

ABHI'S

TIENDA DE INFORMÁTICA



ABHIJEET SINGH
PROYECTO FINAL - 2ºDAW
TUTOR - JOSE FIDEL OLTRA LANDETE
I.E.S. Jaume II el Just
31/05/2024

Datos del Proyecto

Datos del alumno

Nombre y Apellido	Abhijeet Singh
DNI	
Curso y Ciclo Formativo	2º DAW

Datos del proyecto

Título del proyecto	ABHI'S
Nombre de tutor individual	José
Nombre de tutor del grupo	José
Resumen	Tienda E-Commerce de productos de informática y electrónica.
Abstract	E-Commerce Store of IT and electronics products.
Módulos Implicados	<ul style="list-style-type: none">• DWES (Desarrollo web en entorno de Servidor)• DWEC (Desarrollo web en entorno Cliente)• DAW (Despliegue de Aplicaciones Web)• DIW (Diseño de Interfaz web)• Ingles técnico• EIE (Empresa e iniciativa emprendedora)
Fecha de presentación	05 de junio de 2024

Índice

Tabla de contenido

Datos del Proyecto	0
Datos del alumno	0
Datos del proyecto.....	0
Índice	1
1. Introducción / Marco del proyecto	5
1.1 Descripción del proyecto	5
1.2 Objetivos	5
1.3 Tipo de proyecto	5
1.4 Orientaciones para el desarrollo y recursos	6
2. Análisis/Estudio del estado actual	7
2.1 Descripción del sistema actual	7
2.2. Viabilidad del sistema actual.....	7
2.3. Requerimientos del nuevo sistema	8
3. Diseño de la solución	9
3.1. Análisis de las posibles soluciones	9
3.2. Evaluación de las posibles soluciones	10
Justificación de la solución elegida	12
3.3. Descripción de la solución escogida	12
3.3.1. Definición de las tareas y subtareas en las que dividiremos el proyecto ...	12
3.3.2. Estimación del coste temporal de cada una de las tareas	14
3.3.3. Estimación del coste económico.....	14
4. Contenido del Proyecto y Herramientas.....	16
4.1 Contenido.....	16
4.2 Herramientas	16
4.2.1 Backend	16
4.2.2 Frontend.....	18
4.2.3 Base de Datos	24
4.2.4 Otras herramientas utilizadas	25
5. Estructura del proyecto	28

5.1 Estructura de base de datos	28
5.2 Estructura de proyecto.....	32
6. Creación de Base de datos.....	34
7. Creación de Servidor de Correos	37
Configuración del Servidor de Correos con SendGrid	37
8. Autenticación.....	38
8.1 Configuración del Bundle JWT	38
8.2 Mejorando la Experiencia del Usuario	40
Creación de Event Listener.....	40
9. Configuración de pasarela de pagos (API de Stripe)	43
9.1 Instalación de Stripe	43
9.2 Creación del Controlador de Stripe.....	43
9.2 Ventajas de Usar Stripe	45
10. Estructura final del Backend	46
11. Parte Cliente – Frontend Tienda	49
11.1 Creación de la aplicación	49
11.2 Proceso de desarrollo	50
11.3 Interfaz de la aplicación	56
Página de Inicio	58
Navbar.....	61
Página de Productos	62
Carrito	64
Pago	65
Página de Pedidos	66
Perfil.....	67
Pagina de Contacto	67
Error 404.....	68
Pagina de producto individual	69
Reseñas.....	69
12. Parte Cliente – Admin	71
12.1 Contenido.....	71
12.2 Interfaz de la Aplicación	75

Home Inicio.....	75
SideBar.....	76
Página de Productos	77
Añadir producto	77
Editar Producto	77
Marcas Brands.....	77
Añadir Nueva marca	78
Categorías	78
Añadir nueva categoría	78
Editar una categoría.....	79
Pedidos	79
Inventario	79
Usuarios Clientes	80
Contacto	80
13. Despliegue	81
Backend.....	81
Frontend	81
14. Conclusiones y Mejoras Futuras	82
14.1 Conclusiones	82
14.2 Mejoras Futuras.....	82
15. Bibliografía	84

1. Introducción / Marco del proyecto

1.1 Descripción del proyecto

Este proyecto consiste en el desarrollo de una tienda e-Commerce especializada en la venta de productos de informática, tecnología, telefonía y electrónica. La plataforma está diseñada para proporcionar una experiencia de compra fácil, rápida, accesible e intuitiva, ofreciendo una amplia gama de productos, desde ordenadores, teléfonos móviles, tabletas, relojes inteligentes y accesorios hasta los últimos gadgets tecnológicos.

1.2 Objetivos

El proyecto persigue alcanzar los siguientes objetivos:

1. Facilitar la experiencia de compra online para los usuarios.
2. Ofrecer una plataforma moderna y accesible que se adapte a las tendencias actuales del comercio electrónico.
3. Integrar un sistema de gestión eficiente para el inventario y las ventas.
4. Implementar un diseño responsive que garantice la usabilidad en dispositivos móviles.
5. Optimizar los procesos de pago y envío para mejorar la satisfacción del cliente.
6. Proveer soporte técnico y asistencia a los usuarios.

1.3 Tipo de proyecto

Este proyecto es de ámbito externo, ya que está destinado a usuarios finales (clientes compradores) fuera de la organización que lo desarrolla. Los principales requerimientos del proyecto incluyen:

- Requerimientos de software: Desarrollo de una aplicación web utilizando tecnologías como PHP, Symfony 6, API Platform para la creación de la API REST, Stripe para pagos (con tarjeta, Apple Pay, Google Pay, Klarna etc.) y Vue.js para el frontend.
- Requerimientos de infraestructura: Implementación y despliegue de la aplicación backend (REST API) en Microsoft Azure y el frontend en Vercel.
- Requerimientos de usabilidad: Crear una interfaz de usuario intuitiva y accesible, mejorando así la experiencia del usuario durante el proceso de compra.
- Requerimientos de seguridad: Asegurar que todas las transacciones y datos de los usuarios estén protegidos mediante protocolos de seguridad avanzados.

1.4 Orientaciones para el desarrollo y recursos

Para el desarrollo de este proyecto, se han seguido diversas orientaciones y se han utilizado múltiples recursos bibliográficos y web. El desarrollo ha sido guiado por el tutor individual José Fidel Oltra Landete, cuya orientación ha sido crucial en la planificación y ejecución del proyecto. Los recursos consultados incluyen:

- Documentación oficial de cada lenguaje y tecnología utilizada:
 - PHP: php.net
 - Symfony 6: symfony.com
 - API Platform: api-platform.com
 - Vue.js: vuejs.org
- Foros y comunidades:
 - Stack Overflow: Resolución de dudas técnicas y problemas específicos.
 - GitHub: Ejemplos de proyectos, código abierto y contribuciones de la comunidad.
- Plataformas de despliegue:
 - Microsoft Azure: Para el despliegue de la aplicación backend.
 - Vercel: Para el despliegue del panel de administración en Vue.js y el frontend para los usuarios.

Estos recursos han proporcionado una base sólida para la implementación y han asegurado que el proyecto se desarrolle siguiendo las mejores prácticas de la industria. Los detalles completos de los recursos utilizados se encuentran al final del documento en la bibliografía.

2. Análisis/Estudio del estado actual

2.1 Descripción del sistema actual

Actualmente, la situación del mercado de e-Commerce para productos de informática, tecnología y electrónica presenta una serie de desafíos y oportunidades. Muchos de los competidores en este sector utilizan plataformas comerciales genéricas que, aunque funcionales, no están completamente optimizadas para las necesidades específicas de estos productos. Los sistemas actuales a menudo carecen de:

1. Personalización: Interfaces de usuario no adaptables a las preferencias y comportamientos individuales de los usuarios.
2. Integración: Falta de integración eficiente con proveedores y servicios de logística.
3. Escalabilidad: Limitaciones en la capacidad de escalar y gestionar grandes volúmenes de tráfico y datos.
4. Interfaz de usuario: Diseños que no son intuitivos o atractivos, lo que dificulta la navegación y la experiencia de compra.
5. Velocidad y rendimiento: Tiempos de carga lentos y problemas de rendimiento que afectan la satisfacción del usuario.

En base a este análisis, podemos identificar varias áreas críticas que requieren mejoras para satisfacer las expectativas de los consumidores modernos y competir efectivamente en el mercado.

2.2. Viabilidad del sistema actual

Evaluando la viabilidad del sistema actual, se identifican las siguientes conclusiones:

1. Limitaciones de personalización: El sistema actual no permite una personalización adecuada, lo que impide que los usuarios tengan una experiencia de compra única y satisfactoria.
2. Problemas de integración: La falta de integración con sistemas de proveedores y logística resulta en procesos inefficientes y retrasos en la entrega.
3. Dificultades de escalabilidad: El sistema actual no está preparado para manejar un crecimiento rápido en términos de tráfico y volumen de transacciones.
4. Experiencia de usuario deficiente: La interfaz no es intuitiva ni atractiva, lo que reduce la retención de clientes y la tasa de conversión.
5. Rendimiento insuficiente: Tiempos de carga lentos y problemas de rendimiento afectan negativamente la experiencia del usuario y la satisfacción general.

Con base en esta evaluación, el sistema actual no es viable en su forma presente y se requiere una remodelación significativa para alcanzar los objetivos propuestos. Por lo tanto, se justifica plenamente la realización del nuevo proyecto.

2.3. Requerimientos del nuevo sistema

Para abordar las deficiencias identificadas en el sistema actual y alcanzar los objetivos del proyecto, se han establecido los siguientes requerimientos para el nuevo sistema:

1. Personalización avanzada:
 - Implementar funciones que permitan a los usuarios personalizar su experiencia de compra, incluyendo recomendaciones personalizadas basadas en su historial de navegación y compras.
2. Integración eficiente:
 - Integrar la plataforma con proveedores y servicios de logística para asegurar un flujo de información continuo y minimizar los tiempos de entrega.
3. Escalabilidad robusta:
 - Diseñar la arquitectura del sistema para soportar un crecimiento exponencial en el tráfico y el volumen de transacciones, utilizando tecnologías escalables y microservicios.
4. Mejora en la interfaz de usuario:
 - Desarrollar una interfaz de usuario moderna, intuitiva y responsive que facilite la navegación y mejore la experiencia de compra, adaptándose a dispositivos móviles y de escritorio.
5. Rendimiento y velocidad optimizados:
 - Optimizar el rendimiento de la plataforma para reducir los tiempos de carga y mejorar la respuesta del sistema, utilizando técnicas de optimización de base de datos y carga de contenido.
6. Seguridad y protección de datos:
 - Implementar medidas de seguridad avanzadas para proteger la información personal y financiera de los usuarios, cumpliendo con las normativas de protección de datos.

Estos requerimientos aseguran que el nuevo sistema no solo supere las deficiencias del actual, sino que también establezca una base sólida para el crecimiento y la satisfacción del cliente a largo plazo.

3. Diseño de la solución

3.1. Análisis de las posibles soluciones

Para satisfacer los requerimientos del nuevo sistema, se han identificado varias soluciones tecnológicas que podrían implementarse. Cada solución tiene características distintivas que podrían influir en su elección. Las posibles soluciones son:

1. Solución A: Plataforma basada en Magento
 - Características:
 - Plataforma robusta y flexible, ampliamente utilizada en eCommerce.
 - Gran cantidad de extensiones y módulos disponibles.
 - Soporte para personalización avanzada y escalabilidad.
 - Comunidad activa y extensa documentación.
 - Requiere conocimientos avanzados para configuración y mantenimiento.
2. Solución B: Plataforma basada en Shopify
 - Características:
 - Solución de eCommerce alojada en la nube, fácil de configurar y utilizar.
 - Amplia variedad de temas y aplicaciones para personalización.
 - Integración sencilla con proveedores y servicios de logística.
 - Escalabilidad limitada en comparación con soluciones más personalizadas.
 - Dependencia de una plataforma propietaria y tarifas adicionales.
3. Solución C: Desarrollo a medida utilizando Symfony y Vue.js
 - Características:
 - Total control sobre el desarrollo y la funcionalidad del sitio web.
 - Posibilidad de crear una arquitectura completamente personalizada y escalable.
 - Utilización de Symfony para el backend y Vue.js para el frontend, aprovechando sus fortalezas en rendimiento y modularidad.
 - Integración directa con API Platform para la creación de APIs RESTful.
 - Mayor tiempo y costo de desarrollo inicial en comparación con soluciones comerciales.
4. Solución D: Utilización de WooCommerce en WordPress
 - Características:
 - Basado en WordPress, lo que facilita la administración de contenidos y SEO.

- Gran cantidad de plugins y temas disponibles para personalización.
- Adecuado para tiendas pequeñas y medianas.
- Limitaciones en la escalabilidad y rendimiento para tiendas muy grandes.
- Dependencia de plugins de terceros para funcionalidades avanzadas.

3.2. Evaluación de las posibles soluciones

Para evaluar las diferentes soluciones, se utilizará una matriz DAFO (Debilidades, Amenazas, Fortalezas y Oportunidades) para cada una y se analizarán criterios objetivos como la escalabilidad, el costo, la facilidad de implementación, y la capacidad de personalización.

Solución A: Plataforma basada en Magento

- **Fortalezas:**
 - Robusta y flexible.
 - Gran cantidad de extensiones.
 - Comunidad activa y amplia documentación.
- **Debilidades:**
 - Requiere conocimientos avanzados.
 - Complejidad en la configuración y mantenimiento.
- **Oportunidades:**
 - Adaptabilidad para grandes volúmenes de productos y tráfico.
 - Potencial para integraciones avanzadas.
- **Amenazas:**
 - Costo elevado de mantenimiento y desarrollo.
 - Competencia de otras plataformas de eCommerce.

Solución B: Plataforma basada en Shopify

- **Fortalezas:**
 - Fácil de configurar y utilizar.
 - Integración sencilla con logística y proveedores.
 - Hospedaje en la nube.

- **Debilidades:**
 - Limitada escalabilidad.
 - Dependencia de una plataforma propietaria.
- **Oportunidades:**
 - Rápida implementación.
 - Soporte y actualizaciones continuas de la plataforma.
- **Amenazas:**
 - Costos adicionales por tarifas.
 - Menor control sobre personalización avanzada.

Solución C: Desarrollo a medida utilizando Symfony y Vue.js

- **Fortalezas:**
 - Control total sobre el desarrollo y funcionalidades.
 - Arquitectura personalizada y altamente escalable.
 - Uso de tecnologías modernas y flexibles.
- **Debilidades:**
 - Mayor tiempo y costo de desarrollo inicial.
 - Requiere un equipo de desarrollo con experiencia.
- **Oportunidades:**
 - Personalización completa según las necesidades del proyecto.
 - Adaptabilidad a futuros cambios y expansiones.
- **Amenazas:**
 - Riesgo de retrasos en el desarrollo.
 - Dependencia de recursos humanos especializados.

Solución D: Utilización de WooCommerce en WordPress

- **Fortalezas:**
 - Fácil administración de contenidos y SEO.
 - Gran cantidad de plugins y temas.
 - Comunidad activa.

- **Debilidades:**
 - Limitaciones en escalabilidad y rendimiento.
 - Dependencia de plugins de terceros.
- **Oportunidades:**
 - Ideal para tiendas pequeñas y medianas.
 - Rápida implementación.
- **Amenazas:**
 - Competencia de otras plataformas más robustas.
 - Problemas de seguridad con plugins de terceros.

Justificación de la solución elegida

Después de evaluar las posibles soluciones, la Solución C: Desarrollo a medida utilizando Symfony y Vue.js se considera la mejor opción. Esta solución proporciona la mayor flexibilidad y control sobre el sistema, permite una personalización avanzada y asegura una escalabilidad robusta para el crecimiento futuro. Aunque tiene un mayor costo y tiempo de desarrollo inicial, las ventajas en términos de adaptabilidad y potencial de personalización justifican la elección.

La Solución C garantiza que el proyecto podrá evolucionar y adaptarse a las necesidades específicas del mercado de productos de informática, tecnología y electrónica, ofreciendo una experiencia de usuario superior y posicionándose competitivamente en el mercado.

3.3. Descripción de la solución escogida

La solución escogida para este proyecto es el desarrollo a medida utilizando Symfony y Vue.js. Esta elección nos permitirá crear una plataforma eCommerce altamente personalizada, escalable y eficiente, que cumpla con los requerimientos y objetivos establecidos. A continuación, se detallan las diferentes tareas y subtareas necesarias para implementar esta solución, así como la estimación de costes temporales y económicos.

3.3.1. Definición de las tareas y subtareas en las que dividiremos el proyecto

1. Planificación (1 semana)

- 1.1. Reuniones iniciales con stakeholders
- 1.2. Definición de requerimientos

- 1.3. Análisis de viabilidad
- 1.4. Elaboración del plan de proyecto
- 1.5. Definición del cronograma y asignación de recursos

2. Desarrollo del Backend (3 semanas)

- 2.1. Configuración del entorno de desarrollo
- 2.2. Diseño de la arquitectura del backend
- 2.3. Implementación de la base de datos
- 2.4. Desarrollo de APIs REST utilizando Symfony
- 2.5. Integración con API Platform
- 2.6. Pruebas unitarias y de integración

3. Desarrollo del Frontend (6 semanas)

- 3.1. Configuración del entorno de desarrollo para el frontend
- 3.2. Diseño de la interfaz de usuario
 - 3.2.1. Diseño del panel de administración
 - 3.2.2. Diseño del frontend para usuarios
- 3.3. Implementación del panel de administración (Vue.js)
 - 3.3.1. Desarrollo de componentes UI
 - 3.3.2. Integración con APIs del backend
 - 3.3.3. Pruebas y depuración
- 3.4. Implementación del frontend de la tienda (Vue.js)
 - 3.4.1. Desarrollo de componentes UI
 - 3.4.2. Integración con APIs del backend
 - 3.4.3. Pruebas y depuración
- 3.5. Optimización y mejora del rendimiento
- 3.6. Pruebas finales y ajustes

4. Despliegue y pruebas finales (2 semanas)

- 4.1. Configuración del entorno de producción en Microsoft Azure
- 4.2. Despliegue del Backend

- 4.3. Configuración del entorno de producción en Vercel
- 4.4. Despliegue del Frontend
- 4.5. Pruebas de aceptación del usuario (UAT)
- 4.6. Resolución de problemas y ajustes finales

5. Formación y documentación (1 semana)

- 5.1. Creación de documentación técnica
- 5.2. Formación del equipo de soporte y administración
- 5.3. Entrega de la documentación al cliente

6. Lanzamiento y seguimiento (1 semana)

- 6.1. Lanzamiento oficial del eCommerce
- 6.2. Monitoreo y análisis post-lanzamiento
- 6.3. Soporte inicial y resolución de incidencias

3.3.2. Estimación del coste temporal de cada una de las tareas

Tarea	Duración
Planificación	1 semana
Desarrollo del Backend	3 semanas
Desarrollo del Frontend	6 semanas
Integración y despliegue	1 semana
Documentación	1 semana
Duración total estimada	12 semanas

3.3.3. Estimación del coste económico

He elegido que aplicación tenga alojamiento en la nube de Microsoft Azure. Esto ofrece más seguridad, alta disponibilidad, gran escalabilidad, seguridad en el tratamiento de datos y una personalización mayor a la hora de asignar los recursos necesarios al nuestro proyecto.

A continuación, se muestra una tabla con los costes estimados. Los precios pueden variar según el proveedor, y se ha optado por un coste aproximado.

COSTES DE INFRASTRUCTURA

	Coste mensual	Coste Anual
Gastos de alojamiento en Microsoft Azure	50€	600€

Gastos de pasarela de Pagos con tarjeta (Stripe)	40€	480
Tarifa API Google Maps	10€	120€
Servidor de Correo	5€	60€
		1260€

Los costes de diseño y desarrollo se han calculado para un despliegue inicial con la aplicación funcionando con un rendimiento alto. Estos costes podrían aumentar dependiendo de cómo evolucione el proyecto.

COSTES DE DISEÑO Y DESARROLLO (20€/hora, impuestos incluidos)

	Tiempo (horas)	Coste
Planificación y diseño	40	800€
Desarrollo de la aplicación	100	2000€
Despliegue	10	200€
Soporte y Mantenimiento	~ 10 Horas / mes	200€
		3200€

Por lo tanto, el coste estimado final será 1260€ + 3200€ = 4460€.

4. Contenido del Proyecto y Herramientas

4.1 Contenido

El proyecto se compone de los siguientes elementos:

- Backend (Parte servidor) – Desarrollado en Symfony 6 y API Platform.
- Frontend (Parte clientes de la tienda) - Desarrollado en Vue.js
- Admin Panel (Para administradores de la tienda) - Desarrollado en Vue.js
- Base de datos (MySQL y phpMyAdmin)

4.2 Herramientas

Las herramientas empleadas en el desarrollo son las siguientes:

4.2.1 Backend

El backend de la aplicación se desarrolla utilizando una combinación de tecnologías y herramientas que aseguran un sistema robusto, escalable y eficiente. A continuación, se detallan las principales tecnologías empleadas:

Symfony 6

Symfony es un framework PHP altamente flexible y escalable, utilizado para desarrollar aplicaciones web complejas. Symfony 6 ofrece una arquitectura sólida, un conjunto de componentes reutilizables y una comunidad activa que proporciona soporte continuo. En este proyecto, Symfony 6 se utiliza para gestionar la lógica de negocio, la seguridad, el enrutamiento y la comunicación con la base de datos.



PHP

PHP es un lenguaje de programación de código abierto ampliamente utilizado para el desarrollo web. Es la base sobre la que se construye Symfony. PHP permite desarrollar aplicaciones dinámicas e interactivas de manera eficiente. En este proyecto, PHP se utiliza para implementar la lógica de negocio, procesar las solicitudes del cliente y generar respuestas dinámicas.



