1. Introducción

1.1 Presentación Personal

Me llamo **Eduardo García Romera** y soy estudiante del Grado en **Ingeniería Informática** en la **UNED**. Desde que era niño me ha gustado desmontar cosas para ver cómo funcionaban, sobre todo si tenían botones o pantallas. Con el tiempo, esa manía de "ver qué hay detrás" se convirtió en interés real por los ordenadores y la programación. Gracias al grado y a mi experiencia trabajando en Baleària y Minsait, he aprendido tanto de teoría como de la práctica.

Este TFG no lo veo solo como un trámite para titularme, sino como una oportunidad para juntar todo lo aprendido y construir algo que funcione de verdad, como lo haría en un entorno profesional.

1.2 Contexto del Proyecto

Este Trabajo de Fin de Grado gira en torno al desarrollo de software para la gestión de inventarios, algo que hoy en día es clave en muchas empresas, sobre todo en las que venden online.

El proyecto parte de un caso ficticio pero muy realista: una empresa parecida a Amazon que necesita llevar un buen control de su inventario, registrar ventas, gestionar distintos tipos de usuarios y recibir alertas cuando algo va mal, como por ejemplo si se acaba el stock.

Aunque no está pensado para una empresa real, sí que he intentado que el sistema responda a problemas habituales que cualquier e-commerce podría tener. Al final, la idea es que sirva como base para una solución realista, que se pueda adaptar fácilmente a un entorno profesional.

1.3 Objetivos Generales y Específicos

Objetivo general:

El objetivo principal de este TFG es crear una aplicación web que sirva para gestionar inventario online, permitiendo controlar las ventas, el stock disponible y los usuarios según sus permisos. También quiero incluir un sistema de alertas automáticas y gráficos que ayuden a ver la evolución de las ventas o detectar problemas de inventario.

Objetivos específicos:

1. Crear una base de datos relacional bien estructurada que garantice la integridad de los datos y cubra las necesidades del sistema.
2. Desarrollar un backend con Java y Spring Boot que sea fácil de mantener y que pueda crecer si hiciera falta.
3. Montar un frontend con Angular 19 y PrimeNG que sea fácil de usar, que se vea bien en distintos dispositivos y que no resulte confuso.
4. Añadir un sistema de login seguro con autenticación JWT para proteger el acceso a la aplicación.
5. Documentar todos los endpoints del backend con Swagger para poder probar y entender la API sin herramientas externas.
6. Implementar un sistema de roles que limite el acceso según el tipo de usuario.
7. Incluir paneles con gráficos y tablas que muestren información útil, como qué productos se venden más o qué almacén tiene exceso de stock.
8. Probar la aplicación tanto con datos normales como en situaciones límite, y ver que responde bien.

1.4 Justificación del Proyecto

Elegí este proyecto porque me parecía una buena forma de juntar muchas de las cosas que he aprendido durante la carrera: diseño de bases de datos, desarrollo web, arquitectura backend, seguridad, interfaces, etc. Pero sobre todo porque quería hacer algo que tuviera sentido en un contexto real, no solo un CRUD simple.

Mi idea ha sido ir un paso más allá y construir un sistema completo que podría usarse de verdad en una empresa. Sé que hay muchas herramientas de este tipo en el mercado, pero hacer una propia desde cero te obliga a pensar en problemas reales como la seguridad, el control de accesos, la organización del código o la escalabilidad.

Además, este tipo de aplicaciones son muy comunes en el mundo laboral, y me viene bien practicar algo que se alinea con lo que ya hago en el trabajo. Aunque los temas de los TFG suelen estar bastante definidos, esta fue mi primera opción desde el principio porque me permite aplicar conocimientos de casi todas las asignaturas clave del grado.

2. Alcance del Proyecto

2.1 Funcionalidades Esenciales

El sistema que estoy desarrollando está pensado para cubrir las funciones básicas que necesitaría una empresa que vende por internet y quiere controlar su inventario de forma ordenada. A continuación, resumo las funciones más importantes que he incluido:

- **Gestión de productos**: se pueden crear, editar y borrar productos. Cada uno tiene nombre, precio, SKU, descripción, categoría y un límite mínimo de stock para lanzar alertas.

- **Inventario por almacenes**: un mismo producto puede estar en varios almacenes, y el sistema lleva la cuenta de cuánto hay disponible en cada uno. Si se pasa del límite (por arriba o por abajo), se genera una alerta automática.

- **Registro de ventas y movimientos**: todo lo que se mueve en el inventario queda registrado: entradas (add), salidas (remove) y ventas (sale). Así se puede ver el historial completo de lo que ha pasado con cada producto.

- **Usuarios y roles**: hay 4 tipos de usuarios, cada uno con permisos distintos según su función en la empresa (administrador, encargado, marketing y reponedor).

- **Alertas automáticas**: el sistema avisa cuando hay poco stock, demasiado, o cuando un producto lleva tiempo sin moverse.

- **Dashboard con visualizaciones**: hay una sección de estadísticas donde se pueden ver gráficos y tablas sobre ventas, almacenes y productos.

- **Auditoría de acciones**: se guarda un registro de las acciones importantes para saber quién hizo qué y cuándo.

- **Seguridad**: todo el sistema está protegido por tokens JWT. Cada usuario solo puede acceder a las partes que le corresponden según su rol.

