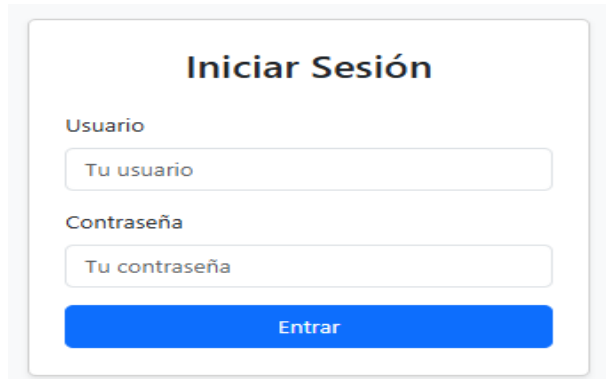


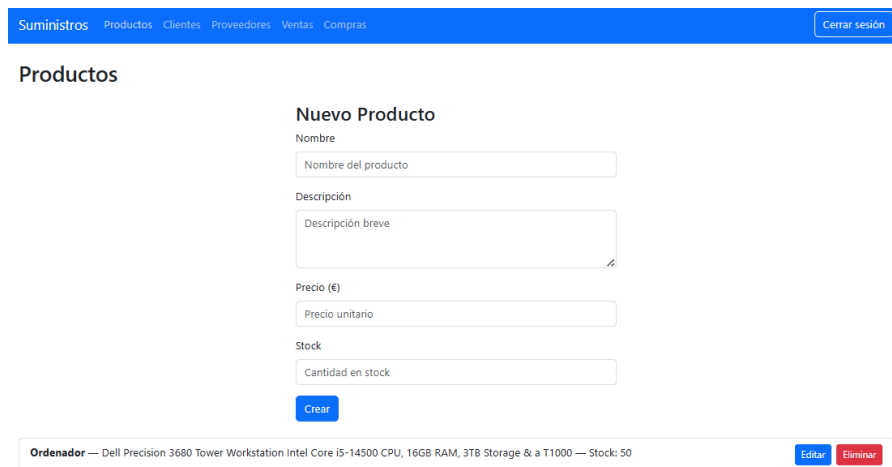
1. OBJETIVOS DEL PROYECTO

El objetivo principal de este proyecto es desarrollar una aplicación web completa para la gestión de productos, clientes, proveedores, compras y ventas dentro del contexto de una empresa de suministros informáticos. Se busca implementar un sistema funcional, modular, seguro y escalable que permita un control eficaz del inventario, totales automáticos y validaciones completas.



The image shows a login form titled "Iniciar Sesión". It contains two input fields: "Usuario" with the placeholder text "Tu usuario" and "Contraseña" with the placeholder text "Tu contraseña". Below these fields is a blue button labeled "Entrar".

Esta es la pantalla inicial del sistema. Para poder acceder a las funcionalidades, el usuario debe iniciar sesión. El login está conectado al backend y usa JWT para verificar que el usuario es correcto



The image shows the "Productos" section of the application. At the top, there is a blue navigation bar with links: "Suministros", "Productos", "Clientes", "Proveedores", "Ventas", "Compras", and a "Cerrar sesión" button. Below the navigation bar, the "Productos" section is displayed. It includes a "Nuevo Producto" form with fields for "Nombre" (Nombre del producto), "Descripción" (Descripción breve), "Precio (€)" (Precio unitario), and "Stock" (Cantidad en stock). There is a "Crear" button below the form. Below the form, there is a table with one row showing a product: "Ordenador — Dell Precision 3680 Tower Workstation Intel Core i5-14500 CPU, 16GB RAM, 3TB Storage & a T1000 — Stock: 50". There are "Editar" and "Eliminar" buttons next to the product name.

Una vez iniciamos sesión correctamente, se muestra la sección de productos. Aquí se pueden ver todos los artículos en stock, así como botones para editar o eliminar. Es la página por defecto tras iniciar sesión.