Doctrine ORM

Doctrine ORM (Object-Relational Mapping) es una herramienta que permite mapear clases de PHP a tablas de bases de datos, facilitando así la interacción con la base de datos. Doctrine ORM se integra perfectamente con Symfony, proporcionando una forma sencilla y eficiente de gestionar la persistencia de datos. Con Doctrine, se puede realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de manera intuitiva y optimizada.



En la imagen siguiente se puede ver un ejemplo básico de como mapear clases fácilmente:

A screenshot of a code editor window showing a PHP file named Product.php. The code defines a class Product with annotations for Doctrine ORM. It includes annotations for the repository class, primary key, generated value, and column length. The code is as follows:

```
// src/Entity/Product.php
namespace App\Entity;

use App\Repository\ProductRepository;
use Doctrine\ORM\Mapping as ORM;

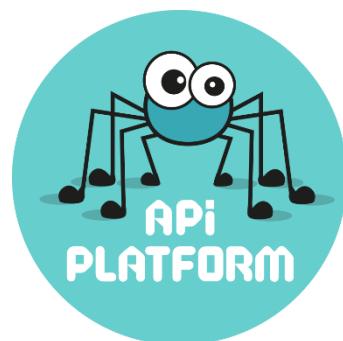
#[ORM\Entity(repositoryClass: ProductRepository::class)]
class Product
{
    #[ORM\Id]
    #[ORM\GeneratedValue]
    #[ORM\Column]
    private ?int $id = null;

    #[ORM\Column(length: 255)]
    private ?string $name = null;

    // ...
}
```

API Platform

API Platform es un framework utilizado para construir APIs RESTful de manera rápida y eficiente. Este framework se integra con Symfony y Doctrine, permitiendo exponer los recursos de la aplicación a través de APIs seguras y escalables. En este proyecto, API Platform se utiliza para desarrollar las APIs que permiten la comunicación entre el frontend y el backend, así como con otros servicios externos.



API de Stripe

Stripe es una plataforma de pagos en línea que permite realizar transacciones de manera segura. La API de Stripe se utiliza para gestionar los pagos en el eCommerce. Con esta API, se pueden crear, procesar y gestionar pagos de manera segura, incluyendo funciones como la gestión de suscripciones y reembolsos. La integración con Stripe permite ofrecer a los clientes una experiencia de pago rápida y confiable.



API de SendGrid

SendGrid es un servicio de correo electrónico que permite enviar correos electrónicos de manera fiable y a gran escala. La API de SendGrid se utiliza en el proyecto para gestionar el envío de correos electrónicos en diversos eventos, como la confirmación de pedidos, notificaciones de envío, y restablecimiento de contraseñas. La integración con SendGrid asegura que los correos electrónicos sean entregados de manera efectiva y profesional, mejorando la comunicación con los usuarios.



Estas tecnologías y herramientas se combinan para crear un backend robusto y eficiente, capaz de manejar las necesidades de un eCommerce moderno. Cada componente juega un papel crucial en la funcionalidad y el rendimiento del sistema, asegurando que los usuarios tengan una experiencia de compra fluida y segura.

4.2.2 Frontend

El frontend de la tienda consta de dos aplicaciones diferentes, ambas desarrolladas utilizando las mismas tecnologías. La primera aplicación está destinada a los clientes para realizar compras, mientras que la segunda aplicación es un panel de administración para los administradores de la tienda. A continuación, se detallan las tecnologías utilizadas en ambas aplicaciones.

Aplicación 1 – Parte cliente para las compras por los usuarios finales (Clientes de la tienda)

Aplicación 2 – Panel de Admin para los administradores de la tienda para el control total sobre la tienda

Tecnologías usadas:

Vue.js

Vue.js es un framework progresivo de JavaScript utilizado para construir interfaces de usuario. Vue.js facilita la creación de aplicaciones web interactivas y reactivas, permitiendo desarrollar componentes reutilizables y gestionar el estado de la aplicación de manera eficiente.



CSS

CSS (Cascading Style Sheets) se utiliza para dar estilo a los componentes de la aplicación. Permite definir la apariencia visual de la interfaz de usuario, incluyendo colores, fuentes, y layout.



Tailwind CSS¹

Tailwind CSS es un framework de utilidades CSS que permite construir interfaces de usuario rápidamente. Tailwind proporciona clases predefinidas que facilitan la creación de estilos personalizados sin necesidad de escribir CSS desde cero.



PostCSS y Autoprefixer

PostCSS es una herramienta para transformar estilos CSS utilizando plugins. Autoprefixer es un plugin de PostCSS que añade prefijos específicos del navegador a las reglas CSS, asegurando compatibilidad entre diferentes navegadores.



¹ Para evitar el conflicto entre las clases de Tailwind CSS y Bootstrap, se han personalizado las clases con el prefijo "tw- " (Ejemplo : La clase **mt-5** en la aplicación se prefijara como **tw-mt-5**).

Bootstrap 5

Bootstrap 5 es un framework de diseño web que facilita la creación de interfaces de usuario responsivas. Bootstrap incluye un conjunto de componentes y utilidades CSS y JavaScript que aceleran el desarrollo de la interfaz de usuario.



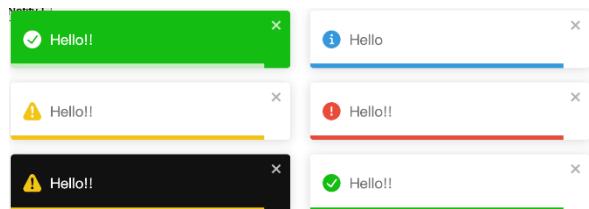
Vue Router

Vue Router es una biblioteca oficial de enrutamiento para Vue.js. Permite definir rutas y gestionar la navegación dentro de la aplicación, facilitando la creación de aplicaciones de una sola página (SPA).



Toastify

Toastify es una biblioteca que permite mostrar notificaciones en la aplicación. Se utiliza para informar a los usuarios sobre diversas acciones, como confirmaciones de pedidos y errores.



JavaScript

JavaScript es el lenguaje de programación utilizado para desarrollar la lógica de la aplicación. En combinación con Vue.js, JavaScript permite crear aplicaciones interactivas y dinámicas.



Google Maps API - para autocompletar el formulario de pedidos.

La API de Google Maps se utiliza para implementar la funcionalidad de autocompletar en campos de direcciones. Esto mejora la experiencia del usuario al proporcionar sugerencias de direcciones a medida que el usuario escribe.



FontAwesome

FontAwesome es una biblioteca de íconos que proporciona una amplia variedad de íconos escalables y personalizables. Se utiliza para mejorar la interfaz de usuario con iconos visualmente atractivos.



Bootstrap Icons y FontAwesome Icons

Bootstrap Icons es una biblioteca de íconos integrada en Bootstrap. Se utiliza junto con FontAwesome Icons para ofrecer una gama completa de iconos en la interfaz de usuario.



Vee Validate para la validación de formularios

Vee Validate es una biblioteca de validación de formularios para Vue.js. Permite validar campos de formulario de manera eficiente y proporcionar retroalimentación en tiempo real a los usuarios.



Axios

Axios es una biblioteca de JavaScript utilizada para realizar solicitudes HTTP. Se utiliza para interactuar con las APIs del backend, enviar y recibir datos, y gestionar la comunicación con el servidor.



JsBarcode

JsBarcode es una biblioteca de JavaScript que genera códigos de barras. Se utiliza en la aplicación para mostrar códigos de barras en los detalles de los pedidos, facilitando la identificación y el seguimiento de los productos.

JsBarcode



Vue Password Meter.

Vue Password Meter es un componente de Vue.js que muestra la fuerza de la contraseña ingresada por el usuario. Proporciona retroalimentación visual sobre la seguridad de la contraseña. Se usa en la pagina de registro.



Vuestic UI Components Library

Vuestic UI es una biblioteca de componentes de interfaz de usuario para Vue.js. Proporciona una serie de componentes predefinidos que aceleran el desarrollo de la interfaz de usuario. Se usan varios componentes en la aplicación que proporciona esta librería (tablas, dropdowns, Avatares etc.).



Vuex

Vuex es una biblioteca de gestión del estado para Vue.js. Permite centralizar el estado de la aplicación, facilitando la gestión y sincronización de datos entre componentes.



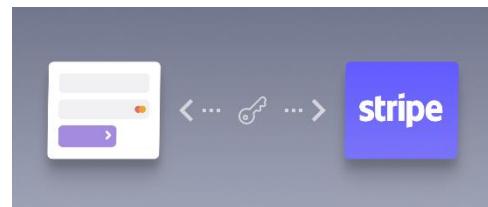
ESLint

ESLint es una herramienta de análisis de código estático para JavaScript. Se utiliza para identificar y corregir problemas en el código, asegurando que siga las mejores prácticas y estándares de codificación.



Stripe API - La pasarela de pagos

La API de Stripe se utiliza para gestionar los pagos en el frontend. Permite generar pagos de manera segura y eficiente, proporcionando una experiencia de compra fluida para los usuarios.



ApexCharts – Se usa en el panel de administración para diagramas y visualización de gráficos.

ApexCharts es una biblioteca de gráficos utilizada para visualizar datos en el panel de administración. Proporciona gráficos interactivos y personalizables que ayudan a los administradores a analizar datos y tomar decisiones informadas.



Moment.js

Moment.js es una biblioteca de JavaScript utilizada para manipular y formatear fechas y horas. Se utiliza para gestionar el tiempo en la aplicación, facilitando la visualización y el manejo de datos temporales.



Estas tecnologías se combinan para crear un frontend eficiente y atractivo, proporcionando una experiencia de usuario óptima tanto para los clientes como para los administradores de la tienda.

4.2.3 Base de Datos

La gestión de la base de datos en este proyecto se realiza utilizando las siguientes herramientas y tecnologías:

phpMyAdmin

phpMyAdmin es una herramienta web que facilita la administración de bases de datos MySQL y MariaDB. Proporciona una interfaz gráfica de usuario intuitiva que permite realizar diversas tareas de administración de manera sencilla. Sus características principales incluyen:

- Interfaz Gráfica de Usuario: Facilita la interacción con la base de datos mediante una interfaz visual amigable.
- Administración de Bases de Datos: Permite crear, modificar y eliminar bases de datos, tablas y columnas.
- Importación y Exportación de Datos: Facilita la importación y exportación de bases de datos en varios formatos, como SQL y XML.
- Mantenimiento de la Base de Datos: Permite ejecutar tareas de mantenimiento como reparar tablas y realizar copias de seguridad.
- Generación de Diagramas: Ofrece la posibilidad de crear esquemas de bases de datos para visualizar las relaciones entre tablas.



MySQL

MySQL es un sistema de gestión de bases de datos relacional (RDBMS) que se utiliza para almacenar y gestionar los datos de la aplicación. Es conocido por su fiabilidad, escalabilidad y rendimiento. MySQL permite realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y es compatible con una amplia variedad de aplicaciones y entornos de desarrollo.



Doctrine ORM

Anteriormente mencionado en el punto 4.2.1 (Backend).

4.2.4 Otras herramientas utilizadas

Además de las tecnologías principales para el desarrollo del backend y frontend, se han utilizado diversas herramientas adicionales que han facilitado el proceso de desarrollo y asegurado la calidad del proyecto. Estas herramientas incluyen:

Git y GitHub

Git es un sistema de control de versiones distribuido que permite realizar un seguimiento de los cambios en el código fuente a lo largo del tiempo. GitHub es una plataforma basada en la web que utiliza Git para gestionar y almacenar repositorios de código. Estas herramientas proporcionan varias ventajas:

- **Control de versiones:** Permiten mantener un historial detallado de los cambios en el código, facilitando la colaboración entre desarrolladores y la gestión de versiones del proyecto.
- **Colaboración:** Facilitan la colaboración entre varios desarrolladores mediante ramas, fusiones y solicitudes de extracción (pull requests).
- **Respaldo:** Ofrecen un almacenamiento seguro y centralizado del código fuente, proporcionando una copia de seguridad confiable.



Postman

Postman es una herramienta de desarrollo de API que facilita el diseño, prueba y documentación de APIs. Se ha utilizado para interactuar con las APIs RESTful del backend de manera eficiente. Las características principales de Postman incluyen:

- **Pruebas de API:** Permite enviar solicitudes HTTP a las APIs, verificar respuestas y realizar pruebas automatizadas.



Figma

Figma es una herramienta de diseño basada en la web que permite a los diseñadores crear interfaces de usuario de manera colaborativa. En este proyecto, Figma se ha utilizado para diseñar las interfaces del frontend y del panel de administración. Sus características principales incluyen:

- Diseño colaborativo: Permite a varios diseñadores trabajar simultáneamente en el mismo archivo, facilitando la colaboración en tiempo real.
- Prototipos interactivos: Permite crear prototipos interactivos que simulan la funcionalidad de la aplicación, ayudando a visualizar y validar el diseño antes de la implementación.
- Componentes reutilizables: Facilita la creación de componentes reutilizables, asegurando la consistencia del diseño en toda la aplicación.



El uso de estas herramientas ha sido fundamental para el desarrollo eficiente del proyecto, asegurando que se mantengan altos estándares de calidad y que los diseños y funcionalidades se implementen de manera efectiva.

Docker

Docker es una plataforma de código abierto que permite a los desarrolladores crear, desplegar, ejecutar y gestionar contenedores, que son componentes estandarizados y ejecutables que combinan el código fuente de aplicación con las dependencias y las bibliotecas del sistema operativo (SO) necesarias para ejecutar dicho código en cualquier entorno.

La aplicación está preparada para ser lanzada en Docker. Utiliza Docker Compose para lanzar tres contenedores:

- **Contenedor 1:** PHP-Symfony
- **Contenedor 2:** Servidor Nginx
- **Contenedor 3:** MySQL

La estructura de los archivos y configuración de Docker es la siguiente:

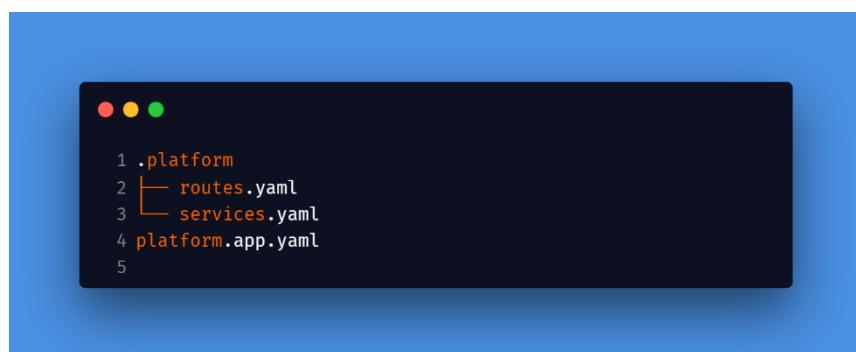
```

1 .docker/
2   -- docker-compose.yml
3   -- nginx
4     -- Dockerfile
5     -- nginx.conf
6     -- templates
7       -- default.conf.template
8   -- php
9     -- Dockerfile
10    -- php.ini
11
12
  
```

Platform.sh

Platform.sh es una plataforma de alojamiento como servicio (PaaS) que está diseñada para facilitar el desarrollo, la implementación y la gestión de aplicaciones web y servicios en la nube. Proporciona un entorno gestionado y automatizado para alojar aplicaciones, con un enfoque particular en la escalabilidad, la colaboración en equipo y la gestión eficiente del ciclo de vida del software.

La aplicación también está preparada para despliegue en Platform.sh, la plataforma nativa de Symfony para el despliegue en la nube. Toda la configuración está en la raíz del proyecto en la carpeta **./platform**:



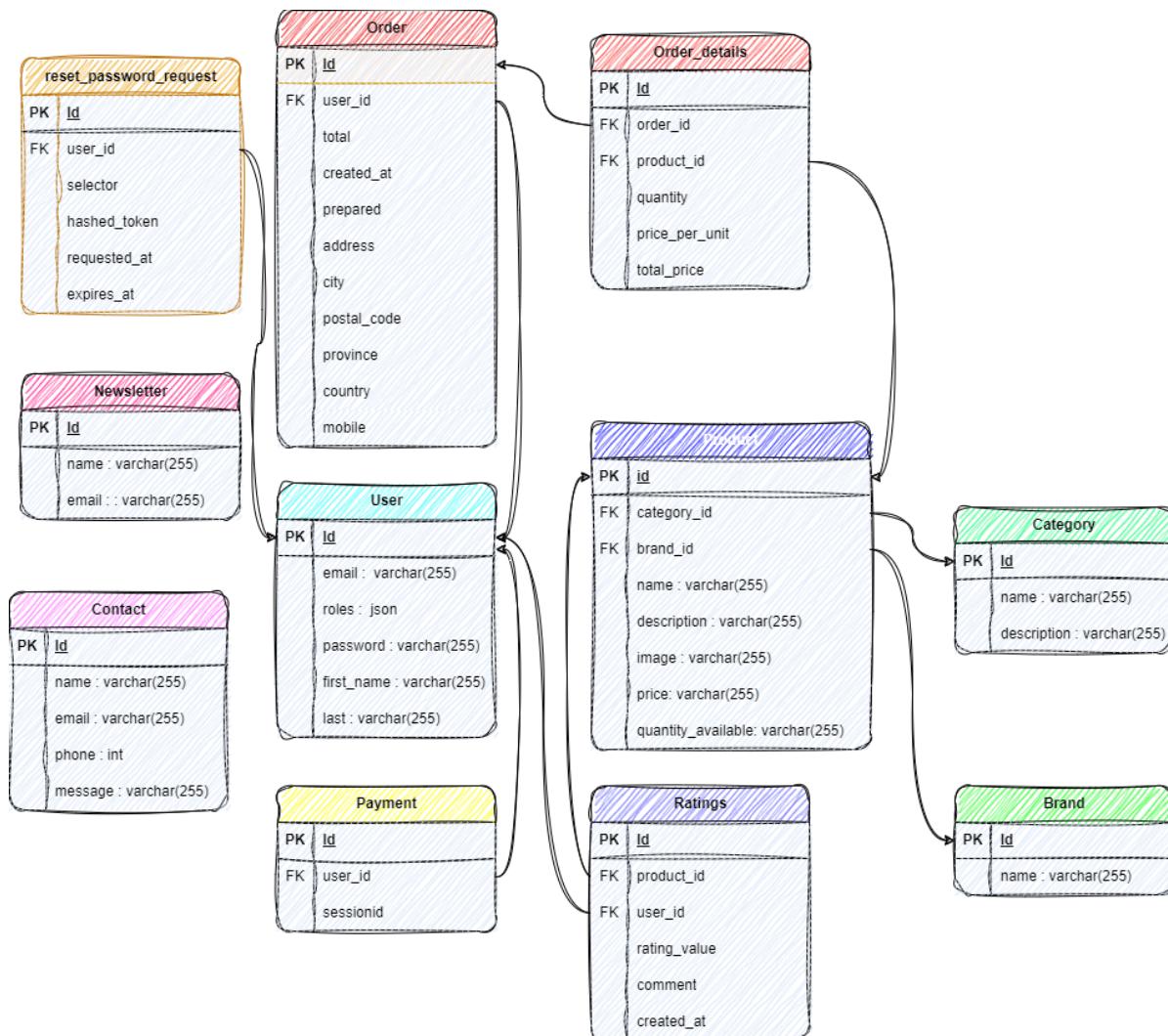
```
1 .platform
2   |- routes.yaml
3   |- services.yaml
4 platform.app.yaml
5
```

5. Estructura del proyecto

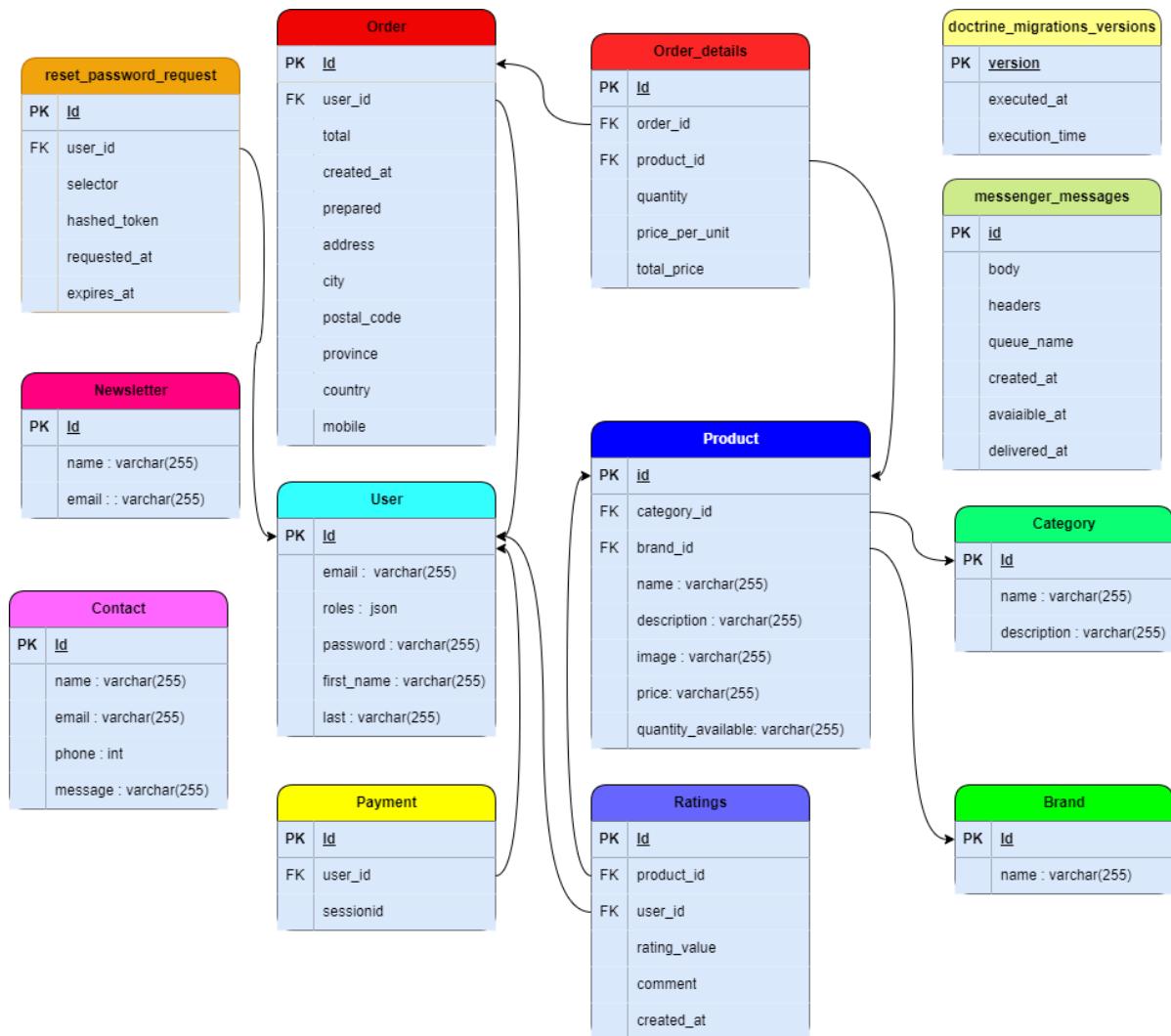
5.1 Estructura de base de datos

Antes de comenzar a programar, el primer paso fue diseñar la estructura de la base de datos. Este diseño es crucial, ya que define cómo se almacenarán y relacionarán los datos en el sistema. Se ha creado un diagrama entidad-relación (ER) que ilustra las tablas, campos y relaciones entre ellas, asegurando una organización eficiente y lógica de los datos.

