Public License*) también es muy comúnmente utilizada por aplicativos de código abierto (software libre). La diferencia principal diferencia, es que GPL es mucho más restrictiva, ya que se enfoca en proteger el software y que continúe siendo libre siempre.

Programas que se han utilizado con esta licencia:

- Git
- MySQL
- HeidiSQL

### 4.2.4. APACHE 2.0

Al igual que las anteriores licencias, Apache 2.0 se emplea para software libre, es decir, permite distribuirlo y modificarlo. La principal diferencia con las otras licencias es que no obliga a que el software creado a partir del original tenga que usar la misma licencia, por lo que a partir de un software libre se podría crear un software privativo.

Programas que se han utilizado con esta licencia:

- IntelliJ PHP Storm

## **5. Metodología y planificación**

### **5.1. Metodología**

La metodología de trabajo utilizada para llevar a cabo el proyecto ha sido Gantt. La creación de hitos, las cuales llamo “banderas”, son aquellas secciones que engloban un conjunto de tareas mucho más extensa.

Para cada bandera es importante tener definido cuanto tiempo se tardará en realizarlo, el cual no debería ser complicado calcular porque ya se debe tener definido cada una de las tareas a desarrollar para cumplir el objetivo de este.

Un paso previo antes de comenzar a desarrollar con esta metodología es detectar las dependencias de cada una de las banderas, ya que, si se prioriza aquellas que tienen una menor dependencia con otras banderas, conseguiremos una eficiencia mucho mayor en el desarrollo, ya que no se tendrá que dejar a medias el desarrollo de banderas para realizar otras y luego volver a terminarlas.

Una versatilidad que proporciona la metodología de Gantt, es combinar cada finalización de una bandera, con una serie de pruebas internas para valorar el estado del desarrollo. Esto es una muy buena praxis, puesto que permite detectar errores de programación que no se han detectado anteriormente y, además, hacer un análisis de la usabilidad de la UI (Interfaz de Usuario) de la web.

Si durante el desarrollo se obtiene bastante ventaja con las respectivas fechas de entrega, se puede aprovechar la finalización de una bandera para revisión y optimización del código. Con esto conseguiremos aumentar de forma considerable el código limpio<sup>4</sup> para que el software sea aún más sostenible a lo largo del tiempo.

Todas estas tareas anteriormente comentadas, deben estar reflejadas correctamente en el diagrama de Gantt para que el tiempo de ejecución del proyecto sea lo más fiel posible a la realidad.

### **5.2. Planificación**

Durante el proceso de definición del proyecto, se definió los principales objetivos a cumplir con el proyecto, teniendo especialmente en cuenta de cubrir todos los aspectos necesarios para el éxito de este.

---

<sup>4</sup> Código limpio es una serie de buenas praxis que permite que el código que se desarrolla sea más legible, primando la sencillez, comentar los bloques de código y evitando recursividades complejas que será muy difícil comprender.

Fases	Duración (horas)	Tareas
Estudio previo / Análisis	45	Tarea 1.1: Estudio de como satisfacer las necesidades del cliente y del vendedor de manera óptima.
		Tarea 1.2: Obtención de los requisitos del usuario de la aplicación.
		Tarea 1.3: Elaboración de una maqueta con la estructura de la aplicación.
		Tarea 1.4: Aprendizaje de la tecnología que se empleará para el desarrollo.
Diseño / Desarrollo / Implementación	175	Tarea 2.1: Realización de las definiciones de las tareas que se ejecutarán durante el desarrollo.
		Tarea 2.2: Diseño y creación de la base de datos.
		Tarea 2.3: Desarrollo del portal basándonos en las definiciones del punto 2.1.
		Tarea 2.4: Aplicar técnicas de código limpio.
Evaluación / Validación / Prueba	30	Tarea 3.1: Pruebas de la aplicación en diferentes pantallas y dispositivos.
		Tarea 3.2: Pruebas de las funcionalidades de la aplicación y su fiabilidad.
Documentación / Presentación	50	Tarea 4.1: Realización de la memoria.
		Tarea 4.2: Realización de la presentación.

Tabla 2 - Tabla de planificación TFT01

En efectos generales, la planificación anterior se ha cumplido satisfactoriamente, ajustándose a las horas estipuladas. Esto ha sido posible puesto que el análisis previo a la estimación se ha hecho de forma exitosa, favoreciendo así la exactitud.

En la planificación hay que tener en cuenta que el autor del proyecto ya era experimentado en el lenguaje de programación, por lo que no se ha contemplado en ningún momento el aprendizaje y estudio de Laravel y su entorno al no ser necesario. Esto suma una ligera ventaja a la hora de estimar con precisión todo el proyecto.

## 6. Tecnologías y herramientas utilizadas

### 6.1. Diseño

Para las labores de diseño de la aplicación se ha utilizado las siguientes herramientas:

- *Figma*: Es un editor online que permite la creación y edición de gráficos de tipo vectorial o prototipos. Se ha empleado esta herramienta para la creación de los bocetos de todo el proyecto, tanto las vistas de administrador, como las de cliente.
- *Drawio*: Es un software online que permite generar de forma rápida e intuitiva diagramas de flujo, UML, entre otros. Con esta herramienta se ha podido crear tanto los casos de uso de la aplicación como diagramas de proceso durante el desarrollo para clarificar las funcionalidades a desarrollar.

## 6.2. Desarrollo

Para el desarrollo de la aplicación, se ha utilizado las siguientes herramientas:

- IntelliJ PHP Storm: Esta aplicación es un *IDE* (entorno de desarrollo) de programación enfocado especialmente en desarrollos de PHP. El uso de un *IDE* durante el desarrollo proporciona ventajas como la comodidad y eficiencia, aprovechando el análisis de código y recomendaciones que aporta.

Para el desarrollo del proyecto se ha utilizado este programa por su gran potencia y ventajas que aporta al desarrollador.

- Git: Es un software de gestión de versiones que permite almacenar en un repositorio el código fuente de tu aplicación y llevar un control de los cambios que se realizan y sus versiones.

Para el control de versiones de este proyecto, se ha utilizado este software.

- HeidiSQL: Es un cliente que proporciona una interfaz de usuario y funcionalidades enriquecedoras para la gestión y mantenimiento de base de datos tanto relacionales como no relaciones. Acepta MySQL (el que se usa en este proyecto) como MariaDB, Microsoft SQL Server, entre otros.

Para la gestión de la base de datos, comprobaciones y estudio de la correcta arquitectura, se ha utilizado este programa.

- MySQL: Es un gestor de base de datos relacional. Aunque actualmente existen otras opciones, MySQL se caracteriza por tener una potencia de escritura y lectura bastante equilibrado para un uso estándar.

La base de datos del proyecto se ha elaborado con esta herramienta.

- Node: Es un conjunto de herramientas y librerías que proporciona todo lo necesario para que el lenguaje de programación funcione correctamente. Además, proporciona al desarrollador herramientas útiles.

Se ha utilizado tanto para compilar y minificar el código JavaScript. También es una dependencia de Laravel para su correcto y eficiente funcionamiento.

- PHP: Lenguaje de programación orientado al desarrollo web e interpretado por el lado del servidor.

Se ha utilizado para el desarrollo de la aplicación, ya que Laravel tiene como base PHP como lenguaje de programación.

- Composer: Es un software que permite la gestión de paquetes para desarrollos enfocados en el lenguaje PHP. Aunque se puede prescindir de este programa, facilita mucho la instalación de librerías y dependencias, automatizando el proceso.
- Laravel: Laravel es un *framework*<sup>5</sup> orientado al desarrollo de aplicaciones y servicios web PHP.

Se ha usado Laravel como base del proyecto web, donde combinado con PHP y las otras herramientas comentadas, el autor es capaz de llevar a cabo el desarrollo.

- Laragon: Es una aplicación que no ejecuta ningún servicio de Windows que ayuda a los desarrolladores a preparar el entorno local para el desarrollo. Entre sus funcionalidades destaca la automatización de los proyectos en el *host virtual* en Apache para que se pueda acceder rápidamente mediante el navegador con una URL personalizada. Además, permite controlar las versiones de PHP, Node, MySQL de forma rápida por si se cambia los requerimientos durante el desarrollo o se cambia de proyecto.

Laragon se ha utilizado como parte de la estructura local de desarrollo para llevar a cabo el desarrollo.

## 7. Análisis

### 7.1 Actores

En esta aplicación existen tres tipos de actores diferentes. A continuación, se detallará cada uno de ellos y cuáles son sus funciones dentro de la plataforma.

#### 7.1.1. Administrador de la plataforma

El administrador de la plataforma es aquel que tiene acceso y control de todas las características.

Este rol, también está sometido a permisos, por lo que se pueden tener administradores con accesos restringidos, como, por ejemplo, personal de marketing que solo puedan acceder a sitios específicos de la web.

Las acciones que puede realizar un administrador de la plataforma en esta versión son las siguientes:

- a. Crear y gestionar empleados de la plataforma.
- b. Crear y gestionar las tiendas de la plataforma. También pueden bloquearlas para que no aparezca en las búsquedas.

---

<sup>5</sup> Un framework es un marco de trabajo que aplica una metodología de trabajo estricto, donde combinado con un conjunto de reglas, hace que el desarrollo sea más efectivo ya que se ahorra tiempo.



- c. Gestionar los métodos de pago. Puede configurar, activar o desactivar los métodos de pago disponibles en la plataforma.

### **7.1.2. Empleados de una tienda**

Los empleados de una tienda son aquellos que acceden al panel de gestión de la tienda donde trabajan.

Este rol, al igual que el anterior, tiene a su disposición la asignación de permisos, por lo que puede haber empleados con diferentes niveles de acceso a la plataforma.

Las acciones que puede realizar un empleado de una tienda son las siguientes:

- a. Crear y gestionar empleados que puedan acceder al panel de la tienda.
- b. Gestionar los pedidos.
- c. Crear y gestionar la carta de productos que será visible en la web.

### **7.1.3. Clientes**

Los clientes son usuarios registrados que acceden a la web para buscar una tienda o producto de interés y realizar un pedido.

Las acciones que puede realizar un cliente son las siguientes:

- a. Buscar tiendas o productos.
- b. Añadir una tienda a la *wishlist* (en español, lista de deseados).
- c. Ir al perfil de una tienda y realizar un pedido de tres formas diferentes:
  - i. Seleccionando productos de la carta disponible por la tienda.
  - ii. Enviando una foto de las notas de compra que el cliente pueda tener en casa
  - iii. Escribiendo en un campo de texto lo que desea.
  - iv. Una combinación de todos los puntos anteriores.

## **7.2. Casos de uso**

En este apartado comentaremos los casos de uso de cada uno de los actores que participan en la aplicación.

### **7.2.1. Administrador de la plataforma**

El administrador de la plataforma tiene los siguientes casos de uso:

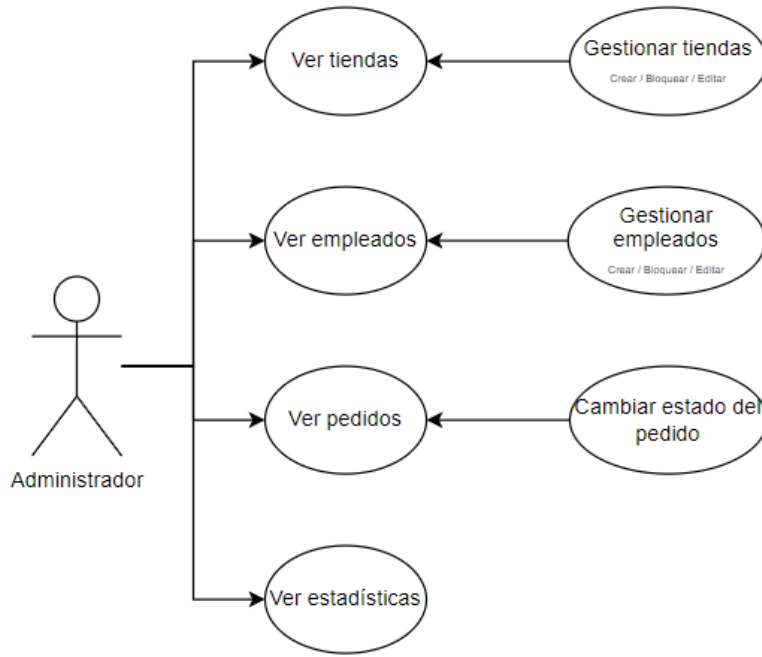


Figura 6 - Casos de uso de un administrador de la plataforma

### 7.2.2. Empleados de una tienda

Los empleados de las tiendas tienen los siguientes casos de uso:

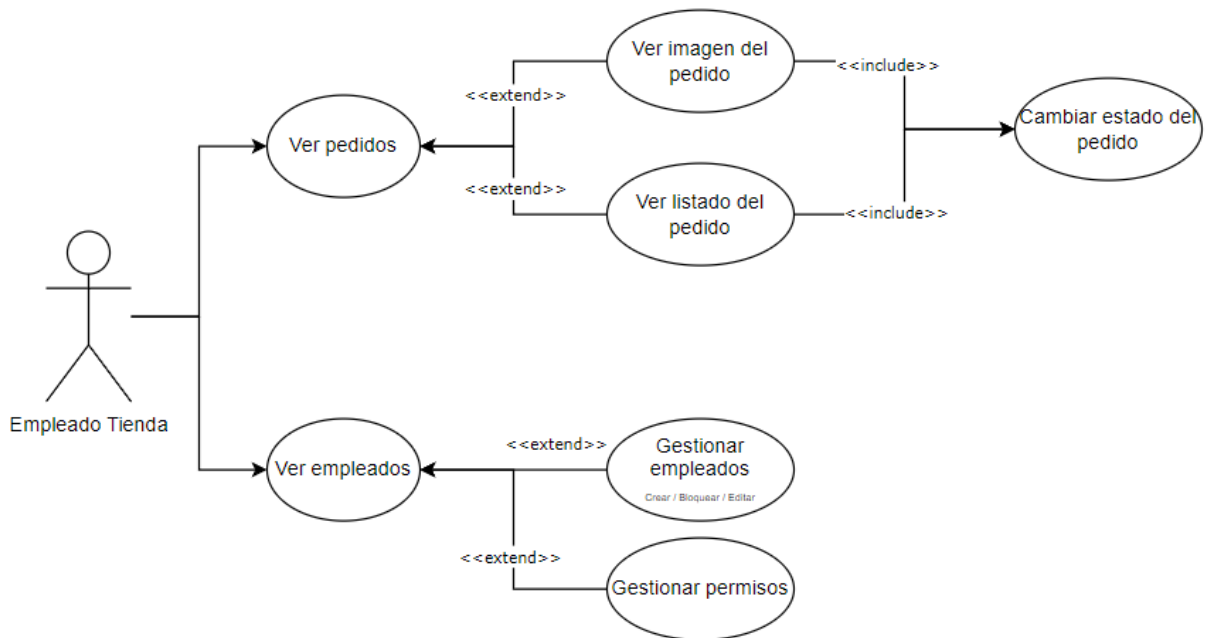


Figura 7 - Casos de uso de empleados de una tienda

### 7.2.3. Clientes

Los clientes tienen los siguientes casos de uso:

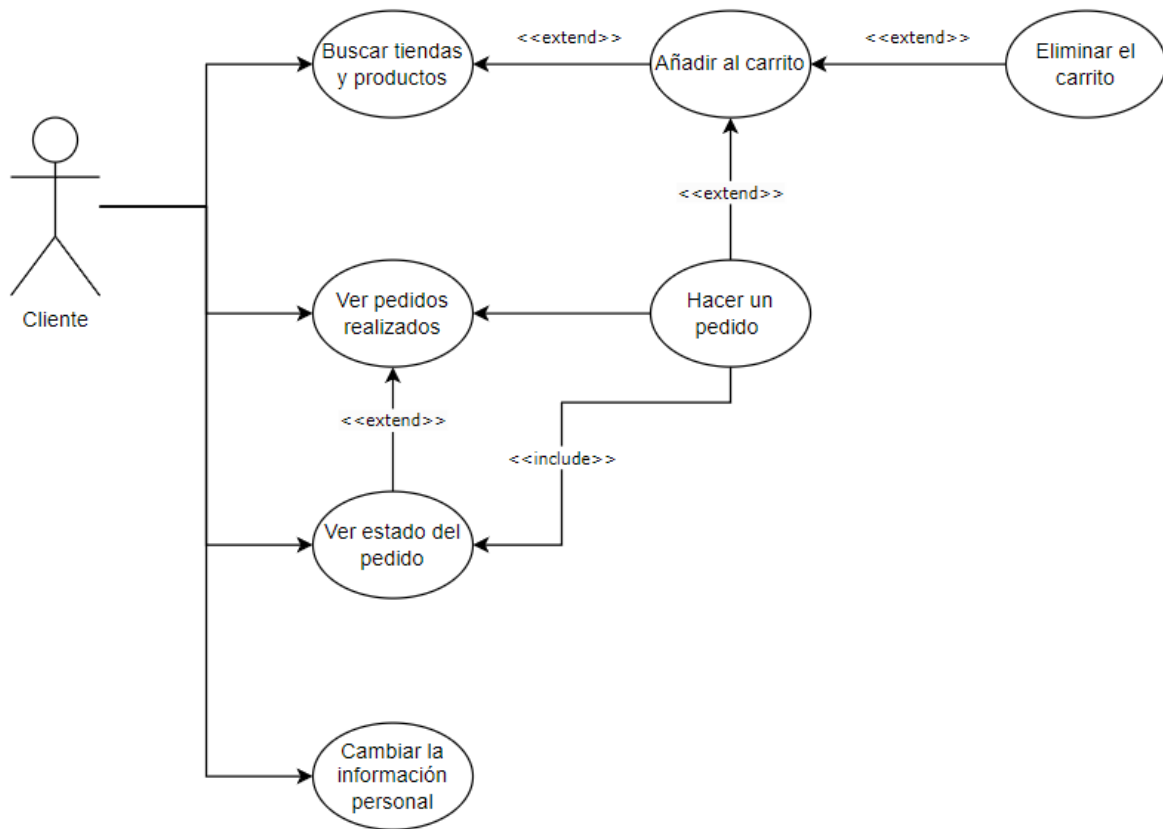


Figura 8 - Casos de uso de los clientes

Se detallan a continuación cada uno de los casos de uso descritos en la *Figura 6*, *Figura 7* y *Figura 8*.

Caso de uso	Descripción	Observación
Buscar tiendas y productos	Se puede buscar tanto tiendas como productos y ver un listado completo	-
Añadir al carrito	Permite añadir al carrito un producto, una imagen de una nota o texto plano	-
Eliminar el carrito	Permite quitar un artículo del carrito	-
Ver pedidos realizados	Permite ver un listado de pedidos realizados	-
Hacer un pedido	Habilita al usuario a realizar un pedido que automáticamente aparecerá en la tienda para procesar	-
Ver estado del pedido	Permite ver el estado del pedido actual	-

Cambiar la información personal	Permite editar la información personal del usuario	-
Ver tiendas	Posibilidad de ver todas las tiendas o por algún criterio de búsqueda	-
Ver empleados	Posibilidad de ver todos los empleados o por algún criterio de búsqueda	-
Ver pedidos	Posibilidad de ver todos los pedidos o por algún criterio de búsqueda	-
Ver estadísticas	Posibilidad de ver todas las estadísticas relacionadas con la tienda	-
Gestionar tiendas	Permite realizar diferentes acciones a una tienda	Crear / Bloquear / Editar
Gestionar empleados	Permite realizar diferentes acciones a los empleados	Crear / Bloquear / Editar
Cambiar estado del pedido	Permite realizar un cambio de estado en el pedido	Esta acción solo la puede realizar un empleado de tienda
Ver imagen del pedido	Permite ver la imagen que ha subido un usuario de su nota de compra	Esta acción solo la puede realizar un empleado de tienda
Ver listado del pedido	Permite ver un listado de los pedidos realizados	-
Gestionar permisos	Permite realizar diferentes acciones con los permisos	-

*Tabla 3 - Descripción de los casos de uso*

## 7.3. Requisitos

Los requisitos son aquellas funcionalidades básicas que el proyecto debe disponer para su correcto funcionamiento e información de los usuarios.

### 7.3.1. Requisitos funcionales

A continuación, se enumeran los requisitos funcionales básicos:

- Se permitirá a un usuario registrarse en la aplicación, tanto como cliente como si quiere darse de alta como una tienda.
- Se permitirá a un usuario con una cuenta creada, poder iniciar sesión en la plataforma.
- Se permitirá a cualquier usuario realizar búsquedas de tiendas y productos en la plataforma.
- Se permitirá únicamente a usuarios con la sesión iniciada poder realizar un pedido.
- Los usuarios con la sesión iniciada podrán realizar un pedido de las siguientes formas:
  - Seleccionando productos de la carta disponible por la tienda.
  - Enviando una foto de las notas de compra que el cliente pueda tener en casa.
  - Escribiendo en un campo de texto lo que desea.
  - Una combinación de todos los puntos anteriores.
- Los usuarios que hayan realizado un pedido podrán ver un historial de pedidos anteriores y el estado actual del pedido.

- Se permitirá a un administrador de la plataforma con permisos de crear empleados, poder crear nuevos empleados con permisos a la plataforma.
- Se permitirá a un empleado de una tienda con permisos para crear empleados, poder crear nuevos empleados con permisos de acceso a la plataforma.
- Se permitirá a un administrador de la plataforma con permisos pertinentes, ver los pedidos de todas las tiendas.
- Se permitirá a los empleados de las tiendas con permisos pertinentes, ver y gestionar los pedidos de la tienda.
- Se permitirá a un empleado de una tienda con permisos pertinentes, ver y gestionar la carta de productos de la tienda.
- Se permitirá a un empleado de una tienda con permisos pertinentes, ver las estadísticas de la tienda.
- Se permitirá a un administrador de la plataforma con permisos pertinentes, poder gestionar los métodos de pago.
- Se le notificará al usuario de cualquier cambio en su pedido.
- Se podrá añadir y eliminar productos del carrito de la compra.
- Se podrá eliminar productos en el *checkout* (en la página de pago).

### 7.3.2. Requisitos no funcionales

A continuación, se enumerarán los requisitos no funcionales básicos:

- Se mostrará al usuario alertas en los formularios para informar de datos incorrectos o faltantes.
- La *UI* (Interfaz de usuario) debe ser sencilla para todo tipo de usuarios.

## 8. Diseño

### 8.1. Marca

Lo más importante de una empresa, es su marca. A continuación, comentaremos aspectos importantes y el enfoque de la marca.

#### 8.1.1. Significado

Con la marca se ha creado una propuesta que de un valor único y que sea diferenciado con otras empresas. Es importante también que la marca sea clara y se entienda que es lo que hay detrás de ella sin tener que explicar demasiado en que consiste.

El nombre que se ha elegido para el proyecto ha sido “*Ordery*”. La combinación de “*Order*” (pedido, en inglés) añadiendo una “*y*” al final de la palabra hace que sea muy fácil de recordar y sencilla de pronunciar.

Como ya se ampliará en apartados posteriores, la plataforma permite realizar un pedido a una tienda, por lo que por ese motivo el autor del trabajo eligió “*Order*” como palabra raíz.

A modo de resumen, lo que se pretende en aspectos generales con la marca es:

- Conseguir que destaque y que sea fácil de recordar.

- Que sea simple tanto de escribir como de decir.
- Que sea llamativo y atractivo para los usuarios.
- Que el nombre de la marca de confianza a los usuarios.

### 8.1.2. Logotipo

Aunque el nombre de la marca también es importante, que le acompañe un logo acorde al mensaje que se pretende dar también lo es.



*Figura 9 - Logo de "Ordery" transparente*

El logo se caracteriza por tener un estilo moderno, donde predomina el nombre del comercio y el modelo de negocio. La elección del color no ha sido aleatoria, ya que al igual que la importancia que tiene un nombre claro y sencillo, lo son los colores. El verde aporta gran valor porque al vender productos tanto comestibles como los que no lo son, este color se asocia a la frescura, bienestar y calma.



