



I.E.S. "ÁGORA"

Dpto.: Tu departamento

Ciclo Formativo de G. Superior: Desarrollo de aplicaciones web



Autor o autores: Aitor Bartolomé  
Puertas Tutor: Lorena Granado Garcia

Cáceres, a 31 de Mayo de 2025

<b>3. Introducción.....</b>	<b>4</b>
<b>4. Objetivos del proyecto.....</b>	<b>5</b>
Objetivo general.....	5
Objetivos específicos.....	5
<b>5. Justificación y motivación.....</b>	<b>6</b>
<b>6. Descripción general de la aplicación.....</b>	<b>7</b>
Funcionalidades para el usuario cliente.....	7
Funcionalidades para el usuario administrador.....	8
Dispositivos compatibles.....	8
<b>7. Análisis y diseño.....</b>	<b>8</b>
7.1 Casos de uso.....	9
Descripción de los principales casos de uso:.....	10
7.2 Diagrama entidad-relación.....	10
7.3 Diagrama de secuencia.....	12
Flujo simplificado de compra:.....	12
<b>8. Tecnologías utilizadas.....</b>	<b>12</b>
Frontend.....	12
Backend.....	13
Base de datos.....	14
Despliegue y entorno.....	15
<b>9. Arquitectura del sistema.....</b>	<b>16</b>
9.1 Estructura del backend.....	16
9.2 Estructura del frontend.....	17
9.3 Conexión con la base de datos.....	18
<b>10. Desarrollo del proyecto.....</b>	<b>19</b>
10.1 Backend.....	19
Registro y autenticación.....	19
Gestión de usuarios.....	19
Gestión de productos y categorías.....	20
Gestión de pedidos.....	20
Controladores y respuestas.....	21
Validaciones y manejo de errores.....	21
<b>10.2 Frontend.....</b>	<b>21</b>
Estructura general.....	21
Componentes clave.....	22
Navegación y rutas.....	23
Comunicación con el backend.....	23
Interfaz y experiencia de usuario.....	23

<b>11. Pruebas realizadas.....</b>	<b>24</b>
11.1 Pruebas del backend.....	24
a) Pruebas unitarias e integración.....	24
b) Pruebas funcionales con Postman.....	25
11.2 Pruebas del frontend.....	25
a) Pruebas visuales y de interacción.....	25
b) Pruebas de navegación.....	26
c) Pruebas responsive y dispositivos.....	26
<b>12. Despliegue.....</b>	<b>27</b>
12.1 Despliegue del backend.....	27
Pasos realizados:.....	27
12.2 Despliegue del frontend.....	28
Pasos realizados:.....	28
12.3 Base de datos en Railway.....	28
Pasos realizados:.....	28
12.4 Seguridad y entorno de producción.....	29
<b>13. Resultados obtenidos.....</b>	<b>29</b>
Resultados técnicos.....	30
Resultados funcionales.....	30
Experiencia de usuario.....	31
Cumplimiento de objetivos.....	31
<b>14. Futuras mejoras.....</b>	<b>31</b>
Mejoras técnicas.....	32
Nuevas funcionalidades.....	32
Mejoras en la experiencia de usuario (UX/UI).....	33
<b>15. Conclusiones.....</b>	<b>33</b>
<b>Anexo A. Manual técnico.....</b>	<b>34</b>
<b>Anexo B. Manual de usuario.....</b>	<b>40</b>
B.1 Acceso a la aplicación.....	40
B.2 Registro de usuario.....	41
B.3 Inicio de sesión.....	42
B.4 Navegación general.....	43
B.5 Visualización de productos.....	44
B.6 Carrito de compra.....	44
B.7 Perfil de usuario.....	46
B.8 Panel de administrador.....	47
Gestión de productos.....	47
Gestión de usuarios.....	48
Gestión de pedidos.....	49
B.9 Cierre de sesión.....	49
B.10 Compatibilidad y accesibilidad.....	50

### 3. Introducción

El propósito principal de este proyecto es el desarrollo de una plataforma web de comercio electrónico centrada en la venta de productos tecnológicos. Esta aplicación permite a los usuarios navegar por un catálogo de productos, añadir artículos a un carrito de compra, gestionar pedidos, y consultar su historial de compras. Además, incorpora un panel de administración para gestionar el inventario, los usuarios y los pedidos registrados en el sistema.

El desarrollo del proyecto ha sido realizado siguiendo los principios de arquitectura en capas, utilizando tecnologías modernas tanto en el frontend como en el backend, y desplegando la aplicación completa en la nube. El objetivo final es ofrecer una solución profesional, robusta y extensible, aplicando todos los conocimientos adquiridos a lo largo del ciclo formativo.

Este documento tiene como finalidad explicar detalladamente el proceso de análisis, diseño, implementación, pruebas y despliegue del sistema. Además, se incluyen anexos con manuales técnico y de usuario, que permiten tanto la instalación como la utilización completa de la plataforma.

AITStore está diseñado para ser accesible desde cualquier dispositivo con conexión a Internet, adaptándose a ordenadores, tablets y móviles gracias a su diseño responsivo y moderno.



#### 4. Objetivos del proyecto

El objetivo general del proyecto **AITStore** es desarrollar una aplicación web funcional y moderna que permita la compraventa de productos tecnológicos a través de una plataforma online. Esta solución integra tanto funcionalidades para usuarios finales como herramientas de administración interna, cubriendo así las necesidades de gestión y operación de una tienda virtual.

##### Objetivo general

Desarrollar una plataforma de comercio electrónico completa, segura y accesible, que permita a los usuarios realizar compras en línea y a los administradores gestionar de forma eficiente el catálogo de productos, los pedidos y los usuarios registrados.

##### Objetivos específicos

- **Diseñar y construir una API REST** segura y bien estructurada utilizando **Spring Boot** como backend.
- **Implementar una interfaz de usuario moderna** y responsiva utilizando **Angular 19** y **Tailwind CSS**.
- **Aplicar autenticación y autorización con JWT**, diferenciando el acceso por roles (CLIENTE y ADMIN).
- **Permitir la gestión de productos, categorías, pedidos y usuarios** desde un panel de administración.
- **Facilitar al usuario final la navegación, búsqueda y compra de productos** de forma intuitiva.
- **Integrar un sistema de carrito de compras persistente** vinculado al usuario autenticado.
- **Diseñar una base de datos relacional** en MySQL estructurada, escalable y optimizada.

- **Realizar el despliegue completo en la nube** utilizando Render (backend y frontend) y Railway (base de datos).
- **Aplicar buenas prácticas de desarrollo**, modularidad, pruebas y separación de responsabilidades.
- **Elaborar documentación técnica y manuales** de uso para facilitar el mantenimiento y la evolución futura del sistema.

## 5. Justificación y motivación

El proyecto **AITStore** nace como una propuesta personal con el objetivo de aplicar de forma práctica y real todos los conocimientos adquiridos a lo largo del ciclo formativo de Desarrollo de Aplicaciones Web. La elección de desarrollar una tienda online no es casual, sino que responde tanto a una motivación académica como profesional.

Por un lado, el comercio electrónico representa actualmente uno de los sectores más dinámicos y en crecimiento dentro del desarrollo web. Esta realidad lo convierte en un contexto ideal para integrar múltiples tecnologías y competencias del ciclo, como el desarrollo full-stack, el diseño de interfaces modernas, la gestión de datos, la seguridad de aplicaciones y el despliegue en la nube.

Por otro lado, desde el punto de vista personal, se trata de un proyecto que permite enfrentarse a problemas reales como la autenticación de usuarios, la persistencia de datos, la protección de rutas, la gestión de productos en tiempo real, la validación de formularios y la adaptación a diferentes dispositivos. Todo ello supone una experiencia muy enriquecedora para consolidar el aprendizaje y prepararse para el entorno laboral.

La decisión de utilizar tecnologías como **Spring Boot, Angular 19, JWT, MySQL, Tailwind CSS** y plataformas de despliegue como **Render o Railway** responde al deseo de trabajar con herramientas actuales y demandadas en el mercado profesional, simular un entorno real de desarrollo y entregar un producto completo, funcional, y preparado para ser ampliado o mantenido en el futuro.

Además, esta elección permite demostrar dominio en todas las fases del desarrollo de una aplicación: análisis, diseño, implementación, pruebas, despliegue, documentación y usabilidad.

## 6. Descripción general de la aplicación

**AITStore** es una aplicación web de comercio electrónico orientada a la venta de productos tecnológicos. La plataforma ha sido diseñada con un enfoque centrado en el usuario final, facilitando la navegación, búsqueda y compra de productos desde cualquier dispositivo, así como con funcionalidades de gestión para administradores.

La aplicación distingue claramente entre dos tipos de usuarios: **clientes** y **administradores**, cada uno con funcionalidades específicas. Mientras que los clientes pueden explorar el catálogo, añadir productos al carrito y realizar pedidos, los administradores disponen de un panel desde el que pueden gestionar el inventario, los usuarios y los pedidos.

Gracias a una arquitectura modular y bien estructurada, la aplicación garantiza una experiencia fluida, rápida y segura, con un diseño responsive y un sistema de autenticación robusto basado en tokens JWT.

#### **Funcionalidades para el usuario cliente**

- Registro y autenticación segura mediante JWT
- Consulta del catálogo de productos por categoría
- Búsqueda de productos por nombre
- Visualización de detalles del producto
- Añadir productos al carrito
- Modificación de cantidades y eliminación de productos del carrito
- Finalización de pedidos
- Consulta del historial de pedidos
- Visualización y edición de datos del perfil personal

#### **Funcionalidades para el usuario administrador**

- Gestión de productos: creación, edición y eliminación

- Gestión de usuarios registrados
- Consulta de todos los pedidos realizados
- Acceso restringido mediante rol ADMIN
- Interfaz administrativa diferenciada visualmente

### **Dispositivos compatibles**

La aplicación está desarrollada con diseño responsive, lo que permite su uso en:

- Ordenadores de escritorio y portátiles
- Tablets
- Teléfonos móviles

El diseño se adapta automáticamente al tamaño de pantalla del dispositivo, manteniendo todas las funcionalidades disponibles y con una experiencia visual coherente.

## **7. Análisis y diseño**

En esta fase del proyecto se definieron las funcionalidades principales del sistema y su estructura de datos, utilizando diagramas que permiten visualizar de forma clara las relaciones entre los distintos actores y componentes del sistema.