Este es el boceto de la diagrama:



Y la estructura final de la base de datos seria lo siguiente incluyendo la tabla de migraciones de doctrine y la tabla especifica de messenger que guarda los datos sobre las notificaciones y correo electronicos enviados desde el mailer.



Descripción de las entidades principales:

1. User (Usuario)

- Campos:**
 - id**: Identificador único del usuario.
 - email**: Correo electrónico del usuario.
 - roles**: Roles asignados al usuario, almacenados en formato JSON.
 - password**: Contraseña del usuario.
 - first_name**: Nombre del usuario.
 - last_name**: Apellido del usuario.
- Relaciones:** Un usuario puede realizar múltiples pedidos (**Order**).

2. Product (Producto)

- **Campos:**
 - **id:** Identificador único del producto.
 - **category_id:** Identificador de la categoría a la que pertenece el producto.
 - **brand_id:** Identificador de la marca del producto.
 - **name:** Nombre del producto.
 - **description:** Descripción del producto.
 - **image:** URL de la imagen del producto.
 - **price:** Precio del producto.
 - **quantity_available:** Cantidad disponible del producto.
- **Relaciones:** Un producto pertenece a una categoría (**Category**) y a una marca (**Brand**). Además, un producto puede tener múltiples valoraciones (**Ratings**).

3. Category (Categoría)

- **Campos:**
 - **id:** Identificador único de la categoría.
 - **name:** Nombre de la categoría.
 - **description:** Descripción de la categoría.
- **Relaciones:** Una categoría puede tener múltiples productos (**Product**).

4. Brand (Marca)

- **Campos:**
 - **id:** Identificador único de la marca.
 - **name:** Nombre de la marca.
- **Relaciones:** Una marca puede tener múltiples productos (**Product**).

5. Order (Pedido)

- **Campos:**
 - **id:** Identificador único del pedido.
 - **user_id:** Identificador del usuario que realizó el pedido.

- **total:** Total del pedido.
- **created_at:** Fecha de creación del pedido.
- **prepared:** Estado de preparación del pedido.
- **address:** Dirección de envío.
- **city:** Ciudad de envío.
- **postal_code:** Código postal de envío.
- **province:** Provincia de envío.
- **country:** País de envío.
- **mobile:** Teléfono móvil de contacto.
- **Relaciones:** Un pedido puede contener múltiples detalles de pedido (**Order_details**).

6. Order_details (Detalles del Pedido)

- **Campos:**
 - **id:** Identificador único del detalle del pedido.
 - **order_id:** Identificador del pedido al que pertenece el detalle.
 - **product_id:** Identificador del producto en el detalle.
 - **quantity:** Cantidad del producto en el detalle.
 - **price_per_unit:** Precio por unidad del producto.
 - **total_price:** Precio total del producto en el detalle.
- **Relaciones:** Un detalle de pedido pertenece a un pedido (**Order**) y a un producto (**Product**).

7. Payment (Pago)

- **Campos:**
 - **id:** Identificador único del pago.
 - **user_id:** Identificador del usuario que realizó el pago.
 - **sessionid:** Identificador de la sesión de pago.
- **Relaciones:** Un pago está asociado a un usuario (**User**).

8. Ratings (Valoraciones)

- **Campos:**

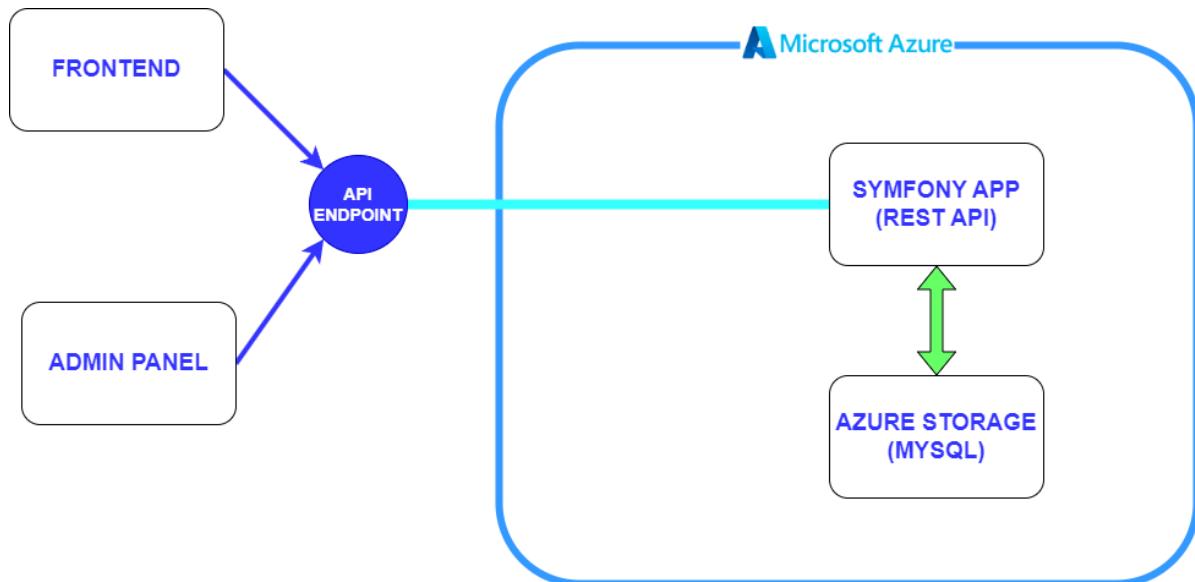
- **id**: Identificador único de la valoración.
- **product_id**: Identificador del producto valorado.
- **user_id**: Identificador del usuario que realizó la valoración.
- **rating_value**: Valor de la valoración.
- **comment**: Comentario de la valoración.
- **created_at**: Fecha de creación de la valoración.
- **Relaciones**: Una valoración pertenece a un producto (**Product**) y a un usuario (**User**).

9. Otros elementos:

- **reset_password_request** (**Solicitudes de restablecimiento de contraseña**)
- **newsletter** (**Boletín informativo**)
- **contact** (**Contacto**)
- **doctrine_migrations_versions** (**Versiones de migraciones de Doctrine**)
- **messenger_messages** (**Mensajes de Messenger**)

Estas tablas y sus relaciones permiten gestionar de manera eficiente los datos necesarios para el funcionamiento de la tienda ecommerce, asegurando integridad y consistencia en las operaciones realizadas sobre la base de datos.

5.2 Estructura de proyecto



La imagen ilustra la estructura de funcionamiento de la aplicación desplegada en máquina virtual de Microsoft Azure, mostrando cómo interactúan los diferentes componentes del sistema.

1. Microsoft Azure VM:

- Toda la infraestructura de la aplicación está alojada en una máquina virtual en Azure.

2. Frontend Clientes:

- Esta es la aplicación que utilizan los usuarios finales para realizar compras en la tienda online.

3. Frontend Admin:

- Este es el panel de administración utilizado por los administradores de la tienda para gestionar productos, pedidos y otras configuraciones.

4. API Endpoint:

- Punto de acceso donde tanto el Frontend Clientes como el Frontend Admin realizan peticiones. Actúa como intermediario entre los frontends y el backend.

5. Symfony App (REST API):

- La aplicación backend desarrollada en Symfony que expone una API RESTful. Esta aplicación recibe las peticiones de los frontend y las procesa.

6. Azure Storage (MySQL):

- La base de datos MySQL almacenada en Azure. La API de Symfony realiza consultas y actualizaciones en esta base de datos para gestionar la información de la tienda.

6. Creación de Base de datos.

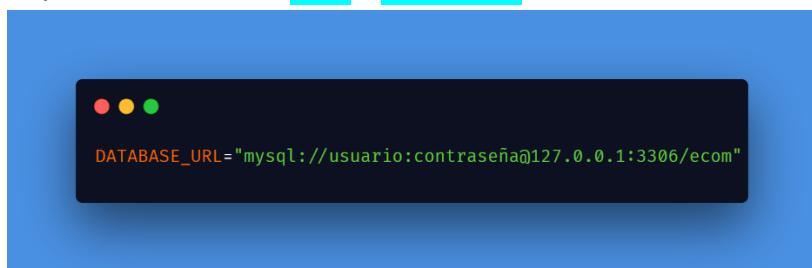
La base de datos de nuestra aplicación, llamada “**ecom**”, ha sido creada utilizando la CLI (Command Line Interface) de Symfony. A continuación, se detallan los pasos necesarios para su creación y configuración, explicando cada uno de ellos.

Paso 1: Configuración del Entorno

Antes de empezar, es crucial asegurarse de que Symfony y las herramientas necesarias estén correctamente instaladas y configuradas en el entorno de desarrollo. Además, es importante verificar que se cuenta con un servidor de base de datos MySQL operativo.

Paso 2: Configuración del Archivo .env

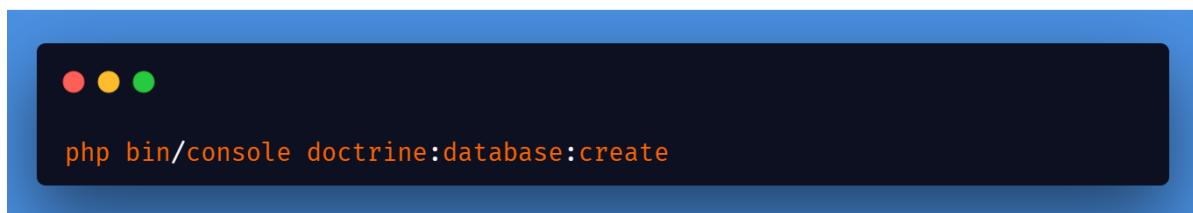
Symfony utiliza el archivo **.env** para configurar las variables de entorno. Es necesario especificar la cadena de conexión a la base de datos MySQL en este archivo. Primero que todo para mantener los secretos y variables de entornos en nuestro entorno local, copiamos el archivo **.env** a **.env.local**.



```
DATABASE_URL="mysql://usuario:contraseña@127.0.0.1:3306/ecom"
```

Paso 3: Crear la Base de Datos

Con la configuración en el archivo **.env** lista, el siguiente paso es crear la base de datos. Esto se puede hacer utilizando el comando Symfony CLI:



```
php bin/console doctrine:database:create
```

Paso 4: Crear las Entidades

Creamos todas las entidades de la aplicación. La manera más fácil es con symfony CLI usando el comando: **symfony console make:entity**
Estas son las entidades de la aplicación :

```
Brand.php  
Category.php  
Contact.php  
Newsletter.php  
Order.php  
OrderDetails.php  
Payment.php  
Product.php  
Ratings.php  
ResetPasswordRequest.php  
User.php
```

Todos los detalles, de cada entidad están explicadas en el [punto 5.1](#) del documento.

Paso 5: Generar las Migraciones

Una vez definidas las entidades, se deben generar las migraciones para crear las tablas correspondientes en la base de datos. Esto se hace con el siguiente comando:

```
php bin/console make:migration
```

Este comando analiza las entidades y crea un archivo de migración que describe las operaciones SQL necesarias para actualizar el esquema de la base de datos.

Paso 6: Ejecutar las Migraciones

Para aplicar las migraciones y crear las tablas en la base de datos, se utiliza el siguiente comando:

```
php bin/console doctrine:migrations:migrate
```

Este comando ejecuta las migraciones y actualiza la base de datos “ecom” con las nuevas tablas y campos definidos en las entidades.

Paso 7: Comprobar la Migración

Entramos en **phpMyAdmin** y comprobamos la migración y podemos ver se ha creado correctamente la base de datos:

The screenshot shows the phpMyAdmin interface for the 'newecom' database. The top navigation bar includes tabs for Structure, SQL, Search, Query, Export, Import, Operations, Privileges, Routines, Events, Triggers, and a Help icon. Below the navigation is a 'Filters' section with a search input field. The main content area displays a table of 13 database tables, each with a checkbox, a star icon, and various action buttons like Browse, Structure, Search, Insert, Empty, Drop, etc. The table columns include: Table, Action, Rows, Type, Collation, Size, and Overhead. The total summary at the bottom indicates 13 tables, 177 rows, InnoDB type, utf8mb4_unicode_ci collation, 560.0 Kib size, and 0 B overhead.

Table	Action	Rows	Type	Collation	Size	Overhead
brand	Browse Structure Search Insert Empty Drop	5	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
category	Browse Structure Search Insert Empty Drop	10	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
contact	Browse Structure Search Insert Empty Drop	6	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
doctrine_migration_versions	Browse Structure Search Insert Empty Drop	19	InnoDB	utf8mb3_unicode_ci	16.0 Kib	-
messenger_messages	Browse Structure Search Insert Empty Drop	32	InnoDB	utf8mb4_unicode_ci	208.0 Kib	-
newsletter	Browse Structure Search Insert Empty Drop	8	InnoDB	utf8mb4_unicode_ci	16.0 Kib	-
order	Browse Structure Search Insert Empty Drop	11	InnoDB	utf8mb4_unicode_ci	32.0 Kib	-
order_details	Browse Structure Search Insert Empty Drop	26	InnoDB	utf8mb4_unicode_ci	48.0 Kib	-
payment	Browse Structure Search Insert Empty Drop	14	InnoDB	utf8mb4_unicode_ci	32.0 Kib	-
product	Browse Structure Search Insert Empty Drop	20	InnoDB	utf8mb4_unicode_ci	48.0 Kib	-
ratings	Browse Structure Search Insert Empty Drop	1	InnoDB	utf8mb4_unicode_ci	48.0 Kib	-
reset_password_request	Browse Structure Search Insert Empty Drop	0	InnoDB	utf8mb4_unicode_ci	32.0 Kib	-
user	Browse Structure Search Insert Empty Drop	25	InnoDB	utf8mb4_unicode_ci	32.0 Kib	-
13 tables	Sum	177	InnoDB	utf8mb4_0900_ai_ci	560.0 Kib	0 B

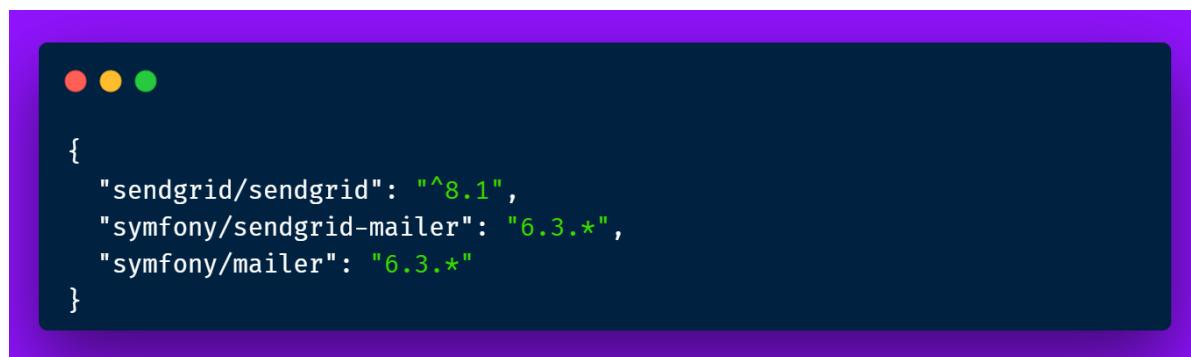
7. Creación de Servidor de Correos

Para gestionar el envío de correos electrónicos en la aplicación, se ha implementado el servidor de correos utilizando SendGrid junto con el Symfony Mailer Bundle. Este enfoque ha sido seleccionado después de evaluar varias opciones, incluyendo Google Mailer y Mailjet, y determinar que SendGrid proporciona la mejor combinación de rapidez y optimización.

Configuración del Servidor de Correos con SendGrid

1. Instalación de Dependencias:

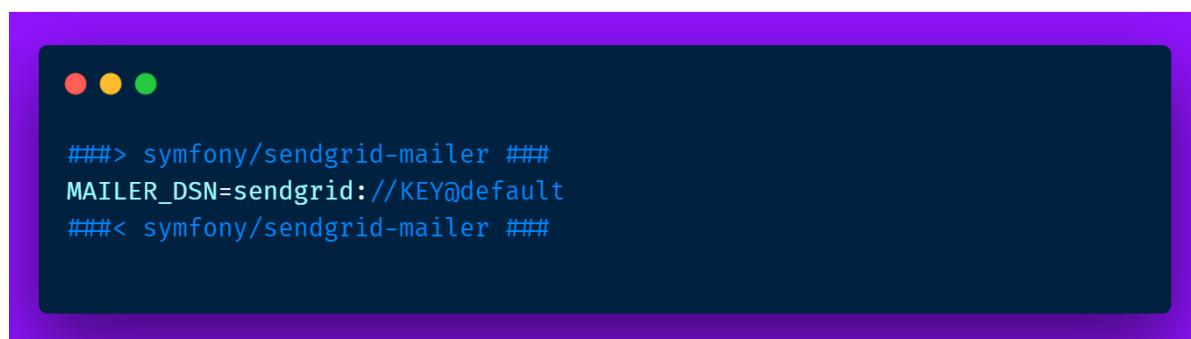
- Se han instalado las siguientes dependencias mediante Composer



```
{  
    "sendgrid/sendgrid": "^8.1",  
    "symfony/sendgrid-mailer": "6.3.*",  
    "symfony/mailer": "6.3.*"  
}
```

2. Configuración en el archivo .env:

- Se ha configurado el Data Source Name (DSN) de SendGrid en el archivo **.env** de Symfony para integrar SendGrid con Symfony Mailer:



```
###> symfony/sendgrid-mailer ###  
MAILER_DSN=sendgrid://KEY@default  
###< symfony/sendgrid-mailer ###
```

De esta forma ya está configurado el servidor para enviar correos.

3. Comparación con Otros Proveedores:

- Google Mailer: Aunque es fiable y ampliamente utilizado, Google Mailer no ofreció la misma rapidez en la entrega de correos como SendGrid.
- Mailjet: Mailjet también es una opción popular, pero en las pruebas realizadas, SendGrid demostró ser óptimo en términos de velocidad y eficiencia.

El uso de SendGrid para el envío de correos electrónicos asegura una gestión eficiente y rápida de las comunicaciones por correo de la aplicación, abarcando desde confirmaciones de pedidos, estado de los pedidos hasta notificaciones y restablecimientos de contraseñas.

8. Autenticación

La autenticación en la aplicación se basa en JSON Web Tokens (JWT) para asegurar que solo los usuarios autenticados puedan acceder a recursos protegidos. Se ha utilizado el bundle **lexik/jwt-authentication-bundle** para gestionar la autenticación JWT en Symfony.