*Figura 10 - Logo de "Ordery" con fondo verde*

Esta tendencia que se ha aplicado en la elección del color podría considerarse “*Neuromarketing*”, ya que se da importancia al factor psicológico del usuario y lo que percibe al ver el logotipo.

### 8.1.3. Paleta de colores

En la paleta de colores empleada para la elaboración de la web, se ha empleado un tono menos llamativo como el verde, ya que la percepción que se pretende dentro de la web es distinta.

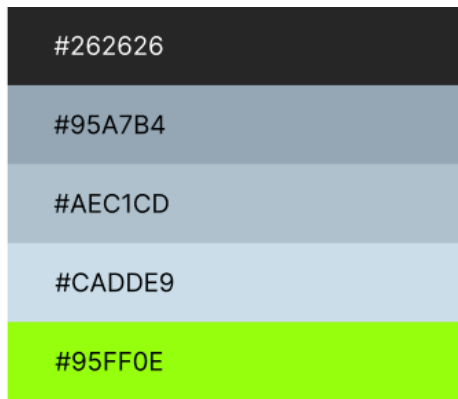


Figura 11 - Paleta de colores

El autor del trabajo ha elegido una escala de grises, con la finalidad de proporcionar una neutralidad generalizada en toda la web, haciendo que los usuarios tengan una percepción de elegancia, profesionalidad y sobre todo moderno.

Este rango de colores proporciona una comodidad añadida durante la fase de diseño, puesto que combinar un rango de colores mucho más llamativo puede ser más complicado. También hay que atender a la accesibilidad de dichos colores, ya que, con colores muy llamativos, hay que pensar muy bien los colores y tamaños de la fuente para que sea visible por cualquier usuario.

Como se comenta anteriormente, para simplificar el diseño, aumentar la accesibilidad y, sobre todo, pensando en la percepción del usuario, se ha elegido un rango de grises.

## 8.2. Mockups

El diseño de *mockups* es una de las partes más fundamentales de la fase de diseño del proyecto, puesto que tiene que contemplar cada una de las funciones que se describen en los casos de uso.

Este proceso puede, en ocasiones, convertirse en algo tedioso cuando el proyecto es grande, ya que, en este caso, al tener tres actores que participan en el entorno del aplicativo, hay que tener en cuenta a cada uno de ellos.

Hay que remarcar que se han creado los componentes para que puedan reutilizarse en las vistas que fueran necesarias, con el fin de optimizar el desarrollo y mostrar una identidad uniforme por toda la web.

El inicio del maquetado se ha realizado con el *login* para acceder a la plataforma tanto clientes como tiendas.



## Bienvenido a Orderly!

Accede al panel para disfrutar de todas las ventajas

Email

Contraseña

[¿Has olvidado la contraseña?](#)

Recuérdame

¿Eres nuevo? [Crea una cuenta](#)



Figura 12 - Login

Para registrarse, en este caso si diferenciamos entre clientes y tiendas. En la *Figura 13* se muestra el registro de clientes, con los datos necesarios para el registro, tales como nombre, apellidos, DNI, teléfono, email y contraseña:



**Es hora de llenar tu despensa** ■

Haz tu pedido y recógelo sin esperas

Nombre  Apellidos

DNI  Teléfono

Email

Contraseña

Acepto los [términos y condiciones](#)

[Registrarme](#)

¿Eres una tienda?

[¡Regístrate y empieza a vender!](#)

¿Ya tienes cuenta? [Acceder](#)

Figura 13 - Registro de cliente

Para que se pueda dar de alta una tienda en Ordery, es necesario cumplimentar el siguiente formulario de registro presente en la *Figura 14*, cuyos campos son referentes tanto a la persona de contacto que crea la tienda, como los propios datos fiscales de la misma:

**Incrementa tus ventas con Ordery**



Tus productos estarán visibles para todo el mundo. ¡Solo ventajas!

Nombre  Apellidos  DNI

DNI  Teléfono  Teléfono

**Información sobre tu tienda**

Imagen de la tienda  Logo de la tienda  Nombre de la tienda

 Recomendado: 1920 x 350 px  Recomendado: 115 x 115 px

Nombre Legal  CIF  Descripción breve

Dirección  Ciudad  Provincia

Código postal  Nombre de persona de contacto  URL

Acepto los [términos y condiciones](#)

[Registrarme](#)

¿Ya tienes cuenta? [Acceder](#)

Figura 14 - Registro de una tienda

Una vez usuario se registra, irá redirigido a la página principal de *Ordery* tal y como se muestra en la *Figura 15*Figura 15. El usuario podrá encontrar establecimientos recomendados cerca de su ubicación y productos recomendados aleatorios.

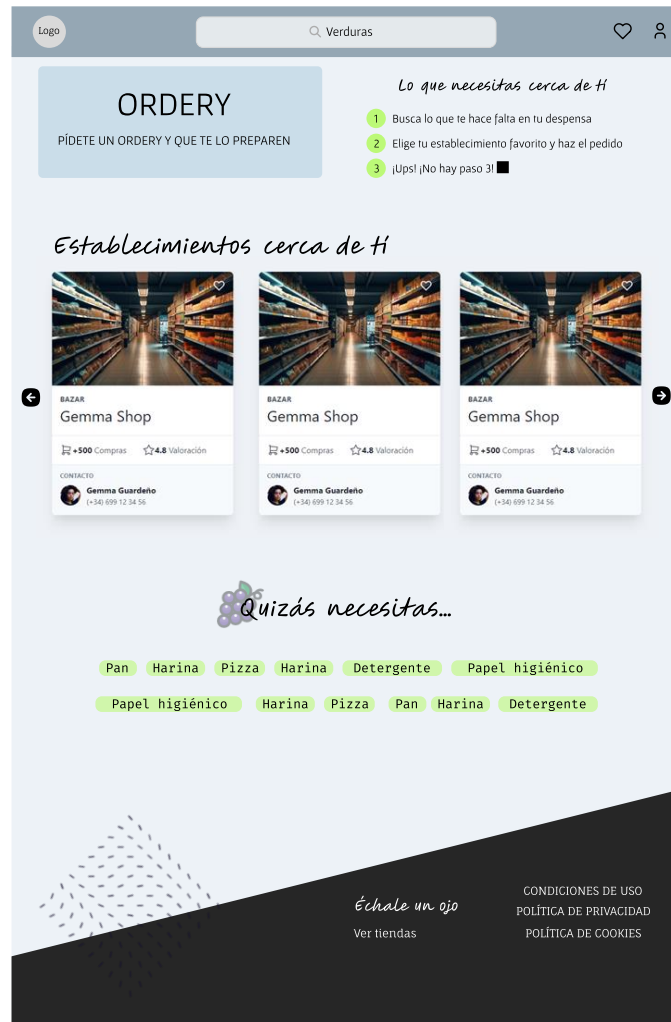


Figura 15- Página principal

En la *Figura 16*Figura 16, se puede ver los resultados de una búsqueda, donde se buscará tanto por tiendas como por productos disponibles en ellas.

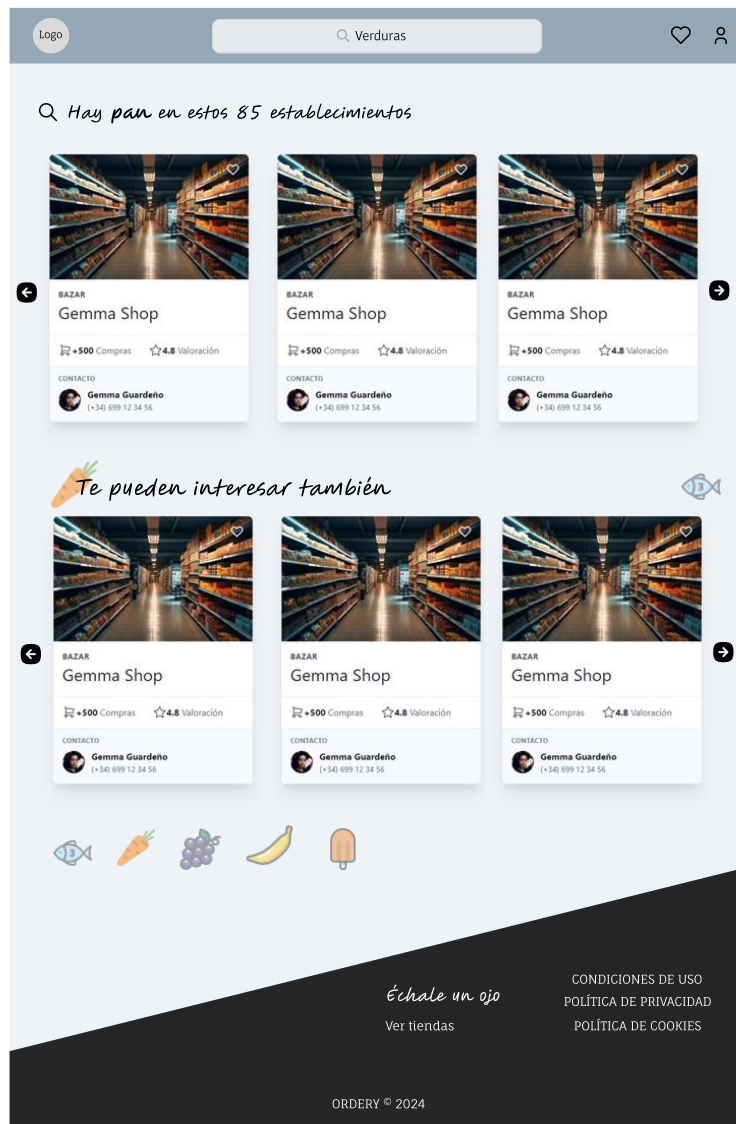


Figura 16 - Página de búsqueda

Al seleccionar una tienda, el usuario puede ver información detallada sobre la misma. En la *Figura 17*, se puede ver la información relacionada con la tienda, los productos disponibles y funciones de solicitud de pedido. Además, también valoraciones y promociones, en caso de que la tienda tenga alguna disponible.

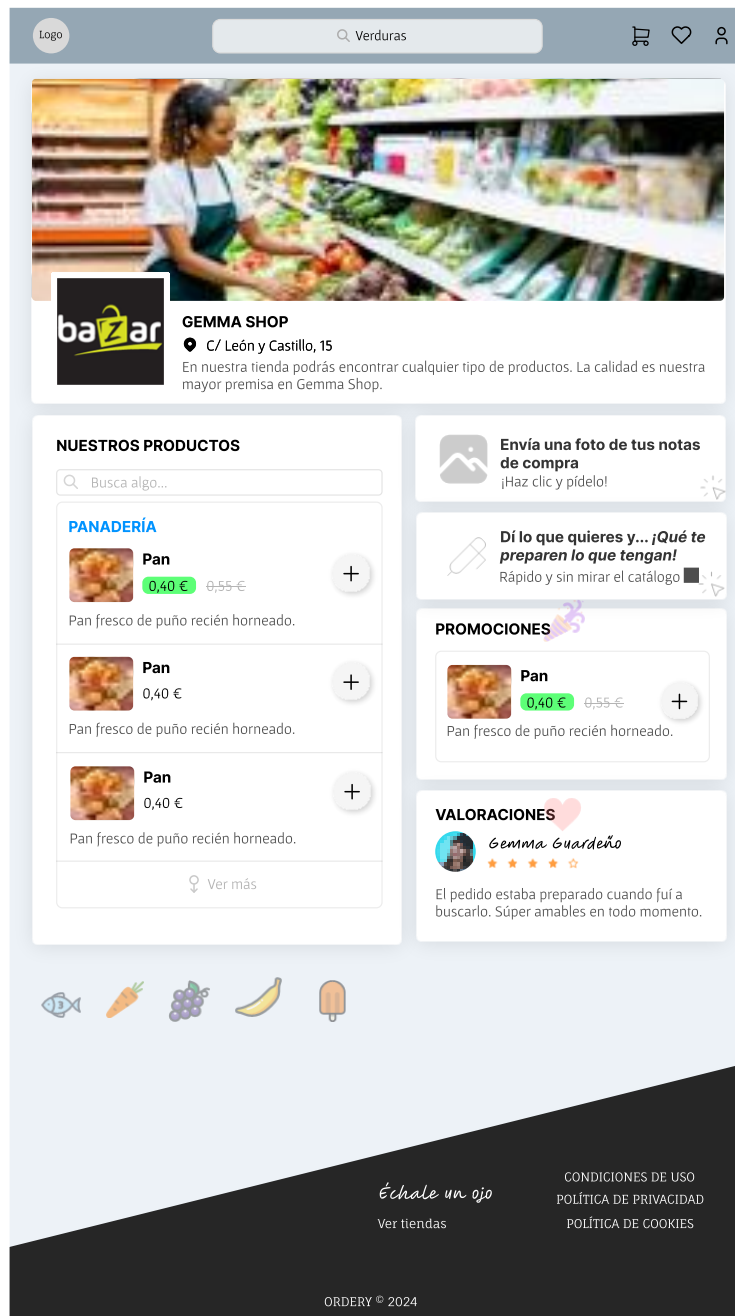


Figura 17 – Vista de tienda

Como proceso final de toda tienda online, es importante tener un *checkout* sencillo y claro. En la *Figura 18* Figura 18, se muestra el listado de productos del carrito, con su cantidad y precio, la dirección de la tienda donde hay que recoger el pedido, el tiempo promedio de preparación y los términos.

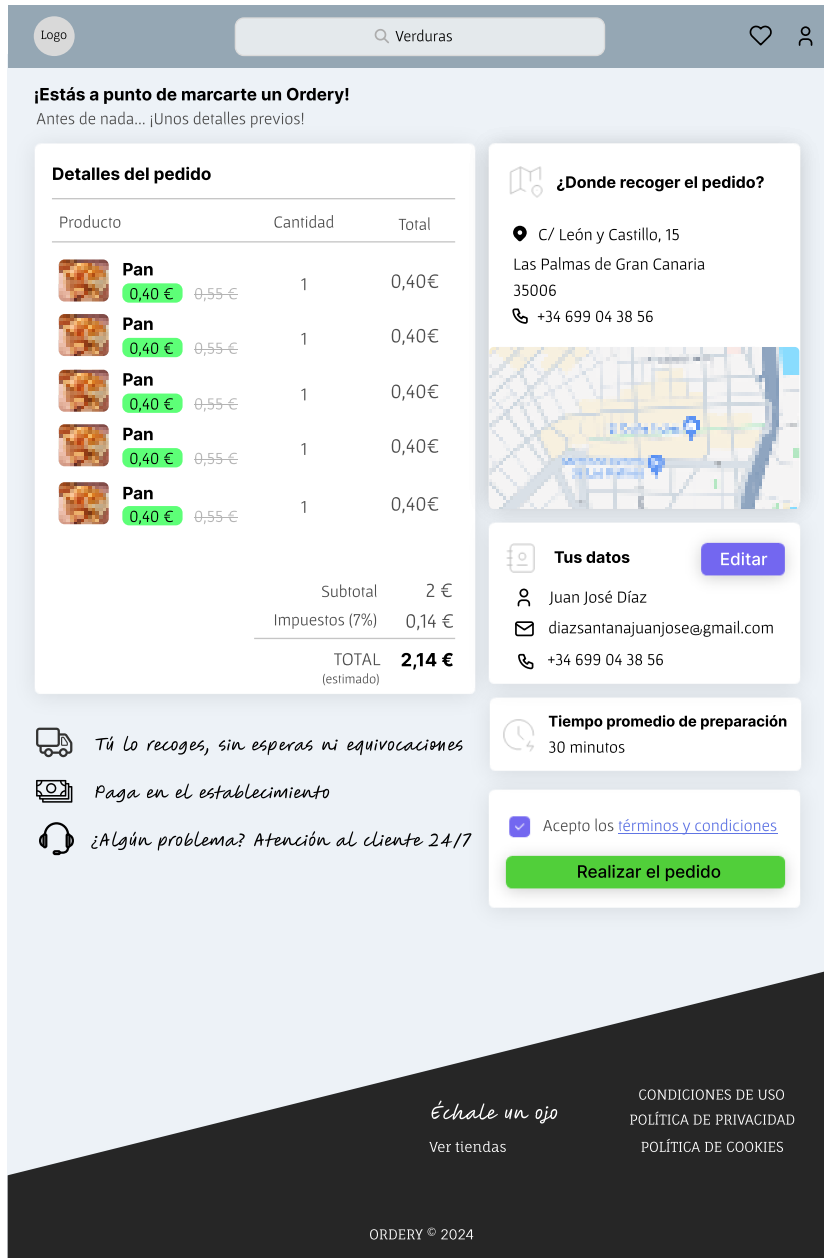


Figura 18 - Vista Checkout

Una vez se realiza el pedido, el usuario puede seguir el estado de este en su perfil de usuario (Figura 19). En esta vista se puede ver el detalle del último pedido y un listado de los últimos pedidos realizados.

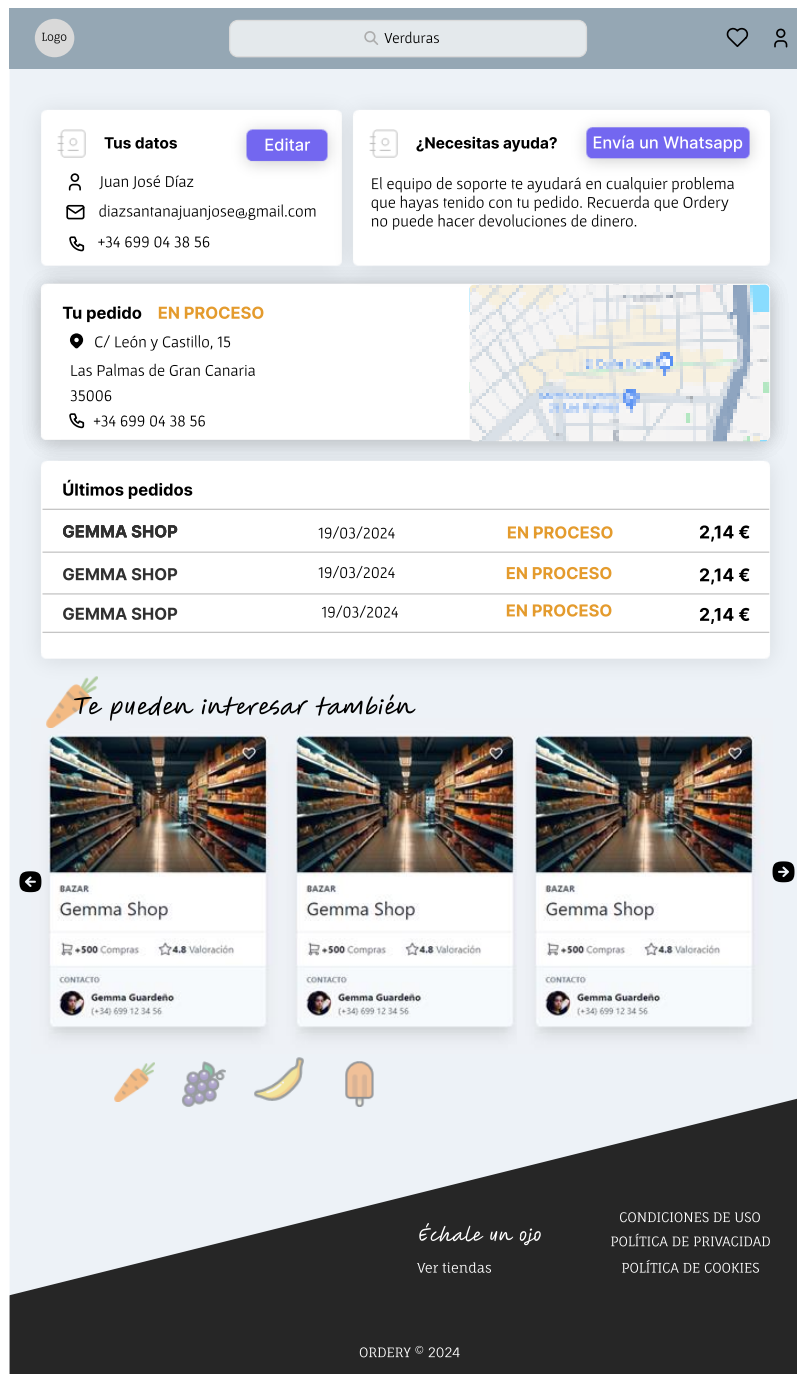


Figura 19 - Perfil de usuario

Cuando el usuario añade productos en el carrito, se pueden visualizar de la siguiente forma, tal y como se representa en la (Figura 20).



Figura 20 - Vista de tienda con productos en el carrito

Cuando una tienda se da de alta en la plataforma, accede directamente al panel de gestión (Figura 21), con un resumen de estadísticas y accesos directos.

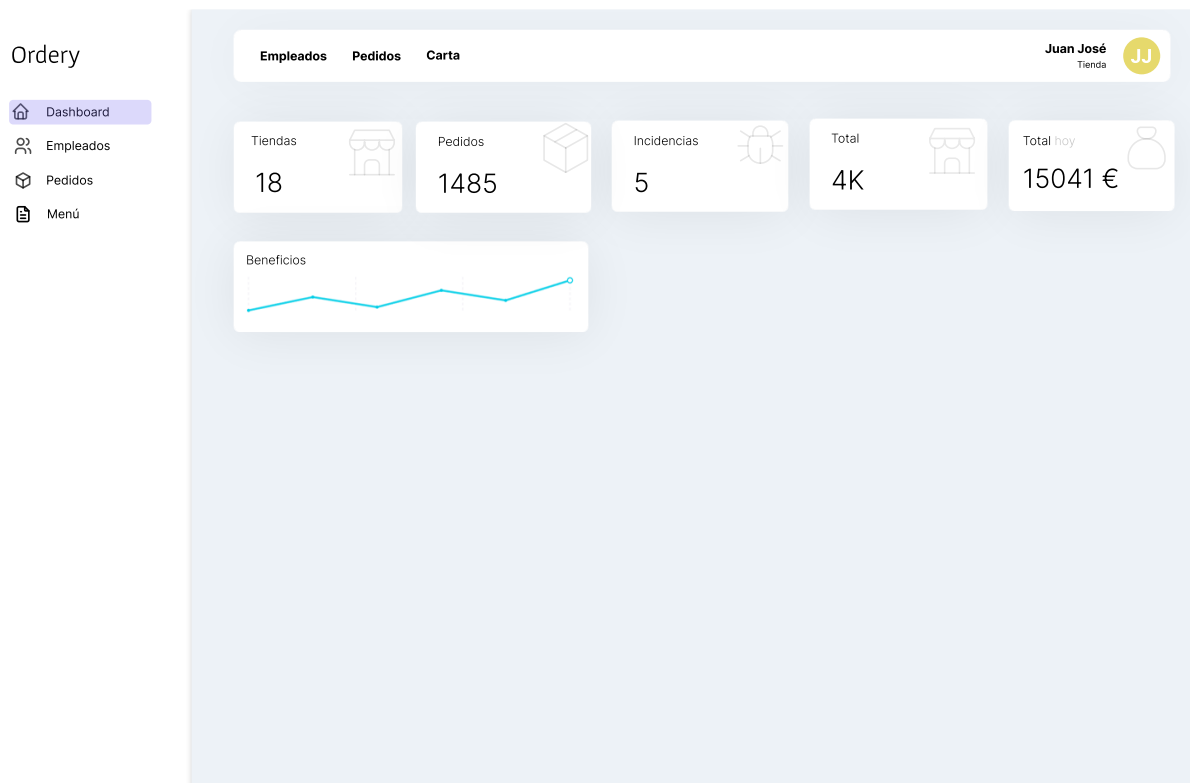


Figura 21 - Dashboard de empleados de una tienda

Como empleado de una tienda, si se tienen los permisos necesarios se puede visualizar y gestionar todos los pedidos de la tienda. En la Figura 22 se muestra un listado coloreado con diferentes colores correspondiente a su estado de pedido.