Los tres aspectos principales analizados fueron:

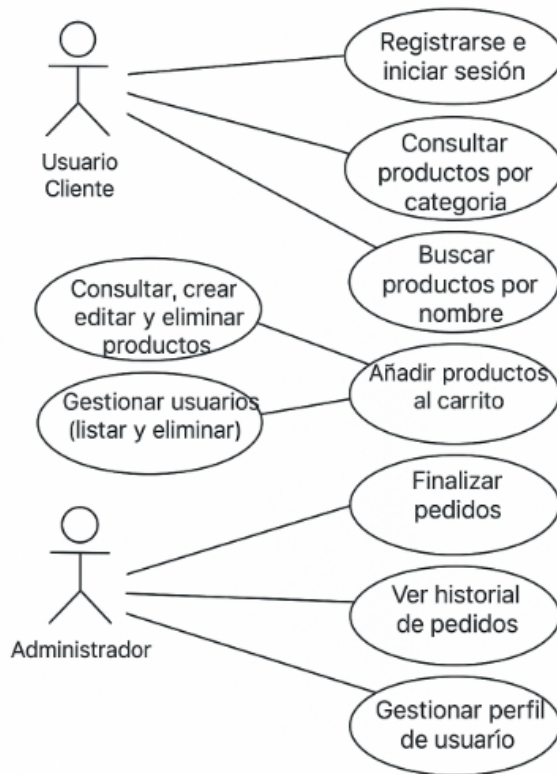
- Las **funcionalidades esperadas** según el tipo de usuario (casos de uso)
- La **estructura de la base de datos** y sus relaciones (entidad-relación)
- El **flujo de interacción** entre componentes clave del sistema (diagrama de secuencia)

### **7.1 Casos de uso**



Se identificaron dos actores principales:

- **Usuario cliente:** consumidor final de la tienda online
- **Administrador:** encargado de gestionar el contenido y usuarios del sistema



### Descripción de los principales casos de uso:

#### Usuario cliente:

- Registrarse
- Iniciar sesión
- Consultar productos
- Buscar productos por nombre
- Filtrar productos por categoría
- Añadir productos al carrito

- Modificar o eliminar productos del carrito
- Finalizar un pedido
- Consultar historial de pedidos
- Ver y editar datos del perfil

**Administrador:**

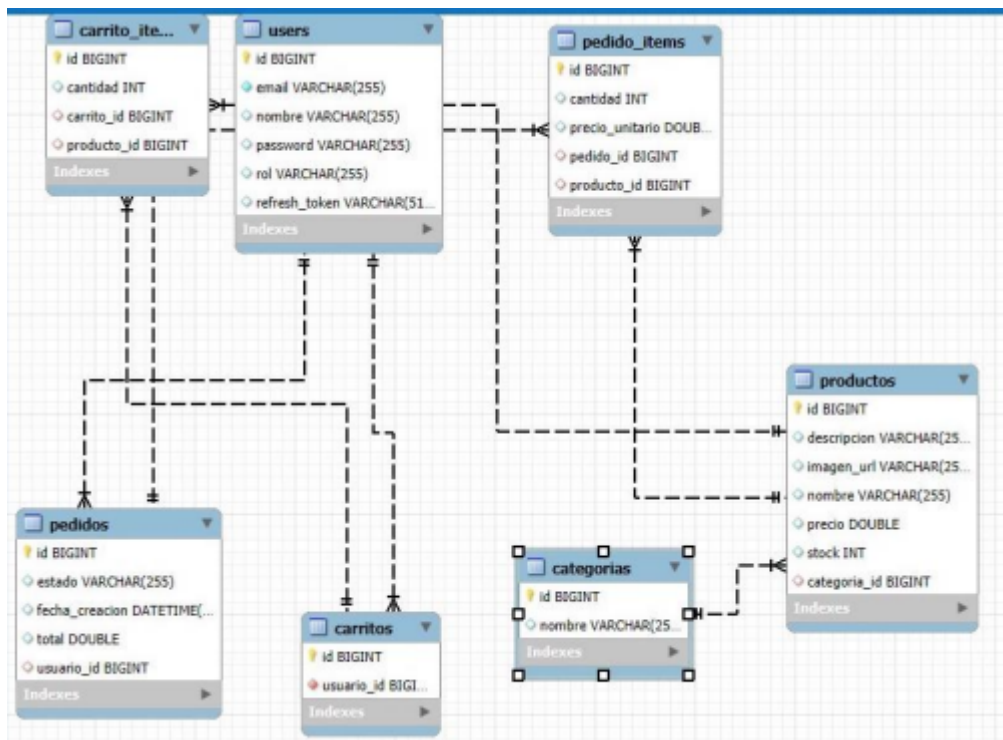
- Acceder al panel de administración
- Crear, editar y eliminar productos
- Ver listado de usuarios
- Eliminar usuarios
- Consultar todos los pedidos

**7.2 Diagrama entidad-relación**

La base de datos está diseñada con un enfoque relacional utilizando **MySQL**, y modelada mediante entidades JPA en el backend. Las tablas principales incluyen:

- User (usuarios)
- Product (productos)
- Category (categorías de producto)
- Order (pedidos)

- OrderItem (detalle de cada producto en el pedido)



Relaciones clave:

- Un User puede tener muchos Order
- Un Order contiene muchos OrderItem
- Cada OrderItem se vincula a un Product
- Un Product pertenece a una única Category
- Las contraseñas en User se almacenan cifradas

### 7.3 Diagrama de secuencia

Se ha representado también un **diagrama de secuencia** para mostrar un flujo representativo del sistema: el proceso completo de compra.

**Flujo simplificado de compra:**

1. El cliente inicia sesión.
2. Selecciona productos y los añade al carrito.
3. Confirma el pedido.
4. El frontend envía el pedido al backend.
5. El backend valida stock, genera el pedido y descuenta cantidades.
6. Se guarda el pedido en la base de datos.
7. El backend responde con confirmación.

## 8. Tecnologías utilizadas

El proyecto **AITStore** ha sido desarrollado utilizando tecnologías modernas y ampliamente adoptadas en el desarrollo web profesional. La elección de cada herramienta se ha realizado en base a su rendimiento, comunidad, escalabilidad y adecuación a los objetivos del proyecto.

A continuación se detallan las principales tecnologías utilizadas, clasificadas por capa:

### Frontend

Tecnología	Descripción y uso en el proyecto
<b>Angular 19</b>	Framework para construir SPA (Single Page Applications). Utilizado para estructurar la interfaz cliente con enrutamiento, componentes reutilizables y servicios.
<b>TypeScript</b>	Lenguaje tipado que mejora el desarrollo seguro y mantenible sobre JavaScript.

<b>Tailwind CSS</b>	Framework de diseño basado en clases utilitarias. Permite un diseño rápido, responsivo y coherente.
<b>Sweetalert2</b>	Librería para mostrar alertas visuales y modales de confirmación con un diseño atractivo.
<b>Angular Animations</b>	Módulo para aplicar transiciones suaves entre vistas, carga de datos y componentes.
<b>LocalStorage</b>	Utilizado para almacenar el token JWT del usuario y mantener su sesión activa entre recargas.

## Backend

<b>Tecnología</b>	<b>Descripción y uso en el proyecto</b>
<b>Spring Boot 3</b>	Framework para crear aplicaciones web con Java. Facilita la construcción de APIs REST robustas.
<b>Java 17</b>	Lenguaje principal para el desarrollo del backend. Ofrece rendimiento, estabilidad y madurez.
<b>Spring Security</b>	Proporciona autenticación y autorización mediante tokens JWT. Protege los endpoints según el rol.
<b>JWT (JSON Web Token)</b>	Sistema de autenticación basado en tokens firmados, sin necesidad de mantener sesiones en servidor.

<b>Spring Data JPA</b>	Permite interactuar con la base de datos de forma abstracta a través de repositorios.
<b>Hibernate</b>	ORM utilizado por JPA para mapear entidades a tablas relacionales.
<b>MapStruct</b>	Framework de mapeo entre entidades y DTOs. Automatiza conversiones y mejora la limpieza del código.
<b>Lombok</b>	Reduce el código repetitivo en Java (getters, setters, constructores...) con anotaciones.
<b>Swagger / OpenAPI</b>	Documentación interactiva de los endpoints REST y pruebas de la API desde el navegador.

## Base de datos

<b>Tecnología</b>	<b>Descripción</b>
<b>MySQL</b>	Sistema de gestión de base de datos relacional. Utilizado para persistir usuarios, productos, pedidos, etc.
<b>MySQL Workbench</b>	Herramienta gráfica para modelar y consultar la base de datos durante el desarrollo.

## Despliegue y entorno

Herramienta	Descripción
<b>Render</b>	Plataforma cloud para desplegar el frontend y el backend de forma gratuita y escalable.
<b>Railway</b>	Plataforma para desplegar y gestionar la base de datos MySQL online.
<b>Docker (opcional)</b>	Usado localmente para facilitar el desarrollo y pruebas en entornos aislados.
<b>Postman</b>	Herramienta para probar peticiones HTTP hacia el backend durante el desarrollo y testing.
<b>Git + GitHub</b>	Control de versiones del código fuente, alojado en repositorio remoto para despliegue.

El uso conjunto de estas tecnologías ha permitido construir un sistema sólido, moderno y con una arquitectura profesional, totalmente desplegable en la nube y fácilmente mantenible.

## 9. Arquitectura del sistema

La aplicación **AITStore** sigue una arquitectura en capas y basada en el patrón cliente-servidor. La división de responsabilidades entre frontend, backend y base de datos permite una mayor escalabilidad, mantenibilidad y claridad en el desarrollo del sistema.