8.1 Configuración del Bundle JWT

Instalación de Dependencias:

- El paquete **lexik/jwt-authentication-bundle** se ha añadido al proyecto para facilitar la gestión de la autenticación basada en tokens JWT:

```
● ● ●
{
    "lexik/jwt-authentication-bundle": "^2.20"
}
```

Configuración en el archivo .env:

- En el archivo **.env**, se han configurado las claves secretas y públicas necesarias para firmar y verificar los tokens JWT:

```
● ● ●
##> lexik_jwt_authentication ###
JWT_SECRET_KEY=%kernel.project_dir%/config/jwt/private.pem
JWT_PUBLIC_KEY=%kernel.project_dir%/config/jwt/public.pem
JWT_PASSPHRASE=el_passphrase
##< lexik_jwt_authentication ###
```

Configuración de Seguridad:

- La configuración de seguridad en **config/packages/security.yaml** asegura que solo los usuarios autenticados puedan acceder a ciertas rutas y define cómo se maneja la autenticación JWT:

```
security:
    password_hashers:

        Symfony\Component\Security\Core\User\PasswordAuthenticatedUserInterface
: 'auto'
    providers:
        app_user_provider:
            entity:
                class: App\Entity\User
                property: email
    firewalls:
        dev:
            pattern: ^/(_(profiler|wdt)|css|images|js)/
            security: false
        api:
            pattern: ^/api/
            stateless: true
            provider: app_user_provider
            jwt: ~
        main:
            json_login:
                check_path: /auth
                username_path: email
                password_path: password
                success_handler:
                    lexik_jwt_authentication.handler.authentication_success
                    failure_handler:
                        lexik_jwt_authentication.handler.authentication_failure

            access_control:
                - { path: ^/reset-password.*$, roles: PUBLIC_ACCESS }
                - { path: ^/user/\d+/orders, roles: PUBLIC_ACCESS }
                - { path: ^/profile, roles: PUBLIC_ACCESS }
                - { path: ^/stripe/create-checkout-session, roles: PUBLIC_ACCESS }
}
            - { path: '^/api/newsletters', roles: PUBLIC_ACCESS }
            - { path: '^/stripe.*$', roles: PUBLIC_ACCESS }
            - { path: '^/api$', roles: PUBLIC_ACCESS } # Allows accessing the
Swagger UI
            - { path: ^/auth, roles: PUBLIC_ACCESS }
            - { path: ^/api/users$, roles: PUBLIC_ACCESS }
            - { path: ^/, roles: IS_AUTHENTICATED_FULLY }
```

En este archivo protegemos cada ruta.

Para obtener el token podemos obtener desde **Postman** o en este caso lo hacemos desde terminal con CURL:

```
curl -X POST -H "Content-Type: application/json"
https://localhost/api/auth -d '{"username":"johndoe", "password":"test"}'
```

Y nos devuelve el token:

```
{
  "token" :
  "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXUYJ9.eyJleHAiOiE0MzQ3Mjc1MzYsInVzZXJuYW
  1lIjoia29ybGVvbISImLhdCI6IjE0MzQ2NDExMzYifQ.nh0L_wuJy6ZKIQWh6OrW5hdLkvi
  Ts1_bau2G...."
}
```

8.2 Mejorando la Experiencia del Usuario

La configuración básica solo devuelve el token JWT. Sin embargo, para mejorar la experiencia del usuario, se ha personalizado la respuesta de autenticación para incluir datos básicos del usuario como el ID, nombre de usuario y roles. Esto permite llenar automáticamente los datos del usuario en formularios o usar el ID de usuario en peticiones

Nota: Conseguir esto ha sido un desafío, ya que requirió mucha búsqueda de documentación y recursos.

Creación de Event Listener

Para lograr esta personalización, se ha creado un **Event Listener** llamado AuthenticationSuccessListener.php :

```
// src/EventListener/AuthenticationSuccessListener.php

namespace App\EventListener;

use Lexik\Bundle\JWTAuthenticationBundle\Event\AuthenticationSuccessEvent;
use Symfony\Component\Security\Core\User\UserInterface;

class AuthenticationSuccessListener
{
    public function onAuthenticationSuccessResponse(AuthenticationSuccessEvent $event)
    {
        $data = $event->getData();
        $user = $event->getUser();

        if (!$user instanceof UserInterface) {
            return;
        }

        // Obtener el ID del usuario
        $userId = $user->getId();
        $userName = $user->getFirstName();
        $userRole = $user->getRoles();

        // Modificar los datos de la respuesta para incluir el token y el ID del usuario
        $responseData = [
            'token' => $data['token'],
            'user_id' => $userId,
            'username' => $userName,
            'role' => $userRole
        ];

        // Establecer los datos modificados en la respuesta
        $event->setData($responseData);
    }
}
```

Este Event Listener modifica la respuesta de autenticación para incluir los datos básicos del usuario, mejorando así la experiencia del usuario final.

Ventajas de la Autenticación JWT

- **Seguridad:** Los tokens JWT son firmados con una clave privada, asegurando que no puedan ser falsificados.
- **Escalabilidad:** Los tokens JWT son independientes del estado, lo que permite que la autenticación funcione bien en aplicaciones distribuidas.
- **Simplicidad:** La autenticación basada en JWT es fácil de implementar y mantener, con tokens que se pueden incluir en las cabeceras HTTP de manera sencilla.

Ahora cuando hagamos peticiones para obtener el token, nos devuelve los siguientes datos:

```
{  
  "token": "jdsbfjvhb_skldbfbhjvbjabas-sakfvahjvabdkbjafs-slavsshafdvadfsbhjfadsbhjadfsbhjfdsvhjmndfsahjfvh",  
  "user_id": 22,  
  "user_name": "Abhijeet",  
  "roles": ["ROLE_USER", "ROLE_ADMIN"]  
}
```

Para hacer peticiones desde el Frontend, se pasa el token en los **Headers** de cada petición de la siguiente forma:

```
const token = localStorage.getItem('token');  
const response = await axios.get(url, {  
  headers: {  
    // Aquí pasamos el token en la header de la petición  
    Authorization: `Bearer ${token}`,  

```

9. Configuración de pasarela de pagos (API de Stripe)

Para integrar una pasarela de pagos en la aplicación, se ha utilizado la API de Stripe. Stripe es una de las plataformas de pago más populares y proporciona una manera sencilla y segura de manejar transacciones en línea.

9.1 Instalación de Stripe

1. Instalación del Paquete:

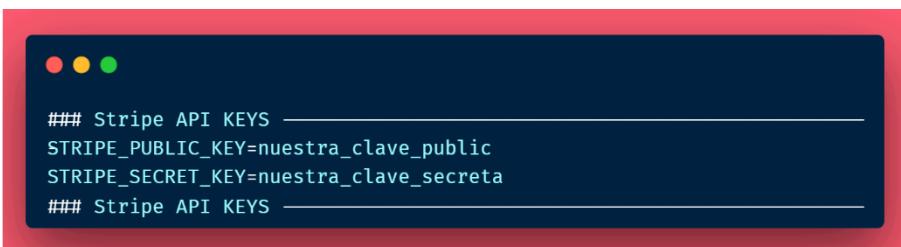
- Se ha añadido el paquete **stripe/stripe-php** a las dependencias del proyecto:



```
{  
    "stripe/stripe-php": "^13.15"  
}
```

2. Registro y Obtención de Claves de API:

- Se debe crear una cuenta en Stripe y obtener las claves de API (clave pública y clave secreta). Estas claves se configuran en el archivo **.env**:

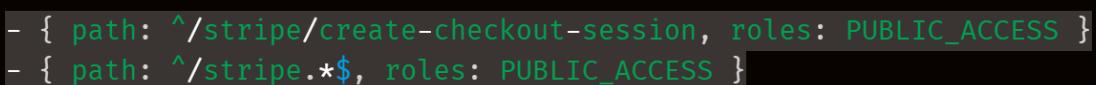


```
### Stripe API KEYS  
STRIPE_PUBLIC_KEY=nuestra_clave_public  
STRIPE_SECRET_KEY=nuestra_clave_secreta  
### Stripe API KEYS
```

9.2 Creación del Controlador de Stripe

El controlador de Stripe maneja la lógica para crear cargos y sesiones de pago. A continuación, se presenta el código del controlador que expone una API para procesar pagos.

Estos son las rutas que expone la API para generar la sesión de pago:



```
- { path: '^/stripe/create-checkout-session', roles: PUBLIC_ACCESS }  
- { path: '^/stripe.*$', roles: PUBLIC_ACCESS }
```

Este es el controlador:

```

// src/Controller/StripeController.php

namespace App\Controller;

use Stripe\Stripe;
use StripeCheckout\Session;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Annotation\Route;

class StripeController extends AbstractController
{
    #[Route('/stripe', name: 'app_stripe')]
    public function index(): Response
    {
        return $this->render('stripe/index.html.twig', [
            'stripe_key' => $_ENV["STRIPE_PUBLIC_KEY"],
        ]);
    }

    #[Route('/stripe/create-charge', name: 'app_stripe_charge', methods: ['POST'])]
    public function createCharge(Request $request)
    {
        Stripe::setApiKey($_ENV["STRIPE_SECRET_KEY"]);
        StripeCharge::create([
            "amount" => 5 * 100, // El monto en centavos
            "currency" => "eur",
            "source" => $request->request->get('stripeToken'),
            "description" => "Pago de prueba"
        ]);
        $this->addFlash('success', '¡Pago exitoso!');
        return $this->redirectToRoute('app_stripe', [], Response::HTTP_SEE_OTHER);
    }

    #[Route('/stripe/create-checkout-session', name: 'app_stripe_create_checkout_session', methods: ['POST'])]
    public function createCheckoutSession(Request $request): JsonResponse
    {
        $data = json_decode($request->getContent(), true);
        $totalAmount = $data['amount_total'];
        $userName = $data['created'];
        $successUrl = $data['successUrl'];
        $cancelUrl = $data['cancelUrl'];

        Stripe::setApiKey($_ENV["STRIPE_SECRET_KEY"]);
        try {
            $session = Session::create([
                'payment_method_types' => ['card'],
                'line_items' => [
                    [
                        'price_data' => [
                            'currency' => 'eur',
                            'unit_amount' => $totalAmount * 100, // Convertir a centavos
                            'product_data' => [
                                'name' => 'Compra',
                            ],
                        ],
                        'quantity' => 1,
                    ],
                ],
                'mode' => 'payment',
                'success_url' => $successUrl,
                'cancel_url' => $cancelUrl,
                'metadata' => [
                    'user_name' => $userName,
                ],
            ]);
            return $this->json(['url' => $session->url]);
        } catch (\Exception $e) {
            return $this->json(['error' => $e->getMessage()], Response::HTTP_INTERNAL_SERVER_ERROR);
        }
    }
}

```

- La ruta **/stripe/create-checkout-session** crea una sesión de pago de Stripe y devuelve una URL para redirigir al usuario a la página de pago de Stripe.

Si accedemos al URL que devuelve, este es el interfaz de pagos:

The screenshot shows a payment interface for StakNine. On the left, there's a summary table with the following details:

Pay StakNine	
\$39.99	
 Phoenix Deployhandbook Test	\$39.99
How to deploy Phoenix app...	
Subtotal	\$39.99
Tax ⓘ	Enter address to calculate
Total due	\$39.99

On the right, there's a "Pay with card" form with fields for Email, Debit/Credit Card information (with placeholder card number 1234 1234 1234 1234 and logos for VISA, MasterCard, American Express, and Discover), Name on card, Billing address (set to United States), and a checkbox for "Save my info for secure 1-click checkout". A large blue "Pay" button is at the bottom.

9.2 Ventajas de Usar Stripe

- Seguridad: Stripe maneja de manera segura los detalles de la tarjeta de crédito, reduciendo el riesgo de fraudes.
- Facilidad de Integración: Proporciona una API sencilla y bien documentada.
- Experiencia de Usuario: La página de pago de Stripe está optimizada para conversiones y proporciona una experiencia de usuario fluida.

Esta configuración garantiza que los usuarios puedan realizar pagos de manera segura y eficiente a través de la plataforma de Stripe, mejorando así la funcionalidad de comercio electrónico de la aplicación.

10. Estructura final del Backend

Después de todo el proceso, la estructura final del Backend sería el siguiente:

```
@iamabhijeet2003 ~/workspaces/backend-proyecto-final (main) $ tree -I 'vendor|var|bundles|migrations'  
.  
├── assets  
│   ├── app.js  
│   ├── bootstrap.js  
│   ├── controllers  
│   │   └── hello_controller.js  
│   ├── controllers.json  
│   └── styles  
│       └── app.css  
├── bin  
│   ├── console  
│   └── phpunit  
├── composer.json  
└── composer.lock  
├── config  
│   ├── bundles.php  
│   ├── packages  
│   │   ├── api_platform.yaml  
│   │   ├── asset_mapper.yaml  
│   │   ├── cache.yaml  
│   │   ├── debug.yaml  
│   │   ├── doctrine.yaml  
│   │   ├── doctrine_migrations.yaml  
│   │   ├── framework.yaml  
│   │   ├── lein_jwt_authentication.yaml  
│   │   ├── mailer.yaml  
│   │   ├── messenger.yaml  
│   │   ├── monolog.yaml  
│   │   ├── nelmio_cors.yaml  
│   │   ├── notifier.yaml  
│   │   ├── reset_password.yaml  
│   │   ├── routing.yaml  
│   │   ├── security.yaml  
│   │   ├── translation.yaml  
│   │   └── twig.yaml  
│   ├── validator.yaml  
│   └── web_profiler.yaml  
├── preload.php  
├── routes  
│   ├── api_platform.yaml  
│   ├── framework.yaml  
│   └── web_profiler.yaml  
├── routes.yaml  
├── services.yaml  
├── importmap.php  
├── newecom.sql  
├── php.info  
├── phpunit.xml.dist  
├── public  
│   └── index.php  
└── src  
    ├── ApiResource  
    ├── Controller  
    │   ├── MailerController.php  
    │   ├── ProfileController.php  
    │   ├── ResetPasswordController.php  
    │   ├── StripeController.php  
    │   └── UserOrdersController.php  
    ├── Entity  
    │   ├── Brand.php  
    │   ├── Category.php  
    │   ├── Contact.php  
    │   ├── Newsletter.php  
    │   ├── Order.php  
    │   ├── OrderDetails.php  
    │   ├── Payment.php  
    │   ├── Product.php  
    │   ├── Ratings.php  
    │   └── ResetPasswordRequest.php  
    ├── User.php  
    ├── EventListener  
    │   ├── AuthenticationSuccessListener.php  
    │   └── JWTCreatedListener.php  
    ├── EventSubscriber  
    │   ├── ContactFormSubscriber.php  
    │   ├── LoginMailSubscriber.php  
    │   ├── NewsletterSubscriber.php  
    │   ├── OrderMailSubscriber.php  
    │   ├── OrderStatusSubscriber.php  
    │   └── WelcomeEmailsSubscriber.php  
    ├── Form  
    │   ├── ChangePasswordFormType.php  
    │   └── ResetPasswordRequestFormType.php  
    ├── Kernel.php  
    ├── Repository  
    │   ├── BrandRepository.php  
    │   ├── CategoryRepository.php  
    │   ├── ContactRepository.php  
    │   ├── NewsletterRepository.php  
    │   ├── OrderDetailRepository.php  
    │   ├── OrderRepository.php  
    │   ├── PaymentRepository.php  
    │   ├── ProductRepository.php  
    │   ├── RatingRepository.php  
    │   └── ResetPasswordRequestRepository.php  
    ├── State  
    │   └── UserPasswordHasher.php  
    └── symfony.lock  
    ├── templates  
    │   ├── base.html.twig  
    │   ├── emails  
    │   │   └── OrderConfirmation.html.twig  
    │   ├── order  
    │   │   └── index.html.twig  
    │   ├── reset_password  
    │   │   ├── check_email.html.twig  
    │   │   ├── email.html.twig  
    │   │   ├── request.html.twig  
    │   │   └── reset.html.twig  
    │   ├── stripe  
    │   │   └── index.html.twig  
    │   └── user_orders  
    │       └── index.html.twig  
    └── tests  
        └── bootstrap.php  
    └── translations
```

Iniciamos el servidor local de Symfony

```
abhijeet@abhijeet-VirtualBox:~/Escritorio/proj-v23/backend$ symfony server:start -d
INFO | A new Symfony CLI version is available (5.8.19, currently running 5.7.3).
If you installed the Symfony CLI via a package manager, updates are going to be automatic.
If not, upgrade by downloading the new version at https://github.com/symfony-cli/symfony-cli/releases
And replace the current binary (symfony) by the new one.

[WARNING] The local web server is already running

Local Web Server
  Listening on http://127.0.0.1:8000
    The Web server is using PHP CLI 8.1.2 (from Platform.sh: /home/abhijeet/Escritorio/proj-v23/backend/.platform.app.yaml)

Local Domains

Workers
  PID 11878: /usr/bin/php8.1 -S 127.0.0.1:40793 -d variables_order=EGPCS /home/abhijeet/.symfony5/php/4241a2fbcd23978b407a
o/proj-v23/backend/php.ini/)

Environment Variables
  None
abhijeet@abhijeet-VirtualBox:~/Escritorio/proj-v23/backend$ |
```

Por defecto, el servidor inicia en el puerto 8000.

Después de toda la configuración, este sería el panel del API, entramos en la ruta /api donde podemos ver todas las rutas disponibles:

Product		
GET	/api/products	Retrieves the collection of Product resources.
POST	/api/products	Creates a Product resource.
GET	/api/products/{id}	Retrieves a Product resource.
PUT	/api/products/{id}	Replaces the Product resource.
DELETE	/api/products/{id}	Removes the Product resource.
Ratings		
GET	/api/ratingss	Retrieves the collection of Ratings resources.
POST	/api/ratingss	Creates a Ratings resource.
GET	/api/ratingss/{id}	Retrieves a Ratings resource.
PUT	/api/ratingss/{id}	Replaces the Ratings resource.
DELETE	/api/ratingss/{id}	Removes the Ratings resource.
PATCH	/api/ratingss/{id}	Updates the Ratings resource.
User		
GET	/api/users	Retrieves the collection of User resources.
POST	/api/users	Creates a User resource.
GET	/api/users/{id}	Retrieves a User resource.
PUT	/api/users/{id}	Replaces the User resource.
Login Check		
POST	/auth	Creates a user token.

OrderDetails			
GET	/api/order_detailss	Retrieves the collection of OrderDetails resources.	🔒 ↴
POST	/api/order_detailss	Creates a OrderDetails resource.	🔒 ↴
GET	/api/order_detailss/{id}	Retrieves a OrderDetails resource.	🔒 ↴
PUT	/api/order_detailss/{id}	Replaces the OrderDetails resource.	🔒 ↴
DELETE	/api/order_detailss/{id}	Removes the OrderDetails resource.	🔒 ↴
PATCH	/api/order_detailss/{id}	Updates the OrderDetails resource.	🔒 ↴
Order			
GET	/api/orders	Retrieves the collection of Order resources.	🔒 ↴
POST	/api/orders	Creates a Order resource.	🔒 ↴
GET	/api/orders/{id}	Retrieves a Order resource.	🔒 ↴
PUT	/api/orders/{id}	Replaces the Order resource.	🔒 ↴
DELETE	/api/orders/{id}	Removes the Order resource.	🔒 ↴
PATCH	/api/orders/{id}	Updates the Order resource.	🔒 ↴
Payment			
GET	/api/payments	Retrieves the collection of Payment resources.	🔒 ↴
POST	/api/payments	Creates a Payment resource.	🔒 ↴
GET	/api/payments/{id}	Retrieves a Payment resource.	🔒 ↴
PUT	/api/payments/{id}	Replaces the Payment resource.	🔒 ↴
DELETE	/api/payments/{id}	Removes the Payment resource.	🔒 ↴
PATCH	/api/payments/{id}	Updates the Payment resource.	🔒 ↴
Brand			
GET	/api/brands	Retrieves the collection of Brand resources.	🔒 ↴
POST	/api/brands	Creates a Brand resource.	🔒 ↴
GET	/api/brands/{id}	Retrieves a Brand resource.	🔒 ↴
PUT	/api/brands/{id}	Replaces the Brand resource.	🔒 ↴
DELETE	/api/brands/{id}	Removes the Brand resource.	🔒 ↴
PATCH	/api/brands/{id}	Updates the Brand resource.	🔒 ↴
Category			
GET	/api/categories	Retrieves the collection of Category resources.	🔒 ↴
POST	/api/categories	Creates a Category resource.	🔒 ↴
GET	/api/categories/{id}	Retrieves a Category resource.	🔒 ↴
PUT	/api/categories/{id}	Replaces the Category resource.	🔒 ↴
DELETE	/api/categories/{id}	Removes the Category resource.	🔒 ↴
PATCH	/api/categories/{id}	Updates the Category resource.	🔒 ↴
Contact			
GET	/api/contacts	Retrieves the collection of Contact resources.	🔒 ↴
POST	/api/contacts	Creates a Contact resource.	🔒 ↴
GET	/api/contacts/{id}	Retrieves a Contact resource.	🔒 ↴
PUT	/api/contacts/{id}	Replaces the Contact resource.	🔒 ↴
DELETE	/api/contacts/{id}	Removes the Contact resource.	🔒 ↴
PATCH	/api/contacts/{id}	Updates the Contact resource.	🔒 ↴
Newsletter			
GET	/api/newsletters	Retrieves the collection of Newsletter resources.	🔒 ↴
POST	/api/newsletters	Creates a Newsletter resource.	🔒 ↴
GET	/api/newsletters/{id}	Retrieves a Newsletter resource.	🔒 ↴

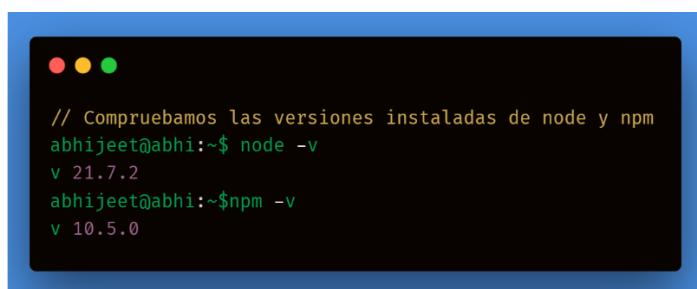
11. Parte Cliente – Frontend Tienda

El frontend de la tienda ha sido desarrollado utilizando Vue.js, un framework progresivo para la construcción de interfaces de usuario. A continuación, se detallan los pasos necesarios para configurar y lanzar la aplicación frontend.