Ordery

Empleados Pedidos Carta

Juan José  
Tienda JJ

Gestionar pedidos

Ver 10 filas

Buscar

Exportar Búsqueda avanzada Campos + Añadir

# ID	CLIENTE	PEDIDO	OBSERVACIONES	FECHA ENTREGA	FECHA CREACIÓN	ESTADO	ACCIONES
#455112	Juan José Díaz	Ver	¡El dulce que sea de hoy!	12/06/2023 20:34	12/06/2023 20:34	Pendiente	⋮
#455112	Juan José Díaz	Ver	¡El dulce que sea de hoy!	12/06/2023 20:34	12/06/2023 20:34	Preparación	⋮
#455112	Juan José Díaz	Ver	¡El dulce que sea de hoy!	12/06/2023 20:34	12/06/2023 20:34	Preparado	⋮
#455112	Juan José Díaz	Ver	¡El dulce que sea de hoy!	12/06/2023 20:34	12/06/2023 20:34	Finalizado	⋮
#455112	Juan José Díaz	Ver	¡El dulce que sea de hoy!	12/06/2023 20:34	12/06/2023 20:34	Cancelado	⋮

Mostrando registros del 1 al 5 de un total de 5 registros

1

Figura 22 - Pedidos de empleados de una tienda

Al ampliar los detalles del pedido, se abrirá una ventana emergente a la derecha con información sobre el mismo. En la siguiente figura (*Figura 23*) se muestra el boceto elaborado para esta situación.



Orderly

Empleados Pedidos Carta

### Gestionar pedidos

Ver 10 filas

# ID	CLIENTE	PEDIDO	OBSERVACIONES	FECHA ENTREGA	FECHA CREACIÓN
#455112	Juan José Díaz	Ver	¡El dulce que sea de hoy!	12/06/2023 20:34	12/06/2023 20:34
#455112	Juan José Díaz	Ver	¡El dulce que sea de hoy!	12/06/2023 20:34	12/06/2023 20:34
#455112	Juan José Díaz	Ver	¡El dulce que sea de hoy!	12/06/2023 20:34	12/06/2023 20:34
#455112	Juan José Díaz	Ver	¡El dulce que sea de hoy!	12/06/2023 20:34	12/06/2023 20:34
#455112	Juan José Díaz	Ver	¡El dulce que sea de hoy!	12/06/2023 20:34	12/06/2023 20:34

Mostrando registros del 1 al 5 de un total de 5 registros

**Pedido #455112** X

**Estado:** Pendiente

**Productos**

Galletas 1x **5€**

Unidades: 1 uds.  
Precio Ud.: 5€  
Observación: Ninguna  
**SUBTOTAL: 5€**

Dulces 2x **10€**

Unidades: 2 uds.  
Precio Ud.: 10€  
Observación: ¡El dulce que sea de hoy!  
**SUBTOTAL: 20€**

**TOTAL: 25€**

Figura 23 - Detalles del pedido

En la *Figura 24*, se muestra una pizarra de empleados donde la tienda puede gestionar los accesos de estos.

Orderly

Empleados Pedidos Carta

Juan José Tienda JJ

### Gestionar empleados

Ver 10 filas

Buscar

AVATAR	NOMBRE	APELLIDOS	DNI	CORREO ELECTRONICO	TELÉFONO	ACCIONES
JJ	Juan José	Díaz Santana	12345678N	correo@gmail.com	65655887	⋮
JJ	Juan José	Díaz Santana	12345678N	correo@gmail.com	65655887	⋮
JJ	Juan José	Díaz Santana	12345678N	correo@gmail.com	65655887	⋮
JJ	Juan José	Díaz Santana	12345678N	correo@gmail.com	65655887	⋮

Mostrando registros del 1 al 5 de un total de 5 registros

Figura 24 - Pizarra de empleados

Naturalmente, una tienda puede crear o gestionar los empleados existentes. En la *Figura 25* se muestra el formulario con todos los campos relacionados con los empleados.

Ordery

Dashboard  
Empleados  
Pedidos  
Menú

Empleados Pedidos Carta

Juan José ADMIN JJ

### Nuevo empleado

#### Información del empleado

Avatar

Nombre \*

Apellidos \*

DNI \*

Teléfono \*

Cargo

Cuenta bloqueada

#### Accesos

Correo electrónico \*

Repetir correo electrónico \*

Contraseña \*


Repetir contraseña \*

#### Permisos





MÓDULO	VER	EDITAR	BORRAR	SELECCIONAR TODO
Empleados	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Empleados	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Empleados	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Empleados	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

*Figura 25 - Formulario de empleados*

Cambiando de rol de usuario, un administrador de la plataforma tiene acceso privilegiado a toda la información. Tal y como se muestra en *Figura 26*, tienen acceso a todas las tiendas actualmente registradas en la plataforma.

Empleados **Tiendas**
Juan José ADMIN 

### Gestionar tiendas

Ver  filas
Buscar 
 Exportar
 Búsqueda avanzada
 Campos
 Añadir


NOMBRE PÚBLICO	NOMBRE LEGAL	CIF	CORREO ELECTRONICO	TELÉFONO	FECHA CREACIÓN	ACCIONES
El goloso	El Goloso S.L	12345678N	correo@gmail.com	656555887	12/08/2023	⋮
El goloso	El Goloso S.L	12345678N	correo@gmail.com	656555887	12/08/2023	⋮
El goloso	El Goloso S.L	12345678N	correo@gmail.com	656555887	12/08/2023	⋮
El goloso	El Goloso S.L	12345678N	correo@gmail.com	656555887	12/08/2023	⋮

Mostrando registros del 1 al 5 de un total de 5 registros

1

Figura 26 - Pizarra de todas las tiendas


Como en casos anteriores, en la *Figura 27* se maqueta el formulario de tiendas.

Empleados **Pedidos** Carta
Juan José ADMIN 

### Nueva tienda

**Información básica**

Avatar



Nombre público \*

Nombre legal \*

CIF \*

Teléfono \*

Correo electrónico

URL de acceso

Tienda bloqueada

Descripción

**Datos necesarios**

Dirección \*

Provincia \*

Ciudad \*

Código postal \*

Guardar
Cancelar

Figura 27 - Formulario de tienda

También el administrador de la plataforma puede crear empleados que gestionen la misma. En la *Figura 28* se muestra la pizarra relacionada.

Orderly

Dashboard  
Empleados  
Tiendas

Empleados Pedidos Carta

Juan José ADMIN JJ

### Gestionar empleados

Ver 10 filas

Buscar

Exportar Búsqueda avanzada Campos Añadir

AVATAR	NOMBRE	APELLIDOS	DNI	CORREO ELECTRONICO	TELÉFONO	ACCIONES
JJ	Juan José	Diaz Santana	12345678N	correo@gmail.com	656555887	⋮
JJ	Juan José	Diaz Santana	12345678N	correo@gmail.com	656555887	⋮
JJ	Juan José	Diaz Santana	12345678N	correo@gmail.com	656555887	⋮
JJ	Juan José	Diaz Santana	12345678N	correo@gmail.com	656555887	⋮

Mostrando registros del 1 al 5 de un total de 5 registros

1

*Figura 28 - Pizarra de empleados*

En la *Figura 29* se muestra el formulario con los permisos para los empleados.

Orderly

Dashboard  
Empleados  
Tiendas

Empleados Pedidos Carta

Juan José ADMIN JJ

### Nuevo empleado

Información del empleado

Avatar

Nombre \*

Apellidos \*

DNI \*

Teléfono \*

Cargo

Cuenta bloqueada

Accesos

Correo electrónico \*

Repetir correo electrónico \*

Contraseña \*

Repetir contraseña \*

Permisos

MÓDULO	VER	EDITAR	BORRAR	SELECCIONAR TODO
Empleados	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Empleados	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Empleados	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Empleados	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Guardar Cancelar

*Figura 29 - Formulario de empleados*

## 9. Desarrollo

En este apartado se abordará todo lo relacionado sobre *Ordery*.

*Ordery* es una tienda online que pretende simplificar el proceso de compra, haciendo que los clientes puedan realizar un pedido de una forma rápida, tanto seleccionando productos de un catálogo, como enviando una foto o escribiendo manualmente lo que deseen.

Por otro lado, *Ordery* también pretende que los pequeños comercios tengan un portal donde visibilizarse y poder gestionar los pedidos.

La plataforma tiene como idea principal, simplificar y optimizar el tiempo que los usuarios pasan en una tienda física seleccionando los productos, reduciendo el proceso a un modelo de negocio “*click & collect*”, donde pagan al recoger.

En los siguientes apartados, abordaremos más extensamente todo el proceso llevado a cabo.

### 9.1 Lenguaje de programación

El lenguaje de programación que se ha elegido para el desarrollo del aplicativo ha sido Laravel. Se ha elegido principalmente este lenguaje ya que su amplio mercado permite encontrar bastante documentación en internet y, sobre todo, el conocimiento que el autor del trabajo tenía con este lenguaje, hacía que se elaboraran funciones más complejas y se indagara mucho más en profundidad en seguridad.

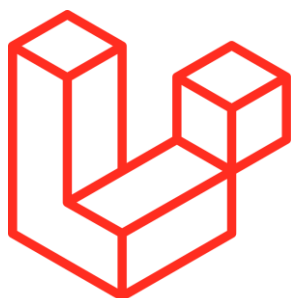


Figura 30 - Logo Laravel

Laravel, tal y como se comentaba en apartados anteriores, es un *framework* de código PHP que permite la elaboración de aplicaciones y servicios web de una forma rápida, cómoda y potente.

Laravel tiene como dependencia otro *framework* llamado *Symfony*, que también se emplea en lenguajes conocidos como *Ruby on Rails*.

Una característica reseñable del lenguaje es que permite a la arquitectura de software para modelo-vista-controlador (MVC).

### 9.2 Estructura del proyecto

El proyecto se ha llevado a cabo utilizando la estructura común de Laravel, donde se ha hecho especial énfasis en respetar la arquitectura y modular el código en subcarpetas con el fin de reutilizar y ordenar el desarrollo.

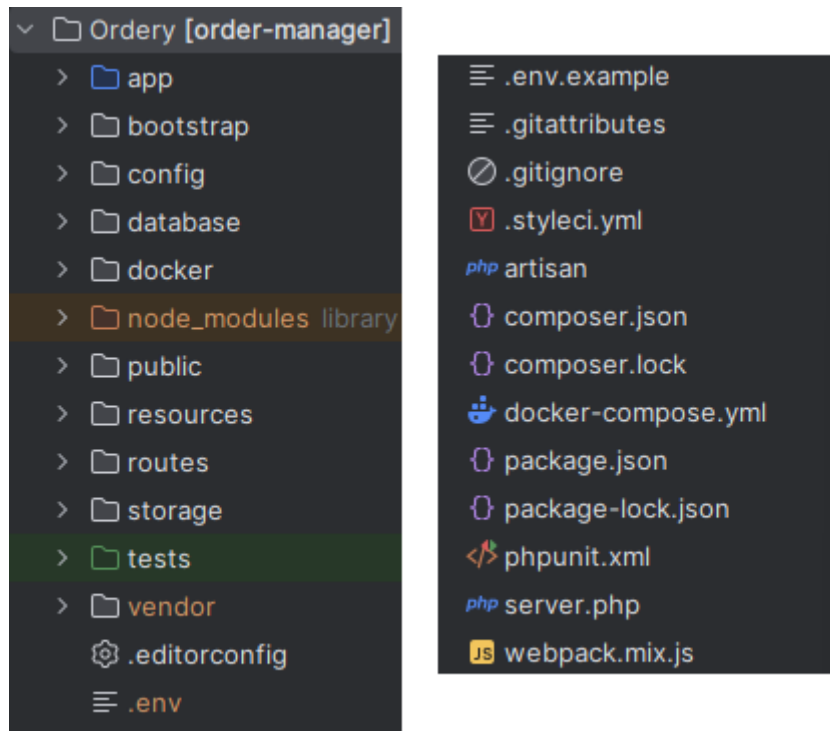


Figura 31 - Estructura de Laravel

En la carpeta *App* encontramos archivos importantes como comandos de terminal que creamos, controladores donde se alojará el código de lado de servidor, los modelos que negociarán con la base de datos y archivos de validación.

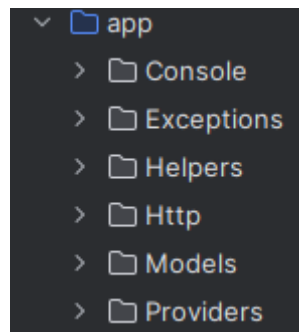


Figura 32 - Carpeta App

En la *Figura 33* se puede ver un ejemplo de un controlador, con una consulta de *eloquent* (herramienta integrada en Laravel que interactúa con la base de datos y devuelve los datos en un formato correcto) y una redirección de una vista.

```

15 class WebpagesController extends Controller
16 {
17
18     // HOMEPAGE
19     Juan José +1 *
20     public function home()
21     {
22         $tiendas = Tienda::query()→inRandomOrder()→limit( value: 10)→get();
23         return view( view: '/web/pages/home/index', compact( var_names: 'tiendas'));
24     }
25 }

```

Figura 33 - Ejemplo de controlador

Para que la consulta anterior pudiera realizarse sin ningún problema, es necesario que previamente exista un modelo. En este caso, el modelo de tiendas. En la *Figura 34*, se muestra un ejemplo del contenido de un modelo.

```

9 class Tienda extends Model
10 {
11     use HasFactory, Notifiable;
12
13     The attributes that are mass assignable.
14     array
15
16     protected $fillable = [
17         'nombre',
18         'nombre_legal',
19         'cif',
20         'telefono',
21         'email',
22         'url',
23         'is_blocked',
24         'imagenes',
25         'ciudad',
26         'provincia',
27         'descripcion',
28         'codigo_postal',
29         'direccion',
30     ];
31
32     no usages Juanjo
33
34     public function categorias(){
35         return $this→hasMany( related: Categoria::class, foreignKey: 'id_tienda', localKey: 'id');
36     }
37
38     Juanjo
39     public function productos(){
40         return $this→hasMany( related: Producto::class, foreignKey: 'id_tienda', localKey: 'id');
41     }
42 }

```

Figura 34 - Ejemplo de modelo

En la Figura anterior, se puede comprobar como existe una variable “\$fillable” que hace referencia a cada uno de los campos de la tabla de la base de datos. Esta variable es opcional, pero su uso permite utilizar métodos que simplifican crear o actualizar en base de datos,

siempre y cuando por método *POST* en *http* las variables coinciden con el nombre de la columna.

Además, se puede ver dos relaciones de base de datos. Una relación llamada “categorias” que permite traer las categorías relacionadas con la tienda. Por otro lado, encontramos una relación “productos”, que al igual que la relación anterior, permite traer los productos relacionadas con la tienda.

Esto permite simplificar en gran medida las consultas a la base de datos y los tiempos de respuesta, puesto que, sin relaciones habría que hacer tres consultas distintas y recorrer cada uno de los resultados e igualarlos. En este caso, solo con una consulta obtenemos todos los datos relacionados correctamente y de forma optimizada.

La carpeta *config* posee información relevante sobre la configuración del proyecto y diferentes áreas de este. En estos archivos se pueden encontrar variables de entornos con valores por defecto que se usarán en el proyecto. Estos pueden ser modificados y consultados desde los controladores.

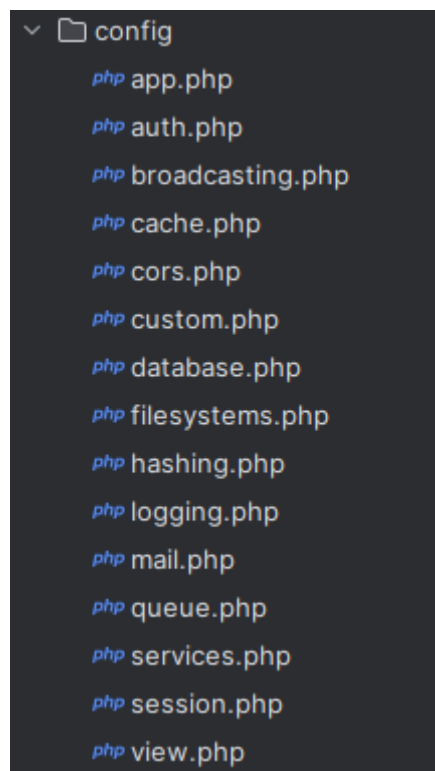


Figura 35 - Carpeta *config*

La carpeta *database* se encuentra los principales archivos correspondientes a la base de datos. En la *Figura 36* se puede ver la existencia de las carpetas “*factories*” y “*seeders*” cuya finalidad es poblar la base de datos con información aleatoria o estándar. La carpeta “*migrations*” son archivos que permite crear la base de datos con la estructura y relaciones correspondientes.



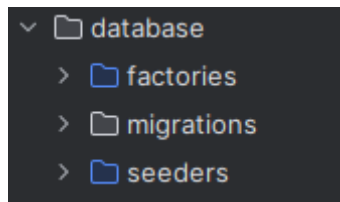


Figura 36 - Carpeta database

Las migraciones se deben informar tanto del tipo de dato de la columna, como su nombre. Además, como se puede apreciar al final de la *Figura 37*, también hay que indicar si la columna es una llave foránea de otra tabla. Con esto conseguiremos que el motor de datos de MySQL funcione a pleno rendimiento y eficacia.

```
public function up()
{
    Schema::create( table: 'pedidos', function (Blueprint $table) {
        $table->id();
        $table->string( column: 'doc');
        $table->longText( column: 'info_pago');
        $table->longText( column: 'pedido');
        $table->float( column: 'precio');
        $table->longText( column: 'observaciones');
        $table->dateTime( column: 'fecha_entrega');
        $table->string( column: 'estado');
        $table->timestamps();

        $table->foreignId( column: 'id_user')->nullable()->references( column: 'id')
            ->on( table: 'users')->onUpdate( action: 'CASCADE')->onDelete( action: 'SET NULL');
        $table->foreignId( column: 'id_tienda')->nullable()->references( column: 'id')
            ->on( table: 'tiendas')->onUpdate( action: 'CASCADE')->onDelete( action: 'SET NULL');
    });
}
```

Figura 37 - Ejemplo de una migración

En la carpeta *public* se encontrarán todos los archivos que entran en juego para el funcionamiento de la web compilados o minificados (reducido en tamaño).

En la carpeta *resources* se encuentran todos los archivos que se usan para el desarrollo del aplicativo. Encontraremos tanto las vistas, las fuentes, las imágenes, los estilos, los scripts y cualquier otro material que se desee añadir.

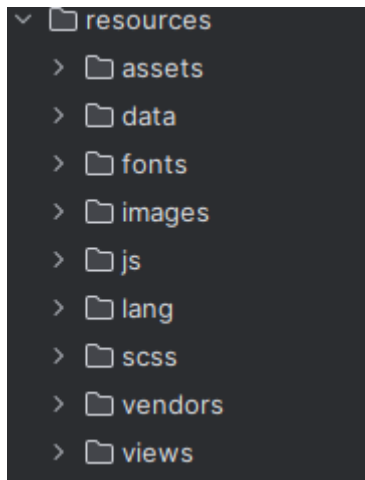


Figura 38 - Carpeta resources

En la carpeta *routes* encontramos los archivos relacionados con la gestión de las rutas y redireccionamiento dentro de la aplicación.

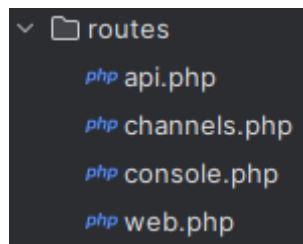


Figura 39 - Carpeta routes

En este proyecto se ha trabajado sobre el archivo “*web.php*”, y contiene todas las rutas correspondientes de la aplicación tal y como se muestra en la siguiente *Figura 40*.

```

1 <?php
2
3 > use ...
12
13 Auth::routes();
14
15 Route::get(uri: '/maintenance', [WebpagesController::class, 'maintenance'])->name(name: 'maintenance');
16 Route::get(uri: '/', [WebpagesController::class, 'home'])->name(name: 'home');
17 Route::get(uri: '/shops/{url}', [TiendaController::class, 'tienda'])->name(name: 'tienda');
18 Route::get(uri: '/shops', [WebpagesController::class, 'search'])->name(name: 'search');
19 Route::get(uri: '/busqueda', [WebpagesController::class, 'busqueda'])->name(name: 'buscador');
20 ⚠ Route::get(uri: '/logout', function (Request $request) { Undefined class 'Request'
21     Auth::logout();
22
23     return redirect(to: '/login');
24 });
25
26 Route::group(['middleware' => ['auth']], function () {
27     Route::group(['prefix' => 'admin', 'as' => 'admin.', 'middleware' => ['rol:admin']], function () {
28         Route::get(uri: 'dashboard', [DashboardController::class, 'indexAdmin'])->name(name: 'dashboard');
29         Route::get(uri: 'perfil', [UserController::class, 'perfil'])->name(name: 'perfil');
30
31         EMPLEADOS *
32
33         Route::prefix(prefix: 'empleados.')->group(function () {
34             Route::group(['middleware' => ['permission:Empleados,Leer']], function () {
35                 Route::get(uri: 'pizarra', [UserController::class, 'index'])->name(name: 'pizarraEmpleados');
36                 Route::post(uri: 'json', [UserController::class, 'getDataJson']);
37                 Route::get(uri: 'show/{id}', [UserController::class, 'show']);
38             });
39         });
40     });
41 }

```

Figura 40 - Fragmento del archivo "web.php"

En la figura anterior, se puede comprobar diferentes aspectos importantes.

- Hay rutas que no se encuentran protegidas porque no es necesario, puesto que cualquier persona puede acceder.
- Hay rutas que están protegidas por “*middleware*” de “*auth*” que es un archivo que valida que el usuario que está intentando acceder tenga los permisos o derechos necesarios para ello. En este caso, el archivo “*auth*” simplemente valida que el usuario esté logueado o no, donde en su defecto, se le redirigirá a la vista del *login*.
- Al igual que el *middleware* anterior, hay rutas protegidas donde se valida el rol del usuario (si es administrador, un empleado de una tienda o cliente) o incluso, si tiene un permiso en específico para acceder a esa ruta.

En la carpeta *storage* se encuentra datos almacenados tanto de Laravel como los propios que se puedan almacenar en el aplicativo (como, por ejemplo, archivos que sube un usuario).

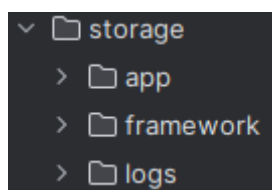


Figura 41 - Carpeta storage

En cuanto a los archivos que están en la carpeta raíz del proyecto, la mayoría son de configuración del entorno de trabajo. Los más importantes a destacar son los siguientes:

- a. El archivo “.env” que contiene variables de entorno relevantes como la conexión a la base de datos, las credenciales de email para el envío de correos, configuración de variables, entre otros.
- b. El archivo “composer.json” que contiene todos los paquetes y las dependencias de los mismos de la herramienta *composer*, para que el proyecto pueda instalar los recursos necesarios y su versión en cualquier lugar.
- c. El archivo “package.json” contiene todos los paquetes y dependencias como el archivo anterior, pero en este caso de la librería *NodeJs*.
- d. El archivo “webpack.mix.js” que contiene un listado de directorios y archivos que deben ser procesados para minificar y compilar para el correcto funcionamiento.

## 9.3. Funciones implementadas destacadas

En este apartado abordaremos algunas de las funciones implementadas. Una vez detallado los bocetos en el apartado de Mockups, se han llevado a cabo los desarrollos pertinentes para hacer realidad la idea original.

### 9.3.1. Búsqueda de tiendas o productos

Para que a los usuarios la búsqueda sea la más cómoda posible en cuanto a rendimiento, se ha optimizado las consultas que se realizan a la base de datos para aprovechar el máximo de sus prestaciones, tanto del motor MySQL como de Laravel.