2. STACK TECNOLÓGICO Y ALTERNATIVAS EVALUADAS

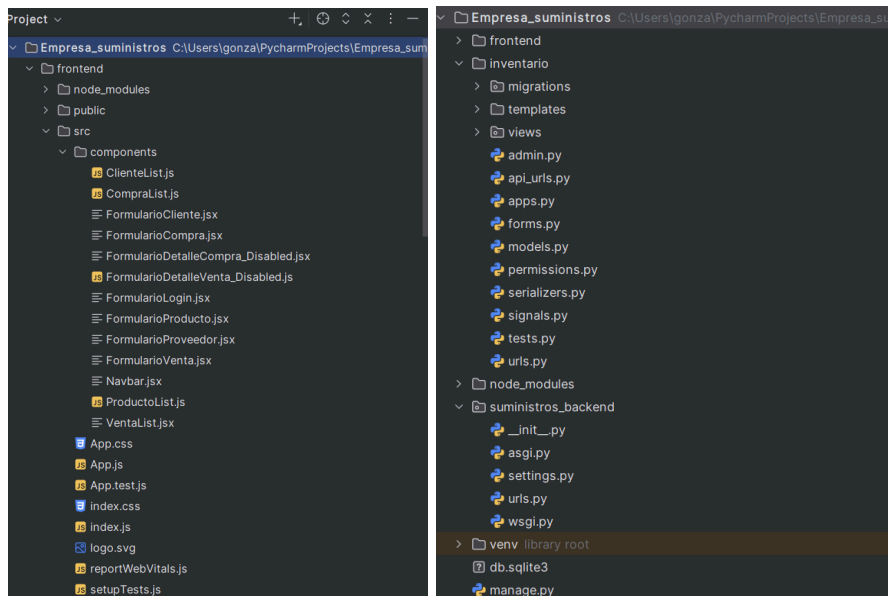
Tecnologías utilizadas:

- Backend: Django y Django REST Framework

- Frontend: React + Bootstrap
- Base de datos: SQLite
- Autenticación: JSON Web Tokens (JWT)

Se evaluaron otras tecnologías como Flask (menos estructurado para APIs REST) y plantillas de Django en lugar de React (menos flexibilidad para SPAs). Se eligió React por su separación clara de lógica y presentación, y Django por su robustez, seguridad y productividad.

Captura de estructura de carpetas del proyecto



En estas capturas se puede ver cómo está dividido el proyecto en dos partes principales: el frontend y el backend.

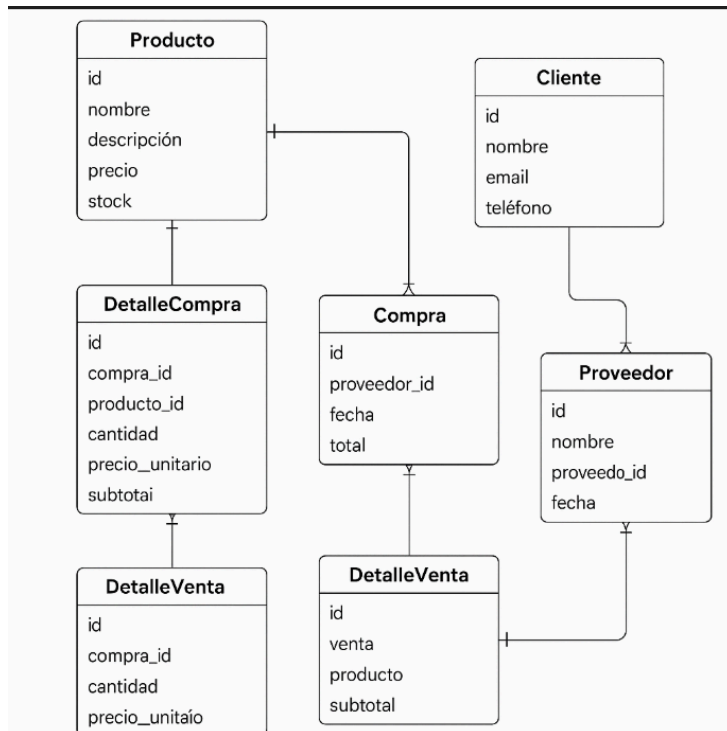
La carpeta frontend contiene el código de React, incluyendo los componentes, estilos y App.js, que es el núcleo del frontend.