11.1 Creación de la aplicación

1. Instalar Node.js y npm

Para comenzar con el desarrollo del frontend, es necesario tener instalado Node.js y npm (Node Package Manager). Node.js es un entorno de ejecución de JavaScript en el servidor, y npm es el gestor de paquetes asociado a Node.js.



```
// Comprobamos las versiones instaladas de node y npm
abhijeet@abhi:~$ node -v
v 21.7.2
abhijeet@abhi:~$npm -v
v 10.5.0
```

2. Crear un Proyecto Vue

Vue CLI (Command Line Interface) facilita la creación y configuración de proyectos Vue.

- **Instalar Vue CLI:**
 - Ejecutamos el siguiente comando en la terminal para instalar Vue CLI globalmente



```
abhijeet@abhi:~$ npm install -g @vue/cli
```

- **Crear un Nuevo Proyecto Vue:**
 - Navegamos al directorio donde queremos crear el proyecto y ejecutamos el siguiente comando:



```
abhijeet@abhi:~$ vue create frontend-tienda
```

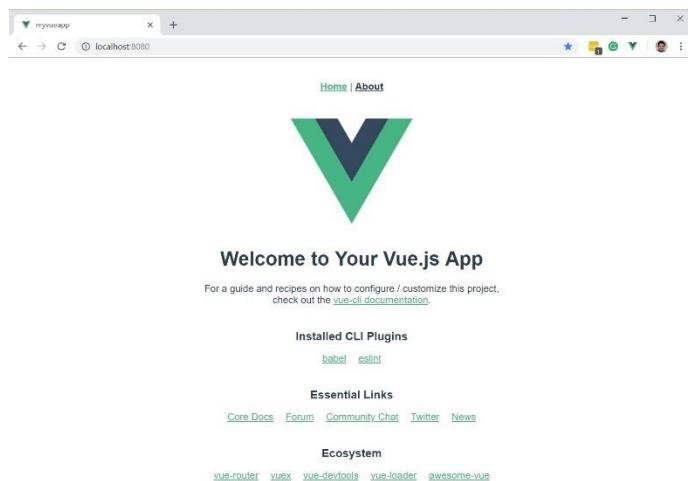
3. Lanzar la Aplicación

Después de configurar el proyecto Vue, podemos iniciar el servidor de desarrollo para lanzar la aplicación.



```
// Instalar Dependencias:  
abhijeet@abhi:~$ npm install -g @vue/cli  
// Iniciar el Servidor de Desarrollo:  
abhijeet@abhi:~$ npm run serve
```

Ya podemos acceder a la aplicación :



11.2 Proceso de desarrollo

A partir de ahora, iremos mostrando los archivos principales de la aplicación final.

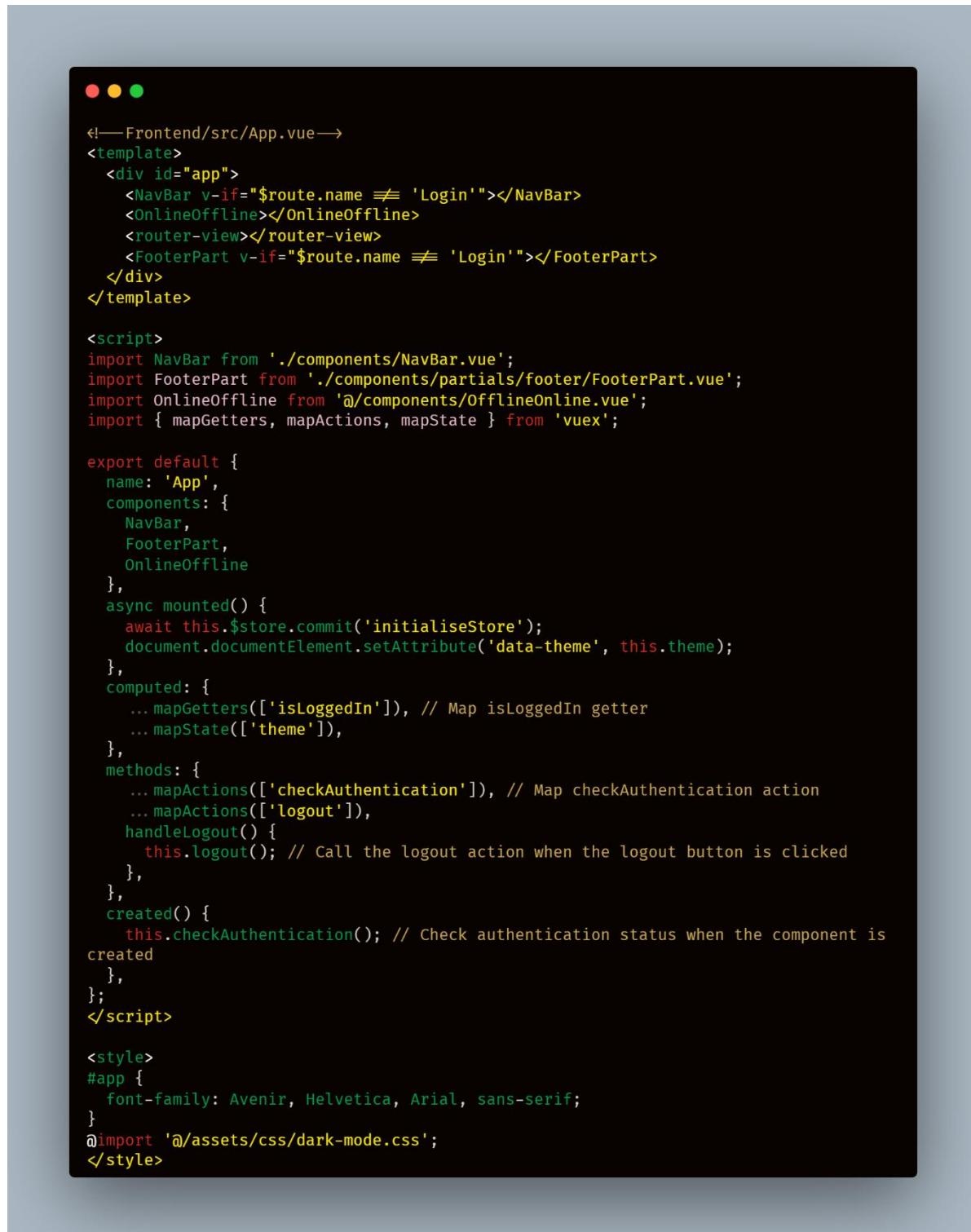
Este sería la configuración de Tailwind CSS (El framework de CSS mas potente):



```
// tailwind.config.js  
/** @type {import('tailwindcss').Config} */  
export default {  
  content: [  
    "./index.html",  
    "./src/**/*.{vue,js,ts,jsx,tsx}",  
  ],  
  prefix: "tw-",  
  theme: {  
    extend: {},  
  },  
  plugins: [],  
}
```

App.vue

El archivo **App.vue** es el componente raíz que sirve como punto de entrada de la aplicación.



```
● ● ●

<!—Frontend/src/App.vue—>
<template>
  <div id="app">
    <NavBar v-if="$route.name !== 'Login'"></NavBar>
    <OnlineOffline></OnlineOffline>
    <router-view></router-view>
    <FooterPart v-if="$route.name !== 'Login'"></FooterPart>
  </div>
</template>

<script>
import NavBar from './components/NavBar.vue';
import FooterPart from './components/partials/footer/FooterPart.vue';
import OnlineOffline from '@/components/OfflineOnline.vue';
import { mapGetters, mapActions, mapState } from 'vuex';

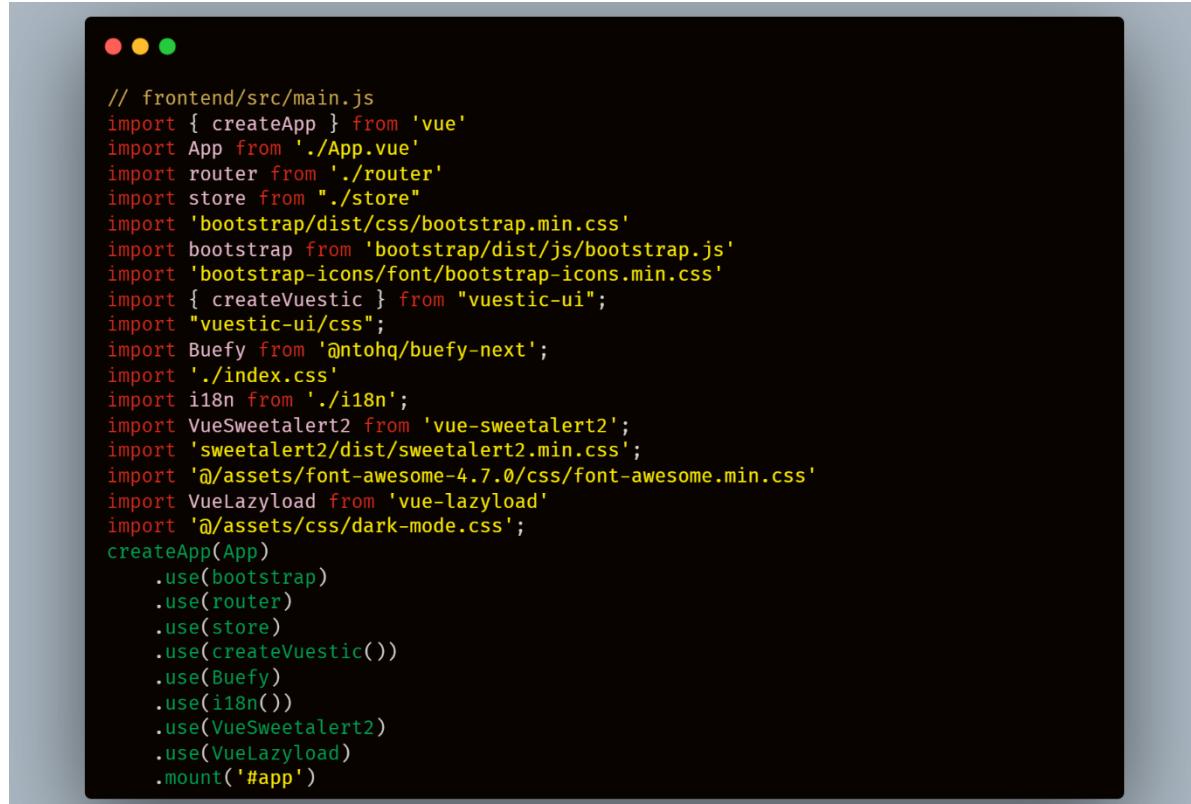
export default {
  name: 'App',
  components: {
    NavBar,
    FooterPart,
    OnlineOffline
  },
  async mounted() {
    await this.$store.commit('initialiseStore');
    document.documentElement.setAttribute('data-theme', this.theme);
  },
  computed: {
    ...mapGetters(['isLoggedIn']),
    ...mapState(['theme']),
  },
  methods: {
    ...mapActions(['checkAuthentication']), // Map checkAuthentication action
    ...mapActions(['logout']),
    handleLogout() {
      this.logout(); // Call the logout action when the logout button is clicked
    },
  },
  created() {
    this.checkAuthentication(); // Check authentication status when the component is created
  },
};
</script>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
}
@import '@/assets/css/dark-mode.css';
</style>
```

Main.js

El archivo main.js en una aplicación Vue.js es el punto de entrada principal donde se crea la instancia de Vue y se monta la aplicación en el DOM.

Aquí importamos todas las librerías que usa globalmente nuestra aplicación.



```
// frontend/src/main.js
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
import 'bootstrap/dist/css/bootstrap.min.css'
import bootstrap from 'bootstrap/dist/js/bootstrap.js'
import 'bootstrap-icons/font/bootstrap-icons.min.css'
import { createVuestic } from "vuestic-ui";
import "vuestic-ui/css";
import Buefy from '@nftohq/buefy-next';
import './index.css'
import i18n from './i18n';
import VueSweetalert2 from 'vue-sweetalert2';
import 'sweetalert2/dist/sweetalert2.min.css';
import '@/assets/font-awesome-4.7.0/css/font-awesome.min.css'
import VueLazyload from 'vue-lazyload'
import '@/assets/css/dark-mode.css';
createApp(App)
  .use(bootstrap)
  .use(router)
  .use(store)
  .use(createVuestic())
  .use(Buefy)
  .use(i18n())
  .use(VueSweetalert2)
  .use(VueLazyload)
  .mount('#app')
```

Router

En una aplicación Vue.js, el archivo router/index.js es donde se configura y define el enrutador de la aplicación. Vue Router es la biblioteca oficial de enrutamiento para Vue.js que permite crear aplicaciones de una sola página (SPA) con navegación sin recargar la página. Nuestra aplicación contiene 19 rutas diferentes.

Para aumentar la velocidad de la carga de la aplicación se hace Lazy Loading² de los componentes lo que nos permite cargar un componente solamente cuando se pide acceder a ello en vez de cargar todas las páginas por defecto.

La rutas con **meta: { requiresAuth: true }** son rutas que requieren la autenticación para acceder a ellos. La aplicación tiene todas las rutas protegidas excepto la página de Login, Página de registro y páginas de errores (error 404, error 500)

² El lazy loading en Vue.js carga componentes de rutas solo cuando se navega a ellas, mejorando el rendimiento inicial de la aplicación. <https://router.vuejs.org/guide/advanced/lazy-loading.html>

```

// frontend/src/router/index.js
import { createRouter, createWebHistory } from 'vue-router'
import { authMiddleware } from '../middleware/auth.js';
const routes = [
  {
    path: '/',
    name: 'Home',
    component: () => import('../views/HomeView.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/products',
    name: 'products',
    component: () => import('../views/ProductsView.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/products/samsung',
    name: 'productssamsung',
    component: () => import('../views/products/SamsungProducts.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/products/apple',
    name: 'productsapple',
    component: () => import('../views/products/AppleProducts.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/product/:id',
    name: 'Product',
    component: () => import('../views/ProductView.vue'),
    props: true, // Pass route params as props to the component
    meta: { requiresAuth: true }
  },
  {
    path: '/cart',
    name: 'Cart',
    component: () => import('../views/CartView.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/checkout',
    name: 'Checkout',
    component: () => import('../views/CheckoutView.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/selectpayment',
    name: 'selectpayment',
    component: () => import('../views/checkout>SelectPaymentMethod.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/profile',
    name: 'profile',
    component: () => import('../views/UserProfile.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/checkout-card',
    name: 'checkout-card',
    component: () => import('../views/checkout/CheckoutCard.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: "/login",
    name: "Login",
    component: () => import("../views/LoginView.vue"),
  },
  {
    path: "/register",
    name: "Register",
    component: () => import("../views/RegisterView.vue"),
  },
  {
    path: '/order-confirmation/:orderId', // Dynamic segment for order ID
    name: 'OrderConfirmation',
    component: () => import('../views/OrderConfirmationView.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/contact',
    name: 'contact',
    component: () => import('../views/ContactForm.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/politicas/politica-privacidad',
    name: 'politica-privacidad',
    component: () => import('@/components/politicas/PoliticaPrivacidad.vue'),
  },
  {
    path: '/orders',
    name: 'Orders',
    component: () => import('../views/UserOrders.vue'),
    meta: { requiresAuth: true },
  },
  {
    path: '/product/search',
    name: 'ProductSearch',
    component: () => import('@/components/product/ProductSearch.vue'),
    meta: { requiresAuth: true }
  },
  {
    path: '/*:catchAll(.*)',
    name: 'NotFound',
    component: () => import('@/components/errors/NotFound404.vue'),
  }
]

const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes
})

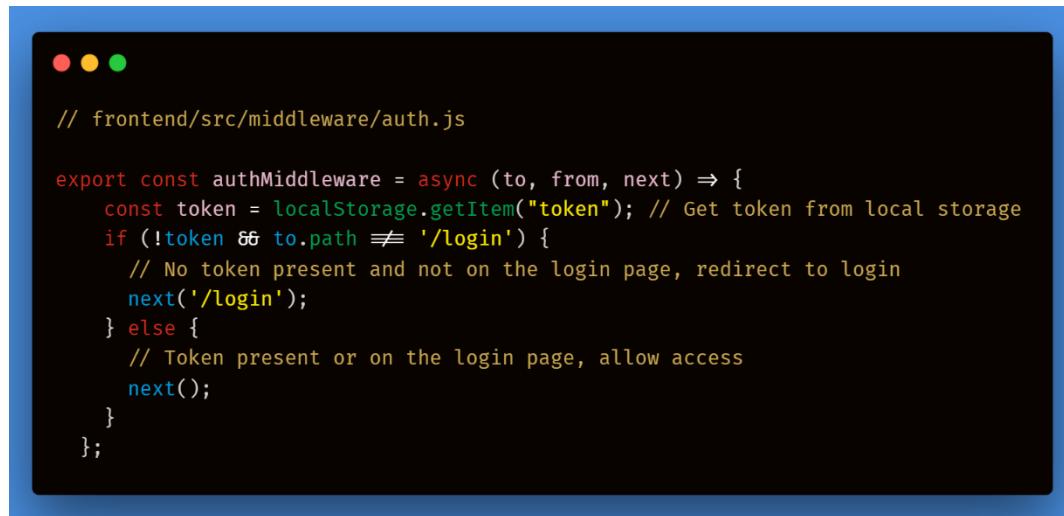
router.beforeEach((to, from, next) => {
  if (to.meta.requiresAuth) {
    // Apply authMiddleware to protected routes
    authMiddleware(to, from, next);
  } else {
    // For non-protected routes, proceed to the next route
    next();
  }
});

export default router

```

Middleware

El middleware es una función que permite realizar verificaciones y ejecutar lógica antes de permitir el acceso a ciertas rutas.



```
// frontend/src/middleware/auth.js

export const authMiddleware = async (to, from, next) => {
  const token = localStorage.getItem("token"); // Get token from local storage
  if (!token || to.path !== '/login') {
    // No token present and not on the login page, redirect to login
    next('/login');
  } else {
    // Token present or on the login page, allow access
    next();
  }
};
```

El middleware obtiene el token (JWT) almacenado en el localStorage. Si no hay token y la ruta no es /login, redirige al usuario a la página de login. Si hay token o la ruta es /login, permite el acceso a la ruta solicitada.

I18n

La aplicación de la tienda está internacionalizada, lo que significa que puede ser utilizada en múltiples idiomas, ofreciendo una mejor experiencia de usuario para una audiencia global. En este proyecto, se ha implementado la internacionalización para dos idiomas: español e inglés.

Para ello hemos usado la librería **vue-i18n** y contiene la siguientes estructura en el proyecto:



```
● ● ●

// src/i18n/index.js

import { createI18n } from "vue-i18n"
import { nextTick } from 'vue';

let i18n;
export const SUPPORT_LOCALES = ['en', 'es'];

export function setI18nLanguage(locale) {
    loadLocaleMessages(locale);

    if (i18n.mode === 'legacy') {
        i18n.global.locale = locale;
    } else {
        i18n.global.locale.value = locale;
    }

    document.querySelector('html').setAttribute('lang', locale);
    localStorage.setItem('lang', locale);
}

export async function loadLocaleMessages(locale) {
    // load locale messages with dynamic import
    const messages = await import(
        /* webpackChunkName: "locale-[request]" */ `./locales/${locale}.json`
    );

    // set locale and locale message
    i18n.global.setLocaleMessage(locale, messages.default);

    return nextTick();
}

export default function setupI18n() {
    if (!i18n) {
        let locale = localStorage.getItem('lang') || 'en';

        i18n = createI18n({
            globalInjection: true,
            legacy: false,
            locale: locale,
            fallbackLocale: 'en'
        });

        setI18nLanguage(locale);
    }
    return i18n;
}
```

```
● ● ●

## i18n/locales/en.json
{
    "locale": {
        "en": "English",
        "es": "Spanish"
    },
    "navbar": {
        "home": "Home",
        "products": "Products",
        ....
    },
}
## i18n/locales/es.json
{
    "locale": {
        "en": "Inglés",
        "es": "Español"
    },
    "navbar": {
        "home": "Inicio",
        "products": "Productos",
        "offers": "Ofertas",
        ....
    }
}
```

Store

Se utiliza Vuex para gestionar el estado de forma centralizada. Vuex facilita el almacenamiento, acceso y modificación del estado compartido entre componentes, manteniendo el código organizado y predecible.