En la *Figura 42*, se muestra la consulta que se realiza tras una búsqueda del usuario. Lo más importante, es que la petición pregunte tanto por tiendas activas como por productos que contengan esas tiendas, de esta forma, cubrimos todo el abanico que le interesaría al usuario.

En dicha función, se incluye diferentes cláusulas “*where*” con el que agrupamos dos condiciones importantes. La primera condición, la más sencilla, se buscará aquellas tiendas que no estén bloqueadas o desactivadas en *Ordery*. La segunda condición agrupará dos escenarios importantes para la búsqueda fluida en la plataforma. El primero de ellos es obtener aquellas tiendas cuyo nombre coincida con el criterio de búsqueda. Por otro lado, el segundo escenario consiste en obtener también aquellas tiendas que, aunque su nombre no coincida con el criterio de búsqueda, tenga algún nombre de producto que sí lo haga.

En adición a todo lo anterior, es importante paginar (fraccionar los resultados) para no traerle al usuario todos los resultados al mismo tiempo. En un entorno de pruebas se puede traer todos los resultados sin problemas, pero de cara a lanzar a producción la plataforma, números muy grandes de productos y tiendas sería inviable obtener todos los resultados al mismo tiempo, por cuestiones de rendimiento y salud del servidor.

```

public function busqueda(Request $request){

    if(!isset($request->search) || $request->search == ""){
        return view( view: '/web/pages/home/index');
    }

    $busqueda = $request->search;

    $tiendas = Tienda::query()
        ->where(function ($query) use ($busqueda) {
            $query->where('nombre', 'LIKE', "%$busqueda%")
                ->orWhereHas('categorias', function ($q) use ($busqueda) {
                    $q->whereHas('productos', function ($q2) use ($busqueda) {
                        $q2->where('nombre', 'LIKE', "%$busqueda%");
                    });
                });
        })
        ->where( column: 'is_blocked', operator: '=', value: 0)
        ->with(['productos'=>function($q) use ($busqueda){
            $q->where('nombre', 'LIKE', "%$busqueda%")->select('id', 'nombre', 'id_tienda');
        }])

        ->paginate( perPage: 8);

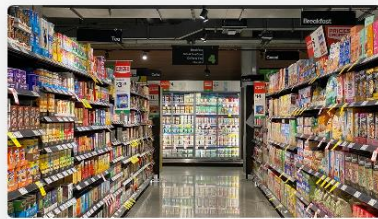
    return view( view: '/web/pages/search/index', [
        'busqueda' => $busqueda,
        'tiendas' => $tiendas,
    ]);
}

```

Figura 42 - Lógica de búsqueda

En definitiva, la consulta busca tiendas y productos de forma unificada y eficiente, completándose en tiempos muy inferiores al segundo (milisegundos), mostrando las tiendas relacionadas con la búsqueda como aquellas que tengan el producto relacionado que coincida con el criterio, tal y como se muestra en la *Figura 43*.

¿Hay "Pan" en estos 9 establecimientos



#### Bazar Gemma

Vendemos productos variados para todas las necesidades. Desde primeras necesidades hasta ¡chocolate!

📍 C/ Luis Morote, 120  
Las Palmas de Gran Canaria

Este establecimiento tiene 1 productos que coinciden con "Pan".



#### Food Shop

Gran variedad de productos cárnicos y verduras frescas.

📍 C/ Paseo de Chill, S/N  
Las Palmas de Gran Canaria

Este establecimiento tiene 1 productos que coinciden con "Pan".



#### Las Mascotas. Pan

Mima a tu mascota con los mejores productos 🐾.

📍 C/ La Unión, 45  
Vecindario

Figura 43 - Resultados de una búsqueda con el criterio "Pan".

### 9.3.2. Pedidos

En el momento de realizar un pedido, es importante tener en cuenta factores de seguridad. Aunque en algunas de las tareas futuras se implementará aún más algunas de estas cuestiones, se ha contemplado el control de fallos en el lado del servidor.

Es muy común que por algún motivo el pedido falle, así sea por lo que el usuario introduzca o por algún error interno durante el proceso de guardado.

```
public function store(Request $request) {  
  
    try {  
        DB::beginTransaction();  
        $userId = $this->user->id;  
  
        $precioTotal = $this->calcularPrecioTotal($request->productos['productos']);  
        $pedido = new Pedido();  
        $pedido->doc = uniqid( prefix: $request->tiendaId . '-' . $userId . '-');  
        $pedido->info_pago = "Sin pago";  
        $pedido->pedido = json_encode($request->productos);  
        $pedido->precio = $precioTotal;  
        $pedido->observaciones = $request->observaciones;  
        $pedido->estado = 'Pendiente';  
        $pedido->id_user = $this->user->id;  
        $pedido->id_tienda = $request->tiendaId;  
        $pedido->save();  
  
        DB::commit();  
        return response(['pedidoDoc' => $pedido->doc], status: 200);  
    } catch (\Exception $e) {  
        DB::rollBack();  
        return response(['errors' => ['errores' => ['No se ha podido generar el pedido.']]], status: 500);  
    }  
}
```

Figura 44 - Lógica de guardado de pedidos

Para tener en cuenta estas casuísticas, se ha implementado el uso de la declaración “try” y “catch”. Si el código que se introduce dentro del “try” falla, se ejecutará el código del “catch”, donde este ejecutará un “rollback”. Este método desharrá cualquier acción llevada a cabo en la base de datos desde que se hizo el último “beginTransaction”. En el caso completamente opuesto, si el guardado es satisfactorio y no ha surgido ningún error, se realizará un “commit”, actualizando la base de datos con los cambios.

Además, como parte del proceso de muestra de errores al usuario, si surge algún error se devuelve uno o varios mensajes de error que se mostrarán en forma de alerta en el *front*.

Otro factor de seguridad es validar los precios en el lado de servidor, y nunca “crear” en los datos que envía el *front*, puesto que pueden estar manipulador por los usuarios. En la *Figura 45* se muestra la lógica que se realiza para calcular el precio del pedido, recorriendo cada uno de los productos.

```
public function calcularPrecioTotal($productos) {
    $total = 0;
    foreach ($productos as $producto) {
        if($producto['tipo'] == 'producto'){
            $total += (floatval($producto['precio']) * intval($producto['cantidad']));
        }
    }
    $impuesto = 1.07;
    return $total * $impuesto;
}
```

Figura 45 - Lógica de cálculo de precio

Hay que tener en cuenta en la figura anterior que esto solo funcionará para aquellos productos físicos introducidos por las tiendas, puesto que las imágenes o textos subidos por los usuarios no se conoce el importe total hasta que un empleado de un comercio lo establece posteriormente.

### 9.3.3. Pizarras

Lo más importante para un empleado de una tienda o de la plataforma es que pueda acceder fácilmente a la información que precise en ese momento. La forma más apropiada de abordar este asunto de forma eficiente en rendimiento es implementar la librería *datatables*, que permite cargar de forma dinámica y optimizada una tabla, facilitando así su filtrado, acciones y personalización de los campos.

En la *Figura 46*, se muestra la consulta inicial del *datatable* para cargar los pedidos.

Primeramente, se comprueba los permisos que tiene el usuario para poder personalizar las acciones de cada una de las filas.

Posteriormente, se llama a la clase *datatables* para generar una consulta. Podemos encontrar diferentes métodos que permiten personalizar la consulta:

1. *addColumn*: Permite añadir columnas personalizadas que no existen en la base de datos a la tabla, por ejemplo, los permisos de los usuarios. No necesariamente estas columnas deben ser visibles.

2. *editColumn*: Permite modificar el campo que se obtiene de la base de datos, por ejemplo, cambiar el formato de la fecha de “Y-m-d” a “d/m/Y”.
3. *filterColumn*: Permite modificar el criterio de filtrado. Normalmente esto es necesario si se necesita filtrar por una relación compleja en la base de datos.
4. *orderColumn*: Permite modificar el criterio de ordenación. Al igual que el anterior, esto es necesario si se necesita filtrar por una relación compleja en la base de datos.

```
public function getDataJson(Request $request) {
    // Traemos todos los pedidos para la tienda
    $pedidos = Pedido::query()>where( column: 'id_tienda', $this->user->tienda_id)->with(['cliente' => function($q){
        $q->select('id', 'nombre', 'apellidos');
    }]);

    $permisoLeer = $this->user->hasPermiso('Pedidos', 'Leer');
    $permisoEditar = $this->user->hasPermiso('Pedidos', 'Editar');
    $permisoBorrar = $this->user->hasPermiso('Pedidos', 'Borrar');

    return DataTables::eloquent($pedidos)
        ->addColumn( name: 'permiso_leer', function ($model) use($permisoLeer){
            return $permisoLeer;
        })
        ->addColumn( name: 'permiso_borrar', function ($model) use($permisoBorrar){
            return $permisoBorrar;
        })
        ->addColumn( name: 'permiso_editar', function ($model) use($permisoEditar){
            return $permisoEditar;
        })
        ->editColumn( name: 'user_id', function ($model){
            return $model->cliente->nombre . ' ' . (isset($model->cliente->apellidos) ? $model->cliente->apellidos : '');
        })
        ->editColumn( name: 'fecha_entrega', function ($model){
            if(isset($model->fecha_entrega)){
                return Carbon::createFromFormat( format: 'Y-m-d H:i:s', $model->fecha_entrega)->format( format: 'd/m/Y H:i');
            }
            return '-';
        })
        ->editColumn( name: 'created_at', function ($model){
            return Carbon::createFromFormat( format: 'Y-m-d H:i:s', $model->created_at)->format( format: 'd/m/Y H:i');
        })
        ->toJson();
}
```

Figura 46 - Lógica de lado de servidor del datatable

En el lado del front, mediante Javascript, inicializamos el datatable. Con el objetivo de modular el código, se ha creado un método único que realiza todo el proceso del *datatable*, tal y como se muestra en la *Figura 47*.

```
// Inicializamos el datatable
initDatatable(columns, columnDefs, nameCrud, tableClass, options);
```

Figura 47 - Método de inicialización datatable

Para que funcione el método, es necesario inicializar cada una de las variables que entran en juego.

En la *Figura 48*, se define las columnas que participan en la tabla. Estos valores deben coincidir con el nombre de la columna de la base de datos. Si esto no fuera así, habría que



elaborar manualmente el *orderColumn* y el *filterColumn* anteriormente mencionados para que el *datatable* pudiera funcionar correctamente.

```
const columns : [(data: string), (data: string... = [
  // columns according to JSON RESPONSE
  {data: 'id'}, // vista movil
  {data: 'id'}, // checkbox
  {data: 'doc'},
  {data: 'user_id'},
  {data: 'pedido'},
  {data: 'observaciones'},
  {data: 'fecha_entrega'},
  {data: 'created_at'},
  {data: 'estado'},
  {data: 'id', className: 'not-export-col'}, // Acciones
];
```

Figura 48 - Definición de la variable "columns"

En la Figura 49, se define el contenido de las columnas. Por regla general, si lo que se desea es mostrar lo mismo que está en la base de datos, no habría que implementar esa columna. Por el contrario, si se desea añadir o personalizar ligeramente el dato, habría que desarrollarlo dentro de *columnDefs*.

```
const columnDefs : [(orderable: boolean, responsi... = [
{
  // For Responsive
  className: 'control',
  orderable: false,
  responsivePriority: 2,
  targets: 0
},
{
  // For Checkboxes
  targets: 1,
  orderable: false,
  responsivePriority: 3,
  render: function (data, type, full, meta) :string {
    return (
      '<div class="form-check form-check-primary custom-checkbox"> <input class="form-check-input dt-checkboxes" type="checkbox"
      data +
      '" /><label class="custom-control-label" for="checkbox" +
      data +
      '"></label></div>'
    );
  },
  checkboxes: {
    selectAllRender:
      '<div class="form-check form-check-primary custom-checkbox"> <input class="form-check-input" type="checkbox" value="" id="d
  }
},
{
  // PEDIDO
  targets: 4,
  orderable: false,
  responsivePriority: 3,
  render: function (data, type, full, meta) :string {
    return (
      '<button class="btn btn-outline-primary waves-effect" type="button" onclick="mostrarPedido('${full.id}', '${btoa(data)}')">
      Ver
      </button>'
    );
  }
},
];
```

Figura 49 - Definición de la variable "columnDefs"

Como extensión desarrollada para ampliar el funcionamiento de *datatable*, se ha añadido filtros para que funcione de forma automática y de una forma intuitiva y fácil. En la *Figura 50* se define la variable "filtros" que incluye cada uno de los campos y el tipo por el que se

podrá filtrar en la tabla. Es importante señalar el número de columna al que hace referencia para que el *datatable* funcione correctamente.

```
const filtros :[{column: number, className: s... = [  
  {  
    title: 'Nombre',  
    column: 3,  
    className: 'col-2 px-50',  
    type: 'text',  
  },  
  {  
    title: 'Apellidos',  
    column: 4,  
    className: 'col-2 px-50',  
    type: 'text',  
  },  
  {  
    title: 'DNI',  
    column: 5,  
    className: 'col-2 px-50',  
    type: 'text',  
  },  
  {  
    title: 'Correo electrónico',  
    column: 6,  
    className: 'col-2 px-50',  
    type: 'text',  
  },  
]
```

Figura 50 - Definición de la variable "filtros"

```
const options : = {  
  order: [6, 'asc'],  
  prefix: '/tienda',  
  exportOptions: {  
    exportButtonPrint: {active: true, exportOptions: {columns: ":visible:not(.not-export-col)"}},  
    exportButtonCSV: {active: true, exportOptions: {columns: ":visible:not(.not-export-col)"}},  
    exportButtonExcel: {active: true, exportOptions: {columns: ":visible:not(.not-export-col)"}},  
    exportButtonPdf: {active: true, exportOptions: {columns: ":visible:not(.not-export-col)"}},  
    exportButtonCopy: {active: true, exportOptions: {columns: ":visible:not(.not-export-col)"}},  
    exportButtonBorrar: {  
      active: true,  
      exportOptions: {columns: ":visible:not(.not-export-col)"}  
    },  
  },  
  addButton: {  
    active: false,  
  },  
  domOptions: {  
    domRows: true,  
    domSearch: true,  
    domFiltros: true,  
  },  
  titleModal: {  
    data: ['nombre', 'apellidos']  
  },  
  filtros: filtros,  
  rowCallback: rowDrawCallback  
}
```

Figura 51 - Definición de la variable "options"

Por otro lado, en la *Figura 51*, se configura el *datatable* para que funcione como se necesita, como filtros, buscador, botón de añadir, botones de exportación, etc.

Por último, las dos variables restantes es “*nameCrud*”, que hace referencia al nombre que se le ha asociado a la clase de HTML, en este caso “pedidos”, y la variable “*tableClass*”, que igual que el anterior, es la clase referenciada al HTML, en este caso “pedidos-list-table”.

Con esto se consigue una tabla completamente funcional y con un rendimiento excepcional, sin importar la cantidad de contenido ya que se carga dinámicamente y la potencia que proporciona.

### 9.3.4. Carta

Una de las funciones a destacar para las tiendas, es que pueden elaborar su carta (o catálogo) de una forma muy sencilla. Haciendo uso de la librería “*dragula*” (librería de JavaScript que facilita la función “*drag and drop*”) se consigue que el usuario tenga una experiencia cómoda arrastrando y soltando elementos, ordenándolos a su criterio.

```
function initCardsDraggable() : void {
  destroyListsDraggable();
  dragulaCard = dragula([document.getElementById( elementId: 'card-drag-area')]);

  $('#card-drag-area .card-body, .btn-add-producto').fadeOut(function () : void {
    $('.btn-edit-producto, .btn-delete-producto').fadeOut(function () : void {
      $('.list-group').fadeOut();
      $('.btn-add-categoria').fadeIn();
      $('.btn-delete-categoria').fadeIn();
      $('.btn-edit-categoria').fadeIn();
    });
  });
}
```

*Figura 52 - Inicialización de dragula para categorías*

En la inicialización de *dragula*, primeramente, se comprueba que no hay ningún componente ya haciendo uso de la librería, puesto que esto generaría errores de duplicidad si se inicializa dos veces. Además, como se puede cambiar entre arrastrar y soltar de categorías y productos, se debe tener en cuenta que hay de destruir el componente que no se desea mover. En la *Figura 53*, se ha desarrollado métodos para destruir de forma rápida y cómoda los componentes *dragula*.

```

function destroyCardsDraggable() : void {
  if(typeof(dragulaCard) ≠ 'undefined') {
    dragulaCard.destroy();
  }
}

function destroyListsDraggable() : void {
  if(typeof(dragulaList) ≠ 'undefined'){
    dragulaList.destroy();
  }
}

```

Figura 53 - Lógica de creación y destrucción dragula

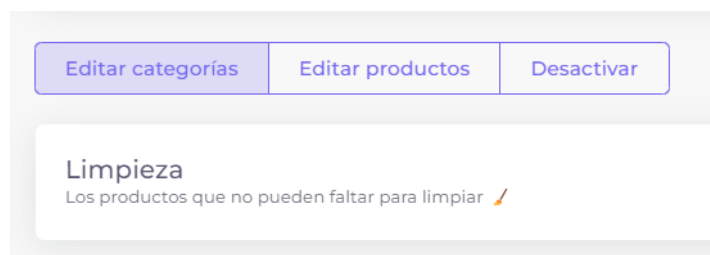


Figura 54 - Editar categorías en la carta

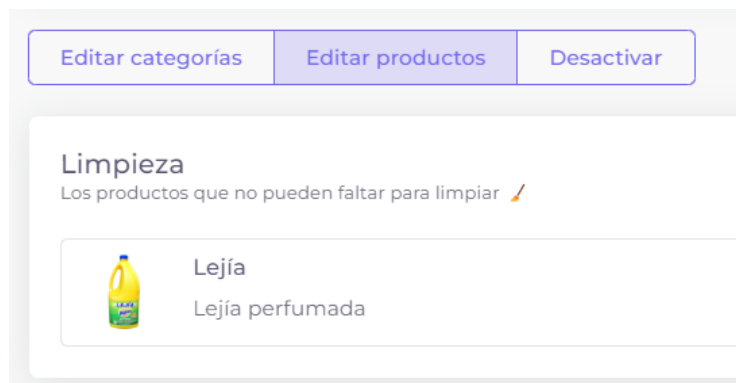


Figura 55 - Editar productos en la carta

En la *Figura 54* se muestra cuando el botón de editar categorías es activado. Esto permite hacer arrastrar y soltar entre las diferentes categorías creadas, pudiendo ordenarlas al gusto del usuario.

Por otro lado, y de igual forma, en la *Figura 55* se muestra cuando el botón de editar productos es activado, permitiendo arrastrar y soltar los diferentes productos.

En ambos casos, tal y como se detalla anteriormente, hay que combinar entre destruir el *dragula* de categorías y productos para poder mover de forma independiente cada componente cuando está activado. Una vez alguna de las opciones es pulsada, por JavaScript

se oculta y se muestra los botones de acción progresivamente para aumentar la satisfacción visual.

En la *Figura 56*, se muestra cómo se realiza el guardado de la carta, teniendo en cuenta que es importante conocer y priorizar el orden que el usuario ha establecido en su catálogo. Para simplificar el proceso, se ha generado un objeto que almacene toda la información necesaria, tanto de categoría como de productos.

```
function guardarCarta() : void {
  let carta : any[] = [];
  let categorias = $('<u>categoria</u>:not(<u>categoria-new</u>');

  categorias.each(function (index) : void {
    console.log($(this));
    let categoriaNombre = $(this).find('<u>categoria-titulo</u>').text();
    let idCategoria = $(this).find('<u>id_categoria</u>').val();
    let categoriaDescripcion = $(this).find('<u>categoria-descripcion</u>').text();
    let productos = $(this).find('<u>producto</u>');
    let cartaProductos : any[] = [];

    productos.each(function (index) : void {
      let arrayProductos : (...) = {
        '<u>titulo</u>': $(this).find('<u>producto-titulo</u>').text(),
        '<u>descripcion</u>': $(this).find('<u>producto-descripcion</u>').text(),
        '<u>precio</u>': $(this).find('<u>producto-precio</u>').val(),
        '<u>imagen</u>': $(this).find('<u>img</u>').attr('<u>src</u>'),
        '<u>id_producto</u>': $(this).find('<u>id_producto</u>').val()
      };
      cartaProductos.push(arrayProductos);
    });

    let categoria : (...) = {
      "id": idCategoria,
      "titulo": categoriaNombre,
      "descripcion": categoriaDescripcion,
      "productos": [
        ... cartaProductos
      ]
    }

    carta[index] = categoria;
  });
}
```

*Figura 56 - Lógica de guardado de carta*

Una vez el objeto se crea, ya se tiene todo lo necesario para enviarlo al *backend* y proceder a guardarlo. Antes de eso, en la *Figura 57*, enviamos el objeto vía *Ajax*.

```
// Se guarda
$('#spinner-loading').fadeIn();

$.ajax({
  url: 'guardarCarta',
  method: 'POST',
  data: {
    carta: JSON.stringify(carta)
  },
}).done(response => {
  standardAjaxResponse('¡Actualizado/a!', response.mensaje);
}).failerrores => {
  customFormAjaxResponse(errores);
}).always(C : void => {
  $('#spinner-loading').fadeOut();
})
activarGuardado( activo: false);
```

Figura 57 - Lógica de envío de la carta

Como se comenta en apartados anteriores, es importante remarcar la comunicación al usuario en todo momento del estado de las acciones, por lo que se retorna tanto un mensaje satisfactorio o de error según el resultado del guardado.

Una vez en el lado del servidor, tal y como se muestra en la *Figura 58*, guardamos primeramente las categorías de la carta, creando o actualizando según corresponda.

```
foreach ($carta as $index => $categoria) {
  if(isset($categoria->id) && $categoria->id != ''){
    $newCategoria = Categoria::query()->find($categoria->id);
    if(!isset($newCategoria)){
      $newCategoria = new Categoria();
    }
  }else{
    $newCategoria = new Categoria();
  }

  $newCategoria->id_tienda = $tiendaId;
  $newCategoria->nombre = $categoria->titulo;
  $newCategoria->descripcion = $categoria->descripcion;
  $newCategoria->orden = $index;
  $newCategoria->save();
}
```

Figura 58 - Guardado de categorías de la carta

Al igual que en la figura anterior, en la *Figura 59* se guarda los productos que el usuario ha creado o editado. Para que este proceso funcione correctamente, es importante conocer cuáles de los productos son nuevos y cuáles de ellos son actualizaciones.

```

foreach ($categoria→productos as $index ⇒ $producto) {

    if(isset($producto→id_producto) && $producto→id_producto ≠ ''){
        $newProducto = Producto::query()→where( column: 'id', $producto→id_producto)→first();
        if(!isset($newProducto)){
            $newProducto = new Producto();
        }
    }else{
        $newProducto = new Producto();
    }

    $newProducto→nombre = $producto→titulo;
    $newProducto→descripcion = $producto→descripcion;
    $newProducto→precio = $producto→precio;
    $newProducto→orden = $index;
    $newProducto→id_categoria = $newCategoria→id;
    $newProducto→id_tienda = $tiendaId;
    $newProducto→save();
}

```

Figura 59 - Guardado de productos de la carta

Al tener guardado tanto las categorías como los productos, se debe almacenar las imágenes. Es importante controlar que el producto se haya creado satisfactoriamente para evitar guardados innecesarios y errores futuros. En la siguiente figura (Figura 60), se muestra el algoritmo realizado para detectar la imagen del producto en la petición, crear una carpeta si es necesario con los permisos (esto evitará errores de lectura y guardado) y convertir la imagen de *base64* a un archivo y almacenarlo en el servidor.

```

$path = public_path('/images/productos/');
if (str_contains($producto→imagen, 'data:image') && $producto→imagen ≠ '' &&
    $producto→imagen ≠ null && $producto→imagen ≠ $newProducto→imagen) {

    if (!(file_exists($path) && is_dir($path))) {
        mkdir($path, permissions: 777, recursive: true);
    }

    $file = Helper::base64ToFile($producto→imagen);

    $filename = $newProducto→id . '-' . Helper::randomString() . '.' . $file['extension'];

    $this→base64_to_image($producto→imagen, output_file: $path.$filename);

    $newProducto→imagen = $filename;
    $newProducto→save();
}

```

Figura 60 - Guardado de imágenes de la carta

Con esto se habría completado el proceso de creación o actualización de la carta para la tienda. El objetivo en todo momento es la facilidad para el usuario de poder gestionar la carta y que esté optimizado para la web.