### 9.1 Estructura del backend

El backend está desarrollado con **Spring Boot**, y estructurado siguiendo una arquitectura en capas, separando claramente las responsabilidades:

com.aitorbp.aitstore

- |— auth      # Controladores y lógica de login/registro
- |— controller # Puntos de entrada de la API (endpoints REST)
- |— dto      # Objetos de transferencia de datos (entrada/salida)
- |— entity    # Entidades JPA que representan las tablas de la base de datos
- |— mapper    # Conversores entre entidades y DTOs (usando MapStruct)
- |— repository # Interfaces JPA que acceden a la base de datos
- |— service    # Interfaces de lógica de negocio
- |— service.impl # Implementación de los servicios
- |— security   # Configuración JWT, filtros, autenticación y roles
- |— config    # Beans, Swagger, seguridad, CORS
- |— AitstoreApplication.java

#### Flujo de una petición típica (por ejemplo, crear pedido):

1. El cliente envía una solicitud HTTP desde Angular.
2. El controller recibe la petición y la delega al service.
3. El service ejecuta la lógica de negocio y accede a los datos a través del repository.
4. Se responde con un DTO convertido por un mapper.



## 9.2 Estructura del frontend

El frontend está construido con **Angular 19** y se organiza de forma modular mediante componentes standalone, lo que mejora la carga de recursos, la reutilización y la claridad estructural.

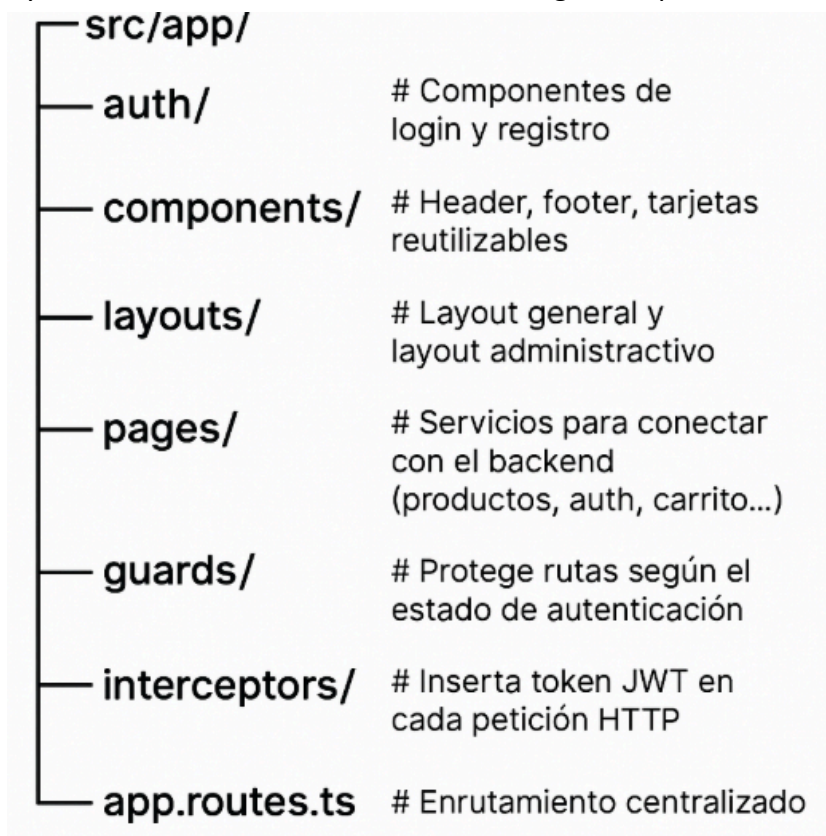
src/app/

— auth/	# Componentes de login y registro
— components/	# Header, footer, tarjetas reutilizables
— layouts/	# Layout general y layout administrativo
— pages/	# Home, perfil, carrito, pedidos, administración...
— services/	# Servicios para conectar con el backend (productos, auth, carrito...)
— guards/	# Protege rutas según el estado de autenticación
— interceptors/	# Inserta token JWT en cada petición HTTP
— app.routes.ts	# Enrutamiento centralizado

Cada parte cumple una función específica:

- pages/ agrupa vistas de alto nivel
- components/ contiene elementos reutilizables
- services/ se comunican con la API

- layouts/ define la estructura visual base según el tipo de usuario



### 9.3 Conexión con la base de datos

La base de datos utilizada es **MySQL**, desplegada en **Railway**. La conexión con ella se realiza desde el backend a través de **Spring Data JPA**.

- Las credenciales están definidas en `application-prod.yml` y gestionadas como variables de entorno en Render.
- Hibernate se encarga de generar las tablas automáticamente a partir de las entidades (`ddl-auto=update`).
- Las relaciones entre entidades están definidas mediante anotaciones JPA (`@OneToMany`, `@ManyToOne`, `@JoinColumn`, etc.)

#### Relaciones clave del modelo:

- Un usuario (User) puede tener muchos pedidos (Order)
- Un pedido contiene múltiples elementos (OrderItem)

- Un producto pertenece a una única categoría (Category)
- Los pedidos se relacionan con los productos a través de los OrderItem

## 10. Desarrollo del proyecto

### 10.1 Backend

El backend de **AITStore** se ha construido como una API REST robusta, segura y bien estructurada, utilizando el framework **Spring Boot 3**, con Java 17 y acceso a base de datos relacional a través de JPA/Hibernate. La API expone todos los recursos necesarios para que el frontend pueda interactuar con el sistema: usuarios, productos, categorías, pedidos, carrito, etc.

#### Registro y autenticación

El sistema implementa un mecanismo de autenticación basado en **JWT (JSON Web Token)**. Cuando un usuario se registra o inicia sesión correctamente, el backend genera un token firmado que contiene su identificador y su rol (CLIENTE o ADMIN).

- El token se devuelve al frontend y se almacena en el localStorage.
- Cada vez que el frontend hace una petición protegida, el token se adjunta en el encabezado Authorization.
- El backend valida el token automáticamente y concede o deniega el acceso según el rol.

La seguridad está centralizada en el paquete security/, que contiene filtros, configuraciones y utilidades para la autenticación.

#### Gestión de usuarios

Los usuarios se registran a través del endpoint `/api/auth/registro`, con validaciones en los DTOs. Los administradores pueden listar y eliminar usuarios desde la ruta protegida `/api/usuarios`.

- Contraseñas almacenadas cifradas con `BCryptPasswordEncoder`
- Validación de email único
- Asignación automática del rol `CLIENTE` al registrarse

### **Gestión de productos y categorías**

Los productos están organizados por categorías, y se pueden consultar mediante endpoints públicos (`/api/productos`, `/api/categorias`). Un administrador puede crear, editar o eliminar productos desde `/api/productos`.

- Cada producto incluye nombre, descripción, precio, imagen, stock y categoría
- Las categorías están gestionadas desde `/api/categorias`
- Uso de DTOs y `MapStruct` para separar lógica interna y datos de presentación

### **Gestión de pedidos**

Cuando un usuario finaliza la compra, se genera un pedido completo desde el carrito. El pedido se guarda en la base de datos y se vincula al usuario.

- Endpoint: `POST /api/pedidos`
- Verificación automática del stock de cada producto
- Descuento de stock en base a la cantidad pedida
- Eliminación del producto si su stock llega a cero
- Visualización de pedidos por usuario (`/api/pedidos/usuario/{id}`)

La lógica se encuentra centralizada en `OrderServiceImpl`, aplicando validaciones y control de errores personalizados (`StockInsuficienteException`, por ejemplo).

### Controladores y respuestas

Cada entidad tiene su propio controlador (`ProductController`, `UserController`, `OrderController`, etc.), y todos los endpoints siguen un esquema RESTful.

- Las rutas públicas están documentadas en Swagger.
- Las rutas protegidas por rol ADMIN están validadas mediante anotaciones `@PreAuthorize`.
- Se utilizan códigos HTTP estándar (200, 201, 400, 403, 404, 500).

### Validaciones y manejo de errores

Se aplican validaciones con anotaciones JSR-380 (`@NotNull`, `@Email`, `@Size`, etc.) y manejo global de excepciones con `@ControllerAdvice`.

- Respuestas claras en caso de errores de validación o autenticación
- Estructura de error uniforme: timestamp, mensaje, estado

## 10.2 Frontend

El frontend de **AITStore** se ha desarrollado utilizando **Angular 19**, un framework moderno y robusto para la construcción de aplicaciones SPA (Single Page Application). La aplicación permite a los usuarios interactuar con la tienda desde cualquier dispositivo mediante una interfaz clara, rápida y totalmente responsive.

Se ha priorizado la experiencia de usuario mediante animaciones suaves, alertas visuales (`SweetAlert2`), modo oscuro, buscador dinámico y navegación protegida según el rol del usuario.

### Estructura general

El frontend se ha organizado en módulos independientes basados en **componentes standalone**, lo que mejora la carga inicial, reduce dependencias innecesarias y permite una estructura más limpia.

src/app/

— auth/	# Login y registro de usuarios
— components/	# Header, footer, tarjetas de producto, buscador...
— layouts/	# Layout general y layout administrativo
— pages/	# Vistas: home, perfil, carrito, pedidos, panel admin...
— services/	# Comunicación con el backend
— guards/	# Protege rutas privadas según el rol
— interceptors/	# Añade token JWT a cada solicitud HTTP
— app.routes.ts	# Rutas del sistema

### Componentes clave

- **HomeComponent:** Página principal que muestra productos por categoría con scroll horizontal.
- **HeaderComponent:** Cabecera fija con logo, buscador, login/logout, avatar y menú hamburguesa.
- **CartComponent:** Muestra los productos añadidos, permite modificar cantidades o finalizar la compra.
- **PerfilComponent:** Datos del usuario autenticado y su historial de pedidos.
- **AdminProductosComponent / AdminUsuariosComponent:** Vistas exclusivas para el administrador, donde puede gestionar el catálogo y los usuarios.

Todos estos componentes están diseñados para funcionar tanto en escritorio como en móvil, con soporte para modo oscuro y animaciones visuales.

## Navegación y rutas

El sistema de rutas (`app.routes.ts`) separa claramente:

- Rutas públicas: `/`, `/login`, `/registro`, `/categoria/:id`
- Rutas protegidas (`AuthGuard`): `/perfil`, `/carrito`, `/pedidos`
- Rutas administrativas (`AdminLayoutComponent`): `/admin/productos`, `/admin/usuarios`

El `AuthGuard` impide el acceso a rutas protegidas si el usuario no está autenticado, y redirige al login si es necesario.