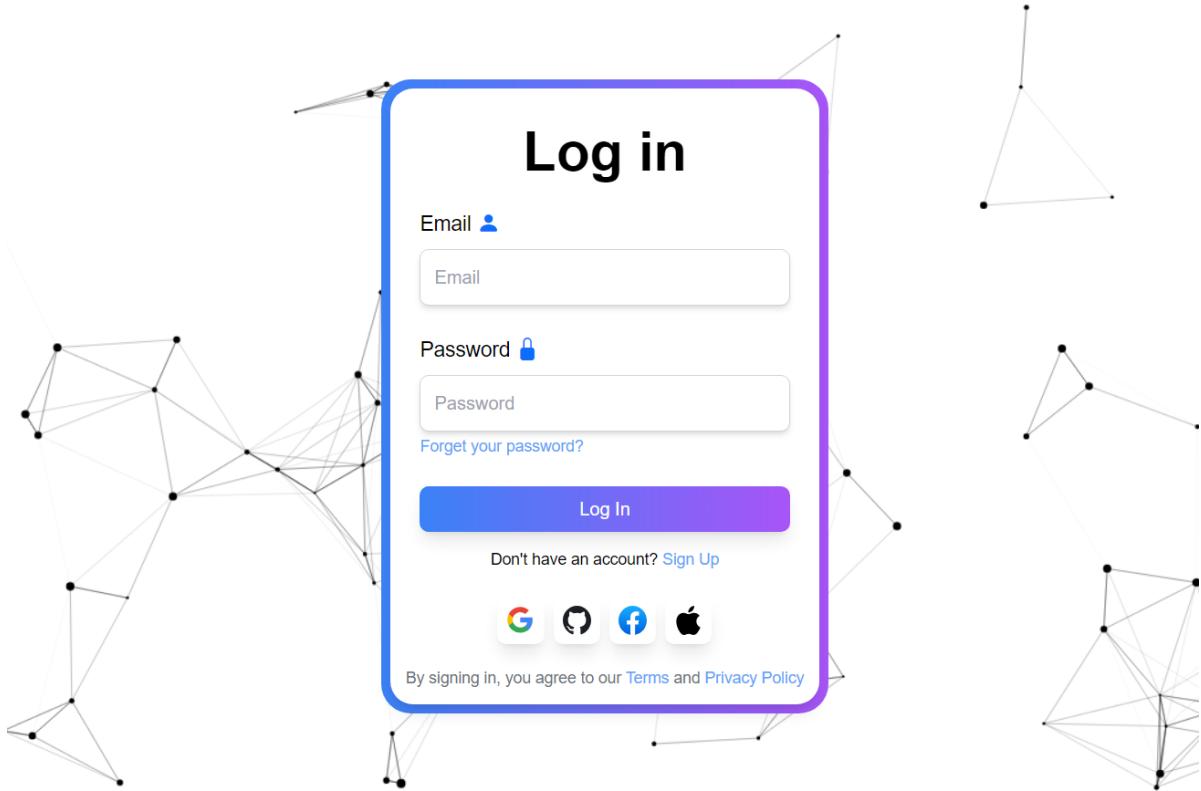
```
1 // frontend/src/store/index.js
2
3 import { createStore } from 'vuex';
4 import createPersistedState from "vuex-persistedstate";
5 /* eslint-disable */
6 export default createStore({
7 >   state: { ... },
16
17 >   mutations: { ... },
54
55   actions: {
56     checkAuthentication({ commit }) {
57       },
58     logout({ commit }) {
59       },
60     changeLocale({ commit }, locale) { ... },
62
63     toggleTheme({ commit, state }) {
64       },
65
66   },
67   modules: {},
68   getters: {
69 >     isLoggedIn(state) { ... },
71       },
72     },
73   plugins: [createPersistedState()],
74 })
```

11.3 Interfaz de la aplicación

La aplicación requiere autenticación para acceder. Por eso, si intentamos acceder a la aplicación sin autenticarse siempre nos lleva a la página de [Login](#).

En esta página si tenemos una cuenta se puede iniciar la sesión insertando los datos o podemos registrarnos.

La página de [login](#) es el siguiente:



Se usa una librería para mostrar animaciones aleatorias en el fondo que aparecen cada vez que se recarga la página.

Nota: Las opciones de inicio de sesión con **Google**, **GitHub**, **Facebook** y **Apple** son mejoras que se aplicarán en futuro.

Al iniciar la sesión se manda un correo al usuario avisándole el inicio de sesión por motivos de seguridad. Se indica el **IP** del dispositivo con el cual se inicia la sesión.

Login Confirmation Inbox x

iamabhiject.dev@gmail.com via sendgrid.net
to me ▾

16:08 (8 minutes ago) ☆ ☺ ← ⋮

Hello abhijeet

Welcome back!

You have successfully logged in to your account.

IP Address: 193.111.53.99
User Agent: Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/125.0.0.0 Mobile Safari/537.36

If this login was not done by you, please contact us immediately and consider changing your password for security.

La pagina de registro es el siguiente e igualmente al registrar se manda un correo de bienvenida al usuario registrado.

REGISTER

Email

First Name

Last Name

Password

Register

¿Ya tienes cuenta? [Inicia Sesión](#)

By registering, you agree to our [Terms](#) and [Privacy Policy](#)

Welcome to Our Website Inbox x

iamabhijeet.dev@gmail.com via sendgrid.net
to me ▾

Hello Abhijeet,

Welcome to our website! We are glad to have you on board.

Thank you for registering with us.

If you have any questions or need assistance, feel free to contact us.

Best regards,

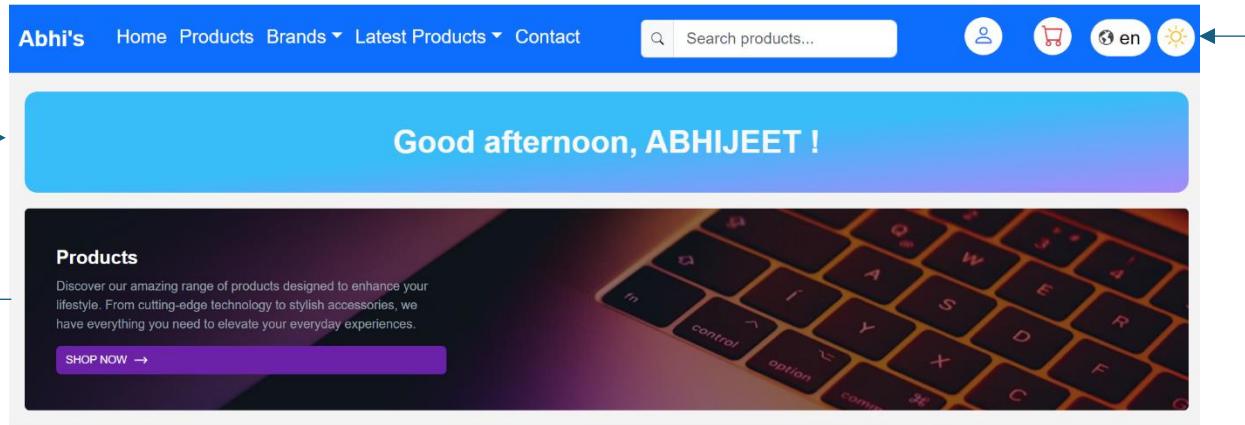
Your Website Team

Correo de bienvenida

Formulario de Registro

Página de Inicio

Al iniciar la sesión se dirige al usuario a la página de inicio:



La página de inicio tiene una barra de navegación donde están los enlaces necesarios para navegar en la página.

- ▶ Este componente recupera el nombre de usuario y saluda al usuario. Dependiendo de la hora se cambia el mensaje (Buenos días, buenas tardes, buenas noches etc.).
- ▶ Este componente redirige al usuario a la página de productos.

A continuación, se detallan los siguientes elementos de la página de inicio.

Explore the latest Samsung products

Discover the newest offerings from Samsung and stay ahead with cutting-edge technology

Explore Now →



Enlace que redirige a los productos de la marca **Samsung**.

iPhone 15 Pro

A total powerhouse.

Buy >



Enlace que redirige al producto mencionado en la imagen.

iPad air

Light. Bright. Full of might.

Buy >



Enlace que redirige al producto mencionado en la imagen.

2,157 people have said how good our store is

Our happy clients say about us



"La mejor tienda del Mundo!!"



JOSEP MAGRANER I RAMÓN

Una Maquina

Testimonios de los clientes.



"Xé, em jubile però ja!"



Jose Fidel Oltra Landete

Escritor, Profesor, Musico, Jugador de ajedrez

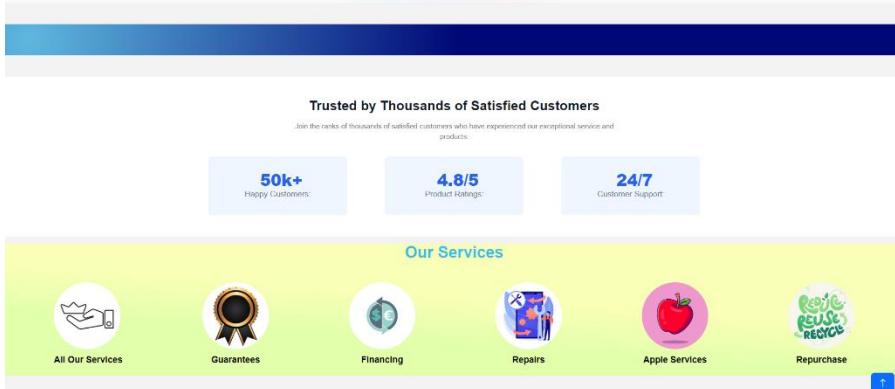


"Otro cliente satisfecholl! Malditos"

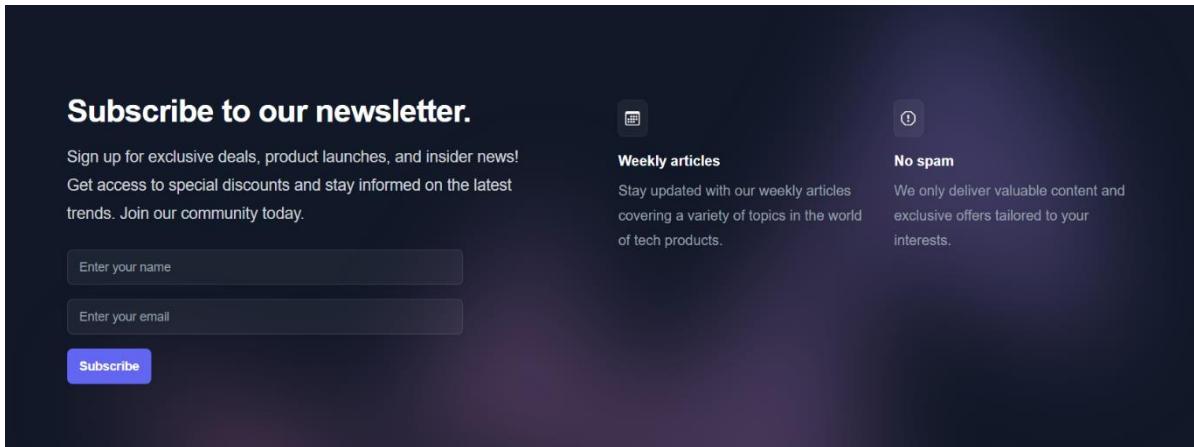


Dr. Juan Bautista Talens Felis

El mejor politico y profesor.



Estadísticas de reseñas y servicios ofrecidos.



Este componente es un formulario para suscribir al boletín de noticias o Newsletter.

Al suscribir se manda un correo al usuario.



Hello Abhijeet,

Welcome to Our Newsletter!

Thank you for subscribing to our newsletter. You will now receive the la'

Stay tuned for exciting content!

Frequently Asked Questions

How can I place an order? ^

You can easily place an order on our website by browsing our product catalog, selecting the items you want, and adding them to your cart. Then, proceed to checkout, where you can provide your shipping and payment information to complete the order.

What payment methods do you accept? ^

We accept various payment methods including credit cards, PayPal, and bank transfers.

How long does shipping take? ^

Shipping times vary depending on the destination and the selected shipping method.

Can I return a product if I'm not satisfied? ^

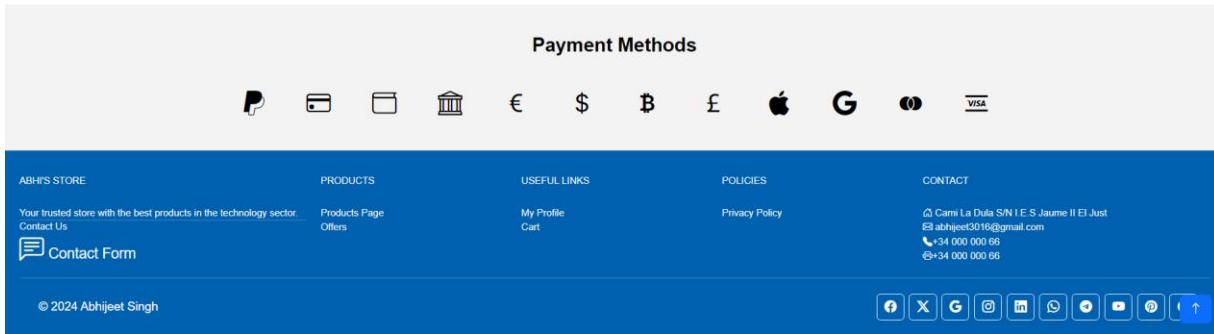
If you're not satisfied with a product, we offer a 30-day return policy.

Do you offer international shipping? ^

We offer international shipping to most countries around the world.

Still have questions? Contact our support

Componente de preguntas frecuentes.



Footer de la página y métodos de pagos aceptados.

Navbar

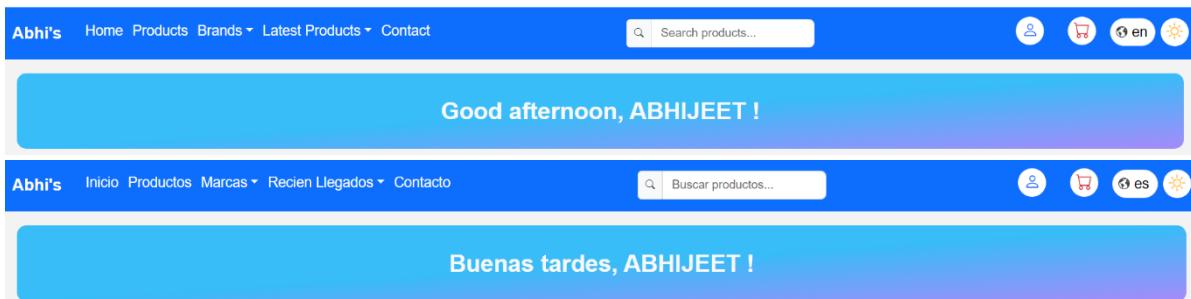
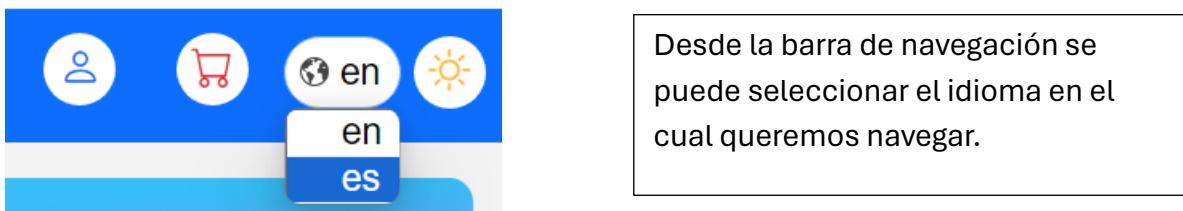
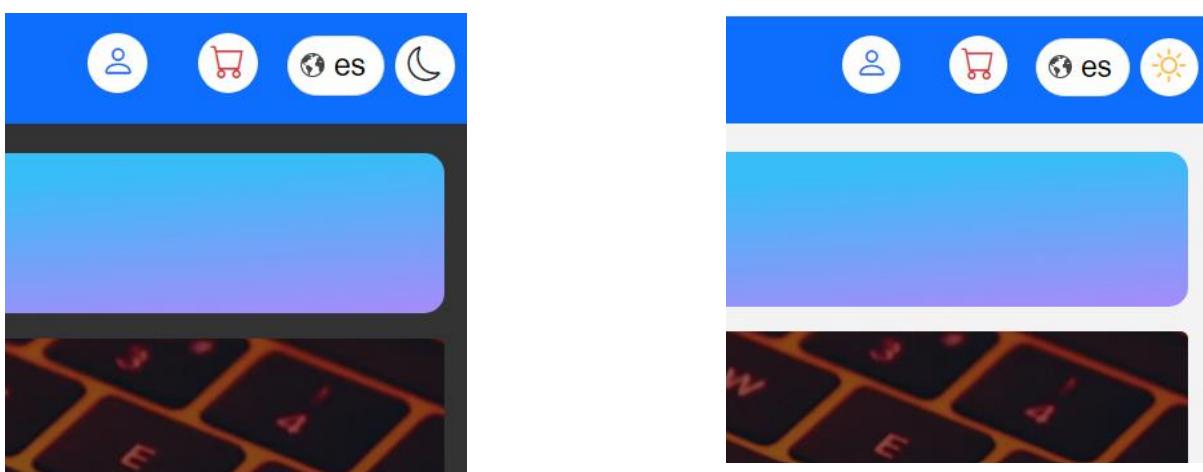
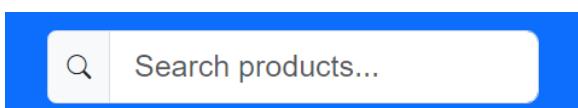


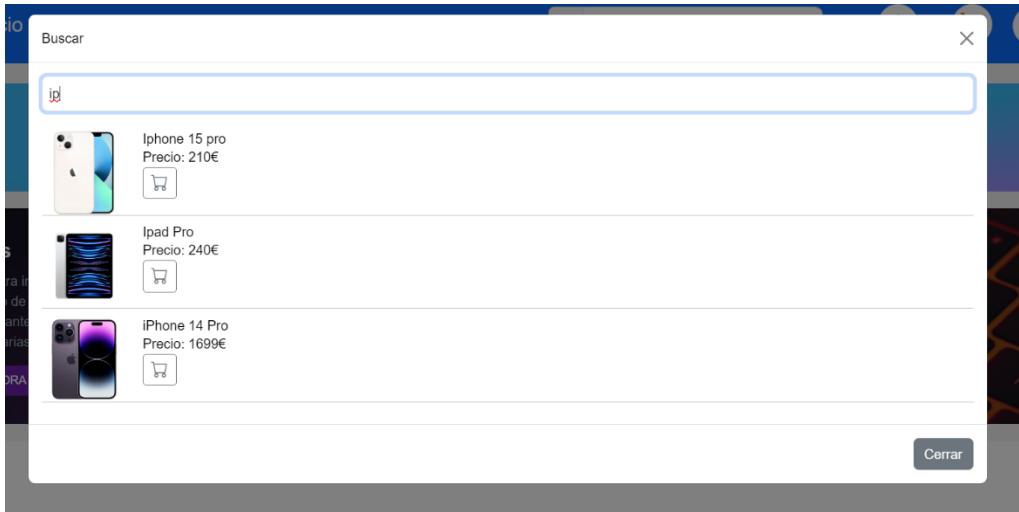
Imagen 1: Pagina en inglés. **Imagen 2:** Pagina en español.

La aplicación también tiene la funcionalidad de modo claro y modo oscuro.

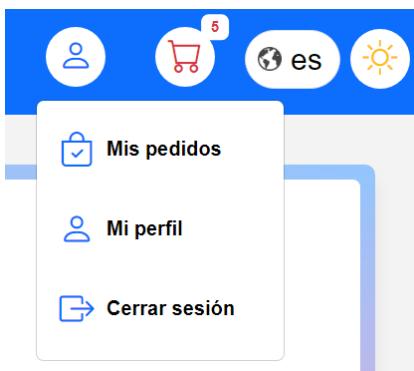


Dentro de la barra de navegación tenemos el formulario para buscar productos.

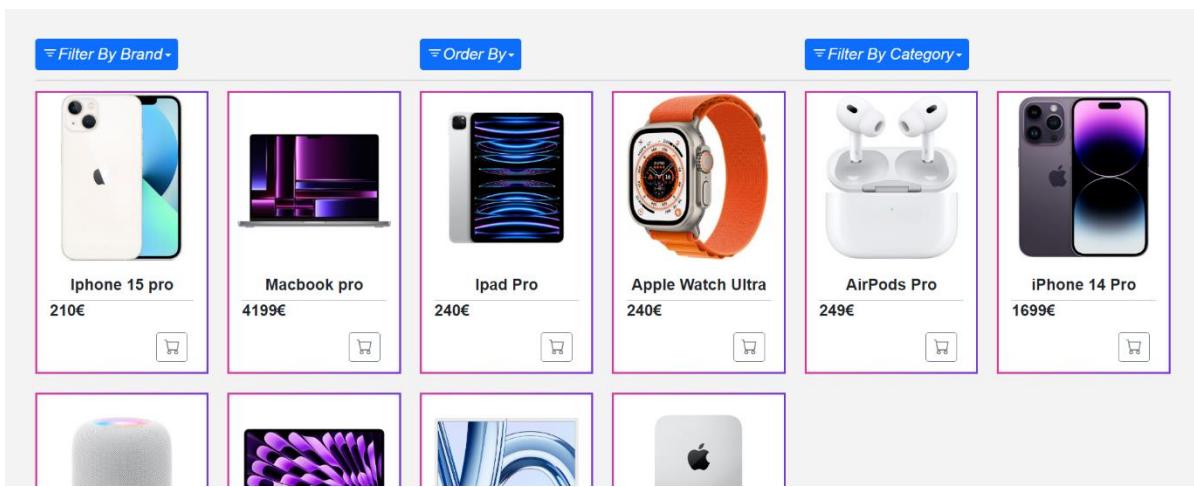




Se puede ver el perfil de usuario, los pedidos y terminar la sesión desde el siguiente dropdown:



Página de Productos



= Filter By Brand -

Apple
Samsung
HP
Huawei
Google

Iphone 15 pro
210€

Macbook pro
4199€

Filtrar productos por la marca

= Filter By Brand -

= Order By -

Iphone 15 pro
210€

iPhone 14 Pro
1699€

Samsung Galaxy S24 Ultra
12GB/1TB
2150€

= Filter By Category -

Smartphone
Tablet
Smartwatch
Computer
Laptop
Monitor
TV
Virtual Glass
Accessories
Headphones

Filtrar por categoría.

= Order By -

Iphone 15 pro
210€

Ipad Pro
240€

Apple Watch Ultra
240€

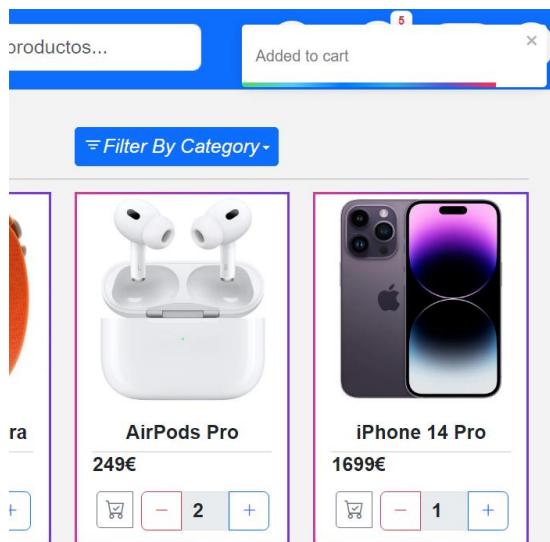
- Ordenar por precio ascendente
- Ordenar por precio descendente
- Ordenar alfabéticamente [A-Z] [Z-A]

« 1 2 »

Para aumentar el rendimiento se usa paginación y se cargan solo 10 productos en una petición en vez de mostrar todos los productos.