## 10. Despliegue del aplicativo

Para el lanzamiento de la web, es necesario conocer algunos aspectos necesarios que se comentarán en este apartado.

### 10.1. Compra del dominio

El dominio web se podría definir como un nombre o dirección única que se escribe en el navegador para encontrar e identificar un sitio web.

El primer paso para la publicación de un sitio web en internet y que los clientes puedan acceder, es la compra de un dominio.

En este trabajo, se ha optado por realizar la compra del dominio en la plataforma *IONOS*, ya que por la experiencia que ha tenido el autor del trabajo, tiene una buena asistencia en caso de problemas.



Figura 61 - Búsqueda de dominio en IONOS



Figura 62 - Resultados búsqueda de dominios en IONOS

Aunque lo ideal sería comprar un dominio llamado *Ordery* o relacionado, se ha optado por comprar un dominio llamado “*juanjoseds*” (el nombre del autor del trabajo y sus apellidos abreviados), y crear un subdominio para lanzar la web en producción.

Para hacer este proceso se ha accedido a *IONOS* y se ha buscado el nombre anteriormente comentado.

En los resultados de la búsqueda, aparecerán diferentes posibilidades de dominios como “.org”, “.es”, “.net” entre otros. En este caso, se ha elegido “.com”.

Como podemos ver en la *Figura 62*, los precios pueden variar dependiendo del dominio que se

elija, pero, por regla general, suele tener un precio bastante económico al año.

Este proceso, simplemente termina al momento de realizar el pago con el dominio que se haya elegido. En apartados posteriores se detallará como vincular y configurar este dominio.



## 10.2. Compra del servidor VPS

Otro de los componentes más importantes a la hora de lanzar a producción una web, es tener un servidor capaz de conectarse a internet y que pueda resolver las DNS, con recursos suficientes para gestionar el tráfico de los visitantes y de la propia web.

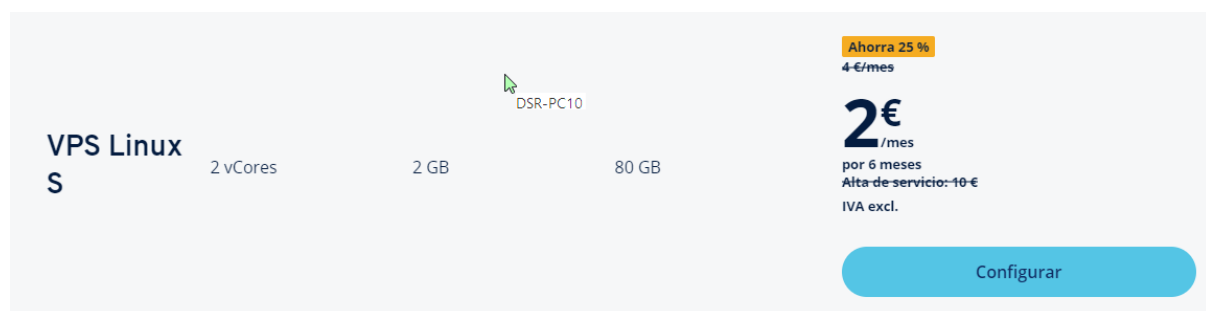
Para cumplir con dichas expectativas, se ha comprado, también en *IONOS*, un servidor privado virtual, también llamado *VPS*. Un servidor *VPS* no es más que una máquina virtual dentro de un servidor físico, donde los recursos, por su tipología, son virtualizados.

Se ha elegido un servidor *VPS* en lugar de un servidor compartido por simples razones de seguridad, rendimiento, estabilidad y escalabilidad. En los servidores de esta tipología los recursos son únicos para cada máquina virtual e independientes de otros sitios webs ajenos que se encuentren en el mismo servidor físico. Esto supone una mejora en seguridad, puesto que la máquina está aislada de otros clientes. También en estabilidad y escalabilidad, ya que se puede monitorizar y llevar un control de los recursos propios de tu servidor, pudiendo ampliarlo en cualquier momento sin tener que hacer migraciones de servidor ni similares.

Por otro lado, y el factor más importante, es que un servidor compartido tiene muchas restricciones que no tiene un servidor *VPS*, como no poder ejecutar comandos, configuraciones, cambios de versiones, etc. Además, una aplicación donde se espera una alta afluencia de personas no es recomendable alojarlas en un servidor compartido por las medidas de seguridad que comentamos anteriormente y porque, si alguna de las aplicaciones que se alojan en el servidor de terceros se encarga de enviar *spam* a través de internet y resulta ser añadido a una *blacklist*, estaría afectando a *Ordery* también.

Para evitar problemas de configuración con el dominio, se optó por comprar el servidor en el mismo hosting donde se adquirió este. Teniendo todo en un mismo lugar, garantizamos que los problemas de configuración y restricciones entre hostings sea mínima.

En cuanto al tamaño se ha elegido un servidor con 2 *vCores*, con 2 GB de RAM y 80 GB de espacio de disco SSD. Como se puede comprobar en la *Figura 63*, se ha optado por un servidor de arquitectura Linux, por su comodidad y versatilidad a la hora de realizar configuraciones y mantenimientos.



The image shows a screenshot of a web hosting configuration page for a VPS Linux S. The configuration details are: 2 vCores, 2 GB RAM, and 80 GB SSD. A price tag indicates a 25% discount, showing 2€/mes for 6 months, with a service fee of 10€ (IVA excl.). A 'Configurar' button is visible at the bottom right.

Figura 63 - Requisitos servidor VPS Linux S

Una vez hemos realizado la compra del servidor, simplemente accedemos al panel de *IONOS* donde nos proporcionará las credenciales de acceso al servidor y todos los detalles importantes relacionados.

## 10.3. Configuraciones del servidor

### 10.3.1. Instalación Plesk

Para llevar un buen control del servidor, tanto en seguridad como en configuraciones, se ha instalado *Plesk*. Es una herramienta que habilita un panel de control para la gestión del servidor enfocado en alojamiento web.

Para su correcta instalación, se ha accedido por *SSH* al servidor y se ha ejecutado los siguientes comandos:

<pre>apt-get install curl</pre>
<pre>sh &lt;(curl https://autoinstall.plesk.com/one-click-installer    wget -O - https://autoinstall.plesk.com/one-click-installer)</pre>

Tabla 4 - Comandos para instalar Plesk

Una vez se haya instalado correctamente, tenemos que entrar a la URL del servidor, añadiendo el puerto “:8443” al final de la ruta, ya que es donde se accede a *Plesk*. El proceso que viene a continuación es bastante rápido, ya que simplemente hay que rellenar los campos del proceso de instalación, preferencias e introducir la clave de activación de *Plesk*. Esta clave la proporciona *IONOS* en el panel, ya que con la compra también se adquiere esta.

Al finalizar el proceso de instalación, accedemos nuevamente a la URL de nuestro servidor con el puerto 8443 y encontramos el *login* del *Plesk*.

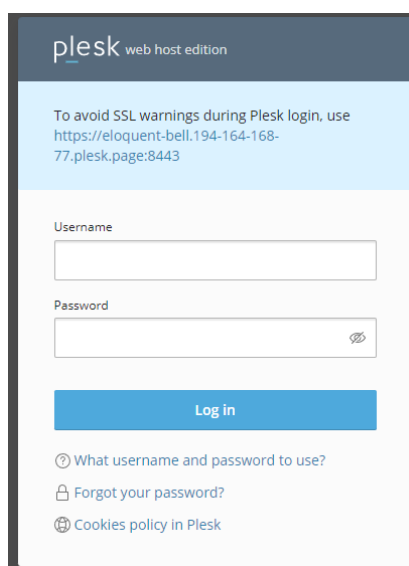


Figura 64 - Login Plesk

Tras iniciar sesión, nos redirigirá a la página principal de *Plesk*, donde podremos encontrar un resumen del servidor, dominios registrados, y un listado de accesos directos de funcionalidades disponibles.

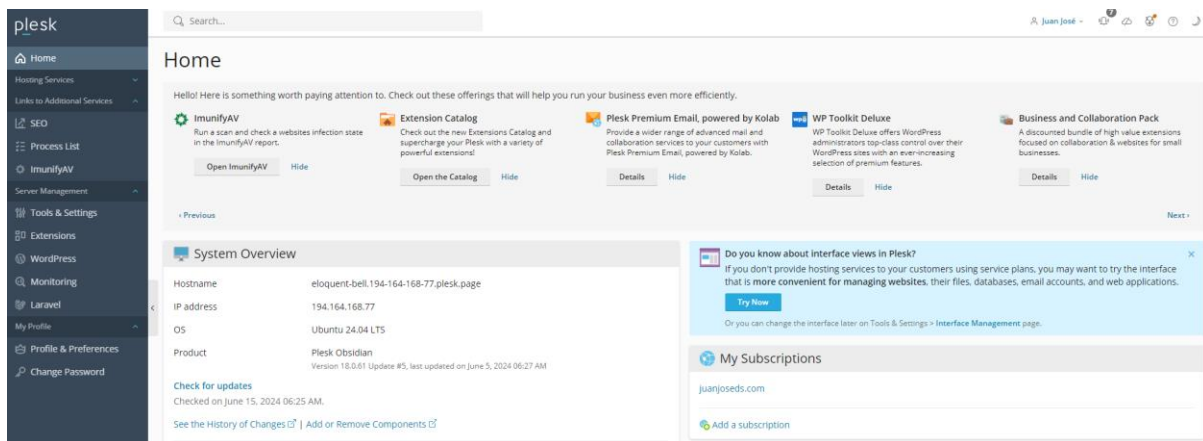


Figura 65 - Entorno de plesk

### 10.3.2. DNS

Según explica Amazon en uno de los apartados de su web: “El DNS, o sistema de nombres de dominio, traduce los nombres de dominios aptos para lectura humana (por ejemplo, *www.amazon.com*) a direcciones IP aptas para lectura por parte de máquinas (por ejemplo, *192.0.2.44*)”.

Es por ello por lo que se tiene que configurar la plantilla de DNS de IONOS, para que apunte al nuevo servidor. Con esto conseguimos que cuando alguien busque el dominio en el navegador, las DNS resuelvan correctamente y pregunte a nuestro servidor VPS sobre el contenido que debe mostrar.

<input type="checkbox"/>	A	@	194.164.168.77	Web Hosting on...	<a href="#">✎</a> <a href="#">🔒</a>
<input type="checkbox"/>	CNAME	www	juanjoseds.com	Web Hosting on...	<a href="#">✎</a> <a href="#">🔒</a>

Figura 66 - Plantilla de DNS de IONOS

En IONOS se ha creado un registro A<sup>6</sup> que apunta al servidor VPS Plesk que se ha adquirido. Con esto conseguimos que cuando alguien escriba el dominio “*juanjoseds.com*” los servidores DNS sepan que tienen que llevarlo a la IP “*194.164.168.77*”.

También se ha creado un registro CNAME con el nombre de dominio canónico, en este caso “*juanjoseds.com*”. Así, cuando se le pregunte al DNS sobre algún subdominio, se sabrá cual es el dominio canónico de este.

Una vez realizado las modificaciones pertinentes, se ha comprobado que se propagaban correctamente haciendo uso de una herramienta online llamada “DNSChecker”. En la Figura 67 se muestra un análisis de las respuestas que ha dado los principales servidores DNS del mundo cuando se le ha preguntado por el nombre del dominio. Esta herramienta es interesante para saber si hemos configurado correctamente la plantilla de DNS y comprobar que se ha redirigido satisfactoriamente.

<sup>6</sup> Un registro A tiene la capacidad de enlazar una IP a un dominio.

### DNS CHECK

juanjoseds.com A  Search

Refresh: 20 sec

San Francisco CA, United States	194.164.168.77	✓
Mountain View CA, United States	194.164.168.77	✓
Berkeley, US	194.164.168.77	✓
San Francisco, US	194.164.168.77	✓
Fort Dodge, United States	194.164.168.77	✓
San Jose, United States	194.164.168.77	✓
United States	194.164.168.77	✓
Burnaby, Canada	194.164.168.77	✓
Yekaterinburg, Russian Federation	194.164.168.77	✓
Cullinan, South Africa	194.164.168.77	✓
Diemen, Netherlands	194.164.168.77	✓
Paris, France	194.164.168.77	✓
Madrid, Spain	194.164.168.77	✓

### CHECK DNS PROPAGATION

Whether you have recently changed your DNS records, switched web host, or started a new website - checking wh records are propagated globally is essential. DNS Checker provides a free DNS propagation check service to check System records against a selected list of DNS servers in multiple regions worldwide. Perform a quick DNS propagat any hostname or domain, and check DNS data collected from all available DNS Servers to confirm that the DNS re propagated.

#### DNS Propagation Map by DNSChecker.org

Server Location ✓ Resolved ✗ Not Resolved

#### DNS Lists

IPs

Public IPv4 Public IPv6

Continents

Figura 67 - Herramienta DNSChecker revisando el dominio

Para el subdominio *ordery.juanjoseds.com*, donde se lanzará el proyecto, seguimos el mismo proceso, ya que había que apuntar desde IONOS al servidor y que reconociera dicho subdominio.

### 10.3.3. SSL

Home > Domains > juanjosed.com

Dashboard Hosting & DNS Mail Get started

Files & Databases

- Connection Info for FTP, Database
- Files
- Databases
- FTP
- Backup & Restore
- Website Copying

Dev Tools

- PHP Version 8.3.8
- Logs
- SSH Terminal
- Scheduled Tasks
- Monitoring (Not connected)
- Performance Booster (Speed boost available)
- PHP Composer
- git
- SEO
- Website Importing

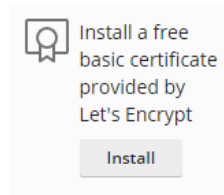
Security

- SSL/TLS Certificates (Security can be improved)
- Password-Protected Directories
- ImunifyAV (Website is clean)
- Web Application Firewall
- Advisor

Figura 68 - Vista de dominio en Plesk

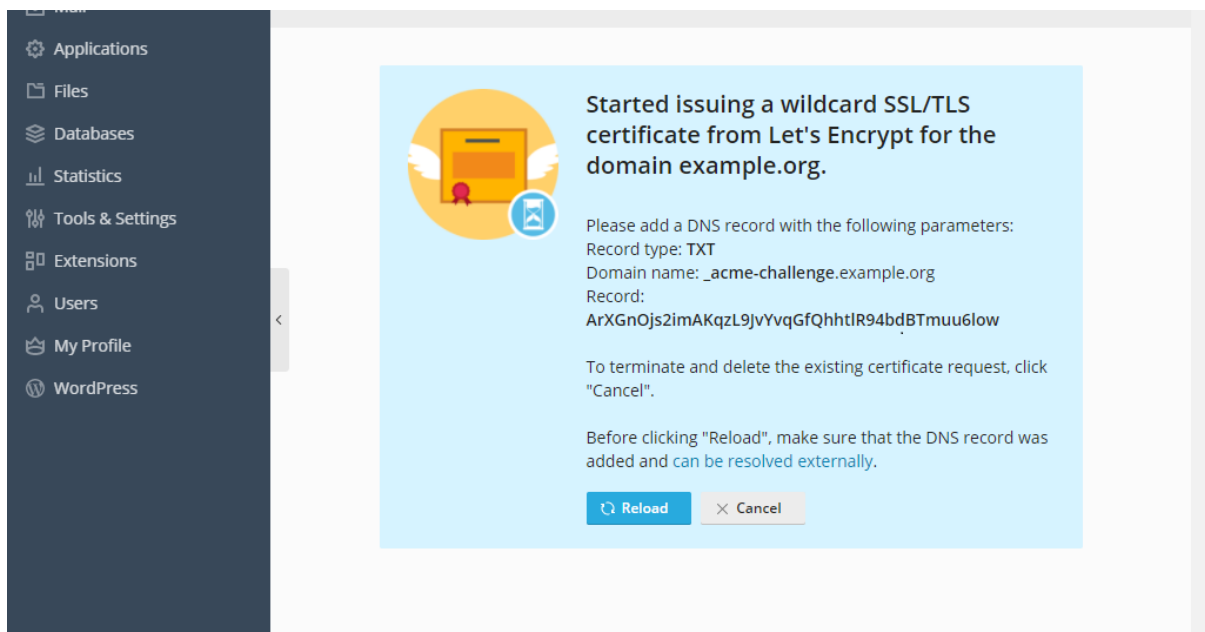
El SSL (*Secure Sockets Layer*), es un protocolo que debemos tener en cuenta a la hora de tener nuestro dominio completamente funcional. Este protocolo de seguridad centrado en el cifrado es esencial para que las personas puedan acceder y confiar en el sitio web. Hoy en día, los principales navegadores de internet prohíben el acceso a las webs que no tienen certificado SSL.

Para configurar esto en *Plesk*, es muy simple. Primero debemos ir al dominio que deseamos generar o renovar el certificado SSL (*Figura 69*) y acceder a la opción “SSL/TLS Certificates”. Si no existe ningún certificado instalado, se puede generar uno gratuito mediante la entidad certificadora *Let's Encrypt*.



*Figura 69 - Botón de instalar certificado Lets Encrypt*

Tras darle al botón, se mostrará la siguiente pantalla, donde se mostrará un registro TXT que deberemos añadir a nuestra plantilla DNS de *IONOS* para verificar el dominio.



*Figura 70 - Registro TXT para validar el certificado*

Una vez añadido el registro TXT a la plantilla DNS, pulsamos sobre el botón “*Reload*”. Esto permitirá que *Plesk* compruebe que todo esté correcto. Acto seguido, si esto fuera así, mostrará un aviso de que nuestro dominio se encuentra protegido por el certificado.

### 10.3.4. Puertos

Para poder trabajar con normalidad en *Plesk*, es necesario abrir algunos puertos en el servidor. Nos dirigimos a *IONOS* al apartado de Red y Políticas de Firewall y editamos la plantilla.

cloudpanel.ionos.es/panel/app/corevps?ionospanelid=ionos9d13e84e-22ca-41a8-8c46-388e71956d0c1839#/firewalls/be0e9c51-5035-4d8f-824e-d5efe1160894

IONOS MENÚ

Buscar funciones, dominios y artículos de ayuda

Último acceso: 20/05/2024 15:00:35 desde 62.117.155.26 (Spain)

Asegúrese de que todos sus dominios estén protegidos con SSL

- ✓ Cifrado de extremo a extremo hacia y desde su sitio web
- ✓ En este momento, dispone de 1 dominio(s) sin certificado SSL.

Gestionar mis certificados SSL

### Políticas de firewall

Nombre	Estado	Puerto
My firewall policy	●	TCP: 22, 80, 443, 8443, 8447, 21, 49152-65535, 53

My firewall policy Disponible

Introduzca una descripción.

#### Configuración

##### Entrada

Acción	IP permitida	Protocolo	Puerto(s)	Descripción
Permitir	Todas	TCP	22	
Permitir	Todas	TCP	80	

Figura 71 - Firewall de IONOS

Para poder trabajar sin ningún problema, tanto por SSH como por FTP y otros protocolos, es necesario abrir el rango de puertos comprendidos entre 49152 y 65535.

## 10.4 Configuración con Git

Plesk incorpora la herramienta Git para poder desplegar el proyecto de forma rápida y sencilla. Antes de usar esta herramienta deberemos vincular el repositorio donde se encuentra alojado el proyecto.



En el dominio, hacemos clic en la herramienta Git, donde se nos abrirá una ventana con los repositorios vinculados al dominio actualmente.

Figura 72 - Botón de git en Plesk

En cualquier caso, para vincular uno nuevo, debemos hacer clic en “Add Repository”, donde acto seguido se nos abrirá una ventana emergente solicitando la URL del repositorio, el nombre del repositorio (cargará automáticamente tras introducir la URL de repositorio), el modo de despliegue y la ubicación en el servidor.

## Create repository >

### Code location

**Remote repository**

Your code is hosted online (a cloud service like GitHub, GitLab, or Bitbucket, or your own server). Plesk will pull code from there.

**Local repository**

Your code is on the computer to which Plesk wouldn't be able to connect. For example it's your laptop or some server unavailable outside of your LAN. You will push code to the server with Plesk yourself.