## Comunicación con el backend

La comunicación se realiza mediante servicios Angular (`HttpClient`), centralizando toda la lógica de negocio en clases reutilizables:

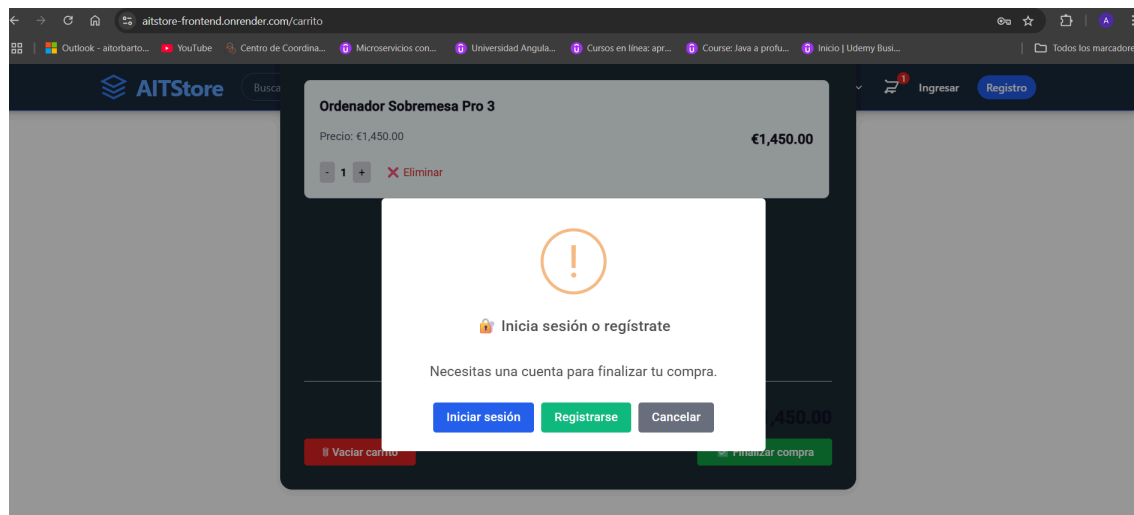
- `AuthService`: login, registro, comprobación del rol, gestión del token
- `ProductService`: obtención de productos, búsqueda, filtrado
- `CarritoService`: lógica del carrito, productos, cantidades, vaciado
- `PedidoService`, `CategoriaService`, `UsuarioService`: servicios complementarios

Un interceptor (`auth.interceptor.ts`) inyecta automáticamente el token JWT en cada petición HTTP.

## Interfaz y experiencia de usuario

- **Tailwind CSS** se ha utilizado como sistema de estilos: ligero, utilitario y 100% responsive.

- Se ha aplicado un **modo oscuro integrado**, que se conserva entre sesiones.
- Se han incluido **SweetAlert2** y **Angular Animations** para mejorar la interacción y mostrar confirmaciones visuales, errores y acciones.
- La interfaz es totalmente adaptativa y usable desde móvil, tablet o escritorio.



Con esta implementación, el frontend ofrece una experiencia de usuario clara, rápida y accesible desde cualquier dispositivo, garantizando una comunicación fluida y segura con el backend.

## 11. Pruebas realizadas

Para garantizar el correcto funcionamiento del sistema **AITStore**, se han realizado pruebas tanto a nivel de servidor (backend) como de cliente (frontend). Estas pruebas permitieron validar la lógica de negocio, la seguridad, la experiencia de usuario y la integración entre componentes.

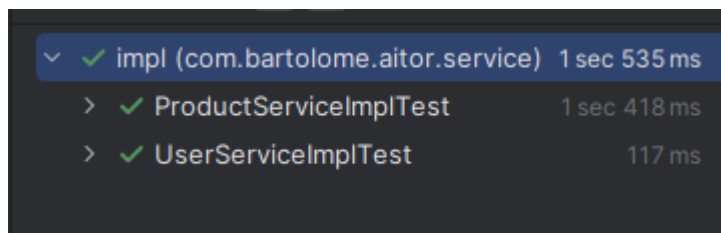
### 11.1 Pruebas del backend

El backend fue sometido a pruebas **unitarias**, **de integración** y **funcionales**, utilizando herramientas como **JUnit 5**, **Mockito**, **Postman** y **Swagger UI**.

#### a) Pruebas unitarias e integración



- Se probaron métodos clave de servicios (ProductService, OrderService, UserService) aislando dependencias con Mockito.
- Se validó la creación de pedidos, control de stock y eliminación automática de productos sin stock.
- También se probaron flujos de autenticación y acceso por roles.
- Las pruebas se ejecutaron en local usando @SpringBootTest y @WebMvcTest.



#### b) Pruebas funcionales con Postman

- Se creó una colección de endpoints en **Postman** para verificar todos los flujos:
  - Registro y login
  - Crear y consultar pedidos
  - Añadir, editar y borrar productos
  - Acceso denegado sin token o con rol incorrecto
- Se probaron respuestas correctas y casos de error (404, 403, 400).

### 11.2 Pruebas del frontend

Se realizaron **pruebas manuales exhaustivas** en diferentes dispositivos y navegadores para garantizar la correcta experiencia de usuario, validación de formularios, visualización y funcionamiento de todas las rutas.

#### a) Pruebas visuales y de interacción

- **Validación de formularios:**
  - Campos obligatorios, email válido, contraseñas coincidentes.
  - Errores mostrados al instante, confirmaciones con SweetAlert2.
- **Carrito de compra:**
  - Añadir, quitar, cambiar cantidad, vaciar, finalizar compra.
  - Validación de stock y mensajes de error si no hay disponibilidad.
- **Login y registro:**
  - Comprobación de rutas protegidas (perfil, carrito, admin)
  - Redirección automática tras autenticarse correctamente.

#### **b) Pruebas de navegación**

- Rutas públicas y privadas correctamente protegidas.
- Menú dinámico según si el usuario está autenticado o no.
- Comprobación del guardado y eliminación del token en localStorage.

#### **c) Pruebas responsive y dispositivos**

- Se probó la aplicación en:
  - Google Chrome, Mozilla Firefox
  - Modo móvil (emulador de navegador)
  - Dispositivos reales (teléfono y tablet)
- Todo el diseño se adaptó correctamente, incluyendo:

- Menú hamburguesa
- Layout admin
- Tarjetas de producto y botones

## 12. Despliegue

El sistema **AITStore** se encuentra desplegado completamente en la nube, accesible desde cualquier navegador web sin necesidad de instalación. El despliegue se ha realizado por separado para el backend, el frontend y la base de datos, empleando plataformas modernas que permiten alta disponibilidad y escalabilidad.

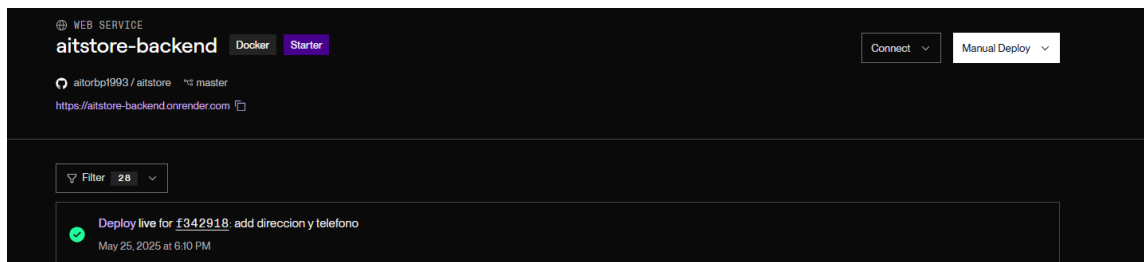
### 12.1 Despliegue del backend

El backend desarrollado con **Spring Boot** ha sido desplegado en la plataforma cloud **Render**, que permite ejecutar aplicaciones Java con facilidad y gestionar variables de entorno de forma segura.

#### Pasos realizados:

1. Se creó un repositorio en GitHub con el proyecto backend.
2. En Render se configuró un nuevo **Web Service** conectado al repositorio.
3. Parámetros establecidos:
  - **Build command:** ./mvnw clean install
  - **Start command:** java -jar target/aitstore-0.0.1-SNAPSHOT.jar
  - **Runtime:** Java 17
  - **Environment Variables:** DB\_URL, DB\_USER, DB\_PASSWORD, entre otras.
4. La base de datos externa (Railway) se conectó mediante JDBC y variables de entorno.

5. Render detecta cambios automáticamente y redepliega tras cada push a GitHub.

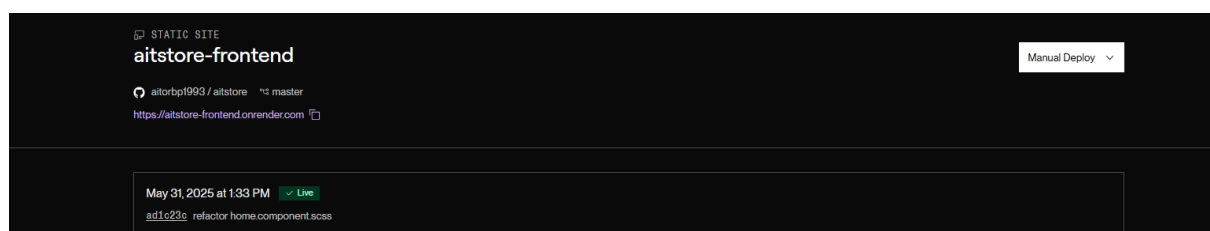


## 12.2 Despliegue del frontend

El frontend desarrollado con **Angular 19** ha sido compilado y desplegado en **Render** como **Static Site**.

### Pasos realizados:

1. Se ejecutó el comando `ng build --configuration=production` para generar la carpeta `dist/`.
2. En Render se creó un servicio tipo **Static Site** conectado al repositorio de GitHub del frontend.
3. Configuración:
  - **Build command:** `npm install && npm run build`
  - **Publish directory:** `dist/aitstore-frontend`
4. La URL pública generada permite acceder a la aplicación desde cualquier dispositivo.