Carrito

Cada producto tiene un botón para añadir al carrito.



Al pulsar el botón de añadir al carrito se añade el producto y se muestra un **toast** (notificación temporal en forma de popup).

Aparecen botones para añadir o restar el numero de productos.

Esta es la vista de la pagina de carrito:

Cart details	
Subtotal	6876 €
Total	6876 €

Aquí se pueden ver todos los productos en el carrito, añadir más productos, quitar productos etc.

El botón de Checkout en el carrito redirige a la siguiente página donde se puede seleccionar el método de pago.

Please Select the payment method.

Pay with Card

CASH ON DELIVERY (COD)

Pago con tarjeta redirige a la página de pasarela de pago.

Contrarrembolsos redirigen directamente a la página de formulario.

Pago

Esta es la página de pago de Stripe:

TEST MODE

Purchase

€6,876.00

Pay with link

Or pay with card

Email
acc1abhi@gmail.com

Card information
4242 4242 4242 4242
02 / 52 225 VISA

Cardholder name
Abhijeet Singh

Country or region
Spain

Securely save my information for 1-click checkout
Pay faster on this site and everywhere Link is accepted.

Processing...

Rellenamos los datos y en pago correcto se nos redirige al formulario donde rellanaremos los datos de la entrega y contacto.

Checkout Page

Total Price
6876

05/30/2024 02:14 PM

Address
Enter a location

City
Enter a location

Province
Choose Province

Postal Code
0

Mobile
0

Place Order

En esta página para mejorar la experiencia del usuario se usa el API de Google Maps para autocompletar la dirección.

Address
IES JAUME

IES Jaume I Avinguda d'Albaida, Ontinyent, Spain

IES Jaume I Borriana Borriana, Spain

IES Jaume II el Just Carrer Doctor Gómez Ferrer, Tavernes de la Valldigna, Spain

IES Jaume I Polígono Número 65, Sagunto, Spain

IES Jaume II Carrer Doctor Cristóbal Pardo, Alicante, Spain

powered by Google

Después de llenar los datos, damos al botón de hacer pedido y nos dirige a la página de confirmación.

Order Confirmation

Order ID: 15
Total Price: 6876
Items:
Ipad Pro - Quantity: - Price: 240

Apple Watch Ultra - Quantity: - Price: 240

AirPods Pro - Quantity: - Price: 249

iPhone 14 Pro - Quantity: - Price: 1699

Macbook pro - Quantity: - Price: 4199


Ya tenemos tu pedido, echa un vistazo a otros productos 
[Ver Mis Pedidos](#) 

Order Confirmation Inbox ×



iamabhiheet.dev@gmail.com via sendgrid.net
to me, abhiheet3016 ▾

Hello abhiheet

Thanks for your order!

Your order #15 has been confirmed.

Order details:

Total: 6,876.00€

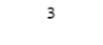
Created At: 2024-05-30 14:14:00

Se manda un correo de confirmación al usuario.

Página de Pedidos

En esta página podemos ver todos nuestros pedidos.

Your Orders

Order ID: 1	Order ID: 2
<p>Total: 4439€ Prepared <input checked="" type="checkbox"/> : No Created At  : May 20, 2024, 12:46:00</p>  1	<p>Total: 5258€ Prepared <input checked="" type="checkbox"/> : No Created At  : May 21, 2024, 06:04:00</p>  2
<p>Total: 9838€ Prepared <input checked="" type="checkbox"/> : No Created At  : May 21, 2024, 06:41:00</p>  3	<p>Total: 9838€ Prepared <input checked="" type="checkbox"/> : No Created At  : May 21, 2024, 06:41:00</p>  4
<p>Total: 9838€ Prepared <input checked="" type="checkbox"/> : No</p>  5	<p>Total: 9838€ Prepared <input checked="" type="checkbox"/> : No</p>  6

Perfil

My Profile

Name: abhijeet
Last Name: singh
Email: acc1abhi@gmail.com

Página de Contacto

Contacta con Nosotros

Your Name

Your Email

Phone Number

Message

Send message

Nuestra Tienda Física



Message Submitted Inbox X



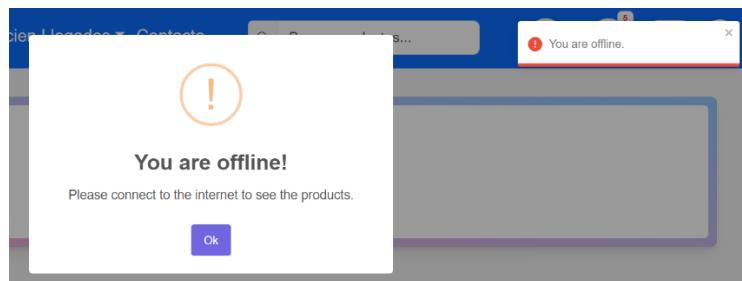
iamabhijeet.dev@gmail.com via sendgrid.net
to me ▾

Hello abhijeet,

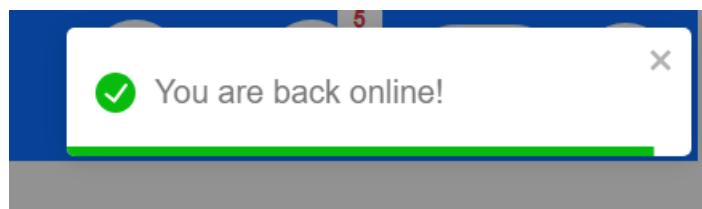
Your message has been submitted successfully. We will get in touch with you soon.

Al llenar el formulario de contacto se manda un correo al usuario.

Funcionalidad de Online/Offline



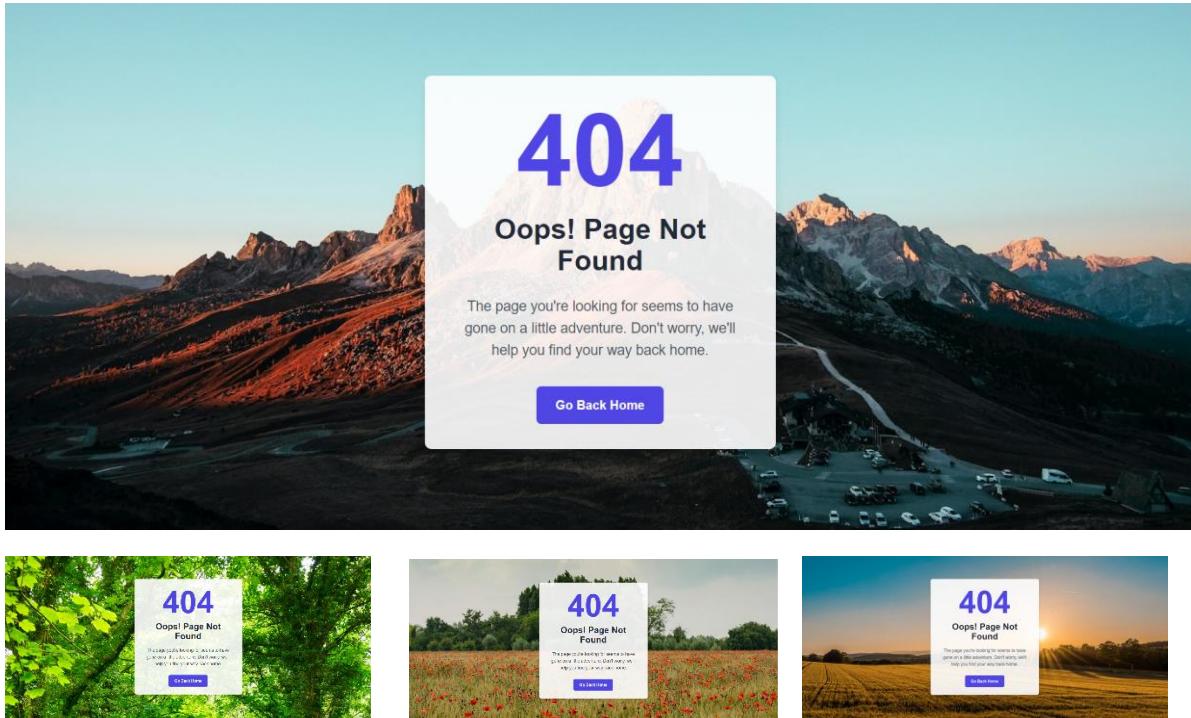
Cuando se pierde conexión con internet, sale un popup y un toast.



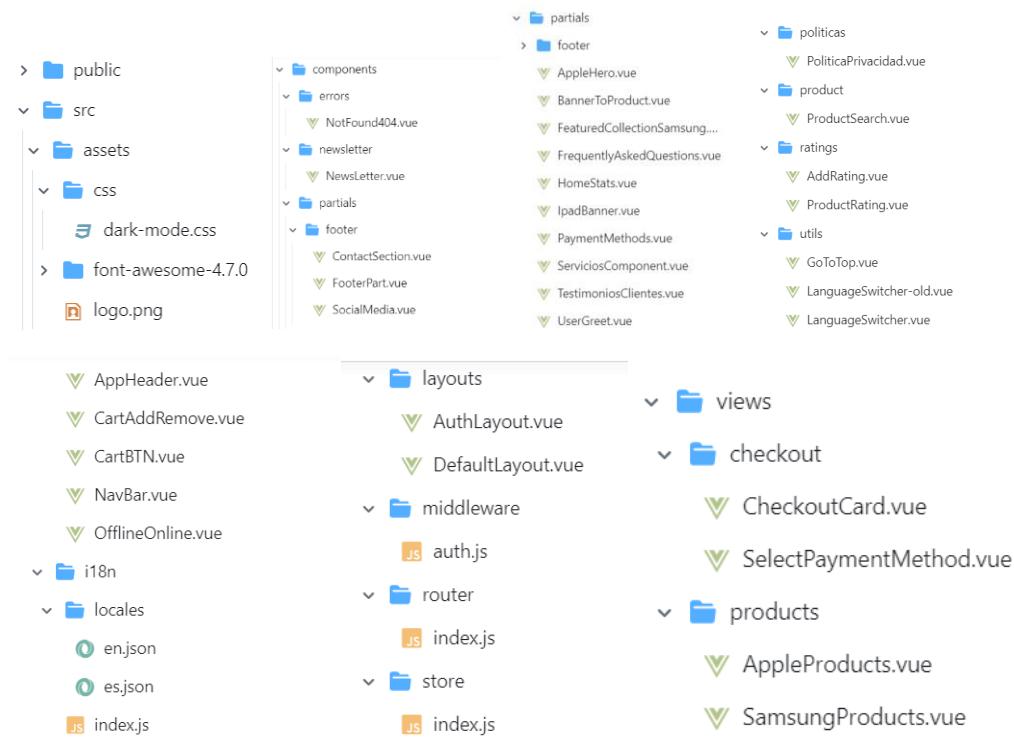
Al volver a tener conexión sale otro toast.

Error 404

La página de error 404, cada vez tiene una imagen de fondo diferente. Para ello utiliza la api de Unsplash (Plataforma de Imágenes de libre uso).



La estructura final de la aplicación es:



```

▼ CartView.vue
▼ CheckoutRedirect.vue
▼ CheckoutView.vue
▼ ContactForm.vue
▼ HomeView.vue
▼ LoginView.vue
▼ OrderConfirmationView.vue
▼ ProductView.vue
▼ ProductsView.vue
▼ RegisterView.vue
▼ StoreMap.vue
js UseEmail.js
▼ UserOrders.vue
▼ UserProfile.vue
▼ App.vue
js BaseEndpoint.js
js apiConfig.js

```

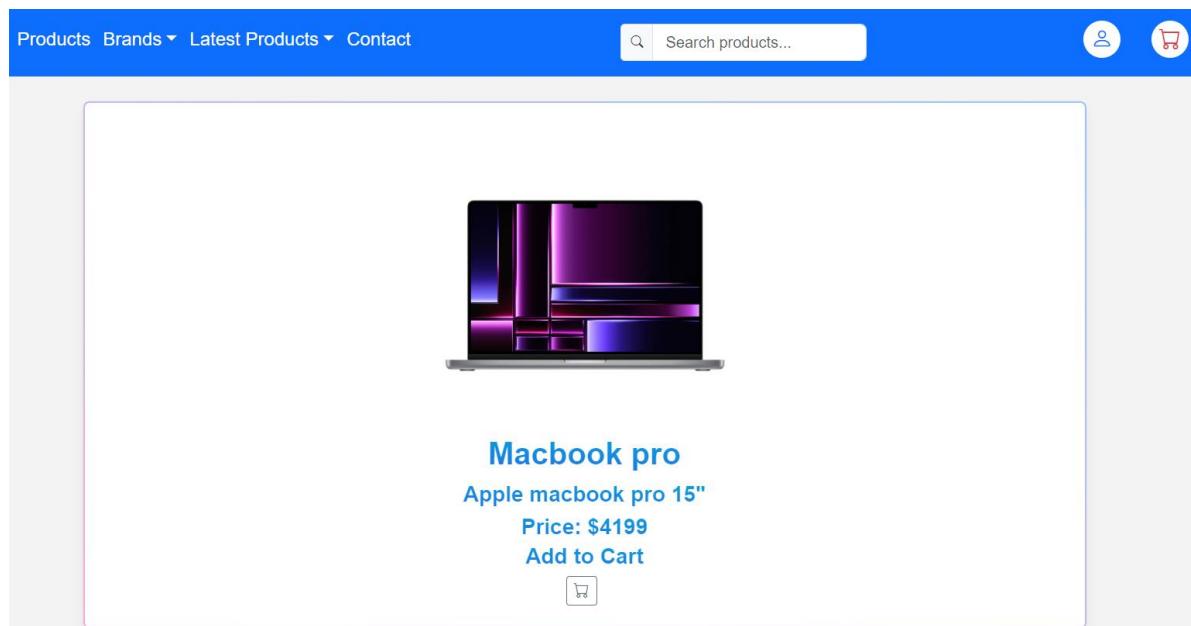
```

js apiUser.js
index.css
main.js
.gitignore
README.md
babel.config.js
jsconfig.json
package-lock.json
package.json
postcss.config.js
tailwind.config.js
vue.config.js

```

Página de producto individual

Se puede ver un producto individual haciendo clic desde la vista de varios productos.



Reseñas

En la página de productos individuales se pueden ver reseñas de los productos de la página. Además, hay un formulario para agregar una reseña.

Add Rating

Rating:

5

Comment:

Muy buen producto

Submit

Ratings

User: abhijeet

Rating:★★★★★

Comment: Muy buen producto

User: david

Rating:★★★★★

Comment: El mejor producto del mercado

12. Parte Cliente – Admin

La aplicación del panel de administración utiliza las mismas tecnologías que la aplicación frontend de la tienda detallada en el punto 11. Sin embargo, hay diferencias clave en la interfaz y las funcionalidades, diseñadas específicamente para los administradores de la tienda.

12.1 Contenido

A continuación, se detallan los archivos principales de la aplicación de Admin.

App.vue

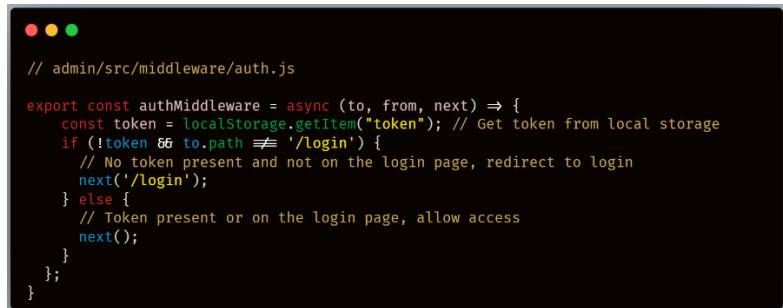


```
<template>
  <Sidebar v-if="$route.name !== 'Login' && $route.name !== 'NotFound'" />
  <NavBar v-if="$route.name !== 'Login' && $route.name !== 'NotFound'"></NavBar>
  <div :style="{ 'margin-left': sidebarWidth }"></div>

  <router-view></router-view>
</template>

<script>
import Sidebar from '@components/sidebar/SideBar'
import { sidebarWidth } from '@components/sidebar/state'
import { mapGetters, mapActions } from 'vuex';
import NavBar from './components/navbar/NavBar.vue';
export default {
  name: 'App',
  async mounted() {
    //await this.$store.commit('initialiseStore')
  },
  components: { Sidebar, NavBar },
  setup() {
    return { sidebarWidth }
  },
  computed: {
    ...mapGetters(['isLoggedIn']), // Map isLoggedIn getter
  },
  methods: {
    ...mapActions(['checkAuthentication']), // Map checkAuthentication action
    ...mapActions(['logout']),
    handleLogout() {
      this.logout(); // Call the logout action when the logout button is clicked
    }
  },
  created() {
    this.checkAuthentication(); // Check authentication status when the component is created
  }
}</script>
```

Middleware/auth.js



```
// admin/src/middleware/auth.js

export const authMiddleware = async (to, from, next) => {
  const token = localStorage.getItem("token"); // Get token from local storage
  if (!token && to.path !== '/login') {
    // No token present and not on the login page, redirect to login
    next('/login');
  } else {
    // Token present or on the login page, allow access
    next();
  }
};
```

Main.js



```
// src/main.js
import "bootstrap/dist/css/bootstrap.min.css";
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
import "bootstrap/dist/js/bootstrap.min";
import 'bootstrap-icons/font/bootstrap-icons.min.css'
import '@fortawesome/fontawesome-free/js/all'
import VueApexCharts from "vue3-apexcharts";
import VueSweetalert2 from 'vue-sweetalert2';
import 'sweetalert2/dist/sweetalert2.min.css';
import './index.css'

createApp(App)
  .use(store)
  .use(router)
  .use(VueApexCharts)
  .use(VueSweetalert2)
  .mount('#app')
```

Store/index.js



```
// src/store/index.js
import { createStore } from 'vuex'
import createPersistedState from "vuex-persistedstate";
/* eslint-disable */
export default createStore({
  state: {
    user: null,
    token: null,
    isAuthenticated: false,
  },
  getters: {
    isLoggedIn(state) {
      return !!state.token;
    },
  },
  mutations: {
    setUser(state, user) {
      state.user = user;
    },
    setToken(state, token) {
      state.token = token;
    },
    setAuthentication(state, isAuthenticated) {
      state.isAuthenticated = isAuthenticated;
    },
  },
  actions: {
    checkAuthentication({ commit }) {
      const token = localStorage.getItem("token");
      if (token) {
        commit("setAuthentication", true);
      } else {
        commit("setAuthentication", false);
      }
    },
    logout({ commit }) {
      localStorage.removeItem('token');
      commit('setToken', null);
      commit('setAuthentication', false);
      commit('setUser', null);
      location.reload();
    },
  },
  modules: {
  },
  plugins: [createPersistedState()]
})
```

Importaciones dinámicas de las rutas:

```
// src/router/imports.js
// Dynamic imports - Lazy Loading
export const HomeView = () => import('../views/HomeView.vue');
export const AddProduct = () => import('../components/AddProduct.vue');
export const ProductView = () => import('../views/ProductsView.vue');
export const SingleProductView = () => import('../views/SingleProductView.vue');
export const CategoriesView = () => import('../views/category/CategoriesView.vue');
export const SingleCategoryView = () => import('../views/category/SingleCategoryView.vue');
export const AddProductPage = () => import('../views/AddProductPage.vue');
export const AddCategoryPage = () => import('../views/category/AddCategoryPage.vue');
export const EditProduct = () => import('../views/EditProduct.vue');
export const EditProductPage = () => import('../views/EditProductPage.vue');
export const InventoryShow = () => import('../views/inventory/InventoryShow.vue');
export const Login = () => import('../views/LoginView.vue');
export const OrderView = () => import('../views/orders/OrdersView.vue');
export const UserView = () => import('../views/user/UsersView.vue');
export const EditCategory = () => import('@/components/forms/EditCategoryForm.vue');
export const Contactlist = () => import('../views/contact/ContactList.vue');
export const Error404 = () => import('@/components/errors/NotFound404.vue');
export const AddBrand = () => import('@/views/brand/AddBrand.vue');
export const ListBrands = () => import('@/views/brand/ListBrands.vue');
```