Repository URL \*

Both HTTP(S) and SSH protocols are supported

SSH public key creation:

Default [Add new deploy key](#)

SSH public key content:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQAC1EihsX9hwqMz/Rj3f3l8aNfr
Eh2W1yeVu33rOKFMK+I+/s6GBJOEYO7PQ56T0Lj1j2Q5G+ZGlu/uyuGoE
t+ieKouG6NuyoEOY20BnDz3rNFYH28FkjsW7R+ZJA+TLFukkiREtezwO/oi
```

This is the public part of the SSH key used for authorization in the remote repository. It must be added to the remote service.

Repository name \*

Specify a name that is unique within a domain.

### Deployment settings

Deployment mode \*

Automatic Manual Disabled

Files will be deployed to the production site as soon as they are available in the Plesk repository.

[Create](#) [Cancel](#)

Figura 73 - Ventana de creación de un nuevo repositorio en Plesk

Al introducir la URL del repositorio, el servidor generará una clave *SHA-256* que deberemos introducir en este en Github. Esta clave debe guardarse como una “*Deploy Key*” para que pueda obtener los permisos necesarios para obtener y desplegar el código.

The screenshot shows the GitHub 'Deploy keys' page. On the left is a sidebar with navigation options: General, Access, Collaborators, Code and automation (Branches, Tags, Rules, Actions, Webhooks, Codespaces, Pages), Security (Code security and analysis, Deploy keys, Secrets and variables), and Integrations (GitHub Apps, Email notifications). The main content area is titled 'Deploy keys' and shows a single key named 'Servidor 77'. The key's details include a key icon, the label 'SSH', the SHA256 fingerprint 'SHA256:BXWntUxwftNcJkjiHzXynpqJzumZekI3Ch5xCIT7rK0', the creator '@Juanjoseds', the date 'Added on May 22, 2024', and the permissions 'Last used within the last 4 weeks — Read-only'. A 'Delete' button is visible next to the key details. An 'Add deploy key' button is in the top right corner.

Figura 74 - Creación de una deploy Key en github

En cuanto al método de despliegue, se ha dejado la opción automática que viene por defecto, para que, cuando se haga un “*pull request*” de los cambios en el servidor y actualizar la web a la última versión, se sustituya de forma automática los archivos con cambios.

## 10.5 Despliegue del proyecto

Una vez ya tenemos los archivos del proyecto en el servidor, solo falta ejecutar los comandos necesarios para instalar las librerías y dependencias del proyecto.

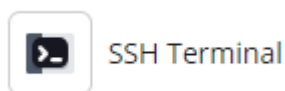


Figura 75 - Botón de SSH Terminal del dominio en Plesk

En el dominio, hacemos clic en la herramienta “SSH Terminal”, que nos habilita una terminal directa en el servidor. También se puede usar como alternativas programas de terceros y realizar una conexión desde el exterior, pero en este caso se prefiere usar esta terminal puesto que no se requiere mucho trabajo y, aunque es algo limitada, permite realizar los comandos que deseamos

ejecutar.

Los comandos más importantes para tener en cuenta son los siguientes.

Comando	Descripción
<code>npm i</code>	Instala las librerías y dependencias de NodeJs
<code>composer i</code>	Instala las librerías y dependencias de Composer
<code>npm run prod</code>	Ejecuta la compilación y minificación de los archivos
<code>php artisan migrate</code>	Ejecuta las migraciones de la base de datos para crear las tablas y columnas que se utilizan en el proyecto
<code>php artisan key:generate</code>	Genera una <i>key</i> que se guarda en el archivo “.env” que es único y necesario para el funcionamiento de Laravel

Tabla 5 - Comandos de despliegue de un proyecto Laravel

Con esto, podemos concluir el lanzamiento a producción de nuestra aplicación, haciendo que sea completamente funcional y accesible desde cualquier punto del mundo.

## 11. Campaña de lanzamiento

Para dar a conocer *Orderly* en el mercado actual y fomentar su uso, es necesario llevar a cabo una estrategia de lanzamiento.

En este apartado, se definirá a grandes rasgos la campaña de lanzamiento, en la cual habría que profundizar mucho más para ejecutarla.



La tendencia de la compra de productos online creció de manera exponencial debido a las circunstancias vividas durante la época del Covid-19, la cual los usuarios aceptaron e instauraron como una herramienta más para agilizar procesos de su día a día. Por lo tanto, los esfuerzos de comunicación deberán ir dirigidos a visibilizar los beneficios de la web para poder posicionarnos en la mente de los consumidores frente a posibles competencias.

Conociendo la situación actual, podemos establecer que el objetivo principal de la campaña de lanzamiento será dar a conocer la web en el mercado, siendo un producto nuevo por su tipología.

El público objetivo al que irá dirigido la campaña serán, hombres y mujeres de entre 50 – 65 años que buscan optimizar el proceso de compra cotidiano.

*Ordery* se diferencia de la competencia por la versatilidad a la hora de generar pedidos en la web. Los clientes pueden realizar compras en una tienda mediante el catálogo de productos, subiendo una imagen de sus notas de compra o, simplemente, escribiendo texto. Además, los empleados de las tiendas encargados de gestionar dichos pedidos, pueden fácilmente revisarlos y editar el precio de cada producto solicitado en caso de que fuera necesario, siendo de necesaria utilidad para poder aceptar cualquier tipo de petición del cliente.

### **11.1. Prelanzamiento**

El prelanzamiento se comenzará 2 meses antes del lanzamiento de la plataforma a producción.

En esta fase de la estrategia se creará y se publicará el contenido en redes sociales creando expectativa a los usuarios. La idea de esta fase es conectar con el público potencial, intentando abarcar toda la audiencia posible.

El contenido que se publicará en las redes sociales se enfocará especialmente en destacar los beneficios del modelo de negocio, las facilidades para el día a día que puede suponer su uso, ventajas tanto para los usuarios como para las tiendas que se registren, entre otros.

Es importante lanzar mensajes directos a la audiencia que hagan empezar a crear comunidad en las plataformas sociales, algo crucial para posicionar la web en la mente de los consumidores.

En esta fase será interesante implementar una página simple solicitando el correo electrónico a aquellas personas interesadas, con la finalidad de suscribirse a una “*newsletter*” (en español, boletín informativo). Esta técnica llamada *Email marketing* nos ayudará a llegar a más público objetivo y avisar de una forma más directa de los tiempos de lanzamiento, aumentando la base de datos de comunicaciones para el futuro.

### **11.2. Lanzamiento**

El lanzamiento se refiere a la semana 0, donde se hace público que la web está en producción y lista para el uso.

En esta fase se publicará un video demostrativo y se compartirá en todas las plataformas donde se tenga presencia. La intención es publicar oficialmente el lanzamiento de la plataforma para que llegue al máximo público posible. Además, se acompañará con valoraciones, contenido de *influencers*<sup>7</sup>, temas recurrentes de beneficios y ventajas del uso.

Para incrementar la captación y rápido crecimiento inicial, se promocionará el lanzamiento de la web con cupones de descuento. Por ejemplo, un descuento de 10€ a los primeros 100 en realizar un pedido, manteniendo en el tiempo un descuento de 20% en tu primer pedido.

### **11.3. Post-lanzamiento**

El post-lanzamiento comenzará durante el primer mes hasta el tercer mes aproximadamente.

Durante esta fase, será interesante recopilar el *feedback* (en español, retroalimentación) de los usuarios que han usado la aplicación, generando una base de datos de incidencias y mejoras que se implementarán según el grado de importancia.

En las redes sociales, se publicará de forma continuada las actualizaciones y nuevas funcionalidades que se implementen, captando así al público más indeciso, ya que hacer ver a la audiencia que la plataforma está en continuo desarrollo, anima a las personas a probarlo.

En adición a lo anterior, se seguirán ofreciendo descuentos y otras acciones como sorteos y colaboraciones con *influencers*, que nos ayuden a incrementar el número de seguidores.

Para finalizar, durante las tres fases se irán analizando y monitorizando los resultados de cada acción por si fuera necesario modificar o invertir más esfuerzos en alguna de ellas para conseguir el objetivo planteado.

## **12. Resultados, conclusiones y trabajos futuros**

### **12.1. Resultados**

Una vez realizado el desarrollo de la plataforma, podemos decir que este trabajo ha cumplido y con creces todas las expectativas que pusimos al inicio de la definición. Aunque puede resultar una idea común, tiene factores diferenciales muy interesantes que combinados con un buen diseño ha hecho que sea único.

Sin lugar a duda se han cumplido todos los objetivos planteados al inicio del proyecto, donde incluso se han ampliado en diferentes aspectos que se han contemplado en este trabajo, como el despliegue en un servidor y la configuración de este.

Como resultados de este trabajo, los usuarios pueden:

- a. Hacer una compra a una tienda de interés tanto con productos disponibles en su catálogo, como enviando una foto de unas notas de compra o con texto plano.
- b. Poder buscar y ver productos y tiendas de interés para el usuario.
- c. Permite a los empleados de las tiendas gestionar tanto su tienda de la web como los pedidos que se generan en ella.
- d. Permite a los administradores llevar a cabo un control minucioso de la plataforma.

---

<sup>7</sup> Los *influencers* son perfiles con alta credibilidad que siguen muchas personas en las redes sociales.

En líneas generales, con este desarrollo creemos que podemos ofrecer la solución a los pequeños comercios para que puedan ganar visibilidad y ventas de forma gratuita, pudiendo gestionar los pedidos directamente en la misma plataforma. Donde, además, hacemos énfasis a las circunstancias vividas con el covid-19, que fue cuando surgió la idea de este proyecto, y por lo tanto, se permite a los usuarios preocupados por su salud recoger rápidamente sus pedidos. Así que podemos afirmar, que se cumple el objetivo inicial.

Este proyecto resulta ser muy ambicioso, y a nivel personal, continuaré trabajando en detalles del aplicativo para mejorarlo cada vez más y poder acabar haciendo realidad el lanzar el proyecto al mundo.

## 12.2. Conclusiones

A modo de conclusión, este proyecto ha sido bastante enriquecedor en todos los sentidos. El uso de tecnologías como Laravel, el uso de *JavaScript* e incluso profundizar en métodos que simplifican el trabajo de PHP, hacen que la experiencia en este lenguaje crezca de forma incuestionable. Cuando te enfrentas a un proyecto de esta envergadura siendo uno mismo el responsable, surgen preguntas importantes que hay que tener en cuenta como la seguridad, el rendimiento o la estabilidad de la plataforma, lo que hace que profundices mucho más en el lenguaje para intentar aprovechar al máximo los recursos disponibles.

Por otro lado, el estudio y profundizar en las necesidades que pueda tener la sociedad no ha sido tarea fácil, puesto que crear una herramienta innovadora cada día es más difícil, pero con dedicación y uniendo las piezas necesarias hemos conseguido crear una plataforma capaz de hacer frente a muchas otras y ayudar en la idea principal del proyecto.

En definitiva, *Ordery* es un proyecto con un potencial increíble que aparte de enseñar a desenvolverte en el mundo de la programación, también te hace comprender la sociedad y cómo funciona. Este proyecto me ha enseñado una gran cantidad de cosas que, sin duda, aplicaré en el futuro.

## 12.3. Futuras mejoras

La magnitud de este proyecto te permite pensar en muchas funcionalidades que se pueden implementar. Algunas de ellas se han pensado para ejecutarlas durante este trabajo, pero por falta de tiempo no se han podido llevar a cabo.

Algunas de las futuras mejoras que sin duda serían importantes llevarlas a cabo son:

- a. Validación de imágenes mediante IA que compruebe que el contenido subido por los usuarios cumpla con las normas de la plataforma.
- b. Motor que analice el texto introducido por los usuarios para censurar cualquier palabra ofensiva.
- c. Preparar la plataforma para que las tiendas se muestren ordenadas por proximidad al usuario que busca.
- d. Implementar un chat para que puedan comunicarse directamente con la tienda.
- e. Implementar un chat para que puedan comunicarse directamente con la asistencia de la plataforma.
- f. Adaptar las vistas de toda la plataforma para dispositivos móviles
- g. Crear una aplicación móvil para que la plataforma tenga más alcance aún.

## 13. Bibliografía

- Amazon. (2024). *Amazon*. Obtenido de <https://aws.amazon.com/es/route53/what-is-dns/>
- Apache. (s.f.). *Apache License*. Obtenido de <https://apache.org/licenses/LICENSE-2.0>
- Commons, C. (2019). *Creative Commons*. Obtenido de <https://creativecommons.org/share-your-work/ccllicenses/>
- datos, A. e. (Mayo de 2024). *Guía sobre el uso de las cookies*. Obtenido de <https://www.aepd.es/guias/guia-cookies.pdf>
- España, G. d. (12 de Julio de 2002). *Ley 34/2002, de 11 de julio, de servicios de la sociedad de la información y de comercio electrónico*. Obtenido de <https://www.boe.es/buscar/pdf/2002/BOE-A-2002-13758-consolidado.pdf>
- España, G. d. (06 de Diciembre de 2018). *Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales*. Obtenido de <https://www.boe.es/buscar/act.php?id=BOE-A-2018-16673>
- Kantar. (2022). *Kantar*. Obtenido de <https://kantar.turtl.co/story/foodservice-2022-p/page/5/1>
- ULPGC. (29 de Febrero de 2024). *ULPGC / Escuela de Ingeniería Informática*. Obtenido de <https://www.eii.ulpgc.es/sites/default/files/2024-05/20240229%20Grado%20en%20Ingenier%C3%ADa%20Inform%C3%A1tica%20%20rev03.pdf>

## 14. Glosario

### A

#### *addons*

En español, extensión, permite añadir características extra a una herramienta o utilidad... 12

### B

#### *backend*

El backend se refiere al lado del servidor de un sitio, o dicho de otra forma, la parte lógica que se comunica con la base de datos. .... 66

### C

#### *click & collect*

Es un modelo de negocio que permite al cliente hacer una compra por internet y recogerlo en una tienda física..... 12

coronavirus

El coronavirus, o también llamado COVID-19, fue una pandemia originada en 2019, donde las personas que contraían el virus sufrían una enfermedad respiratoria, en muchos casos, grave. .... 12

## **D**

### *datatable*

Librería de Javascript que permite potenciar las capacidades y la optimización de una tabla, proporcionando herramientas útiles para su configuración y presentación. ....60

### *DOM*

En español, modelo de objeto de documento, es una interfaz para documentos HTML y XML.....66

## **F**

### *framework*

Un framework es un marco de trabajo que aplica una metodología de trabajo estricto, donde combinado con un conjunto de reglas, hace que el desarrollo sea más efectivo ya que se ahorra tiempo. ....23

### *FTP*

En español, Protocolo de transferencia de ficheros, es empleado comunmente en el puerto 21 y permite el intercambio de archivos entre equipos.....78

## **M**

### *minificar*

Es una técnica de compresión que elimina los espacios, comentarios, saltos de línea y tabulaciones haciendo que el fichero pese menos.....22