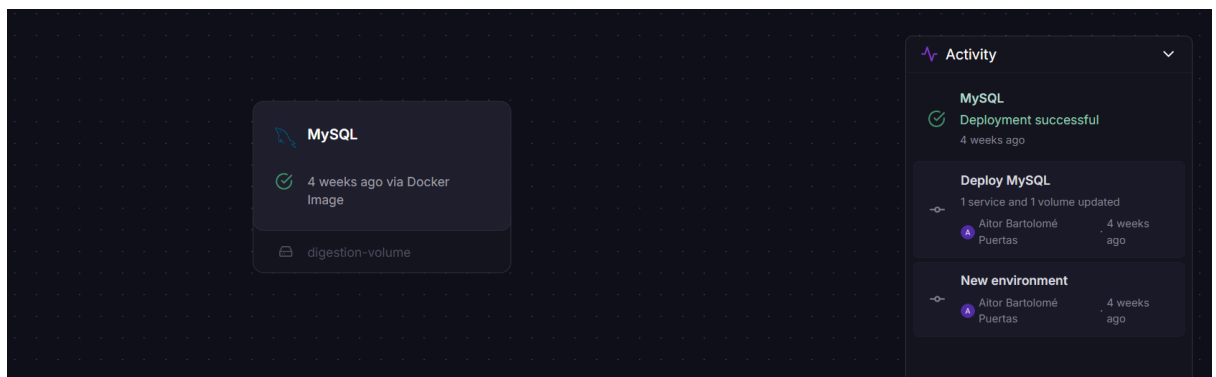


## 12.3 Base de datos en Railway

La base de datos **MySQL** ha sido desplegada en **Railway**, una plataforma que ofrece acceso persistente y seguro a bases de datos en la nube.

### Pasos realizados:

1. Se creó un nuevo proyecto en Railway con un contenedor MySQL.
2. Se generaron las credenciales de conexión: host, puerto, usuario, contraseña, y nombre de base de datos.
3. Estas credenciales se añadieron a Render como variables de entorno para que el backend pueda acceder a la base de datos.
4. Se permitió acceso externo para pruebas locales.
5. Railway ofrece una consola web para consultas SQL, backup, y visualización de datos.



## 12.4 Seguridad y entorno de producción

- Las credenciales están almacenadas como variables de entorno en Render, nunca embebidas en el código fuente.
- El backend está configurado con perfil prod y acceso solo desde orígenes permitidos (CORS).
- La base de datos está protegida con contraseña fuerte y acceso limitado.
- Todo el tráfico entre cliente y servidor se realiza a través de HTTPS.

Con esta estructura de despliegue distribuida y profesional, la aplicación queda lista para ser utilizada en un entorno real, accesible 24/7 y escalable.

## 13. Resultados obtenidos

El desarrollo de **AITStore** ha dado como resultado una plataforma de comercio electrónico completa, funcional y accesible, que cumple con todos los objetivos marcados al inicio del proyecto.

Tanto desde el punto de vista técnico como desde el punto de vista funcional, el sistema permite gestionar eficientemente el proceso de compraventa de productos tecnológicos, proporcionando una experiencia fluida al usuario final y herramientas sólidas para el administrador.

### Resultados técnicos

- Se ha implementado una **arquitectura por capas profesional**, con separación clara de responsabilidades en backend y frontend.
- El backend construido con **Spring Boot** expone una API REST segura, validada y documentada con Swagger.
- El frontend desarrollado con **Angular 19** es completamente responsive, soporta modo oscuro y se comunica eficientemente con la API.
- La aplicación ha sido **desplegada exitosamente en la nube**, utilizando Render para backend y frontend, y Railway para la base de datos.
- Se han seguido buenas prácticas de seguridad (JWT, cifrado de contraseñas, control de acceso por roles, manejo de errores).

### Resultados funcionales

- El sistema distingue correctamente entre **usuarios cliente y administradores**, ofreciendo funcionalidades adaptadas a cada perfil.
- Los **clientes** pueden:
  - Registrarse e iniciar sesión
  - Consultar productos por categoría

- Buscar productos por nombre
- Añadir productos al carrito y realizar pedidos
- Consultar historial de compras y modificar su perfil
- Los **administradores** pueden:
  - Gestionar productos (crear, editar, eliminar)
  - Gestionar usuarios registrados
  - Consultar pedidos realizados
  - Acceder a un panel exclusivo con layout separado

### Experiencia de usuario

- La interfaz es clara, moderna y visualmente atractiva gracias al uso de Tailwind CSS y SweetAlert2.
- La navegación es fluida y se adapta a cualquier dispositivo (escritorio, tablet, móvil).
- Se ofrece **retroalimentación inmediata al usuario** en todas las acciones (formularios, errores, confirmaciones).
- El modo oscuro mejora la accesibilidad y experiencia visual en diferentes entornos.

### Cumplimiento de objetivos

Todos los objetivos planteados al inicio del proyecto han sido alcanzados, incluyendo el diseño técnico, la implementación completa, el despliegue en la nube y la elaboración de documentación técnica y de usuario.

### 14. Futuras mejoras

Aunque el proyecto **AITStore** cumple con los requisitos funcionales y técnicos establecidos, se han identificado posibles mejoras que permitirían ampliar su funcionalidad, optimizar su rendimiento o aumentar su valor añadido en un entorno real de producción.

Estas mejoras se presentan clasificadas por tipo:

### **Mejoras técnicas**

- **Mejorar cobertura de testing**  
Ampliar las pruebas unitarias y de integración del backend utilizando JUnit y Mockito para asegurar el comportamiento de todos los servicios críticos.
- **Implementar paginación y ordenación en endpoints**  
Especialmente útil en el listado de productos, usuarios o pedidos cuando el número de registros aumente significativamente.
- **Alerta de stock mínimo**  
Añadir un sistema que notifique al administrador cuando el stock de un producto esté por debajo de un umbral definido.
- **Optimización de imágenes**  
Aplicar compresión y carga diferida (lazy loading) para mejorar el rendimiento de la aplicación en redes lentas.

### **Nuevas funcionalidades**

- **Pasarela de pago real**  
Integrar servicios como Stripe o PayPal para permitir transacciones reales desde la plataforma.
- **Seguimiento del pedido**  
Añadir estados de pedido (pendiente, enviado, entregado...) para que el cliente pueda seguir el proceso.
- **Sistema de valoraciones y reseñas**  
Permitir a los usuarios dejar opiniones sobre los productos que han comprado.



- **Gestión avanzada de usuarios**  
Añadir perfiles editables, roles adicionales (como moderador), o recuperación de contraseña por email.
- **Sistema de notificaciones por correo electrónico**  
Envío automático de confirmaciones de registro, pedido completado, cambios de estado, etc.

### Mejoras en la experiencia de usuario (UX/UI)

- **Buscador inteligente con autocompletado**  
Mostrar sugerencias a medida que el usuario escribe, mejorando la experiencia de búsqueda.
- **Modo multiidioma**  
Permitir al usuario cambiar el idioma de la interfaz (ej. español e inglés).
- **Personalización de la tienda**  
Añadir temas de color o personalización visual según preferencias del usuario.
- **Accesibilidad mejorada**  
Asegurar el cumplimiento de estándares de accesibilidad (WAI-ARIA, contraste, teclado...).

## 15. Conclusiones

El desarrollo del proyecto **AITStore** ha supuesto un reto completo que ha permitido aplicar de forma práctica y real todos los conocimientos adquiridos durante el ciclo formativo de **Desarrollo de Aplicaciones Web**. La realización de una plataforma funcional de comercio electrónico, desde cero y hasta su despliegue en la nube, ha aportado una visión global y realista del trabajo que implica construir una aplicación profesional.

En el transcurso del proyecto se han integrado herramientas y tecnologías actuales como **Angular, Spring Boot, JWT, MySQL, Tailwind CSS**, y se ha conseguido desplegar la aplicación de forma totalmente operativa mediante plataformas como **Render y Railway**.

Desde el punto de vista técnico, el proyecto ha cumplido con todos los objetivos marcados: arquitectura limpia, autenticación segura, API bien documentada, frontend responsivo y

funcional, y separación de roles de usuario. También se ha trabajado la validación de formularios, la experiencia de usuario y el diseño visual, aplicando buenas prácticas en todo el ciclo de vida del desarrollo.

Desde el punto de vista personal, el proyecto ha servido para reforzar competencias clave como la organización, la toma de decisiones técnicas, la resolución de problemas, el trabajo autónomo y la documentación profesional del trabajo realizado. Además, ha sido una oportunidad para enfrentarse a situaciones similares a las que se viven en un entorno laboral real.

Finalmente, se ha demostrado que es posible construir una aplicación web robusta, moderna y con potencial de crecimiento, completamente desarrollada, desplegada y documentada por una sola persona, aplicando los conocimientos adquiridos durante el ciclo y enfrentándose con éxito a los distintos desafíos técnicos y de diseño que han ido surgiendo.

## **Anexo A. Manual técnico**

Este manual está orientado a desarrolladores o técnicos que necesiten instalar, configurar o desplegar AITStore en su entorno local o en producción.

### **1. Requisitos previos**

#### **Backend**

- Java 17+
- Maven 3+
- MySQL 8+
- Docker (opcional)
- IDE: IntelliJ IDEA o Eclipse

#### **Frontend**

- Node.js v18+
- Angular CLI v19+
- Docker (opcional)
- IDE: VS Code recomendado

## 2. Instalación local

### Base de datos

1. Crear una base de datos MySQL llamada aitstore.
2. Al ejecutar el backend, Hibernate generará automáticamente las tablas.

### Backend (Spring Boot)

1. Clonar el repositorio:

```
git clone https://github.com/aitorbp1993/aitstore
```

Entrar en /backend y configurar application.properties:

```
spring.datasource.url=jdbc:mysql://localhost:3306/aitstore
```

```
spring.datasource.username=root
```

`spring.datasource.password=tu_contraseña`

2.

3. Ejecutar:

`mvn spring-boot:run`

4. Acceder a la API en:

`http://localhost:8081/swagger-ui/index.html`

Frontend (Angular)

Ir a /frontend:

`npm install`

`ng serve`

1. Abrir en navegador:

`http://localhost:4200/`

3. Despliegue en producción

Docker (opcional)

- El proyecto incluye Dockerfile y docker-compose.yml para backend y frontend.

Basta con ejecutar:

`docker-compose up --build`

-

### Servicios utilizados

- Frontend: Render (como web estática Docker)
- Backend: Render (servicio web Docker)
- BBDD: Railway (MySQL)

### 4. Variables de entorno

En producción se deben definir:

- JWT\_SECRET
- SPRING\_DATASOURCE\_URL
- SPRING\_DATASOURCE\_USERNAME
- SPRING\_DATASOURCE\_PASSWORD

### 5. Observaciones

- El backend se puede testear directamente desde Swagger.
- El frontend permite probar toda la experiencia completa.
- El sistema es modular y ampliable: se pueden añadir más funcionalidades sin romper la arquitectura actual.