Por otro lado, en la carpeta inventario dentro del backend, están los modelos de Django, vistas, formularios, serializadores y señales. También se puede ver el módulo suministros_backend, donde se encuentra la configuración general del proyecto, incluyendo settings.py y urls.py.

Esta separación permite trabajar de forma ordenada y mantener cada parte bien organizada.

3. EXPLICACIÓN Y ESQUEMA DE LA BASE DE DATOS

La base de datos está compuesta por siete modelos principales: Producto, Cliente, Proveedor, Compra, DetalleCompra, Venta y DetalleVenta. Se utilizan relaciones 1:N entre entidades principales y sus detalles. El stock se actualiza automáticamente con señales de Django.



Esquema de la base de datos

El sistema está compuesto por siete modelos principales:

- **Producto**: representa cada artículo gestionado en inventario.
- **Cliente**: almacena los datos de los clientes a quienes se venden productos.
- **Proveedor**: almacena los datos de los proveedores a quienes se compra.
- **Compra**: documento principal para registrar una compra a un proveedor.
- **DetalleCompra**: línea de cada producto comprado, enlazado a una compra y a un producto.
- **Venta**: documento principal para registrar una venta a un cliente.
- **DetalleVenta**: línea de cada producto vendido, enlazado a una venta y a un producto.

Las relaciones siguen la lógica de una empresa de suministros: un proveedor puede tener muchas compras, cada compra puede tener varios productos (detalle), y lo mismo para clientes y ventas. Además, el stock de cada producto se actualiza automáticamente a través de los detalles de compra y venta.

4. EXPLICACIÓN DE LOS REQUISITOS DE LA APLICACIÓN

Requisitos funcionales cubiertos por la aplicación:

- CRUD de productos, clientes y proveedores
- Registro de compras y ventas con detalles
- Cálculo automático de subtotales y totales
- Validaciones en frontend y backend
- Control de stock en tiempo real
- Autenticación segura mediante JWT
- Interfaz responsive y amigable

Suministros

Productos

Clientes

Proveedores

Ventas

Compras

Cerrar sesión

Productos

Nuevo Producto

Nombre

Nombre del producto

Descripción

Descripción breve

Precio

Stock

Crear Producto

Ordenador — Dell Precision 3680 Tower Workstation Intel Core i5-14500 CPU, 16GB RAM, 3TB Storage & a T1000 — Stock: 10

Editar

Eliminar

Teclado — Tempest Combo 3100 RGB Teclado + Ratón Gaming — Stock: 300

Editar

Eliminar

Ratón — Logitech G Pro Ratón Gaming Inalámbrico 25600DPI Negro — Stock: 50

Editar

Eliminar

Monitor — ASUS TUF Gaming VG32UQA1A 31.5" LED UltraHD 4K 160Hz FreeSync Premium — Stock: 30

Editar

Eliminar

En esta pantalla se muestran todos los productos disponibles. El usuario puede consultar su nombre, descripción, precio y stock, y tiene opciones para editar o eliminar cada uno. Esta interfaz permite registrar un nuevo producto o modificar uno existente. Los campos incluyen validaciones para evitar valores incorrectos.