## **P**

### *Prestashop*

Prestashop es una plataforma de código abierto que permite crear una tienda online con tan solo unos pocos clics, gestionar pedidos, clientes y un largo etcétera de funcionalidades. ....14

## **S**

### *seeders*

La palabra anglosajón “Seeder” se traduce textualmente como “sembradores”. Es una funcionalidad que dispone los lenguajes de programación más modernos para poblar una base de datos. Son capaces de llenar muchas filas de la base de datos con información aleatorios. ....16

## **Anexo I. Manual de usuario**

En este anexo, se detallará el manual de usuario de *Ordery*, donde se profundizará en cada una de las funciones implementadas en la plataforma con una breve explicación de uso para el usuario final.

### **AI.1. Login**

El *login*, como bien se comentó en apartados anteriores, permite el acceso tanto de clientes como de empleados de las tiendas. Este formulario simplificado permite comodidad y uniformidad en toda la plataforma.

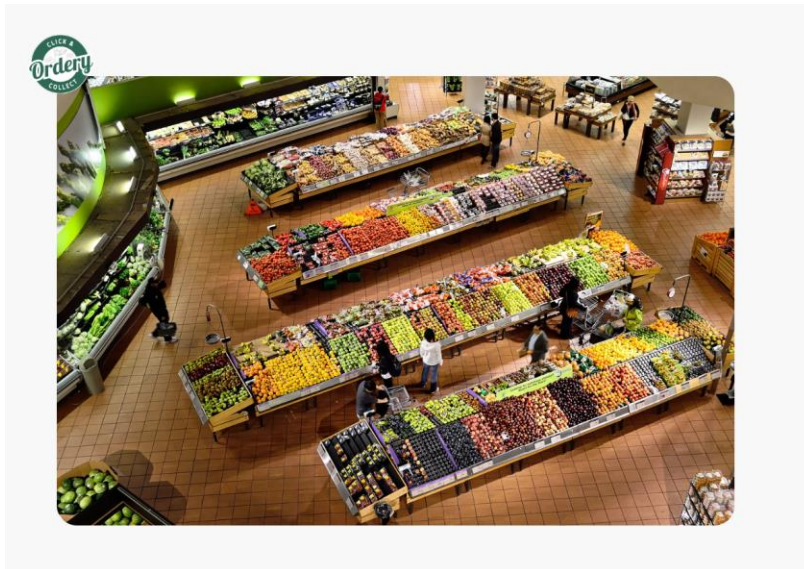


Figura 76 - Login

A diferencia de lo ideado en la maqueta, el *login* presenta cambios tanto en la imagen de la izquierda, representando algo mucho más realista y enfocado al mercado de destino, y los botones de las redes sociales, que se han adaptado a la actualidad.

También permite a los usuarios restablecer su contraseña o incluso redirigirle rápidamente al registro para comenzar con su inicio en *Orderly*.

## AI.2. Registro de usuarios

En la vista de registro de usuarios, se ha respetado el estilo maquetado anteriormente. La intención es solicitar únicamente los datos necesarios obligatorios para el registro, evitando campos innecesarios que hacen que la mayoría de los usuarios desconfíen y rechacen el uso de la aplicación.

Es hora de llenar tu despensa 🍷  
Haz tu pedido y recógelo sin esperas

Nombre: Tu nombre  
Apellidos: Tu apellido

DNI: Introduce tu DNI  
Teléfono: +34 699123111

Email: tienda@gmail.com

Contraseña: \*\*\*\*\*

Acepto los términos y condiciones

[Regístrame](#)

¿Eres una tienda?  
[¡Regístrate y empieza a vender!](#)

¿Ya tienes cuenta? [Acceder](#)

Figura 77 - Registro de usuarios

En adición a lo anterior, se ha intentado poner *placeholders* (texto dentro del campo a modo informativo) para que cualquier persona entienda que dato deben introducir.

Si un responsable de una tienda desea registrarse como vendedora, puede hacerlo pulsando el botón de la parte inferior, que le redirigirá a otro formulario.


Una vez el usuario se registre, se le redirigirá a la página principal.

### AI.3. Registro de tiendas

El registro de tiendas es un proceso más largo y extendido que un usuario, puesto que es necesario que introduzca información legal sobre la tienda, imágenes y datos de contacto.

Siguiendo el criterio del anterior formulario, se ha empleado el uso de *placeholders*, fomentando la legibilidad y sencillez de este.



Una vez completado el formulario de la *Figura 78*, se redirigirá al usuario a su panel de administración.

Incrementa tus ventas con Orderly 

Tus productos estarán visibles para todo el mundo. ¡Solo ventajas!

Nombre <input type="text" value="Tu nombre"/>	Apellidos <input type="text" value="Tu apellido"/>	DNI <input type="text" value="Introduce tu DNI"/>
Teléfono <input type="text" value="+34 699123111"/>	Email <input type="text" value="tienda@gmail.com"/>	Contraseña <input type="password" value="*****"/>

**Información sobre tu tienda**

Imagen de la tienda  <small>Recomendado: 1920 x 350 px</small>	Logo de la tienda  <small>Recomendado: 115 x 115 px</small>	Nombre de la tienda <input type="text" value="Mi tiendita"/>
Nombre Legal <input type="text" value="Mi tiendita S.L."/>	CIF <input type="text" value="12345678X"/>	Descripción breve <input type="text" value="¡Vendemos los mejores panes!"/>
Dirección <input type="text" value="C/ Tomás Miller, 115"/>	Ciudad <input type="text" value="Las Palmas de Gran Canaria"/>	Provincia <input type="text" value="Las Palmas"/>
Código postal <input type="text" value="35002"/>	Nombre de persona de contacto <input type="text" value="RECEPCION"/>	URL que los clientes visitarán <input type="text" value="mi-tienda"/>

Acepto los términos y condiciones

[Regístrame](#)

[¿Ya tienes cuenta? Acceder](#)

Figura 78 - Registro de tiendas

## AI.4. Página principal

En la página principal se optó por añadir pequeños cambios y mejoras visuales para hacerlo más atractivo visualmente. Aunque hay ligeras variaciones como en el primer bloque donde aparece el nombre de la marca, se ha seguido en todo momento el diseño original.

En la vista podemos ver un listado de establecimientos donde destaca el número de compras realizadas en el establecimiento y su valoración promedio. Aunque estas funciones aún no están implementadas en su totalidad, son datos atractivos para los clientes.

Cada tienda que se muestra debe contener información de contacto para que los usuarios puedan contactar con el comercio en caso de tener algún problema con el proceso. Además, el autor del trabajo de fin de grado tenía la creencia de que el simple hecho de que se muestre un contacto fiable de la tienda haría que aumente la confianza de cualquier comprador, aunque no haya tenido ventas aún.

En la página principal y en cualquier otra, tendrá el usuario a su disposición un buscador en la parte superior para buscar los artículos que necesite o tiendas que conozca.



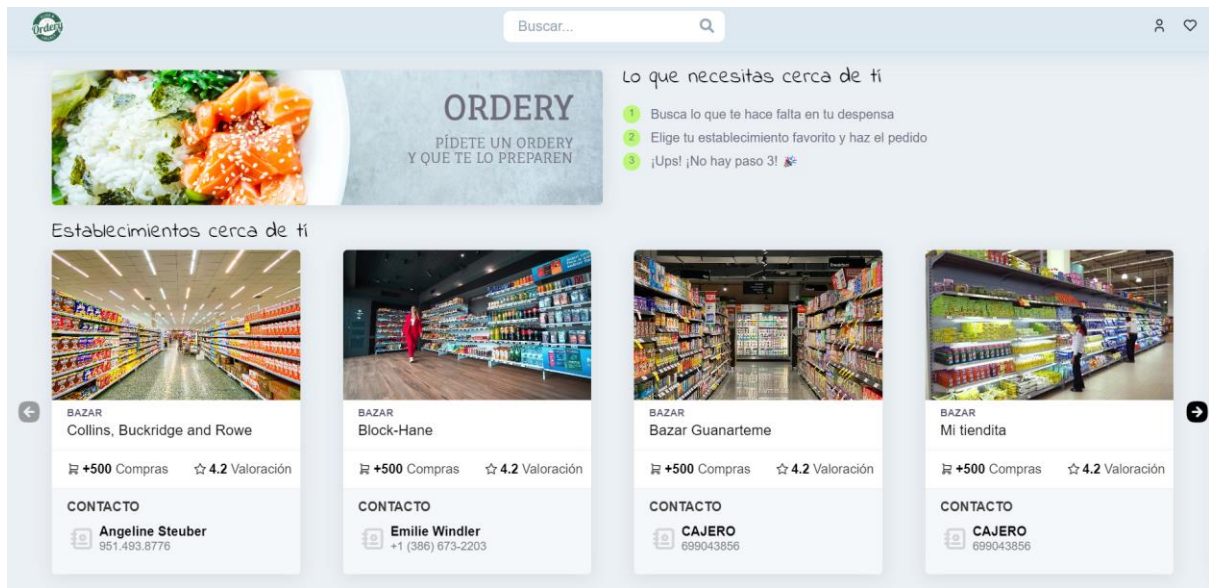


Figura 79 - Página principal



Figura 80 - Página principal, parte inferior

En la *Figura 80*, se muestra la sección de “Quizás necesitas” que muestra un listado sencillo de productos cargados aleatoriamente de la base de datos.

Por último, el *footer* de la web, que permite accesos directos a los textos legales como Condiciones de uso, política de privacidad y política de cookies.

## AI.5. Búsqueda de tiendas o productos

El usuario puede buscar en cualquier momento tiendas que le guste o productos que les sea de interés para comprar. Esta búsqueda la puede hacer en el menú superior, simplemente escribiendo cualquier texto.

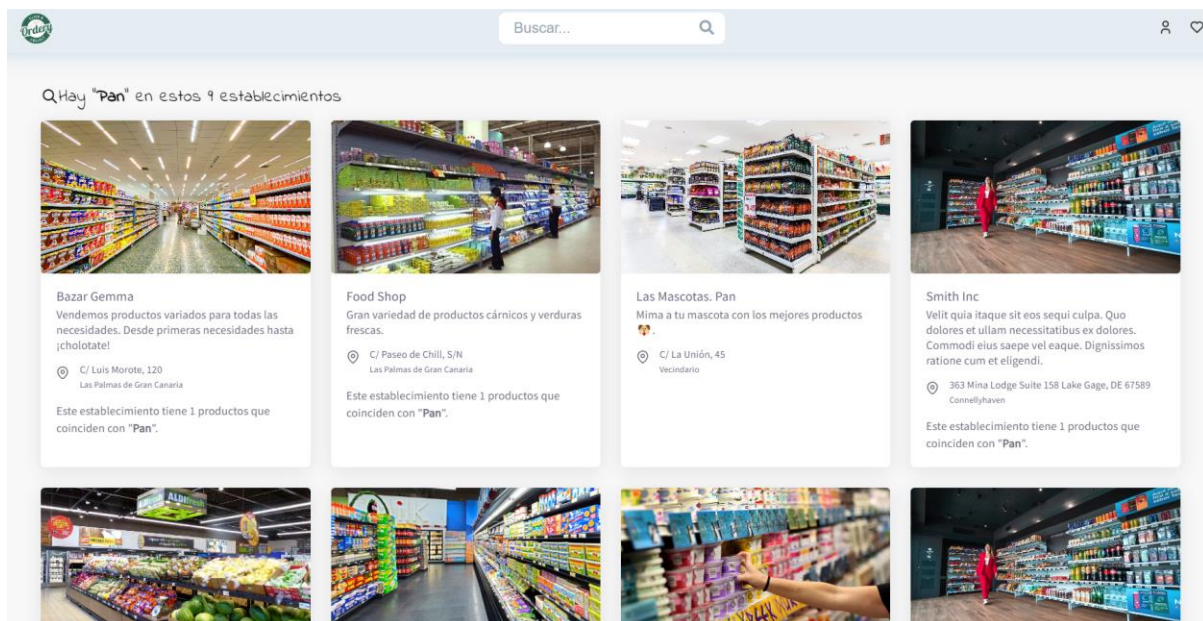


Figura 81 - Listado de tiendas tras una búsqueda

En el listado de tiendas que se muestra en la *Figura 81*, cada uno de los resultados cuentan con el nombre de la tienda, descripción propia de este y la dirección. Al realizar una búsqueda, si el texto que escribe el usuario coincide con un producto de la tienda, se mostrará con el siguiente mensaje de ejemplo: “Este establecimiento tiene 1 productos que coinciden con Pan”. Con esto conseguimos que la búsqueda sea realmente útil, ya que se estará filtrando tanto por tienda como por productos de estas.

Esta sección está cuidadosamente programada, con la finalidad de que sea completamente eficiente y los tiempos de carga sean mínimos. Se realizó una única consulta, donde incluso se paginan para que en la primera página no carguen todos los resultados al mismo tiempo.

En el siguiente apartado, se detallará cuando un usuario hace clic en uno de los resultados.

## AI.6. Detalles de una tienda

Cuando alguno de los resultados es de interés para el usuario, este puede acceder a la tienda para ver su información y los productos que tienen en el catálogo.

Esta plataforma tiene una particularidad, y es que no es necesario que una tienda tenga productos para poder realizar pedidos, por lo que un usuario igualmente puede tramitar uno escribiendo texto plano manualmente o enviando una foto de las notas de compra que tenga.

Si la tienda tiene productos en promoción, se verán reflejados en una lista para hacerlos destacar sobre el resto, fomentando la interacción y la retención de los potenciales clientes.

Una versatilidad y un factor diferenciador de *Ordery* es que un usuario puede combinar una foto de sus notas de compra, productos e incluso texto plano al mismo tiempo. Esto permite mucha mayor personalización en los pedidos y comodidad para los usuarios.

En la *Figura 82*, se muestra el resultado final del detalle de una tienda.

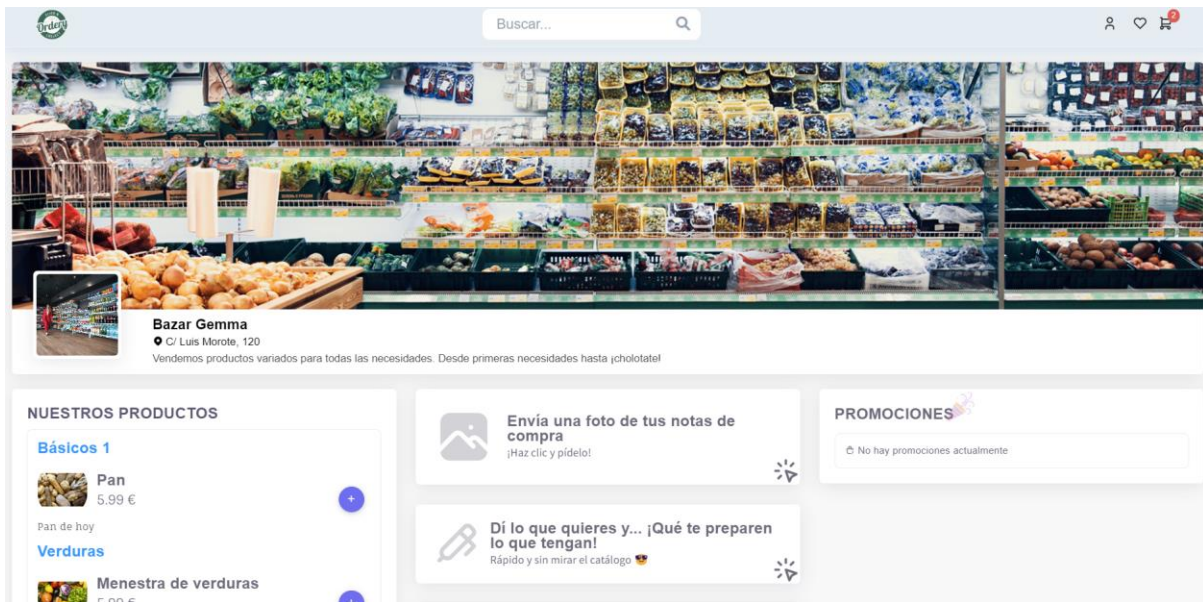


Figura 82 - Detalles de una tienda

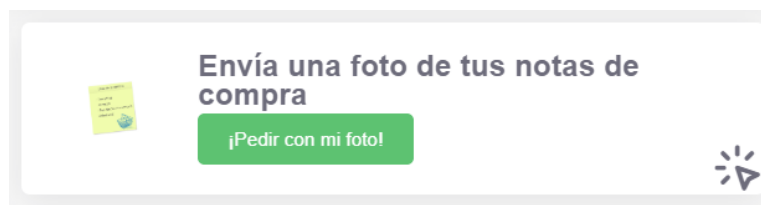


Figura 83 - Detalles de la tienda con imagen de nota de compra

En la anterior *Figura 83*, vemos cómo ha cambiado el campo de enviar una foto de tus notas, tras que el usuario seleccione una foto de su galería. En este punto, simplemente tendría que hacer *clic* en el botón “¡Pedir con mi foto!” y se añadirá correctamente al carrito.

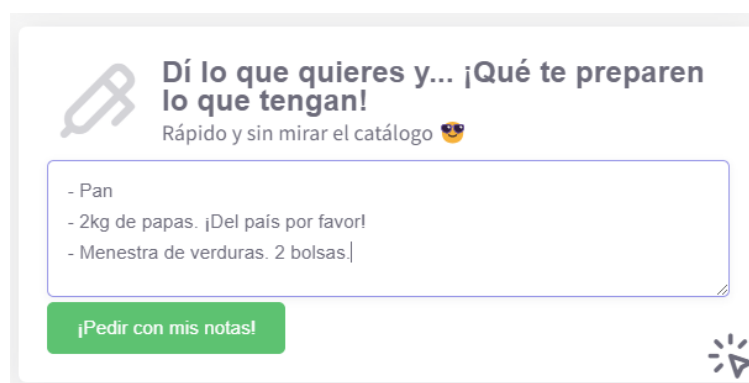


Figura 84 - Detalles de la tienda con nota de compra

En la *Figura 84*, cuando un usuario selecciona escribir texto manual, aparece un *textarea* que le habilita para detallar todo lo que desee. Al igual que el proceso anterior, al hacer *clic* en el botón “¡Pedir con mis notas!” se añadirá al carrito, como se muestra en la *Figura 85*.



Figura 85 - Carrito de compra

Como podemos observar, al añadir los productos, aparecen en el carrito de compra satisfactoriamente, con nomenclaturas claras para que el usuario entienda todo el proceso sin problemas.

Ahora, simplemente, el usuario tendría que hacer *clic* en el botón “¡Realizar el pedido!” y se le redirigirá a la vista de detalles del pedido para que pueda realizar el pedido.

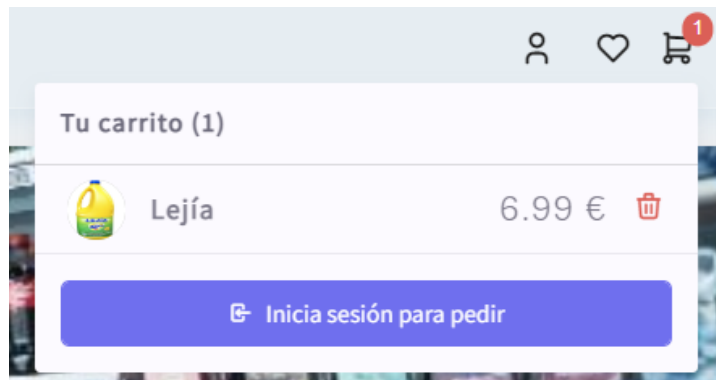


Figura 86 - Carrito de compra sin hacer login

Si un usuario no ha iniciado sesión en la plataforma, no podrá realizar el pedido hasta que lo haga. En ese caso, se le muestra un botón de “Inicia sesión para pedir”, donde se le redirigirá al *login* para que pueda iniciar sesión o registrarse si ese fuera el caso.

Por otro lado, cuando una tienda no tiene productos, se mostrará el siguiente mensaje de aviso, recomendando a los potenciales clientes que existen otras formas de realizar su pedido.



Figura 87 - Alerta cuando no hay productos en una tienda

## AI.7. Resumen del pedido

Una vez el cliente se ha decidido a tramitar su pedido, llega al resumen del pedido (o también *checkout*), donde podrá ver todo lo relacionado con lo que va a comprar.

En el resumen del pedido, puede ver un listado de productos (que puede ser productos, imágenes subidas o texto plano) con un cálculo del precio total del mismo.

Además, aparte de mostrarle al usuario sus datos de contacto para que los verifique, también se muestran los datos de la tienda donde se realizará el pedido. De forma automática, se carga un mapa de Google con la dirección, facilitando al usuario su localización y evitar confusiones de donde pueda estar situada.

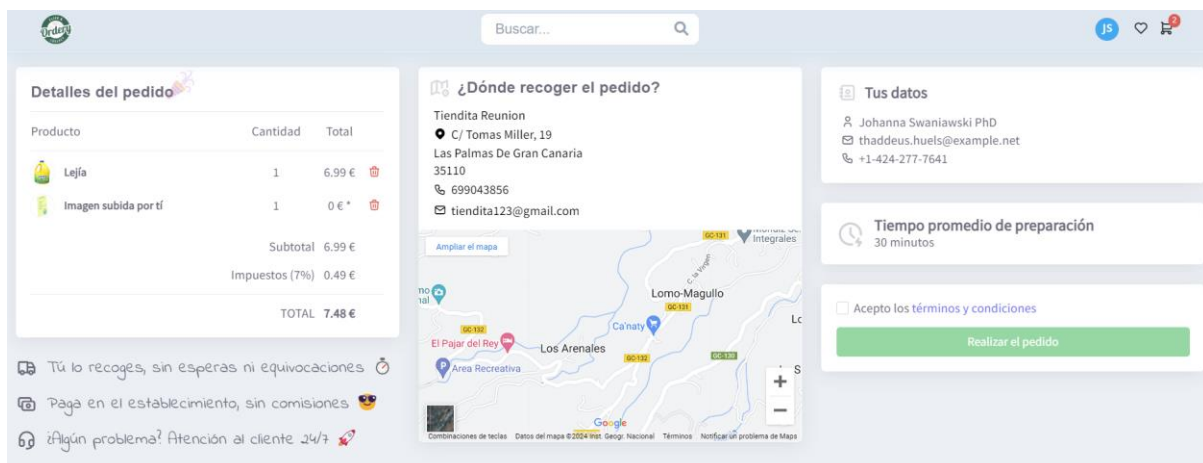


Figura 88 - Resumen del pedido

Una vez el usuario acepte los términos y condiciones de *Ordery*, se desbloqueará el botón “Realizar el pedido”, donde se tramitará y comenzará a gestionarse por parte del vendedor.

El tiempo promedio de preparación se calcula en base al tiempo que ha tardado la tienda de cambiar el estado de “Pendiente” a “Preparado”.

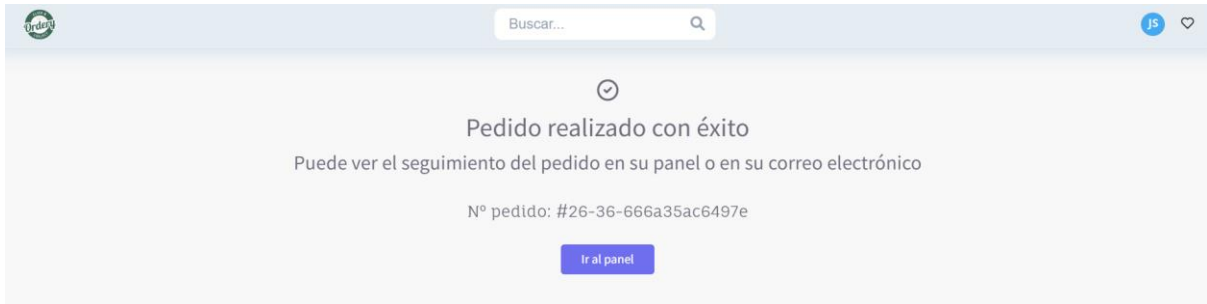


Figura 89 - Pedido realizado con éxito

Tras realizar el pedido, el usuario recibirá una notificación como que el pedido se ha realizado correctamente, su número de pedido y un botón que le permitirá acceder rápidamente a su panel de usuario para visualizar el estado y evolución de este.

## AI.8. Perfil del usuario

En el perfil del usuario podrá hacer seguimiento de su último pedido y de los anteriores. Como sucedía en el resumen del pedido (Figura 88), se muestra la localización y el mapa de la tienda, únicamente del último pedido.

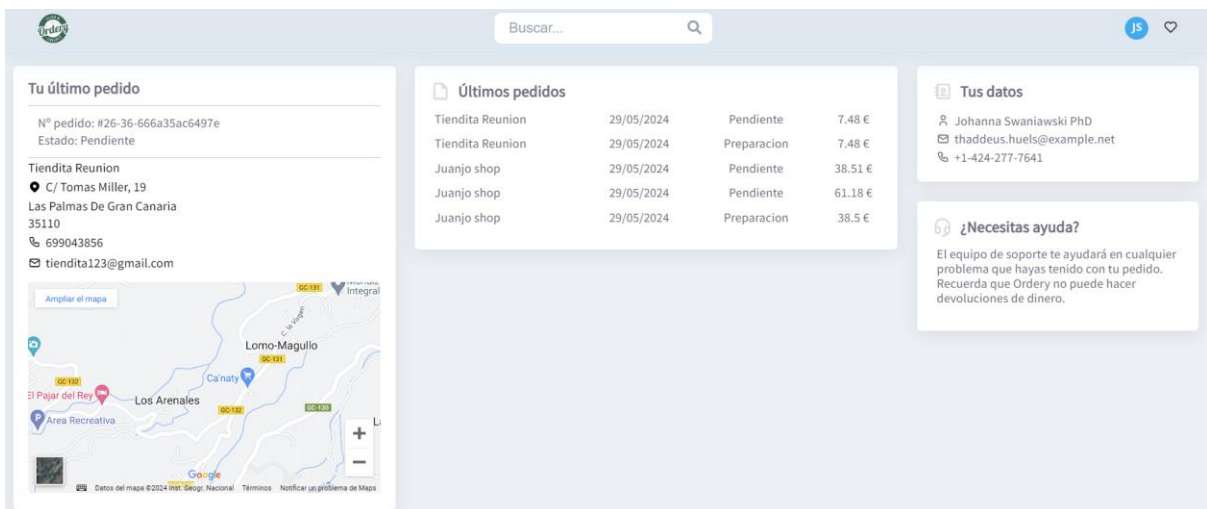


Figura 90 - Perfil de usuario

## AI.9. Panel de administración de administradores

El panel de administración es la parte más importante de una plataforma, ya que se lleva el control absoluto de las funciones y conocer el estado de la aplicación.

Aunque no se han elaborado complejas funciones de gestión y en apartados posteriores se comentarán posibles mejoras, se ha implementado elementos básicos.

Al iniciar sesión como administrador, se le redirige al usuario al *Dashboard* que se muestra en la *Figura 91*, donde se muestra un resumen con información relevante para el administrador.

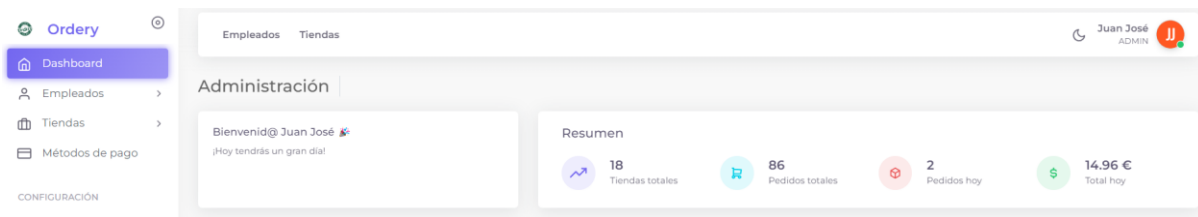


Figura 91 - Dashboard administrador

En el *dashboard* se muestra un contador de tiendas registradas, el número de pedidos totales, número de pedidos de hoy y los ingresos totales en pedidos.

En el panel lateral se encuentran los accesos directos a las secciones del panel de administrador: Empleados, Tiendas y Métodos de pago.

En la siguiente figura se mostrará el contenido de la pizarra de empleados y las posibilidades.

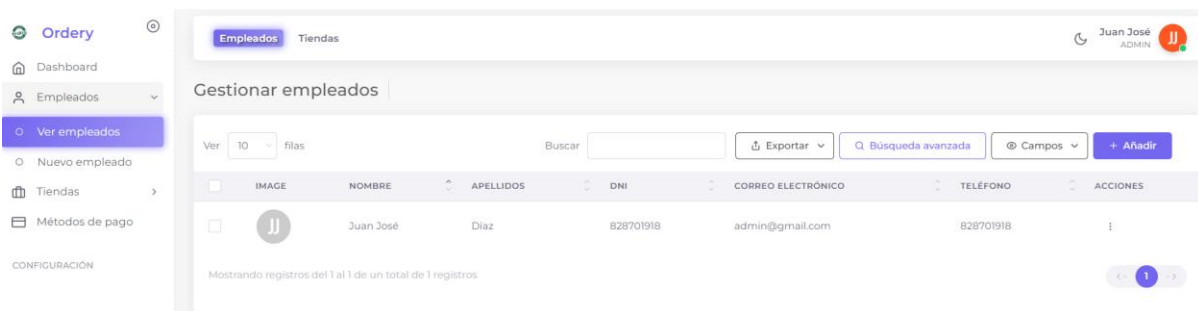


Figura 92 - Pizarra de empleados

En la pizarra de empleados, se muestra un “*datatable*<sup>8</sup>”, donde el contenido carga de forma optimizada y paginada. Cada empleado tiene una serie de acciones que se pueden llevar a cabo: ver, editar, eliminar o bloquear.

En la siguiente figura, se puede ver el formulario de ver o editar un empleado. Este formulario se ha perseguido la simplicidad que se mostraba en la maqueta y la legibilidad de todos los campos necesarios. Hay que tener en cuenta que el panel de administración no es necesario tener tanta consideración con la legibilidad o con la accesibilidad de este, pero igualmente el autor del trabajo consideró que era necesario no pasarlo por alto.

En el formulario se solicitará la información básica del empleado, una imagen, los datos de acceso a la plataforma y los permisos que tendrá.

---


<sup>8</sup> Librería de Javascript que permite potenciar las capacidades y la optimización de una tabla, proporcionando herramientas útiles para su configuración y presentación.

También desde el formulario se puede bloquear el perfil para evitar accesos no autorizados a la plataforma.

Nuevo empleado

🔍 Información del usuario

Imagen del empleado



Tamaño recomendado: 600 x 700

Nombre\*

Apellidos\*

DNI\*

Teléfono\*

Cargo en la empresa\*

Cuenta bloqueada

🔑 Acceso

Correo electrónico\*

Confirmar correo electrónico\*

Contraseña\*

Confirmar contraseña\*

📄 Lista de permisos

MÓDULO	VER	EDITAR	BORRAR	<input type="checkbox"/> SELECCIONAR TODOS
empleados	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
tiendas	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
metodos de pago	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
configuracion	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Figura 93 - Formulario de nuevo empleado

En la *Figura 94*, se muestra una pizarra de tiendas, en la que se muestra información tanto relevante para un cliente como para la administración: Nombre, nombre legal, CIF de la empresa, email, teléfono, URL y fecha de creación.

La particularidad que tiene las tiendas es que pueden elegir la URL donde los clientes pueden encontrarlas. En esta pizarra encontramos también la URL para que sea fácil acceder para los administradores.

Gestionar tiendas

Ver  filas

Buscar





<input type="checkbox"/>	IMAGE	NOMBRE	NOMBRE LEGAL	CIF	EMAIL	TELÉFONO	URL	FECHA	ACCIONES
<input type="checkbox"/>		Bähringer Ltd	Miller and Sons PLC	B37807897M	goldner.chaya@bradtke.com	1-216-995-4090	ut-et	19/05/2024	⋮
<input type="checkbox"/>		Bazar Gemma	Gemmazar S.L.	X382900A	bazargemma@gmail.com	888551125	bazar-gemma	19/05/2024	⋮