### 20.2 Manual de usuario

Este manual está dirigido a usuarios finales (clientes y administradores) que utilicen la aplicación AITStore.

## 1. Acceso

- URL: <https://aitstore-frontend.onrender.com/>
- Registro gratuito como cliente.
- Acceso de administrador con credenciales proporcionadas por el desarrollador.

## 2. Funcionalidades del cliente

### Registro y login

- Formulario de registro con validaciones.
- Inicio de sesión con token de seguridad.

### Navegar productos

- Inicio con productos por categoría.
- Búsqueda por nombre desde la barra superior.

### Carrito de compras

- Añadir, eliminar o modificar cantidad.
- Cálculo automático del total.

### Realizar pedido

- Botón de compra en el carrito.

- Pedido registrado y asociado al usuario.

Ver pedidos

- En el perfil personal aparece el historial de compras.

### 3. Funcionalidades del administrador

Panel de administración

- Acceso desde menú superior si el usuario tiene rol ADMIN.
- Vistas para:
  - Gestión de productos (crear, editar, eliminar)
  - Gestión de usuarios
  - Gestión de pedidos

### 4. Modo oscuro y diseño responsive

- Activable desde el header.
- Funciona en móvil, tablet y escritorio.

### 5. Errores comunes y soluciones

Error Solución

“Token inválido” Cerrar sesión y volver a iniciar

“No hay stock suficiente” Disminuir la cantidad en el carrito

“No puede acceder a esta  
página”

Verificar que esté logueado o tenga  
permisos

## **Anexo B. Manual de usuario**

Este manual describe el uso general de la plataforma **AITStore**, diferenciando entre usuarios **clientes** y **administradores**, e ilustrando los principales flujos con ejemplos visuales.

El objetivo es facilitar la comprensión y el manejo de la aplicación por parte de cualquier usuario, sin necesidad de conocimientos técnicos previos.

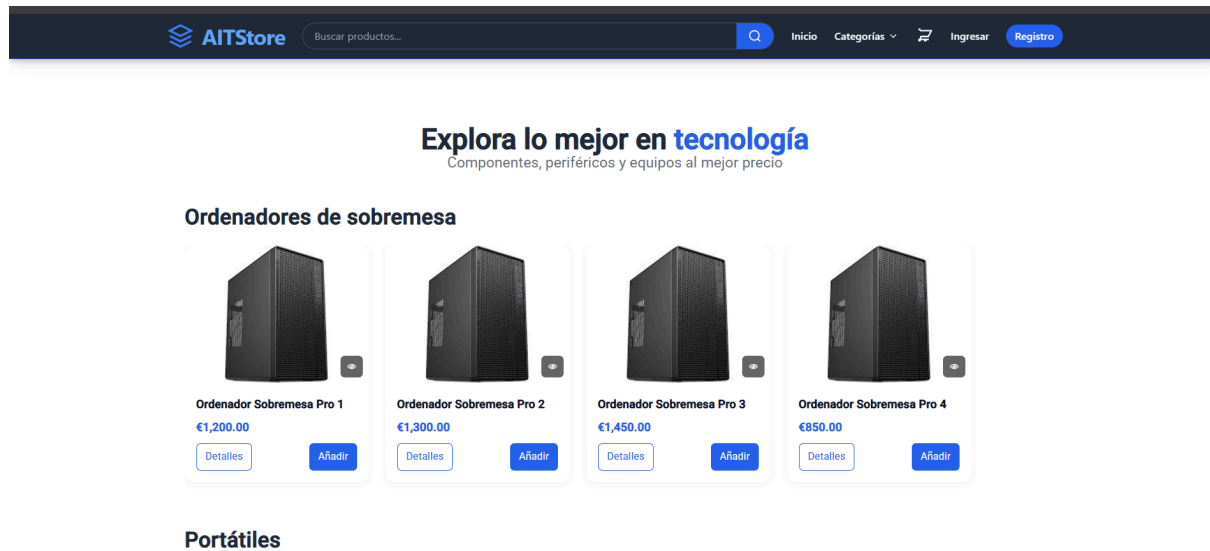
### **B.1 Acceso a la aplicación**

#### **URL pública:**

<https://aitstore-frontend.onrender.com> (reemplazar por tu URL real si es distinta)

La aplicación es accesible desde cualquier navegador moderno y está optimizada para ordenador, tablet y móvil.





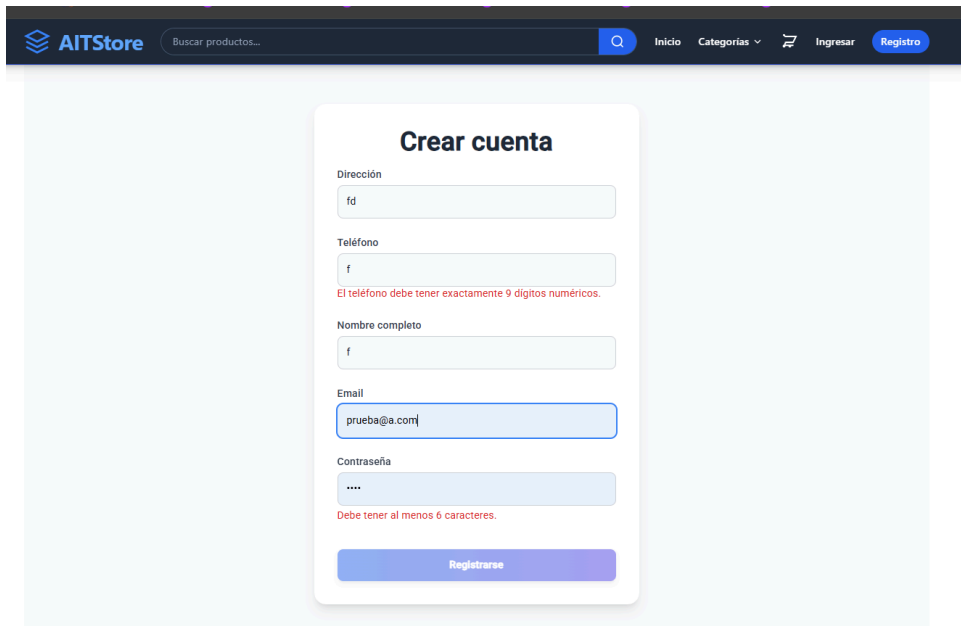
## B.2 Registro de usuario

1. Accede al botón **“Registrarse”** en el menú superior o desde el menú hamburguesa en móvil.
2. Completa el formulario con tu nombre, correo electrónico, contraseña, dirección y teléfono.
3. Pulsa **“Crear cuenta”**. Se mostrará una confirmación visual.

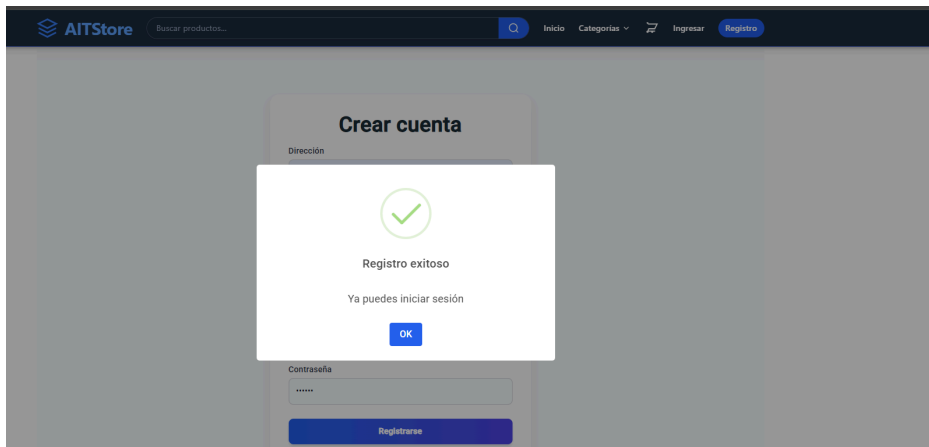
The screenshot shows the "Crear cuenta" registration form on the AITStore website. The form is a white card with a light blue background. It contains the following fields and labels:

- Dirección:** Calle Ejemplo 123, Ciudad
- Teléfono:** 612345678
- Nombre completo:** Tu nombre completo
- Email:** admin@mail.com
- Contraseña:** (masked with dots)

At the bottom of the form is a blue button labeled "Registrarse".

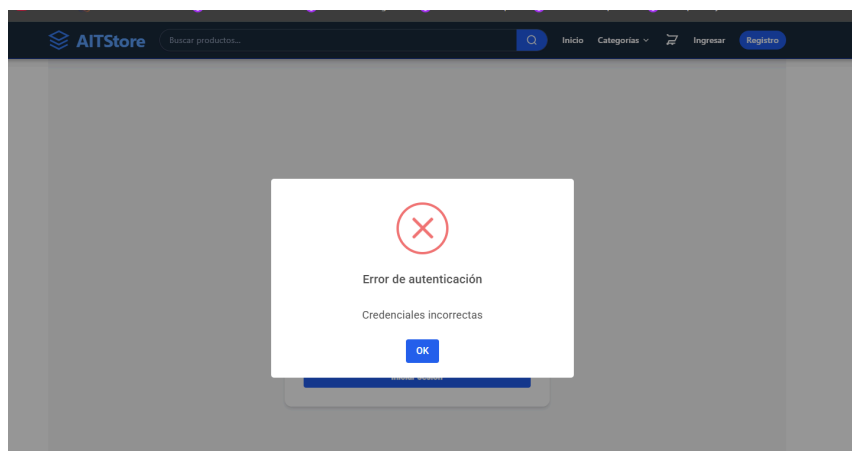
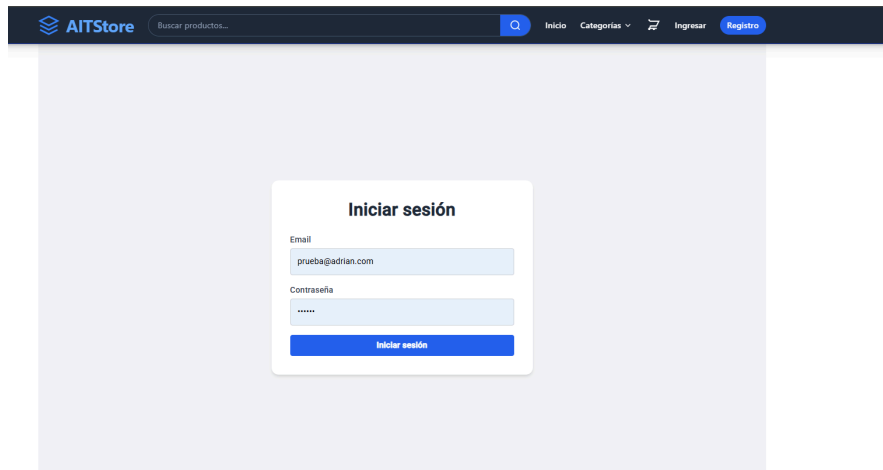


The screenshot shows the AITStore website header with a search bar and navigation links. The main content area features a 'Crear cuenta' (Create account) form. The form includes fields for 'Dirección' (Address), 'Teléfono' (Phone), 'Nombre completo' (Full name), 'Email', and 'Contraseña' (Password). The 'Teléfono' field has a red error message: 'El teléfono debe tener exactamente 9 dígitos numéricos.' (The phone number must have exactly 9 numeric digits). The 'Contraseña' field has a red error message: 'Debe tener al menos 6 caracteres.' (It must have at least 6 characters). A 'Registrarse' (Register) button is at the bottom of the form.