[Suministros](#) [Productos](#) [Clientes](#) [Proveedores](#) [Ventas](#) [Compras](#) [Cerrar sesión](#)

Clientes

Nuevo Cliente

Nombre

Email

Teléfono

[Crear cliente](#)

Mediamarkt — Mediamarkt@gmail.com	Editar	Eliminar
Neobyte — NeoByte@gmail.com	Editar	Eliminar
PcComponents — PcComponents@gmail.com	Editar	Eliminar

En esta vista se observa el listado de clientes registrados, junto con el formulario para añadir nuevos. Cada cliente puede ser editado o eliminado mediante los botones correspondientes. Se aplican validaciones básicas como formato de email y longitud del teléfono.

[Suministros](#) [Productos](#) [Clientes](#) [Proveedores](#) [Ventas](#) [Compras](#) [Cerrar sesión](#)

Proveedores

Nuevo Proveedor

Nombre

Email

Teléfono

[Crear proveedor](#)

Componentes_Inf_China — 789487489	Editar	Eliminar
Ordenadores De China — 456456456	Editar	Eliminar

La sección de proveedores funciona de forma similar a la de clientes. Permite registrar nuevos proveedores con su información de contacto, así como modificar o eliminar los ya existentes. La validación de email y teléfono garantiza la calidad de los datos.

Ventas

Nueva Venta

Cliente

-- Selecciona un cliente --

Fecha

dd/mm/aaaa

Productos

+ Agregar producto

Total: €0.00

Crear

Cliente: Mediamarkt — Fecha: 16/7/2025 — Total: €180000.00

Editar

Eliminar

Ordenador — 90 uds

Cliente: PcComponents — Fecha: 16/7/2025 — Total: €16000.00

Editar

Eliminar

Monitor — 20 uds

En esta pantalla se gestiona el registro de ventas. El usuario puede seleccionar un cliente, indicar la fecha, y añadir uno o varios productos con sus cantidades. El sistema calcula automáticamente el total en función de los precios y las unidades seleccionadas. En la parte inferior se muestran las ventas ya registradas, con su respectiva fecha, cliente, productos vendidos y el total generado. Además, cada venta puede ser editada o eliminada desde esta misma vista.

Ventas

Nueva Venta

Cliente

PcComponents

Fecha

17/07/2025

Productos

Ratón (Stock: 50)

020

010

€200.00

+ Agregar producto

Total: €200.00

Crear

En esta vista se observa el formulario de registro de una venta con todos los campos rellenos. El usuario ha seleccionado un cliente, una fecha y ha añadido un producto desde un menú desplegable que incluye información del stock disponible. Además, se introducen unidades y precio, y el sistema calcula automáticamente el subtotal y el total general de la venta antes de registrarla.

Suministros Productos Clientes Proveedores Ventas Compras Cerrar sesión

Compras

Nueva Compra

Proveedor

-- Selecciona uno --

Fecha

dd/mm/aaaa

Productos

+ Agregar producto

Total: €0.00

Crear

Proveedor: Ordenadores De China — Fecha: 16/7/2025 — Total: €15000.00

Editar Eliminar

o Ordenador — 30 uds

Proveedor: Componentes_Inf_China — Fecha: 16/7/2025 — Total: €2487.50

Editar Eliminar

o Teclado — 250 uds

Esta sección permite registrar compras a proveedores. El usuario selecciona un proveedor, indica la fecha y añade los productos comprados con su cantidad. El sistema calcula automáticamente el total de la operación. En la parte inferior se muestran las compras ya realizadas, indicando proveedor, fecha, total y productos comprados con su cantidad. Esta vista también permite editar o eliminar cualquier compra registrada.

VALIDACIONES:

Email

prueba.com

Teléfono

Incluye un signo "@" en la dirección de correo electrónico. La dirección "prueba.com" no incluye el signo "@".

El formulario detecta automáticamente errores de formato en campos como el email o teléfono. En este ejemplo, el usuario ha introducido un valor sin el símbolo "@", y el navegador muestra una advertencia clara. Este tipo de validaciones evita errores antes de enviar los datos al backend.