2.2 Limitaciones Previstas

A pesar de intentar desarrollar un sistema lo más completo posible, hay ciertas limitaciones que se aceptan como parte del alcance razonable de un TFG:

- No se incluye pasarela de pagos ni gestión económica detallada.
- La generación de informes se limita a filtros básicos por fecha o categoría, no incluye BI (Business Intelligence) avanzado ni análisis predictivo.
- Aunque se aplican buenas prácticas de seguridad, no se integran herramientas de escaneo automático de vulnerabilidades ni pentesting profesional.
- El sistema no contempla internacionalización para múltiples monedas/idiomas.

Estas limitaciones no afectan a la funcionalidad esencial del sistema, pero se podrían tener en cuenta en futuras ampliaciones o versiones comerciales.

2.3 Público Objetivo

El sistema está pensado para empresas pequeñas o medianas que necesitan tener un control centralizado de su inventario distribuido en varios almacenes, con distintos perfiles de trabajadores implicados en su gestión. Es ideal para negocios de venta online que manejan un catálogo de productos de tamaño medio y necesitan herramientas de análisis sin pagar licencias de software comerciales.

2.4 Requisitos Mínimos del Sistema

Requisitos para el entorno de desarrollo:

- **Frontend**:
    - Navegador moderno (Chrome, Firefox)
    - Node.js v20+
    - Angular CLI

- **Backend**:
    - Java 17+
    - Spring Boot 3
    - Maven
- **Base de datos**:
    - PostgreSQL 15 o superior
- **Otros**:
    - Docker (opcional para despliegue)
    - Git (control de versiones)
    - IDEs recomendados: Visual Studio Code (frontend), IntelliJ IDEA (backend)

Requisitos para el usuario final:

- Acceso a un navegador web
- Resolución mínima 1280x720
- Conexión a internet

3. Tecnologías Elegidas y Justificación

La elección del conjunto de tecnologías ha sido uno de los primeros pasos importantes del proyecto. He optado por herramientas que ya conocía previamente, ya fuera por mi experiencia laboral o por proyectos anteriores, pero utilizando versiones más actualizadas con el objetivo de profundizar en su uso y ampliar mis conocimientos. A continuación, detallo cada una de las tecnologías empleadas, junto con una breve justificación de su elección.

3.1 Angular 19 con PrimeNG – Frontend

Para el desarrollo del frontend he utilizado Angular en su versión 19. Aunque en mi entorno profesional suelo trabajar con Angular 7, he querido aprovechar este proyecto para familiarizarme con una versión más moderna del framework. Esta actualización me ha permitido beneficiarme de mejoras en rendimiento, organización de módulos y nuevas funcionalidades.

En cuanto a la interfaz de usuario, he optado por PrimeNG como librería de componentes. Entre las alternativas disponibles, PrimeNG destaca por su amplia variedad de componentes reutilizables, su facilidad de integración y su aspecto visual profesional sin necesidad de realizar grandes esfuerzos de personalización. Esto me ha permitido centrarme en la lógica funcional de la aplicación, asegurando al mismo tiempo una experiencia de usuario adecuada.

3.2 Java + Spring Boot – Backend

Para la parte del backend he trabajado con Java 17 y Spring Boot 3. Esta elección responde tanto a mi familiaridad con estas tecnologías como a su robustez y popularidad en el desarrollo de aplicaciones empresariales. Aunque ya tenía experiencia previa con versiones anteriores, en este proyecto he querido utilizar versiones más actuales para explorar mejoras como el soporte a Jakarta EE o la configuración simplificada.

Spring Boot ha facilitado la estructuración de la aplicación en capas claramente diferenciadas (controladores, servicios, repositorios) y ha permitido integrar de forma sencilla otras herramientas esenciales como Swagger para documentación, JWT para seguridad y PostgreSQL como sistema de base de datos.

3.3 PostgreSQL – Sistema de Bases de Datos

Para el sistema de almacenamiento he elegido PostgreSQL, un motor de base de datos relacional ampliamente utilizado en entornos profesionales. Destaca por su cumplimiento de estándares, buen rendimiento, escalabilidad y soporte para tipos de datos avanzados como `ENUM` y `JSONB`, que han resultado especialmente útiles en este proyecto.

Además, su integración con Spring Boot ha sido directa y sin complicaciones, lo cual ha facilitado el desarrollo y la puesta en marcha del sistema desde las primeras fases.

3.4 JWT con Bearer Token – Seguridad

La autenticación de los usuarios se ha implementado mediante JWT (JSON Web Tokens), una solución moderna y ligera para gestionar sesiones sin necesidad de almacenar información en el servidor. Una vez el usuario se autentica correctamente, se le asigna un token que se incluye en cada petición como Bearer Token. El backend valida dicho token y autoriza el acceso según el rol del usuario.

Esta técnica se ajusta bien a aplicaciones SPA (Single Page Applications) como la desarrollada en este TFG, y ofrece una solución segura y escalable.

3.5 Swagger – Documentación de APIs

Con el objetivo de facilitar tanto la prueba como la comprensión de los endpoints del backend, se ha utilizado Swagger (OpenAPI). Gracias a esta herramienta, la documentación de la API se genera de forma automática y se presenta mediante una interfaz gráfica que permite ejecutar peticiones directamente desde el navegador. Esta funcionalidad ha sido útil tanto en fase de desarrollo como para la validación de funcionalidades.

3.6 Git y GitHub – Control de Versiones

Durante todo el desarrollo del proyecto se ha utilizado Git como sistema de control de versiones, con GitHub como plataforma de alojamiento del repositorio. Esta herramienta ha permitido mantener un registro ordenado de los avances, organizar las distintas funcionalidades por ramas, y facilitar la colaboración y revisión del código cuando ha sido necesario.