Router/index.js

La aplicación de admin tiene 19 rutas diferentes. Cada ruta requiere autenticación.

```
import { createRouter, createWebHistory } from 'vue-router'
const routes = [
  {
    path: '/',
    name: 'Home',
    component: components.HomeView,
    meta: { requiresAuth: true }
  },
  {
    path: '/addproduct',
    name: "addproduct",
    component: components.AddProduct,
    meta: { requiresAuth: true }
  },
  {
    path: '/products',
    name: "products",
    component: components.ProductView,
    meta: { requiresAuth: true }
  },
  {
    path: '/product/:id',
    name: 'SingleProductView',
    component: components.SingleProductView,
    meta: { requiresAuth: true }
  },
  {
    path: '/categories',
    name: 'Categories',
    component: components.CategoriesView,
    meta: { requiresAuth: true }
  },
  {
    path: '/categories/:id',
    name: 'SingleCategoryView',
    component: components.SingleCategoryView,
    props: true,
    meta: { requiresAuth: true }
  },
  {
    path: "/addproduct",
    name: "addproduct",
    component: components.AddProductPage,
    meta: { requiresAuth: true }
  },
  {
    path: "/addcategory",
    name: "addcategory",
    component: components.AddCategoryPage,
    meta: { requiresAuth: true }
  },
  {
    path: '/products/:id/edit',
    name: 'EditProduct',
    component: components.EditProduct,
    props: true,
    meta: { requiresAuth: true }
  },
  {
    path: '/edit-product',
    name: 'EditProductPage',
    component: components.EditProductPage,
    meta: { requiresAuth: true }
  },
]
```

```

{
  path: '/inventory',
  name: 'InventoryShow',
  component: components.InventoryShow,
  meta: { requiresAuth: true }
},
{
  path: "/orders",
  name: "Orders",
  component: components.OrderView,
  meta: { requiresAuth: true }
},
{
  path: "/users",
  name: "users",
  component: components.UserView,
  meta: { requiresAuth: true }
},
{
  path: "/editcategory",
  name: "editcategory",
  component: components.EditCategory,
  meta: { requiresAuth: true }
},
{
  path: "/login",
  name: "Login",
  component: components.Login,
},
{
  path: "/contacts",
  name: "contacts",
  component: components.ContactList,
  meta: { requiresAuth: true }
},
{
  path: "/:catchAll(.*)",
  name: "NotFound",
  component: components.Error404,
},
{
  path: "/addrbrand",
  name: "addrbrand",
  component: components.AddBrand,
  meta: { requiresAuth: true }
},
{
  path: "/brands",
  name: "brands",
  component: components.ListBrands,
  meta: { requiresAuth: true }
}
]

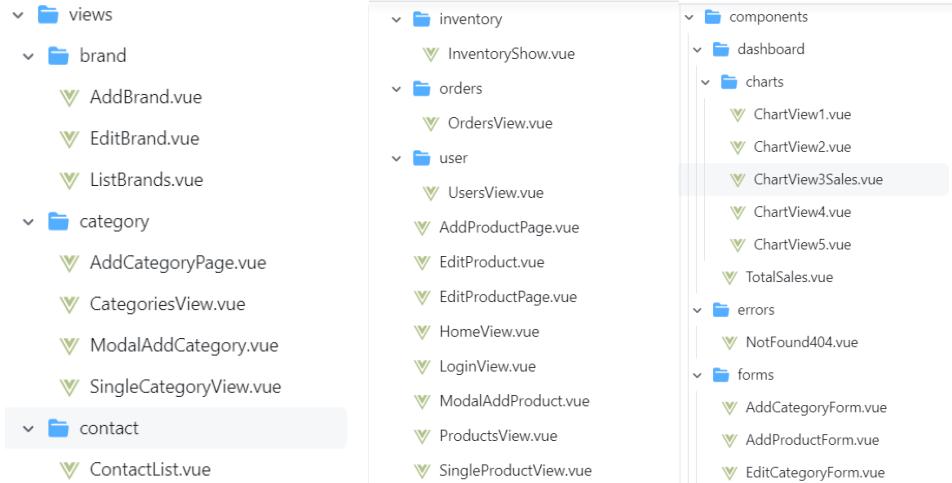
const router = createRouter({
  history: createWebHistory(process.env.BASE_URL),
  routes
})

router.beforeEach((to, from, next) => {
  if (to.meta.requiresAuth) {
    // Apply authMiddleware to protected routes
    authMiddleware(to, from, next);
  } else {
    // For non-protected routes, proceed to the next route
    next();
}
});

export default router

```

Las vistas y los componentes de la aplicación son las siguientes:

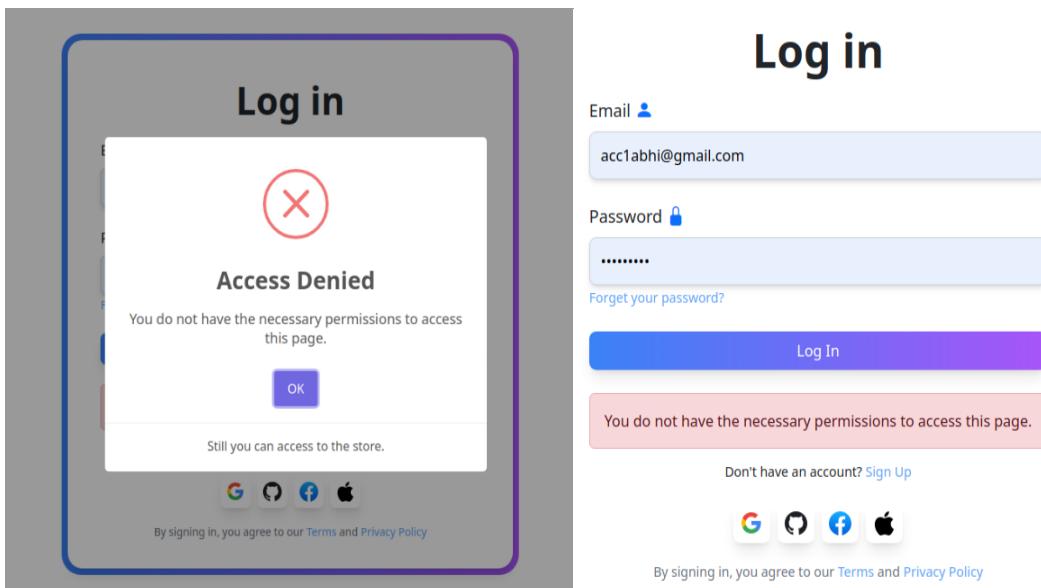


✓	navbar	✓	layouts
	└─ NavBar.vue		└─ LoginLayout.vue
✓	sidebar	✓	└─ MainLayout.vue
	└─ SideBar.vue	✓	middleware
	└─ SidebarLink.vue		└─ auth.js
	└─ state.js	✓	router
	└─ AddCategory.vue		└─ imports.js
	└─ AddProduct.vue	✓	└─ index.js
		✓	store
			└─ index.js

12.2 Interfaz de la Aplicación

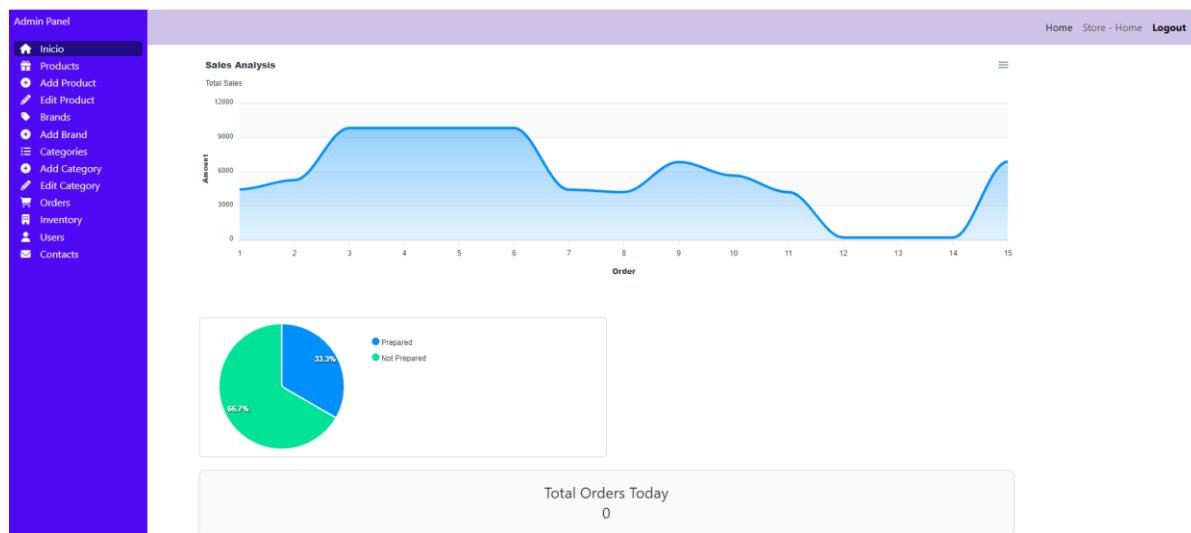
La página de login de la página de admin es igual que la tienda, pero lo que diferencia esta página del otro es en la parte de la autenticación. Solo los usuarios con el rol **ROLE_ADMIN** pueden acceder al panel de administración.

Si intentamos acceder sin ser admin nos sale este aviso denegando nos el acceso.



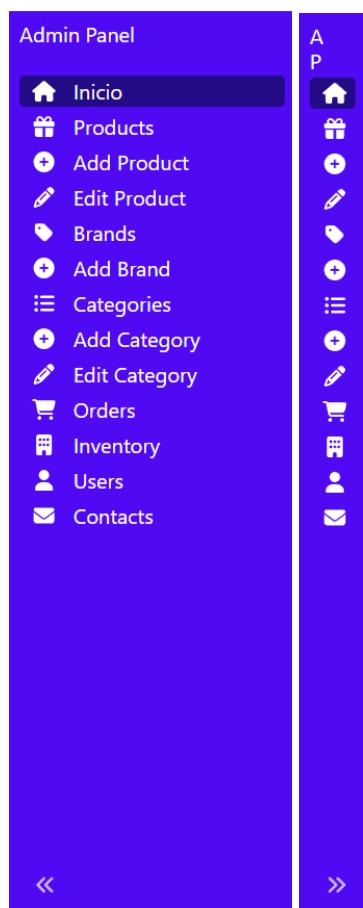
Home | Inicio

Y si el usuario que intenta acceder tiene el ROL de admin será redirigido a la página de inicio.



SideBar

Desde la barra lateral se accede a todas las funcionalidades y páginas disponibles para el administrador. Se puede colapsar la barra lateral (Imagen 2).



Página de Productos

Products Page

[Add new product](#)

[Sort by Price \(Asc\)](#) [Sort by Price \(Desc\)](#)

Product Id	Image	Name	Description	Price	Quantity	View	Edit	Delete
1		Iphone 15 pro	Apple New Iphone 15 pro	210€	11			
2		Macbook pro	Apple macbook pro 15"	4199€	15			
3		Ipad Pro	Apple Ipad PRO	240€	4			
4		Apple Watch Ultra	Apple Watch Pro Ultra	240€	0			

Se pueden ver un único producto, editar y borrar productos desde esta página.

Añadir producto

Add New Product

Desde esta vista tambien se puede crear un producto.

[Go to products page](#)

[Add Product](#)

Editar Producto

Desde este componente se puede editar un producto insertando el ID o en caso de no saber el id, desde la página de Productos.

Edit Product

Product ID:

[Edit Product](#)

Si no sabes el ID del producto, tambien puedes editar el producto en la Página de Productos [Ir a la pagina de productos →](#)

Marcas | Brands

Esta es la vista de las marcas de productos. Se puede editar, crear y borrar una marca.

Brands

Apple	Edit	Delete
Samsung	Edit	Delete
HP	Edit	Delete
Huawei	Edit	Delete
Google	Edit	Delete

Añadir Nueva marca

Add Brand

Brand Name:

Add Brand

Categorías

Se pueden ver, modificar, añadir y borrar categorías desde esta vista.

Categories Page

Id	Name	Description	Created At	Operations		
				View Category	Edit	Delete
1	Smartphone	Mobile Phones		View Category	Edit	Delete
2	Tablet	Tablet		View Category	Edit	Delete
3	Smartwatch	Smartwatch		View Category	Edit	Delete
4	Computer	Computer		View Category	Edit	Delete
5	Laptop	Laptop		View Category	Edit	Delete
6	Monitor	Monitor		View Category	Edit	Delete
7	TV	Television		View Category	Edit	Delete
8	Virtual Glass	AR Glass		View Category	Edit	Delete
9	Accessories	accessories		View Category	Edit	Delete
10	Headphones	Audio and Headphones		View Category	Edit	Delete

Añadir nueva categoría

Add a new category

Add Category

Name:

Description:

Submit

Go back to category

Editar una categoría

Edit Category
Select Category:

Name:
Tablet

Description:
Tablet

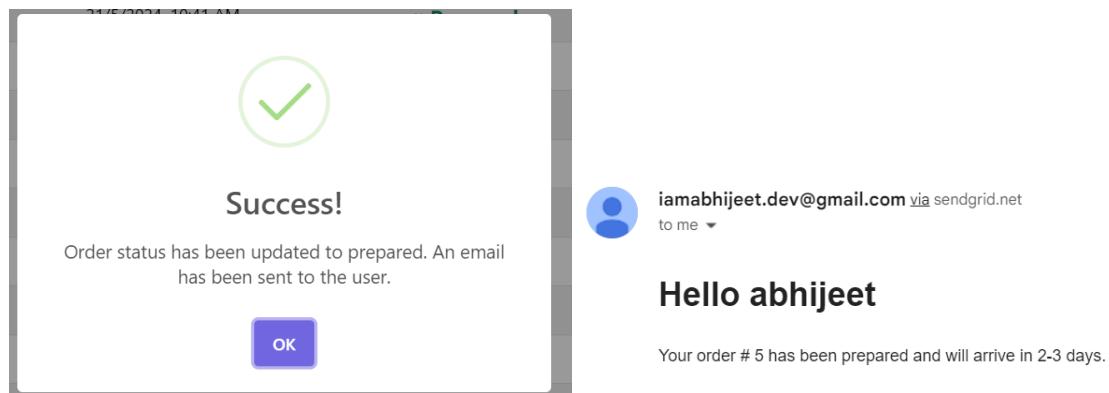
Submit

Pedidos

Orders

Order ID	User ID	Total Price	Created At	Status	Action
1	22	4439	20/5/2024, 4:46 PM	X Pending	Mark as Prepared
2	22	5258	21/5/2024, 10:04 AM	X Pending	Mark as Prepared
3	22	9838	21/5/2024, 10:41 AM	✓ Prepared	Mark as Unprepared
4	22	9838	21/5/2024, 10:41 AM	✗ Canceled	Mark as Prepared
5	22	9838	21/5/2024, 10:53 AM	X Pending	Mark as Prepared
6	22	9838	21/5/2024, 10:59 AM	X Pending	Mark as Prepared

Desde este panel se pueden ver todos los pedidos. Se puede marcar un pedido como preparado, cancelado etc. Al marcar el pedido como preparado se envía un correo de confirmación al usuario.



Inventario

Inventory					
Product ID	Image	Name	Available Units	Operations	
1		Iphone 15 pro	11	-	11 +
2		Macbook pro	15	-	15 +
3		Ipad Pro	4	-	4 +
4		Apple Watch Ultra	0	-	0 +

Desde esta vista se puede modificar el inventario de productos. Si quedan menos de 10 productos sale un aviso en color rojo de Inventario Bajo.

Usuarios | Clientes

User Management

ID	Email	First Name	Last Name
21	abhi@user.com	Abhijeet	singh
22	acc1abhi@gmail.com	abhijeet	singh
23	abhijeet3016@gmail.com	abhi	abhi
24	iamabhijeet2003@gmail.com	abhi	abhi

Desde esta vista se pueden ver todos los usuarios registrados.

Contacto

En esta vista se pueden ver todos los mensajes llenados desde el formulario de contacto en el Frontend de la tienda.

User Contact Form

Name	Email	Phone	Message
abhijeet	acc1abhi@gmail.com	1122222	test
abhijeet	acc1abhi@gmail.com	1122222	test
abhijeet	acc1abhi@gmail.com	125402554	test
abhijeet	acc1abhi@gmail.com	125402554	test
abhijeet	acc1abhi@gmail.com	125402554	test
abhijeet	acc1abhi@gmail.com	125402554	test

13. Despliegue

El despliegue del proyecto ha sido llevado a cabo utilizando varias plataformas para garantizar la disponibilidad, escalabilidad y rendimiento óptimo de la aplicación tanto en el backend como en el frontend. A continuación, se detallan los procesos y las herramientas utilizadas para el despliegue:

Backend

Para la **API** del proyecto:

- **Máquina Virtual en Azure:** Se creó una VM en Azure con Linux para ejecutar la API de Symfony, optimizando rendimiento y seguridad.
- **Servidor Web:** Se instaló y configuró Apache y PHP para la ejecución de Symfony.
- **Balanceador de Carga:** Azure Load Balancer distribuyó eficientemente las solicitudes entre múltiples instancias de la API.

Frontend

Para el Frontend y el Admin:

- **Vercel:** Se desplegaron el frontend y el admin en Vercel, con integración continua desde repositorios de GitHub.
- **Integración Continua:** Se configuró la integración continua para desplegar automáticamente las últimas versiones del frontend y admin.
- **Configuración de Entorno:** Se establecieron variables de entorno en Vercel para la comunicación con la API en Azure.
- **Pruebas y Verificación:** Se realizaron pruebas de rendimiento y seguridad para asegurar el funcionamiento correcto del frontend y admin.

Este enfoque combinado utilizando Azure para el backend y Vercel para el frontend asegura una solución robusta y eficiente, proporcionando una excelente experiencia de usuario.

14. Conclusiones y Mejoras Futuras

14.1 Conclusiones

El desarrollo de este proyecto ha representado un esfuerzo significativo y una experiencia de aprendizaje profunda. Desde la conceptualización hasta la implementación, cada paso ha ofrecido retos y oportunidades para crecer en habilidades técnicas y de gestión de proyectos. Estoy muy satisfecho con los resultados obtenidos y con el conocimiento adquirido a lo largo de este proceso. La creación de una tienda online completa, con funcionalidades de administración y un sistema de autenticación robusto, ha demostrado ser una tarea compleja pero gratificante. Este proyecto no solo ha mejorado mis capacidades como desarrollador, sino que también ha reforzado mi habilidad para resolver problemas y adaptarme a nuevas tecnologías y metodologías.

14.2 Mejoras Futuras

Para seguir mejorando y expandiendo la funcionalidad de la aplicación, se han identificado varias mejoras y nuevas características que podrían implementarse en el futuro:

1. Autenticación OAuth:

- Integrar sistemas de autenticación mediante OAuth para permitir a los usuarios iniciar sesión utilizando sus cuentas de Google, GitHub, Facebook, Apple, etc. Esto mejorará la experiencia de usuario al proporcionar múltiples opciones de autenticación.

2. Chatbot en Tiempo Real:

- Implementar un chatbot usando WebSocket para proporcionar atención al cliente en tiempo real. Esto permitirá a los usuarios obtener respuestas rápidas a sus preguntas y solucionar problemas de manera más eficiente.

3. IA para Preguntas Frecuentes:

- Desarrollar un modelo de Inteligencia Artificial utilizando la API de OpenAI para responder a preguntas frecuentes y resolver dudas de los clientes. Esto mejorará la experiencia del usuario al proporcionar soporte automatizado y eficiente.

4. Productos en Realidad Aumentada (AR):

- Implementar la funcionalidad de productos en realidad aumentada (AR) para que los clientes puedan visualizar los productos en su entorno

antes de realizar una compra. Esto aumentará el atractivo de la tienda y mejorará la experiencia de compra.

5. **Lista de Deseos (Wishlist):**

- Añadir una funcionalidad de lista de deseos para que los usuarios puedan guardar productos que les interesen para comprarlos en el futuro. Esto mejorará la retención de clientes y fomentará compras futuras.

6. **Aplicación Móvil Nativa:**

- Crear una aplicación móvil nativa para iOS y Android, ofreciendo a los usuarios una experiencia más optimizada y accesible desde dispositivos móviles. Esto ampliará el alcance de la tienda y facilitará el acceso desde cualquier lugar.

7. **Funcionalidades Avanzadas:**

- **Sistema de Recomendaciones Personalizadas:** Implementar un sistema de recomendaciones personalizadas basado en el comportamiento de los usuarios y sus preferencias.
- **Análisis de Datos y Reportes Avanzados:** Desarrollar herramientas de análisis de datos y generación de reportes avanzados para que los administradores puedan tomar decisiones informadas basadas en métricas de rendimiento.
- **Integración de Pagos con Criptomonedas:** Ofrecer opciones de pago con criptomonedas para atraer a una base de usuarios más amplia y moderna.

Estas mejoras no solo añadirán valor a la aplicación, sino que también la mantendrán competitiva y alineada con las tendencias tecnológicas actuales. Continuar trabajando en estas mejoras futuras garantizará que la aplicación siga evolucionando y proporcionando una experiencia de usuario excepcional.

15. Bibliografía

- <https://symfony.com/>
- <https://vuejs.org/>
- <https://learn.microsoft.com/en-us/azure/?product=popular>
- <https://api-platform.com/>
- <https://jwt.io/>
- <https://stripe.com/es>
- <https://sendgrid.com/>
- <https://router.vuejs.org/>
- <https://vuex.vuejs.org/>
- <https://getbootstrap.com/>
- <https://icons.getbootstrap.com/>
- <https://tailwindcss.com/>
- <https://developers.google.com/maps>
- <https://fontawesome.com/v4/>
- <https://axios-http.com>
- <https://ui.vuestic.dev/>
- <https://github.com/>
- <https://git-scm.com/>
- <https://www.docker.com/>
- <https://platform.sh/>
- <https://www.postman.com/>
- <https://www.figma.com/>
- <https://symfony.com/bundles/LexikJWTAuthenticationBundle/current/index.html>
- <https://momentjs.com/>
- <https://vee-validate.logaretm.com/v4/>
- <https://stackoverflow.com/>
- <https://apexcharts.com/>
- <https://sweetalert2.github.io/>
- <https://www.npmjs.com/package/vue3-toastify>
- <https://symfony.com/doc/current/security.html>
- <https://www.mysql.com/>
- <https://developer.mozilla.org/es/docs/Web/JavaScript>
- <https://www.digitalocean.com/community/tutorials>
- <https://buefy.org/documentation>
- <https://openwebinars.net>