Nuevo Producto

El precio debe ser un número positivo

Nombre

Ordenador

Descripción

Descripción breve

Precio

-1

Stock

1

Crear Producto

En la creación de productos, se aplican validaciones adicionales para asegurar que el precio y el stock sean positivos. Aquí se muestra un mensaje personalizado al intentar introducir un precio negativo. Esta validación se realiza en el frontend antes de enviar la solicitud.

Ventas

Nueva Venta

Uno o más productos no tienen suficiente stock.

Cliente

Neobyte

Fecha

17/07/2025

Productos

Ordenador (Stock: 0)	020	03000	€60000.00	x
----------------------	-----	-------	-----------	---

+ Agregar producto

Total: €60000.00

Crear

Al registrar una venta, el sistema verifica que haya suficiente stock disponible para los productos seleccionados. Si no hay stock, se bloquea la operación y se muestra un mensaje de error en rojo. Esta validación se realiza en el backend para garantizar integridad de los datos, incluso si se intenta manipular desde el cliente.

Nota sobre los mensajes de confirmación:

En la versión actual del frontend en React no se incluyen mensajes visuales de verificación tras crear, editar o eliminar registros.

En el backend de Django, sin embargo, sí se implementaron correctamente los mensajes success mediante el sistema de `django.contrib.messages`, visibles por ejemplo tras crear o editar registros desde las vistas basadas en plantillas (como productos o clientes).

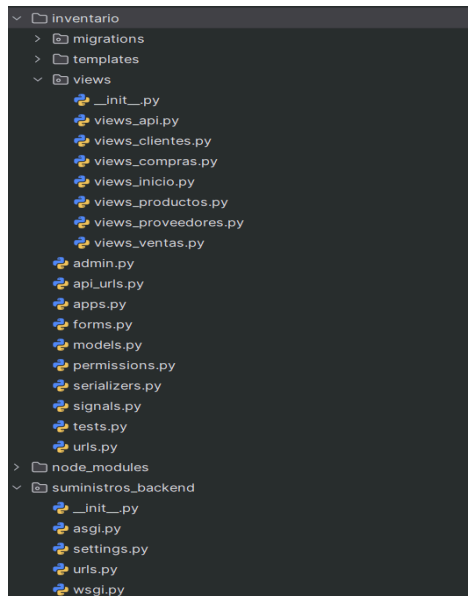
Para las vistas de eliminación (DeleteView), se configuró todo lo necesario para mostrar un mensaje de éxito tras eliminar un objeto:

- La vista delete() fue sobrescrita correctamente con messages.success()
- La plantilla confirm_delete.html hereda de base.html donde se renderizan los bloques de mensajes
- La ruta correspondiente en urls.py (path('clientes/eliminar/<int:pk>/')) estaba correctamente conectada

A pesar de esto, el mensaje de éxito **no llegó a visualizarse** tras la redirección, y no se logró identificar la causa exacta antes de la entrega.

Se documenta como un comportamiento inesperado, pendiente de revisión futura. La lógica está implementada correctamente, pero el resultado visual no se mostró en el flujo de eliminación.

BACKEND:



El backend se organiza dentro de la app inventario, donde se encuentran todos los archivos relacionados con la lógica del sistema. Se han separado las vistas en varios archivos (views_productos.py, views_clientes.py, etc.) para mejorar la legibilidad. Se utilizan archivos como models.py para definir las entidades de la base de datos, serializers.py para la API REST, signals.py para automatizar actualizaciones de stock y totales, y permissions.py para controlar el acceso. Esta estructura refleja un enfoque modular y profesional.

Administración de Django

Nombre de usuario:

Contraseña:

Iniciar sesión

Acceso al panel administrativo mediante usuario y contraseña de superusuario.