### B.3 Inicio de sesión

1. Accede al botón **“Ingresar”**.
2. Introduce el email y la contraseña registrados.
3. Si los datos son correctos, serás redirigido a la página principal o al panel admin (si eres administrador).



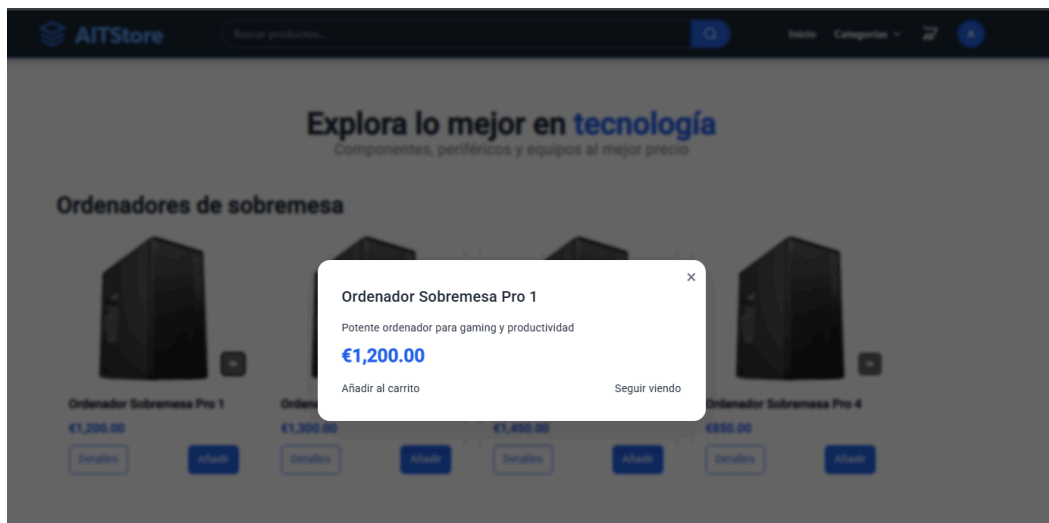
## B.4 Navegación general

- **Menú superior:** acceso a productos, buscador, perfil, carrito, login/logout.
- **Buscador:** permite encontrar productos por nombre de forma dinámica.
- **Modo oscuro:** conmutador automático según preferencia del usuario o sistema operativo.
- **Menú móvil:** accesible desde el icono de hamburguesa.



## B.5 Visualización de productos

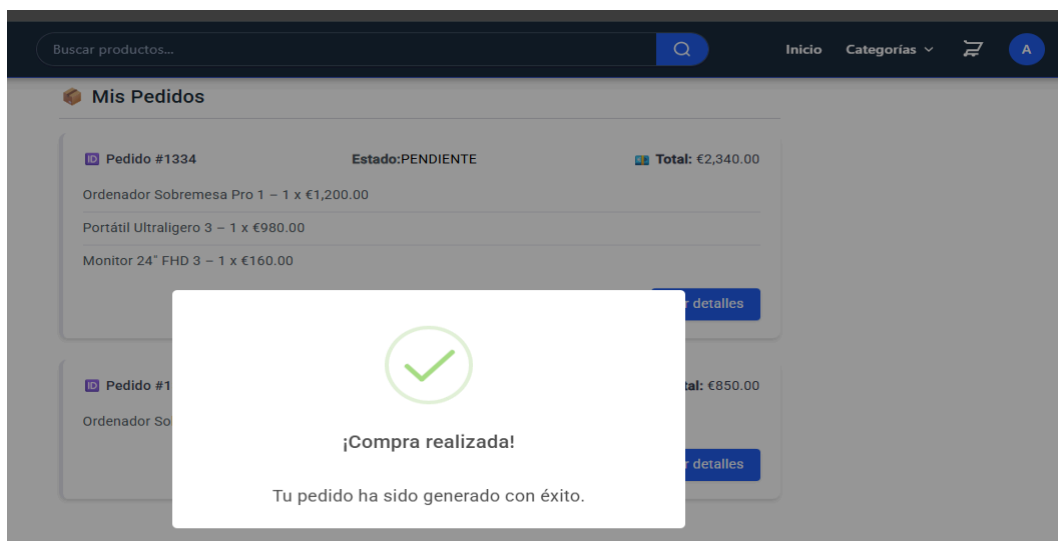
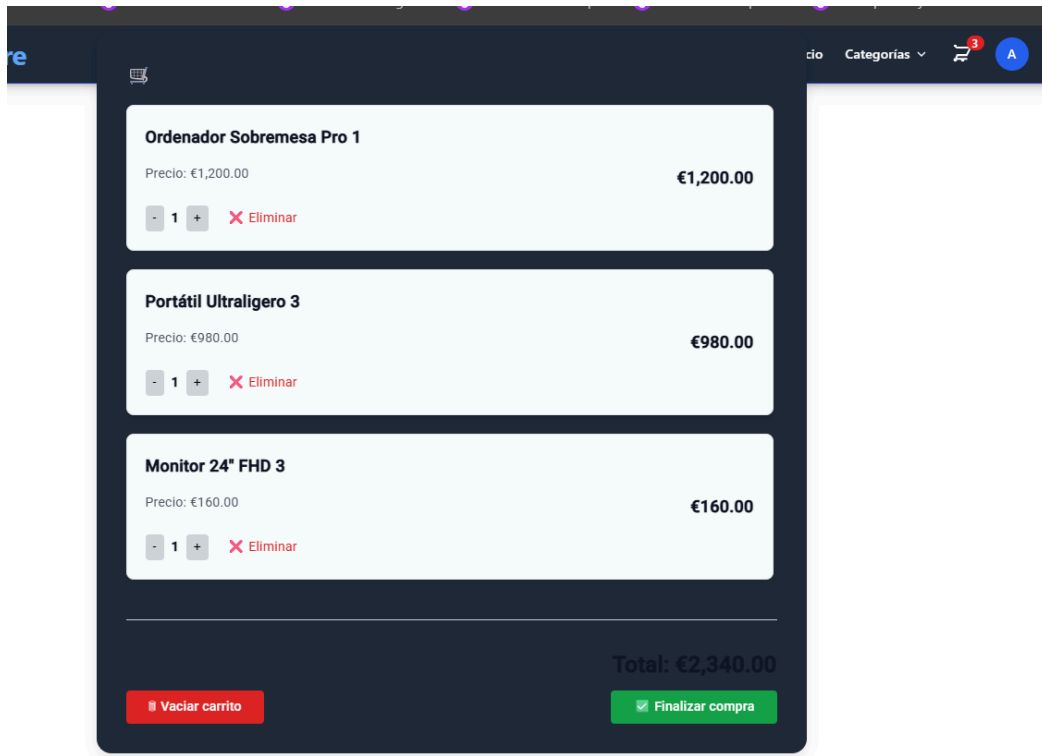
- Los productos se muestran organizados por **categoría**.
- Se puede hacer scroll horizontal para navegar por cada sección.
- Al hacer clic en un producto, se abre una vista emergente con más información.

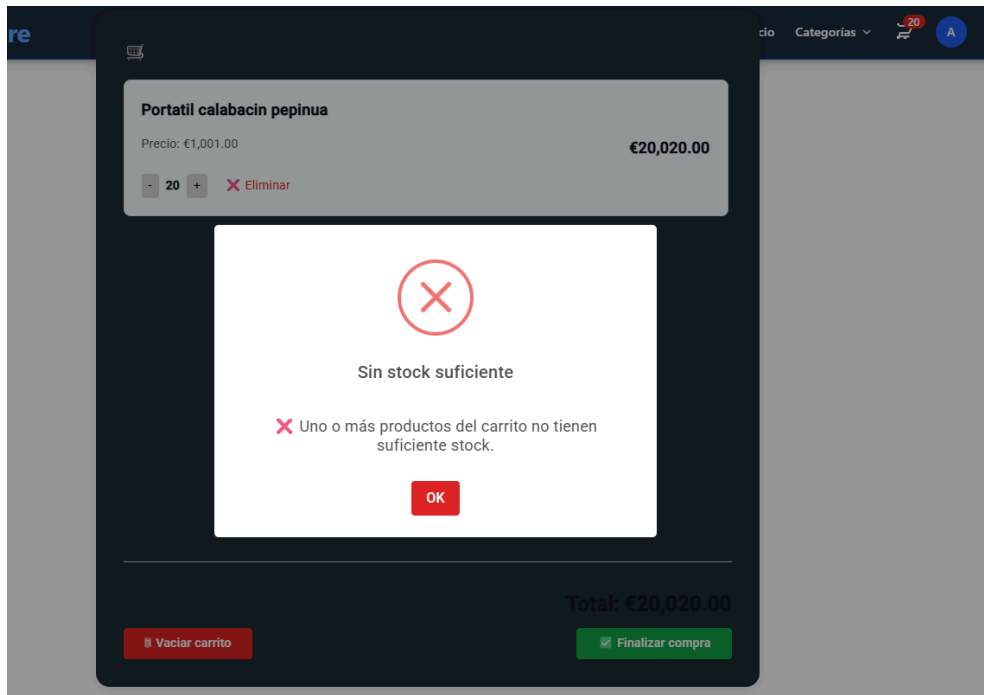


## B.6 Carrito de compra

1. Haz clic en **“Añadir al carrito”** desde cualquier producto.
2. El carrito muestra los productos añadidos, cantidades y subtotales.
3. Puedes aumentar/disminuir unidades o eliminar productos.
4. Pulsa **“Finalizar compra”** para confirmar tu pedido.