<input type="checkbox"/>		Block-Hane	Casper-McLaughlin LLC	M10763755O	rschoen@sipes.org	+1 (386) 673-2203	itague-sed	19/05/2024	⋮
<input type="checkbox"/>		Collins, Buckridge and Rowe	O'Kon-Rau and Sons	M85132860H	litzy.collins@baumbach.net	951.493.8776	similique-quidem	19/05/2024	⋮

Figura 94 - Pizarra de tiendas



La parte positiva que tiene usar *datatables* para gestionar la información es la posibilidad de añadir filtros eficientes que realizan una consulta optimizada para traer la información solicitada. Además, otro punto a favor es que se puede modificar el criterio de ordenación o de búsqueda para contemplar más o menos cosas a la hora de filtrar por campo. En la *Figura 95* se muestra los filtros que aparecen después de hacer clic en “Búsqueda avanzada”.

The screenshot shows a search interface with the following elements:
 

- Top left: "Ver 10 filas" (View 10 rows).
- Top center: "Buscar" (Search) input field.
- Top right: "Exportar" (Export) button, "Búsqueda avanzada" (Advanced search) button, "Campos" (Fields) dropdown, and "+ Añadir" (Add) button.
- Section: "Búsqueda avanzada" (Advanced search).
- Filters:
  - Nombre (Name): "Buscar por nombre" (Search by name)
  - Apellidos (Surnames): "Buscar por apellidos" (Search by surnames)
  - DNI: "Buscar por dni" (Search by dni)
  - Correo electrónico (Email): "Buscar por correo electrónico" (Search by email)
  - Teléfono (Phone): "Buscar por teléfono" (Search by phone)
- Bottom right: "Buscar" (Search) and "Limpiar" (Clear) buttons.

Figura 95 - Filtros de las pizarras

En la figura de a continuación, se mostrará el formulario para crear o editar una tienda. Al igual que la información que se solicitaba en la tabla, se solicita información tanto pública como información legal de la sociedad.

Es importante que se añada correctamente la dirección de la tienda, ya que en la web se muestra el mapa acorde a la dirección introducida.

The screenshot shows a form titled "Nueva tienda" (New store) with the following sections:
 

- Section: "Información básica" (Basic information).
  - Imagen de la tienda (Store image): A placeholder with a green circular icon and a recommended size of 600 x 700.
  - Nombre público (Public name), Nombre legal (Legal name), CIF, Teléfono (Phone), Correo electrónico (Email), and URL de acceso (Access URL) input fields.
  - Descripción (Description) text area.
- Section: "Datos necesarios" (Required data).
  - Dirección (Address), Código postal (Postal code), Ciudad (City), and Provincia (Province) input fields.
- Buttons: "Guardar" (Save) and "Cancelar" (Cancel).

Figura 96 - Formulario de tienda

Los métodos de pago son una característica importante en el desarrollo. En esta versión no se ha implementado en su totalidad, pero si se ha elaborado el *front* para dejarlo todo preparado para futuros desarrollos. En este caso, se ha implementado el pago con efectivo, *Paypal* y tarjeta de débito.

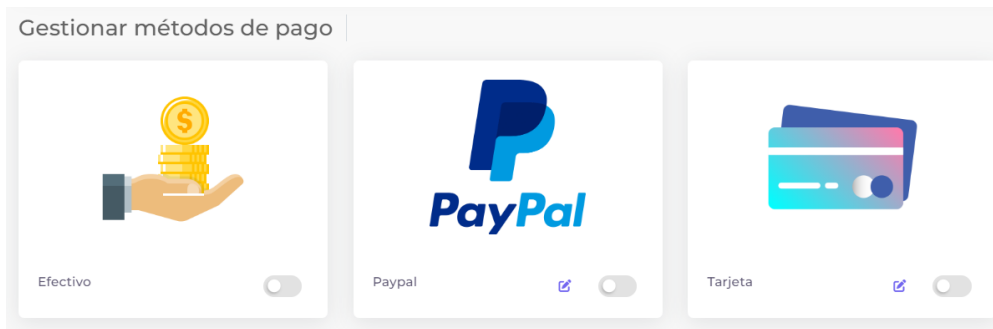


Figura 97 - Métodos de pago disponibles

Al editar el método de pago, aparecerá una ventana emergente para rellenar la información necesaria para su correcto funcionamiento. En la *Figura 98*, se muestra los campos necesarios para *Paypal*.

Figura 98 - Configuración de Paypal

Para continuar con la creciente tendencia de usar el modo nocturno, el panel de administración también se le puede activar el modo noche simplemente haciendo clic en el icono de la luna que aparece en el menú de navegación superior.

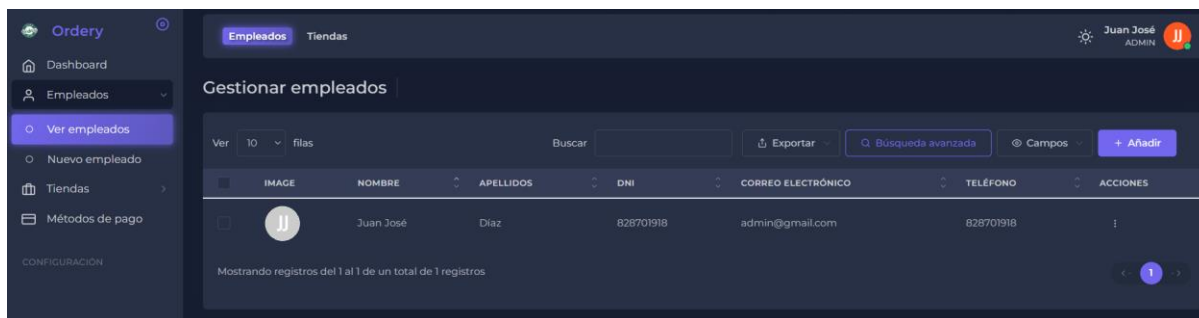


Figura 99 - Modo noche del panel admin

## AI.10. Panel de administración de tiendas

El panel de administración de una tienda es importante para llevar la gestión de los pedidos que se realizan, el control de los empleados y la personalización del catálogo de la tienda.

Hay que tener en cuenta que en el panel de administración de tiendas solo tendrá acceso aquellos empleados que hayan sido creados desde el propio panel, por un usuario perteneciente a la misma tienda. Además, al acceder al panel, solo será visible las secciones que el usuario tenga permiso.

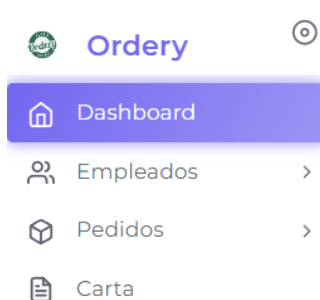


Figura 100 - Accesos directos laterales

Al igual que sucedía en el panel de administradores, al acceder los empleados de una tienda encontrarán un panel adaptado a sus necesidades con una apariencia similar.

Aunque puede parecer que todo está en la misma vista, las funciones y rutas están bloqueadas por permisos y tipo de usuario, por lo que es técnicamente imposible acceder a funciones privilegiadas. Además, las vistas están separadas, aumentando aún más la seguridad del entorno.

Como se puede ver en la *Figura 100*, se muestran los principales accesos directos del panel, donde se encuentra *Dashboard*, *Empleados*, *Pedidos* y *Carta*.

*Dashboard* es la página principal del panel, donde accederán los empleados al iniciar sesión. Una vez dentro, podrán ver un breve resumen de las estadísticas de la tienda como la cantidad de pedidos totales de hoy, la cantidad vendida hoy, productos disponibles en el catálogo y la fecha del último pedido.

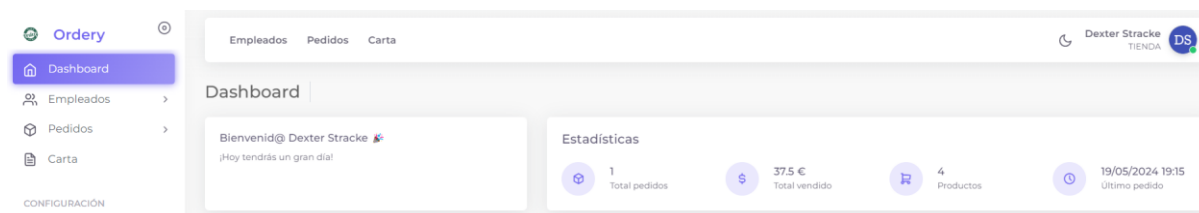


Figura 101 - Dashboard de empleado de una tienda

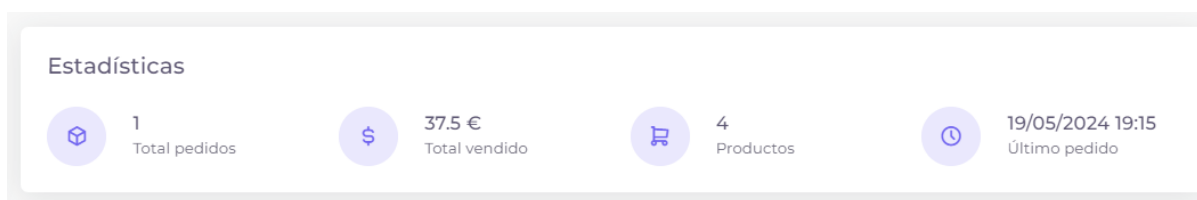


Figura 102 - Estadísticas de una tienda

En esta versión no se ha profundizado demasiado en las estadísticas, donde sería atractivo indagar más en los intereses de los empleados y administradores para añadir información útil.

Los empleados de las tiendas con permisos de gestionar empleados pueden acceder a la pizarra de usuarios con acceso a la tienda.

Como se muestra en la *Figura 103*, el listado de empleados contiene coloreados de rojo aquellos que se encuentren bloqueados actualmente. Además, como ya se ha visto en apartados anteriores, el formato de la tabla es similar a ellos.

Esta pizarra contiene un avatar del usuario, el nombre, apellidos, DNI, correo electrónico, teléfono y acciones.

Gestionar empleados

Ver 10 filas

Buscar

Exportar

Búsqueda avanzada

Campos

Añadir

IMAGE	NOMBRE	APELLIDOS	DNI	CORREO ELECTRÓNICO	TELÉFONO	ACCIONES
<input checked="" type="checkbox"/> DS	Dexter Stracke	Jacobs	51992891K	scarlett21@example.org	+1 (347) 918-6229	:
<input checked="" type="checkbox"/> DR	Dr. Rex Rempel	Littel	51999996O	jeramy43@example.net	571.895.4025	:
<input type="checkbox"/> DT	Dr. Thalia Senger III	Rutherford	69493657S	percival.berge@example.org	929.615.2151	:
<input type="checkbox"/> GB	Graciela Bailey	Berge	33773778I	ressie.west@example.net	+1-262-917-6438	:

Figura 103 - Pizarra de empleados

Esta pizarra y las siguientes que se detallarán en este apartado, contienen el botón de “Búsqueda avanzada”, que muestra una serie de campos por el cual se puede filtrar la pizarra.

La diferencia de realizar la búsqueda en “Buscar” en lugar de un campo específico en búsqueda avanzada, es que, el campo buscar lo hace en todas las columnas participantes del *datatable*.



Figura 104 - Filtros de la pizarra de empleados

También se incluye el botón “Campos” que permite mostrar u ocultar columnas del *datatable* para elegir visualmente los datos que se deseen.



Figura 105 - Filtro de visibilidad del *datatable*

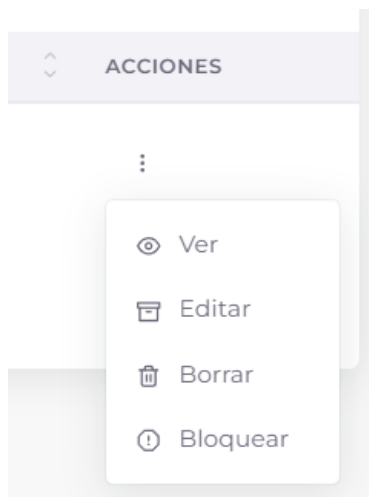


Figura 106 - Acciones de la pizarra de empleados

Cada registro (fila) que existen en el *datatable*, contiene acciones que se pueden utilizar como atajos rápidos, como por ejemplo la acción “bloquear”. Esta acción permite bloquear al empleado de forma directa sin tener que editar el registro.

Las acciones pasan por una verificación de permisos del usuario, donde se muestran o no en el *DOM* (Modelo de objeto de documento)<sup>9</sup> dependiendo de los permisos. Es importante que estas funciones aparte de tenerlas controladas en el *backend* del proyecto también se controle estos aspectos en el *front*, puesto que alguna persona con conocimientos de informática puede alterar el código fuente en el navegador y

mostrar los botones ocultos y realizar acciones privilegiadas.

Por otro lado, también es posible exportar los datos de la tabla haciendo clic en el botón “Exportar” de la tabla, donde nos aparecerá diferentes opciones como PDF, CSV o Imprimir.

<sup>9</sup> DOM es un conjunto de APIs que nos permite gestionar los elementos de un documento.

En la siguiente figura se muestra un ejemplo de exportación de empleados en un PDF.

## Gestionar empleados

Nombre	Apellidos	DNI	Correo electrónico	Teléfono
Dexter Stracke	Jacobs	51992891K	scarlett21@example.org	+1 (347) 918-6229
Dr. Rex Rempel	Littel	51999996O	jeramy43@example.net	571.895.4025
Dr. Thalia Senger III	Rutherford	69493657S	percival.berge@example.org	929.615.2151
Graciela Bailey	Berge	33773778I	ressie.west@example.net	+1-262-917-6438
Jordan Abshire IV	Jacobson	06672831P	swift.marilou@example.net	956-574-6535
Prof. Ena Barton IV	Kulas	96645889Y	heller.giuseppe@example.com	351.748.3632
Prof. Mortimer Johns Sr.	Keebler	51106566G	creichert@example.com	218-847-3525
Theresa Waelchi	Bruen	51754462Q	vbeer@example.org	+1 (770) 450-4191

Figura 107 - Ejemplo de exportación en PDF de empleados

Uno de los apartados más relevantes para las tiendas, es la pizarra de pedidos. En esta pizarra deberán llevar el control y gestión de todas las órdenes.

En esta pizarra se muestran las columnas documento (ID del pedido), cliente que realizó el pedido, un botón de ver que explicaremos a continuación, observaciones del pedido, fecha de entrega del pedido, fecha de creación, estado y acciones.

Para hacer más rápido la identificación de los pedidos y visualmente cómodo, se establece colores para cada uno de los posibles estados. Estos estados pueden ser:

- Pendiente
- En preparación
- Preparado
- Finalizado
- Cancelado

DOCUMENTO	CLIENTE	PEDIDO	OBSERVACIONES	FECHA ENTREGA	FECHA CREACIÓN	ESTADO	ACCIONES
test1	Juan José Díaz	Ver	¡El dulce que sea de hoy!	19/05/2024 19:15	19/05/2024 19:15	Pendiente	⋮
test2	Juan José Díaz	Ver	¡El dulce que sea de hoy!	19/05/2024 19:15	19/05/2024 19:15	Preparacion	⋮
test3	Juan José Díaz	Ver	¡El dulce que sea de hoy!	19/05/2024 19:15	19/05/2024 19:15	Preparado	⋮
test4	Juan José Díaz	Ver	¡El dulce que sea de hoy!	19/05/2024 19:15	19/05/2024 19:15	Finalizado	⋮
test5	Juan José Díaz	Ver	¡El dulce que sea de hoy!	19/05/2024 19:15	19/05/2024 19:15	Cancelado	⋮

Figura 108 - Datatable de pedidos

En la columna pedido, se muestra un botón “Ver” que permite activar una ventana emergente que muestra el resumen del pedido.

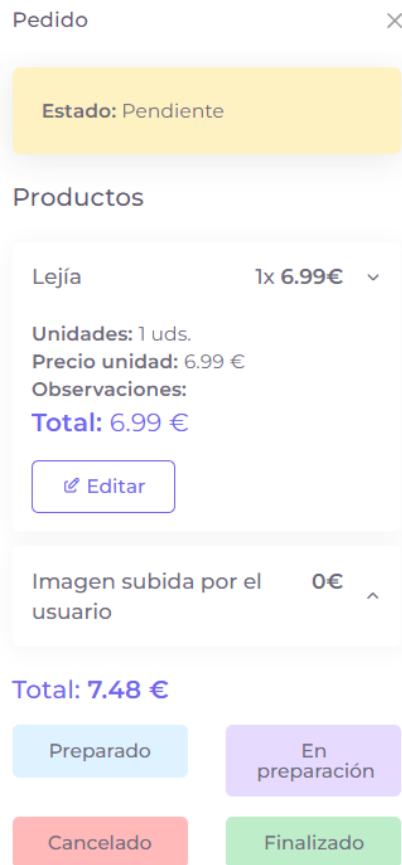


Figura 109 - Resumen del pedido para la tienda

En primera instancia, se muestra el estado del pedido y su estado, para una rápida identificación.

Posteriormente, se muestra un listado de los productos con un formato elegante de acordeón, donde aparece cada uno de los ítems replegados y al hacer clic se muestra la información.

Dentro de cada producto, se muestra la cantidad, el precio, y las observaciones si las hubiera.

Una particularidad de *Ordery* es que el empleado de la tienda puede editar cada línea del pedido modificando el precio total del mismo. Esto es útil cuando las solicitudes o los productos tienen ligeras variaciones, como lo pueden ser los productos a granel o que la tienda no tenga todas las unidades que solicita el cliente.

Si el empleado hace clic en editar, se muestra un campo donde introducir el nuevo precio total del producto y un botón de guardar, tal y como se muestra en la *Figura 110*.

Al realizar un cambio en una línea del pedido, se recalcula el total nuevamente, teniendo en cuenta los impuestos y se

muestra un mensaje de actualización satisfactoria (*Figura 111*).

Para facilitar la edición, en la parte inferior del mismo se encuentran botones de acción rápida para cambiar el estado del pedido con solamente un clic.



Figura 110 - Editando el total de un producto del pedido

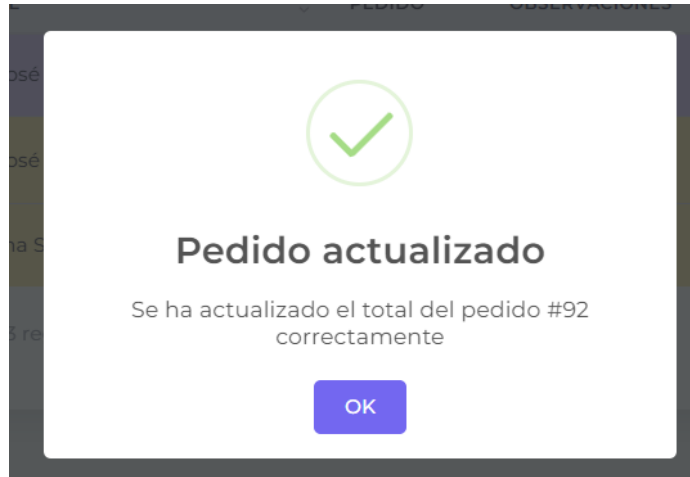


Figura 111 - Mensaje de pedido actualizado

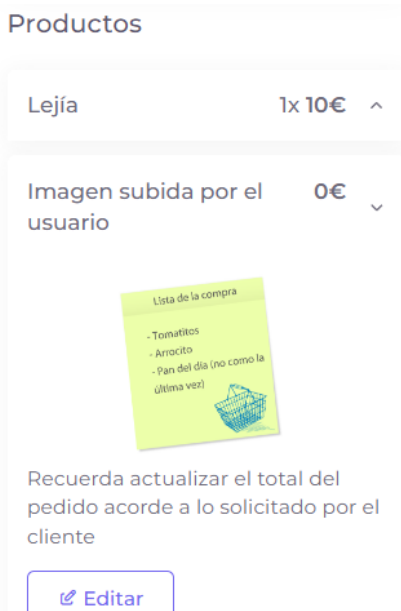


Figura 112 - Pedido con imagen subida por el usuario

También es posible ver las imágenes subidas por el usuario y los textos que han introducido a la hora de realizar el pedido.

En un principio esta línea aparece como 0€ ya que un empleado de la tienda tendrá que calcular lo solicitado por el cliente. Una vez modificado se le enviará un correo al cliente con las actualizaciones.

Además, se le muestra un recordatorio al empleado para que tenga en cuenta que debe actualizar el pedido.

Para mejorar la visibilidad de las imágenes subidas por los clientes, se puede hacer clic en ella para ampliarla.

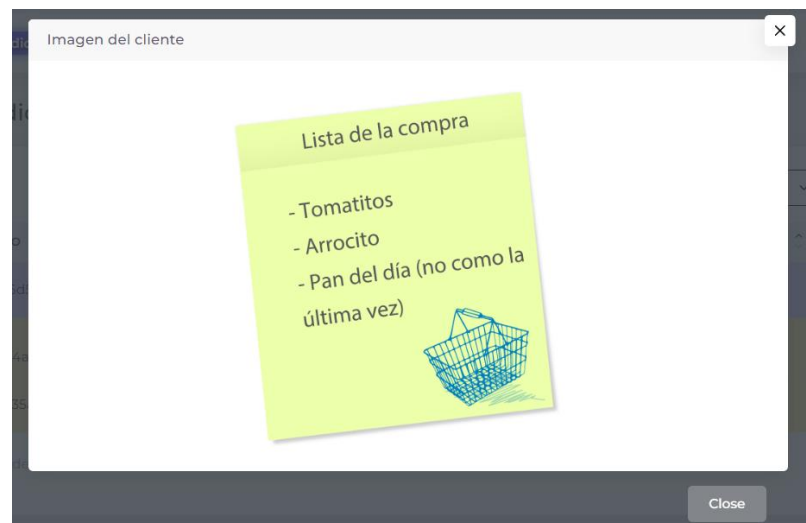


Figura 113 - Zoom de la imagen subida por el usuario



Por último, la sección del catálogo de la tienda. Esta sección permite poder configurar tanto categorías como productos que posteriormente se harán visibles en la página de la tienda.

Es importante que las tiendas tengan un catálogo para dar visibilidad a sus productos, aumentando la captación y retención de posibles clientes.

Al acceder al catálogo, entramos directamente en el modo editar categoría. Esta función muestra las categorías que se han creado con su descripción, como se muestra en la *Figura 114*.

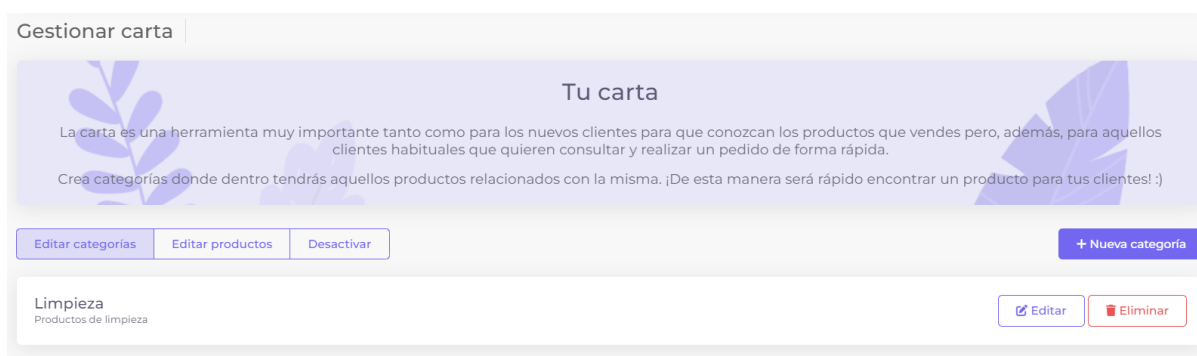


Figura 114 - Carta editando categorías

Cada categoría se puede editar o eliminar. También este apartado tiene implementado la funcionalidad de “*drag and drop*” (en español, arrastrar y soltar), que permite reordenar las categorías en el orden que se desee.

Cuando se añade o se edita una categoría, se muestra una ventana emergente por la derecha con los campos relacionados, como nombre de la categoría y descripción.

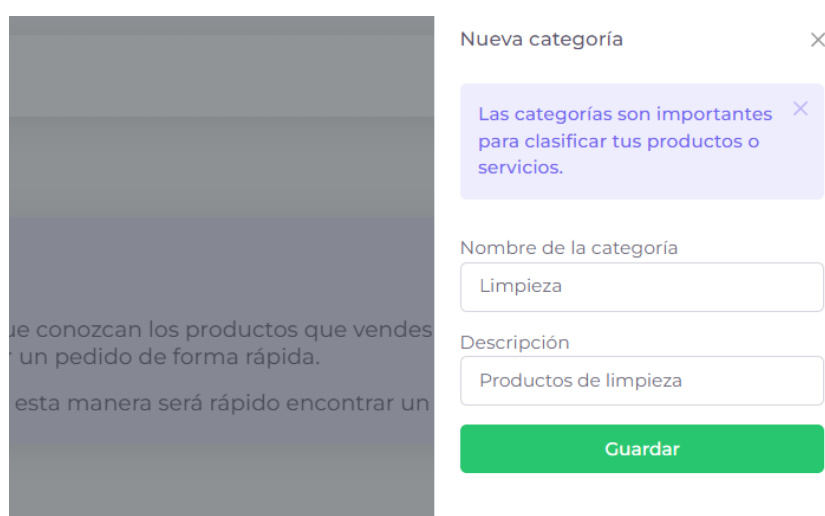


Figura 115 - Editando una categoría del catálogo

Tras crear y editar las categorías, es turno de poblarlos con productos. Para ello, se debe hacer clic en el botón de la parte superior que indica “editar productos”. Con esta acción ya

no se puede ordenar las categorías, pero ahora se activa el *drag and drop* para los productos, pudiendo elegir su posición.

Ahora, tras seleccionar “editar productos” se muestran los productos asignados a cada una de las categorías.

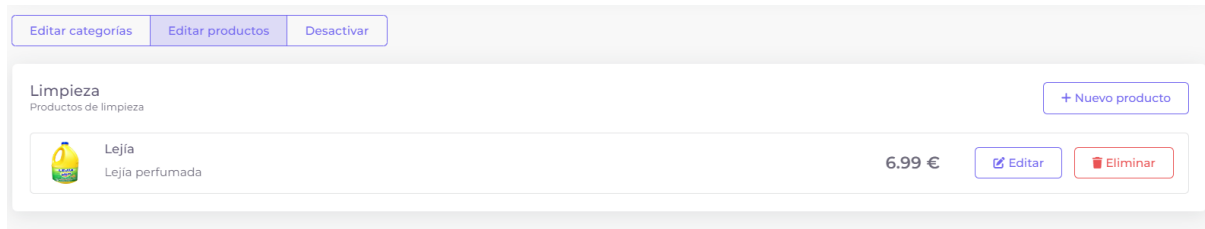


Figura 116 - Producto en una categoría del catálogo



Al igual que sucedía con las categorías, al crear y editar un producto, aparece una ventana emergente por la derecha de la pantalla que muestra los campos necesarios. En este caso, se le solicita al empleado de una tienda una imagen del producto, nombre del producto, descripción y precio.

Hay que tener en cuenta que la imagen no puede superar el megabyte de espacio, puesto que superar este límite implica perjudicar el rendimiento de la plataforma a la hora de cargar el perfil de la tienda. Además, se muestra un mensaje con el tamaño recomendado de los productos para su correcta visualización en la plataforma.

Una vez guardamos el producto, se asigna a la categoría, estando automáticamente visible en la plataforma para su compra.

Figura 117 - Editando un producto del catálogo

A modo de conclusión de este apartado, en los paneles de administración tanto para tienda como administradores, son capaces de personalizar, gestionar y controlar diferentes aspectos de los que tienen permisos. Con la implementación de una interfaz sencilla y ayudas a lo largo del panel, permite que cualquier persona se sienta cómoda utilizándolo sin ningún inconveniente.