Administración de Django

BIENVENIDOS, GONZALO. VER EL SITIO / CAMBIAR CONTRASEÑA / CERRAR SESIÓN

Sitio administrativo

AUTENTICACIÓN Y AUTORIZACIÓN

Grupos

Añadir

Modificar

Usuarios

Añadir

Modificar

INVENTARIO

Clientes

Añadir

Modificar

Compras

Añadir

Modificar

Detalle compras

Añadir

Modificar

Detalle ventas

Añadir

Modificar

Productos

Añadir

Modificar

Proveedores

Añadir

Modificar

Ventas

Añadir

Modificar

TOKEN DE AUTENTICACIÓN

Tokens

Añadir

Modificar

Acciones recientes

Mis acciones

Compra #39 - Gonzalo

Compra

Gonzalo

Cliente

Gonzalo

Proveedor

12 x calculadora

Detalle venta

Venta #27 - gonzalo

Venta

12 x calculadora

Detalle compra

Compra #35 - pcomponents

Compra

200 x calculadora

Detalle compra

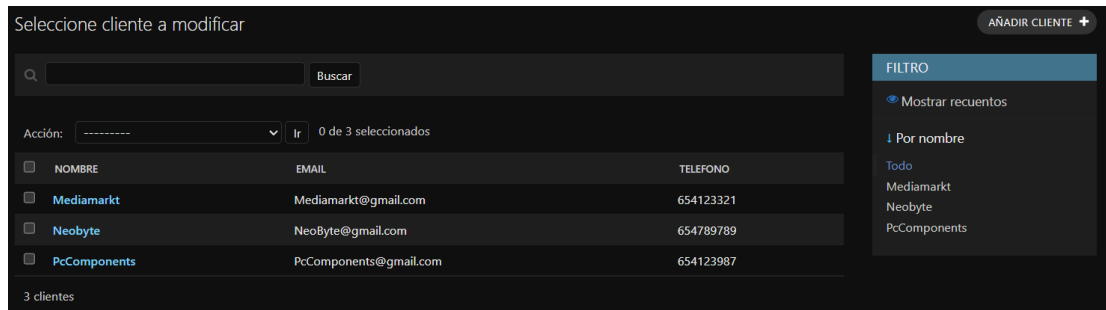
Compra #25 - pcomponents

Compra

10 x calculadora

Detalle compra

Se muestran todos los modelos principales registrados: productos, clientes, proveedores, compras, ventas y sus detalles.



El panel permite modificar, filtrar, buscar y eliminar registros de clientes ,productos,compras y ventas, facilitando la administración avanzada de los datos del sistema.

5. MANUAL DE INSTALACIÓN

Backend (Django)

1. Clona el proyecto o descomprime el archivo ZIP en tu ordenador.
2. Abre una terminal y sitúate en la carpeta principal del backend del proyecto.
3. Crea y activa un entorno virtual para Python.
4. Instala las dependencias necesarias del backend usando el gestor de paquetes (pip).
5. Realiza las migraciones de la base de datos.
6. Crea un superusuario para poder acceder al panel de administración de Django.
7. Inicia el servidor de desarrollo del backend.
8. Accede al panel de administración desde el navegador web, en la dirección <http://localhost:8000/admin/>.
9. También puedes acceder a las vistas y la API del sistema en <http://localhost:8000/>.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

(venv) PS C:\Users\gonza\PycharmProjects\Empresa_suministros> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
July 17, 2025 - 13:55:28
Django version 5.2.3, using settings 'suministros_backend.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.

WARNING: This is a development server. Do not use it in a production setting. Use a production WSGI or ASGI server instead.
For more information on production servers see: https://docs.djangoproject.com/en/5.2/howto/deployment/
█
```

En la imagen se muestra la consola tras iniciar el servidor de desarrollo de Django.

Se observa el entorno virtual activado, la ausencia de errores y el mensaje de que el servidor está disponible en <http://127.0.0.1:8000/>.

Esto confirma que la instalación de dependencias y migraciones se realizó correctamente.

Frontend (React)

1. Abre una nueva terminal y sitúate en la carpeta frontend del proyecto.
2. Instala las dependencias del frontend usando el gestor de paquetes de Node.js (npm).
3. Inicia el servidor de desarrollo de React.
4. La aplicación web se abrirá en el navegador en la dirección <http://localhost:3000/>.