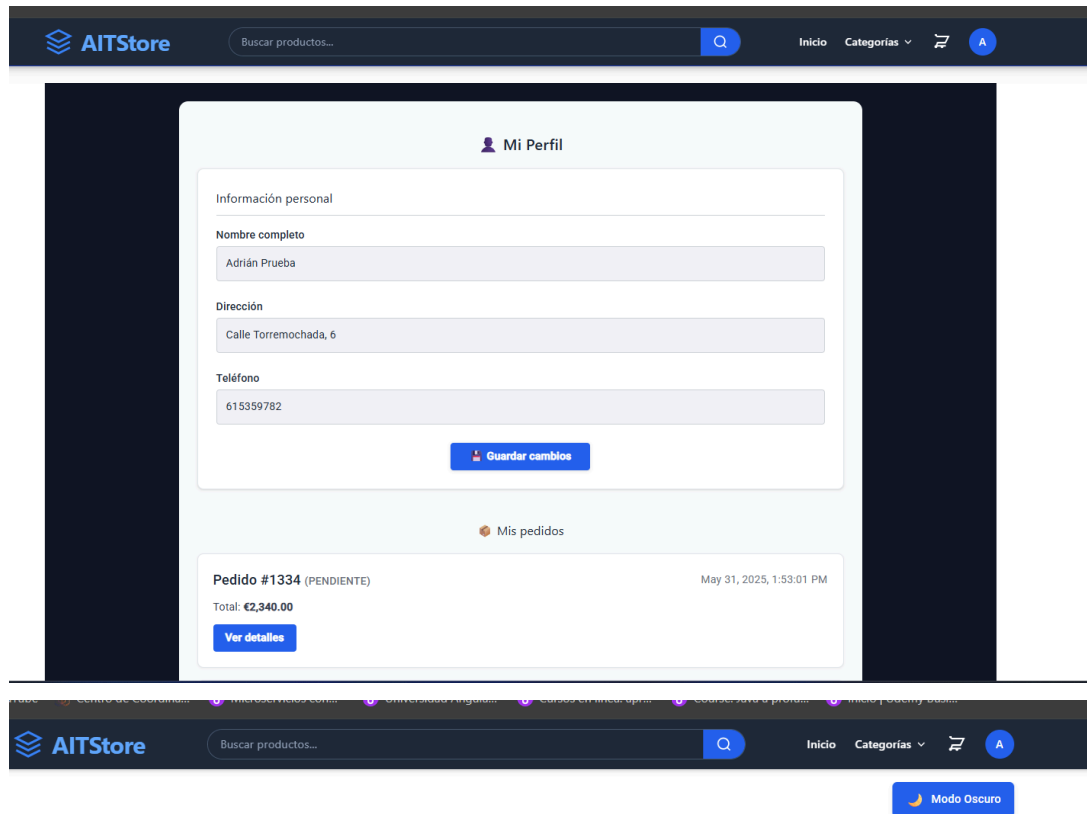
Se mostrará una alerta de confirmación y el stock se actualizará.





### B.7 Perfil de usuario

- Muestra los datos personales (no editables en esta versión).
- Lista de **pedidos realizados**.
- Botón para ver los **detalles de cada pedido** (productos comprados, cantidades, precios).



Detalle del Pedido	
Ordenador Sobremesa Pro 1 x1	€1,200.00
Portátil Ultraligero 3 x1	€980.00
Monitor 24" FHD 3 x1	€160.00
Total: €2,340.00	

## B.8 Panel de administrador

Solo accesible con cuenta con rol ADMIN.

Acciones disponibles desde el menú **“Administración”**:

### Gestión de productos

- Crear, editar y eliminar productos.
- Añadir nombre, precio, descripción, imagen, stock y categoría.

Panel de Administración Salir de admin

**Productos**  
Usuarios  
Pedidos

### Gestión de Productos

[Crear nuevo producto](#)

ID	Nombre	Precio	Stock	Acciones
1	Ordenador Sobremesa Pro 1	€1,200.00		<a href="#">Editar</a> <a href="#">Eliminar</a>
2	Ordenador Sobremesa Pro 2	€1,300.00		<a href="#">Editar</a> <a href="#">Eliminar</a>
3	Ordenador Sobremesa Pro 3	€1,450.00		<a href="#">Editar</a> <a href="#">Eliminar</a>
4	Ordenador Sobremesa Pro 4	€850.00		<a href="#">Editar</a> <a href="#">Eliminar</a>
5	Ordenador Sobremesa Pro 5	€1,400.00		<a href="#">Editar</a> <a href="#">Eliminar</a>
6	Ordenador Sobremesa Pro 6	€1,350.00		<a href="#">Editar</a> <a href="#">Eliminar</a>
7	Ordenador Sobremesa Pro 7	€1,500.00		<a href="#">Editar</a> <a href="#">Eliminar</a>
8	Ordenador Sobremesa Pro 8	€900.00		<a href="#">Editar</a> <a href="#">Eliminar</a>
9	Ordenador Sobremesa Pro 9	€1,250.00		<a href="#">Editar</a> <a href="#">Eliminar</a>
10	Ordenador Sobremesa Pro 10	€1,600.00		<a href="#">Editar</a> <a href="#">Eliminar</a>
11	Ordenador Sobremesa Pro 11	€300.00		<a href="#">Editar</a> <a href="#">Eliminar</a>

### + Crear Producto

Nombre del producto

Descripción

Precio (€)  Stock disponible

URL de la imagen

Categoría

Selecciona una categoría ▼

[Crear producto](#)

## Gestión de usuarios

- Visualización de todos los usuarios registrados.
- Eliminación de usuarios si es necesario.



### Gestión de Usuarios

ID	Nombre	Email	Rol	Acciones
516	Admin	admin@mail.com	ADMIN	
518	Federico	ferico@gmail.com	CLIENTE	<a href="#">Eliminar</a>
521	Marta	marta_saviola5@hotmail.com	CLIENTE	<a href="#">Eliminar</a>
523	Sergio Javier	sejagarrido@gmail.com	CLIENTE	<a href="#">Eliminar</a>
527	Manuel Cobos	manue@gmail.com	CLIENTE	<a href="#">Eliminar</a>
528	Lgranadog04	lorena.granado@gmail.com	CLIENTE	<a href="#">Eliminar</a>
530	prueba	prueba@b.com	CLIENTE	<a href="#">Eliminar</a>
531	aa	prueba@c.com	CLIENTE	<a href="#">Eliminar</a>
532	2	prueba@e.com	CLIENTE	<a href="#">Eliminar</a>
533	paquito	prueba@t.com	CLIENTE	<a href="#">Eliminar</a>
534	paquito	prueba@ee.com	CLIENTE	<a href="#">Eliminar</a>
535	dfd	prueba@cdf.com	CLIENTE	<a href="#">Eliminar</a>

### Gestión de pedidos

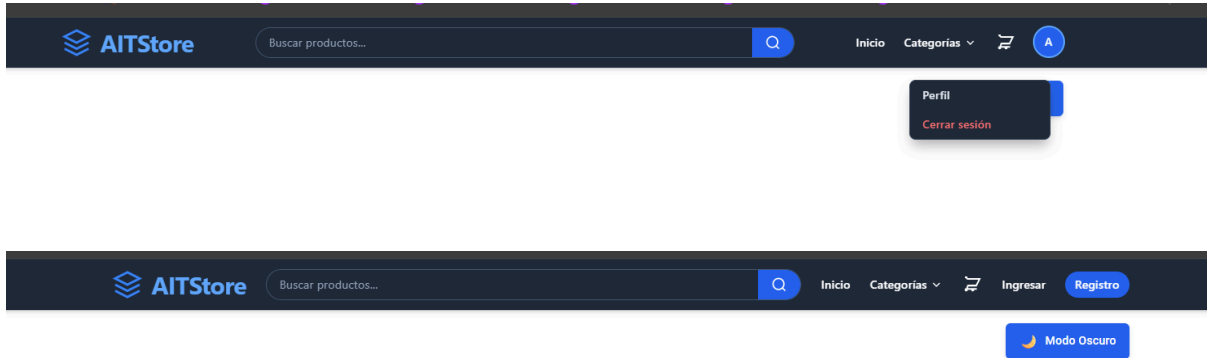
- Visualización de todos los pedidos realizados por todos los usuarios.

### Gestión de Pedidos

<b>Pedido #1278</b> (PENDIENTE)	4/15/25, 7:24 PM
Cliente ID: 516	Total: €3,950.00
<b>Pedido #1279</b> (PENDIENTE)	4/26/25, 9:39 AM
Cliente ID: 518	Total: €2,400.00
<b>Pedido #1280</b> (PENDIENTE)	4/26/25, 9:54 AM
Cliente ID: 518	Total: €2,750.00
<b>Pedido #1281</b> (PENDIENTE)	5/2/25, 1:01 PM
Cliente ID: 518	Total: €2,125.00
<b>Pedido #1283</b> (PENDIENTE)	5/2/25, 3:17 PM
Cliente ID: 518	Total: €1,300.00
<b>Pedido #1291</b> (PENDIENTE)	5/9/25, 7:56 AM
Cliente ID: 527	Total: €1,200.00

### B.9 Cierre de sesión

- Puedes cerrar sesión desde el icono de perfil → “Cerrar sesión”.
- Esto elimina el token del navegador y redirige a la pantalla de inicio.



## B.10 Compatibilidad y accesibilidad

- El sistema ha sido probado en:
  - Google Chrome, Firefox
  - Escritorio, móvil y tablet

- El diseño se adapta correctamente a todas las resoluciones.