```
Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

(venv) PS C:\Users\gonza\PycharmProjects\Empresa_suministros> cd frontend
(venv) PS C:\Users\gonza\PycharmProjects\Empresa_suministros\frontend> npm start

> frontend@0.1.0 start
> react-scripts start
v Something is already running on port 3000.

Would you like to run the app on another port instead? ... yes
(node:11672) [DEP_WEBPACK_DEV_SERVER_ON_AFTER_SETUP_MIDDLEWARE] DeprecationWarning: 'onAfterSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
(Use 'node --trace-deprecation ...' to show where the warning was created)
(node:11672) [DEP_WEBPACK_DEV_SERVER_ON_BEFORE_SETUP_MIDDLEWARE] DeprecationWarning: 'onBeforeSetupMiddleware' option is deprecated. Please use the 'setupMiddlewares' option.
Starting the development server...
Compiled successfully!

You can now view frontend in the browser.

Local: http://localhost:3000
On Your Network: http://192.168.1.130:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Aquí se ve la consola tras ejecutar el comando para arrancar el frontend con React.

El sistema indica que el proyecto se ha compilado correctamente y está disponible en el navegador en la dirección <http://localhost:3000/>.

Este mensaje certifica que todas las dependencias del frontend están correctamente instaladas y la aplicación React está lista para usarse.

Nota:

Para ejecutar el comando `npm start` y arrancar el frontend, es imprescindible situarse primero dentro de la carpeta frontend del proyecto.

Si el comando se ejecuta fuera de esta carpeta, no funcionará correctamente y la aplicación no podrá arrancar.

Durante el desarrollo del frontend con React, se implementaron inicialmente formularios individuales para gestionar los detalles de venta y los detalles de compra. Estos formularios permitían controlar directamente los productos, cantidades y precios unitarios de forma separada.

Sin embargo, para simplificar la experiencia del usuario, se optó por desactivar temporalmente esos formularios y centralizar toda la lógica de compras y ventas en los formularios principales (``FormularioVenta`` y ``FormularioCompra``). Esta decisión permitió una gestión más clara y directa, evitando errores por duplicidad o confusión de formularios secundarios.

Más adelante, y de forma controlada, los formularios de detalle fueron reactivados exclusivamente para pruebas en el panel de administración de Django, manteniéndolos ocultos en la interfaz principal del usuario final. Esta estrategia permitió conservar la funcionalidad avanzada sin afectar la usabilidad general del sistema.

En resumen, el sistema permite registrar y consultar todos los detalles de compras y ventas, pero concentra la gestión en formularios principales para mantener una experiencia limpia y controlada para el usuario.

Nota sobre los formularios de detalle de ventas y compras

Durante las pruebas del sistema clásico (plantillas Django en el puerto 8000), se detectó una limitación en la edición de ventas y compras. Cuando se edita una operación, el sistema genera automáticamente una fila vacía en el formulario por si se desea añadir más productos. Sin embargo, si esta fila queda vacía y no se elimina, el formulario completo se invalida y no se guarda la edición.

Esta limitación es inherente al uso de formsets de Django. Por eso, se recomienda:

- Eliminar manualmente las filas vacías antes de guardar.
- O realizar estas operaciones desde el frontend en React, donde el proceso es más claro y controlado.

Nota sobre la edición de ventas y compras desde el servidor 8000

Nota sobre las capturas del puerto 8000: No se han incluido capturas de las vistas servidas en el puerto 8000 (como inicio.html) ya que resultaban redundantes. Estas vistas reproducen funcionalidades similares al panel de administración y a la aplicación React, por lo que se optó por centrarse en documentar esta última, que representa la interfaz principal de usuario.

Esta separación de capas refleja una arquitectura profesional donde el backend administra los datos y el frontend (React) los consume y presenta de forma interactiva.

3. Aplicación React (<http://localhost:3000/>): Interfaz principal y moderna dirigida al usuario final. Está conectada por completo con la API REST del backend, lo que permite un funcionamiento ágil como SPA (Single Page Application).

2. Plantillas Django clásicas (<http://localhost:8000/>): Vistas tradicionales con HTML renderizado desde el servidor. Fueron útiles para pruebas y visualización básica de modelos.

1. Django Admin (<http://localhost:8000/admin/>): Interfaz administrativa destinada al superusuario. Permite filtrar, buscar, editar o eliminar todos los registros de forma rápida y eficaz.

Durante el desarrollo del sistema se implementaron tres formas distintas de interactuar con la aplicación, cada una con un propósito específico:

Interfaces del sistema y sus propósitos

6. CONCLUSIONES Y EVOLUTIVOS DEL PROYECTO

Conclusiones personales y técnicas

Este proyecto ha supuesto para mí un auténtico reto técnico y personal. No solo me ha servido para poner en práctica todo lo aprendido en Django y React, sino que me ha obligado a buscar soluciones a problemas reales, investigar documentación, y sobre todo, aprender de mis propios errores.

Una de las mayores dificultades ha sido coordinar correctamente la comunicación entre el backend y el frontend, sobre todo cuando intervienen conceptos como la autenticación JWT, la gestión de tokens y los estados en React. También me he encontrado con errores inesperados (como la no visualización de mensajes de éxito en Django tras eliminar), lo que me ha obligado a analizar el flujo interno de las vistas, las plantillas y las redirecciones.

Durante el desarrollo, descubrí que mi proyecto acabó teniendo tres “frontends” distintos:

- ***El admin de Django: para gestión interna de superusuario.***
- ***El frontend HTML clásico (Django Templates en el puerto 8000): para vistas tradicionales.***

- *El frontend moderno en React (puerto 3000): pensado para la experiencia de usuario final.*

Esto me ayudó a comprender la diferencia entre backend puro, backend con vistas, y una Single Page Application (SPA) real usando React. Al principio me costó entender por qué tenía varias “caras” el mismo proyecto, pero ahora lo veo como un aprendizaje clave de cómo las empresas realmente separan responsabilidades y tecnología.

Problemas, bloqueos y soluciones:

- *El mayor desafío fue la sincronización de datos y los errores de stock y validación entre frontend y backend.*
- *También he documentado y aprendido de errores no resueltos, como la no visualización de mensajes flash tras eliminar registros usando DeleteView en Django, a pesar de tener bien configuradas las vistas, plantillas y rutas.*
- *En React, me topé con problemas al crear, editar o eliminar productos, muchos de los cuales solucioné gracias a la depuración, el uso de consola y la revisión de props.*
- *Ha habido momentos de frustración, pero también mucha satisfacción al ver que todo iba tomando forma y era realmente funcional.*

Evolutivos y mejoras futuras:

- *Añadir mensajes de confirmación y notificaciones visuales en React (usando react-toastify o similar).*
- *Mejorar la experiencia de usuario en formularios, con más validaciones y diseños visuales atractivos.*
- *Implementar generación automática de facturas en PDF.*
- *Incluir gráficos y paneles de estadísticas para ventas, compras y stock.*
- *Desplegar el proyecto en la nube (por ejemplo, usando Heroku o Render) y conectar con una base de datos PostgreSQL.*
- *Añadir roles de usuario y permisos avanzados.*
- *Implementar tests automáticos para asegurar la robustez del sistema.*
- *Documentar aún mejor el proyecto y mejorar el README para su publicación en GitHub.*

En resumen, este proyecto me ha permitido desarrollar competencias técnicas clave, pero también habilidades personales como la organización, la gestión del tiempo y la capacidad de no rendirme ante los problemas. Siento que he dado un salto real como desarrollador y estoy motivado para seguir aprendiendo y mejorando esta plataforma.